

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ**

*Єгоршин О. О.*  
*Малярець Л. М.*  
*Сінкевич Б. В.*

**ДОВІДНИК З МАТЕМАТИЧНОЇ  
СТАТИСТИКИ З ПРИКЛАДАМИ  
ОБЧИСЛЕНЬ У MATLAB**

**Навчально-практичний посібник**

**Частина 2**

**Харків. Вид. ХНЕУ, 2009**

УДК519.22(035)

ББК22.172я2

Є72

Рецензенти: докт. екон. наук, професор, зав. кафедри математики і математичних методів економіки Донецького національного університету *Христіановський В. В.*; докт. фіз.-мат. наук, професор, зав. відділу радіофізичної інтроскопії Інституту радіофізики та електроніки ім. О. Я. Усикова НАН України *Масалов С. О.*

Затверджено на засіданні вченої ради Харківського національного економічного університету.

Протокол № 5 від 02.12.2008 р.

**Рекомендовано Міністерством освіти і науки України як  
навчальний посібник для студентів вищих навчальних закладів  
(лист № 14/18-Г-2490 від 03.12.2008 р.)**

**Єгоршин О. О.**

Є72 Довідник з математичної статистики з прикладами обчислень у MatLab : навчально-практичний посібник. Ч. 2 / О. О. Єгоршин, Л. М. Малярець, Б. В. Сінкевич. – Харків : Вид. ХНЕУ, 2009. – 508 с. (Укр. мов.)

Детально описано функції з математичної статистики, що реалізовані в обчислювальній системі MatLab, а також наведено необхідні теоретичні відомості. Викладення супроводжується великою кількістю прикладів розрахунків у MatLab, що можна розглядати як реалізацію типових питань при розв'язанні задач математичної статистики.

Рекомендовано для студентів, аспірантів та науковців, які використовують інструменти теорії ймовірності і математичної статистики.

**ISBN 978-966-676-378-8**

**ISBN 978-966-676-379-5**

**УДК 519.22(035)**

**ББК 22.172я2**

© Харківський національний економічний університет, 2009

© Єгоршин О. О.

Малярець Л. М.

Сінкевич Б. В.

2009

## Вступ

При написанні даного посібника автори ставили за мету посилити професійну компетенцію майбутніх фахівців різних галузей знань, за рахунок уміння застосовувати інструменти математичної статистики у вирішенні практичних завдань з обчисленнями в **MatLab**.

Користуючись другою частиною довідника студенти, аспіранти, дослідники мають змогу дізнатися про сучасні методи математичної статистики, включаючи методи багатовимірного статистичного аналізу. В другій частині описані також функції керування статистичними процесами, планування експерименту і марківські моделі. В останньому розділі наведено способи введення і збереження числової і текстової інформації при роботі в **MatLab**.

Вивчивши матеріал другої частині довідника дослідник може кваліфіковано перевіряти статистичні гіпотези; застосовувати дисперсійний, коваріаційний та регресійний аналізи; використовувати нелінійні моделі; набути навички аналізу багатовимірних об'єктів, зокрема застосовувати методи кластерного аналізу, багатовимірного шкалювання, а також функцій зниження розмірності багатовимірного простору. Деякі статистичні методи є найновітнішими і викладені лише у спеціальній науковій літературі.

Опис кожної статистичної функції супроводжується великою кількістю зразків програмних кодів, що реалізують найпоширені запити. Працездатність цих кодів перевірена копіюванням програмних фрагментів у робочу область **MatLab** з виконанням розрахунків.

У стандартний перелік статистичних функцій **MatLab** нами були додані функції **CHI2TEST** – критерій згоди Пірсона  $\chi^2$ , **BTEST** – критерій Бартлета для порівняння дисперсій, **COCHTEST** – критерій Кокрена для порівняння дисперсій. Тексти цих функцій наведено у додатку А.

Даний навчально - практичний посібник відповідає вимогам підготовки фахівців бакалаврського та магістерського рівнів пвдготовки за всіма економічними спеціальностями.

Для зручності користування довідником для кожного розділу, підрозділів, переліків функцій додані навігатори – скорочені змісти підрозділів і перелік функцій кожного підрозділу з посиланням на відповідні сторінки довідника. Нумерація сторінок – своя для кожної частини. У додатку Б наведено перелік за абеткою всіх функцій, описаних у другій частині. Цей перелік також є навігатором і дозволяє швидко знаходити опис необхідних функцій.

Нижче наведено скорочений зміст другої частини довідника.

## Зміст

Вступ .....	1
10. Перевірка статистичних гіпотез про згоду розподілу експериментальним даним .....	6
11. Перевірка статистичних гіпотез (порівняння вибіркових характеристик) .....	37
12. Дисперсійний, коваріаційний і регресійний аналізи .....	75
13. Нелінійні моделі .....	237
14. Функції кластерного аналізу .....	256
15. Функції зниження розмірності задачі .....	294
16. Функції аналізу багатовимірних випадкових величин .....	327
17. Функції нелінійного регресійного аналізу на підставі графа можливих рішень .....	344
18. Керування статистичними процесами .....	380
19. Планування експерименту .....	402
20. Марківські моделі .....	444
21. Функції введення, форматування й збереження інформації .....	463
Використана література .....	506
Додаток А .....	511
Додаток Б .....	516
Зміст частини 2 .....	516



## 10. Перевірка статистичних гіпотез про згоду розподілу експериментальним даним

На жаль, навіть у 5-й версії **Statistics Toolbox** відсутній  $\chi^2$ -критерій згоди Пірсона (хоча там є інші широко розповсюджені критерії перевірки відповідності даних вибірки до теоретичного розподілу). Проте, за адресою

<http://www.mathworks.com/matlabcentral/fileexchange/loadcategory.do> у категорії **Statistics and Propability** можна знайти m-файл Chi2test.m, де реалізовано  $\chi^2$ -критерій згоди Пірсона, імпортувати його в установлену версію **MatLab** і застосовувати за необхідності. На момент написання довідника у категорії **Statistics and Propability** перебувало 435 m-файлів. Всі вони доступні для вільного копіювання й використання.

Нижче розглянуто наступні функції критеріїв узгодження:

<b>CHI2TEST</b> $\chi^2$ -критерій згоди Пірсона .....	6
<b>KSTEST</b> Тест Колмогорова-Смірнова на непротириччя розподілу значень випадкової величини заданому закону .....	11
<b>KSTEST2</b> Тест Колмогорова-Смірнова для порівняння розподілів двох сукупностей значень випадкових величин .....	19
<b>JBTEST</b> Тест Яркі-Бера на непротириччя розподілу значень випадкової величини нормальному закону .....	25
<b>LILLIETEST</b> Тест Лільєфорса на непротириччя розподілу значень випадкової величини нормальному закону .....	31

### **CHI2TEST** $\chi^2$ -критерій згоди Пірсона

Проводиться  $\chi^2$ -тест Пірсона для перевірки 0-гіпотези про те, що задана вибірка взята з генеральної сукупності з відомим теоретичним розподілом. Автор цієї функції – Леонардо Саломоне (Leonardo Salomone).

Цей найбільш поширений критерій вимагає дотримання деяких (у певному сенсі) суперечливих вимог, які можуть бути задовільнені лише для вибірок великого розміру ( $n > 200$ ). Тому розглянемо припущення, що полягають в основі  $\chi^2$ -критерію.

Якщо деякі випадкові величини  $x_i$  розподілені за стандартним нормальним законом  $x_i \sim N(0;1)$ , то сума їх квадратів  $\chi^2 = \sum x_i^2$  розподілена за законом "хі-квадрат" Пірсона (частковий випадок гамма-розподілу).

На цьому факті і засновано критерій згоди з цією ж назвою.

Дані слід угрупувати на  $k$  інтервалів, і в кожному інтервалі підрахувати кількість спостережень  $m_i$ , що в сумі складають об'єм вибірки  $\sum m_i = n$ . Згідно з прийнятим теоретичним законом, обчислюються ймовірності потрапляння випадкової величини в ці самі інтервали  $f_i$  і теоретичні частоти  $\kappa_i = n f_i$ . Для того, щоб сума теоретичних частот дорівнювала загальному числу спостережень  $\sum \kappa_i = n$  (наслідок вимоги  $\sum f_i = 1$ ), треба границі крайніх інтервалів розширити до  $\pm\infty$ . Випадкові частоти  $m_i$  в кожному інтервалі розподілені за біноміальним законом Бернуллі з характеристиками  $M(m_i) = n f_i$ ,  $D(m_i) = n f_i (1 - f_i)$  або

$$M(m_i) = \kappa_i, \quad D(m_i) = \kappa_i \left(1 - \frac{\kappa_i}{n}\right).$$

Згідно з центральною граничною теоремою, зі збільшенням числа спостережень розподіл Бернуллі наближається до нормального (для цього досить  $n > 30$  і  $M(m_i) = \kappa_i > 5$ ). Щоб задовольнити останню вимогу, малонасичені

інтервали слід укрупнити. Тоді статистика  $\sum \frac{(m_i - \kappa_i)^2}{\kappa_i \left(1 - \frac{\kappa_i}{n}\right)}$  буде

розподілена за законом "Хі-квадрат" Пірсона (як сума квадратів стандартизованих нормально розподілених величин). Якщо відносні частоти невеликі порівнянне з 1 ( $\frac{\kappa_i}{n} < 0.05$ ), то статистика Пірсона

прийме стандартний вигляд  $\chi^2 = \sum \frac{(m_i - \hat{\mu}_i)^2}{\hat{\mu}_i}$ . Саме тут з'являються

дещо суперечливі вимоги – число спостережень в кожному інтервалі повинно бути більше 5 і одночасно складати менше 5% об'єму вибірки. Ці вимоги можуть бути задоволені лише для вибірок великого об'єму ( $n > 200$ ).

Теоретичний розподіл Пірсона залежить від одного параметра – числа ступенів свободи  $df$ . Звичайно параметри теоретичного розподілу оцінюються за даними вибірки, для чого теоретичні характеристики прирівнюються до їх вибіркових оцінок:

$$\begin{cases} M = \bar{x} \\ \sigma_x^2 = s_x^2 \end{cases}$$

Число цих умов дорівнює числу параметрів теоретичного закону розподілу. Для однопараметричних законів (експоненціального, Пуасона) прирівнюються лише центри розподілів, для двохпараметричних – прирівнюються також міри мінливості (дисперсії), для трьохпараметричних (і більше) – прирівнюються ще моменти вищих порядків.

Це приводить до того, що на  $k$  різниць  $(m_i - \hat{\mu}_i)$  накладається  $(p + 1)$  лінійних зв'язків, де  $p$  – число параметрів теоретичного закону:

$$\begin{cases} \sum (m_i - \hat{\mu}_i) = 0 & \text{наслідок } \sum \hat{p}_i = 1 \\ \sum (m_i - \hat{\mu}_i)x_i = 0 & \text{наслідок } M(x) = \bar{x} \\ \sum (m_i - \hat{\mu}_i)x_i^2 = 0 & \text{наслідок } \sigma_x^2 = s_x^2 \end{cases}$$

Тут через  $x_i$  позначені центри інтервалів.

Отже, число ступенів свободи для різниць  $(m_i - \hat{\mu}_i)$  дорівнює  $df = k - p - 1$ . Обчислену величину статистики Пірсона порівнюють з табличними значеннями  $\chi_{\alpha}^2(k - p - 1)$ , де  $\alpha$  – рівень значимості (0.05 чи 0.01). Якщо обчислене значення статистики Пірсона буде більше табличного, нуль-гіпотеза про згоду розподілів відхиляється.

*Синтаксис:*

**[a,b] = chi2test(data,k,alpha,dist,x,y,z)**

Опис:

Функція **[a,b] = chi2test(data,k,alpha,dist,x,y,z)** перевіряє за  $\chi^2$ -критерієм згоди Пірсона дані, що записані у векторі-рядку *data*. Об'єм вибірки *n* має бути не менше 20. Аргумент *k* – кількість ділянок, на які розбивається інтервал зміни даних в *data*. За  $20 \leq n < 50$  повинно бути  $k \in [5; 10]$ ; за  $50 \leq n < 100$  –  $k \in [10; 20]$ ; далі для  $n \geq 100$  –  $k \in [\sqrt{n}; n/5]$ .

Аргумент *alpha* визначає рівень значимості. На відміну від інших функцій, його треба задавати обов'язково (немає значень за замовченням). Аргумент *dist* – це рядок символів, що задає вид теоретичного розподілу, з яким порівнюється вибірка. Можливі значення цього параметра (імена розподілів) перелічені в табл. 10.1 (допустимо задавати як коротке, так і повне ім'я).

Таблиця 10.1

**Можливі значення аргументу dist**

Вид розподілу	Повне ім'я	Коротке ім'я
Біноміальний	'Binomial'	'bino'
Негативний біноміальний	'Negative Binomial'	'nbin'
Геометричний	'Geometric'	'geo'
Гіпергеометричний	'Hypergeometric'	'hyge'
Дискретний рівномірний	'Discrete Uniform'	'unid'
Пуасонівський	'Poisson'	'poiss'
Бета	'Beta'	'beta'
Експоненціальний	'Exponential'	'exp'
Гама	'Gamma'	'gam'
Логнормальний	'Lognormal'	'logn'
Нормальний	'Normal'	'norm'
Релеївський	'Rayleigh'	'rayl'
Вейбулівський	'Weibull'	'wbl'
Безперервний рівномірний	'Uniform'	'unif'
$\chi^2$ -розподіл Пірсона	'Chisquare'	'chi2'
Зміщений $\chi^2$ -розподіл Пірсона	'Noncentral Chi-square'	'ncx2'
F-розподіл Фішера	'F'	'f'
Зміщений F-розподіл Фішера	'Noncentral F'	'ncf'
t-розподіл Стюдента	'T'	't'
Зміщений t-розподіл Стюдента	'Noncentral t'	'nct'
Гамбела (крайніх значень)	'Extreme Value'	'ev'

Останні три аргументи функції  $x$ ,  $y$  і  $z$  – це параметри перевірюваного теоретичного розподілу. Залежно від виду розподілу таких параметрів може бути 1, 2 або 3.

У результативному параметрі  $a$  повертається обчислена  $\chi^2$ -статистика,  $a$  і  $b$  – критичне значення цієї статистики для перевірки 0-гіпотези. Якщо  $a < b$ , то на рівні значимості  $alpha$  0-гіпотезу про згоду розподілу з теоретичним законом можна прийняти.

*Приклади:*

1. Генерується вибірка зі 100 елементів, що розподілена за стандартним нормальним законом. Вибірка розбивається на 10 інтервалів і порівнюється з рівномірним законом на рівні значимості 0.05.

```
>> x=normrnd(0,1,1,100);
>> [a,b]=chi2test(x,10,0.05,'unif',1.5,2.9) | a = | b =
| 89.4000 | 14.0671
```

Оскільки одержане  $a > b$ , то нуль-гіпотеза про рівномірний розподіл відхиляється.

2. Та ж вибірка порівнюється зі стандартним нормальним законом на рівні значимості 0.01.

```
>> x=normrnd(0,1,1,100);
>> [a,b]=chi2test(x,10,0.01,'norm',0,1) | a = | b =
| 8.8000 | 18.4753
```

Тепер  $a < b$ , тобто нуль-гіпотеза про нормальний розподіл може бути прийнята.

3. Генерується вибірка з 200 елементів, розподіл якої випадково обраний з чотирьох: релеївського, експоненціального, нормального й рівномірного. Вибірка розбивається на 20 інтервалів. Далі провадиться тестування цієї вибірки на рівні значимості 0.05.

```
>> tdist={'exp','rayl','norm','unif'}; % види розподілів
>> tdistru={'експоненціальний','релеївський',...
'нормальний','рівномірний'}; % назви українською мовою
>> cd=length(tdist); % кількість розподілів
>> nd=unidrnd(cd); % випадково вибрали розподіл
>> disp(['Випадково обраний ' tdistru{nd} ' розподіл']);
>> n=200; % об'єм вибірки
>> k=20; % кількість інтервалів
>> if nd<3, % 1- параметричний розподіл з параметром 2
x=random(tdist{nd},2,1,n); % випадкова вибірка
else % 2-параметричний розподіл з параметрами 1 і 3
```

```

    x=random(tdist{nd},1,3,1,n); % випадкова вибірка
end
>> ph=cell(1,cd); % для найкращих параметрів
>> for id=1:cd, % визначаємо найкращі параметри
    if id<3, % потрібні позитивні значення
        phid=mle(abs(x),'dist',tdist{id}); % найкращі параметри
    else
        phid=mle(x,'dist',tdist{id}); % найкращі параметри
    end
    ph{id}=phid; % зберегли найкращі параметри
end
>> a=[]; % для статистик
>> for id=1:cd, % перевіряємо за chi2-критерієм
    phid=ph{id}; % знайдені параметри
    if id<3, % 1-параметричний розподіл
        [a1,b1]=chi2test(x,k,0.3,tdist{id},phid);
    else % 2-параметричний розподіл
        [a1,b1]=chi2test(x,k,0.3,tdist{id},phid(1),phid(2));
    end
    a=[a;a1]; % додали параметр
end
>> [am,iam]=min(a); % найменша статистика
>> disp([' Найкраще підходить ' tdistru{iam} ' розподіл'])
Випадково обраний нормальний розподіл
Найкраще підходить нормальний розподіл

```

## **KSTEST** Тест Колмогорова-Смірнова на непротириччя розподілу значень випадкової величини заданому закону

Перевіряється вибірка на відповідність заданому розподілу за допомогою критерію згоди Колмогорова-Смірнова. Цей тест заснований на порівнянні емпіричної функції розподілу (кумуляти) з інтегральною функцією теоретичного закону, параметри якого вважаються відомими.

За проведення тесту Колмогорова-Смірнова для серії заданих значень випадкової величини  $x$  означає частку вибірових значень  $X(:) \leq x$ . Максимальна різниця між очікуваною часткою й часток значень  $X(:) \leq x$  за вибіркою використовується як вибіркова статистика Колмогорова-Смірнова. Іншими словами, вибіркова статистика розраховується з вираження  $\max |F(x) - G(x)|$ , де  $F(x) = P(X(:) \leq x)$  – частка вибірових значень не більших заданого значення  $x$  (емпірична

функція розподілу);  $G(x)$  – значення інтегральної функції теоретичного закону розподілу для заданого  $x$ .

*Синтаксис:*

**H = kstest(X)**

**H = kstest(X,cdf)**

**H = kstest(X,cdf,alpha,tail)**

**[H,P,KSSTAT,CV] = kstest(X,cdf,alpha,tail)**

*Опис:*

Функція **H = kstest(X)** призначена для виконання тесту Колмогорова-Смірнова на непротивіччя розподілу значень випадкової величини стандартному нормальному закону (нормальному закону з нульовим математичним очікуванням і одиничною дисперсією) за вибіркою  $X$ , де  $X$  задається як вектор. Функція повертає скаляр  $H$ , що є результатом перевірки нульової гіпотези для критичного рівня значимості  $p_{кр}$  рівного 0.05. Нульова гіпотеза полягає в тому, що розподіл сукупності не суперечить стандартному нормальному закону. Альтернативна гіпотеза тесту Колмогорова-Смірнова полягає в тому, що розподіл генеральної сукупності суперечить стандартному нормальному закону. Нульова гіпотеза ухвалюється, якщо  $H = 0$  за  $p_{кр} = 0.05$ . Якщо  $H = 1$ , то нульова гіпотеза має бути відкинута для рівня значимості  $p_{кр} = 0.05$ .

Функція **H = kstest(X,cdf)** призначена для виконання тесту Колмогорова-Смірнова на непротивіччя розподілу значень випадкової величини довільному закону розподілу ймовірностей  $cdf$  за вибіркою  $X$ . Вхідний параметр  $cdf$  задається як матриця з розмірністю  $m \times 2$ , де  $m$  – число заданих значень випадкової величини  $X$ . Перший стовпець матриці  $cdf$  відповідає заданим значенням випадкової величини  $X$ . Другий стовпець матриці  $cdf$  розраховується за інтегральною теоретичною функцією розподілу в точках  $X$ . Завдання  $cdf$  у зазначеному вигляді є рекомендованим. Якщо  $cdf$  буде визначена як вектор, без стовпця відповідного  $X$ , значення  $G(x)$  буде розраховано методом

інтерполяції. Значення  $X$  повинні перебувати в інтервалі між мінімальним і максимальним значеннями першого стовпця матриці  $cdf$ . Якщо другий аргумент заданий як порожній вектор або матриця,  $cdf = []$ , у якості теоретичного розподілу буде використано стандартний нормальний закон.

Тест Колмогорова-Смірнова використовується у випадках, коли явно задана теоретична функція розподілу і її параметри. Якщо параметри закону розподілу розраховуються за вибірковими значеннями, використання тесту Колмогорова-Смірнова приведе до істотних похибок. Тому, якщо необхідно виконати перевірку на сукупності нормальному закону з невизначеними параметрами, рекомендується використовувати тест Пірсона або Лільєфорса.

Функція  **$H = kstest(X,cdf,alpha,tail)$**  у вхідному параметрі  $alpha$  дозволяє задати значення критичного рівня значимості для перевірки нульової гіпотези. За замовчуванням  $alpha = 0.05$ . Умовою прийняття нульової гіпотези є  $p \geq p_{кр}$ , де  $p_{кр}$  – критичний рівень значимості;  $p$  – рівень значимості, відповідний до вибіркової статистики Колмогорова-Смірнова. Вибір величини  $p_{кр}$  наданий досліднику. У більшості практичних випадків  $p_{кр}$  приймається рівним 0.05 або 0.01.

Вхідний параметр  $tail$  призначений для задання виду альтернативної гіпотези. Якщо  $tail = 0$ , альтернативною гіпотезою є  $F \neq G$  й перевірка гіпотези виконується для двосторонньої критичної області;  $tail = -1$  відповідає альтернативній гіпотезі  $F < G$ ;  $tail = 1$  відповідає альтернативній гіпотезі  $F > G$ . Значення за замовчуванням  $tail = 0$ . Вираження для розрахунку статистики Колмогорова-Смірнова будуть залежати від виду альтернативної гіпотези (табл. 10.2):

Таблиця 10.2

**Статистики Колмогорова-Смірнова**

Значення параметра $tail$	Вираження для статистик Колмогорова-Смірнова
$tail = 0$	$max F(x) - G(x) $

$tail = 1$	$max(G(x) - F(x))$
$tail = 1$	$max(F(x) - G(x))$

У функції **[H,P,KSSTAT,CV] = kstest(X,cdf,alpha,tail)** є наступні результативні параметри:

результат перевірки нульової гіпотези  $H$  для заданого критичного рівня значимості;

рівень значимості  $P$ , відповідний до вибіркового значення статистики Колмогорова-Смірнова;

значення вибіркової статистики Колмогорова-Смірнова  $KSSTAT$ ;

критичне значення статистики Колмогорова-Смірнова  $CV$ .

Величина  $CV$  призначена для перевірки нульової гіпотези. Якщо  $KSSTAT < CV$ , то нульова гіпотеза може бути прийнята, а якщо ні, то ухвалюється альтернативна гіпотеза. Таким чином, зазначена умова є альтернативною стосовно попередньої. Якщо результативний параметр  $CV = NaN$ , то величина  $P$  буде розрахована за асимптотичною формулою і для визначення значень  $H$  буде використано попереднє вираження замість умови  $KSSTAT < CV$ .

*Приклади:*

1. Тест Колмогорова-Смірнова на непротириччя генеральної сукупності стандартному нормальному закону. Об'єм вибірки – 15 елементів. Функція повертає результат перевірки нульової гіпотези для критичного рівня значимості 0.05. Графічне представлення розподілу вибірових значень виконується за допомогою функції **histfit** (рис. 10.1).

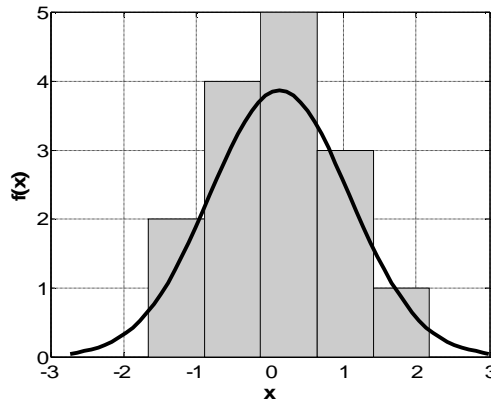


Рис. 10.1. Гістограма вибірки і графік щільності нормального закону

Оператори:

```
>> x=normrnd(0,1,15,1);
>> H=kstest(x)
H=
    0
>> histfit(x,5), grid on
```

```
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bff(x)') % мітка осі OY
```

2. Тест Колмогорова-Смірнова на непротириччя закону Вейбула з параметром масштабу рівним 1 і коефіцієнтом форми 2. Вибірка розподілена за рівномірним законом. Об'єм вибірки – 40 елементів. Функція повертає наступні результати перевірки нульової гіпотези для критичного рівня значимості 0.01: рівень значимості, відповідний до вибіркового значення статистики Колмогорова-Смірнова; значення вибіркової статистики Колмогорова-Смірнова; критичне значення статистики Колмогорова-Смірнова.

```
>> x=unifrnd(0,4,40,1);
>> X=0:0.5:4;
>> Y=wblcdf(X,1,2);
>> cdf=[X' Y'];
>> [H,P,KSSTAT,CV]=kstest(x,cdf,0.01)
H =
    1
```

```
P =
    4.2964e-013
KSSTAT =
    0.5910
CV =
    0.2521
```

3. Вид розподілу вибіркової статистики Колмогорова-Смірнова. Розподіл генеральної сукупності формується як композиція нормального й рівномірного законів. Нульова гіпотеза полягає в непротириччі генеральної сукупності стандартному нормальному закону. Об'єм вибірки

дорівнює 50 елементам. Кількість вибірок дорівнює 25. Графічна оцінка відповідності нормальному закону вибіркової статистики Колмогорова-Смірнова виконується за допомогою функції **qqplot** (рис. 10.2):

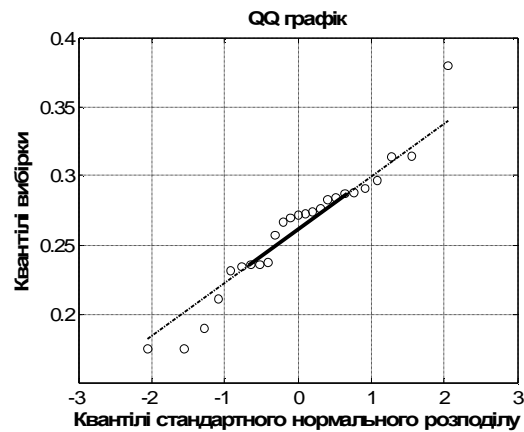


Рис. 10.2. QQ-графіки вибірок

Оператори:

```
>> m_norm=0;
>> sigma_norm =1;
>> m_unif=0;
>> sigma_unif=3/1.73;
>> n=50;
>> N=25;
>> for i=1:N
    x = normrnd(m_norm,sigma_norm,n,1)+...
        unifrnd(m_unif-1.73*sigma_unif,m_unif+1.73*sigma_unif,n,1);
    [H,P,KSSTAT(i),CV] = kstest(x);
end;
>> qqplot(KSSTAT), grid on
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14) % шрифт
>> title('\bfQQ графік');
>> xlabel('\bfКвантілі стандартного нормального розподілу') % мітка осі OX
>> ylabel('\bfКвантілі вибірки') % мітка осі OY
```

4. Залежність величини рівня значимості за перевірки нульової гіпотези від об'єму вибірки й дисперсії рівномірного закону, що є складовою композиції законів розподілу випадкової величини  $x$ . Розподіл генеральної сукупності формується як композиція нормального й рівномірного законів. Нульова гіпотеза полягає в непротириччі генеральної сукупності стандартному нормальному закону. Параметри нормальної складової дорівнюють: математичне очікування – 0,

дисперсія – 1. Математичне очікування рівномірної складової прийняте рівним нулю (рис. 10.3).

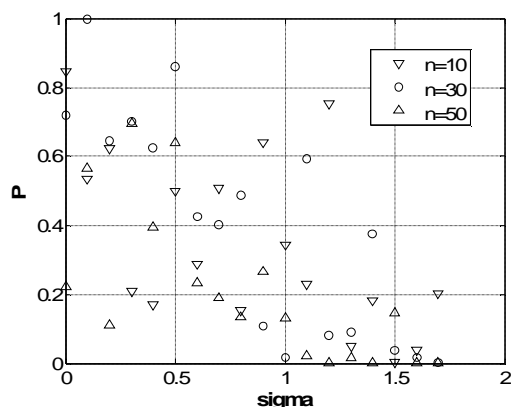


Рис. 10.3. Залежність величини рівня значимості від об'єму вибірки й дисперсії рівномірного закону

Оператори:

```
>> sigma = 0:0.1:3/1.73;
>> n=10;
>> for i=1:length(sigma)
    x = normrnd(0,1,n,1)+unifrnd(-1.73*sigma(i), 1.73*sigma(i),n,1);
    [H,p1(i),KSSTAT,CV] = kstest(x);
end;
>> n=30;
>> for i=1:length(sigma)
    x = normrnd(0,1,n,1)+unifrnd(-1.73*sigma(i), 1.73*sigma(i),n,1);
    [H,p2(i),KSSTAT,CV] = kstest(x);
end;
>> n=50;
>> for i=1:length(sigma)
    x = normrnd(0,1,n,1)+unifrnd(-1.73*sigma(i), 1.73*sigma(i),n,1);
    [H,p3(i),KSSTAT,CV] = kstest(x);
end;
>> plot(sigma,p1,'vk', sigma,p2,'ok', sigma,p3,'^k'), grid on
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14)
>> xlabel('\bfsigma') % мітка осі OX
>> ylabel('\bfP') % мітка осі OY
```

5. У прикладі генеруються релеївська вибірка, далі вирішується задача добору найкращого теоретичного розподілу. Вид розподілу передбачається релеївським, найкращий параметр визначається за допомогою функції `raylfit`. За допомогою двостороннього критерію

Колмогорова-Смірнова перевіряється відповідність вибірки передбачуваному теоретичному розподілу (рис. 10.4).

```
>> b=2; % теоретичний параметр
>> n=100; % об'єм вибірки
>> x=raylrnd(b,n,1); % випадкова вибірка
>> bhat=raylfit(x); % оцінка параметра за даними
>> fprintf('Теоретичне b=%10.8f;\nЗнайдене b=%10.8f.\n',b,bhat);
>> xt=[0:0.01:max(x)+0.1]'; % рівномірні абсциси
>> yt=raylcdf(xt,bhat); % теоретичні ординати
>> [h,p]=kstest(x,[xt yt],0.3); % тест Колмогорова-Смірнова
>> fprintf('Критичне р-значення р=%10.8f.\n',p);
>> if h,
    disp('Відкидаємо 0-гіпотезу.');
```

else  
disp('Ухвалюємо 0-гіпотезу.');

```
end
>> cdfplot(x); % графік емпіричної F(x)
>> hold on;
>> plot(xt,yt,'k'); % теоретична F(x)
hold off;
>> set(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title(['\bfВибіркова й теоретична функції розподілу'])
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfF(x)') % мітка осі OY
Теоретичне b=2.00000000;
Знайдене b=2.03861660.
Критичне р-значення р=0.98981036.
Ухвалюємо 0-гіпотезу.
```

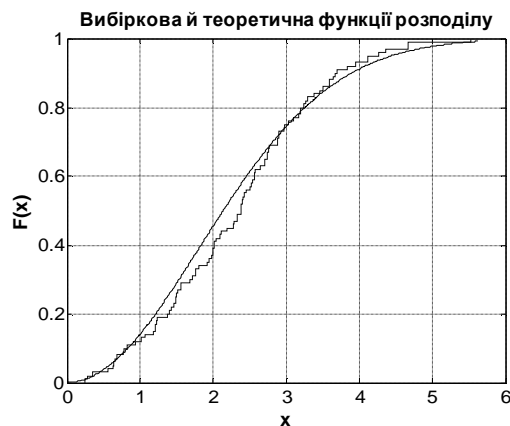


Рис. 10.4. Теоретична й вибіркова функції розподілу

## **KSTEST2** Тест Колмогорова-Смірнова для порівняння розподілів двох сукупностей значень випадкових величин

Перевіряються дві вибірки на відповідність їх функцій розподілу за допомогою критерію згоди Колмогорова-Смірнова. В ході проведення тесту Колмогорова-Смірнова для серії заданих значень випадкової величини  $x$  порівнюються частки вибірових значень, що задовольняють умовам  $X_1(\cdot) \leq x$  і  $X_2(\cdot) \leq x$ . Максимальна різниця між зазначеними частками вибірових значень, відповідних до зазначених умов, використовується як вибірова статистика Колмогорова-Смірнова. Тобто, статистику Колмогорова-Смірнова можна представити як  $\max |F_1(x) - F_2(x)|$ , де  $F_1(x)$ ,  $F_2(x)$  – частки вибірових значень, що задовольняють умовам  $X_1(\cdot) \leq x$  і  $X_2(\cdot) \leq x$  відповідно (кумуляти двох вибірок). Елементи векторів  $X_1(i)$  і  $X_2(i)$ , що дорівнюють NaN, вважаються як пропущені значення й ігноруються під час розрахунку.

*Синтаксис:*

**H = kstest2(X1,X2)**  
**H = kstest2(X1,X2,alpha,tail)**  
**[H,P,KSSTAT] = kstest2(...)**

*Опис:*

Функція **H = kstest2(X1,X2)** дозволяє виконати тест Колмогорова-Смірнова для порівняння законів розподілу двох сукупностей значень випадкових величин за вибірками  $X_1$ ,  $X_2$ . Вибірki  $X_1$ ,  $X_2$  задаються як вектори з розмірностями  $n_1 \times 1$ ,  $n_2 \times 1$  відповідно. Функція повертає скаляр  $H$ , що є результатом перевірки нульової гіпотези для критичного значення рівня значимості  $p_{кр}$  рівного 0.05. Нульова гіпотеза полягає в тому, що дві генеральні сукупності значень випадкових величин розподілені по тому самому безперервному закону. Альтернативна гіпотеза полягає в тому, що дві генеральні сукупності значень випадкових величин розподілені за різними безперервними законами. Нульова гіпотеза ухвалюється якщо  $H = 0$  при  $p_{кр} = 0.05$ . Якщо  $H = 1$ , то нульова гіпотеза має бути відкинута при  $p_{кр} = 0.05$ .

Функція **H = kstest2(X1,X2,alpha,tail)** в параметрі *alpha* дозволяє задати значення критичного рівня значимості для перевірки нульової гіпотези. За замовчуванням *alpha* = 0.05. Умова прийняття нульової гіпотези:  $p \geq p_{кр}$ , де  $p_{кр}$  – критичний рівень значимості;  $p$  – рівень значимості, відповідний до вибіркової статистики Колмогорова-Смірнова. Вибір величини  $p_{кр}$  наданий досліднику. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

Вхідний параметр *tail* призначений для задання виду альтернативної гіпотези. Його можливі значення наведено у табл. 10.3.

Таблиця 10.3

### Статистики Колмогорова-Смірнова

Значення параметра tail	Вираження для статистик Колмогорова-Смірнова
$tail = 0$	$max F_1(x) - F_2(x) $
$tail = -1$	$max(F_2(x) - F_1(x))$
$tail = 1$	$max(F_1(x) - F_2(x))$

Якщо  $tail = 0$ , альтернативною гіпотезою є  $F_1 \neq F_2$  і перевірка виконується для двосторонньої критичної області;  $tail = -1$  відповідає альтернативній гіпотезі  $F_1 < F_2$ ;  $tail = 1$  відповідає альтернативній гіпотезі  $F_1 > F_2$ . Значення за замовчуванням  $tail = 0$ . Вираження для розрахунку вибіркової статистики Колмогорова-Смірнова буде залежати від виду альтернативної гіпотези.

У функції **[H,P,KSSTAT] = kstest2(...)** результативними параметрами є:

результат перевірки нульової гіпотези  $H$  для заданого критичного рівня значимості;

рівень значимості  $P$ , відповідний до вибіркового значення статистики Колмогорова-Смірнова;

значення вибіркової статистики Колмогорова-Смірнова  $KSSTAT$ .

Асимптотичне значення рівня значимості  $P$  має мінімальну похибку за великих об'ємів вибірок  $X_1$  і  $X_2$ , або якщо виконується умова  $(n_1 * n_2) / (n_1 + n_2) \geq 4$ .

Приклади:

1. Тест Колмогорова-Смірнова про згоду розподілів двох генеральних сукупностей за вибірковими значенням. Об'єм вибірок – 15 елементів. Функція повертає результат перевірки нульової гіпотези для критичного рівня значимості 0.05.

```
>> x=normrnd(0,1,15,1);
>> y=normrnd(0,1,15,1);
>> H=kstest2(x,y)
H=
    0
>> [fx,xx]=ksdensity(x);
>> [fy,xy]=ksdensity(y);
>> plot(xx,fx,'-',xy,fy,'--'), grid on
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx, y') % мітка осі OX
>> ylabel('\bff(x), f(y)') % мітка осі OY
```

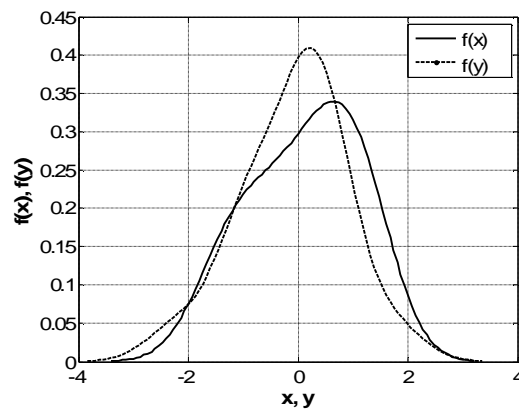


Рис. 10.5. Розподіл вибіркових значень

Графічне представлення розподілів вибіркових значень (рис. 10.5) виконується за допомогою функції непараметричного згладжування **ksdensity**. 0-гіпотеза про згоду двох розподілів приймається ( $H = 0$ ).

2. Тест Колмогорова-Смірнова на згоду розподілів двох сукупностей за вибірковими значеннями. Вибірки розподілені за нормальним і рівномірним законами з однаковими математичними очікуваннями й дисперсіями. Об'єм вибірок – 20 елементів. Функція повертає результат перевірки нульової гіпотези для критичного рівня значимості 0.05. Графічне представлення розподілів вибіркових значень виконується за допомогою функції непараметричного згладжування **ksdensity** (рис.10.6).

```
>> x=normrnd(0,1,20,1);
>> y=unifrnd(-1.73,1.73,20,1);
>> H=kstest2(x,y)
H=
    0
>> [fx,xx]=ksdensity(x);
>> [fy,xy]=ksdensity(y);
>> plot(xx,fx,'-',xy,fy,'--'), grid on
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx, y') % мітка осі OX
>> ylabel('\bff(x), f(y)') % мітка осі OY
```

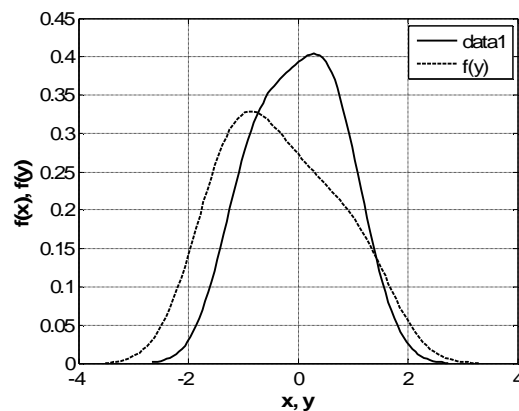


Рис. 10.6. Функції розподілу двох вибірок

0-гіпотеза приймається.

3. Тест Колмогорова-Смірнова про згоду розподілів двох генеральних сукупностей за вибірковими значеннями. Вибірки розподілені за нормальним і рівномірним законами з різними математичними очікуваннями й дисперсіями. Об'єм вибірок – 30 і 50 елементів. Функція повертає результат перевірки нульової гіпотези для критичного рівня значимості **0.02**. Графічне представлення розподілів вибіркових значень (рис. 10.7) виконується за допомогою функції непараметричного згладжування **ksdensity**.

Оператори:

```
>> x=normrnd(0,1,30,1);
>> y=unifrnd(-5,5,50,1);
>> H=kstest2(x,y,0.02)
H=
    1
>> [fx,xx]=ksdensity(x);
```

```
>> [fy,xy]=ksdensity(y);
>> plot(xx,fx,'-',xy,fy,'--'), grid on
>> set(get(gcf,'CurrentAxes'),...
'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx, y') % мітка осі OX
>> ylabel('\bff(x), f(y)') % мітка осі OY
```

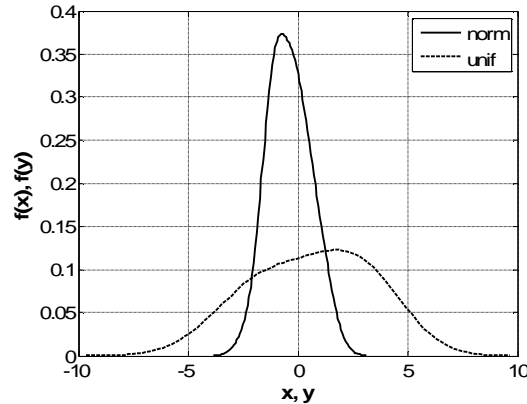


Рис.10.7. Функції розподілу двох вибірок

0-гіпотеза не приймається ( $H = 1$ ).

4. Вид розподілу вибіркової статистики Колмогорова-Смірнова за перевірки нульової гіпотези для генеральних сукупностей, розподілених за нормальним і рівномірним законами. Об'єм вибірок дорівнює 50 елементам. Кількість вибірок дорівнює 25. Графічна оцінка відповідності вибіркової статистики Колмогорова-Смірнова нормальному закону виконується за допомогою функції **qqplot** (рис. 10.8).

```
>> m_norm=0;
>> sigma_norm =1;
>> m_unif=0;
>> sigma_unif=3/1.73;
>> n=50;
>> N=25;
>> for i=1:N
    x=normrnd(m_norm, sigma_norm,n,1);
    y=unifrnd(m_unif-1.73*sigma_unif,m_unif+1.73*sigma_unif,n,1);
    [H,P,KSSTAT(i)]=kstest2(x,y);
end;
>> qqplot(KSSTAT), grid on
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14) % шрифт
>> title('\bfQQ графік');
```

```
>> xlabel('\bfКвантили стандартного нормального розподілу') % мітка осі OX
>> ylabel('\bfКвантили вибірки') % мітка осі OY
```

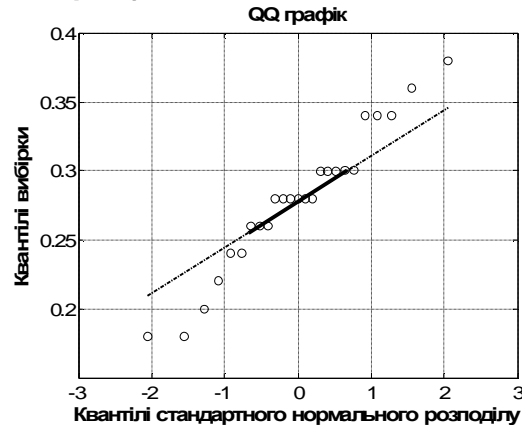


Рис. 10.8. QQ графіки даних вибірки і стандартного нормального розподілу

На рис. 10.8 бачимо помітні відхилення точок від прямої лінії, тобто є різниця між розподілами вибірок.

5. У прикладі генеруються дві вибірки нормальна і рівномірна з однаковими генеральними середніми й дисперсіями. Вони порівнюються за допомогою двостороннього критерію Колмогорова-Смірнова на збіг законів розподілу генеральних сукупностей.

```
>> x=normrnd(3,2/sqrt(3),100,1); % 1-а випадкова вибірка
>> y=unifrnd(1,5,1,200); % 2-а випадкова вибірка
>> [h,p]=kstest2(x,y,0.3); % тест Колмогорова-Смірнова
>> fprintf('Критичне р-значення р=%10.8f.\n',p);
>> if h,
    disp('Відкидаємо 0-гіпотезу.');
```

```
else
    disp('Ухвалюємо 0-гіпотезу.');
```

```
end
>> cdfplot(x); % графік емпіричної F1(x)
>> hold on;
>> cdfplot(y); % графік емпіричної F2(x)
>> hold off;
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title(['\bfВибіркові функції розподілу']) % заголовок
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfF(x)') % мітка осі OY
```

Критичне р-значення  $p=0.27451548$ .  
Відкидаємо 0-гіпотезу.

На рис. 10.9 показані графіки вибірових функцій розподілу.

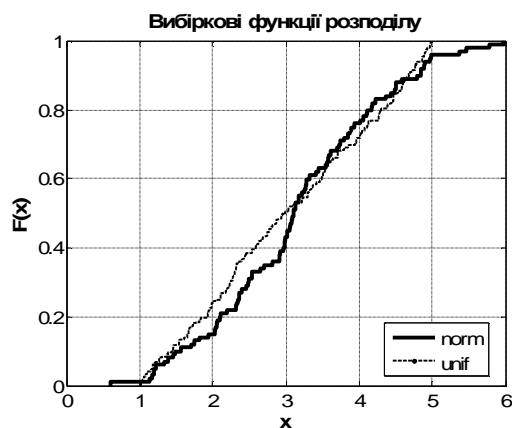


Рис. 10.9. Функції розподілу двох вибірок

**JBTEST** Тест Ярки-Бера на непротириччя розподілу значень випадкової величини нормальному закону

Тест Ярки-Бера використовується у випадках, коли невідомі середнє й дисперсія генеральної сукупності. Перевірка нульової гіпотези заснована на розрахунку оцінок коефіцієнтів асиметрії й ексцесу за вибіркою  $X$ . У випадку справедливості нульової гіпотези вибіркоче значення коефіцієнта асиметрії повинне приблизно дорівнювати нулю, а вибіркоче значення ексцесу – трьом. Слід зазначити, що у вітчизняній літературі коефіцієнт ексцесу визначається за формулою

$$k = \frac{E(X - \bar{X})^4}{s_x^4} - 3, \quad \text{де}$$

$s_x$  – точкова оцінка середнього квадратичного відхилення,  $E(X - \bar{X})^4$  – емпіричний четвертий центральний момент,  $\bar{X}$  – вибіркоче середнє арифметичне,  $X$  – вектор вибіркових значень. Отже, за цією формулою коефіцієнт ексцесу нормального закону буде дорівнювати 0.

Під час проведення тесту Ярки-Бера визначаються відхилення вибіркових коефіцієнтів асиметрії й ексцесу щодо очікуваних значень за умови справедливості нульової гіпотези для заданого об'єму вибірки. Для оцінки відхилень вибіркових значень коефіцієнтів асиметрії і ексцесу використовується статистика хі-квадрат. Тест Ярки-Бера

використовується за умови великого об'єму вибірки  $X$ . Для малого об'єму вибірки використовується тест Лільєфорса.

*Синтаксис:*

**$h = \text{jbtest}(X)$**

**$h = \text{jbtest}(X, \alpha)$**

**$[h, P, \text{JBSTAT}, \text{CV}] = \text{jbtest}(X, \alpha)$**

*Опис:*

Функція  **$h = \text{jbtest}(X)$**  призначена для виконання тесту Яркі-Бера на непротириччя розподілу сукупності значень випадкової величини нормального закону за вибіркою  $X$ , що задається як вектор. Функція повертає скаляр  $h$ , що є результатом перевірки нульової гіпотези для критичного рівня значимості  $p_{кр}$  рівного 0.05. Нульова гіпотеза полягає в тому, що розподіл сукупності значень випадкової величини не суперечить нормальному закону. Альтернативна гіпотеза тесту Яркі-Бера полягає в тому, що розподіл генеральної сукупності суперечить нормальному закону. Нульова гіпотеза ухвалюється, якщо  $h = 0$  за  $p_{кр} = 0.05$ . Якщо  $h = 1$ , то нульова гіпотеза має бути відкинута за  $p_{кр} = 0.05$ .

Функція  **$h = \text{jbtest}(X, \alpha)$**  в параметрі *alpha* дозволяє задати значення критичного рівня значимості для перевірки нульової гіпотези. За замовчуванням  **$\alpha = 0.05$** . Умовою прийняття нульової гіпотези є  $p \geq p_{кр}$ , де  $p_{кр}$  – критичний рівень значимості;  $p$  – рівень значимості, відповідний до вибіркової статистики хі-квадрат. Вибір величини  $p_{кр}$  наданий досліднику. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

У функції  **$[h, P, \text{JBSTAT}, \text{CV}] = \text{jbtest}(X, \alpha)$**  результативними параметрами є:

результат перевірки нульової гіпотези  $h$  для заданого критичного рівня значимості;

рівень значимості  $P$ , відповідний до вибіркового значення статистики хі-квадрат;

значення вибіркової статистики хі-квадрат  $\text{JBSTAT}$ ;

критичне значення статистики  $\chi^2$ -квадрат  $CV$ .

Величина  $CV$  призначена для перевірки нульової гіпотези. Якщо  $JBSTAT < CV$ , то нульова гіпотеза може бути прийнята, а якщо ні, то ухвалюється альтернативна гіпотеза. Таким чином, зазначена умова є альтернативною стосовно попередньої.

*Приклади:*

1. Тест Ярки-Бера на непротириччя нормальному закону генеральної сукупності, розподіленої за нормальним законом, за вибірковими значеннями. Об'єм вибірки – 25 елементів. Функція повертає результат перевірки нульової гіпотези. Графічне представлення розподілу вибіркових значень виконується за допомогою функції `histfit` (рис. 10.10).

```
>> x = normrnd(0,1,25,1);  
>> h = jbstest(x)  
h =  
    0  
>> histfit(x,7), grid on
```

```
>> set(get(gcf,'CurrentAxes'),...  
      'Fontname','Arial Cyr','FontSize',14)  
>> xlabel('\bfx') % мітка осі OX  
>> ylabel('\bff(x)') % мітка осі OY
```

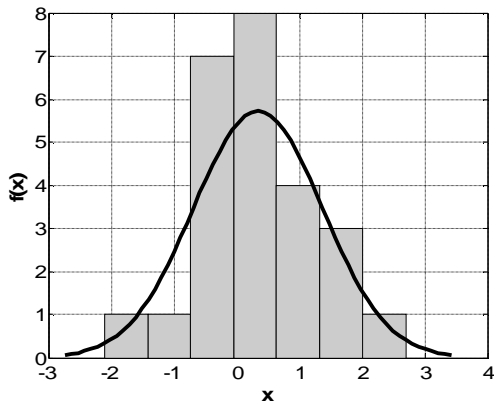


Рис. 10.10. Гістограма і графік щільності ймовірностей вибірки

0-гіпотеза приймається ( $h = 0$ ).

2. Тест Ярки-Бера на непротириччя нормальному закону генеральної сукупності, розподіленої за нормальним законом, за вибірковими значеннями. Об'єм вибірки – 30 елементів. Функція повертає результат перевірки нульової гіпотези для критичного значення

рівня значимості 0.01. Графічне представлення розподілу вибірових значень виконується за допомогою функції **qqplot** (рис. 10.11).

```
>> x=normrnd(0,1,30,1);  
>> alpha=0.01;  
>> h=jbtest(x,alpha)  
h=  
    0  
>> qqplot(x), grid on  
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14)  
>> title("\bfQQ графік");  
>> xlabel("\bfКвантили стандартного нормального розподілу") % мітка осі OX  
>> ylabel("\bfКвантили вибірки") % мітка осі OY
```

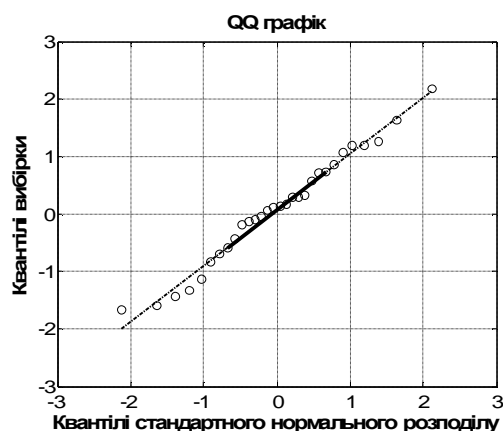


Рис. 10.11. **Графік розподілу вибірових значень**

0-гіпотеза приймається ( $h = 0$ ).

3. Тест Ярки-Бера на непротириччя нормальному закону генеральної сукупності за вибіровими значеннями. Розподіл генеральної сукупності представляє композицію нормального й рівномірного законів. Об'єм вибірки – 30 елементів. Функція повертає результат перевірки нульової гіпотези для критичного рівня значимості 0.01. Графічне представлення розподілу вибірових значень(рис. 10.12) виконується за допомогою функції **qqplot**.

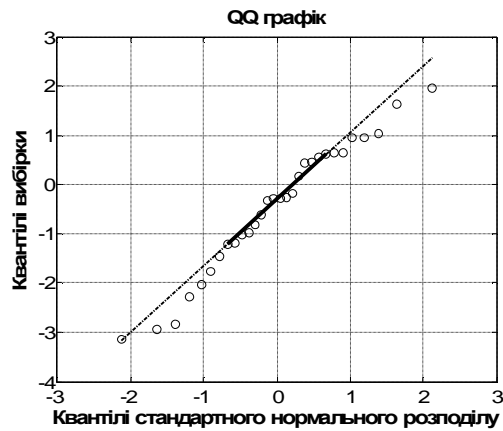


Рис. 10.12. Графік розподілу вибірових значень

Нульова гіпотеза не відхиляється ( $h = 0$ ).

Оператори:

```
>> x = normrnd(0,1,30,1)+unifrnd(-2,1,30,1);
>> alpha=0.01;
>> h = jbtest(x,alpha)
h =
    0
>> qqplot(x), grid on
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14)
>> title('\bfQQ графік');
>> xlabel('\bfКвантілі стандартного нормального розподілу')
>> ylabel('\bfКвантілі вибірки')
```

4. З декількох розподілів випадковим образом вибирається одне й перевіряється на відповідність нормальному розподілу за допомогою критерію згоди Ярки-Бера

```
>> namedistr={'beta' 'chi2' 'exp' 'norm' 'rayl' 'unif' 'wbl'}; % види розподілів
>> ndistr=length(namedistr); % кількість розподілів
>> ns=20; % кількість вибірок
>> distr=unidrnd(ndistr,1,ns); % випадкові види розподілів
>> for k=1:ns, % проводимо тести Ярки-Бера
    d=distr(k); % номер розподілу
    name=char(namedistr{d}); % назва розподілу
    if ismember(d,[2,3,5]), % однопараметричні розподіли
        x=random(name,1,100,1); % вибірка
    else % двухпараметричні розподіли
        x=random(name,1,2,100,1); % вибірка
    end
    h=jbtest(x,0.2); % перевіряємо на нормальність
    fprintf('%2.0f. Теоретичний розподіл %s; тест Ярки-Бера=%d.\n',k,name,h);
end
```

1. Теоретичний розподіл exp; тест Ярки-Бера=1.
2. Теоретичний розподіл beta; тест Ярки-Бера=1.
3. Теоретичний розподіл exp; тест Ярки-Бера=1.

4. Теоретичний розподіл wbl; тест Яркі-Бера=1.
5. Теоретичний розподіл beta; тест Яркі-Бера=1.
6. Теоретичний розподіл rayl; тест Яркі-Бера=1.
7. Теоретичний розподіл exp; тест Яркі-Бера=1.
8. Теоретичний розподіл chi2; тест Яркі-Бера=1.
9. Теоретичний розподіл exp; тест Яркі-Бера=1.
10. Теоретичний розподіл unif; тест Яркі-Бера=1.
11. Теоретичний розподіл unif; тест Яркі-Бера=1.
12. Теоретичний розподіл beta; тест Яркі-Бера=1.
13. Теоретичний розподіл exp; тест Яркі-Бера=1.
14. Теоретичний розподіл norm; тест Яркі-Бера=0.
15. Теоретичний розподіл unif; тест Яркі-Бера=1.
16. Теоретичний розподіл unif; тест Яркі-Бера=1.
17. Теоретичний розподіл wbl; тест Яркі-Бера=1.
18. Теоретичний розподіл wbl; тест Яркі-Бера=1.
19. Теоретичний розподіл norm; тест Яркі-Бера=0.
20. Теоретичний розподіл exp; тест Яркі-Бера=1.

Бачимо, що логічний символ 0 відповідає нормальному розподілу.

5. Залежність величини рівня значимості під час перевірки нульової гіпотези від об'єму вибірки і дисперсії рівномірного закону, що є складовою композиції законів розподілу випадкових величин. Розподіл генеральної сукупності є композицією нормального й рівномірного законів. Параметри нормальної складової дорівнюють: математичне очікування – 0, дисперсія – 1. Математичне очікування рівномірної складової прийняте рівним нулю (рис. 10.13).

```
>> sigma=0:0.1:3/1.73;
>> n=10;
>> for i=1:length(sigma)
    x=normrnd(0,1,n,1)+unifrnd(-1.73*sigma(i), 1.73*sigma(i),n,1);
    [H,p1(i),JBSTAT,CV]=jbtest(x);
end;
>> n=30;
>> for i=1:length(sigma)
    x=normrnd(0,1,n,1)+unifrnd(-1.73*sigma(i), 1.73*sigma(i),n,1);
    [H,p2(i),JBSTAT,CV]=jbtest(x);
end;
>> n=50;
>> for i=1:length(sigma)
    x=normrnd(0,1,n,1)+unifrnd(-1.73*sigma(i),1.73*sigma(i),n,1);
    [H,p3(i),JBSTAT,CV]=jbtest(x);
end;
>> plot(sigma,p1,'vk', sigma,p2,'ok', sigma,p3,'+k'), grid on
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14)
>> xlabel('\bfsigma') % мітка осі OX
>> ylabel('\bfP') % мітка осі OY
>> legend('n = 10', 'n = 30', 'n = 50')
```

Подивимось на вихідні дані функції **jbtest**:

```
>> [H,p1(i),JBSTAT,CV]=jbtest(x)
```

```
H=
```

```
0
```

```
p1 =
```

```
Columns 1 through 9
```

```
0.9691 0.5292 0.3125 0.3153 0.2835 0.7230 0.5455 0.8328 0.4811
```

```
Columns 10 through 18
```

```
0.6693 0.8297 0.2048 0.8641 0.1946 0.3173 0.5321 0.7457 0.3668
```

```
JBSTAT =
```

```
2.0058
```

```
CV =
```

```
5.9915
```

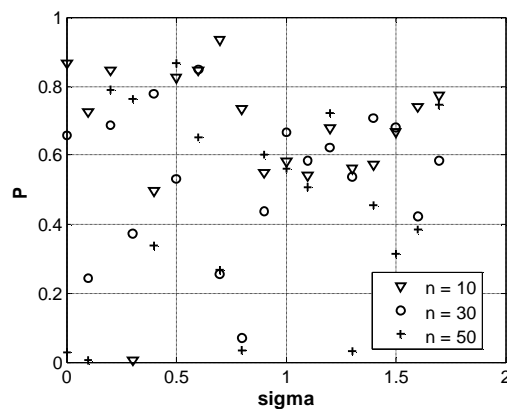


Рис. 10.13. Залежність величини рівня значимості від об'єму вибірки й дисперсії рівномірного закону

Нульова гіпотеза приймається ( $H = 0$ ). Результативні дані 3-х рівнів значимості  $P$  до вибіркового значення статистики хі-квадрат для 18 вибірок відповідають різним об'ємам вибірок. Значення вибіркової статистики хі-квадрат  $JBSTAT = 2.0058$ . Критичне значення статистики хі-квадрат  $CV = 5.9915$ . Бачимо, що  $JBSTAT < CV$ . Остання нерівність використана для висновку  $H = 0$ , що відповідає нульовій гіпотезі.

### **LILLIETEST** Тест Лільєфорса на непротириччя розподілу значень випадкової величини нормальному закону

У критерії згоди Лільєфорса перевіряється 0-гіпотеза про те, що вибірка  $x$  узята з генеральної сукупності з нормальним розподілом з будь-якими середнім і дисперсією. Альтернативною є гіпотеза про те, що

генеральна сукупність має розподіл, відмінний від нормального. У тесті порівнюється емпіричний розподіл  $x$  з нормальним, що має ті ж самі середнє й дисперсію, що й  $x$ . Це схоже на тест Колмогорова-Смірнова, але параметри теоретичного розподілу обчислюються за самою вибіркою, а не відомі заздалегідь.

*Синтаксис:*

**H = lillietest(X)**

**H = lillietest(X,alpha)**

**[H,P,LSTAT,CV] = lillietest(X,alpha)**

*Опис:*

Функція **H = lillietest(X)** призначена для виконання тесту Лільєфорса на непротириччя розподілу генеральної сукупності значень випадкової величини нормальному закону за вибіркою  $X$ , що задається як вектор. Функція повертає скаляр  $H$ , що є результатом перевірки нульової гіпотези для критичного рівня значимості  $p_{кр}$  рівного 0.05. Нульова гіпотеза полягає в тому, що розподіл генеральної сукупності значень випадкової величини не суперечить нормальному закону з невідомими середнім арифметичним і дисперсією. Альтернативна гіпотеза тесту Лільєфорса полягає в тому, що розподіл генеральної сукупності значень випадкової величини суперечить нормальному закону. Нульова гіпотеза ухвалюється якщо  $H = 0$  при  $p_{кр} = 0.05$ . Якщо  $H = 1$ , то нульова гіпотеза має бути відкинута для  $p_{кр} = 0.05$ .

Функція **H = lillietest(X,alpha)** в параметрі *alpha* дозволяє задати значення критичного рівня значимості для перевірки нульової гіпотези. За замовчуванням  $alpha = 0.05$ . Умова прийняття нульової гіпотези  $p \geq p_{кр}$ , де  $p_{кр}$  – критичний рівень значимості;  $p$  – рівень значимості, що відповідає вибірковій тестовій статистиці. Вибір величини  $p_{кр}$  наданий досліднику й повинен змінюватися в межах від 0.01 до 0.1. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

У функції **[H,P,LSTAT,CV] = lillietest(X,alpha)** результативними параметрами є:

результат перевірки нульової гіпотези  $H$  для заданого критичного рівня значимості;

рівень значимості  $P$  відповідний до вибіркового значення тестової статистики;

значення вибіркової тестової статистики  $LSTAT$ ;

критичне значення тестової статистики  $CV$ .

Величина  $CV$  призначена для перевірки нульової гіпотези. Якщо  $LSTAT < CV$ , то нульова гіпотеза може бути прийнята, а якщо ні, то ухвалюється альтернативна гіпотеза. Таким чином, зазначена умова є альтернативною стосовно  $p \geq p_{кр}$ .

Величина  $P$  визначається методом лінійної інтерполяції по таблиці Лільєфорса на основі тестової статистики  $LSTAT$ . Якщо  $LSTAT$  виходить за межі табличних значень, функція повертає  $P = \text{NaN}$ , але  $H$  показує можливість прийняття нульової гіпотези.

*Приклади:*

1. Тест Лільєфорса на непротириччя нормальному закону розподілу генеральної сукупності за вибілковими значеннями. Об'єм вибірки – 10 елементів. Функція повертає результат перевірки нульової гіпотези. Графічне представлення розподілу вибіркових значень виконується за допомогою функції **histfit** (рис. 10.14).

```
>> x = normrnd(0,1,10,1);
>> H = lillietest(x)
H =
    0
>> histfit(x,5), grid on
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14)
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfn') % мітка осі OY
```

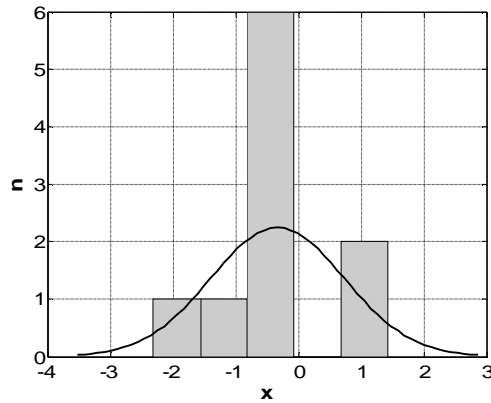


Рис. 10.14. Гістограма і графік щільності ймовірностей

2. Тест Лільєфорса на непротириччя нормальному закону розподілу генеральної сукупності за вибірковими значеннями. Розподіл генеральної сукупності є композицією нормального й рівномірного законів. Об'єм вибірки – 10 елементів. Функція повертає результат перевірки нульової гіпотези для критичного рівня значимості 0.01. Графічне представлення розподілу вибіркових значень (рис. 10.15) виконується за допомогою функції **qqplot**.

```
>> x=normrnd(0,1,10,1)+unifrnd(-2,1,10,1);
>> alpha=0.01;
>> H=lillietest(x,alpha)
H=
    0
>> qqplot(x), grid on
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14) % шрифт
>> title('\bfQQ графік');
>> xlabel('\bfКвантили стандартного нормального розподілу') % мітка осі OX
>> ylabel('\bfКвантили вибірки') % мітка осі OY
```

0-гіпотеза приймається.

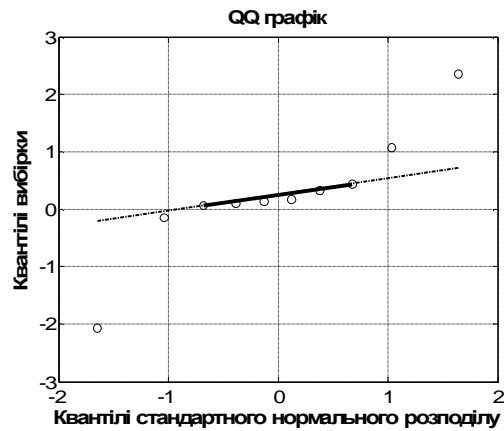


Рис. 10.15. Розподіл вибірових значень

3. Тест Лільєфорса на непротириччя нормальному закону розподілу генеральної сукупності за вибіровими значеннями. Розподіл генеральної сукупності є композицією нормального й рівномірного законів. Об'єм вибірки – 15 елементів. Графічне представлення розподілу вибірових значень виконується за допомогою функції **qqplot** (рис. 10.16). Функція повертає результат перевірки нульової гіпотези для критичного рівня значимості 0.01, рівень значимості  $P$ , відповідний до значення вибірової тестової статистики, значення вибірової тестової статистики, критичне значення тестової статистики.

```
>> x=normrnd(0,1,15,1)+unifrnd(-3,3,15,1);
```

```
>> alpha=0.01;
```

```
>> [H,P,LSTAT,CV]=lillietest(x,alpha)
```

H =	P =	LSTAT =	CV =
0	NaN	0.1143	0.2570

```
>> qqplot(x), grid on
```

```
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14)
```

```
>> xlabel('\bfКвантілі стандартного нормального розподілу') % мітка осі OX
```

```
>> ylabel('\bfКвантілі вибірки') % мітка осі OY
```

```
>> title('\bfQQ графік');
```



Рис. 10.16. Розподіл вибірових значень

Функція повертає  $P = \text{NaN}$ . Це ознака того, що  $LSTAT$  виходить за межі табличних значень, але  $H = 0$  показує можливість прийняття нульової гіпотези.

# 11. Перевірка статистичних гіпотез (порівняння вибірових характеристик)

В **Statistics Toolbox** цей розділ називається **Hypothesis Tests** (перевірка гіпотез), але така назва занадто загальна, тому що там є лише функції для перевірки гіпотез про збіг середніх або медіан вибірок.

В розділ додатків включено опис функцій Бартлета і Кокрена для порівняння дисперсій.

<b>ZTEST</b>	Перевірка параметричної гіпотези про значення середнього за відомої дисперсії нормальної сукупності .....	37
<b>TTEST</b>	Перевірка параметричної гіпотези про значення середнього за невідомої дисперсії нормальної сукупності .....	42
<b>TTEST2</b>	Перевірка параметричної гіпотези про різницю середніх двох сукупностей .....	48
<b>RANKSUM</b>	Ранговий тест Вілкоксона на рівність медіан двох незалежних сукупностей .....	53
<b>SIGNRANK</b>	Знаковий тест Вілкоксона .....	58
<b>SIGNTEST</b>	Знаковий тест .....	64
<b>BTEST</b>	Критерій Бартлета для порівняння дисперсій .....	69
<b>COCHTES</b>	Критерій Кокрена для порівняння дисперсій .....	72

## **ZTEST** Перевірка параметричної гіпотези про значення середнього за відомої дисперсії нормальної сукупності

Під Z-тестом розуміється перевірка статистичної гіпотези про те, що математичне очікування вибірки з відомою дисперсією  $\sigma_x^2$  дорівнює заданій величині  $m$ . Цей тест заснований на тому, що якщо вихідна генеральна сукупність  $X$  має нормальний розподіл з математичним очікуванням  $m_x$  і дисперсією  $\sigma^2$ , то статистика  $Z = \frac{\sqrt{n} \cdot (\bar{X} - m_x)}{\sigma_x}$ , де  $\bar{X}$

– випадкова величина, реалізацією якої є середнє арифметичне елементів вибірки, а  $n$  – об'єм вибірки, має стандартний нормальний розподіл. Якщо ж замість генерального середнього  $m_x$  віднімати в чисельнику задане число  $m$ , то отримана величина буде Z-статистикою, яка й обчислюється в даному тесті. Рівень значимості  $p$  є ймовірністю того, що випадкова величина прийме значення не менше вибіркового значення статистики  $Z$  для використаної критичної області.

*Синтаксис:*

**$h = ztest(x,m,sigma)$**

**$h = ztest(x,m,sigma,alpha)$**

**$[h,sig,ci,zval] = ztest(x,m,sigma,alpha,tail)$**

*Опис:*

Функція  **$h = ztest(x,m,sigma)$**  призначена для перевірки нульової гіпотези, яка полягає в тому, що вибірка  $x$  отримана з нормально розподіленої сукупності із середнім  $m$  і середнім квадратичним відхиленням  $sigma$ . Перевірка нульової гіпотези виконується на підставі статистики  $Z$ . Вибірка  $x$  задається як вектор. Функція повертає скаляр  $h$ , що є результатом перевірки нульової гіпотези для критичного рівня значимості  $p_{кр}$  рівного 0.05. Нульова гіпотеза ухвалюється, якщо  $h = 0$  за  $p_{кр} = 0.05$ . Якщо  $h = 1$ , то нульова гіпотеза відхиляється для  $p_{кр} = 0.05$ .

Функція  **$h = ztest(x,m,sigma,alpha)$**  в параметрі *alpha* дозволяє задати значення критичного рівня значимості для перевірки нульової гіпотези. За замовчуванням *alpha* = 0.05. Умовою прийняття нульової гіпотези є  $p \geq p_{кр}$ , де  $p_{кр}$  – критичний рівень значимості,  $p$  – рівень значимості, відповідний до вибіркової статистики  $Z$ . Вибір величини  $p_{кр}$  надано досліднику. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

Функція  **$[h,sig,ci,zval] = ztest(x,m,sigma,alpha,tail)$**  у додатковому аргументі *tail* дозволяє задати вид альтернативної гіпотези. Можливі три альтернативні гіпотези (табл. 11.1):

**Параметр *tail* і альтернативні гіпотези**

Значення <i>tail</i>	Альтернативна гіпотеза
'both'	Середнє генеральної сукупності не дорівнює $m$ : $M(x) \neq m$ . Перевірка гіпотези виконується для двосторонньої критичної області. Значення за замовчуванням
'right'	Середнє генеральної сукупності більше $m$ : $M(x) > m$ . Перевірка гіпотези виконується для правобічної критичної області
'left'	Середнє генеральної сукупності менше $m$ : $M(x) < m$ . Перевірка гіпотези виконується для лівосторонньої критичної області

Результативними параметрами функції є:

результат перевірки нульової гіпотези  $h$  для заданого критичного рівня значимості;

рівень значимості  $sig$ , відповідний до вибіркового значення статистики  $Z$ ;

вектор границь довірчого інтервалу вибіркового середнього арифметичного  $ci$  для довірчої ймовірності  $P = 1 - \alpha$ ;

значення вибіркової статистики  $Z$ ,  $zval$ .

*Приклади:*

1. Перевірка рівності нулю середнього генеральної сукупності, розподіленої за нормальним законом з математичним очікуванням і дисперсією рівними 0 і 1, за вибіркою  $x$ . Об'єм вибірки дорівнює 10 елементам. Функція повертає результат перевірки нульової гіпотези  $h$ .

```
>> x=normrnd(0,1,10,1);           | h=
>> h=ztest(x,0,1)                 | 0
```

Нульова гіпотеза приймається

2. Перевірка рівності  $m = 0.5$  для середнього сукупності, розподіленої за нормальним законом з математичним очікуванням і дисперсією 0.5 і 2. Об'єм вибірки  $x$  дорівнює 25 елементам. Функція повертає результат перевірки нульової гіпотези  $h$ . Графічне представлення розподілу значень вибірки виконується за допомогою функції **histfit** (рис. 11.1).

```
>> x=normrnd(0.5,2,25,1);
>> h=ztest(x,0.5,2)
h=
    0
>> histfit(x,7), grid on
```

```
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfn') % мітка осі OY
```

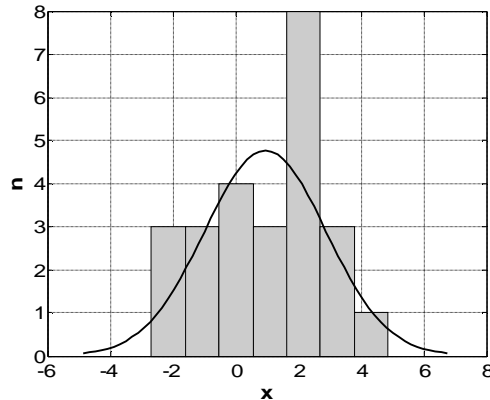


Рис. 11.1. Гістограма і графік щільності ймовірностей вибірки

Нульова гіпотеза приймається

3. Перевірка рівності нулю середнього генеральної сукупності, розподіленої за нормальним законом з математичним очікуванням і дисперсією рівними 0 і 1, за вибіркою  $x$ . Перевірка нульової гіпотези виконується для критичного значення рівня значимості 0.01. У якості альтернативної гіпотези ухвалюється припущення, що середнє генеральної сукупності більше нуля. Перевірка гіпотези проводиться для правобічної критичної області. Об'єм вибірки дорівнює 15 елементам. Функція повертає: результат перевірки нульової гіпотези  $h$  для заданого критичного рівня значимості, рівень значимості  $sig$ , відповідний до розрахованого значення статистики  $Z$ , вектор границь довірчого інтервалу на середнє  $ci$ , значення вибіркової статистики  $zval$ . Рівень довіри зв'язаний з рівнем значимості як  $(1 - \alpha)$ .

```
>> x=normrnd(0,1,15,1);
>> alpha=0.01;
>> [h,sig,ci,zval]=ztest(x,0,1,alpha,'right')
```

h = 0	ci = -0.5749	zval = 0.0998
sig = 0.4602	Inf	

4. Залежність величини рівня значимості від об'єму вибірки і математичного очікування нормально розподіленої генеральної сукупності. Перевіряється нульова гіпотеза на рівність нулю середніх генеральних сукупностей. Дисперсія вибірок дорівнює 1 (рис. 11.2).

```
>> delta=0:0.1:1;
>> n=10;
>> for i=1:length(delta)
    x = normrnd(delta(i),1,n,1);
    [h,p1(i),ci,zval] = ztest(x,0,1);
end;
>> n=30;
>> for i=1:length(delta)
    x = normrnd(delta(i),1,n,1);
    [h,p2(i),ci,zval] = ztest(x,0,1);
end;
>> n=50;
>> for i=1:length(delta)
    x = normrnd(delta(i),1,n,1);
    [h,p3(i),ci,zval] = ztest(x,0,1);
end;
>> plot(delta,p1,'vk',delta,p2,'ok',delta,p3,'+k')
>> grid on
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
>> xlabel("\bfx") % мітка осі OX
>> ylabel("\bfn") % мітка осі OY
>> legend('n = 10', 'n = 30', 'n = 50')
```

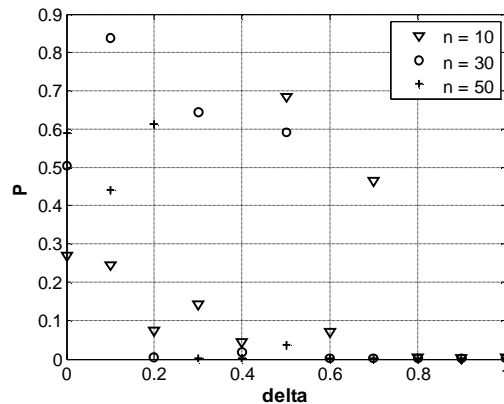


Рис. 11.2. Залежність величини рівня значимості від математичного очікування

5. Вид розподілу статистики  $Z$ . Генеральна сукупність розподілена за нормальним законом з нульовим математичним очікуванням і одиничною дисперсією (рис. 11.3). Об'єм вибірки дорівнює 5. Кількість вибірок дорівнює 25.

```
>> m=0; sigma=1;
>> n=5; N=25;
>> for i=1:N
    x = normrnd(m,sigma,n,1);
    [h,p,ci,zval(i)] = ztest(x,m,sigma);
end;
>> histfit(zval), grid on
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
>> xlabel("\bfx") % мітка осі OX
>> ylabel("\bfn") % мітка осі OY
```

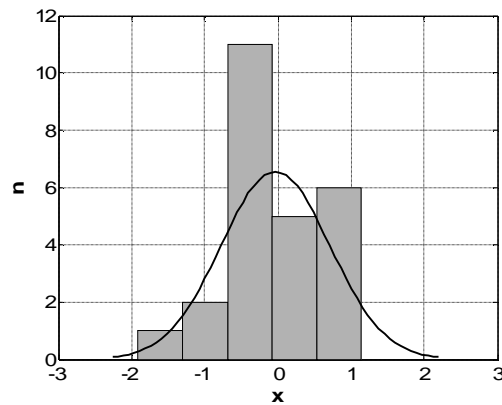


Рис 11.3. Гістограма і графік щільності ймовірностей для вибірки

### **TTEST** Перевірка параметричної гіпотези про значення середнього за невідомої дисперсії нормальної сукупності

Під Т-тестом для однієї вибірки розуміється перевірка статистичної гіпотези про те, що математичне очікування вибірки з невідомою дисперсією дорівнює заданій величині  $m$ . Цей тест заснований на тому, що, якщо вихідна генеральна сукупність  $X$  має нормальний розподіл з математичним очікуванням  $m_x$ , то статистика  $t = \frac{\sqrt{n} \cdot (\bar{X} - m_x)}{s_x}$ , де  $\bar{X}$  – випадкова величина, реалізацією якої є середнє арифметичне елементів вибірки,  $n$  – об'єм вибірки,  $s_x$  – точкова незміщена оцінка середнього квадратичного відхилення, має  $t$ -розподіл Стюдента.

Рівень значимості  $p$  є ймовірністю того, що випадкова величина  $t$  прийме значення не менше вибіркової статистики  $t$  для використаної критичної області. Розподіл статистики  $t$  підкоряється закону Стюдента з  $(n - 1)$  ступенями свободи. Якщо ж замість генерального середнього  $m_x$  віднімати в чисельнику задане число  $m$ , то отримана величина буде  $t$ -статистикою, яка й обчислюється в даному тесті.

*Синтаксис:*

- h = ttest(x)**
- h = ttest(x,m)**
- h = ttest(x,y)**
- h = ttest(x,m,alpha)**

**h = ttest(x,m,alpha,tail)**  
**[h,p,ci] = ttest(...)**  
**[h,p,ci,stats] = ttest(...)**

*Опис:*

Функція **h = ttest(x)** призначена для перевірки нульової гіпотези, що полягає в тому, що вибірка  $x$  отримана з генеральної сукупності з нульовим середнім. Перевірка нульової гіпотези виконується на підставі статистики  $t$ . Вибірка  $x$  задається як вектор. Функція повертає скаляр  $h$ , що є результатом перевірки нульової гіпотези для критичного значення рівня значимості  $p_{кр}$ . Нульова гіпотеза ухвалюється, якщо  $h = 0$  при  $p_{кр} = 0.05$ . Якщо  $h = 1$ , то нульова гіпотеза має бути відхилена при  $p_{кр} = 0.05$ . Передбачається, що вибірка  $x$  отримана з генеральної сукупності, що має нормальний розподіл з невідомою дисперсією.

Функція **h = ttest(x,m)** призначена для перевірки нульової гіпотези, яка полягає в тому, що вибірка  $x$  отримана з нормальної генеральної сукупності з математичним очікуванням рівним  $m$ . Перевірка нульової гіпотези виконується на підставі статистики  $t$ .

Функція **h = ttest(x,y)** дозволяє провести перевірку нульової гіпотези, яка полягає в тому, що вибірки  $x$  і  $y$  отримані з генеральних сукупностей з однаковими середніми значеннями. Передбачається, що різниця вибірових значень  $(x - y)$  буде відповідати генеральній сукупності, що має нормальний розподіл з нульовим математичним очікуванням і невідомою дисперсією. Число елементів у векторах  $x$  і  $y$  має бути однаковим. Перевірка нульової гіпотези виконується на підставі статистики  $t$ .

Функція **h = ttest(...,alpha)** в параметрі *alpha* дозволяє задати значення критичного рівня значимості для перевірки нульової гіпотези. За замовчуванням  $alpha = 0.05$ . Умова прийняття нульової гіпотези  $p \geq p_{кр}$ , де  $p_{кр}$  – критичний рівень значимості,  $p$  – рівень значимості, відповідний до розрахованої статистики  $t$ . Вибір величини  $p_{кр}$  наданий досліднику. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

Функція  $h = \text{ttest}(\dots, \alpha, \text{tail})$  в параметрі *tail* дозволяє задати вид альтернативної гіпотези.

Можливі три альтернативні гіпотези (табл. 11.2):

Таблиця 11.2

**Значення параметру *tail* і альтернативні гіпотези**

Значення <i>tail</i>	Альтернативна гіпотеза
'both'	Середнє генеральної сукупності не дорівнює нулю або значенню $m$ . Перевірка гіпотези провадиться для двосторонньої критичної області. Значення за замовчуванням
'right'	Середнє генеральної сукупності більше нуля, або величини $m$ . Перевірка гіпотези провадиться для правобічної критичної області
'left'	Середнє генеральної сукупності менше нуля, або величини $m$ . Перевірка гіпотези провадиться для лівосторонньої критичної області

Функція  $[h, p, ci, stats] = \text{ttest}(\dots)$  повертає:

результат перевірки нульової гіпотези  $h$  для заданого критичного рівня значимості;

рівень значимості  $p$ , відповідний до розрахованого значення статистики  $t$ ;

вектор границь довірчого інтервалу на середнє  $ci$  з довірчою ймовірністю  $(1 - \alpha)$ ;

структуру  $stats$ , що має наступні поля:

'tstat' – значення статистики  $t$ , розраховане по вибірці;

'df' – число ступенів свободи;

'sd' – незміщена точкова оцінка середнього квадратичного відхилення.

Якщо виконується парний тест для двох вибірок, то точкова оцінка середнього квадратичного відхилення розраховується для вибірки різниць  $(x - y)$ .

*Приклади:*

1. Перевірка рівності нулю середнього арифметичного генеральної сукупності, розподіленої за нормальним законом з математичним

очікуванням і дисперсією рівними 0 і 1. Об'єм вибірки  $x$  дорівнює 10 елементам. Функція повертає результат перевірки нульової гіпотези  $h$ .

```
>> x=normrnd(0,1,10,1);  
>> h=ttest(x)  
h=  
    0
```

2. Перевірка рівності одиниці середнього генеральної сукупності, розподіленої за нормальним законом з математичним очікуванням і дисперсією рівними 0.5 і 2. Об'єм вибірки  $x$  дорівнює 25 елементам. Функція повертає результат перевірки нульової гіпотези  $h$ . Графічне представлення розподілу значень вибірки виконується за допомогою функції `histfit` (рис. 11.4).

```
>> x=normrnd(0.5,2,25,1);  
>> h=ttest(x)  
h=  
    0  
>> histfit(x,7), grid on
```

```
>> set(get(gcf,'CurrentAxes'),...  
      'FontName','Arial Cyr','FontSize',14)  
>> xlabel('\bfx') % мітка осі OX  
>> ylabel('\bfn') % мітка осі OY
```

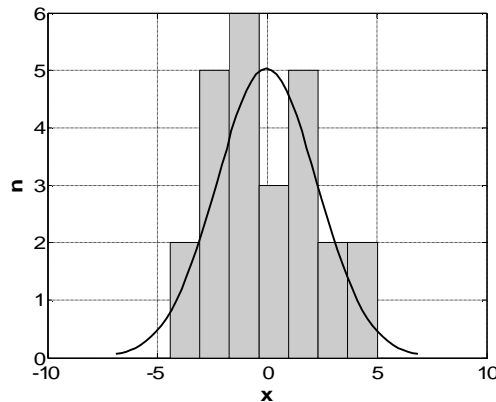


Рис. 11.4. Гістограма і графік функції розподілу вибірки

Оскільки  $h = 0$ , нульова гіпотеза приймається.

3. Перевірка нульової гіпотези, яка полягає в тому, що вибірки  $x$  і  $y$  отримані з генеральних сукупностей з однаковими середніми значеннями рівними нулю. Вибірka  $x$  розподілена за нормальним законом, а  $y$  – за рівномірним законом. Вибірki  $x$ ,  $y$  мають різні дисперсії. Функція повертає результат перевірки нульової гіпотези  $h$ . Графічне

представлення розподілів значень вибірок  $x$ ,  $y$ ,  $(x - y)$  виконується за допомогою функції **ksdensity** (рис. 11.5).

```
>> x=normrnd(0,1,30,1);
>> y=unifrnd(-1,1,30,1);
>> h=ttest(x,y)
h=
    0
>> [fx,xx]=ksdensity(x);
>> [fy,xy]=ksdensity(y);
>> [f_xy,x_xy]=ksdensity(x-y);
>> plot(xx,fx,'-',xy,fy,'--',x_xy,f_xy,':')
>> grid on
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
>> xlabel('\bfx, y, x-y') % мітка осі ОХ
>> ylabel('\bff') % мітка осі ОУ
```

Нульова гіпотеза приймається.

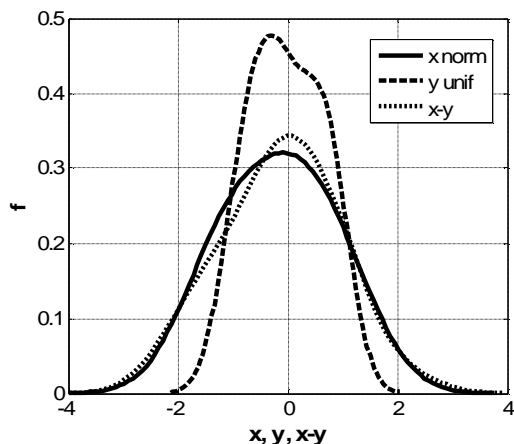


Рис. 11. 5. Графіки розподілів вибірок

4. Перевірка рівності нулю середнього генеральної сукупності, розподіленої за нормальним законом з математичним очікуванням і дисперсією рівними 0 і 1. Перевірка нульової гіпотези виконується для критичного значення рівня значимості рівного 0.01. У якості альтернативної гіпотези ухвалюється припущення, що середнє генеральної сукупності більше нуля. Перевірка гіпотези провадиться для правобічної критичної області. Об'єм вибірки  $x$  дорівнює 15 елементам. Функція повертає: результат перевірки нульової гіпотези  $h$  для заданого критичного рівня, рівень значимості  $p$ , відповідний до розрахованого значення статистики  $t$ , вектор границь довірчого інтервалу на середнє  $ci$  для довірчої ймовірності  $P = 1 - \alpha$ , структуру  $stats$ .

```
>> [h,p,ci,stats]=ttest(x,0,0.01,'right')
h=
    0
p=
    0.7343
ci=
    -0.5424
    Inf
stats=
    tstat: -0.6335
    df: 29
    sd: 0.9597
```

5. Залежність величини рівня значимості від математичного очікування й об'єму вибірок, розподілених за нормальним законом. Перевіряється нульова гіпотеза на рівність нулю середніх генеральних сукупностей. Дисперсія вибірок дорівнює 1 (рис. 11.6).

```
>> delta=0:0.1:1;
>> n=10;
>> for i=1:length(delta)
    x=normrnd(delta(i),1,n,1);
    [h,p1(i),ci,stats]=ttest(x,0);
end;
>> n=30;
>> for i=1:length(delta)
    x=normrnd(delta(i),1,n,1);
    [h,p2(i),ci,stats]=ttest(x,0);
End;
>> n=50;
>> for i=1:length(delta)
    x=normrnd(delta(i),1,n,1);
    [h,p3(i),ci,stats]=ttest(x,0);
end;
>> plot(delta,p1,'vk',delta,p2,'ok',delta,p3,'+k')
>> grid on
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
>> xlabel('\bdelta') % мітка осі OX
>> ylabel('\bfP') % мітка осі OY
>> legend('n = 10', 'n = 30', 'n = 50')
```

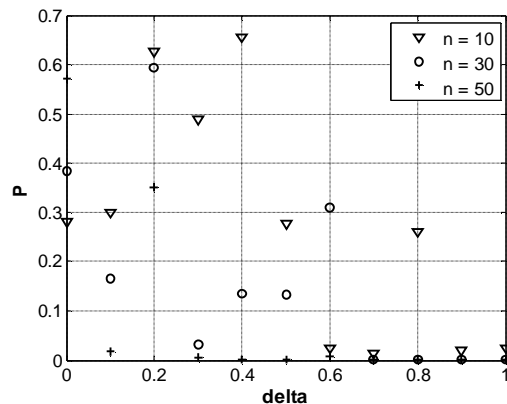


Рис. 11. 6. Залежність величини рівня значимості від математичного очікування й об'єму вибірок

6. Перевірка 0-гіпотези для нормально розподіленої випадкової величини за рівнем значимості 0.01. Об'єм вибірки – 100 елементів.

```
>> m=5; % середнє, що перевіряється
>> n=100; % об'єм вибірки
```

```

>> alpha=0.01; % рівень значимості
>> x=normrnd(6,4,n,1); % вибірка M(x) = 6
>> [h,p,ci]=ttest(x,m,alpha,'right'); % t-тест
>> fprintf('Критичне значення p=%10.8f;\n',p)
>> if h
    disp('ухвалюємо альтернативну гіпотезу: M(X)>m.')
else
    disp('ухвалюємо 0-гіпотезу: M(X)=m.')
end
>> fprintf('Довірчий інтервал: %10.8f<=M(X)<=%10.8f\n',ci)
Критичне значення p=0.24533657;
ухвалюємо 0-гіпотезу: M(X)=m.
Довірчий інтервал: 4.35799489<=M(X)<=      Inf

```

## **TTEST2** Перевірка параметричної гіпотези про різницю значень середніх двох сукупностей

Під t-тестом для двох виборок розуміється перевірка статистичної гіпотези про рівність генеральних середніх двох нормально розподілених сукупностей з невідомими, але однаковими генеральними дисперсіями.

Якщо ця гіпотеза має місце, то статистика  $t = \frac{\bar{X} - \bar{Y}}{s_0 \sqrt{\frac{1}{n} + \frac{1}{m}}}$  має t-розподіл

Стьюдента з  $(n + m - 2)$  ступенями свободи. Тут  $\bar{X}, \bar{Y}$  – випадкові величини, реалізаціями яких є вибіркові середні,  $n, m$  – об'єми виборок  $x$  і  $y$  ( $n = \text{length}(x)$ ,  $m = \text{length}(y)$ ),  $s_0$  – точкова оцінка об'єднаного середньоквадратичного відхилення. Рівень значимості  $p$  є ймовірністю того, що випадкова величина  $t$  прийме значення не більше вибіркового значення статистики  $t$  для використовуваної критичної області.

*Синтаксис:*

```

h = ttest2(x,y)
[h,significance,ci]= ttest2(x,y,alpha)
[h,significance,ci,stats]= ttest2(x,y,alpha)
[...] =ttest2(x,y,alpha,tail)
h = ttest2(x,y,alpha,tail,'unequal')

```

Опис:

Функція **h = ttest2(x,y)** призначена для перевірки нульової гіпотези, яка полягає в тому, що вибірки  $x$ ,  $y$  отримані з нормально розподілених генеральних сукупностей з однаковими середніми значеннями. Передбачається, що середньоквадратичні відхилення генеральних сукупностей невідомі, але дорівнюють між собою. Вектори  $x$  і  $y$  можуть мати різну кількість елементів. Параметр  $h$  є результатом перевірки нульової гіпотези для критичного значення рівня значимості  $p_{кр} = 0.05$ . Нульова гіпотеза ухвалюється якщо  $h = 0$  при  $p_{кр} = 0.05$ . Якщо  $h = 1$ , то нульова гіпотеза має бути відхилена для  $p_{кр} = 0.05$ . Перевірка нульової гіпотези виконується на підставі статистики  $t$ .

Умовою прийняття нульової гіпотези є  $p \geq p_{кр}$ , де  $p_{кр}$  – критичний рівень значимості,  $p$  – рівень значимості, відповідний до розрахованої статистики  $t$ . Вибір величини  $p_{кр}$  наданий досліднику. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

Функція **[h,significance,ci]= ttest2(x,y,alpha)** повертає:

результат перевірки нульової гіпотези  $h$  для заданого критичного рівня значимості;

рівень значимості *significance*, відповідний до вибіркового значення статистики  $t$ ;

вектор границь довірчого інтервалу на різниці середніх  $ci$  з довірчою ймовірністю  $(1 - alpha)$ . Додатковий аргумент *alpha* дозволяє задати величину критичного рівня значимості для перевірки нульової гіпотези. Значення за замовчуванням  $alpha = 0.05$ .

У функції **[h,significance,ci,stats] = ttest2(x,y,alpha)** результативний параметр *stats* є структурою з наступними полями:

*tstat* – значення статистики  $t$ , розрахованої за вибіркою;

*df* – число ступенів свободи;

$sd$  – точкова оцінка об'єднаного середнього квадратичного відхилення, якщо дисперсії вибірок рівні, а якщо ні, то  $sd$  являє собою вектор вибірових середніх квадратичних відхилень.

Функція [...] = **ttest2(x,y,alpha,tail)** в параметрі *tail* дозволяє задати вид альтернативної гіпотези, три види яких описані в табл. 11.3.

У функції **h = ttest2(x,y,alpha,tail,'unequal')** вхідний параметр *'unequal'* призначений для перевірки нульової гіпотези, якщо вибірки  $x$ ,  $y$  мають нормальний розподіл з різними дисперсіями. Ця задача відома як проблема Беренса-Фішера. Для її рішення в **ttest2** використовується апроксимація Сатервейта за визначення ефективного числа ступенів свободи.

Таблиця 11.3

**Параметр *tail* і альтернативні гіпотези**

<b>Значення <i>tail</i></b>	<b>Альтернативна гіпотеза</b>
<i>'both'</i>	Середні генеральних сукупностей нерівні $\mu_x \neq \mu_y$ . Перевірка гіпотези провадиться для двосторонньої критичної області. Значення за замовчуванням
<i>'right'</i>	Середнє генеральної сукупності $\mu_x$ більше $\mu_y$ : $\mu_x > \mu_y$ . Перевірка гіпотези провадиться для правобічної критичної області
<i>'left'</i>	Середнє генеральної сукупності $\mu_x$ менше $\mu_y$ : $\mu_x < \mu_y$ . Перевірка гіпотези провадиться для правобічної критичної області

*Приклади:*

1. Перевірка нульової гіпотези, яка полягає в тому, що вибірки  $x$ ,  $y$  отримані з нормально розподілених генеральних сукупностей з однаковими середніми значеннями, для критичного рівня значимості 0.05. Дисперсії вибірок дорівнюють одиниці. Вибірki  $x$ ,  $y$  мають однаковий об'єм. Графічне представлення вибірок  $x$ ,  $y$  виконується за допомогою функції непараметричного згладжування **ksdensity** (рис. 11.7). Функція **ttest2** повертає результат перевірки нульової гіпотези  $h$ .

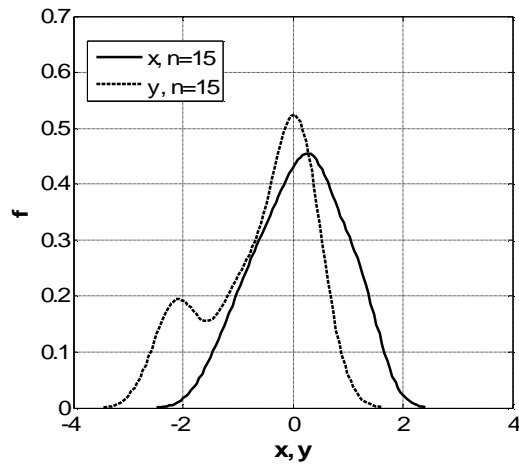


Рис. 11.7. Функції щільності ймовірностей для двох вибірок

Нульова гіпотеза приймається.

Оператори:

```
>> x=normrnd(0,1,15,1);
>> y=normrnd(0,1,15,1);
>> h=ttest2(x,y)
h=
    0
>> [fx,xx]=ksdensity(x);
>> [fy,xy]=ksdensity(y);
>> plot(xx,fx,'-',xy,fy,'--'), grid on
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
>> xlabel('\bfx, y') % мітка осі OX
>> ylabel('\bff') % мітка осі OY
```

2. Перевірка нульової гіпотези про те, що вибірки  $x$ ,  $y$  отримані з нормально розподілених генеральних сукупностей з однаковими середніми значеннями, для критичного рівня значимості 0.05. Дисперсії вибірок дорівнюють одиниці. Вибірki  $x$ ,  $y$  мають однаковий об'єм. Функція повертає: результат перевірки нульової гіпотези  $h$ , рівень значимості *significance*, відповідний до розрахованого значення статистики  $t$ , вектор границь 95%-го довірчого інтервалу на різниці середніх  $ci$ .

```
>> x=normrnd(0,1,15,1);
>> y=normrnd(0,1,15,1);
>> [h,significance,ci]=ttest2(x,y)
h=
    0
significance=
    0.7013
ci=
   -1.1165
    0.7612
```

3. Перевірка нульової гіпотези про те, що вибірки  $x$ ,  $y$  отримані з нормально розподілених генеральних сукупностей з однаковими середніми значеннями, для критичного рівня значимості  $\alpha = 0.01$ . Альтернативна

гіпотеза ухвалюється, якщо середнє сукупності  $x$  більше середнього  $y$ :  $\mu_x > \mu_y$ . Перевірка гіпотези проводиться для правобічної критичної області. Дисперсії вибірок дорівнюють одиниці. Вибірki  $x$ ,  $y$  мають різний об'єм. Функція повертає: результат перевірки нульової гіпотези  $h$ , рівень значимості *significance*, відповідний до розрахованого значення статистики  $t$ , вектор границь 99%-го довірчого інтервалу різниці середніх  $ci$ ; структуру *stats*.

```
>> x = normrnd(0,1,15,1);
>> y = normrnd(0,1,20,1);
>> alpha= 0.01;
>> [h,significance,ci,stats]=ttest2(x,y,alpha,'right')
h=
    0
```

significance =	stats=
0.7674	tstat: -0.7390
ci=	df: 33
-1.0088	sd: 0.9277
Inf	

4. Перевірка нульової гіпотези про те, що вибірки об'ємом 100 і 200, отримані з нормально розподілених генеральних сукупностей з різними середніми значеннями, для критичного рівня значимості  $\alpha = 0.1$ . Альтернативна гіпотеза полягає у тому, що в генеральних сукупностях середнє  $x$  більше середнього  $y$ .

```
>> alpha=0.1; % рівень значимості
>> x=normrnd(9,4,100,1); % вибірка x
>> y=normrnd(6,12,200,1); % вибірка y
>> Dx=var(x); % дисперсії вибірок
>> Dy=var(y);
>> fx=length(x)-1; % числа ступенів свободи
>> fy=length(y)-1;
>> Fbord=finv([alpha/2 1-alpha/2],fx,fy); % довірчий інтервал
>> if prod(Fbord-Dx/Dy)<=0,
    vt='equal'; % дисперсії однакові
else
    vt='unequal'; % дисперсії різні
end
>> [h,p,ci]=ttest2(x,y,alpha,'both',vt); % t-тест
>> fprintf('Критичне значення p=%10.8f;\n',p);
>> if h,
    disp('ухвалюємо альтернативну гіпотезу: M(X)<>M(Y).');
else
    disp('ухвалюємо 0-гіпотезу: M(X)=M(Y).');
end
>> fprintf('Довірчий інтервал: %10.8f<=M(X)-M(Y)<=% 10.8f',ci);
Критичне значення p=0.00035227
ухвалюємо альтернативну гіпотезу: M(X)<>M(Y).
Довірчий інтервал: 1.72594059<=M(X)-M(Y)<= 4.61845802
```

## **RANKSUM** Ранговий тест Вілкоксона на рівність медіан двох незалежних сукупностей

Порівнюються медіани двох вибірок.

*Синтаксис:*

```
p = ranksum(x,y)  
[p,h] = ranksum(x,y)  
[p,h] = ranksum(x,y,alpha)  
[p,h,stats] = ranksum(...)
```

*Опис:*

Функція **p = ranksum(x,y)** призначена для проведення двостороннього рангового тесту Вілкоксона для перевірки нульової гіпотези, яка полягає в тому, що незалежні вибірки  $x$  і  $y$  узяті з генеральних сукупностей з рівними медіанами. Вибірki  $x$  і  $y$  задаються як вектори. Функція повертає значення рівня значимості  $p$  для перевірки нульової гіпотези. Рівень значимості – це імовірність помилки першого роду, тобто ймовірність невірно відкинути нульову гіпотезу у випадку її справедливості. Якщо одержано значення  $p \approx 0$ , нульова гіпотеза має бути відкинута, тобто медіани порівнюваних вибірок статистично значимо відрізняються між собою. Вибір критичного рівня значимості  $p_{кр}$  для умови прийняття нульової гіпотези  $p \geq p_{кр}$  наданий досліднику. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

Передбачається, що генеральні сукупності, з яких витягають вибірки, мають однаковий безперервний розподіл, але вони можуть відрізнятися середніми значеннями. Об'єми вибірок, і як наслідок, число елементів векторів  $x$ ,  $y$ , можуть бути різними.

Ранговий тест Вілкоксона є еквівалентом U-тесту Манна-Уїтні.

Функція **[p,h] = ranksum(x,y)** повертає значення рівня значимості  $p$  в ході перевірки нульової гіпотези й результат перевірки нульової гіпотези  $h$  для критичного значення  $p_{кр}$  рівного 0.05. Нульова гіпотеза ухвалюється якщо  $h = 0$  за  $p_{кр} = 0.05$ . Якщо  $h = 1$ , то нульова гіпотеза

має бути відкинута за  $p_{кр} = 0.05$ , тобто медіани вибірок статистично значимо відрізняються між собою.

Функція **[p,h] = ranksum(x,y,alpha)** в параметрі *alpha* дозволяє задати значення критичного рівня значимості в разі перевірки нульової гіпотези. За замовчуванням *alpha* = 0.05.

Функція **[p,h,stats] = ranksum(...)** повертає структуру *stats* з одним або двома полями:

*stats.ranksum* – значення рангової статистики;

*stats.zval* – значення Z-статистики. Ця статистика розраховується за умови великого об'єму вибірки. Значення рівня значимості *p* буде розраховано, виходячи з нормальної апроксимації розподілу статистики *Z*.

*Приклади:*

1. Перевірка рівності медіан двох незалежних вибірок розподілених за безперервним рівномірним законом з однаковою величиною розмаху й величинами математичних очікувань рівних 0.5 і 0.75 для вибірок *x* і *y* відповідно. Об'єми вибірок дорівнюють 10 і 15 елементам. Функція повертає значення рівня значимості *p*.

```
>> x=unifrnd(0,1,10,1);  
>> y=unifrnd(.25,1.25,15,1);  
>> p=ranksum(x,y)  
p=  
0.1416
```

Нема підстави відкидати 0-гіпотезу про те, що незалежні вибірки *x* і *y* узяті з генеральних сукупностей з рівними медіанами.

2. Перевірка гіпотези про рівність медіан двох незалежних вибірок розподілених за нормальним законом з однаковими величинами середніх квадратичних відхилень і величинами математичних очікувань рівними 0 і 0.1 для вибірок *x* і *y* відповідно. Об'єми вибірок дорівнюють 10 елементам. Функція повертає значення рівня значимості *p* і результат перевірки нульової гіпотези для критичного рівня значимості рівного 0.02. Крім значень *p* і *h* функція повертає структуру *stats*. Графічно

результати теста можна представити за допомогою функції **boxplot** (рис. 11.8).

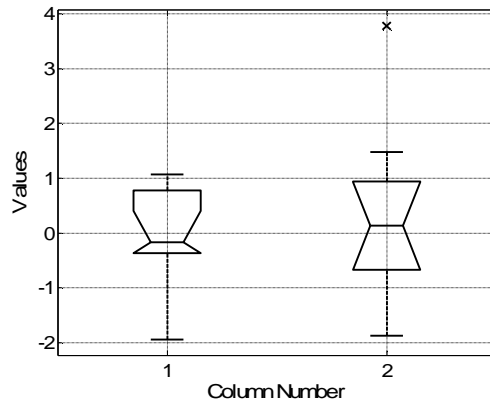


Рис.1 1.8."Коробки" Тьюки для двох вибірок

Приймається 0-гіпотеза.

Оператори:

```
>> x=normrnd(0,1,10,1);
>> y=normrnd(0.1,1,10,1);
>> [p h stats]=ranksum(x,y,0.01)
p=
    0.6232
h =
     0
```

```
0
stats =
    zval: -0.4914
    ranksum: 98
>> boxplot([x y],1), grid
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
```

3. Залежність величини рівня значимості від різниці між середніми арифметичними вибірок, розподілених за нормальним законом. Об'єми вибірок прийняті рівними 10 елементам (рис. 11.9).

```
>> delta=0:0.1:1;
>> x=normrnd(0,1,10,1);
>> for i=1:length(delta)
    y=normrnd(delta(i),1,10,1);
    p(i)=ranksum(x,y);
end;
```

```
>> plot(delta,p,'ok'), grid on
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
>> xlabel('\bfdelta') % мітка осі OX
>> ylabel('\bfP') % мітка осі OY
```

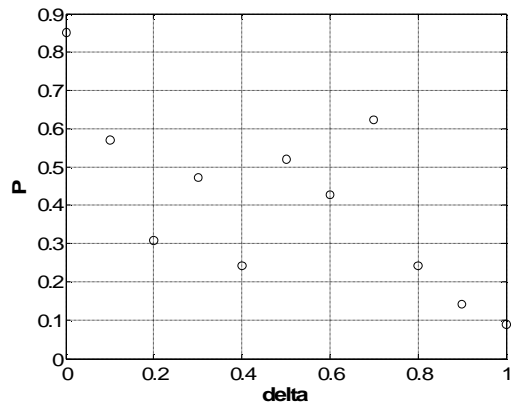


Рис. 11.9. Залежність величини рівня значимості

4. Залежність величини рівня значимості від різниці між математичними очікуваннями й об'ємом вибірок, розподілених за нормальним законом. Об'єми вибірок прийняті рівними 10, 30 і 50 елементам.

```
>> delta=0:0.1:1;
>> n=10;
>> x=normrnd(0,1,n,1);
>> for i=1:length(delta)
    y=normrnd(delta(i),1,n,1);
    p1(i)=ranksum(x,y);
end;
>> n=30;
>> x=normrnd(0,1,n,1);
>> for i=1:length(delta)
    y=normrnd(delta(i),1,n,1);
    p2(i)=ranksum(x,y);
end;
>> n=50;
>> x=normrnd(0,1,n,1);
>> for i=1:length(delta)
    y=normrnd(delta(i),1,n,1);
    p3(i)=ranksum(x,y);
end;

>> plot(delta,p1,'vk',delta,p2,'ok',delta,p3,'+k')
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14)
>> grid on
>> legend('n = 10', 'n = 30', 'n = 50')
>> xlabel('\bdelta') % мітка осі OX
>> ylabel('\bfP') % мітка осі OY
```

На рис. 11.10 зображена залежність рівня значимості для кожної з 11 вибірок з різними об'ємами.

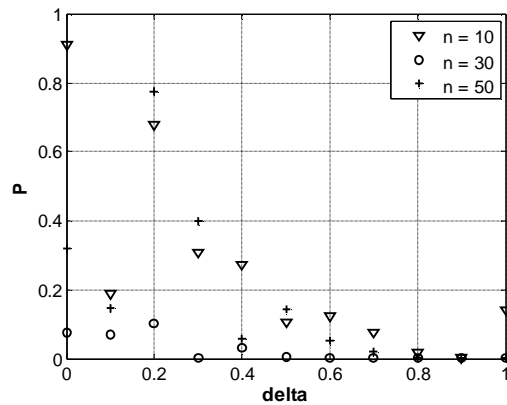


Рис.11.10. Залежність величини рівня значимості для вибірок різних об'ємів

5. У прикладі генеруються дві релеївські вибірки з параметрами 5 і 4.5, які далі порівнюються за допомогою рангового тесту Вілкоксона на рівні значимості 0.01.

```
>> x=raylrnd(5,100,1); % 1-а вибірка
>> y=raylrnd(4.5,200,1); % 2-а вибірка
>> [p,h]=ranksum(x,y,0.01); % ранговий тест Вілкоксона
>> fprintf('Критичне значення p=%10.8f;\n',p);
>> if h,
    disp('справедлива альтернативна гіпотеза.');
```

```
else
    disp('справедлива 0-гіпотеза.');
```

```
end
>> cdfplot(x); % графік F(x)
>> hold on;
>> cdfplot(y); % графік F(y)
>> hold off;
>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14)
>> title(['\bfВибіркові функції розподілу']) % заголовок
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfF(x)') % мітка осі OY
Критичне значення p=0.06468601;
справедлива 0-гіпотеза.
```

На рис. 11.11 показані вибіркові функції розподілу двох релеївських вибірок.

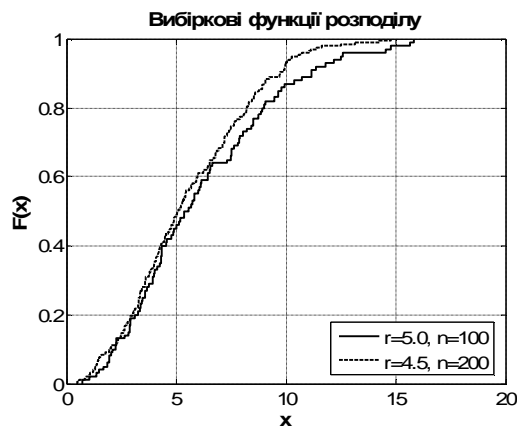


Рис. 11.11. Функції розподілу двох вибірок

### **SIGNRANK** Знаковий тест Вілкоксона

Дана функція за допомогою знакового тесту Вілкоксона (Wilcoxon) перевіряє статистичну гіпотезу про рівність нулю (або будь-якому заданому значенню) медіани вибірки або медіани різниці двох вибірок.

*Синтаксис:*

```

p = signrank(x)
p = signrank(x,m)
p = signrank(x,y)
[p,h] = signrank(...)
[p,h] = signrank(...,alpha)
[p,h,stats] = signrank(...)
[p,h] = signrank(...,'method',method)

```

*Опис:*

Функція **p = signrank(x)** провадить двосторонній знаковий тест для перевірки статистичної гіпотези про те, що вибірка  $x$  узята з генеральної сукупності з нульовою медіаною й нульовим середнім, якщо воно існує (0-гіпотеза). У результативному параметрі  $p$  функція повертає  $p$ -значення для перевірки даної гіпотези. Мале значення  $p$  (менше рівня значимості) свідчить про незастосовність 0-гіпотези, тобто медіана вибірки  $x$  статистичне значимо відрізняється від нуля. Вважається, що вибірка взята з генеральної сукупності з безперервним розподілом, симетричним щодо його медіани.

Функція  **$p = \text{signrank}(x,m)$**  провадить двосторонній знаковий тест для перевірки статистичної гіпотези про те, що вибірка у векторі  $x$  узята з генеральної сукупності з медіаною  $m$ . Аргумент  $m$  повинен бути скаляром.

Функція  **$p = \text{signrank}(x,y)$**  провадить попарний двосторонній знаковий тест для перевірки статистичної гіпотези про те, що різниця між вибірками  $x$  і  $y$  узята з генеральної сукупності з нульовою медіаною. Вважається, що різниця між вибірками  $x$  і  $y$  взята з генеральної сукупності з безперервним розподілом, симетричним щодо його медіани. Вектори  $x$  і  $y$  повинні мати однакову довжину.

*Зауваження.* Оскільки на відміну від математичних очікувань, медіана різниці в загальному випадку не дорівнює різниці медіан, то гіпотеза про рівність нулю медіани  $(x - y)$  не еквівалентна гіпотезі про рівність медіан  $x$  і  $y$ .

Функція  **$[p,h] = \text{signrank}(\dots)$**  в результативному параметрі  $h$  повертає результат перевірки 0-гіпотези на рівні значимості 5%. Якщо  $h = 0$ , то 0-гіпотезу можна прийняти, а за  $h = 1$  її слід відкинути на цьому рівні значимості.

Функція  **$[p,h] = \text{signrank}(\dots, 'alpha', alpha)$**  в аргументі *alpha* дозволяє задати рівень значимості для перевірки 0-гіпотези (за замовчуванням 0.05).

Функція  **$[p,h] = \text{signrank}(\dots, 'method', method)$**  в аргументі *method* дозволяє задати метод обчислення  $p$ -значень, використовуваний в алгоритмі. Можливі значення: *'exact'* для точного обчислення або *'approximate'* для нормальної апроксимації. За замовчуванням використовується *'exact'* для малих вибірок, або *'approximate'* для великих.

Функція  **$[p,h,stats] = \text{signrank}(\dots)$**  у результативному параметрі *stats* повертає структуру з одним або двома полями. Поле *signedrank* містить знакову статистику. Якщо вибірка велика, то  $p$  обчислюється з викори-

станням нормальної апроксимації і в цьому випадку поле *zval* містить значення нормальної Z-статистики.

*Приклади:*

1. Перевірка рівності заданому значенню 0.5 медіани вибірки, розподіленої за нормальним законом з величиною математичного очікування рівної 0.1. Об'єм вибірки дорівнює 15 елементам. Функція повертає значення рівня значимості й результат перевірки нульової гіпотези. Під час перевірки нульової гіпотези критичне значення рівня значимості ухвалюється рівним 0.1. Графічне представлення тесту виконується за допомогою функції **boxplot** (рис. 11.12).

```
>> x=normrnd(0.1,1,15,1);
>> alpha=0.1;
>> [p h]=signrank(x,0.5, alpha)
p=
    0.0215
h=
     1
>> boxplot(x,1), grid
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
```

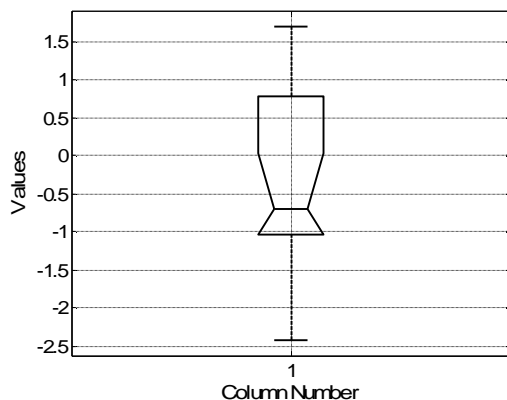


Рис. 11.12. "Коробка" Тьюки.

2. Перевірка рівності заданому значенню 0.7 медіан декількох вибірок, розподілених за нормальним законом з величинами математичних очікувань рівних 0; 0.5; 1; 1.5. Об'єми вибірок дорівнюють 15 елементам. Функція повертає значення рівня значимості, результат перевірки нульової гіпотези й структуру *stats*. Під час перевірки нульової гіпотези критичне значення рівня значимості ухвалюється рівним 0.07. Графічне представлення тесту виконується за допомогою функції **boxplot** (рис. 11.13).

```
>> x1=normrnd(0,1,15,1);
>> x2=normrnd(0.5,1,15,1);
>> x3=normrnd(1,1,15,1);
>> x4=normrnd(1.5,1,15,1);
```

```
>> [p1 h1 stats1]=signrank (x1,0.7, alpha)
p1=
    0.0067
h1 =
     1
```

```
stats1=
    signedrank: 14
>> [p2 h2 stats2]=signrank (x2,0.7, alpha)
p2=
    0.9341
h2=
     0
stats2=
    signedrank: 55
>> [p3 h3 stats3]=signrank (x3,0.7,alpha)
p3=
    0.4887
h3=
     0
```

```
stats3=
    signedrank: 47
>> [p4 h4 stats4]=signrank (x4,0.7, alpha)
p4=
    0.0125
p4=
     1
stats4=
    signedrank: 17
>> boxplot([x1 x2 x3 x4],1)
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
>> grid on
```

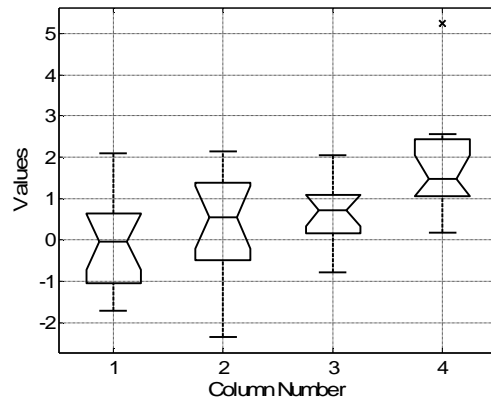


Рис. 11.13. "Коробки" Тьюки для чотирьох вибірок

3. Залежність величини рівня значимості від різниці між медіанами генеральних сукупностей і об'ємами вибірок, розподілених за нормальним законом (рис. 11.14).

```
>> delta=0:0.1:1;
>> n=10;
>> for i=1:length(delta)
    x=normrnd(delta(i),1,n,1);
    p1(i)=signrank(x,0);
end;
```

```
>> n=30;
>> for i=1:length(delta)
    x=normrnd(delta(i),1,n,1);
    p2(i)=signrank (x,0);
end;
```

```
>> n=50;
>> for i=1:length(delta)
    x=normrnd(delta(i),1,n,1);
    p3(i)=signrank (x,0);
end;
```

```
>> plot(delta,p1,'vk',delta,p2,'ok',delta,p3,'+k')
>> grid on
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
```

```
>> xlabel('\bfdelta') % мітка осі OX
>> ylabel('\bfP') % мітка осі OY
>> legend('n = 10', 'n = 30', 'n = 50')
```

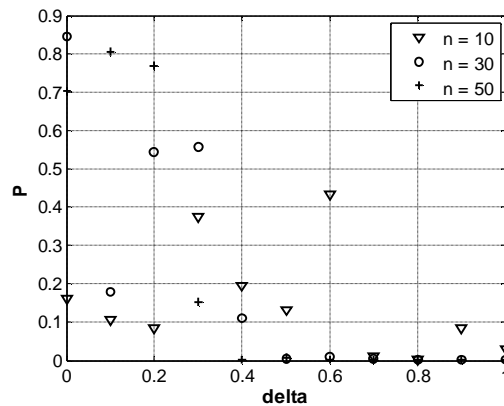


Рис. 11.14. Залежність величини рівня значимості від різниці між медіанами для вибірок різних об'ємів

4. Перевірка рівності нулю медіани вибірки з різницевої генеральної сукупності  $(x - y)$ . За перевірки нульової гіпотези критичне значення рівня значимості ухвалюється рівним 0.01. Функція повертає значення рівня значимості  $p$ , результат перевірки нульової гіпотези  $h$  і структуру  $stats$ . Графічне представлення тесту виконується за допомогою функції `boxplot` (рис. 11.15).

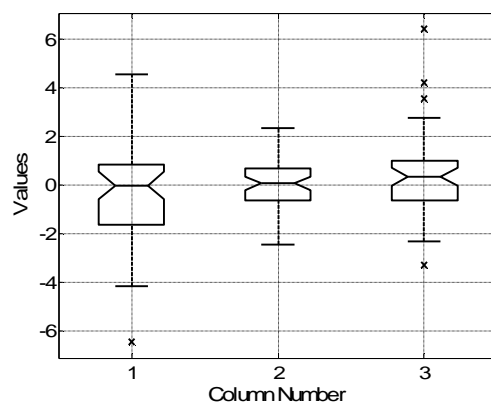


Рис. 11.15. "Коробки" Тьюки

Оператори:

```
>> x=normrnd(0,1,50,1);
>> y=normrnd(0.5,2, 50,1);
>> alpha=0.01;
>> [p h stats]=signrank (x,y, alpha)
p=
    0.3983
```

```
h=
    0
stats=
    zval: -0.8447
    signedrank: 550
```

```
>> boxplot([x-y,x,y],1)
>> grid on
```

```
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
```

5. У прикладі генеруються дві вибірки нормальна й рівномірна з математичним очікуванням (і медіаною) 5, які далі порівнюються за допомогою знакового тесту Вілкоксона на рівні значимості 0.01.

```
>> x=normrnd(5,4,100,1); % 1-а вибірка
>> y=unifrnd(0,10,100,1); % 2-а вибірка
>> [p,h]=signrank(x,y,0.01); % знаковий тест Вілкоксона
>> fprintf('Критичне значення p=%10.8f;\n',p);
>> if h,
    disp('справедлива альтернативна гіпотеза. ');
else
    disp('справедлива 0-гіпотеза. ');
end
>> cdfplot(x); % графік F(x)
>> hold on;
>> cdfplot(y); % графік F(y)
>> hold off;
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title(['\bfВибіркові функції розподілу']) % заголовок
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfF(x)') % мітка осі OY
Критичне значення p=0.68494663;
справедлива 0-гіпотеза.
```

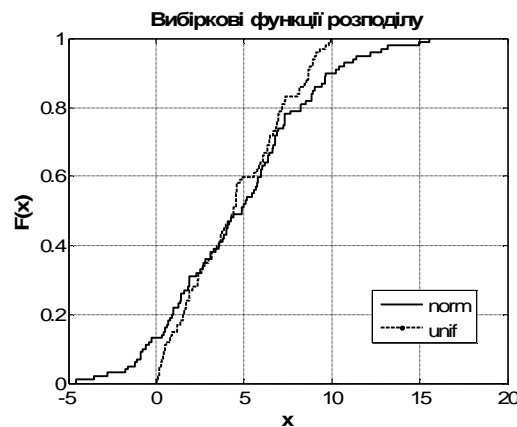


Рис. 11.16. Вибіркові функції розподілу для двох вибірок

На рис. 11.16 показані вибіркові функції розподілу вибірок, розподілених за нормальним і рівномірним законами.

### **SIGNTEST** Знаковий тест

Ця функція за допомогою знакового тесту перевіряє статистичну гіпотезу про рівність нулю (або будь-якому заданому значенню) медіани сукупності або медіани різниці двох сукупностей.

*Синтаксис:*

**p = signtest(x)**  
**p = signtest(x,m)**  
**p = signtest(x,y)**  
**[p,h] = signtest(...)**  
**[p,h] = signtest(...,alpha)**  
**[p,h,stats] = signtest(...)**

*Опис:*

Функція **p = signtest(x)** призначена для проведення двостороннього знакового тесту для перевірки нульової гіпотези, яка полягає в тому, що медіана сукупності дорівнює нулю. Вибірка  $x$  задається як вектор. Функція повертає значення рівня значимості  $p$  для перевірки нульової гіпотези. Рівень значимості – це ймовірність помилки першого роду, тобто ймовірність невірно відкинути нульову гіпотезу у випадку її справедливості. Якщо значення  $p \approx 0$ , то нульова гіпотеза має бути відкинута, тобто медіана вибірки  $x$  статистично значимо відрізняється від нуля. Вибір критичного рівня значимості  $p_{кр}$  для умови прийняття нульової гіпотези  $p \geq p_{кр}$  наданий досліднику. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

Вважається, що вибірка  $x$  отримана з генеральної сукупності, що має безперервний розподіл симетричний щодо медіани.

Функція **p = signtest(x,m)** дозволяє провести двосторонній знаковий тест на рівність медіани вибірки  $x$  значенню  $m$ . Величина  $m$  має бути задана як скаляр.

Функція **p = signtest(x,y)** дозволяє провести парний двосторонній знаковий тест нульової гіпотези, який полягає в тому, що медіана

різницевого розподілу генеральних сукупностей, з яких були узяті вибірки  $x$  і  $y$ , дорівнює нулю. Вважається, що різницевий генеральний розподіл, з якого отримана вибірка  $(x - y)$ , є довільним безперервним законом. Об'єми вибірок  $x$  і  $y$  має бути однаковими, тобто  $length(x) = length(y)$ .

*Примітка:* Гіпотеза про рівність нулю медіани генерального різницевого розподілу не є еквівалентною гіпотезі про рівність медіан генеральних розподілів, з яких були отримані вибірки  $x$  і  $y$ .

Функція **[p,h] = signtest(...)** повертає значення рівня значимості  $p$  і результат перевірки нульової гіпотези  $h$  для значення  $p_{кр} = 0.05$ . Нульова гіпотеза ухвалюється якщо  $h = 0$  при  $p_{кр} = 0.05$ . Якщо  $h = 1$ , то нульова гіпотеза має бути відкинута за  $p_{кр} = 0.05$ .

Функція **[p,h] = signtest(...,alpha)** в параметрі  $alpha$  дозволяє задати значення критичного рівня значимості під час перевірки нульової гіпотези. За замовчуванням  $alpha = 0.05$ .

Функція **[p,h,stats] = signtest(...)** повертає структуру  $stats$  з одним або двома полями:

$stats.sign$  – значення знакової статистики;

$stats.zval$  – значення  $Z$ -статистики.

Статистика  $Z$  розраховується за умови великого об'єму вибірки й її рівень значимості  $p$  буде розраховано, виходячи з нормальної апроксимації розподілу цієї статистики.

*Приклади:*

1. Перевірка рівності нулю медіан сукупностей, розподілених за безперервним законом рівної ймовірності з величинами математичного очікування рівними 0 і 0.1. Об'єми вибірок дорівнюють 10 елементам. Функція повертає значення рівня значимості  $p$ .

```
>> x=unifrnd(-0.9,1.1,10,1);  
>> p=signtest(x,0)  
p=  
0.3438
```

```
>> x=unifrnd(-1,1.1,10,1);  
>> p=signtest(x,0.1)  
p=  
0.1094
```

2. Перевірка рівності заданому значенню 0.22 медіан сукупностей, розподілених за нормальним законом з величинами математичного

очікування рівними 0 і 0.5. Об'єми вибірок дорівнюють 35 елементам. Функція повертає значення рівня значимості  $p$  і результат перевірки нульової гіпотези  $h$ .

```
>> x=normrnd(0,1,35,1);
>> [p h]=signtest (x,0.22)
p=
    1
h=
    0
```

```
>> x=normrnd(0.5,1,35,1);
>> [p h]=signtest (x,0.22)
p=
    0.0410
h=
    1
```

3. Перевірка рівності заданому значенню 0.7 медіан декількох сукупностей, розподілених за нормальним законом з величинами математичних очікувань рівними 0; 0.5; 1; 1.5. Об'єми вибірок дорівнюють 15 елементам. Функція повертає значення рівня значимості, результат перевірки нульової гіпотези й структуру *stats*. За перевірки нульової гіпотези критичне значення рівня значимості ухвалюється рівним 0.07.

```
>> x1=normrnd(0,1,15,1);
>> x2=normrnd(0.5,1,15,1);
>> x3=normrnd(1,1,15,1);
>> x4=normrnd(1.5,1,15,1);
>> alpha=0.07;
>> [p1 h1 stats1]=signtest (x1,0.7, alpha)
p1=
    0.0352
h1=
    1
stats1=
    sign: 3
>> [p2 h2 stats2]=signtest (x2,0.7,alpha)
p2=
    0.6072
h2=
    0
```

```
stats2=
    sign: 6
>> [p3 h3 stats3]=signtest (x3,0.7,alpha)
p3=
    0.6072
h3=
    0
stats3=
    sign: 6
>> [p4 p4 stats4]=signtest (x4,0.7,alpha)
p4=
    0
p4=
    0
stats4=
    sign: 5
```

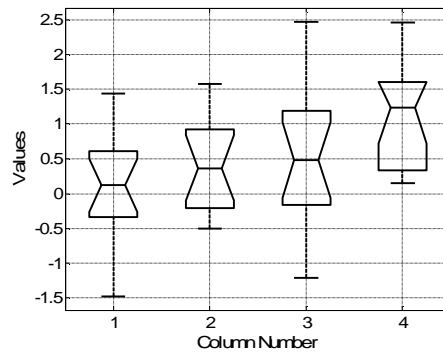


Рис. 11.17. "Коробки" Тьюки

Графічне представлення тесту виконане за допомогою функції **boxplot** (рис. 11.17):

```
>> boxplot([x1 x2 x3 x4],1), grid
>> set(get(gcf,'CurrentAxes'),'Fontname','Arial Cyr','FontSize',14)
```

4. Залежність величини рівня значимості від різниці між математичними очікуваннями генеральних сукупностей і об'ємами вибірок, розподілених за нормальним законом (рис. 11.18).

```
>> delta=0:0.1:1;
>> n=10;
>> for i=1:length(delta)
    x=normrnd(delta(i),1,n,1);
    p1(i)=signtest(x,0);
end;
>> n=30;
>> for i=1:length(delta)
    x=normrnd(delta(i),1,n,1);
    p2(i)=signtest(x,0);
end;
>> n=50;
>> for i=1:length(delta)
    x=normrnd(delta(i),1,n,1);
    p3(i)=signtest(x,0);
end;
>> plot(delta,p1,'vk',delta,p2,'ok',delta,p3,'+k')
>> grid on
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
>> xlabel('\bdelta') % мітка осі OX
>> ylabel('\bP') % мітка осі OY
>> legend('n = 10', 'n = 30', 'n = 50')
```

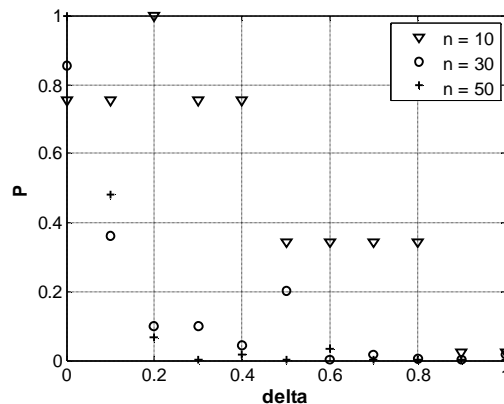


Рис. 11.18. Залежність величини рівня значимості

5. Перевірка рівності нулю медіани з різницевої генеральної сукупності  $(x - y)$ . В ході перевірки нульової гіпотези критичне значення рівня значимості ухвалюється рівним 0.01. Функція повертає значення рівня значимості  $p$ , результат перевірки нульової гіпотези  $h$  і структуру  $stats$ . Графічне представлення тесту виконується за допомогою функції **boxplot** (рис. 11.19).

```
>> x=normrnd(0,1,50,1);
>> y=normrnd(0.5,2, 50,1);
```

```
p=
    0.3222
h=
    0
```

```
>> alpha=0.01;
>> [p h stats]=signtest(x,y, alpha)
```

```
stats=
    sign: 21
>> boxplot([x-y,x,y],1), grid
>> set(get(gcf,'CurrentAxes'),...
'FontName','Arial Cyr','FontSize',14)
```

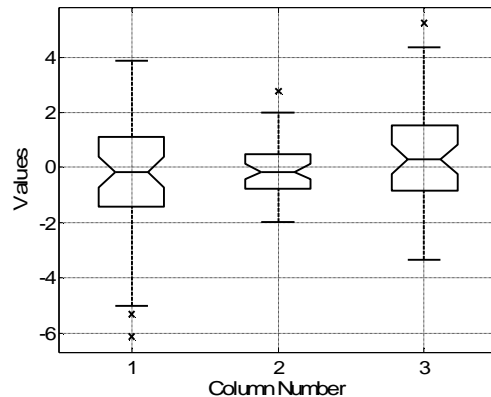


Рис. 11.19. "Коробки" Тьюки

6. У прикладі генеруються дві вибірки нормальна й експоненціальна з математичним очікуванням 5, які далі порівнюються за допомогою знакового тесту на рівні значимості 0.01.

```
>> x=exprnd(5,100,1); % 1-а вибірка
>> y=normrnd(5,4,100,1); % 2-а вибірка
>> [p,h]=signtest(x,y,0.01); % знаковий тест
>> fprintf('Критичне значення p=%10.8f;\n',p);
>> if h,
    disp('справедлива альтернативна гіпотеза. ');
else
    disp('справедлива 0-гіпотеза. ');
end
>> cdfplot(x); % графік F(x)
>> hold on;
>> cdfplot(y); % графік F(y)
```

```

>> hold off;
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title(['\bfВибіркові функції розподілу']) % заголовок
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfF(x)') % мітка осі OY
Критичне значення  $p=0.19360097$ ;
справедлива 0-гіпотеза.

```

На рис. 11.20 показані вибіркові функції розподілу вибірок, розподілених за нормальним й експоненціальним законами.

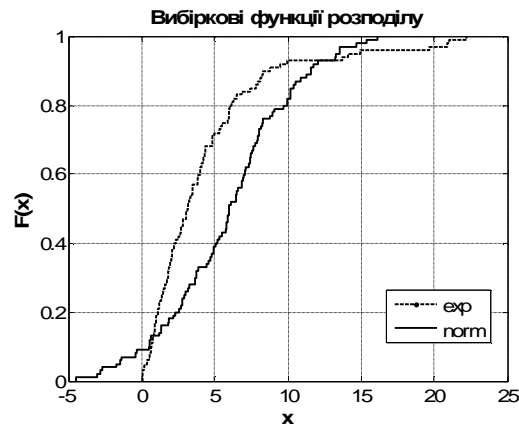


Рис. 11.20. Вибіркові функції розподілу двох вибірок

### **BTEST** Критерій Бартлета для порівняння дисперсій

Наявна в **MatLab** функція **barttest** (що описана в розділі 15: "Функції аналізу багатовимірних випадкових величин") вирішує трохи іншу задачу: вона з'ясовує кількість незалежних стовпців у матриці даних.

Критерій Бартлета у вигляді алгоритму порівняння декількох дисперсій в **MatLab** відсутній, але він є на сайті серед програм для вільного копіювання й використання.

Ця функція провадить тест Бартлета для порівняння дисперсій декількох вибірок, які можуть бути різного об'єму. Автори – А. Трухильо-Ортис (A. Trujillo-Ortiz) і Р. Ернандес-Уоллс (R. Hernandez-Walls).

Алгоритм методу наступний. За вибірковими дисперсіями  $D_1, D_2, \dots, D_k$  обчислюють їх середньозважене значення  $D_x$ , де в якості вагових коефіцієнтів приймають числа ступенів свободи  $df_1, df_2, \dots, df_k$

( $df_j = n_j - 1$ ,  $n_j$  – об'єм  $j$ -тої вибірки). Загальне число ступенів свободи дорівнює їх сумі:

$$df = \sum df_i ; \quad D_x = \frac{1}{df} \sum df_i D_i .$$

Далі обчислюють дві допоміжні величини  $B$ ,  $C$  і їх відношення  $\chi^2$ :

$$B = df \ln D_x - \sum df_i \ln D_i ; \quad C = 1 + \frac{1}{3(k-1)} \left( \sum \frac{1}{df_i} - \frac{1}{df} \right) ; \quad \chi^2 = \frac{B}{C} .$$

Бартлет довів, що коли всі числа ступенів свободи  $df_i \geq 5$ , і всі вибірки мають однакову генеральну дисперсію, то відношення  $B/C$  має приблизно  $\chi^2$ -розподіл Пірсона з  $(k-1)$  ступенями свободи. Отже, обчислене (вбіркове) значення цієї величини може бути критерієм перевірки 0-гіпотези про рівність усіх генеральних дисперсій, тобто, якщо виконується нерівність  $\chi^2 \leq \chi_{\alpha}^2(k-1)$ , 0-гіпотезу можна прийняти. Якщо ж ця нерівність порушується, приймається альтернативна гіпотеза про те, що генеральні дисперсії різні.

*Синтаксис:*

**Btest(X,alpha)**

*Опис:*

Функція **Btest(X,alpha)** провадить тест Бартлета для вибірок, заданих у матриці  $X$ , на рівні значимості  $alpha$  (за замовчуванням 0.05). Параметрів, що повертаються, немає; результати тесту виводяться на екран. Це трохи незручно, але в розпорядженні користувача є вихідний текст функції (файл **Btest.m**), і можна дописати код декількома командами, що не представляє утруднень. Вибірki можуть бути різного об'єму, тому аргумент  $X$  є матрицею із двома стовпцями, де 1-й стовпець – це самі дані, а 2-й – номер вибірки (натуральне число).

*Приклади:*

1. В табл. 11.4 наведено дані чотирьох вибірок різних об'ємів. Порівняємо їх дисперсії за критерієм Бартлета.

Таблиця 11.4

## Дані 4-х вибірок

1	2	3	4
60.8	68.7	102.6	87.9
57.0	67.7	102.1	84.2
65.0	74.0	100.2	83.1
58.6	66.3	96.5	85.7
61.7	69.8		90.3

Набираємо дані з табл. 11.4 у матрицю  $X$ :

```
>> X=[60.8 1; 57.0 1; 65.0 1; 58.6 1; 61.7 1; ...
      68.7 2; 67.7 2; 74.0 2; 66.3 2; 69.8 2; ...
      102.6 3; 102.1 3; 100.2 3; 96.5 3; ...
      87.9 4; 84.2 4; 83.1 4; 85.7 4; 90.3 4]; % Дані 4-х вибірок
```

```
>> Btest(X,0.1); % тест Бартлета
```

The number of samples are: 4

Sample	Size	Variance
1	5	9.3920
2	5	8.5650
3	4	7.6567
4	5	8.3880

Bartlett's Test for Equality of Variances  $X^2=0.0328$ ,  $df= 3$ ,  $F= 0.0109$ ,  $df1= 3$ ,  $df2=14$   
 Probability associated to the Chi-squared statistic = 0.9984  
 The associated probability for the Chi-squared test is equal or larger than 0.10  
 So, the assumption of homoscedasticity was met.

Тест Барлетта для порівняння дисперсій:  $\chi^2=0.0328$ ,  $df= 3$ ,  $F= 0.0109$ ,  $df2=14$   
 Ймовірність появи такого значення статистики  $\chi^2$ -квадрат = 0.9984  
 Ця ймовірність (що зв'язана з  $\chi^2$ -квадрат статистикою) більше, ніж 0.10  
 Таким чином, нульова гіпотеза не може бути відхилена

Тест показав, що можна прийняти 0-гіпотезу: дисперсії однакові.

Пояснимо отримані дані тесту:  $X^2=0.0328$  – це величина  $\chi^2 = \frac{B}{C}$ ;  
 $k = 4$  – кількість вибірок;  $df = k - 1 = 3$  – число ступенів свободи (ЧСС)  
 для критерію  $\chi^2$ -квадрат;  $df1 = df$  – ЧСС чисельника дисперсійного  
 відношення Фішера;  $df2 = \sum n_i - k - 1 = 14$  – ЧСС знаменника  
 дисперсійного відношення Фішера;  $F = \frac{X^2}{k-1} = 0.0109$  – статистика  
 Фішера. Далі ясно за текстом.

2. Генеруються чотири вибірки різних об'ємів, елементи яких розподілені за стандартним нормальним законом. Порівнюються їх дисперсії на рівні значимості 0.1.

```
>> a1=[normrnd(0,1,10,1);normrnd(0,1,8,1);normrnd(0,1,4,1);normrnd(0,1,7,1)];
>> a2=[ones(10,1);2*ones(8,1);3*ones(4,1);4*ones(7,1)];
>> X=[a1 a2];
>> Btest(X,0.1);
```

Кількість вибірок: 4

Вибірка	Об'єм	Дисперсія
1	10	0.4600
2	8	0.5129
3	4	0.5619
4	7	1.6060

Тест Бартлета порівняння дисперсій:  $\chi^2$ -кв=1.8214,  $df=3$ ,  $F=0.6071$ ,  $df1=3$ ,  $df2=24$

Ймовірність статистики  $\chi^2$ -квадрат:  $P = 0.6103$

$P$  не менше, ніж 0.10

Нульова гіпотеза не може бути відхиленою (тобто приймається)

### **COCHTEST** Критерій Кокрена для порівняння дисперсій

Дана функція провадить тест Кокрена для порівняння дисперсій декількох вибірок. У даній реалізації передбачена можливість порівняння вибірок різних об'ємів. Автори – А. Трухильо-Ортіс (A. Trujillo-Ortiz) і Р. Ернандес-Уоллс (R. Hernandez-Walls).

Алгоритм методу наступний. Нехай з  $k$  генеральних сукупностей з однаковими дисперсіями, але, можливо, з різними середніми узяти вибірки однакового об'єму  $n$ . Для всіх вибірок треба знайти вибіркові середні і дисперсії. Вони є оцінками відповідних генеральних характеристик. Позначимо через  $D_{max}$  максимальну за величиною вибіркову дисперсію. Знайдемо відношення:

$G = \frac{D_{max}}{\sum D_i - D_{max}}$ . Розподіл

величини  $G$  називається  $g$ -розподілом Кокрена. Він залежить від двох параметрів: об'єму кожної вибірки  $n$  (або її числа ступенів свободи  $df$ ) і їх кількості  $k$ . На рівні значимості  $\alpha$  0-гіпотезу можна прийняти, якщо виконується співвідношення:  $G \leq g_\alpha(df, k)$  де  $G$  – емпіричне значення

(реалізація) величини  $g$ , а  $g_a$  – квантиль  $g$ -розподілу Кокрена для  $k$  вибірок зі  $df$  ступенями свободи. Якщо ця нерівність не виконується,  $0$ -гіпотеза відхиляється.

*Синтаксис:*

**Cochtest(X,alpha)**

*Опис:*

Функція **Cochtest(X,alpha)** провадить тест Кокрена для вибірок, заданих у матриці  $X$ , на рівні значимості  $alpha$  (за замовчуванням 0.05). Як і в тесті Бартлета, тут немає параметрів, що повертаються, результати виводяться на екран. Вибірki можуть бути різного об'єму, тому аргумент  $X$  є матрицею із двома стовпцями: 1-й стовпець – це самі дані, а 2-й – номер вибірки (натуральне число).

*Приклади:*

1. Порівняємо дисперсії чотирьох вибірок різних об'ємів. Дані наведені у табл. 11.4 у прикладі застосування критерію Бартлета.

```
>> X=[60.8 1; 57.0 1; 65.0 1; 58.6 1; 61.7 1;  
      68.7 2; 67.7 2; 74.0 2; 66.3 2; 69.8 2;  
      102.6 3; 102.1 3; 100.2 3; 96.5 3;  
      87.9 4; 84.2 4; 83.1 4; 85.7 4; 90.3 4]; % 4 вибірки  
>> Cochtest(X,0.1); % тест Кокрена з рівнем значимості 0.1
```

Кількість вибірок: 4

Вибірka	Об'єм	Дисперсія
1	5	9.3920
2	5	8.5650
3	4	7.6567
4	5	8.3880

Cochran's Test for Equality of Variances  $C = 0.2762$

Probability associated to the Cochran's statistic = 0.4719

The associated probability for the Cochran' test is equal or larger than 0.10

So, the assumption of homoscedasticity was met.

Значення статистики Кокрена для порівняння дисперсій  $C = 0.2762$

Ймовірність значення статистики Кокрена = 0.4719

Ця ймовірність не менше, ніж 0.10

Нульова гіпотеза не відхиляється

2. Генеруються чотири вибірки різних об'ємів, елементи яких розподілені за стандартним нормальним законом. Порівнюються їх дисперсії на рівні значимості 0.1.

```
>> a1=[normrnd(0,1,10,1);normrnd(0,1,8,1);normrnd(0,1,4,1);normrnd(0,1,7,1)];  
>> a2=[ones(10,1);2*ones(8,1);3*ones(4,1);4*ones(7,1)];  
>> X=[a1 a2];  
>> Cochtest(X,0.1); % тест Кокрена
```

Кількість вибірок: 4

Вибірка Об'єм Дисперсія

1	10	0.5830
2	8	0.9397
3	4	0.7517
4	7	0.7960

Значення статистики Кокрена для порівняння дисперсій  $C = 0.3060$

Ймовірність значення статистики Кокрена = 0.8468

Ця ймовірність не менше, ніж 0.10

Нульова гіпотеза не відхиляється

## 12. Дисперсійний, коваріаційний і регресійний аналізи

В **Statistics Toolbox** є множина функцій для рішення задач дисперсійного аналізу, регресійного аналізу й побудови узагальнених лінійних моделей. Майже всі вони описуються в цьому розділі. Проте декілька функцій регресійного аналізу в **MatLab** традиційно включені в розділ 9 – "Статистична графіка". Це функції **lsline**, **refline**, **refcurve**. Функція **addedvarplot** у різних версіях **Statistics Toolbox** включається в різні розділи. У даному довіднику вона описана в цьому розділі. Функція **lsqnonneg** також розглянута в розділі 12, хоча в деяких версіях **Statistics Toolbox** вона включена в розділ 13 – "Нелінійні моделі".

Дисперсійний і регресійний аналізи базуються на однакових передумовах і використовують для оцінки параметрів моделей однаковий метод найменших квадратів (МНК). Різниця полягає у тому, що в регресійному аналізі всі змінні, у тому числі всі пояснючі змінні (аргументи) – кількісні (метричні), а в дисперсійному аналізі пояснючі змінні – якісні, що виміряні в найбільш загальній шкалі імен. Окремі стани якісної змінної називаються категоріями, які визначають належність спостережень до тієї чи іншої групи. Зовні ці два аналізи не схожі один з одним, а дисперсійний аналіз навіть має ще інше ім'я – "Порівняння вибірок".

У коваріаційному аналізі припускаються як якісні, так і кількісні пояснючі змінні (аргументи). З одного боку, дисперсійний аналіз був узагальнений, щоб урахувати в дисперсійній моделі лінійні ефекти кількісних змінних (і ця техніка дуже схожа на звичайний дисперсійний аналіз). Коваріаційний аналіз у такій формі реалізовано в **MATLAB** функцією **AOSTOOL**. З іншого боку, регресійний аналіз був узагальнений, щоб урахувати в регресійній моделі дані різних груп спостережень, тобто щоб урахувати в моделі кілька класифікаційних змінних-аргументів. Досягнуто це за допомогою так званих *dummy*-змінних, які дорівнюють одиниці для однієї категорії обраної якісної змінної, і нулю – для решти спостережень. Загальна кількість *dummy*-

змінних дорівнює сумі числа категорій за всіма якісними змінними. У кожній класифікаційній (якісній) змінній один зі станів приймається за еталонний, і відповідну *dummy*-змінну в модель *не включають*. Далі провадиться звичайний множинний регресійний аналіз, наприкінці якого одержують рівняння регресії для еталонних станів з виправленнями для всіх інших станів якісних змінних. *Dummy*-змінні можна створити за допомогою функції **DUMMYVAR**.

Результативна змінна в дисперсійному і регресійному аналізах має бути кількісною і неперервною. На противагу цієї традиційної постанови, в узагальнених лінійних моделях результативна ознака може бути дискретною (і навіть якісною), що приймає декілька окремих станів. Так, в логіт і пробіт моделях результативна ознака може бути дихотомічною, тобто такою, що має лише два стани, які кодуються числами {0, 1} (або словами "так", чи "ні"). В узагальнених лінійних моделях оцінюють функцію розподілу результативної змінної, причому деякий параметр цієї функції розподілу лінійно залежить від пояснюючих змінних. Класична (традиційна) регресійна модель є окремим (і найпростішим) випадком узагальненої моделі, так звана Normal Identify Model (модель з нормальною функцією розподілу і тотожною функцією зв'язку).

Матеріал в цьому розділі подано у наступній послідовності:

12.1. Дисперсійний аналіз .....	76
12.2. Коваріаційний аналіз .....	132
12.3. Регресійний аналіз .....	150
12.4. Діагностика регресійних моделей .....	180
12.5. Спеціальні процедури регресійного аналізу .....	197
12.6. Узагальнені лінійні моделі .....	212

## 12.1. Дисперсійний аналіз

**Дисперсійний аналіз реалізований в MATLAB функціями:**

<b>ANOVA1</b> Однофакторний дисперсійний аналіз (ANOVA) .....	77
<b>ANOVA2</b> Двохфакторний дисперсійний аналіз .....	85
<b>ANOVAN</b> Багатофакторний дисперсійний аналіз (MANOVA) .....	92

<b>MULTCOMPARE</b> Перевірка параметричних гіпотез за парного порівняння групових середніх .....	106
<b>KRUSKALWALLIS</b> Непараметричний тест Краскала-Уоліса .....	119
<b>FRIDMAN</b> Тест Фрідмана .....	123
<b>MANOVA1</b> Однофакторний багатовимірний дисперсійний аналіз .....	127
<b>MANOVACLUSTER</b> Дендрограма результатів однофакторного багатовимірної дисперсійного аналізу .....	130

## **ANOVA1** Однофакторний дисперсійний аналіз (ANOVA)

Вирішується задача збалансованого і незбалансованого однофакторного дисперсійного аналізу. Алгоритм, реалізований у цій функції, дозволяє порівнювати вибірки різного об'єму й з різними вибірковими дисперсіями. Функція повертає критичне значення  $p$ , яке треба порівнювати із заданим рівнем значимості для ухвалення того чи іншого рішення.

*Синтаксис:*

```

p = anova1(X)
p = anova1(X,group)
p = anova1(X,group,'displayopt')
[p,table] = anova1(...)
[p,table,stats] = anova1(...)

```

*Опис:*

Функція **p = anova1(X)** дозволяє провести однофакторний дисперсійний аналіз для порівняння середніх арифметичних значень однієї або декількох вибірок однакового об'єму. Вибірки визначаються вхідним

аргументом  $X$ , який задається як матриця з розмірністю  $m \times n$ , де  $m$  – число спостережень у вибірці (число рядків  $X$ ),  $n$  – кількість вибірок (число стовпців матриці  $X$ ). Вибірки мають бути незалежними. Результативним параметром функції є рівень значимості  $p$  нульової гіпотези. Нульова гіпотеза полягає в тому, що всі вибірки в матриці  $X$

узяті з однієї генеральної сукупності (або з різних генеральних сукупностей, але з рівними середніми). Значення  $p$  є ймовірністю помилки першого роду, тобто ймовірністю необґрунтовано відкинути нульову гіпотезу. Якщо значення  $p \gg 0$ , то нульова гіпотеза має бути відхилена, тобто хоча б для однієї вибірки середнє арифметичне явно відрізняється від інших значень. Вибір критичного рівня значимості  $p_{кр}$  для умови прийняття нульової гіпотези  $p \geq p_{кр}$  наданий досліднику. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

Під час проведення дисперсійного аналізу загальна дисперсія підрозділяється на дві складові:

міжгрупову дисперсію – дисперсію середніх арифметичних значень вибірок матриці  $X$  щодо загального середнього;

внутрішньогрупову дисперсію – сумарну дисперсію значень кожної вибірки щодо своїх вибірових середніх.

Результати розрахунку відображаються у двох графічних вікнах. У перше вікно виводиться таблиця з результатами однофакторного дисперсійного аналізу, у друге – діаграма розмаху для середніх арифметичних для заданих вибірок.

Таблиця з результатами однофакторного дисперсійного аналізу містить 6 стовпців.

1. Джерело мінливості (*Source*):

між групами (*Columns*);

усередині груп (*Error*);

загальна (*Total*).

2. Суму квадратів відхилень (*SS*) спостережуваних значень від середніх арифметичних за кожним видом мінливості.

3. Число ступенів свободи (*df*) за кожним видом мінливості.

4. Середні суми квадратів (*MS*) за кожним видом мінливості, обумовлені як відношення  $MS = SS / df$ .

5. Значення F-статистики Фішера – дисперсійне відношення  $F = MS / MSE$ .

6. Значення рівня значимості  $p$  для розрахованого значення статистики  $F$ . Якщо величина  $F$  збільшується, то значення  $p$  зменшується.

Діаграма розмаху середніх арифметичних значень будується функцією **boxplot** за вибірками, тобто за стовпцями матриці  $X$ . Чим більше різниця між центральними лініями на діаграмі розмаху (медіанами, або середніми арифметичними вибірок), тим більшою є статистика  $F$  і меншою відповідне значення рівня значимості  $p$ .

У функції **p=anova1(X,group)** другий аргумент *group* задає мітки вибірок  $X$  і назви відповідних їм графіків на діаграмі розмаху. Мітки вибірок *group* задаються вектором символів або масивом рядків. Число елементів вектора *group* має дорівнює кількості стовпців матриці  $X$ .

Вибірки можуть бути задані як вектор  $X$ . Тоді підрозділення елементів  $X$  на різні вибірки виконується за допомогою аргументу *group*, який може бути заданий як вектор, символічний масив або масив рядків. Належність елемента вектора  $X$  до вибірки визначається однаковим значенням відповідного елемента вектора *group*. Розмірність векторів  $X$  і *group* повинна збігатися. Значення вектора *group* є мітками відповідних графіків вибірок на діаграмі розмаху середніх арифметичних.

Векторна форма завдання вибірок  $X$  за класифікаційною змінною *group* дозволяє провадити однофакторний дисперсійний аналіз з вибірками неоднакового об'єму. Якщо елемент змінної *group* містить порожній рядок, порожній елемент масиву або нечислове значення NaN, то відповідний елемент даних у векторі  $X$  ігнорується під час розрахунку.

У функції **p=anova1(X,group,'displayopt')** третій аргумент *'displayopt'* дозволяє за *'displayopt'='on'* явно задати обов'язкове відображення графічних вікон з таблицею результатів дисперсійного аналізу й діаграмою розмаху, або не виводити ці графічні вікна за *'displayopt'='off'*. Значення за замовчуванням *'displayopt'='on'*.

Функція `[p,table]=anova1(...)` повертає таблицю з результатами однофакторного дисперсійного аналізу в текстовій формі в командне вікно **MatLab**.

Функція `[p,table,stats]=anova1(...)` додатково повертає структуру *stats*, що використовується для проведення парних порівнянь середніх арифметичних різних вибірок. Перевірка параметричної гіпотези про рівність двох середніх арифметичних виконується за допомогою функції **multcompare**. Структура даних *stats* передається у цю функцію як аргумент.

Однофакторний дисперсійний аналіз заснований на наступних припущеннях:

дані у вибірках розподілені нормально;

дані всіх вибірок мають однакову дисперсію;

спостереження у кожній вибірці взаємно незалежні.

Однофакторний дисперсійний аналіз нечутливий до малих відхилень від першої й другої умов.

*Приклади:*

1. Однофакторний дисперсійний аналіз для п'яти вибірок з математичними очікуваннями від 1 до 5. Спочатку створюється матриця розміром  $5 \times 5$  – 5 вибірок по 5 спостережень. Далі до всіх спостережень додається нормально розподілена похибка.

<pre>&gt;&gt; X = meshgrid(1:5)</pre>	<pre>&gt;&gt; X = X + normrnd(0,1,5,5)</pre>
X =	X =
1 2 3 4 5	1.4254 2.7910 3.4819 2.5098 5.8232
1 2 3 4 5	0.5884 3.8792 4.9166 5.3241 3.8831
1 2 3 4 5	-0.4684 2.7780 1.2092 3.0028 4.7924
1 2 3 4 5	-0.1533 3.0392 5.1634 5.3195 4.4529
1 2 3 4 5	0.4247 3.4649 2.7069 2.3626 4.2490

Результати порівняння стовпців матриці *X*:

```
>> p = anova1(X)
p =
1.2765e-006
```

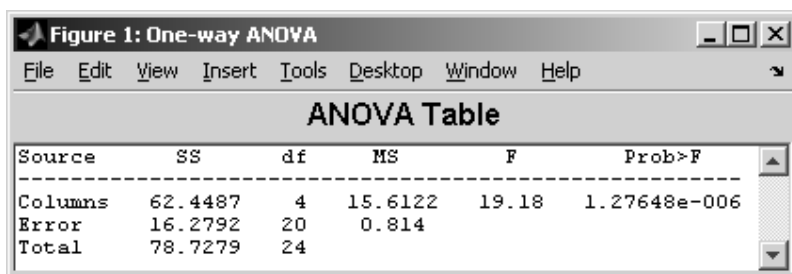


Рис. 12.1. Вікно результату обчислень

На рис. 12.1 представлені результати дисперсійного аналізу п'яти вибірок. Оскільки рівень значимості для обчисленого значення статистики Фішера  $F=10.46$  виявився меншим за  $0,01$  ( $p = 1.2765 \cdot 10^{-6}$ ), то робимо висновок про наявність значимих відмінностей між середніми стовпців матриці  $X$ . На рис. 12.2 зображені "вусаті коробки" Тюки для цих п'яти вибірок. Як видно з цієї діаграми, "коробка" Тюки для першої вибірки (першого стовпця матриці  $X$ ) явно відрізняється від інших "коробок" (для інших вибірок). Уже по цьому результату можна зробити висновок про існування значимих відмінностей між групами. Інші чотири "коробки" попарно перекриваються, що говорить про близькість другої і третьої, третьої і четвертої, четвертої і п'ятої вибірок і про незначимість відмінностей між їхніми середніми.

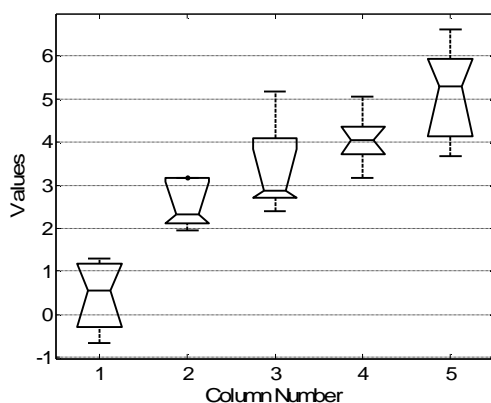


Рис.12.2. "Коробки" Тюки для п'яти вибірок

2. Однофакторний дисперсійний аналіз для п'яти вибірок з математичними очікуваннями, що змінюються в межах від 1 до 5. Мітки вибірок задані змінною *group*.

>> X = meshgrid(1:5)

|| 1.5391 3.0197 3.2894 5.3598 4.1570

```

X =
  1  2  3  4  5
  1  2  3  4  5
  1  2  3  4  5
  1  2  3  4  5
  1  2  3  4  5
>> X = X + normrnd(0,1,5,5)
X =
  0.4148  2.1474  3.4819  3.4055  6.3921
  1.2029  4.2595  4.5988  6.0550  3.6275
  0.0393  1.7689  2.3042  3.5807  4.4150
  1.0160  3.4167  3.1351  3.0396  2.7739
>> group = ['Aa'; 'Bb'; 'Cc'; 'Dd'; 'Ee']
group =
  Aa
  Bb
  Cc
  Dd
  Ee
>> p = anova1(X, group)
p =
  5.3531e-004

```

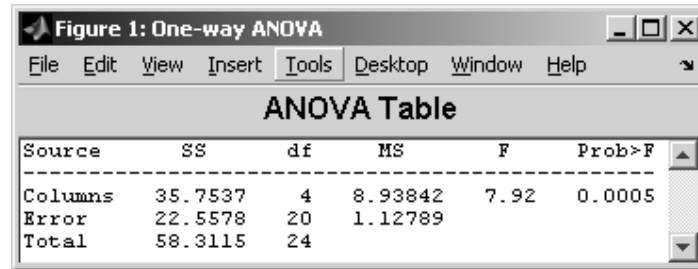


Рис. 12.3. Вікно результату обчислень

На рис. 12.3 показані результати обчислень. Оскільки виявилось, що  $p=0.0005 < 0.01$ , робимо висновок про існування значимості відмінностей між вибірками спостережень (стовпцями матриці  $X$ ).

На рис. 12.4 побудовані коробки Тьюки. На відміну від попереднього рис. 12.2 кожна вибірка тепер має своє ім'я (а не просто номер). Ця діаграма явно вказує на значиму відмінність між першою вибіркою й усіма іншими.

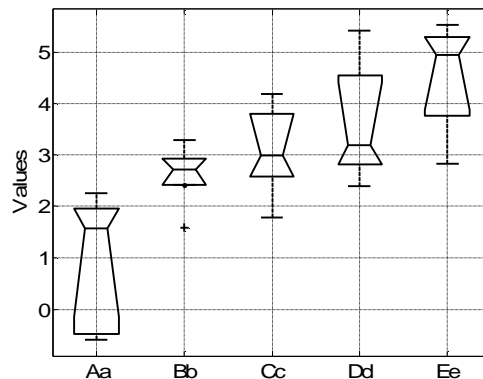


Рис. 12.4. "Коробки" Тьюки для п'яти вибірок з мітками

3. Однофакторний дисперсійний аналіз для чотирьох вибірок зі змінним об'ємом. Угрупування значень вектора  $X$  за вибірками виконується вектором цілих чисел  $group$  (рис. 12.5).

```
>> X = normrnd(0,1,104,1);  
>> group = unidrnd(4,104,1);  
>> p = anova1(X, group)  
p =  
    0.3104
```

Оскільки отримане значення  $p=0.3104>0.05$ , робимо висновок про відсутність значимих відмінностей між вибірками (нуль-гіпотеза не може бути відхилена).

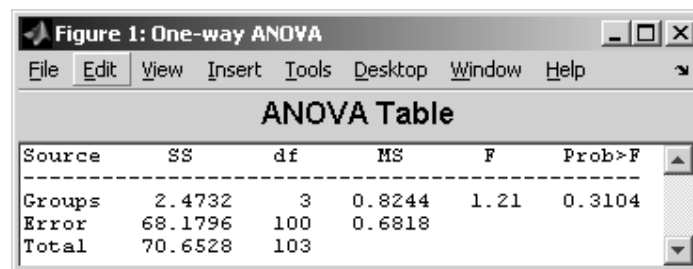


Рис. 12.5. Вікно результату обчислень

На рис. 12.6 видно, що всі "вусаті коробки" цього прикладу перекриваються. Оскільки відмінності між середніми лініями "коробок" перебувають у межах випадкових похибок, робимо висновок про відсутність значимих відмінностей.

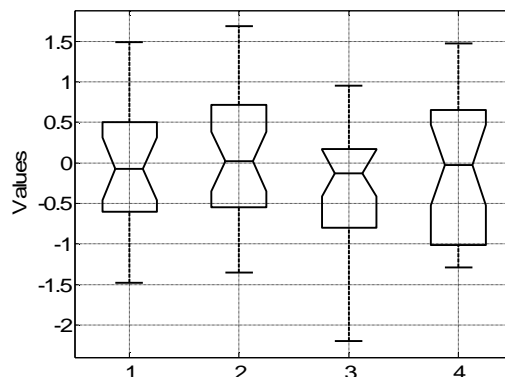


Рис. 12.6. "Коробки" Тьюки для чотирьох вибірок

4. Однофакторний дисперсійний аналіз для п'яти вибірок з математичними очікуваннями, що змінюються в межах від 1 до 5. Мітки вибірок задані змінною *group*. Функція повертає таблицю з результатами однофакторного дисперсійного аналізу в текстовій формі в командне вікно **MatLab**. Функція **anova1** повертає структуру даних *stats*.

```
>> X = meshgrid(1:5);
>> X = X + normrnd(0,1,5,5);
>> group = ['Aa'; 'Bb'; 'Cc'; 'Dd'; 'Ee'];
>> [p,table,stats] = anova1(X, group)
p =
    2.4289e-005
table =
    'Source'    'SS'        'df'    'MS'        'F'        'Prob>F'
    'Columns'  [46.2330]   [ 4]    [11.5583]   [12.8570]  [2.4289e-005]
    'Error'    [17.9796]  [20]    [ 0.8990]   []         []
    'Total'    [64.2127]  [24]    []          []         []
stats =
    gnames: {5x1 cell} % масив рядків
           n: [5 5 5 5 5] % усі вибірки одного розміру n=5
    source: 'anova1'
    means: [1.0530 1.5254 2.7926 4.0541 4.5313] % середні за вибірками
    df: 20 % dfe – залишкове число ступенів свободи
    s: 0.9481 % середньоквадратична похибка (корінь з MSE)
```

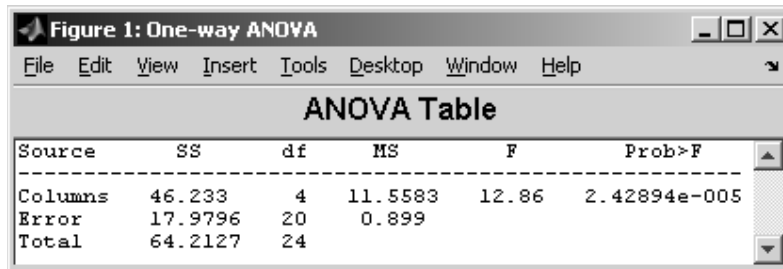


Рис. 12.7. Вікно результату обчислень

Цього разу таблиця дисперсійного аналізу виведена як текст (а не лише в графічному вікні рис. 12.7). Коментарі до структури *stats* додані вручну. За результатами таблиці можна встановити, що між вибірками є значущі відмінності.

Ще наочніше це видно, якщо розглянути графічну інформацію із цього аналізу на рис. 10.8:

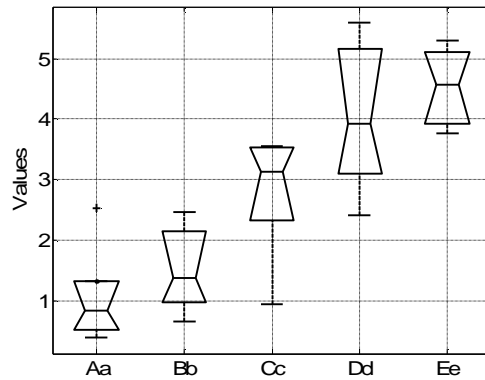


Рис. 12.8. "Коробки" Тьюки для п'яти вибірок

Із цієї діаграми видно, що перші дві вибірки явно відрізняються від інших трьох, тобто є два споріднених класи. У перший попадають перша й друга вибірки, а в другий клас – третя, четверта й п'ята.

### **ANOVA2** Двофакторний дисперсійний аналіз

Вирішується задача збалансованого двофакторного дисперсійного аналізу. Фактор – це якісна класифікаційна змінна, що визначає належність даних до певної групи (певної категорії цього фактора). В двофакторному аналізі є дві класифікаційні змінні із різними принципами угруповування. Дані називаються збалансованими, якщо є спостереження для всіх комбінацій категорій першого і другого фактора (такий план спостережень називається також повноблочним). Якщо спостереження заздалегідь не планувалися, то дані називаються незбалансованими. Для незбалансованого аналізу дивись функцію **anovan** (с. 86).

*Синтаксис:*

```
p = anova2(X, reps)
p = anova2(X, reps, 'displayopt')
[p, table] = anova2(...)
[p, table, stats] = anova2(...)
```

*Опис:*

Функція **anova2(X, reps)** дозволяє провести двофакторний дисперсійний аналіз для порівняння середніх арифметичних значень  $m$

стовпців і  $n$  рядків у матриці  $X$ . Дані в стовпцях і рядках відповідають різним рівням факторів  $A$  і  $B$ . Кількість повторних спостережень для всіх комбінацій рівнів факторів  $A$  і  $B$  задається вхідним аргументів  $reps$ . Значення  $reps$  повинне бути цілим числом, однаковим для всіх комбінацій. Дисперсійний аналіз для змінного числа спостережень за рівнями факторів використовує функцію багатофакторного дисперсійного аналізу **anovan**.

Якщо фактор  $A$  має два рівні, фактор  $B$  – три рівні й розглядаються по два спостереження для кожної комбінації рівнів факторів  $A$  і  $B$  ( $reps = 2$ ), то матрицю  $X$  (спостережень результативної ознаки) слід сформулювати так, як показано на рис. 12.9.

Кожна позиція в матриці відповідає конкретному рівню факторів  $A$ ,  $B$  і номеру повторення. У фактора  $A$  є 2 рівні, у фактора  $B$  – 3 рівні, і щодо кожної комбінації рівнів – по 2 виміри. На рис. 12.9 індексами у вимірів  $x_{ijk}$  відзначені:  $i$  – рівень фактора  $B$ ,  $j$  – номер стовпця (рівень фактора  $A$ );  $k$  – номер повторення експерименту за даної комбінації рівнів факторів.

За замовчуванням  $reps = 1$ , і в цьому випадку у результативному параметрі  $p$  повертаються два  $p$ -значення у вигляді

вектора. Його перший елемент – це  $p$ -значення для перевірки 0-гіпотези для фактора  $A$  (про рівність середніх стовпців), а другий – для фактора  $B$  (про рівність середніх рядків). Якщо  $reps > 1$ , то повертається вектор із трьох елементів. Паралельні спостереження дозволяють додатково оцінити взаємодію факторів, тому в  $p(3)$  повертається  $p$ -значення для перевірки 0-гіпотези про незалежність факторів. Мала величина будь-

$A=1$	$A=2$		
$x_{111}$	$x_{121}$	}	$B = 1$
$x_{112}$	$x_{122}$		
$x_{211}$	$x_{221}$	}	$B = 2$
$x_{212}$	$x_{222}$		
$x_{311}$	$x_{321}$	}	$B = 3$
$x_{312}$	$x_{322}$		

**Рис. 12.9. Матриця даних для двофакторного дисперсійного аналізу**

якого  $p$ -значення (що менше прийнятого рівня значимості, наприклад, 0.01) вимагає відкинути відповідну 0-гіпотезу.

Функція **anova2** за замовчуванням обчислює стандартну таблицю дисперсійного аналізу. Вона показана нижче в прикладах. Усі відмінності в даних розділяються на три або чотири групи залежно від відсутності або наявності повторних спостережень:

розкид між середніми стовпців (вплив фактора  $A$ );

розкид між середніми рядків (вплив фактора  $B$ );

розкид через взаємодію факторів (якщо  $reps > 1$ );

випадковий розкид.

Таблиця дисперсійного аналізу складається з 6 стовпців.

1. Джерело мінливості даних (*Source*). Таких джерел три або чотири: розкид між середніми стовпців (*Columns*), між середніми рядків (*Rows*), через взаємний вплив факторів за наявності повторних вимірів (*Interaction*) і випадковий розкид (*Errors*). Крім того, видається також сумарний розкид (*Total*).

2. Сума квадратів відхилень (*Sum of Squares*, скорочено *SS*).

3. Число ступенів свободи (*the degrees of freedom*, або *df*).

4. Дисперсії (*Mean of Squares*, або *MS*), які підраховуються як  $SS/df$ .

5.  $F$ -статистика Фішера ( $F$ ), що є відношенням величин  $MS$  до  $MSE$ .

6. Величина  $p$ -значення, яка обчислена за допомогою функції розподілу Фішера. Чим більше  $F$ -статистика, тим менше  $p$ -значення.

Величина  $p$  є ймовірністю помилки першого роду, або ймовірністю необґрунтовано відхилити нульову гіпотезу. Якщо значення  $p$  близько до нуля, то нульова гіпотеза має бути відхилена. Значення  $p \approx 0$  для гіпотези  $H_{0A}$  означає, що хоча б одне із середніх арифметичних, розрахованих по стовпцях, відрізняється від інших значень, тобто існує значимий головний ефект фактора  $A$ . Значення  $p \approx 0$  для гіпотези  $H_{0B}$  означає, що хоча б одне із середніх арифметичних, розрахованих по рядках, відрізняється від інших значень, тобто існує значимий головний

ефект фактора  $B$ . Значення  $p \approx 0$  для гіпотези  $H_{0AB}$  означає, що існує значимий ефект взаємодії між факторами  $A$  і  $B$ .

Вибір критичного рівня значимості  $p_{кр}$  для умови прийняття нульової гіпотези  $p \geq p_{кр}$  надано досліднику. У більшості практичних випадків  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

В ході проведення дисперсійного аналізу загальна дисперсія підрозділяється на 3 або 4 складові залежно від значення  $reps$ :

дисперсію між середніми арифметичними за стовпцями матриці  $X$  (дисперсія між середніми арифметичними за фактором  $A$ );

дисперсію між середніми арифметичними за рядками матриці  $X$  (дисперсія між середніми арифметичними за фактором  $B$ );

дисперсію, що обумовлена взаємодією між рядками й стовпцями  $X$  (розраховується, якщо  $reps > 1$ );

залишкову дисперсію.

Функція **`p=anova2(X,reps,'displayopt')`** в третьому аргументі *'displayopt'* дозволяє відобразити графічне вікно з таблицею результатів дисперсійного аналізу за *'displayopt'='on'*, або не виводити це графічне вікно за *'displayopt'='off'*. Значення за замовчуванням *'displayopt'='on'*.

Функція **`[p,table]=anova2(...)`** повертає таблицю з результатами двофакторного дисперсійного аналізу в текстовій формі в командне вікно **MatLab**.

Функція **`[p,table,stats]=anova2(...)`** у третьому результативному параметрі *stats* повертає структуру, яку надалі можна використовувати для більш точного порівняння середніх. Функція *anova2* перевіряє 0-гіпотези про рівність середніх стовпців, рядків і, можливо, про взаємний вплив факторів, а альтернативними є гіпотези про відмінність відповідних середніх. Іноді потрібно більш точне дослідження: саме які пари середніх мають значимі різниці, а які не мають. Для цього можна використовувати функцію **`multcompare`** (див. нижче у цьому розділі), на вхід якої й потрібно подати структуру *stats*.

Приклади:

1. Двофакторний дисперсійний аналіз для шести рівнів за кожним фактором без повторень.

```
>> X=unifrnd(0,1,6,6); % матриця 6x6  
>> p=anova2(X,1) % одне повторення  
p =  
0.2580 0.0894
```

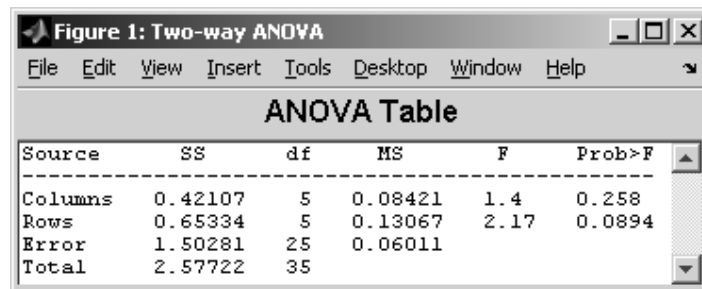


Рис. 12.10. Результати обчислень головних ефектів

На рис. 12.10 в графічному вікні представлено результати аналізу. Оскільки всі  $p > 0.05$ , то приймаємо висновок про відсутність значимих відмінностей між стовпцями й рядками матриці  $X$ .

2. Двофакторний дисперсійний аналіз для шести рівнів за першим фактором, двох рівнів за другим фактором і 3-ма повтореннями.

```
>> X = normrnd(0,1,6,6); % матриця 6x6  
>> p=anova2(X,3) % три повторення  
p =  
0.6437 0.5074 0.8311
```

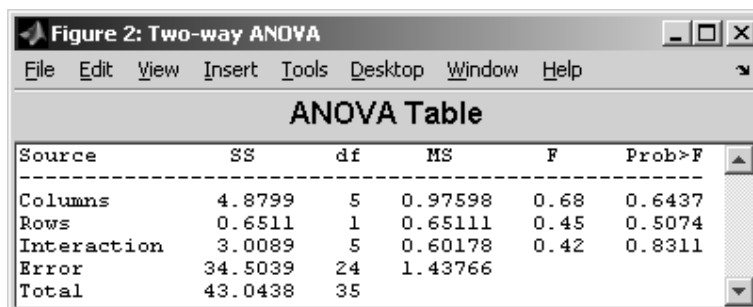


Рис. 12.11. Результати обчислень із взаємодією АВ

На рис. 12.11 представлені результати аналізу. Значення всіх трьох імовірностей більше 0.05. Звідси виходить, що немає значимих відмінностей між ефектами  $A$ ,  $B$  і  $AB$ .

3. Двофакторний дисперсійний аналіз для шести рівнів за першим фактором, двох рівнів за другим фактором і 3-ма повтореннями. Функція повертає вектор рівнів значимості  $p$ , таблицю двофакторного дисперсійного аналізу  $table$ , структуру  $stats$  для парних порівнянь середніх арифметичних вибірок.

```
>> X = normrnd(0,1,6,6);
>> [p,table,stats] = anova2(X,3) % дані 6x2x3
p =
    0.2623    0.9415    0.5236
table =
    'Source'      'SS'      'df'      'MS'      'F'      'Prob>F'
    'Columns'    [ 6.8848] [ 5]      [1.3770]  [1.3927] [0.2623]
    'Rows'       [ 0.0054] [ 1]      [0.0054] [0.0055] [0.9415]
    'Interaction' [ 4.2383] [ 5]      [0.8477] [0.8574] [0.5236]
    'Error'      [23.7278] [24]      [0.9887]  []        []
    'Total'      [34.8563] [35]      []         []        []
stats =
    source: 'anova2'
    sigmasq: 0.9887          % MSE – незміщена оцінка залишкової дисперсії
    colmeans: [-0.5274 0.3065 0.1038 0.7801 0.3728 -0.3201] % середні по стовпцях
    coln: 6
    rowmeans: [0.1070 0.1316] % середні рядків
    rown: 18
    inter: 1                % взаємодія AB
    pval: 0.5236           % Взаємодія незначима, тому що pval > 0.05
    df: 24                 % dfe
```

На рис. 12.12 представлені результати аналізу. Можна зробити висновок, що всі ефекти незначимі.

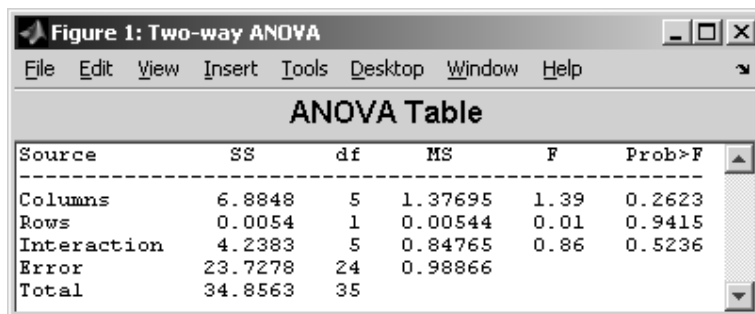


Рис. 12.12. Результати обчислень

4. У прикладі, взятому з бази **MatLab**, досліджується вплив марки й способу приготування попкорну на прибуток від продажів. Стівпці матриці – це марки попкорну: Gourmet, National і Generic. Рядки – спосіб прготування: у маслі (Oil) або повітряний (Air). За кожною комбінацією рівнів факторів проведено по 3 виміри прибутків. Дані перебувають у файлі popcorn.mat у папці ststs:

```
>> load popcorn
>> whos popcorn
Name      Size      Bytes Class
popcorn   6x3       144   double array
Grand total is 18 elements using 144 bytes
>> popcorn
```

popcorn =					
5.5000	4.5000	3.5000	6.0000	4.0000	3.0000
5.5000	4.5000	4.0000	6.5000	5.0000	4.0000
7.0000	5.5000	5.0000	7.0000	5.0000	4.5000

```
>> [p,table,stats] = anova2(popcorn,3)
p =
    0.0000    0.0001    0.7462
table =
    'Source'      'SS'      'df'      'MS'      'F'      'Prob>F'
    'Columns'    [15.7500] [ 2]      [7.8750]  [56.7000] [7.6790e-007]
    'Rows'       [ 4.5000] [ 1]      [4.5000]  [32.4000] [1.0037e-004]
    'Interaction' [ 0.0833] [ 2]      [0.0417] [ 0.3000] [ 0.7462]
    'Error'      [ 1.6667] [12]      [0.1389]  []         []
    'Total'      [22]      [17]      []         []         []
stats =
    source: 'anova2'
    sigmasq: 0.1389
    colmeans: [6.2500 4.7500 4]
    coln: 6
    rowmeans: [4.5000 5.5000]
    rown: 9
    inter: 1
    pval: 0.7462
    df: 12
```

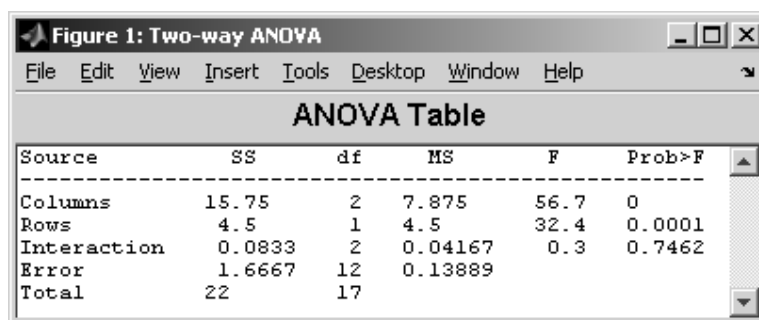


Рис. 12.13. Результати обчислень

Вибравши рівень значимості 0.01, можна зробити висновок про те, що й торговельна марка, і спосіб готування суттєво впливають на рівень продажів. У той же час ці фактори практично незалежні. Дійсно,  $p(3) = 0.7462 > 0.05$ , а  $p(1)$  і  $p(2)$  близькі до нуля (рис. 12.3).

## **ANOVA** Багатофакторний дисперсійний аналіз (MANOVA)

Вирішує задачу збалансованого й незбалансованого багатофакторного дисперсійного аналізу.

*Синтаксис:*

```
p = anovan(X,group)
p = anovan(X,group,'model')
p = anovan(X,group,'model',sstype)
p = anovan(X,group,'model',sstype,gnames)
p = anovan(X,group,'model',sstype,gnames,'displayopt')
[p,table] = anovan(...)
[p,table,stats] = anovan(...)
[p,table,stats,terms] = anovan(...)
```

*Опис:*

Функція **p=anovan(X,group)** дозволяє провести багатофакторний дисперсійний аналіз для порівняння середніх арифметичних значень  $N$  вибірок. Вибірki задаються у вигляді вектора  $X$ . Фактори і їх рівні визначаються масивом рядків *group*. Кожен рядок аргументу *group* містить перелік рівнів факторів, відповідних до спостережень у векторі  $X$ . Перелік у кожному рядку може бути вектором, масивом символів, або масивом рядків, що містять рядкові змінні. Кількість елементів **group** повинне збігатися із числом елементів у векторі  $X$ . Повертається вектор  $p$ -значень для перевірки 0-гіпотез для відповідних факторів і їх взаєдій (при наявності паралельних спостережень). Мала величина будь-якого  $p$ -значення (меншого прийнятого рівня значимості) вимагає відхилити відповідну 0-гіпотезу.

Приклад завдання вхідних аргументів вибірки  $X$  з 8 елементів:  
 $X=[x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8]$ , де замість  $x_i$  треба набирати числа і аргументи *group* для 3-х факторів, що мають по два рівні:

```
>> group={[1 2 1 2 1 2 1 2]; ['hi';'hi';'lo';'lo';'hi';'hi';'lo';'lo'];...  
  {'may' 'may' 'may' 'may' 'june' 'june' 'june' 'june'}}  
group =  
 [1×8 double] % числовий масив (у таких дужках уводиться масив чисел);  
 [8×2 char] % масив символів (у таких дужках уводиться масив символів);  
 {1×8 cell} % масив рядків (у таких дужках уводиться масив рядків).
```

Рівні першого фактора кодуються цілими числами, другого – вектором двосимвольних елементів, третього – символьними рядками різної довжини. Значення  $x_1$  відповідає рівню першого фактора рівного 1, другого – 'hi', третього – 'may', і т.д.

Результативний вектор  $p$  містить значення рівнів значимості нульової гіпотези щодо головних ефектів. Елемент  $p(1)$  є рівнем значимості нульової гіпотези  $H_{0A}$ , яка полягає у тому, що вибірки, відповідні до рівнів фактора  $A$ , взяті з однієї генеральної сукупності. Елемент  $p(2)$  є рівнем значимості нульової гіпотези  $H_{0B}$ , яка полягає у тому, що вибірки, відповідні до рівнів фактора  $B$ , узяті з однієї генеральної сукупності. І так далі щодо всіх інших елементів вектора  $p(i)$ .

Рівень значимості  $p$  є ймовірністю помилки першого роду, або ймовірністю необґрунтовано відкинути нульову гіпотезу ( $H_0$ ).  $H_0$  має бути відкинута, якщо  $p$  близько до нуля. Значення  $p \approx 0$  для гіпотези  $H_{0A}$  означає, що хоча б одне із середніх арифметичних, розрахованих за вибірками, відповідних до різних категорій фактора  $A$ , відрізняється від інших значень, тобто існує значимий головний ефект фактора  $A$ . І так саме за всіма іншими факторами (елементами вектора  $p(i)$ ).

Вибір критичного рівня значимості  $p_{кр}$  для умови прийняття нульової гіпотези  $p \geq p_{кр}$  надано досліднику. У більшості практичних задач  $p_{кр}$  ухвалюють рівним 0.05 або 0.01.

Під час проведення дисперсійного аналізу загальна мінливість за замовчуванням підрозділяється на:

мінливість, обумовлену змінами рівнів факторів для заданої моделі; залишкову мінливість.

За замовчуванням функція будує графічне вікно зі стандартною таблицею багатофакторного дисперсійного аналізу. Вона показана нижче в прикладах. Усі відмінності в даних розділяються на групи залежно від обліку тих або інших факторів і їх комбінацій. Таблиця дисперсійного аналізу складається з 6 стовпців.

1. Джерело розкиду даних (*Source*). Спочатку перелічуються джерела, що пов'язані з факторами і їх комбінаціями. Далі враховується випадковий розкид (*Error*). Наприкінці видається сумарний розкид (*Total*).

2. Сума квадратів відхилень (*Sum of Squares*, скорочено *Sum Sq.*) значень вибірок за кожним видом мінливості від середніх вибірок.

3. Число ступенів свободи (*the degrees of freedom*, або *d.f.*).

4. Середні квадрати (*Mean Squares*, або *Mean Sq.*) – незміщені оцінки дисперсій, які підраховується як  $(Sum Sq.) / (d.f.)$ .

5. F-статистика Фішера (*F*), яка є відношенням відповідної величини *Mean Sq* до **Mean Sq** випадкових похибок.

6. Рівень значимості – *p*-значення, обчислене за допомогою функції розподілу Фішера. Чим більше F-статистика, тим менше *p*-значення.

Функція **`p=anovan(X,group,'model')`** дозволяє провести багатофакторний дисперсійний аналіз для заданої моделі оцінюваних ефектів. Результативний вектор *p* містить результати розрахунку рівнів значимості для нульових гіпотез щодо факторів моделі. Вид моделі факторів визначається вхідним аргументом '*model*'. Передбачені наступні види моделей: '*linear*' – лінійна модель, що включає головні ефекти; '*interaction*' – модель, що включає ефекти парних взаємодій; '*full*' – повна модель, що включає головні ефекти й ефекти парних взаємодій.

Крім зазначених рядкових значень, вхідний аргумент '*model*' може бути заданий як ціле число або вектор. Якщо вхідний аргумент '*model*'

заданий як ціле число  $k$  ( $k \leq n$ ) функція **anovan** дозволяє оцінити значимість взаємодій факторів до  $k$  рівня. Значення  $k = 1$  і  $k = 2$  відповідають '*model*'='linear' і '*interaction*' відповідно;  $k = N$  відповідає '*model*'='full'.

Використання вектора цілих чисел для визначення '*model*' дозволяє оцінити значимість заданих головних ефектів і ефектів взаємодій факторів. Кожний елемент вектора '*model*' кодує в десятковій формі бінарне число, що визначає наявність головних ефектів або ефектів взаємодій. Приклади кодування головних ефектів або ефектів взаємодій для 3 факторів наведені в табл. 12.1:

Таблиця 12.1

**Таблиця кодування ефектів факторів**

Бінарне кодування	Десятковий елемент вектора ' <i>model</i> '	Вид ефекту в дисперсійному аналізі
[0 0 1]	1	Головний ефект А
[0 1 0]	2	Головний ефект В
[1 0 0]	4	Головний ефект С
[0 1 1]	3	Ефект взаємодії АВ
[1 0 1]	5	Ефект взаємодії АС
[1 1 0]	6	Ефект взаємодії ВС
[1 1 1]	7	Ефект взаємодії АВС

Наприклад, якщо '*model*' = [2 4 6], то результативний вектор  $p$  містить значення рівня значимості для перевірки нульових гіпотез значимості головних ефектів  $B$ ,  $C$  і ефекту взаємодії  $BC$  у наведеному порядку.

Одним з можливих способів завдання вектора '*model*' є модифікація параметра *terms* функції **anovan**. Вектор *terms* кодує головні ефекти й ефекти взаємодій факторів згідно з форматом вхідного аргументу '*model*'. Наприклад, якщо після проведення дисперсійного аналізу за трьома факторами для '*model*' = [2 4 6] ефект взаємодій  $BC$  виявився статистично незначимим, то для корекції головних ефектів  $B$  і  $C$  необхідно використовувати '*model*' = [2 4].

Функція **p=anovan(X,group,'model',sstype)** дозволяє провести багатофакторний дисперсійний аналіз для заданого способу розрахунку суми квадратів відхилень  $SS$ . Спосіб розрахунку суми квадратів відхилень визначається аргументом *sstype*. Передбачено 3 способи розрахунку (*sstype* = 1, 2, 3). Значення за замовчуванням *sstype* = 3. Спосіб розрахунку суми квадратів відхилень впливає на результати багатофакторного дисперсійного аналізу для вибірок неоднакового об'єму.

Сума квадратів відхилень для ефекту кожного фактора визначається при порівнянні двох складових. За першим способом розрахунку для *model(i)*  $SS$  є різницею залишкових сум квадратів відхилень множини факторів *model(j)*, де  $j=1..(i-1)$ , і поточного фактора. За другим способом розрахунку для *model(i)*  $SS$  є різницею залишкової суми квадратів відхилень головних ефектів за винятком *model(i)* і залишкової суми квадратів відхилень головних ефектів з додаванням *model(i)*. За третім способом розрахунку для *model(i)*  $SS$  є різницею залишкової суми квадратів відхилень для всіх факторів в *model* за винятком *model(i)* і залишкової суми квадратів відхилень за всіма факторами в *model*.

Розглянемо приклад формування сум квадратів відхилень 1, 2, 3 типів для двох факторів  $A$  і  $B$ . Модель може містити два головні ефекти факторів  $A$ ,  $B$  і ефект взаємодії  $AB$ . Порядок ефектів у векторі '*model*' відповідає послідовності:  $A$ ,  $B$ ,  $AB$ . Величина  $R(\cdot)$  представляє залишкову суму квадратів відхилень для моделі, наприклад,  $R(A,B,AB)$  – залишкова сума квадратів відповідно до повної моделі,  $R(A)$  – залишкова сума квадратів, що відповідає одному головному ефекту фактора  $A$ ,  $R(1)$  – залишкова сума квадратів, що відповідає загальному середньому арифметичному. Три різні способи розрахунку суми квадратів для розглянутого прикладу наведені в табл. 12.2:

Таблиця 12.2

## Три моделі розрахунку суми квадратів

Ефект	Спосіб I ( <i>sstype</i> = 1)	Спосіб II ( <i>sstype</i> = 2)	Спосіб III ( <i>sstype</i> = 3)
A	$R(1) - R(A)$	$R(B) - R(A,B)$	$R(B,AB) - R(A,B,AB)$
B	$R(A) - R(A,B)$	$R(A) - R(A,B)$	$R(A,AB) - R(A,B,AB)$
AB	$R(A,B) - R(A,B,AB)$	$R(A,B) - R(A,B,AB)$	$R(A,B) - R(A,B,AB)$

Моделі розрахунку  $SS$  за третім способом мають обмеження щодо середнього квадратичного відхилення. Це означає, наприклад, що для залишкової суми квадратів  $R(B,AB)$ , масив ефектів  $AB$  обмежений за сумою від 0 до  $A$  для кожного значення  $B$  і до  $B$  для кожного значення  $A$ .

У функції **p=anovan(X,group,'model',sstype,gnames)** аргумент *gnames* служить для визначення міток  $N$  факторів у таблиці дисперсійного аналізу, де *gnames* може бути заданий як вектор символів або масив рядків, що містить рядкові змінні, а кожний елемент масиву відповідає одному спостереженню. Якщо вхідний аргумент **gnames** не заданий, у якості міток за замовчуванням ухвалюються  $'X_1'$ ,  $'X_2'$ ,  $'X_3'$ , ...,  $'X_N'$ .

У функції **p=anovan(X,group,'model',sstype,gnames,'displayopt')** аргумент *'displayopt'* дозволяє відобразити графічне вікно з таблицею результатів дисперсійного аналізу за *'displayopt'='on'*, або не виводити графічне вікно за *'displayopt'='off'*. Значення за замовчуванням *'displayopt'='on'*.

Функція **[p,table]=anovan(...)** повертає таблицю з результатами багатофакторного дисперсійного аналізу (включаючи мітки факторів) у текстовій формі в командне вікно **MatLab**.

Функція **[p,table,stats]=anovan(...)** повертає структуру **stats**, що використовується для проведення парних порівнянь середніх арифметичних вибірок. Перевірка параметричної гіпотези про рівність двох середніх арифметичних виконується за допомогою функції **multcompare**. Структура даних **stats** передається функції **multcompare** як вхідний аргумент.

Функція `[p,table,stats,terms]=anovan(...)` повертає вектор номерів головних ефектів і ефектів взаємодій *terms*, що використовувалися в ході проведення багатofакторного дисперсійного аналізу. Формат завдання оцінюваних ефектів *terms* відповідає формату вектора *'model'*. Якщо за виклику `anovan` був використаний вектор *'model'*, то *terms = 'model'*.

Вважається зручним усе вищесказане звести в табл. 12.3.

Таблиця 12.3

### Значення додаткових аргументів функції `anovan`

Ім'я	Можливі значення
<i>'sstype'</i>	1, 2 або 3, щоб задати тип обчислення впливу фактора (за замовчуванням 3, це найбільш точний спосіб обліку факторів)
<i>'varnames'</i>	Матриця символів або масив рядків, що задає імена факторів, по одному на кожний. За замовчуванням використовуються імена <i>'X1'</i> , <i>'X2'</i> , <i>'X3'</i> тощо
<i>'display'</i>	<i>'on'</i> (за замовчуванням) – виводиться таблиця дисперсійного аналізу; <i>'off'</i> – не виводиться.
<i>'random'</i>	Вектор індексів, що показує, які групуючи змінні є випадковими ефектами (за замовчуванням – усі)
<i>'alpha'</i>	Рівень значимості (за замовчуванням 0.05)
<i>'model'</i>	Модель обліку факторів. Можливі значення: <i>'linear'</i> (за замовчуванням) – обчислюються <i>p</i> -значення для перевірки впливу лише самих <i>n</i> факторів; <i>'interaction'</i> – обчислюються <i>p</i> -значення для перевірки впливу самих <i>n</i> факторів і всіх їх двофакторних взаємодій; <i>'full'</i> – обчислюються <i>p</i> -значення для перевірки впливу самих <i>n</i> факторів і всіх їх взаємодій з будь-якою кількістю факторів; ціле позитивне число <i>k</i> – максимальна кількість факторів, взаємодії яких враховуються. Повинне бути $k \leq n$ . Випадок $k = 1$ відповідає моделі <i>'linear'</i> , $k = 2$ – <i>'interaction'</i> , а $k = n$ – <i>'full'</i> ; Матриця факторів, що враховуються, – матриця з нулів і одиниць такої ж форми, як і у функції <code>x2fx</code> (див. нижче).

Для більш точного задання факторів і їх взаємодій, які потрібно оцінювати, для параметра *'model'* можна задати значення у вигляді матриці факторів з нулів і одиниць. Кожний рядок матриці визначає фактори і їх комбінації, які потрібно враховувати в моделі. У кожному рядку повинне бути *n* чисел (нулів або одиниць). Одиниці визначають ті фактори, комбінації яких враховуються. Зокрема, одинична матриця визначає модель *'linear'* (усі фактори враховуються поодиноці, без взаємодій). Якщо, наприклад, задати для імені *'model'* значення `[1 0 0; 0`

1 0; 1 0 1], то функція **anovan** поверне у результативному параметрі  $p$  вектор із трьох  $p$ -значень для перевірки впливу фактора  $A$ , фактора  $B$  і взаємодії факторів  $A$  і  $C$ .

Для прикладу в табл. 12.4 показана матриця факторів, що враховуються, для трьохфакторного дисперсійного аналізу і моделі *'full'*.

Таблиця 12.4

**Приклад матриці моделі для функції anovan**

Рядок матриці	Фактори, що враховуються
[1 0 0]	Фактор А
[0 1 0]	Фактор В
[0 0 1]	Фактор С
[1 1 0]	Взаємодія факторів А і В
[0 1 1]	Взаємодія факторів В і С
[1 0 1]	Взаємодія факторів А і С
[1 1 1]	Взаємодія факторів А, В і С

*Приклади:*

1. Дисперсійний аналіз для 3-х факторів на двох рівнях мінливості.

```
>> X=normrnd(0,1,8,1)
X =
-0.3775
-0.2959
-1.4751
-0.2340
0.1184
0.3148
1.4435
-0.3510

>> group={1 2 1 2 1 2 1 2}; ['hi';'hi';'lo';'lo';'hi';'hi';'lo';'lo'];
{'may' 'may' 'may' 'may' 'june' 'june' 'june' 'june'}
group =
[1x8 double]
[8x2 char]
{1x8 cell}
>> p=anovan(X,group)
p =
0.9122
0.8803
0.1711
```

В таблиці дисперсійного аналізу на рис. 12.4 використані стандартні позначення  $X_1$ ,  $X_2$ ,  $X_3$  для 3-х факторів. Оскільки паралельних спостережень немає, взаємодії факторів не обчислюються.

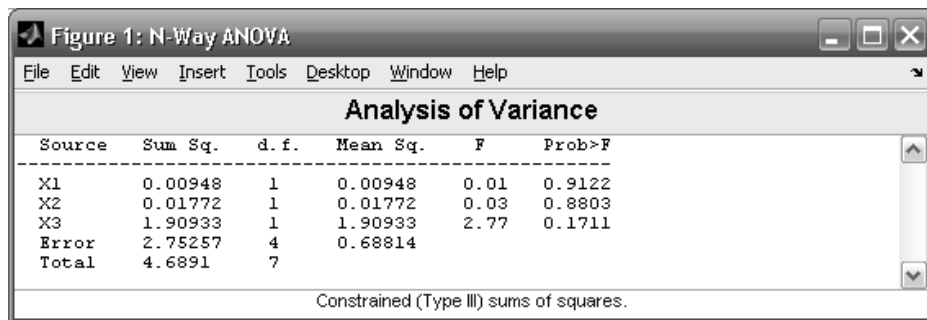


Рис. 12.14. Результати дисперсійного аналізу

За результатами аналізу видно, що значущі ефекти відсутні.

2. Дисперсійний аналіз для 3-х факторів, що змінюються на двох рівнях, і моделі, що містить ефекти взаємодій факторів.

```
>> X=normrnd(0,1,8,1);
>> group = {[1 2 1 2 1 2 1 2];
['hi';'hi';'lo';'lo';'hi';'hi';'lo';'lo']; {'may' 'may'
'may' 'may' 'june' 'june' 'june' 'june'}};
>> model='interaction';
>> p = anovan(X,group,model)
```

```
p =
0.3304 % фактор A
0.9088 % фактор B
0.3508 % фактор C
0.9941 % взаємодія факторів AB
0.8861 % взаємодія факторів AC
0.5117 % взаємодія факторів BC
```

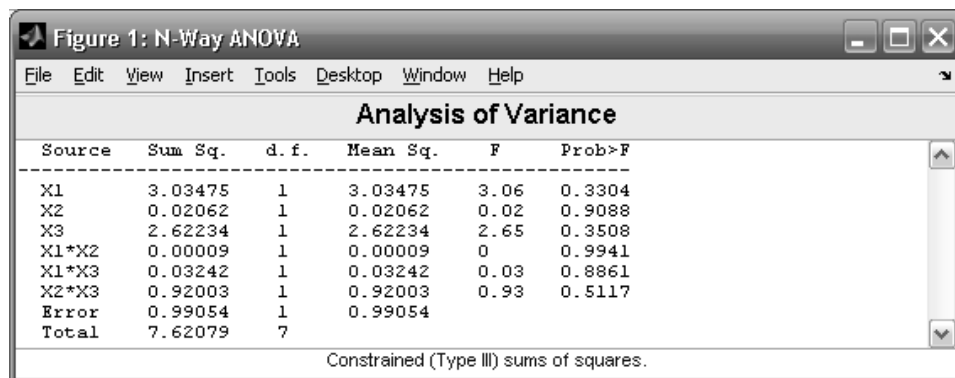


Рис. 12.15. Результати дисперсійного аналізу

На рис. 12.15 наведені результати дисперсійного аналізу повної моделі для 3-х факторів і трьох їх можливих взаємодій першого порядку. З результатів виходить, що всі шість ефектів незначимі.

3. Дисперсійний аналіз для 3-х факторів:  $A$ ,  $B$  і  $C$ , що змінюються на двох рівнях, і  $model = [2 \ 4 \ 6]$  для перевірки нульових гіпотез щодо головних ефектів  $B$ ,  $C$  і ефекту взаємодії  $BC$ .

```
>> X=normrnd(0,1,8,1);
>> group=[1 2 1 2 1 2 1 2];
['hi';'hi';'lo';'lo';'hi';'hi';'lo';'lo']; {'may' 'may'
'may' 'may' 'june' 'june' 'june' 'june'};
>> model=[2 4 6];
```

```
>> p=anovan(X,group,model)
p=
    0.3028    % фактор В
    0.0611    % фактор С
    0.0724    % взаємодія факторів ВС
```

На рис. 12.16 наведені результати дисперсійного аналізу, з якого виходить, що всі враховані в моделі ефекти незначимі.

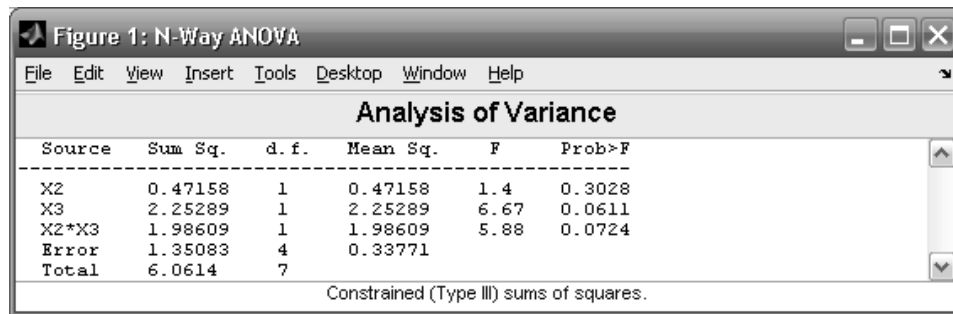


Рис. 12.16. Результати дисперсійного аналізу

4. Дисперсійний аналіз для 3-х факторів:  $A$ ,  $B$  і  $C$ , що змінюються на двох рівнях, і  $model = [2 \ 4 \ 6]$  для перевірки нульових гіпотез щодо головних ефектів  $B$ ,  $C$  і ефекту взаємодії  $BC$ , а також другого способу розрахунку суми квадратів різниць ( $SS$ ) за кожним видом мінливості (рис. 12.17).

```
>> X=normrnd(0,1,8,1);
>> group=[1 2 1 2 1 2 1 2];
['hi';'hi';'lo';'lo';'hi';'hi';'lo';'lo']; {'may' 'may'
'may' 'may' 'june' 'june' 'june' 'june'};
>> model=[2 4 6];
>> sstype=2;
```

```
>> p=anovan(X,group,model, sstype)
p=
    0.6796
    0.6457
    0.1240
```

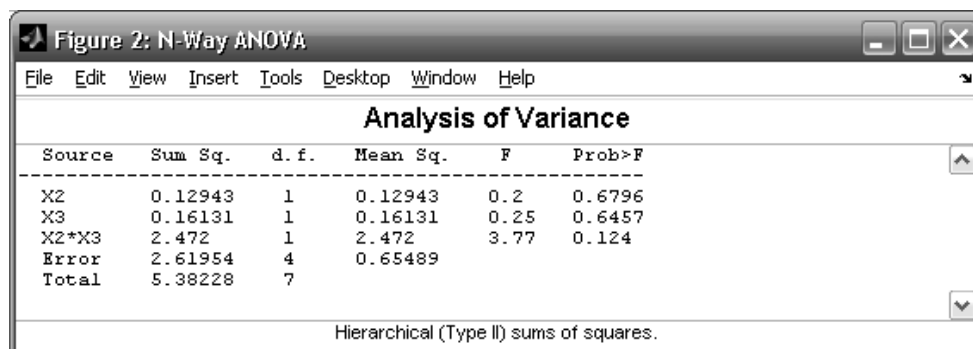


Рис. 12.17. Результати дисперсійного аналізу

5. Дисперсійний аналіз для 3-х факторів:  $A$ ,  $B$  і  $C$ , що змінюються на двох рівнях, і  $model = [1 \ 2 \ 4 \ 6]$  для перевірки нульових гіпотез щодо головних ефектів  $A$ ,  $B$ ,  $C$  і ефекту взаємодії  $BC$  за другим способом розрахунку сум квадратів відхилень ( $SS$ ) за кожним видом мінливості. У якості міток використовується масив рядків  $gnames = \{ 'Xx'; 'Yy'; 'Zz' \}$ .

```
>> X=normrnd(0,1,8,1);
>> group=[1 2 1 2 1 2 1 2];
['hi';'hi';'lo';'lo';'hi';'hi';'lo';'lo']; {'may'
'may' 'may' 'may' 'june' 'june' 'june'
'june'};
>> model=[1 2 4 6];
>> sstype=2;
```

```
>> gnames= {'Xx'; 'Yy'; 'Zz'};
>> p=anovan(X,group,model,sstype,gnames)
p=
    0.8156
    0.4569
    0.3284
    0.4489
```

Таблиця дисперсійного аналізу наведена на рис. 12.18.

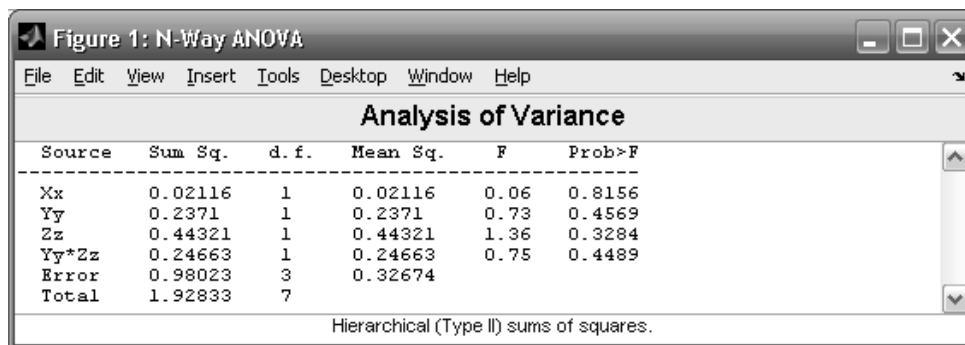


Рис. 12.18. Результати дисперсійного аналізу

6. Дисперсійний аналіз для 3-х факторів:  $A$ ,  $B$  і  $C$ , що змінюються на двох рівнях, і моделі, що містить ефекти взаємодій факторів за другим способом розрахунку сум квадратів відхилень ( $SS$ ) за кожним видом мінливості. У якості міток використовується масив рядків  $gnames = \{ 'Xx'; 'Yy'; 'Zz' \}$ . Крім рівнів значимості для зазначених нульових гіпотез, виводяться таблиця багатофакторного дисперсійного аналізу, структура для проведення парного порівняння середніх, вектор використаних факторів і їх взаємодій (рис. 12.19).

```
>> X=normrnd(0,1,8,1);
>> group=[1 2 1 2 1 2 1 2]; ['hi';'hi';'lo';'lo';'hi';'hi';'lo';'lo'];...
{'may' 'may' 'may' 'may' 'june' 'june' 'june' 'june'};
>> model='interaction';
>> sstype=2;
>> gnames={'Xx'; 'Yy'; 'Zz'};
>> [p,table,stats,terms]=anovan(X,group,model,sstype,gnames)
```

```

p =
0.2191
0.8722
0.2539
0.2315
0.2801
0.1122

table =
'Source' 'Sum Sq.' 'd.f.' 'Singular?' 'Mean Sq.' 'F' 'Prob>F'
'Xx' [ 0.4517] [1] [0] [0.4517] [ 7.7858] [0.2191]
'Yy' [ 0.0024] [1] [0] [0.0024] [ 0.0414] [0.8722]
'Zz' [ 0.3267] [1] [0] [0.3267] [ 5.6309] [0.2539]
'Xx*Yy' [ 0.4006] [1] [0] [0.4006] [ 6.9046] [0.2315]
'Xx*Zz' [ 0.2618] [1] [0] [0.2618] [ 4.5133] [0.2801]
'Yy*Zz' [ 1.8286] [1] [0] [1.8286] [31.5187] [0.1122]
'Error' [ 0.0580] [1] [0] [0.0580] [] []
'Total' [ 3.3298] [7] [0] [] [] []

stats =
source: 'anovan'
resid: [8x1 double]
coeffs: [19x1 double]
rtr: [7x7 double]
rowbasis: [7x19 double]
varnames: {3x1 cell}

dfe: 1
mse: 0.0580
nullproject: [19x7 double]
terms: [6x3 double]
nlevels: [3x1 double]
grpnames: {3x1 cell}

continuous: [0 0 0]
vmeans: [3x1 double]
termcols: [7x1 double]
coeffnames: {19x1 cell}
vars: [19x3 double]

terms =
1 0 0
0 1 0
0 0 1
1 1 0
1 0 1
0 1 1

```

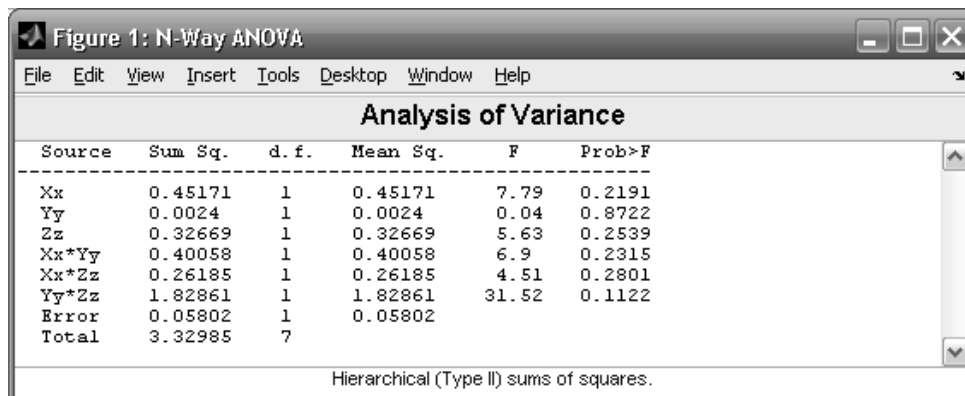


Рис. 12.19. Результати дисперсійного аналізу

Якщо подивитися на результати дисперсійного аналізу в наведених прикладах, то видно, що всі результати дають незначущі відповіді. Це логічно, тому що вибірки і фактори для умов задач формувалися випадково і не мали практичної основи.

7. У папці stats є файл carbig.mat, що містить реальну статистичну вибірку за 9-ма характеристиками 406 легкових автомобілів, виготовлених у США, Японії і Європі в період з 1970 року по 1982 за кожний рік. Серед них найбільше європейських авто (70). Фрагмент цього файла представлено в табл. 12.5.

## Фрагмент даних файла carbig.mat

Model	Origin	Year	Accel	Displ	Power	MPG	Weight	Cyl	cyl4	when	org
Audi	German	73	14.0	114	91	24.0	2582	4	Four	Early	E
Volvo	Sweden	73	15.5	121	112	20.0	2868	4	Four	Early	E
Dodge	USA	73	11.0	318	150	19.0	3399	8	Other	Early	A
Saab	Sweden	73	14.0	121	110	15.0	2660	4	Four	Early	E
Toyota	Japan	73	13.5	156	122	24.0	2807	6	Other	Early	J
oldsmobile	USA	73	11.0	350	180	20.0	3664	8	Other	Early	A
plymouth	USA	74	16.5	198	95	11.0	3102	6	Other	Mid	A
ford	USA	74	17.0	200	NAN	20.0	2875	6	Other	Mid	A
amc	USA	74	16.0	232	100	21.0	2901	6	Other	Mid	A
datsun	Japan	78	14.9	119	97	23.8	2405	4	Four	Mid	J
Audi	German	78	15.9	131	103	23.9	2830	5	Other	Mid	E
Volvo	Sweden	78	13.6	163	125	20.3	3140	6	Other	Mid	E
Saab	Sweden	78	15.7	121	115	17.0	2795	4	Four	Mid	E
peugeot	France	78	15.8	163	133	21.6	3410	6	Other	Mid	E
volkswagen	German	78	14.9	89	71	16.2	1990	4	Four	Mid	E
Honda	Japan	78	16.6	98	68	31.5	2135	4	Four	Mid	J
pontiac	USA	79	15.4	231	115	29.5	3245	6	Other	Late	A
mercury	USA	79	18.2	200	85	21.5	2990	6	Other	Late	A
Ford	USA	79	17.3	140	88	19.8	2890	4	Four	Late	A
amc	USA	79	18.2	232	90	22.3	3265	6	Other	Late	A

Ці дані мають наступні відмінності від даних файла carsmall.mat:

кількість автомобілів збільшена до 406 (проти 100);

дати виготовлення обрані підряд з 1970 року до 1982.

Для зручності проведення дисперсійного аналізу в матрицю даних уведено три стовпці, у яких фактори сформовані в такий спосіб:

стовпець *cyl4* містить дві категорії – чотири (*Four*) циліндри й інші (*Other*);

стовпець *org* містить три категорії – *Europe (E)*, *Japan (J)*, *USA (A)*;

стовпець *when* містить три категорії – *Early* (випуск 1970 – 1973 рр), *Mid* (випуск 1974 – 1978 рр) і *Late* (випуск 1978 – 1982 рр).

Структуру файла можна переглянути у вікні Workspace після його завантаження в командне вікно, як це показано на рис. 12.20.

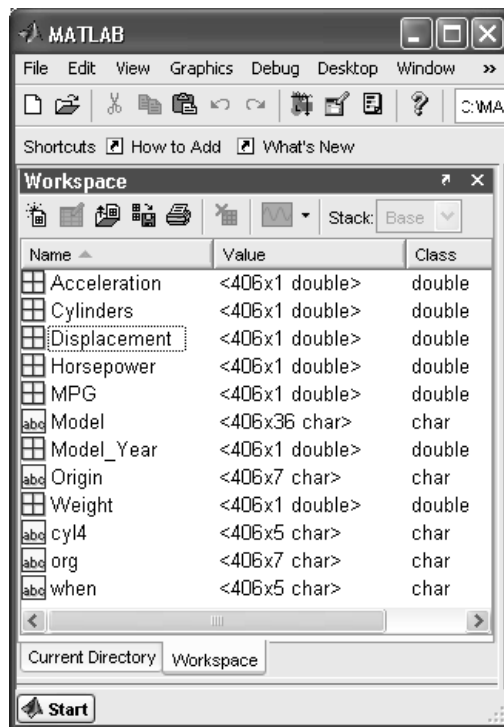


Рис. 12.20. Структура файла carbig.mat

У прикладі досліджується вплив регіону й року виготовлення автомобіля на його питомий пробіг. Для цього використовується база даних, одне з полів якої (*MPG*) – це досліджувана величина. Два інших поля (*org* і *when*) – це масиви символів, що вказують регіон й час виготовлення (рівні відповідних факторів). За викликом функції **anovan** вони поєднуються в один рядок. Задається модель, в якій враховуються обидва фактори і їх взаємодія. Тип обчислення впливу фактора береться за третім способом (цю пару "ім'я – значення" можна було вилучити). І, нарешті, задаються мітки для імен факторів.

```
>> load carbig % завантажуємо базу даних по автомобілях
>> p=anovan(MPG,{org when},'model',2,'sstype',3,'varnames',...
{'Місце складання';'Час складання'})
p =
```

```
0 % pval місця складання
0 % pval час складання
0.3059 % pval їх взаємодії
```

Результати аналізу наведено на рис. 12.21.

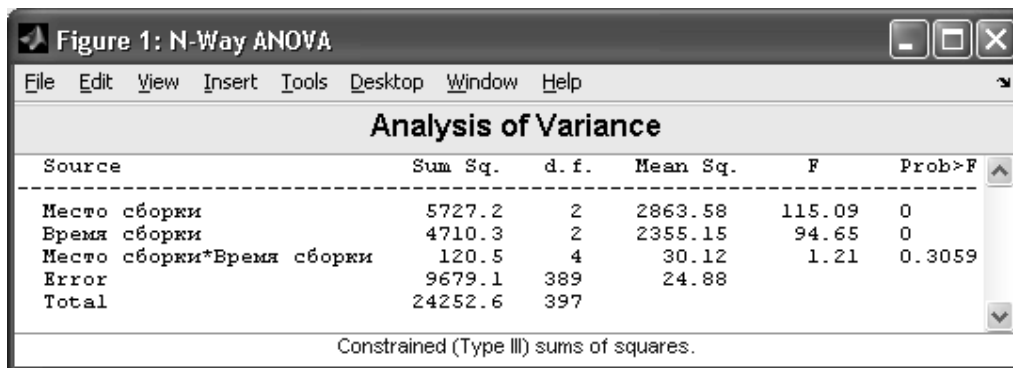


Рис. 12.21. Результати дисперсійного аналізу

Вибравши рівень значимості 0.01, можна зробити висновок про те, що й регіон, і рік виготовлення автомобілів суттєво впливають на питомий пробіг. У той же час ці фактори практично незалежні. Зауважимо також, що створене вікно є інтерактивним.

### **MULTCOMPARE** Перевірка параметричних гіпотез за парного порівняння групових середніх

Уточнюється рішення різних задач дисперсійного аналізу: провадиться попарне порівняння середніх. Таке порівняння не зводиться просто до порівняння всіх пар середніх за t-критерієм Стюдента. за кожного попарного порівняння враховується наявність інших середніх.

*Синтаксис:*

**C = multcompare(stats)**

**C = multcompare(stats,alpha)**

**C = multcompare(stats,alpha,'displayopt')**

**C = multcompare(stats,alpha,'displayopt','ctype')**

**C = multcompare(stats,alpha,'displayopt','ctype','estimate')**

**C = multcompare(stats,alpha,'displayopt','ctype','estimate',dim)**

**[C,m] = multcompare(...)**

**[C,m,h] = multcompare(...)**

*Опис:*

Функція **C=multcompare(stats)** призначена для перевірки параметричних гіпотез за парного порівняння середніх арифметичних або інших оцінок на основі інформації в структурі даних *stats*, яка має бути попередньо отримана в результаті рішення основної задачі

дисперсійного аналізу. Результативним параметром функції *multcompare* є матриця *C* з результатами перевірки параметричних гіпотез, тобто із результатами попарного порівняння середніх. Ця матриця має 5 стовпців і стільки рядків, скільки є пар середніх. В 1-му і 2-му елементах кожного рядка наведені номери порівнюваних змінних, в 4-му – оцінка різниці середніх, а в 3-му і 5-му – нижня й верхня границі довірчого інтервалу для різниці середніх за рівнем значимості 5%. Якщо довірчий інтервал не містить нуля, то на даному рівні значимості потрібно прийняти гіпотезу про значиму відмінність досліджуваних середніх. Також функція дозволяє побудувати інтерактивний графік за результатами перевірки множини параметричних гіпотез.

Під час проведення дисперсійного аналізу за множиною вибірок перевіряється нульова гіпотеза, що полягає в тому, що всі математичні очікування вибірок рівні між собою, вибірки витягнуті з однієї генеральної сукупності, або з декількох генеральних сукупностей але з однаковими значеннями математичних очікувань. Альтернативна гіпотеза полягає в тому, що значення хоча б одного вибіркового середнього арифметичного значимо відрізняється від інших.

Після проведення дисперсійного аналізу доцільно визначити, для якої пари вибірок середні арифметичні значення мають статистично значимі відмінності. Для перевірки такої параметричної гіпотези використовується процедура множинного порівняння.

В ході перевірки простої параметричної гіпотези (нульової гіпотези) про рівність середніх однієї групи вибірок стосовно іншої за статистикою Стюдента *t* необхідно задати рівень значимості  $\alpha_{кр}$ , що визначає критичне значення статистики  $t_{кр}$ . У більшості практичних випадків  $\alpha_{кр}$  ухвалюють рівним 0.05 або 0.01. Це означає, що в 5%, або в 1% випадків буде невірно відкинута нульова гіпотеза. За умови збільшення числа груп вибірок, збільшується число гіпотез, що перевіряються. Якщо використовується проста параметрична гіпотеза за статистикою *t*, рівень значимості  $\alpha_{кр}$  буде застосовуватися до кожної гіпотези окремо, що спричиняє до росту ймовірності невірно відкинути нульову гіпотезу пропорційно

кількості виконаних перевірок, тобто, невірно визначити значиму відмінність вибірових середніх. Процедура множинного порівняння забезпечує заданий рівень значимості для кожної перевірки.

Результативний параметр  $C$  представляє результати множинного порівняння у вигляді матриці з 5 стовпців. Рядок матриці  $C$  відповідає результатам перевірки однієї параметричної гіпотези. Таким чином, кожний рядок  $C$  відповідає одній парі вибірок. Перші два значення в рядку  $C$  показують номери порівнюваних вибірок, четвертий – величину різниці середніх арифметичних порівнюваних вибірок, третій і п'ятий стовпці – нижню і верхню границі 95%-го довірчого інтервалу отриманої різниці середніх арифметичних.

Наприклад, якщо рядок  $C$  містить наступні значення:

2.0000 5.0000 1.9442 8.2206 14.4971 ,

то вони показують, що порівнюються середні арифметичні значення 2-ї і 5-ї вибірок, величина їх різниці дорівнює 8.2206, а 95%-й довірчий інтервал отриманої різниці середніх арифметичних склав [1.9442, 14.4971]. Оскільки в довірчий інтервал не потрапило нульове значення, різниця середніх арифметичних 2-ї і 5-ї вибірок значуща, тобто середні арифметичні цих вибірок статистично значущо відрізняються між собою для  $\alpha_{кр} = 0.05$ .

Функція **multcompare** дозволяє графічно відобразити значення середніх арифметичних і їх довірчих інтервалів. Два вибірових середніх значимо відрізняються, якщо їх довірчі інтервали не перекриваються на графіку. За умови перекривання границь довірчих інтервалів двох середніх арифметичних відмінність між ними можна вважати статистично незначимою. За графіком можна визначити вибірки із середніми арифметичними, які значимо відрізняються від вибірового середнього, що нас цікавить. Для цього необхідно мишкою виділити графік бажаного вибірового середнього і відповідні графіки будуть виділені іншим кольором.

Функція **C=multcompare(stats,alpha)** у другому аргументі дозволяє задати рівень значимості  $\alpha$  для розрахунку довірчих інтервалів на

різниці середніх арифметичних, представлених у матриці  $C$ . Довірча ймовірність визначається як  $100*(1 - \alpha)\%$ . Значення за замовчуванням  $\alpha = 0.05$ .

Функція **C=multcompare(stats,alpha,'displayopt')** з аргументом *'displayopt'* дозволяє відобразити графік з результатами множинного порівняння за *'displayopt'='on'*, або не виводити цей графік за *'displayopt'='off'*. Значення за замовчуванням *'displayopt'='on'*.

Функція **C=multcompare(stats,alpha,'displayopt','ctype')** в аргументі *'ctype'* дозволяє задати спосіб визначення критичних значень статистик за перевірки нульової гіпотези. В табл. 12.6 наведені значення *'ctype'*.

Таблиця 12.6

### Можливі значення аргументу *'ctype'*

Значення <i>'ctype'</i>	Спосіб визначення критичних значень статистик
<i>'tukey-kramer'</i>	Використовується різницевий критерій Тьюки (Tukey). Значення за замовчуванням. Критерій заснований на розподілі Стюдента. Цей критерій – найкращий для збалансованого однофакторного дисперсійного аналізу й інших тестів з однаковими об'ємами вибірок. Доведено також, що цей критерій прийнятний і для дисперсійного аналізу з різними об'ємами вибірок. Згідно з недоведеним припущенням Тьюки – Крамера (Kramer), цей критерій також можна застосовувати й для корельованих вибірок, наприклад, у задачі дисперсійного аналізу з незбалансованими корельованими вибірками
<i>'lsd'</i>	Використовується процедура визначення критичних значень статистики за найменш значущою різницею критерію Тьюки. Ця процедура заснована на використанні простого t-тесту. Вона прийнятна, якщо попередній тест (мається на увазі F-статистика за однофакторного дисперсійного аналізу) показала значущу відмінність порівнюваних середніх
<i>'bonferroni'</i>	Заснований на розрахунку критичних значень розподілу Стюдента з коректуванням Бонфероні для використання в процедурі множинного порівняння
<i>'dunn-sidak'</i>	Використовуються критичні значення, розраховані за розподілом Стюдента після коректування для процедури множинного порівняння, запропонованої Даном
<i>'scheffe'</i>	Використовуються критичні значення, розраховані за процедурою Шеффе, заснованої на розподілі Фішера. Ця процедура дозволяє забезпечити однаковий рівень значимості за порівняння лінійних комбінацій середніх

Функція **C=multcompare(stats,alpha,'displayopt','ctype','estimate')** в аргументі *'estimate'* дозволяє задати вид оцінок для процедури множинного порівняння. Можливі види оцінок визначаються функцією, використаною для розрахунку структури *stats*, і наведені в табл. 12.7.

Таблиця 12.7

**Види порівнюваних оцінок**

Функція	Значення <i>'estimate'</i>
<i>'anova1'</i>	Значення <i>'estimate'</i> ігноруються. Завжди порівнюються групові середні
<i>'anova2'</i>	<i>'estimate' = 'column'</i> (значення за замовчуванням) – для порівняння середніх за стовпцями; <i>'estimate' = 'row'</i> – для порівняння середніх за рядками
<i>'anovan'</i>	Значення <i>'estimate'</i> ігноруються. Завжди порівнюються маргінальні середні генеральної сукупності, обумовлені згідно з аргументом <i>dim</i> .
<i>'aoctool'</i>	<i>'estimate' = 'slope', 'intercept', 'pmm'</i> для порівняння значень початкового зсуву, коефіцієнтів за першого ступеня незалежної змінної, маргінальних середніх генеральної сукупності. Якщо модель кореляційного аналізу не включає розділені постійні зсуви, тоді використовувати значення <i>'estimate' = 'slope'</i> заборонене. Також неможливо застосувати значення <i>'estimate' = 'intercept'</i> , якщо немає розділених коефіцієнтів за лінійного ступеня незалежної змінної моделі
<i>'friedman'</i>	Значення ігноруються. Завжди порівнюються середні за стовпцями
<i>'kruskalwallis'</i>	Значення ігноруються. Завжди порівнюються середні за групами

**C=multcompare(stats,alpha,'displayopt','ctype','estimate',dim)** – у цьому синтаксисі аргумент *dim* задає маргінальні (окремі) середні розподілів, які треба порівнювати. Цей аргумент необхідний, якщо за розрахунку *stats* використовувалася функція **anovan**. Під час проведення багатofакторного дисперсійного аналізу з *n* факторами, *dim* може бути заданий як скаляр або вектор цілих чисел у діапазоні від 1 до *n*. Значення за замовчуванням *dim = 1*.

Наприклад, якщо *dim = 1*, то порівнянні точкові оцінки є середніми арифметичними груп, обумовлених значеннями першої класифікаційної змінної, скоректованих видаленням ефектів інших класифікаційних змінних, так, якби план був збалансований, тобто задані вибірки однакового об'єму. Якщо *dim = [1 3]*, то маргінальні середні розподілів обчислюються для кожної комбінації першої й третьої класифікаційних

змінних з ігноруванням ефектів інших факторів. Якщо використовується сингулярна (вироджена) модель, середні арифметичні за деякими групами не можуть бути розраховані й відповідні їм маргінальні середні приймуть значення NaN.

Функція **[C,m]=multcompare(...)** повертає матрицю  $m$  точкових й інтервальних оцінок середніх арифметичних значень, або інших порівнюваних оцінок. Перший стовпець матриці  $m$  містить точкові оцінки середніх арифметичних, або порівнюваних статистик, для кожної групи. Другий стовпець  $m$  містить величини стандартних похибок порівнюваних оцінок.

Функція **[C,m,h]=multcompare(...)** повертає покажчик  $h$  на графік, сформований у результаті множинного порівняння. Слід зазначити, що в рядку заголовка графіка виводиться інструкції по роботі з інтерактивним графіком, підписи по осі абсцис містять список груп, у яких середні арифметичні значимо відрізняються від середнього арифметичного для виділеного графіка. Для видалення заголовка графіка й підпису по осі X необхідно використовувати інтерактивні інструменти графічного вікна або команди:

```
>> title("")
>> xlabel("")
```

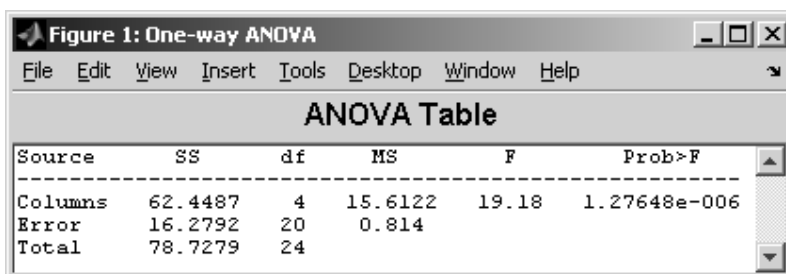
*Приклади:*

1. Однофакторний дисперсійний аналіз для п'яти вибірок з математичними очікуваннями, що змінюються в межах від 1 до 5, з наступним проведенням множинного порівняння оцінок середніх арифметичних

```
>> X=meshgrid(1:5);
>> X= X+normrnd(0,1,5,5);
>> group=['Aa'; 'Bb'; 'Cc'; 'Dd'; 'Ee'];
>> [p,table,stats]=anova1(X, group);
>> c=multcompare(stats)
```

```
c =
 1.0000  2.0000 -3.8426 -2.1352 -0.4277    % різниця значима
 1.0000  3.0000 -4.6733 -2.9658 -1.2584    % різниця значима
 1.0000  4.0000 -5.3362 -3.6287 -1.9213    % різниця значима
 1.0000  5.0000 -6.3946 -4.6872 -2.9797    % різниця значима
 2.0000  3.0000 -2.5381 -0.8306  0.8768
 2.0000  4.0000 -3.2010 -1.4935  0.2139
 2.0000  5.0000 -4.2594 -2.5520 -0.8445    % різниця значима
 3.0000  4.0000 -2.3703 -0.6629  1.0446
 3.0000  5.0000 -3.4288 -1.7214 -0.0139    % різниця значима
 4.0000  5.0000 -2.7659 -1.0585  0.6490
```

Результати аналізу представлені в числовій формі, у графічній (рис. 12.22) і інтерактивній (рис. 12.23). Пари вибірок, довірчі інтервали яких не містять нуля, є значимими.



Source	SS	df	MS	F	Prob>F
Columns	62.4487	4	15.6122	19.18	1.27648e-006
Error	16.2792	20	0.814		
Total	78.7279	24			

Рис. 12.22. Результати дисперсійного аналізу

На рис. 12.22 представлена загальна значимість ( $pval = 1.27648 \times 10^{-6}$ ), що менша за 0.01 (тобто між групами є значущі відмінності).

Рис. 12.23 дозволяє побачити загальну картину розшарування вибірок на однорідні групи й області, у яких перекриваються довірчі інтервали. Виявляється, що перша група (*Aa*) значимо відрізняється від усіх інших.

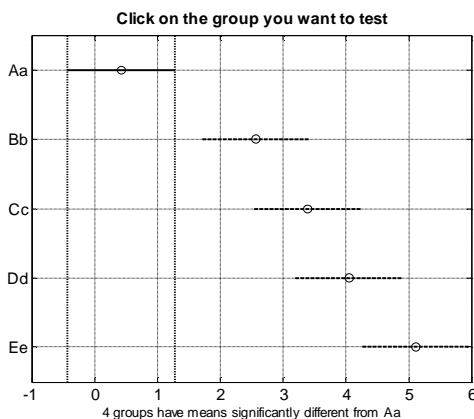


Рис. 12.23. Інтерактивне вікно результатів аналізу

Далі середні повільно збільшуються з номером групи, причому середні для двох сусідніх груп мало відрізняються. Так, друга група (*Bb*) ще мало відрізняється від сусідньої третьої групи (*Cc*), але вже значимо відрізняється від п'ятої групи (*Ee*). Третя група (*Cc*) значимо відрізняється від п'ятої групи (*Ee*).

Вікно на рис. 12.23 є інтерактивним. Якщо клацнути мішкою по довірчому інтервалу будь-якої групи, група буде виділена кольором, і знизу вікна з'явиться результат порівняння обраної групи зі всіма іншими. Зараз на рис. 12.23 виділена група *Aa*, і наведено висновок, що 4 групи мають значимі відмінності від групи *Aa*.

2. Двофакторний дисперсійний аналіз для шести рівнів першого фактора, двох рівнів другого фактора, 3-х повторень з наступним проведенням множинного порівняння оцінок середніх арифметичних з 99%-ми довірчими інтервалами ( $\alpha_{кр} = 0.01$ ).

```
>> X = normrnd(0,1,6,6);
>> [p,table,stats] = anova2(X,3);
>> alpha = 0.01;
>> c = multcompare(stats,alpha)
```

Note: Your model includes an interaction term. A test of main effects can be difficult to interpret when the model includes interactions.

```
c =
 1.0000  2.0000 -2.9346 -1.0198  0.8951
 1.0000  3.0000 -1.4097  0.5051  2.4200
 1.0000  4.0000 -1.7510  0.1638  2.0787
 1.0000  5.0000 -2.4294 -0.5146  1.4002
 1.0000  6.0000 -1.8167  0.0981  2.0129
 2.0000  3.0000 -0.3899  1.5249  3.4397
 2.0000  4.0000 -0.7312  1.1836  3.0984
 2.0000  5.0000 -1.4096  0.5052  2.4200
 2.0000  6.0000 -0.7970  1.1179  3.0327
 3.0000  4.0000 -2.2561 -0.3413  1.5735
 3.0000  5.0000 -2.9346 -1.0197  0.8951
 3.0000  6.0000 -2.3219 -0.4070  1.5078
 4.0000  5.0000 -2.5933 -0.6784  1.2364
 4.0000  6.0000 -1.9806 -0.0657  1.8491
 5.0000  6.0000 -1.3021  0.6127  2.5275
```

У висновках аналізу виведене зауваження: "Ваша модель включає член взаємодії. При цьому може бути утруднена інтерпретація результатів перевірки значимості головних ефектів".

Дійсно, досить складно за числовими результатами без міток визначати взаємодіючі пари. Досить просто це зробити на рис. 12.24, на якому зображені результати аналізу в інтерактивному режимі. Оскільки всі довірчі інтервали перекриваються, то немає груп, представлених стовпцями матриці зі значимими відмінностями із групою 1, яка виділена для порівняння суцільним відрізком.

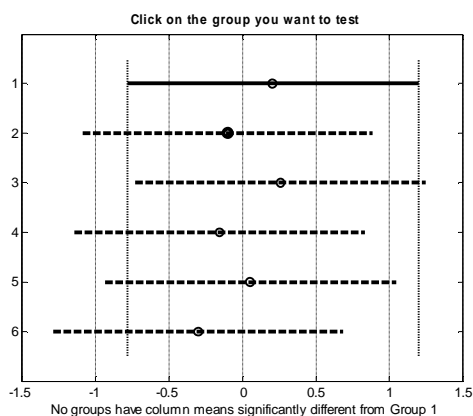


Рис. 12.24. Інтерактивне вікно результатів аналізу

3. Однофакторний дисперсійний аналіз для п'яти вибірок з математичними очікуваннями, що змінюються в межах від 1 до 5, з наступним проведенням множинного порівняння оцінок середніх арифметичних з 99%-ми довірчими інтервалами ( $\alpha = 0.01$ ).

```
>> X=meshgrid(1:5)
X =
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
>> X=X+normrnd(0,1,5,5);
>> group=['Aa'; 'Bb'; 'Cc'; 'Dd';
'Ee'];
>> [p,table,stats]=anova1(X,
group);
>> alpha=0.01;
>> displayopt='on';
>> ctype='lsd';
>>
c=multcompare(stats,alpha,displayopt,ctype)
c =
     1.0000     2.0000    -2.8283    -1.4736    -0.1190
     1.0000     3.0000    -2.7817    -1.4270    -0.0723
     1.0000     4.0000    -4.7391    -3.3844    -2.0297
     1.0000     5.0000    -5.5488    -4.1942    -2.8395
     2.0000     3.0000    -1.3080     0.0467     1.4013
     2.0000     4.0000    -3.2655    -1.9108    -0.5561
     2.0000     5.0000    -4.0752    -2.7205    -1.3659
     3.0000     4.0000    -3.3121    -1.9574    -0.6028
     3.0000     5.0000    -4.1219    -2.7672    -1.4125
     4.0000     5.0000    -2.1644    -0.8097     0.5449
```

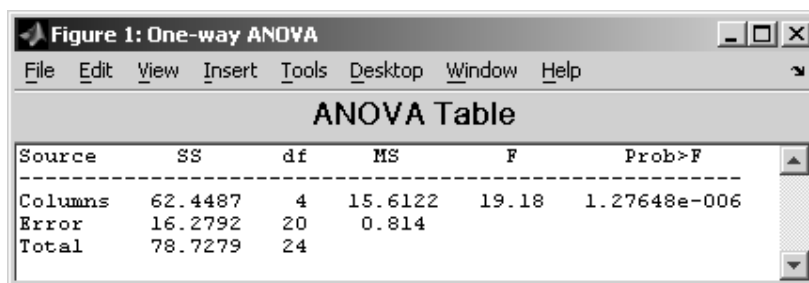


Рис. 12.25. Результати дисперсійного аналізу

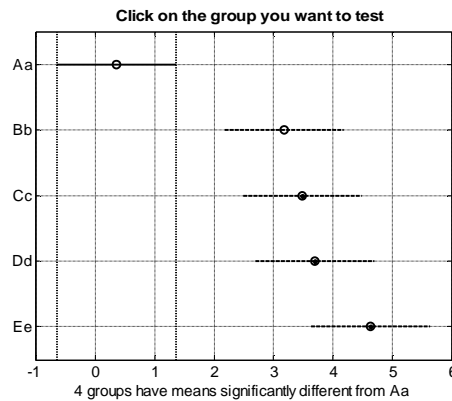


Рис. 12.26. Інтерактивне вікно результатів аналізу

На рис. 12.25 і 12.26 представлені результати аналізу. З них видна загальна значимість (рис. 12.25) і значимість відмінності 1-ої групи (*Aa*) від 3-ої, 4-ої і 5-ої. В інтерактивному вікні це видно х вертикальних штрихових ліній і висновку знизу інтерактивного вікна англійською мовою.

Під час розрахунку критичного рівня статистики використовувалася процедура найменшої значущої різниці (НСР – '*lsd*') за критерієм Тьюки.

4. Виконується багатофакторний дисперсійний аналіз для 3-х факторів з двома рівнями, з наступним множинним порівнянням і з 99%-ми довірчими інтервалами ( $\alpha = 0.01$ ). При розрахунку критичного рівня статистики використовується процедура Шеффе. Маргінальні (окремі) середні розподілів, що підлягають порівнянню, задані змінною  $dim = [1 \ 3]$ . Маргінальні середні розподілів обчислюються для кожної комбінації першого й третього факторів, видаляючи ефекти другого фактора.

```
>> X=normrnd(0,1,8,1);
>> group={[1 2 1 2 1 2 1 2]; ['hi';'hi';'lo';'lo';'hi';'hi';'lo';'lo']; {'may' 'may' 'may' 'may'
'june' 'june' 'june' 'june'}};
>> [p,table,stats,terms]=anovan(X,group);
>> alpha=0.01;
>> displayopt='on';
>> ctype='scheffe';
>> estimate=1;
>> dim=[1 3];
>> c=multcompare(stats,alpha,displayopt,ctype,estimate,dim)
c =
    1.0000    2.0000   -4.0683    0.2935    4.6554
    1.0000    3.0000   -4.3823   -0.0205    4.3414
```

1.0000	4.0000	-5.8955	0.2731	6.4416
2.0000	3.0000	-6.4825	-0.3140	5.8546
2.0000	4.0000	-4.3823	-0.0205	4.3414
3.0000	4.0000	-4.0683	0.2935	4.6554

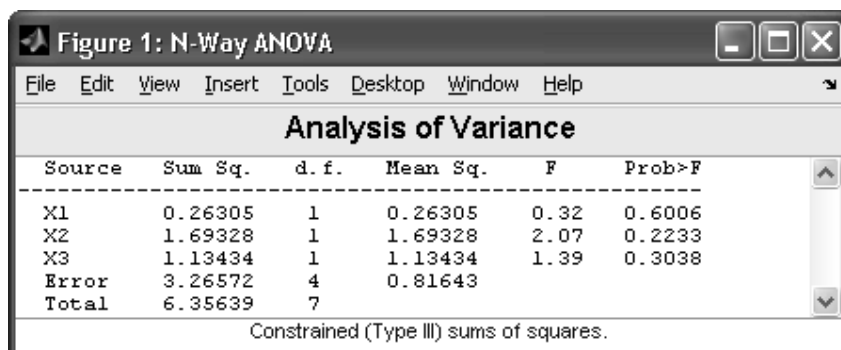


Рис. 12.27. Результати дисперсійного аналізу

На рис. 12.27 і 10.28 представлені результати аналізу в графічному вікні **anova** і в інтерактивному вікні. Значення *pval* для всіх груп виявилися більшими за 0.05, тобто немає значимих різниць між групами.

В інтерактивному вікні (рис. 12.28) видно, що довірчі інтервали для усіх груп перекриваються, що свідчить про незначимість різниць між групами. Виділена перша група ( $X_1 = 1$ ,  $X_3 = \text{may}$ ), для якої зроблено висновок про відсутність інших груп зі значимими різницями від першої.

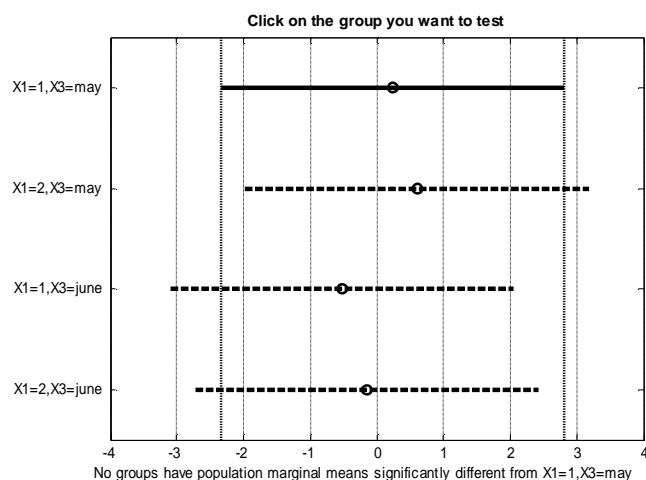


Рис. 12.28. Інтерактивне вікно результатів аналізу

5. Однофакторний дисперсійний аналіз для п'яти вибірок з математичними очікуваннями, що змінюються в межах від 1 до 5, з

наступним проведенням множинного порівняння середніх. Функція повертає матрицю  $m$  точкових і інтервальних оцінок середніх арифметичних значень і показчик (дескриптор)  $h = 2$  на графік множинного порівняння.

```
>> X=meshgrid(1:5);
>> X=X + normrnd(0,1,5,5);
>> group=['Aa'; 'Bb'; 'Cc'; 'Dd'; 'Ee'];
>> [p,table,stats]=anova1(X, group);
>> [c,m,h]=multcompare(stats)
```

```
c =
1.0000 2.0000 -2.0501 -0.3580 1.3341
1.0000 3.0000 -3.4140 -1.7219 -0.0298
1.0000 4.0000 -4.4701 -2.7780 -1.0859
1.0000 5.0000 -4.8882 -3.1961 -1.5040
2.0000 3.0000 -3.0561 -1.3640 0.3281
2.0000 4.0000 -4.1121 -2.4200 -0.7279
```

```

2.0000 5.0000 -4.5302 -2.8381 -1.1460
3.0000 4.0000 -2.7481 -1.0560 0.6361
3.0000 5.0000 -3.1663 -1.4742 0.2179
4.0000 5.0000 -2.1102 -0.4181 1.2740
m =
1.0044 0.3998
1.3624 0.3998
2.7264 0.3998
3.7824 0.3998
4.2005 0.3998
h =
2
```

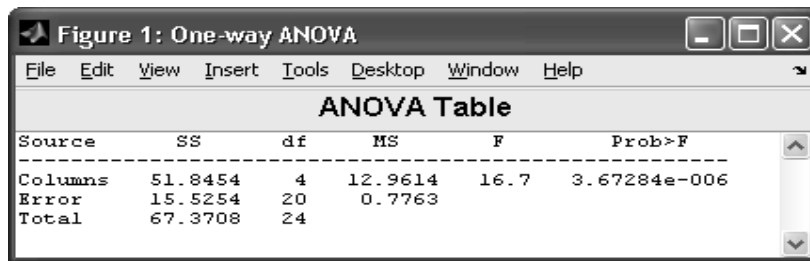


Рис. 12.29. Результати дисперсійного аналізу

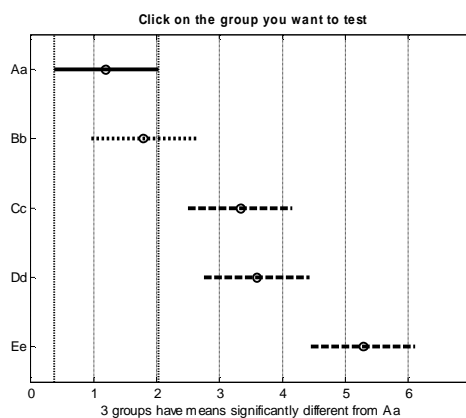


Рис. 12.30. Інтерактивне вікно результатів аналізу

З рис. 12.29 і 10.30 можна зробити висновок про загальну значимість і про те, що дві групи ( $Dd$  і  $Ee$ ) значимо відрізняються від

першої (*Aa*). З рис. 12.30 можна одержати більш детальні результати аналізу.

6. Наведемо реальний приклад. У прикладі порівнюються твердості трьох сплавів – сталевого й двох алюмінієвих, які позначені *st*, *al<sub>1</sub>*, *al<sub>2</sub>*. Для порівняння використовується однофакторний дисперсійний аналіз з подальшим попарним порівнянням.

```
>> q=0.05; % рівень значимості
>> strength=[82 86 79 83 84 85 86 87 74 82 78 75 76 77 79 79 77 78 82 79];
      % твердість
>> alloy={'st','st','st','st','st','st','st','st','st','al1','al1','al1','al1','al1',...
'al2','al2','al2','al2','al2','al2'}; % сплави
>> [p,a,s]=anova1(strength,alloy,'off'); % одержали структуру s
>> c=multcompare(s,q); % множинне порівняння
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bфОберіть мишкою групу для тесту') % заголовок
>> ylabel('Сплав')
>> disp('Результати порівняння груп');
>> disp('Сплави Нижня межа Середнє Верхня межа');
>> fprintf(' %1.0f-%1.0f %14.8f %7.2f %14.8f\n',c');
```

Результати порівняння груп

Сплави	Нижня межа	Середнє	Верхня межа
1-2	3.60635148	7.00	10.39364852
1-3	1.60635148	5.00	8.39364852
2-3	5.62796287	2.00	1.62796287

У виведеній в командне вікно інформації відзначені порівнювані сплави, середні значення й нижня і верхня границі довірчих інтервалів. На рис. 12.31 аналіз, що наведений у графічному вікні, створений функцією **multcompare**. Довірчі інтервали для середніх груп *al<sub>1</sub>* і *al<sub>2</sub>* перекриваються, а середнє *st* значно від них відрізняється.

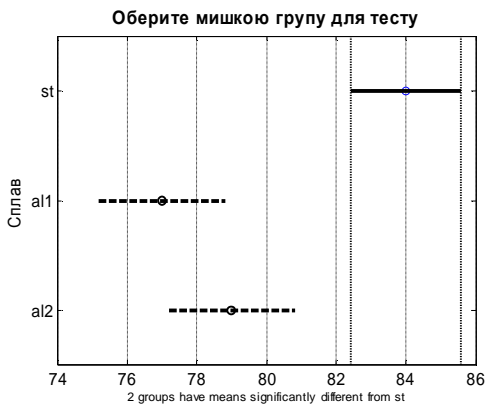


Рис. 12.31. Інтерактивне вікно результатів аналізу

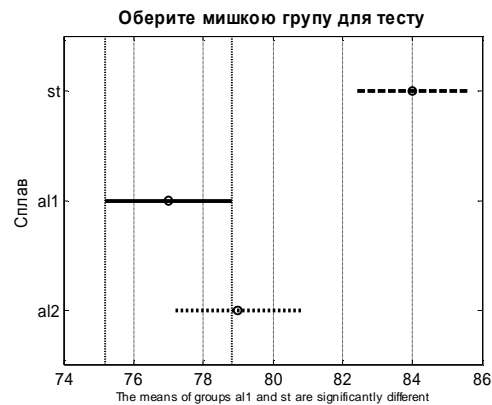


Рис. 12.32. Те ж вікно після натискання курсором на групу *al<sub>1</sub>*

На рис. 12.32 наведено результат інтерактивної дії, що полягає в тому, що мишкою клацнули по групі  $al_1$ . Функція все перерахувала й вивела висновок: група  $al_1$  значимо відрізняється від  $st$  і не відрізняється від групи  $al_2$ . Під рис. 12.32 виведено напис: середні груп  $al_1$  і  $st$  значимо розрізняються.

### **KRUSKALWALLIS** Непараметричний тест Краскала-Уоліса

Вирішується задача однофакторного дисперсійного аналізу за допомогою непараметричного тесту Краскала – Уоліса. Даний тест використовує ранговий підхід і є більш робастним (стійким) стосовно похибок у даних, ніж звичайний однофакторний дисперсійний аналіз.

*Синтаксис:*

```
p=kruskalwallis(X)  
p = kruskalwallis(X,group)  
p = kruskalwallis(X,group,displayopt)  
[p,table] = kruskalwallis(...)  
[p,table,stats] = kruskalwallis(...)
```

*Опис:*

Функція **p=kruskalwallis(X)** провадить тест Краскала – Уоліса для порівняння двох або більш вибірок. Кожний стовпець матриці  $X$  розміром  $m \times n$  являє собою незалежну вибірку, яка містить  $m$  взаємно-незалежних спостережень. У тесті порівнюються медіани вибірок і повертається  $p$ -значення для перевірки 0-гіпотези про те, що всі вибірки взяті з однієї генеральної сукупності (або з різних генеральних сукупностей з однаковими розподілами, що те саме). Якщо  $p$ -значення близьке до нуля (наприклад, менше заданого рівня значимості), то 0-гіпотезу потрібно відкинути й прийняти альтернативну гіпотезу про те, що медіана принаймні однієї вибірки значимо відрізняється від інших.

Функція **kruskalwallis** за замовчуванням будує дві фігури, які показані нижче у прикладі. Перша з них – це стандартна таблиця дисперсійного аналізу, де усі відмінності в даних підрозділяються на дві групи:

розкид між медіанами стовпців (відмінності між групами);

розкид даних у кожному стовпці (відмінності усередині груп).

Таблиця дисперсійного аналізу складається з 6 стовпців:

Джерело мінливості даних (*Source*). Таких джерел два: розкид між групами (*Groups*) і розкид даних у кожному стовпці (*Error*). Крім того, видається також сумарний розкид (*Total*).

Сума квадратів відхилень (*Sum of Squares*, скорочено *SS*).

Число ступенів свободи (*the degrees of freedom*, або *df*).

Середні квадратів (*Mean of Squares*, або *MS*), які підраховуються як відношення  $MS = SS / df$ .

$\chi^2$ – статистика Пірсона (*Chi-sq*), вона є відношенням величин *MS* до *MSE*.

Величина *p*-значення, що обчислена за допомогою функції розподілу Пірсона. Чим більше  $\chi^2$ -статистика, тим менше *p*-значення.

На другій фігурі зображуються "вусаті ящики Тьюки" (**boxplot**) для даних кожного стовпця. Велика відмінність між центральними лініями (тонкі паски на кожному прямокутнику) відповідає великому значенню  $\chi^2$ -статистики й малому *p*-значенню.

Функція **p = kruskalwallis(X,group)** використовує значення елементів *group* (масив символів або масив символічних рядків) як мітки для коробок Тьюки, якщо *X* є матрицею. Кожний рядок *group* містить мітку для даних відповідного стовпця *X*, тому довжина *group* повинна дорівнювати кількості стовпців *X*.

Якщо *X* є вектором, функція **kruskalwallis** провадить однофакторний дисперсійний аналіз, розділяючи елементи *X* на окремі вибірки з використанням *group* як індексного масиву. У цьому випадку аргумент **group** повинен бути вектором, масивом символів або масивом символічних рядків, що містять мітки груп. Довжина *group* повинна бути такою ж, як і *X*. Елементи *X* попадають в одну вибірку, якщо відповідні

елементи *group* однакові. Значення елементів *group* використовуються також як мітки для коробок Тьюки.

Мітки вибірок не обов'язково повинні нумеруватися послідовно (1, 2, 3, ...). Якщо, наприклад, *X* містить виміри за трьома різними температурами:  $-27^{\circ}$ ,  $65^{\circ}$ ,  $110^{\circ}$ , то ці числа можна використовувати як мітки вибірок в *group*. Якщо рядок в *group* містить порожній елемент або порожній рядок, то спостереження у відповідному рядку *X* відкидається. Нечислові значення NaN у будь-якому вхідному аргументі також ігноруються.

Функція **p = kruskalwallis(X,group,displayopt)** у третьому вхідному аргументі *displayopt* дозволяє управляти створенням фігур. За *displayopt = 'on'* фігури будуються, а за *displayopt = 'off'* – ні. За замовчуванням прийняте *displayopt = 'on'*.

Функція **[p,table] = kruskalwallis(...)** у другому результативному параметрі *table* повертає у командне вікно стандартну таблицю дисперсійного аналізу (дублюючи першу фігуру). Текстовий зміст цієї таблиці може бути скопійований в буфер обміну за допомогою меню Edit / Copy Text (Редагувати / Копіювати текст).

Функція **[p,table,stats] = kruskalwallis(...)** у третьому результативному параметрі *stats* повертає структуру, яку надалі можна використовувати для більш детального порівняння вибірок. Функція **kruskalwallis** перевіряє 0-гіпотезу про рівність усіх медіан; альтернативною є гіпотеза про відмінність вибірок. Проте іноді потрібно більш детальне дослідження: які пари вибірок – порівнянні, а які ні. Для цього можна використовувати функцію **multcompare** (див. вище), на вхід якої й потрібно подати цю структуру *stats*.

Тест Краскала – Уоліса використовує наступні припущення про дані. Усі вибірки беруться з генеральних сукупностей з однаковим безперервним розподілом з можливою відмінністю в параметрі зсуву. Усі спостереження повинні бути взаємо-незалежними. Класичний однофакторний дисперсійний аналіз вимагає більш обтяжливих обмежень: усі вибірки повинні ще бути нормальними.

У прикладі порівнюються твердості трьох сплавів, які позначені  $st$ ,  $al_1$  і  $al_2$ . Для порівняння використовуються два тести: класичний однофакторний дисперсійний аналіз і непараметричний тест Краскала – Уоліса. Поки немає викидів (великих похибок вимірів), обидва тесту дають однакові результати. Але, якщо внести в дані викиди, то класичний однофакторний дисперсійний аналіз буде давати помилковий результат, а непараметричний тест Краскала – Уоліса – правильний.

```
>> q=0.05; % рівень значимості
>> strength=[82 86 79 83 84 85 86 87 74 82 78 75 76 77 79 79 77 78 82 79];
% твердість, усього 20 спостережень
>> alloy={'st','st','st','st','st','st','st','st','st','al1','al1','al1','al1','al1','al1','al2','al2','al2','al2',
'al2','al2'}; % три сплави
>> ppar=anova1(strength,alloy,'off') % фігури не виводити
>> pnonpar=kruskalwallis(strength,alloy,'off') % фігури не виводити
>> if ppar<=q,
    disp('Параметричний аналіз: середні різні.');
```

```
else,
    disp('Параметричний аналіз: середні однакові.');
```

```
end
>> if pnonpar<=q,
    disp('Непараметричний аналіз: середні різні.');
```

```
else,
    disp('Непараметричний аналіз: середні однакові.');
```

```
end
>> strength(20)=120; % змінили одне з даних на викид
>> ppar=anova1(strength,alloy,'off') % фігури не виводити
>> pnonpar=kruskalwallis(strength,alloy,'on') % виводити фігури
>> if ppar<=q,
    disp('Параметричний аналіз: середні різні.');
```

```
else,
    disp('Параметричний аналіз: середні однакові.');
```

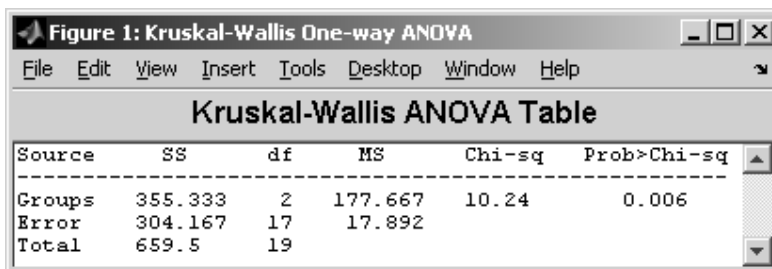
```
end
>> if pnonpar<=q,
    disp('Непараметричний аналіз: середні різні.');
```

```
else,
    disp('Непараметричний аналіз: середні однакові.');
```

```
end
ppar =
    1.5264e-004
pnonpar =
    0.0018
Параметричний аналіз: середні різні.
Непараметричний аналіз: середні різні.
ppar =
    0.2501
pnonpar =
    0.0060
```

Параметричний аналіз: середні однакові.  
Непараметричний аналіз: середні різні.

З таблиці дисперсійного аналізу (рис. 12.33) також виходить висновок про значимі різниці між групами ( $Prob = 0,006 < 0,01$ ).



Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Groups	355.333	2	177.667	10.24	0.006
Error	304.167	17	17.892		
Total	659.5	19			

Рис. 12.33. Результати дисперсійного аналізу

Діаграми Тьюки (рис. 12.34) демонструють причину цих різниць – це наявність викиду.

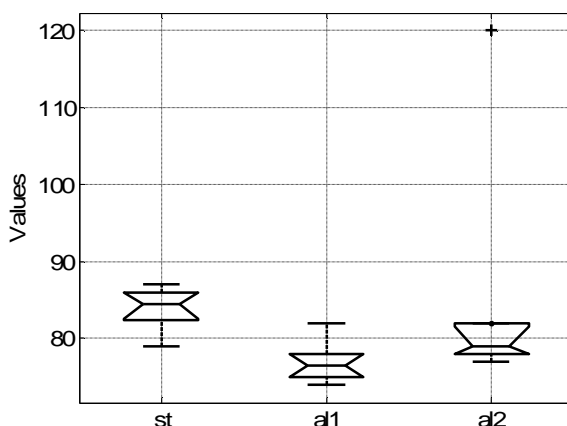


Рис. 12.34. "Коробки" Тьюки з викидом

Видно, що є помилкове дане – викид (плюстик над третьою графічною коробкою Тьюки), але це не приводить до зсуву всієї коробки, тобто висновки про відмінності між твердостями сплавів не зіпсуються.

### **FRIDMAN** Тест Фрідмана

Аналогічно тесту Краскала – Уоліса, непараметричний тест Фрідмана використовує ранговий підхід і тому він є більш робастним стосовно похибок у даних, ніж звичайний однофакторний дисперсійний аналіз. Традиційно тест Фрідмана називається "двофакторним", хоча в

ньому вивчається вплив лише однієї класифікаційної змінної – порівнюються середні стовпців матриці  $X$  (окремих вибірок спостережень). Але з обчислювальної точки зору, тест Фрідмана дійсно схожий на збалансований двофакторний дисперсійний аналіз, оскільки в ньому порівнюються середні стовпців матриці  $X$  після виключення можливого впливу рядків. Як однофакторний аналіз тест не перевіряє розкид середніх за рядками і взаємодію факторів стовпці – рядки. Цей тест застосовується, коли стовпці відповідають рівням досліджуваного фактора, а рядки – сторонніх факторів, вплив яких потрібно відсіяти.

*Синтаксис*

```
p = friedman(X, reps)
p = friedman(X, reps, displayopt)
[p, table] = friedman(...)
[p, table, stats] = friedman(...)
```

*Опис:*

Функція **p = friedman(X, reps)** – проводить непараметричний тест Фрідмана в порівнянні середніх стовпців матриці  $X$  з попередньою нейтралізацією рядків.

Різні стовпці матриці  $X$  – це різні рівні досліджуваного фактора  $A$ , а рядки – рівні фактора  $B$ , що має бути нейтралізованим. Якщо є більше одного виміру при кожній комбінації факторів, то в змінній *reps* вказується кількість повторень (ціле позитивне число, однакове для всіх комбінацій (за замовчуванням *reps* = 1). За *reps* > 1 матриця даних має бути заповнена як на рис. 12.35, де кожному рівню фактора  $B$  відповідає *reps* послідовних рядків.

Тест Фрідмана повертає *p*-значення для перевірки 0-гіпотези про рівність середніх стовпців. Мала величина *p*-значення (що менше прийнятого рівня значущості) вимагає відкинути 0-гіпотезу.

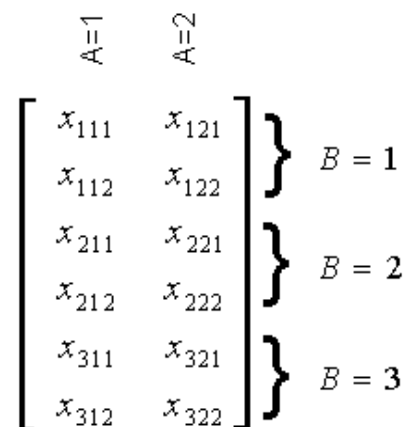


Рис. 12.35. Матриця даних для двофакторного

Функція **friedman** за замовчуванням виводить стандартну таблицю дисперсійного аналізу. Вона показана нижче у прикладі. Усі відмінності в даних розділяються на дві або три групи залежно від відсутності або наявності повторних спостережень:

розкид між середніми стовпців (вплив фактору *A*);

розкид через взаємодію факторів (якщо *reps* > 1);

випадковий розкид.

Таблиця дисперсійного аналізу складається з 6 стовпців.

Джерело розкиду даних (*Source*). Таких джерел два або три: розкид між середніми стовпців (*Columns*), розкид через взаємодію рядків і стовпців (*Interaction*), якщо *reps* > 1, і випадковий розкид (*Errors*).

Крім того, приводиться також сумарний розкид (*Total*).

Сума квадратів відхилень (*Sum of Squares*, скорочено *SS*).

Число ступенів свободи (*the degrees of freedom*, або *df*).

Середні квадрати (*Mean of Squares*, або *MS*) – підраховуються як відношення *SS* / *df*.

$\chi^2$ -статистика Пірсона (*Chi-sq*) – відношення величин *MS* до *MSE*.

Величина *p*-значення – обчислюється за допомогою функції розподілу Фішера. Чим більше  $\chi^2$ -статистика, тим менше *p*-значення.

Функція **p = friedman(X, reps, displayopt)** у третьому аргументі *displayopt* дозволяє управляти графічним вивідом результатів аналізу. За *displayopt* = 'on' створюються графічні вікна, а за *displayopt* = 'off' – ні. За замовчуванням прийняте *displayopt* = 'on'.

Функція **[p, table] = friedman(...)** у другому результативному параметрі *table* повертає стандартну таблицю дисперсійного аналізу (замість фігури), включаючи мітки рядків і стовпців, у вигляді масиву рядків. Текстовий зміст цієї таблиці може бути також скопійований в буфер обміну за допомогою меню Edit – Copy Text (Редагувати – Копіювати текст).

Функція `[p,table,stats] = friedman(...)` у третьому результативному параметрі `stats` повертає структуру, яку надалі можна використовувати для більш детального порівняння середніх. Функція `friedman` перевіряє  $H_0$ -гіпотезу про рівність середніх усіх стовпців; альтернативною є гіпотеза про відмінність цих середніх. Проте іноді потрібне більш детальне дослідження: які саме пари середніх порівнянні, а які – ні. Для цього можна використовувати функцію `multcompare`, на вхід якої й потрібно подати цю структуру `stats`.

Тест Фрідмана використовує наступні припущення про дані  $X$ . Вважається, що всі вибірки мають однаковий безперервний розподіл з можливими відмінностями середніх у стовпцях і випадковими відмінностями в рядках. Усі виміри також припускаються взаємно незалежними. Класичний двофакторний дисперсійний аналіз накладає на дані більш обтяжливі обмеження про нормальний розподіл даних.

У *прикладі* досліджується вплив марки попкорну на прибутки від продажів. Стовпці матриці – це марки попкорну: *Gourmet*, *National* і *Generic*. Рядки – спосіб приготування: у маслі (*Oil*) або повітряний (*Air*). За кожної комбінації рівнів факторів проведено по 3 виміри прибутку.

```
>> clear all % очищаємо пам'ять
>> load popcorn % завантажуюмо базу даних за продажів попкорну
>> popcorn
popcorn =
    5.5000    4.5000    3.5000
    5.5000    4.5000    4.0000
    6.0000    4.0000    3.0000
    6.5000    5.0000    4.0000
    7.0000    5.5000    5.0000
    7.0000    5.0000    4.5000
>> p=friedman(popcorn,3) % тест Фрідмана
p =
    0.0010
```

Friedman's ANOVA Table					
Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	99.75	2	49.875	13.76	0.001
Interaction	0.0833	2	0.0417		
Error	16.1667	12	1.3472		
Total	116	17			

Test for column effects after row effects are removed

Рис. 12.36. Результати обчислень

На рис. 12.36 наведені результати тесту Фрідмана. Величина *pval* свідчить про те, що прибутки значимо змінюються зі зміною марки продукту.

### **MANOVA1** Однофакторний багатовимірний дисперсійний аналіз

Класифікаційна змінна – одна, тобто аналіз – однофакторний. Але результативних ознак може бути декілька, тому в назві методу присутній термін "багатовимірність". З цих кількісних результативних ознак складається канонічна (дискримінантна) функція, яка максимізує розбіжності між групами, і з цією (латентною) змінною провадиться звичайний однофакторний дисперсійний аналіз.

#### *Синтаксис*

**d = manova1(X,group)**

**d = manova1(X,group,alpha)**

**[d,p] = manova1(...)**

**[d,p,stats] = manova1(...)**

#### *Опис:*

Функція **d = manova1(X,group)** – проводить багатовимірний однофакторний дисперсійний аналіз у порівнянні з багатовимірними середніми стовпців матриці даних *X*, згрупованих відповідно до масиву *group*.

Аргумент *X* – матриця розміром  $m \times n$ , кожний рядок якої – один вимір *n*-вимірної величини. Аргумент *group* – це масив, що групує. Він повинен бути вектором, символьним масивом або масивом рядків; його довжина

– *m*. Два спостереження попадають в одну групу, якщо відповідні значення *group* однакові. Спостереження з кожної групи представляють вибірки з однієї генеральної сукупності. Результативний параметр *d* є оцінкою розмірності підпростору середніх груп. Функція **manova1** перевіряє 0-гіпотезу про те, що середні всіх груп – однакові *n*-вимірні вектори, а будь-які відмінності між групами – суто випадкові. Якщо *d* = 0, немає ніяких підстав відкидати 0-гіпотезу. Якщо *d* = 1, то на 5%-му рівні значимості 0-гіпотезу потрібно відкинути, але слід прийняти гіпотезу про те, що всі *n*-вимірні середні лежать на одній прямій (тобто розмірність підпростору середніх дорівнює 1). Аналогічно, якщо *d* = 2, то *n*-вимірні середні лежать в одній площині (підпросторі розмірності 2), але не на одній прямій тощо.

Функція **d = manova1(X,group,alpha)** у третьому аргументі *alpha* дозволяє задати рівень значимості (за замовчуванням 0.05). Значення *d*, що повертається, можна трактувати так: *d* – це найменша розмірність, для якої  $p > alpha$ , де *p* – *p*-значення для перевірки гіпотези про те, що всі середні лежать у підпросторі даної розмірності.

Функція **[d,p] = manova1(...)** – у другому результативному параметрі *p* повертає вектор *p*-значень для перевірки гіпотез про те, що всі середні лежать у підпросторах розмірності 0, 1, 2 тощо. Найбільша можлива розмірність простору середніх на одиницю менше числа груп. На кожну розмірність доводиться по одному елементу *p*, який строго менше рівня значимості. Якщо *p*(*i*) близько до нуля (наприклад, менше заданого рівня значимості), потрібно відкинути гіпотезу про те, що середні груп лежать у підпросторі розмірності (*i* – 1).

Функція **[d,p,stats] = manova1(...)** у третьому результативному параметрі *stats* повертає структуру з додатковою інформацією. Її поля перелічені в табл. 12.8.

Таблиця 12.8

**Поля структури *stats* функції **manova1****

Поле	Значення
------	----------

$W$	Матриця $W$ внутрішньогрупових сум квадратів і взаємних добутків
$B$	Матриця $B$ міжгрупових сум квадратів і взаємних добутків
$T$	Загальна матриця $T$ сум квадратів і взаємних добутків
$Dfw$	Число ступенів свободи для $W$
$Dfb$	Число ступенів свободи для $B$
$Dft$	Число ступенів свободи для $T$
$Lambda$	Вектор значень $\lambda$ -статистик Уїлка (Wilk) для перевірки гіпотез про те, що підпростір середніх має розмірність 0, 1 тощо.
$Chisq$	Перетворення $\lambda$ -статистик до наближеного $\chi^2$ -розподілу
$Chisqdf$	Число ступенів свободи для $chisq$
$Eigenval$	Власні значення матриці $W^{-1}B$
$Eigenvec$	Власні вектори $W^{-1}B$ , які є коефіцієнтами канонічних змінних $C$ , що масштабовані так, щоб дисперсії канонічних змінних були рівними 1
$Canon$	Канонічні змінні $C$ , що дорівнюють $XC * Eigenvec$ , де $XC$ – це матриця $X$ зі стовпцями, центрованими вирахуванням їх середніх
$Mdist$	Вектор відстаней Махаланобіса від точки до їх середніх групових
$Gmdist$	Матриця відстаней Махаланобіса між середніми груп

Канонічні змінні  $C$  – це такі лінійні комбінації вихідних змінних, які максимізують відмінність між групами. Так, 1-й стовпець  $C$  – це така лінійна комбінація стовпців  $X$ , яка серед усіх можливих лінійних комбінацій дають найбільш значиму F-статистику для однофакторного дисперсійного аналізу. Далі, 2-й стовпець  $C$  – це така лінійна комбінація стовпців  $X$ , ортогональна 1-му стовпцю  $C$ , яка серед усіх можливих таких лінійних комбінацій дає найбільш значиму F-статистику для однофакторного дисперсійного аналізу тощо.

Результативні параметри функції **manova1** можуть бути використані для подальшого аналізу даних. Можна, наприклад, за допомогою функції **gplotmatrix** провести угруповання змінних. Можна використовувати функцію **gscatter** для візуалізації груп, використовуючи перші дві канонічні змінні. Функція **manovacluster** будує дендрограму кластерів групових середніх.

Даний тест використовує наступні припущення про дані  $X$ : усі сукупності повинні мати нормальний розподіл, однакові коваріаційні матриці, а всі виміри мають бути взаємно незалежними.

У прикладі досліджується відмінність 4-вимірному вектора характеристик автомобілів (питомий пробіг, прискорення, маса, об'єм циліндрів) залежно від країни виробника.

```

>> clear all % очищаємо пам'ять
>> load carbig % завантажуюємо базу даних щодо автомобілів
>> [d,p]=manova1([MPG Acceleration Weight Displacement],Origin)
d =
    3
p =
    0
    0.00000031405833
    0.00751099957774
    0.19341007458976

```

На рівні значимості 5% розмірність підпростору середніх виявилася рівною 3. Це означає, що із чотирьох координат векторів середніх одна є лінійною комбінацією інших з довірчою ймовірністю 95%.

### **MANOVA CLUSTER** Дендрограма результатів однофакторного багатовимірного дисперсійного аналізу

Використовуючи результати рішення задачі однофакторного багатовимірного дисперсійного аналізу за допомогою функції **manova1**, будується дендрограма розподілу груп на кластери.

*Синтаксис*

**manovacluster(stats)**

**manovacluster(stats,method)**

**h = manovacluster(stats,method)**

*Опис:*

Функція **manovacluster(stats)** створює фігуру й будує на ній дендрограму середніх груп після багатовимірного однофакторного дисперсійного аналізу. Аргумент *stats* – це результативний параметр функції **manova1**. Для знаходження кластерів застосовується метод послідовної побудови матриці відстаней Махаланобіса між середніми груп.

Функція **manovacluster(stats,method)** у другому аргументі *method* дозволяє задати в явному вигляді метод побудови ієрархічного дерева кластерів. Цей аргумент повинен бути текстовим рядком. Його можливі значення:

*'single'* – метод найкоротших відстаней (за замовчуванням);

'complete' – метод найбільших відстаней;

'average' – метод середніх відстаней;

'centroid' – метод відстаней між центрами груп;

'ward' – метод мінімізації внутрішньокластерної дисперсії.

Функція **h = manovacluster(stats,method)** – повертає вектор показчиків ліній (дескрипторів) на фігурі.

У прикладі демонструються результати однофакторного багатовимірного дисперсійного аналізу і роботи функції **manovacluster(stats)**. Вони зображені на рис. 12.37 у вигляді дендрограми розподілу груп на кластери. Це дозволяє оцінити, автомобілі яких країн найбільш відрізняються від інших за досліджуваними параметрами, а які – не дуже сильно.

```
>> load carbig % завантажуюмо базу даних щодо автомобілів
>> [d,p,stats]=manova1([MPG Acceleration Weight Displacement],Origin);
>> manovacluster(stats); % будуємо дендрограму 'single'
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',12)
>> title('\bfДендрограма схожих автомобілів') % заголовок
>> xlabel('\bfКраїна') % мітка осі OX
>> grid on, box on
```

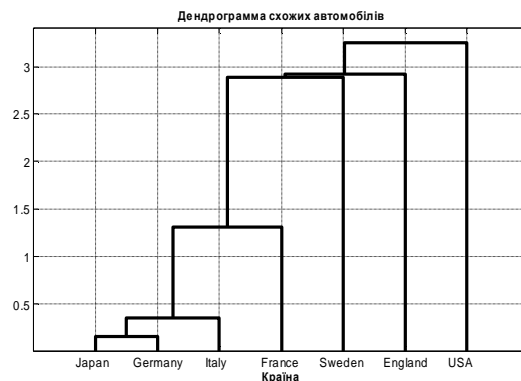


Рис. 12.37. Дендрограма угруповання автомобілів

З рис. 12.37 виходить, що найбільш відрізняються від інших американські автомобілі, далі – британські, шведські і т.д. Зауважимо, що за замовченням тут прийнятий метод кластеризації 'single', який зараз вважається лише початковим. Встановлено, що за цього методу

правильно об'єднуються дійсно найближчі об'єкти, а далі до створених кластерів

почерзі приєднуються (притягуються) інші об'єкти. Найбільш ефективною методикою зараз вважається метод Уорда ('ward').

## 12.2. Коваріаційний аналіз

Коваріаційний аналіз об'єднує дисперсійний і регресійний аналізи. У довіднику представлений нижче двома функціями:

<b>AOCTOOL</b>	Інтерактивний графічний коваріаційний аналіз .....	132
<b>DUMMYVAR</b>	Формування матриці індикаторних змінних .....	146

### **AOCTOOL** Інтерактивний графічний коваріаційний аналіз

Дана функція дозволяє в діалоговому режимі вирішувати задачу коваріаційного аналізу лінійної кількісної залежності, коли дані належать до різних груп спостережень. Функція **aoctool** виводить на екран:

- графіки лінійних математичних моделей;
- таблицю з оцінками параметрів математичних моделей;
- таблицю дисперсійного (коваріаційного) аналізу.

*Синтаксис:*

```
aoctool(x,y,g)  
aoctool(x,y,g,alpha)  
aoctool(x,y,g,alpha,xname,yname,gname)  
aoctool(x,y,g,alpha,xname,yname,gname,'displayopt')  
aoctool(x,y,g,alpha,xname,yname,gname,'displayopt','model')  
h = aoctool(...)  
[h,atab,ctab] = aoctool(...)  
[h,atab,ctab,stats] = aoctool(...)
```

*Опис:*

Функція **aoctool(x,y,g)** дозволяє провести парний коваріаційний аналіз, сприймаючи вектор-стовпець *x* як абсциси, вектор-стовпець *y* такої ж довжини як ординати, *g* – групуючий масив і знаходить найкращі

лінійні залежності для всіх груп даних. Групуєчий масив  $g$  може бути вектором такої ж довжини, як  $x$  і  $y$ , масивом символів або масивом рядків. Окремі точки попадають в одну групу, якщо для них значення відповідних елементів  $g$  однакові. В англійській літературі цей вид аналізу відомий за абрєвіатури **ANOCOVA**.

*Примітка:* групуєча змінна  $g$  повинна містити значення, що поділяють вектори  $x$ ,  $y$  як мінімум на дві вибірки, а якщо ні, то буде видано наступне повідомлення про помилку:

```
>> x=normrnd(100,1);
>> y=normrnd(100,1);
>> g=ones(100,1);
>> aoctool(x,y,g)
??? Error using ==> aoctool
Must have at least two groups % Має бути не менш двох груп
```

Виклик функції приводить до побудови трьох фігур: інтерфейсу користувача з даними й побудованими графіками прямих, стандартної таблиці дисперсійного аналізу і таблиці з оцінками параметрів моделі.

Створені вікна є інтерактивними. На першій фігурі є стандартне меню, яке є також на всіх інших фігурах **MatLab**, і додаткові елементи керування (детально описані нижче). За замовченням будуються окремі лінійні моделі для кожної групи спостережень. Ці установи можна змінити елементами керування, які розташовані на першому інтерактивному вікні, але вони впливають також на друге і третє вікна.

У стандартній таблиці дисперсійного аналізу (друге графічне вікно) є 6 стовпців:

Джерело розкиду даних (*Source*). Кількість цих джерел залежить від моделі. Ураховується також випадковий розкид (*Errors*);

Сума квадратів відхилень (*Sum of Squares*, скорочено *Sum Sq*);

Число ступенів свободи (*the degrees of freedom*, або *d.f.*).

Середні квадрати (*Mean of Squares*, або *Mean Sq*), які підраховуються як  $(Sum Sq) / (d.f.)$ .

F-статистика Фішера, що є відношенням величин *Mean Sq* до *MSE*;

Величина  $p$ -значення, яка обчислена за допомогою функції розподілу Фішера. Чим більше F-статистика, тим менше  $p$ -значення.

На фігурі з оцінками параметрів лінійних залежностей (третє графічне вікно) – 5 стовпців:

Параметри ліній (*Term*). Таких елементів – по два на кожну групу даних плюс ще два на об'єднані дані: константа в рівнянні прямої (*Intercept*) і кутовий коефіцієнт (*Slope*);

Оцінка параметра (*Estimate*);

Середньоквадратична похибка наближення (*Standard Error*, скорочено *Std. Err.*);

t-статистика Стьюдента (*T*);

Величина  $p$ -значення, яка обчислена за допомогою функції розподілу Стьюдента. Чим більше t-статистика, тим менше  $p$ -значення.

Функція **aoctool(x,y,g,alpha)** в аргументі *alpha* дозволяє задати рівень значимості для розрахунку границь довірчого інтервалу. Довірча ймовірність визначається як  $100*(1 - alpha)\%$ . За замовчуванням рівень значимості ухвалюється як 0.05.

У функції **aoctool(x,y,g,alpha,xname,yname,gname)** додаткові аргументи *xname*, *yname*, *gname* дозволяють задати назви змінним *x*, *y*, *g* на графіках і в таблицях графічних вікон. Якщо в якості вхідних аргументів *x*, *y*, *g* використовуються змінні (не вектори), *aoctool* використовує їх ідентифікатори як назви на графіках і в таблицях. Вхідні аргументи *xname*, *yname*, *gname* слід використовувати у випадках, коли для завдання *x*, *y*, *g* використовуються формули (вираження). Наприклад, якщо в якості *x* було використане виділення 1 стовпця матриці *m* - *m(:,1)*, то необхідно в якості назви *x* використовувати деяку рядкову константу *xname* = 'Col1'.

Функція **aoctool(x,y,g,alpha,xname,yname,gname,'displayopt')** в аргументі 'displayopt' дозволяє явно задати відображення графічних вікон

з результатами аналізу за *'displayopt' = 'on'*, або не виводити графічні вікна за *'displayopt' = 'off'*. За замовчуванням *'displayopt' = 'on'*.

**aoctool(x,y,g,alpha,xname,yname,gname,'displayopt','model')** – у цьому синтаксисі вхідний аргумент *'model'* призначений для задання початкової моделі в ході проведення коваріаційного аналізу. Можливі наступні значення параметра *'model'*:

*'same mean'* – визначення загального середнього арифметичного з ігноруванням угруповання значень  $x, y$ ;

*'separate means'* – визначення середніх арифметичних для кожної підгрупи окремо;

*'same line'* – визначення параметрів лінійної регресійної моделі з ігноруванням угруповання;

*'parallel lines'* – визначення параметрів лінійних регресійних моделей для кожної групи за умови, що лінії рівнянь регресії повинні бути паралельні;

*'separate lines'* – визначення параметрів лінійних регресійних моделей для кожної групи без будь-яких обмежень.

Вид коваріаційних моделей наведено в табл. 12.9.

Таблиця 12.9

### Коваріаційні моделі

Значення параметра <i>'model'</i>	Рівняння
<i>Same mean</i>	$y = \alpha + \varepsilon$
<i>Separate means</i>	$y = (\alpha + \alpha_i) + \varepsilon$
<i>Same line</i>	$y = \alpha + \beta x + \varepsilon$
<i>Parallel lines</i>	$y = (\alpha + \alpha_i) + \beta \cdot x + \varepsilon$
<i>Separate lines</i>	$y = (\alpha + \alpha_i) + (\beta + \beta_j)x + \varepsilon$

Тут  $\varepsilon$  – похибка математичної моделі;  $\alpha$  – загальне середнє за всіма підгрупами (загальний початковий зсув на графіку);  $\alpha_i$  – додатковий зсув для  $i$ -ої підгрупи експериментальних значень відносно  $\alpha$ ;  $\beta$  – коефіцієнт за першого ступеня незалежної змінної для всіх підгруп (кут нахилу лінійної регресійної моделі до осі абсцис);  $\beta_j$  – додаток до

коефіцієнта за першого ступеня незалежної змінної для  $j$ -ої підгрупи експериментальних значень відносно  $\beta$ . Для регресійних моделей окремих груп з паралельними прямими, *'model'='Parallel lines'*, величини кутів нахилу  $\beta_j$  приймуть однакові значення, а значення початкового зсуву можуть бути різними  $\alpha_i = var$ .

Функція **h = aoctool(...)** повертає вектор  $h$  дескрипторів (покажчиків) об'єктів – ліній графіків на графіку.

Функція **[h,atab,ctab] = aoctool(...)** крім вектора покажчиків  $h$  повертає масиви рядків *atab*, що містять результати однофакторного дисперсійного аналізу у вигляді стандартної таблиці і відформатовану в текстовому вигляді таблицю значень точкових оцінок коефіцієнтів *ctab*, включаючи мітки рядків і стовпців.

Функція **[h,atab,ctab,stats] = aoctool(...)** у четвертому результативному параметрі *stats* повертає структуру, яку надалі можна використовувати для більш детального порівняння даних. Функція *aoctool* перевіряє 0-гіпотези про рівність середніх і кутів нахилу різних груп; альтернативними є гіпотези про відмінність цих величин. Проте іноді потрібно більш детальне дослідження: саме які пари середніх і кутів нахилу порівнянні, а які – ні. Для цього можна використовувати функцію **multcompare** (див. вище), на вхід якої й потрібно подати структуру *stats*. Під час проведення процедури парного порівняння можна порівнювати: початкові зсуви регресійних моделей, коефіцієнти за лінійних ступенів регресійних моделей або загальні середні (середні по підгрупах).

*Приклад:*

Розглядається приклад (дані з файла *carsmall.mat*) коваріаційного аналізу з метою визначення наявності й виду взаємозв'язку між масою автомобіля й величиною його питомого пробігу. Автотранспортні засоби згруповані за роками випуску: 1970, 1976, 1982.

Завантаження даних:

```
>> load carsmall
```

Змінні вибірових даних файла `catsmall.mat` переглянути у вікні `Workspace Browser` (рис. 12.38):

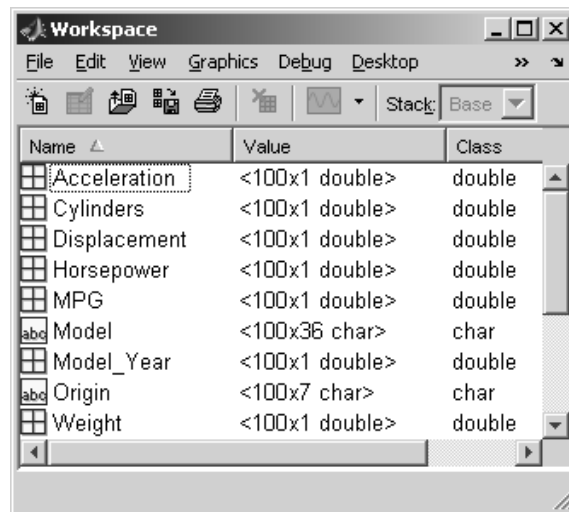


Рис. 12.38. Структура файла `carsmall.mat`

Тут:

*Acceleration* – час розгону автомобіля з місця до 60 миль/год,

*Cylinders* – кількість циліндрів,

*Displacement* – об'єм циліндрів у куб. дюймах,

*Horsepower* – потужність у к. с.,

*MPG* – питомий пробіг у милях на галон пального,

*Model* – назва моделі,

*Model\_Year* – рік випуску,

*Origin* – країна-виробник: США, Японія, Європа,

*Weight* – маса автомобіля у фунтах.

Розглядається залежність між масою автомобіля – *Weight* (це незалежна кількісна змінна) і питомим пробігом – *MPG* (це результативна ознака) з урахуванням якісного фактора – року випуску – *Model\_Year* (це класифікаційна змінна). Функція `aoctool` повертає вектор дескрипторів *h* – покажчиків на лінії графіка, результати однофакторного дисперсійного (коваріаційного) аналізу *atab* і таблицю значень точкових й інтервальних оцінок коефіцієнтів регресійної моделі *ctab*.

```
>> [h,atab,ctab,stats]=aoctool(Weight,MPG,Model_Year)
```

```

h =
    157.0020
    158.0015
    159.0015
    215.0017
    216.0017
    217.0017
    218.0017
    219.0017

atab =
    'Source'          'd.f.'  'Sum Sq'  'Mean Sq'  'F'      'Prob>F'
    'Model_Year'     [2]    [807.6896] [403.8448] [51.9762] [1.2212e-015]
    'Weight'         [1]    [2.0502e+003] [2.0502e+003] [263.8679] [0]
    'Model_Year*Weight' [2]    [81.2188] [40.6094] [5.2266] [0.0072]
    'Error'          [88]   [683.7420] [7.7698] [] []

ctab =
    'Term'      'Estimate'  'Std. Err.'  'T'      'Prob>|T|'
    'Intercept' [ 45.9798] [1.5208]     [30.2330] [2.9266e-048]
    ' 70'       [-8.5805] [1.9619]     [-4.3737] [3.3417e-005]
    ' 76'       [-3.8902] [1.8686]     [-2.0818] [ 0.0403]
    ' 82'       [12.4707] [2.5568]     [4.8775] [4.7391e-006]
    'Slope'     [-0.0078] [5.5717e-004] [-14.0031] [4.0955e-024]
    ' 70'       [0.0020] [6.6152e-004] [2.9605] [0.0039]
    ' 76'       [0.0011] [6.5328e-004] [1.7425] [0.0849]
    ' 82'       [-0.0031] [9.9914e-004] [-3.0994] [0.0026]

stats =
    source: 'aoctool'          s: 2.7874          intercepts: [3x1 double]
    gnames: {3x1 cell}        model: 5           intercov: [3x3 double]
    n: [3x1 double]          slopes: [3x1 double] pmm: [3x1 double]
    df: 88                   slopecov: [3x3 double] pmmcov: [3x3 double]

```

Крім цієї інформації, результати розрахунку функції **aoctool** представлені також в 3-х графічних вікнах (рис. 12.39, 12.42, 12.43).

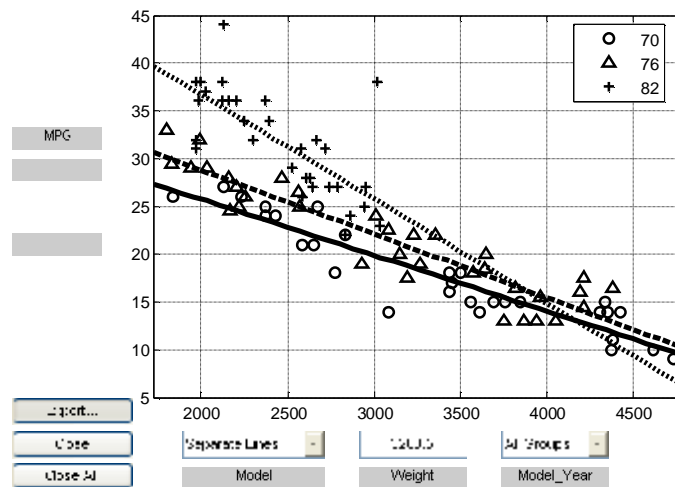


Рис. 12.39. Вікно графіків лінійної регресійної моделі з експериментальними точками

У першому графічному вікні (рис. 12.39) відображуються графіки лінійних залежностей від кількісної змінної-аргументу для кожної групи даних.

В цьому інтерактивному вікні є наступні елементи керування, які впливають також на друге і третє графічні вікна:

Кнопка **Export** (Експорт) дозволяє передати в робочий простір **MatLab** усі знайдені величини, змінивши за необхідністю їх ідентифікатори (рис. 12.40);

Кнопка **Close** (Закрити) закриває графічне вікно рис. 12.39;

Кнопка **Close All** (Закрити все) закриває всі три графічних вікна;

В списку **Model** (Модель), що розкривається, можна задати вид моделі, яка відразу відображається на цьому графіку, а дані розрахунку – в інших вікнах;

Поле уведення з назвою змінної-аргументу, де можна задати будь-яке значення кількісного аргументу по осі абсцис. Зміна числа у цьому полі приводить до зсуву вертикальної штрихової лінії. Цю штрихову лінію можна також пересувати мишкою, тоді у полі уведення відображається нова абсциса. За замовчуванням за завантаження вікна штрихова лінія встановлюється в середнє положення;

У списку, що розкривається, з назвою фактора (класифікаційної змінної) міститься перелік груп. Якщо вибрати опцію *Всі групи (All Groups)*, будуються лінії регресії для всіх груп. Якщо ж вибрати одну із груп, то будуються лінія регресії й довірчі інтервали лише для неї. За замовчуванням за завантаження вікна вибирається опція *Всі групи*.

Види моделей (рис. 12.41):

Загальне середнє (*Same Mean*) – знаходиться одна найкраща константа (середнє) для всіх груп даних;



Рис. 12.40. Панель експорту

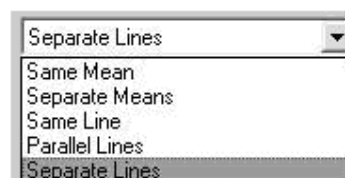


Рис. 12.41. Види моделей

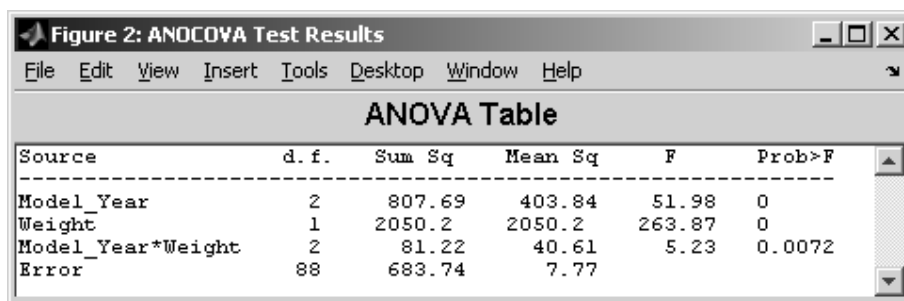
Різні середні (*Separate Means*) – для кожної групи даних знаходяться свої найкращі константи;

Загальна пряма (*Same Line*) – для всіх груп даних знаходиться одна найкраща пряма;

Паралельні лінії (*Parallel Lines*) – для кожної групи даних знаходиться своя найкраща константа, але приймається однаковий найкращий кут нахилу для всіх прямих;

Різні лінії (*Separate Lines*) – для кожної групи даних знаходиться своя найкраща пряма. За замовчуванням ця опція включається за завантаження вікна.

Результати оцінки значимості ефекту взаємодії й головних ефектів наведені в таблиці однофакторного дисперсійного аналізу (друге графічне вікно ANOCOVA Test Results) на рис. 12.42.



The screenshot shows a window titled "Figure 2: ANOCOVA Test Results" with a menu bar (File, Edit, View, Insert, Tools, Desktop, Window, Help). Below the menu bar is an "ANOVA Table" with the following data:

Source	d. f.	Sum Sq	Mean Sq	F	Prob>F
Model_Year	2	807.69	403.84	51.98	0
Weight	1	2050.2	2050.2	263.87	0
Model_Year*Weight	2	81.22	40.61	5.23	0.0072
Error	88	683.74	7.77		

Рис. 12.42. Вікно однофакторного дисперсійного аналізу

Ефект взаємодії  $Model\_Year \times Weight$  дозволяє оцінити значимість відмінностей в коефіцієнтах за лінійного ступеня незалежною змінною (кути нахилу ліній регресії на рис. 12.39). Значення статистики Фішера  $F = 5.23$  і рівня значимості  $p = 0.0072 < 0.01$  дозволяє зробити висновок про значиму відмінність у лінійних коефіцієнтах за першого ступеня незалежної змінної.

У третьому графічному вікні ANOCOVA Coefficients (рис. 12.43) наведені значення коефіцієнтів регресійної моделі з виправленнями для кожної групи окремо. Крім точкових оцінок параметрів лінійних регресійних моделей, це вікно містить також t-статистики Стюдента і рівні значимості *Prob*, призначені для перевірки нульової гіпотези про значимість отриманих оцінок.

Term	Estimate	Std. Err.	T	Prob> T
Intercept	45.9798	1.52085	30.23	0
70	-8.5805	1.96186	-4.37	0
76	-3.8902	1.86864	-2.08	0.0403
82	12.4707	2.5568	4.88	0
Slope	-0.0078	0.00056	-14	0
70	0.002	0.00066	2.96	0.0039
76	0.0011	0.00065	1.74	0.0849
82	-0.0031	0.001	-3.1	0.0026

Рис. 12.43. Вікно оцінок параметрів лінійних регресійних моделей

З рис. 12.43 можна виписати рівняння регресії за кожний рік окремо і в середньому за всі ці роки:

В середньому: 
$$y = 45.9798 - 0.0078x + \varepsilon$$

Рік випуску 1970: 
$$y = (45.9798 - 8.5805) + (-0.0078 + 0.0020)x + \varepsilon$$

Рік випуску 1976: 
$$y = (45.9798 - 3.8902) + (-0.0078 + 0.0011)x + \varepsilon$$

Рік випуску 1982: 
$$y = (45.9798 + 12.4707) + (-0.0078 - 0.0031)x + \varepsilon$$

Для оцінки міри відповідності лінійних регресійних моделей з обмеженнями на рівність коефіцієнтів за перших ступенів незалежних змінних слід в графічному вікні експериментальних даних (рис. 12.39) змінити опцію *'model' = 'separate lines'* на *'model' = 'parallel lines'*.

Після зміни виду регресійних моделей будуються нові залежності й перераховуються всі значення відповідних статистик в графічних вікнах (рис. 12.44 – 12.46).

На рис. 12.44 показані графіки залежностей за установки опції *'Parallel lines'* у списку *Model*, що розкривається.

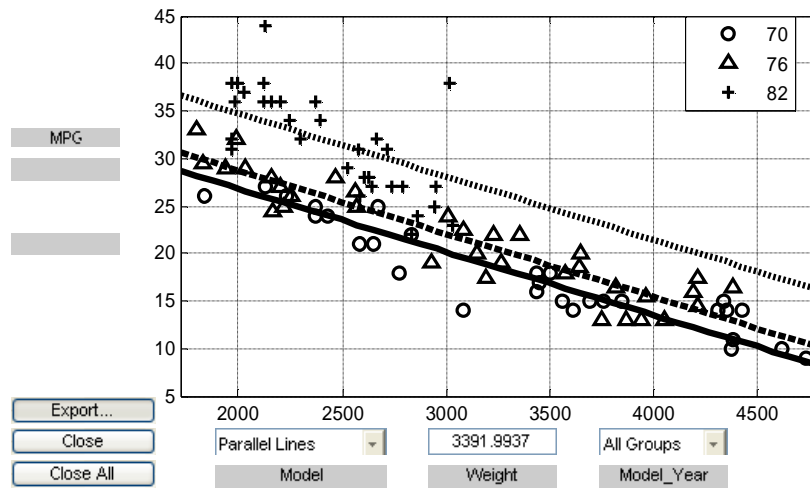


Рис. 12.44. Вікно графіків лінійних регресійних моделей для *'Model'='parallel lines'*

Опція *'Parallel lines'* визначає параметри лінійних регресійних моделей для кожної групи за умови, що лінії регресії повинні бути паралельні між собою. Експериментальні точки добре групуються навколо цих ліній. Отримані результати свідчать про те, що лінійні моделі без обмежень щодо значень лінійних коефіцієнтів і з однаковими лінійними коефіцієнтами майже еквівалентні.

На рис. 12.45 і 12.46 зображені друге і третє графічні вікна.

The screenshot shows a window titled 'Figure 2: ANCOVA Test Results'. It contains an ANOVA table with the following data:

Source	d. f.	Sum Sq	Mean Sq	F	Prob>F
Model_Year	2	807.69	403.84	47.51	0
Weight	1	2050.2	2050.2	241.21	0
Error	90	764.96	8.5		

Рис. 12.45. Вікно однофакторного дисперсійного аналізу для *'Model'='parallel lines'*

Figure 3: ANCOVA Coefficients

Term	Estimate	Std. Err.	T	Prob> T
Intercept	43.3898	1.30575	33.23	0
70	-3.2795	0.4674	-7.02	0
76	-1.3504	0.41996	-3.22	0.0018
82	4.6298	0.48037	9.64	0
Slope	-0.0066	0.00043	-15.53	0

Рис. 12.46. Вікно точкових оцінок параметрів лінійних регресійних моделей для *'Model'='parallel lines'*

Якщо замовити дослідження залежності між пробігом і масою автотранспортних засобів в одній окремій групі, то разом з графіком ліній регресії будуть побудовані границі 95%-ї довірчої полоси на розрахункові значення (рис. 12.47). Для цього в списку *Model\_Year*, що розкривається, обираємо ім'я потрібної групи. У припущенні лінійної залежності між змінними границі довірчого інтервалу будуть відповідати 95%-й довірчій імовірності.

Для прикладу аналізуємо моделі випуску 1982 року. В основному вікні ANCOVA Prediction Plot (рис. 12.47) з'являється залежність між змінними *Weight – MPG* з границями 95%-го довірчого інтервалу лінію регресії (*Model='Separate Lines', Model\_Year=82*). Графіки залежностей для інших груп (вже без довірчих інтервалів) будуть побудовані сірим кольором.

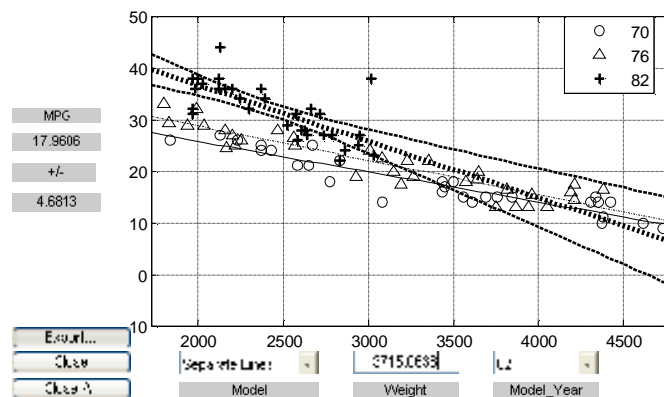



Рис. 12.47. Вікно графіків експериментальних даних, лінійної регресійної моделі й границь довірчих інтервалів для *'Model'='separate lines'* і *Model\_Year = 82*

Ліворуч з'являється результат  $MPG = 17.9606 \pm 4.5813$  для заданого значення  $Weight = 3715.0638$ .

У деяких випадках доцільно розраховувати границі довірчого інтервалу для окремого або нового спостереження (скласти прогноз), а не для середнього значення залежної змінної. Для перемикання способу розрахунку границь довірчого інтервалу використовується команда основного меню Bounds. У підменю, що з'являється, ставимо прапорець  напроти Bounds – Observation. Команда Bounds – Line дозволяє повернутися для розрахунку границь довірчого інтервалу за середніми значеннями. Перемикання між способами розрахунку границь довірчого інтервалу має сенс лише для окремих груп. Тому спочатку слід в списку Model вибрати групу, що нас цікавить. Для  $Model\_Year = 82$  прогнозні границі довірчого інтервалу для окремих значень  $x = Weight$  приймуть вигляд, показаний на рис. 12.48. Ліворуч з'являється результат  $MPG = 17.9606 \pm 8.3719$  для заданого значення  $Weight = 3715.0638$ .

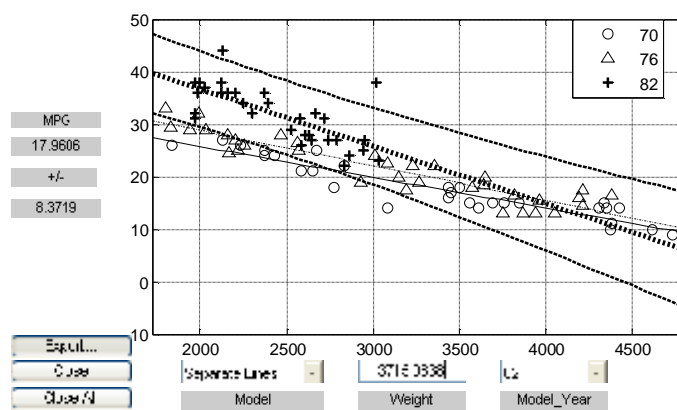


Рис. 12.48 Вікно графіків лінійної регресійної моделі й границь довірчих інтервалів для 'Model' = 'separate lines' і  $Model\_Year = 82$

Діючи мишкою (або набираючи нове значення в поле Weight), можна змінити положення пунктирної лінії по осі абсцис, і одержати нове значення залежної змінної  $MPG$  щодо регресійної моделі й границь довірчого інтервалу.

Цей спосіб визначення точкових й інтервальних оцінок використовується лише для обраної певної підгрупи в списку

класифікаційної змінної `Model_Year`. Якщо в списку значень класифікаційної змінної `Model_Year` обране значення "All Groups", цей спосіб визначення точкових й інтервальних оцінок не працює.

Для проведення процедури парного порівняння використовується функція `multcompare`. Вхідним аргументом `multcompare` є структура `stats`, що попередньо отримана в результаті виклику функції `aocool`. Процедура парного порівняння може бути застосована для параметрів лінійних регресійних моделей, отриманих для різних значень класифікаційної змінної, або маргінальних середніх. Маргінальними середніми для розглянутого прикладу є розраховані за регресійною моделлю значення залежної змінної *MPG* для середньої маси кожної підгрупи.

Виконаємо парне порівняння коефіцієнтів на першому ступені незалежної змінної. Як було показано в попередніх пунктах коваріаційного аналізу, кут нахилу лінійних регресійних моделей має бути різним для різних груп. Застосування процедури парного порівняння дозволяє визначити пари коефіцієнтів, що статистично значимо відрізняються між собою.

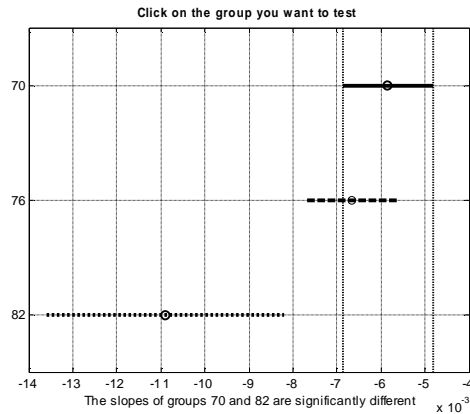
Виклик функції парного порівняння:

```
>> multcompare(stats,0.05,'on')
ans =
    1.0000    2.0000   -0.0012    0.0008    0.0029
    1.0000    3.0000    0.0013    0.0051    0.0088
    2.0000    3.0000    0.0005    0.0042    0.0079
```

З першого рядка матриці `ans` виходить, що різниця між коефіцієнтами на першому ступені незалежної змінної для першої й другої підгруп експериментальних значень (моделі випуску 1970 і 1976 років) склала 0.0008 із границями довірчого інтервалу  $[-0.0012, 0.0029]$ . Оскільки цей довірчий інтервал включає нульове значення, то різниця між коефіцієнтами статистично незначима. З 2-х наступних рядків матриці `ans` можна зробити висновок про значиму відмінність між відповідними коефіцієнтами 1 і 3 підгрупами (моделі 1970 і 1982 років), а також між 2 і 3 підгрупами (моделі 1976 і 1982 років), тобто група 3 значимо відрізняється від груп 1 і 2 за величинами кутового коефіцієнта

за незалежної кількісної змінної, оскільки нульове значення не попадає в межі довірчих інтервалів на різниці зазначених коефіцієнтів.

Графік значень коефіцієнтів на першому ступені незалежної змінної для 1, 2, 3 груп і їх довірчих інтервалів представлено на рис. 12.49.



**Рис. 12.49. Графічне вікно результатів парного порівняння значень коефіцієнтів на першому ступені незалежної змінної для груп "70", "76", "82"**

З рис. 12.49 видно, що довірчі інтервали для груп автомобілів, виготовлених в 1970 і 1976 роках, не перекриваються довірчим інтервалом групи автомобілів, випущених в 1982 році. Це графічно підтверджує, що кутовий коефіцієнт лінії регресії для 3-ї групи значимо відрізняється від кутових коефіцієнтів ліній 1-ї і 2-ї першої групи.

Слід зауважити, що структура *stats* була отримана для значення "model" за замовчуванням і не може бути змінена в інтерактивному режимі. Проведення парного порівняння для інших моделей вимагатиме явно вхідний аргумент 'model' функції **aocool**, і далі отриману структуру *stats* передати як вхідний аргумент **multcompare**.

### **DUMMYVAR** Формування матриці індикаторних змінних

Інша назва думпу-змінних – макетні змінні (dummy – макет). Але у вітчизняній літературі також часто вживається неправильна термінологія: "фіктивні змінні". Фіктивною є "змінна"  $X_0 \equiv 1$ , яку уводять в регресійну модель для урахування в неї вільного члена.

Функція **dummyvar** здійснює умовне кодування класифікаційних змінних, вона повертає матрицю одиниць і нулів. Ця матриця містить число стовпців, що дорівнює сумі різних значень (категорій) у стовпцях вихідної матриці (матриці класифікаційних змінних). Одиниці й нулі характеризують наявність або відсутність певної категорії в кожному стовпчику вихідної матриці.

*Синтаксис:*

**D = dummyvar(group)**

*Опис:*

Функція **D = dummyvar(group)** дозволяє одержати матрицю індикаторних змінних  $D$ , що містить одиниці й нулі. Одиничні значення в  $D$  означають наявність певного елемента в матриці  $group$ , нульові – відсутність цього елемента. Елементами матриці  $group$  має бути позитивні цілі числа. Кількість рядків матриць  $group$  і  $D$  буде однаковим. Кількість стовпців у матриці  $D$  буде визначатися сумою різних значень у стовпцях матриці  $group$ .

Матрицю  $D$  формується в такий спосіб:

Спочатку створюємо нульову матрицю  $D$  із числом рядків рівним кількості рядків матриці  $group$  і кількістю стовпців рівним сумі чисел можливих значень у стовпцях матриці  $group$ . Наприклад, якщо матриця  $group$  містить два стовпці (дві класифікаційні змінні) з числом рівнів 2 – для першого стовпця і 3 – для другого стовпця

$$group = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 2 & 1 \\ 2 & 2 \\ 2 & 3 \end{bmatrix},$$

то майбутня матриця індикаторних змінних  $D$  прийме наступний вигляд:

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Кількість стовпців матриці  $D$  буде дорівнювати  $2 + 3 = 5$ , де 2 – число можливих значень у першому стовпці матриці *group* (1 і 2); 3 – число можливих значень у другому стовпці матриці *group* (1, 2, 3).

Далі деякі елементи матриці  $D$  змінюємо на "1". Перші два стовпці матриці  $D$  відповідають двом можливим значенням першого стовпця матриці *group*, останні три стовпці матриці  $D$  відповідають трьом можливим значенням другого стовпця матриці *group*. У першому стовпці матриці  $D$  ставимо одиниці в тих рядках, де перша класифікаційна змінна (перший стовпець матриці *group*) приймає своє перше значення (1); у другому стовпці матриці  $D$  ставимо одиниці в тих рядках, де перша класифікаційна змінна (перший стовпець матриці *group*) приймає своє друге значення (2). Аналогічно заповнюються останні три стовпці матриці  $D$ , які відповідають першому, другому і третьому рівням другої класифікаційної змінної (другому стовпцю матриці *group*).

Для розглянутого вище прикладу одержимо:

$$D = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

*Приклади:*

1. Розглянемо формування матриці індикаторних змінних відповідно до матриці *group*, наведеної в описі функції **dummyvar**.

```
>> group = [1 1;1 2;1 3;2 1;2 2;2 3]
```

```
group =
```

```
1 1
1 2
1 3
2 1
2 2
2 3
```

```
>> D = dummyvar(group)
```

```
D =
```

```
1 0 1 0 0
1 0 0 1 0
1 0 0 0 1
0 1 1 0 0
0 1 0 1 0
0 1 0 0 1
```

2. Оскільки матриця індикаторних змінних часто використовується в плануванні експерименту, розглянемо наступний приклад.

Вивчаються ефекти від роботи 3 операторів на 3 верстатах.

```
>> group = [1 1 1; 1 1 2; 1 1 3; 1 2 1; 1 2 2; 1 2 3; 1 3 1; 1 3 2; 1 3 3; 2 1 1; 2 1 2; 2 1 3;
2 2 1; 2 2 2; 2 2 3; 2 3 1; 2 3 2; 2 3 3; 3 1 1; 3 1 2; 3 1 3; 3 2 1; 3 2 2; 3 2 3; 3 3 1; 3 3 2;
3 3 3]
```

```
group =
```

```
1 1 1 | 2 1 1 | 3 1 1
1 1 2 | 2 1 2 | 3 1 2
1 1 3 | 2 1 3 | 3 1 3
1 2 1 | 2 2 1 | 3 2 1
1 2 2 | 2 2 2 | 3 2 2
1 2 3 | 2 2 3 | 3 2 3
1 3 1 | 2 3 1 | 3 3 1
1 3 2 | 2 3 2 | 3 3 2
1 3 3 | 2 3 3 | 3 3 3
```

```
>> D = dummyvar(group)
```

```
D =
```

```
1 0 0 1 0 0 1 0 0 | 0 1 0 0 1 0 0 0 1
1 0 0 1 0 0 0 1 0 | 0 1 0 0 0 1 1 0 0
1 0 0 1 0 0 0 0 1 | 0 1 0 0 0 1 0 1 0
1 0 0 0 1 0 1 0 0 | 0 1 0 0 0 1 0 0 1
1 0 0 0 1 0 0 1 0 | 0 0 1 1 0 0 1 0 0
1 0 0 0 1 0 0 0 1 | 0 0 1 1 0 0 0 1 0
1 0 0 0 0 1 1 0 0 | 0 0 1 1 0 0 0 0 1
1 0 0 0 0 1 0 1 0 | 0 0 1 0 1 0 1 0 0
1 0 0 0 0 1 0 0 1 | 0 0 1 0 1 0 0 1 0
0 1 0 1 0 0 1 0 0 | 0 0 1 0 1 0 0 0 1
0 1 0 1 0 0 0 1 0 | 0 0 1 0 0 1 1 0 0
0 1 0 1 0 0 0 0 1 | 0 0 1 0 0 1 0 1 0
0 1 0 0 1 0 1 0 0 | 0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0 | 0 0 1 0 0 1 0 0 1
```

У цьому випадку матриця *group* буде відповідати комбінаціям всіх рівнів всіх факторів, кількість яких дорівнює  $3 \times 3 = 27$ .

## 12.3. Регресійний аналіз

Функції регресійного аналізу **lsgline**, **refline**, **refcurve** розглянуті в розділі 9, присвяченому статистичній графіці. Функції **lsgline** і **refline** – еквівалентні, вони додають на діаграму з експериментальними точками графік найкращої (у сенсі МНК) лінійної залежності (саме рівняння регресії не виводиться), а функція **refcurve** додає на діаграму з експериментальними точками графік поліноміальної залежності. Функція **refcurve** працює після того, коли вже знайдені коефіцієнти поліноміальної регресії функцією **polyfit** (див. нижче).

<b>POLYFIT</b> Розрахунок коефіцієнтів однофакторної регресійної поліноміальної моделі довільного порядку .....	150
<b>POLYVAL</b> Розрахунок значень однофакторної регресійної поліноміальної моделі довільного порядку .....	153
<b>POLYCONF</b> Розрахунок значень залежної змінної і довірчих інтервалів для однофакторної регресійної поліноміальної моделі довільного порядку .....	157
<b>REGRESS</b> Множинна лінійна регресія .....	160
<b>STEPWISE</b> Покрокова регресія в інтерактивному режимі .....	167
<b>STEPWISEFIT</b> Послідовна регресія .....	171
<b>X2FX</b> Перетворення матриці факторів у матрицю експерименту .....	173
<b>RSTOOL</b> Візуалізація багатовимірної поверхні відгуку .....	174

### **POLYFIT** Розрахунок коефіцієнтів однофакторної регресійної поліноміальної моделі довільного порядку

Дана функція ядра **MatLab** вирішує задачу поліноміальної інтерполяції або апроксимації методом найменших квадратів. Вона знаходить коефіцієнти найкращого полінома. Для обчислення самого полінома можна використовувати функції **polyval** або **polyconf**.

*Синтаксис:*

**p = polyfit(x,y,n)**

## **[p,S] = polyfit(x,y,n)**

Опис:

Функція **p = polyfit(x,y,n)** дозволяє розрахувати коефіцієнти  $p$  поліноміальної регресійної моделі  $n$ -го ступеня, починаючи від вищого ступеня, для вибірки  $(x, y)$  методом найменших квадратів, де  $x$  – незалежна змінна,  $y$  – залежна змінна. Залежна й незалежна змінні задаються як вектори з однаковим числом елементів. Вектор коефіцієнтів  $p$  містить  $(n + 1)$  елементи, розташовані за убутанням ступеня незалежної змінної згідно з формулою  $p(x) = p_1x^n + p_2x^{n-1} + p_3x^{n-2} + \dots + p_{n+1}$ .

Функція **[p,S] = polyfit(x,y,n)** повертає коефіцієнти  $p$  поліноміальної регресійної моделі  $n$ -го ступеня й матриці  $S$  для вибірки  $(x, y)$ . Матриця  $S$  використовується в якості вхідного аргументу функції *polyval* для обчислення границь довірчих інтервалів на розрахункові значення поліноміальної регресійної моделі в заданих точках. Якщо відхилення значень залежної змінної  $y$  від регресійної моделі незалежні й розподілені за нормальним законом з однаковою дисперсією, границі довірчого інтервалу, що отримані функцією **polyval**, включають, як мінімум, 50% розрахованих значень. Функція **polyfit** є стандартною функцією **MatLab**.

Приклади:

1. Розрахунок коефіцієнтів лінійної моделі:

```
>> x=1:1:10;
>> d=normrnd(0,2,1,10);
>> y=x+d;
>> p=polyfit(x,y,1)
p =
    1.286   -1.5736
>> f=p(1)*x+ p(2);
>> plot(x,y,'o',x,f)
```

```
>> grid on, box on
>> hold on
>> y=x;
>> plot(x,y, '-')
>> set(get(gcf,'Currentaxes'),...
    'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfy') % мітка осі OY
>> legend('Данные', 'y = x+d', 'y = x')
```

На рис. 12.50 зображено 10 точок даних, розподілених за нормальним законом, апроксимація лінійної залежності  $y_p = 1.2866x -$

1.5736 і графік функції  $y = x$ . Видні відхилення (похибки) у ту або іншу сторону.

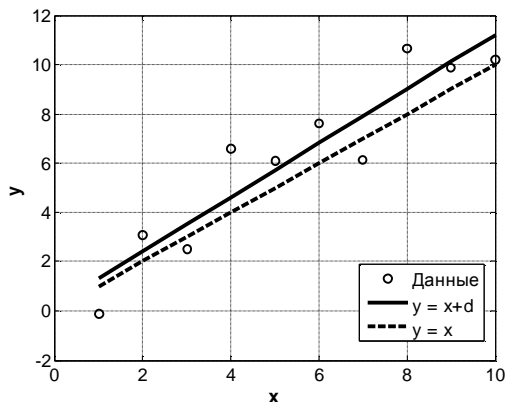


Рис. 12.50. Лінійна апроксимація

2. Розрахунок коефіцієнтів квадратичної моделі:

```
>> x=1:1:10;
```

```
>> d=normrnd(0,2,1,10);
```

```
>> y=x.^2+d;
```

```
>> p=polyfit(x,y,2)
```

```
p =  
0.9192 0.7858 -0.9987
```

Побудова графіків:

```
>> f=p(1)*x.^2+p(2)*x+p(3);
```

```
>> plot(x,y,'o',x,f)
```

```
>> grid on, box on, hold on
```

```
>> y=x.^2;
```

```
>> plot(x,y,'-')
```

```
>> set(get(gcf,'Currentaxes'),...  
'Fontname','Arial Cyr','FontSize',14)  
>> xlabel('\bfx') % мітка осі OX  
>> ylabel('\bfy') % мітка осі OY  
>> legend('Дані','y = x^2+d','y = x^2')
```

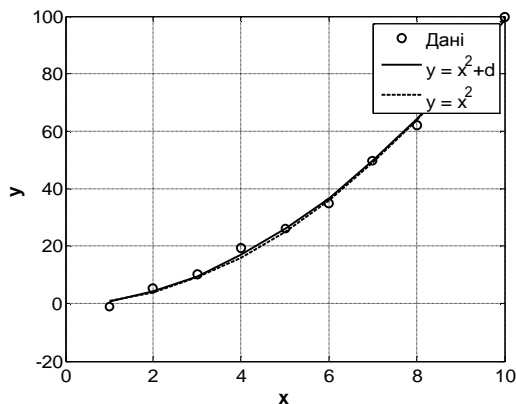


Рис. 12.51. Квадратичная апроксимація

На рис. 12.51 зображено 10 точок даних, графік квадратичної апроксимації  $y = 0.8816x^2 + 1.2455x - 3.1322$ , а також графік функції  $y = x^2$ . Відмінності (похибки) практично незначні.

3. Приклад квадратичної апроксимації з виведенням на екран апроксимуючого поліному у вигляді формули.

```
>> x=[0:10]; % абсциси
>> y=2*x.^2-3*x+5+normrnd(0,1,1,length(x)); % ординати
>> p=polyfit(x,y,2); % квадратична апроксимація
>> fprintf('y(x)=%7.4f*x^2%+7.4f*x%+7.4f\n',p);
y(x)= 1.9783*x^2-2.8261*x+4.9876
```

Отримуємо рівняння квадратичної регресії (а не лише його коефіцієнти).

### **POLYVAL** Розрахунок значень однофакторної регресійної поліноміальної моделі довільного порядку

Дана функція ядра **MatLab** працює в парі з **polyfit**. Вона обчислює значення знайденого полінома в потрібних точках.

*Синтаксис:*

```
Y = polyval(p,X)  
[Y,DELTA] = polyval(p,X,S)
```

*Опис:*

Функція **Y = polyval(p,X)** дозволяє розрахувати значення залежної змінної  $Y$  однофакторної регресійної поліноміальної моделі довільного порядку за заданих значеннях незалежної змінної  $X$ . Коефіцієнти регресійної моделі задаються вектором  $p$ .

Вектор коефіцієнтів  $p$  містить  $(n + 1)$  елементи, розташовані за убутанням ступеня незалежної змінної згідно з формулою:

$$p(x) = p_1x^n + p_2x^{n-1} + p_3x^{n-2} + \dots + p_{n+1},$$

де  $n$  – порядок полінома.

Функція **[Y,DELTA] = polyval(p,X,S)** дозволяє розрахувати значення  $Y$  однофакторної регресійної поліноміальної моделі довільного порядку й границі 95%-го довірчого інтервалу  $DELTA$  у заданих точках  $X$ . Результати розрахунку значень залежної змінної можуть бути

представлені як  $Y \pm DELTA$ . Вхідний аргумент  $S$  розраховується з використанням функції *polyfit* (див. вище). Якщо відхилення значень залежної змінної  $y$  від регресійної моделі незалежні й розподілені за нормальним законом з однаковою дисперсією, границі довірчого інтервалу, що отримані функцією *polyval*, включають як мінімум половину розрахованих значень.

Вхідний аргумент  $X$  може бути заданий як вектор або матриця. Значення регресійної поліноміальної моделі розраховуються для кожного елемента  $X$ . Розмірність  $X$ ,  $Y$  і  $DELTA$  буде однаковою.

Функція **polyval** є стандартною функцією **MatLab**.

*Приклади:*

1. Розрахунок значень лінійної моделі

```
>> x=1:1:10;
>> d=normrnd(0,2,1,10);
>> y=x+d;
>> p=polyfit(x,y,1)
p =
    1.1807   -0.4297
```

```
>> X=0:2:20;
>> Y=polyval(p,X)
Y =
   -0.4297    1.9318    4.2933    6.6548    9.0163   11.3778
   13.7392   16.1007   18.4622   20.8237   23.1852
```

Побудова графіків:

```
>> f=p(1)*X+ p(2);
>> plot(X,Y,'+',x,y,'o',X,f)
>> grid on
>> box on
```

```
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfy') % мітка осі OY
>> legend(' y = polyval(p,x)', 'y = x+d', 'y = p(1)*X+ p(2)')
```

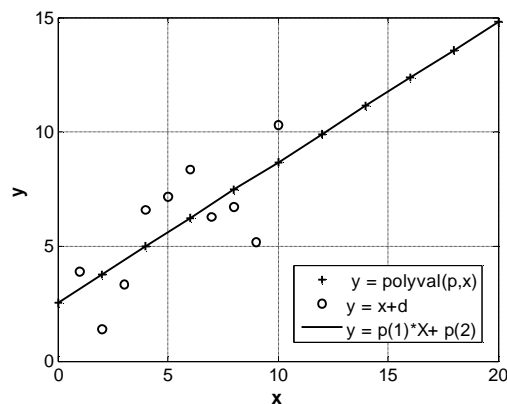


Рис. 12.52. Лінійна апроксимація даних

На рис. 12.52 представлені результати послідовного застосування функцій **polyfit** і **polyval**. Спочатку обчислюються значення за лінійною моделлю з похибкою (кружки на графіку). Далі функцією **polyfit** обчислюються коефіцієнти апроксимуючої прямої  $y = 1.1807x - 0.4297$  (суцільна лінія на графіку). Нарешті функцією **polyval** за цією формулою обчислюються ординати 11 точок на інтервалі  $[0; 20]$  із кроком 2 (хрестики на графіку).

2. Розрахунок значень квадратичної моделі і їх 95%-х довірчих інтервалів

```
>> x=1:1:10;
>> d=normrnd(0,2,1,10);
>> y=x.^2+d;
>> [p S]=polyfit(x,y,2)
p =
    1.0269   -0.0224   -1.0081
S =
    R: [3x3 double]
    df: 7
    normr: 3.7839
>> X = 0:1.2:12;
>> [Y DELTA]=polyval(p,X,S)
>> Y_min=Y-DELTA;
>> Y_max=Y+DELTA;
```

```
Y =
-1.0081  0.4436  4.8527  12.2191  22.5428  35.8238
52.0622  71.2579  93.4109  118.5213  146.5890
DELTA =
    2.2079  1.7648  1.5797  1.5565  1.5797  1.5824
    1.5598  1.5680  1.7177  2.1145  2.7885
>> f=p(1)*X.^2+ p(2).*X+p(3);
>> plot(X,Y,'+', X, Y_min,X,Y_max,x,y,'o',X,f)
>> grid on, box on
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfy') % мітка осі OY
>> legend('y = polyval(p,x)', '95%ymin ', '95%ymax',
'Дані', 'y = p(1)x^2+ p(2)x+p(3)')
```

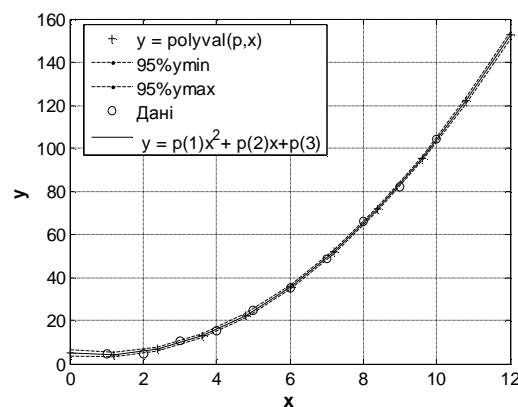


Рис. 12.53. Квадратична інтерполяція з 95% довірчим інтервалом

На рис. 12.53 приведені в графічному вікні результати розрахунків. Їх коментувати складно із-за великого масштабу, який вибрала сама програма. На рис. 12.54 представлені ті ж результати, але в укрупненому вигляді (відредаговані вручну).

На рис. 12.54 видно апроксимуючу криву (суцільна лінія), верхню (пунктирну) і нижню (точкову) границі довірчого інтервалу. Кружками позначені дані, похибки яких розподілені за нормальним законом. Хрестиками позначені точки, що розраховані за апроксимуючим квадратичним поліномом.

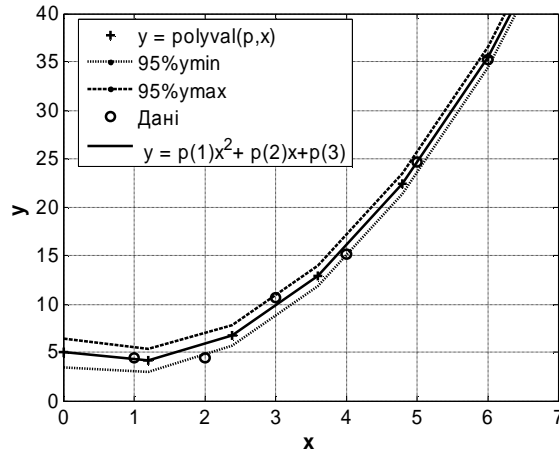


Рис. 12.54. Той же графік, що й на рис. 12.53, але збільшений на інтервалі [0; 6]

3. Розрахунок значень квадратичної моделі і їх 95%-х довірчих інтервалів для кожного значення матриці  $X$ .

```
>> x=1:1:10;
>> d=normrnd(0,2,1,10);
>> y=x.^2/5+d;
>> [p S]=polyfit(x,y,2)
p =
    0.1192    0.7858   -0.9987
S =
    R: [3x3 double]
    df: 7
    normr: 4.5852
>> X=magic(3)
X =
     8     1     6
     3     5     7
     4     9     2
>> [Y DELTA]=polyval(p,X,S)
Y =
    12.9142   -0.0937    8.0061
     2.4312    5.9095   10.3410
     4.0512   15.7257    1.0496
DELTA =
     1.8852    2.2046    1.9175
     1.8852    1.9175    1.8949
     1.8949    1.9598    1.9598
```

У якості матриці даних взято магічний квадрат розміром  $3 \times 3$ .

**POLYCONF** Розрахунок значень залежної змінної і довірчих інтервалів для однофакторної регресійної поліноміальної моделі довільного порядку

Дана функція працює в парі з **polyfit** і розширює можливості стандартної функції **polyval**; вона дозволяє знаходити не лише 95%-і довірчі інтервали для ординат кривої, але й будь-які інші.

*Синтаксис:*

**[Y,DELTA] = polyconf(p,X,S)**  
**[Y,DELTA] = polyconf(p,X,S,alpha)**

*Опис:*

Функція **[Y,DELTA]=polyconf(p,X,S)** дозволяє розрахувати значення залежної змінної  $Y$  однофакторної регресійної поліноміальної моделі довільного порядку в заданих точках  $X$ . Коефіцієнти регресійної моделі задаються вектором  $p$ .

Вектор коефіцієнтів  $p$  містить  $(n + 1)$  елементи, розташовані за збуванням ступеня незалежної змінної згідно за формулою:

$$p(x) = p_1x^n + p_2x^{n-1} + p_3x^{n-2} + \dots + p_{n+1},$$

де  $n$  – порядок полінома. Вхідний аргумент  $S$  розраховується з використанням функції **polyfit**. Результати розрахунку значень залежної змінної можуть бути представлені як  $Y \pm DELTA$ , де  $DELTA$  відповідає 95%-у довірчому інтервалу. Під час розрахунку  $DELTA$  передбачається, що відхилення значень залежної змінної у від регресійної моделі незалежні й розподілені за нормальним законом з однаковою дисперсією.

Функція **[Y,DELTA] = polyconf(p,X,S,alpha)** дозволяє розрахувати границі довірчого інтервалу  $DELTA$  відповідно до рівня значимості  $alpha$ . Довірча ймовірність визначається як  $100(1 - alpha)\%$ .

Вхідний аргумент  $X$  може бути заданий як вектор або матриця. Значення регресійної моделі розраховуються для кожного елемента  $X$ . Розмірність  $X$ ,  $Y$  і  $DELTA$  буде однаковою.

Приклади:

1. Розрахунок значень лінійної моделі і їх 95%-х довірчих інтервалів

(рис. 12.55).

```
>> x=1:1:10;  
>> d=normrnd(0,2,1,10);  
>> y=x+d;  
>> [p S]=polyfit(x,y,1)  
p =  
    1.2866  -1.5736  
S =  
    R: [2x2 double]  
    df: 8  
    normr: 4.7547
```

```
>> X=0:2:20;  
>> [Y,DELTA]=polyconf(p,X,S)  
Y =  
 -1.5736  0.9996  3.5727  6.1458  8.7190  11.2921  
13.8653  16.4384  19.0115  21.5847  24.1578  
DELTA =  
 4.6947  4.3314  4.1158  4.0713  4.2034  4.4965  
 4.9219  5.4488  6.0507  6.7075  7.4045
```

```
>> f=p(1)*X+ p(2);  
>> plot(X,Y,'+',x,y,'o',X,f)  
>> grid on  
>> box on
```

```
>> set(get(gcf,'Currentaxes'),...  
    'Fontname','Arial Cyr','FontSize',14)  
>> xlabel('\bfx') % мітка осі OX  
>> ylabel('\bfy') % мітка осі OY
```

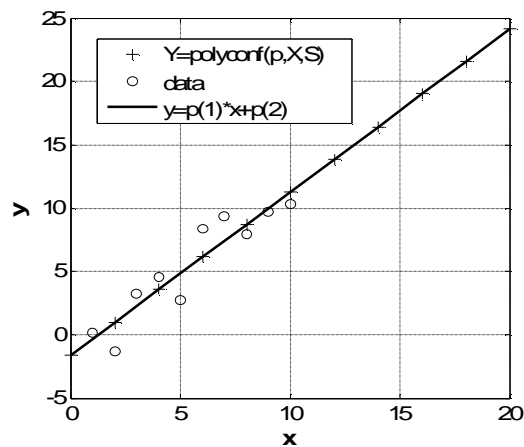


Рис. 12.55. Лінійна апроксимація

2. Розрахунок значень квадратичної моделі і 99%-х довірчих інтервалів (рис. 12.56).

```
>> x=1:1:10;
>> d=normrnd(0,2,1,10);
>> y=x.^2/5+d;
>> [p S] = polyfit(x,y,2)
p =
    0.0941    1.0270   -1.1075
S =
     R: [3x3 double]
     df: 7
     normr: 4.5337
>> X=0:1.2:12; alpha=0.01;
```

```
>> [Y,DELTA]=polyconf(p,X,S,alpha)
Y =
   -1.1075    0.2603    1.8990    3.8085    5.9890    8.4403
   11.1624   14.1555   17.4194   20.9542   24.7598
DELTA =
    9.2577    7.3998    6.6235    6.5266    6.6235    6.6351
    6.5401    6.5745    7.2024    8.8662   11.6920
>> Y_min=Y-DELTA;
>> Y_max=Y+DELTA;
>> f=p(1)*X.^2+ p(2)*X + p(3);
```

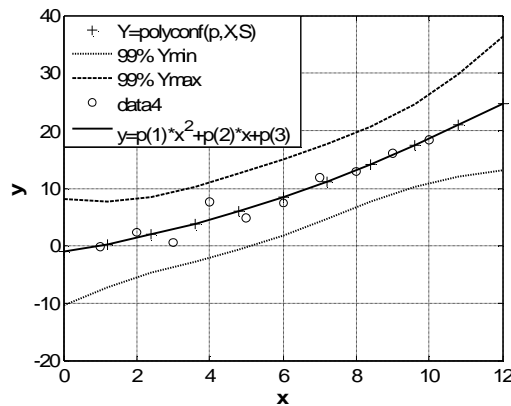


Рис. 12.56. Квадратична апроксимація

На рис. 12.56 представлені дані вибірки (кружечки), графік апроксимуючого поліному (суцільна лінія), графіки верхньої й нижньої 99%-их границь (пунктир) й точки прогнозу (хрестики). Цей рисунок було створено так:

```
>> plot(X,Y,'+', X, Y_min,X,Y_max,x,y,'o',X,f)
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfy') % мітка осі OY
>> grid on, box on
```

3. Розрахунок значень квадратичної моделі і їх 99%-х довірчих інтервалів для матриці значень  $X$ .

```
>> x=1:1:10;
>> d=normrnd(0,2,1,10);
>> y=x.^2/5+d;
>> [p S]=polyfit(x,y,2)
p =
    0.2471   -0.4654    0.0395
S =
     R: [3x3 double]
     df: 7
     normr: 5.5348
```

```
      8    1    6
      3    5    7
      4    9    2
>> alpha=0.01;
>> [Y,DELTA]=polyconf(p,X,S,alpha)
Y =
   12.1287   -0.1788    6.1416
    0.8670    3.8893    8.8881
    2.1311   15.8635    0.0970
DELTA =
```

```
>> X=magic(3)
X =
```

```
7.9637  9.3126  8.1002
7.9637  8.1002  8.0043
8.0043  8.2786  8.2786
```

## **REGRESS** Множинна лінійна регресія

Будується лінійна регресійна модель: методом найменших квадратів знаходить вектор параметрів  $B$  розміром  $p \times 1$  з рівняння  $Y = XB + E$ , де  $Y$  – вектор спостережень розміром  $n \times 1$ ;  $X$  – матриця  $n \times p$ , стовпці якої представляють значення базисних функцій у заданих точках; а  $E$  – вектор  $n \times 1$  нормально розподілених центрованих незалежних величин з однаковою дисперсією.

*Синтаксис:*

```
b = regress(y,X)
```

```
[b,bint,r,rint,stats] = regress(y,X)
```

```
[b,bint,r,rint,stats] = regress(y,X,alpha)
```

*Опис:*

Функція **b = regress(y,X)** призначена для розрахунку точкових оцінок коефіцієнтів  $b$  лінійного рівняння регресії.

Розмірності векторів значень залежної змінної  $y$  і випадкових похибок  $\epsilon$  становлять  $n \times 1$ , де  $n$  – кількість спостережень. Розмірність матриці  $X$  дорівнює  $n \times p$ , де  $p$  – кількість незалежних (пояснювальних) змінних. Стовпці матриці  $X$  відповідають незалежним змінним, рядки – спостереженням. Розмірність вектора коефіцієнтів  $b$  лінійної регресійної моделі дорівнює  $p \times 1$ . Коефіцієнти множинної лінійної регресійної моделі у векторі  $b$  розташовуються за зростанням номера незалежних змінних.

Функція **[b,bint,r,rint,stats] = regress(y,X)** повертає:  $b$  – вектор точкових оцінок коефіцієнтів лінійного рівняння регресії,  $bint$  – матрицю інтервальних оцінок параметрів лінійної регресії,  $r$  – вектор залишків моделі,  $rint$  – матрицю 95%-х довірчих інтервалів,  $stats$  – структуру, що

містить значення статистик з відповідними їм F-статистикою й рівнем значимості  $p$  для регресійної моделі.

Розмірність матриці  $bint$  становить  $p \times 2$ , де перший стовпець матриці задає нижню границю 95%-го довірчого інтервалу, другий – верхню границю 95%-го довірчого інтервалу для  $b$ . Кількість елементів вектора  $r$  дорівнює  $n$ .

Розмірність матриці  $rint$  становить  $n \times 2$ , де перший і другий стовпці використовуються для завдання нижньої й верхньої границь 95%-го довірчого інтервалу на розрахункові значення по кожному з  $n$  спостережень.

Функція **[b,bint,r,rint,stats] = regress(y,X,alpha)** в параметрі  $alpha$  дозволяє задати величину рівня значимості. Рівень значимості використовується для розрахунку границь довірчих інтервалів  $bint$  і  $rint$  з довірчою ймовірністю обумовленою як  $100(1 - alpha)\%$ . Значення  $alpha = 0.2$  буде відповідати 80%-м границям довірчих інтервалів  $bint$  і  $rint$ .

Якщо модель включає вільний член (константу), у матрицю  $X$  потрібно включити стовпець із одиниць. F-статистика й  $p$ -значення обчислюються в припущенні, що в моделі завжди є вільний член, і за його відсутності будуть помилковими. Величина  $R^2$ -статистики дорівнює одиниці мінус відношення суми квадратів похибок до загальної суми квадратів. Для моделей без константи ця величина може виявитися від'ємною, що вказує на невідповідність такої моделі даним.

Якщо стовпці  $X$  лінійно-залежні, функція  $regress$  задає максимально можливе число елементів  $b$  нульовими, щоб одержати базисне рішення. У цьому випадку відповідні елементи  $bint$  також будуть нульовими.

Функція  $regress$  розглядає значення NaN в  $X$  або в  $y$  як відсутні, і видаляє їх перед аналізом.

Приклади:

1. Розглянемо лінійну регресійну модель вигляду  $y = 10 + X + \varepsilon$ , де  $\varepsilon \gg N(0; 0.1)$ .

1.1. Моделювання матриці значень незалежної змінної

```
>> X=[ones(10,1) (1:10)']
X =
     1     1     1     1
     1     2     3     4
     1     3     4     5
     1     4     5     6
     1     5     6     7
     1     6     7     8
     1     7     8     9
     1     8     9    10
     1     9    10    11
     1    10    11    12
```

1.2. Моделювання вектора значень залежної змінної

```
>> y=X*[10;1]+normrnd(0,0.1,10,1)
y =
 10.9600 12.0690 13.0816 14.0712 15.1290 16.0669 17.1191 17.8798 18.9980 19.9843
```

1.3. Розрахунок параметрів лінійної регресійної моделі і їх 95%-вих границь довірчих інтервалів.

```
>> [b,bint]=regress(y,X,0.05)
b =
 10.0761
  0.9927
bint =
  9.9505 10.2016
  0.9725  1.0129
```

Якщо лінійна модель розглядається у вигляді  $y_p = b_0 + b_1 x$ , то одержимо рівняння лінії регресії:  $y_p = 10.0761 + 0.9927 x$  і 95%-ві довірчі інтервали для коефіцієнтів:  $9.9505 \leq b_0 \leq 10.2016$ ,  $0.9725 \leq b_1 \leq 1.0129$ .

1.4. Графічне представлення вибірових даних, ліній регресії з границями 95%-х довірчих інтервалів:

```
>> Xx = 1:1:10;
>> Yy = b(1)+ b(2).*Xx;
>> Yn = bint(1,1)+ bint(2,1).*Xx;
>> Yv = bint(1,2)+ bint(2,2).*Xx;
>> plot(Xx,Yy,'-', Xx,Yn,'g--', Xx,Yv,'g--',X(:,2),y,'o')
>> grid on, box on
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx'), ylabel('\bfy') % мітка осей OX, OY
```

На рис. 12.57 представлено 10 точок вибірки, пряма лінія регресії з 95%-ми довірчими границями.

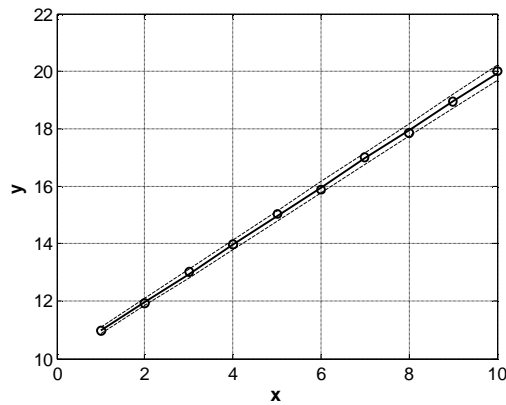


Рис. 12.57. Прямая лінія регресії з довірчими границями

1.5. Розрахунок параметрів лінійної регресійної моделі, їх 99%-х границь довірчих інтервалів, вектора значень залишків, їх 99%-х границь довірчих інтервалів і структури *stats*.

```
>> [b,bint,r,rint,stats] = regress(y,X,0.01)
b =
    10.0761
     0.9927
bint =
    9.8934  10.2587
    0.9633   1.0221
r =
   -0.1088
    0.0075
    0.0274
    0.0243
    0.0895
    0.0346
    0.0942
-0.1379
-0.0123
-0.0187

rint =
   -0.2944   0.0769
   -0.2401   0.2552
   -0.2298   0.2846
   -0.2409   0.2896
   -0.1563   0.3353
   -0.2325   0.3018
-0.1447  0.3330
-0.3297  0.0539
-0.2596  0.2350
-0.2487  0.2114
stats =
    1.0e+004 *
    0.0001  1.2801  0.0000  0.0000
```

За умови рівня значимості 0.01 довірчі інтервали для коефіцієнтів лінійної моделі становлять  $9.8934 \leq b_0 \leq 10.2587$ ,  $0.9633 \leq b_1 \leq 1.0221$ . Значення коефіцієнтів залишаються тими ж. У цьому прикладі результат не виводимо в графічне вікно, тому що він практично не відрізняється від рис. 12.57.

1.6. Графічне представлення вектора залишків і їх границь довірчих інтервалів (рис. 12.58).

```
>> plot(X(:,2),r,'ko', X(:,2),rint(:,1),'k+', X(:,2),rint(:,2),'k+')
>> grid on, box on
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
```

```
>> title('\bfГрафік залишкових похибок')
>> xlabel('\bfНомер точки'), ylabel('\bfЗалишкові похибки')
```

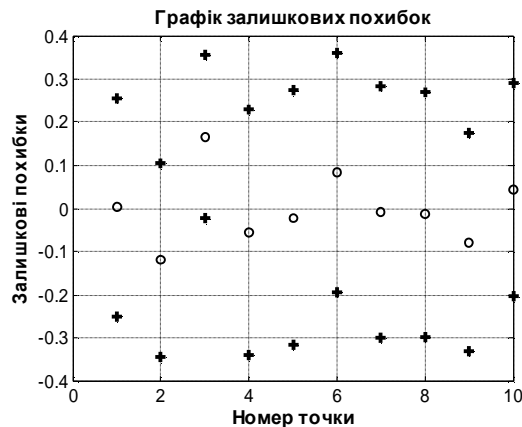


Рис. 12.58. Графік залишкових похибок і їх довірчі інтервали

2. Розглянемо лінійну регресійну модель вигляду

$$y = AX_1 + BX_2 + C + \varepsilon,$$

де  $\varepsilon \approx N(0; 0.1)$ ;  $A, B, C$  – параметри лінійної регресійної моделі,  $A = 0.2$ ,  $B = 0.5$ ,  $C = 1.5$ ;  $X_1, X_2$  – незалежні змінні.

2.1. Моделювання матриці й вектора значень незалежних і залежної змінних

```
>> X1=unidrnd(10,10,1);
>> X2=unidrnd(20,10,1);
>> X=[ones(10,1) X1 X2];
>> y=X*[1.5;2.5; 1.2]+normrnd(0,0.1,10,1);
```

2.2. Розрахунок параметрів лінійної регресійної моделі, їх 99%-х границь довірчих інтервалів, вектора значень залишків, їх 99%-х границь довірчих інтервалів і структуру *stats*.

```
>> [b,bint,r,rint,stats] = regress(y,X,0.01) % alpha=0.01
```

b =		-0.0310		-0.3535	0.2914
1.6314		0.1620		-0.1498	0.4738
2.4796		-0.0151		-0.2941	0.2639
1.1967		0.1145		-0.2165	0.4456
bint =		0.0558		-0.3040	0.4155
0.8278	2.4350	-0.1399		-0.3624	0.0826
2.4181	2.5411	-0.1350		-0.4452	0.1752

1.1626	1.2307	0.0246	-0.3527	0.4019
r =		rint =	stats =	
-0.0446		-0.3598	1.0e+004 *	
0.0088		-0.3448	0.0001	1.0541 0.0000 0.0000
		0.2706		
		0.3624		

2.3. Графічне представлення вибірових даних і регресійної моделі разом з границями довірчих інтервалів.

```

>> [Xx1 Xx2]=meshgrid([1:1:10],[1:2:20]);
>> Yy=b(1)+ b(2).*Xx1+ b(3).*Xx2;
>> Yn=bint(1,1)+ bint(2,1).*Xx1+ bint(3,1).*Xx2;
>> Yv=bint(1,2)+ bint(2,2).*Xx1+ bint(3,2).*Xx2;
>> mesh(Xx1, Xx2, Yy)
>> hold on
>> plot3(X1,X2,y,'o')
>> hidden off, box on
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx1'), ylabel('\bfx2'), zlabel('\bfz')

```

На рис. 12.59 зображені точки вибірки й площина регресії. Щоб проглядалися всі точки, площина зроблена прозорою, залишена тільки координатна сітка (каркас). Оскільки точки мають розкид, то на графіку з непрозорою площиною (>> **hidden on**) будуть видні лише ті точки, які перебувають у полі зору над площиною регресії, як показано на рис. 12.60.

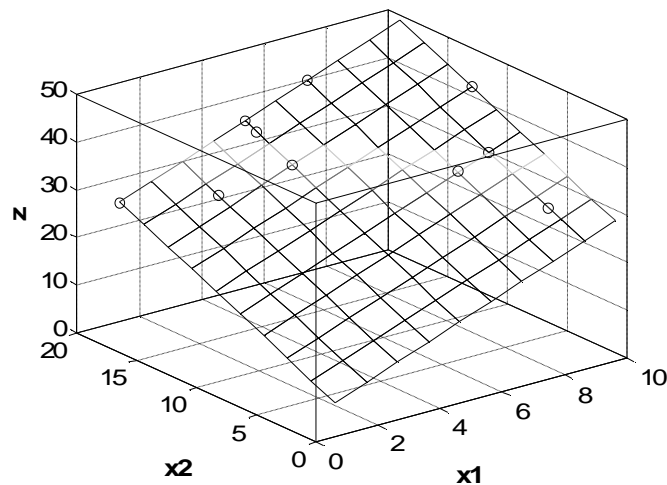


Рис. 12.59. Графік лінійної регресійної моделі від двох змінних

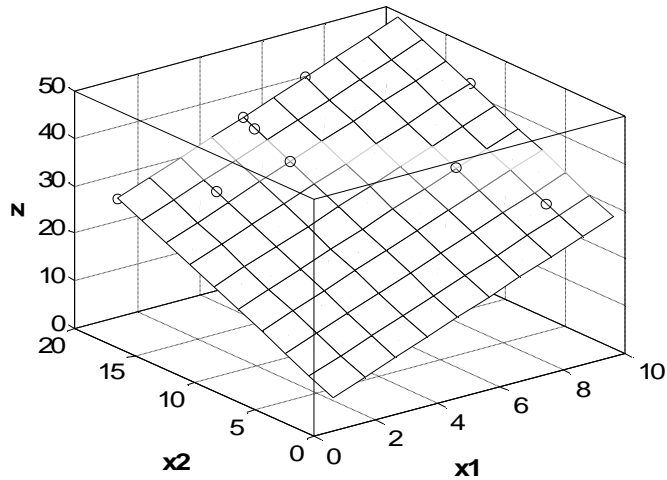


Рис. 12.60. Та ж площина регресії, але непрозора

Додамо на графік рис. 12.60 довірчі площини нижньої й верхньої границь 95%-го довірчого інтервалу й зробимо їх прозорими.

```
>> mesh(Xx1, Xx2, Yn)
>> hold on
>> mesh(Xx1, Xx2, Yv)
>> hidden off
>> hold off
```

На рис. 12.61 зображені площина регресії й довірчі 95%-ві границі. Оскільки довірчий інтервал невеликий, різниці цих площин малопомітні на графіку. Тому можна або повернути графік і знайти більш зручну точку огляду, або спробувати змінити масштаб.

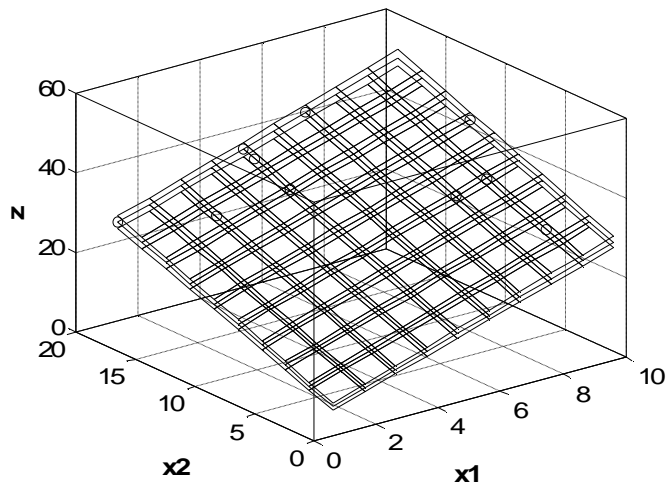


Рис. 12.61. Площина регресії й довірчі площини

На рис. 12.62 показані всі три площини з іншої точки огляду.

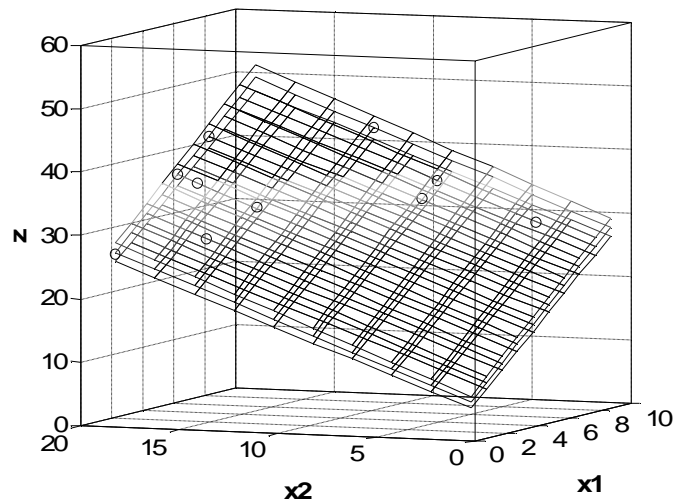


Рис. 12.62. Ті ж площини з іншої точки огляду

2.4. Графічне представлення вектора залишків і границь їх довірчих інтервалів (рис. 12.63).

```
>> plot(1:10,r,'ko', 1:10,rint(:,1),'k+', 1:10,rint(:,2),'k+')  
>> grid on, box on  
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)  
>> xlabel('\bfNumber Point'), ylabel('\bfResidual Error'), title('\bfError Plot')
```

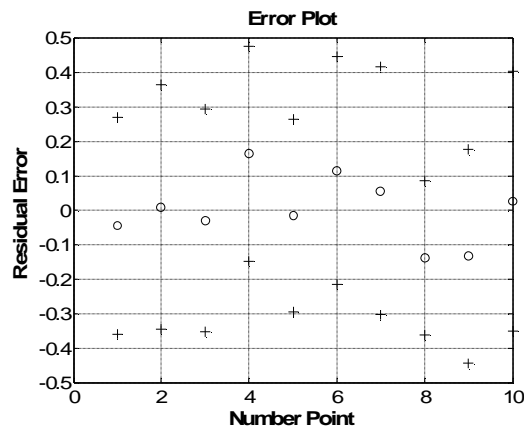


Рис. 12.63. Графічне представлення залишків і їх довірчих границь

### **STEPWISE** Покрокова регресія в інтерактивному режимі

Створюється графічний інтерфейс користувача для покрокової регресії.

Синтаксис:

**stepwise(x,y)**  
**stepwise(x,y,inmodel,penter,premove)**

Опис:

Функція **stepwise(x,y)** виводить інтерактивний інструментарій для послідовної побудови регресійної моделі за значеннями функції  $y$ , використовуючи різні підмножини стовпців матриці  $x$ . Створене вікно є інтерактивним і не закривається. За створенням вікна в модель жодні стовпці ще не включені, як показано на рис.12.64.

У лівому верхньому вікні "Коефіцієнти і їх розкид" (Coefficients with Error Bars) показані змінні, які можна додавати в модель або виключати з неї. Включені в модель змінні відзначаються синім кольором, а невключені – червоним. Горизонтальні лінії показують 90%-і довірчі інтервали (кольорові лінії) і 95%-і (чорні). У правому вікні відображаються значення коефіцієнтів,  $t$ -статистики й  $p$ -значення. Клацання мишкою на них також приводить до включення (вилучення) параметрів у модель. Мітка Next step (наступний крок) показує, що ще можна додати в модель. Фактори можна додавати в модель послідовно (кнопка **Next Step**) або всі відразу (кнопка **All Steps** – усі кроки).

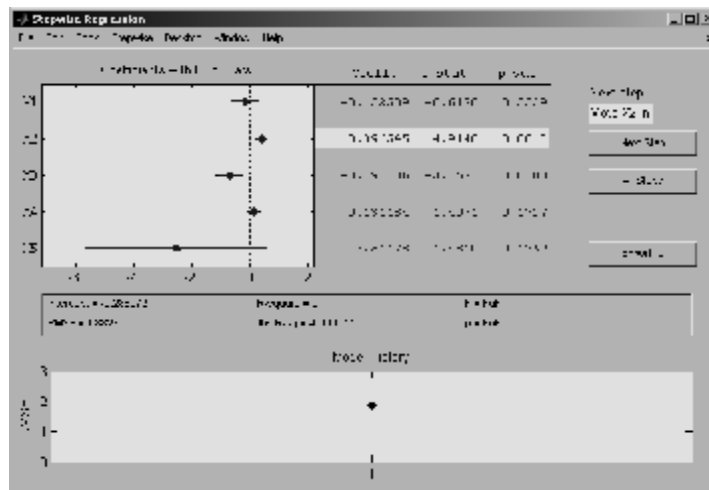


Рис. 12.64. Вікно інтерактивної послідовної регресії в момент завантаження

У нижній частині основного вікна (рис. 12.64) відображається історія побудови моделі. Мітки осі абсцис – номери кроків побудови моделі. По

осі ординат показується середньоквадратична похибка. За правильної побудови моделі ця похибка повинна убувати.

На кожній точці в цьому вікні можна клацнути мишкою, при цьому відкриється таке ж нове вікно, але в ньому історія почне відлічуватися від обраного кроку.

Кнопкою **Export** можна експортувати в робочий простір **MatLab** наступні змінні, при причому можна змінити їх імена (рис. 12.65):

коефіцієнти моделі (вектор-стовпець);

їх довірчі інтервали (матриця із двох стовпців);

включені фактори (вектор-рядок);

виключені фактори (вектор-рядок);

статистичні дані моделі (структура з полями, що повторюють величини в середній частині вікна);

таблицю коефіцієнтів (структура з полями, що повторюють величини в правій верхній частині вікна);

історію побудови моделі (структура з 3-ма полями: середньоквадратичні похибки, номери включених змінних і порядок їх додавання-видалення).

Функція **stepwise(X,y,inmodel,penter,remove)** в 3-му, 4-му і 5-му аргументах дозволяє встановити параметри вікна в момент його створення. Аргумент *inmodel* – логічний вектор, довжина якого дорівнює кількості стовпців матриці  $X$ , або вектор з натуральних чисел від 1 до кількості стовпців матриці  $X$ , що задає номери змінних, включених у початкову модель. За замовчуванням ніякі стовпці  $X$  у початкову модель не включаються. Аргумент *penter* задає максимальне  $p$ -значення, яке використовується у вікні для рекомендації додавання фактора до моделі (за замовчуванням 0.05). Аргумент *remove* задає мінімальне  $p$ -



Рис. 12.65. Вікно експорту результатів послідовної регресії

значення, яке використовується у вікні для рекомендації видалення фактора з моделі (за замовчуванням 0.1).

Дана функція розглядає не числові значення NaN в  $X$  або в  $y$  як відсутні дані, і виключає відповідні рядки з розгляду до побудови моделі.

*Приклад:*

Розглядається регресійна модель

$$y = AX_1 + BX_2 + CX_3 + DX_4 + EX_5 + \varepsilon,$$

де  $\varepsilon \approx N(0; 0.05)$ ;  $A, B, C, D, E$  – параметри лінійної регресійної моделі,  $A = -0.2, B = 0.3, C = -0.5, D = 0.01, E = 0.001$ ;  $X_1, X_2, X_3, X_4, X_5$  – незалежні змінні.

```
% Моделювання матриці значень незалежних змінних
>> x1 = unidrnd(10,10,1);
>> x2 = unidrnd(15,10,1);
>> x3 = unidrnd(8,10,1);
>> x4 = normrnd(0,8,10,1);
>> x5 = weibrnd(1,2,10,1);
>> x = [x1 x2 x3 x4 x5]
% Моделювання вектора значень залежної змінної
>> y = -0.2*x1 + 0.3*x2 - 0.5*x3 + 0.01*x4 + 0.001*x5 + normrnd(0,0.5,10,1);
% Інтерактивний покроковий регресійний аналіз
>> stepwise(x,y)
```

З'являється вікно інтерактивної послідовної регресії в момент заантаження (рис. 12.64), в якому послідовними кроками в модель підключаються значимі члени (за критерієм Стьюдента, або *p-val*).

Фінальне вікно показане на рис. 12.66. На кожній точці в цьому вікні можна клацнути мишкою, при цьому відкриється таке ж нове вікно, але в ньому історія почне відлічуватися від обраного кроку.

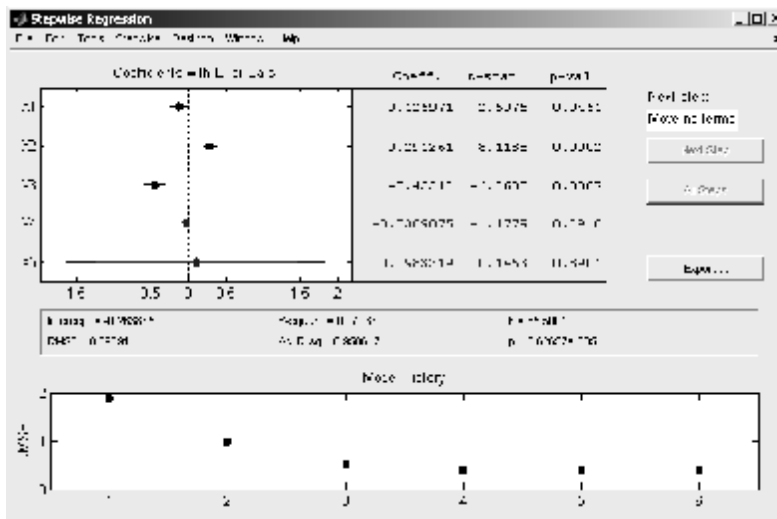


Рис. 12.66. Вікно інтерактивної послідовної регресії після добору моделі

### **STEPWISEFIT** Послідовна регресія

Будується послідовна регресія, але, на відміну від попередньої функції, без створення користувальницького інтерфейсу.

*Синтаксис:*

**b = stepwisefit(x,y)**

**[b,se,pval,inmodel,stats,nextstep,history] = stepwisefit(...)**

**[...] = stepwisefit(x,y,param1,value1,param2,value2,...)**

*Опис:*

Функція **b = stepwisefit(x,y)** провадить послідовну побудову регресійної моделі за значеннями функції відгуку  $y$  і матриці  $x$ , стовпці якої представляють фактори, що враховуються (значення базисних функцій в експериментальних точках). Результат – вектор  $b$  оцінок коефіцієнтів регресії для всіх стовпців  $x$ . Значення  $b$  для тих стовпців, що не включені в модель – це ті значення, які ми одержали б, включаючи їх у модель. За замовчуванням у командному вікні **MatLab** виводиться інформація про хід послідовної регресії: які фактори включаються й у якому порядку; знайдені значення коефіцієнтів; похибки поточної моделі; статус кожного фактора (включений він у модель, чи ні) і  $p$ -значення для коефіцієнтів  $b$ .

Функція **[b,se,pval,inmodel,stats,nextstep,history] = stepwisefit(...)** повертає наступні додаткові результати:

*se* – вектор стандартних помилок для коефіцієнтів *b*;

*pval* – вектор *p*-значень для перевірки значимості впливу факторів;

*inmodel* – логічний вектор, довжина якого дорівнює кількості стовпців *x*, що показує, які фактори включені в кінцеву модель;

*stats* – структуру, що містить додаткову статистичну інформацію;

*nextstep* – рекомендований наступний крок: номер фактора, який потрібно включити на наступному кроці або 0, якщо нічого більше не потрібно підключати;

*history* – структуру, що містить інформацію про історію проведення послідовної регресії.

Функція **[...] = stepwisefit(X,y,param1,value1,param2,value2,...)** визначає додаткові аргументи у вигляді пар "ім'я – значення". Можливі імена (символьні рядки) і відповідні їм значення:

*'inmodel'* – логічний вектор, що задає фактори, які обов'язково включаються в початкову модель (за замовчуванням усі 0);

*'penter'* – задає максимальне *p*-значення, яке використовується для рекомендації додавання факторів до моделі (за замовчуванням 0.05);

*'premove'* – задає мінімальне *p*-значення, яке використовується для рекомендації видалення фактора з моделі (за замовчуванням 0.1);

*'display'* – *'on'* (за замовчуванням), якщо потрібно друкувати в команднім вікні **MatLab** інформацію про хід послідовної регресії, і *'off'*, якщо непотрібно;

*'maxiter'* – максимальне число кроків послідовної регресії (за замовчуванням не обмежується);

*'keep'* – логічний вектор, який задає фактори, що втримуються в початковому стані (за замовчуванням усі 0);

*'scale'* – *'on'*, якщо кожний стовець матриці *X* потрібно масштабувати до одиничного стандартного відхилення перед

включенням у модель, і 'off', якщо непотрібно (за замовчуванням 'off').

Приклад:

Розглянемо модель:  $y = 3 \cdot x^2 - 5x + 2 + \varepsilon$ .

```
>> x=[0:10]'; % точки
>> n=length(x); % кількість точок
>> y=3*x.^2-5*x+2+normrnd(0,1,n,1); % ординати
>> [b,se,pval,stats]=stepwisefit([x,x.^2],y)
Initial columns included: none
Step 1, added column 2, p=6.11911e-013
Step 2, added column 1, p=3.82771e-006
Final columns included: 1 2
   'Coeff'      'Std.Err.'  'Status'      'P'
   [-4.8373]   [ 0.4351]   'In'          [3.8277e-006]
   [ 2.9775]   [ 0.0419]   'In'          [1.7140e-012]
b =
-4.8373
 2.9775
```

## **X2FX** Перетворення матриці факторів у матрицю експерименту

Перетворюється матриця рівнів факторів у матрицю експерименту, яку можна подавати на вхід різних функцій дисперсійного й регресійного аналізу.

*Синтаксис:*

**D = x2fx(x)**  
**D = x2fx(x,model)**

*Опис:*

Функція **D = x2fx(x)** додає ліворуч до матриці **X** стовпець одиниць і тим самим перетворює її в матрицю експерименту **D** для лінійної адитивної моделі зі сталим доданком (константою).

Функція **D = x2fx(x,model)** у другому аргументі *model* дозволяє задати вид моделі для побудови матриці експерименту. Аргумент *model* має бути рядком символів. Можливі значення:

*'linear'* – лінійна модель, що включає константу й лінійні члени, що складаються (значення за замовчуванням);

'interaction' – включає константу, лінійні члени і добутки змінних, що складаються;

'quadratic' – 'interaction' плюс квадрати змінних;

'purequadratic' – включає константу, лінійні й квадратичні доданки.

Порядок стовпців для найбільш загальної квадратичної моделі 'quadratic' наступний.

Константа (стовпець із одиниць).

Лінійні члени (стовпці матриці  $X$  з номерами 1, 2, ...,  $k$ ).

Взаємодії факторів, що сформовані із добутків елементів стовпців  $X$  у порядку: (1, 2); (1, 3); ... ; (1,  $k$ ); (2, 3); ...; ( $k-1$ ,  $k$ ).

Квадратичні члени – квадрати елементів стовпців матриці  $X$  з номерами 1, 2, ...,  $k$ .

Інші моделі: 'linear', 'interaction' і 'purequadratic' є підмоделями повної моделі 'quadratic', і порядок стовпців у них такий же.

Можна задавати й більш складні моделі: поліноміальні будь-якого ступеня за будь-якими змінними. Для цього аргумент **model** має бути матрицею, кожний рядок якої представляє один фактор, що враховується. Довжина рядка матриці *model* в цьому випадку дорівнює кількості стовпців  $X$ , а елементи цього рядка (цілі позитивні числа) – ступені, у які має бути піднесені відповідні стовпці  $X$ . Іншими словами, у цьому випадку  $D(i,j)=prod(x(i,:).^model(j,:))$ .

Приклад:

```
>> x=[1 2 3;4 5 6]' % матриця рівнів факторів
>> d=x2fx(x,'quadratic') % матриця аргументів
x =
  1  4
  2  5
d =
  3  6
  1  1  4  4  1  16
  1  2  5  10  4  25
  1  3  6  18  9  36
```

## **RSTOOL** Візуалізація багатовимірної поверхні відгуку

Графічний інтерфейс користувача для добору й візуалізації багатовимірної поверхні відгуку.

*Синтаксис:*

```
rstool(x,y)  
rstool(x,y,'model')  
rstool(x,y,'model',alpha,'xname','yname')
```

*Опис:*

Функція **rstool(x,y)** призначена для розрахунку параметрів і побудови графіків множинної лінійної регресійної моделі для матриці незалежних змінних  $x$  і вектора значень залежної змінної  $y$ . Стовпці матриці  $x$  задають значення незалежних змінних. Рядки матриці  $x$  відповідають спостереженням. Кількість рядків  $x$  і елементів вектора  $y$  має бути однаковими.

Багатовимірну залежність  $y = f(X_1, X_2, X_3)$  графічно представляють серією одновимірних графіків за кожним аргументом окремо при фіксованих значеннях інших аргументів. Ці фіксовані значення можна змінювати в інтерактивному режимі, спостерігаючи за відповідними змінами графіків.

Детально графічне вікно буде розглянуте нижче після опису всіх варіантів функції **rstool**, а зараз лише відмітимо, що результати розрахунків можна експортувати з інтерактивного вікна в робочий простір **MatLab** натискаючи на кнопку **Export** в цьому вікні.

Вікно експорту поведено на рис. 12.67.

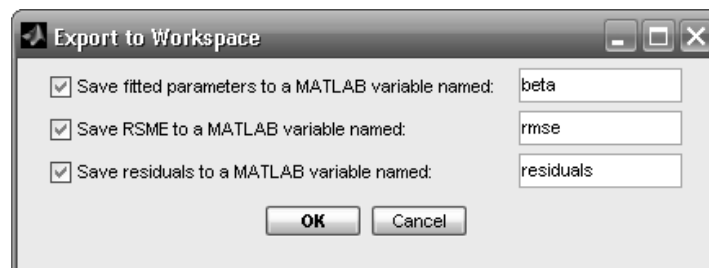


Рис. 12.67. Вікно експорту результатів у робочий простір **MatLab**

Можна експортувати в робочий простір **MatLab** наступні величини:  
знайдені параметри моделі (*fitted parameters*);  
середньоквадратичну похибку (*RSME*);  
залишки моделі (*residuals*).

Якщо в цьому є потреба, змінним можна дати нові імена.

Після натискання кнопки **OK** у вікні Експорту з'являється повідомлення: Variables have been created in the current workspace (Значення отриманих після обчислення змінних розташовані в поточному вікні Workspace).

Функція **rstool(x,y,'model')** в параметрі *'model'* дозволяє користувачеві задати вид початкової регресійної моделі, що буде відображено в графічному вікні множинної регресії. Параметр *'model'* може приймати наступні значення: *'interaction'*, *'quadratic'*, *'purequadratic'*. За замовчуванням *'model' = 'linear'*.

Опис регресійних моделей наведено в таблиці 12.10.

Таблиця 12.10

### Допустимі види регресійних моделей

Значення <i>'model'</i>	Склад ефектів множинної регресійної моделі
<i>'linear'</i>	Лінійна модель, що включає лінійні ефекти факторів і константу (вільний член). Ухвалюється за замовчуванням
<i>'interaction'</i>	Лінійна модель, що включає лінійні ефекти, ефекти взаємодій факторів і константу
<i>'quadratic'</i>	Квадратична модель, що включає лінійні ефекти, квадратичні ефекти, ефекти взаємодій факторів і константу.
<i>'purequadratic'</i>	Квадратична модель, що включає квадратичні й лінійні ефекти факторів, константу
<i>'User Specified'</i>	Модель, що визначена користувачем за допомогою вхідного параметра <i>'model'</i>

Функція **rstool(x,y,'model',alpha)** в параметрі *alpha* дозволяє задати рівень значимості. Довірча ймовірність для границь довірчого інтервалу визначається як  $100(1 - \alpha)\%$ . Наприклад, за  $\alpha = 0.01$  довірча ймовірність буде дорівнювати 99%.

Якщо в якості *y* задати матрицю, то кожний її стовець буде трактуватися як окрема залежна змінна від усіх незалежних змінних *x* (стовпців матриці *x*). Задача множинної регресії буде вирішуватися по черзі спочатку для першої залежної змінної (першого стовпця *y*), далі для другої залежної змінної і т.д. Графічне представлення такої задачі буде мати вигляд матриці графіків, де "Predicted Y1" є першим стовпцем

матриці  $y$ , "Predicted Y2" – другим стовпцем  $y$  і т.д. Усі графіки будуть побудовані в однаковому масштабі для незалежних змінних  $x$ . Зміна значення однієї з незалежних змінних мишкою у вікні графіка або введенням її значення у відповідне поле введення приведе до автоматичного перерахування точкових і інтервальних оцінок усіх залежних змінних. Вектори параметрів регресійних моделей *Parameters* і залишків *Residuals* в ході експорту будуть перетворені в матриці із числом стовпців, що дорівнює кількості регресійних моделей або залежних змінних  $y$ . Перший стовпець *Parameters* і *Residuals* буде відповідати  $y(:,1)$  тощо. Скаляр *RMSE* буде перетворений у вектор-рядок із числом елементів, що дорівнює  $size(y,2)$ . В разі зміни виду регресійної моделі нова модель буде застосована до всіх залежностей, і зазначеним вище чином будуть перераховані експортовані параметри.

Функція **`rstool(x,y,'model',alpha,'xname','yname')`** у параметрах *'xname'* і *'yname'* дозволяють задати мітки на графіках для незалежних і залежні змінних. Параметр *'xname'* і задається як вектор рядкових змінних. Параметр *'yname'* визначається як рядкова змінна.

В інтерактивному графічному вікні суцільними лініями зображені графіки регресійної моделі для кожної незалежної змінної, а пунктирними лініями – відповідні границі довірчого інтервалів на лінії регресії. Значення фіксованих аргументів відображаються у маленьких віконцях і вертикальними пунктирними лініями. Змінюючи мишкою положення вертикальної пунктирної лінії або вводячи нові значення фіксованих незалежних змінних у поля введення (маленкі віконця) під графіками, можна розрахувати нові точкові й інтервальні оцінки залежної змінної в інтерактивному режимі. Результати розрахунків для фіксованої точки (всі  $X_i = Const$ ) у вигляді (наприклад, див. рис. 12.68)  $Predicted Y1 = 74.17 \pm 10.74$  будуть автоматично перераховуватися після зміни фіксованого значення хоча б однієї змінної.

*Приклади:*

1. Розглядається повна квадратична регресійна модель виду  $y = A(X_1)^2 + B(X_2)^2 + CX_1 + DX_2 + EX_1X_2 + F + \varepsilon$ , де  $A = 1.2$ ,  $B = 2.5$ ,

$C = 1.5$ ,  $D = -1.3$ ,  $E = -0.89$ ,  $F = 0$  – параметри квадратичної регресійної моделі;  $X_1$ ,  $X_2$  – незалежні змінні;  $\varepsilon \approx N(0; 2)$ .

Моделювання матриці значень незалежних змінних

```
>> X1 = unidrnd(5,10,1);
>> X2 = unidrnd(7,10,1);
>> X=[X1.^2 X2.^2 X1 X2 X1.*X2]
X =
```

25	25	5	5	25
4	36	2	6	12
16	49	4	7	28

```

9 36 3 6 18
25 4 5 2 10
16 9 4 3 12
9 49 3 7 21
1 49 1 7 7
25 9 5 3 15
9 49 3 7 21
```

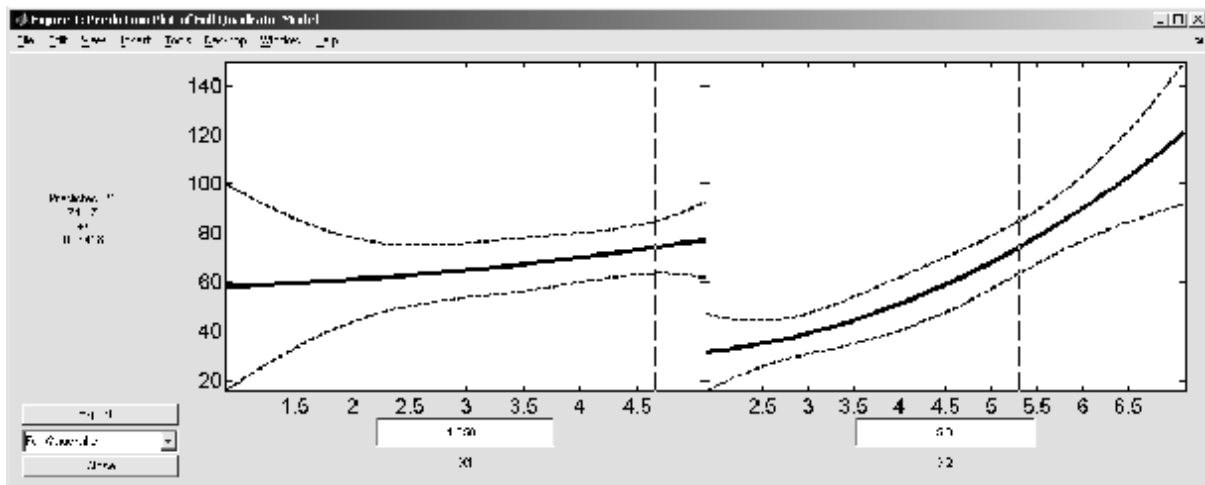
Моделювання вектора значень залежної змінної

```
>> y = X*[1.2; 2.5; 1.5; -1.3; -0.89] + normrnd(0,2,10,1)
y =
```

70.3849	33.7071
75.9888	35.5018
113.9307	112.3883
82.0554	109.7947
	43.4046
	110.3593

Інтерактивний множинний регресійний аналіз

```
>> rstool([X1 X2],y,'quadratic')
```



**Рис. 12.68. Графічне вікно інтерактивного розрахунку й результату розрахунку параметрів множинної регресії**

З'являється графічне вікно (рис. 12.68). Розраховані значення параметрів можна екпортувати у поточне вікно Workspace (рис. 12.69).

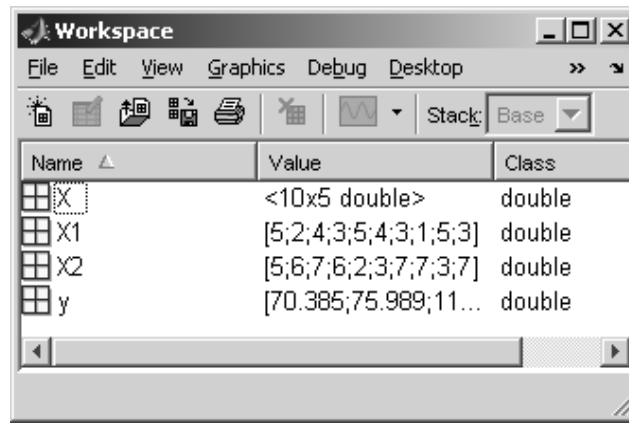


Рис. 12.69. Поточне вікно Workspace після обчислень й експорту

Змінні можна переглянути також командами, наприклад:

```
>> residuals
residuals =
```

-0.3016	1.4212
-0.0315	1.3829
-1.9945	1.0719
2.2855	-0.6079

2. Вид моделі можна змінити прямо в інтерактивному вікні. Наприклад, перейдемо в інтерактивному режимі до неповної квадратичесей моделі. У меню, що розкривається під кнопкою **Export**, вибираємо команду *Pure Quadratic*. Одержуємо новий результат (рис. 12.70):

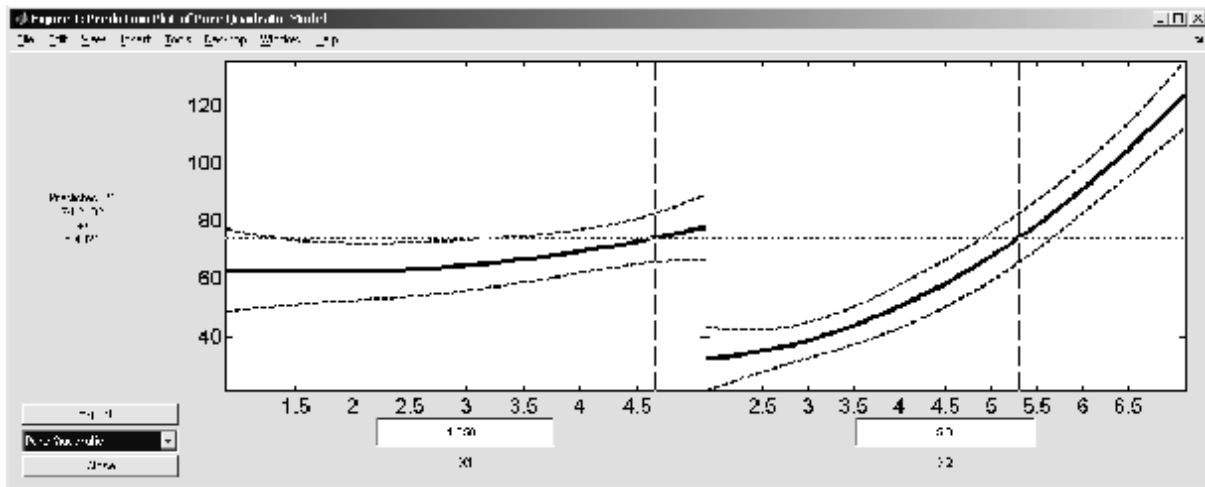


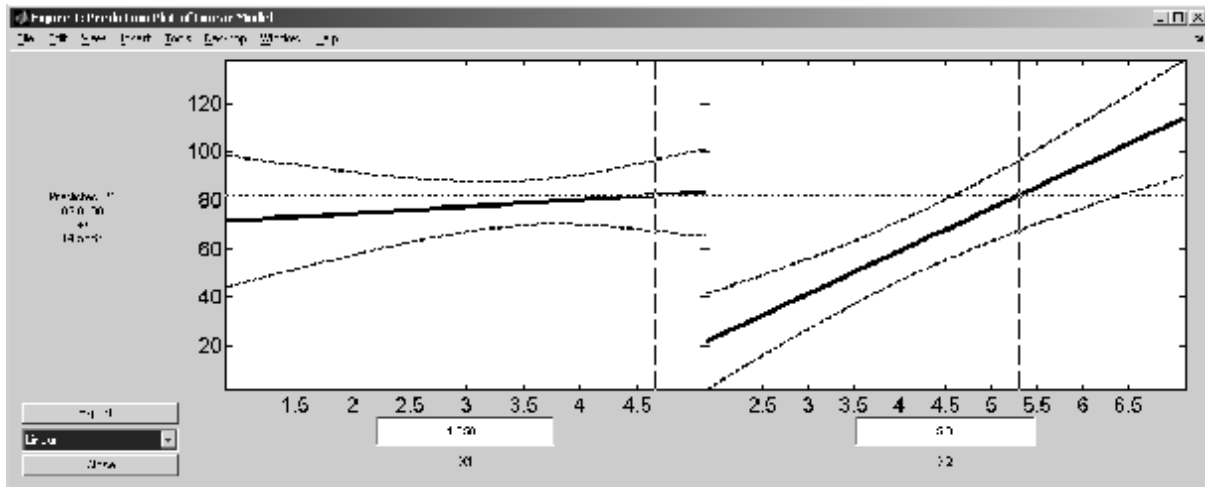
Рис. 12.70. Графічне вікно інтерактивного розрахунку й результату розрахунків для неповної квадратичної моделі

В порівнянні з повною квадратичною моделлю в неповній моделі вилучено член взаємодії:  $y = A(X_1)^2 + B(X_2)^2 + CX_1 + DX_2 + F + \varepsilon$ , де

$A = 1.2$ ,  $B = 2.5$ ,  $C = 1.5$ ,  $D = -1.3$ ,  $E = -0.89$ ,  $F = 0$  – параметри неповної моделі;  $X_1$ ,  $X_2$  – незалежні змінні;  $\varepsilon \approx N(0; 2)$ .

3. Перейдемо в інтерактивному режимі до лінійної регресійної моделі виду  $y = CX_1 + DX_2 + F + \varepsilon$ , де  $\varepsilon \approx N(0; 2)$ ;  $C = 1.5$ ,  $D = -1.3$ ,  $F = 0$  – параметри лінійної регресійної моделі;  $X_1$ ,  $X_2$  – незалежні змінні.

У меню, що розкривається під кнопкою **Export**, вибираємо команду *Linear*. Одержуємо новий результат (рис. 12.71):



**Рис. 12.71. Графічне вікно інтерактивного розрахунку й результату розрахунку для лінійної регресійної моделі**

У всіх прикладах границі довірчих інтервалів регресійних моделей розраховувалися для довірчої ймовірності 95% (за замовчуванням).

## 12.4. Діагностика регресійних моделей

<b>RCOPLLOT</b>	Графік залишкових похибок .....	181
<b>REGSTATS</b>	Розрахунок параметрів і діагностика множинної регресійної моделі .....	182
<b>LEVERAGE</b>	Розрахунок коефіцієнтів впливу окремих спостережень на параметри поліноміальної регресійної моделі .....	190
<b>ADDEDVARPLOT</b>	Графік регресії з доданою змінною .....	193

## **RCOPLOT** Графік залишкових похибок

Ця функція є доповненням до функції **regress**. Вона будує графік залишкових помилок для всіх точок і їх довірчі інтервали.

*Синтаксис:*

**rcoplot(r,rint)**

*Опис:*

Функція **rcoplot(r,rint)** будує графік залишкових помилок і довірчі інтервали для них, використовуючи вихідні параметри *r* і *rint* функції **regress**. Якщо довірчий інтервал не охоплює 0, він виділяється іншим кольором.

```
>> x=[0:10]'; % точки
>> n=length(x); % кількість точок
>> X=[ones(n,1),x,x.^2]; % стовпці 1, x, x2
>> y=X*[5;-3;2]+normrnd(0,1,n,1); % ординати
>> [b,bint,r,rint]=regress(y,X,0.1); % регресія
>> rcoplot(r,rint);
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfГрафік залишкових похибок') % заголовок
>> xlabel('\bfНомер точки')
>> ylabel('\bfЗалишкові похибки')
>> grid on, box on
```



Рис. 12.72. **Графік залишкових похибок**

На рис. 12.72 наведено графік залишків, на якому довірчі інтервали відзначені відрізками. Тут наведено чорно-білий варіант, а довірчий інтервал, що не охоплює 0, виділено напівжирною лінією.

## **REGSTATS** Розрахунок параметрів і діагностика множинної регресійної моделі

Проводиться дослідження побудованої регресійної моделі за багатьма критеріями.

*Синтаксис:*

```
regstats(responses,DATA,'model')
stats = regstats(responses,DATA,'model','whichstats')
stats = regstats(...)
```

*Опис:*

Функція **regstats(responses,DATA,'model')** призначена для розрахунку параметрів множинної регресійної моделі для вектора значень залежної змінної *responses*, матриці незалежних змінних *DATA*, регресійної моделі виду *'model'*. Функція відображає графічне вікно з набором статистик, що призначені для оцінки якості множинної регресійної моделі (рис.12.73, див. приклад нижче). Для вибору статистик їх необхідно відзначити відповідними прапорцями. Після натискання кнопки **OK** обрані статистики із заданими ідентифікаторами змінних будуть розраховані й експортовані в середовище **MatLab**.

Передбачені наступні види регресійних моделей (табл. 12.11):

Таблиця 12.11

**Можливі види регресійних моделей**

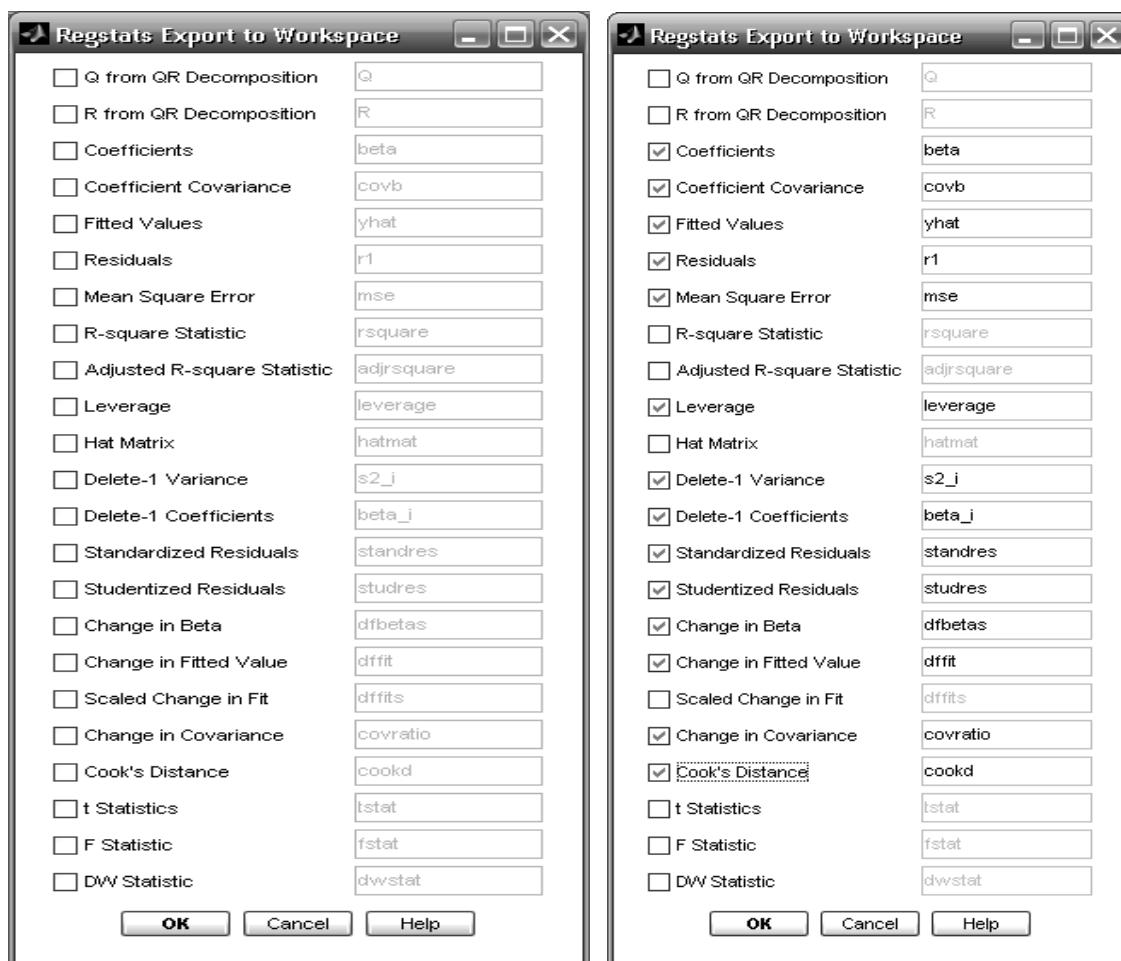
Значення <i>'model'</i>	Склад ефектів множинної регресійної моделі
<i>'linear'</i>	Лінійна модель, що включає лінійні ефекти факторів і константу. Приймається за замовчуванням
<i>'interaction'</i>	Лінійна модель, що включає лінійні ефекти, ефекти взаємодії факторів і константу
<i>'quadratic'</i>	Квадратична модель, що включає лінійні і квадратичні ефекти, ефекти взаємодії факторів, константу
<i>'purequadratic'</i>	Неповна квадратична модель, що включає квадратичні й лінійні ефекти факторів, константу

Послідовність коефіцієнтів множинної регресійної моделі відповідає їх порядку, використовуваному у функції **x2fx** (див. вище).

Функція **stats = regstats(responses,DATA,'model','whichstats')** повертає структуру *stats*, що містить статистики множинної регресійної

моделі. Склад полів структури *stats* визначає вхідний аргумент *'whichstats'*, який може бути заданий як символічна константа, наприклад *'leverage'*, або масив рядків, утримуючий рядкові константи, наприклад *{'leverage' 'standres' 'studres'}*. Припустимі рядкові константи (статистики регресійної моделі) за умови формування *'whichstats'* наведено в табл. 12.12.

Потрібні статистики можна вибрати для розрахунку в графічному вікні (рис. 12.73):



**Рис. 12.73. Графічне вікно вибору статистик множинної регресійної моделі. Праворуч – вікно із обраними опціями**

Припустимі рядкові константи '*whichstats*'

Значення константи	Статистика, що повертається
' <i>Q</i> '	Матриця $Q$ за QR-розкладу матриці незалежних змінних <i>DATA</i>
' <i>R</i> '	Матриця $R$ за QR-розкладу матриці незалежних змінних <i>DATA</i>
' <i>beta</i> '	Вектор точкових оцінок коефіцієнтів регресійної моделі
' <i>covb</i> '	Коваріаційна матриця коефіцієнтів регресійної моделі
' <i>yhat</i> '	Вектор залежної змінної, розрахований за регресійною моделлю для значень незалежних змінних <i>DATA</i>
' <i>r</i> '	Вектор залишків
' <i>mse</i> '	Середня квадратична похибка
' <i>leverage</i> '	Вектор ступеня впливу окремих спостережень на коефіцієнти регресійної моделі
' <i>hatmat</i> '	Матриця, що проектує вектор спостережень залежної змінної на вектор розрахункових значень залежної змінної за регресійною моделлю
' <i>s2_i</i> '	Вектор середніх квадратичних похибок залежної змінної, отриманих без обліку поточного спостереження
' <i>beta_i</i> '	Матриця коефіцієнтів регресійної моделі. Стовпці матриці – вектори коефіцієнтів регресійної моделі, що отримані послідовно без обліку одного зі спостережень
' <i>standres</i> '	Вектор стандартизованих залишків (залишків, поділених на величину їх середнього квадратичного відхилення)
' <i>studres</i> '	Вектор стандартизованих залишків без обліку поточного спостереження (залишків, поділених на величину відповідного елемента вектора ' <i>s2_i</i> ')
' <i>dfbetas</i> '	Матриця впливу коефіцієнтів регресійної моделі. Стовпці матриці – вектори коефіцієнтів регресійної моделі без відповідного коефіцієнта, що отримані послідовно без обліку одного зі спостережень
' <i>dffit</i> '	Вектор виправлень щодо значень залежної змінної, розрахованих без обліку поточного спостереження
' <i>dffits</i> '	Вектор виправлень щодо значень залежної змінної, розрахованих без обліку поточного спостереження, і нормований на величину стандартної похибки
' <i>covratio</i> '	Вектор відношень узагальненої дисперсії за умови визначення значень коефіцієнтів регресійної моделі без поточного спостереження до узагальненої дисперсії коефіцієнтів регресії з урахуванням усіх спостережень
' <i>cookd</i> '	Вектор відстаней Кука. Елементи вектора відстаней Кука є нормалізованими виправленнями до вектора коефіцієнтів без обліку поточного спостереження
' <i>all</i> '	Створення структури <i>stats</i> , що включає всі перераховані вище статистики

Послідовність статистик на рис 12.73 і в табл. 12.12 збігається. Для одержання більш детальної інформації для кожної статистики використовується кнопка **Help** у вікні вибору (рис. 12.73).

Алгоритм розрахунку коефіцієнтів рівняння регресії.

Розрахунок точкових оцінок коефіцієнтів виконується методом найменших квадратів з наступного рівняння лінійної моделі:  $Y = XB + \varepsilon$ , де  $Y$  – вектор значень залежної змінної;  $B$  – вектор коефіцієнтів лінійної моделі;  $X$  – матриця значень незалежних змінних;  $\varepsilon$  – вектор випадкових похибок, розподілених за нормальним законом з нульовим математичним очікуванням і однаковою дисперсією  $\sigma^2$ :  $\varepsilon \approx N(0; \sigma)$ .

Розмірності векторів значень залежної змінної  $Y$  й випадкових похибок  $\varepsilon$  дорівнюють  $n \times 1$ , де  $n$  – кількість спостережень. Розмірність матриці  $X$  рівна  $n \times p$ , де  $p$  – кількість незалежних змінних. Стовпці матриці  $X$  відповідають незалежним змінним, рядки – спостереженням. Розмірність вектора коефіцієнтів лінійної регресійної моделі  $B$  складає  $p \times 1$ .

Для розрахунку коефіцієнтів використовується QR-розкладання матриці спостережень незалежних змінних  $X=Q \cdot R$ , де  $Q$  – прямокутна ортогональна матриця,  $R$  – трикутна матриця. Матриці  $Q$  і  $R$  використовуються для розрахунку інших статистик, наведених у табл. 12.12.

Основним алгоритмом МНК, що приводиться в літературі, є визначення коефіцієнтів регресії у вигляді добутку матриць  $B = (X'X)^{-1}X'Y$ . Проте такий спосіб рішення має деякі істотні недоліки. Зокрема, за мультиколінеарністю не завжди можна знайти обернену матрицю  $(X'X)^{-1}$  з причини кінцевої точності обчислень. Більш стабільним чисельним алгоритмом вважається  $B = R^{-1}Q'Y$ . Цей алгоритм розрахунку точкових оцінок множинної регресійної моделі використовується функцією **regstats**.

*Приклади:*

1. Розглядається двофакторна лінійна регресійна модель виду  $y = AX_1 + BX_2 + C + \varepsilon$ , де  $A = 1.2$ ,  $B = 2.5$ ,  $C = 0$  – параметри лінійної регресійної моделі;  $X_1, X_2$  – незалежні змінні;  $\varepsilon \approx N(0; 0.1)$ .

Моделювання матриці незалежних змінних:

```
>> X1=unidrnd(10,10,1);
>> X2=unidrnd(20,10,1);
>> X=[X1 X2]
X =
     1     1
     4    15
     9     9
```

1	19
2	10
3	9
2	17
7	11
3	5
2	14

Моделювання вектора залежної змінної

```
>> y = X * [2.5; 1.2] + normrnd(0,0.1,10,1)
y =
  3.5664
 28.0714
 33.4624
 25.2308
```

17.0858
18.4254
25.2406
30.5559
13.5571
21.7600

Розрахунок параметрів і діагностика лінійної регресійної моделі

```
>> regstats(y,X, 'linear')
```

З'являється вікно вибору статистик (рис. 12.73).

У цьому вікні прапорцями  можна виділити наступні статистики, що будуть експортовані в робочу область **Matlab**:

*Q* – ортогональна матриця *Q* після QR-розкладу матриці даних (*DATA*):  $X = QR$ ;

*R* – трикутня матриця *R* після QR-розкладу матриці даних (*DATA*):  $X = QR$ ;

*Coefficients* – коефіцієнти регресії;

*Coefficient Covariance* – коваріаційна матриця коефіцієнтів регресії;

*Fitted Values* – розрахункові значення;

*Residuals* – залишки;

*Mean Square Error* – середньоквадратична похибка;

*R-square Statistic* – коефіцієнт детермінації;

*Adjusted R-square Statistic* – коефіцієнт детермінації, скоректований на ЧСС;

*Leverage* – ступінь впливу окремих спостережень;

*Hat Matrix* – матриця, що проектує;  
*Delete-1 Variance* – вилучити сталий доданок (константу) в ході обчислення дисперсії;  
*Delete-1 Coefficients* – вилучити сталий доданок (константу) в ході обчислення коефіцієнтів;  
*Standardized Residuals* – стандартизовані залишки;  
*Studentized Residuals* – стьюдентизовані залишки;  
*Change in Beta* – змінити коефіцієнти;  
*Change in Fitted Value* – змінити розрахункові значення;  
*Scaled Change in Fit* – змінити масштаб у розрахунках;  
*Change in Covariance* – змінити ковариаційну матрицю;  
*Cook's Distance* – відстані Кука;  
*t-Statistic* – статистика Стьюдента;  
*F-Statistic* – статистика Фішера;  
*DVV-Statistic* – статистика Дарбіна-Уотсона;

У нашому прикладі вибираються статистики, відзначені прапорцями на рис. 12.73 праворуч. Висвітлюються назви обраних статистик, які за замовчуванням утримуються в текстових полях на рис. 12.73. Їх зміст:

*beta* – коефіцієнти регресії;  
*covb* – ковариаційна матриця коефіцієнтів регресії;  
*yhat* – розрахункові значення ординат;  
*r1* – різниці між експериментальними й теоретичними ординатами;  
*mse* – залишкова дисперсія;  
*leverage* – рівень впливу спостереження на регресію;  
*s2\_i* – дисперсія для моделі без сталого доданка (без константи);  
*beta\_i* – коефіцієнти для моделі без сталого доданка;  
*standres* – нормовані різниці між експериментальними й теоретичними ординатами;  
*studres* – зведені до розподілу Стьюдента різниці між експериментальними й теоретичними ординатами;  
*dfbetas* – масштабовані зміни коефіцієнтів регресії;  
*dffit* – зміни в значеннях ординат;  
*covratio* – зміни в дисперсіях;  
*cookd* – відстані Кука (Cook).

Після натискання кнопки  одержуємо результати розрахунків:

>> y	33.4624	25.2406
y =	25.2308	30.5559
3.5664	17.0858	13.5571
28.0714	18.4254	21.7600

Значення статистик можна вивести на екран через командний рядок. Нижче виведено дві статистики.

```
>> beta
beta =
-0.0236
 2.5155
 1.1969

>> covb
covb =
 0.0124 -0.0010 -0.0007
-0.0010  0.0003  0.0000
-0.0007  0.0000  0.0001
```

Значення змінних за відзначеними статистиками можна також одержати у вікні Workspace (рис. 12.74), як і значення всіх уведених і обчислених інших змінних. Зміст будь-якої змінної може бути переглянуто у вікні Array Editor шляхом подвійного клацання по її назві у вікні Workspace.

2. Розглянемо повну квадратическую регресійну модель виду  $y = A(X_1)^2 + B(X_2)^2 + CX_1 + DX_2 + EX_1X_2 + \varepsilon$ , де  $\varepsilon \approx N(0; 0.1)$ ;  $A = 1.2$ ,  $B = 2.5$ ,  $C = 1.5$ ,  $D = -1.3$ ,  $E = -0.89$ ,  $F = 0$  – параметри лінійної регресійної моделі;  $X_1$ ,  $X_2$  – незалежні змінні.

```
% Моделювання матриці значень незалежних змінних
>> X1 = unidrnd(10,10,1);
>> X2 = unidrnd(20,10,1);
% Моделювання матриці значень незалежних змінних.
>> y=1.2.*X1.^2+2.5.*X2.^2+1.5.*X1-1.3.*X2-0.89.*X1.*X2
```

y =	449.2400
113.4600	181.4400
462.0400	744.0000
139.0900	747.7200
274.2400	330.8400
113.4600	

```
% Розрахунок параметрів і діагностика квадратичної регресійної моделі
>> regstats(y,[X1 X2],'quadratic')
```

З'являється графічне вікно рис. 12.75 для вибору статистик.

Результатирозрахунку й експорту:

```
>> y
```

139.0900	181.4400
274.2400	744.0000
113.4600	747.7200
449.2400	330.8400

```
y =
113.4600
462.0400
```

Фукція **stats = regstats(...)** – у цьому синтаксисі виклику інтерфейс користувача не створюється, а повертається структура *stats*, поля якої містять усі характеристики регресійної моделі, що включені в графічний інтерфейс користувача. Назви полів повторюють імена змінних, які за замовчуванням утримуються в текстових полях на рис. 12.75.

Наприклад:

```
>> stats = regstats(y, [X1 X2], 'quadratic')
```

```
stats =
    source: 'regstats'
      Q: [10x6 double]
      R: [6x6 double]
    beta: [6x1 double]
    covb: [6x6 double]
    yhat: [10x1 double]
      r: [10x1 double]
    mse: 2.9484e-026

    rsquare: 1
    adjrsquare: 1
    leverage: [10x1 double]
    hatmat: [10x10 double]
    s2_i: [10x1 double]
    beta_i: [6x10 double]
    standres: [10x1 double]
    studres: [10x1 double]
    dfbetas: [6x10 double]

    dffit: [10x1 double]
    dffits: [10x1 double]
    covratio: [10x1 double]
    cookd: [10x1 double]
    tstat: [1x1 struct]
    fstat: [1x1 struct]
    dwstat: [1x1 struct]
```

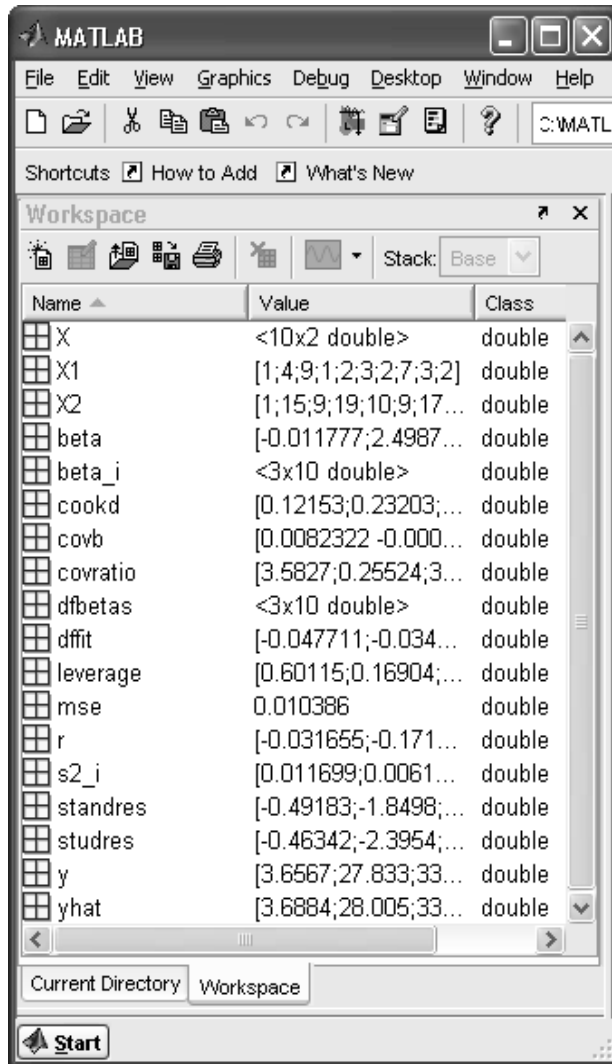


Рис. 12.74. Вікно Workspace з розрахунками

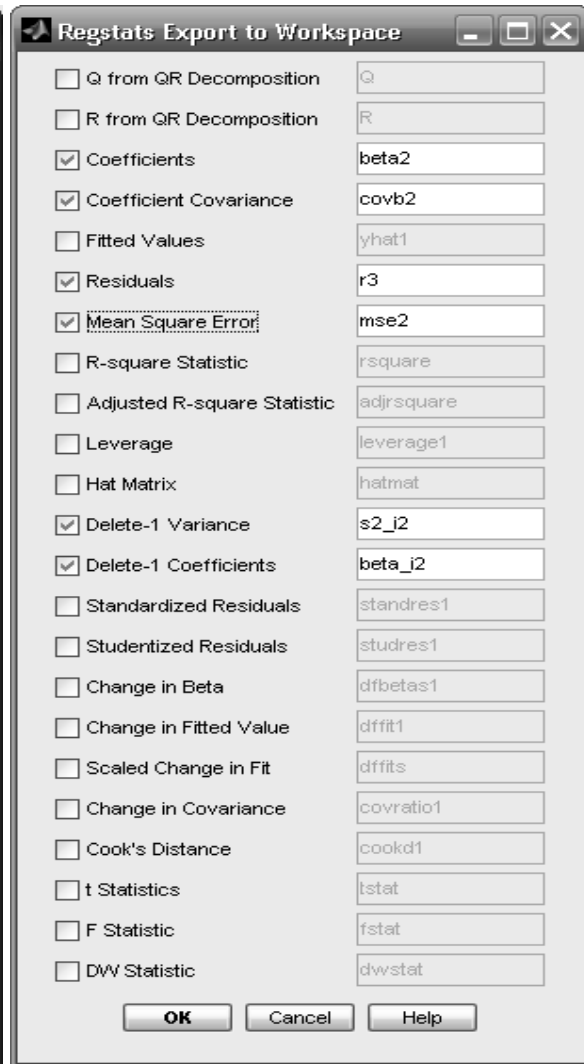


Рис. 12.75. Вікно вибору статистик для повної квадратичної моделі

## **LEVERAGE** Розрахунок коефіцієнтів впливу окремих спостережень на параметри поліноміальної регресійної моделі

Оцінюються ступені впливу окремих спостережень у множині даних на параметри регресійної моделі (**leverage** – важіль).

*Синтаксис:*

```
h = leverage(data)  
h = leverage(data,'model')
```

*Опис:*

Функція **h = leverage(data)** дозволяє оцінити коефіцієнти впливу  $h$  кожного зі спостережень (рядків у матриці *data*) на коефіцієнти лінійного рівняння регресії. Результативний параметр  $h$  визначається як вектор  $length(h)=size(data,1)$ .

Функція **h = leverage(data,'model')** призначена для оцінки коефіцієнтів впливу окремих спостережень на коефіцієнти поліноміальної регресійної моделі, обумовленої вхідним параметром *'model'*. Передбачені наступні види регресійних моделей:

*'linear'* – лінійна модель, що включає лінійні ефекти незалежних змінних і константу;

*'interaction'* – модель взаємодій, що включає лінійні ефекти, ефекти взаємодій незалежних змінних і константу.

*'quadratic'* – квадратична модель, включає ефекти взаємодій і квадратичні ефекти незалежних змінних;

*'purequadratic'* – квадратична модель без обліку ефектів взаємодій незалежних змінних; включає лінійні, квадратичні ефекти й константу.

Коефіцієнти впливу  $h$  на параметри поліноміальної регресійної моделі визначаються з урахуванням положення кожного зі спостережень у просторі незалежних змінних.

Алгоритм розрахунку коефіцієнтів впливу.

```
>> [Q,R] = qr(x2fx(data,'model'));  
>> leverage = (sum(Q'.*Q))'
```

Ця функція дозволяє перевіряти наступне правило: максимальний рівень впливу не повинен перевищувати величини  $2p/n$ , де  $p$  – кількість параметрів моделі, а  $n$  – кількість спостережень.

*Приклади:*

1. Перевіримо це правило на реальній базі даних, що перебувають у файлі `hald.mat`. Фрагмент даних наведено в табл. 12.13.

Таблиця 12.13

**Фрагмент даних файлу `hald.mat`**

Інгредієнти				Температура
7	26	6	60	78,5
1	29	15	52	74,3
11	56	8	20	104,3
11	31	8	47	87,6
7	52	6	33	95,9
11	55	9	22	109,2
3	71	17	6	102,7
1	31	22	44	72,5
2	54	18	22	93,1
21	47	4	26	115,9
1	40	23	34	83,8
11	66	9	12	113,3
10	68	8	12	109,4

```
>> load hald.mat % Завантажуємо дані
>> [n,rows]=size(ingredients)
n =
    13
rows =
     4
% Кількість параметрів моделі:
>> p=rows+1
p =
     5
```

Обчислюємо величину  $2p/n$ :

```
>> hmax=2*p/n
hmax =
    0.7692
```

Оцінюємо коефіцієнти впливу  $h$ .

```
>> h=leverage(ingredients)
h =
    0.2952    0.3576    0.1242    0.3671    0.4085
    0.2943    0.7004    0.4255    0.2630    0.3037
    0.5503
    0.3332
    0.5769
```

Знаходимо максимальне значення.

```
>> h1=max(leverage(ingredients))
h1 =
    0.7004
```

Тепер можна порівняти  $h1$  і  $hmax$ . Друге число більше першого. Виходить, правило виконується.

Усе це можна виконати з виведенням на екран лише останнього висновку:

```
>> load hald.mat % завантажуюмо базу даних
>> [n,rows]=size(ingredients); % розміри потрібного масиву
>> p=rows+1; % число параметрів лінійної моделі
>> hmax=2*p/n;
>> fprintf('Кількість вимірів n=%d\n',n);
>> fprintf('Число параметрів лінійної моделі p=%d\n',p);
>> fprintf('Параметр 2*p/n=%12.8f\n',hmax);
>> h=max(leverage(ingredients));
>> fprintf('Максимальний вплив на параметри моделі h=%12.8f\n',h);
>> if h<hmax,
    disp('Правило виконується');
else
    disp('Правило порушується');
end
```

Кількість вимірів n=13

Число параметрів лінійної моделі p=5

Параметр  $2*p/n= 0.76923077$

Максимальний вплив на параметри моделі h= 0.70040277

Правило виконується

2. Порівняння коефіцієнтів впливу спостережень нормально розподіленої двовимірної випадкової величини без викидів і з викидами в вибірці для лінійної моделі.

2.1. Моделювання матриці незалежних випадкових величин, розподілених за нормальним законом з нульовим математичним очікуванням і одиничною дисперсією параметрів лінії регресії.

```
>> r=normrnd(0,1,10,2)
r =
    -0.9530    2.0941
     0.7782    0.0802
    -0.0063   -0.9373
     0.5245    0.6357
     1.3643    1.6820
     0.4820    0.5936
    -0.7871    0.7902
     0.7520    0.1053
    -0.1669   -0.1586
    -0.8162    0.8709
```

2.2. Розрахунок коефіцієнтів впливу спостережень

```
>> h = leverage(r)
h =
    0.5689
    0.1998
     0.4476
     0.1318
     0.6281
     0.1246
     0.2460
     0.1914
     0.2037
     0.2579
```

3. Порівняння коефіцієнтів впливу спостережень нормально розподіленої двовимірної випадкової величини без викидів і з викидами у вибірці для квадратичної моделі.

3.1. Моделювання матриці 3-х незалежних випадкових величин, розподілених за нормальним законом з нульовим математичним очікуванням і одиничною дисперсією.

```
>> r=normrnd(0,1,10,3);
>> r=r.^2
r =
    0.0379    0.2609    0.0074    1.4122    2.5706    0.9089
    0.0057    0.0621    0.1058    0.0618    0.1152    0.0546
```

3.2. Розрахунок коефіцієнтів впливу спостережень

```
>> h = leverage(r,'purequadratic')
h =
    0.7693
    0.3874
```

0.3268	0.2663
0.7456	1.0000
0.5418	0.9628
1.0000	1.0000

3.3. Додавання викида в останнє спостереження

```
>> r(10,1)=10;r(10,2)=-10; r(10,3)=-5;
```

3.4. Розрахунок коефіцієнтів впливу спостережень

```
>> h1=leverage(r,'purequadratic')
h1 =
    0.7762
    0.3347
```

0.2641	0.2694
0.7748	1.0000
0.5961	0.9848
0.9999	1.0000

### **ADDEDVARPLOT** Графік регресії з доданою змінною

Багатофакторні регресійні моделі традиційно складаються за стандартною методикою підключення – вилучення незначимих членів. Згідно з принципом економії Л. Клейна, незначимі члени мають бути вилучені з регресійної моделі. Але, з теоретичних або будь-яких інших змістовних міркувань, фахівець може вважати обов'язковою присутність в моделі деяких, нехай навіть незначущих членів. Функція **addedvarplot** дозволяє включити в модель такі обов'язкові члени і дослідити статистичні наслідки цього директивного втручання в загальноприйнятний алгоритм складання багатофакторної моделі.

*Синтаксис:*

**addedvarplot(X,y,vnum,inmodel)**  
**addedvarplot(X,y,vnum,inmodel,stats)**

*Опис:*

Функція **addedvarplot(X,y,vnum,inmodel)** створює діаграму з експериментальними точками й лінією регресії за однієї зі змінних. Матриця  $X$  розміром  $n \times p$  містить стовпці значень базисних функцій (регресорів – членів регресійної кривої). Експериментальні значення функції записані у векторі-стовпці  $y$  довжиною  $n$ . Скаляр  $vnum$  показує, за якою змінною потрібно будувати регресію. Аргумент *inmodel* – логічний масив довжини  $p$  – показує, які змінні (стовпці  $X$ ) вже включені в модель. Спочатку за замовчуванням усі елементи *inmodel* дорівнюють 0 (в модель нічого не включене, тобто зараз в модель додається лише одна змінна за номером  $vnum$ ). Для створення вектора *inmodel* для найкращої моделі можна використовувати функцію **stepwisefit**, що описана вище.

Функція **addedvarplot(X,y,vnum,inmodel,stats)** використовує структуру *stats*, яка містить характеристики найкращої регресійної моделі, створеною функцією **stepwisefit**. Бажано створити цю структуру заздалегідь, до виклику функції **addedvarplot**, щоб скоротити час її роботи.

На діаграмі будуть показані експериментальні точки й найкраща теоретична крива. Якщо позначити через  $X_1$  стовпець з номером  $vnum$  матриці  $X$ , то додається графік залежності  $y$  від  $X_1$  після видалення впливу всіх інших змінних, зазначених у векторі *inmodel*. Теоретична крива, що побудована за допомогою методу найменших квадратів, буде накреслена суцільною кривою. Її кутовий коефіцієнт – це значення, яке мав би коефіцієнт при  $X_1$  за включення цього стовпця в модель. Пунктирними лініями будуть показані 95%-і довірчі інтервали для директивно підключеного члена.

Приклад побудови лінійної моделі із завантаженням бази даних файлу `hald.mat`. Фрагмент даних цього файлу наведено в табл. 12.13.

Спочатку визначимо структуру `stats`:

```
>> load hald
>> [b,se,p,inmodel,stats]=stepwisefit(ingredients,heat)
Initial columns included: none
Step 1, added column 4, p=0.000576232
Step 2, added column 1, p=1.10528e-006
Final columns included: 1 4

'Coeff'   'Std.Err.' 'Status'   'P'           | b =           | se =
[ 1.4400] [0.1384]   'In'      [1.1053e-006] | 1.4400        | 0.1384
[ 0.4161] [0.1856]   'Out'     [0.0517]      | 0.4161        | 0.1856
[-0.4100] [0.1992]   'Out'     [0.0697]      | -0.4100       | 0.1992
[-0.6140] [0.0486]   'In'      [1.8149e-007] | -0.6140       | 0.0486

p =
0.0000
0.0517
0.0697
0.0000
inmodel =
1 0 0 1
stats =
source: 'stepwisefit'
dfe: 10
df0: 2
SStotal: 2.7158e+003
SSresid: 74.7621
fstat: 176.6270
pval: 1.5811e-008
rmse: 2.7343
xr: [13x2 double]
yr: [13x1 double]
B: [4x1 double]
SE: [4x1 double]
TSTAT: [4x1 double]
PVAL: [4x1 double]
intercept: 103.0974
wasnan: [13x1 logical]
```

Функцією `stepwisefit` виведена наступна інформація. Методом послідовного підключення складено найкращу модель, що містить лише дві змінні  $x_1$  і  $x_4$ . Її рівняння:  $Y_p = b_0 + b_1x_1 + b_4x_4$ . Далі зроблено пробні підключення до найкращої двофакторної моделі інших змінних  $x_2$  і  $x_3$ . У невеличкій таблиці зазначено статус `'In'` – для підключених і `'Out'` – для інших змінних. Рівень значущості  $p$  для підключених змінних менше 0.01, для непідключених – більше 0.05, тобто найкраща модель складається лише зі значущих членів. Далі наведено значення коефіцієнтів регресії  $b$ , їх середньоквадратичні похибки  $Sb$  (позначені  $se$ ) і рівні значущості  $p$  (ймовірності 0-гіпотези, яка стверджує, що генеральні значення параметрів дорівнюють нулю). Ще раз відзначається, що модель містить лише 1-й і 4-й аргументи: `inmodel = [1 0 0 1]`. Після цього виведено структуру `stats`, що описує розмірності всіх масивів і містить інформацію для дисперсійного аналізу: числа ступенів свободи  $dfe$ ,  $dfRegr$  (позначено  $df0$ ); суми квадратів  $SSY$ ,  $SSE$  (позначено  $SStotal$ ,  $SSresid$ ),

дисперсійне відношення Фішера  $F$  ( $fstat$ ), його рівень значимості  $pval$ . Модель значима в цілому, тому що  $pval < 0.01$ . Нарешті, наведено значення кореня квадратного із залишкової дисперсії  $mse$  ( $rmse$ ) і значення вільного члена  $b_0$  ( $Intercept$ ).

Тепер розглянемо результати роботи функції **addedvarplot**:  
**>> addedvarplot(ingredients, heat,2,inmodel,stats)**

На рис. 12.76 показаний результат директивного підключення в модель змінної  $x_2$  (третій аргумент функції **addedvarplot** – 2, тобто  $x_2$ ).

Обчислені залишки  $E_y = Y - Y_p$  (позначено  $Y$  residuals). Далі складено двофакторну модель для  $X_2(x_1, x_4)$  і обчислено залишки  $EX_2 = X_2 - X_{2p}$  (позначено  $X_2$  residuals). Між цими залишками знайдено залежність  $E_y = b_2 EX_2$ , де параметр  $b_2 = 0.4161$ , і розраховано 95%-і довірчі інтервали на теоретично очікувані значення цього параметра  $b_2 - t_{0.05} Sb_2 < \beta_2 < b_2 + t_{0.05} Sb_2$  ( $0.0076 < \beta_2 < 0.8246$ ).

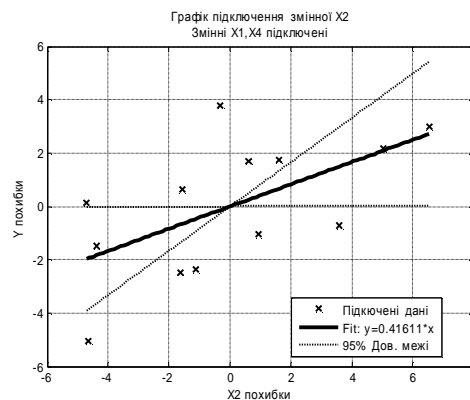


Рис. 12.76. Лінія регресії між залишками  $Y - X_2$  після обліку  $x_1$  і  $x_4$

На рис. 12.76 побудовано графіки залежності між залишками при  $b_2 = [0.0076 \ 0.4161 \ 0.8246]$ . Хрестиками на графіку відзначено дані, скоректовані (*adjust*) на середні значення вже врахованих змінних  $x_1$  і  $x_4$ .

Наведемо графік залишків за умови додавання третьої змінної (рис. 12.77).

**>> addedvarplot(ingredients,heat,3,inmodel,stats)**

Фахівцю, що наполягає на обов'язковій присутності в моделі цих статистично незначимих членів, надається інформація, на підставі якої він має зробити свої висновки.

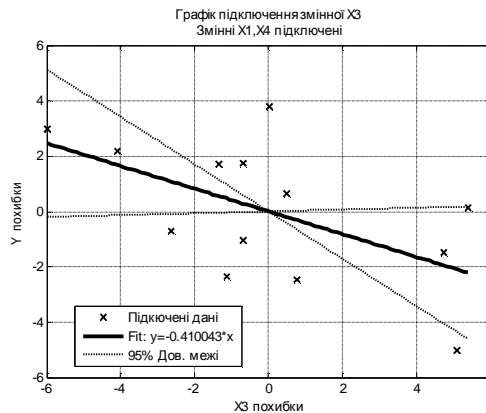


Рис. 12.77. Лінія регресії за умови включення третьої змінної

## 12.5. Спеціальні процедури регресійного аналізу

<b>RIDGE</b> Ридж-регресія .....	197
<b>ROBUSTFIT</b> Робастна лінійна регресія .....	201
<b>LSCOV</b> Метод найменших квадратів із заданою коваріаційною матрицею .....	206
<b>LSQNONNTG</b> Метод найменших квадратів з невід'ємними коефіцієнтами .....	209

### **RIDGE** Ридж-регресія

Будується лінійна регресійна модель за методом гребінних оцінок. Це один з найефективніших прийомів подолання наслідків сильної мультіколінеарності.

*Синтаксис:*

**b1 = ridge(y,X,k)**  
**b0 = ridge(y,X,k,0)**  
**b = ridge(y,X,k)**

Опис:

Функція **bk = ridge(y,X,k)** призначена для розрахунку гребінних оцінок параметрів лінійної регресійної моделі:  $Y = XB + E$ , де  $Y$  – вектор значень залежної змінної;  $B$  – вектор коефіцієнтів лінійної моделі;  $X$  – матриця значень незалежних змінних;  $E$  – вектор випадкових похибок, розподілених за нормальним законом з нульовим математичним очікуванням і однаковою дисперсією  $\sigma^2$ .

Розмірності векторів значень залежної змінної  $Y$  і випадкових похибок  $E$  –  $(n \times 1)$ , де  $n$  – кількість спостережень. Розмірність матриці  $X$  дорівнює  $n \times p$ , де  $p$  – кількість незалежних (пояснюючих) змінних. Стовпці матриці  $X$  відповідають незалежним змінним, рядки – спостереженням. Розмірність вектора коефіцієнтів лінійної регресійної моделі  $B$  складає  $p \times 1$ .

Гребінні оцінки параметрів лінійної регресійної моделі  $B$  розраховуються за наступним вираженням:  $b_k = B = (X'X + kI)^{-1}X'Y$ , де  $k$  – ридж-параметр, що задається як скалярна величина;  $I$  – одинична матриця, в якій відмінні від нуля лише одиничні елементи головної діагоналі.

Якщо  $k = 0$ , то гребінні оцінки параметрів лінійної регресійної моделі  $b_k$  збігаються із точковими оцінками, отриманими методом найменших квадратів. Якщо величина  $k$  збільшується, то збільшується зсув коефіцієнтів  $b_k$ , але зменшується дисперсія оцінки. Для матриці незалежних змінних  $X$  з низьким числом обумовленості зменшення величини дисперсії оцінок коефіцієнтів відбувається швидше, ніж зростає похибка зсуву, тобто оцінки будуть декілька зміщеними, але більш ефективними. Загальна похибка складається з випадкової і систематичної складових; за наявності мультиколінеарності маємо дуже велику випадкову похибку; в ридж-регресії допускають невелику систематичну похибку, але суттєво знижують випадкову складову.

Матриця  $X$  не повинна містити стовпець одиниць.

Функція **b0 = ridge(y,X,k,0)** будує ридж-регресію без центрування й масштабування. Результат – вектор  $b_0$  довжиною  $(p + 1)$ , причому перший елемент відповідає сталому доданку моделі (константі).

Співвідношення між  $b_k$  і  $b_0$  у функції **ridge** наступне:

```
>> m = mean(X);
>> s = std(X,0,1)';
>> temp = bk./s;
>> b0 = [mean(Y)-m*temp; temp]
```

Якщо  $k = 0$ , то результат  $b_0$  або  $b_k$  – це оцінки звичайного методу найменших квадратів. За збільшення  $k$  зростає зсув коефіцієнтів  $b$ , але зменшуються їх дисперсії. Для погано обумовлених  $X$  зниження дисперсії впливає більше, ніж збільшення зсуву.

*Приклади:*

1. У прикладі будуються дві лінійні моделі: за стандартним методом найменших квадратів і за методом гребінних оцінок.

```
>> x=[0:10]'; % точки
>> n=length(x); % кількість точок
>> X=[ones(n,1),x,x.^2]; % стовпці 1, x, x^2
>> y=X*[5;-1;0.5]+normrnd(0,2,n,1); % ординати
>> b=regress(y,X); % регресія за МНК
>> disp('Регресія за методом найменших квадратів:');
>> fprintf('y(x)=%8.5f*x^2%+8.5f*x%+8.5f\n',flipud(b));
>> br0=ridge(y,X(:,2:end),0.2,0); % Регресія за гребінними оцінками
>> disp('Регресія за методом гребінних оцінок:');
>> fprintf('y(x)=%8.5f*x^2%+8.5f*x%+8.5f\n',flipud(br0));
>> xplot=linspace(x(1),x(end)); % абсциси для графіка
>> yp=polyval(flipud(b),xplot); % ординати регресії за МНК
>> yr=polyval(flipud(br0),xplot); % ординати за гребінними оцінками
>> plot(x,y,'ks',xplot,yp,'k-',xplot,yr,'k-');
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfРегресія за гребінними оцінками') % заголовок
>> xlabel('\bfx')
>> ylabel('\bfy')
>> grid on
```

Регресія за методом найменших квадратів:

$y(x) = 0.42429 \cdot x^2 - 0.04503 \cdot x + 2.84348$

Регресія за методом гребінних оцінок:

$y(x) = 0.34678 \cdot x^2 + 0.71569 \cdot x + 1.75250$

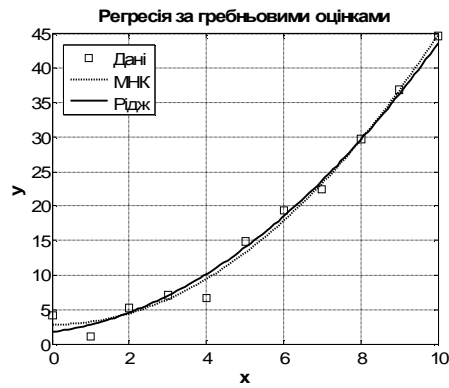


Рис. 12.78. Лінії регресії

Результати у вигляді двох рівнянь виведені на екран і представлені в графічному вікні на рис. 12.78. Лінії регресії, обчислені за методом найменших квадратів і методом гребінних оцінок мало відрізняються, хоча мають різні оцінки коефіцієнтів регресії.

2. Будується залежність зміни параметрів лінійної регресійної моделі даних файла `hald.mat` (див. вище зміст файла в описі функції **leverage**) від величини коефіцієнта  $k$  ридж-регресії. Графік будується для кожного з 4-х коефіцієнтів регресійної моделі за незалежних змінних – *ingredients* (стовпці матриці даних).

```
>> load hald;
>> b = zeros(4,100);
>> kvec = 0.01:0.01:1;
>> count = 0;
>> for k = 0.01:0.01:1; count=count + 1; b(:,count) = ridge(heat,ingredients,k); end
>> plot(kvec,'b'), grid on
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14) % шрифт
>> xlabel('\bfk'), ylabel('\bfbeta')
```

На рис. 12.79 у графічному вікні представлені результати розрахунку методом гребінних оцінок графіків гребінних слідів для кожного коефіцієнта регресії лінійної моделі. Видно, що дуже малі значення гребінного параметра  $k$  суттєво впливають на значення всіх коефіцієнтів регресії, які стабілізуються лише за  $k = 0.1$ . У файлі `hald.mat` пояснюючі змінні *ingredients* пов'язані мультиколінеарною залежністю, що призводить до зміщення МНК-оцінок параметрів моделі.

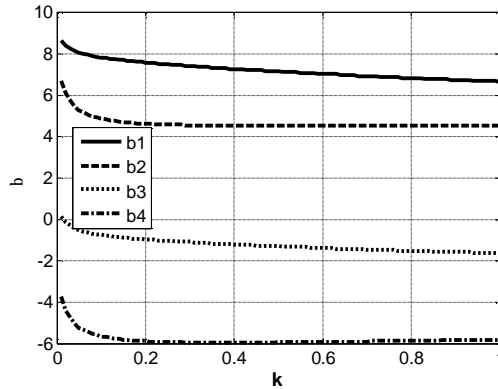


Рис. 12.79. Гребінні сліди для чотирьох параметрів лінійної моделі

### **ROBUSTFIT** Робастна лінійна регресія

Будується лінійна регресійна модель, що нечутлива до викидів – великих похибок вимірів (*robust* – стійкість).

*Синтаксис:*

**b = robustfit(X,y)**

**[b,stats] = robustfit(X,y)**

**[b,stats] = robustfit(X,y,wfun,tune,const)**

*Опис:*

Функція **b = robustfit(X,y)** будує робастну лінійну регресію  $y$  від  $X$  і повертає коефіцієнти регресії  $b$ . У матриці  $X$  не повинно бути одиничного стовпця; сталий доданок (константа) у лінійній моделі враховується автоматично. Якщо  $y$  – вектор розміром  $n \times 1$ ,  $X$  – матриця  $n \times p$ , то в  $b$  повертається вектор розмірності  $(p + 1) \times 1$ , причому перший елемент відповідає константі моделі. Функція **robustfit** використовує ітеративний зважений метод найменших квадратів, на кожній ітерації якого ваги обчислюються як біквадратичні функції від нев'язок попередньої ітерації. Такий алгоритм приводить до малих ваг для викидів (точок з великими відхиленнями). Результат малочутливий до похибок в даних в порівнянні зі звичайним методом найменших квадратів.

Функція **[b,stats] = robustfit(X,y)** у другому результативному параметрі *stats* повертає структуру з наступними полями:

*stats.ols\_s* – оцінка середньоквадратичного відхилення за методом найменших квадратів;

*stats.robust\_s* – робастна оцінка середньоквадратичного відхилення;

*stats.mad\_s* – оцінка середньоквадратичного відхилення, що обчислена як медіана абсолютних відхилень нев'язок від їх медіани з використанням масштабування нев'язок у ході ітераційного процесу;

*stats.s* – оцінка середньоквадратичного відхилення залишків, це більша із двох величин: зваженого середнього *ols\_s* і *robust\_s*;

*stats.se* – стандартні похибки оцінок коефіцієнтів;

*stats.t* – відношення *b* до *stats.se*;

*stats.p* – *p*-значення для *stats.t*;

*stats.coeffcorr* – оцінка кореляції оцінок коефіцієнтів;

*stats.w* – вектор ваг для робастної моделі;

*stats.h* – вектор значень рівнів впливу для моделі методу найменших квадратів;

*stats.dfe* – число ступенів свободи залишкової похибки;

*stats.R* – матриця *R* після QR-розкладу матриці *X*.

У процесі побудови робастної регресії розраховується коваріаційна матриця коефіцієнтів регресії *V*. Розрахунок *V* виконується як  $V = \text{inv}(X' * X) * \text{stats.s}^2$ . Стандартні похибки й коефіцієнти кореляції параметрів регресійної моделі є похідними від *V*.

Функція **[b,stats] = robustfit(X,y,wfun,tune,const)** в 3-му, 4-му і 5-му аргументах дозволяє задати відповідно вагову функцію, константу настроювання й наявність або відсутність константи в моделі. Аргумент *wfun* повинен бути текстовим рядком. Його можливі значення перелічено в табл. 12.14.

Можливі імена й значення аргументу *wfun* в функції *robustfit*

Ім'я	Значення	Константа <i>tune</i>
'andrews'	$w = (\text{abs}(r) < \pi) \cdot \sin(r) / r$	1,339
'bisquare'	$w = (\text{abs}(r) < 1) \cdot (1 - r.^2).^2$	4,685
'cauchy'	$w = 1 / (1 + r.^2)$	2,385
'fair'	$w = 1 / (1 + \text{abs}(r))$	1,400
'huber'	$w = 1 / \max(1, \text{abs}(r))$	1,345
'logistic'	$w = \tanh(r) / r$	1,205
'talwar'	$w = 1 * (\text{abs}(r) < 1)$	2,795
'welsch'	$w = \exp(-r.^2)$	2,985

Значення  $r$  у ваговій функції дорівнює  $\text{resid} / (\text{tune} * s * \sqrt{1 - h})$ , де  $\text{resid}$  – вектор нев'язок попередньої ітерації,  $\text{tune}$  – константа настроювання,  $h$  – вектор рівнів впливу в методі найменших квадратів, а  $s$  – оцінка стандартного відхилення похибок;  $s = \text{mad} / 0.6745$ . Тут  $\text{mad}$  – медіана абсолютного відхилення нев'язок від їх медіани. Константа 0,6745 робить цю оцінку незміщеною для нормального розподілу.

На додаток до імен, перелічених вище, у якості параметра *wfun* можна також задати рядок 'ols', щоб використовувати звичайний незважений метод найменших квадратів.

Аргумент *tune* дозволяє змінити константу настроювання, значення за замовчуванням для якої наведено в табл. 12.14. Зменшення значення *tune* приводить до зниження вагових коефіцієнтів вилучених точок більш строго, а збільшення *tune* – менш строго. Значення за замовчуванням, що наведені в табл. 12.14, приблизно в 95% випадків дають оцінки настільки ж ефективні, як і у звичайному методі найменших квадратів, якщо експериментальні значення  $y$  мають нормальний розподіл і не містять викидів.

Аргумент *const* може бути 'on' (за замовчуванням), якщо в моделі потрібно врахувати константу, або 'off', якщо ні. Якщо в моделі є сталий доданок (константа), потрібно задати *const='on'*, а не додавати в матрицю  $X$  одиничний стовпець.

Можна також задати власну вагову функцію. Вона повинна брати в якості аргументу  $r$  вектор нев'язок точок, і видавати в результаті вектор вагових коефіцієнтів  $w$  такої ж довжини. Цю функцію можна записати у файл (наприклад, `Mywfun.m`), і використовувати в якості аргументу  $wfun$  його ім'я '`mywfun`', дескриптор `@mywfun`, або використовувати механізм `inline`-функцій.

*Приклади:*

1. Порівняння лінійних регресійних моделей, отриманих методом найменших квадратів і методом робастної регресії. Розглядається лінійна регресійна модель виду  $y = Ax + B + \varepsilon$ , де  $\varepsilon \approx N(0, 1)$ ;  $A$ ,  $B$  – параметри лінійної регресійної моделі,  $A = -2$ ,  $B = 10$ ;  $x$  – незалежна змінна.

```
>> x = (1:10)'; % Моделювання вектора значень незалежної змінної
>> y = 10 - 2*x + normrnd(0,1,10,1); % Моделювання значень залежної змінної
>> y(10) = 0; % Додавання викиду у вектор значень залежної змінної
>> [bls bls_i] = regress(y,[ones(10,1) x]) % МНК-параметри лінійної моделі
bls =
    8.8528
   -1.5642
bls_i =
    3.9900    13.7155
   -2.3479   -0.7805
% Методом найменших квадратів одержали значення параметрів лінійної регресійної
% моделі і їх 95%-ві довірчі інтервали.
>> brob=robustfit(x,y) % Робастні значення параметрів
brob =
    10.8535
    -2.1232
```

Робастні значення параметрів близькі до заданих і суттєво відрізняються від відповідних МНК-оцінок

Представимо графічно отримані регресійні моделі й експериментальні точки.

```
>> scatter(x,y) % Експериментальні точки з викидом
>> hold on
>> plot(x,bls(1)+bls(2)*x,'g:') % Лінія регресії МНК
>> plot(x,bls_i(1,1)+bls_i(2,1)*x,'b:') % Нижня довірча границя
>> plot(x,bls_i(1,2)+bls_i(2,2)*x,'b:') % Верхня довірча границя
>> plot(x,brob(1)+brob(2)*x,'r-') % Робастна лінія регресії
>> grid on, box on, hold off
>> set(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
```

```
>> xlabel('\bfx'), ylabel('\bfy')
```

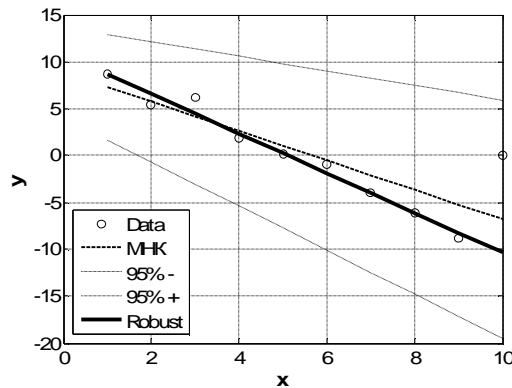


Рис. 12.80. Лінії регресії лінійної моделі

На рис. 12.80 представлена лінія регресії, що обчислена методом найменших квадратів з 95%-ми довірчими границями. Там же суцільною лінією показана робастна лінія регресії. Видно, що вона менш чутлива до викиду в точці (10; 0).

2. Будуються дві квадратичні моделі: за стандартним методом найменших квадратів і за допомогою робастної лінійної регресії. Розглянемо наслідки наявності в даних одного, але дуже великого викида.

```
>> x=[0:10]'; % точки
>> n=length(x); % кількість точок
>> X=[ones(n,1),x,x.^2]; % стовпці 1, x, x2
>> y=X*[5;-3;2]+normrnd(0,1,n,1); % ординати
>> y(n)=0; % одне помилкове значення
>> b=regress(y,X); % регресія pf МНК
>> disp('Регресія за методом найменших квадратів:');
>> fprintf('y(x)=%8.5f*x^2%+8.5f*x%+8.5f\n',flipud(b));
>> br=robustfit(X(:,2:end),y); % Робастна регресія
>> disp('Робастна регресія:');
>> fprintf('y(x)=%8.5f*x^2%+8.5f*x%+8.5f\n',flipud(br));
>> plot(x,y,'ks',x,X*b,'k:',x,X*br,'k-');
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfРобастная регресія') % заголовок
>> xlabel('\bfx'), ylabel('\bfy')
>> grid on, box on
Регресія за методом найменших квадратів:
y(x)=-1.07514*x^2+19.75249*x-17.08168
Робастна регресія:
y(x)= 1.96386*x^2-2.73565*x+4.80926
```

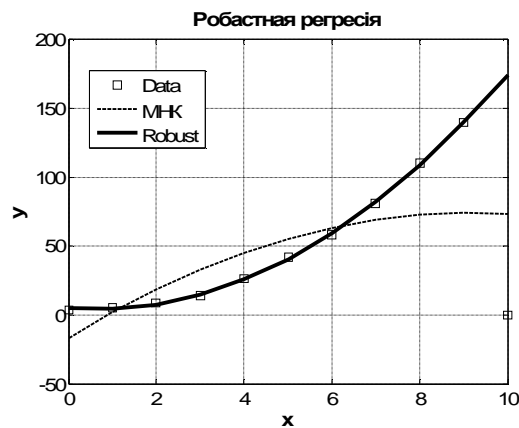


Рис. 12.81. Порівняльні лінії регресії для квадратичної моделі

На рис. 12.81 пунктирна лінія відповідає методу найменших квадратів, суцільна – робастної регресії. Видно, що за наявності викиду (10; 0) робастна регресія дає значно кращі результати.

### **LSCOV** Метод найменших квадратів із заданою коваріаційною матрицею

Вирішується перевизначена система лінійних рівнянь із заданою коваріаційною матрицею. Знаходяться параметри моделі за допомогою методу найменших квадратів, у тому числі зваженого МНК й з відомою коваріаційною матрицею.

*Синтаксис:*

**$x = \text{lscov}(A,b,V)$**   
 **$[x,dx] = \text{lscov}(A,b,V)$**

*Опис:*

Функція  **$x = \text{lscov}(A,b,V)$**  повертає вектор  $x$ , що є рішенням системи рівнянь  $AX = B + E$ , де  $E$  – вектор нормально розподілених значень похибок з нульовим математичним очікуванням і коваріацією  $V$ . У розглянутій задачі метод найменших квадратів застосовується для рішення перевизначеної системи рівнянь. Розмірність матриці  $A$  –  $n \times m$ , де  $n$  – кількість спостережень,  $m$  – число незалежних змінних,  $n > m$ . Розмірність матриці  $V$  дорівнює  $n \times n$ . Матриця  $V$  повинна бути

позитивно визначеною. За замовчуванням функція **lscov** використовує розклад Холецкого (Cholesky) для  $V$  і використовує отриманий розклад для обчислення оберненої матриці  $V^{-1}$  і до приведення задачі до звичайного методу найменших квадратів. Проте, якщо  $V$  виявиться напіввизначеною, використовується алгоритм ортогонального розкладання, який дозволяє обійтися без обернення матриці  $V$ .

Функція **[x,dx] = lscov(A,b,V)** призначена для розрахунку вектора рішень  $X$  системи рівнянь  $AX = B + E$  і їх стандартних похибок  $dx$ . Розмірність векторів  $X$  і  $dx$  буде збігатися.

Алгоритм методу.

Вектор  $X$  знаходять з умови мінімізації вираження

$$(AX - B)'V^{-1}(AX - B).$$

Класичне рішення цієї задачі, відоме з лінійної алгебри, має такий вигляд:  $X = (A'V^{-1}A)^{-1}A'V^{-1}B$ .

У функції **lscov** використана QR декомпозиція матриці  $A$  з наступною модифікацією матриці  $Q$  за заданою коваріацією  $V$ . Величини стандартних похибок розраховуються за формулами:

$$mse = B'(V^{-1} - V^{-1}A(A'V^{-1}A)^{-1}A'V^{-1})B / (m - n);$$

$$dx = sqrt(diag((A'V^{-1}A)^{-1}*mse)).$$

*Приклад:*

Для заданих експериментальних точок будуються дві прямі: з використанням звичайного методу найменших квадратів і методу найменших квадратів із заданою коваріаційною матрицею.

```
>> x=[1:10]; % аргументи
>> y=[2.5 0.8 3.7 4.2 2.6 5.4 4.2 6.2 6.1 7.4];
% значення функції
>> m=length(x) % довжина даних
m =
    10
>> A=[x' ones(m,1)] % матриця A
A =
     1     1
     2     1
     3     1
     4     1
     5     1
     6     1
     7     1
     8     1
     9     1
    10     1
>> b=y'; % вектор b
>> p=lscov(A,b)
% параметри звичайної лінійної моделі
p =
    0.5848
    1.0933
```

```

>> fprintf('Звичайна лінійна модель:\ny=%8.6f*x%+8.6f\n',p);
Звичайна лінійна модель:
y=0.584848*x+1.093333
>> V=diag(1.0*ones(1,m))+diag(0.5*ones(1,m-1),1)+ diag(0.5*ones(1,m-1),-1);
% коваріаційна матриця
>> p1=lscov(A,b,V)
p1 =
    0.6836
    0.8000
>> disp('Лінійна модель із заданою коваріаційною матрицею:');
>> fprintf('y=%8.6f*x%+8.6f\n',p1);
Лінійна модель із заданою коваріаційною матрицею:
y=0.683636*x+0.800000
>> xplot=[x(1) x(end)]; % аргументи для графіка
>> yplot=polyval(p',xplot); % ординати лінійної моделі
>> yplot1=polyval(p1',xplot); % ординати лінійної моделі із заданої матрицею V
>> plot(x,y,'ks',xplot,yplot,'k-',xplot,yplot1,'k:');
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14) % шрифт
>> title('\bfДві лінійні моделі') % заголовок
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfy') % мітка осі OY
>> grid on, box on

```

На рис. 12.82 суцільною лінією показана пряма, знайдена за допомогою звичайного методу найменших квадратів, а штриховою – коли спостереження не є незалежними й мають наступну тридіагональну коваріаційну матрицю:

$$K = \begin{pmatrix} 1 & 0.5 & 0 & L & 0 \\ 0.5 & 1 & 0.5 & L & 0 \\ 0 & 0.5 & 1 & L & 0 \\ L & L & L & L & L \\ 0 & 0 & 0 & L & 1 \end{pmatrix}$$

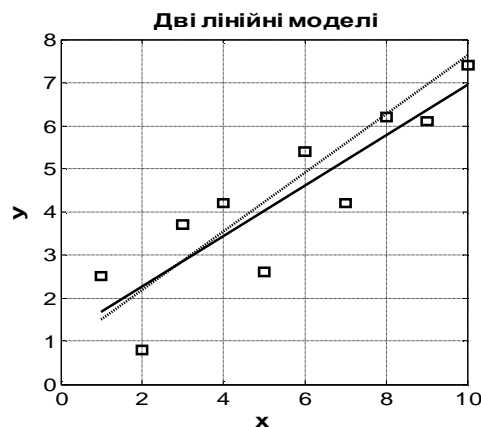


Рис. 12.82. Вибірка й дві лінійні моделі

**LSQNONNTG** Метод найменших квадратів з невід’ємними коефіцієнтами

Іноді в ході побудови лінійної моделі за методом найменших квадратів на оцінки параметрів накладається додаткова вимога невід’ємності. Цю задачу вирішує дана функція.

*Синтаксис:*

```
x = lsqnonneg(C,d)
x = lsqnonneg(C,d,x0)
x = lsqnonneg(C,d,x0,options)
[x,resnorm] = lsqnonneg(...)
[x,resnorm,residual] = lsqnonneg(...)
[x,resnorm,residual,exitflag] = lsqnonneg(...)
[x,resnorm,residual,exitflag,output] = lsqnonneg(...)
[x,resnorm,residual,exitflag,output,lambda] = lsqnonneg(...)
```

*Опис:*

Функція **x = lsqnonneg(C,d)** для заданих матриці *C* і вектора *d* повертає вектор *x*, мінімізуючий  $norm(C*x - d)$  за обмежень  $x \geq 0$ . Аргументи *C* і *d* повинні бути дійсними.

Функція **x = lsqnonneg(C,d,x0)** у третьому аргументі  $x_0$  дозволяє задати початкову точку для пошуку оптимуму; всі проєкції цієї точки мають бути невід’ємними  $x_0 \geq 0$ . Для використання значення за замовчуванням можна вилучити цей аргумент або задати порожній масив [], якщо потрібно задавати наступні аргументи.

Функція **x = lsqnonneg(C,d,x0,options)** у четвертому аргументі *options* дозволяє задати різні параметри оптимізації. Цей аргумент є структурою, значення якої за замовчуванням можна одержати за допомогою функції *optimset*:

```
>>options=optimset('lsqnonneg')
options =
Display: 'notify'           Hessmult: []           Nodedisplayinterval: []
Maxfunvals: []             Hesspattern: []       Nodesearchstrategy: []
Maxiter: []                Hessupdate: []        Nonleqnalgorithm: []
Tolfun: []                 Initialhesstype: []   Nostopifflatinfeas: []
Tolx:                      Initialhessmatrix: [] Phaseonetotalscaling: []
```

'10*eps*norm(c,1)*length(c)'	Jacobian: []	Preconditioner: []
Funvalcheck: []	Jacobmult: []	Precondbandwidth: []
Outputfcn: []	Jacobpattern: []	Rellinesrchbnd: []
Activeconstrtol: []	Largescale: []	Rellinesrchbndduration: []
Branchstrategy: []	Levenbergmarquardt: []	Showstatuswindow: []
Derivativecheck: []	Linesearchtype: []	Simplex: []
Diagnostics: []	Maxnodes: []	Tolcon: []
Diffmaxchange: []	Maxpcgiter: []	Tolpcg: []
Diffminchange: []	Maxrlpiter: []	Tolrlpfun: []
Goalsexactachieve: []	Maxsqpiter: []	Tolxinteger: []
Gradconstr: []	Maxtime: []	Typicalx: []
Gradobj: []	Meritfunction: []	
Hessian: []	Minabsmax: []	

Функція **Isqnonneg** використовує лише два поля цієї структури: *Display* (рівень повідомлень) і *Tolx* (точність щодо аргументів).

Функція **[x,resnorm] = Isqnonneg(...)** у другому результативному параметрі *resnorm* повертає квадрат норми нев'язок  $norm(C*x - d)^2$ .

Функція **[x,resnorm,residual] = Isqnonneg(...)** у третьому результативному параметрі *residual* повертає вектор нев'язок  $(C*x - d)$ .

Функція **[x,resnorm,residual,exitflag] = Isqnonneg(...)** у четвертому результативному параметрі *exitflag* повертає умову закінчення процесу оптимізації у функції **Isqnonneg**. Можливі значення цього параметра:

1 – процес оптимізації зійшовся до одержання рішення;

0 – перевищена максимально припустима кількість ітерацій. У цьому випадку може допомогти зменшення величини *options.Tolx*.

Функція **[x,resnorm,residual,exitflag,output] = Isqnonneg(...)** у п'ятому результативному параметрі *output* повертає структуру з такими полями:

*output.iterations* – кількість проведених ітерацій;

*output.algorithm* – використаний алгоритм оптимізації;

*output.message* – повідомлення про результати роботи.

**[x,resnorm,residual,exitflag,output,lambda] = Isqnonneg(...)** – у цьому синтаксисі шостий результативний параметр *lambda* повертає вектор двоїстих змінних (множників Лагранжа).

У прикладі для заданих експериментальних точок будуються дві прямі: з використанням звичайного методу найменших квадратів

(суцільна лінія) і методу найменших квадратів з невід'ємними параметрами (штрихова лінія). Вони показані на рис. 12.83.

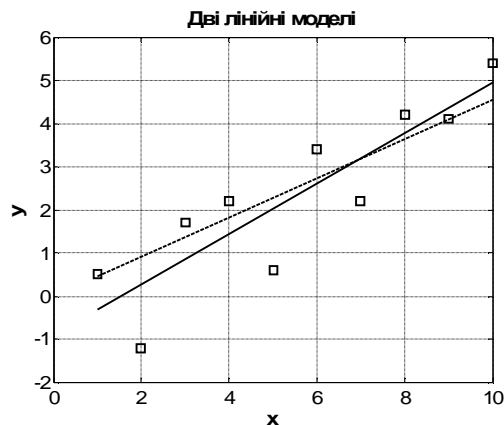


Рис. 12.83. Вибірка й дві лінійні моделі

Оператори:

```
>> clear all % очищаємо пам'ять
>> x=[1:10]; % аргументи
>> y=[2.5 0.8 3.7 4.2 2.6 5.4 4.2 6.2 6.1 7.4]-2; % значення функції
>> m=length(x); % довжина даних
>> C=[x' ones(m,1)]; % матриця C
>> d=y'; % вектор d
>> p=lscov(C,d); % параметри звичайної лінійної моделі
>> fprintf('Звичайна лінійна модель:\ny=%8.6f*x%+8.6f\n',p);
>> p1=lsqnonneg(C,d);
>> disp('Лінійна модель із невід'ємними параметрами:');
>> fprintf('y=%8.6f*x%+8.6f\n',p1);
>> xplot=[x(1) x(end)]; % аргументи для графіка
>> yplot=polyval(p',xplot); % ординати лінійної моделі
>> yplot1=polyval(p1',xplot); % ординати лінійної моделі з b>=0
>> plot(x,y,'ks',xplot,yplot,'k-',xplot,yplot1,'k:');
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfДві лінійні моделі') % заголовок
>> xlabel('\bfx'), ylabel('\bfy') % мітки осей OX, OY
>> grid on, box on
Звичайна лінійна модель:
y=0.584848*x-0.906667
Лінійна модель із невід'ємними параметрами:
y=0.455325*x+0.000000
```

Можна ці результати порівняти з отриманими для цих же даних за використанням функції **lscov** (див. вище)

## 12.6. Узагальнені лінійні моделі

<b>GLMFIT</b> Добір узагальненої лінійної моделі .....	212
<b>GLMVAL</b> Розрахунок значень залежної змінної за узагальненою регресійною моделлю .....	223

### **GLMFIT** Добір узагальненої лінійної моделі

За заданими експериментальними даними будемо узагальнену лінійну модель для обраного виду розподілу. Далі за допомогою функції `glmval` (дивися наступну функцію) можна обчислити значення функції відгуку в проміжних точках.

*Синтаксис:*

```
b = glmfit(X,Y,'distr')  
b = glmfit(X,Y,'distr','link','estdisp',offset,pwts,'const')  
[b,dev,stats] = glmfit(...)
```

*Опис:*

Функція `b = glmfit(X,Y,'distr')` призначена для оцінки параметрів узагальненої лінійної регресійної моделі. Вхідними аргументами функції є:  $Y$  – залежна змінна,  $X$  – матриця незалежних змінних, *'distr'* – рядкова константа, що визначає вид розподілу залежної змінної.

Можливі види розподіли залежної змінної перелічені в табл. 12.15.

Для всіх розподілів, крім біноміального, залежна змінна  $Y$  задається як вектор. Для біноміального закону  $Y$  має бути визначене як матриця із двома стовпцями: у першому стовпці задається число сприятливих подій, у другому – число повторних незалежних випробувань. Матриця незалежних змінних  $X$  повинна містити таку ж кількість рядків, що й  $Y$ .

Таблиця 12.15

### Види розподілу залежної змінної

Значення <i>'distr'</i>	Вид розподілу
<i>'binomial'</i>	Біноміальний
<i>'gamma'</i>	Гама
<i>'inverse gaussian'</i>	Зворотний розподіл Гауса

<i>'lognormal'</i>	Логнормальний
<i>'normal'</i>	Нормальний (значення за замовчуванням)
<i>'poisson'</i>	Пуасона

Вихідна змінна  $b$  – вектор точкових оцінок коефіцієнтів лінійної узагальненої регресійної моделі. Цей варіант синтаксису виклику функції **glmfit** використовує канонічний взаємозв'язок (див. нижче) параметрів розподілу з незалежними змінними.

У функції **b = glmfit(X,Y,'distr','link','estdisp',offset,pwts,'const')** вхідні аргументи *'link'*, *'estdisp'*, *offset*, *pwts*, *'const'* призначені для керування процесом оцінки параметрів регресійної моделі. Вхідний аргумент *'link'* задає вид взаємозв'язку між параметром розподілу  $\mu$  й оцінюваною лінійною комбінацією незалежних змінних  $Xb$ .

В табл. 12.16 перелічені стандартні значення *'link'*.

Таблиця 12.16

### Значення параметру *'link'* функції glmfit

Значення <i>'link'</i>	Вид залежності	Взаємозв'язок за замовчуванням	
<i>'identity'</i>	$\mu = xb$	<i>'normal'</i>	
<i>'log'</i>	$\log(\mu) = xb$	<i>'poisson'</i>	
<i>'logit'</i>	$\log(\mu / (1 - \mu)) = xb$	<i>'binomial'</i>	logit – логіт-аналіз probit – пробіт-аналіз
<i>'probit'</i>	$norminv(\mu) = xb$		
<i>'comploglog'</i>	$\log(-\log(1 - \mu)) = xb$		
<i>'logloglink'</i>	$\log(-\log(\mu)) = xb$		
<i>'reciprocal'</i>	$1 / \mu = xb$	<i>'gamma'</i>	
$p$ (число)	$\mu^p = xb$	<i>'inverse gaussian'</i> (якщо $p = -2$ )	

Реалізація іншого виду взаємозв'язків здійснюється за допомогою *inline* або *m*-функцій. Аргумент *'link'*, що визначає вид довільного взаємозв'язку, задається як масив рядків, що містить 3 елемента: 1-й елемент містить визначення функції взаємозв'язків; 2-й елемент – похідну від функції взаємозв'язку; 3-й елемент – її зворотну функцію.

Нижче наведено приклад формування масиву рядків *mylinks* функцій взаємозв'язків на підставі *inline*-функцій:

$$FL = inline('x^{-.5}') \Rightarrow y = \frac{1}{\sqrt{x}} - \text{функція};$$

$$FD = inline('-.5*x^{-1.5}') \Rightarrow y' = -\frac{1}{2\sqrt{x^3}} - \text{похідна};$$

$$FI = inline('x^{-2}') \Rightarrow y = \frac{1}{x^2} - \text{зворотна даної};$$

$$mylinks = \{FL FI FD\}.$$

Якщо використовуються  $m$ -функції, масив рядків формується за наступним правилом:  $mylinks = \{@FL @FD @FI\}$ .

Вхідний аргумент *'estdisp'* дозволяє задати величину дисперсії вибірки для біноміального закону й розподілу Пуасона. Якщо *'estdisp'='on'*, то величина дисперсії оцінюється за вибірковим даними  $X, Y$ . Для *'estdisp'='off'* точкова оцінка дисперсії приймається рівною 1. Для інших розподілів в **glmfit** значення дисперсії розраховується у всіх випадках.

Аргумент *offset* є спеціальною незалежною змінною з коефіцієнтом у регресійній моделі, рівним одиниці. Як приклад використання *offset*, можна розглянути моделювання числа дефектів на різних поверхнях. Потрібно одержати регресійну модель, у якої залежною змінною є відносна кількість дефектів на одиницю площі поверхні. У цьому випадку кількість дефектів буде використовуватися як залежна змінна, розподілена за законом Пуасона, логарифмічна функція (*log*) буде використана для задання виду взаємозв'язку між параметром розподілу й оцінюваною лінійною комбінацією незалежних змінних (параметр *'link'*), і в якості параметра *offset* буде виступати вектор логарифма від площі поверхні.

Аргумент *pwts* дозволяє задати вектор вагових значень залежної змінної. Наприклад, якщо значення залежної змінної  $Y(i)$  є середнім арифметичним від  $f(i)$  вимірів, то  $f$  може бути використаний як вектор вагових значень.

Вхідні параметри *offset* і *pwts* можуть бути задані як вектори або пропущені під час виклику функції. Порожні вектори *offset* і *pwts* будуть

тракуватися як пропущені вхідні параметри. Розмірність  $Y$ ,  $offset$  і  $pweights$  повинна збігатися.

Вхідний параметр *'const'* дозволяє в явному виді визначити, буде розраховуватися оцінка константи регресійної моделі *'const' = 'on'*, чи ні – *'const' = 'off'*. Значення за замовчуванням *'const' = 'on'*. Якщо необхідно одержати оцінку вільного члена регресійної моделі, то рекомендується використовувати *'const' = 'on'* замість завдання одиничного стовпця в матриці незалежних змінних.

Функція **[b,dev,stats] = glmfit(...)** повертає відхилення у векторі рішень *dev* і структуру *stats*. Відхилення у векторі рішень є узагальненням залишкової суми квадратів і використовуються для порівняння ряду регресійних моделей, склад коефіцієнтів однієї з яких є підмножиною коефіцієнтів іншої. У результаті проведеного порівняння має бути зроблене статистично значимий висновок, що похибка опису експериментальних даних регресійною моделлю з більшою кількістю коефіцієнтів є меншою, ніж у моделі з меншим числом коефіцієнтів. Таким чином, вибирається регресійна модель із найменшим числом коефіцієнтів, що забезпечує мінімальну похибку. Структура *stats* містить наступну інформацію:

*stats.dfe* – число ступенів свободи похибок регресійної моделі;

*stats.s* – теоретична або оцінена дисперсія параметрів;

*stats.sfit* – оцінка дисперсії параметра;

*stats.estdisp* = 1 – якщо оцінка дисперсії була розрахована, 0 – якщо оцінка дисперсії була задана;

*stats.beta* – вектор коефіцієнтів лінійної узагальненої регресійної моделі (дорівнює результативному параметру  $b$ );

*stats.se* – вектор стандартних похибок коефіцієнтів лінійної узагальненої регресійної моделі  $b$ ;

*stats.coeffcorr* – матриця коефіцієнтів кореляції коефіцієнтів  $b$ ;

*stats.t* – значення статистики t-Стьюдента для вектора  $b$ ;

*stats.p* – значення рівня значимості *p* статистики *t* Стьюдента для вектора коефіцієнтів *b*;

*stats.resid* – вектор залишків;

*stats.residp* – вектор залишків Пірсона;

*stats.residd* – вектор залишків узагальненої залишкової суми квадратів;

*stats.resida* – вектор залишків Анскомбе (Anscombe).

Якщо розраховується оцінка дисперсії параметра для біноміального закону або розподілу Пуасона, то *stats.s = stats.sfit*. Елементи вектора *stats.se* будуть відрізнятися від їх теоретичних значень на величину *stats.s*.

*Приклади:*

1. Розрахунок параметрів узагальненої лінійної регресійної моделі для 5 незалежних змінних, розподілених за нормальним законом, і залежної змінної, розподіленої за нормальним законом.

```
>> x=normrnd(0,1,100,5); >> b=glmfit(x,y,distr)
>> y=normrnd(0,1,100,1); b =
>> distr='normal';      -0.0796      0.0073      -0.1033
                        -0.0456      -0.1262
                        0.1781
```

Аналітичне вираження моделі:

$$y = -0.0796 + 0.0073 \cdot x_1 - 0.0456 \cdot x_2 + 0.1781 \cdot x_3 - 0.1033 \cdot x_4 - 0.1262 \cdot x_5.$$

2. Розрахунок параметрів узагальненої лінійної регресійної моделі для 3-х незалежних змінних, розподілених за гама-законом. Вид розподілу залежної змінної заданий також як гама-розподіл. Вид взаємозв'язку між параметром розподілу  $\mu$  й оцінюваною лінійною комбінацією незалежних змінних  $Xb$  заданий як логарифмічний.

```
>> x=gamrnd(1,2,100,3); >> link='log';
>> y = gamrnd(1,2,100,1); >> b = glmfit(x,y,distr, link);
>> distr='gamma';      >> b'      ans =
                        0.581 0.000 -0.032 0.002
```

Аналітичне вираження моделі:

$$y = 0.5814 + 0.0004 \cdot x_1 - 0.0318 \cdot x_2 + 0.0021 \cdot x_3.$$

3. Розрахунок параметрів узагальненої лінійної регресійної моделі для 3-х незалежних змінних, розподілених за нормальним законом. Вид

розподілу залежної змінної заданий як нормальний. Залежність між параметром розподілу й оцінюваною лінійною комбінацією незалежних змінних задана як  $X^{1/2}$  у вигляді inline функцій.

```
>> x= normrnd (0,1,100,3);
>> y = normrnd (0,1,100,1);
>> distr='normal';
>> FL = inline('x.^.5');
>> FD = inline('.5*x.^-0.5');
>> FI = inline('x.^2');
>> link = {FL FI FD}
link =
[1x1 inline] [1x1 inline] [1x1 inline] | >> b = glmfit(x,y,distr,link)
Warning: Iteration limit reached
> In glmfit at 334
b =
1.1230 + 4.2739i
0.0081 - 0.2360i
0.0646 - 0.4430i
0.1076 - 0.8553i
```

У цьому прикладі отримано негативний результат. По-перше, у командному рядку з'явилося повідомлення: досягнута межа ітераційного процесу, не виконалася команда m-файлу glmfit.m, що прописана в рядку 142 (на це можна подивитися, відкривши файл у редакторі за зазначеному в наступному рядку адресі). По-друге, якщо подивитися на вибірку змінної  $X$ , помічаємо, що вона містить від'ємні значення й не може бути оброблена програмою (модель містить корені квадратні від незалежної змінної). Звідси висновок: модель обрана неправильно.

4. Розрахунок параметрів узагальненої лінійної регресійної моделі для 3-х незалежних змінних, розподілених за нормальним законом, і залежною змінною, розподіленою за законом Пуасона. Вид взаємозв'язку між параметром розподілу й оцінюваною лінійною комбінацією незалежних змінних заданий як логарифмічний.

```
>> x=normrnd(0,1,100,3);
>> y = poissrnd(0.5,100,1);
>> distr='poisson';
>> link='log';
>> estdisp='on';
>> b = glmfit(x,y,distr, link, estdisp) | b =
-0.7063
0.1773
-0.1130
-0.2340
```

Аналітичне вираження моделі:

$$y = -0.7063 + 0.1773 \cdot x_1 - 0.1130 \cdot x_2 - 0.2340 \cdot x_3 .$$

5. Розрахунок параметрів узагальненої лінійної регресійної моделі для 3-х незалежних змінних, розподілених за нормальним законом, і залежною змінною, розподіленою за законом Пуасона. Вид взаємозв'язку між параметром розподілу й оцінюваною лінійною комбінацією

незалежних змінних заданий як логарифмічний. У якості дисперсії вибірки прийнята одиниця. Задана спеціальна незалежна змінна *offset* як натуральний логарифм від залежної змінної.

```
>> x=normrnd(0,1,100,3);
>> y=poissrnd(5,100,1);
>> distr='poisson';
>> link='log';
>> estdisp='off';
>> offset=log(y);
>> b=glmfit(x,y,distr,link,estdisp,offset)
```

b =
1.0e-015 *
0.1008
0.0001
0.0157
-0.0035

Усі коефіцієнти моделі вкрай малі й мають порядок  $10^{-15}$ . Модель неприйнятна.

6. Розрахунок параметрів узагальненої лінійної регресійної моделі для 3-х незалежних змінних. Вид розподілу залежної змінної заданий як нормальний закон. Вид взаємозв'язку між параметром розподілу й оцінюваною лінійною комбінацією незалежних змінних заданий як логарифмічний. У якості додаткового параметра задано вектор вагових коефіцієнтів *pwts*. Вагові коефіцієнти розподілені за рівномірним законом. Розрахунок виконується без обліку вільного члена (константи) регресійної моделі.

```
>> x=1:1:10;
>> x=[x' x' x'];
>> xs=normrnd(0,1,10,3);
>> x=x+xs;
>> y=1:1:10;
>> ys=normrnd(0,1,10,1);
>> y=y+ys';
>> distr='normal';
>> link='identity';
```

>> estdisp=[];
>> offset = [];
>> pwts=unidrnd(10,10,1);
>> const='off';
>> b=glmfit(x,y,distr,link,estdisp,offset,pwts,const)
b =
-0.0282
0.7478
0.3297

Аналітичний вираз моделі:  $y = -0.0282x_1 + 0.7478x_2 + 0.3297x_3$ .

7. Розрахунок параметрів узагальненої лінійної регресійної моделі для 3-х незалежних змінних. Вид розподілу залежної змінної заданий як нормальний. Функція **glmfit** повертає вектор коефіцієнтів регресійної моделі, відхилення у векторі рішень *dev* і структуру *stats*.

```

>> x=normrnd(0,1,10,3);
>> y =normrnd(0,1,10,1);
>> distr='normal'
>> [b,dev,stats] = glmfit(x,y,distr)
b =
    -0.2641
    -0.0104
     0.2923
     0.2356
dev =
    2.1690

stats =
    coeffcorr: [4x4 double]
    t: [4x1 double]
    p: [4x1 double]
    resid: [10x1 double]
    residp: [10x1 double]
    residd: [10x1 double]
    resida: [10x1 double]
    beta: [4x1 double]
    dfe: 6
    sfit: 0.6012
    estdisp: 1
    s: 0.6012
    se: [4x1 double]

```

Результативна змінна *stats* містить різноманітну статистику прямих і прихованих обчислень таких як конкретні значення змінних, вид структури змінних, розміри матриць і іншу інформацію. Значення полів структури *stats* можна прочитати, викликавши їх прямо в командному рядку.

```

>> stats.beta
% Параметри регресійної моделі
ans =
    -0.2641
    -0.0104
     0.2923
     0.2356

>> stats.coffcorr
% Кореляційна матриця
ans =
    1.0000  0.2628  0.1062  -0.2891
    0.2628  1.0000  0.1491   0.2243
    0.1062  0.1491  1.0000  -0.1506
   -0.2891  0.2243 -0.1506  1.0000

>> stats.resid % Залишки моделі
ans =
    0.2618
   -0.6770
    0.7166
   -0.1264
    -0.2909
     0.3721
     0.3868
    -0.0347
     0.2220
    -0.8304

```

8. Розглянемо приклад узагальненої лінійної регресії для числа автомобілів, що задовольняють вимогам пробігу з витратою палива в 1 галон, залежно від їхньої ваги. Вибірка експериментальних значень задана наступними випадковими величинами:  $w$  – маса автомобіля у фунтах, *poor* – кількість автомобілів відповідних специфікацій із числа використаних під час проведення випробувань для кожного значення  $w$ , *total* – загальне число автотранспортних засобів для кожної величини маси, використаних на випробуваннях. Порівнюються дві моделі

біноміального розподілу: *'logit'* і *'probit'* і вибирається краща з них. Далі для кращої із двох моделей спробуємо врахувати квадратичний доданок і оцінити результат за допомогою  $\chi^2$ -критерія. І, нарешті, відносно кращої моделі проводимо остаточне дослідження й оцінюємо значущість коефіцієнтів.

```
>> w = (2100:200:4300)'
w =
    2100
    2300
    2500
    2700
    2900
    3100
    3300
    3500
    3700
    3900
    4100
    4300
>> poor = [1 2 0 3 8 8 14 17 19 15 17 21]';
>> total = [48 42 31 34 31 21 23 23 21 16 17 21]';
```

Оскільки в задачі розглядається число вдалих результатів за повторних незалежних випробувань, то в якості закону розподілу доцільно прийняти біноміальний розподіл. У якості регресійних моделей на початковому етапі узагальненого лінійного регресійного аналізу приймаються залежності *logit* і *probit*.

```
>> [bl,dl,sl] = glmfit(w,[poor total],'binomial');
>> [bp,dp,sp] = glmfit(w,[poor total],'binomial','probit');
>> dl
dl =
    6.4842
    >> dp
dp =
    7.5693
```

Значення відхилення у векторі рішень моделі *logit* менше ніж відхилення для залежності *probit*, тобто  $dl < dp$ . Незважаючи на те, що це відношення не є формальним загальноприйнятим тестом, цей результат дає підставу розглядати модель *logit* як більш кращу стосовно *probit*.

Розглянемо дві *logit* моделі. У першій моделі, розглянутої вище, використовується лінійний ефект незалежної змінної  $w$ . У якості другої приймемо модель із квадратичним і лінійним ефектами  $w$ . Для порівняння регресійних залежностей буде використаний тест на основі статистики  $\chi^2$ . Нульова гіпотеза полягає в тому, що якщо ефект квадратичного ефекту  $w$  статистично незначимий, то рівень значимості буде менше критичного значення 0.05. Значення рівня значимості розраховується для правобічної критичної області розподілу  $\chi^2$ . У якості

параметрів закону  $\chi^2$  за розрахунку рівня значущості виступають різниця відхилень у векторах рішень для розглянутих моделей і її ступінь свободи, що дорівнює 1.

Розрахунок відхилення у векторі рішень для другої *logit* моделі:

```
>> [b2,d2,s2] = glmfit([w w.^2],[poor total],'binomial');  
>> d2  
d2 =  
    5.7815
```

Величина різниці відхилень у векторах рішень для розглянутих моделей:

```
>> dl-d2  
ans =  
    0.7027
```

Розрахунок рівня значущості для різниці відхилень у векторах рішень для розглянутих моделей:

```
>> chi2cdf(dl-d2,1)  
ans =  
    0.5981
```

Оскільки отримане значення рівня значимості 0.5981 більше критичного 0.05, то немає підстав відкинути нульову гіпотезу, тобто квадратичний ефект маси автотранспортного засобу *w* є статистично незначимим в порівнянні з лінійним. Відповідно, немає підстав вважати повну регресійну модель із обліком квадратичного й лінійного ефектів маси автотранспортних засобів більш переважною, ніж лінійну модель.

Нижче наведено значення коефіцієнтів *bl* лінійної моделі, стандартних похибок *sl.se*, *t*-статистик *sl.t*, рівнів значимості *sl.p* для цих коефіцієнтів:

```
>> [bl sl.se sl.t sl.p]  
ans =  
%      bl      sl.se      sl.t      sl.p  
-13.3801  1.3940 -9.5986  0.0000  
    0.0042  0.0004  9.4474  0.0000
```

Отримати ці відповіді можна так:

```
>> w = (2100:200:4300)'; % вага автомобілів  
total = [48 42 31 34 31 21 23 23 21 16 17 21]'; % загальна кількість  
poor = [1 2 0 3 8 8 14 17 19 15 17 21]'; % кількість авто з малим пробігом  
logit='logit'; % назва моделі logit
```

```

probit='probit'; % назва моделі probit
[bl,dl,sl] = glmfit(w,[poor total],'binomial',logit); % модель logit
[bp,dp,sp] = glmfit(w,[poor total],'binomial',probit); % модель probit
fprintf('Похибка моделі binomial-%s: d=%7.5f.\n',logit,dl,probit,dp);
if dl<dp,
    bestd=dl;
    bestmodel=logit; % модель logit переважніше
else
    bestd=dp;
    bestmodel=probit; % модель probit переважніше
end
fprintf('Модель %s переважніше.\n',bestmodel);
[b2,d2,s2] = glmfit([w w.^2],[poor total],'binomial',bestmodel);
gros=bestd-d2; % поліпшення для квадратичних ваг
chi2gr=chi2cdf(gros,1); % функція chi2-розподілу для нього
fprintf('Різниця відхилень =%7.5f.\n',gros);
fprintf('Функція chi2-розподілу для нього =%7.5f.\n',chi2gr);
if chi2gr>gros,
    bestp=2;
    disp('Квадратичне зважування w застосовувати потрібно');
else
    bestp=1;
    disp('Квадратичне зважування w застосовувати не потрібно');
end
wp=[];
for k=1:bestp, % формуємо матрицю даних
    wp=[wp w.^k];
end
[b,d,s] = glmfit(wp,[poor total],'binomial',bestmodel);
fprintf(' Краща модель:\n Коефіцієнти моделі  р-значення\n');
fprintf(' %16.12f  %16.12f\n',[b s.p]);
q=0.1; % рівень значимості
if s.p(1)>q,
    disp(' Постійний доданок ураховувати не потрібно');
else
    disp(' Постійний доданок ураховувати потрібно');
end
if s.p(2)>q,
    disp(' Лінійний член ураховувати не потрібно');
else
    disp(' Лінійний член ураховувати потрібно');
end
if length(s.p)>2,
    if s.p(3)>q,
        disp(' Квадратичний член ураховувати не потрібно');
    else
        disp(' Квадратичний член ураховувати потрібно');
    end
end
Похибка моделі binomial-logit: d=6.48422.
Похибка моделі binomial-probit: d=7.56930.

```

Модель logit переважніше.  
Різниця відхилень =0.70269.  
Функція  $\chi^2$ -розподілу для нього =0.59812.  
Квадратичне зважування w застосовувати не потрібно  
Краща модель:  
Коефіцієнти моделі р-значення  
-13.380147448676 0.000000000000  
0.004181184746 0.000000000000  
Постійний доданок ураховувати потрібно  
Лінійний член ураховувати потрібно

Отримані результати дозволяють зробити висновок про неможливість далі спрощувати отриману лінійну модель. Обидва коефіцієнти – константа і коефіцієнт при лінійному ступені  $x$  – значимо відрізняються від нуля, оскільки в обох випадках величина рівня значимості дорівнює  $0.0000 < 0.01$ .

### **GLMVAL** Розрахунок значень залежної змінної за узагальненою регресійною моделлю

Виконується прогнозування з використанням узагальненої лінійної моделі. Ця функція працює в парі з **glmfit** (дивися вище) і повинна викликатися після неї. Вона за знайденими параметрами узагальненої лінійної моделі обчислює значення функції відгуку в будь-яких точках (більш точно – у проміжних точках, менш точно – поза тою множиною даних, за якою були визначені параметри моделі).

*Синтаксис:*

```
yfit = glmval(b,X,'link')  
[yfit,dlo,dhi] = glmval(b,X,'link',stats,clev)  
[yfit,dlo,dhi] = glmval(b,X,'link',stats,clev,N,offset,'const')
```

*Опис:*

Функція **yfit = glmval(b,X,'link')** призначена для розрахунку залежної змінної *yfit* для значень незалежної змінної  $X$  на підставі вектора коефіцієнтів лінійної узагальненої регресійної моделі  $b$  і функції взаємозв'язку *'link'*. У загальному випадку вектор коефіцієнтів регресійної моделі  $b$  розраховується за допомогою функції **glmfit**.

Вхідний аргумент *'link'* задає вид взаємозв'язку між параметром розподілу й оцінюваною лінійною комбінацією незалежних змінних. Параметр *'link'* задається відповідно до вимог до такого ж вхідного аргументу функції **glmfit**. Передбачені наступні значення *'link'* (табл. 12.17):

Таблиця 12.17

**Значення параметру *'link'***

Значення <i>'link'</i>	Вид залежності
<i>'identity'</i>	$\mu = xb$
<i>'log'</i>	$\log(\mu) = xb$
<i>'logit'</i>	$\log(\mu / (1 - \mu)) = xb$
<i>'probit'</i>	$norminv(\mu) = xb$
<i>'comploglog'</i>	$\log(-\log(1 - \mu)) = xb$
<i>'logloglink'</i>	$\log(-\log(\mu)) = xb$
<i>'reciprocal'</i>	$1 / \mu = xb$
<i>p</i> (число)	$\mu^p = xb$

Крім наведених вище значень *'link'* можна задати довільний вид залежності за допомогою inline-функцій або m-файлів. Правила й приклади використання такого способу див. в опису функції **glmfit**.

Результативний параметр *yfit* є значенням зворотної функції взаємозв'язку лінійної комбінації *Xb*.

Функція **[yfit,dlo,dhi] = glmval(b,X,'link',stats,clev)** – у цьому варіанті синтаксису повертає нижнє *dlo* і верхнє *dhi* відхилення від *yfit*, тобто границі довірчого інтервалу для параметрів закону розподілу при заданих *b*, *X*, *'link'*. Для розрахунку *dlo*, *dhi* використовується структура *stats*, що отримана як результативний параметр функції **glmfit**. Вхідний параметр *clev* задає довірчу ймовірність для розрахунку границь довірчого інтервалу. За замовчуванням *clev* приймається рівною 0.95, що відповідає 95%-у довірчому інтервалу. Границі довірчого інтервалу розраховуються як  $[yfit - dlo, yfit + dhi]$ .

У функції **[yhat,dlo,dhi] = glmval(beta,X,'link',stats,clev,N,offset,'const')** додаткові вхідні параметри *N*, *offset*, *'const'* потрібні для забезпечення

однакових умов з розрахунками, проведеними за допомогою функції **glmfit**. За умови використання біноміального закону для роботи з **glmfit** необхідно задати вхідний параметр  $N$ , відповідний до числа повторних незалежних випробувань. Якщо для виклику функції **glmfit** використовувалися параметри *offset* і *'const'*, то необхідно для виклику функції **glmval** використовувати ці ж параметри з тими ж значеннями.

*Приклади:*

Звичайна лінійна регресійна модель включає добір лінійної залежності або функції, у яку параметри моделі входять лінійно, а похибки даних розподілені за нормальним законом. Проте, така модель не завжди реалістично описує ситуацію. Узагальнені лінійні моделі включають, як би, два види лінійних моделей. Перша припускає лінійність залежності результативної ознаки (рівняння регресії) відносно параметрів, друга припускає лінійність деякого параметра розподілу результативної ознаки, причому цей розподіл необов'язково нормальний.

Функція **glmfit** за даними знаходить параметри узагальненої лінійної моделі, а **glmval** за даними параметра цієї моделі обчислює значення функції відгуку в будь-яких точках. У цій функції аргумент *link* визначає відношення між параметром зсуву обраного розподілу  $\mu$  і підбраною лінійною комбінацією  $Xb$ . У таблиці 12.17 наведені можливі значення параметра *link*. Якщо жоден із цих канонічних зв'язків не підходить, можна визначити іншу функцію зв'язку користувача й задати її в аргументі *link*.

1. Регресійна модель визначає характеристики розподілу залежної змінної (позначаємо  $y$ ) від однієї або більше незалежних змінних (позначаємо  $x_1, x_2$  і так далі). Звичайна лінійна регресія визначає  $y$  як випадково розподілені набори чисел, середнє значення яких лінійно залежать від змінних  $x_i$ :  $y_p = b_0 + b_1 * x_1 + b_2 * x_2 + \dots$  і чиї дисперсії постійні. У найпростішому випадку однієї змінної модель являє собою пряму лінію з гаусовським розподілом поблизу кожної точки.

Набираємо команди.

```
>> clear all
>> mu=@(x)-1.9+.23*x; % лінійна функцію зв'язку з користувачьким покажчиком
```

```

>> x=5:1:15; % 101 значення змінної x
>> yhat=mu(x); % ці значення x будемо використовувати для визначення y
>> dy=-3.5:1:3.5; % задаємо 71 значення y
sz=size(dy) % перевіряємо кількість точок
sz =
    1    71
k=(length(dy)+1)/2 % кількість точок для графіка
k =
    36
>> x1=repmat(7,sz); % задаємо значення x1=7
>> y1=mu(x1)+dy; % 71 значення y(x1) з обліком заданої лінійної залежності mu
z1=normpdf(y1,mu(x1),1); % обчислюємо 71 значення функції щільності розподілу
нормального закону з математичним очікуванням, що дорівнює 7, і середнім
квадратичним відхиленням, що дорівнює 1
>> x2=repmat(10,sz); % задаємо x2=10
>> y2=mu(x2)+dy; z2=normpdf(y2,mu(x2),1);
>> x3=repmat(13,sz); % задаємо x3=13
>> y3=mu(x3)+dy; z3=normpdf(y3,mu(x3),1);
plot3(x,yhat,zeros(size(x)),'k-', ...
x1,y1,z1,'k-',x1([k k]),y1([k k]),[0 z1(k)],'k:', ...
x2,y2,z2,'k-',x2([k k]),y2([k k]),[0 z2(k)],'k:', ...
x3,y3,z3,'k-',x3([k k]),y3([k k]),[0 z3(k)],'k:');
% будуємо 3 графіка кривих Гауса й пряму лінію регресії
>> zlim([0 1]); % обмежуємо значення z
>> xlabel('X'); ylabel('Y'); zlabel('Щільність імовірності');
>> grid on; view([-45 45]);

```

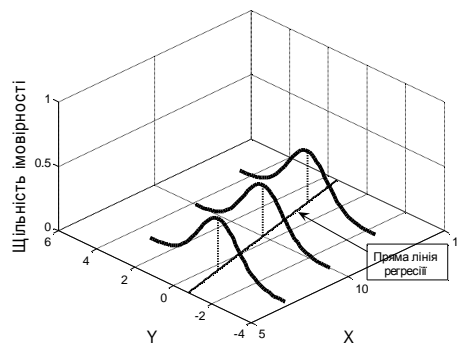


Рис. 12.84. Пряма лінія регресії

На рис. 12.84 побудовано графіки трьох функцій щільності ймовірності нормального закону з  $\mu = 7, 10, 13$  і  $\sigma = 1$ , а також лінію регресії як би в об'ємі.

2. Далі в прикладі розглядається регресія Пуасона з параметром 'link', рівним  $\log$ , і однією незалежною змінною  $x$ , що змінюється за експоненціальним законом з розподілом Пуасона поблизу кожної точки.

```

>> clear all
>> mu=@(x)exp(-1.9+.23*x);
>> x=5:1:15;
>> yhat=mu(x);
>> x1= repmat(7,1,5); y1=0:4; z1=poisspdf(y1,mu(x1));
>> x2= repmat(10,1,7); y2=0:6; z2=poisspdf(y2,mu(x2));
>> x3= repmat(13,1,9); y3=0:8; z3=poisspdf(y3,mu(x3));
>> plot3(x,yhat,zeros(size(x)),'k-', ...
[x1;x1],[y1;y1],[z1;zeros(size(y1))],'k',x1,y1,z1,'k.', ...
[x2;x2],[y2;y2],[z2;zeros(size(y2))],'k',x2,y2,z2,'k.', ...
[x3;x3],[y3;y3],[z3;zeros(size(y3))],'k',x3,y3,z3,'k. ');
>> zlim([0 1]);
>> xlabel('X'); ylabel('Y'); zlabel('Імовірність');
>> grid on; view([-45 45]);

```

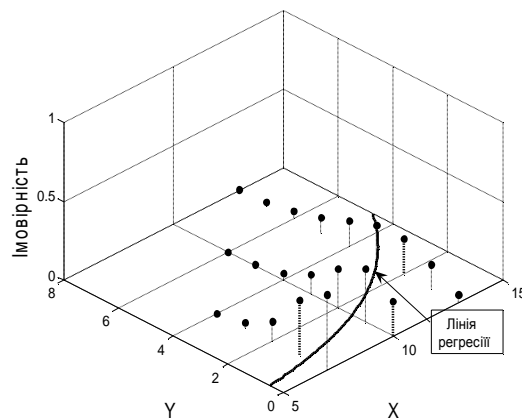


Рис. 12.85. Лінія регресії

На рис. 12.85 лінія регресії вже нелінійна.

3. Розраховується значення залежної змінної, середнього арифметичного  $y_{fit}$ , для значень незалежної змінної  $x_{new}$ . У якості моделі розподілу залежної змінної прийнято нормальний закон. За вид взаємозв'язку середнього арифметичного від лінійної моделі незалежної змінної прийнята логарифмічна залежність:  $\log(\mu) = xb$ . Також розраховуються границі 95%-го довірчого інтервалу залежної змінної.

```

>> x = 1:1:10;
>> y = 1:2:20;
>> y1 = normrnd(0,1,10,1);
>> y = y.^2+y1';
>> [b,dev,stats] = glmfit(x,y,'normal', 'log')

```

b =	2.8832
	0.3057
dev =	2.9747e+003

```
stats =
  beta: [2x1 double]
  dfe: 8
  sfit: 19.2832
  estdisp: 1
  s: 19.2832
  se: [2x1 double]
  coeffcorr: [2x2 double]
  t: [2x1 double]
  p: [2x1 double]
  resid: [10x1 double]
  residp: [10x1 double]
  residd: [10x1 double]
  resida: [10x1 double]
```

```
>> xnew = 0.5:1:10;
>> [yfit,ylo,yhi] = glmval(b,xnew, 'log',stats,0.95)
```

% Розрахункові значення yfit =	% Нижня довірча границя ylo =	% Верхня довірча границя yhi =
20.8229	7.3595	10.9996
28.2695	9.0451	12.8536
38.3792	10.9368	14.7808
52.1042	12.9484	16.6498
70.7376	14.9096	18.2538
96.0346	16.5393	19.3047
130.3783	17.4771	19.5000
177.0039	17.6222	18.9147
240.3036	18.7130	19.6356
326.2403	26.9608	28.4464

```
>> errorbar(xnew,yfit,ylo,yhi);
>> grid on, box on
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx'), ylabel('\bfy')
```

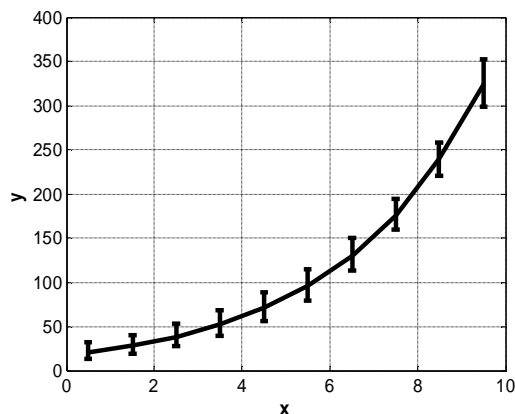


Рис. 12.86. Лінія розрахункових значень із інтервалами похибок

На рис. 12.86 представлено графік моделі й для 10 точок розраховано інтервали похибок.

Виведемо в графічне вікно рис. 12.87 зображені на рис. 12.86 точки й додамо туди довірчі границі для моделі:

```
>> plot(x,y, 's', xnew,yfit, 'r', xnew, yfit-ylo, 'b', xnew, yfit+yhi, 'b');
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx'), ylabel('\bfy'), grid on, box on
```

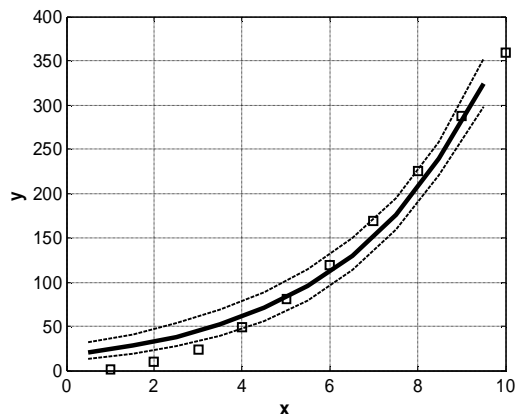


Рис. 12.87. Лінія регресії з довірчими границями й експериментальні точки

3. Для даних попереднього прикладу порівнюються моделі: лінійна  $y = b_0 + b_1X$  і квадратична  $y = b_0 + b_1X + b_2X^2$ :

На рис. 12.88 показано порівняльні лінійну й квадратичну регресійні моделі. Із-за близькості ліній, що побудовані програмою, масштаб було змінено вручну.

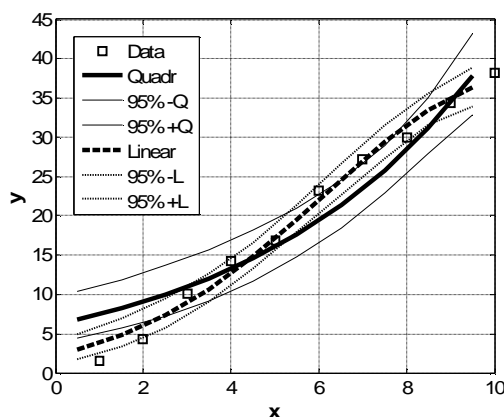


Рис. 12.88. Лінійна й квадратична моделі з довірчими границями

Ці графіки були побудовані операторами:

```
>> [b2,dev2,stats2] = glmfit([x' x.^2],y,'normal', 'log');
>> [yfit2,ylo2,yhi2] = glmval(b2, [xnew' xnew.^2], 'log',stats2,0.95);
>> plot(x,y,'s',xnew,yfit,'-',xnew,yfit-ylo,':',xnew,yfit+yhi,':',xnew,yfit2,'--',...
    xnew, yfit2-ylo2, ':', xnew, yfit2+yhi2, ':');
>> grid on, box on
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> xlabel('\bfx'), ylabel('\bfy')
```

4. Розглядається моделювання числа автомобілів з низькою витратою палива, як продовження прикладу 8 у описі попередньої функції. Побудуємо графік експериментальних даних.

```
>> weight=[2100:200:4300]; % маса
>> tested=[48 42 31 34 31 21 23 23 21 16 17 21]; % кількість випробовувань
>> failed=[1 2 0 3 8 8 14 17 19 15 17 21]; %число невдалих випробувань
>> proportion= failed./ tested;
>> plot(weight, proportion, 's'), grid
>> xlabel('Маса автомобіля'); ylabel('Відносне число')
```

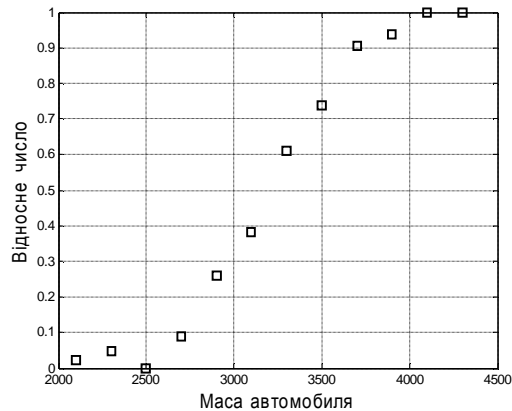


Рис. 12.89. Експериментальні дані

З рис. 12.89 виходить, що резонно припустити біноміальний розподіл з параметром імовірності  $p$ , який зростає з ростом ваги автомобіля. Але як визначити точно цю залежність?

Можна спробувати побудувати лінійну модель (рис. 12.90).

```
>> lincoef=polyfit(weight,proportion,1);
>> linfit=polyval(lincoef,weight);
>> plot(weight,proportion,'s',weight,linfit,'k-'), grid
>> xlabel('Маса автомобіля'); ylabel('Відносне число')
```

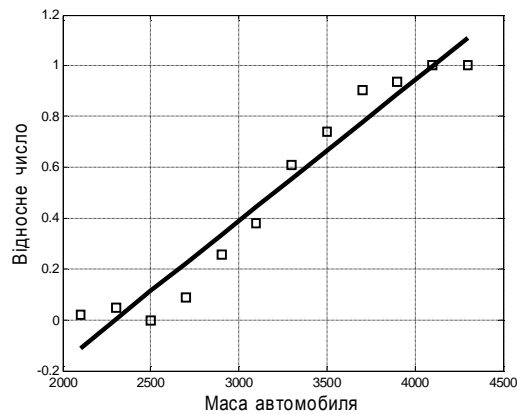


Рис.12.90. Пряма лінія регресії

З отриманого на рис. 12.90 результату виходять дві проблеми.

1. Прогноз  $p$  розширюється на від'ємні значення й на значення, що більші одиниці.

2. Змінна не має нормального розподілу, як це необхідно для моделі. Ці порушення умов вимагають удосконалення моделі.

Будемо поліпшувати модель. Залежність даних на рис.12.90 нагадує сигмоїдну криву. Застосуємо функцію **glmfit** для побудови логістичної регресійної моделі. Логістична модель є спеціальним випадком узагальненої лінійної моделі і вона краще, ніж лінійна модель, описує поведінку змінної із двох причин. По-перше, відпадає вимога про нормальний розподіл і можна використовувати біноміальний розподіл. По-друге, на залежну змінну відразу накладається логістичний зв'язок, що обмежує область значень відрізком  $[0; 1]$ .

Для застосування логістичної моделі створюємо матрицю, що містить у першому стовпці кількість автомобілів даної маси, що не пройшли випробування, а в другому – відповідну кількість випробуваних. Установлюємо біноміальний розподіл і значення аргументу *link*: 'logit'.

Далі для прогнозу застосуємо функцію **glmval**.

```
>> [logcoef,dev]=glmfit(weight,[failed; tested'],'binomial','logit');
>> logfit=glmval(logcoef,weight,'logit');
>> plot(weight,proportion,'bs',weight,logfit,'k-'), grid
>> xlabel('Маса автомобіля'); ylabel('Відносне число')
```

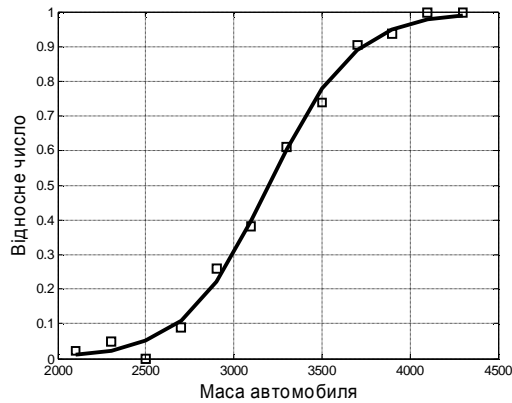


Рис. 12.91. Лінія регресії на основі логістичної моделі

Тепер з рис. 12.91 виходить, що залежна змінна – відносне число автомобілів, асимптотично прагне до нуля за умови зменшенні маси й до одиниці за умови збільшенні маси транспортного засобу, тобто модель правильно описує експериментальні дані.

Функція **glmfit** дозволяє виводити на екран вихідні аргументи для тестування застосованої моделі. Наприклад, можна порівняти величини дисперсій двох моделей, визначивши, чи потрібно включати квадратичний член у модель.

```
>> [logcoef2,dev2]=glmfit([weight; weight.^2],[failed; tested'],'binomial','logit');
>> pval=1-chi2cdf(dev-dev2,1)
pval =
    0.4019
```

Досить значна величина  $pval = 0.4019$  для наших даних говорить про те, що квадратичний член незначущий, тобто його включення в модель не принесе помітних змін. Це видно і на графіках.

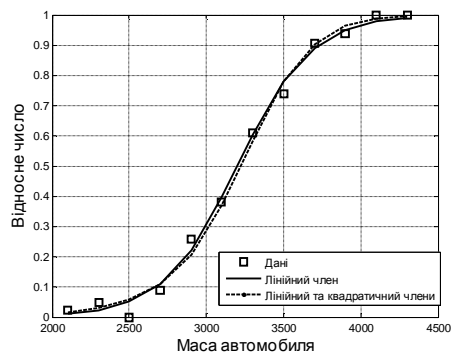


Рис.12.92. Лінії регресії з урахуванням лінійного, а також квадратичного членів

Рис. 12.92 було побудовано так:

```
>> logfit2=glmval(logcoef2,[weight; weight.^2'],'logit');  
>> plot(weight,proportion,'bs',weight,logfit,'k-',weight,logfit2), grid  
>> xlabel('Маса автомобіля'); ylabel('Відносне число')  
>> legend('Дані','Лінійний член','Лінійний та квадратичний члени')
```

Для перевірки якості апроксимації подивимось на розподіл помилок на графіку Пирсона за допомогою функції **normplot**. Ця функція нормалізує дані і якщо їх оцінки досить гарні, то похибки розподіляються за стандартним нормальним законом і точки розташовуються на діаграмі поблизу прямої. А якщо апроксимація не гарна, то точки будуть розташовуватися уздовж деякої кривої.

Перевіримо це для нашого прикладу.

```
>> [logCoef,dev,stats]=glmfit(weight,[failed; tested'],'binomial');  
>> normplot(stats.residp)
```

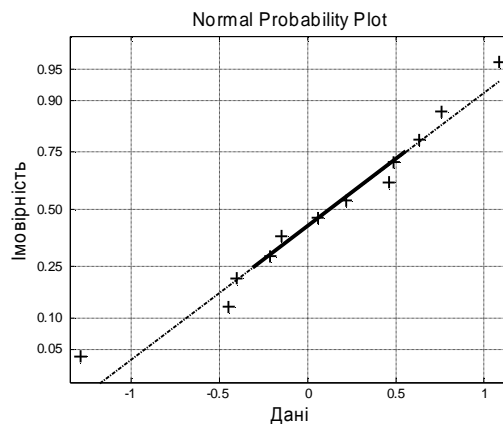


Рис.12.93. Розподіл похибок

Графік похибок (рис.12.93) показує гарний збіг з нормальним розподілом.

Розібравшись із моделлю, тепер можна зробити прогнози, включаючи обчислення довірчих інтервалів. Обчислимо очікувану кількість автомобілів з 100, які тестуються, що можуть не пройти тест, для чотирьох категорій їх мас.

```
>> weightpr=2500:500:4000;  
>> [failtp,dlo,dhi]=glmval(logcoef,weightpr,'logit',stats);  
>> errorbar(weightpr,failtp,dlo,dhi,'k-'), grid  
>> xlabel('Маса автомобіля'); ylabel('Відносне число')
```

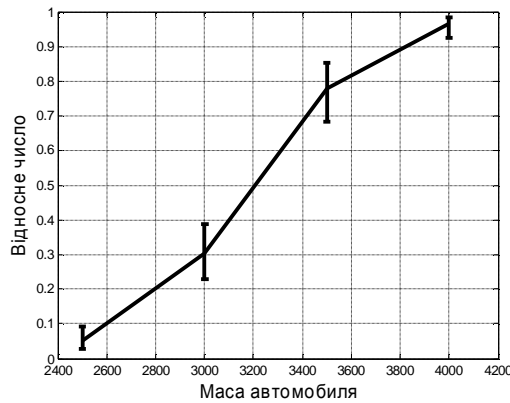


Рис. 12.94. Очікувана відносна кількість автомобілів з довірчими інтервалами

Розглянемо вплив значення функцій зв'язку *link*.

Для кожного з п'яти розподілів, які підтримує функція **glmfit**, є канонічне (за замовчуванням) значення функції зв'язку *link*. Для біноміального розподілу таким канонічним значенням є *logit*. Є ще три інших функції, які також застосовуються в біноміальній моделі. Усі чотири функції – *logit*, *probit*, *comploglog* і *loglog* – мають область значень відрізок [0; 1].

Їх аналітичні вираження:

$$\textit{logit}: \frac{1}{1 + \exp^{-x}} \text{ – логістична функція;}$$

$$\textit{probit}: \textit{normcdf}(x) \text{ – функція розподілу нормального закону;}$$

$$\textit{comploglog}: 1 - e^{-e^x};$$

$$\textit{loglog}: e^{-e^x}.$$

Для порівняння побудуємо їх графіки (рис. 12.95).

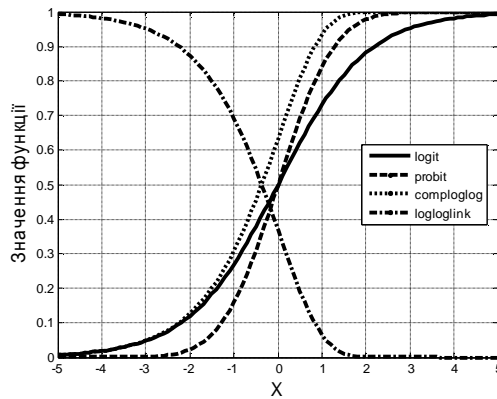


Рис. 12.95. Графіки функцій зв'язку

Оператори:

```
>> x=-5:1:5;
>> plot(x,1./(1+exp(-x)),x,normcdf(x),x,1-exp(-exp(x)),x,exp(-exp(x)));
>> xlabel('X'); ylabel('Значення функції')
>> legend('logit','probit','comploglog','logloglink'),grid
```

Наприкінці, порівняємо лінії регресії, отримані за допомогою logit-аналізу і probit-аналізу.

```
>> prcoef=glmfit(weight,[failed; tested]','binomial','probit');
>> prfit=glmval(prcoef,weight,'probit');
>> plot(weight,proportion,'bs',weight,logfit,'k', weight,prfit), grid
>> xlabel('Маса автомобіля'); ylabel('Відносне число')
>> legend('Дані',' Logit-модель',' Probit-модель')
```

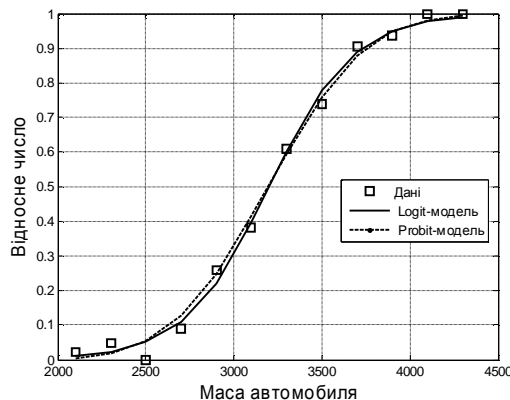


Рис. 12.96. Порівняння двох моделей

Рис. 12.96 свідчить про те, що для даного прикладу відмінність між логіт і пробіт моделями незначна. Але, можливо, в інших прикладах ця відмінність буде істотнішою.

Усі моделі, що розглянуті в цьому розділі були лінійними за параметрами (деякі з цих моделей – квадратичні, тобто нелінійні за пояснючими змінними-аргументами).

## 13. Нелінійні моделі

У цьому розділі описуються чотири функції для побудови регресійних моделей, що нелінійно залежать від параметрів апроксимації. Функція **lsqnonneg**, яка в деяких версіях **Statistics Toolbox** включена в цей розділ, у цьому довіднику описана в розділі 12.

<b>NLINFIT</b> Нелінійна регресія за методом найменших квадратів з використанням ітераційної процедури Гауса-Ньютона .....	237
<b>NLINTOOL</b> Інтерактивний нелінійний регресійний аналіз .....	244
<b>NLPARC</b> Розрахунок інтервальних оцінок коефіцієнтів нелінійного рівняння регресії .....	248
<b>NLPREDCI</b> Розрахунок довірчих інтервалів для нелінійної регресійної моделі .....	251

### **NLINFIT** Нелінійна регресія за методом найменших квадратів з використанням ітераційної процедури Гауса-Ньютона

За допомогою методу найменших квадратів вирішує задачу побудови нелінійної регресійної моделі.

*Синтаксис:*

```
beta = nlinfit(x,y,FUN,beta0)
[beta,r,J] = nlinfit(X,y,FUN,beta0)
[...] = nlinfit(X,y,fun,beta0,options)
```

*Опис:*

Функція **beta = nlinfit(x,y,FUN,beta0)** призначена для розрахунку точкових оцінок коефіцієнтів довільної нелінійної регресійної моделі методом найменших квадратів. Тут  $y$  – вектор значень залежної змінної,  $x$  – матриця значень незалежних змінних,  $beta_0$  – вектор початкових значень коефіцієнтів регресійної моделі,  $FUN$  – покажчик на функцію, що реалізує регресійну модель. Ця функція може бути створена або з використанням дескриптора @, або як m-файл із синтаксисом виклику

$yhat = myfun(beta, X)$ , або як inline-функція. Значення незалежних змінних визначаються стовпцями матриці  $x$ . Кількість рядків матриці  $x$  повинне дорівнювати числу елементів вектора  $y$ . Рівняння регресії задається як функція в наступному вигляді:  $yhat = FUN(beta, X)$ , де  $beta$  – вектор коефіцієнтів регресійної моделі,  $X$  – матриця незалежних змінних,  $yhat$  – вектор або матриця значень залежної змінної.

Вектор початкових значень коефіцієнтів регресійної моделі  $beta_0$  може бути заданий довільним масивом. Основна вимога – специфікація вектора  $beta_0$  повинна збігатися зі специфікацією вхідного аргументу  $beta$  функції  $FUN$ .

Функція **[beta,r,J] = nlinfit(X,y,FUN,beta0)** повертає точкові оцінки коефіцієнтів  $beta$ ,  $r$  – вектор залишків,  $J$  – матрицю Якобі. Параметри  $beta$ ,  $r$ ,  $J$  використовуються як вхідні аргументи функцій **nlpredci** і **nlparci** для розрахунку довірчих інтервалів регресійної моделі й довірчих інтервалів коефіцієнтів моделі відповідно.

Функція **[...] = nlinfit(X,y,fun,beta0,options)** в аргументі *options* дозволяє задати параметри алгоритму оптимізації, використовувані в процесі побудови нелінійної регресійної моделі. Цей аргумент є структурою, її значення з полями за замовчуванням можна згенерувати за допомогою функції **statset**:

<pre>&gt;&gt; options=statset('nlinfit') options =   Display: 'off'   Maxfunevals: []   Maxiter: 200   Tolbnd: []   Tolfun: 1.0000e-008</pre>	<pre>Tolx: 1.0000e-008 Gradobj: [] Derivstep: 6.0555e-006 Funvalcheck: 'on' Robust: 'off' Wgtfun: [] Tune: []</pre>
---	---

Тут:

*Display* – рівень повідомлень про хід рішення задачі:

*'off'* – повідомлення не виводиться;

*'iter'* – виводиться повідомлення на кожній ітерації;

*'final'* – виводиться фінальне повідомлення;

*Maxiter* – максимальна кількість ітерацій;

*Tolfun* – критерій виходу за сумою квадратів нев'язок;

*Tolx* – критерій виходу за приростами змінних *beta*;

*Derivstep* – відносна різниця для апроксимації градієнта; може бути скаляром або вектором такої ж довжини, як і *beta<sub>0</sub>*;

*Funvalcheck* – чи перевіряти значення цільової функції на наявність помилкових значень типу NaN або *Inf*; можливі значення 'off' і 'on'.

Функція **nlinfit** розглядає нечислові значення NaN в *y* або *fun(beta, X)* як помилкові й відкидає відповідні рядки.

*Приклади:*

1. Одновимірна модель. Розрахунок параметрів регресійної моделі

виду  $y = 5(1 - e^{-2x})$ :

```
>> x=[0:0.1:10]'; % абсциси
>> n=length(x) % число точок
n =
    101
>> y=5*(1-exp(-2*x))+normrnd(0,0.1,n,1); % експериментальні ординати
>> fun=inline('b(1)*(1-exp(-b(2)*x))','b','x')
fun =
    Inline function:
    fun(b,x) = b(1)*(1-exp(-b(2)*x))
>> b=nlinfit(x,y,fun,[1 1]) % знаходимо параметри
b =
    5.0020    2.0292
>> disp('Крива насичення:');
>> fprintf('y(x)=%7.4f*(1-exp(%+7.4f*x))\n',b(1),-b(2));
>> yt=fun(b,x); % теоретичні точки
>> plot(x,y,'k',x,yt,'k-'); % графік
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfКрива насичення') % заголовок
>> xlabel('\bfx'), ylabel('\bfy')
>> grid on, box on
Крива насичення:
y(x)= 5.0020*(1-exp(-2.0292*x))
```

На рис. 13.1 представлено графічно результати розрахунку параметрів моделі за вибірки зі 101 елементів. Рівняння регресії виведено на екран.

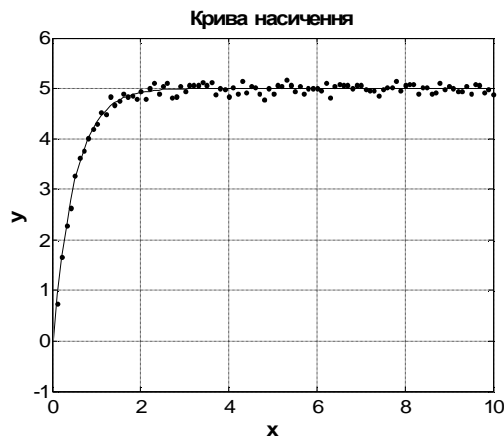


Рис. 13.1. Нелінійна лінія регресії

2. Двовимірна модель. Розрахунок параметрів неповної квадратичної регресійної моделі виду:  
 $Y = A(X_1)^2 + B(X_2)^2 + CX_1 + DX_2 + E$ , де  $A, B, C, D, E$  -- параметри регресійної моделі;  $Y$  – залежна змінна;  $X_1, X_2$  – незалежні змінні.

2.1. Створюємо m-файл функції (polynom.m), що реалізує квадратичну регресійну модель.

```
function yhat=polynom(beta,x)
% Поділ вектора коефіцієнтів beta за окремими змінними A,B,C,D,E
A=beta(1);
B=beta(2);
C=beta(3);
D=beta(4);
E=beta(5);
% Поділ матриці незалежних змінних x за окремими векторами x1, x2,
x1=x(:,1);
x2=x(:,2);
% Квадратична залежність
yhat = A.*x1.^2+B.*x2.^2+C.*x1+D.*x2+E;
```

Цей текст треба скопіювати в редактор m-файлів **MatLab** (File \ New \ M-File) і зберегти (File \ Save As) під назвою polynom.m у папку work.

2.2. Моделювання експериментальних даних

```
>> A=1;
>> B=-2;
>> C=-1.5;
>> D=2.6;
>> E=2;
>> X1=[1 1 10 10 1 5.5 10 5.5 5.5];
>> X2=[2 20 20 2 11 20 11 2 11];
```

```

>> Y = A.* X1.^2+B.*X2.^2+C.*X1+D.*X2+E;
>> XX1=normrnd(0,1,length(X1),1);
>> XX2=unifrnd(-0.1,0.1, length(X2),1);
>> YY=normrnd(0,2, length(Y),1);
>> X1= X1+ XX1';
>> X2= X2+ XX2';
>> Y= Y+ YY';

```

### 2.3. Початкові значення коефіцієнтів

```

>> beta0=[0.1 1 0 1 1];

```

### 2.4. Розрахунок точкових оцінок коефіцієнтів $A, B, C, D, E$ .

```

>> [beta,r,J]=nlinfit([X1' X2'],Y',@polynom,beta0)

```

beta =

```

0.5734 -1.9911 2.3351 2.0694 3.9873

```

```

r =
    -7.0880    2.2034   -5.3702
    4.2348    7.2384
    -2.0078   -2.2299

```

```

J =
    4.9615  122.0504  2.2274  11.0476  1.0000
    23.0764 397.4143  4.8038  19.9353  1.0000
    100.151 120.5855 10.0075 10.9811 1.0000
    22.2511  4.3560  4.7171  2.0871 1.0000
    37.0508 122.8413  6.0869 11.0834 1.0000

```

### 2.5. Графічне представлення отриманої залежності

```

>> [Xx1 Xx2]=meshgrid([1:0.5:10],[2:0.5:20]);
>> z= beta(1).*Xx1.^2+ beta(2).*Xx2.^2+ beta(3).*Xx1+ beta(4).*Xx2+ beta(5);
>> plot3(X1,X2,Y,'o',Xx1,Xx2,z)
>> grid on, box on

```

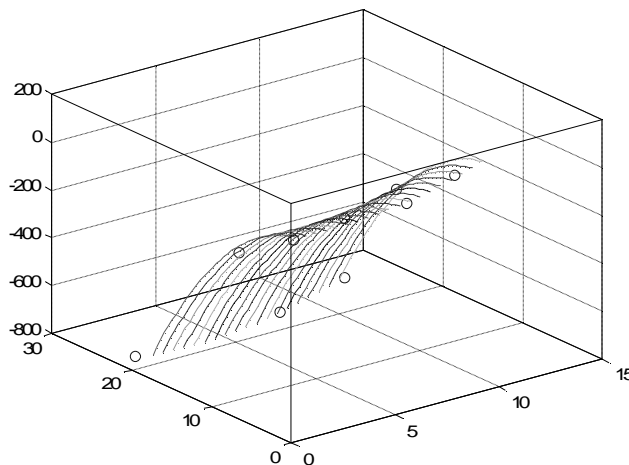


Рис. 13.2. Графік двовимірної регресійної моделі

На рис. 13.2 зображена регресійна поверхня другого порядку, розрахована за 9 випадковими точками. Її параметри виведені на екран і рівняння поверхні регресії має вигляд:

$$Y = 0.5734 \cdot (X_1)^2 - 1.9911 \cdot (X_2)^2 + 2.3351 \cdot X_1 + 2.0694 \cdot X_2 + 3.9873$$

3. В даному прикладі використовуються дані, зібрані для вивчення забруднення води індустріальними і побутовими відходами. Ці дані детально описані в Box, G.P., W.G. Hunter, and J.S. Hunter, *Statistics for Experimenters* (Wiley, 1978, pp. 483-487). Результативна ознака – біохімічний показник кисню у воді в мг/л, а незалежна змінна – інкубаційний період в добах.

```
>> x = [1 2 3 5 7 10]';
>> y = [109 149 149 191 213 224]';
>> plot(x,y,'ko'); grid
>> xlabel('Інкубаційний період (доба), x'); ylabel('Біохімічний показник (мг/л), y');
```

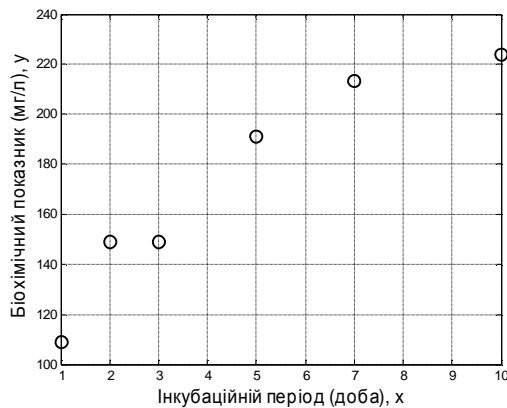


Рис. 13.3. Дані забруднення води

Відомо, що перші два спостереження проводилися з меншою точністю, ніж інші. Вони могли, наприклад, бути проведені з іншим інструментом. Друга загальна причина навантажити дані є та, що кожне записане спостереження – фактично середнє декількох вимірювань. Припускаємо, що два перші значення представляють єдине вимірювання кожне. Останні чотири – середнє з 5 вимірювань. Ці спостереження доцільно навантажити числом вимірювань.

```
>> w = [1 1 5 5 5 5]';
```

Абсолютний масштаб ваги фактично не впливає на наслідки. Тому, вони могли бути нормалізовані деяким чином. Для цілей оцінки

мінливості  $y$ , корисно подумати про вагу 1, як представлення точності "стандартного" вимірювання. У цьому прикладі, де вага представляє число вимірювань, сприяючих спостереженню, природне обчислення для ваги очевидне.

Модель, яку ми оберемо до цих даних, – показникова крива, яка вирівнюється, коли  $x$  стає великим.

```
>> modelFun = @(b,x) b(1).*(1-exp(-b(2).*x));
```

Візуально виявляється, що крива, можливо, вирівнялась би при значенні  $y$  близько 240 поблизу  $x = 15$ . Отже ми використаємо 240, як стартове значення для  $b_1$ , і 0.5? як стартові значення для  $b_2$ , тому що  $e^{(-0.5*15)}$  значно менше одиниці. Отже

```
>> start = [240; .5];
```

Використовуємо нелінійний метод найменших квадратів.

```
>> yw = sqrt(w).*y;  
>> modelFunw = @(b,x) sqrt(w).*modelFun(b,x);  
>> [bFitw,rw,Jw] = nlinfit(x,yw,modelFunw,start);  
>> bFitw  
bFitw =  
 225.1719  
  0.4008
```

Отримали зважені оцінки параметрів нелінійної функції регресії:  
 $b_1 = 225.1719$ ,  $b_2 = 0.4008$ .

Можемо використовувати другий і третій вихідні аргументи **nlinfit**, щоб одержати коваріаційну матрицю оцінених параметрів і від цього отримати оцінки стандартних погрешностей.

```
>> [Qw,Rw] = qr(Jw,0);  
>> msew = sum(abs(rw).^2)/(sum(w)-length(bFitw));  
>> Rinvw = inv(Rw);  
>> Sigmaw = Rinvw*Rinvw*msew;  
>> seFitw = sqrt(diag(Sigmaw))  
seFitw =  
 4.7850  
 0.0288
```

Важлива частина будь-якого аналізу – оцінка точності визначеної моделі. Використовуючи функцію **nlparci** (дивись далі), ми можемо розрахувати довірчі інтервали для параметрів.

```
>> bCIw = nlparsi(bFitw,rw,Jw)
bCIw =
    195.4650  254.8788
     0.2223   0.5793
```

Побудуємо знайдену нелінійну зважену лінію регресії і довірчу смугу на неї.

```
>> xgrid = linspace(min(x),max(x),100);
>> [yFitw, deltaw] = nlpredci(modelFun,xgrid,bFitw,rw,Jw);
>> plot(x,y,'ko', xgrid,yFitw,'b-',xgrid,yFitw+deltaw,'b-',xgrid,yFitw-deltaw,'b:');
>> xlabel('x'); ylabel('y'); grid
>> legend('Дані', 'Зважена лінія регресії', '95% довірчі межі');
```

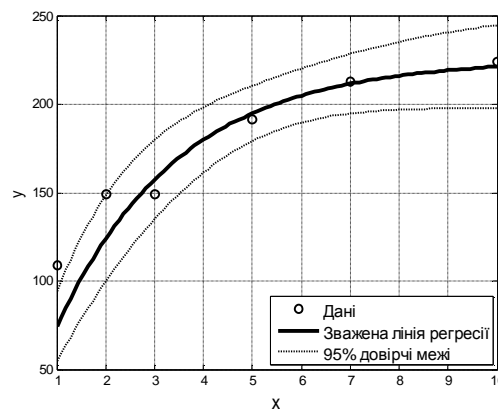


Рис. 13.4. Зважена лінія регресії

Відзначимо, що дві початкові точки на рис. 13.4 виходять за рамки довірчої смуги.

### **NLINTOOL** Інтерактивний нелінійний регресійний аналіз

Графічний інтерфейс користувача для нелінійної регресії за допомогою методу найменших квадратів.

*Синтаксис:*

```
nlintool(x,y,FUN,beta0)
nlintool(x,y,FUN,beta0,alpha,'xname','yname')
```

*Опис:*

Функція **nlintool(x,y,FUN,beta0)** дозволяє одержати інтерактивний графік нелінійної регресійної моделі, обумовленою функцією *FUN*, границь її 95%-го довірчого інтервалу для матриці незалежних змінних *x* і

вектора залежної змінної  $y$ . Кількість рядків матриці  $x$  повинне дорівнювати числу елементів вектора  $y$ . Вхідний аргумент  $beta_0$  є вектором початкових значень коефіцієнтів нелінійної моделі  $FUN$ . Рівняння регресії задається як функція в наступному вигляді:

$$\hat{y} = FUN(beta, X),$$

де  $beta$  – вектор коефіцієнтів функції,  $X$  – матриця незалежних змінних,  $\hat{y}$  – вектор або матриця вихідних значень.

Вектор початкових значень коефіцієнтів регресійної моделі  $beta_0$  може бути заданий у довільній формі. Основна вимога – специфікація вектора  $beta_0$  повинна збігатися зі специфікацією вхідного аргументу  $beta$  функції  $FUN$ .

Функція **nlintool** явно викликає **nlinfit**. Тому формат вхідних параметрів  $x$ ,  $y$ ,  $FUN$ ,  $beta_0$  цих функцій повинен збігатися. Докладне визначення параметрів  $x$ ,  $y$ ,  $FUN$ ,  $beta_0$  див. в опису функції **nlinfit**.

У функції **nlintool(x,y,FUN,beta0,alpha,'xname','yname')** додаткові аргументи *'xname'* і *'yname'* визначають назви незалежних змінних (стовпців матриці  $x$ ) і залежної змінної  $y$ . Аргумент *'yname'* задається як рядок, *'xname'* – як вектор рядків. Назви незалежних і залежної змінних відображаються в графічному вікні (рис. 13.5).

```
>> x=[0:0.1:10]'; % абсциси
>> n=length(x); % число точок
>> y=5*(1-exp(-2*x))+normrnd(0,0.1,n,1); % експериментальні ординати
>> fun=inline('b(1)*(1-exp(-b(2)*x))','b','x');
>> nlintool(x,y,fun,[1 1],'alpha',0.1,'xname','x','yname','y');
```

Результатом роботи функції **nlintool** є інтерактивний графік (рис. 13.5), побудований для одновимірної моделі виду  $y = 5(1 - e^{-2x})$  з довірчою смугою на розрахункові значення (рис. 13.5). Це інтерактивне вікно відображає "вектор" графіків, кожний з яких відповідає стовпцю матриці незалежних змінних. Штрихові лінії на графіках служать для задання поточних значень незалежних змінних і розрахунку за ними залежної змінної. Клацанням лівої кнопки миші можна змінити положення штрихової лінії на відповідному графіку й змінити значення відповідної незалежної змінної.

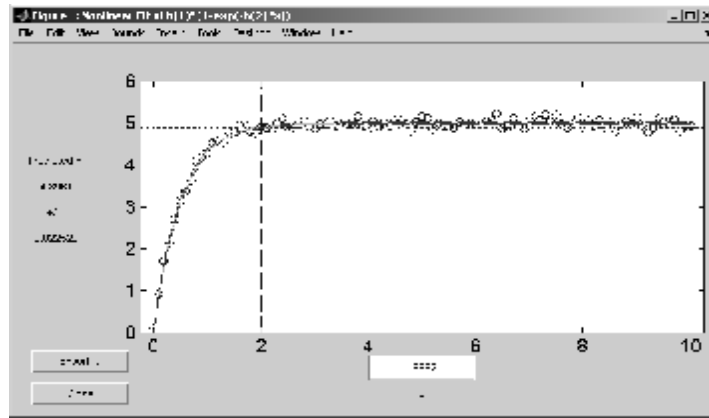


Рис. 13.5. Інтерфейс вікна для інтерактивної нелінійної регресії

Значення змінної відгуку буде перераховано за умови інтерактивної зміни хоча б однієї незалежної змінної в графічному вікні. Величина незалежної змінної може також бути змінена введенням числа в поле введення під графіком відповідної незалежної змінної. Кнопка Export графічного вікна використовується для експорту розрахованих змінних у робочу область **MatLab** (рис. 13.6).



Рис. 13.6. Діалогове вікно експорту результатів інтерактивного графічного аналізу

Кнопкою Export можна експортувати у середовище **MatLab** наступні змінні:

*Parameters* – вектор точкових оцінок коефіцієнтів регресійної моделі;

*Parameters CI* – матриця інтервальних оцінок коефіцієнтів моделі;

*Prediction* – точкова оцінка залежної змінної;

*Prediction CI* – вектор інтервальних оцінок залежної змінної;

*RMSE* – значення кореня квадратного із залишкової дисперсії;

*Residuals* – вектор залишків.

Кнопка Close на рис. 13.5 призначена для закриття графічного вікна.

Меню Bounds (рис. 13.7) дозволяє задати спосіб розрахунку довірчих інтервалів.

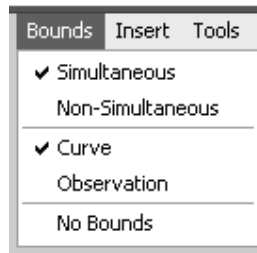


Рис. 13.7. Склад меню Bounds

Пункти *Simultaneous* і *Non-Simultaneous* визначають, яким чином стосовно незалежних змінних будуть розраховуватися границі довірчого інтервалу: *Simultaneous* – у діапазоні зміни незалежних змінних  $x$ ; *Non-Simultaneous* – за вибірковими значенням  $x(j,:)$ . Пункти: *Curve* задає розрахунок значень довірчого інтервалу за регресійною моделлю, *Observation* – за вибірковими значенням  $y$ . Пункт *No Bounds* призначений для скасування розрахунку границь довірчого інтервалу залежної змінної і їх відображення на графіку.

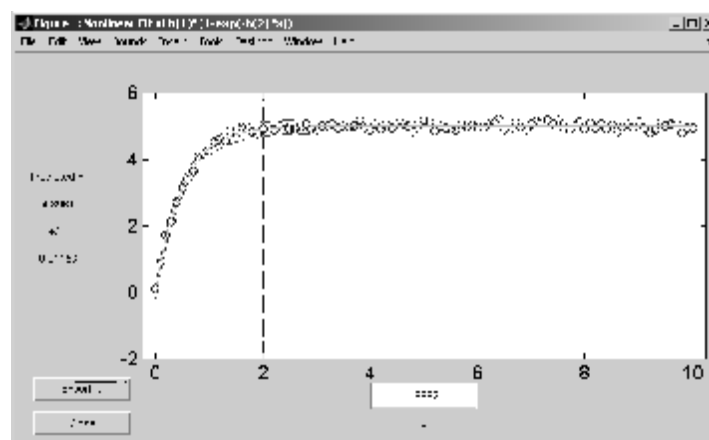


Рис. 13.8. Інтерфейс вікна для інтерактивної нелінійної регресії в режимі Bounds-Observation

На рис. 13.8 показаний результат перерахування моделі в режимі *Bounds–Observation*. За замовчуванням в графічному вікні наведено експериментальні точки, розрахункова лінія регресії і 95%-ві довірчі інтервали на розкидування спостережень навколо неї.

### **NLPARCI** Розрахунок інтервальних оцінок коефіцієнтів нелінійного рівняння регресії

Використовуючи результати, що повертає функція **nlinfit**, знаходяться довірчі інтервали для коефіцієнтів нелінійної регресії

*Синтаксис:*

**ci = nlparci(beta,r,J)**  
**ci = nlparci(beta,resid,J,alpha)**

*Опис:*

Функція **ci = nlparci(beta,r,J)** призначена для розрахунку 95%-х границь довірчого інтервалу *ci* для коефіцієнтів нелінійного рівняння регресії *beta*, вектора залишків *r* і матриці Якобі *J*. Вектор коефіцієнтів *beta* розраховується за методом найменших квадратів. Вхідні аргументи *beta*, *r*, *J* можуть бути розраховані за допомогою функції **nlinfit**.

Межі 95%-х довірчих інтервалів на коефіцієнти регресійної моделі будуть представлені у вигляді матриці *ci*. Розмірність *ci* –  $m \times 2$ , де *m* – число рядків. Кількість рядків *ci* буде дорівнювати числу коефіцієнтів регресійної моделі *beta*. У першому стовпці *ci* приводяться нижні границі довірчих інтервалів, у другому – верхні границі.

Функція **ci = nlparci(beta,resid,J,alpha)** в аргументі *alpha* дозволяє задати рівень значимості для знаходження довірчих інтервалів (за замовчуванням 0.05).

Функція **nlparci** розглядає нечислові значення NaN в *resid* або *J* як відсутні і ігнорує відповідні спостереження.

Довірчі інтервали будуть обчислені правильно, якщо довжина *resid* більше довжини *beta*, а *J* має повний стовпцевий ранг. Якщо *J* погано обумовлена, довірчі інтервали будуть обчислюватися неточно.

### Приклади:

1. Розрахунок параметрів одновимірної моделі виду  $y = 5(1 - e^{-2x})$ :

```
>> x=[0:0.1:10]'; % абсциси
>> n=length(x); % число точок
>> y=5*(1-exp(-2*x))+normrnd(0,0.1,n,1); % експериментальні ординати
>> fun=inline('b(1)*(1-exp(-b(2)*x))','b','x');
>> [b,r,J]=nlinfit(x,y,fun,[1 1]); % обчислюємо параметри
>> disp('Крива насичення:');
>> fprintf('y(x)=%7.4f*(1-exp(%+7.4f*x))\n',b(1),-b(2));
>> ci=nlparsci(b,r,J); % довірчі інтервали
>> disp('Довірчі інтервали для параметрів моделі:');
>> disp('Параметр Нижня границя Верхня границя');
>> fprintf(' b(%d): %10.5f %10.5f\n',[[1;2],ci]);
```

Крива насичення:

$y(x) = 4.9945 \cdot (1 - \exp(-1.9592 \cdot x))$

Довірчі інтервали для параметрів моделі:

Параметр Нижня границя Верхня границя

b(1): 4.97310 5.01466

b(2): 1.97867 2.12822

2. Розрахунок параметрів неповної квадратичної регресійної моделі виду:  $Y = A(X_1)^2 + B(X_2)^2 + CX_1 + DX_2 + E$ , де  $A, B, C, D, E$  -- параметри регресійної моделі;  $Y$  – залежна змінна;  $X_1, X_2$  – незалежні змінні.

2.1. Створюємо m-файл функції (polynom.m), що реалізує неповну квадратичну залежність

```
function yhat=polynom(beta,x)
```

```
% Поділ вектора коефіцієнтів beta за окремими змінними A,B,C,D,E
```

```
A=beta(1);
```

```
B=beta(2);
```

```
C=beta(3);
```

```
D=beta(4);
```

```
E=beta(5);
```

```
% Поділ матриці незалежних змінних x за окремими векторами x1, x2,
```

```
x1=x(:,1);
```

```
x2=x(:,2);
```

```
% Квадратична залежність
```

```
yhat = A.*x1.^2+B.*x2.^2+C.*x1+D.*x2+E;
```

2.2. Моделювання експериментальних даних

```
>> A=1;
```

```
>> B=-2;
```

```
>> C=-1.5;
```

```
>> D=2.6;
```

```
>> E=2;
```

```
>> X1=[1 1 10 10 1 5.5 10 5.5 5.5];
```

```
>> X2=[2 20 20 2 11 20 11 2 11];
```

```
>> Y = A.* X1.^2+B.*X2.^2+C.*X1+D.*X2+E;
>> YY=normrnd(0,2, length(Y),1);
>> Y= Y+ YY';
```

### 2.3. Початкові значення коефіцієнтів

```
>> beta0=[0.1 1 0 1 1];
```

### 2.4. Розрахунок точкових оцінок коефіцієнтів

```
>> [beta,r,J] = nlinfit([X1' X2'],Y',@polynom,beta0)
```

```
beta =
    0.9108   -1.9913   -0.6081    2.3955    1.2243
r =
    1.4494
   -1.6293
    1.5409
   -2.8635
    0.1799
    0.0884
    1.3226
    1.4141
   -1.5025
J =
    1.0000    4.0000    1.0000    2.0000    1.0000
    1.0000   400.0000    1.0000   20.0000    1.0000
   100.0000   400.0000   10.0000   20.0000    1.0000
   100.0000    4.0000   10.0000    2.0000    1.0000
    1.0000   121.0000    1.0000   11.0000    1.0000
   30.2500   400.0000    5.5000   20.0000    1.0000
   100.0000   121.0000   10.0000   11.0000    1.0000
   30.2500    4.0000    5.5000    2.0000    1.0000
   30.2500   121.0000    5.5000   11.0000    1.0000
```

2.5. Розрахунок границь 95%-го довірчого інтервалу на коефіцієнти квадратичної регресійної моделі

```
>> ci = nlparci(beta,r,J)
```

```
ci =
    0.6867    1.1350
   -2.0473   -1.9352
   -3.1412    1.9251
    1.1289    3.6621
   -6.0748    8.5234
```

### 2.6. Графічне представлення отриманої залежності

```
>> [Xx1 Xx2]=meshgrid([1:0.5:10],[2:0.5:20]);
>> z= beta(1).*Xx1.^2+ beta(2).*Xx2.^2+ beta(3).*Xx1+ beta(4).*Xx2+ beta(5);
>> zn= ci(1,1).*Xx1.^2+ ci(2,1).*Xx2.^2+ ci(3,1).*Xx1+ ci(4,1).*Xx2+ ci(5,1);
>> zv= ci(1,2).*Xx1.^2+ ci(2,2).*Xx2.^2+ ci(3,2).*Xx1+ ci(4,2).*Xx2+ ci(5,2);
>> surf(Xx1,Xx2,z)
>> shading flat
>> hold on
>> plot3(X1,X2,Y,'o',Xx1,Xx2,zn)
>> hold on
>> plot3(Xx1,Xx2,zv)
>> grid on, box on
```

На рис. 13.9 представлена поверхня регресії, побудована за 9-ма експериментальними точками (суцільна сітка). За обчисленими значеннями довірчих інтервалів (див. значення *ci*) для коефіцієнтів

неповної квадратичної моделі побудовані довірчі площини (лінійчасті поверхні).

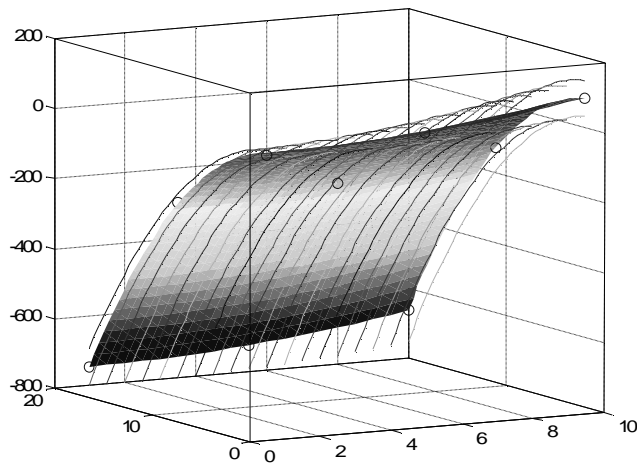


Рис. 13.9. Поверхня регресії двовимірної моделі з довірчими границями

### **NLPREDCI** Розрахунок довірчих інтервалів для нелінійної регресійної моделі

Використовуючи результати, що повертає функція **nlinfit**, знаходяться довірчі інтервали для розрахункових значень функції відгуку в задачі нелінійної регресії.

*Синтаксис:*

```
ypred = nlpredci(FUN,inputs,beta,r,J)  
[ypred,delta] = nlpredci(FUN,inputs,beta,r,J)  
ypred = nlpredci(FUN,inputs,beta,r,J,alpha,'simopt','predopt')
```

*Опис:*

Функція **ypred = nlpredci(FUN,inputs,beta,r,J)** повертає вектор значень залежної змінної *ypred* регресійної моделі *FUN*, що розраховані за матрицею незалежних змінних *inputs*, коефіцієнти регресійної моделі *beta*, вектор залишків *r* і матрицю Якоби *J*. Значення незалежних змінних розташовані в стовпцях матриці *inputs*. Кількість рядків матриці *inputs* і число елементів вектора *y* буде однаковим.

Рівняння регресії задається як *m*-функція в наступному вигляді:

$$\hat{y} = FUN(\beta_0, X),$$

де  $\beta_0$  – вектор коефіцієнтів функції,  $X$  – матриця незалежних змінних,  $\hat{y}$  – вектор або матриця розрахункових значень.

Вектор значень коефіцієнтів регресійної моделі  $\beta$  функції **nlpredci** може бути заданий у довільній формі. Основна вимога – специфікація вектора  $\beta$  повинна збігатися зі специфікацією вхідного аргументу  $\beta_0$  функції  $FUN$ .

Функція **[ypred,delta] = nlpredci(FUN,inputs,beta,r,J)** повертає вектори значень  $y_{pred}$  і половину довірчого інтервалу  $\delta$  залежної змінної. Довірчий інтервал розраховується для 95%-ї довірчої ймовірності. Довірчий інтервал визначається як  $[y_{pred} - \delta, y_{pred} + \delta]$  і розраховується в точках  $inputs(j,:)$  незалежної змінної за отриманим рівнянням регресії.

У функції **ypred = nlpredci(FUN,inputs,beta,r,J,alpha,'simopt','predopt')** додаткові вхідні параметри  $\alpha$ ,  $'simopt'$ ,  $'predopt'$  дозволяють управляти видом довірчого інтервалу, що розраховується. Параметр  $\alpha$  задає рівень значимості довірчого інтервалу. Довірча ймовірність визначається як  $100(1 - \alpha)\%$ . Параметр  $'simopt'$  визначає, яким чином стосовно незалежних змінних, будуть розраховуватися границі довірчого інтервалу:  $'on'$  – у діапазоні зміни незалежних змінних  $inputs$ ;  $'off'$  – у вибіркових значеннях  $inputs(j,:)$ . Значення за замовчуванням  $'simopt'='off'$ . Вхідний аргумент  $'predopt'$  дозволяє задати:  $'curve'$  – розрахунок значень довірчого інтервалу за регресійною моделлю;  $'observation'$  – розрахунок значень довірчого інтервалу за вибірковими значенням. Значення за замовчуванням  $'predopt'='curve'$ .

Вхідні параметри  $\beta$ ,  $r$ ,  $J$  можуть бути розраховані з використанням функції **nlfit**. Формат завдання функції  $FUN$  див. в опису функції **nlfit**.

*Приклади:*

1. Розрахунок параметрів одновимірної моделі  $y = 5(1 - e^{-2x})$ :

```
>> x=[0:0.1:10]; % абсциси
```

```

>> n=length(x); % число точок
>> y=5*(1-exp(-2*x))+normrnd(0,0.1,n,1); % експериментальні ординати
>> fun=inline('b(1)*(1-exp(-b(2)*x))','b','x');
>> [b,r,J]=nlinfit(x,y,fun,[1 1]); % знаходимо параметри
>> disp('Крива насичення:');
>> fprintf('y(x)=%7.4f*(1-exp(%+7.4f*x))\n',b(1),-b(2));
>> xt=linspace(x(1),x(end),1000); % розрахункові точки
>> [yt,dyt]=nlpredci(fun,xt,b,r,J,0.01,[],'observation');
>> plot(x,y,'k.',xt,yt,'k-',xt,yt+dyt,'k--',xt,yt-dyt,'k--'); % графік
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14) % шрифт
>> title('\bфДовірчі інтервали для кривої насичення') % заголовок
>> xlabel('\bfx'), ylabel('\bfy'), grid on, box on
Крива насичення:
y(x)= 4.9987*(1-exp(-1.9911*x))

```

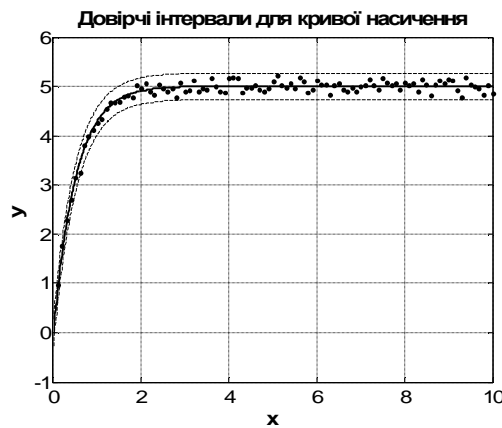


Рис. 13.10. Крива насичення одновимірної моделі

2. Розрахунок значень залежної змінної й границь довірчих інтервалів неповної квадратичної регресійної моделі виду:

$$Y = A(X_1)^2 + B(X_2)^2 + CX_1 + DX_2 + E,$$

де  $A$ ,  $B$ ,  $C$ ,  $D$ ,  $E$  — параметри регресійної моделі;  $Y$  — залежна змінна;  $X_1$ ,  $X_2$  — незалежні змінні.

2.1. Створюємо m-файл функції (polynom.m), що реалізує неповну квадратичну залежність:

```

function yhat=polynom(beta,x)
% Поділ вектора коефіцієнтів beta за окремими змінними A,B,C,D,E
A=beta(1);
B=beta(2);
C=beta(3);
D=beta(4);
E=beta(5);
% Поділ матриці незалежних змінних x за окремими векторами x1, x2,
x1=x(:,1);
x2=x(:,2);

```

```
% Квадратична залежність
yhat = A.*x1.^2+B.*x2.^2+C.*x1+D.*x2+E;
```

## 2.2. Моделювання експериментальних даних

```
>> A=1;
>> B=-2;
>> C=-1.5;
>> D=2.6;
>> E=2;
>> X1=[1 1 10 10 1 5.5 10 5.5 5.5];
>> X2=[2 20 20 2 11 20 11 2 11 ];
>> Y = A.* X1.^2+B.*X2.^2+C.*X1+D.*X2+E;
>> YY=normrnd(0,2, length(Y),1);
>> Y= Y+ YY';
```

## 2.3. Початкові значення коефіцієнтів

```
>> beta0=[0.1 1 0 1 1];
```

## 2.4. Розрахунок точкових оцінок коефіцієнтів

```
>> [beta,r,J] = nlinfit([X1' X2'],Y',@polynom,beta0)
beta =
    0.9850   -1.9877   -1.4003    2.5273    0.7292
r =
    0.7320
    2.1348
    1.3129
   -2.0218
   -2.8668
   -3.4477
    0.7089
    1.2898
    2.1579
J =
    1.0000    4.0000    1.0000    2.0000    1.0000
    1.0000   400.0000    1.0000   20.0000    1.0000
  100.0000   400.0000   10.0000   20.0000    1.0000
  100.0000    4.0000   10.0000    2.0000    1.0000
    1.0000   121.0000    1.0000   11.0000    1.0000
   30.2500   400.0000    5.5000   20.0000    1.0000
  100.0000   121.0000   10.0000   11.0000    1.0000
   30.2500    4.0000    5.5000    2.0000    1.0000
   30.2500   121.0000    5.5000   11.0000    1.0000
```

2.5. Матриця значень незалежних змінних, для яких розраховуються значення залежної змінної

```
>> XX1=min(X1):(max(X1)- min(X1))/20: max(X1);
>> XX2=min(X2):(max(X2)- min(X2))/20: max(X2);
>> inputs=[ XX1' XX2'];
```

2.6. Розрахунок значень залежної змінної  $Y_r$ , її нижньої  $YN$  і верхньої  $YV$  границь 95%-го довірчого інтервалу

```
>> [Yr, delta]= nlpredci(@polynom,inputs,beta,r,J)
Yr =
   -2.5822
   -8.6174
  -17.4738
  -29.1512
   -60.9694
  -81.1101
 -104.0719
 -129.8548
 -158.4589
  -224.1302
  -261.1976
  -301.0860
  -343.7955
  -389.3262
  -488.8508
  -542.8447
  -599.6597
  -659.2959
```

-43.6497	-189.8840	-437.6779	
delta =	5.1802	6.3074	4.7207
6.3643	5.5413	6.1404	4.8689
5.4115	5.8765	5.8765	5.4115
4.8689	6.1404	5.5413	6.3643
4.7207	6.3074	5.1802	
4.8689	6.3643	4.8689	
<b>&gt;&gt; YN=Yr-delta</b>			
YN =	-66.1495	-230.4376	-493.5714
-8.9465	-86.6514	-267.3380	-547.7136
-14.0289	-109.9484	-306.9625	-605.0712
-22.3427	-135.9953	-349.3368	-665.6602
-33.8719	-164.7662	-394.5064	
-48.5187	-196.2483	-442.5469	
<b>&gt;&gt; YV=Yr+delta</b>			
YV =	-55.7892	-217.8229	-484.1301
3.7821	-75.5688	-255.0571	-537.9757
-3.2060	-98.1954	-295.2095	-594.2482
-12.6048	-123.7144	-338.2543	-652.9315
-24.4305	-152.1515	-384.1460	
-38.7808	-183.5197	-432.8090	

## 2.7. Графічне представлення отриманої залежності

```

>> [X_1 X_2]=meshgrid([1:0.5:10],[2:0.5:20]);
>> z= beta(1).*X_1.^2+ beta(2).*X_2.^2+ beta(3).*X_1+ beta(4).*X_2+ beta(5);
>> plot3(XX1,XX2,YN,'ko', XX1,XX2,YV,'+',X_1,X_2,z)
>> grid on, box on
>> xlabel('\bfX_1'), ylabel('\bfX_2'), zlabel('\bfZ')

```

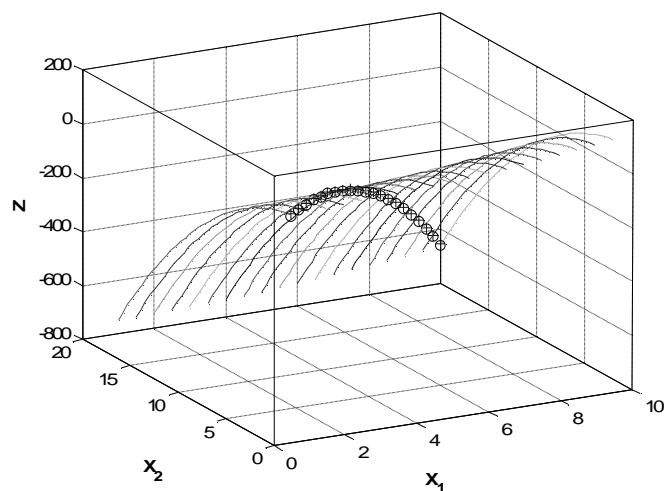


Рис. 13.11. Зображення залежної змінної з довірчими інтервалами

## 14. Функції кластерного аналізу

Будь-яка велика сума знань становиться наукою, якщо ці знання класифіковані. Отже класифікації є початковим етапом наукового дослідження. Правильні класифікації не лише привносять порядок в дослідження, але часто є джерелом формулювання наукових гіпотез, бо “структура відображає процес”. Так, класифікація хімічних елементів на цей час є основою неорганічної хімії і атомної теорії побудови матерії, класифікація організмів є базою еволюційної теорії, класифікація хвороб сприяла відкриттю нових засобів лікування.

Методи кластерного аналізу призначені для створення класифікацій в умовах повної відсутності розробленої теорії або моделі явища. За допомогою кластерного аналізу можна одержати теоретично виправдані результати без будь-якої попередньої інформації. Проте слід пам'ятати, що всі методи кластерного аналізу – евристичні, тобто не мають ніякого теоретичного обґрунтування. Вони базуються на прийнятті низки гіпотез, відносно яких можна привести багато критичних зауважень. Але, на диво, методи кластерного аналізу з успіхом застосовуються у самих різноманітних дослідженнях і є поштовхом для більш детального вивчення проблеми. Теперішнього часу багаторічною практикою вже відсіяно явно невдалі техніки, деякі методики витримали суворі перевірки за аналізу об'єктів з відомими класифікаціями.

У цьому розділі описуються функції кластерного аналізу, що призначені для розбиття множини точок в багатовимірному просторі на підмножини за різними критеріями. Саме поняття "кластер" (cluster – гроно) у цьому випадку можна визначити як сукупність точок вибірки, близьких за деякими характеристиками (критеріями). В **Statistics Toolbox** таким критерієм є метрика простору.

Функції кластерного аналізу будемо вивчати у послідовності:

<b>PDIST</b> Розрахунок парних відстаней між об'єктами вихідної множини даних .....	257
<b>LINKAGE</b> Формування ієрархічного дерева бінарних кластерів .....	264

<b>DENDROGRAM</b>	Графік дендрограми ієрархічного дерева кластерів .....	270
<b>CLUSTER</b>	Кластеризація об'єктів за результативним параметром функції linkage .....	273
<b>CLUSTERDATA</b>	Побудова кластерів за даними .....	277
<b>COPHENET</b>	Коефіцієнт кофенетичної (якісної) кореляції .....	280
<b>INCONSISTENT</b>	Коефіцієнти несумісності .....	281
<b>KMEANS</b>	Кластеризація об'єктів за внутрішньогруповими середніми .....	283
<b>SILHOUETTE</b>	Силуети кластерів .....	289
<b>SQUAREFORM</b>	Перетворення вектора відстаней в матрицю .....	292

Щоб зрозуміти результати застосування функцій кластерного аналізу, слід усвідомити, як працюють разом функції **pdist** і **linkage**.

### **PDIST** Розрахунок парних відстаней між об'єктами вихідної множини даних

Ця функція знаходить відстані між будь-якою парою точок. Далі ця інформація використовується для побудови ієрархічного дерева кластерів.

*Синтаксис:*

**Y = pdist(X)**  
**Y = pdist(X,'metric')**  
**Y = pdist(X,distfun,p1,p2,...)**  
**Y = pdist(X,'minkowski',p)**

*Опис:*

Функція **Y = pdist(X)** дозволяє розрахувати вектор евклідових відстаней  $Y$  між парами об'єктів множини даних, заданих матрицею  $X$ . Розмірність матриці  $X$  дорівнює  $m \times n$ , де  $m$  – число спостережень  $n$ -вимірної випадкової величини. Кількість пар відстаней для  $m$

спостережень, або інакше – для  $m$  об'єктів, буде визначатися формулою

$$C_m^2 = \frac{m(m-1)}{2}.$$

Результативний параметр  $Y$  є вектором, що містить значення парних відстаней між об'єктами. Число елементів  $Y$  дорівнює  $C_m^2$ .

Відстані між об'єктами у векторі  $Y$  розташовані в наступному порядку – спочатку йдуть відстані з 1-м об'єктом, далі з 2-м, 3-м, і т. д., тобто: (1, 2), (1, 3), ... , (1,  $m$ ); (2, 3) ,... , (i, j), ... , (2,  $m$ ); ... , ... , ( $m-1$ ,  $m$ ); де  $i, j$  – номери рядків матриці  $X$ .

За допомогою функції **squareform** вектор  $Y$  можна конвертувати у квадратну матрицю. Елемент ( $i, j$ ) отриманої матриці буде відповідати відстані між  $i$ -м і  $j$ -м об'єктами заданої множини даних.

Функція **Y = pdist(X,'metric')** у вхідному параметрі '*metric*' визначає вид відстані між об'єктами множини даних. Параметр '*metric*' задається як рядкова змінна (табл. 14.1):

Таблиця 14.1

### Види відстаней між об'єктами

Значення параметра ' <i>metric</i> '	Метод розрахунку відстаней між об'єктами
'euclidean'	Евклідова відстань. Значення за замовчуванням
'seuclidean'	Нормалізована евклідова відстань
'mahalanobis'	Відстань Махаланобіса
'cityblock'	“Манхетенівська” відстань. Визначається як сума абсолютних величин відхилень за всіма вимірами
'minkowski'	Метрика Мінковського
'cosine'	Косинусна відстань. Визначається як одиниця мінус косинус кута між об'єктами. Об'єкти в багатомірному просторі розглядаються як вектори
'correlation'	Кореляційна відстань. Визначається як одиниця мінус вибіркового коефіцієнта кореляції між спостереженнями. Вектори спостережень трактуються як вибірки
'hamming'	Відстань Хемінга. Визначається як відсоток відмінних однойменних координат від їх загального числа
'jaccard'	Відстань Жакара. Визначається як одиниця мінус коефіцієнта Жакара. Коефіцієнт Жакара є відсоток відмінних однойменних ненульових координат від їх загального числа

Функція  $Y = \text{pdist}(X, \text{distfun}, p_1, p_2, \dots)$  у параметрі  $\text{distfun}$  показує на функцію розрахунку парних відстаней між об'єктами, що задається користувачем у вигляді:  $d = \text{distfun}(X_i, X_j, p_1, p_2, \dots)$ . Функція  $\text{distfun}$  одержує дві матриці-рядки  $X_i, X_j$ , тобто два рядки матриці  $X$ . Розмірність матриць  $X_i, X_j$  дорівнює  $1 \times n$ . Необов'язкові вхідні параметри  $p_1, p_2, \dots$  прямо передаються функції  $\text{distfun}$  з  $\text{pdist}$ . Функція  $\text{distfun}$  має повертати вектор відстаней  $d$ . Елемент  $d(k)$  є відстанню між спостереженнями  $X_i(k, :)$  і  $X_j(k, :)$ .

Функція  $Y = \text{pdist}(X, \text{'minkowski'}, p)$  призначена для розрахунку парних відстаней між об'єктами, обумовленими матрицею  $X$ , з використанням метрики Мінковського. Вхідний параметр  $p$  є показником ступеня метрики Мінковського. За замовчуванням  $p = 2$ .

Види парних відстаней між об'єктами, що передбачені в  $\text{pdist}$ :

1. Евклідова відстань (її квадрат):

$$d_{ij}^2 = (\mathbf{X}_i - \mathbf{X}_j)(\mathbf{X}_i - \mathbf{X}_j)' = \sum_{k=1}^n (x_{ik} - x_{jk})^2,$$

де  $X_i, X_j$  – рядки матриці  $X$  для об'єктів  $i, j$  ( $i, j = 1 \dots m$ ).

Як було сказано вище, вчені відносно кожної гіпотези (припущенню) кластерного аналізу висловили низку зауважень. Не приводимо повністю цих цікавих роздумів, але звернемо увагу на те, що за винятком відстаней 2 і 3, для всіх інших відстаней дані показників *мають бути попередньо нормовані*. Для евклідової відстані обчислюються суми квадратів значень різних показників (точніше – різниць цих значень для двох об'єктів). Така сума має сенс, якщо всі її складові – однієї розмірності, або безрозмірні.

2. Стандартизована евклідова відстань (її квадрат):

$$d_{ij}^2 = (\mathbf{X}_i - \mathbf{X}_j)\mathbf{D}^{-1}(\mathbf{X}_i - \mathbf{X}_j)' = \sum_{k=1}^n \frac{(x_{ik} - x_{jk})^2}{s_k^2},$$

де  $D$  – діагональна матриця, діагональними елементами якої є вибіркові дисперсії ознак (стовпців матриці  $X$ ).

Тут у якості норми приймаються стандартизовані відхилення (попередня стандартизація показників). Укажемо хоча б на один випадок, коли цей вид нормування є недоцільним. Уявимо, що знаходимо евклідові відстані у скороченому просторі головних компонент. Відомо, що дисперсія першої компоненти – найбільша, ця компонента найбільш важлива, вона пояснює максимум загальної мінливості даних. Навпаки, дисперсії останніх компонент – найменші, ці компоненти майже нічого не пояснюють. Але після стандартизації всі компоненти будуть рівноправними, і тому ця операція буде привносити у розрахунки зайві похибки.

Незважаючи на всі критичні зауваження, ця міра є найбільш поширеною у наукових дослідженнях.

3. Відстань Махаланобіса (його квадрат):

$$d_{ij}^2 = (\mathbf{X}_i - \mathbf{X}_j) \mathbf{V}^{-1} (\mathbf{X}_i - \mathbf{X}_j)' = \sum_{p=1}^n \sum_{q=1}^n w_{pq} (x_{ip} - x_{jp}) (x_{iq} - x_{jq}),$$

де  $V$  – коваріаційна матриця показників, що розрахована за вибіркою  $X$ ;  $w_{pq}$  – елементи оберненої матриці  $W = V^{-1}$ , ( $p, q = 1 \dots n$ ).

Евклідова відстань – це геометрична відстань між об'єктами. Якщо координати точок (значення окремих показників) – взаємоортогональні (у статистичному сенсі – некорельовані), квадрат евклідової відстані є сумою квадратів окремих компонент. Але якщо координати не декартові (не взаємоортогональні, тобто є кореляції між показниками), то формула для розрахунку відстаней ускладнюється і в статистиці називається відстанню Махаланобіса. Деяке теоретичне покращення евклідової метрики з лихвою компенсується новими недоліками відстані Махаланобіса.

4. Манхетенівська відстань (відстань кварталів мегаполісу).

$$d_{ij} = \sum_{k=1}^n |x_{ik} - x_{jk}|.$$

Розумна міра (якщо правильно вибрано нормування). Дійсно, чому для відстані між об'єктами, окремі показники яких до нормування мають неоднакові розмірності, повинне обов'язково приймати геометричну відстань?

5. Метрика Мінковського:

$$d_{ij} = \left[ \sum_{k=1}^n |x_{ik} - x_{jk}|^p \right]^{1/p}.$$

За  $p = 1$  ця метрика еквівалентна Манхетенівській відстані. За  $p = 2$  метрика Мінковського перетворюється в евклідову відстань. За збільшення показника  $p$  метрика Мінковського наближається до відстані Чебишева. Самостійного значення ця метрика не має.

6. Косинусна відстань:

$$d_{ij} = 1 - \frac{\mathbf{X}_i \mathbf{X}_j'}{\sqrt{\mathbf{X}_i \mathbf{X}_i'} \sqrt{\mathbf{X}_j \mathbf{X}_j'}}.$$

Попереднє нормування обов'язкове.

7. Кореляційна відстань:

$$d_{ij} = 1 - \frac{(\mathbf{X}_i - \bar{\mathbf{X}}_i)(\mathbf{X}_j - \bar{\mathbf{X}}_j)'}{\sqrt{(\mathbf{X}_i - \bar{\mathbf{X}}_i)(\mathbf{X}_i - \bar{\mathbf{X}}_i)'} \sqrt{(\mathbf{X}_j - \bar{\mathbf{X}}_j)(\mathbf{X}_j - \bar{\mathbf{X}}_j)'}}$$

де  $\bar{X}_i = \frac{1}{n} \sum_{k=1}^n x_{ik}$ ,  $\bar{X}_j = \frac{1}{n} \sum_{k=1}^n x_{jk}$ , – середні рядків матриці  $X$ . Попереднє нормування обов'язкове.

Ці дві метрики (косинусна і кореляційна) не витримали перевірок на прикладах з відомими класифікаціями.

8. Відстань Хемінга, яка дорівнює частці координат, що відрізняються:  $d_{ij} = \frac{\#(x_{ik} \neq x_{jk})}{n}$ , де функція  $\#( )$  підраховує кількість елементів;

9. Коефіцієнт Жакара, який дорівнює частці ненульових координат, що відрізняються, серед загальної кількості ненульових координат

$$d_{ij} = \frac{\#((x_{ik} \neq x_{jk}) \cap ((x_{ik} \neq 0) \cup (x_{jk} \neq 0)))}{\#((x_{ik} \neq 0) \cup (x_{jk} \neq 0))}.$$

10. Відстань Чебишева, що дорівнює максимальній за модулем різниці координат:  $d_{ij} = \max_k |x_{ik} - x_{jk}|$ . Нормування обов'язкове.

11. Додамо до цього переліку ще метрику користувача, яка чомусь називається “дивергенцією”:

$$d_{ij} = \sum_{k=1}^n \left| \frac{x_{ik} - x_{jk}}{x_{ik} + x_{jk}} \right|$$

В цієї метриці ніяких нормувань непотрібно. На думку деяких фахівців це є найперспективніша відстань.

*Приклади:*

1. Сформуємо нормально розподілений тривимірний масив, що містить сім спостережень і знайдемо  $7 \times 6 / 2 = 21$  відстань між кожною парою спостережень:

```
>> X=normrnd(0,1,7,3)      >> Y = pdist(X)
X =                          Y =
-0.4326 -0.0376 -0.1364      Columns 1 through 7
-1.6656  0.3273  0.1139      1.3100  1.3431  0.7611  1.0460  1.8502  2.7835  2.0344
 0.1253  0.1746  1.0668      Columns 8 through 14
 0.2877 -0.1867  0.0593      2.0205  0.6872  3.1454  3.4098  1.0826  1.8090  2.3074
-1.1465  0.7258 -0.0956      Columns 15 through 21
 1.1909 -0.5883 -0.8323      2.4005  1.7069  1.3312  2.5464  2.7808  2.7805  2.9918
 1.1892  2.1832  0.2944
```

Одержано 21 значення евклідових (за замовчуванням) відстаней між точками вибірки. Конвертуємо вектор  $Y$  у квадратну матрицю:

```
>> S = squareform(Y)
S =
 0 1.3100 1.3431 0.7611 1.0460 1.8502 2.7835
 1.3100 0 2.0344 2.0205 0.6872 3.1454 3.4098
 1.3431 2.0344 0 1.0826 1.8090 2.3074 2.4005
 0.7611 2.0205 1.0826 0 1.7069 1.3312 2.5464
 1.0460 0.6872 1.8090 1.7069 0 2.7808 2.7805
 1.8502 3.1454 2.3074 1.3312 2.7808 0 2.9918
 2.7835 3.4098 2.4005 2.5464 2.7805 2.9918 0
```

2. Розрахунок парних відстаней між об'єктами вихідної множини даних. У якості вихідної множини даних використовується двовимірний випадкова величина. Кількість об'єктів у множині вихідних даних рівна 7.

```
>> X = [3 1.7; 1 1; 2 3; 2 2.5; 1.2 1; 1.1 1.5; 3 1]
X =      || 1.0000 1.0000 | 2.0000 2.5000 || 1.1000 1.5000
```

```

3.0000 1.7000 || 2.0000 3.0000 | 1.2000 1.0000 || 3.0000 1.0000
>> Y = pdist(X)'
Y =
2.1190 0.7000 0.5000 1.8028
1.6401 2.2361 2.1541 0.5099
1.2806 1.8028 1.7493 1.8000
1.9313 0.2000 2.2361 1.9647
1.9105 0.5099 1.7000
2.0000 1.3454

```

Одержано 21 значення відстаней.

3. Розрахунок парних відстаней між об'єктами заданої множини даних. У якості множини даних використовується двовимірна випадкова величина. Кількість об'єктів у множині вихідних даних рівна 7. Порівнюються евклідова відстань, нормалізована евклідова відстань, відстань Махаланобса, Манхетенівська відстань, метрика Мінковського, косинусна відстань, відстань Хемінга, відстань Жакара.

```

>> X = [3 1.7; 1 1; 2 3; 2 2.5; 1.2 1; 1.1 1.5; 3 1];
>> Y1 = pdist(X, 'euclidean');
>> Y2 = pdist(X, 'seuclidean');
>> Y3 = pdist(X, 'mahalanobis');
>> Y4 = pdist(X, 'cityblock');
>> Y5 = pdist(X, 'minkowski');
>> Y6 = pdist(X, 'cosine');
>> Y7 = pdist(X, 'hamming');
>> Y8 = pdist(X, 'jaccard');

>> Y = [Y1'; Y2'; Y3'; Y4'; Y5'; Y6'; Y7'; Y8']
Y =
2.1190 2.4992 2.3879 2.7000 2.1190 0.0362 1.0000 1.0000
1.6401 2.0036 2.1983 2.3000 1.6401 0.1072 1.0000 1.0000
1.2806 1.5399 1.6946 1.8000 1.2806 0.0715 1.0000 1.0000
1.9313 2.2815 2.1684 2.5000 1.9313 0.0160 1.0000 1.0000
1.9105 2.2378 2.2284 2.1000 1.9105 0.0879 1.0000 1.0000
0.7000 0.8757 0.8895 0.7000 0.7000 0.0187 0.5000 0.5000
2.2361 2.7621 2.6097 3.0000 2.2361 0.0194 1.0000 1.0000
1.8028 2.2115 2.0616 2.5000 1.8028 0.0061 1.0000 1.0000
0.2000 0.2341 0.2378 0.2000 0.2000 0.0041 0.5000 0.5000
0.5099 0.6363 0.6255 0.6000 0.5099 0.0116 1.0000 1.0000
2.0000 2.3408 2.3778 2.0000 2.0000 0.1056 0.5000 0.5000
0.5000 0.6255 0.6353 0.5000 0.5000 0.0038 0.5000 0.5000
2.1541 2.6713 2.5522 2.8000 2.1541 0.0412 1.0000 1.0000
1.7493 2.1518 2.0153 2.4000 1.7493 0.0010 1.0000 1.0000
2.2361 2.7621 2.9890 3.0000 2.2361 0.2106 1.0000 1.0000
1.7000 2.0970 1.9750 2.3000 1.7000 0.0202 1.0000 1.0000
1.3454 1.6354 1.5106 1.9000 1.3454 0.0009 1.0000 1.0000
1.8028 2.2115 2.4172 2.5000 1.8028 0.1604 1.0000 1.0000
0.5099 0.6363 0.6666 0.6000 0.5099 0.0295 1.0000 1.0000
1.8000 2.1067 2.1400 1.8000 1.8000 0.0688 0.5000 0.5000
1.9647 2.3101 2.4517 2.4000 1.9647 0.1840 1.0000 1.0000

```

## **LINKAGE** Формування ієрархічного дерева бінарних кластерів

Будується ієрархічне дерево кластерів, використовуючи результати функції **pdist**. Одержаний результат, у свою чергу, використовується у функції **cluster** для розбивки множини точок на кластери.

*Синтаксис:*

```
Z = linkage(Y)  
Z = linkage(Y,'method')
```

*Опис:*

Функція **Z = linkage(Y)** дозволяє сформувати ієрархічне дерево бінарних кластерів з використанням алгоритму "найближчого сусіда". Вхідний аргумент  $Y$  є вектором відстаней між парами об'єктів множини даних у багатовимірному просторі. Число елементів вектора  $Y$  дорівнює

$C_m^2 = \frac{m(m-1)}{2}$ , де  $m$  – кількість об'єктів у множині даних. Вектор  $Y$  може

бути отриманий як результат роботи функції **pdist**. У загальному випадку вхідний аргумент  $Y$  може бути заданий як матриця відстаней між парами об'єктів вихідної множини даних, згідно з форматом параметра функції **pdist**.

Результативний параметр  $Z$  є матрицею, що містить інформацію про дерево кластерів. Розмірність  $Z$  дорівнює  $(m - 1) \times 3$ . Кінцеві вузли дерева кластерів є об'єктами заданої множини даних – спостережень багатовимірної випадкової величини  $Y$ , пронумерованих від 1 до  $m$ . Кінцеві вузли є одиничними кластерами. Вони поєднуються в кластери вищерозташованими вузлами дерева. Кожному наступному вищерозташованому вузлу дерева кластерів відповідає  $i$ -й рядок матриці  $Z$ . Йому відповідає індекс  $m + i$ .

Стовпці 1 і 2 матриці  $Z$  містять індекси об'єктів, зв'язаних у новий кластер. Кількість сформованих бінарних кластерів буде дорівнювати  $(m - 1)$ . 3-й стовпець матриці  $Z$  містить значення відстаней між парами об'єктів, об'єднаних у кластери.

Припустимо, що дерево кластерів містить 30 початкових вузлів. Якщо 100-й кластер був сформований об'єднанням 5-го й 7-го об'єктів і

відстань між ними дорівнює 1.5, тоді 100-й рядок матриці  $Z$  буде містити наступні значення  $Z(:,100)=[5 \ 7 \ 1.5]$ . Цей кластер буде мати індекс рівний  $10 + 30 = 40$ . Якщо індекс 40 буде виявлений у наступних рядках  $Z$ , то це означає, що 40-й бінарний кластер був об'єднаний у новий вищерозташований кластер.

Функція  $Z = \text{linkage}(Y, 'method')$  у додатковому аргументі *'method'* дозволяє задати алгоритм кластеризації. Значення аргументу *'method'* задається як тестовий рядок. Передбачені наступні алгоритми кластеризації (табл. 14.2):

Таблиця 14.2

### Алгоритми кластеризації

Значення <i>'method'</i>	Назва алгоритму
<i>'single'</i>	Алгоритм "найближчого сусіда". Значення за замовчуванням.
<i>'complete'</i>	Алгоритм "найдалшого сусіда"
<i>'average'</i>	Алгоритм "середнього зв'язку"
<i>'centroid'</i>	Центроїдний алгоритм, що використовує відстань між центрами ваги кластерів
<i>'ward'</i>	Покроковий алгоритм Уорда

Формування дерева кластерів засновано на об'єднанні двох нижчерозташованих вузлів в один вищерозташований і т.д. Критерієм об'єднання вузлів у кластер є відношення відстаней між парами об'єктів або кластерів. Для цього і подальшого треба прийняти якусь міру відстаней між кластерами різних розмірів. В **linkage** передбачені наступні міри відстаней між кластерами (табл. 14.3):

Таблиця 14.3

### Міри відстаней між кластерами

Вид алгоритму	Виразення для розрахунку відстані між кластерами
Алгоритм "найближчого сусіда"	$d(r, s) = \min\{dist(x_{ri}, x_{sj})\}, \quad i \in (1, \mathbf{K}, n_r), j \in (1, \mathbf{K}, n_s),$ <p>де <math>n_r</math> – кількість об'єктів у кластері <math>r</math>; <math>n_s</math> – число об'єктів у кластері <math>s</math>;  <math>x_{ri}</math> – <math>i</math>-й об'єкт у кластері <math>r</math>; <math>x_{sj}</math> – <math>j</math>-й об'єкт у кластері <math>s</math>.  Алгоритм "найближчого сусіда" заснований на визначенні найменшої відстані між об'єктами у двох групах</p>

Вид алгоритму	Вирази для розрахунку відстані між кластерами
Алгоритм "найдалшого сусіда"	$d(r, s) = \max\{dist(x_{ri}, x_{sj})\}, \quad i \in (1, \mathbf{K}, n_r), j \in (1, \mathbf{K}, n_s).$ <p>Алгоритм "найдалшого сусіда" заснований на визначенні найбільшої відстані між об'єктами у двох групах</p>
Алгоритм "середнього зв'язку"	$d(r, s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dist(x_{ri}, x_{sj}).$ <p>Алгоритм "середнього зв'язку" заснований на розрахунку відстаней між усіма можливими парами об'єктів у кластерах <math>r</math> і <math>s</math></p>
Центроїдний алгоритм	$d(r, s) = d(\bar{x}_r, \bar{x}_s), \quad \text{де } \bar{x}_r = \frac{1}{n_r} \sum_{i=1}^{n_r} x_{ri}, \quad \bar{x}_s = \frac{1}{n_s} \sum_{j=1}^{n_s} x_{sj} -$ <p>центроїди кластерів <math>r</math> і <math>s</math>. У результаті застосування центроїдного алгоритму може бути отримане немонотонне дерево кластерів. Це може відбутися у випадку, коли відстань від об'єднання двох кластерів <math>r</math> і <math>s</math> до третього кластера <math>g</math> менше, чим відстань від <math>r</math> або <math>s</math> до <math>g</math>. У цьому випадку дендрограма може змінити свій напрямок, що є підставою для використання іншого алгоритму кластеризації</p>
Покроковий алгоритм Уорда	$d(r, s) = \frac{n_r n_s}{(n_r + n_s)} d_{rs}^2, \quad \text{де } d_{rs}^2 - \text{квадрат відстані між кластерами } r \text{ і } s,$ <p>визначеної за центроїдним алгоритмом. Покроковий алгоритм заснований на мінімізації збільшення загальної внутрішньогрупової суми квадратів у результаті об'єднання груп <math>r</math> і <math>s</math>. Внутрішньогрупова сума квадратів кластера визначається як сума квадратів відстаней між усіма об'єктами в кластері й центроїдом кластера</p>

### Приклади:

1. Формування ієрархічного дерева бінарних кластерів для 10-ти вимірної нормально розподіленої випадкової величини. Кількість об'єктів у множині вихідних даних рівна 20. За замовченням використовується алгоритм "найближчого сусіда":

```
>> X=normrnd(0,1,20,10);
>> Y = pdist(X);
>> Z = linkage(Y)
Z =
  5.0000 20.0000 1.9330 16.0000 23.0000 2.6409 13.0000 36.0000 2.9005
  7.0000 19.0000 2.0429  4.0000 29.0000 2.6727 17.0000 37.0000 2.9329
 12.0000 15.0000 2.0979 14.0000 30.0000 2.7020  2.0000 38.0000 3.4541
  6.0000 10.0000 2.3100 25.0000 28.0000 2.7839
```

Графічне представлення дерева бінарних кластерів виконується за допомогою функції **dendrogram** (рис. 14.1).

```
>> H=dendrogram(Z);
>> box on
```

```
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',16)
>> title('\bfАлгоритм найближчого сусіда');
```

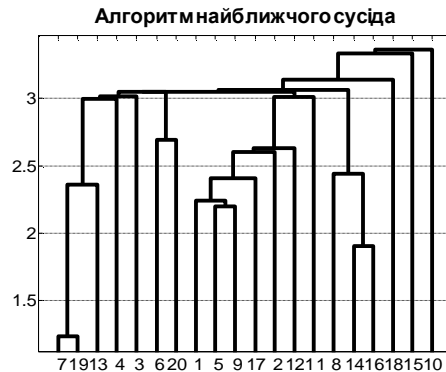


Рис. 14.1. Кластеризація за допомогою алгоритму "найближчого сусіда" (за замовчуванням)

Тривалий час вважалося, що об'єднання кластерів за принципом "найближчого сусіда" має певні переваги, нібито цей метод спроможний виділяти протяженні кластери, навіть кільцевої форми. Практика не виправдувала цих сподівань. Виявилось, що за використання принципу "найближчого сусіда", починаючи з деякого кроку, до великого кластера по черзі підключаються (притягуються) всі інші об'єкти. В результаті найчастіше буде одержана неправильна структура розшарування об'єктів.

2. Кластеризація за допомогою алгоритму "найдалшого сусіда" (рис. 14.2):

```
>> Z = linkage(Y, 'complete');
>> H=dendrogram(Z);
>> box on
```

```
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',16)
>> title('\bfАлгоритм найдалшого сусіда');
```

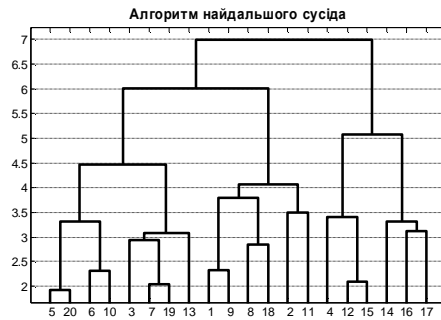


Рис. 14.2. Кластеризація за допомогою алгоритму "найдальшого сусіда"

Цей метод був розроблений як альтернативний попередньому, але він має інші недоліки – за цим принципом завжди знаходять чітке розшарування об'єктів, навіть у випадках, коли ніякого розшарування немає (всі об'єкти належать до однієї однорідної сукупності). Цей недолік (привнесення структури у дані) у той чи іншій мірі присутній для будь-якого ієрархічного алгоритму.

3. Кластеризація за допомогою алгоритму "середнього зв'язку" (рис. 14.3):

```
>> Z = linkage(Y, 'average');
>> H=dendrogram(Z);
>> box on
```

```
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',16)
>> title('\bfАлгоритм середнього зв'язку');
```

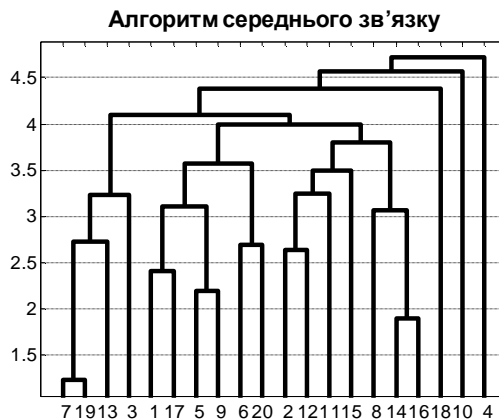


Рис. 14.3. Кластеризація за допомогою алгоритму "середнього зв'язку"

#### 4. Кластеризація за допомогою центроїдного алгоритму

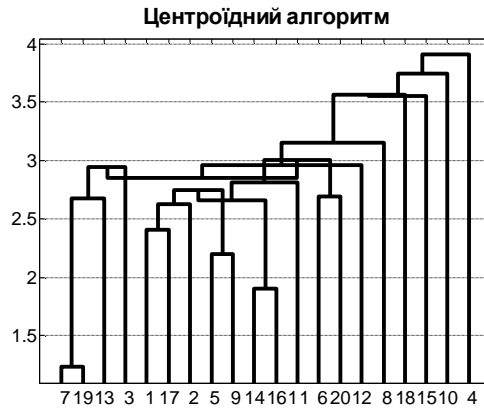


Рис. 14.4. Кластеризація за допомогою центроїдного алгоритму

Оператори:

```
>> Z = linkage(Y, 'centroid');
>> H=dendrogram(Z);
>> box on
```

```
Warning: Non-monotonic cluster tree -- the centroid linkage is probably not appropriate
In linkage at 153
```

```
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',16)
>> title('\bfЦентроїдний алгоритм');
```

Було видано повідомлення: “Увага: Дерево кластерів немонотонне – імовірно, що центроїдний алгоритм не підходить”.

#### 5. Кластеризація за допомогою алгоритму Уорда (рис. 14.5):

```
>> Z = linkage(Y, 'ward');
>> H=dendrogram(Z);
>> box on
```

```
>> set(get(gcf,'Currentaxes'),...
'Fontname','Arial Cyr','FontSize',16)
>> title('\bfАлгоритм Уорда');
```

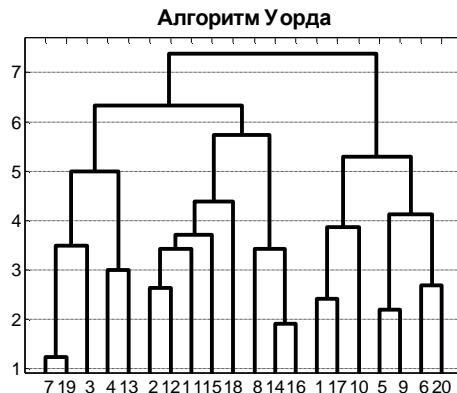


Рис. 14.5. Кластеризація за допомогою алгоритму Уорда

Зараз алгоритм Уорда вважається самим надійним з ієрархічних алгоритмів.

## **DENDROGRAM** Графік дендрограми ієрархічного дерева кластерів

Будує дендрограму ієрархічного дерева кластерів. Ця функція була використана у попередніх прикладах (рис. 14.1 – 14.5)

*Синтаксис:*

```
dendrogram(Z)  
dendrogram(Z,p)  
h = dendrogram(...)  
[h,t] = dendrogram(...)  
[h,t,perm] = dendrogram(...)  
[...] = dendrogram(...,'colorthreshold',tc)  
[...] = dendrogram(...,'orientation',orient)  
[...] = dendrogram(...,'labels',S)
```

*Опис:*

Функція **dendrogram(Z)** будує дендрограму ієрархічного дерева кластерів  $Z$ . Аргумент  $Z$  – матриця розміром  $(m - 1) \times 3$ , що створена за допомогою функції **linkage**,  $m$  – кількість об'єктів у даних. Дендрограма складається з великої кількості П-подібних ліній, що з'єднують об'єкти в ієрархічному дереві. Висота кожної такої фігури відповідає відстані між двома об'єктами, що з'єднуються.

Функція **dendrogram(Z,p)** будує дендрограму лише з  $p$  кластерами (за замовчуванням 30). За умови більшого значення  $p$  рисунок може бути занадто переповненим лініями. Щоб показати всі вузли, потрібно задати  $p = 0$ .

Функція **h = dendrogram(...)** повертає вектор дескрипторів (покажчиків) створених ліній.

Функція **[h,t] = dendrogram(...)** у результативному параметрі  $t$  повертає вектор довжини  $m$ , що містить номер вузла дерева для кожного об'єкта з заданого набору даних. Це може бути корисним, якщо  $p$  менше, ніж загальне число об'єктів. Для таких даних вузли дерева можуть відповідати декільком об'єктам. У цьому випадку, щоб з'ясувати, які

об'єкти втримуються у вузлі дендрограми з номером  $k$ , можна використовувати команду **find(t==k)**. Якщо у даних менше, ніж  $p$  об'єктів, то всі об'єкти зображуються на дендрограмі. У цьому випадку повертається  $t=(1:m)'$ , тобто кожний вузол відповідає одному об'єкту.

Функція **[h,t,perm] = dendrogram(...)** у результативному параметрі **perm** повертає вектор перестановок міток вузлів і листів дендрограми. Мітки в **perm** нумеруються ліворуч-праворуч у горизонтальних дендрограмах і знизу-нагору у вертикальних.

Функція **[...] = dendrogram(...,'colorthreshold',tc)** призначає індивідуальний колір кожній групі вузлів дендрограми, які зв'язані менше, ніж на граничне значення  $tc$ . Величина  $tc$  повинна перебувати в межах від 0 до  $\max(Z(:,3))$ . Завдання  $tc$  у вигляді рядка *'default'* дає значення  $tc = 0.7 * \max(Z(:,3))$ . Задати значення 0 – це однаково, що не задавати взагалі цей параметр. У цьому випадку всі лінії будуть накреслені одним кольором. Максимально можливе значення  $\max(Z(:,3))$  також приводить до креслення всіх ліній одним кольором.

Функція **[...] = dendrogram(...,'orientation',orient)** орієнтує дендрограму у вікні фігури. Можливі значення аргументу *orient* відповідають таким зображенням дендрограми:

*'top'* – зверху-вниз (за замовчуванням);

*'bottom'* – знизу-нагору;

*'left'* – ліворуч-праворуч;

*'right'* – праворуч-ліворуч.

Функція **[...] = dendrogram(...,'labels',S)** дозволяє задати в аргументі  $S$  мітки для кожного спостереження (точки). Цей аргумент повинен бути масивом символів або масивом символічних рядків. Кожний вузол дерева, що містить одне спостереження, буде відзначено на дендрограмі відповідною міткою.

*Приклад:*

Створимо 100 точок, випадковим чином розподілених у квадраті розміром  $1 \times 1$  за рівномірним законом. Представимо графічно їх у вигляді фактичних точок і гістограми розподілу їх абсцис (рис. 14.6):

```

>> n=100; % кількість точок
>> x=lhsdesign(n,2); % рівномірно розподілені у квадраті точки
>> subplot(1,2,1); plot(x(:,1), x(:,2), 'o'); grid, box % фактичні точки
>> subplot(1,2,2); hist(x(:,1)); grid, box % Гістограма розподілу

```

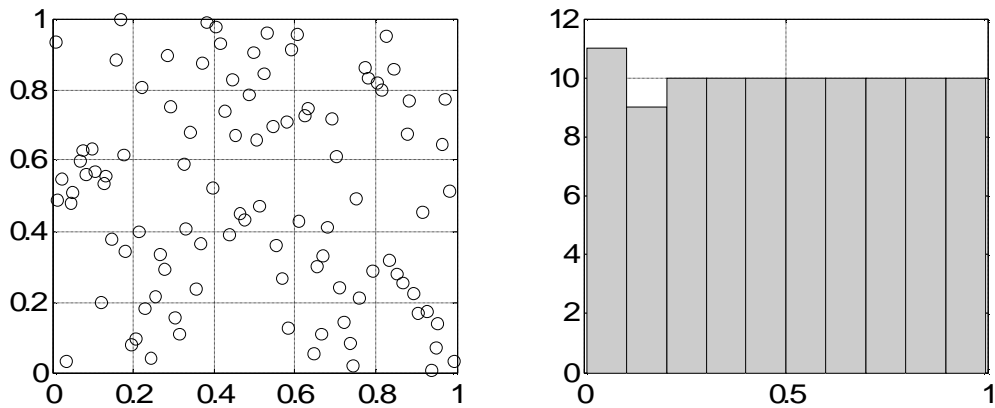


Рис. 14.6. Рівномірно розподілені у квадраті точки і їх гістограма

На рис. 14.6 зображено поле розкиду точок і графік гістограми. Візуально бачимо, що розподіл дійсно рівномірний.

Будуємо дендрограму з мітками для кожної точки (рис. 147):

```

>> S=cell(1,n); % масив символьних рядків
>> for k=1:n,
    S{k}=[s' num2str(k)]; % мітка кожної точки
end
>> Y=pdist(x); % відстані між точками
>> Z=linkage(Y); % ієрархічне дерево кластерів
>> [h,t]=dendrogram(Z,'labels',S); box on
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfДерево кластерів');
>> xlabel('\bfГрупи') % мітка осі ОХ

```

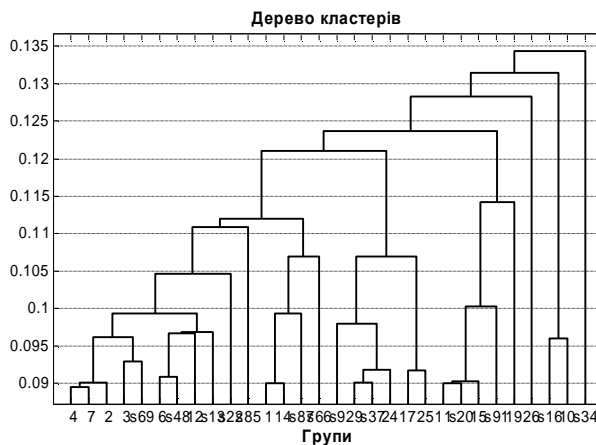


Рис. 14.7. Дерево кластерів

На рис. 14.7 показане ієрархічне дерево кластерів, побудоване за допомогою функції **dendrogram**. Вузли, що містять по одному об'єкту, відзначені міткою  $s$  з номером точки. Якщо ж вузол включає кілька точок, то біля нього показано просто номер, без букви  $s$ .

### **CLUSTER** Кластеризація об'єктів за результативним параметром функції **linkage**

Будуються кластери з ієрархічного дерева, отриманого в результаті роботи функції **linkage** (див. вище). Ця функція, у свою чергу, використовує результати функції **pdist** (див. вище).

*Синтаксис:*

```
T = cluster(Z,'cutoff',c)
T = cluster(Z,'maxclust',n)
T = cluster(...,'criterion','crit')
T = cluster(...,'depth',d)
```

*Опис:*

Функція **T = cluster(Z,'cutoff',c)** призначена для поділу на окремі кластери  $T$  ієрархічного дерева кластерів  $Z$ , отриманого за допомогою функції **linkage**. Вхідний параметр  $Z$  є матрицею з розмірністю  $(m - 1) \times 3$ , де  $m$  – кількість спостережень багатовимірної випадкової величини множини даних. Вхідний параметр  $c$  – гранична величина, що призначена для ділення ієрархічного дерева  $Z$  на кластери. Кластер формується шляхом виключення зв'язків ієрархічного дерева кластерів, для яких значення коефіцієнтів несумісності менше величини  $c$  (докладніше див. опис функції **inconsistent**). Результативний параметр  $T$  є вектором номерів кластерів, до яких віднесені об'єкти вихідної множини даних. Число елементів вектора  $T$  дорівнює  $m$ .

У функції **T = cluster(Z,'maxclust',n)** вхідний параметр  $n$  визначає максимальну кількість сформованих кластерів, на яку розділяється ієрархічне дерево кластерів  $Z$ .

Функція **T = cluster(...,'criterion','crit')** у вхідному параметрі *'crit'* визначає критерій формування кластерів. Можливі значення аргументу *crit*:

*'inconsistent'* – несумісність (використовується за замовчуванням);

*'distance'* – відстань.

Функція **T = cluster(...,'depth',d)** дозволяє задати глибину перегляду ієрархічного дерева кластерів для обчислення рівня несумісності. В ході обчислення коефіцієнта несумісності порівнюється зв'язок між двома об'єктами в дереві кластерів до зазначеної глибини. Вхідний параметр *d* задає число вищерозташованих рівнів ієрархії в дереві кластерів в процесі розрахунку коефіцієнтів несумісності. Значення за замовчуванням  $d = 2$ .

*Приклади:*

1. Кластеризація об'єктів за результативним параметром функції **linkage**. У якості даних використовується п'ятивимірний випадковий величина *X*. Розподіл *X* є композицією нормального закону й закону Вейбулла. Порівнюється розподіл об'єктів по кластерах для граничної величини *t* рівної  $0.1 * (\max(Z(:,3)))$ ,  $0.3 * (\max(Z(:,3)))$ ,  $0.4 * (\max(Z(:,3)))$ . Графічне представлення ієрархічного дерева кластерів виконується за допомогою функції **dendrogram**.

```
>> X=normrnd(0,1,15,5)+weibrnd(1,2,15,5);  
>> Y=pdist(X);  
>> Z=linkage(Y, 'ward');  
>> H=dendrogram(Z); box on
```

На рис. 14.8 показано повне дерево кластерів.

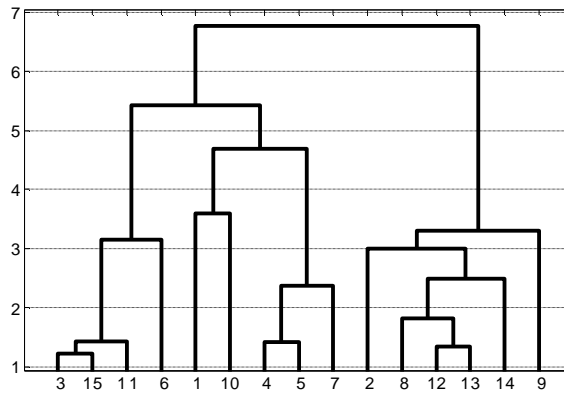


Рис. 14.8. Дерево кластерів

Далі розділяємо це дерево на три кластери.

```
>> t1=0.1*(max(Z(:,3)));
>> t2=0.3*(max(Z(:,3)));
>> t3=0.4*(max(Z(:,3)));
```

>> T1 = cluster(Z,'cutoff', t1);	>> T1 = cluster(Z,'cutoff', t1);
>> T2 = cluster(Z,'cutoff', t2);	>> T2 = cluster(Z,'cutoff', t2);
>> T3 = cluster(Z,'cutoff', t3);	>> T3 = cluster(Z,'cutoff', t3);

```
>> cat(2,T1,T2,T3)
```

ans =	2	1	1	8	1	1
6	1	1	7	7	1	1
6	1	1	5	9	1	1
1	1	1	4	5	1	1
9	1	1	3	8	1	1
			4			

Функція **cat(dim,A,B, ...)** зчіплює матриці одного розміру уздовж розмірності *dim*. Для *dim* = 1 – зчеплювання по рядках, *dim* = 2 – по стовпцях.

Зображемо кластери на площині xOy.

```
>> k=3; % кількість кластерів
>> s='vo+.'; % види точок
>> T=cluster(Z,'maxclust',k); % розбили на 3 кластери
>> figure
>> hold on
>> for kk=1:k,
    nk=find(T==kk); % номери точок, що потрапляють у кластер
    plot(X(nk,1),X(nk,2),['k' s(kk)])
end
>> grid, box, hold off
>> legend('cluster1', 'cluster2', 'cluster3')
```

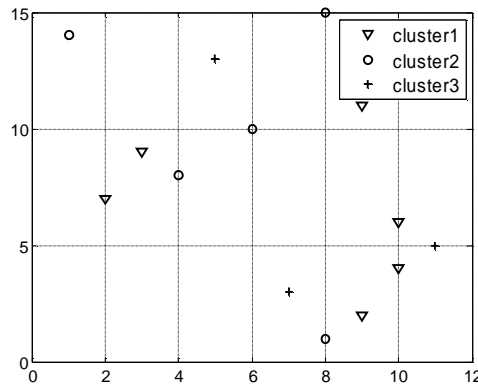


Рис. 14.9. Розбивка масиву на три кластери

На рис. 14.9 показано три кластери, на які було розбите дерево рис. 14.8. Номери кластерів можна вивести на екран:

```
>> T=cluster(Z,'maxclust',3); % розбили на 3 кластери
>> nk1=find(T==1)
nk1 =
    2
    4
    6
    7
    9
   11
>> nk2=find(T==2)
nk2 =
    1
    8
   10
   14
   15
>> nk3=find(T==3)
nk3 =
    3
    5
   12
   13
```

2. Кластеризація об'єктів за результативним параметром функції **linkage**. У якості вихідних даних використовується 10-ти вимірна випадкова величина  $X$ . Об'єм вибірки дорівнює 20 елементам. Вибірka  $X$  розподілена за нормальним законом. Графічне представлення ієрархічного дерева кластерів виконується за допомогою функції **dendrogram**. Критерій кластеризації: '*crit*' = '*inconsistent*'. Максимальне число кластерів прийняте рівним 6. Число вищерозташованих рівнів ієрархії в дереві кластерів за умови розрахунку коефіцієнтів несумісності прийняте рівним 3.

```
>> X=normrnd(0,1,20,10);
>> Y = pdist(X);
>> Z = linkage(Y,'ward');
>> H=dendrogram(Z); box
>> T = cluster(Z,'maxclust', 6, 'criterion', 'inconsistent', 'depth', 3);
>> No=1:1:lenght(T);
>> [No',T]
ans =
```

1	4	5	3	9	6	13	1	17	2
2	5	6	2	10	6	14	6	18	4
3	2	7	6	11	3	15	5	19	4
4	4	8	2	12	5	16	5	20	6

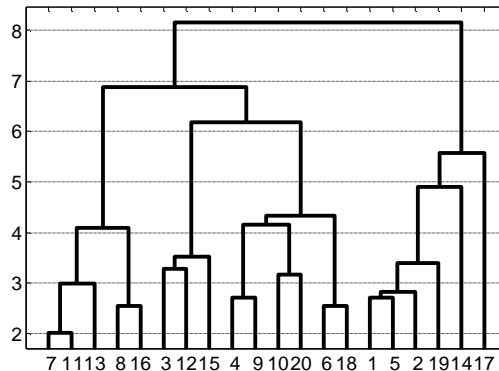


Рис. 14.10. Дерево кластерів

Можна переглянути склад кожного із шести кластерів:

<code>&gt;&gt; find(T==1)</code>	<code>&gt;&gt; find(T==2)</code>	<code>&gt;&gt; find(T==3)</code>	<code>&gt;&gt; find(T==4)</code>	<code>&gt;&gt; find(T==5)</code>	<code>&gt;&gt; find(T==6)</code>
ans = 13	ans = 3 6 8 17	ans = 5 11	ans = 1 4 18 19	ans = 2 12 15 16	ans = 7 9 10 14 20

### **CLUSTERDATA** Побудова кластерів за даними

За заданими координатами точок розбиваємо їх на кластери еквівалентно послідовному застосуванню функцій **pdist**, **linkage** і **cluster**.

*Синтаксис:*

**T = clusterdata(X,cutoff)**

**T = clusterdata(X,param1,val1,param2,val2,...)**

*Опис:*

Функція **T = clusterdata(X,cutoff)** будує кластери за даними *X*. Аргумент *X* повинен бути матрицею розміру  $m \times n$ , де *m* – кількість спостережень, а *n* – кількість змінних. Аргумент *cutoff* – поріг несумісності для об'єднання точок у кластери з їх ієрархічного дерева.

Якщо  $0 < cutoff < 2$  функція **clusterdata** поєднує точки в кластери, якщо рівень їх несумісності більше, ніж *cutoff*. Якщо *cutoff*  $\geq 2$  і ціле, функція **clusterdata** сприймає його як максимальну кількість кластерів. Вихідний параметр *T* – вектор довжини *m*, що містить номери кластерів для кожної точки даних *X*.

Функція **T = clusterdata(X,param1,val1,param2,val2,...)** визначає додаткові аргументи у вигляді пар "ім'я – значення". Можливі імена (рядки) і відповідні їм значення:

*'distance'* – задає метрику для обчислення відстані у функції **pdist**. Можливі значення перелічені в опису функції **pdist**. Якщо задана метрика *'minkowski'*, після неї потрібно вказати ще один параметр *p* – показник ступеня метрики;

*'linkage'* – метод побудови ієрархічного дерева кластерів для функції **linkage**. Можливі значення перелічені в опису функції **linkage**;

*'cutoff'* – поріг несумісності для об'єднання точок у кластери;

*'maxclust'* – максимальна кількість кластерів;

*'criterion'* – критерій для об'єднання точок у кластери. Можливі значення: *'inconsistent'* (несумісність, використовується за замовчуванням) або *'distance'* (відстань);

*'depth'* – глибина перегляду ієрархічного дерева кластерів для обчислення рівня несумісності (за замовчуванням 2).

*Приклад:*

```
>> n=100; % кількість точок
>> k=4; % кількість кластерів
>> s='vo^s'; % види точок
>> X=lhsdesign(n,2); % рівномірно розподілені у квадраті точки
>> T=clusterdata(X,'maxclust',k,'linkage','ward'); % розклад на k=4 кластери
>> figure
>> hold on
>> for kk=1:k,
    nk=find(T==kk); % номери точок, що потрапляють у кластер
    plot(X(nk,1),X(nk,2),['k' s(kk)])
end
>> hold off, grid, box
>> axis equal % однаковий масштаб
```

```

>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfРозклад множини точок на кластери');
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfy') % мітка осі OY

```

На рис. 14.11 показані кластери, на які розбито 100 точок за допомогою функції **clusterdata**. Як видно, отримані кластери розрізняються між собою за кількістю точок. Крім того, деякі кластери можуть перебувати усередині інших. Такий же результат дає й послідовне застосування функцій **pdist**, **linkage** і **cluster**.



Рис. 14.11. Розклад 100 точок на чотири кластери

Побудуємо дендрограму для цього прикладу.

```

>> Y = pdist(X);
>> Z = linkage(Y,'ward');
>> H=dendrogram(Z); box

```

На рис. 14.12 наведено дендрограму 100 точок за замовченням розбитих на 30 вузлів.

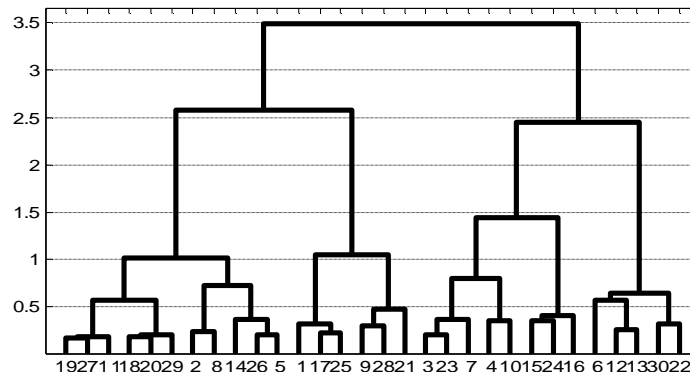


Рис. 14.12. Дерево кластерів

## **COPHENET** Коефіцієнт кофенетичної (якісної) кореляції

Дана функція обчислює коефіцієнт кофенетичної (якісної) кореляції (cophenetic correlation coefficient).

*Синтаксис:*

**c = cophenet(Z,Y)**

*Опис:*

Функція **c = cophenet(Z,Y)** обчислює коефіцієнт якісної кореляції, що порівнює інформацію про відстані  $Z$ , отриману за допомогою функції **linkage**, і інформацію про відстані  $Y$ , згенеровану функцією **pdist**. Аргумент  $Z$  – матриця розміром  $(m - 1) \times 3$ , з інформацією про відстані у 3-му стовпці. Аргумент  $Y$  – це вектор довжиною  $C_m^2 = \frac{m(m-1)}{2}$ .

Оберемо, наприклад, групу з  $m$  об'єктів з відстанями між ними  $Y$  і за допомогою функції **linkage** побудуємо з них ієрархічне дерево кластерів. Функція **cophenet** оцінює відмінність у цих класифікаціях, указуючи, як дані вписуються в побудоване дерево.

Результативний параметр  $c$  – це коефіцієнт якісної кореляції. Для якісного рішення його величина має бути близькою до 1. Значення  $c$  може бути використане для порівняння різних алгоритмів побудови ієрархічних дерев. Коефіцієнт якісної кореляції між  $Z(:,3)$  і  $Y$  обчислюється за формулою:

$$c = \frac{\sum_{i < j} (Y_{ij} - y)(Z_{ij} - z)}{\sqrt{\sum_{i < j} (Y_{ij} - y)^2 \sum_{i < j} (Z_{ij} - z)^2}},$$

де  $Y_{ij}$  – відстань між об'єктами з номерами  $i$  і  $j$  в  $Y$ ;  $Z_{ij}$  – відстань між об'єктами з номерами  $i$  і  $j$  в  $Z(:,3)$ ;  $y$  і  $z$  – середні відповідно  $Y$  і  $Z(:,3)$ .

*Приклад:*

```
>> n = 100; % кількість точок
>> X = lhsnorm(0,1,n); % нормально розподілені у квадраті точки
>> Y = pdist(X); % відстані між точками
>> Z = linkage(Y, 'ward'); % ієрархічне дерево кластерів за методом Уорда
>> c = cophenet(Z,Y) % коефіцієнт якісної кореляції
c =
```

0.6133

Значна відмінність  $c$  від 1 свідчить про погане рішення: точки розбиваються на кластери дуже нерівномірно.

### **INCONSISTENT** Коефіцієнти несумісності

Обчислюються коефіцієнти несумісності в ієрархічному дереві кластерів.

*Синтаксис:*

**$Y = inconsistent(Z)$**

**$Y = inconsistent(Z,d)$**

*Опис:*

Функція  **$Y = inconsistent(Z)$**  обчислює коефіцієнти несумісності для кожного зв'язку в ієрархічному дереві кластерів. Аргумент  $Z$  – це матриця розміром  $(m - 1) \times 3$ , створена функцією **linkage**, а  $m$  – це кількість спостережень (точок) у даних. Коефіцієнт несумісності характеризує кожний зв'язок у дереві, порівнюючи її довжину із середньою довжиною інших зв'язків на тому ж самому рівні ієрархії. Чим більше значення цього коефіцієнта, тем менш сумісні об'єкти, що з'єднані даним зв'язком.

Функція  **$Y = inconsistent(Z,d)$**  обчислює коефіцієнти несумісності для кожного зв'язку в ієрархічному дереві кластерів до глибини  $d$  (ціле позитивне число, за замовченням прийняте 2).

Вихідний параметр  $Y$  – матриця розміром  $(m - 1) \times 4$ , стовпці якої містять величини, надані в табл. 14.4:

Таблиця 14.4

#### **Зміст стовпців параметра $Y$**

<b>Стовпці</b>	<b>Опис</b>
1	Середнє арифметичне величин усіх зв'язків, включених в обчислення
2	Середньоквадратичне відхилення всіх зв'язків, включених в обчислення
3	Кількість зв'язків, включених в обчислення
4	Коефіцієнти несумісності

Для кожного зв'язку  $k$ , коефіцієнт несумісності обчислюється як

$$Y(k,4) = \frac{Z(k,3) - Y(k,1)}{Y(k,2)}$$

Для вузлів, які не мають вузлів під ними, коефіцієнт несумісності вважається рівним 0.

*Приклад:*

Побудова дерева кластерів з використанням коефіцієнтів несумісності для 10 точок, рівномірно розподілених в одиничному квадраті.

```
>> n=10; k=4; % кількість точок, кількість кластерів
>> X=lhsdesign(n,2); % рівномірно розподілені у квадраті точки
>> Y=pdist(X); % відстані між точками
>> Z=linkage(Y,'ward'); % ієрархічне дерево кластерів
>> W=inconsistent(Z,3); % коефіцієнти несумісності
>> T=cluster(Z,'maxclust',k); % розклад на 4 кластери
>> disp('Зв'язок Точки Коеф. несумісності')
>> fprintf(' %2.0f %2.0f %2.0f %12.10f\n',[1:n-1;Z(:,1:2)];W(:,4)')
>> disp('Точка кластер')
>> fprintf(' %2.0f %2.0f\n',[1:n;T'])
```

Зв'язок	Точки	Коеф. несумісності	Точка	кластер
1	6 9	0.0000000000	1	3
2	1 3	0.0000000000	2	2
3	8 10	0.0000000000	3	3
4	4 5	0.0000000000	4	4
5	7 13	0.7071067812	5	4
6	2 11	0.7071067812	6	2
7	15 16	1.4476843081	7	1
8	12 17	1.6309639675	8	1
9	14 18	1.2484896108	9	2
			10	1

Ліворуч у перших трьох стовпцях одержаної таблиці показані значення коефіцієнтів несумісності для всіх зв'язків. Спочатку поєднуються точки з найменшим значенням коефіцієнта несумісності. Так, точки 6 і 9 (після об'єднання в цієї пари буде номер 11) потраплять в один кластер, точки 1 і 3 – теж (у цієї пари тепер буде номер 12), а також точки 8 і 10 (у цієї пари тепер буде номер 13), точки 4 і 5 (у цієї пари тепер буде номер 14); далі об'єднується кластер 13 (це пара 8, 10) з точкою 7 (номер нового кластеру 15); кластер 11 (це пара 6, 9) об'єднується з точкою 2 (новому кластеру привласнюється номер 16); далі об'єднується кластери 15 і 16 (номер об'єднанного кластера 17); до цього кластера приєднується кластер 12 (це пара 1, 3) і наприкінці, до

одержаного кластера 18 приєднується кластер 14 (пара 4, 5). Цей процес триває, поки не залишиться потрібна кількість кластерів.

>> H=dendrogram(Z); box

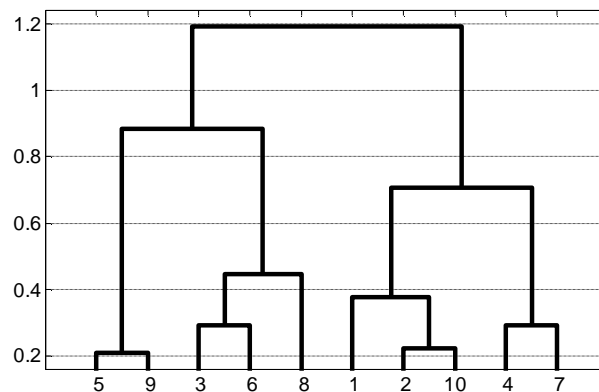


Рис. 14.13. Ієрархічне дерево кластерів

На рис. 14.13 представлена дендрограма  $Z$ .

### **KMEANS** Кластеризація об'єктів за внутрішньогруповими середніми

Можливо, це найкраща функція пакета кластеризації даних. Побудовані за її допомогою кластери добре розрізняються й мають приблизно однакову кількість точок.

*Синтаксис:*

```
IDX = kmeans(X,k)
[IDX,C] = kmeans(X,k)
[IDX,C,sumd] = kmeans(X,k)
[IDX,C,sumd,D] = kmeans(X,k)
[...] = kmeans(...,'param1',val1,'param2',val2,...)
```

*Опис:*

Функція **IDX = kmeans(X, k)** дозволяє розділити множину заданих об'єктів, заданих матрицею  $X$ , на  $k$  кластерів. Розмірність  $X$  дорівнює  $m \times n$ , де  $m$  – кількість спостережень  $n$ -вимірної випадкової величини. Рядки  $X$  відповідають спостереженням, стовпці  $X$  – ознакам багатовимірної випадкової величини. Процедура кластеризації є ітераційною.

Критерієм кластеризації є мінімум внутрішньокластерної суми квадратів відстаней точок кластера до його центроїда. Сума квадратів відстаней обчислюється за всіма кластерами. У якості відстаней точок кластера до його центроїда використовується евклідова міра. Функція повертає вектор  $IDX$  індексів кластерів для кожного спостереження багатовимірної випадкової величини. Число елементів  $IDX$  дорівнює  $m$ .

Функція  $[IDX,C] = kmeans(X,k)$  повертає також матрицю координат центроїдів кластерів  $C$ . Розмірність матриці  $C$  дорівнює  $k \times n$ .

Функція  $[IDX,C,sumd] = kmeans(X,k)$  повертає ще вектор сум квадратів відстаней об'єктів кластера до його центроїда  $sumd$ . Число елементів вектора  $sumd$  дорівнює  $k$ .

Функція  $[IDX,C,sumd,D] = kmeans(X,k)$  повертає матрицю квадратів відстаней об'єктів кластера до його центроїда  $D$ . Розмірність матриці  $D$  дорівнює  $n \times k$ .

У функції  $[...] = kmeans(...,'param1',val1,'param2',val2,...)$  додаткові вхідні параметри  $'param_1'$ ,  $'param_2'$ , ... призначені для керування роботою алгоритму кластеризації й задаються у вигляді пари "назва параметра – його значення". Можливі імена і значення параметрів наведено в табл. 14.5.

Алгоритм процедури **kmeans**.

Функція **kmeans** використовує ітераційний алгоритм мінімізації внутрішньокластерної суми квадратів відстаней об'єктів кластера до його центроїду за всіма  $k$  кластерами. Алгоритм складається двох етапів.

1. Перша фаза алгоритму призначена для пошуку наближеного значення центроїдов кластерів і попереднього угруповання об'єктів у кластери. Усі об'єкти групуються в найближчі до них кластери з наступним розрахунком відстаней між об'єктами й центроїдами. Однократне наближення за угруповання всіх об'єктів виконується за одну ітерацію.

Таблиця 14.5

**Можливі імена й значення додаткових аргументів функції *kmeans***

Ім'я	Можливі значення
------	------------------

'distance'	Відстань в n-вимірному просторі, яку мінімізує функція <b>kmeans</b> . У різних метриках центри кластерів обчислюються за різними формулами:	
	'squeclidean'	Квадратична евклідова метрика (за замовчуванням). Координати центру кластера – середнє арифметичне відповідних координат його вершин
	'cityblock'	Сума модулів різниць, тобто метрика $L_1$ . Координати центра кластера – медіани відповідних координат його вершин
	'cosine'	Одиниця мінус косинус кута між точками, розглянутими як вектори. Координати центра кластера – середнє арифметичне відповідних координат його точок після масштабування координат вершин (векторів) до одиничної довжини в евклідовій метриці
	'correlation'	Одиниця мінус вибірковий коефіцієнт кореляції між точками, розглянутими як набір значень. Координати центра кластера – середнє арифметичне відповідних координат його точок після центрування до нульового середнього й нормування до одиничного середньоквадратичного відхилення
	'Hamming'	Відсоток, що відрізняє біти (придатне лише для бінарних даних). Координати центра кластера – медіани відповідних координат його вершин
'start'	Метод, використовуваний для побудови початкових центрів кластерів (у теорії кластерного аналізу вони називаються "насіння"):	

Продовження табл. 14.5

Ім'я	Можливі значення	
'sample'	Вибирає $k$ рядків (точок) з матриці $X$ випадковим образом (за замовчуванням)	
'uniform'	Вибирає $k$ точок випадковим чином (рівномірно) з діапазону зміни $X$ . Не може застосовуватися, якщо задана метрика Хемінга ('Hamming')	
'cluster'	Центри кластерів вибираються за результатами рішення задачі кластеризації для 10% випадково обраних з $X$ точок. Для цієї допоміжної задачі використовується 'sample'	
Матриця	Матриця розміром $k \times n$ центрів кластерів у явному виді. У цьому випадку за звертання до функції <b>kmeans</b> можна вилучити аргумент $k$ , задавши замість нього порожній масив []. Функція визначить кількість кластерів як першу розмірність матриці. Можна також задати 3-вимірний масив, припускаючи довжину 3-й розмірності як значення параметра 'replicates'	
'replicates'	Кількість повторень рішення задачі кластеризації з різними початковими центрами кластерів. Повертається рішення з мінімальною величиною sumd. Цей аргумент можна задати неявно, обравши в якості параметра 'start' 3-вимірний масив	
'maxiter'	Максимальна кількість ітерацій (за замовчуванням 100).	
'emptyaction'	Дії у випадку, якщо в кластері не виявляється ні однієї точки:	
	'error'	Розглядає появу порожнього кластера як помилку (за замовчуванням)

	<i>'drop'</i>	Видаляє всі порожні кластери. Відповідні значення у параметрах C і D установлюються в NaN
	<i>'singleton'</i>	Створює новий кластер з однієї точки, найбільш далекою від центра її кластера
<i>'display'</i>	Рівень повідомлень:	
	<i>'off'</i>	Немає ніяких повідомлень
	<i>'iter'</i>	Видає інформацію про кожну ітерацію, включаючи її номер, стадію процесу оптимізації, кількість переміщених точок і загальну суму відстаней
	<i>'final'</i>	Фінальна інформація
	<i>'notify'</i>	Видаються тільки попередження й повідомлення про помилки ( за замовчуванням)

2. Друга фаза алгоритму призначена для пошуку точного й остаточного рішення. Угрупування кожного з об'єктів за кластерами, з наступним розрахунком його відстані до центроїда на цій фазі виконується за критерієм мінімізації внутрішньокластерної суми квадратів відстаней об'єктів до центроїдів кластерів. На кожній ітерації розрахунок виконується на всіх об'єктах множини даних.

Недоліком ітераційного алгоритму функції **kmeans** є можливість знаходження локального мінімуму внутрішньокластерної суми квадратів відстаней точок кластера до його центроїда за  $k$  кластери. Ця проблема може бути вирішена відповідним вибором початкової точки на початку процедури кластеризації об'єктів.

#### *Приклади:*

1. Кластеризація об'єктів на 5 кластерах за методом  $k$ -середніх. У якості даних використовується 10-ти вимірний випадковий вектор. Кількість об'єктів для кластеризації дорівнює 20. Функція повертає вектор індексів кластерів для спостережень багатовимірної випадкової величини, матрицю координат центроїдів кластерів, вектор сум квадратів відстаней об'єктів кластера до його центроїда, матрицю відстаней об'єктів кластера до його центроїда. Для розрахунку відстаней від об'єктів кластера до його центроїда використовується Манхетенівська відстань. Для визначення початкових координат центроїдів кластерів використовується випадковий вибір за рівномірним законом з діапазону зміни багатовимірної випадкової величини  $X$ .

```
>> X=normrnd(0,1,30,10)+unifrnd(-3,3,30,10);
```

```

>> [IDX,C,sumd,D] = kmeans(X,5, 'distance', 'cityblock', 'start', 'uniform')
IDX =
    4      1      5      3      3      5
    5      2      4      1      3      1
    5      1      4      1      1      3
    2      4      3      5      3      5
    3      5      2      1      4      5

C =
-0.4518  0.6920  1.2888 -0.7640  1.3453 -1.8543 -2.1945  1.1020  1.0953  0.4089
-0.7797  0.5285  2.3458 -0.1675 -0.8305 -0.1272  0.0100 -3.2330  0.1333  1.0556
 0.2036  3.2682 -1.0045 -0.5124  1.6211  1.0194  3.2315  3.2928  1.4886 -1.7915
 2.5607 -1.0515  0.3769  0.5003  2.5332 -0.9078 -0.5635 -1.1308  0.7978  1.3286
 2.2557 -2.6249 -0.9125 -2.6613 -0.3568 -3.1029 -0.5546  1.7305 -1.6048 -0.2651

sumd =
102.9955
 93.0386
 23.7344
 52.3930
 48.2298

D =
26.2894 18.3066 20.3579 29.2452 14.7314 | 8.1651 24.3271 19.7185  9.7363 21.7065
25.4628 14.2178 17.3870 23.0641  9.5695 | 8.0522 26.3177 20.4996 17.1167 25.5316
26.8366 20.6316 23.1257 13.8907 22.0910 |22.4875  7.3064 24.1471 20.8901 16.2809
27.4722 26.2328 22.5549 22.3403 16.4215 |19.9452 17.5697 12.3538 15.8344 17.1489
28.6374 14.9919 24.8707 25.2776 21.5788 |29.3167 22.7284 15.1958 28.0448 28.3705
13.3365 25.9077 22.9011 22.1990 29.1632 |13.6520 28.0968 25.7865 16.1216 25.2172
23.0609 10.3061 20.1993 25.8862 21.5349 |20.6221 19.9376 10.1631 19.9049 17.4120
30.2319  7.4816 23.5677 22.0519 21.9655 |18.2764 21.1649 12.7243 20.9071 18.7853
27.5288 21.1528 28.4354 19.2362 14.3646 |15.7617 17.8770 15.8084  8.4234 16.2312
28.1608 25.5356 20.9205 26.9543 17.3136 |28.6144 16.6454 24.7240 27.7566 11.0728
17.1987 20.7293 14.5715  4.2292 18.1665 |16.6468 24.9905 20.1514 21.9665 26.4340
23.4465 15.8990 17.6744 22.5166 12.9444 |19.2391  9.5484 10.1169 18.8261 19.3277
22.5472 17.0781 26.8716 25.3404 13.0376 |16.7605 24.8983 29.7204 23.7635 30.3281
23.0630 23.1143 21.9569 18.7928 14.5450 |21.6671 23.0071 20.2692 10.2674 18.7994
21.5271 16.2848 21.8404 19.4746  8.9038 |22.8236 25.4878 11.5379 24.0330 26.7112

```

2. Кластеризація на чотири кластери 100 точок, рівномірно розподілених в одиничному квадраті. Використовується евклідова метрика.

```

>> n =100; % кількість точок
>> k=4; % кількість кластерів
>> X=lhsdesign(n,2); % рівномірно розподілені у квадраті точки
>> T=kmeans(X,k); % розклад на 4 кластери
>> plot(X(:,1),X(:,2),'k-')
>> hold on
>> for kk=1:k,
    nk=find(T==kk); % номери точок, що попадають у кластер
    nc=convhull(X(nk,1),X(nk,2)); % опукла оболонка
    plot(X(nk(nc),1),X(nk(nc),2),'k-')
end

```

```

>> hold off, grid, box on
>> axis equal % однаковий масштаб
>> xlim([0 1]) % границі по осях
>> ylim([0 1])
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfРозклад множини точок на кластери');
>> xlabel('\bfx') % мітка осі ОХ
>> ylabel('\bfy') % мітка осі ОУ

```

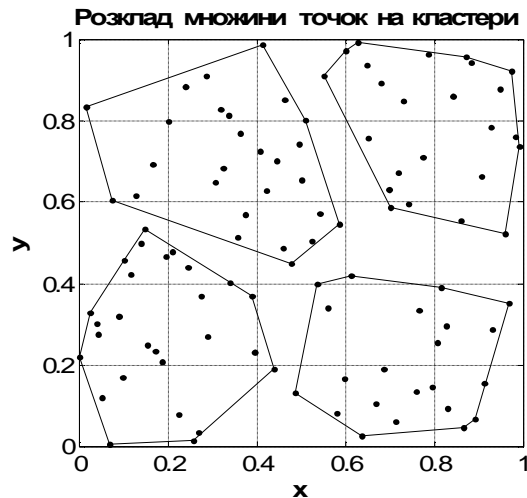


Рис. 14.14. Розклад 100 точок на чотири кластери

На рис. 14.14 показані кластери, на які розбито 100 точок за допомогою функції `kmeans`. Кластери добре відділені між собою, мають приблизно однакові розміри (в евклідовій метриці), і в кожний кластер входить приблизно однакова кількість точок.

3. Кластеризація об'єктів на 3 кластери за методом  $k$ -середніх. У якості даних використовується 5-ти вимірний випадковий вектор. Кількість об'єктів для кластеризації дорівнює 20. Функція повертає вектор індексів кластерів для спостережень багатовимірної випадкової величини. Для визначення початкових координат центроїдів кластерів використовується 3-х вимірний масив значень  $C$ . Кількість кластерів в `kmeans` задається першою розмірністю  $C$ . Число повторень процедури кластеризації визначається 3-ю розмірністю  $C$ . Максимальна кількість ітерацій обмежена 25. Робота функції `kmeans` у випадку, якщо кластер втрачає всі свої спостереження в процесі кластеризації, задана як

'singleton'. Інформація про ітерації **kmeans** відображається за кожною ітерацією.

```
>> X=normrnd(0,1,30,5)+unifrnd(-3,3,30,5);
>> A=unifrnd(-1,1,3,5)+weibrnd(2,1,3,5);
>> C = cat(3,A,A/2,A/5,A/10,A/100);
>> IDX = kmeans(X,[ ],'start',A, 'maxiter', 25, 'emptyaction', 'singleton', 'display',
'iter')
```

iter	phase	num	sum
1	1	30	385.531
2	1	2	352.065
3	1	2	345.371
4	1	1	336.842
5	2	0	336.842

5 iterations, total sum of distances = 336.842

IDX =

1	1	3	1	1	2
2	2	2	2	1	3
1	3	2	3	2	1
3	2	3	2	1	2
2	3	2	2	3	1

## **SILHOUETTE** Силуети кластерів

Обчислюються силуетні значення для всіх точок і створюється графік їх розподілу за кластерами. Силуетне значення  $i$ -ї точки  $s_i$  – це міра відповідності точки з іншими точками кластера стосовно точок інших кластерів. Вона визначається так:

Вона визначається так:  $s_i = \frac{\min_k(b_{ik}) - a_i}{\max_k(\min_k(b_{ik}), a_i)}$ , де  $a_i$  – середня

відстань від  $i$ -ї точки до всіх інших точок її кластера, а  $b_{ik}$  – середня відстань від  $i$ -ї точки до всіх інших точок іншого кластера номер  $k$ .

Визначене у такий спосіб силуетне значення буде  $s_i \in [-1; 1]$ .

*Синтаксис:*

**silhouette(X,clust)**

**s = silhouette(X,clust)**

**[s,h] = silhouette(X,clust)**

**[...] = silhouette(X,clust,distance)**

Опис:

Функція **silhouette(X,clust)** – викреслює силуети кластерів для матриці даних  $X$  розміру  $n \times p$  із кластерами, заданими в *clust*. Рядки  $X$  відповідають спостереженням (точкам), а стовпці – змінним. Аргумент *clust* – числовий вектор з номерами кластерів для кожної точки, або масив символів, або масив символівних рядків з іменами кластерів для кожної точки. Функція **silhouette** розглядає нечислові значення NaN або порожні рядки в *clust* як відсутні значення й відкидає відповідні рядки  $X$ . За замовченням **silhouette** використовує квадрат евклідової відстані між точками.

Функція **s = silhouette(X,clust)** у результативному параметрі *s* повертає вектор силуетних значень розміром  $n \times 1$ .

Функція **[s,h] = silhouette(X,clust)** – у другому результативному параметрі *h* повертає дескриптор створеної фігури.

Функція **[...] = silhouette(X,clust,distance)** в аргументі *distance* дозволяє задати формулу для обчислення відстаней. Цей аргумент повинен бути текстовим рядком. Можливі значення:

'euclidean' – евклідові відстані:

$$d_{ij} = \sqrt{(\mathbf{X}_i - \mathbf{X}_j)(\mathbf{X}_i - \mathbf{X}_j)'} = \sqrt{\sum_{k=1}^n (x_{ik} - x_{jk})^2};$$

'sqeuclidean' – квадрати евклідових відстаней (за замовчуванням):

$$d_{ij}^2 = (\mathbf{X}_i - \mathbf{X}_j)(\mathbf{X}_i - \mathbf{X}_j)' = \sum_{k=1}^n (x_{ik} - x_{jk})^2;$$

'cityblock' – відстані за метрикою "міських кварталів" (сума модулів різниць):  $d_{ij} = \sum_{k=1}^n |x_{ik} - x_{jk}|$ ;

'cosine' – одиниця мінус косинус кута між об'єктами, розглянутими

як вектори значень ознак:  $d_{ij} = 1 - \frac{\mathbf{X}_i \mathbf{X}_j'}{\sqrt{\mathbf{X}_i \mathbf{X}_i'} \sqrt{\mathbf{X}_j \mathbf{X}_j'}}$ ;

'correlation' – одиниця мінус вибіркового коефіцієнта кореляції між об'єктами, розглянутими як вибірки:

$$d_{ij} = 1 - \frac{(\mathbf{X}_i - \bar{\mathbf{X}}_i)(\mathbf{X}_j - \bar{\mathbf{X}}_j)'}{\sqrt{(\mathbf{X}_i - \bar{\mathbf{X}}_i)(\mathbf{X}_i - \bar{\mathbf{X}}_i)' } \sqrt{(\mathbf{X}_j - \bar{\mathbf{X}}_j)(\mathbf{X}_j - \bar{\mathbf{X}}_j)' }},$$

де  $\bar{X}_i = \frac{1}{n} \sum_{k=1}^n x_{ik}$ ,  $\bar{X}_j = \frac{1}{n} \sum_{k=1}^n x_{jk}$ , – центри вибірок.

'*hamming*' – відстані Хемінга (Hamming), які дорівнюють частці координат, що відрізняються:  $d_{ij} = \frac{\#(x_{ik} \neq x_{jk})}{n}$ ; де функція  $\#( )$  – кількість елементів;

'*jaccard*' – відстань Жакара (Jaccard), дорівнює частці ненульових координат, що відрізняються, серед загальної кількості ненульових координат:  $d_{ij} = \frac{\#((x_{ik} \neq x_{jk}) \cap ((x_{ik} \neq 0) \cup (x_{jk} \neq 0)))}{\#((x_{ik} \neq 0) \cup (x_{jk} \neq 0))}$ ;

числовий вектор – відстань між точками в явному вигляді, яка повертається функцією **pdist**. У цьому випадку аргумент  $X$  непотрібний, і його можна задати як порожній масив [ ].

Функція **[...] = silhouette(X, clust, distfun, p1, p2, ...)** в аргументі *distfun* дозволяє задати власну функцію користувача для обчислення відстаней між точками. Цей аргумент може бути рядком символів (ім'я m-файла) або дескриптором функції, створеним за допомогою знака @. Ця функція повинна мати наступний синтаксис виклику:  
**>> d = distfun(u, V, p1, p2, ...)**

Перший аргумент  $u$  повинен бути вектором-рядком  $1 \times p$ , що відповідають одному рядку  $X$ . Другий аргумент  $V$  має бути матрицею розміру  $m \times p$ , що відповідає декільком рядкам матриці  $X$ . Інші необов'язкові аргументи  $p_1, p_2, \dots$  – будь-які параметри, які можна передати функції **distfun**. Функція **distfun** повинна повертати у результативному параметрі  $d$  вектор-стовпець розміром  $m \times 1$ ,  $k$ -й елемент якого – відстань між точками  $u$  і  $V(k, :)$ .

*Приклад:*

1. 100 точок, рівномірно розподілені в одиничному квадраті, розбиваються на 4 кластера.

```
>> n=100; % кількість точок
>> k=4; % кількість кластерів
>> X=lhsdesign(n,2); % рівномірно розподілені у квадраті точки
>> T=kmeans(X,k); % розклад на 4 кластера
>> silhouette(X,T), grid
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfСилуети кластерів');
>> xlabel('\bfСилуетні значення') % мітка осі OX
>> ylabel('\bfКластери') % мітка осі OY
```

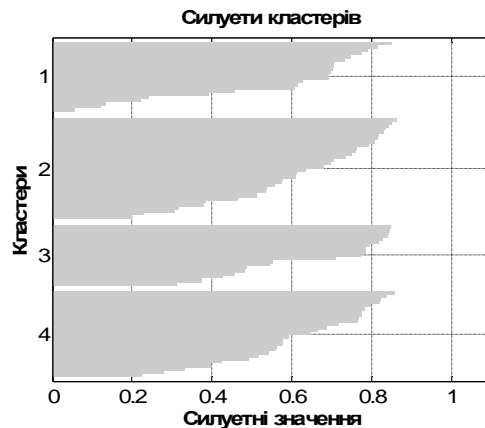


Рис. 14.15. Силуети кластерів

### **SQUAREFORM** Перетворення вектора відстаней в матрицю

Функція `pdist` обчислює попарні відстані між  $n$ -точками й повертає їх у вигляді вектора довжиною  $C_m^2 = \frac{m(m-1)}{2}$ . Але іноді зручніше мати їх у вигляді квадратної матриці  $m \times m$ . Дана функція перетворює один вид в інший, і була використана вище у прикладах до опису функції `pdist`.

*Синтаксис:*

```
Z = squareform(y)
y = squareform(Z)
Z = squareform(y,'tovector')
Y = squareform(Z,'tomatrix')
```

*Опис:*

Функція **Z = squareform(y)** для вектора  $y$ , що повертається функцією **pdist**, створює симетричну квадратну матрицю  $Z$  з нулями на головній діагоналі, у якої кожний елемент  $Z(i,j)$  є відстань між  $i$ -ю і  $j$ -ю точками.

Функція **y = squareform(Z)** для квадратної симетричної матриці  $Z$  з нулями на головній діагоналі створює вектор  $y$  з відстанями у форматі функції **pdist**.

Функція **Z = squareform(y,'tovector')** примусово змушує функцію **squareform** розглядати  $y$  як вектор.

Функція **Y = squareform(Z,'tomatrix')** примусово змушує функцію **squareform** розглядати  $Z$  як матрицю.

Два останні формати корисні, якщо вхідний параметр – одне число, так що незрозуміло, чи є параметр вектором або матрицею.

*Приклад:*

```
>> n=4; % кількість точок
>> X=lhsdesign(n,2); % рівномірно розподілені у квадраті точки
>> y=pdist(X) % вектор
>> Z=squareform(y) % матриця
y =
    0.5540    0.6907    0.5386    0.6632    0.5717    1.0992
Z =
     0    0.5540    0.6907    0.5386
    0.5540     0    0.6632    0.5717
    0.6907    0.6632     0    1.0992
    0.5386    0.5717    1.0992     0
```

У цьому розділі розглянуті функції автоматичної класифікації даних – це альтернативна назва кластерного аналізу. Як синоніми вживають також назви “таксономія”, “стратифікація”, “розпізнавання образів без вчителя”. Слід відрізнити цей аналіз від звичайного угруповання даних за однією або кількома ознаками. На відміну від групування, мета кластерного аналізу – знайти природне розшарування об’єктів; не виключено, що всі об’єкти складають один кластер.

Кластерний аналіз ефективно працює у скорочених просторах латентних змінних, що розглядаються у наступному розділі.

## 15. Функції зниження розмірності задачі

Коли досліджуваний процес залежить від великої кількості ознак, актуальною стає проблема зниження розмірності задачі. Дійсно, якщо дві (або більше) змінних зв'язані кореляційними зв'язками, то або одна зі змінних визначає іншу, або ж обидві ці змінні є наслідками якоїсь загальної причини (фактора). Модель факторного аналізу зв'язана саме з останнім припущенням. Сам фактор – це латентна (прихована) ознака, що породжує дані. У моделі факторного аналізу вважається, що спостережувані випадкові величини є наслідками невеликої кількості деяких причин (факторів), які безпосередньо не спостерігаються.

У цьому розділі описуються різні функції методу головних компонент і факторного аналізу, що дозволяють знизити розмірність розв'язуваних задач.

Модель простого факторного аналізу (лінійна нормальна модель із взаємно ортогональними загальними факторами й з некорельованими залишками) може бути представлена в матричному вигляді:

$$X = FL\zeta + e, \quad (1)$$

де  $X$  – матриця стандартизованих даних розміром  $n \times m$ , стовпці якої є стандартизовані ознаки  $X = \{X_1, X_2, \dots, X_m\}$ , а рядки – спостереження (об'єкти);  $F$  – матриця взаємно незалежних, стандартизованих загальних факторів розміром  $n \times d$ :  $F = \{f_1, f_2, \dots, f_d\}$ ;  $\Lambda$  – матриця коефіцієнтів лінійного перетворення, що називається матрицею навантажень факторів на досліджувані змінні, розмір цієї матриці  $m \times d$  ( $d \leq m$ );  $e$  – діагональна матриця залишків (похибок), які мають  $m$ -вимірний нормальний розподіл з нульовими середніми і з невідомою але однаковою дисперсією  $\sigma_e^2$ . Елемент матриці  $\Lambda_{ij}$  називається навантаженням  $i$ -ї змінної на  $j$ -й фактор, або навпаки, навантаженням  $j$ -го фактора на  $i$ -ю змінну.

Іншою формою запису моделі факторного аналізу є вираження:

$$\text{cov}(X) = \Lambda\Lambda' + \Psi, \quad (2)$$

де  $\Psi = cov(e)$  є діагональною матрицею дисперсій незалежних специфічних факторів. Розмірність матриці  $\Psi$  дорівнює  $d \times d$ .

Як виходить з (1) або (2), значення ознаки багатовимірної випадкової величини  $X_i$  може бути виражене зваженою сумою загальних факторів  $f$  і залишкового члена  $e_i$ . Вагами факторів  $f$  є елементи матриці навантажень  $\Lambda$ . Кількість факторів  $f$  має бути меншою числа ознак багатомірної випадкової величини  $X$ , що дозволяє зменшити розмірність координатного простору з  $m$  до  $d$ . У простому факторному аналізі передбачається, що гіпотетичні фактори взаємно незалежні і їх дисперсії дорівнюють одиниці, а специфічні фактори  $e_i$  не залежні від будь-якого  $f_i$ , де  $i = 1..m, j = 1..d$ .

Суму квадратів навантажень, що відповідають  $X_i$  називають спільністю  $i$ -ї ознаки. Спільність характеризує частину дисперсії  $\sigma^2(X_i)$ , яка пояснюється загальними факторами  $f$ . Величина  $(e_i)^2$  показує, яка частка дисперсії  $\sigma^2(X_i)$   $i$ -ї ознаки залишається непоясненою для обраного набору загальних факторів  $f$ . Цю величину називають специфічністю  $i$ -ї ознаки.

Коефіцієнт кореляції двох ознак  $X_i$  і  $X_j$  можна виразити сумою добутків навантажень некорельованих факторів

$$r_{ij} = r(X_i, X_j) = \sum_{k=1}^m \Lambda_{ik} \Lambda_{jk}$$

Задача факторного аналізу – розрахунок матриці навантажень  $\Lambda$  для заданого вектора  $f$  – не має однозначного рішення. Представлення коваріаційної матриці ознак  $cov(X)$  багатовимірної випадкової величини  $X$  факторами  $f$ , або інакше, факторизацію  $cov(X)$ , можна провести нескінченно великим числом способів. Якщо знайдено одну матрицю навантажень, то будь-яке лінійне ортогональне перетворення  $T$ , або інакше, ортогональне обертання  $T$  приведе до еквівалентної факторизації за моделлю (1).

Головні компоненти є однією з найпоширених початкових систем ортогональних факторів. Вони складаються за екстремальним принципом – перша компонента пояснює максимум загальної мінливості даних; друга компонента, що ортогональна до першої, пояснює максимум залишкової мінливості; внесок останніх компонент вже близький до нуля, тобто останні компоненти практично не варіюють. Перші найбільш значимі компоненти називаються головними. Саме перехід від  $m$ -вимірного простору вихідних змінних  $X_i$  до  $p$ -вимірного простору головних компонент  $f_j$  дозволяє розглянути проблему у скороченому факторному просторі. Подальші перетворення початкової системи головних компонент провадять з метою покращення інтерпретації факторного розв'язку.

У цьому розділі наведені наступні функції факторного аналізу:

<b>PRINCOMP</b>	Аналіз головних компонент за матрицею даних .....	296
<b>PCARES</b>	Розрахунок залишків (нев'язок) після аналізу головних компонент .....	299
<b>PCACOV</b>	Аналіз головних компонент за коваріаційною матрицею .....	304
<b>ROTATEFACTORS</b>	Обертання для факторного аналізу .....	306
<b>FACTORAN</b>	Загальний факторний аналіз .....	310
<b>BIPLOT</b>	Діаграма факторних навантажень і об'єктів .....	322
<b>BARTTEST</b>	Тест Бартлета для оцінки розмірності простору головних компонент .....	324

### **PRINCOMP** Аналіз головних компонент за матрицею даних

Проводиться виділення головних компонент, використовуючи матрицю вихідних даних. Якщо потрібно провести аналіз головних компонент за коваріаційною матрицею, використовується функція **pcacov**.

*Синтаксис:*

**PC = princomp(X)**

**[PC,SCORE,latent,tsquare] = princomp(X)**

*Опис:*

Функція **PC = princomp(X)** призначена для проведення аналізу головних компонент багатовимірної випадкової величини  $X$ . Аргумент  $X$  є матрицею даних розмірності  $n \times m$ . Стовпці матриці  $X$  відповідають ознакам, рядки – спостереженням багатовимірної випадкової величини. Функція повертає матрицю  $PC$  – матрицю переходу від ознак  $X$  до компонент і навпаки. Матриця  $PC$  є множиною власних векторів коваріаційної матриці  $cov(X)$ . Розмірність матриці  $PC$  дорівнює  $n \times n$ , де  $n$  – кількість ознак багатовимірної випадкової величини, або число стовпців матриці  $X$ .

*Примітка:* Вхідний параметр  $X$  повинен бути заданий як матриця вихідних даних, а не як коваріаційна матриця. Щоб провести аналіз головних компонент зі стандартизованими змінними, використовуйте команду: **COEFF = princomp(zscore(X))**.

Функція **[PC,SCORE,latent,tsquare] = princomp(X)** повертає матрицю головних компонент  $PC$ , матрицю  $Z$ -множини даних  $SCORE$ , власні значення  $latent$  коваріаційної матриці  $cov(X)$ , вектор значень статистики  $T^2$  Хотелінга  $tsquare$  для кожного зі спостережень.

$Z$ -множина даних формується проектуванням матриці спостережень  $X$  у просторі компонент. Елементи вектора  $latent$  є дисперсіями стовпців матриці  $SCORE$ . Статистика  $T^2$  Хотелінга є мірою відстані окремих спостережень щодо центру групування вихідних даних в багатовимірному просторі.

*Приклади:*

1. У прикладі аналізуються головні компоненти вибірки з 5-вимірною нормального розподілу, у якій сильно корелюють між собою 1-а, 3-я й 5-а координати, а також 2-а й 4-а, а кореляція між цими групами змінних – слабка.

**>> n=1000;** % число точок

```

>> mu=[1 2 3 -1 0]; % середні
>> s=[ 2 0.01 1.9 0.02 1.4;...
      0.01 1 0.01 0.95 0.04;...
      1.9 0.01 3 0.05 2.2;...
      0.02 0.95 0.05 1 0.02;...
      1.4 0.04 2.2 0.02 2]; % коваріаційна матриця
>> X=mvnrnd(mu,s,n); % багатовимірна нормальна вибірка
>> [c,sc,lat]=princomp(X);
>> expl=lat/sum(lat)*100; % відсотки
>> fprintf('Частка головних компонент у загальній сумі\n N    Відсоток\n')
>> fprintf('%2.0f %12.8f\n',[[1:length(mu)];expl'])
>> disp('Матриця коефіцієнтів головних компонент')
>> fprintf('%12.8f %12.8f %12.8f %12.8f %12.8f\n',c')
Частка головних компонент у загальній сумі
N    Відсоток
1 67.64643143
2 22.16524586
3  7.13098451
4  2.55692074
5  0.50041746
Матриця коефіцієнтів головних компонент
0.52279408  0.01134456 -0.83999751 -0.14453444  0.00846350
0.00911954 -0.70171685  0.01044408 -0.12386608 -0.70146876
0.67026787  0.00026085  0.30078290  0.67019314 -0.10541211
0.01160553 -0.71218007 -0.01326047  0.10378747  0.69405849
0.52650686  0.01625592  0.45127476 -0.70981428  0.12264197

```

Результати розрахунку показують, що перші дві головні компоненти даних охоплюють майже 90% сумарного розкиду даних. Перший стовпець матриці  $c$  показує, що 1-а головна компонента є лінійною комбінацією 1-ї, 3-ї і 5-ї координат матриці  $X$ . Другий стовпець матриці показує, що 2-а компонента є лінійною комбінацією 2-ї і 4-ї координат матриці  $X$ .

2. Розглянемо файл даних `hald.mat` і проведемо аналіз головних компонент 4-х вимірної випадкової величини. Функція повертає матрицю головних компонент, матрицю  $Z$ -множини даних, власні значення коваріаційної матриці  $cov(X)$ , вектор значень статистики  $T^2$  Хотелінга для кожного зі спостережень.

Завантажимо файл і виділимо з нього змінну `ingredients` у вигляді матриці розміром  $13 \times 4$ :

```

>> load hald          || 11 31 8 47          || 11 66 9 12
>> ingredients        || 7 52 6 33          || 10 68 8 12
ingredients =         || 11 55 9 22         || 21 47 4 26
  7 26 6 60           || 3 71 17 6          || 1 40 23 34
  1 29 15 52          || 1 31 22 44         ||
  11 56 8 20          || 2 54 18 22         ||

```

Знайдемо власні значення і власні вектори коваріаційної матриці, а також обчислимо значення компонент:

```

>> [pc,score,latent,tsquare]=princomp(ingredients)
pc =
  0.0678  0.6460 -0.5673  0.5062      0.5532  4.4617  6.0874  0.1424
  0.6785  0.0200  0.5440  0.4933      10.8125  3.6466 -0.9130 -0.1350
  -0.0290 -0.7553 -0.4036  0.5156      32.5882 -8.9798  1.6063  0.0818
  -0.7309  0.1085  0.4684  0.4844     -22.6064 -10.7259 -3.2365  0.3243
  -0.7309  0.1085  0.4684  0.4844      9.2626 -8.9854  0.0169 -0.5437
score =
  -36.8218  6.8709  4.5909  0.3967      3.2840  14.1573 -7.0465  0.3405
  -29.6073 -4.6109  2.2476 -0.3958     -9.2200 -12.3861 -3.4283  0.4352
  12.9818  4.2049 -0.9022 -1.1261      25.5849  2.7817  0.3867  0.4468
  -23.7147  6.6341 -1.8547 -0.3786     26.9032  2.9310  2.4455  0.4116

latent =
  517.7969      3.0758      2.6086
  67.4964       6.0002      7.4818
  12.4054       2.6198      4.1830
  0.2372        3.3681      2.2327
  0.2372        0.5668      2.7216

tsquare =
  5.6803      3.4818
  5.6803      3.9794

```

Приведемо матрицю коваріацій для цих даних:

```

>> cov(ingredients)
ans =
  34.6026  20.9231 -31.0513 -24.1667
  20.9231  242.1410 -13.8782 -253.4167
 -31.0513 -13.8782  41.0256  3.1667
 -24.1667 -253.4167  3.1667  280.1667

```

## **PCARES** Розрахунок залишків (нев'язок) після аналізу головних компонент

Оцінюється залишкова похибка після виділення головних компонент.

*Синтаксис:*

```

residuals = pcares(X,ndim)
[residuals,reconstructed] = pcares(X,ndim)

```

Опис:

Функція **residuals = pcares(X,ndim)** повертає матрицю залишків *residuals* за *ndim* головними компонентами. Вхідний параметр *X* є матрицею вихідних даних. Стовпці матриці *X* відповідають ознакам, рядки – спостереженням багатовимірної випадкової величини. Вхідний параметр *ndim* повинен бути заданий як скаляр. Величина *ndim* повинна бути менше числа стовпців матриці *X*. Розмірність матриць *residuals* і *X* буде однаковою.

*Примітка:* Матриця *X* повинна бути задана як матриця вихідних даних, а не коваріаційна матриця. Функція **pcares** не нормує стовпці *X*. Якщо потрібно провести аналіз головних компонент для стандартизованих змінних, використовуйте **residuals = pcares(zscore(X),ndim);**

Функція **[residuals,reconstructed] = pcares(X,ndim)** у результативному параметрі **reconstructed** повертає відновлені спостереження, тобто апроксимацію *X* після збереження в ньому лише *ndim* головних компонент.

*Приклади:*

1. Оцінимо нев'язки п'ятивимірної випадкової величини, що містить 1000 спостережень із заданими вектором середніх і матрицею коваріацій. Результат представимо в графічному вигляді.

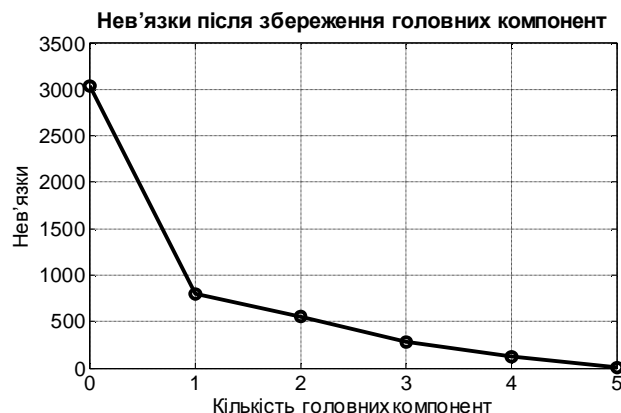


Рис. 15.1. Величина нев'язок після збереження в даних різної кількості головних компонент

На рис. 15.1 показана величина нев'язок після збереження в даних різної кількості головних компонент. Зі збільшенням числа головних компонент нев'язки зменшуються.

Оператори:

```
>> n=1000; % число точок
>> mu=[1 2 3 -1 0]; % середні
>> s=[ 2 0.01 1.9 0.02 1.4;...
      0.01 1 0.01 0.95 0.04;...
      1.9 0.01 3 0.05 2.2;...
      0.02 0.95 0.05 1 0.02;...
      1.4 0.04 2.2 0.02 2]; % коваріаційна матриця
>> X=mvnrnd(mu,s,n); % багатовимірна нормальна вибірка
>> nn=norm(X,1); % норми
>> for kk=1:5,
    res=pcares(X,kk); % нев'язки з kk головними компонентами
    nn=[nn norm(res,1)];
end
>> plot(0:5,nn)
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',16)
>> title('\bфНев'язки після збереження головних компонент');
>> xlabel('Кількість головних компонент') % мітка осі OX
>> ylabel('Нев'язки') % мітка осі OY
```

2. Розрахунок залишків щодо 1, 2, 3 головних компонент для 4-вимірної випадкової величини, узятій з файла hald.mat. Об'єм вибірки дорівнює 13 елементам.

```
>> load hald
>> ingredients
ingredients =
    7    26    6    60    11    31    8    47    21    47    4    26
    1    29   15    52    7    52    6    33    1    40   23    34
    11   56    8    20   11   55    9    22   11   66    9    12
    7    26    6    60    3    71   17    6   10   68    8    12
    1    29   15    52    1    31   22   44
    11   56    8    20    2    54   18   22
```

```
>> [residuals1,reconstructed1] = pcares(ingredients,1)
residuals1=
    2.0350    2.8304   -6.8378    3.0879
   -4.4542    0.9352    2.3715    0.3608
    2.6583   -0.9622   -3.3925   -0.5120
    5.1463   -1.0630   -4.4575   -0.3325
   -0.4990    3.4708   -5.7532    3.4043
    2.8054   -0.4903   -2.4554   -0.0974
   -6.6710    0.7346    6.1765   -0.1822
   -4.9288   -1.8150    9.5747   -2.5224
   -6.0895   -0.4387    6.4996   -1.2302
   13.3158   -3.3821   -7.6739   -1.5998
   -5.8364   -1.8979   10.9632   -2.7387
    1.8038    0.4864   -2.0267    0.6993
    0.7144    1.5919   -2.9885    1.6628
reconstructed1=
    4.9650   23.1696   12.8378   56.9121
    5.4542   28.0648   12.6285   51.6392
    8.3417   56.9622   11.3925   20.5120
    5.8537   32.0630   12.4575   47.3325
    7.4990   48.5292   11.7532   29.5957
    8.1946   55.4903   11.4554   22.0974
    9.6710   70.2654   10.8235    6.1822
    5.9288   32.8150   12.4253   46.5224
    8.0895   54.4387   11.5004   23.2302
    7.6842   50.3821   11.6739   27.5998
    6.8364   41.8979   12.0368   36.7387
    9.1962   65.5136   11.0267   11.3007
    9.2856   66.4081   10.9885   10.3372
```

Одержали нев'язки й відновлені змінні після збереження лише однієї (першої) головної компоненти, яка пояснює 87% мінливості даних.

Тепер обчислимо нев'язки й відновлені змінні після збереження перших двох головних компонент (що пояснює 98% мінливості даних).

```
>> [residuals2,reconstructed2] = pcares(ingredients,2)
residuals2=
-2.4037  2.6930 -1.6482  2.3425
-1.4755  1.0274 -1.1111  0.8610
-0.0582 -1.0463 -0.2165 -0.9681
 0.8606 -1.1957  0.5533 -1.0521
-3.3814  3.3816 -2.3832  2.9203
 0.4496 -0.5632  0.2988 -0.4930
-0.8699  0.9141 -0.6061  0.7920
 2.0003 -1.6006  1.4733 -1.3589
-0.2848 -0.2590 -0.2872 -0.2555
 4.1699 -3.6651  3.0192 -3.1356
 2.1652 -1.6503  1.6079 -1.3950
 0.0068  0.4308  0.0743  0.3976
-1.1790  1.5333 -0.7747  1.3449
reconstructed2=
 9.4037  23.3070  7.6482  57.6575
 2.4755  27.9726 16.1111  51.1390
11.0582  57.0463  8.2165  20.9681
10.1394  32.1957  7.4467  48.0521
10.3814  48.6184  8.3832  30.0797
10.5504  55.5632  8.7012  22.4930
 3.8699  70.0859 17.6061  5.2080
-1.0003  32.6006 20.5267  45.3589
 2.2848  54.2590 18.2872  22.2555
16.8301  50.6651  0.9808  29.1356
-1.1652  41.6503 21.3921  35.3950
10.9932  65.5692  8.9257  11.6024
11.1790  66.4667  8.7747  10.6551
```

Наприкінці обчислимо нев'язки й відновлені змінні після збереження трьох головних компонент:

```
>> [residuals3,reconstructed3] = pcares(ingredients,3)
residuals3=
 0.2008  0.1957  0.2045  0.1921
-0.2004 -0.1953 -0.2041 -0.1918
-0.5700 -0.5555 -0.5806 -0.5455
-0.1916 -0.1867 -0.1952 -0.1834
 0.0721  0.0702  0.0734  0.0690
-0.0683 -0.0666 -0.0696 -0.0654
 0.0414  0.0403  0.0422  0.0396
 0.1642  0.1600  0.1672  0.1571
-0.2752 -0.2682 -0.2803 -0.2634
 0.1724  0.1680  0.1756  0.1649
 0.2203  0.2146  0.2244  0.2108
 0.2262  0.2204  0.2304  0.2164
 0.2083  0.2030  0.2122  0.1994
reconstructed3=
 6.7992  25.8043  5.7955  59.8079
 1.2004  29.1953 15.2041  52.1918
11.5700  56.5555  8.5806  20.5455
11.1916  31.1867  8.1952  47.1834
 6.9279  51.9298  5.9266  32.9310
11.0683  55.0666  9.0696  22.0654
 2.9586  70.9597 16.9578  5.9604
 0.8358  30.8400 21.8328  43.8429
 2.2752  54.2682 18.2803  22.2634
20.8276  46.8320  3.8244  25.8351
 0.7797  39.7854 22.7756  33.7892
10.7738  65.7796  8.7696  11.7836
 9.7917  67.7970  7.7878  11.8006
```

Порівняємо отримані результати з даними файлу.

Після збереження лише першої головної компоненти:

```
>> for k=1:4
subplot(2,2,k)
plot(ingredients(:,k),reconstructed1(:,k),'o'),grid
title(char(48+k))
end
```

Після збереження двох компонент:

```
>>for k=1:4  
subplot(2,2,k)  
plot(ingredients(:,k),reconstructed2(:,k),'o'),grid  
title(char(48+k))  
end
```

Після збереження трьох компонент:

```
>> for k=1:4  
subplot(2,2,k)  
plot(ingredients(:,k),reconstructed3(:,k),'o'),grid  
title(char(48+k))  
end
```

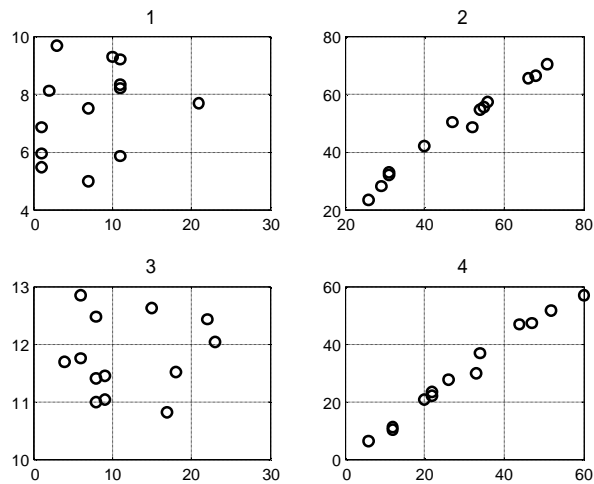


Рис. 15. 2. Змінні після збереження першої головної компоненти

На рис. 15.2 бачимо, що одна перша компонента визначає (і визначає досить точно) лише другу і четверту змінні, її внесок – 87%.

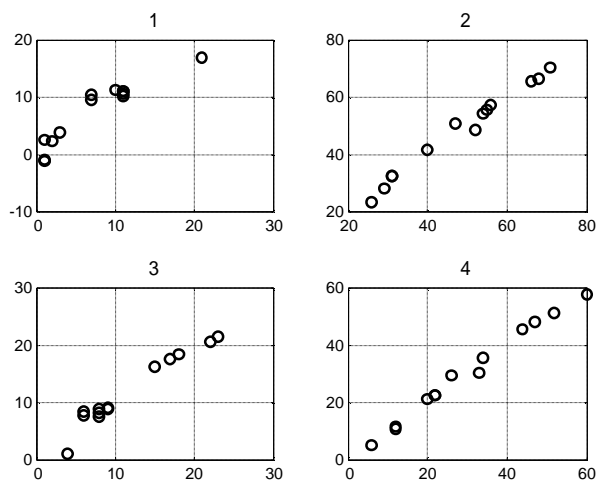


Рис.15.3. Змінні після збереження двох компонент

З рис 15.3 бачимо, що дві компоненти визначають вже всі 4 змінні (хоча першу змінну відтворює не дуже якісно). Їх внесок дорівнює 98%.

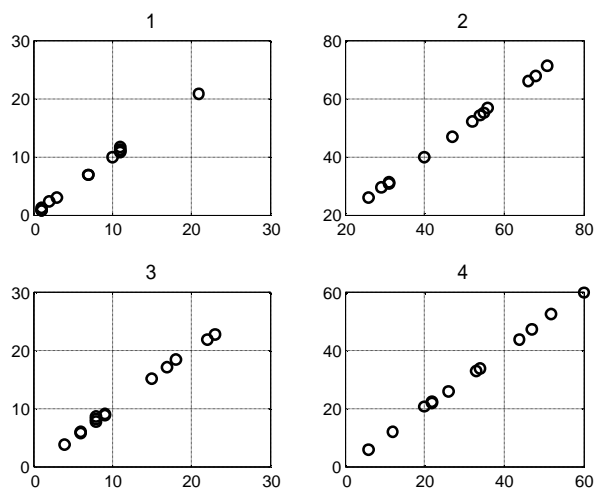


Рис.15.4. Змінні після збереження трьох компонент

З рис. 15.4 видно, що три компоненти визначають всі чотири змінні майже точно (їх внесок дорівнює  $\approx 100\%$ ).

### **PCA** Аналіз головних компонент за коваріаційною матрицею

Проводиться виділення головних компонент, використовуючи коваріаційну матрицю даних. Якщо потрібно провести аналіз головних

компонент за матрицею вихідних даних, використовуйте функцію **princomp**.

*Синтаксис:*

```
pc = pcacov(X)  
[pc,latent,explained] = pcacov(X)
```

*Опис:*

Функція **pc = pcacov(X)** дозволяє провести аналіз головних компонент на підставі коваріаційної матриці  $X$  вихідних даних. Функція повертає матрицю головних компонент  $pc$ . Матриця  $pc$  є множиною власних векторів коваріаційної матриці  $X$ . Розмірності матриць  $pc$  і  $X$  будуть однаковими.

Функція **[pc,latent,explained] = pcacov(X)** повертає матрицю головних компонент  $pc$ , вектор власних значень коваріаційної матриці  $latent$ , і вектор часток  $explained$ , що пояснюються власними векторами (головними компонентами), у відсотках від загальної дисперсії. Число елементів векторів  $latent$  і  $explained$  дорівнює кількості стовпців або рядків матриці  $X$ .

*Приклади:*

1. У прикладі аналізуються головні компоненти вибірки з 5-вимірного нормального розподілу, у якій сильно корелюють між собою 1-а, 3-я й 5-я координати, а також 2-а й 4-а, але кореляція між цими групами змінних – слабка.

```
>> n=1000; % число точок  
>> mu=[1 2 3 -1 0]; % середні  
>> s=[ 2 0.01 1.9 0.02 1.4;...  
      0.01 1 0.01 0.95 0.04;...  
      1.9 0.01 3 0.05 2.2;...  
      0.02 0.95 0.05 1 0.02;...  
      1.4 0.04 2.2 0.02 2]; % коваріаційна матриця  
>> X=mvnrnd(mu,s,n); % багатовимірна нормальна вибірка  
>> V=cov(X); % коваріаційна матриця  
>> [c,lat,expl]=pcacov(V);  
>> fprintf('Частка головних компонент у загальній сумі\n N   відсоток\n')  
>> fprintf('%2.0f  %12.8f\n',[[1:length(mu)];expl'])  
>> disp('Матриця коефіцієнтів головних компонент')  
>> fprintf('%12.8f  %12.8f  %12.8f  %12.8f  %12.8f\n',c')
```

```
Частка головних компонент у загальній сумі  
N   відсоток
```

```

1 66.23844167
2 23.50879056
3 6.94281739
4 2.80573037
5 0.50422001

```

Матриця коефіцієнтів головних компонент

```

-0.47118944  0.01053335  0.86917975  -0.14950616  -0.00663629
 0.02101990  -0.70375042  0.00870520  -0.09644729  0.70350242
-0.69468773  -0.02839940  -0.26139244  0.66391016  0.08660096
 0.01572669  -0.70950217  0.02689463  0.08781002  -0.69851659
-0.54286307  -0.02060441  -0.41880958  -0.72101215  -0.09805692

```

Результати розрахунку підтверджують, що перші дві головні компоненти даних охоплюють майже 90% сумарного розкиду даних. Перший стовпець матриці с показує, що 1-а головна компонента є лінійною комбінацією 1-ї, 3-ї і 5-ї координат, а другий стовпець, що 2-га компонента є лінійною комбінацією 2-ї і 4-ї координат.

2. Застосуємо цю функцію до файла даних hald.mat

```

>>load hald
>> [c,lat,expl]=pcacov(V);
c =
-0.1279  0.0428  0.6459  -0.7514
-0.8398  0.5092  0.0181  0.1875
-0.1984  -0.0721  -0.7557  -0.6199
-0.4889  -0.8566  0.1067  0.1261
lat =
211.3369
expl =
77.2356
28.4597
10.2667
64.5700
23.5979
8.6953
3.1368

```

## **ROTATEFACTORS** Обертання для факторного аналізу

Це – допоміжна функція. Вона використовується іншими функціями для різних обертань матриці навантаження факторів. Коли використовуються ортогональні перетворення, вони називаються обертаннями (серед ортогональних перетворень крім обертань є ще перетворення віддзеркалення). Нові фактори після обертання зберігають ортогональність. Мета перетворення початкового факторного рішення – одержати матрицю факторних навантажень, коефіцієнти якої близьки або до одиниці, або до нуля. Таке факторне рішення має змістовну інтерпретацію.

*Синтаксис:*

**B = rotatefactors(A)**

```
B = rotatefactors(A,'Method','orthomax','Coeff',gamma)  
B = rotatefactors(A,'Method','procrustes','Target',target)  
B = rotatefactors(A,'Method','pattern','Target',target)  
B = rotatefactors(A,'Method','promax')
```

Опис:

Функція **B = rotatefactors(A)** перетворює матрицю навантажень факторів  $A$  розміру  $d \times m$  з метою максимізації дисперсії квадратів ступенів навантажень за методом *'varimax'* і повертає результат у вигляді матриці  $B$ . Рядки  $A$  і  $B$  відповідають змінним, а стовпці – факторам, тобто, наприклад, елемент  $A(i, j)$  – це коефіцієнт для  $i$ -ї змінної в  $j$ -му факторі. Звичайно матриця  $A$  містить коефіцієнти головних компонентів, створені функціями **princomp** або **pcacov**, або оцінки факторів, що завантажуються для функції **factoran**.

Функція **B = rotatefactors(A,'Method','orthomax','Coeff',gamma)** обертає матрицю  $A$  з метою максимізації за методом *'orthomax'* з коефіцієнтом  $gamma$ , тобто  $B$  – це ортогональне обертання  $A$  з метою максимізації виразу:  $sum(B*sum(B.^4, 1) - gamma*sum(B.^2, 1).^2)$ .

За замовчуванням  $gamma = 1$ , що відповідає методу *'varimax'*. Інші можливі значення  $gamma$  – це  $0$ ,  $m/2$  і  $d \times (m - 1) / (d+m-2)$ , відповідні до методів *'quartimax'*, *'equamax'* і *'parsimax'*. Можна також задати рядок *'varimax'*, *'quartimax'*, *'equamax'* або *'parsimax'* для параметра *'method'* і вилучити параметр *'Coeff'*.

Якщо *'Method'* заданий як *'orthomax'*, *'varimax'*, *'quartimax'*, *'equamax'* або *'parsimax'*, то для них є додатковий параметр *'Normalize'* – прапорець, що показує, чи повинна матриця факторів, що завантажуються, бути нормалізованою за рядками перед обертанням. Якщо задано *'on'* (за замовчуванням), то рядки  $A$  нормалізуються перед обертанням до одиничної евклідової норми, і не нормалізуються після обертання. Якщо ж *'Normalize'* встановлено в *'off'*, обертаються вихідні дані. Параметр *'Reltol'* – відносна точність збіжності ітераційного процесу, використовуваного для знаходження матриці обертання  $T$ . Його значення за замовчуванням дорівнює  $\sqrt{eps}$ . Параметр *'Maxit'* –

максимальна кількість ітерацій в алгоритмі знаходження  $T$  (за замовчуванням 250).

Функція  **$B = \text{rotatefactors}(A, 'Method', 'procrustes', 'Target', target)$**  проводить кутове обертання матриці  $A$  за методом *'procrustes'* з матрицею  $target$  такого ж розміру  $d \times m$ , заданої у вигляді параметра *'Target'*.

Функція  **$B = \text{rotatefactors}(A, 'Method', 'pattern', 'Target', target)$**  проводить кутове обертання матриці факторів  $A$  до матриці зразка  $target$  розміром  $d \times m$  і повертає результату вигляді матриці  $B$ . Аргумент  $target$  визначає "обмежені" елементи  $B$ , тобто елементи  $B$ , відповідні до нульових елементів  $target$ , які метод *'pattern'* прагне зробити якнайменшими (за модулем), у той час як усі інші елементи  $B$  можуть мати будь-яку величину.

Якщо *'Method'* заданий як *'procrustes'* або *'pattern'*, додатковий параметр *'Type'* визначає тип обертання. Якщо *'Type'* задано як *'orthogonal'*, то проводяться ортогональні обертання, і фактори залишаються некорельованими. Якщо *'Type'* встановлено в *'oblique'* (за замовчуванням), проводяться кутові обертання, і фактори після обертання можуть виявитися корельованими.

Якщо *'Method'* задано як *'pattern'*, то є обмеження на значення  $target$  параметра *'Target'*. А саме: якщо в  $A$  є  $m$  стовпців, то для ортогонального обертання  $j$ -й стовпець  $target$  повинен мати принаймні  $(m - j)$  нулів. Для кутового обертання кожний стовпець  $target$  повинен мати принаймні  $(m - 1)$  нулів.

Функція  **$B = \text{rotatefactors}(A, 'Method', 'promax')$**  обертає  $A$  для максимізації за критерієм *'promax'*, що еквівалентне кутовому прокрустовому обертанню з параметром  $target$ , створеним за допомогою обертання за критерієм *'orthomax'*. Тут можна використовувати 4 параметри керування обертанням *'orthomax'*; вони будуть використовуватися всередині *'promax'*.

Додатковим параметром для 'promax' є 'Power', показник для створення матриці *target* для 'promax'. Цей параметр повинен бути рівний 1 або більше. Значення за замовчуванням – 4.

Функція **[B,T]=rotatefactors(A, ...)** у вихідному параметрі *T* повертає матрицю обертання, використану при створенні *B*. Виконується  $B = A * T$ ; а  $inv(T' * T)$  – це кореляційна матриця факторів, що обертаються. Для ортогонального обертання це буде одинична матриця, а для кутового – містити одиниці на головній діагоналі, але мати деякі недіагональні елементи, відмінні від нуля.

*Приклад:*

```
>> n=1000; % число точок
>> mu=[1 2 3 -1 0]; % середні
>> s=[ 2 0.01 1.9 0.02 1.4;...
      0.01 1 0.01 0.95 0.04;...
      1.9 0.01 3 0.05 2.2;...
      0.02 0.95 0.05 1 0.02;...
      1.4 0.04 2.2 0.02 2]; % коваріаційна матриця
>> X=mvnrnd(mu,s,n); % багатовимірна нормальна вибірка
>> [c,sc,lat]=princomp(X); % головні компоненти
>> disp('Матриця коефіцієнтів головних компонент')
>> fprintf('%12.8f %12.8f %12.8f %12.8f %12.8f\n',c')
>> [L,T]=rotatefactors(c);
>> disp('Матриця головних компонент після обертання')
>> fprintf('%12.8f %12.8f %12.8f %12.8f %12.8f\n',L')
>> disp('Матриця обертання')
>> fprintf('%12.8f %12.8f %12.8f %12.8f %12.8f\n',T')
```

Матриця коефіцієнтів головних компонент

-0.50917967	-0.02558183	0.83897281	0.18868206	0.02460368
-0.01937805	0.70939095	0.00549152	0.11020628	-0.69585443
-0.68333015	-0.02381906	-0.26100434	-0.67189001	-0.11372383
-0.02579333	0.70393032	0.01274646	-0.12200669	0.69911995
-0.52225967	-0.00498084	-0.47729313	0.69708811	0.11610105

Матриця головних компонент після обертання

-0.00000001	0.00000003	1.00000000	-0.00000000	-0.00000001
0.00000003	1.00000000	-0.00000003	-0.00000014	0.00000401
-1.00000000	0.00000003	-0.00000001	0.00000010	0.00000013
0.00000013	-0.00000401	0.00000001	0.00000031	1.00000000
0.00000010	0.00000014	0.00000000	1.00000000	-0.00000031

Матриця обертання

0.68333010	-0.01937805	-0.50917966	-0.52225975	-0.02579333
0.02381917	0.70938812	-0.02558184	-0.00498073	0.70393316
0.26100428	0.00549143	0.83897281	-0.47729315	0.01274659
0.67189007	0.11020686	0.18868206	0.69708799	-0.12200655
0.11372392	-0.69585722	0.02460371	0.11610135	0.69911711

У цьому прикладі спочатку функція **princomp** виділила матрицю головних факторів. Потім розглянута функція зробила обертання цієї матриці з метою максимізації за методом *'varimax'*. Як бачимо, додаткових обертань майже не знадобилося: матриця головних компонент відразу була близькою до одиничної з точністю до перестановки стовпців і відбиттів. Це свідчить про те, що функція **princomp** досить добре виділила головні фактори.

## **FACTORAN** Загальний факторний аналіз

Знаходяться максимально правдоподібні оцінки загальних факторів у моделі загального факторного аналізу.

*Синтаксис:*

```
lambda = factoran(X,m)  
[lambda,psi] = factoran(X,m)  
[lambda,psi,T] = factoran(X,m)  
[lambda,psi,T ,stats] = factoran(X,m)  
[lambda,psi,T ,stats,F] = factoran(X,m)  
[...] = factoran(...,'param1',value1,'param2',value2,...)
```

*Опис:*

Функція **lambda = factoran(X,m)** призначена для розрахунку матриці навантажень *lambda* методом максимальної правдоподібності для  $d$  факторів. Вхідний параметр  $X$  є матрицею спостережень багатовимірної випадкової величини. Рядки  $X$  відповідають спостереженням, стовпці  $X$  – ознакам багатовимірної випадкової величини. Розмірність  $X$  дорівнює  $n \times m$ , де  $n$  – число спостережень,  $m$  – кількість ознак багатовимірної випадкової величини. Вхідний параметр  $m$  має бути заданий як скаляр. Розмірність матриці навантажень *lambda* дорівнює  $d \times m$ . Елемент  $lambda(i, j)$  є навантаженням  $j$ -го фактора на  $i$ -у ознаку багатовимірної випадкової величини. За замовчуванням функція **factoran** викликає функцію **rotatefactors** для обертання оцінок навантажень факторів з опцією *'varimax'*, що дозволяє спростити стовпці матриці навантажень,

наближаючи їх значення до 1 або до 0 (по можливості найближче до цих граничних значень).

Функція **[lambda,psi] = factoran(X,m)** повертає параметр *psi* – вектор оцінок дисперсій специфічних факторів, розрахований методом максимальної правдоподібності. Число елементів вектора *psi* дорівнює *d*.

Функція **[lambda,psi,T] = factoran(X,m)** повертає матрицю обертання навантажень *T*. Розмірність *T* дорівнює  $m \times m$ .

Функція **[lambda,psi,T,stats] = factoran(X,m)** повертає структуру *stats*, що дозволяє перевірити нульову гіпотезу  $H_0$ , яка полягає в тому, що число факторів дорівнює *m*.

В табл. 15.1 перелічені поля структури *stats*. Значення полів *stats.chisq* і *stats.p* будуть розраховані у випадку, якщо *stats.dfe* > 0 і всі елементи вектора *psi* – позитивні. Якщо вхідний параметр *X* буде заданою як коваріаційна матриця, то для розрахунку *stats.chisq* і *stats.p* необхідно визначити значення додаткового вхідного параметра '*nobs*'.

Таблиця 15.1

### Поля структури *stats*

Поля	Значення структури <i>stats</i>
<i>stats.loglike</i>	Значення логарифма функції максимальної правдоподібності
<i>stats.dfe</i>	Число ступенів свободи залишків, виражене як $((d-m)^2 - (d+m))/2$
<i>stats.chisq</i>	Значення статистики хі-квадрат для перевірки нульової гіпотези
<i>stats.p</i>	Правобічний рівень значимості для перевірки нуль-гіпотези

Функція **[lambda,psi,T,stats,F] = factoran(X,m)** повертає розраховані за моделлю (1) значення багатовимірної випадкової величини *F*. Обчислення значень багатовимірної випадкової величини провадиться для *m* загальних факторів за розрахованою матрицею навантажень *lambda*. *F* є матрицею із розмірністю  $n \times m$ . Рядки матриці *F* відповідають спостереженням, а стовпці – ознакам багатовимірної випадкової величини. *F* не може бути розрахована, якщо вхідний параметр *X* заданий як коваріаційна матриця.

У функції `[...] = factoran(...,'param1',value1,'param2',value2,...)` керування роботою алгоритму здійснюється за допомогою додаткових вхідних параметрів `'param1'`, `'param2'`, ... . Додаткові вхідні параметри задаються у вигляді пари "назва параметра, його значення". Можливі значення цих параметрів наведені в табл. 15.2:

Таблиця 15.2

**Можливі значення параметрів `'param'`**

Параметр <code>'param'</code>	Призначення <code>'param'</code> і його можливі значення ( <i>value</i> )	
<code>'xtype'</code>	Тип аргументу $X$ . Можливі значення:	
	<code>'data'</code>	$X$ задається як матриця спостережень значень багатовимірної випадкової величини (за замовчуванням)
	<code>'covariance'</code>	$X$ задана як позитивно визначена коваріаційна або кореляційна матриця
<code>'scores'</code>	Метод розрахунку значень багатовимірної випадкової величини $F$ за моделлю (1). Параметр <code>'scores'</code> ігнорується, якщо $X$ задана як коваріаційна або кореляційна матриця. Можливі значення <code>'scores'</code> :	
	<code>'wls'</code>	Зважений метод найменших квадратів за фіксованих $F$
	<code>'Bartlett'</code>	(за замовчуванням)

Продовження табл. 15.2

Параметр <code>'param'</code>	Призначення <code>'param'</code> і його можливі значення ( <i>value</i> )	
<code>'scores'</code>	<code>'regression'</code> <code>'Thomson'</code>	Мінімальна середньоквадратична похибка (гребінна оцінка), що еквівалентно ридж-регресії
<code>'start'</code>	Початкове значення дисперсій специфічних факторів $\psi_i$ для процедури оптимізації методом максимальної правдоподібності. Можливі значення <code>'start'</code> :	
	<code>'random'</code>	Вектор дисперсії генерується за безперервним законом рівної ймовірності в інтервалі можливих значень $[0,1]$ . Число елементів вектора дисперсій дорівнює $d$
	<code>'Rsquared'</code>	Початковий вектор дисперсій розраховується за вираженням $diag(inv(corrcoef(X)))$ . Значення за замовчуванням
	Ціле позитивне значення	Вектор оцінок дисперсій генерується за безперервним законом рівної ймовірності в інтервалі можливих значень $[0,1]$ , <code>'start' = 'random'</code> . Розрахунок вихідних параметрів виконується заданим цілим числом кілька разів. Функція <b>factoran</b> повертає результат розрахунку, відповідний до максимального значення функції правдоподібності
	Matrix	Початкове значення дисперсій специфічних факторів

		задається у вигляді матриці. Стівпці матриці відповідають векторам початкових наближень для дисперсій специфічних факторів. Кількість рядків матриці початкових наближень повинно дорівнювати $d$
'rotate'	Метод, використовуваний для обертання $\lambda$ і $F$ . Його значення – це те ж саме, що й параметр 'method' для функції <b>rotatefactors</b> . Можливі значення 'rotate':	
	'none'	Обертання відсутнє
	'orthomax'	Ортогональне обертання. Критерієм оптимізації матриці навантажень за умови ортогонального обертання є максимум дисперсії, що пояснюється простими факторами. Наведені нижче додаткові вхідні параметри 'coeffom', 'normalize', 'reltol', 'maxit' призначені для керування процедурою ортогонального обертання
	'varimax'	Метод ортогонального обертання варімакс. Значення 'rotate' за замовчуванням. Обертання методом варімакс дозволяє спростити стівпці матриці $\lambda$ , наближаючи значення елементів до 1 або 0. Наведені нижче додаткові вхідні параметри 'normalize', 'reltol', 'maxit' дозволяють управляти процедурою обертання методом варімакс

Продовження 2 табл. 15.2

Параметр 'param'	Призначення 'param' і його можливі значення (value)	
'rotate'	'procrustes'	Косокутне або ортогональне обертання для одержання максимальної відповідності матриці навантажень до цільової матриці. Критерієм відповідності є сума квадратів відхилень коефіцієнтів цільової і розрахованої матриць навантажень. За замовчуванням використовується косокутне обертання. Наведені нижче додаткові вхідні параметри 'typeprocr', 'targetprocr' призначені для задання виду обертання й цільової матриці навантажень
	'promax'	Косокутне обертання для одержання матриці навантажень факторів наближеної до цільової матриці. Цільова матриця навантажень розраховується функцією <b>factoran</b> з параметром 'rotate' рівним 'orthomax'. Додатковий вхідний параметр 'powerpm' задає ступінь цільової матриці навантажень. Оскільки <b>factoran</b> з параметром 'rotate' = 'orthomax' для розрахунку цільової матриці навантажень викликається з основної функції, то допустимо використовувати параметри 'coeffom', 'normalize', 'reltol', 'maxit'
	Function	Показчик на функцію обертання заданий в наступній формі: $[B,T] = myrotation(A,...)$ , де $A$ – вихідна матриця навантажень факторів з розмірністю $d \times m$ ; $B$ – матриця

	навантажень після обертання з розмірністю $d \times m$ ; $T$ – матриця обертання з розмірністю $m \times m$ . Для передачі вхідних аргументів функції обертання використовується додатковий параметр <i>'userargs'</i>
<i>'coeffom'</i>	Коефіцієнт ортогонального обертання. У літературі часто позначається як $\gamma$ . Використовується з параметром <i>'rotate'='orthomax'</i> . Повинне бути $\gamma \in [0; 1]$ Значення <i>'coeffom' = 0</i> відповідає методу ортогонального обертання кватримакс, <i>'coeffom' = 1</i> – методу ортогонального обертання варімакс. Значення за замовчуванням <i>'coeffom' = 1</i> . Метод обертання кватримакс ставить метою спрощення рядків факторної матриці, зводячи значення в рядках до 1 або 0. Використання варімакс дозволяє спростити стовпці факторної матриці, зводячи значення елементів стовпців до 1 або 0. Проміжні значення <i>'coeffom'</i> будуть відповідати ортогональному обертанню еквімакс. Метод еквімакс використовується для одночасного спрощення рядків і стовпців матриці навантажень
<i>'normalize'</i>	Нормалізація по рядках матриці навантажень. Для <i>'normalize' = 1</i> буде виконуватися порядкова нормалізація значень елементів матриці навантажень. Для <i>'normalize' = 0</i> нормалізація не виконується. <i>'normalize'</i> використовується для <i>'rotate' = 'orthomax'</i> , або <i>'rotate' = 'varimax'</i> . Значення за замовчуванням <i>'normalize' = 1</i>

Продовження 3 табл. 15.2

Параметр <i>'param'</i>	Призначення <i>'param'</i> і його можливі значення ( <i>value</i> )
<i>'reitol'</i>	Граничне значення похибки сходження алгоритму для методів ортогонального обертання <i>'orthomax'</i> і <i>'varimax'</i> . Значення за замовчуванням <i>sqrt(eps)</i>
<i>'maxit'</i>	Максимальне число ітерацій за ортогонального обертання методами <i>'orthomax'</i> і <i>'varimax'</i> . Значення за замовчуванням 250
<i>'targetprocr'</i>	Цільова матриця навантажень. Використовується для <i>'rotate' = 'procrustes'</i> . Значення за замовчуванням відсутні
<i>'typeprocr'</i>	Вид обертання матриці навантажень для <i>'rotate' = 'procrustes'</i> . Для <i>'typeprocr' = 'oblique'</i> використовується косокутне обертання. Якщо <i>'typeprocr' = 'orthogonal'</i> використовується ортогональне обертання. Значення за замовчуванням <i>'typeprocr' = 'oblique'</i>
<i>'powerpm'</i>	Ступінь цільової матриці. Використовується для <i>'rotate' = 'promax'</i> . Значення <i>'powerpm'</i> повинне бути більше або рівне 1. За замовчуванням <i>'powerpm' = 4</i>
<i>'userargs'</i>	Передача додаткових вхідних параметрів функції обертання, заданої користувачем, <i>'rotate'=Function</i> . Список переданих параметрів формується за правилом: першим параметром <i>Function</i> є матриця $A$ , далі йде перелік змінних і констант функції <b>factoran</b> після вхідного параметра <i>'userargs'</i>
<i>'nobs'</i>	Число спостережень багатовимірної випадкової величини. Вхідний

	параметр <i>'nobs'</i> використовується, якщо $X$ задана як кореляційна або коваріаційна матриця. Якщо $X$ задана як матриця спостережень багатовимірної випадкової величини, то <i>'nobs'</i> ігнорується. Величина <i>'nobs'</i> необхідна для розрахунку рівня значимості $p$ перевірки нульової гіпотези. Значення за замовчуванням відсутнє
<i>'delta'</i>	Мінімальні значення дисперсій специфічних факторів $\psi_i$ , використовувани в процесі оптимізації методом максимальної правдоподібності. Значення за замовчуванням 0,005
<i>'optimopts'</i>	Структура, що визначає параметри процедури оптимізації методом максимальної правдоподібності. Значення вхідного параметра <i>'optimopts'</i> формується за допомогою структури <i>optimset</i> . Функція <b>factoran</b> використовує наступні поля <i>optimset</i> : <i>'Display'</i> , <i>'Maxfunevals'</i> , <i>'Maxiter'</i> , <i>'Tolfun'</i> , <i>'Tolx'</i> . Значення полів структури <i>optimset</i> за замовчуванням: <i>'Display'</i> = <i>'notify'</i> ; <i>'Maxfunevals'</i> =100*d; <i>'Maxiter'</i> =400; <i>'Tolfun'</i> =1e-8; <i>'Tolx'</i> =1e-8

*Примітка:* в **Statistics Toolbox 4.1** структура, що визначає параметри процедури оптимізації, формується за допомогою функції **statset**.

Команда **statset('factoran')** дозволяє одержати список полів структури, що використані функцією **factoran**, і їх значення за замовчуванням:

```
>> statset('factoran')
ans =
    Display: 'notify'
    Maxfunevals: 400
    Maxiter: 100
    Tolbnd: []
    Tolfun: 1.0000e-008
    Tolx: 1.0000e-008
```

Склад полів структури параметрів оптимізації алгоритму **factoran** і їх призначення наведено в табл. 15.3:

Таблиця 15.3

### Призначення полів структури *'optimopts'*

Поле структури (параметр функції <i>statset</i> )	Призначення	Припустимі значення
<i>Display</i>	Відображення проміжної інформації про роботу алгоритму оптимізації функції	<i>'off'</i> – проміжна інформація не відображається; <i>'final'</i> – відображається інформація на останньому кроці оптимізації; <i>'notify'</i> – проміжна інформація відображається лише у випадку відсутності збіжності за роботи алгоритма. Значення за замовчуванням.
<i>Maxfunevals</i>	Максимальна кількість	Позитивне ціле число

	викликів цільової функції	
<i>Maxiter</i>	Максимальна кількість ітерацій	Позитивне ціле число
<i>Tolbnd</i>	Обмеження на можливі значення оптимізованого параметра	Позитивне дійсне число
<i>Tolfun</i>	Значення вихідного параметра функції за яким припиняється робота алгоритма оптимізації	Позитивне дійсне число
<i>Tolx</i>	Значення параметрів функції за яким припиняється робота алгоритма оптимізації	Позитивне дійсне число

*Примітки:*

1. Ознаки багатомірної випадкової величини  $X$ , заданої як матриця спостережень, повинні бути лінійно незалежні, тобто ранг матриці коваріацій повинен дорівнювати числу стовпців або рядків  $cov(X)$ . За дотримання цієї умови можливе використання методу максимальної правдоподібності. Перед визначенням матриці навантажень  $lambda$  функцією **factoran** розраховується кореляційна матриця щодо  $X$  незалежно від того, задана  $X$  як матриця спостережень або матриця коваріацій.

Перед розрахунком матриці навантажень функція **factoran** стандартизує значення рядків матриці спостережень  $X$ . Після стандартизації середнє спостережень  $X$  буде дорівнювати нулю, а дисперсія – одиниці. Стандартизація  $X$  не впливає на оцінку матриці навантажень, оскільки метод максимальної правдоподібності в розглянутій моделі простого факторного аналізу інваріантний до зазначеної операції. Після стандартизації оцінка кореляційної матриці для вихідних даних  $X$  буде визначатися з вираження  $(lambda*lambda)'+diag(psi)$ . Наведене вираження не поширюється на випадок косокутного обертання матриці навантажень.

2. Якщо елементи вектора  $psi$  приблизно дорівнюють значенню параметра  $'delta'$ , тобто приблизно дорівнюють нулю, то інтерпретація результатів розрахунку може бути неоднозначною. Така ситуація відома як випадок Хейвуда. Зокрема, можлива наявність одного або декількох локальних максимумів функції максимальної правдоподібності, кожний з яких відповідає різній оцінці матриці навантажень і дисперсій специфічних факторів. Зокрема випадок Хейвуда може виникнути в процесі використання великого або малого числа простих факторів.

3. Якщо явно не задається  $'rotate' = 'none'$ , функція **factoran** використовує той або інший алгоритм обертання матриць  $lambda$  і  $F$ . Вихідна матриця  $T$  використовується в процедурі обертання навантажень факторів згідно з вираженням  $lambda = lambda_0 * T$ , де  $lambda_0$  – початкова матриця навантажень, що отримана методом максимальної правдоподібності.  $T$  є ортогональною матрицею для ортогонального обертання. У випадку  $'rotate' = 'none'$   $T$  є тотожною матрицею. Матриця зворотна до  $T$  називається матрицею обертання первинних осей. За ортогонального обертання ці дві матриці взаємно транспоновані.

Матриця  $F$  обчислюється за формулою  $F = F_0 * inv(T')$ , де  $F_0$  – матриця розрахованих значень багатовимірної випадкової величини до обертання. Оцінка матриці коваріацій  $F$  визначається як  $inv(T' * T)$ . Коваріаційна матриця  $F$  за ортогонального обертання або за відсутності обертання є одиничною матрицею. Обертання матриць  $lambda$  і  $F$  призначене для одержання більш простих в інтерпретації структур матриць, після розрахунку їх оцінок методом максимальної правдоподібності.

*Приклади:*

1. Розрахунок матриці навантажень  $lambda$  методом максимальної правдоподібності для 3 загальних факторів. Вхідний параметр  $X$  є матрицею спостережень 10-ти вимірної випадкової величини, розподіленої за стандартизованим нормальним законом.

```
>> X=normrnd(0,1,10,6)
```

```
% Одержали матрицю спостережень:
```

```
X =
-0.4326 -0.1867 0.2944 -0.3999 -1.6041 -1.0106
-1.6656 0.7258 -1.3362 0.6900 0.2573 0.6145
0.1253 -0.5883 0.7143 0.8156 -1.0565 0.5077
0.2877 2.1832 1.6236 0.7119 1.4151 1.6924
-1.1465 -0.1364 -0.6918 1.2902 -0.8051 0.5913
1.1909 0.1139 0.8580 0.6686 0.5287 -0.6436
1.1892 1.0668 1.2540 1.1908 0.2193 0.3803
-0.0376 0.0593 -1.5937 -1.2025 -0.9219 -1.0091
0.3273 -0.0956 -1.4410 -0.0198 -2.1707 -0.0195
0.1746 -0.8323 0.5711 -0.1567 -0.0592 -0.0482
```

```
>> [lambda,psi,T,stats,F] = factoran(X,3)
```

```
% Матриця навантажень
lambda =
-0.1391 0.0841 0.7006
0.2439 0.9628 0.0923
0.4104 0.1550 0.8958
0.7336 0.1451 0.1232
0.3828 0.5647 0.3701
0.8706 0.4140 -0.0868

% оцінки дисперсій
специфічних факторів
psi =
0.4827
0.0050
0.0050
0.4256
0.3976

0.0631
% Матриця обертання
T =
0.4246 0.6898 0.5864
0.8960 -0.2273 -0.3815
0.1299 -0.6874 0.7146
```

Значення полів *stats* не розраховані, тому що число ступенів свободи *dfe* дорівнює нулю

```
stats =
loglike: -0.0253
dfe: 0
```

Значення трьох факторів, що розраховані за отриманою матрицею навантажень *lambda*:

```
F =
-1.2493 -0.2642 0.8559
0.3514 0.6436 -1.5513
1.1577 -1.2913 0.3383
1.2121 1.9330 0.6059
0.9647 -0.5821 -1.0149
-0.7890 -0.0457 1.1606
0.1069 0.8616 0.9602
-1.8483 0.3402 -0.7297
-0.3273 -0.1904 -1.1884
0.4211 -1.4046 0.5635
```

2. У прикладі генерується вибірка з 5-вимірним нормальним розподілом, у якого сильно корелюють між собою 1-а, 3-я й 5-а координати, а також 2-а й 4-а, але кореляція між цими групами змінних – слабка. Перевіряється 0-гіпотеза про те, що з п'яти координат лише дві незалежні.

```
>> n=500; % число точок
>> k=2; % число незалежних аргументів
>> alpha=0.05; % рівень значущості
```

```

>> mu=[1 2 3 -1 0]; % середні
>> s=[ 1 0.4 0.8 0.4 0.7;...
      0.4 1 0.4 0.6 0.4;...
      0.8 0.4 1 0.4 0.7;...
      0.4 0.6 0.4 1 0.4;...
      0.7 0.4 0.7 0.4 1]; % коваріаційна матриця
>> X=mvnrnd(mu,s,n); % багатовимірна нормальна вибірка
>> [Lambda,Psi,T,stats]=factoran(X,2,'scores','regression','maxit',1000);
>> fprintf(' 0-гіпотеза: серед %d параметрів %d незалежних:\n',length(mu),k)
>> if stats.p>=alpha,
    fprintf([' P-значення %10.8f більше або рівне рівню значущості ' ...
            '%5.2f =>\nухвалюємо 0-гіпотезу.\n'],stats.p,alpha)
else
    fprintf([' P-значення %10.8f менше рівня значущості ' ...
            '%5.2f =>\nвідхиляємо 0-гіпотезу.\n'],stats.p,alpha)
end
0-гіпотеза: серед 5 параметрів 2 незалежних:
P-значення 0.86498874 більше або рівне рівню значущості 0.05 =>
ухвалюємо 0-гіпотезу.

```

3. Аналіз структури 5-ти вимірної випадкової величини, яка характеризує параметри автотранспортних засобів, що втримуються у файлі *carbig.mat*. Цей файл відрізняється від описаного раніше файла *carsmall.mat* лише тим, що містить параметри 406 автомобілів (*big* – великий) замість 100 (*small* – маленький). У якості ознак багатовимірної випадкової величини виступають: *Acceleration* – прискорення автотранспортного засобу, *Displacement* – об'єм циліндрів двигуна, *Horsepower* – потужність двигуна, *MPG* – питомий пробіг, *Weight* – маса автотранспортного засобу.

3.1. Виконується завантаження даних з файла *carbig* і розрахунок параметрів моделі простого факторного аналізу для двох факторів. У якості методу обертання використовується косокутне обертання, *'rotate' = 'promax'*, для одержання матриці навантажень факторів наближеної до цільової матриці навантажень. Ступінь цільової матриці, *'powerpm'*, дорівнює 4.

```

>> load carbig
>> X = [Acceleration Displacement Horsepower MPG Weight];

```

База даних *X* містить у деяких рядках нечислові дані (див. фрагмент даних в табл. 15.4):

## Фрагмент даних

Acceleration	Displacement	Horsepower	MPG	Weight
0.0085	0.3900	0.1900	0.0150	3.8500
0.0175	0.1330	0.1150	NAN	3.0900
0.0115	0.3500	0.1650	NAN	4.1420
0.0110	0.3510	0.1530	NAN	4.0340
0.0105	0.3830	0.1750	NAN	4.1660
0.0110	0.3600	0.1750	NAN	3.8500
0.0100	0.3830	0.1700	0.0150	3.5630
0.0080	0.3400	0.1600	0.0140	3.6090
0.0080	0.3020	0.1400	NAN	3.3530
0.0095	0.4000	0.1500	0.0150	3.7610

З нечисловими даними функція **factoran** не працює. Наступна команда даних видаляє з бази всі рядки, що містять нечислові дані

```
>> X = X(all(~isnan(X),2),:);
```

Розрахунки факторів  $F$ :

```
>> [Lambda,Psi,T,stats,F] = factoran(X,2,'rotate','promax','powerpm',4)
Lambda =      0.0804      loglike: -0.0030      0.7630 -1.6397
  0.0964  0.9426      0.0680      dfe: 1      0.7458 -1.6992
  0.8879 -0.1057      0.2859      chisq: 1.1544
  0.6526 -0.4076      0.0152      p: 0.2826      .....
 -0.8413  0.0059      T =      F =      -0.3121  0.4117
  1.0800  0.1472      1.0045  0.0070      0.7381 -1.1402      -1.2123  2.5957
Psi =      0.8254  1.3001      1.1048 -1.9752      -0.7148 -0.7258
  0.2184      stats =      0.7853 -1.8663      -0.5372  1.0875
                                -0.4689  1.2723
```

3.2. Розрахунок кореляційної матриці для значень  $F$ . Кореляційна матриця повинна не бути рівною  $eye(2)$  – одиничній матриці другого порядку.

```
>> inv(T'*T)
ans =      1.0000 -0.6391
      -0.6391  1.0000
```

3.3. Розрахунок кореляційної матриці для матриці вихідних даних  $X$ .

```
>> Lambda*inv(T'*T)*Lambda' + diag(Psi)
ans =
  1.0000 -0.5424 -0.6893  0.4309 -0.4167
 -0.5424  1.0000  0.8979 -0.8078  0.9328
 -0.6893  0.8979  1.0000 -0.7730  0.8647
  0.4309 -0.8078 -0.7730  1.0000 -0.8326
 -0.4167  0.9328  0.8647 -0.8326  1.0000
```

3.4. Графічне представлення результатів факторного аналізу після косокутного обертання матриці навантажень.

```
>> invt = inv(T)
invt =
    1.0000 -0.0054
   -0.6349  0.7726
>> Lambda0 = Lambda*invt
Lambda0 =
   -0.5020  0.7277
    0.9550 -0.0865
    0.9113 -0.3185
   -0.8450  0.0091
    0.9865  0.1079
>> plot(Lambda0(1,1),Lambda0(1,2),'ko',Lambda0(2,1),Lambda0(2,2), ...
'k^',Lambda0(3,1),Lambda0(3,2),'k+',Lambda0(4,1),Lambda0(4,2),'k.',...
Lambda0(5,1),Lambda0(5,2),'k>'); grid on
l>> abels = num2str((1:5)');
>> text(Lambda0(:,1)+.05,Lambda0(:,2),labels,'Color','k');
>> line([-invt(1,1) invt(1,1) NaN -invt(2,1) invt(2,1)],[-invt(1,2) invt(1,2) NaN -invt(2,2)
invt(2,2)]);
>> set(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> text(invt(:,1), invt(:,2),[' I ' ; ' II']);
>> title('\bfНеортогональне факторне рішення');
>> xlabel('Фактор 1') % мітка осі ОХ
>> ylabel('Фактор 2') % мітка осі ОУ
>> legend('1-Accel','2-Displace','3-Horse','4-MPG','5-Weight')
```

На рис. 15.5 зображені факторні навантаження після перетворення в косокутній системі координат.

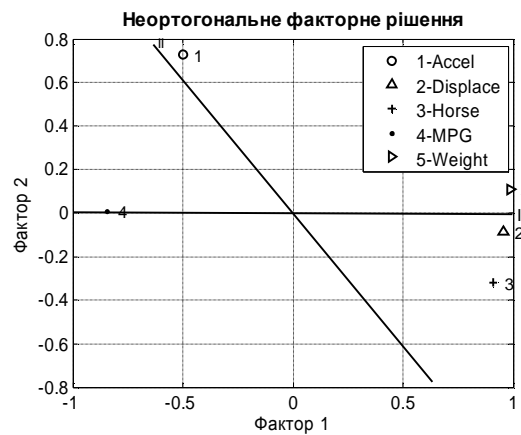


Рис. 15.5. Графічне представлення результатів факторного аналізу

4. Аналіз структури 5-ти вимірної випадкової величини для двох факторів. У якості методу обертання використовується ортогональне обертання з коефіцієнтом  $\gamma = 0.5$ . Вхідні дані задані у вигляді

коваріаційної матриці. Для розрахунку полів структури *stats.chisq* і *stats.p* задано додатковий вхідний параметр 'nobs' = 100.

```
>> load carbig
>> X = [Acceleration Displacement Horsepower MPG Weight];
>> X = X(all(~isnan(X),2),:);
>> C= cov(X)
C =
1.0e+005 *
    0.0001 -0.0016 -0.0007  0.0001 -0.0098
   -0.0016  0.1095  0.0361 -0.0066  0.8293
   -0.0007  0.0361  0.0148 -0.0023  0.2827
    0.0001 -0.0066 -0.0023  0.0006 -0.0552
   -0.0098  0.8293  0.2827 -0.0552  7.2148
>> [lambda,psi,T,stats] = factoran(C,2,'xtype','cov', 'nobs', 100)
lambda =
   -0.2432  -0.8500
    0.8773  0.3871
    0.7618  0.5930
   -0.7978  -0.2786
psi =
    0.9692  0.2129
    0.2184
    0.0804
    0.0680
T =
    0.2859
    0.0152
    0.9476  0.3195
    0.3195 -0.9476
stats =
    loglike: -0.0030
         dfe: 1
    chisq: 0.2838
         p: 0.5943
```

### **BIPLOT** Діаграма факторних навантажень і об'єктів

Будується діаграма векторів факторних навантажень і розташування експериментальних точок у просторі головних компонент.

*Синтаксис:*

```
biplot(coefs)
biplot(coefs,....,'Scores',scores)
biplot(coefs,....,'Varlabels',varlabs)
biplot(coefs,....,'Scores',scores,'Obslabels',obslabs)
biplot(coefs,....,PropertyName,Propertyvalue,...)
h = biplot(coefs,...)
```

*Опис:*

Функція **biplot(coefs)** – буде графік векторів коефіцієнтів, заданих у матриці *coefs*. Якщо в матриці *coefs* два стовпці, будується двовимірний графік, а якщо три – тривимірний. Звичайно *coefs* містить факторні навантаження за головних компонент, що створені функціями **princomp**, **pcacov** або оцінку матриці факторних навантажень, створену функцією **factoran**. Осі створеного графіка – це головні компоненти або фактори (стовпці *coefs*), а змінні (рядки *coefs*) представлені як вектори.

Функція **biplot(coefs,...,'Scores',scores)** будує й вектори, що задані в *coefs*, і самі змінні (точками), що задані в матриці *scores*. Звичайно *scores* містить головні компоненти, створені функцією **princomp**, або множини оцінок факторів, створену функцією **factoran**. Кожне спостереження (рядок в *scores*) представлено на графіку точкою.

Дана діаграма дозволяє наочно показати величину й напрямок впливу кожної змінної на перші 2 або 3 головні компоненти, і як кожне спостереження представлено через ці компоненти.

У функції **biplot** використовується наступна угода про знаки: показуються елементи з найбільшими значеннями в кожному стовпці *coefs* із числа позитивних.

Функція **biplot(coefs,...,'Varlabels',varlabs)** в аргументі *varlabs* дозволяє задати мітки для кожного вектора (змінної) у вигляді масиву символів або масиву символівних рядків.

Функція **biplot(coefs,...,'Scores',scores,'Obslabels',obslabs)** в аргументі *obslabs* дозволяє задати мітки для кожної точки (спостереження) у вигляді масиву символів або масиву з рядків символів.

Функція **biplot(coefs,...,PropertyName,PropertyValue,...)** установлює значення заданих властивостей усіх графічних об'єктів типу *line* на фігурі.

Функція **h = biplot(coefs,...)** повертає вектор-стовпець із дескрипторів усіх об'єктів на фігурі. Порядок проходження дескрипторів наступний: спочатку йдуть дескриптори, що відповідають змінним (лінії, далі маркери, потім текстові мітки). За ними йдуть дескриптори, відповідні до спостережень (спочатку маркери, якщо вони є, а далі текстові мітки). Наприкінці – лінії осей.

У *прикладі* завантажується база даних щодо автомобілів, і досліджується простір п'яти ознак – техніко-економічних характеристик автомобілів: прискорення, об'єму двигуна, потужності, питомого пробігу на 1 галон пального й ваги.

```
>> load carsmall % база даних щодо автомобілів
>> x=[Acceleration Displacement Horsepower MPG Weight]; % змінні
>> x=x(all(~isnan(x),2),:); % вилучення NAN
```

```

>> [coefs,score]=princomp(zscore(x)); % ГОЛОВНІ КОМПОНЕНТИ
>> vlabs={'Accel','Disp','HP','MPG','Wgt'};
>> biplot(coefs(:,1:3),'scores',score(:,1:3),'varlabels',vlabs);
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfГрафік коефіцієнтів і спостережень'); % заголовок
>> xlabel('Компонента 1'), ylabel('Компонента 2'), zlabel('Компонента 3')

```



Рис. 15.6. **Зображення об'єктів і факторних навантажень у просторі перших трьох компонент**

На рис. 15.6 показані експериментальні точки й вектори п'яти факторів у просторі перших трьох головних компонент. Вектори змінних "об'єм двигуна" і "потужність" дуже близькі за напрямком й величиною, що говорить про сильний зв'язок між ними.

### **BARTTEST** Тест Бартлета для оцінки розмірності простору головних компонент

Цей тест визначає кількість незалежних розмірностей (стовпців) у випадковій матриці даних.

*Синтаксис:*

```

ndim = barttest(x,alpha)
[ndim,prob,chisquare] = barttest(x,alpha)

```

*Опис:*

Функція **ndim = barttest(x,alpha)** дозволяє оцінити розмірність *ndim* простору головних компонент, потрібного для того, щоб пояснити не випадкові зміни в матриці вихідних даних *x* за заданого рівня

значущості  $alpha$ . Величина розмірності  $ndim$  визначається в результаті перевірки послідовності нульових гіпотез. Нульова гіпотеза для  $ndim = 1$  полягає в тому, що дисперсії вихідних даних за всіма головними компонентами однакові. Нульова гіпотеза для  $ndim = 2$  полягає в тому, що дисперсії вихідних даних за другою й наступними головними компонентами однакові. І так далі.

Функція **[ndim,prob,chisquare] = barttest(x,alpha)** повертає розмірність простору головних компонентів  $ndim$ , що пояснюють не випадкові зміни в матриці вхідних даних  $x$ ; вектор рівнів значимості  $prob$ , отриманий в ході перевірки послідовності нульових гіпотез про рівність  $ndim$  заданому значенню; вектор значень статистики хі-квадрат  $chisquare$ , використаних для перевірки кожної із серії нульових гіпотез.

#### Приклади:

Тест Бартлета для перевірки нульової гіпотези про величину розмірності простору головних компонент. У якості вихідних даних виступає матриця, що полягає з 3-х двовимірних випадкових величин, розподілених за багатовимірним нормальним законом з коваріаційною матрицею  $\begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$  і нульовим математичним очікуванням. Об'єм вибірки 20 елементів. Функція повертає розмірність простору головних компонентів, вектор рівнів значимості, отриманий в ході перевірки серії нульових гіпотез, вектор значень статистики хі-квадрат, використаної для перевірки кожної із серії нульових гіпотез. Перевірка гіпотез провадиться для рівня значимості 0.01.

```
>> x=mvnrnd([0 0],[1 0.8; 0.8 1],20);
```

```
>> x(:,3:4)=mvnrnd([0 0],[1 0.8; 0.8 1],20);
```

```
>> x(:,5:6)=mvnrnd([0 0],[1 0.8; 0.8 1],20)
```

```
x =
```

```
 0.1167  0.6447  0.2427  1.4770 -0.9065 -0.7262
-0.4311  0.0007  1.7214  0.4105 -0.0097 -0.1227
-0.0704 -0.1224 -0.4345 -0.4935  1.2083  1.1354
 0.4427  1.4141  0.7728  0.4220 -0.2587  0.2724
 0.3722  0.3298 -0.6067  0.1932 -0.3014  0.0916
 2.2534  3.5342 -1.1585 -0.8332  0.7894  0.6681
 0.7223  0.4815  1.0296  0.8418  0.3495 -0.1610
 0.2414 -0.1126  0.3999  0.1246  0.2743 -0.1477
 0.3179  0.3356 -1.7791 -1.9980  0.0029  1.3347
 1.8590  1.9471  1.1533  0.7156 -0.7833 -0.9621
```

```

0.1785  0.0511  -1.3143  -0.6806  -0.5184  -1.2498
-0.6566 -0.9563  0.4541  0.5938  -1.2218  -2.6552
0.5669  0.5790  -1.5175  -1.4817  0.5771  0.2657
-0.4907 -0.0424  1.3700  1.3030  0.4961  1.5407
-0.1609 -0.8036  -1.6280  -1.3797  -1.6784  -2.1194
0.2913  1.3834  0.0836  0.7831  0.5393  -0.0553
0.3500  -0.1302  -0.6968  -0.1712  0.7985  0.6930
-0.9587 -0.9769  0.9665  0.7374  -1.6303  -0.8945
1.4769  0.4442  -0.5637  -0.1948  -0.5208  -1.0061
1.0455  0.7801  -0.1870  -0.9878  -0.6465  -0.8061
>> [ndim,prob,chisquare]=barttest(x,0.01)
ndim =          | chisquare =
   3           | 78.7965
prob =          | 63.3217
 0.0000        | 41.7510
 0.0000        | 2.6267
 0.0000        | 1.7525
 0.7573
 0.4163

```

Отже, розмірність скороченого факторного простору дорівнює 3, тобто для адекватного опису даних досить 3-х головних компонент.

## 16. Функції аналізу багатовимірних випадкових величин

Під цим заголовком в **Statistics Toolbox** зібрані функції канонічних кореляцій, дискримінантного, прокрустова аналізу й багатовимірного шкалювання:

<b>CANONCORR</b>	Канонічний кореляційний аналіз .....	327
<b>CLASSIFY</b>	Лінійний дискримінантний аналіз .....	330
<b>CMDSCALE</b>	Класичне багатовимірне шкалювання .....	333
<b>MDSCALE</b>	Метричне й неметричне багатовимірне шкалювання .....	336
<b>PROCRUSTES</b>	Прокрустів аналіз .....	340
<b>MAHAL</b>	Відстані Махаланобіса .....	342

### **CANONCORR** Канонічний кореляційний аналіз

За допомогою канонічного кореляційного аналізу визначають коефіцієнт кореляції між двома групами ознак  $X$ ,  $Y$ . Для цього складають дві лінійні комбінації  $U$ ,  $V$  зі змінних групи  $X$  і зі змінних групи  $Y$ . Коефіцієнти лінійних комбінацій  $U$ ,  $V$  добирають так, щоб коефіцієнт кореляції між цими “канонічними змінними” був максимальним. Визначивши першу пару канонічних змінних, знаходять другу пару, що в деякій метриці ортогональна до першої. Таким чином визначають всі найважливіші кореляційні зв'язки між двома групами ознак.

*Синтаксис:*

**[A,B] = canoncorr(X,Y)**

**[A,B,r] = canoncorr(X,Y)**

**[A,B,r,U,V] = canoncorr(X,Y)**

**[A,B,r,U,V,stats] = canoncorr(X,Y)**

Опис:

Функція  $[A,B] = \text{canoncorr}(X,Y)$  обчислює вибіркові канонічні коефіцієнти для матриць даних  $X$  (розміром  $n \times d_1$ ) і  $Y$  (розміром  $n \times d_2$ ). Аргументи  $X$  і  $Y$  повинні мати однакову кількість спостережень (рядків), але можуть мати різну кількість змінних (стовпців). Канонічні коефіцієнти  $A$ ,  $B$  – параметри, що повертаються, – будуть матрицями розмірів відповідно  $d_1 \times d$  і  $d_2 \times d$ , де  $d = \min(\text{rank}(X), \text{rank}(Y))$ . Стовпці матриць  $A$  і  $B$  з номером  $j$  містять канонічні коефіцієнти (тобто коефіцієнти лінійних комбінацій вихідних ознак  $X$  і  $Y$  відповідно) для  $j$ -ї пари канонічних змінних. Стовпці матриць  $A$  і  $B$  масштабуються так, щоб кореляційні матриці канонічних змінних стали одиничними. Якщо  $X$  або  $Y$  мають неповний ранг, функція **canoncorr** видає про це попередження й повертає нулі в рядках  $A$  або  $B$ , відповідних до залежних стовпців  $X$  або  $Y$ .

Функція  $[A,B,r] = \text{canoncorr}(X,Y)$  у результативному параметрі  $r$  повертає вектор розміру  $1 \times d$ , що містить вибіркові канонічні кореляції. Кожний  $j$ -й елемент  $r$  – це кореляція між  $j$ -мі стовпцями  $U$  і  $V$ .

Функція  $[A,B,r,U,V] = \text{canoncorr}(X,Y)$  у результативних параметрах  $U$  і  $V$  повертає канонічні змінні для  $X$  і  $Y$  відповідно. Це – матриці розміром  $n \times d$ , які обчислюються так:

```
>> U=(X-repmat(mean(X),n,1))*A;  
>> V=(Y-repmat(mean(Y),n,1))*B;
```

Функція **repmat** як би "розмножує" дану матрицю до заданих розмірів.

Функція  $[A,B,r,U,V,stats] = \text{canoncorr}(X,Y)$  у результативному параметрі  $stats$  повертає структуру з інформацією щодо послідовності  $d$  0-гіпотез  $H_{0k}$  про те, що всі кореляції з  $(k + 1)$ -ї по  $d$ -ю дорівнюють нулю. У структурі  $stats$  сім полів, кожне з яких є вектором  $1 \times d$  з елементами, що відповідають різним  $k$ . Зміст полів  $stats$  наступний:

*Wilks* –  $\lambda$ -статистика Уилкса (Wilks);

*chisq* – апроксимація Бартлета  $\chi^2$ -статистики для  $H_0^k$  з модифікацією Лоулі (Lawley);

*pchisq* – правобічний рівень значимості для *chisq*;

*F* – апроксимація Рао (Rao) F-статистики для  $H_0^k$ ;

*pf* – правобічний рівень значимості для *F*;

*df*<sub>1</sub> – число ступенів свободи для  $\chi^2$ -статистики, воно ж число ступенів свободи чисельника для F-статистики;

*df*<sub>2</sub> – число ступенів свободи знаменника для F-статистики.

*Приклад:*

У прикладі генерується вибірка 5-вимірного нормального розподілу, у якого сильно корелюють між собою 1-а, 3-а й 5-а координати, а також 2-а й 4-а, а кореляція між цими групами змінних – слабка. Далі перші 3 стовпця виділяються в групу *X*, а останні два – в *Y*, і провадиться канонічний кореляційний аналіз цих матриць. У кожному з матриць входить, принаймні, один стовпець, що сильно корелює зі стовпцем іншої матриці, тому кореляція між канонічними змінними буде значущою.

```
>> n=1000; % число точок
>> mu=[1 2 3 -1 0]; % середні
>> s=[ 2 0.01 1.9 0.02 1.4;...
      0.01 1 0.01 0.95 0.04;...
      1.9 0.01 3 0.05 2.2;...
      0.02 0.95 0.05 1 0.02;...
      1.4 0.04 2.2 0.02 2]; % коваріаційна матриця
>> XY=mvnrnd(mu,s,n); % багатовимірна нормальна вибірка
>> X=XY(:,1:3); % перші 3 стовпці
>> Y=XY(:,4:5); % останні 2 стовпці
>> [A,B,r,U,V,stats]=canoncorr(X,Y); % канонічний кореляційний аналіз
>> disp(' N   r')
>> fprintf('%3.0f %12.10fn',[1 2;r])
>> ku=1; % номер координати U
>> kv=1; % номер координати V
>> plot(U(:,ku),V(:,kv),'k.') % графік
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title(['\bфКореляція між стовпцями \rm\bфU\rm_' num2str(ku) ...
        '\bф \rm\bфV\rm_' num2str(kv)]);
>> xlabel(['\bфU\rm_' num2str(ku)]) % мітка осі OX
>> ylabel(['\bфV\rm_' num2str(kv)]) % мітка осі OY
>> grid on, box on
 N   r
 1  0.9567725960
 2  0.8973305893
```

На рис. 16.1 показані експериментальні точки в системі координат 1-ї пари канонічних змінних.

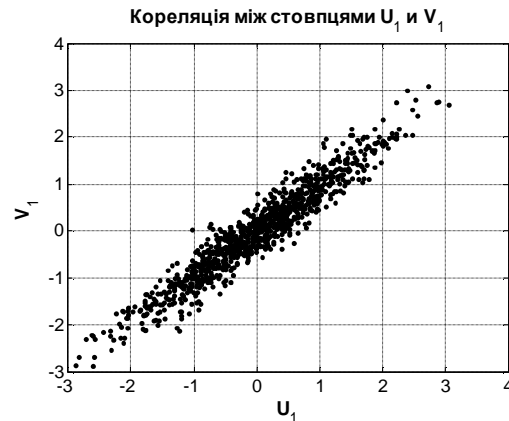


Рис. 16.1. Кореляція між канонічними змінними

### **CLASSIFY** Лінійний дискримінантний аналіз

Інша назва методу – автоматична діагностика, розпізнавання образів з вчителем. Розподіляє спостереження за відомими групами за принципом подібності.

*Синтаксис:*

```
class = classify(sample,training,group)  
class = classify(sample,training,group,type)  
class = classify(sample,training,group,type,prior)  
[class,err] = classify(...)  
[class,err,posterior,logp] = classify(...)
```

*Опис:*

Функція **class = classify(sample,training,group)** класифікує кожний рядок даних у матриці *sample* в одну із груп відповідно відомим навчальним вибіркам *training*. Матриці *sample* і *training* повинні мати однакову кількість стовпців (змінних). Аргумент *group* – це групуюча змінна для *training*. Унікальні значення *group* задають групи, а кожний елемент визначає номер групи, у яку попадає відповідний елемент *training*. Аргумент *group* може бути вектором, масивом символів або масивом символівних рядків. Кількість елементів в *group* повинна збігатися із числом рядків в *training*. Функція *classify* сприймає нечислові

значення NaN або порожні рядки в *group* як відсутні значення, і ігнорує відповідні рядки *training*. Результативний параметр *class* показує, у яку групу попадає кожний рядок *sample*. Це вектор такого ж типу, як і *group*, а число його елементів дорівнює числу рядків *sample*.

Функція **class = classify(sample,training,group,type)** в аргументі *type* дозволяє задати тип дискримінантної функції. Можливі значення цього параметра:

*'linear'* – використовує багатовимірний нормальний розподіл для кожної групи з однаковими оцінками дисперсій (за замовчуванням);

*'diaglinear'* – те ж саме, що й *'linear'*, за винятком того, що дисперсії можуть бути різними, але коваріаційна матриця залишається діагональною;

*'quadratic'* – використовує багатовимірний нормальний розподіл з оцінками дисперсій, однаковими для всіх груп;

*'diagquadratic'* – те ж саме, що й *'quadratic'*, за винятком того, що дисперсії можуть бути різними, але коваріаційна матриця залишається діагональною;

*'mahalanobis'* – використовуються відстані Махаланобіса за недіагональною матрицею коваріацій.

Функція **class = classify(sample,training,group,type,prior)** в аргументі *prior* дозволяє задати апіорні ймовірності для груп одним із трьох способів.

Аргумент *prior* може бути:

числовим вектором такої ж довжини, як і число унікальних елементів в *group*. Якщо аргумент *group* – числовий вектор, то порядок елементів в *prior* повинен відповідати розсортованим у порядку зростання значенням *group*. Якщо ж *group* містить рядки, то порядок елементів в *prior* має відповідати порядку входження елементів в *group*;

скалярною структурою з полями:

*prob* – числовий вектор;

*group* – такого ж типу, як і аргумент *group*, але містить лише унікальні значення в тому порядку, у якому впливають імовірності в *prob*. Тут можуть також бути значення, яких немає в аргументі *group*. Це може бути корисно, якщо аргумент *training* – підмножина деякої іншої, більш великої множини. У цьому випадку потрібно просто задати відповідні елементи *prob* нульовими;

рядком *'empirical'*, який вказує, що функція *classify* повинна оцінити апіорну ймовірність кожної групи за відносною частотою появи її елементів в *training*.

За замовчуванням аргумент *prior* задається як числовий вектор з однаковими значеннями ймовірностей, тобто використовується рівномірний розподіл. Цей параметр не повинен використовуватися при дискримінантному аналізі з відстанями Махаланобіса, інакше це викличе похибку обчислень.

Функція **[class,err] = classify(...)** у результативному параметрі *err* повертає оцінку похибки неправильної класифікації: відсоток рядків *training*, які були неправильно класифіковані.

Функція **[class,err,posterior] = classify(...)** у результативному параметрі *posterior* повертає матрицю оцінок апостеріорних ймовірностей того, що *j*-я навчальна група була джерелом *i*-го спостереження. Цей параметр не обчислюється за використання відстаней Махаланобіса.

Функція **[class,err,posterior,logp] = classify(...)** у вихідному параметрі *logp* повертає вектор, що містить оцінки логарифмів безумовної щільності розподілу спостережень (рядків *sample*). Цей параметр не обчислюється за використання відстаней Махаланобіса.

*Приклад:*

У прикладі генерується вибірка 2-вимірною рівномірною розподілу з незалежними координатами. Масив *training* формується як регулярна квадратна сітка, а номери груп відповідають чвертям квадрата. За цими даними проводиться розподіл точок за групами.

```

>> n=500; % кількість точок
>> sample=lhsdesign(n,2); % рівномірно розподілені у квадраті точки
>> [X,Y]=meshgrid(linspace(0,1,12)); % сітка
>> training=[X(:) Y(:)]; % масив для тренінгу
>> gr=(sign(X-0.5)+2*sign(Y-0.5)+3)/2+1; % номери точок: 1, 2, 3, 4
>> group=gr(:); % такий же стовпець
>> class=classify(sample,training,group); % класифікували
>> s='o^.+'; % види точок
>> figure
>> hold on
>> for kk=1:4,
    nk=find(class==kk); % номери точок, що потрапляють в один кластер
    plot(sample(nk,1),sample(nk,2),'k' s(kk))
end
>> hold off
>> axis equal % однаковий масштаб
>> xlim([0 1]) % границі по осі OX
>> ylim([0 1]) % границі по осі OY
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bфКласифікація точок у групи');
>> xlabel('\bfx') % мітка осі OX
>> ylabel('\bfy') % мітка осі OY
>> grid on, box on

```

На рис. 16.2 точки, що потрапляють у різні групи, показані різними символами.

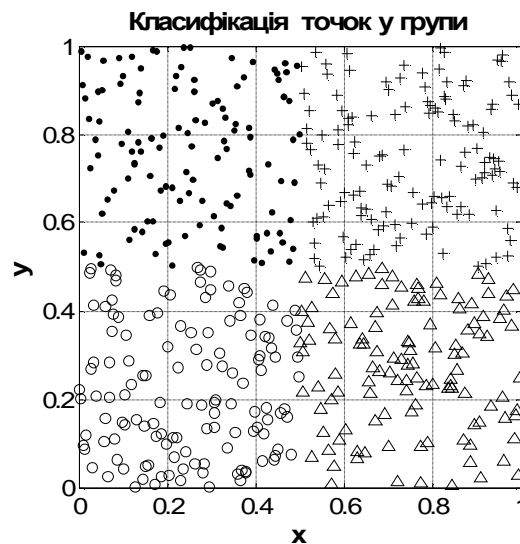


Рис. 16.2. Розподіл масиву даних по кластерам

### **CMDSCALE** Класичне багатовимірне шкалювання

За матрицею відстаней обчислюються координати точок у просторі мінімально можливої розмірності за допомогою класичного

багатовимірного шкалювання. Мета багатовимірного шкалювання співпадає з метою кластерного аналізу – в деякій метриці намагаються визначити відносні відстані між об'єктами, показати розшарування об'єктів у скороченому метричному просторі, виділити компактні угруповання (кластери), якщо вони є. Але на відміну від кластерного аналізу, методики якого не мають ніякого теоретичного обґрунтування, класичне багатовимірне шкалювання дуже подібне до методу головних компонент, більш того, один з варіантів метричного (класичного) шкалювання, коли приймаються евклідові відстані з попередньою стандартизацією ознак, просто співпадає з методом головних компонент.

*Синтаксис:*

**$Y = \text{cmdscale}(D)$**

**$[Y, e] = \text{cmdscale}(D)$**

*Опис:*

Функція  **$Y = \text{cmdscale}(D)$**  бере в якості аргументу матрицю відстаней  $D$  розміру  $n \times n$ , і повертає матрицю конфігурації  $Y$  розміром  $n \times p$ . Рядки  $Y$  містять координати  $n$  точок в  $p$ -вимірному просторі для деякого  $p < n$ . У якості  $p$  береться найменша можлива розмірність простору, у якому відстані між  $n$  точками, що задані в  $D$ , можуть бути обчислені.

Функція  **$[Y, e] = \text{cmdscale}(D)$**  у результативному параметрі  $e$  повертає власні значення матриці  $Y^*Y'$ . Якщо відстані в  $D$  – евклідові, то перші  $p$  елементів  $e$  – позитивні, а інші – нульові. Якщо перші  $k$  елементів  $e$  суттєво більше інших ( $n - k$ ), то можна залишити перші  $k$  стовпців  $Y$  як  $k$ -вимірні точки, відстані між якими приблизно дорівнюють  $D$ . Це може бути корисно в задачах скорочення розмірностей і візуалізації (якщо  $k = 2$ ).

Відстані у матриці  $D$  необов'язково повинні бути евклідові. Якщо це матриця неевклідових відстаней або більш загальна матриця відмінностей, то деякі елементи  $e$  можуть виявитися від'ємними. У цьому випадку *cmdscale* визначає  $p$  як число позитивних власних значень. У

цьому випадку скорочення розмірності до  $p$  дає розумну апроксимацію до  $D$  лише у тому випадку, коли від'ємні елементи є малими за модулем.

Матрицю  $D$  можна задавати або повністю, або у вигляді вектора, що містить елементи верхньої трикутної матриці, як у результативному параметрі функції **pdist**. У першому випадку це повинна бути дійсна симетрична матриця з нулями на головній діагоналі й позитивними іншими елементами. У випадку векторного завдання це повинен бути вектор з позитивними дійсними числами такої довжини, яка дозволяє відновити  $n$ . Можна також задати  $D$  як повну матрицю подібності. У цьому випадку  $D$  має бути дійсною симетричною матрицею з одиницями на головній діагоналі й іншими елементами, що менше 1. Функція *cmdscales* перетворить цю матрицю так, що відстані між точками в  $Y$  будуть приблизно дорівнювати  $\sqrt{1 - D}$ .

*Приклад:*

У прикладі генерується вибірка 5-вимірною нормального розподілу, у якого сильно корелюють між собою 1-а, 3-а й 5-а координати, а також 2-а й 4-а, а кореляція між цими групами змінних – слабка. Для них обчислюється матриця відстаней, за якою робиться спроба відновити координати точок у просторі найменшої розмірності. Усі ненульові розмірності виводяться на екран: їх виявилось 5.

```
>> n=100; % число точок
>> mu=[1 2 3 -1 0]; % середні
>> s=[ 2 0.01 2.2 0.02 1.8;...
      0.01 1 0.01 0.98 0.02;...
      2.2 0.01 3 0.05 2.4;...
      0.02 0.98 0.05 1 0.01;...
      1.8 0.02 2.4 0.01 2]; % коваріаційна матриця
>> X=mvnrnd(mu,s,n); % багатовимірною нормальною вибіркою
>> D=pdist(X); % відстані
>> [Y,e]=cmdscales(D); % багатовимірною шкалювання
>> dim=find(abs(e)>eps^(3/4)); % ненульові розмірності
>> fprintf('Ненульові розмірності\n N   e\n')
>> fprintf('%2.0f %14.10f\n',[dim,e(dim)])
>> plot(Y(:,1),Y(:,2),'k.', grid % графік
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bПерші дві координати');
>> xlabel('\ity\rm_1') % мітка осі OX
>> ylabel('\ity\rm_2') % мітка осі OY
Ненульові розмірності
```

N e  
 1 504.0629684892  
 2 176.3792297019  
 3 23.3725490211  
 4 5.6635816986  
 5 0.6404253311

На рис. 16.3 показані перші дві координати точок, що відповідають найбільшим власним значенням  $e$ .

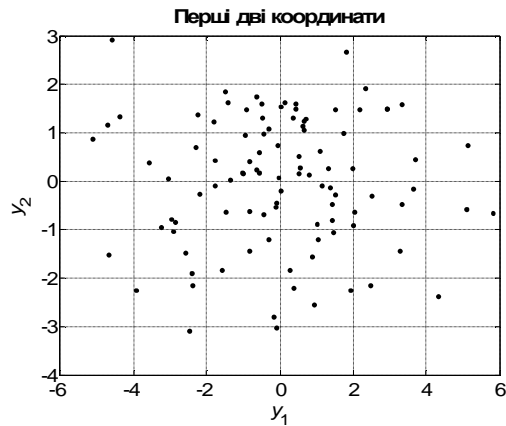


Рис. 16.3. Розшарування об'єктів у скороченому 2-вимірному просторі

### **MDSCALE** Метричне й неметричне багатовимірне шкалювання

За матрицею відстаней відновлюються координати точок у просторі мінімально можливої розмірності за допомогою метричного й неметричного багатовимірного шкалювання.

*Синтаксис:*

**Y = mdscale(D,p)**

**[Y, stress] = mdscale(D,p)**

**[Y, stress, disparities] = mdscale(D,p)**

**[...] = mdscale(..., param1, val1, param2, val2, ...)**

*Опис:*

Функція **Y = mdscale(D,p)** провадить неметричне багатовимірне шкалювання матриці відстаней  $D$  розміру  $n \times n$ , і повертає матрицю координат  $Y$ , що містить  $n$  точок (рядків) і  $p$  вимірів (стовпців). Евклідові відстані між точками в  $Y$  апроксимують монотонне перетворення

відповідних відстаней в  $D$ . За замовчуванням використовується критерій нормалізації Краскала (Kruskal)  $stress_1$ .

Матрицю  $D$  можна задавати або повністю розміром  $n \times n$ , або у вигляді перетвореної у вектор верхньої трикутної матриці, як у результативному параметрі функції **pdist**. У першому випадку це повинна бути дійсна симетрична матриця з нулями на головній діагоналі й позитивними іншими елементами. Функція **mdscale** розглядає значення NaN в  $D$  як відсутні, і ігнорує ці елементи. Значення  $Inf$  не сприймаються.

Можна також задати  $D$  як повну матрицю подібності. У цьому випадку  $D$  повинна бути дійсною симетричною матрицею з одиницями на головній діагоналі й іншими елементами, що менше 1. У цьому випадку функція **mdscale** перетворить матрицю  $D$  так, щоб відстані між точками в  $Y$  приблизно дорівнювали  $\sqrt{1 - D}$ .

Функція **[Y, stress] = mdscale(D, p)** у результативному параметрі  $stress$  повертає мінімальну міру відмінності, тобто міру відмінності, обчислену за  $Y$ .

Функція **[Y, stress, disparities] = mdscale(D, p)** у результативному параметрі  $disparities$  повертає відмінності за кожною відстанню, тобто монотонне перетворення відстаней  $D$ .

Функція **[...] = mdscale(..., param1, val1, param2, val2, ...)** враховує додаткові аргументи у вигляді пар "ім'я – значення". Можливі такі імена (рядки) і відповідні їм значення:

'*Criterion*' – критерій мінімізації в процедурі добору найкращих параметрів. Він також визначає тип шкалювання (метричне або неметричне). У наступних варіантах значень перші два відповідають неметричному шкалюванню, а наступні чотири – метричному;

'*stress*' – міра відмінності нормалізується як сума квадратів відстаней між точками, відома також як  $stress_1$  (значення за замовчуванням);

'*sstress*' – сума 4-х ступенів відстаней між точками;

'*metricstress*' – сума квадратів відмінностей у відстанях;

*'metricsstress'* – сума 4-х ступенів відмінностей у відстанях;

*'sammon'* – нелінійний картографічний критерій Семона (Sammon); для нього всі недіагональні елементи повинні бути строго позитивними;

*'strain'* – критерій, використовуваний у класичному багатовимірному шкалюванні;

*'weights'* – матриця або вектор такого ж розміру, як і  $D$ , з невід'ємною вагою відстаней. Ці вагові коефіцієнти використовуються для зміни внеску кожної відстані з  $D$  у міру відмінності, що мінімізується. Елементи  $D$ , відповідні до нульової ваги, ігноруються.

*'Start'* – метод, використовуваний для вибору початкової конфігурації точок в  $Y$ . Можливі його значення:

*'cmdscale'* – використовується рішення за методом класичного багатовимірного шкалювання (значення за замовчуванням; недоступне за наявності нульової ваги);

*'random'* – вибираються випадково з відповідним чином масштабованого  $p$ -мірного нормального розподілу з некорельованими координатами, як матриця початкових точок розміром  $n \times p$ , де  $n$  – розмір матриці  $D$ , а  $p$  – кількість стовпців у вихідній матриці  $Y$ . У цьому випадку замість аргументу  $p$  можна вказати порожній масив [], тому що аргумент  $p$  буде визначений як другий розмір даної матриці. Можна також використовувати тривимірний масив, припускаючи, що значення параметра *'Replicates'* дорівнює третій розмірності;

*'Replicates'* – кількість повторень рішення задачі шкалювання з новими початковими конфігураціями (за замовчуванням 1).

*'Options'* – структура з параметрами алгоритму, використовуваного для мінімізації критерію відповідності. Значення цієї структури з полями за замовчуванням може бути отримане за допомогою функції **statset**:

```
>> options=statset('mdscale')
```

```
| Tolfun: 1.000000000000000e-006
```

```
options =  
  Display: 'off'  
  Maxfunevals: []  
  Maxiter: 200  
  Tolbnd: []
```

```
Tolx: 1.0000000000000000e-006  
Gradobj: []  
Derivstep: []  
Funvalcheck: []
```

Використовувані поля:

*'Display'* – рівень повідомлень про похибки; можливі значення *'off'*, *'iter'* і *'final'*;

*'Maxiter'* – максимальна кількість ітерацій;

*'Tolfun'* – критерій закінчення ітераційного процесу мінімізації за значенням функції;

*'Tolx'* – критерій закінчення ітераційного процесу мінімізації за значеннями аргументів.

*Приклад:*

У прикладі генерується вибірка 5-вимірного нормального розподілу, у якого сильно корелюють між собою 1-а, 3-а й 5-а координати, а також 2-а й 4-а, а кореляція між цими групами змінних – слабка. Для них обчислюється матриця відстаней, за якою відновлюються координати точок у двовимірному просторі.

```
>> n=100; % число точок  
>> mu=[1 2 3 -1 0]; % середні  
>> s=[ 2 0.01 2.2 0.02 1.8;...  
      0.01 1 0.01 0.98 0.02;...  
      2.2 0.01 3 0.05 2.4;...  
      0.02 0.98 0.05 1 0.01;...  
      1.8 0.02 2.4 0.01 2]; % коваріаційна матриця  
>> X=mvnrnd(mu,s,n); % багатовимірна нормальна вибірка  
>> D=pdist(X); % відстані  
>> [Y,s]=mdscale(D,2); % двовимірне шкалювання  
>> fprintf('Міра відмінності = %10.8f\n',s) %  
>> plot(Y(:,1),Y(:,2),'k.').grid % діаграма  
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)  
>> title('\bf Двовірна апроксимація');  
>> xlabel('\ity\rm_1') % мітка осі OX  
>> ylabel('\ity\rm_2') % мітка осі OY  
Міра відмінності = 0.03674252
```

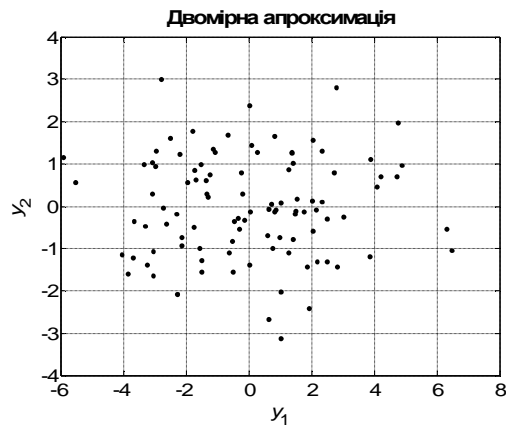


Рис.16.4. Відновлені координати точок у двовимірному просторі

На рис. 16.4 показані ці точки в нових координатах.

### **PROCRUSTES** Прокрустів аналіз

Прокрустовим аналізом називаються лінійні перетворення даних (точок): паралельний перенос, віддзеркалення, ортогональне обертання й шкалювання. Дана функція виконує ці дії.

*Синтаксис:*

```
d = procrustes(X,Y)
[d,Z] = procrustes(X,Y)
[d,Z,transform] = procrustes(X,Y)
```

*Опис:*

Функція **d = procrustes(X,Y)** провадить прокрустів аналіз точок (рядків) матриці  $Y$  для їх найкращої відповідності точкам матриці  $X$ . Критерій найкращої відповідності – мінімум суми квадратів відхилень. Функція *procrustes* повертає мінімізоване значення цієї величини в  $d$ . Величина  $d$  є стандартизованою величиною якості шкалювання  $X$ , заданого як:

$$\text{sum}(\text{sum}((X - \text{repmat}(\text{mean}(X, 1), \text{size}(X, 1), 1)).2, 1));$$

тобто це сума квадратів центрованих значень  $X$ . Проте, якщо  $X$  включає однакові точки, сума квадратів похибок не стандартизується.

Аргументи  $X$  і  $Y$  повинні мати однакову кількість точок (рядків); функція **procrustes** порівнює  $i$ -у точку  $Y$  з  $i$ -ю точкою  $X$ . Точки в  $Y$  можуть

мати меншу розмірність (число стовпців), ніж в  $X$ . У цьому випадку **procrustes** додає потрібну кількість нульових стовпців в  $Y$ .

Функція **[d,Z] = procrustes(X,Y)** у результативному параметрі  $Z$  повертає трансформовані значення  $Y$ .

Функція **[d,Z,transform] = procrustes(X,Y)** у результативному параметрі **transform** повертає інформацію про проведене перетворення у вигляді структури з полями:

$c$  – матриця паралельного переносу;

$T$  – матриця ортогонального обертання;

$b$  – масштабний множник (скаляр).

Зв'язок між параметрами, що повертаються:

$$Z = transform.b*Y*transform.T+transform.c$$

*Приклад:*

```
>> X=normrnd(0,1,[10 2]); % зразок
>> S=[0.5 -sqrt(3)/2; sqrt(3)/2 0.5]; % коваріаційна матриця
>> Y=normrnd(0.5*X*S+2,0.05,size(X)); % точки для аналізу
>> [d,Z,tr]=procrustes(X,Y); % прокрустів аналіз
>> plot(X(:,1),X(:,2),'kx',...
        Y(:,1),Y(:,2),'k.',...
        Z(:,1),Z(:,2),'ko');
>> axis equal, grid
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
>> title('\bfПрокрустів аналіз');
>> xlabel('\ity\rm_1') % мітка осі OX
>> ylabel('\ity\rm_2') % мітка осі OY
```

На рис. 16.5 показано результати рішення задачі прокрустова аналізу. Хрестиками позначені зразки, точками – значення даних, а кружками – результат рішення задачі: вихідні точки, пристосовані за допомогою прокрустова аналізу для найкращої відповідності зразкам.

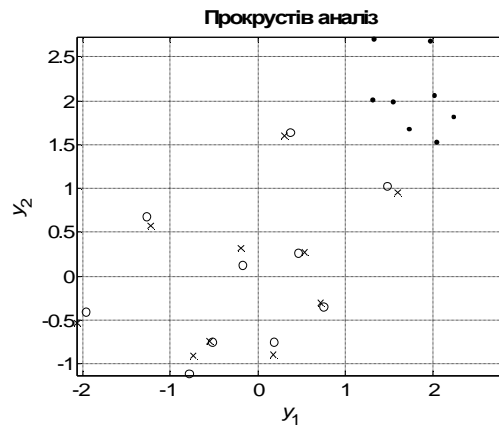


Рис.16.5. Результат прокрустова аналізу

### **MAHAL** Відстані Махаланобіса

Обчислюються відстані Махаланобіса між точками.

*Синтаксис:*

**D2 = mahal(Y,X)**

*Опис:*

Функція **D2 = mahal(Y,X)** обчислює відстані Махаланобіса (у квадратних одиницях) кожної точки (рядка матриці)  $Y$  від вибірки, заданої в матриці  $X$ . Кількість стовпців в  $Y$  повинне дорівнювати числу стовпців  $X$ , але кількість рядків може бути різним. Кількість рядків в  $X$  має бути більше числа стовпців.

Відстань Махаланобіса – це міра відстані від однієї точки до множини точок у багатовимірному просторі. Воно є критерієм мінімізації у лінійному дискримінантному аналізі.

*Приклад:*

У прикладі обчислюються відстані Махаланобіса від кожної точки, заданої в матриці  $Y$ , до множини з п'яти точок  $X$ .

```
>> X=lhsdesign(5,2)/10+ones(5,2)*diag([0.7 0.8]); % зразок
>> Y=[ 0 0; 0.1 0.3; 0.5 0.1; 0.4 0.2]; % точки
>> D2=mahal(Y,X); % відстані Махаланобіса
>> fprintf('Відстані Махаланобіса від Y до X\n Yi   Di\n')
>> fprintf('%2.0f %16.10f\n',[[1:4];D2'])
>> plot(X(:,1),X(:,2),'k.',Y(:,1),Y(:,2),'ko'), grid % діаграма
>> axis equal % однаковий масштаб
>> xlim([0 1]) % границі по осі OX
>> ylim([0 1]) % границі по осі OY
```

```
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)
```

```
>> title('\bfМножина точок X і Y');
```

Відстані Махаланобіса від  $Y$  до  $X$

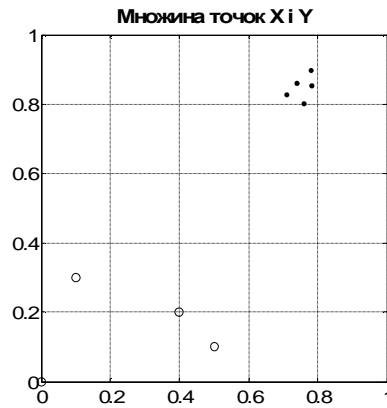
$Y_i$     $D_i$

1   803.2497679204

2   487.6765429641

3   448.1463771910

4   348.5920790876



**Рис.16.6. Відстані Махаланобіса від чотирьох точок, заданих в матриці  $Y$ , до множини з п'яти точок  $X$**

На рис. 16.6 показано ці точки: для  $Y$  – кружками, а для  $X$  – точками.

## 17. Функції нелінійного регресійного аналізу на підставі графа можливих рішень

У цьому розділі розглядаються функції **Statistics Toolbox** для рішення задач регресійного і дискримінантного аналізів із використанням дерева можливих рішень. Ці функції мають у своїй назві префікс **tree** (дерево).

<b>TREEFIT</b>	Розрахунок параметрів ієрархічної нелінійної регресійної моделі або бінарного дерева класифікації спостережень .....	344
<b>TREEDISP</b>	Графічне представлення ієрархічної нелінійної регресійної моделі або бінарного дерева класифікації спостережень .....	358
<b>TREEPRUNE</b>	Розрахунок параметрів скороченого бінарного дерева рішень ієрархічної регресійної моделі .....	363
<b>TREETEST</b>	Визначення похибок ієрархічної нелінійної регресійної моделі .....	369
<b>TREEVAL</b>	Розрахунок залежної змінної за ієрархічною нелінійною регресійною моделлю / .....	374

### **TREEFIT** Розрахунок параметрів ієрархічної нелінійної регресійної моделі або бінарного дерева класифікації спостережень

Будується дерево можливих рішень для різних класифікаційних і регресійних задач.

*Синтаксис:*

**T = treefit(X,y)**

**T = treefit(X,y,'param1',val1,'param2',val2,...)**

*Опис:*

Функція **T = treefit(X,y)** створює дерево рішень *T* для аргументів *X* і значень залежної змінної *y*. Аргумент *X* – матриця розміром  $n \times m$ , де кожний рядок – це точка (спостереження), а стовпець – змінна. Аргумент

у для задач регресії – числовий вектор з  $n$  значень залежної змінної, а для задач класифікації – масив символів або масив рядків символів, що містить  $n$  назв класів. Результативний параметр  $T$  – бінарне дерево, кожний проміжний вузол якого розщеплюється у відповідності зі значеннями стовпців  $X$ . У якості умови вибору напрямку переходу виступає обмеження на значення незалежних змінних. Виводить результати в графічному вигляді в інтерактивному вікні Regression tree viewer. Параметр  $T$  являє собою структуру з великою кількістю полів, яка використовується в інших функціях, що розглянуті у даному розділі. Зокрема, обчислене дерево зручно графічно зображувати за допомогою функції **treedisp**.

Функція **T = treefit(X,y,'param1',val1,'param2',val2,...)** в додаткових параметрах '*param*<sub>1</sub>', '*param*<sub>2</sub>',... задає у вигляді пари "назва параметра – його значення" низку установ, які перелічені для задач регресії – в табл. 17.1, а для задач класифікації – у табл. 17.2.

Таблиця 17.1

### Значення параметра '*param*' для регресійних задач

Назва параметра ' <i>param</i> '	Призначення й можливі значення <i>val</i>
<i>'catidx'</i>	Вектор номерів стовпців матриці незалежних змінних $X$ , заданих на категоріальній шкалі
<i>'method'</i>	Вид задачі: ' <i>method</i> ' = ' <i>classification</i> ' – вирішується задача класифікації, ' <i>method</i> ' = ' <i>regression</i> ' – вирішується задача регресії. Якщо спостереження залежної змінної у задані як рядкові значення, то ' <i>method</i> ' за замовчуванням ухвалюється рівним ' <i>classification</i> '. Для $y(i)$ , заданих на числовій шкалі ' <i>method</i> ', за замовчуванням ухвалюється рівним ' <i>regression</i> '
<i>'splitmin'</i>	Мінімальне число спостережень у вихідній вибірці $(X, y)$ , відповідне до вузла бінарного дерева рішень, за якого можливе його наступне ділення. Значення за замовчуванням – 10
<i>'prune'</i>	Параметр дозволяє задати можливість розрахунку скорочених ієрархічних регресійних моделей. Для ' <i>prune</i> ' = ' <i>on</i> ' розраховуються параметри повного й послідовності скорочених ієрархічних регресійних моделей. Якщо ' <i>prune</i> ' = ' <i>off</i> ', то визначаються параметри лише повної ієрархічної регресійної

Для задач класифікації додатково можна враховувати матрицю вартості (втрат) – наслідків від неправильної класифікації, якщо така матриця може бути складена; додатково можна врахувати апіорні ймовірності класів, що може бути вирішальним у невизначених випадках; крім того, можна задати критерій класифікації.

Таблиця 17.2

**Значення параметра *'param'* для задач класифікації**

Назва параметра <i>'param'</i>	Призначення й можливі значення <i>val</i>
<i>'cost'</i>	Матриця вартості (втрат) $C$ розміру $p \times p$ , де $p$ – число різних значень відповідей або імен класів в $y$ . Елемент $C(i, j)$ – вартість неправильної класифікації точки в клас $i$ , якщо насправді її дійсний клас $j$ . Цей аргумент може також бути структурою $S$ із двома полями: <i>group</i> – імена груп; <i>cost</i> – матриця вартості (за замовчуванням $C$ – одинична матриця)
<i>'splitcriterion'</i>	Критерій розщеплення бінарного дерева рішень. Можливі значення <i>'splitcriterion'</i> : <i>'gdi'</i> – формування вузлів дерева рішень виконується на основі індексів відмінності Джині; <i>'twoing'</i> – використовується правило максимальної відповідності пари значень, тобто правило роздвоєння; <i>'deviance'</i> – ділення виконується за правилом мінімізації максимальної дисперсії. Значення за замовчуванням <i>'splitcriterion' = 'gdi'</i>
<i>'priorprob'</i>	Параметр задає апіорні ймовірності для кожного класу. Значення параметра <i>'priorprob'</i> визначаються вектором. Кожний елемент вектора значень <i>'priorprob'</i> відповідає апіорній ймовірності для окремого класу щодо $y$ . Значення <i>'priorprob'</i> може бути задане як структура $S$ із двома полями: <i>S.group</i> – назви класів щодо $y$ ; <i>S.prob</i> – вектор відповідних ймовірностей

*Приклади:*

1. Розглядається задача регресії виду  $y = f(X(:, 1), X(:, 2), X(:, 3))$ .

Залежна змінна  $y$  визначена на числовій шкалі.

```
>> x=1:1:100; % рядок перших 100 натуральних чисел
>> x=[x' x' x']; % три стовпці по 100 натуральних чисел.
>> x1=normrnd(0,10,100,3);
```

```

% три стовпці по 100 випадкових чисел нормального розподілу
>> X=x+x1; % вибірки по 100 елементів трьох незалежних випадкових величин
>> y=1:2:200; % рядок перших 100 непарних чисел
>> y1=normrnd(0,2,1,100); % рядок 100 випадкових чисел нормального розподілу
>> y=y+y1; % 100 значень випадкової залежної змінної
>> T=treefit(X,y'); % обчислюємо дерево рішень
>> treedisp(T); % графічне представлення

```

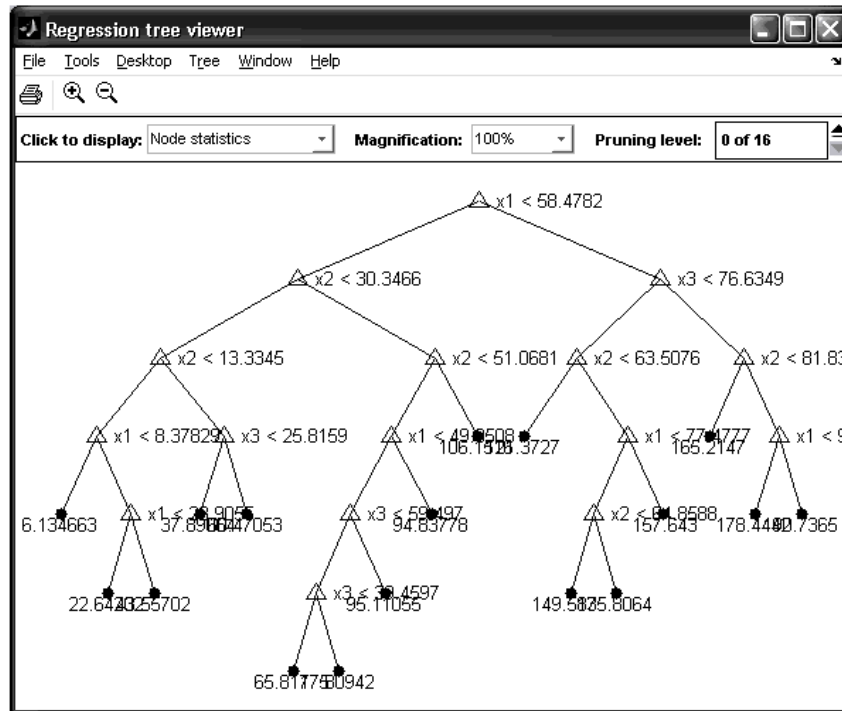


Рис. 17.1. Бінарне дерево класифікації спостережень

На рис. 17.1 наведені результати обчислень. Проміжні вузли дерева рішень відзначені мітками  $\Delta$  з умовами вибору щодо значення тієї чи іншої незалежної змінної. Кожний з кінцевих вузлів дерева рішень має мітку  $\bullet$  зі значенням залежної змінної. Для вибору однієї із двох можливих гілок дерева рішень, що виходять із проміжного вузла, діє правило: ліва гілка відповідає виконанню умови щодо незалежної змінної, права – невиконанню цієї умови. Визначення значення залежної змінної починається з першого проміжного вузла, далі, дотримуючись заданих умови вибирається права або ліва гілка. Аналогічно проглядаються наступні вузли доти, поки не буде досягнутий останній кінцевий вузол. Числа або категоріальна мітка, відповідна до останнього вузла, є значенням залежної змінної.

Меню "Click to display" в цьому інтерактивному вікні призначене для вибору інформації про виділений вузол дерева рішень (рис. 17.2).

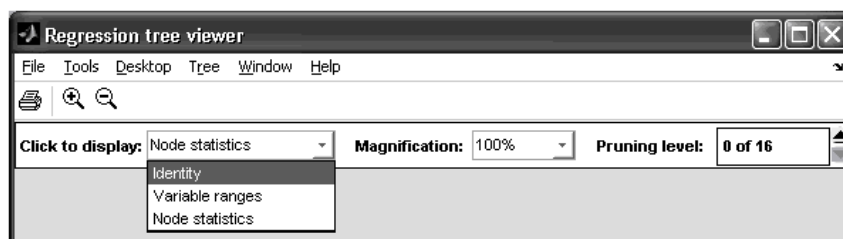


Рис. 17.2. Меню "Click to display"

В табл. 17.3 перелічені можливі пункти меню "Click to display":

Таблиця 17.3

**Пункти меню "Click to display"**

Пункт меню "Click to display"	Призначення
<i>Identity</i>	Відображається номер вузла, вид вузла: кінцевий або проміжний, і правило вибору вузла (рис. 17.3 а)
<i>Variable ranges</i>	Приводиться розмах кожної незалежної змінної для зазначеного вузла (рис. 17.3 б)
<i>Node statistics</i>	Статистичні дані щодо виділеного вузла: кількість незалежних змінних, що відповідні до виділеного вузла за вихідною вибіркою; середнє арифметичне й стандартне відхилення залежної змінної (рис. 17.3 в)

Для відображення інформації про вузли необхідно спочатку вибрати відповідну команду в меню "Click to display", далі виділити мишкою один з вузлів дерева рішень.

Як приклад на рис. 17.3 наведено інформація про вузол 2.

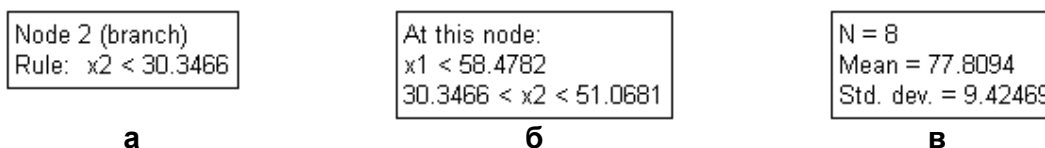


Рис. 17.3. Виконання пунктів меню Click to display:


а – *Identity*; б – *Variable ranges*; в – *Node statistics*

На рис. 17.3 (а) показано результат виконання пункту *Identity*: вузол 2 (проміжний), правило переходу  $x_2 < 30.3466$ .

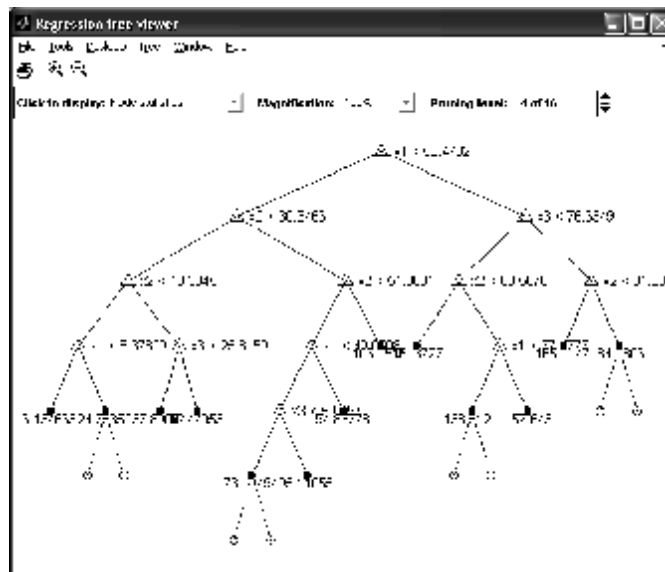
На рис. 17.3 (б) наведено результат виконання пункту *Variable ranges*: розмах незалежних змінних  $x_1 < 58.4782$  і  $30.3446 < x_2 < 51.0681$ .

На рис. 17.3 (в) наведено результат виконання пункту *Node statistics*: кількість незалежних змінних 8, середнє значення залежної змінної 77.8094, стандартне відхилення залежної змінної 9.42469.

Поле "Magnification" призначене для зміни масштабу відображення дерева рішень у графічному вікні.

Поле "Pruning level" відображає число рівнів повного дерева рішень і кількість рівнів, які були вилучені з нього. Наприклад, значення "0 of 16" відповідають повному дереву рішень, що складається з 16 рівнів, жодний рівень поки не вилучено (рис. 17.1). Зміна числа рівнів, що віддаляються, виконується за допомогою кнопок .

На рис. 17.4 наведено приклад скороченого дерева рішень, де вилучено чотири рівні з 16. У полі "Pruning level" відображається значення "4 of 16", тобто при формуванні скороченого дерева рішень були вилучені чотири рівня з 16-и.



**Рис.17.4. Приклад скороченого дерева рішень.  
Вилучено чотири рівні з 16**

Якщо є потреба, можна переглянути в Array Editor (редакторі масиву) усі поля обчисленої змінної  $T$ , розміри й типи цих змінних (рис 17.5).

Field	Value
method	'regression'
node	<33x1 double>
parent	<33x1 double>
class	<33x1 double>
var	<33x1 double>
cut	<33x1 double>
children	<33x2 double>
nodeprob	<33x1 double>
nodeerr	<33x1 double>
risk	<33x1 double>
nodesize	<33x1 double>
npred	3
catcols	[ ]
catsplit	<0x2 cell>
prunelist	<33x1 double>
alpha	<17x1 double>
ntermnodes	<17x1 double>

Рис. 17.5. Змінні структури T

Кожну зі змінних також можна переглянути, клацнувши на неї мишкою двічі. Нижче приводяться ймовірності перших 10 вузлів дерева із загальної кількості 33.

T.nodeprob	0,23
1	0,21
0,56	0,16
0,44	0,15
0,31	0,18
0,25	

2. Розглядається задача регресії виду  $y = f(X(:, 1), X(:, 2), X(:, 3))$ . Залежна змінна  $y$  визначена на числовій шкалі. Задане мінімальне число спостережень для подальшого ділення проміжного вузла  $'splitmin' = 20$  і скасована можливість розрахунку параметрів послідовності скорочених ієрархічних регресійних моделей  $'prune' = 'off'$ .

```
>> x=1:1:100;
>> x=[x' x' x'];
>> x1=normrnd(0,10,100,3);
>> X=x+x1;
>> y=1:2:200;
>> y1=normrnd(0,2,1,100);
>> y=y+y1;
>> T=treesfit(X,y,'prune','off','splitmin',20);
```

% мінімальне число спостережень для розщеплення 20  
 >> **treedisp(T);**

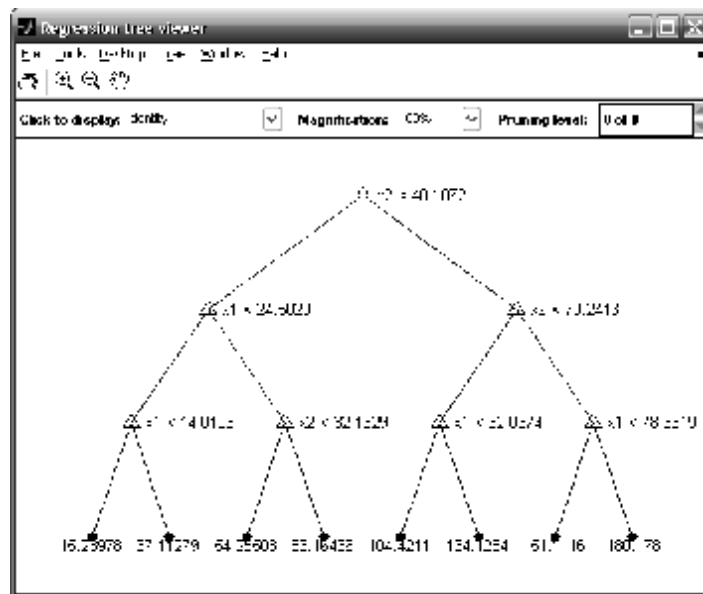


Рис. 17.6. Бінарне дерево класифікації спостережень

Кінцеві вузли дерева, що отримані з мінімальними середньоквадратичними відхиленнями (їх можна переглянути за методикою, описаною в попередньому прикладі), дають прогноз значень залежної змінної  $y$ .

3. У наведеному нижче прикладі регресії використовується категоріальна незалежна змінна *Model\_Year* з файлу *carsmall.mat* (у функції `treefit 'catidx' = 2`). Цей файл можна переглянути в Array Editor. Наведемо його фрагмент (табл. 17.4):

Таблица 17.4

Фрагмент файла *carsmall.mat*

Acc	Cyl	Dis	Hor	MPG	Model	Model_Year	Origin	Weight
8,5	8	390	190	15	amc	70	USA	3850
17,5	4	133	115	NAN	citroen	70	France	3090
11,5	8	350	165	NAN	chevrolet	70	USA	4142
11	8	351	153	NAN	ford	70	USA	4034
10,5	8	383	175	NAN	plymouth	70	USA	4166
11	8	360	175	NAN	amc rebel	70	USA	3850
10	8	383	170	15	dodge	70	USA	3563
8	8	340	160	14	plymouth	70	USA	3609
8	8	302	140	NAN	ford mustang	70	USA	3353

```

>> load carsmall
>> x=[Weight,Model_Year];
>> y=MPG;
>> t=treefit(x,y,'catidx',2);
>> treedisp(t,'name',{'Wt' 'Yr'});

```

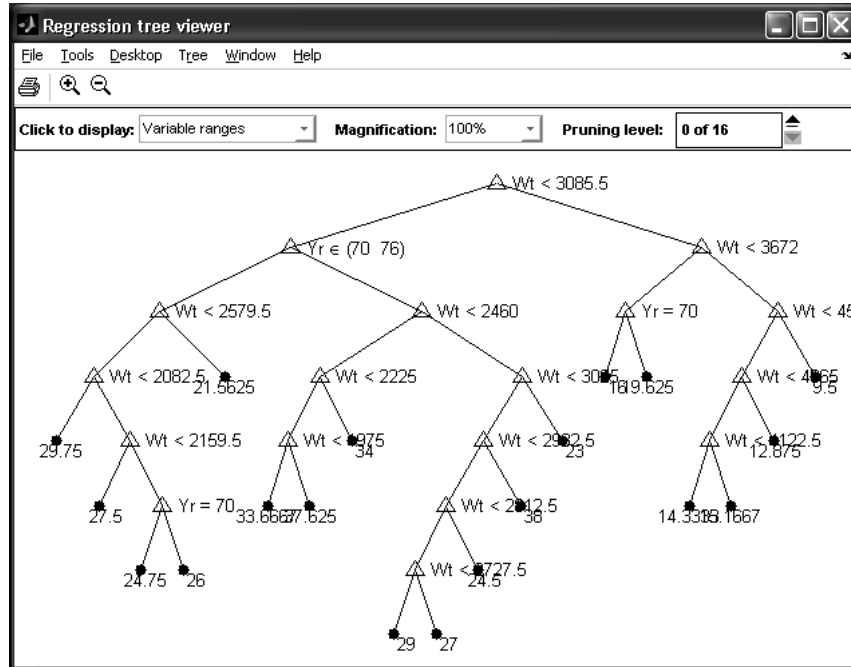


Рис. 17.7. Бінарне дерево класифікації спостережень файлу carsmall.mat

Використовуючи інтерактивність вікна, проаналізуємо отримані результати (рис. 17.7). Незалежними змінними є *Weight* (вага у фунтах) і *Model\_Year* (рік випуску). Залежна змінна – *MPG* (питомий пробіг в милях на галон пального).

Розглянемо параметри цього бінарного дерева.

N = 94
Mean = 23.7181
Std. dev. = 8.03573

Рис. 17.8. Параметри кореневого вузла

Кореневий вузол має такі параметри (рис.17.8): відібрано 94 авто (виключені рядки з нечисловими даними *MPG*, що наведені в табл. 17.4).

Середнє значення *MPG* дорівнює 27.7181, стандартне відхилення 8.03573.

Обране обмеження на значення *Weight* < 3085.5.

Далі йде розщеплення по цьому обмеженню: ліва гілка – авто, для яких виконується це обмеження, права для яких не виконується. Отримано два проміжні вузли: Параметри лівого показано на рис. 17.9. Можна відзначити, що туди ввійшли 58 авто випуску лише 1970 і 1976 років (див. обмеження на рік випуску для цього вузла).

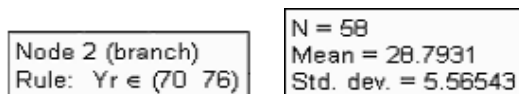


Рис. 17.9. Параметри проміжного вузла 2 (ліва гілка)

На рис. 17.10 показані параметри вузла 3 правої гілки.

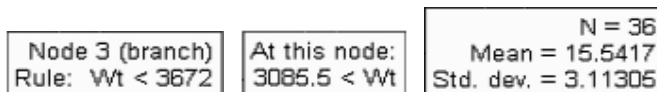


Рис. 17.10. Параметри проміжного вузла 3 (права гілка)

У цих вузлах знову йде розщеплення на дві гілки (створюється бінарне дерево) і так далі.

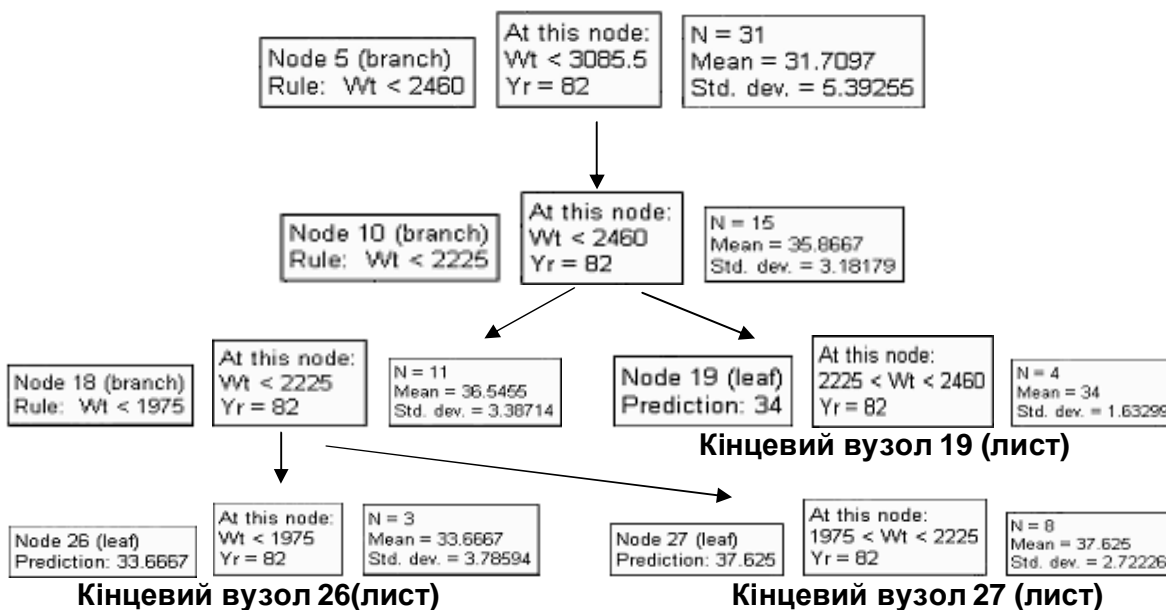


Рис. 17.11. Параметри вузлів гілки, що починається з п'ятого вузла

На рис. 17.11 наведені параметри вузлів лівої гілки, що починається з п'ятого проміжного вузла, у який потрапило 31 авто. У ліву гілку відсортовано 15 авто. Гілка наведена цілком аж до кінцевих вузлів. На рис. 17.11 видно зменшення дисперсії за переходу до наступного вузла. У кінцевих вузлах вказуються значення залежної змінної. Нумерація вузлів і гілок ведеться зверху – вниз і ліворуч – праворуч.

4. Вирішується задача класифікації даних, наведених у файлі *fisheriris.mat* для 4 незалежних змінних *SL*, *SW*, *PL*, *PW* і залежної змінної *species*. У якості критерію формування бінарного дерева рішень використовується правило мінімізації максимальної дисперсії. В табл. 17.5 наведено фрагмент цього файла на границі даних про сорти ірису.

Таблиця 17.5

**Фрагмент файла *fisheriris.mat***

Назва ірису	<i>SL</i>	<i>SW</i>	<i>PL</i>	<i>PW</i>
'setosa'	5.1	3.8	1.9	0.4
'setosa'	4.8	3.0	1.4	0.3
'setosa'	5.1	3.8	1.6	0.2
'setosa'	4.6	3.2	1.4	0.2
'setosa'	5.3	3.7	1.5	0.2
'setosa'	5.0	3.3	1.4	0.2
'versicolor'	7.0	3.2	4.7	1.4
'versicolor'	6.4	3.2	4.5	1.5
'versicolor'	6.9	3.1	4.9	1.5
'versicolor'	5.5	2.3	4.0	1.3
'versicolor'	6.5	2.8	4.6	1.5

Позначки змінних означають: *L* – довжина, *W* – ширина, *P* – пелюсток, *S* – квітколоже. Незалежні змінні: *SL*, *SW*, *PL*, *PW*. Залежна змінна – *species* – вид сорту ірисів. Вибірка складається з 150 елементів, по 50 зразків кожного сорту ірисів. Змінні розташовані у двох матрицях: числової *meas* розміром 150 × 4 (змінна типу "double") і масиву рядків *species* (змінна типу "cell").

```
>> load fisheriris
>> t = treefit(meas,species, 'splitcriterion', 'deviance');
treedisp(t);
```

На рис. 17.12 наведено бінарне дерево класифікації спостережень трьох сортів ірису за 4 незалежними змінними і залежною категоріальною змінною – сортом ірису.

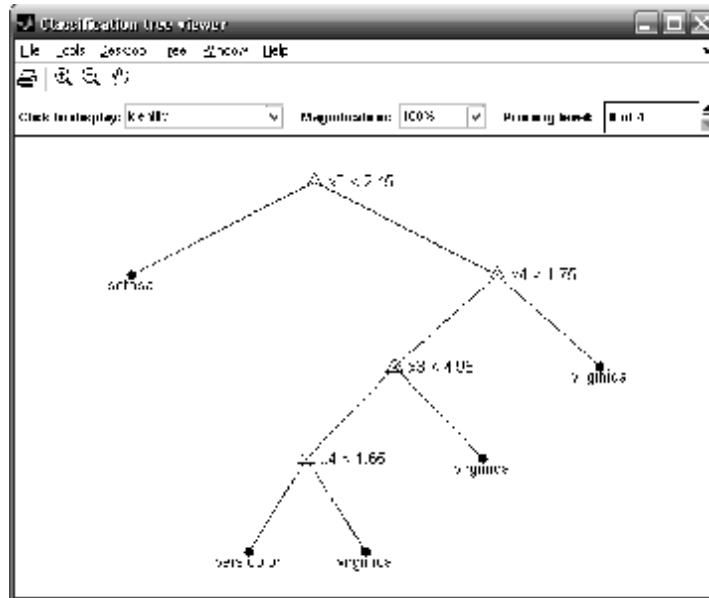


Рис. 17.12. Бінарне дерево класифікації спостережень файла fisheriris .mat

Приведемо параметри двох вузлів: кінцевого другого й проміжного четвертого.

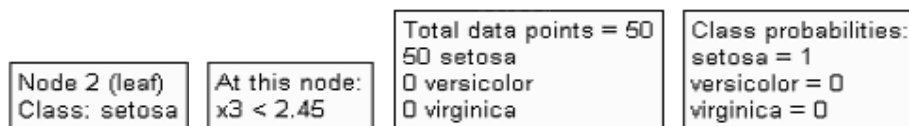


Рис. 17.13. Параметри кінцевого вузла 2

На рис. 17.13 видно, що в кінцевому вузлі 2 критерієм відбору була змінна  $PL$ , у вузол потрапили всі 50 ірисів *setosa*.

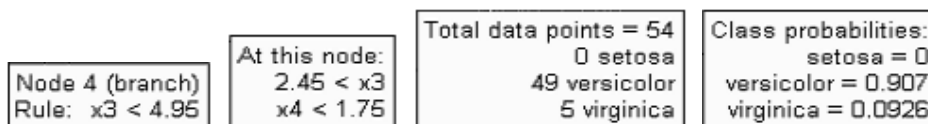


Рис. 17.14. Параметри проміжного вузла 4

На рис. 17.14 видно, що в проміжному вузлі 4 критерієм відбору були змінні  $PL$  і  $PW$ , у вузол потрапили 49 ірисів *versicolor* і 5 *virginica*.

Оскільки це права гілка, то в цьому вузлі не дотримуються умови:  $PL > 2.45$ ,  $PW < 1.75$ . В останньому віконці показані ймовірності цих подій. Таким чином, класифікація за довжиною й шириною квітколожа виявилася несуттєвою й функція не формувала за ними критеріїв.

5. Вирішується задача класифікації для 4 незалежних змінних *meas* і залежної змінної *species*. У якості додаткових параметрів задані: матриця втрат *cost* і вектор апіорних ймовірностей *priorprob*.

```
>> load fisheriris
>> cost=[0 0.8 0.9; 0.8 0 0.9; 0.8 0.9 0];
>> priorprob=[0.3 0.5 0.2];
>> T=treefit(meas,species,'cost',cost,'priorprob',priorprob);
>> treedisp(T);
```

На рис. 17.15 показано бінарне дерево класифікації спостережень файла *fisheriris.mat* щодо додаткових вхідних аргументів *cost* і *priorprob*. Збільшилося число вузлів і гілок, змінилися деякі критерії, але незалежні змінні залишилися ті ж самі.

Найбільш повну інформацію можна одержати, якщо вивести на екран структуру *T* і переглянути в Array Editor зміст будь-якої матриці або символічного рядка даних:

```
>> T=treefit(meas,species,'cost',cost,'priorprob',priorprob);
T =
method: 'classification' % – метод
node: [13x1 double] % – нумерація вузлів
parent: [13x1 double] % – номери проміжних вузлів
class: [13x1 double] % – нумерація класів об'єктів (їх три сорти ірисів)
var: [13x1 double] % – номери змінних у проміжних вузлах, за якими іде розщеплення
cut: [13x1 double] % – обмеження щодо вузлів на незалежні змінні
children: [13x2 double] % – номери дочірніх вузлів (проти номера проміжного вузла розташовані номери вузлів після розщеплення)
nodeprob: [13x1 double] % – ймовірності переходу у вузли
nodeerr: [13x1 double] % – помилки
risk: [13x1 double] % – ймовірності помилок
nodesize: [13x1 double] % – кількість елементів даних у вузлі
npred: 4 % – кількість незалежних змінних
catcols: []
prior: [0.3000 0.5000 0.2000] % – апіорні ймовірності (задається як аргумент)
nclasses: 3 % – кількість класів даних
cost: [3x3 double] % – матриця вартості (задається як аргумент)
classprob: [13x3 double] % – ймовірності влучення у вузол конкретного класу даних
classcount: [13x3 double] % – кількість у вузлі конкретного класу даних
classname: {3x1 cell} % – назви класу даних (три сорти ірисів)
catsplit: {0x2 cell}
```

prunelist: [13x1 double]  
alpha: [5x1 double]  
ntermnodes: [5x1 double]

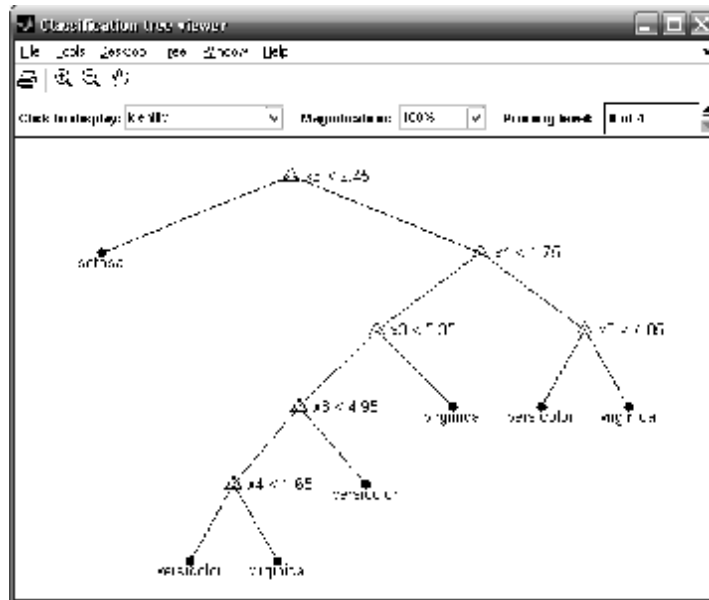


Рис.17.15. Бінарне дерево класифікації спостережень файлу fisheriris.mat

6. Вирішується задача класифікації для 4 незалежних змінних *meas* і залежної змінної *species*. Додаткові параметри – вартості помилок класифікації й апіорні ймовірності класів залежної змінної – задані як структури  $S_1$ ,  $S_2$  відповідно.

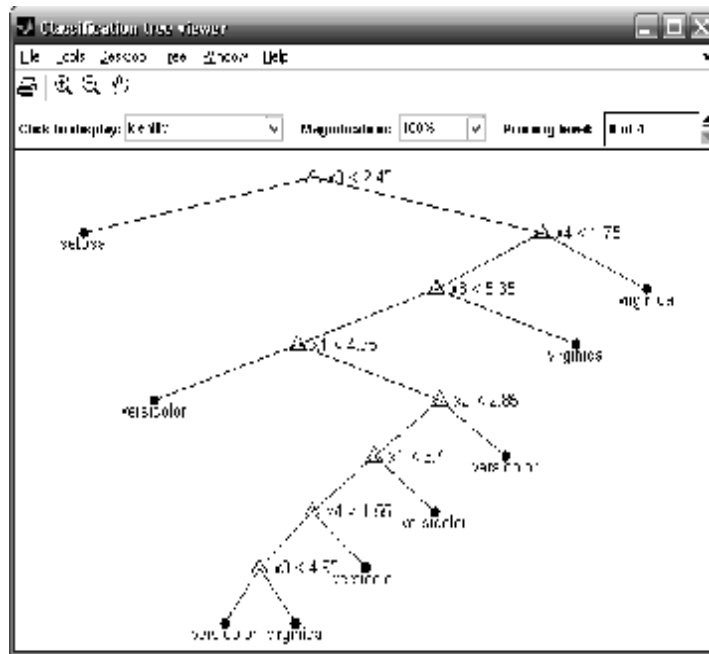


Рис.17.16. Бінарне дерево класифікації спостережень файлу fisheriris .mat

На рис. 17.16 видно, що в цьому випадку в процесі розщеплення й класифікації беруть участь усі чотири змінні.

Оператори:

```
>> load fisheriris
>> S1.group=['setosa ','versicolor','virginica'];
>> S1.cost=[0 0.9 0.9; 0.8 0 0.6; 0.7 0.8 0];
>> S2.group=['setosa',4,'versicolor',4,'virginica',4];
>> S2.prob=[0.3 0.5 0.2];
>> t = treefit(meas,species,'cost',S1.cost,'priorprob',S2.prob);
>> treedisp(t);
```

**TREEDISP** Графічне представлення ієрархічної нелінійної регресійної моделі або бінарного дерева класифікації спостережень

Будується дерево можливих рішень для різних класифікаційних і регресійних задач.

*Синтаксис:*

```
treedisp(T)
treedisp(T,'param1',val1,'param2',val2,...)
```

Опис:

Функція **treedisp(T)** дозволяє одержати графічне представлення ієрархічної нелінійної регресійної моделі, що задається в  $u$ , або бінарного дерева для класифікації спостережень на підставі вхідного аргументу  $T$ . Параметр  $T$  визначає вид дерева рішень і задається як структура. Цей параметр є результативним аргументом функції **treefit**, яка описана вище. Результат відображається в графічному вікні. Параметр  $T$  являє собою структуру з великою кількістю полів, яка використовується в інших функціях, розглянутих у даному розділі. Проміжні вузли дерева рішень відзначаються мітками  $D$  з умовами вибору щодо значення тої чи іншої незалежної змінної. Для вибору однієї із двох можливих гілок дерева рішень, що виходять із проміжного вузла, діє правило: ліва гілка відповідає виконанню умови щодо незалежної змінної, права – невиконанню цієї умови. Визначення значення залежної змінної починається з першого проміжного вузла, далі, дотримуючись заданої умови, вибирається права або ліва гілка. Аналогічно проглядаються наступні вузли доти, поки не буде досягнутий останній кінцевий вузол. Числова або категоріальна мітка, що відповідає до останнього вузла, є значенням залежної змінної.

Функція **T = treedisp(T,'param1',val1,'param2',val2,...)** урахує додаткові аргументи у вигляді пар "ім'я – значення". Деякі з параметрів застосовуються для всіх видів дерев, а деякі – лише для класифікаційних. Додаткові параметри ' $param_1$ ', ' $param_2$ ', ...: задаються у вигляді пари "назва параметра – його значення". Для всіх дерев можливі такі параметри (табл. 17.6).

Таблиця 17.6

### Значення параметру ' $param$ ' у функції **treedisp**

Назва параметра ' $param$ '	Призначення й можливі значення $val$
$'names'$	Визначає мітки незалежних змінних у графічному вікні. Значення ' $names$ ' задаються у вигляді масиву символьних рядків. Порядок елементів у масиві рядків повинен відповідати послідовності незалежних змінних (стовпців) матриці $X$ , що передане як вхідний аргумент функції <b>treefit</b>

'prunelevel'


Початковий рівень за формування скороченого дерева рішень.  
Задається як ціле число



Приклади:

1. У прикладі завантажується база даних з файла `fisheriris.mat` щодо ірисів Фішера, і за критерієм *species* будується їх класифікаційне дерево.

```
>> load fisheriris; % база даних щодо ірисів Фішера  
>> t=treefit(meas,species,'splitmin',1); % розрахували дерево  
>> treedisp(t,'names',{'SL' 'SW' 'PL' 'PW'}); % зобразили дерево
```

Класифікаційне дерево показано на рис. 17.17. Створене вікно є інтерактивним і має наступні елементи керування.

Меню *File* – *Print* дозволяє друкувати фігуру. Воно дублюється кнопкою  з підказкою *Print Figure*.

Меню *Tools* – *Zoom In* і *Zoom Out* дозволяють змінити масштаб фігури. Для цього потрібно включити обрану опцію й клацнути на будь-якому місці фігури. Ці елементи меню також дублюються кнопками з підказками  .

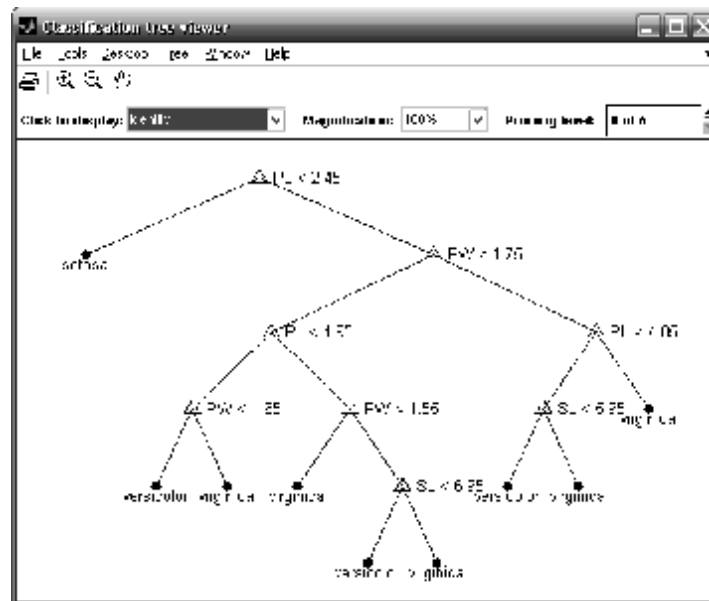


Рис. 17.17. Класифікаційне дерево

Меню *Desktop* – *Dock Classification tree viewer* вставляє фігуру в основне вікно **MatLab**, а в командному вікні показує поля структури дерева рішень.

В меню *Tree* можна перемкнути показ повного поточного дерева на показ нескороченого дерева й навпаки (*Show Full Tree – Show Unpruned Nodes*), включити-виключити показ міток проміжних вузлів (*Label Branch Nodes*) і включити-виключити показ міток вузлів-листів (*Label Leaf Nodes*).

*Click to display* (клацніть, щоб показати) – перелік що розкривається, дозволяє вибрати інформацію про вузол, яка з'являється за клацання мишкою на вузлі: *Identity* – номер вузла, чи є цей вузол проміжним чи фінальним (листом), правило ділення для проміжного вузла або теоретичне значення функції відгуку для вузла-листа; *Variable ranges* – діапазон значень кожного аргументу для даного вузла; *Class membership* – загальна кількість елементів, відповідних до даного вузла, ділення і їх розподіл за значеннями функції відгуку; *Estimated probabilities* – оцінки ймовірностей влучення в даний клас точок з різним значенням функції відгуку.

*Magnification* – перелік, що розкривається, дозволяє змінити масштаб діаграми, і є доповненням до елементів меню *Tools – Zoom In* і *Zoom Out* (і кнопкам зміни масштабу).

За допомогою шкали *Pruning level* (рівень скорочення) можна сховати або показати різні рівні скорочення побудованого дерева.

На рис. 17.18 наведено вигляд інформації за вибіру опції *Class membership* в переліку *Click to display*. Після виділення курсором точки  $PL < 4.85$  у вікні з'являється повідомлення про кількість елементів, відповідних до даного вузла, і їх розподілу за значеннями функції відгуку.

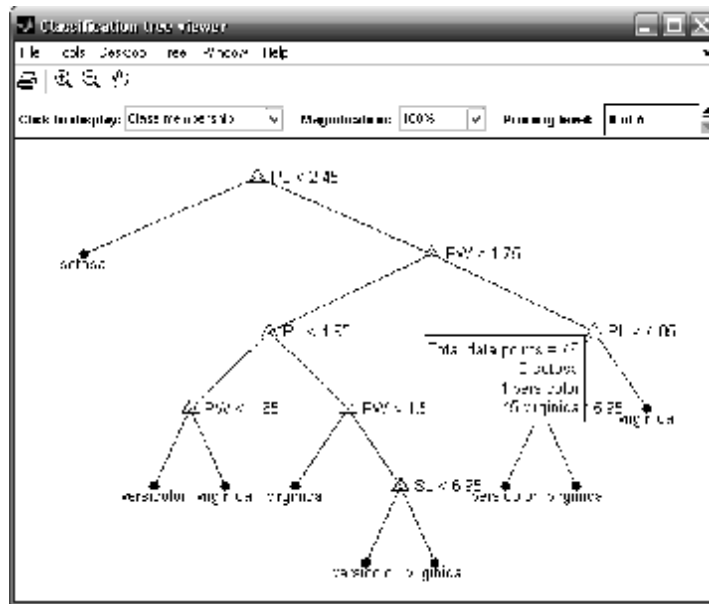


Рис.17.18. Вид спливаючого меню Class membership

На рис. 17.18 видно, що в п'ятому проміжному вузлі ставиться умова розщеплення щодо змінної  $PL$ . Спливаюче меню показує, що в цей вузол потрапило 46 елементів даних, з них групи *setosa* – 0, *versicolor* – 1, *virginica* – 45.

2. Графічне відображення ієрархічної нелінійної регресійної моделі виду  $y = f(X(:, 1), X(:, 2), X(:, 3))$ . Залежна змінна задана на числовій шкалі. Мітки незалежних змінних у графічному вікні визначені за допомогою масиву рядків *names*. Для відображення дерева рішень задано початковий рівень *prunelevel* (рис. 17.19).

```
>> x=1:1:100;
>> x=[x' x' x'];
>> x1=normrnd(0,10,100,3);
>> X=x+x1;
>> y=1:2:200;
>> y1=normrnd(0,2,1,100);
>> y=y+y1;
>> T=treefit(X,y');
>> names={'first', 'second', 'third'};
>> prunelevel=1;
>> treedisp(T, 'names', names, 'prunelevel', prunelevel);
```



першу чергу віддаляються проміжні вузли відповідні до найменшого зменшення похибки ієрархічної регресійної моделі. Іншими словами, використовується оптимальна схема скорочення, яка спочатку обрізає гілки, що мінімально впливають на вартість.

Функція **T2 = treeprune(T1,'nodes',nodes)** дозволяє сформувати скорочене бінарне дерево рішень  $T_2$  з початкової ієрархічної регресійної моделі  $T_1$ . Кінцеві вузли  $T_2$  визначаються як проміжні вузли, номери яких перераховані у векторі *nodes*. Якщо *nodes(i)* є підлеглим *nodes(j)*, то *nodes(i)* буде вилучено, а *nodes(j)* стане кінцевим вузлом  $T_2$ , тобто будь-який вузол  $T_1$ , внесений у перелік *nodes*, стане вузлом-листом в  $T_2$ , якщо його предок уже скорочений. Елементами вектора *nodes* повинні бути цілі позитивні числа. Номера вузлів можна визначити на графіку бінарного дерева рішень, отриманого за допомогою функції **treedisp**.

Функція **T2 = treeprune(T1)** повертає структуру  $T_2$ , що задає те саме бінарне дерево рішень, що й вхідний параметр  $T_1$ . У відмінності від  $T_1$  у  $T_2$  буде внесена інформація з оптимального скорочення бінарного дерева рішень. Такий варіант синтаксису функції **treeprune** використовується в двох випадках:

якщо  $T_1$  було отримане за допомогою скорочення іншого дерева рішень;

якщо  $T_1$  було розраховане з використанням **treefit** з параметром *'prune' = 'off'*.

Процедура скорочення вихідного дерева рішень дозволяє перевести проміжні вузли в кінцеві з видаленням останніх з початкової ієрархічної регресійної моделі. Це корисно, якщо дерево  $T_1$  створювалося шляхом скорочення іншого дерева або за допомогою функції **treefit**, у якій параметр *'prune'* установлений в *'off'*. Якщо дерево потрібно буде в майбутньому багаторазово скорочувати, то більш ефективно заздалегідь створити оптимальну послідовність скорочень.

У кожному конкретному застосуванні цієї функції корисно переглядати структури  $T_1$  і  $T_2$ . Програма іноді видає рішення задачі

класифікації (*method: 'classification'*), а іноді регресійної (*method: 'regression'*).

*Приклади:*

1. Розглядається задача регресії виду  $y = f(X(:, 1), X(:, 2), X(:, 3))$ . Залежна змінна  $y$  визначена на числовій шкалі. Вихідне бінарне дерево рішень скорочується на 5 рівнів. На рис. 17.20 наведено графіки початкового й скороченого бінарних дерев рішень.

```
>> x=1:1:100;  
>> x=[x' x' x'];  
>> x1=normrnd(0,10,100,3);  
>> X=x+x1;  
>> y=1:2:200;  
>> y1=normrnd(0,2,1,100);  
>> y=y+y1;  
>> T1=treefit(X,y'); % одержали первісне бінарне дерево  
>> treedisp(T1); % його графік
```

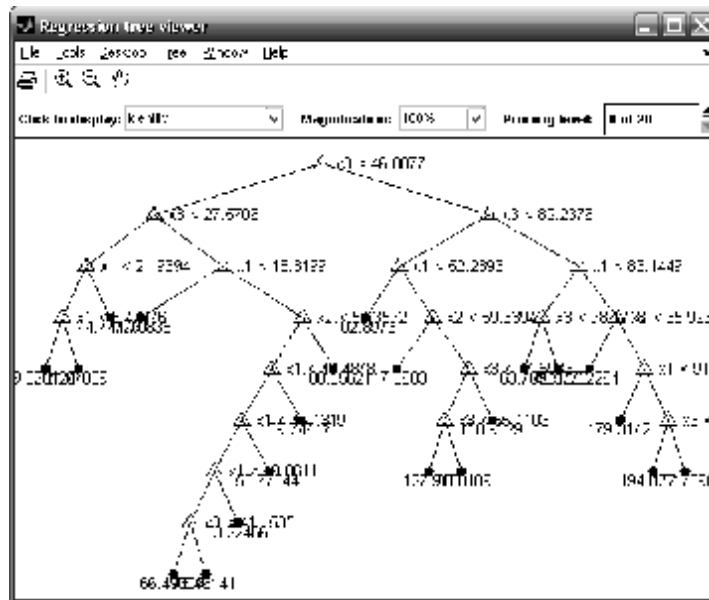


Рис. 17.20. Двадцатирівневе класифікаційне дерево

Скорочуємо одержане дерево на 5 рівнів (рис. 17.21):

```
>> level=5; % скоротили на п'ять рівнів  
>> T2=treeprune(T1,'level',level); % одержали дерево на 5 рівнів менше  
>> treedisp(T2);
```

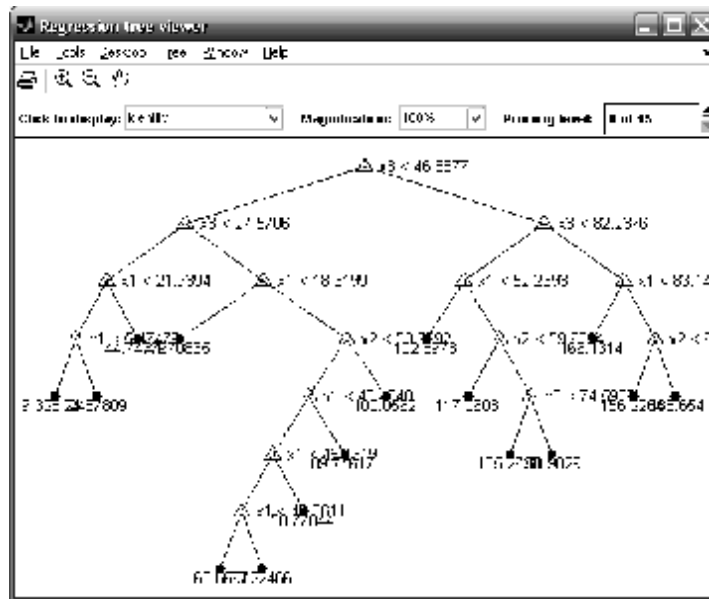


Рис. 17.21. Класифікаційне дерево з 15 рівнів для тієї ж вибірки

2. Розглядається задача регресії виду  $y = f(X(:, 1), X(:, 2))$ . Залежна змінна  $y$  визначена на числовій шкалі. Скорочення виконується для проміжних вузлів заданих вектором номерів *nodes*.

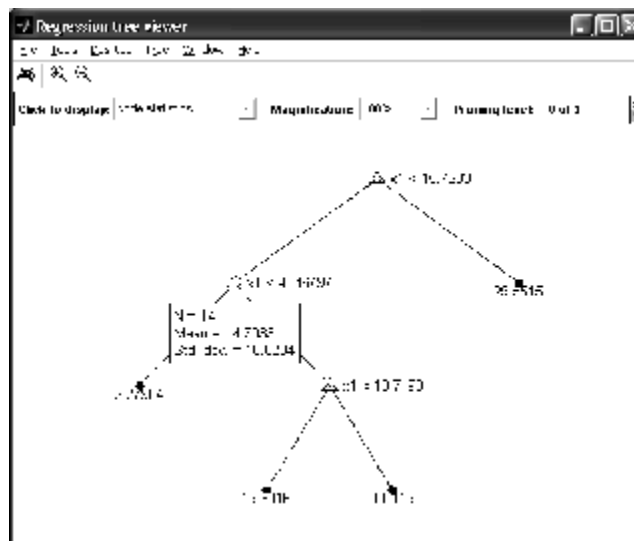


Рис.17.22. Класифікаційне дерево

Первісне класифікаційне дерево (рис. 17.22) обчислене так:

```
>> x=1:1:20;
>> x=[x' x'];
>> x1=normrnd(0,10,20,2);
>> X=x+x1;
>> y=1:2:40;
>> y1=normrnd(0,2,1,20);
```

```
>> y=y+y1;
>> T1=treefit(X,y'); % одержали первісне бінарне дерево
>> treedisp(T1); % його графік
```

Скорочуємо 2-й і 4-й рівні:

```
>> nodes = [2 4]; % скоротили 2 і 4 рівні
>> T2 = treeprune(T1,'nodes',nodes); % одержали скорочене дерево
>> treedisp(T2); % його графік
```

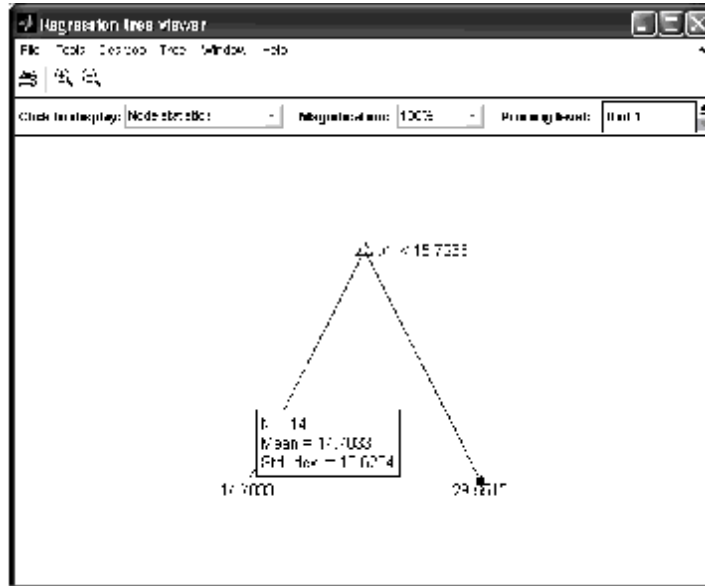


Рис. 17.23. Піддерево класифікаційного дерева

Зверніть увагу на однакові значення параметрів проміжного вузла на рис. 22 і кінцевого листа піддерева на рис. 17.23.

Наведемо також структуру  $T_2$ , у якій показано, що в цьому прикладі застосовувався метод регресії:

T2 =	cut: [3x1 double]	npred: 2
method: 'regression'	children: [3x2 double]	catcols: []
node: [3x1 double]	nodeprob: [3x1 double]	catsplit: {0x2 cell}
parent: [3x1 double]	nodeerr: [3x1 double]	alpha: [2x1 double]
class: [3x1 double]	risk: [3x1 double]	ntermnodes: [2x1 double]
var: [3x1 double]	nodesize: [3x1 double]	prunelist: [3x1 double]

3. Задача класифікації для 4 незалежних змінних `meas` файла `fisheriris.mat`, заданих на числовій шкалі, і залежної змінної `species`, елементи вектора якої задані по 3-м класам на категоріальній шкалі. У прикладі будується те ж дерево, що й у прикладі попередньої функції, а далі воно скорочується на один рівень.

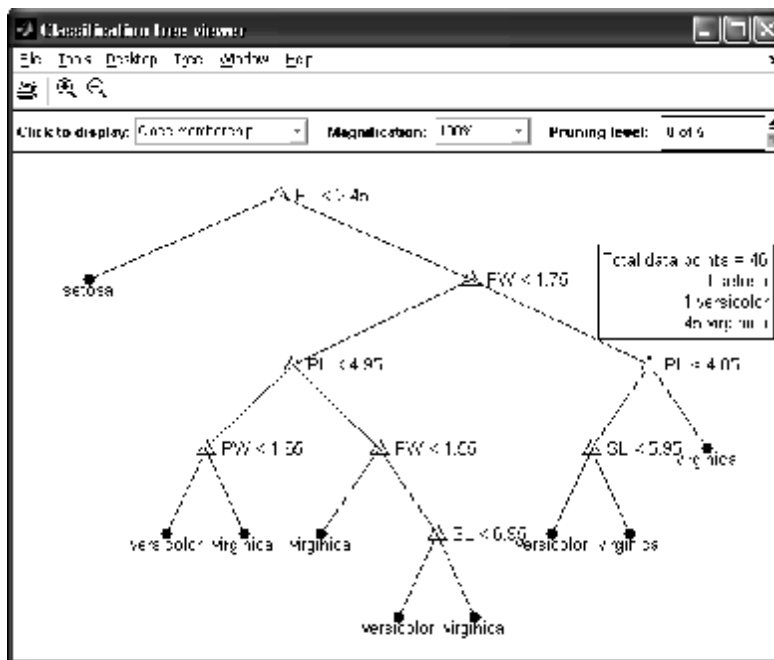


Рис. 17.24. Класифікаційне дерево щодо ірисів Фішера

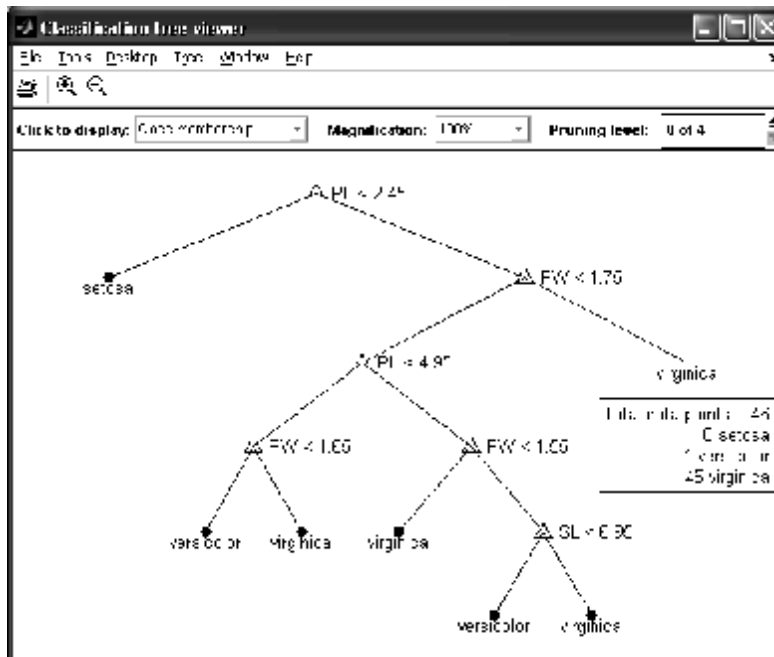


Рис. 17.25. Піддерево класифікаційного дерева для ірисів Фішера

Всі обчислення були виконані операторами:

```
>> load fisheriris; % база даних щодо ірисів Фішера
>> t=treefit(meas,species,'splitmin',1); % побудували дерево
>> treedisp(t,'names',{'SL' 'SW' 'PL' 'PW'}); % зобразили його
>> t1=treeprune(t,'level',1); % скоротили на 1 рівень
>> treedisp(t1,'names',{'SL' 'SW' 'PL' 'PW'}); % зобразили його
```

Цього разу у віконцях спливаючих меню на рис. 17.24 і 17.25 показані розподіли даних у проміжному вузлі дерева й кінцевому піддереві.

### **TREETEST** Визначення похибок ієрархічної нелінійної регресійної моделі

*Синтаксис:*

```
cost = treetest(T,'resubstitution')  
cost = treetest(T,'test',X,Y)  
cost = treetest(T,'crossvalidate',X,Y)  
[cost,secost,ntnodes,bestsize] = treetest(...)  
[...] = treetest(...,'param1',val1,'param2',val2,...)
```

*Опис:*

Функція **cost = treetest(T,'resubstitution')** обчислює вартість помилки дерева  $T$ , використовуючи метод підстановки (*'resubstitution'*). Аргумент  $T$  – це дерево рішень, що створене функцією **treefit**. Вартість дерева – це сума за всіма вузлами-листами оцінок ймовірностей вартостей вузлів. Якщо  $T$  – класифікаційне дерево, то вартість вузла – це число неправильних класифікацій значень функції в цьому вузлі. Якщо  $T$  – регресійне дерево, то вартість вузла – це середньоквадратична похибка за спостереженнями в даному вузлі. Результативний параметр *cost* – це вектор значень вартості для всіх піддерев в оптимальній послідовності їх скорочення для  $T$ . Обчислення вартості методом підстановки базується на тій самій вибірці, що й для побудови вихідного дерева, тому виходить нижня оцінка вартості за застосування дерева з іншими даними.

Функція **cost = treetest(T,'test',X,Y)** призначена для розрахунку вектора похибок *cost* ієрархічної нелінійної регресійної моделі, обумовленою структурою  $T$  за тестовою (*'test'*) вибіркою  $X$ ,  $Y$ . Тут  $X$  – матриця значень незалежних змінних,  $Y$  – вектор значень залежної змінної. Стовпці  $X$  відповідають незалежним змінним, рядки – спостереженням. Тестова вибірка й вихідна вибірка, що використана в

**treefit** для розрахунку  $T$ , повинні бути різними. Кількість і порядок незалежних змінних – стовпців матриць  $X$  у вихідної й тестової вибірки, має бути однаковим.

Функція **cost = treetest(T, 'crossvalidate', X, Y)** призначена для розрахунку вектора похибок  $cost$  ієрархічної нелінійної регресійної моделі, заданою структурою  $T$ , методом перехресної підстановки ('crossvalidate'). Матриця незалежних змінних  $X$  і вектор значень залежної змінної  $Y$  є заданою вибіркою, за якою була розрахована структура  $T$ .

Алгоритм розрахунку похибок регресійної моделі методом перехресної підстановки включає наступні етапи.

1. Вихідна вибірка ( $X, Y$ ) підрозділяється на 10 підгруп із приблизно рівними об'ємами, елементи яких вибираються випадковим чином. Якщо за формування  $T$  вирішувалася задача класифікації, то підгрупи формуються із приблизно однаковими ймовірностями витягти значення  $y(i)$  з кожної підгрупи.

2. Вибирається одна з отриманих підгруп. Визначаються значення з ( $X, y$ ), що не потрапили в обрану підгрупу. Ці значення використовуються для розрахунку бінарного дерева рішень ієрархічної нелінійної регресійної моделі.

3. За отриманим бінарним деревом рішень на основі елементів підгрупи розраховується похибка моделі.

4. Розрахунок по п. 2 – 3 повторюється в підгрупах, що залишилися. На підставі отриманих похибок для окремих підгруп розраховується загальна похибка регресійної моделі для заданої вибірки.

Функція **[cost, secost, ntnodes, bestsize] = treetest(...)** дозволяє розрахувати:

$cost$  – похибки ієрархічної нелінійної регресійної моделі;

$secost$  – вектор стандартних похибок для елементів вектора  $cost$ ;

$ntnodes$  – вектор чисел кінцевих вузлів послідовності скорочених бінарних дерев рішень;

*bestsize* - скаляр, що визначає оптимальний рівень скорочення повного дерева рішень *T*. Якщо *bestsize* = 0, то оптимальним буде повне бінарне дерево рішень *T*. Величина *bestsize* повинна відповідати найменшому бінарному дереву рішень, що забезпечує похибку ієрархічної нелінійної регресійної моделі рівною мінімальної похибці плюс її стандартна похибка.

У функції [...] = **treetest**(..., 'param1', val1, 'param2', val2, ...) додаткові вхідні параметри 'param<sub>1</sub>', 'param<sub>2</sub>', ... задаються у вигляді пари "назва параметра – їх значення" і дозволяють визначити (табл. 17.7):

Таблиця 17.7

### Значення параметра 'param' у функції treetest

Назва параметра 'param'	Призначення й можливі значення val
'nsamples'	Кількість підгруп за використання методу перехресної підстановки для розрахунку вектора похибок послідовності регресійних моделей. Значення за замовчуванням – 10
'treesize'	Критерій розрахунку величини <i>bestsize</i> . Можливі значення: 'se' – величина <i>bestsize</i> повинна відповідати найменшому бінарному дереву рішень, що забезпечує похибку ієрархічної нелінійної регресійної моделі рівною мінімальної похибки плюс її стандартна похибка; 'min' – <i>bestsize</i> вибирається так, щоб забезпечувалося мінімальне значення похибки регресійної моделі

#### Приклади:

1. Розглядається задача регресії виду  $y = f(X(:, 1), X(:, 2))$ . Залежна змінна *y* визначена на числовій шкалі. Похибка регресійної моделі розраховується методом зворотної підстановки (рис. 17.26).

```
>> x=1:1:20;
>> x=[x' x'];
>> x1=normrnd(0,10,20,2);
>> X=x+x1;
>> y=1:2:40;
>> y1=normrnd(0,2,1,20);
>> y=y+y1;
>> T=treetest(X,y);
>> treedisp(T);
```

Перевіряємо, яка була використана модель.

```
T =
method: 'regression'
>> cost = treetest(T, 'resubstitution')
cost =
42.9768
65.4293
144.5146
```

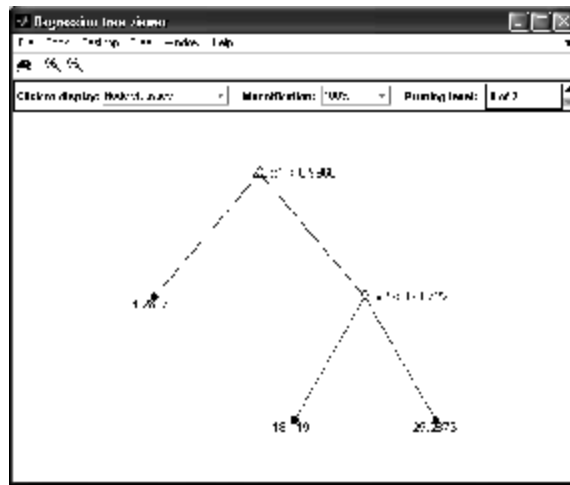


Рис. 17.26. Класифікаційне дерево

Оскільки використовувалася регресійна модель, то вартість вузла (міра похибки) – це середньоквадратична похибка за спостереженнями в даному вузлі. Вихідний параметр *cost* – це вектор значень вартості для всіх піддерев в оптимальній послідовності їх скорочення для дерева рішень *T*.

2. Розглядається задача регресії виду  $y = f(X(:, 1), X(:, 2))$ . Залежна змінна *y* визначена на числовій шкалі. Похибка регресійної моделі розраховується методом перехресної підстановки. Функція повертає вектор похибок ієрархічної нелінійної регресійної моделі, вектор стандартних помилок для елементів вектора похибок, вектор чисел кінцевих вузлів послідовності скорочених бінарних дерев рішень, оптимальний рівень скорочення повного дерева рішень. У якості додаткових параметрів задані: кількість підгруп за використання методу перехресної підстановки – 9, критерій розрахунку величини *bestsize* – 'min' (рис. 17.27).

```
>> x=1:1:20;
>> x=[x' x'];
>> x1=normrnd(0,1,20,2);
>> X=x+x1;
>> y=1:2:40;
>> y1=normrnd(0,2,1,20);
>> y=y+y1;
>> T=treefit(X,y);
>> treedisp(T);
```

Перевіряємо, яка використана модель.

```
>> T
T =
```

```

method: 'regression'
>> [cost,secost,ntnodes,bestsize]=
  treetest(T,'crossvalidate',X,y,'nsamples',9,'treesize','min')
cost =          || secost =          || ntnodes =          || bestsize =
  30.5886        || 8.7742           || 3                 || 0
  52.3030        || 14.2492          || 2                 ||
  148.7048       || 33.7068          || 1                 ||

```

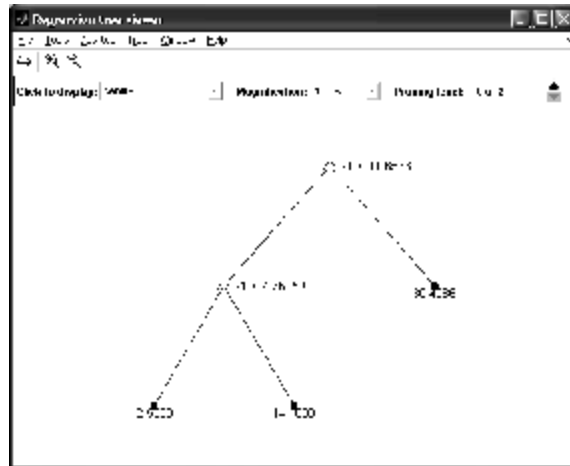


Рис. 17.27. Класифікаційне дерево прикладу

Одержали результат: *cost* – середньоквадратичні помилки відхилень за спостереженнями в даних вузлах; *secost* – вектор стандартних похибок для елементів вектора *cost*; *ntnodes* – вектор кількості кінцевих вузлів, *bestsize* – оскільки це значення 0, то найкращим буде повне, нескорочене дерево.

3. У прикладі визначається оптимальний розмір дерева. Спочатку завантажуються база даних щодо ірисів Фішера і за критерієм *species* будується повне класифікаційне дерево. Воно показано на рис. 17.24. Далі за критерієм '*crossvalidate*' провадиться оцінка дерева й усіх його піддерев, і дерево скорочується до найкращого рівня.

```

>> load fisheriris; % база даних щодо ірисів Фішера
>> t=treefit(meas,species,'splitmin',1); % побудували дерево
>> [c,s,n,best]=treetest(t,'cross',meas,species); % провадимо оцінку
>> tmin=treeprune(t,'level',best); % скоротили до найкращого рівня
>> [mincost,minloc]=min(c); % мінімальна оцінка і її номер
>> plot(n,c,'k-',n(best+1),c(best+1),'ks',n,(mincost+s(minloc))*ones(size(n)), 'k:');grid
>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14) % шрифт
>> title('\bfОцінка скорочення дерева')
>> xlabel('Кількість кінцевих вузлів-листів')
ylabel('Оцінка'), grid on, box on

```

```

>> t % Перевіряємо модель
t =
  method: 'classification'
% Виводимо на екран результати
c =          | s =          | n =          | best =
0.0600      | 0.0191         | 9            | 3
0.0600      | 0.0191         | 7            |
0.0667      | 0.0200         | 4            |
0.0667      | 0.0200         | 3            |
0.3333      | 0              | 2            |
0.6667      | 0              | 1            |

```

Усі ці результати показано на рис. 17.28. Суцільною лінією показана залежність вартості від кількості вузлів-листв. Квадратиком відзначено найкраще скорочення. Штриховою лінією позначено рівень мінімальної вартості (0.06) плюс одна стандартна похибка.

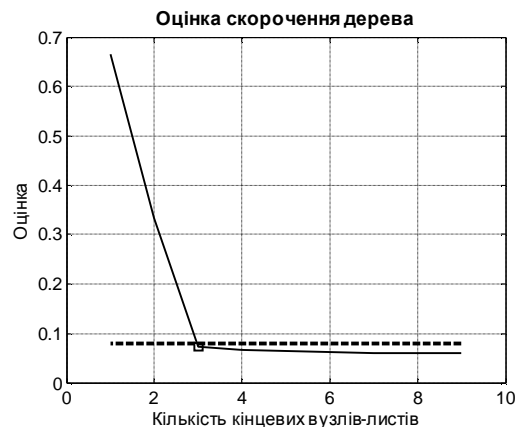


Рис. 17.28. Графік залежності вартості від кількості вузлів

Як бачимо, найкраще скорочення відповідає дереву з мінімальною кількістю листів, вартість якого ще не виходить за цей рівень. Рівень скорочення повного дерева рішень 3 ( $best = 3$ ). Саме бінарне дерево рішень для ірисів Фішера показано на рис. 17.24 – 17.25.

### **TREEVAL** Розрахунок залежної змінної за ієрархічною нелінійною регресійною моделлю

На підставі побудованого дерева рішень обчислюється теоретичні значення функції відгуку за заданими аргументами.

*Синтаксис:*

**YFIT = treeval(T,X)**

**YFIT = treeval(T,X,subtrees)**  
**[YFIT ,NODE] = treeval(...)**  
**[YFIT ,NODE,CNAME] = treeval(...)**

*Опис:*

Функція **YFIT = treeval(T,X)** дозволяє визначити значення залежної змінної *YFIT* за ієрархічною нелінійною регресійною моделлю, параметри якої задаються структурою *T* для значень незалежних змінних *X*. Структура *T* формується як результат роботи функції **treefit**. *X* задається як матриця з розмірністю  $n \times m$ , де  $n$  – число спостережень,  $m$  – кількість незалежних змінних. Рядки *X* відповідають спостереженням, стовпці *X* – незалежним змінним. Результативна змінна *YFIT* є вектором значень із числом елементів рівним  $n$ . Значення *YFIT(j)* розраховується згідно з рядками  $X(j,:)$ . Якщо за формування структури *T* вирішувалася задача класифікації, то *YFIT(j)* є номером класу, до якого належить точка з координатами  $X(j,:)$  у просторі незалежних змінних. Для визначення назви класу відповідного до номера *YFIT(j)* використовується третій результативний параметр *cname* (див. нижче).

Функція **YFIT = treeval(T,X,subtrees)** у додатковому вхідному параметрі *subtrees* задає число рівнів, що видаляються за формування послідовності скорочених ієрархічних регресійних моделей і розрахунку за цими моделями значень залежної змінної *YFIT* для матриці незалежних змінних *X*. Параметр *subtrees* задається як вектор цілих чисел: нульове значення відповідає повному дереву рішень, 1 – дереву рішень скороченому на один рівень і т.д. Структура *T* повинна бути сформована з наявністю в ній інформації про послідовність скорочених бінарних дерев функціями **treefit** або **prunetree**. Якщо вектор *subtrees* містить  $k$  елементів, а матриця *X* –  $n$  рядків, то *YFIT* буде матрицею з розмірністю  $n \times k$ , де  $j$ -й стовпець значень залежних змінних буде розрахований за деревом рішень, скороченому до *subtrees(j)* рівня. Елементи у векторі *subtrees* повинні бути розташовані за зростанням.

Функція `[YFIT,NODE] = treeval(...)` крім значень залежної змінної функція повертає масив номерів вузлів *NODE*, співвіднесених з кожним рядком матриці незалежних змінних  $X(j,:)$ . Розмірність *YFIT* і *NODE* має збігатися. За допомогою функції `treedisp` в інтерактивному режимі можна визначити номер кожного виділеного в графічному вікні вузла бінарного дерева рішень.

Функція `[YFIT,NODE,CNAME] = treeval(...)` повертає масив символічних рядків *CNAME*, що містить назви класів, знайдених за деревом рішень *T*. Результативна змінна *CNAME* використовується лише тоді, коли структура *T* була отримана після рішення задачі класифікації.

Нечислові значення NaN у матриці *X* розглядаються як відсутні. Якщо функція `treeval` зустрічається з таким значенням у проміжному (розгалуженому) вузлі, вона не може вирішити, за яким шляхом піти. У цьому випадку вона встановлює теоретичне значення, що дорівнює значенню функції відгуку в цьому проміжному вузлі.

#### Приклади:

1. Розглядається задача регресії виду  $y = f(X(:, 1), X(:, 2), X(:, 3))$ . Залежна змінна *y* визначена на числовій шкалі. Значення залежної змінної розраховуються в точках  $x_{new} = 2.5 : 10 : 100$ .

<pre>&gt;&gt; x=1:1:100; &gt;&gt; x=[x' x' x']; &gt;&gt; x1=normrnd(0,10,100,3); &gt;&gt; X=x+x1; &gt;&gt; y=1:2:200; &gt;&gt; y1=normrnd(0,2,1,100); &gt;&gt; y=y+y1; &gt;&gt; T=treefit(X,y'); &gt;&gt; xnew=2.5:10:100; &gt;&gt; xnew=[xnew' xnew' xnew']; &gt;&gt; YFIT=treeval(T, xnew)</pre>	<pre>YFIT =     6.9326    31.1466    51.2528    57.4721    85.6034   118.8870   134.8997   134.8997   134.8997   181.5062</pre>
--	---

Тут генеруються 100 спостережень трьох незалежних змінних. Фрагмент перших шести елементів представлений нижче:

```
>> X
X =
-3.3256 -10.8778  7.3527
-14.6558 -20.0232 -4.0141
4.2533 12.8634  8.5118
6.8768 -1.1864 -6.9984
-6.4647  8.2737  5.8599
17.9092  8.3406 -14.0456
```

Далі генеруються 100 спостережень залежної змінної. Фрагмент перших шести значень представлено нижче:

```
Columns 1 through 7
-2.5842  3.5407  4.4438  2.3464 13.7349 10.9383 15.1449
```

Структура *T* визначає параметри регресійної моделі:

```
>> T
T =
method: 'regression'
```

Як повністю переглянути й вивчити цю модель, було описано вище.

Визначаємо значення залежної змінної в інших 10 точках (усі три незалежні змінні ухвалюють однакові значення), використовуючи ієрархічну регресійну модель. Будуємо графік регресії (рис. 17.29):

```
>> plot(xnew,YFIT)
>> grid on
```

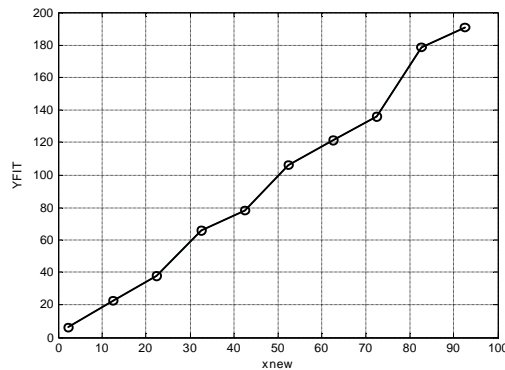
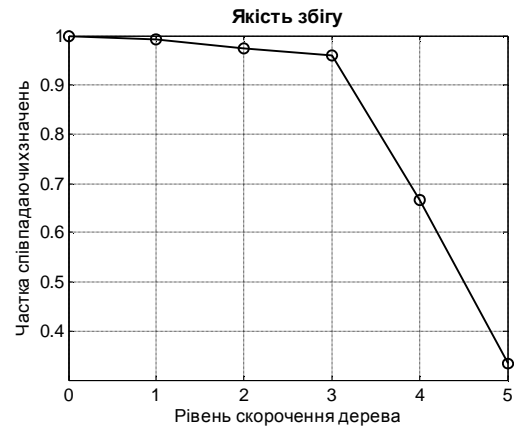


Рис.17.29. Графік регресії

2. Розглядається задача регресії виду  $y = f(X(:, 1), X(:, 2))$ . Залежна змінна  $y$  визначена на числовій шкалі. Значення залежної змінної розраховуються в точках  $xnew_1 = 2.5 : 2 : 20$ ,  $xnew_2 = 1.5 : 2 : 20$ . Залежні змінні розраховуються для повного дерева рішень і ієрархічних моделей, скорочених на 1 і 2 рівні:  $subtrees = [0 \ 1 \ 2]$ . Функція повертає матриці значень залежної змінної *YFIT* і відповідних їм вузлів бінарного дерева рішень *NODE*.



добре узгоджується з результатами роботи функції `treetest`: саме 3-му рівню скорочення відповідають 3 вузли-листя.



**Рис 17.30. Частка співпадаючих значень від рівня скорочення дерева**

## 18. Керування статистичними процесами

Ідея поточного попереджувального статистичного контролю якості полягає в тому, що протягом виготовлення великої партії продукції періодично випадковим чином на аналіз відбираються невеликі вибірки. За цими пробами невеликого обсягу контролюється хід процесу виробництва з метою своєчасного виявлення і попередження в ньому відхилень, які створюють погрозу погіршення необхідного якості продукції. Результати аналізів проміжних вибірок оформляються у вигляді так званих карт контролю якості. Звичайно стежать за стабільністю лише деяких найважливіших характеристик, зокрема, за змінами центру настроювання устаткування і за збільшенням мінливості. Відповідно, послідовно будуються контрольні карти для центру розсіювання і для самої міри розсіювання. У цьому розділі розглянуто обидва типи контрольних карт.

Вхідні дані можуть бути вектором, або матрицею. Випадковий вектор можна розглядати як значення деякого випадкового процесу в різні моменти часу, а матрицю – як кілька реалізацій випадкового процесу. Ряд функцій **Statistics Toolbox** сформульовані для задач саме у такий постановці. Ці функції описуються в цьому розділі у послідовності:

<b>CAPABLE</b>	Розрахунок індексів відтворюваності .....	381
<b>CAPAPLOT</b>	Графік відтворюваності процесу .....	384
<b>EWMAPLOT</b>	Контрольна карта експоненціально зваженого ковзного середнього (EWMA контрольна карта) .....	386
<b>SCHART</b>	Контрольна карта середніх квадратичних відхилень (s-контрольна карта) .....	392
<b>XBARPLOT</b>	Контрольна карта середніх арифметичних значень ( $\bar{X}$ -контрольна карта) .....	395

## **CAPABLE** Розрахунок індексів відтворюваності

Обчислюються параметри стабільності статистичного процесу. Функція **capable** працює у парі з наступною функцією **capaplot**.

*Синтаксис:*

**p = capable(data,specs)**  
**[p,Cp,Cpk] = capable(data,specs)**

*Опис:*

Функція **p = capable(data,specs)** дозволяє розрахувати ймовірність  $p$  виходу значень вибірки *data* за межі допусків *specs*. Вибірка *data* задається як вектор. Границі допусків представляються у вигляді двоелементного вектора: *specs*(1) – нижня границя допуску, *specs*(2) – верхня границя допуску.

Припущеннями для розрахунку  $p$  є:

1. Непротириччя вибірки *data* нормальному закону зі сталими математичним очікуванням і дисперсією,
2. Статистичною незалежністю результатів вимірів у вибірці.

Функція **[p,Cp,Cpk] = capable(data,specs)** дозволяє розрахувати ймовірність  $p$  виходу значень вибірки *data* за межі допусків *specs* і індекси відтворюваності процесу  $C_p$ ,  $C_{pk}$ .

Індекс  $C_p$  (індекс потенційної придатності) являє собою відношення різниці верхньої *USL* і нижньої *LSL* границь поля допуску до добутку  $6s$ , де  $s$  – точкова оцінка середнього квадратичного відхилення вибірки:

$$C_p = \frac{USL - LSL}{6s}.$$

Для центрованого технологічного процесу (вибіркове середнє збігається з номінальним значенням параметра технологічного процесу) значення  $C_p = 1$  відповідає відношенню числа дефектів до кількості виробів рівному  $1/1000$ . Згідно зі стандартами якості частку дефектів необхідно знижувати до величин  $1/1000000$  і менше. Величина частки дефектів  $1/1000000$  відповідає значенню  $C_p = 1.6$ . Таким чином, збільшення величини коефіцієнта  $C_p$  відповідає поліпшенню якості

технологічного процесу за рівнем дефектності. Чим більше значення  $C_p$ , тим більше пристосовується процес.

Індекс  $C_{pk}$  (індекс зміщеності технологічного процесу) розраховується за формулою:

$$C_{pk} = \min\left[\frac{USL - \mu}{3s}, \frac{\mu - LSL}{3s}\right],$$

де  $\mu$  – середнє арифметичне вибірки *data*.

З наведеної вище формули виходить, що індекс відтворюваності  $C_{pk}$  є відношенням мінімальної різниці середнього арифметичного вибірки *data* від верхньої або нижньої границі поля допуску параметра до трьох точкових оцінок середнього квадратичного відхилення. За  $C_p = 1$  для центрованого технологічного процесу буде також  $C_{pk} = 1$ . Якщо метою керування процесом не є його середнє, величина  $C_{pk}$  буде більш інформативною характеристикою пристосовності процесу.

*Приклади:*

1. Розрахунок імовірності браку.

```
>> data=normrnd(0,1,100,1); % модель процесу ( $\mu = 0$ ,  $\sigma = 1$ )
>> specs=[-2 2]; % границі контрольного інтервалу
>> p = capable(data,specs)
p =
    0.0413
```

Імовірність того, що значення вибірки будуть виходити за межі інтервалу  $[-2; 2]$  дорівнює 0,0413.

2. Приклад центрованого процесу із границями розсіювання параметра  $\mu \pm 3s$  співпадаючими із границями допусків.

```
>> x=normrnd(1,1,100,1); % модель процесу ( $\mu = 1$ ,  $\sigma = 1$ )
>> spec=[-3 3]; % границі контрольного інтервалу
>> [p,cp,cpk]=capable(x,spec); % параметри стійкості
>> fprintf('P(%d<x<%d)=%12.10f\n',spec,p)
>> fprintf('Cp=%12.10f\ncpk=%12.10f\n',cp,cpk)
P(-3<x<3)=0.0220139443
Cp=1.0022674907
`Cpk=0.6714769289
```

3. Графічне представлення центрованого технологічного процесу з границями розсіювання параметра, співпадаючими із границями допусків.

```
>> data=normrnd(0,1,100,1);
>> specs=[-3 3];
>> [p,Cp,Cpk] = capable(data,specs)
p =
    0.0021
Cp =
    1.0353
Cpk =
    0.9844
>> capaplot(data,specs); % див. нижче
```

```
>> H = line([0 0],[0 0.5]);
>> set(H,'Linewidth',3, 'Color','k')
>> H1=line([specs(1) specs(1)],[0 0.5]);
>> set(H1,'Color','m')
>> text(specs(1) + 0.1,0.4,'LSL');
>> H2=line([specs(2) specs(2)],[0 0.5]);
>> set(H2,'Color','m')
>> text(specs(2) + 0.1, 0.4,'USL');
>> grid on
```

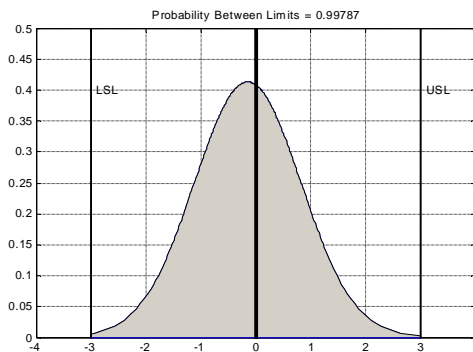


Рис. 18.1. Центрований технологічний процес із границями допусків

На рис. 18.1 представлено центрований технологічний процес ( $\mu = 1$ ,  $s = 1$ ) і границі допусків ( $-3s$ ;  $3s$ ). Результати обчислень показали, що ймовірність виходу процесу за межі стійкості не перевищує 0,0021. Там же наведені параметри потенційної придатності і зміщеності технологічного процесу. Вони, як і очікувалося, мало відрізняються від одиниці (вбірка центрована й нормальна).

4. Приклад зміщеного праворуч параметра технологічного процесу із границями розсіювання, що збігаються із границями допусків (за  $\mu = 0$ ).

```
>> data=normrnd(1.5,1,100,1); % модель процесу ( $\mu = 1.5$ ;  $s = 1$ )
>> specs=[-3 3];
>> [p,Cp,Cpk] = capable(data,specs);
>> fprintf('P(%d<x<%d)=%12.10f\n',spec,p)
>> fprintf('Cp=%12.10f\nCpk=%12.10f\n',cp,cpk)
P(-3<x<3)=0.0484239280
Cp=1.0022674907
Cpk=0.6714769289
```

Параметр зсуву виявився рівним 0.6715 ( $\mu = 1.5$ ).

Його графічне представлення

```
>> capaplot(data,specs);  
>> H = line([0 0],[0 0.4]); % номінальне значення ( $\mu=0$ )  
>> set(H,'Linewidth',3, 'Color','k')  
>> H0 = line([mean(data) mean(data)],[0 0.4]); % зсув праворуч ( $\mu=1.5$ )  
>> set(H0,'Linewidth',3, 'Color','b')  
>> H1=line([specs(1) specs(1)],[0 0.4]);  
>> set(H1,'Color','m')  
>> text(specs(1) + 0.1,0.4,'LSL');  
>> H2=line([specs(2) specs(2)],[0 0.4]);  
>> set(H2,'Color','m')  
>> text(specs(2) + 0.1, 0.4,'USL');  
>> grid on
```

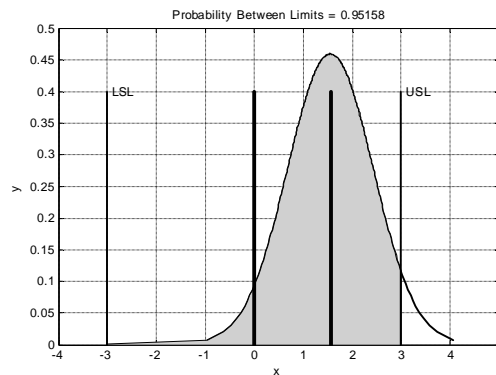


Рис. 18.2. Процес, зміщений щодо центру ( $m = 0$ )

На рис.18.2 представлені в графічному вигляді результати розрахунків розглянутої функції. Процес зміщений вправо щодо номінального на 1,5 одиниці. Пунктирними лініями позначені контрольні границі.

### **CAPAPLOT** Графік відтворюваності процесу

Наочно зображується розподіл спостережень процесу й інтервал значень, влучення в який контролюється.

*Синтаксис:*

```
p = capaplot(data,specs)  
[p,h] = capaplot(data,specs)
```

Опис:

Функція **p = caplot(data,specs)** дозволяє побудувати графік функції щільності нормального закону за точковими оцінками математичного очікування й середнього квадратичного відхилення вибірки *data* з накладеними границями допусків параметра *specs*. Вибірка вхідних значень *data* має бути представлена як вектор. Передбачається, що вибірка *data* не суперечить нормальному закону. Результативний параметр *p* є ймовірністю влучення значення випадкової величини в границі допусків *specs*. Границі допусків задаються у вигляді двоелементного вектора: *specs(1)* – нижня границя допуску, *specs(2)* – верхня границя допуску. Імовірність влучення значень параметра технологічного процесу на графіку представляється зафарбованою областю між границями допусків *specs*, віссю абсцис і кривою функції щільності розподілу ймовірності нормального закону.

Функція **[p,h] = caplot(data,specs)** крім величини *p* повертає масив покажчиків на елементи графіка відтворюваності *h*.

Приклади:

1. Приклад незміщеного технологічного процесу з нульовим математичним очікуванням і одиничною дисперсією. Нижня й верхня границі допусків дорівнюють  $LSL = -2$ ;  $USL = 2$ .

```
>> data = normrnd(0,1,50,1);  
>> p = caplot(data,[-2 2])  
>> grid on
```

```
| p =  
0.8809
```

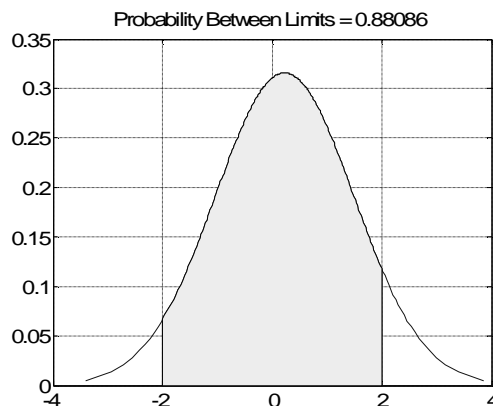


Рис 18.3. Незміщений процес

На рис. 18.3 показано незміщений процес ( $\mu=0$ ,  $s=1$ ) із контрольними границями. У титулі зазначена ймовірність влучення в ці границі – ймовірність влучення елементів вибірки в контрольний інтервал дорівнює 0.8809.

2. Приклад зміщеного технологічного процесу з одиничним математичним очікуванням і одиничною дисперсією. Нижня й верхня границі допусків дорівнюють  $LSL = -2$ ;  $USL = 2$ .

```
>> data = normrnd(1,1,50,1);
>> p = capplot(data,[-2 2])
>> grid on
```

p =  
0.8167

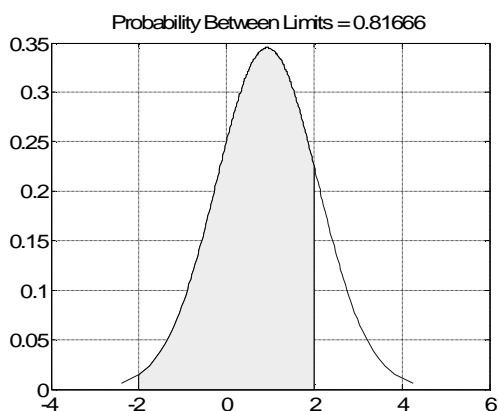


Рис. 18.4. Зміщений процес із контрольними границями

На рис. 18.4 представлена вибірка з параметрами  $\mu = 1$ ,  $\sigma = 1$  й контрольні границі  $[-2; 2]$ . Ймовірність влучення елементів вибірки в контрольний інтервал дорівнює 0.8167. Це ж число зазначене в титулі рисунка.

**EWMAPLOT Контрольна карта експоненціально зваженого ковзного середнього (EWMA контрольна карта)**

Ця функція будує графік експоненціально зваженого ковзного середнього для контролю статистичних процесів. Вважається, що вплив попередніх спостережень спадає за експоненціальним законом з параметром  $\lambda < 1$ . Випадковий вектор можна розглядати як

значення деякого випадкового процесу в різні моменти часу, а матрицю – як кілька реалізацій випадкового процесу.

*Синтаксис:*

```
ewmaplot(data)  
ewmaplot(data,lambda)  
ewmaplot(data,lambda,alpha)  
ewmaplot(data,lambda,alpha,specs)  
h = ewmaplot(...)
```

*Опис:*

Функція **ewmaplot(data)** призначена для побудови контрольної карти експоненціально зваженого ковзного середнього для вибірки *data*. За замовчуванням прийнято  $lambda = 0.4$ . Розмірність матриці *DATA* дорівнює  $n \times m$ , де  $n$  – кількість вибірок (рядків *DATA*),  $m$  – об'єм вибірки (число стовпців *DATA*). Послідовність вибірок (рядків матриці *data*) повинна відповідати порядку збору вихідних даних. На графіку контрольної карти відображаються вибіркові ковзні середні  $\bar{X}_i$ ,  $i = 1, \dots, n$ , центральна лінія, що відповідає загальному середньому  $\bar{X}$ , верхня *UCL* і нижня *LCL* контрольні границі. Верхня й нижня контрольні границі визначаються як  $UCL = \bar{X} + 3\sigma$ ,  $LCL = \bar{X} - 3\sigma$ , де  $\sigma$  – середнє квадратичне відхилення  $\bar{X}$ . Зображується також середній рівень (горизонтальна лінія), який позначається *CL*.

Функція **ewmaplot(data,lambda)** призначена для побудови контрольної карти експоненціально зваженого ковзного середнього для вибірки *data* і змінної *lambda*, відповідальної за ступінь впливу попередніх значень на поточне ковзне середнє. Більші значення *lambda* відповідають більшому значенню ваги попередніх спостережень. Величина *lambda* має перебувати в інтервалі  $[0 \ 1]$ . За замовчуванням  $lambda = 0.4$ .

Функція **ewmaplot(data,lambda,alpha)** дозволяє побудувати контрольну карту експоненціально зваженого ковзного середнього для вибірки *data* із заданою вагою попередніх спостережень *lambda* і рівнем

значимості  $alpha$  для верхньої й нижньої контрольних границь. За замовчуванням  $alpha = 0,0027$ , що відповідає відхиленням  $\pm 3\sigma$ :

```
>> norminv(1-0.0027/2)
ans =
    3.0000
```

Для розрахунку рівня значимості  $alpha$  відповідного  $\pm k$  середніх квадратичних відхилень ковзного середнього використовується вираження  $2*(1 - normcdf(k))$ . Наприклад, для  $k = 2$  значення  $alpha$  дорівнює 0.0455:

```
>> k = 2;
>> alpha =2*(1-normcdf(k))
alpha =
    0.0455
```

Функція **ewmaplot(data,lambda,alpha,specs)** призначена для побудови контрольної карти експоненціально зваженого ковзного середнього для вибірки  $data$  із заданою вагою попередніх спостережень  $lambda$ , рівнем значимості  $alpha$  і границь допуску параметра  $specs$ . Границі допусків визначаються як вектор із двома елементами:  $specs(1)$  – нижня границя допуску,  $specs(2)$  – верхня границя допуску.

Функція **h = ewmaplot(...)** у цьому варіанті синтаксису припускає будь-які перелічені вище вхідні параметри, результативним параметром  $h$  є вектор покажчиків на об'єкти графіка контрольної карти.

*Приклади:*

1. EWMA контрольна карта для технологічного процесу зі сталим нульовим математичним очікуванням і одиничною дисперсією.

```
>>data=normrnd(0,1,20,5)
data =
    0.4408 -1.6466 -0.8533 -1.0082  0.4938
    0.5654  0.4287  0.3456 -0.6646 -0.8709
   -0.6936 -0.7372  0.1098  0.5582  0.0798
    0.8339  0.5649 -1.1330 -1.1885 -0.5216
   -2.2374 -1.3842 -0.6831 -0.7755 -1.4139
    1.0976  0.4603 -0.2779  0.2710 -0.3843
   -0.0016  0.6294  0.6548  1.5350 -0.4579
   -1.6146  0.3798 -1.2484 -1.0523 -0.2915
   -1.2287 -1.0133 -0.5975  0.6256 -0.3012
    0.2074 -0.3472 -0.4818 -0.7976 -1.5886
    0.2209  0.4419  0.9834 -0.3135  1.0943
   -1.0061 -1.5902  1.7621 -0.6022  1.3242
   -0.4531 -0.7014  1.4274  1.2591 -0.1265
    1.3995 -1.0776  0.9118  0.8585 -0.7372
   -0.4620  1.0022  0.3268 -2.1053  0.2137
    0.0327  1.7295  0.0696 -0.3609 -0.4005
    0.7988  0.7090 -1.4998  0.5536  0.0649
    0.8968 -0.7479 -0.4182 -1.5564 -1.7580
    0.1379  0.2289 -0.0210 -0.2067  1.6867
   -1.6191 -0.2235  0.2284 -0.4256  0.3274
```

Елементи кожного рядка матриці DATA – це повторювані спостереження в той самий момент часу. Рядки розташовані в порядку зростання часу.

```
>>ewmplot(data)
```

На рис. 18.5 представлена контрольна карта нормального процесу з контрольними границями. Кожна з 10 точок є експоненціально зважене середнє даних 10 рядків матриці. Розташування точок демонструє, як процес розвивається у часі (ковзне середнє процесу).

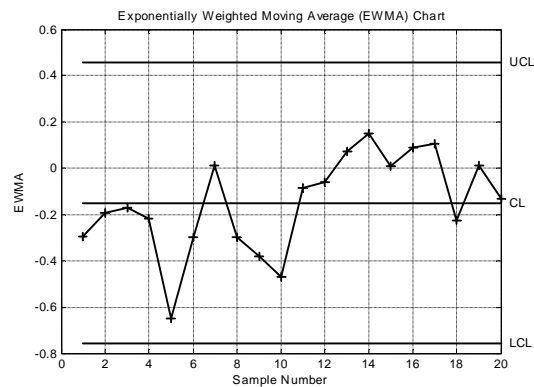


Рис. 18.5. Контрольна карта процесу з параметрами  $m = 0$ ,  $s = 1$

З рис. 18.5 видно, що жодна із точок не виходить за контрольні границі. Мітки полів позначає програма.

2. Вид EWMA контрольної карти для технологічного процесу зі сталим нульовим математичним очікуванням, одиничною дисперсією і коефіцієнтом впливу  $\lambda = [0.1 \ 0.3 \ 0.6 \ 1]$ . Більші значення  $\lambda$  відповідають більшій вазі попередніх спостережень.

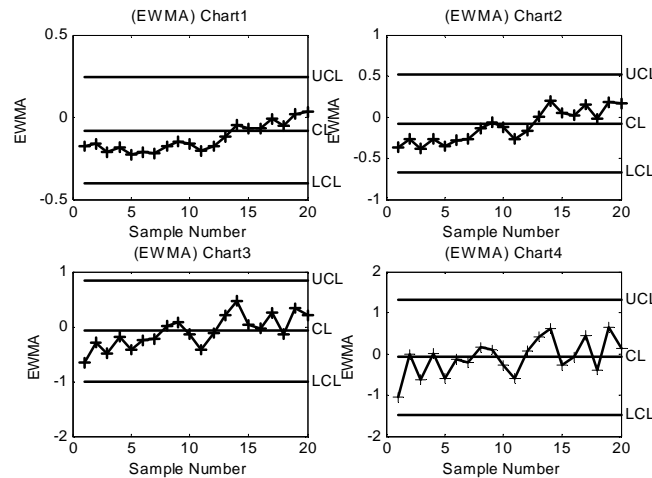


Рис. 18.6. Ступінь впливу значень  $\lambda = 0.1, 0.3, 0.6, 1$

На рис. 18.6 представлена в графічному вигляді вагова залежність наступних значень від попередніх, що обумовлено величиною  $\lambda$ . Середнє значення за вибіркою залишається тим самим. Змінюються значення контрольних точок і границь.

Оператори:

```
>>data=normrnd(0,1,20,5);
>> lambda=0.1;
>> subplot(2,2,1)
>> ewmaplot(data,lambda)
>> lambda=0.3;
>> subplot(2,2,2)
>> ewmaplot(data,lambda)
>> lambda=0.6;
>> subplot(2,2,3)
>> ewmaplot(data,lambda)
>> lambda=1;
>> subplot(2,2,4)
>> ewmaplot(data,lambda)
```

3. EWMA контрольна карта для технологічного процесу з лінійно наростаючим математичним очікуванням, що змінюються за лінійним законом  $10 + 0.02 \cdot t$  для 28 тимчасових точок і сталою дисперсією, рівною 0.52. Вибірка вихідних даних  $data$  є матрицею з розмірністю  $28 \times 4$ , де кількість вибірок дорівнює 28, число вимірів у вибірці – 4. Контрольна карта експоненціально зваженого ковзного середнього побудована для значення константи згладжування  $\lambda = 0.4$  і контрольних границь,

відповідних до рівня значимості  $\alpha = 0.01$ . Границі допусків параметра дорівнюють  $LSL = 9,75$ ;  $USL = 10,75$ .

```
>> t = (1:28)';
>> data = normrnd(10+0.02*t(:,ones(4,1)),0.5);
>> lambda =0.4;
>> alpha =0.01;
>> specs=[9.75 10.75];
>> ewmplot(data, lambda, alpha, specs)
```

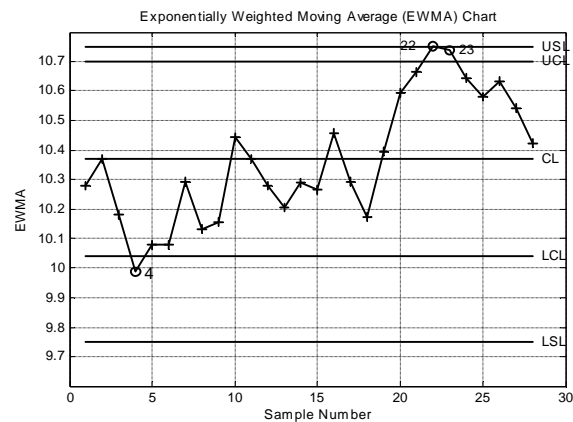


Рис. 18.7. Контрольна карта з лінійно наростаючим математичним очікуванням

На рис. 15.7 зображена EWMA контрольна карта процесу з лінійно наростаючим у часі математичним очікуванням із заданими контрольними границями  $[LSL; USL] = [9.75; 10/75]$ . На рисунку видно, що за ці границі виходить вибірка 22 (22 рядок матриці даних). Програма будує контрольний інтервал  $[LSD; USD] = [\bar{X}(CL) - 3\sigma, \bar{X}(CL) + 3\sigma]$ . За цей інтервал виходять три вибірки: 4, 22 і 23.

4. EWMA контрольна карта для технологічного процесу зі змінним математичним очікуванням за законом  $y = 10 + 0.2\sin(0.1t)$  і сталою дисперсією, яка дорівнює 0.52, для 100 тимчасових точок. Вибірка даних *data* є матрицею з розмірністю  $100 \times 4$ , де кількість вибірок дорівнює 100, число вимірів у вибірці – 4. Контрольна карта експоненціально зваженого ковзного середнього побудована для значення константи згладжування  $\lambda = 0.4$  і контрольних границь, відповідних до рівня

значимості  $\alpha = 0.01$ . Границі допусків параметра дорівнюють  $LSL = 9.5$ ;  $USL = 10.5$ .

```
>> t=(1:100)'; % моменти часу
>> r=normrnd(10+0.2*sin(t(:,ones(4,1))/10),0.5); % модель процесу
>> ewmaplot(r,0.4,0.01,[9.5 10.5]); grid % EWMA карта
>> set(get(gcf,'Currentaxes'),'Fontname','Times New Roman Cyr','FontSize',14)
>> title('\bфГрафік експоненційно зваженого ковзного середнього');
>> xlabel('\itt') % мітка осі OX
>> ylabel('EWMA') % мітка осі OY
```

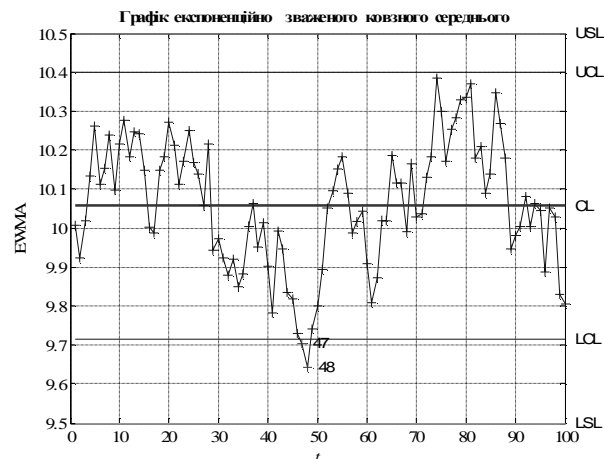


Рис. 18.8. Контрольна карта з наростаючим за синусоїдальним законом математичного очікування

На рис. 18.8 зображена контрольна карта з синусоїдальним законом математичним очікуванням. Мітки осей і титолу введені вручну. На карті видні вибірки, що виходять як за межі заданих контрольних границь, так і за межі смуги шириною  $6\sigma$ .

### **SCHART** Контрольна карта середніх квадратичних відхилень (s-контрольна карта)

Для випадкового процесу, заданого декількома реалізаціями, будується графік зміни в часі середньоквадратичного відхилення.

*Синтаксис:*

```
schart(DAT A,conf)
schart(DAT A,conf,specs)
schart(DAT A,conf,specs)
[outliers,h] = schart(DAT A,conf,specs)
```

Опис:

Функція **schart(data)** дозволяє одержати контрольну карту середніх квадратичних відхилень вибірових значень *DATA*. Розмірність матриці *DATA* дорівнює  $n \times m$ , де  $n$  – кількість вибірок (рядків *DATA*),  $m$  – об'єм вибірки (число стовпців *DATA*). Послідовність вибірок повинна відповідати порядку збору вихідних даних. На графіку контрольної карти відображаються вибіркові середні квадратичні відхилення  $s_i$ ,  $i = 1..n$ ), центральна лінія ( $\bar{s}$ ), верхня *UCL* і нижня *LCL* контрольні границі.

Середнє арифметичне вибірових середніх квадратичних відхилень розраховується за формулою:  $\bar{s} = \frac{1}{n} \sum_{i=1}^n s_i$ .

Верхня й нижня контрольні границі розраховуються за формулами:  $UCL = \bar{s} + 3\sigma_s$ ,  $LCL = \bar{s} - 3\sigma_s$ , де  $\sigma_s$  – стандартне відхилення для  $\bar{s}$ .

Якщо технологічний процес є статистично керованим, то за 1000 вибірок кількість точок, що виходять за контрольні межі, не повинна перевищувати 3-х у випадковому порядку. Таким чином, за малої кількості вибірок вихід вибірового середнього квадратичного відхилення за контрольні границі означає втрату процесом статистичної керованості.

Функція **schart(DATA,conf)**призначена для побудови *s*-контрольної карти вибірки *DATA* із заданою довірчою ймовірністю *conf* для верхньої й нижньої контрольних границь. За замовчуванням значення *conf* приймається рівним 0.9973. Величина  $conf = 0.9973$  відповідає інтервалу  $\pm 3\sigma_s$ .

```
>> norminv(1-(1-0.9973)/2)
ans =
    3.0000
```

Для розрахунку довірчої ймовірності **conf** відповідної  $\pm k\sigma_s$  використовується вираження  $1 - 2*(1 - normcdf(k))$ . Наприклад, обчислимо значення *conf* для  $k = 2$ :

```
>> k=2;
>> conf=1-2*(1-normcdf(k))
conf =
    0.9545
```

Функція **schart(DATA,conf,specs)** призначена для побудови  $s$ -контрольної карти із заданою довірчою ймовірністю розташування контрольних границь *conf* і границями допусків *specs*. Границі допусків визначаються як вектор із двома елементами: *specs*(1) – нижня границя допуску, *specs*(2) – верхня границя допуску.

У функції **[outliers,h] = schart(data,conf,specs)** параметрами є: *outlier* – вектор номерів вибірок (рядків матриці *DATA*), середні квадратичні відхилення яких вийшли за контрольні границі, *h* – вектор покажчиків на об'єкти графіка контрольної карти.

*Приклади:*

1.  $s$ -контрольна карта з параметрами за замовчуванням:

```
>> DATA=normrnd(0,1,20,5);
>> schart(DATA)
```

Контрольна карта цього прикладу зображена на рис. 18.9.

2.  $s$ -контрольна карта для контрольних границь, що дорівнюють  $\pm 1.5\sigma_s$  і границями допусків  $LSL = 0.2$ ;  $USL = 1.5$ . У якості результативних параметрів виступають: *outlier* – вектор номерів вибірок, що вийшли за контрольні границі, і *h* – вектор покажчиків на об'єкти графіка.

```
>> DATA=normrnd(0,1,20,10);
>> k=1.5;
>> conf =1-2*(1-normcdf(k));
>> specs=[0.2 1.5];
>> [outlier,h] = schart (DATA,conf,specs)
>> grid on
outlier =
     9
    13
    17
    20
h =
154.0034
155.0015
156.0015
157.0015
158.0015
159.0020
170.0023
171.0021
```

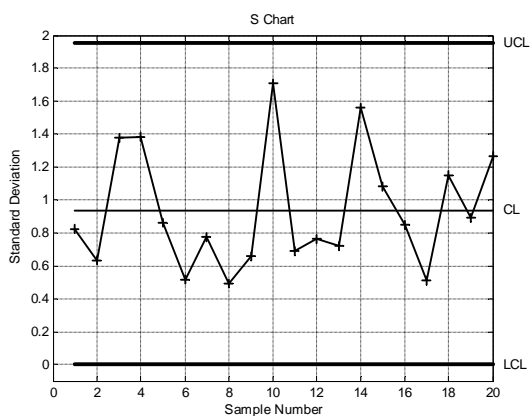


Рис. 18.9. Контрольна  $s$ -карта з параметрами за замовчуванням

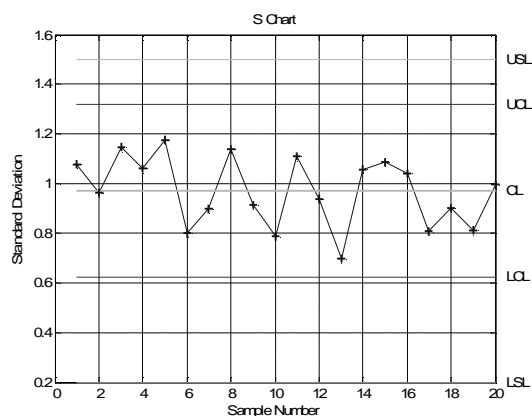


Рис. 18.10. Контрольна  $s$ -карта із заданими значеннями параметрів

У результативних параметрах і на рис. 18.10 зазначено результати розрахунків. Видно контрольні границі  $[LCL; UCL]$ , довірчий інтервал  $[LSL; USL]$ , точки (вибірки) 9, 13, 17, 20, що виходять за смугу контрольних границь, але що залишаються в довірчому інтервалі. Ламана – це контрольна карта середніх квадратичних відхилень. Суцільна лінія – це середній рівень даних. Значення  $h$  – це дескриптори (показники) на 8 елементів, з яких складається діаграма.

### **XBARPLOT** Контрольна карта середніх арифметичних значень ( $\bar{X}$ - контрольна карта)

Для випадкового процесу, заданого декількома реалізаціями, дана функція будує графік зміни в часі вибірових середніх.

*Синтаксис:*

```
xbarplot(DAT A)
xbarplot(DAT A,conf)
xbarplot(DAT A,conf,specs)
xbarplot(DAT A,conf,specs,'sigmaest')
[outlier,h] = xbarplot(...)
```

*Опис:*

Функція **xbarplot(DAT A)** дозволяє одержати контрольну карту середніх арифметичних значень ( $\bar{X}$ ) для вибірки *DATA*. Розмірність

матриці *DATA* дорівнює  $n \times m$ , де  $n$  – кількість вибірок (рядків *DATA*),  $m$  – об'єм вибірки (число стовпців *DATA*). Послідовність вибірок повинна відповідати порядку збору вихідних даних. На графіку контрольної карти відображаються вибіркові середні значення  $\bar{X}_i, i = 1, \dots, n$ ; центральна лінія, що відповідає загальному середньому арифметичному  $\bar{X}$ ; верхня *UCL* і нижня *LCL* контрольні границі. Границі проводяться на відстані трьох похибок середнього  $\sigma_{\bar{X}}$ , що відповідає 99,73%-ї довірчої ймовірності для нормального процесу.

Загальне середнє арифметичне розраховується за формулою:

$\bar{X} = \frac{1}{n} \sum_{i=1}^n x_i$ . Верхня й нижня контрольні границі визначаються як

$UCL = \bar{X} + 3\sigma_{\bar{X}}, \quad LCL = \bar{X} - 3\sigma_{\bar{X}}, \quad \sigma_{\bar{X}}$  – похибка середнього (стандартна похибка).

Якщо технологічний процес є статистично керованим, то за 1000 вибірок кількість точок, що вийшли за контрольні межі, не повинна перевищувати 3-х у випадковому порядку. Таким чином, за малої кількості вибірок вихід вибіркового середнього арифметичного  $\bar{X}_i$  за контрольні границі означає втрату процесом статистичної керованості.

Функція **xbarplot(DATA,conf)** призначена для побудови  $\bar{X}$ -контрольної карти для вибірки *DATA* із заданою довірчою ймовірністю *conf* для верхньої й нижньої контрольних границь. За замовчуванням значення *conf* приймається рівним 0.9973. Величина *conf* = 0.9973 відповідає інтервалу  $\pm 3\sigma_{\bar{X}}$  (три стандартних похибки):

```
>> norminv(1-(1-.9973)/2)
ans =
3.0000
```

Для розрахунку довірчої ймовірності *conf*, відповідної до значення  $\pm k\sigma_{\bar{X}}$  використовується вираження  $1 - 2*(1 - normcdf(k))$ . Наприклад, для  $k = 2$  значення *conf* дорівнює:

```
>> k=2;
>> conf=1-2*(1-normcdf(k))
conf =
```

0.9545

Функція **xbarplot(DAT A,conf,specs)** призначена для побудови контрольної карти середніх арифметичних значень із заданою довірчою ймовірністю розташування контрольних границь *conf* і границями допусків на параметр технологічного процесу *specs*. Границі допусків визначаються як вектор із двома елементами: *specs(1)* – нижня границя допуску, *specs(2)* – верхня границя допуску.

Функція **xbarplot(DAT A,conf,specs,'sigmaest')** дозволяє одержати  $\bar{X}_i$ -контрольну карту із заданою довірчою ймовірністю *conf*. Рядковий параметр *'sigmaest'* служить для визначення способу розрахунку точкової оцінки стандартного відхилення загального середнього. Можливі наступні способи розрахунку  $\sigma_{\bar{X}}$  наведено у табл. 18.1.

Таблиця 18.1

### Способи розрахунку $\sigma_{\bar{X}}$

Значення <i>'sigmaest'</i>	Спосіб розрахунку точкової оцінки стандартного відхилення
<i>'range', 'r'</i>	Для розрахунку точкової оцінки $\sigma_{\bar{X}}$ використовується середній вибірковий розмах. Застосовується для об'єму вибірки не більш 25 елементів
<i>'variance', 'v'</i>	Точкова оцінка $\sigma_{\bar{X}}$ приймається рівною кореню квадратному з дисперсії, розрахованої за всіма вибірками
<i>'s'</i>	Точкова оцінка $\sigma_{\bar{X}}$ приймається рівною середньому арифметичному середніх квадратичних відхилень вибірок

У функції **[outlier,h] = xbarplot(DAT A,conf,specs)** результативними параметрами функції є: *outlier* – вектор номерів вибірок (рядків матриці *DATA*), середні арифметичні значення яких вийшли за контрольні границі, *h* – вектор показників на об'єкти графіка контрольної карти.

Приклади:

1.  $\bar{X}$  - контрольна карта.

```
>> DATA=normrnd(0,1,20,4)
DATA =
    0.8675    0.2751    0.8035   -0.0636
    0.3608    0.8262    0.5809    0.6453
    1.0195    0.0161   -0.6216    2.4152
    0.9344    1.3692    0.8095   -0.4021
    1.2286    0.4169    1.9288    0.9141
   -0.2495    0.0687    0.3961   -0.1360
```

-0.0804	-0.9793	1.7786	-1.7713	-0.7076	0.2942	-0.8614	1.3142
0.7493	-0.1043	-0.9381	0.0596	-0.5938	0.4726	2.4319	0.3224
-1.7920	0.1278	-0.9167	-0.7602	-0.2623	1.7827	-0.8405	-0.4765
1.2132	0.0625	0.3760	-1.6909	1.2428	0.2606	0.2805	0.0762
-0.0605	0.3716	0.9098	1.1037	-1.5489	1.5107	0.8204	-0.1051
-0.3925	-0.1040	0.1542	1.4625	-0.3868	0.3170	1.2278	1.4170
0.6095	-0.6968	-0.2023	0.2362	<b>&gt;&gt;xbarplot(DATA)</b>			
0.6436	-0.3868	1.4887	-1.0977	<b>&gt;&gt; grid on</b>			

Матриця *DATA* містить 20 виборок по 4 виміри кожна. Вони розподілені за стандартизованим нормальним законом. На рис. 18.11 зображена контрольна карта середніх за вибірками, контрольні границі, а також загальне середнє (суцільна горизонтальна лінія).

Його можна обчислити так

```
>> mean(mean(DATA))
ans =
    0.0591
```

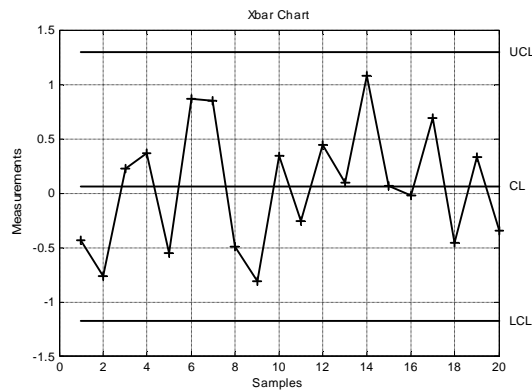


Рис. 18.11. Контрольна карта середніх з контрольними границями

2.  $\bar{X}$  - контрольна карта з контрольними границями рівними  $\pm 2\sigma_{\bar{X}}$  і границями допусків технологічного параметра  $[-1; 1]$ .

```
>> DATA=normrnd(0,1,20,4);
>> k=2;
>> conf =1-2*(1-normcdf(k));
>> specs=[-1 1];
>> xbarplot(DATA,conf,specs), grid on, box on
```

На рис. 18.12 зображена  $\bar{X}$  - контрольна карта 20 вибірок з контрольними границями  $[LCL; UCL]$  і границями допусків. Маркування осей і титулу проводить програма.

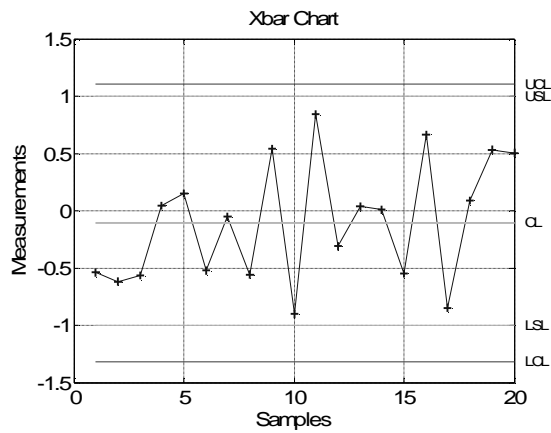


Рис. 18.12.  $\bar{X}$  - контрольна карта з контрольними границями й границями допусків

3.  $\bar{X}$  - контрольна карта з контрольними границями рівними  $\pm 1,5\sigma_{\bar{X}}$  і границями допусків технологічного параметра в інтервалі  $[-1; 1]$ . У якості результативних параметрів виступають: *outlier* – вектор номерів вибірок, що вийшли за контрольні границі, і *h* – вектор покажчиків об'єктів графіка.

```
>> DATA=normrnd(0,1,20,4);
>> k=1.5;
>> conf =1-2*(1-normcdf(k));
>> specs=[-1 1];
>> [outlier,h] = xbar-
plot(DATA,conf,specs)
>> grid on
```

```
outlier =      155.0018
              3      156.0018
              4      157.0018
              5      158.0018
              7      159.0018
h =           170.0017
154.0052      171.0017
```

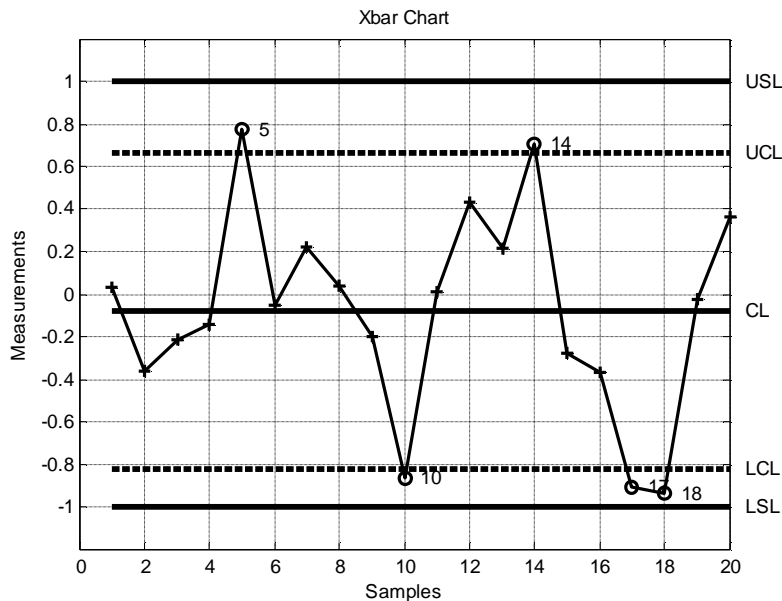


Рис. 18.13.  $\bar{X}$  - контрольна карта вибірки

На рис. 18.13 видно, що п'ять точок (вибірок) 5, 10, 14, 17 і 18 виходять за рамки контрольних границь.

4. Уводимо статистичні дані з файла parts.mat і будуємо  $\bar{X}$ -контрольну карту з відзначеними точками, що випадають за межі допусків технологічного параметра. Цей файл містить числову матрицю розміром  $36 \times 4$  без позначень.

```
>> load parts; % завантажуюмо базу даних
>> outl=xbarplot(runout,0.999,[-0.5 0.5]); grid on % графік стабільності
>> if isempty(outl),
    disp('Значень, що випадають, немає')
else
    fprintf('Випадають точки з номерами:')
    fprintf(' %d',outl)
    fprintf('\n')
end
>> set(gcf,'Currentaxes','Fontname','Arial Cyr','FontSize',14)
>> title('\bфГрафік середнього');
>> xlabel('\bft') % мітка осі OX
>> ylabel('\bfm_x') % мітка осі OY
```

На рис. 18.14 видно, що лише дві точки 21 і 25 (середні чотирьох вимірів, наведених в 21 і 25 рядках матриці даних) випадають за межі допусків технологічного процесу.

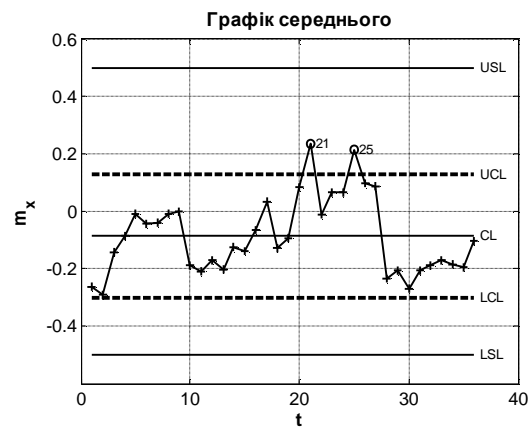


Рис.18.14.  $\bar{X}$  - контрольна карта даних файла parts.mat

## 19. Планування експерименту

Цей розділ математичної статистики вивчає раціональну організацію вимірів з випадковими похибками. Звичайно розглядається наступна схема планування експерименту. З випадковими похибками вимірюється функція  $f(\theta, x)$ , що залежить від невідомих параметрів (вектор  $\theta$ ) і від змінних  $x$ , які за вибором експериментатора можуть приймати значення з деякої припустимої множини  $X$  (це, так званий, активний експеримент). Метою експерименту є оцінка всіх, або деяких параметрів  $\theta$ , або їх функцій, або перевірка деяких гіпотез про параметри  $\theta$ . Виходячи з мети експерименту, формулюється критерій оптимальності плану експерименту. Під планом експерименту розуміється сукупність значень, що задаються змінними  $x$  в експерименті. Як правило, значення  $x$  у плані (схемі експерименту) можуть бути закодовані у вигляді цілих чисел, кількість рівнів яких (фіксованих значень) обмежена. Різні змінні (керуючі змінні) у теорії планування експерименту називаються "факторами". У трифакторному дворівневному плані три фактори  $x_1, x_2, x_3$  змінюються на двох рівнях варіювання. Як підкреслив засновник цього напрямку Р. Фішер, раціональне планування експерименту дає не менш істотний виграш у точності оцінок, ніж оптимальна обробка результатів вимірів. Р. Фішер запропонував схеми повних факторних експериментів, які містять всі можливі комбінації рівнів всіх факторів. Ці схеми найбільш інформативні, містять найбільшу кількість парних порівнянь варіантів досліду, що дозволяє оцінити параметри математичної моделі незалежно один від одного (така корисна властивість називається ортогональністю схеми досліду). На жаль, повні факторні схеми вимагають великого числа варіантів в порівнянні з іншими прийомами планування експерименту, що розглянуті нижче.

Матеріал цього розділу розташовано за темами:

19.1. Повні факторні експерименти (ПФЕ) ..... 403

19.2. Дробові факторні експерименти (ДФЕ) .....	407
19.3. Композиційні плани експерименту .....	414
19.4. D-оптимальні плани експерименту .....	420
19.5. Латинські квадрати, куби, гіперкуби .....	438

## 19.1. Повні факторні експерименти (ПФЕ)

<b>FF2N</b> Генерування матриці повного факторного експерименту для факторів, що варіюють на 2-х рівнях .....	403
<b>FULLFACT</b> Генерування матриці повного факторного експерименту для довільного числа факторів й кількості їх рівнів .....	404

### **FF2N** Генерування матриці повного факторного експерименту для факторів, що варіюють на 2-х рівнях

Створюється матриця повного факторного експерименту з двома рівнями варіювання факторів.

*Синтаксис:*

**X = ff2n(n)**

*Опис:*

Функція **X = ff2n(n)** дозволяє одержати матрицю плану  $X$  повного факторного експерименту (ПФЕ) за умови, що всі фактори, що входять у план, змінюються на двох рівнях. Вхідний аргумент  $n$  – кількість факторів в експерименті. Розмірність матриці плану експерименту дорівнює  $2^n \times n$ , де  $2^n$  – число рядків або число варіантів в експерименті,  $n$  – число стовпців або число факторів. Елементи матриці плану експерименту виражаються в кодованій формі і дорівнюють або 0, або 1, де 0 відповідає мінімальному значенню фактору, а 1 – максимальному значенню фактору. Матриця плану експерименту  $X$ , що генерується функцією **ff2n**, не рендомізована (рендомізація – статистична процедура, яка призначена для страхування результатів аналізу від систематичних впливів не врахованих в моделі сторонніх факторів; для схеми

планування експерименту рендомізація полягає у спеціальному порядку розташування варіантів досліду у просторі і часу). Схеми ПФЕ являє собою послідовний перебір усіх можливих комбінацій значень факторів.

*Приклади:*

1. Матриця плану повного трифакторного експерименту  $2 \times 2 \times 2$ .

```
>> n=3;
>> X = ff2n(n);
X =
```

0	0	0	0	1	1	1	1	0
0	0	1	1	0	0	1	1	1
0	1	0	1	0	1	1	0	1

Матриця  $X$  дає номери рядків в бінарному коді від 0 до  $2^n - 1$  (рядки 1, 2, ..., 7).

1. Графічно матрицю планування повного факторного експерименту  $2 \times 2 \times 2$  для трьох факторів (рис. 19.1) можна представити в такий спосіб:

```
>> n=3;
>> X = ff2n(n)
>> x= X (:,1);
>> y= X (:,2);
>> z= X (:,3);
>> plot3(x,y,z,'o')
>> grid on, box on
```

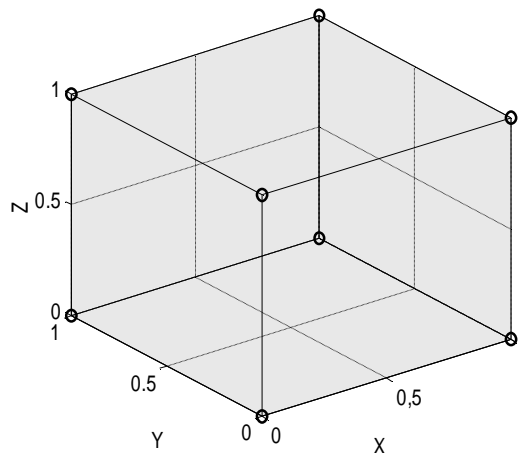


Рис. 19.1. Графічне зображення ПФЕ  $2 \times 2 \times 2$

### **FULLFACT** Генерування матриці повного факторного експерименту для довільного числа факторів й кількості їх рівнів

Будується матриця повного факторного експерименту (ПФЕ) із вказаною кількістю рівнів і факторів.

*Синтаксис:*

**design = fullfact(levels)**

*Опис:*

Функція **design = fullfact(levels)** призначена для генерування матриці повного факторного експерименту *design* за довільного числа факторів і заданої кількості рівнів кожного фактора. Кількість факторів і число рівнів визначається вектором *levels* у форматі *levels = [n m k ...]*, де *n, m, k, ...* – число рівнів першого, другого, третього факторів тощо. Елементи матриці плану експерименту представляються кодованими значеннями і відповідно дорівнюють *1..n, 1..m, 1..k* тощо, де 1 відповідає мінімальному значенню фактора й далі за зростанням. Матриця плану експерименту не рендомізована (рендомізація – статистична процедура, що призначена для страхування результатів аналізу від систематичних впливів не врахованих в моделі сторонніх факторів) і являє собою послідовний перебір усіх можливих комбінацій значень факторів.

*Приклади:*

1. Генерування матриці плану експерименту для трьох факторів, що варіюють на 3 рівнях ( $3 \times 3 \times 3$ ).

```
>> levels=[3 3 3]
levels =
     3     3     3
>> design = fullfact(levels)
design =
     1     1     1
     2     1     1
     3     1     1
     1     2     1
     2     2     1
     3     2     1
     1     3     1
     2     3     1
     1     1     2
     2     1     2
     3     1     2
     1     2     2
     2     2     2
     3     2     2
     1     3     2
     2     3     2
     3     3     2
     1     1     3
     2     1     3
     3     1     3
     1     2     3
     2     2     3
     3     2     3
     1     3     3
     2     3     3
     3     3     3
```

2. Графічно матрицю планування експерименту  $3 \times 3 \times 3$  для трьох факторів з трьома рівнями варіювання можна представити у такий спосіб (рис. 19.2):

```
>> levels=[3 3 3];
>> design = fullfact(levels);
>> x=design(:,1);
>> y=design(:,2);
>> z=design(:,3);
>> plot3(x,y,z,'o')
>> box on
>> grid on
```

У подальшому всі варіанти схеми  $3 \times 3 \times 3$ , що зображені на рис. 19.2, буде доцільно сгрупувати на стандартні множини: вершини куба (8 варіантів), середини ребер (12 варіантів), центри граней (6 варіантів), центр куба (1 варіант) – усього 27 варіантів.

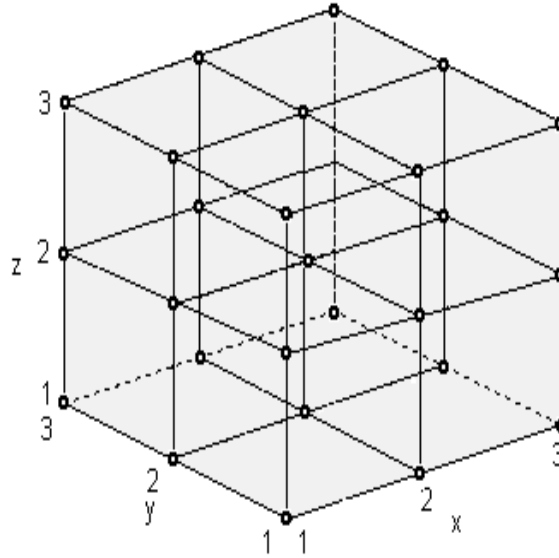


Рис. 19.2. Графічне зображення ПФЕ  $3 \times 3 \times 3$

3. Генерування матриці плану експерименту для трьох факторів, які варіюють на  $2 \times 4 \times 3$  рівнях відповідно:

```
>> levels=[2 4 3]
levels =
    2  4  3
>> design = fullfact(levels)
design =
    1  1  1      |      1  1  2      |      1  1  3
    2  1  1      |      2  1  2      |      2  1  3
    1  2  1      |      1  2  2      |      1  2  3
    2  2  1      |      2  2  2      |      2  2  3
    1  3  1      |      1  3  2      |      1  3  3
    2  3  1      |      2  3  2      |      2  3  3
    1  4  1      |      1  4  2      |      1  4  3
    2  4  1      |      2  4  2      |      2  4  3
```

4. Графічно матрицю планування експерименту  $2 \times 4 \times 3$  для трьох факторів із рівнями 2, 4, 3 можна представити у такий спосіб (рис. 19.3):

```
>> levels=[2 4 3];
>> design = fullfact(levels);
>> x=design(:,1);
>> y=design(:,2);
```

```
>> z=design(:,3);
>> plot3(x,y,z,'o')
>> box on
>> grid on
```

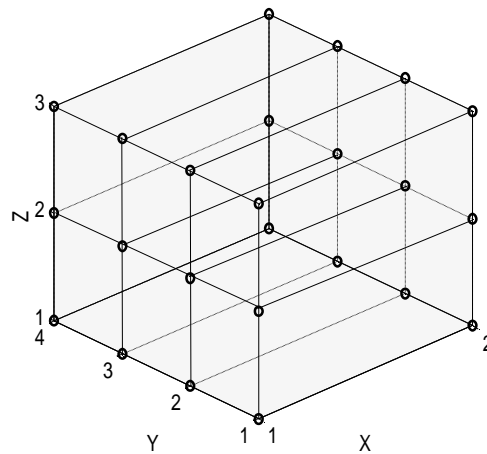


Рис. 19.3. Графічне зображення ПФЕ  $2^{4-3}$

## 19.2. Дробові факторні експерименти (ДФЕ)

<b>FRACFACT</b>	Генерування матриці дробового факторного експерименту для довільного числа факторів з двома рівнями варіювання .....	407
<b>HADAMARD</b>	Генерування матриці Адамара .....	412

**FRACFACT** Генерування матриці дробового факторного експерименту для довільного числа факторів з двома рівнями варіювання

Будується дробовий факторний експеримент ДФЕ  $2^{n-m}$ , який є підвибіркою варіантів повного факторного експерименту ПФЕ  $2^n$ .

*Синтаксис:*

```
x = fracfact('gen')
[x,conf] = fracfact('gen')
```

Опис:

Функція `x = fracfact('gen')` призначена для генерування матриці  $x$  плану дробового факторного експерименту згідно з генератором *gen*. Аргумент *gen* має бути текстовим рядком, що містить 'слова', розділені пропусками. Значення факторів  $x$  у матриці плану експерименту дорівнюють 1 і -1, де 1 відповідає максимальному рівню фактора, а -1 – мінімальному рівню. Генератор '*gen*' є символьною змінною, що складається зі 'слів', розділених пропусками. 'Слово' може складатися з однієї й більше літер латинського алфавіту. У першу чергу задаються односимвольні 'слова', що кодують фактори в матриці повного факторного експерименту. Ця матриця є основою дробового факторного плану. Правила визначення значень інших факторів у матриці дробового плану задаються у вигляді комбінацій символів, що кодують фактори повного факторного експерименту. Для цих факторів значення визначаються як добутки значень кодованих змінних, відповідних до першої групи факторів. Наприклад, для генератора '*gen*'='a b ab' перші два фактори  $a$  і  $b$  утворюють матрицю повного факторного плану вигляду  $2 \times 2$ :

```
>> x = ff2n(2)
x =
    0    0
    0    1
    1    0
    1    1
```

'Слово' *ab* визначає значення третього фактора як порядковий добуток  $a$  і  $b$ .

```
>> x = fracfact('a b ab')
x =
   -1   -1    1
   -1    1   -1
    1   -1   -1
    1    1    1
```

Для факторів другої групи комбінація символів у 'слові' визначає вид змішування їх головних ефектів. Тобто, у попередньому прикладі головний ефект, наприклад, третього фактора  $\beta_3$  буде змішаний з

парним ефектом взаємодії перших двох факторів  $\beta_{12}$ :  $\beta_3 \Rightarrow \beta_3 + \beta_{12}$ . Те ж саме стосується головних ефектів інших факторів. Для цього прикладу матриця  $x$  буде напівреплікою з повного факторного експерименту (ПФЕ)  $2^3$ , її позначення – ДФЕ  $2^{3-1}$ .

Матриця дробового факторного плану  $x$  є частиною матриці повного факторного плану експерименту, утвореного всіма факторами. Число рядків матриці плану  $x$  дорівнює рівно  $2^n$ , де  $n$  – кількість односимвольних 'слів' у генераторі. Кількість стовпців у матриці плану повинне бути дорівнювати числу всіх 'слів' у генераторі.

Матриця  $x$  – це дробова репліка з повного факторного експерименту (ПФЕ)  $2^n$ . Якщо в  $gen$  є  $m$  'слів', сформованих із перших  $n$  літер алфавіту, то в  $x$  буде не  $2^n$ , а лише  $2^{n-m}$  рядків та  $n$  стовпців. Якщо  $F$  – план ПФЕ  $2^n$ , створений функцією **ff2n(n)**, то  $j$ -й стовпець  $x$  виходить множенням всіх стовпців  $F$ , які відповідають буквам, що входять в  $j$ -е слово  $gen$ .

Функція **[x,conf] = fracfact('gen')** додатково повертає матрицю  $conf$  як таблицю, що показує спосіб змішування головних ефектів факторів і їх парних взаємодій.

*Приклади:*

1. Метою експерименту є оцінка величин головних (лінійних) ефектів 4-х факторів на функцію відгуку за проведення восьми спостережень. Кожне спостереження проводиться без повторень.

Проведення повного факторного експерименту потребує  $2^4 = 16$  варіантів. Проте, якщо на підставі апріорних даних припустити незначимість потрійних взаємодій, то можна оцінити величини головних ефектів чотирьох факторів лише у восьми спостереженнях ( $2^{4-1} = 8$ ).

Матриця планування дробового факторного експерименту  $x$  у цьому випадку прийме вигляд:

```

>> [x,conf] = fracfact('a b c abc')
x =
  -1  -1  -1  -1
  -1  -1   1   1
  -1   1  -1   1

conf =
  'Term'   'Generator' 'Confounding'
  'X1'     'a'         'X1'
  'X2'     'b'         'X2'
  'X3'     'c'         'X3'
  'X4'     'abc'       'X4'

```

-1	1	1	-1
1	-1	-1	1
1	-1	1	-1
1	1	-1	-1
1	1	1	1

'X1*X2'	'ab'	'X1*X2 + X3*X4'
'X1*X3'	'ac'	'X1*X3 + X2*X4'
'X1*X4'	'bc'	'X1*X4 + X2*X3'
'X2*X3'	'bc'	'X1*X4 + X2*X3'
'X2*X4'	'ac'	'X1*X3 + X2*X4'
'X3*X4'	'ab'	'X1*X2 + X3*X4'

Перші 3 стовпці матриці  $x$  складають матрицю повного факторного експерименту для трьох факторів  $a$ ,  $b$ ,  $c$ . Четвертий стовпець отриманий як добуток значень перших трьох факторів. Матриця  $conf$  показує, що таке планування експерименту дозволяє оцінити незмішані головні ефекти для всіх чотирьох факторів. Парні взаємодії змішано один з одним. Наприклад, ефект від парної взаємодії першого й другого факторів змішаний з ефектом парної взаємодії третього й четвертого факторів:  $X_1 * X_2 + X_3 * X_4$ . Це не дозволяє роздільно оцінити ефекти взаємодій.

У випадку, якщо після проведення зазначених восьми спостережень з'ясується, що сумарний ефект від взаємодії  $X_1 * X_2 + X_3 * X_4$  значущий, то доцільно визначити, яка пара взаємодій значима:  $X_1 * X_2$  або  $X_3 * X_4$ . Для цього дробову напіврепліку з перших восьми варіантів можна доповнити другою напівреплікою до повного факторного експерименту. Матриця другої репліки задається тим же генератором, але зі зворотним знаком для четвертого фактора:

```

>> fracfact('a b c -abc')
ans =
  -1  -1  -1   1
  -1  -1   1  -1
  -1  -1   1  -1

```

-1	1	-1	-1
-1	1	1	1
1	-1	-1	-1

1	-1	1	1
1	1	-1	1
1	1	1	-1

2. Потрібно оцінити величини лінійних ефектів восьми факторів. Оскільки у більшості випадків парні ефекти взаємодії важливіші ефектів взаємодій вищих порядків, то слід так скласти дробову репліку, щоб лінійні ефекти не були змішані з ними за мінімальної кількості варіантів. Проведення повного факторного експерименту вимагало  $2^8=256$

спостережень. Використання дробової репліки повного факторного експерименту дозволяє за 16 варіантів одержати оцінки головних ефектів 8-и факторів, не змішаних з парними взаємодіями. 1 / 16 репліку для названих умов можна одержати з використанням наступного генератора:

```
>> [x, conf] = fracfact('a b c d abc acd abd bcd')
```

```
x =
```

-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	1	1	1	1	-1
-1	-1	-1	1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	1
-1	-1	1	-1	1	1	-1	1	1	-1	1	-1	-1	-1	1	1	1
-1	-1	1	1	1	-1	1	1	-1	1	1	-1	-1	1	-1	-1	1
-1	1	-1	-1	1	-1	1	1	1	1	-1	-1	-1	-1	1	1	-1
-1	1	-1	1	1	1	-1	-1	-1	1	1	-1	1	-1	-1	-1	-1
-1	1	1	-1	-1	1	1	1	-1	1	1	1	1	1	1	1	-1
-1	1	1	1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1

```
conf =
```

Term	Generator	Confounding	X3	c	X3	X7	abd	X7
X1	a	X1	X4	d	X4	X8	bcd	X8
X2	b	X2	X5	abc	X5			
			X6	acd	X6			

X1*X2	ab	X1*X2+X3*X5+X4*X7+X6*X8	X3*X5	ab	X1*X2+X3*X5+X4*X7+X6*X8
X1*X3	ac	X1*X3+X2*X5+X4*X6+X7*X8	X3*X6	ad	X1*X4+X2*X7+X3*X6+X5*X8
X1*X4	ad	X1*X4+X2*X7+X3*X6+X5*X8	X3*X7	abcd	X1*X8+X2*X6+X3*X7+X4*X5
X1*X5	bc	X1*X5+X2*X3+X4*X8+X6*X7	X3*X8	bd	X1*X7+X2*X4+X3*X8+X5*X6
X1*X6	cd	X1*X6+X2*X8+X3*X4+X5*X7	X4*X5	abcd	X1*X8+X2*X6+X3*X7+X4*X5
X1*X7	bd	X1*X7+X2*X4+X3*X8+X5*X6	X4*X6	ac	X1*X3+X2*X5+X4*X6+X7*X8
X1*X8	abcd	X1*X8+X2*X6+X3*X7+X4*X5	X4*X7	ab	X1*X2+X3*X5+X4*X7+X6*X8
X2*X3	bc	X1*X5+X2*X3+X4*X8+X6*X7	X4*X8	bc	X1*X5+X2*X3+X4*X8+X6*X7
X2*X4	bd	X1*X7+X2*X4+X3*X8+X5*X6	X5*X6	bd	X1*X7+X2*X4+X3*X8+X5*X6
X2*X5	ac	X1*X3+X2*X5+X4*X6+X7*X8	X5*X7	cd	X1*X6+X2*X8+X3*X4+X5*X7
X2*X6	abcd	X1*X8+X2*X6+X3*X7+X4*X5	X5*X8	ad	X1*X4+X2*X7+X3*X6+X5*X8
X2*X7	ad	X1*X4+X2*X7+X3*X6+X5*X8	X6*X7	bc	X1*X5+X2*X3+X4*X8+X6*X7
X2*X8	cd	X1*X6+X2*X8+X3*X4+X5*X7	X6*X8	ab	X1*X2+X3*X5+X4*X7+X6*X8
X3*X4	cd	X1*X6+X2*X8+X3*X4+X5*X7	X7*X8	ac	X1*X3+X2*X5+X4*X6+X7*X8

Матриця *conf* показує, що лінійні ефекти восьми факторів не змішані з парними взаємодіями. Парні взаємодії змішані один з одним. Застосування інших генераторів не завжди дозволяє досягти такого гарного змішування.

3. Графічне порівняння планів дробового й повного факторних експериментів для трьох факторів з двома рівнями варіювання.

```

>> n=3;
>> X=fracfact('a b ab')
X =
  -1  -1  1
  -1   1 -1
   1  -1 -1
   1   1  1
>> X=(X +1)/2
X =
  0  0  1
  0  1  0
  1  0  0
  1  1  1
>> x=X(:,1);
>> y=X(:,2);
>> z=X(:,3);
>> subplot(1,2,1)
>> plot3(x,y,z,'o')

>> grid on
>> box on
>> Y=ff2n(n)
Y =
  1  0  1
  1  1  0
  1  1  1
  0  0  0
  0  0  1
  0  1  0
  0  1  1
  1  0  0
>> x1=Y(:,1);
>> y1=Y(:,2);
>> z1=Y(:,3);
>> subplot(1,2,2)
>> plot3(x1,y1,z1,'dr')
>> box on, grid on

```

На рис. 19.4 зображена одна напіврепліка ДФЕ  $2^{3-1}$  – "тетраедр". Поряд, на рис. 19.5, для прорівняння зображена повна схема ПФЕ  $2^3$  – "куб".

Дробові факторні схеми використовуються, в основному, для рендомізації повного факторного експерименту – близькі у просторі або у часі варіанти повного плану повинні складати рівноправні блоки (дробові репліки).

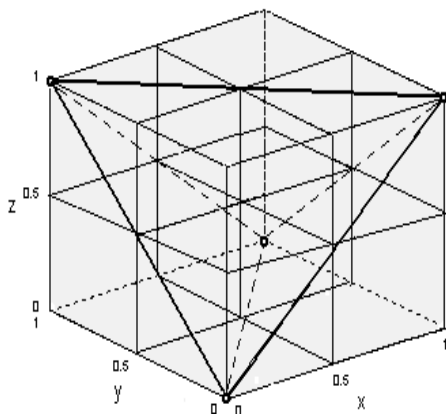
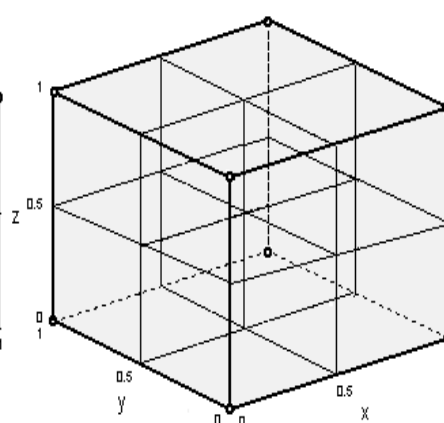


Рис. 19.4. План дробового експерименту  $2^{3-1}$



19.5. План повного експерименту  $2^3$

### **HADAMARD** Генерування матриці Адамара

Матриця Адамара – це ортогональна матриця, елементами якої є числа 1 і -1. Для  $n > 2$  існують матриці Адамара лише порядку, кратного 4. Для матриці Адамара повинне виконуватися умова  $H' * H = n * I$ , де  $size(H) = n \times n$ ,  $I = eye(n, n)$ .

Матриця Адамара розмірністю  $n \times n$ , при  $n > 2$ , існує, якщо  $rem(n, 4) = 0$ . Для вхідного аргументу  $n$  повинне виконуватися умова –  $n$ ,  $n/12$ ,  $n/20$  є ступенем 2.

*Синтаксис:*

**H = hadamard(n)**

*Опис:*

Функція **H = hadamard(n)** – призначена для генерування матриці Адамара  $n$ -го порядку.

*Приклади:*

1. Генерування матриці Адамара з розмірністю  $4 \times 4$ .

```
>> H = hadamard(4)
```

```
H =
    1    1    1    1
    1   -1    1   -1
    1    1   -1   -1
    1   -1   -1    1
```

Перевірка виконання умови  $H' * H = n * I$ ,

```
>> n=4
```

```
n =
```

```
    4
```

```
>> H = hadamard(n);
```

```
>> I = eye(n,n)
```

```
I =
    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1
```

```
>> I*4==H'*H
```

```
ans =
```

```
    1    1    1    1
    1    1    1    1
    1    1    1    1
    1    1    1    1
```

2. Зробимо контурний графік матриці Адамара (рис. 19.6).

```
>> [x,y]=meshgrid([1:4]); % аргументи
```

```
>> contour(x,y,H) % контурний графік
```

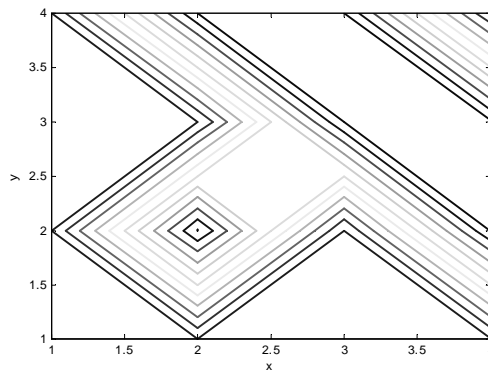


Рис. 19.6. Контурний графік матриці Адамара 4-го порядку

### 3. Контурний графік матриці Адамара 16-го порядку (рис. 19.7).

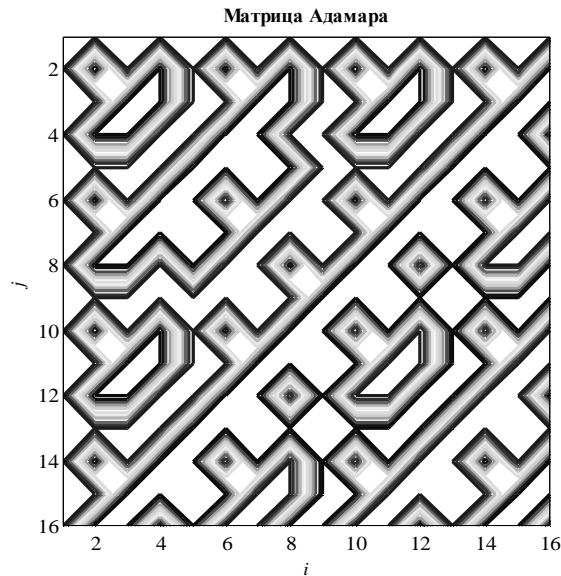


Рис. 19.7. Контурний графік матриці Адамара 16-го порядку

Цей графік було побудовано операторами:

```
>> clear all % очищаємо пам'ять
>> n=16; % порядок матриці
>> H=hadamard(n); % матриця Адамара
>> [x,y]=meshgrid([1:n]); % аргументи
>> contour(x,y,H) % контурний графік
>> set(get(gcf,'Currentaxes'),'Fontname','Times New Roman Cyr','FontSize',14)
>> title('\bfМатрицз Адамара') % заголовок
>> xlabel('\bfi') % мітка осі ОХ
>> ylabel('\bfj') % мітка осі ОУ
>> axis ij % початок координат – у лівому верхньому куті
>> axis equal % однакові масштаби по осях координат
>> xlim([1 n]) % границі по осі ОХ
>> ylim([1 n]) % границі по осі ОУ
```

## 19.3. Композиційні плани експерименту

<b>CCDESIGN</b> Генерування матриці центрального композиційного плану .....	415
<b>BBDESIGN</b> Генерування матриці плану Бокса-Бенкена .....	418

## **CCDESIGN** Генерування матриці центрального композиційного плану

Будується центральний композиційний план експерименту Дж. Бокса. Плани Дж. Бокса складаються зі стандартних множин варіантів (точок у факторному просторі). Розглянемо стандартні множини точок (блоки), з яких складається ПФЕ  $3 \times 3 \times 3$ . Блок 3 (ядро схеми) – "вершини куба", тобто варіанти, де всі 3 фактори приймають мінімальні або максимальні значення (це схема  $2 \times 2 \times 2$  – 8 варіантів). Блок 2 – "кубооктаедр", "середини ребер", коли один з факторів зафіксований на середньому рівні, а інші 2 приймають мінімальні або максимальні значення (таких варіантів  $3 \times 2 \times 2 = 12$ ). Блок 1 – "зіркові точки", "середини граней", коли два фактори зафіксовані на середніх рівнях, а один з факторів приймає мінімальне або максимальне значення (таких варіантів  $3 \times 2 = 6$ ). Блок 0 – центральний варіант, коли всі фактори зафіксовані на середніх рівнях (таких варіантів 1).

Композиційні схеми складаються з трьох блоків з різними рівнями варіювання: блок 3 – "вершини куба"  $2^n$ , блок 1 – "зіркові точки" з більшим розмахом варіювання факторів і кілька центральних точок. Обираючи різні діапазони варіювання у блоці "зіркових точок" і різну кількість центральних варіантів, одержують композиційні плани з різними властивостями: можна досягти ортогональності плану (що дозволяє оцінювати параметри моделі незалежно один від одного), можна забезпечити рототабельність, тобто однакову точність прогнозів у будь-якому напрямку (якщо фактори змінюються у певній пропорції), можна наблизитися до  $D$ -оптимальності, коли мінімізуються похибки прогнозів. Для  $n > 3$  ядро схеми можна скоротити удвічі, використовуючи замість ПФЕ  $2^n$  напіврепліку  $2^{n-1}$  і навіть чвертьрепліку  $2^{n-2}$ .

Якщо з повної схеми  $3 \times 3 \times 3$  вилучити блоки 3 і 1 (композиційну схему), то одержимо план Бокса-Бенкена ("кубооктаедр" плюс центральні варіанти).

Плани Бокса майже вдвічі менші за повні факторні схеми за однакової з ними інформативності.

Синтаксис:

**D = ccdesign(nfactors)**

**D = ccdesign(nfactors,'pname1',pvalue1,'pname2',pvalue2,...)**

**[D,blk] = ccdesign(...)**

Опис:

Функція **D = ccdesign(nfactors)** призначена для генерування матриці *D* центрального композиційного плану для *nfactors* факторів. Розмірність матриці плану експерименту *D* дорівнює  $n \times nfactors$ , де *n* – кількість точок плану. Значення факторів нормалізовані таким чином, щоб вершини куба плану були рівні -1 і 1.

Функція **[D,blk] = ccdesign(nfactors)** повертає вектор номерів *blk* варіантів у плані експерименту. Блоки дозволяють згрупувати варіанти таким чином, щоб мінімізувати вплив змін зовнішніх умов на значення оцінюваних параметрів (рендомізація).

Функція **[...] = ccdesign(nfactors,'pname1',pvalue1,'pname2',pvalue2,...)** дозволяє задати наступні додаткові параметри '*pname*' і їх значення *pvalue* (табл. 19.1).

Додаткові аргументи задаються у вигляді пари: 'назва параметра', 'значення параметра'.

Таблиця 19.1

### Значення *pvalue* параметра '*pname*'

Параметр ' <i>pname</i> '	Його значення <i>pvalue</i>	
'center'	Число центральних точок у плані:	
	Ціле число	Визначає число центральних точок у плані
	'uniform'	Число центральних точок вибирається для забезпечення рівної точності в області проведення експерименту
	'orthogonal'	Число центральних точок вибирається для забезпечення ортогональності плану. Ухвалюється як значення за замовчуванням
'fraction'	Вид дробової репліки від повного факторного плану. Задається у вигляді ступеня 1/2. Наприклад:	
	0	Повний план
	1	1/2 репліка
	2	1/4 репліка
'type'	Вид центрального композиційного плану. Можливі значення ' <i>inscribed</i> ', ' <i>circumscribed</i> ', ' <i>faced</i> '	

'blocksize'

Максимальна кількість точок у блоці плану

Приклад:

1. Генерування матриці центрального композиційного плану для 5 факторів розділеного на блоки. Ядро – напіврепліка від ПФЕ 2<sup>5</sup> (16 варіантів). Подвійний діапазон розмаху варіювання у блоці зіркових точок і число центральних варіантів (6) задане з метою забезпечення ортогональності плану. Вид центрального композиційного плану – 'inscribed'.

```
>> [D,blk]=ccdesign(5,'center','uniform','fraction',1,'type','inscribed')
```

D =

-0.5000	-0.5000	-0.5000	-0.5000	0.5000	0.5000	-0.5000	-0.5000	-0.5000	-0.5000
0.5000	-0.5000	-0.5000	0.5000	0.5000	-0.5000	-0.5000	-0.5000	0.5000	-0.5000
0.5000	-0.5000	0.5000	-0.5000	0.5000	-0.5000	-0.5000	0.5000	-0.5000	-0.5000
0.5000	-0.5000	0.5000	0.5000	-0.5000	-0.5000	-0.5000	0.5000	0.5000	0.5000

-0.5000	0.5000	-0.5000	-0.5000	-0.5000	0.5000	0.5000	-0.5000	-0.5000	0.5000
-0.5000	0.5000	-0.5000	0.5000	0.5000	0.5000	0.5000	-0.5000	0.5000	-0.5000
-0.5000	0.5000	0.5000	-0.5000	0.5000	0.5000	0.5000	0.5000	-0.5000	-0.5000
-0.5000	0.5000	0.5000	0.5000	-0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
-1.0000	0	0	0	0	0	0	0	0	-1.0000
1.0000	0	0	0	0	0	0	0	0	1.0000
0	-1.0000	0	0	0	0	0	0	0	0
0	1.0000	0	0	0	0	0	0	0	0
0	0	-1.0000	0	0	0	0	0	0	0
0	0	1.0000	0	0	0	0	0	0	0
0	0	0	-1.0000	0	0	0	0	0	0
0	0	0	1.0000	0	0	0	0	0	0

blk =

1	1	2	2
1	1	2	2
1	1	2	1
1	1	2	1
1	1	2	1
1	1	2	2
1	1	2	2
1	1	2	2

Цей план практично ортогонален відносно всіх членів квадратичної моделі. Невеличка кореляція буде лише між квадратичними ефектами  $R(x_{11}, x_{22}) = -0.0667$ , що близько до нуля.

2. Графічне представлення 3-х видів композиційного плану.

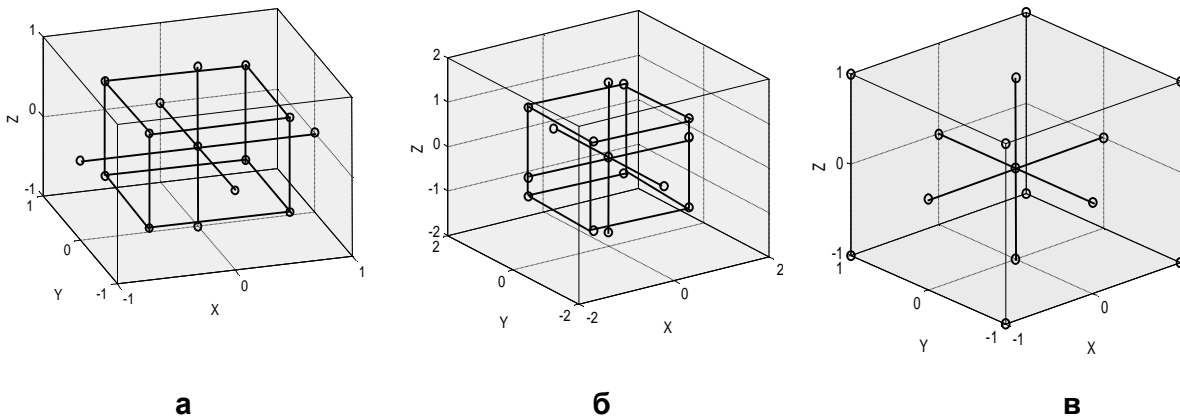


Рис. 19.8. Центральні композиційні плани типу (а) *inscribed*, (б) *circumscribed*, (в) *faced*

На рис. 19.8 зображено 3 види центрального композиційного плану для 3 факторів з різними розмахами варіювання в блоці зіркових точок (і з різними властивостями), які були сгенеровані операторами:

<pre>&gt;&gt; nfactors=3; &gt;&gt; D = ccdesign(nfactors, 'type', 'inscribed'); &gt;&gt; x= D(:,1); &gt;&gt; y= D (:,2); &gt;&gt; z= D (:,3); &gt;&gt; plot3(x,y,z,'o'), grid on &gt;&gt; box on</pre>	<pre>&gt;&gt; D1 = ccdesign(nfactors, 'type', 'circumscribed'); &gt;&gt; x1= D1(:,1); &gt;&gt; y1= D1 (:,2); &gt;&gt; z1= D1 (:,3); &gt;&gt; plot3(x1,y1,z1,'o') &gt;&gt; grid on, box on</pre>	<pre>&gt;&gt; D2 = ccdesign(nfactors, 'type', 'faced'); &gt;&gt; x2= D2(:,1); &gt;&gt; y2= D2 (:,2); &gt;&gt; z2= D2 (:,3); &gt;&gt; plot3(x2,y2,z2,'o') &gt;&gt; grid on, box on</pre>
--	---	---

### **BBDESIGN** Генерування матриці плану Бокса-Бенкена

Дана функція генерує план Бокса-Бенкена (Box-Behnken) для експерименту із числом факторів 3-7, 9-12 або 16. Для іншого числа факторів план генерується подібним же чином, але він може виявитися дуже великим для практичних цілей.

*Синтаксис:*

```
D = bbdesign(nfactors)
D = bbdesign(nfactors,'pname1',pvalue1,'pname2',pvalue2,...)
[D,blk] = bbdesign(...)
```

*Опис:*

Функція **D = bbdesign(nfactors)** дозволяє одержати матрицю *D* плану Бокса-Бенкена для *nfactors* факторів. Розмірність матриці плану

експерименту  $D$  дорівнює  $n \times n_{factors}$ , де  $n$  – кількість точок плану. Кожен рядок – це перелік рівнів для всіх факторів, масштабованих між -1 і 1, тобто елементами матриці  $D$  є 1 і -1, що кодують максимальні й мінімальні значення факторів відповідно.

Функція **[D,blk] = bbdesign(nfactors)** додатково повертає вектор *blk* номерів блоків варіантів. Це вектор-стовпець номерів блоків довжиною  $n$ . Блоки – це групи варіантів, які мають бути проведені в схожих умовах (наприклад, в один день). План, розбитий на блоки, мінімізує ефект міжблокової різниці в оцінках параметрів, тобто блоки дозволяють згрупувати варіанти таким чином, щоб мінімізувати вплив змін зовнішніх умов на значення оцінюваних параметрів.

Функція **[...] = bbdesign(nfactors,'pname1',pvalue1,'pname2',pvalue2,...)** дозволяє задати додаткові параметри '*pname*' і їх значення *pvalue*:

'center' – кількість повторень спостережень у центральних точках плану;

'blocksize' – максимальна кількість точок у блоці плану.

Додаткові аргументи задаються у вигляді пари: 'назва параметра', значення параметра.

*Приклади:*

1. Генерування матриці плану Бокса-Бенкена для 4 факторів, розділеного на блоки, за умови 5 повторних спостережень у центральній точці плану й максимальному числі точок у блоці плану, що дорівнює 10.

**>> [D,blk] = bbdesign(4, 'center', 5, 'blocksize', 10)**

D =

-1	-1	0	0	-1	0	0	-1	-1	0	-1	0	0	0	0	0
-1	1	0	0	-1	0	0	1	-1	0	1	0	0	0	0	0
1	-1	0	0	1	0	0	-1	1	0	-1	0	0	0	0	0
1	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0
0	0	-1	-1	0	-1	-1	0	0	-1	0	-1	0	0	0	0
0	0	-1	1	0	-1	1	0	0	-1	0	1				
0	0	1	-1	0	1	-1	0	0	1	0	-1				
0	0	1	1	0	1	1	0	0	1	0	1				

blk =	1	1	2	2	3	3	2
1	1	2	2	3	3	1	3
1	1	2	2	3	3	1	
1	1	2	2	3	3	2	

2. Графічно матрицю плану Бокса-Бенкена для 3 факторів можна побудувати в такий спосіб

```
>> D = bbdesign(3)
D =
    -1  -1  0
    -1   1  0
     1  -1  0
     1   1  0
    -1   0  -1
    -1   0   1
     1   0  -1
     1   0   1
     0  -1  -1
     0  -1   1
     0   1  -1
     0   1   1
     0   0   0
     0   0   0
     0   0   0
    >> x= D(:,1);
    >> y= D(:,2);
    >> z= D(:,3);
    >> plot3(x,y,z,'o'), grid
```

На рис. 19.9. відображені 15 точок плану з 3-ма центральними варіантами (що позначені п'ятипроменевою зіронькою).

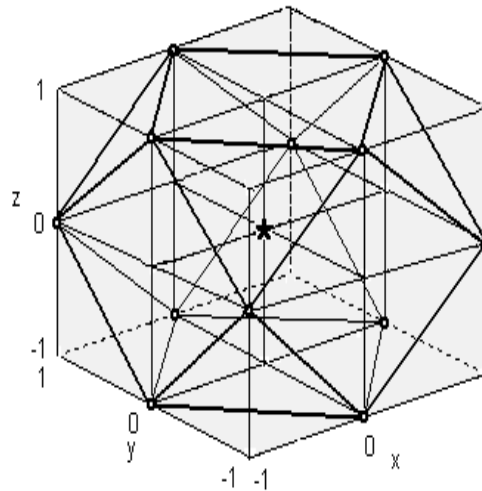


Рис. 19.9. Графічне зображення плану Бокса-Бенкена

### 19.4. D-оптимальні плани експерименту

<b>CANDGEN</b>	Генерування початкової множини точок у факторному просторі для D-оптимального плану .....	421
<b>CANDEXCH</b>	Генерування D-оптимального плану з початкової множини точок у факторному просторі на підставі алгоритму перестановки рядків .....	424
<b>ROWEXCH</b>	Генерування матриці D-оптимального плану на підставі алгоритму перестановки рядків .....	426
<b>CORDEXCH</b>	Генерування матриці D-оптимального плану на підставі алгоритму зміни координат .....	429

<b>DAUGMENT</b> Доповнення матриці значень факторів до D-оптимального плану .....	432
<b>DCOVARY</b> Генерування D-оптимального блокового плану .....	435

**CANDGEN** Генерування початкової множини точок у факторному просторі для D-оптимального плану

Генерується множина точок – кандидатів для обміну рядками за побудові D-оптимального плану функцією **candexch**.

*Синтаксис:*

```
xcand = candgen(nfactors,'model')
[xcand,fxcand] = candgen(nfactors,'model')
```

*Опис:*

Функція **xcand = candgen(nfactors,'model')** призначена для генерування початкової множини точок (рядків з координатами точок) *xcand* у факторному просторі із числом факторів *nfactors* і видом регресійної моделі *'model'*. Матриця результативних значень *xcand* містить *nfactors* стовпців.

В табл. 19.2 наведено можливі види математичних моделей:

Таблиця 19.2

**Види математичних моделей**

<b>Значення <i>'model'</i></b>	<b>Вид математичної моделі</b>
<i>'linear'</i>	Лінійна модель із константою. Значення за замовчуванням
<i>'interaction'</i>	Лінійна модель із константою і ефектами взаємодії
<i>'quadratic'</i>	Повна квадратична модель, що включає лінійні, квадратичні ефекти, ефекти взаємодій, константу
<i>'purequadratic'</i>	Неповна квадратична модель, що включає лінійні, квадратичні ефекти, константу

Крім рядкового значення вхідний аргумент *model* може бути заданий як вектор або матриця аналогічно такому ж аргументу функції **x2fx**. Функція **x2fx** дозволяє виконати перетворення матриці значень факторів *X* у матрицю плану експерименту *D*. У випадку, якщо *X* і *model* задані як вектори, то матриця плану експерименту формується за

правилом: кожний стовпець  $D$  є послідовним піднесенням  $X$  у ступінь елемента вектора  $model$ . Розмірність матриці  $D$  дорівнює  $n \times m$ , де  $n$  – число елементів вектора  $X$ ,  $m$  – число елементів вектора  $model$ . Тобто, вектор  $model$  є переліком ступенів полінома регресійної моделі для одного фактора  $X$ . Якщо  $X$  і  $model$  задані як матриці, то стовпець матриці  $D$  формуються за формулою:  $D(i, j) = prod(X(i, :).^model(j, :))$ , тобто, елемент матриці  $D_{ij}$  є добутком елементів  $i$ -го рядка матриці  $X$ , зведених послідовно в ступені  $j$ -го рядка матриці  $model$ . Таким чином, кількість стовпців  $model$  має дорівнювати числу стовпців матриці  $X$ .

Функція **[xcand, fxcand] = candgen(nfactors, model)** повертає матрицю значень факторів  $xcand$  і матрицю значень ступенів  $fxcand$ .

*Примітка:* Функція *rowexch* генерує початкову множину точок у факторному просторі за допомогою функції **candgen** і формує матрицю D-оптимального плану за допомогою функції **candexch**. Роздільне використання функцій **candgen** і **candexch** доцільне у випадку, коли необхідно змінити початкову множину точок.

*Приклади:*

1. Генерування початкової множини у вигляді матриці значень факторів D-оптимального плану для 3-х факторів і повної квадратичної моделі

```
>> [xc, fxc]=candgen(3,'quadratic');
>> disp(' x0  x1  x2  x3  x1*x2  x1*x3  x2*x3  x1^2  x2^2  x3^2')
>> fprintf([' %2.0f  %2.0f  %2.0f  %2.0f  %2.0f  %2.0f  %2.0f  ...\n', fxc'])
' %2.0f  %2.0f  %2.0f  %2.0f\n', fxc')
x0 x1 x2 x3 x1x2 x1x3 x2x3 x1x1 x2 x2 x3 x3 | 1 0 0 0 0 0 0 0 0 0 0
1 -1 -1 -1 1 1 1 1 1 1 1 | 1 1 0 0 0 0 0 0 1 0 0
1 0 -1 -1 0 0 1 0 1 1 1 | 1 -1 1 0 -1 0 0 1 1 0
1 1 -1 -1 -1 -1 1 1 1 1 1 | 1 0 1 0 0 0 0 0 0 1 0
1 -1 0 -1 0 1 0 1 0 1 1 | 1 1 1 0 1 0 0 1 1 0
1 0 0 -1 0 0 0 0 0 1 1 | 1 -1 -1 1 1 -1 -1 1 1 1
1 1 0 -1 0 -1 0 1 0 1 1 | 1 0 -1 1 0 0 -1 0 1 1
1 -1 1 -1 -1 1 -1 1 1 1 1 | 1 1 -1 1 -1 1 -1 1 1 1
1 0 1 -1 0 0 -1 0 1 1 1 | 1 -1 0 1 0 -1 0 1 0 1
1 1 1 -1 1 -1 -1 1 1 1 1 | 1 0 0 1 0 0 0 0 0 1
1 -1 -1 0 1 0 0 1 1 0 1 | 1 1 0 1 0 1 0 1 0 1
1 0 -1 0 0 0 0 0 1 0 1 | 1 -1 1 1 -1 -1 1 1 1 1
1 1 -1 0 -1 0 0 1 1 0 1 | 1 0 1 1 0 0 1 0 1 1
1 -1 0 0 0 0 0 1 0 0 1 | 1 1 1 1 1 1 1 1 1 1
```

2. Генерування початкової множини у вигляді матриці значень факторів і матриці D-оптимального плану для 3-х факторів і регресійної моделі, заданої у вигляді матриці *model*.

```
>> nfactors=3;
>> model=[0 1 2; 0 1 2; 0 1 2; 0 1 2]
model =
    0   1   2
    0   1   2
    0   1   2
    0   1   2

```

0	1	2
0	1	2

```
>> [xcand,fxcand]=candgen(nfactors, model)
xcand =
 -1 -1 -1
  1 -1 -1
 -1  1 -1
  1  1 -1
 -1 -1  0
  1 -1  0

```

-1	1	0
1	1	0
-1	-1	1
1	-1	1
-1	1	1
1	1	1

```
fxcand =
  0  0  0  0
 -1 -1 -1 -1
 -1 -1 -1 -1
  1  1  1  1
  1  1  1  1
  0  0  0  0
  0  0  0  0

```

0	0	0	0
0	0	0	0
-1	-1	-1	-1
-1	-1	-1	-1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

3. Графічне представлення початкової множини *xcand* значень 3-х факторів і повної квадратичної моделі (рис. 19.10).

```
>> nfactors=3;
>> model='quadratic';
>> xcand=candgen(nfactors,model);
>> x=xcand (:,1);

```

>> y=xcand (:,2);
>> z=xcand (:,3);
>> plot3(x,y,z,'o')
>> grid on, box on

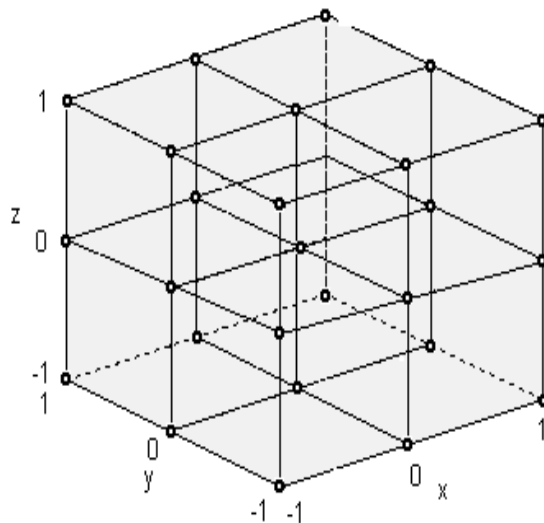


Рис. 19.10. Початкова множина значень 3-х факторів для повної квадратичної моделі (ПФЕ 3 ^ 3 ^ 3)

## **CANDEXCH** Генерування D-оптимального плану з початкової множини точок у факторному просторі на підставі алгоритму перестановки рядків

Будується D-оптимальний план, використовуючи алгоритм заміни рядків і точок-кандидатів, отримані за допомогою функції **candgen** або будь-яким іншим способом.

*Синтаксис:*

```
rlist = candexch(C,nrows)  
rlist = candexch(C,nrows,'param1',value1,'param2',value2,...)
```

*Опис:*

Функція **rlist = candexch(C,nrows)** призначена для генерування плану експерименту *rlist* з початкової множини точок (рядків-кандидатів) *C* у факторному просторі на основі алгоритму перестановки рядків. Матриця *C* розмірністю  $n \times p$  містить значення  $p$  ефектів моделі (кількість додатків у моделі) для кожної з  $n$  точок. Вхідний параметр *nrows* задає кількість варіантів у плані експерименту. Результативний параметр *rlist* є вектором із числом елементів *nrows* і задає номери обраних рядків з початкової множини точок *C*.

Функція **candexch** вибирає матрицю початкового плану *X* випадковим чином. Далі на підставі алгоритму заміни рядків *X* рядками початкової множини точок *C* формується D-оптимальний план експерименту. Обчислення припиняються, якщо буде виконана умова  $\max\{det[X'X]\}$ , або досягнуте максимальне число ітерацій.

Функція **rlist = candexch(C,nrows,'param1',value1,'param2',value2,...)** дозволяє визначити наступні додаткові вхідні аргументи (табл. 19.3):

Таблиця 19.3

### Додаткові аргументи *param* – *value*

Параметр 'param'	Його значення <i>value</i>
<i>'display'</i>	Відображення значення лічильника ітерацій. Можливі значення <i>'display'</i> : <i>'on'</i> – відобразити у командне вікно, <i>'off'</i> – скасування відображення. Значення за замовчуванням <i>'on'</i>

'init'	Початкова матриця значень факторів з розмірністю $nrows \times p$ . За замовчуванням ухвалюється випадкова підмножина рядків початкової множини $C$
'maxiter'	Максимальне число ітерацій. Значення за замовчуванням – 10

Додаткові аргументи задаються у вигляді пари: 'назва параметра', 'значення параметра'.

*Приклади:*

1. Генерування матриці значень факторів  $X$  D-оптимального плану з початкової множини точок у факторному просторі для 12 варіантів. Початкова множина  $xcand$  генерується за допомогою функції **candgen** для 3-х факторів і повної квадратичної моделі.

```
>> nfactors=3;
>> model='quadratic';
>> xcand = candgen(nfactors,model)
xcand =
```

-1	-1	-1	-1	1	-1	0	0	0	1	-1	1
-1	-1	-1	0	1	-1	1	0	0	-1	0	1
0	-1	-1	1	1	-1	-1	1	0	0	0	1
1	-1	-1	-1	-1	0	0	1	0	1	0	1
-1	0	-1	0	-1	0	1	1	0	-1	1	1
0	0	-1	1	-1	0	-1	-1	1	0	1	1
1	0	-1	-1	0	0	0	-1	1	1	1	1

```
>> nrows=12;
>> rlist = candexch(xcand,nrows)
rlist' = 1 7 3 7 9 1 9 3 9 3 1 7
```

2. Генерування матриці значень факторів  $X$  D-оптимального плану з початкової множини точок у факторному просторі для 12 варіантів. Початкова множина  $xcand$  генерується за допомогою функції **candgen** для 3-х факторів і повної квадратичної моделі. У якості додаткового параметра задана матриця  $A$  початкового наближення за пошуку D-оптимального плану із множини  $xcand$ . Матриця  $A$  є підматрицею  $xcand$ , перші 12 рядків.

```
>> nfactors=3;
>> model='quadratic';
>> xcand = candgen(nfactors,model)
xcand =
```

-1	-1	-1	-1	-1	0	0	-1	1
-1	-1	-1	0	-1	0	1	-1	1
0	-1	-1	1	-1	0	-1	0	1
1	-1	-1	-1	0	0	0	0	1
-1	0	-1	0	0	0	1	0	1
0	0	-1	1	0	0	-1	1	1

```

1  0 -1
-1 1 -1
0  1 -1
1  1 -1

```

```

-1 1 0
0  1 0
1  1 0
-1 -1 1

```

```

0 1 1
1 1 1

```

```

>> nrows=12;
>> A=xcand(1: nrows,:);
A =
-1 -1 -1
0 -1 -1

```

```

1 -1 -1
-1 0 -1
0 0 -1
1 0 -1
-1 1 -1

```

```

0 1 -1
1 1 -1
-1 -1 0
0 -1 0
1 -1 0

```

```

>> rlist = candexch(xcand,nrows,'init',A)
>> X = xcand(rlist,:);
rlist =
1
1
3
7
3
9

```

```

>> X=xcand(rlist,:);
X =
-1 -1 -1
-1 -1 -1
1 -1 -1
-1 1 -1
1 -1 -1

```

```

1 1 -1
-1 1 -1
-1 -1 -1
1 1 -1
1 1 -1
1 -1 -1
-1 1 -1

```

3. Графічне представлення початкової множини *xcand*, матриці початкового наближення *A* і матриці значень факторів *X* (рис. 19.11 – 19.13).

```

>> x= xcand (:,1);
>> y= xcand (:,2);
>> z= xcand (:,3);
>> plot3(x,y,z,'o')
>> grid on, box on

```

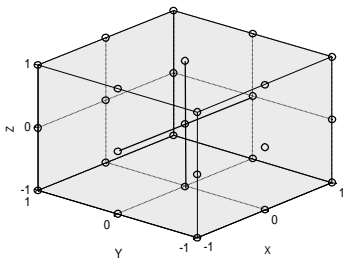


Рис. 19.11. Початкові значення *X*

```

>> x1= A(:,1);
>> y1= A(:,2);
>> z1= A(:,3);
>> plot3(x1,y1,z1,'o')
>> grid on, box on

```

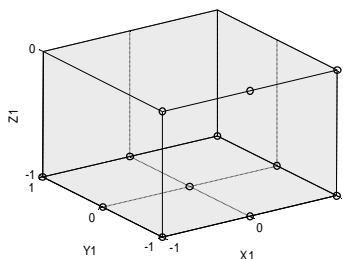


Рис. 19.12. Початкові значення матриці *A*

```

>> x2= X(:,1);
>> y2= X(:,2);
>> z2= X(:,3);
>> plot3(x2,y2,z2,'o')
>> grid on, box on

```

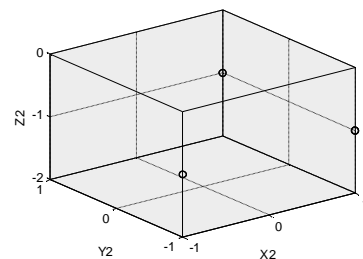


Рис. 19.13. Матриця значень факторів *X*

### **ROWEXCH** Генерування матриці D-оптимального плану на підставі алгоритму перестановки рядків

Будується D-оптимальний план, використовуючи алгоритм заміни рядків. Спочатку відбираються точки-кандидати для включення в план, а

далі провадиться багаторазова заміна точок (рядків) плану на точки-кандидати з метою поліпшити план. Якщо є необхідність самому задати (або змінити) точки-кандидати, використовуйте замість даної функції **candgen** і **candexch**.

*Синтаксис:*

```
settings = rowexch(nfactors,nruns)  
[settings,X] = rowexch(nfactors,nruns)  
[settings,X] = rowexch(nfactors,nruns,'model')  
[settings,X] = rowexch(...,'param1',value1,'param2',value2,...)
```

*Опис:*

Функція **settings = rowexch(nfactors,nruns)** дозволяє одержати матрицю значень факторів D-оптимального плану *settings* на підставі алгоритму перестановки рядків у кодованих факторів, що змінюються на двох рівнях (-1 – мінімальне значення фактора, 1 – максимальне значення фактора). Результат роботи функції *settings* являє собою матрицю D-оптимального плану експерименту за винятком стовпця відповідного до константи моделі. Вхідними параметрами є число факторів *nfactors* і кількість варіантів *nruns*. У якості рівняння регресії за замовчуванням ухвалюється лінійна модель. Розмірність матриці *settings* дорівнює добутку *nruns* рядків на *nfactors* стовпців.

Функція **[settings,X] = rowexch(nfactors,nruns)** крім матриці значень факторів *settings* повертає матрицю D-оптимального плану експерименту *X*.

Функція **[settings,X] = rowexch(nfactors,nruns,'model')** передбачає генерацію матриць *settings* і *X* для заданих числа факторів *nfactors*, кількості варіантів *nruns* і виду математичної моделі '*model*'. Можливі види математичних моделей наведено в табл. 19.2.

У функції **[settings,X] = rowexch(...,'param1',value1,'param2',value2,...)** крім числа факторів *nfactors*, кількості варіантів *nruns*, виду математичної моделі '*model*' передбачено додаткові вхідні аргументи '*param*' = '*display*', '*init*', '*maxiter*', які повністю еквівалентні додатковим вхідним аргументам функції **candexch**, що описано в табл. 19.3.

Додаткові аргументи задаються у вигляді пари: 'назва параметра', 'значення параметра'.

*Примітка.* Функція **rowexch** здійснює пошук D-оптимального плану на підставі алгоритму перестановки рядків. На першому етапі генерується початкова множина точок у факторному просторі, що можуть бути включені в план експерименту. На другому етапі за допомогою перестановки рядків у початковій матриці плану формується результуюча матриця плану експерименту за критерієм мінімізації дисперсії коефіцієнтів рівняння регресії. Якщо необхідно задати початкову множину точок відмінну від множини точок, що генерується за замовчуванням функцією **rowexch**, використовуються функції **candgen** і **candexch**.

Число точок повинне бути не менше числа членів моделі, інакше матриця плану експерименту не буде мати повного стовпцевого рангу.

*Приклади:*

1. Генерування матриці значень факторів і матриці D-оптимального плану для 4 факторів, 4 варіантів і неповної квадратичної моделі. Матриця  $A$  є початковою матрицею для генерації D-оптимального плану експерименту.

```
>> nfactors=4;
>> nruns=4;
>> model='purequadratic';
>> A=[1 0 1 0; 0 0 0 0; 1 1 1 1; 0 0 1 1]
A =
```

1	0	1	0
0	0	0	0
1	1	1	1
0	0	1	1

```
>> [settings,X] = rowexch(nfactors,nruns,model,'init',A)
settings =
```

-1	1	-1	1
1	-1	-1	-1
1	1	1	0
0	-1	1	1

```
X =
```

1	-1	1	-1	1	1	1	1	1
1	1	-1	-1	-1	1	1	1	1
1	1	1	1	0	1	1	1	0
1	0	-1	1	1	0	1	1	1

2. Графічне представлення матриці D-оптимального плану, розрахованої для 3 факторів, 14 варіантів і повної квадратичної моделі (рис. 19.14):

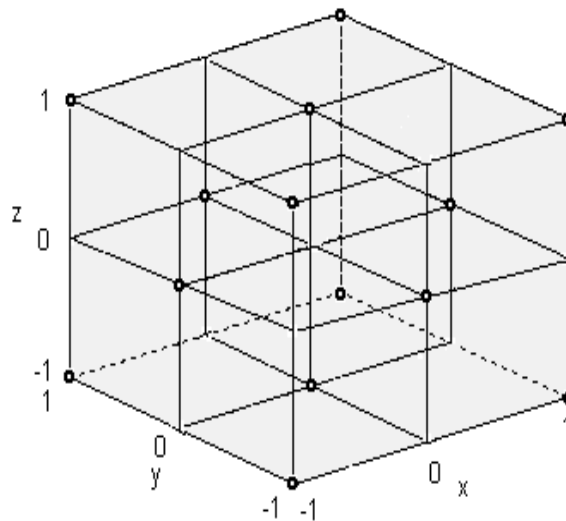


Рис. 19.14. Зображення D-оптимального плану

План 19.14 було побудовано операторами:

```
>> nfactors=3;
>> nruns=14;
>> model='quadratic';
>> settings = rowexch(nfactors,nruns,model);
>> x= settings (:,1);
>> y= settings (:,2);
>> z= settings (:,3);
>> plot3(x,y,z,'o')
>> box on % обмежуючий паралелепіпед
>> grid on
```

Одержано один з варіантів композиційної схеми Бокса з однаковими розмахами варіювання факторів блоку ядра і блоку зіркових точок. Цей план, найкращий (у деякому сенсі) за багатьма критеріями, у вітчизняній науковій літературі позначений як план  $B_3$ .

### **CORDEXCH** Генерування матриці D-оптимального плану на підставі алгоритму зміни координат

Будується D-оптимальний план, використовуючи алгоритм заміни координат.

*Синтаксис:*

```
settings = cordexch(nfactors,nruns)
[settings,X] = cordexch(nfactors,nruns)
[settings,X] = cordexch(nfactors,nruns,'model')
[settings,X] = cordexch(...,'param1',value1,'param2',value2,...)
```

Опис:

Функція **settings = cordexch(nfactors,nruns)** дозволяє одержати матрицю значень факторів *settings* D-оптимального плану на підставі алгоритму зміни координат для лінійної регресійної моделі. Кількість факторів задається вхідним аргументом *nfactors*, число варіантів плану – *nruns*. Розмірність матриці *settings* становить *nruns* рядків, *nfactors* стовпців.

Функція **[settings,X] = cordexch(nfactors,nruns)** додатково повертає матрицю D-оптимального плану *X*.

Функція **[settings,X] = cordexch(nfactors,nruns,'model')** дозволяє одержати матрицю значень факторів *settings* і матрицю *X* D-оптимального плану експерименту на основі алгоритму зміни координат для заданого числа факторів *nfactors*, числа варіантів *nruns* і виду регресійної моделі '*model*'. Можливі види регресійних моделей перелічені вище в табл. 19.2.

Крім рядкового значення вхідний аргумент *model* може бути заданий як вектор або матриця аналогічно такому ж аргументу функції **x2fx** (більш детально див. текст до табл. 19.2).

У функції **[settings,X] = cordexch(...,'param1',value1,'param2',value2,...)** додаткові вхідні аргументи *param* – *value* аналогічні додатковим аргументам функцій **candexch** і **rowexch** (табл. 19.3).

Число точок повинне бути не менше числа членів моделі, інакше матриця плану експерименту не буде мати повного стовбцевого рангу.

*Примітка.* Функція **cordexch** виконує пошук D-оптимального плану на підставі алгоритму перестановки координат. На першому етапі генерується початковий план експерименту. На другому етапі виконується зміна кожної координати точок плану з метою мінімізації дисперсії коефіцієнтів рівняння регресії.

*Приклади:*

1. Генерування матриць значень факторів D-оптимального плану для трьох факторів, 12 варіантів, повної квадратичної моделі.

```
>> [settings,X]=cordexch(3,12,'quadratic','display','off','maxiter',50);  
>> disp(' x0  x1  x2  x3  x1*x2  x1*x3  x2*x3  x1^2  x2^2  x3^2')
```

```
>> fprintf([' %2.0f %2.0f %2.0f %2.0f %2.0f %2.0f' ...
' %2.0f %2.0f %2.0f %2.0f\n'],X')
x0 x1 x2 x3 x1x2 x1x3 x2x3 x1^2 x2^2 x3^2
1 1 1 1 1 1 1 1 1 1
1 -1 -1 -1 1 1 1 1 1 1
1 1 -1 -1 -1 -1 1 1 1 1
1 -1 0 0 0 0 0 1 0 0
1 -1 1 1 -1 -1 1 1 1 1
1 1 1 -1 1 -1 -1 1 1 1
1 -1 1 -1 -1 1 -1 1 1 1
1 -1 -1 1 1 -1 -1 1 1 1
1 1 -1 1 -1 1 -1 1 1 1
1 0 1 1 0 0 1 0 1 1
1 0 0 -1 0 0 0 0 0 1
1 0 -1 0 0 0 0 0 1 0
```

2. Генерування матриць значень факторів D-оптимального плану для трьох факторів, 12 варіантів, повної квадратичної моделі. У якості додаткового вхідного аргументу задається матриця початкового наближення  $A$ .

```
>> nfactors=3;
>> nruns=12;
>> A=[0 0 0; 1 0 0; -1 0 0; 0 1 0; 1 1 -1; -1 0 -1; 1 0 0; 0 0 1; -1 1 -1; 1 0 -1; 1 1 1; 0 -1 0]
```

```
A =
 0 0 0 | 0 1 0 | 1 0 0 | 1 0 -1
 1 0 0 | 1 1 -1 | 0 0 1 | 1 1 1
-1 0 0 | -1 0 -1 | -1 1 -1 | 0 -1 0
```

```
>> [settings,X] = cordexch(nfactors,nruns,'quadratic','init',A)
```

```
settings =
 0 0 -1 | -1 1 1 | -1 0 0 | 1 0 0
 1 -1 -1 | 1 1 -1 | 1 -1 1 | 1 1 1
-1 -1 1 | -1 -1 -1 | -1 1 -1 | 0 -1 0
```

```
X =
 1 0 0 -1 0 0 0 0 0 1 | 1 -1 0 0 0 0 0 1 0 0
 1 1 -1 -1 -1 -1 1 1 1 1 | 1 1 -1 1 -1 1 -1 1 1 1
 1 -1 -1 1 1 -1 -1 1 1 1 | 1 -1 1 -1 -1 1 -1 1 1 1
 1 -1 1 1 -1 -1 1 1 1 1 | 1 1 0 0 0 0 0 1 0 0
 1 1 1 -1 1 -1 -1 1 1 1 | 1 1 1 1 1 1 1 1 1 1
 1 -1 -1 -1 1 1 1 1 1 1 | 1 0 -1 0 0 0 0 0 1 0
```

3. Графічне представлення матриці значень факторів D-оптимального плану для трьох факторів, 16 варіантів і повної квадратичної моделі (рис. 19.15):

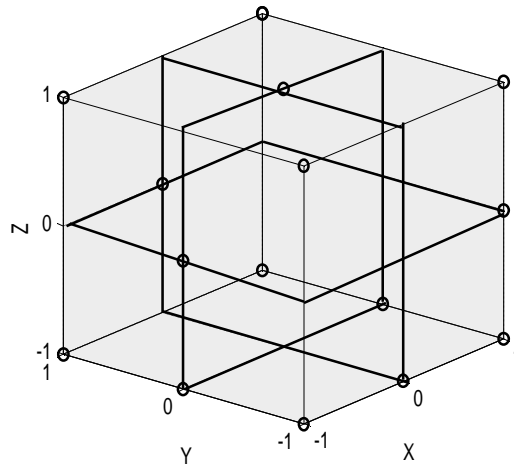


Рис. 19.15. D-оптимальний план для трьох факторів і 16 варіантів

Цей план було побудовано операторами:

```
>> nfactors=3; nruns=16;
>> settings = cordexch(nfactors,nruns,'quadratic');
>> x= settings (:,1);
>> y= settings (:,2);
>> z= settings (:,3);
>> plot3(x,y,z,'o')
>> grid on, box on
```

```
settings =
```

-1	-1	1	0	-1	1	1	-1	1
1	1	1	-1	0	1	0	-1	0
1	-1	-1	1	0	0	0	1	1
-1	-1	-1	-1	1	1	0	0	-1
1	1	-1	-1	1	0	1	1	-1

### **DAUGMENT** Доповнення матриці значень факторів до D-оптимального плану

Збільшується побудований D-оптимальний план на задану кількість точок.

*Синтаксис:*

```
settings = daugment(startdes,nruns)
[settings,X] = daugment(startdes,nruns)
[settings,X] = daugment(startdes,nruns,'model')
[settings, X] = daugment(...,'param1',value1,'param2',value2,...)
```

Опис:

Функція **settings = daugment(startdes,nruns)** для доповнення *nruns* варіантів до вхідної матриці *startdes* з метою одержання D-оптимальної матриці значень факторів плану експерименту *settings*. Функція заснована на алгоритмі зміни координат.

Функція **[settings,X] = daugment(startdes,nruns)** додатково повертає матрицю *X* D-оптимального плану експерименту.

Функція **[settings,X] = daugment(startdes,nruns,'model')** дозволяє одержати матрицю значень факторів *settings* і матрицю *X* D-оптимального плану експерименту на підставі алгоритму зміни координат для заданої матриці значень факторів *startdes*, заданого числа варіантів *nruns* і виду регресійної моделі *'model'*. Можливі моделі регресії наведені вище в табл. 19.2.

Крім рядкового значення вхідний аргумент *model* може бути заданий як вектор або матриця відповідно такому ж аргументу функції **x2fx** (більш детально див. текст до табл. 19.2).

**[settings,X] = daugment(...,'param1',value1,'param2',value2,...)** – при такому синтаксисі функції визначаються додаткові аргументи у вигляді пар "ім'я – значення". Ці аргументи аналогічні додатковим аргументам функцій **candexch**, **rowexch** і **cordexch** (табл. 19.3).

Приклади:

1. Доповнення матриці значень факторів повного факторного експерименту  $2^2$  до D-оптимальної матриці значень факторів. Регресійна модель ухвалюється за замовчуванням лінійною. Кількість варіантів, що додаються, дорівнює 5. У якості додаткового вхідного аргументу задається матриця початкового наближення *A*.

```
>> nruns=5;
>> startdes = [-1 -1; 1 -1; -1 1; 1 1]
startdes =
-1 -1
 1 -1
-1  1
 1  1
>> A=[0 0; 0 1; 1 0; 1 1; -1 1]
A =
 0  0
 0  1
 1  0
 1  1
-1  1
>> [settings,X] = daugment(startdes,nruns,'quadratic','init',A)
```

```

settings =
-1 -1
 1 -1
-1  1
 1  1
 0  0
 0 -1
 1  0
 0  1
-1  0

```

$$X = \begin{pmatrix} 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

2. Графічне представлення матриці значень факторів  $X$  повного факторного експерименту  $2^3$  (функція **ff2n**) і доповненої до D-оптимальної матриці значень факторів *settings*. Кількість варіантів, що додаються, 5.

На рис. 19.16 зображено початковий план ПФЕ  $2^3$ , а на рис. 19.17 – план, що доповнений 5-ю варіантами до D-оптимального.

```

>> startdes = ff2n(3)
startdes =
 0 0 0
 0 0 1

```

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

```

>> nruns=5;
>> settings = daugment(startdes,nruns)
settings =
 0 0 0
 0 0 1
 0 1 0

```

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} -1 & -1 & 1 \\ 1 & -1 & -1 \end{pmatrix}$$

```
>> x= X (:,1);
>> y= X (:,2);
>> z= X (:,3);
>> plot3(x,y,z,'o')
>> grid on, box on
```

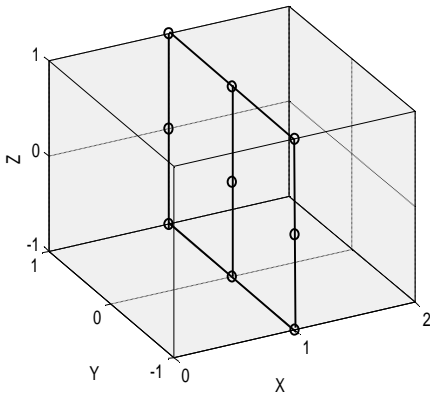


Рис. 19.16. Матриця даних

```
>> x1= settings (:,1);
>> y1= settings (:,2);
>> z1= settings (:,3);
>> plot3(x1,y1,z1,'o')
>> grid on, box on
```

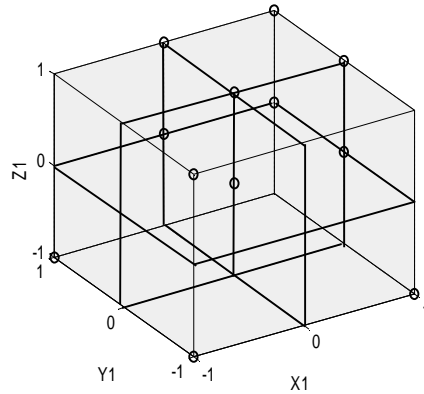


Рис. 19.17. Доповнення до D-оптимальної матриці

### **DCOVARY** Генерування D-оптимального блокового плану

Ця функція будує D-оптимальний план із заданими рівнями деяких факторів.

*Синтаксис:*

```
settings = dcovary(nfactors,covariates)
[settings,X] = dcovary(nfactors,covariates)
[settings,X] = dcovary(nfactors,covariates,'model')
[settings,X] = dcovary(...,'param1',value1,'param2',value2,...)
```

*Опис:*

Функція **settings = dcovary(nfactors,covariates)** дозволяє одержати матрицю значень факторів *settings* D-оптимального плану, розділеного на блоки *covariates* для заданого числа факторів *nfactors*. Кількість варіантів у плані експерименту задається числом рядків у матриці *covariates*. Матриця значень факторів *settings* формується послідовно із двох матриць: матриці значень факторів D-оптимального плану й матриці, що кодує блоки плану. Розмірність матриці *settings* рівна  $n \times m$ , де  $n$  – кількість рядків матриці *covariates*,  $m = nfactors + k$ , де  $k$  –

кількість стовпців матриці *settings*. Функція заснована на алгоритмі зміни координат.

Функція **[settings,X] = dcovary(nfactors,covariates)** додатково повертає матрицю D-оптимального плану *X*.

Функція **[settings,X] = dcovary(nfactors,covariates,'model')** дозволяє одержати матрицю значень факторів *settings* і матрицю *X* D-оптимального плану експерименту на підставі матриці *covariates*, що задає блоки, кількість варіантів, заданого числа факторів *nfactors*, і виду рівняння регресії '*model*'. Можливі моделі регресії описані в табл. 19.2:

Крім рядкового значення вхідний аргумент *model* може бути заданий як вектор або матриця відповідно такому ж аргументу функції **x2fx** (більш детально див. текст до табл. 19.2).

**[settings,X] = dcovary(...,'param1',value1,'param2',value2,...)** – при такому синтаксисі функції додаткові вхідні аргументи визначається у вигляді пар "ім'я-значення". Можливі імена (рядки) і відповідні їм значення наведені в табл. 19.3 (вони аналогічні для всіх попередніх функцій).

*Примітка.* Функція **dcovary** виконує пошук D-оптимального плану на підставі алгоритму перестановки координат. На першому етапі генерується початковий план експерименту, що містить кодові значення, що розділяють матрицю плану на блоки, і значення рівнів факторів. На другому етапі виконується зміна для кожного фактора координат точок плану з метою мінімізації дисперсії коефіцієнтів рівняння регресії.

*Приклади:*

1. Складається D-оптимальний план експерименту з 8 спостережень, розбитих на 4 блоки по 2 спостереження у кожному для двофакторної лінійної моделі.

```
>> covariates = dummyvar([1 1 2 2 3 3 4 4]);
>> settings = dcovary(2,covariates(:,1:3),'linear')
```

```
settings =
```

1	1	1	0	0	-1	1	0	1	0	-1	-1	0	0	1
1	1	1	0	0	1	-1	0	1	0	-1	1	0	0	0
-1	-1	1	0	0	1	1	0	0	1	1	-1	0	0	0

Перші два стовпці є вихідною матрицею для двох факторів. Останні три стовпці – це dummy-змінні, які кодують чотири блоки.

2. Генерування матриці значень факторів і матриці D-оптимального плану для 3 факторів розділеного на 3 блоку по 3 варіанти в кожному. Ділення на блоки задає вектор *covariates*. У якості додаткового параметра використовується матриця початкового наближення *A*.

```
>> nfactors=3;
>> covariates=[-1 -1 -1 1 1 1 2 2 2]';
>> A=[ 0 0 0; 1 0 0; 0 1 0; -1 0 0; -1 -1 0; 0 0 -1; 1 -1 1; -1 -1 1; 1 -1 -1]
A =
```

0	1	0	0	0	-1	1	-1	-1
0	0	0	-1	0	0	1	-1	1
1	0	0	-1	-1	0	-1	-1	1

```
>> [settings,X] = dcovary(nfactors,covariates,'purequadratic')
settings =
```

0	0	-1	-1
-1	-1	1	-1
1	1	0	-1
1	-1	-1	1
1	0	1	1
-1	1	0	1
0	-1	0	2
0	1	1	2
-1	0	-1	2

```
X =
```

1	0	0	-1	-1	0	0	1	1
1	-1	-1	1	-1	1	1	1	1
1	1	1	0	-1	1	1	0	1
1	1	-1	-1	1	1	1	1	1
1	1	0	1	1	1	0	1	1
1	-1	1	0	1	1	1	0	1
1	0	-1	0	2	0	1	0	4
1	0	1	1	2	0	1	1	4
1	-1	0	-1	2	1	0	1	4

3. Графічне представлення матриці D-оптимального плану експерименту для 3 факторів, розділеного на 2 блоки по 5 варіантів у кожному. Матриця D-оптимального плану розраховується для неповної квадратичної моделі (рис. 19.18 – 19.20).

```
>> nfactors=3;
>> covariates=[0 0 0 0 0 1 1 1 1 1]';
>> settings = dcovary(nfactors,covariates,'purequadratic');
% Перший блок      % Другий блок      % Загальний графік
>> x= settings (1:5,1); >> x1= settings (6:10,1); >> plot3(x,y,z,'o',x1,y1,z1,'dr')
>> y= settings (1:5,2); >> y1= settings (6:10,2); >> grid, box on
>> z= settings (1:5,3); >> z1= settings (6:10,3);
>> plot3(x,y,z,'o'), >> plot3(x1,y1,z1,'o'),
>> grid, box on >> grid, box on
```

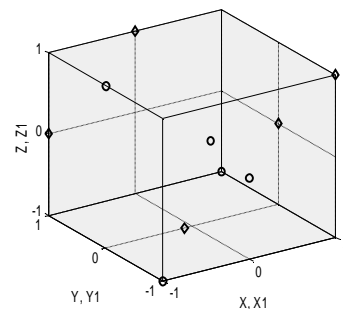
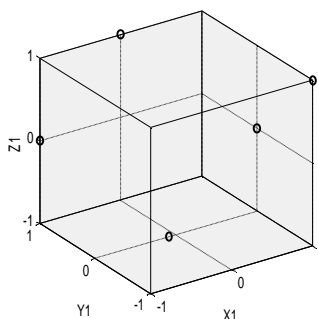
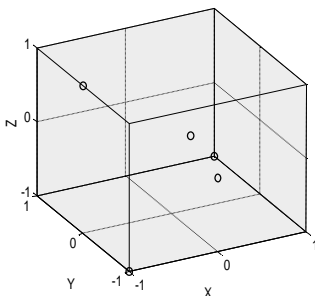


Рис. 19.18. Перший блок

Рис. 19.19. Другий блок .

Рис. 19.20. Загальний графік

## 9.5. Латинські квадрати, куби, гіперкуби

<b>LHSDESIGN</b> Функція генерування вибірки чисел латинського гіперкуба .....	438
<b>LHSNORM</b> Функція генерування вибірки чисел за нормально розподіленим латинським гіперкубом .....	441

### **LHSDESIGN** Функція генерування вибірки чисел латинського гіперкуба

Латинський квадрат – це квадратна матриця порядку  $n$ , кожний рядок і кожний стовець якої є перестановками елементів кінцевої множини, що полягає з  $n$  елементів. Число  $n$  називають порядком латинського квадрата. Аналогічно будуються латинські куби й гіперкуби.

Латинські квадрати (куби, гіперкуби) корисні, коли потрібно побудувати план багатовимірного експерименту з випадковими значеннями координат, що мають рівномірний розподіл. Дана функція будує латинський гіперкуб будь-якого порядку із заданою однаковою кількістю точок щодо кожної розмірності.

*Синтаксис:*

```
X = lhsdesign(n,p)  
X = lhsdesign(...,'smooth','off')  
X = lhsdesign(...,'criterion','c')  
X = lhsdesign(...,'iterations',k)
```

*Опис:*

Функція **X = lhsdesign(n,p)** призначена для генерування вибірки латинського гіперкуба  $X$ , що містить  $n$  значень для кожного значення змінної  $p$  (генерується матриця розмірністю  $n \times p$ ). Столпчик матриці

містить  $n$  випадкових чисел отриманих на одному з інтервалів  $(0, 1/n)$ ,  $(1/n, 2/n)$ , ...,  $(1-1/n, 1)$  з наступною їх випадковою перестановкою.

Функція **X = lhsdesign(...,'smooth','off')** виконує випадкове генерування чисел з інтервалів  $(0, 1/n)$ ,  $(1/n, 2/n)$ , ...,  $(1-1/n, 1)$ . Значення параметра *'smooth'* визначає:

1. **X = lhsdesign(...,'smooth','off')** – випадкове генерування чисел з інтервалів  $(0, 1/n)$ ;  $(1/n, 2/n)$ ; ... ;  $(1 - 1/n, 1)$ ;

2. **X = lhsdesign(...,'smooth','on')** – випадкове генерування значень вибірки як центральних точок вказаних інтервалів:  $0.5/n$ ,  $1.5/n$ , ... ,  $1 - 0.5/n$ .

Значення параметра *'smooth'* за замовчанням – *'on'*.

Параметр *'criterion'* задає ітераційну процедуру генерації вибірки латинського гіперкуба  $X$  у відповідності зі значенням *'c'*. Можливі значення *'criterion'* наведені в наступній табл. 19.4.

Таблица 19.4

**Значення параметра *'criterion'***

Параметр <i>'criterion'</i>	Значення параметра
<i>'none'</i>	Відключення ітераційної процедури
<i>'maximin'</i>	Максимізація мінімальної відстані між значеннями вибірки
<i>'correlation'</i>	Мінімізація значення коефіцієнта кореляції

Параметр *'iterations'* дозволяє задати число ітерацій  $k$  для генерування вибірки  $X$  ітераційною процедурою. За замовчуванням число ітерацій  $k = 5$ .

План за латинським гіперкубом використовується у випадках, коли необхідно одержати випадкову багатовимірну вибірку рівномірно розподілену за кожним виміром.

*Приклади:*

1. Генерування матриці латинського гіперкуба з розмірністю  $3 \times 5$ .

```
>> n=3; p=5;
>> X = lhsdesign(n,p)
X =
```

```
0.5293 0.1867 0.9291 0.6221 0.1234
0.0851 0.3555 0.3869 0.2643 0.8765
0.9107 0.7722 0.1237 0.7976 0.4750
```

2. Використання параметра *'smooth'* для генерування матриці латинського гіперкуба з розмірністю  $3 \times 5$ .

```
>> X = lhsdesign(n,p,'smooth','off')
X =
    0.5000    0.1667    0.5000    0.5000    0.1667
    0.8333    0.8333    0.1667    0.8333    0.5000
    0.1667    0.5000    0.8333    0.1667    0.8333
>> X = lhsdesign(n,p,'smooth','on')
X =
    0.0626    0.3026    0.9446    0.5161    0.7100
    0.4633    0.5250    0.0556    0.0145    0.4102
    0.7662    0.8748    0.3871    0.9509    0.1853
```

3. Використання критерію генерування вибірки латинського гіперкуба.

```
>> X = lhsdesign(n,p,'criterion','none')
X =
    0.2513    0.5117    0.8861    0.8628    0.4990
    0.8042    0.8195    0.1994    0.2380    0.0927
    0.4980    0.0192    0.5641    0.5353    0.8979
>> X = lhsdesign(n,p,'criterion','maximin')
X =
    0.0570    0.1597    0.5131    0.5188    0.7161
    0.6112    0.4273    0.1849    0.8779    0.0802
    0.8687    0.8103    0.9708    0.2325    0.3501
>> X = lhsdesign(n,p,'criterion','correlation')
X =
    0.8333    0.8333    0.1667    0.1667    0.1667
    0.5000    0.1667    0.5000    0.5000    0.8333
    0.1667    0.5000    0.8333    0.8333    0.5000
```

4. Генерування латинського гіперкуба із заданим числом ітерацій

```
>> X = lhsdesign(n,p,'criterion','correlation','iterations',10)
X =
    0.5000    0.1667    0.5000    0.5000    0.5000
    0.8333    0.5000    0.1667    0.8333    0.1667
    0.1667    0.8333    0.8333    0.1667    0.8333
```

5. Побудуємо латинський куб з рівномірно розподіленими в ньому 500 точками. Координати цих точок можна використовувати як значення рівнів факторів в 3-факторному експерименті, якщо потрібно досліджувати дуже складну поверхню відгуку (рис. 19.21).

```

>> n=500; % кількість точок
>> x=lhsdesign(n,3); % n точок в 3-вимірному кубі
>> plot3(x(:,1),x(:,2),x(:,3),'k.');
```

```

>> set(get(gcf,'CurrentAxes'),'FontName','Arial Cyr','FontSize',14)
>> title('\bfРівномірний розподіл точок в кубі') % заголовок
>> axis equal % рівні масштаби
>> box on % обмежуючий паралелепіпед
>> grid on % сітка
```



Рис. 19.21. Латинський куб з 500 точками

### **LHSNORM** Функція генерування вибірки чисел за нормально розподілим латинським гіперкубом

Будується латинський гіперкуб будь-якого порядку з багатовимірним нормальним розподілом точок у ньому.

*Синтаксис:*

```

X = lhsnorm(mu,SIGMA,n)
X = lhsnorm(mu,SIGMA,n,'onoff')
```

*Опис:*

Функція **X = lhsnorm(mu,SIGMA,n)** генерує вибірку латинського гіперкуба  $X$  розмірністю  $n$ , відповідну багатовимірному нормальному розподілу з вектором середнього  $\mu$  і коваріаційною матрицею  $SIGMA$ . Матриця  $SIGMA$  повинна бути квадратною, позитивно визначеною, з числом рядків (стовпців), що дорівнює числу елементів у векторі  $\mu$ . Вибірka  $X$  розподілена за багатовимірним нормальним законом.

Граничним розподілом кожного стовпчика масиву є одновимірний нормальний закон.

У функції  $X = \text{lhsnorm}(\mu, \text{SIGMA}, n, 'onoff')$  параметр *'onoff'* визначає коефіцієнт згладжування вибірки. Якщо *'onoff' = 'off'*, кожний стовпчик масиву є перестановкою значень  $G(0.5/n), G(1.5/n), \dots, G(1-0.5/n)$ , де  $G$  зворотна функція нормального розподілу. Якщо *'onoff' = 'on'*, кожний стовпчик масиву включає значення розподілені за рівномірним законом. У цьому випадку, у якості першого значення рівного  $0.5/n$  генеруються значення, що розподілене за законом рівної ймовірності на інтервалі  $(0/n, 1/n)$ . За замовчуванням ухвалюється *'onoff' = 'on'*.

*Приклади:*

1. Генерування матриці псевдовипадкових чисел з розмірністю  $2 \times 5$ .

```
>> mu=[1 2];
>> n=5;
>> SIGMA=pascal(2)
SIGMA =
    1    1
    1    2
>> X = lhsnorm(mu,SIGMA,n)
X =
    1.0958    3.9176
   -1.0147    1.2795
    1.6578    1.7158
    1.8508    2.5286
    0.6324    0.4250
```

2. Генерування матриці псевдовипадкових чисел з розмірністю  $2 \times 5$  і використанням параметра *'onoff'*.

```
>> mu=[1 2];
>> n=5;
>> SIGMA=pascal(2);
>> X = lhsnorm(mu,SIGMA,n,'off')
X =
    0.4756    2.0000
    1.5244    2.7416
   -0.2816    1.2584
    2.2816    3.8124
    1.0000    0.1876
>> X = lhsnorm(mu,SIGMA,n,'on')
X =
    0.7668    3.3196
   -0.1022    2.3487
    1.3242   -0.0904
    1.8630    3.1277
    0.4836    1.6030
```

3. Побудуємо латинський куб, точки якого мають нормальний розподіл із заданими середніми й коваріаційною матрицею (рис. 19.22).

```
>> n=500; % кількість точок
>> mu=[0.5 0.4 0.3]; % середні
>> s=[0.3 0.08 0.09;0.08 0.2 0.05;0.09 0.05 0.1]; % коваріаційна матриця
```

```
>> x=lhsnorm(mu,s,n); % n точок в 3-вимірному кубі
>> plot3(x(:,1),x(:,2),x(:,3),'k.');
```

**>> set(get(gcf,'Currentaxes'),'Fontname','Arial Cyr','FontSize',14)**  
**>> title('\bfНормальний розподіл точок')** % заголовок  
**>> axis equal** % однакові масштаби  
**>> box on** % обмежуючий паралелепіпед  
**>> grid on** % сітка

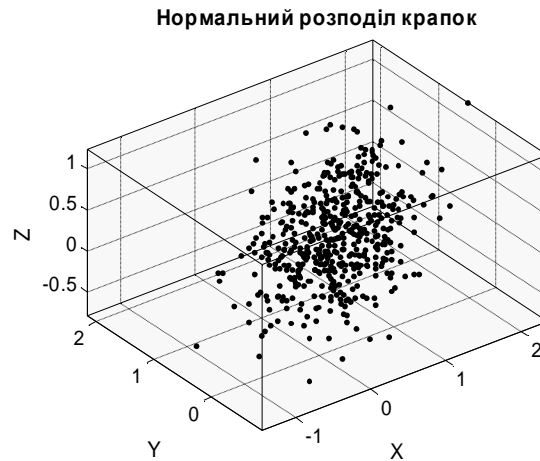


Рис. 19.22. Латинський куб з нормальним розподілом

## 20. Марківські моделі

Марківські процеси є прикладами стохастичних процесів, тобто процесів, що генерують випадкові послідовності результатів або станів відповідно до певної ймовірності. Марківські процеси відрізняються тим, що "майбутнє" і "минуле" процесу не залежать один від одного при відомому "сьогодення". Моделі марківських процесів використовуються в широкому спектрі додатків від щоденних цін на товари до позиції генів у хромосомі.

Марківські ланцюги.

Про марківські ланцюги дає уявлення так звана діаграма станів (рис.20.1).

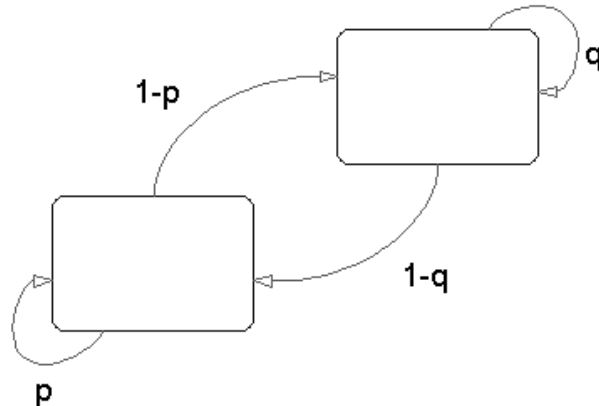


Рис. 20.1. Діаграма станів

Прямокутники на схемі представляють можливі стани процесу, що моделюється, а стрілки являють собою переходи між станами. Позначення на кожній стрілці представляє ймовірність такого переходу. На кожному етапі цього процесу модель може перейти в інший стан залежно від стану, у якому вона перебуває в цей момент.

Важливою характеристикою марківських моделей є те, що наступний стан залежить лише від поточного стану, а не від історії переходів, які привели до нинішнього стану. Наприклад, кидання монети приводить до двох результатів: на верхній грані з'являється або герб, або цифра. Провадимо серію кидків. Результат останнього кидка визначає поточний стан моделі, а наступні кидки визначають перехід в інший стан. Якщо монета симетрична, то ймовірність такого переходу

дорівнює  $1/2$ . У більш складних моделях випадкових процесів переходи в інший стан можуть відбуватися з іншими ймовірностями.

Марківські ланцюги – це математичний опис марківських моделей з дискретним набором станів. Марківські ланцюги характеризуються:

набором початкових станів  $\{1, 2, \dots, m\}$ ;

матрицею переходів  $T$  (матрицею перехідних ймовірностей) розміром  $m \times m$ , елемент якої з індексами  $i, j$  є ймовірністю переходу зі стану  $i$  у стан  $j$ . Сума елементів у кожному рядку повинна дорівнювати 1, тому що це є сума ймовірностей здійснити перехід з конкретного стану в кожний з можливих;

набором можливих кінцевих значень, або емісій  $\{s_1, s_2, \dots, s_n\}$ . За замовчуванням, такий набір позначають як  $\{1, 2, \dots, n\}$ , де  $n$  є номер можливого кінцевого стану, але можна вибрати будь-які інші числа або символи;

матрицею емісій  $E$  (матрицею кінцевих станів) розміром  $m \times n$ , кожний елемент якої з індексами  $l, k$  є ймовірність появи  $l$ -го значення, якщо модель перебуває в  $k$ -му стані. Сума елементів кожного рядка матриці  $E$  також повинна дорівнювати 1.

Нехай початковий стан марківського ланцюга буде  $i_0$  (крок 0). Далі ланцюг переходить у стан  $i_1$  з ймовірністю  $T_{1i_1}$  й значенням  $s_k$  з ймовірністю  $E_{i_1k_1}$ . Отже, ймовірність спостереження послідовності станів  $i_1 i_2 \dots i_r$  і послідовності значень  $s_{k_1}, s_{k_2}, \dots, s_{k_r}$  через перших  $r$  кроків дорівнює  $T_{1i_1} E_{i_1k_1} T_{i_1i_2} E_{i_2k_2} \dots \mathbf{K} T_{i_{r-1}i_r} E_{i_rk_r}$ .

Схована марківська модель – коли ви спостерігаєте послідовність емісійних значень, але не знаєте послідовність станів, через які пройшла модель для одержання цих значень. Аналіз схованих марківських моделей ставить метою відновити послідовність станів за спостережуваними даними.

Як приклад розглянемо марківську модель із двома можливими станами й шістьма можливими значеннями. Модель використовує:

Червоний кубик, що має шість граней, позначених цифрами від 1 до 6.

Зелений дванадцятигранник, п'ять граней із яких позначені цифрами від 2 до 6, а інші сім позначені цифрою 1.

Асиметричну червону монету, за кидка якої ймовірність випадання герба дорівнює 0.9, а ймовірність випадання цифри – 0.1.

Асиметричну зелену монету, за кидка якої ймовірність випадання герба дорівнює 0.95, а ймовірність випадання цифри – 0.05.

На рис.20.2 представлена діаграма станів для цієї моделі, що має два стани, червоне й зелене.

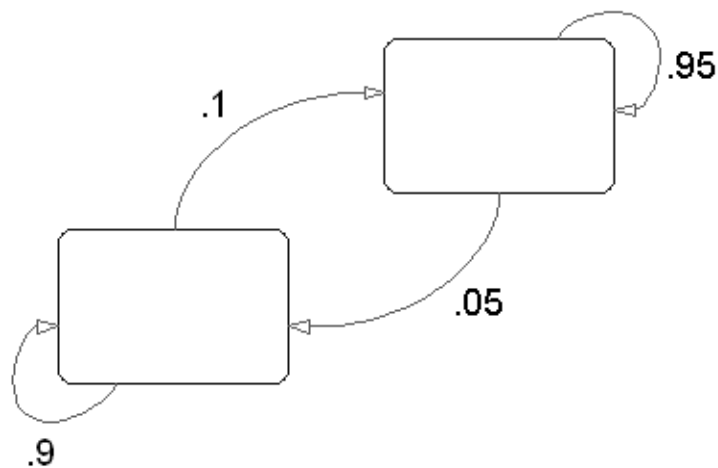


Рис.20.2. Діаграма станів

У цієї моделі емісійне значення визначається за кольором кинутого "кубика", а перехід в інший стан визначається результатом кидання монети того ж кольору.

Матриця переходу має вигляд  $T = \begin{pmatrix} 0.9 & 0.1 \\ 0.05 & 0.95 \end{pmatrix}$ , а матриця

емісійних значень –  $E = \begin{pmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{7}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \end{pmatrix}$ .

Модель не є схованою, тому що ми знаємо послідовність станів за кольором монети й кубика. Припустимо, однак, що хтось інший генерує емісійні значення, а ви не бачите кубика або монети. Усе, що ви бачите – це послідовність емісійних значень. Ви можете припускати наступне

значення, але не можете бути в ньому впевнені, тому що не можете бачити кольору кинутого кубика.

За такої моделі виникають наступні питання.

1. Знаючи послідовність емісійних значень, як з'ясувати шляхи їх досягнення?

2. Знаючи послідовність емісійних значень, як можна обчислити ймовірності переходів і емісій у конкретній моделі?

3. Яка апіорна ймовірність того, що модель згенерує певну послідовність?

4. Яка апостеріорна ймовірність настання конкретного стану в будь-якій точці послідовності?

Нижче розглянуті наступні функції марківських моделей:

<b>HMMGENERATE</b>	Генерування схованої марківської послідовності....	431
<b>HMMVITERBI</b>	Відновлення послідовності станів .....	450
<b>HMMESTIMATE</b>	Оцінка параметрів моделі за відомої послідовності станів .....	453
<b>HMMTRAIN</b>	Оцінка параметрів моделі за невідомої послідовності станів .....	456
<b>HMMDECODE</b>	Апостеріорні ймовірності станів .....	459

## **HMMGENERATE** Генерування схованої марківської послідовності

*Синтаксис:*

```
[seq,states] = hmmgenerate(len,TRANS,EMIS)  
hmmgenerate(...,'Symbols',SYMBOLS)  
hmmgenerate(...,'Statenames',STATENAMES)
```

*Опис:*

Функція **[seq,states] = hmmgenerate(len,TRANS,EMIS)** використовує марківську модель, задану матрицею ймовірностей переходу *TRANS* і матрицею ймовірностей емісії *EMIS*, генерує випадкові послідовності вихідних значень *seq* і станів *states* довжиною *len* кожна. Елементи мат-

риці  $TRANS(i, j)$  є ймовірності переходу зі стану  $i$  у стан  $j$ . Елементи матриці  $EMIS(k, l)$  є ймовірність того, що символ  $l$  емітується зі стану  $k$ .

**Зауваження.** Функція **hmmgenerate** використовує початковий стан з одиничною ймовірністю для 1-го стану й нульовим для всіх інших. Цей стан вважається станом номер 0 і не включається ані в *seq*, ані в *states*. Модель далі здійснює перехід у стан  $i_1$  з імовірністю  $T_{1i_1}$  й генерує значення  $a_{k_1}$  з імовірністю  $E_{i_1k_1}$ . Функція **hmmgenerate** повертає  $i_1$  як перше значення *states* і  $a_{k_1}$  як перше значення *seq*.

Функція **[...] = hmmgenerate(..., 'Symbols', symbols)** в аргументі *symbols* дозволяє задати символи для результативних значень. Це може бути числовий масив або масив символівних рядків. За замовчуванням використовуються цілі числа від 1 до  $n$ .

Функція **[...] = hmmgenerate(..., 'Statenames', statenames)** в аргументі *statenames* дозволяє задати імена для станів. Це може бути числовий масив або масив символівних рядків. За замовчуванням використовуються цілі числа від 1 до  $m$ .

*Приклади:*

1. У прикладі задаються ймовірності переходів із двох станів і ймовірностей шести значень у цих станах. Реалізована марківська послідовність із 10 елементів.

```
>> trans=[0.95,0.05; % імовірності переходу з 1-го стану
           0.10,0.90]; % імовірності переходу з 2-го стану
>> emis=[1/6 1/6 1/6 1/6 1/6 1/6; % імовірності значень в 1-му стані
          1/10 1/10 1/10 1/10 1/10 1/2]; % імовірності значень в 2-му стані
>> [seq,states]=hmmgenerate(10,trans,emis) % генерування послідовності
seq =
    1    5    6    2    2    6    1    6    6    6
states =
    1    1    1    1    1    1    1    1    2    2
```

Оскільки модель завжди починає працювати з першого стану, ймовірності переходів з яких представлені в першому рядку матриці *trans*, перше значення *states* може бути 1 з імовірністю 0.95 і 2 – з імовірністю 0.05. У прикладі ймовірності всіх значень у першому стані однакові.

2. У прикладі досліджується марківська модель із 3-ма станами й 4-ма значеннями. Початковий стан – перший.

```
>> TR=[0.5 0.3 0.2; % імовірності переходу з 1-го стану
       0.3 0.5 0.2; % імовірності переходу з 2-го стану
       0.2 0.3 0.5]; % імовірності переходу з 3-го стану
>> EM=[0.7 0.1 0.1 0.1; % імовірності значень в 1-му стані
       0.1 0.7 0.1 0.1; % імовірності значень в 2-му стані
       0.1 0.1 0.7 0.1]; % імовірності значень в 3-му стані
>> [seq,st]=hmmgenerate(10,TR,EM) % генерування послідовності
>> plot(seq) % графік
>> set(get(gcf,'Currentaxes'),'Fontname','Times New Roman Cyr','FontSize',16)
>> title('\bfМарківська послідовність') % заголовок
>> xlabel('Номер елемента') % мітка осі OX
>> ylabel('Значення виходу') % мітка осі OY
>> grid on
seq =
    1    2    3    4    2    1    2    2    3    3
st =
    1    1    3    2    2    1    2    2    3    1
```

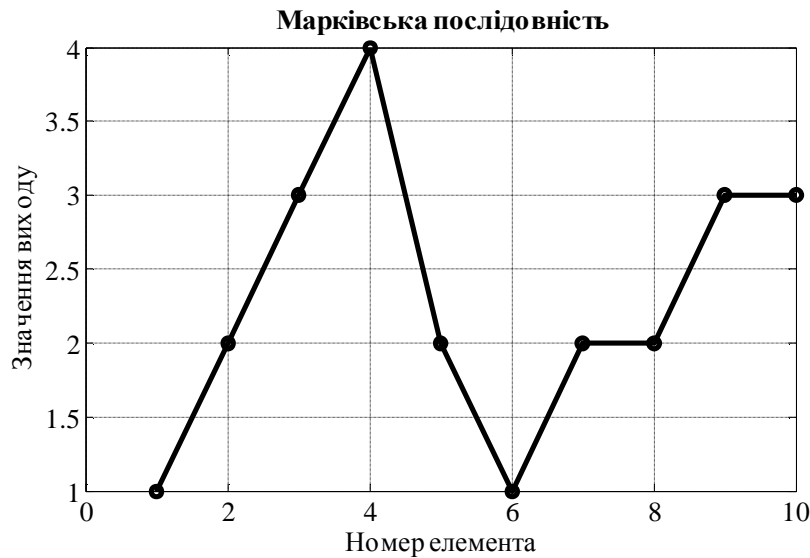


Рис. 20.3. Одна реалізація марківської послідовності з 10 елементів

На рис. 20.3 показано графік однієї з реалізацій марківської послідовності з 20 елементів. Послідовність перебуває у вихідному параметрі *seq*.

3. У прикладі за тих же матриць переходів і станів, що в прикладі 1, у функції задаються символи для вихідних значень і імена для станів.

```
>> trans=[0.95,0.05;
```

```

0.10,0.90];
>> emis = [1/6 1/6 1/6 1/6 1/6 1/6;
           1/10 1/10 1/10 1/10 1/10 1/2];
>> [seq,states]=hmmgenerate(10,trans,emis,...
'Symbols',{'one','two','three','four','five','six'},...
'Statenames',{'home';'work'})
seq =
Columns 1 through 10
'six' 'six' 'six' 'six' 'two' 'five' 'six' 'six' 'three' 'six'
states =
Columns 1 through 10
'work' 'work' 'work' 'work' 'work' 'work' 'work' 'work' 'home' 'home' 'home'

```

### **HMMVITERBI** Відновлення послідовності станів

За заданою послідовністю вихідних значень відновлюється найбільш імовірна послідовність станів, використовуючи алгоритм Вітербі (Viterbi).

*Синтаксис:*

```

STATES = hmmviterbi(seq,TRANS,EMIS)
hmmviterbi(...,'Symbols',SYMBOLS)
hmmviterbi(...,'Statenames',STATENAMES)

```

*Опис:*

Функція **STATES = hmmviterbi(seq,TRANS,EMIS)** обчислює максимально правдоподібну послідовність станів, використовуючи послідовність значень *seq*, що побудована функцією **hmmgenerate**. Аргумент *TRANS* – матриця ймовірностей переходу, а *EMIS* – матриця ймовірностей емісії використовуваної схованою марківською моделлю. Їх формат описаний у попередній функції.

Алгоритм Вітербі, що реалізований у функції **hmmviterbi**, припускає, що перед першим кроком система перебувала в стані 1. Якщо потрібно використовувати інші початкові стани, можна використовувати наступне приймання.

Нехай треба задати інший початковий розподіл ймовірностей у вигляді вектора-рядка  $p = (p_1, p_2, \dots, p_n)$ , сума елементів якого дорівнює 1. Це можна зробити так.

Побудувати нову матрицю ймовірностей переходу:  $T_1 = \begin{pmatrix} 0 & p \\ 0 & T \end{pmatrix}$ , де

$T$  – матриця ймовірностей переходу *TRANS*;

Побудувати нову матрицю ймовірностей емісії:  $E_1 = \begin{pmatrix} 0 \\ E \end{pmatrix}$ , де  $E$  –

матриця ймовірностей емісії *EMIS*;

Викликати функцію **hmmgenerate** із цими новими аргументами.

Функція **states = hmmviterbi(...,'Symbols',symbols)** в аргументі *symbols* дозволяє задати символи для вихідних значень. Це може бути числовий масив або масив символічних рядків. За замовчуванням використовуються цілі числа від 1 до  $n$ , де  $n$  – кількість вихідних значень.

Функція **states = hmmviterbi(...,'Statenames',statenames)** в аргументі *statenames* дозволяє задати імена для станів. Це може бути числовий масив або масив символічних рядків. За замовчуванням використовуються цілі числа від 1 до  $m$ , де  $m$  – кількість станів.

*Приклади:*

1. У прикладі досліджується та ж марківська модель із 3-ма станами й 4-ма виходами, що й у прикладі для попередньої функції. Для неї генерується послідовність із 100 вихідних значень, по них відновлюються найбільш імовірні стани, які порівнюються з дійсними.

```
>> TR=[0.5 0.3 0.2; % імовірності переходу з 1-го стану
        0.3 0.5 0.2; % імовірності переходу з 2-го стану
        0.2 0.3 0.5]; % імовірності переходу з 3-го стану
>> EM=[0.7 0.1 0.1 0.1; % імовірності значень в 1-му стані
        0.1 0.7 0.1 0.1; % імовірності значень в 2-му стані
        0.1 0.1 0.7 0.1]; % імовірності значень в 3-му стані
>> n=100;
>> [seq,st]=hmmgenerate(n,TR,EM); % генерування послідовності
>> estst=hmmviterbi(seq,TR,EM); % відновлюємо стан
>> st10=reshape(st,10,n/10) % порції по 10 елементів вихідної послідовності
st10 =
     1     2     2     1     1     2     3     1     3     3
     3     2     2     2     2     2     2     2     1     2     3
     3     2     2     2     1     3     3     3     1     1     2
     3     2     2     1     2     1     1     2     2     2     2
     2     2     1     2     1     3     2     2     2     2     2
     2     3     2     3     2     3     1     1     1     1     1
```

```

3 2 3 3 2 3 3 3 1 3
2 2 3 3 3 2 2 3 1 3
2 2 3 3 3 3 2 3 1 3
2 2 2 1 3 3 1 3 1 1
>> estst10=reshape(estst,10,n/10)
% порції по 10 елементів оптимальної послідовності.
estst10 =
3 2 2 1 1 2 3 1 3 1
3 2 2 2 2 2 2 1 2 3
3 3 2 2 1 3 3 3 1 3
1 2 2 1 2 2 1 2 1 2
2 2 1 2 1 3 2 1 2 2
2 3 1 2 2 3 1 1 1 2
2 2 3 3 2 3 3 1 1 2
3 2 3 3 3 3 2 3 1 2
3 2 2 3 1 3 1 3 1 3
2 2 2 1 3 3 1 3 1 3
>> perc=sum(estst10==st10) % число збігів
perc =
9 8 7 9 8 6 5 9 8 7
>> plot(perc), grid on % графік
>> set(get(gcf,'Currentaxes'),'Fontname','Times New Roman Cyr','FontSize',16)
>> title('\bfКількість збігів станів') % заголовок
>> xlabel('\bfНомер десятки') % мітка осі ОХ
>> ylabel('\bfЧисло збігів') % мітка осі ОУ

```

На рис. 20.4 показано кількість збігів за кожними 10 послідовними елементами, на які попередньо було розбито обидві послідовності.

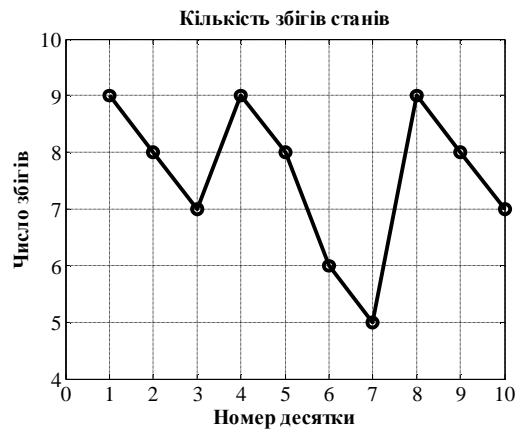


Рис. 20.4. Кількість збігів

2. У цьому прикладі значенням станів привласнені імена. Генерується марківська послідовність із двома переходами й двома станами, що складаються зі ста елементів, далі генерується оптимальна

послідовність і без розбиття шляхом порівняння визначається число збігів.

```
>> trans=[0.95,0.05;
           0.10,0.90];
>> emis=[1/6, 1/6, 1/6, 1/6, 1/6, 1/6;
          1/10, 1/10, 1/10, 1/10, 1/10, 1/2;];
>>[seq,states]=hmmgenerate(100,trans,emis,'Statenames',{'home';'work'})
states =
Columns 1 through 7
'home' 'home' 'work' 'work' 'work' 'work' 'work'
>>States=hmmviterbi(seq,trans,emis,'Statenames',{'home';'work'})
States =
Columns 1 through 7
'home' 'home' 'home' 'home' 'home' 'home' 'home'
>> i=strcmp(States, states)
i =
Columns 1 through 12
1 1 0 0 0 0 0 0 0 0 0 1
>> d=sum(i)
d =
76
```

У результаті одержано 76 збігів. У прикладі показані перші елементи двох послідовностей. Порівняння провадиться логічною функцією *strcmp*, тому що послідовності є масивами типу *cell*. Далі підсумовується кількість збігів.

### **HMESTIMATE** Оцінка параметрів моделі за відомої послідовності станів

За заданими послідовностей станів і вихідних значень відновлюються найбільш імовірні матриці ймовірностей переходу й ймовірностей емісії.

*Синтаксис:*

```
[TRANS, EMIS] = hmestimate(seq, states)
hmestimate(..., 'Symbols', SYMBOLS)
hmestimate(..., 'Statenames', STATENAMES)
hmestimate(..., 'Pseudoemissions', PSEUDOEM)
hmestimate(..., 'Pseudotransitions', PSEUDOTR)
```

*Опис:*

Функція **[TRANS,EMIS] = hmestimate(seq,states)** обчислює максимально правдоподібні оцінки матриць ймовірностей переходу

*TRANS* і ймовірностей емісії *EMIS* за заданими послідовностями значень *seq* і станів *states*.

Функція `[...] = hmestimate(..., 'Symbols', symbols)` в аргументі *symbols* дозволяє задати символи для вихідних значень. Це може бути числовий масив або масив символічних рядків. За замовчуванням використовуються цілі числа від 1 до  $n$ , де  $n$  – кількість вихідних значень.

Функція `[...] = hmestimate(..., 'Statenames', statenames)` в аргументі *statenames* дозволяє задати імена для станів. Це може бути числовий масив або масив символічних рядків. За замовчуванням використовуються цілі числа від 1 до  $m$ , де  $m$  – кількість станів.

Функція `[...] = hmestimate(..., 'Pseudoemissions', PSEUDOE)` – в аргументі *PSEUDOE* дозволяє задати передбачувані значення елементів матриці ймовірностей емісії. Цей аргумент можна використовувати, щоб уникнути нульових оцінок ймовірностей емісії, якщо в послідовності *seq* якесь вихідне значення не представлено. Цей аргумент має бути матрицею  $m \times n$ , де  $m$  – кількість станів, а  $n$  – кількість вихідних значень.

Функція `[...] = hmestimate(..., 'Pseudotransitions', PSEUDOTR)` в аргументі *PSEUDOTR* дозволяє задати передбачувані значення елементів матриці ймовірностей переходу. Цей аргумент можна використовувати, щоб уникнути нульових оцінок ймовірностей переходу, якщо в послідовності *states* даний стан не представлений. Цей аргумент повинен бути матрицею  $m \times m$ , де  $m$  – кількість станів.

Пояснимо зміст і застосування аргументів *Pseudotransitions* і *Pseudoemissions*.

Якщо ймовірність будь-якого конкретного переходу або емісії дуже мала, перехід або емісія може не відбутися. У кожному такому випадку, алгоритм повертає ймовірність 0 для даного переходу або емісії в *TRANS* або *EMIS*. Ви можете компенсувати відсутність переходу за допомогою аргументів '*Pseudotransitions*' і '*Pseudoemissions*'. Найпростіший спосіб зробити це полягає в тому, щоб установити в

*PSEUDO* або *PSEUDOTR* значення рівне 1. Наприклад, якщо перехід  $i \rightarrow j$  не відбувається, слід установити  $PSEUDOTR(i, j) = 1$ . Це приведе до того, що  $TRANS(i, j)$  буде позитивним. Якщо ви маєте оцінку для очікуваного переходу  $i \rightarrow j$  в послідовності тієї ж довжини що й стану, а фактичне число переходів менше, чим ця оцінка, ви можете встановити в  $PSEUDOTR(i, j)$  необхідне число. Це збільшить значення  $TRANS(i, j)$ . Для переходів, які відбуваються із частотою, яку ви очікуєте, установлюють відповідно значення *PSEUDOTR* рівне 0, яке не збільшує *TRANS*.

*Приклади:*

1. Приклад на основі марківської моделі із трьома переходами й чотирма значеннями станів з матрицями ймовірностей переходу і ймовірностей емісії:

$$TR = \begin{pmatrix} 0.5 & 0/3 & 0.2 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}; \quad EM = \begin{pmatrix} 0.7 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.7 & 0.1 \end{pmatrix}.$$

```
>> TR=[0.5 0.3 0.2; % імовірності переходу з 1-го стану
      0.3 0.5 0.2; % імовірності переходу з 2-го стану
      0.2 0.3 0.5]; % імовірності переходу з 3-го стану
>> EM=[0.7 0.1 0.1 0.1; % імовірності значень в 1-му стані
      0.1 0.7 0.1 0.1; % імовірності значень в 2-му стані
      0.1 0.1 0.7 0.1]; % імовірності значень в 3-му стані
>> n=10000;
>> [seq,st]=hmmgenerate(n,TR,EM); % генерування послідовності
>> [etr,eem]=hmmestimate(seq,st); % оцінюємо матриці
>> disp('Оцінка матриці ймовірностей переходу')
>> fprintf('%12.8f %12.8f %12.8f\n',etr)
>> disp('Оцінка матриці ймовірностей емісії')
>> fprintf('%12.8f %12.8f %12.8f %12.8f\n',eem)
Оцінка матриці ймовірностей переходу
 0.48793893  0.30473282  0.20732824
 0.29705336  0.49907088  0.20387576
 0.18836659  0.30064254  0.51099087
Оцінка матриці ймовірностей емісії
 0.68885496  0.09465649  0.10442748  0.11206107
 0.09822140  0.70560127  0.09822140  0.09795593
 0.10750507  0.10513861  0.68864097  0.09871535
```

На виході методом максимальної правдоподібності отримані оптимальні матриці.

2. Розглядається марківська модель із матрицями переходів і емісії

$$TRANS = \begin{pmatrix} 0.95 & 0.05 \\ 0.1 & 0.9 \end{pmatrix}; \quad EMIS = \begin{pmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{2} \end{pmatrix}.$$

Генерується послідовність із 1000 елементів. Обчислюються оптимальні матриці.

```
>> trans=[0.95,0.05; 0.10,0.90];
>> emis=[1/6, 1/6, 1/6, 1/6, 1/6, 1/6;
         1/10, 1/10, 1/10, 1/10, 1/10, 1/2];
>> [seq,states]=hmmgenerate(1000,trans,emis);
>> [estimatetr,estimatee]=hmmestimate(seq,states)
estimatetr =
    0.9527    0.0473
    0.0795    0.9205
estimatee =
    0.1593    0.1625    0.1562    0.1593    0.1877    0.1751
    0.0984    0.0956    0.1038    0.1284    0.1311    0.4426
```

## **HMMTRAIN** Оцінка параметрів моделі за невідомої послідовності станів

За заданою послідовністю вихідних значень відновлюються найбільш імовірні матриці ймовірностей переходу й ймовірностей емісії.

*Синтаксис:*

```
[ESTTR, ESTEMIT] = hmmtrain(seq,TRGUESS,EMITGUESS)
hmmtrain(...,'Algorithm', algorithm)
hmmtrain(...,'Symbols', SYMBOLS)
hmmtrain(...,'Tolerance', tol)
hmmtrain(...,'Maxiterations', maxiter)
hmmtrain(...,'Verbose', true)
hmmtrain(...,'Pseudoemissions', PSEUDOE)
hmmtrain(...,'Pseudotransitions',PSEUDOTR)
```

*Опис.*

Функція **[ESTTR,ESTEMIT] = hmmtrain(seq,TRGUESS,EMITGUESS)** обчислює оцінки матриць ймовірностей переходу *ESTTR* і ймовірностей емісії *ESTEMIT* схованої марківської моделі за заданою послідовністю вихідних значень *seq*, використовуючи алгоритм Баума-Уэлча (Baum-Welch). Аргумент *seq* може бути вектором, що містить одну

послідовність, матрицею, кожний рядок якої містить послідовність або масив рядків, що містять послідовності. Аргументи *TRGUESS* і *EMITGUESS* – це початкові оцінки матриць, вони повинні мати такі ж розміри, як і самі матриці. *TRGUESS(i, j)* є оцінка ймовірності переходу зі стану *i* у стан *j*. *EMITGUESS(i, k)* є оцінка ймовірності того, що значення (символ) *k* емітується зі стану *i*.

Функція [...] = **hmmtrain(...,'Algorithm',algorithm)** в аргументі *algorithm* визначає використовуваний алгоритм. Цей аргумент повинен бути текстовим рядком. Можливі значення: *'Baumwelch'* (за замовчуванням) або *'Viterbi'*.

Функція [...] = **hmmtrain(...,'Symbols',symbols)** в аргументі *symbols* дозволяє задати символи для вихідних значень. Це може бути числовий масив або масив символічних рядків. За замовчуванням використовуються цілі числа від 1 до *n*, де *n* – кількість вихідних значень.

Функція [...] = **hmmtrain(...,'Tolerance',tol)** в аргументі *tol* визначає точність для критерію збіжності в ітераційному алгоритмі (за замовчуванням 10<sup>-4</sup>).

Функція [...] = **hmmtrain(...,'Maxiterations',maxiter)** в аргументі *maxiter* визначає максимальну кількість ітерацій в ітераційному алгоритмі (за замовчуванням 100).

Функція [...] = **hmmtrain(...,'Verbose',true)** видає стан алгоритму на кожній ітерації.

Функція [...] = **hmmtrain(...,'Pseudoemissions',PSEUDOIE)** в аргументі *PSEUDOIE* дозволяє задати передбачувані значення елементів матриці ймовірностей емісії, якщо використовується алгоритм Вітербі. Цей аргумент можна використовувати, щоб уникнути нульових оцінок ймовірностей емісії, якщо в послідовності *seq* якесь значення не представлено. Цей аргумент має бути матрицею  $m \times n$ , де *m* – кількість станів, а *n* – кількість вихідних значень.

Функція [...] = **hmmtrain(...,'Pseudotransitions',PSEUDOTR)** в аргументі *PSEUDOTR* дозволяє задати передбачувані значення елементів матриці ймовірностей переходу, якщо використовується

алгоритм Вітербі. Цей аргумент можна використовувати, щоб уникнути нульових оцінок ймовірностей переходу, якщо в послідовності *states* даний стан не представлений. Цей аргумент повинен бути матрицею  $m \times m$ , де  $m$  – кількість станів.

Окремо пояснимо зміст двох аргументів – *Tolerance* і *Maxiterations*.

Вхідний аргумент '*Tolerance*' контролює кількість кроків виконання алгоритму функції до повернення відповіді.

Аргумент *Maxiterations* контролює число максимальних кроків алгоритму. Якщо за заданого значення *maxiter* не досягається зазначена точність, то функція повертає попередження. У цьому випадку можна збільшити значення '*maxiterations*', щоб досягнути задану точність.

*Приклади:*

1. Генеруються дві марківські послідовності, що містять 100 і 200 елементів. За ними оцінюються найбільш оптимальні матриці переходів і емісій.

```
>> tr=[0.95,0.05;  
       0.10,0.90];  
>> e=[1/6, 1/6, 1/6, 1/6, 1/6, 1/6;  
      1/10, 1/10, 1/10, 1/10, 1/10, 1/2;];  
>> seq1=hmmgenerate(100,tr,e);  
>> seq2=hmmgenerate(200,tr,e);  
>> seqs={seq1,seq2};  
>> [esttr,este]=hmmtrain(seqs,tr,e)  
esttr =  
    0.9876    0.0124  
    0.0109    0.9891  
este =  
    0.1717    0.1576    0.2126    0.1915    0.1265    0.1401  
    0.1329    0.1895    0.0512    0.1275    0.1184    0.3805
```

2. У прикладі генерується марківська послідовність із трьома переходами й станами й із чотирма значеннями, що полягає з 10000 елементів. Задається марківська модель із такими ж параметрами своїми матрицями переходів і емісій. Функція оцінює з точністю  $2 \cdot 10^{-4}$  найбільш оптимальні матриці переходів і емісій.

```
>> TR=[0.5 0.3 0.2; % імовірності переходу з 1-го стану  
       0.3 0.5 0.2; % імовірності переходу з 2-го стану  
       0.2 0.3 0.5]; % імовірності переходу з 3-го стану  
>> EM=[0.7 0.1 0.1 0.1; % імовірності значень в 1-му стані
```

```

0.1 0.7 0.1 0.1; % імовірності значень в 2-му стані
0.1 0.1 0.7 0.1]; % імовірності значень в 3-му стані
>> n=10000;
>> [seq,st]=hmmgenerate(n,TR,EM); % генерування послідовності
>> TRG=[0.4 0.3 0.4; % імовірності переходу з 1-го стану
0.3 0.4 0.3; % імовірності переходу з 2-го стану
0.3 0.3 0.4]; % імовірності переходу з 3-го стану
>> EMG=[0.4 0.2 0.2 0.2; % імовірності значень в 1-му стані
0.2 0.4 0.2 0.2; % імовірності значень в 2-му стані
0.2 0.2 0.4 0.2]; % імовірності значень в 3-му стані
>> [etr,eem]=hmmtrain(seq,TRG,EMG,'Tolerance',2e-4); % оцінюємо
>> disp('Оцінка матриці ймовірностей переходу')
>> fprintf('%12.8f %12.8f %12.8f\n',etr')
>> disp('Оцінка матриці ймовірностей емісії')
>> fprintf('%12.8f %12.8f %12.8f %12.8f\n',eem')
Оцінка матриці ймовірностей переходу
0.58584014 0.23143265 0.18272721
0.29644797 0.48463573 0.21891630
0.17863759 0.24757423 0.57378818
Оцінка матриці ймовірностей емісії
0.60409006 0.17745234 0.11166937 0.10678823
0.13787256 0.69396429 0.07419274 0.09397041
0.12957097 0.11730603 0.64014897 0.11297403

```

## **HMMDECODE** Апостеріорні ймовірності станів

Для кожного елемента послідовності вихідних значень функція обчислює умовні ймовірності того, що система перебуває в кожному  $k$ -му стані, за умови, що отримане саме це вихідне значення.

*Синтаксис:*

```

PSTATES = hmmdecode(seq, TRANS, EMIS)
[PSTATES, logpseq] = hmmdecode(seq, TRANS, EMIS)
[PSTATES, logpseq, FORWARD, BACKWARD, S] =
hmmdecode(seq,TRANS,EMIS)
hmmdecode(...,'Symbols', SYMBOLS)

```

*Опис.*

Функція **PSTATES = hmmdecode(seq,TRANS,EMIS)** обчислює апостеріорні ймовірності станів *PSTATES* за заданою вихідною послідовністю *seq*, матриці ймовірностей переходу *TRANS* і матриці ймовірностей емісії *EMIS*. Вихідний параметр *PSTATES* – матриця розміром  $m \times p$ , де  $m$  – кількість станів системи, а  $p$  – довжина

послідовності  $seq$ . Кожний елемент  $PSTATES(i,j)$  – це ймовірність того, що система перебувала в  $i$ -му стані, за умови, що після  $j$ -го кроку було отримано вихідне значення  $seq(j)$ .

Функція **hmmdecode** використовує початковий стан (перед першим переходом) з одиничною ймовірністю для 1-го стану й нульовим для всіх інших. Алгоритм задання інших початкових станів розглянуто у попередніх функціях.

Функція **[PSTATES,logpseq] = hmmdecode(...)** у результативному параметрі  $logpseq$  повертає логарифм імовірності побудови тієї послідовності  $seq$ , яка побудована. Зі збільшенням довжини послідовності ймовірність її побудови прагне до нуля, тому обчислюється її логарифм.

Функція **[PSTATES,logpseq,forward,backward,s] = hmmdecode(...)** додатково повертає вихідні й переоцінені ймовірності (в оригіналі: the forward and backward probabilities) для елементів послідовності, що масштабовані по  $s$ . Дійсні значення цих ймовірностей можна одержати так:

```
>> f=forward.*repmat(cumprod(s),size(forward,1),1);  
>> bscale = fliplr(cumprod(fliplr(s)));  
>> b = backward.*repmat([bscale(2:end), 1],size(backward,1),1);
```

Функція **[...] = hmmdecode(...,'Symbols',symbols)** в аргументі  $symbols$  дозволяє задати символи для вихідних значень. Це може бути числовий масив або масив символівних рядків. За замовчуванням використовуються цілі числа від 1 до  $n$ , де  $n$  – кількість вихідних значень.

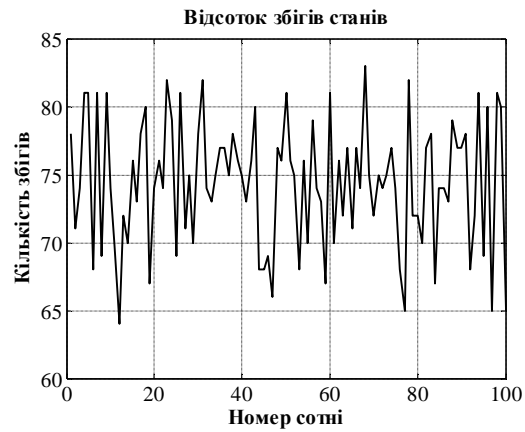
У прикладі досліджується та ж марківська модель, що й у попередній функції. Для неї генерується послідовність із 10000 вихідних значень, за ними обчислюються апостеріорні ймовірності станів. Стан з максимальною ймовірністю порівнюється з дійсними. На рис. 20.5 показана кількість збігів за кожним із 100 послідовних елементів.

```
>> TR=[0.5 0.3 0.2; % імовірності переходу з 1-го стану  
      0.3 0.5 0.2; % імовірності переходу з 2-го стану  
      0.2 0.3 0.5]; % імовірності переходу з 3-го стану  
>> EM=[0.7 0.1 0.1 0.1; % імовірності значень в 1-му стані  
      0.1 0.7 0.1 0.1; % імовірності значень в 2-му стані
```

```

0.1 0.1 0.7 0.1]; % імовірності значень в 3-му стані
>> n=10000;
>> [seq,st]=hmmgenerate(n,TR,EM); % генерування послідовності
>> ps=hmmdecode(seq,TR,EM); % оцінюємо ймовірності станів
>> st100=reshape(st,100,n/100); % порції по 100 елементів
>> [m,im]=max(ps);
>> estst100=reshape(im,100,n/100);
>> perc=sum(estst100==st100); % кількість угадувань
>> plot(perc) % графік
>> set(get(gcf,'Currentaxes'),'Fontname','Times New Roman Cyr','FontSize',16)
>> title('\bfВідсоток збігів станів') % заголовок
>> xlabel('\bfНомер сотні') % мітка осі ОХ
>> ylabel('\bfКількість збігів') % мітка осі ОУ

```



**Рис. 20.5. Кількість збігів апіорних і апостеріорних ймовірностей елементів марківської послідовності**



## 21. Функції введення, форматування й збереження інформації

Матеріал цього розділу будемо вивчати за темами:

21.1. Введення даних .....	463
21.2. Форматування даних .....	490
21.3. Збереження інформації .....	492

### 21.1. Введення даних

Обробка даних починається з уведення інформації. Розглянемо найбільш типові види файлів даних і функції або команди введення інформації з цих файлів:

<b>LOAD</b> Введення числових даних з txt-файла .....	464
<b>TBLREAD</b> Введення даних з форматованої таблиці .....	465
<b>CASEREAD</b> Введення символної інформації .....	466
<b>TEXTREAD</b> Універсальна функція введення даних .....	467
<b>TYPE</b> Функція читання даних з файлів різних форматів .....	470
<b>FOPEN</b> Відкривання файла .....	470
<b>FREAD</b> Читання бінарних файлів .....	472
<b>FOPEN, FGETL</b> Читання даних за рядками .....	477
<b>EXEL LINK</b> Зв'язок MATLAB з EXCEL .....	478
<b>XLSREAD</b> Введення даних з xls-файла Excel .....	481
<b>File/Open</b> Відкрити файл засобами Windows .....	483
<b>File/Import Data</b> Імпорт даних у робочу область Workspace .....	485
<b>UIIMPORT</b> Відкриває Майстер імпорту .....	489

## **LOAD** Введення числових даних з txt-файла

Найчастіше числові дані зберігаються у звичайному текстовому файлі з розширенням \*.txt. Він може бути створений у будь-якому текстовому редакторі. Дані мають бути записані у вигляді прямокутної таблиці чисел, розділених у кожному рядку пропусками й утримуючих в рядках однакову кількість чисел. Десятковий роздільник – точка. Перед числом допускається знак "+" або "-". Будь-яке число може бути задане в експоненціальній формі, наприклад, -12.4587E+04 або 3.9e-2. Де б не зберігався файл даних, його краще скопіювати в піддиректорію work основної директорії **MatLab** (тоді можна не вказувати шлях до файла даних).

*Синтаксис:*

Команда **A=load('ім'я файла')** вводить у командне вікно **MatLab** дані із txt-файла у вигляді матриці *A*.

*Приклад:*

Нехай у редакторі "Блокнот" створена таблиця чисел розміром  $2 \times 5$ :

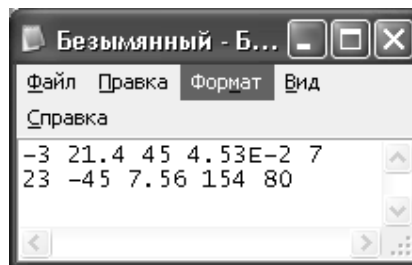


Рис. 21.1. Таблиця чисел у редакторі "Блокнот"

Збережемо цей файл під іменем mdt.txt в директорії work.

Вводимо ці дані у командне вікно **MatLab**:

```
>> M=load('mdt.txt')
M =
-3.0000 21.4000 45.0000 0.0453 7.0000
23.0000 -45.0000 7.5600 154.0000 80.0000
```

## **TBLREAD** Введення даних з форматованої таблиці

Статистична інформація в текстовому файлі може бути доповнена заголовками рядків (записи) і стовпців (поля). Заголовки стовпців, що розділяються пропусками або іншими символами, утворюють перший рядок таблиці. Заголовки рядків утворюють перший стовпець і можуть розділятися тими ж символами. Дані цієї форматованої таблиці зручно вводити за допомогою функції **TBLREAD**.

*Синтаксис:*

```
[data,varnames,casenames] = tblread  
[data,varnames,casenames] = tblread('filename')  
[data,varnames,casenames] = tblread(filename,delimiter)
```

*Опис:*

Функція **[data,varnames,casenames] = tblread** відображає стандартне діалогове вікно відкриття файлу, у якому можна вибрати для відкриття файл із табульованими даними. Повертає числові дані в матрицю *data*, заголовки полів – у матрицю символів *varnames* (за рядками, більш короткі рядки доповнюються пропусками) і імена записів – у матрицю символів *casenames* (також за рядками, більш короткі рядки доповнюються пропусками).

Функція **[data,varnames,casenames] = tblread('filename')** відкриває відразу файл з ім'ям *'filename'*, якщо він перебуває в директорії **MatLab** (інакше треба вказувати повний шлях до нього).

Функція **[data,varnames,casenames] = tblread('filename','delimiter')** в аргументі *'delimiter'* дозволяє задати символ-роздільник полів у кожному запису. Цей аргумент має бути текстовим рядком і може приймати наступні значення:

- ' '* або *'space'* – пропуск (за замовчуванням);
- '\t'* або *'tab'* – символ табулювання;
- ' , '* або *'comma'* – кома;
- ' ; '* або *'semi'* – крапка з комою;
- ' / '* або *'bar'* – вертикальна риска.

Приклад:

В директорії C:\Program Files\MATLAB71\toolbox\stats знаходиться файл sat.dat із таблицю:

	Male	Female
Verbal	470	530
Quantitative	520	480

Прочитаємо цю інформацію функцією **tblread**:

```
>> [data,varnames,casenames]=tblread
```

З'являється діалогове вікно (рис. 21.2), в якому знаходимо файл sat.dat і клацаємо по кнопці **Открыть** (повідомлення русифікованої версії **Windows** наводимо курсивом, не перекладаючи їх на українську мову).

В результаті одержуємо:

data =	varnames =	casenames =
470 530	Male	Verbal
520 480	Female	Quantitative

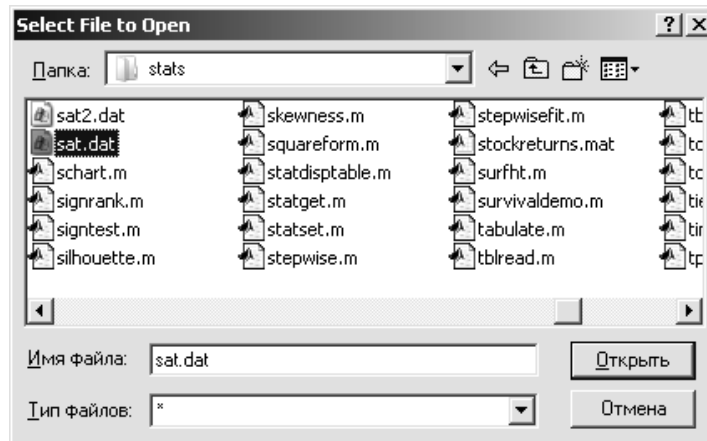


Рис. 21.2. Вікно відкриття файла

Оскільки файл sat.dat знаходиться у піддиректорії **MatLab**, то можна прочитати дані відразу командою

```
>> [data,varnames,casenames]=tblread('sat.dat')
```

### **CASEREAD** Введення символної інформації

Функція **caseread** використовується, якщо необхідне читання текстового файла як прямокутного масиву символів без перетворення в числа.

*Синтаксис:*

```
names = caseread  
names = caseread('filename')
```

*Опис:*

Функція **names = caseread** відображає стандартне діалогове вікно відкриття файла, у якому можна вибрати для відкриття текстовий файл. Символи кирилиці вводяться в тому кодуванні, у якому вони записані у файлі, байт в байт.

Функція **names = caseread('filename')** відразу відкриває файл з ім'ям *'filename'*, якщо він записаний у поточній директорії

*Приклад:*

```
>> A=caseread('sat.dat')
```

```
A =
```

	Male	Female
Verbal	470	530
Quantitative	520	480

Перевіримо тип змінної *A*:

```
>> whos A
```

Name	Size	Bytes	Class
A	3x32	192	char array

Grand total is 96 elements using 192 bytes

## **TEXTREAD** Універсальна функція введення даних

Читання будь-яких числових і текстових даних може бути проведене за допомогою функції **textread**. В ній є множина ключів, що дозволяють настроїти формат вводу даних, символи-роздільники даних і інше. Але, щоб користуватися цими можливостями, треба знати формат файла (див. >> **help textread**). Альтернативою цієї функції може бути застосування **Import Wizard**.

*Синтаксис:*

```
A = textread('filename')  
[A,B,C,...] = textread('filename','format')  
[A,B,C,...] = textread('filename','format',N)
```

Опис:

Функція **A = textread('filename')** використовується, коли всі дані у файлі одного формату, наприклад, числові; функція зчитує дані з файла *'filename'* у змінну *A*.

Функція **[A,B,C,...] = textread('filename','format')** зчитує дані з файла з іменем *'filename'* у змінні *A*, *B*, *C* тощо, використовуючи специфікатори *format*. Ця функція корисна для читання текстів файлів з відомими форматами тексту. Число й тип змінних визначається специфікаторами форматів, які вказуються в рядковій змінній або в рядку *format*. Специфікатори форматів наведено в табл. 21.1.

Таблиця 21.1

### Специфікатори форматів функції textread

Специфікатор	Дія	Формат даних
% d	Читання цілого числа зі знаком	Числовий масив
% u	Читання цілого числа	Числовий масив
% f	Читання дійсного числа із плаваючою крапкою	Числовий масив
% s	Читання рядка із пропусками, символами табуляції або інших символів	Масив символівних рядків
% c	Читання символів, у тому числі й пробілів	Масив символів
% []	Читання рядка, що містить символи у квадратних дужках	Масив символівних рядків
% *	Пропускає відповідну позицію	

Перед кожною буквою специфікатора можна поставити натуральне число (цифру). У цьому випадку буде зчитуватися дана кількість символів.

У функції **[A,B,C,...] = textread('filename','format',N)** аргумент *N* визначає кількість рядків, що треба прочитати. Він задається як ціле позитивне число.

Приклад:

1. Зчитування числових даних.

```
>> m=textread('mdt.txt')
m =
-3.0000 21.4000 45.0000 0.0453 7.0000
23.0000 -45.0000 7.5600 154.0000 80.0000
>> whos m
Name      Size      Bytes   Class
m         2x5       80     double array
```

Grand total is 10 elements using 80 bytes

2. У текстовому редакторі Блокнот складемо файл із табличною структурою, зміст якого показано на рис. 21.3, і запам'ятемо його в поточному каталозі під іменем scan2.txt.

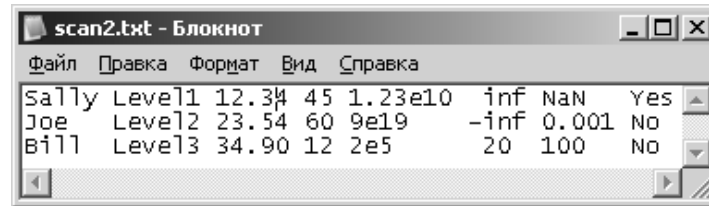


Рис. 21.3. Зміст файла scan2.txt

Перенесемо змінні цього файла в робочий простір **MatLab**. Перший стовпець таблиці є масив символічних рядків *names*; другий стовпець – символічний масив (рядки однакової довжини) *types*; третій і четвертий стовпці – числові масиви *x*, *y*; п'ятий і шостий стовпці пропускаємо; сьомий стовпець – числовий масив *z*; восьмий стовпець – масив символічних рядків *code*.

```
>> [names,types,x,y,z,code]=textread('scan2.txt','%s %s %f %d %f %*s %f %s')
names =      types =      x =      y =      z =      code =
'Sally'      'Level1'      12.3400      45      NaN      'Yes'
'Joe'        'Level2'      23.5400      60      0.001      'No'
'Bill'       'Level3'      34.9000      12      100        'No'

>> whos
Name      Size      Bytes Class
code      3x1      194 cell array
names     3x1      204 cell array
types     3x1      216 cell array
x         3x1      24 double array
y         3x1      24 double array
z         3x1      24 double array
```

3. Прочитаємо лише один перший рядок файла 'scan2.txt' (без п'ятого і шостого стовпців):

```
>> [names,types,x,y,z,code]=textread('scan2.txt','%s %s %f %d %f %*s %f %s',1)
names =      types =      x =      y =      z =      code =
'Sally'      'Level1'      12.3400      45      NaN      'Yes'
```

## **TYPE** Функція читання даних з файлів різних форматів

Відкрити файл із розширеннями .txt, .dat, .m і іншими можна за допомогою функції **type**.

*Синтаксис:*

**type ім'я файла**

*Приклади:*

1. Читання даних з dat-файла:

**>> type sat.dat**

	Male	Female
Verbal	470	530
Quantitative	520	480

2. Введення даних з txt-файла:

**>> type mdt1.txt**

	Mex1	Mex2	Mex3	Mex4
Узел1	3	5	7	1
Узел2	6	0	2	8
Узел3	4	9	3	4

3. Читання тексту з m-файла

**>> type gauslike.m**

```
function z=gauslike(mu,sigma,p1) % Це текст m-файла
% Обчислюється функція правдоподібності нормального розподілу
n=length(p1); % кількість точок
z=ones(size(mu)); % спочатку одиниці
for i=1:n, % добуток щільностей розподілів
    z=z.*(normpdf(p1(i),mu,sigma));
end
```

## **FOPEN** Відкривання файла

Перш ніж робити будь-які операції з файлом, у тому числі читання або запис інформації, файл потрібно відкрити або створити. Це робить спеціальна функція **fopen**.

*Синтаксис.*

**fid = fopen('filename')**

**fid = fopen('filename', mode)**

**fids = fopen('all')**

Опис.

Функція **fid = fopen('filename')** відкриває файл відповідного імені і повертає *fid* – ціле число – ідентифікатор файла. Надалі ідентифікатор файла використовується як перший аргумент у функціях запису і читання інформації. Якщо файл не відкривається, то функція **fopen** повертає -1. Крім того, значення ідентифікатора *fid* = 1 означає (стандартний висновок) і *fid* = 2 (стандартна помилка).

Функція **fid = fopen('filename', mode)** відкриває файл відповідного імені (якщо файл не перебуває в поточній директорії сеансу, то треба прописати повний шлях до нього). Другий аргумент – так званий прапор відкриття файла, що говорить про спосіб доступу до файла. Його можливі значення наведено в табл. 21.2.

Таблиця 21.2

### Можливі значення аргументу *mode*

<i>mode</i>	Дії
'r'	Відкриває бінарний файл для читання
'w'	Відкриває або створює бінарний файл лише для запису. Попередній зміст файла стирається, а неіснуючий файл створюється
'a'	Відкриває файл або створює, додаючи інформацію в кінець файла
'r+'	Відкриває бінарний файл для читання або запису
'w+'	Відкриває бінарний файл для читання й запису, видаляючи існуючий зміст, якщо він є
'a+'	Відкриває або створює бінарний файл для читання й запису, додаючи інформацію в кінець файла
'rt'	Відкриває текстовий файл для читання
'rt+'	Відкриває текстовий файл для читання й запису
'wt'	Відкриває новий текстовий файл для запису, видаляючи існуючий зміст, якщо він є
'wt+'	Відкриває новий текстовий файл для читання й запису
'at'	Відкриває новий або існуючий текстовий файл для запису, додаючи інформацію в кінець файла
'at+'	Відкриває новий або існуючий текстовий файл для читання й запису, додаючи інформацію в кінець файла

Функція **fids = fopen('all')** повертає вектор-рядок, що містить ідентифікатори всіх файлів, що відкриваються (крім *fid* = 1 – стандартний висновок і *fid* = 2 – стандартна помилка). Номер елемента у векторі дорівнює номеру файла, що відкривається.

Запис інформації в бінарний файл здійснюється функцією **fwrite**. Попередньо файл відкривається за допомогою функції **fopen**, запис завершується функцією **fclose(fid)** – закрити файл із ідентифікатором *fid*.

Читання інформації з бінарного файла здійснюється функцією **fread**. Попередньо файл має бути відкритим функцією **fopen**.

### **FREAD** Читання бінарних файлів

*Синтаксис:*

**A = fread(fid)**

**[A,count] = fread(fid,size,precision)**

**A = fread(fid, count, precision, skip)**

*Опис:*

Функція **A = fread(fid)** читає дані бінарного файла з ідентифікатором *fid*, що привласнюється файлу функцією **fopen**, і записує дані в матрицю *A*.

Функція **[A,count] = fread(fid,size,precision)** читає бінарний файл і записує його в матрицю *A*. Результативний параметр *count* повертає кількість успішно прочитаних елементів файла з ідентифікатором *fid*. Аргумент *size* визначає, скільки елементів даних має бути прочитано. Якщо аргумент *size* не заданий, функція **fread** читає весь файл. Якщо *size = n*, функція читає *n* елементів стовпця; *size = inf* – файл читається до кінця, результат приводиться до вигляду стовпця, що містить кількість елементів, що й файл; *size = [m, n]* – дані зчитуються в матрицю розміром  $m \times n$ , причому, якщо даних менше, то матриця доповнюється нулями до заданої розмірності.

В табл. 21.3 наведені значення аргументу *precision*.

Функція **A = fread(fid, count, precision, skip)** включає аргумент *skip*, значення якого вказує на кількість байт, що пропускаються за читання кожного елемента, точність якого визначається аргументом *precision*.

Можливі значення аргументу *precision*

Значення	Опис
'schar'	Символ зі знаком, 8 біт
'uchar'	Символ без знака, 8 біт
'int8'	Ціле число, 8 біт
'int16'	Ціле число, 16 біт

Продовження табл. 21.3

Значення	Опис
'int32'	Ціле число, 32 біта
'int64'	Ціле число, 64 біта
'uint8'	Ціле число без знака, 8 біт
'uint16'	Ціле число без знака, 16 біт
'uint32'	Ціле число без знака, 32 біта
'uint64'	Ціле число без знака, 64 біта
'float32'	Дійсне число із плаваючою крапкою, 32 біта
'float64'	Дійсне число із плаваючою крапкою, 64 біта
'double'	Дійсне число із плаваючою крапкою, 64 біта (подвійна точність)

*Приклад:*

Вище було показано, що стандартні файлові функції системи **MatLab** досить зручні для запису (й для читання) числових даних у вигляді векторів і матриць, а також текстових даних. У цьому прикладі показано, як записати у файл такий складний тип даних пакета **MatLab** як масив символічних рядків різної довжини, а надалі прочитати його за допомогою функції **fread**.

Стандартні файлові функції не можуть здійснювати безпосередній запис у файли й читання масивів структур і символічних рядків, але для кожного конкретного типу масиву можна написати власні m-функції, що виконують цю роботу. Оскільки лише розроблювач знає структуру цих масивів, то бінарні файли з такими записами мають так званий "закритий" ("приватний" – private, custom) формат.

Отже створимо структуру даних і опишемо її.

```
>> Mysan=struct('field1',[1.2 3.4 5.45],'field2','Привіт');
>> Mucel(1,1)={'Прошу пробачення'};
>> Mucel(1,2)={[1.3 5.6 7;7.9 5.6 9.4;5 9 8]};
```

```
>> Mucel(2,1)={Mysan};
>> Mucel(2,2)={[1:7]};
```

Структура *Mysan* складається з двох полів, одне з яких заповнено числовим вектором, друге – текстом. Масив символічних рядків *Mucel* розміром  $2 \times 2$  складається із чотирьох елементів різних типів. Можна створити будь-який масив більшого розміру, але у процесі обробки ані розмір масиву, ані типи вхідних у них елементів вже не можуть змінюватися. Проте параметри кожного елемента можуть змінюватися (тобто задаватися розроблювачем) у широких межах.

Елемент *Mucel*(1, 1) містить текстовий рядок, у якого можуть змінюватися як її довжина, так і зміст. Елемент *Mucel*(1, 2) містить матрицю розміром  $m \times m$ , де натуральне число  $m$  може бути різним на різних стадіях роботи. Елемент *Mucel*(2, 1) завжди є структурою *Mysan*, у якої можуть змінюватися довжини її числових і текстових полів. І, нарешті, *Mucel*(2, 2) є числовим вектором-рядком довільної довжини. Запишемо в бінарний файл елементи масиву в переліченому порядку, разом з допоміжною інформацією, що характеризує змінні атрибути елементів, які описані вище.

Для цього у редакторі **MatLab** (File/New/M-file) напишемо тексти двох  $m$ -функції для запису створеного масиву в бінарний файл і читання масиву з бінарного файлу. Тексти  $m$ -функцій запам'ятаємо в поточному каталозі у вигляді  $m$ -файлів.

Для запису використовуємо функцію **fwrite**.

```
function Wrmyc(Ar) % записує бінарний файл
f1=fopen('Mybinfile1.bin','wb'); %відкриваємо файл для запису
len1=length(Ar{1,1}); % визначаємо розмір елемента (1,1)
fwrite(f1,len1,'float64'); %записуємо його
fwrite(f1,double(Ar{1,1}),'float64'); %записуємо сам елемент
[m,n]=size(Ar{1,2}); % визначаємо розмір елемента (1,2)
fwrite(f1,m,'float64'); % записуємо його
fwrite(f1,Ar{1,2},'float64'); % записуємо сам елемент
k1=length(Ar{2,1}.field1); % визначаємо розміри полів елемента (2,1)
k2=length(Ar{2,1}.field2);
fwrite(f1,k1,'float64'); % записуємо їх
fwrite(f1,k2,'float64');
fwrite(f1,Ar{2,1}.field1,'float64'); % записуємо поля
fwrite(f1,double(Ar{2,1}.field2),'float64');
len2=length(Ar{2,2});% визначаємо розмір елемента (2,2)
```

```

fwrite(f1,len2,'float64'); % записуємо його
fwrite(f1,Ar{2,2},'float64'); % записуємо сам елемент
fclose(f1); %закриваємо файл

```

Запам'ятовуємо цей текст під іменем *Wrmys.m* у поточній директорії.

Створюємо *m*-файл для читання масиву *Mucel*. Для цього використовуємо функцію **fread**.

```

function Cellar=Rwmys % функція читає бінарний файл
f1=fopen('Mybinfile1.bin','rb');% відкриваємо сам файл
len1=fread(f1,[1,1],'float64');
Cellar(1,1)={char(fread(f1,[1,len1],'float64'))}; %читаємо елемент (1,1) з атрибутами
m=fread(f1,[1,1],'float64');
Cellar(1,2)={fread(f1,[m,m],'float64')}; % читаємо елемент (1,2) з атрибутами
k=fread(f1,[1,2],'float64');
vd=fread(f1,[1,k(1)],'float64');
vs={char(fread(f1,[1,k(2)],'float64'))};
Cellar(2,1)={struct('field1',vd,'field2',vs)}; % читаємо елемент (2,1) з атрибутами
len2=fread(f1,[1,1],'float64');
Cellar(2,2)={fread(f1,[1,len2],'float64')}; %читаємо елемент (2,2) з атрибутами
fclose(f1); % закриваємо файл

```

Запам'ятовуємо цей текст з іменем *Rwmys.m* у поточній директорії.

Тепер за допомогою створеної функції **Wrmys** записуємо заповнений вище масив символічних рядків *Mucel* у бінарний файл *Mybinfile1.bin*:

```
>> WrMys(Mucel)
```

У поточній директорії повинен з'явитися файл *Mybinfile1.bin*. Прочитаємо вміст цього файла, очистивши попередньо робочий простір

**MatLab:**

```

>> clear
>> Cellar=Rwmys
Cellar =
    'Прошу пробачення' [3x3 double]
    [1x1 struct] [1x7 double]

```

Видно, що прочитана з файла змінна має правильну структуру. Далі перевіряємо значення окремих елементів цього масиву символічних рядків:

```

>> Cellar(1,1)
ans =
    'Прошу пробачення'
>> Cellar{1,2}

```

```

ans =
    1.3000    5.6000    7.0000
    7.9000    5.6000    9.4000
    5.0000    9.0000    8.0000
>> Cellar{2,1}
ans =
    field1: [1.2000 3.4000 5.4500]
    field2: 'Привіт'
>> Cellar{2,2}
ans =
     1     2     3     4     5     6     7

```

Із приводу функції **dlmread**, а заодно й функції **dlmwrite**, які записують і читають файли даних з роздільниками у форматі ASCII з текстового файла в матрицю, коротко роз'яснимо лише цей формат. Синтаксис функцій можна подивитися в Help.

Формат ASCII (American Standard Code for Information Interchange) – це спосіб кодування символів.

Символами можна оперувати, як числами й, навпаки, змінні можна інтерпретувати як символи (порівнювати із символами або друкувати як символи). У системі **MatLab** є дві функції для конвертації символів у коди й навпаки:

**S = char(X)** – функція конвертує (перетворює) масив *X*, що містить цілі позитивні числа як коди, у символи, букви й числа (використовуються перші 127 кодів ASCII).

**X=double('S')** – функція конвертує масив символів *S* у числові коди.

*Приклади:*

1. 96 кодів з 32 по 127 конвертуються в символи, цифри й букви латиниці.

```

>> ascii = char(reshape(32:127, 32, 3))
ascii =
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~□

```

Усі символи розташовуються в три рядки, код 32 відповідає пропуску.

2. Наступні коди конвертуються в прописні й малі літери кирилиці, що розташовані у два рядки.

```

>> asciik=char(reshape(1040:1103,32,2))

```

```
asciik=
АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ
абвгдежзийклмнопрстуфхцчшщъыьэюя
```

3. Конвертуємо масив, що містить букви кирилиці, символи коми й відсотків. Транспонуємо матрицю X для зручного вгляду на екрані.

```
>> X=double('б,Э,ю я %')
X =
1073 44 1069 44 1102 32 1103 32 37
```

4. Конвертуємо текстову змінну в коди й навпаки.

```
>> N=double('Початок екзаменаційної сесії 20 січня')
N =
1090 1082 1085 1085 1077 50 1095
1055 1086 1079 1072 1086 1089 48 1085
1086 1082 1072 1094 1111 1110 32 1103
1095 32 1084 1110 32 1111 1089
1072 1077 1077 1081 1089 32 1110
```

```
>> char(N)
ans =
Початок екзаменаційної сесії 20 січня
```

## FOPEN, FGETL

 Читання даних за рядками

Функціями **fopen**, **fget** можна прочитати дані файла за рядками.

*Синтаксис:*

**fopen('ім'я файла'), fgetl(цифра)**

*Опис:*

Функція **fopen('ім'я файла')** відкриває файл і привласнює йому ідентифікатор у вигляді цифри. Далі командою **fgetl(цифра)** зчитується перший рядок. Повторне уведення цієї ж команди дозволяє читати послідовно всі рядки. Процес закінчується виводом цифри **-1**. Це означає, що рядки у файлі закінчилися. Тут "цифра" – ідентифікатор файла даних.

*Приклад:*

```
>> fopen('mdt1.txt')
ans =
3
>> fgetl(3)
ans =
Mex1 Mex2 Mex3 Mex4
>> fgetl(3)
ans =
Узел1 3 5 7 1
>> fgetl(3)
ans =
Узел2 6 0 2 8
>> fgetl(3)
ans =
Узел3 4 9 3 4
>> fgetl(3)
ans =
-1
```

## EXEL LINK Зв'язок MATLAB з EXCEL

Інтегрування **MatLab** в **Excel** дозволяє користувачеві **Excel** звертатися до численних функцій **Matlab** обробки даних і навпаки. Надбудова **exellink.xla** реалізує дане розширення можливостей **Excel**, але її спочатку треба настроїти на спільну роботу з **Matlab**. У папці за адресою **MATLAB7.1\toolbox\exlink** (для **Matlab 7.1**) перебуває файл із надбудовою **exlink.xla**. Запустіть **Excel** і в меню *Сервис* виберіть пункт *Надстройки*. Відкриється діалогове вікно, (рис. 21.4) з інформацією про доступні на цей момент надбудови. Використовуючи кнопку **Обзор**, укажіть шлях до файла **exlink.xla**. У списку надбудов діалогового вікна з'явиться рядок **Excel Link 2.3 for use with MATLAB** із установленим прапорцем. Натисніть **OK**, і необхідна надбудова буде додана в **Excel** у вигляді панелі інструментів (рис. 21.5), що містить чотири кнопки: **startmatlab**, **putmatrix**, **getmatrix**, **evalstring**. Ці кнопки реалізують основні дії взаємозв'язку між **Excel** і **MatLab**.

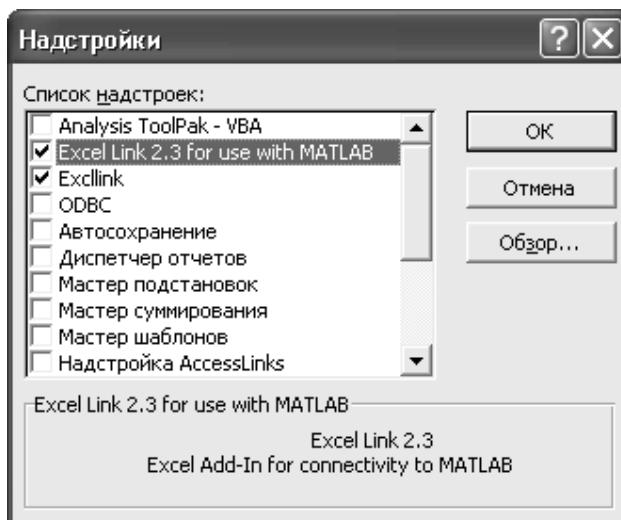


Рис. 21.4. Надбудови Excel

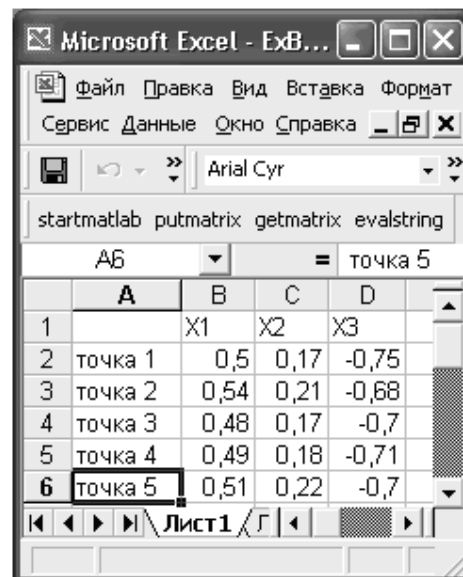


Рис. 21.5. Встановлена надбудова Excel Link

Слід перевірити ще дві установки в **Excel**. У меню *Сервис* у пункті *Параметры* на вкладці *Общие* треба зняти прапорець *Стиль ссылок*

*RICI*. На вкладці *Правка* повинен бути встановлений прапорець *Переход к другой ячейке после ввода*. Після цього можна здійснювати обмін даними між **MatLab** і **Excel**.

Призначення кнопок нової панелі інструментів:

**startmatlab** – ініціює вікно **MatLab** для сумісної роботи з **Excel**;

**putmatrix** – експорт даних виділеного діапазону клітинок **Excel** як матрицю в робочу область **MatLab**;

**evalstring** – обчислення в **MatLab** з робочого листа **Excel**;

**getmatrix** – імпорт в виділений діапазон клітинок **Excel** результатів розрахунків з **MatLab**.

Запустіть **Excel**. Виділіть на робочому листі діапазон клітинок з інформацією і натисніть кнопку **putmatrix**. Якщо попередньо не було ініційовано **MatLab**, з'явиться вікно з попередженням про те, що **MatLab** ще не відкрито. Тоді натисніть **OK** і дочекайтеся відкриття **MatLab**. У діалоговому вікні, що з'явиться з рядком уведення, привласніть ім'я змінної в робочому середовищі **MatLab**, у яку необхідно експортувати виділені дані. Перейдіть у командне вікно **MatLab** і переконаєтесь, що в робочому середовищі створилася змінна із привласненим іменем, що містить виділений масив клітинок **Excel**.

Щоб імпортувати дані з **MatLab** в **Excel**, треба у вікні **Excel** натиснути кнопку **getmatrix** і в діалоговому вікні, що з'явиться з рядком уведення, увести ім'я імпортованої змінної.

Поряд із числовими масивами, об'єктами, що підлягають обміну між **MatLab** і **Excel**, можуть бути текстові дані. У цьому випадку корисно з'ясувати тип змінної, експортованої в **MatLab**. Справа у тому, що текстова інформація із діапазону клітинок **Excel** записується в **MatLab** у масив символних рядків (cell array), а експорт тексту лише однієї клітинки приводить до символного масиву (char array).

Обмін даними між додатками може бути здійснений не лише за допомогою кнопок панелі інструментів Excel Link, але й з використанням функцій, що визначені у цій надбудові. Усього визначено одинадцять функцій, з них три основні функції дублюються кнопками панелі інструментів Excel Link:

**Mlputmatrix**("ім'я змінної"; діапазон клітинок) – поміщає дані із клітинок робочого листа **Excel** у масив робочого середовища **MatLab**;

**Mlgetmatrix**("ім'я змінної"; "діапазон клітинок") – поміщає дані з робочого середовища **MatLab** у діапазон листа **Excel**.

**Mlevalstring**("імена команди **Matlab**") – доступ із середовища **Excel** до виконуючих команд **MatLab**. Команди, що підлягають виконанню, задають в єдиному вхідному аргументі (допускається цілий рядок команд), але весь рядок (також як одна команда) береться в лапки. Команди **MatLab** у середовищі **Excel** набираються у будь-якій вільній клітинці, і мають починатися зі знака "=" .

*Приклад:*

Відкриваємо в середовищі **Excel** файл **Exbook2.xls** (або набираємо дані діапазону **A1:D6** самі – див. рис. 21.6). Зверніть увагу, що в русифікованій версії **Excel** роздільник десяткової частини числа – кома, а не крапка, як у **MatLab**.

Надбудова **Excel Link** встановлена, кнопка **startmatlab** натиснута.

Потрібно матрицю даних, що розташована в діапазоні **B2:D4**, конвертувати у середовище **MatLab** з ім'ям **M**, далі засобами **MatLab** знайти обернену матрицю  $IM = M^{-1}$  і транспонувати її.

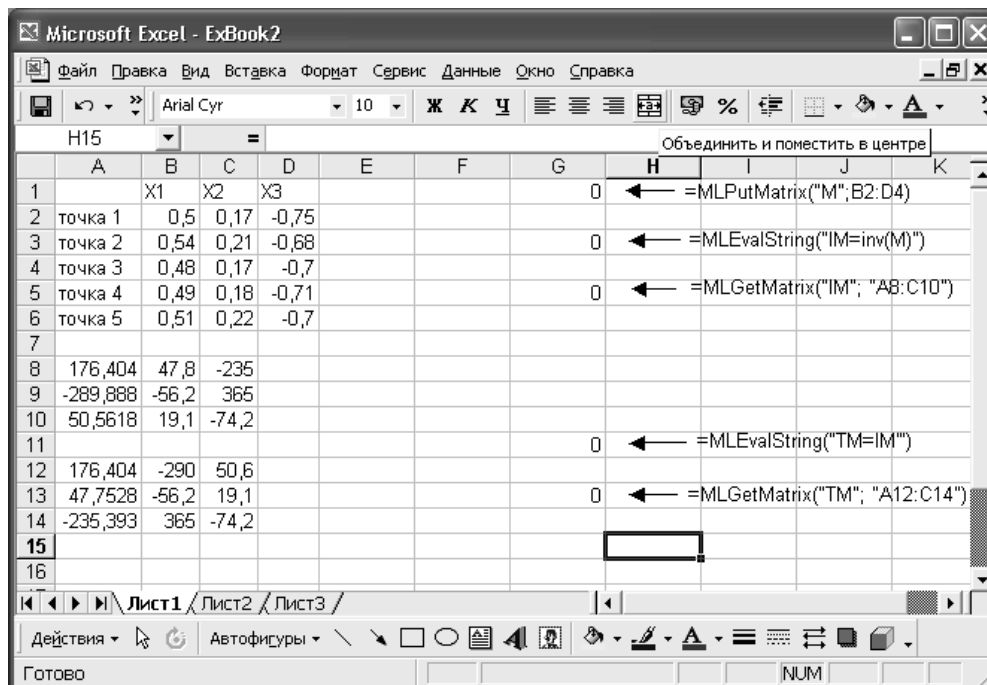


Рис.21.6. Лист **Excel** з розрахунками

Покажемо, як реалізувати поставлене завдання за допомогою функцій надбудови. Тоді усі команди **MatLab** (і результати їх роботи) будуть відбиватися на листі **Excel**, як це видно на рис. 21.6.

Отже, в клітинках G1, G3, G5 набрані команди обміну даних:

```
=MLPutMatrix("M";B2:D4) % Тут адреса діапазону набирається без лапок
=MLEvalString("IM=inv(M)") % Команда MatLab обернення матриці
=MLGetMatrix("IM";"A8:C10") % Тут адреса діапазону має бути в лапках
```

В діапазоні клітинок A8:C10 з'явилася обернена матриця:

$$\begin{pmatrix} 0,50 & 0,17 & -0,75 \\ 0,54 & 0,21 & -0,68 \\ 0,48 & 0,17 & -0,70 \end{pmatrix}^{-1} = \begin{pmatrix} 176,404 & 47,8 & -235 \\ -289,888 & -56,2 & 365 \\ 50,5618 & 19,1 & -74,2 \end{pmatrix}$$

Щоб транспонувати знайдену обернену матрицю і записати результат в діапазон A12:C14, в клітинках G11, G13 набрані команди

```
=MLEvalString("TM=IM'") % Команда MatLab транспонування матриці
=MLGetMatrix("TM"; "A12:C14") % Наявність пропусків не має значення
```

Всі ці операції можна виконати за допомогою командних кнопок панелі надбудови Excel Link. Так для транспонування матриці  $IM$  можна натиснути на кнопку **evalstring** і у вікні (рис. 21.7), що з'являється, набрати команду  $TM=IM'$ :

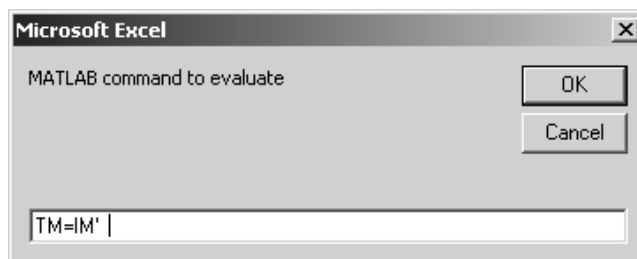


Рис. 21.7. Вікно набору команд Matlab

### **XLSREAD** Введення даних з xls-файла Excel

Якщо дані збережені у вигляді файла **MS Excel** з розширенням .xls, його можна прочитати за допомогою функції **xlsread**.

*Синтаксис:*

```
A=xlsread('ім'я xls-файла')
[A,text]=xlsread('ім'я xls-файла')
[A,text]=xlsread('ім'я xls-файла ','робочий лист','діапазон')
```

*Опис:*

Функція **A=xlsread('ім'я xls-файла')** повертає лише числові дані таблиці, що записана у файлі. Усі тексти (заголовки таблиці) ігноруються.

Функція **[A,text]=xlsread('ім'я xls-файла')** додаткове в параметрі *text* повертає тексти заголовків стовпців і рядків таблиці.

В функції **[A,text]=xlsread('ім'я xls-файла ','робочий лист','діапазон')** можна указати конкретний діапазон клітинок з даними; якщо в діапазоні є текстові заголовки, вони будуть переписані в змінну *text*.

*Приклади:*

Збережемо в директорії work **MatLab** файл Exbook1.xls з таблицею даних в діапазоні A1:D6. Прочитаємо цю таблицю в **MatLab**:

```
>> A=xlsread('Exbook1.xls')
```

```
A =  
 0.5000  0.1700 -0.7500  
 0.5400  0.2100 -0.6800  
 0.4800  0.1700 -0.7000  
 0.4900  0.1800 -0.7100  
 0.5100  0.2200 -0.6950
```

Відмітимо, що всі дробові числа були конвертовані у формат **MatLab** з роздільником-крапкою між цілою і дробовою частинами числа.

Щоб прочитати всі текстові заголовки, використовуємо команду:

```
>>[A,text]=xlsread('Exbook1.xls')
```

```
A =  
 0.5000  0.1700 -0.7500  
 0.5400  0.2100 -0.6800  
 0.4800  0.1700 -0.7000  
 0.4900  0.1800 -0.7100  
 0.5100  0.2200 -0.6950  
text =  
    " 'X1' 'X2' 'X3'  
'точка1' " " "  
'точка2' " " "  
'точка3' " " "  
'точка4' " " "  
'точка5' " " "
```

Перевіримо формат вихідних аргументів *A*, *text* .:

```
>> whos A
```

```
  Name      Size      Bytes   Class  
  A         5x3         120   double array
```

```
Grand total is 15 elements using 120 bytes
```

```
>> whos text
```

Name	Size	Bytes	Class
text	6x4	1522	cell array

Grand total is 65 elements using 1522 bytes

Змінна *text* виведена як масив символічних рядків (cell array).

Можна окремо вивести числові дані й назви рядків:

```
>> [A,points]=xlsread('Exbook1.xls','Лист1','A2:D6') % Діапазон без 1-го рядка Excel
A =
    0.5000    0.1700   -0.7500
    0.5400    0.2100   -0.6800
    0.4800    0.1700   -0.7000
    0.4900    0.1800   -0.7100
    0.5100    0.2200   -0.6950
points =
'точка 1'
' точка 2'
' точка 3'
' точка 4'
' точка 5'
```

Або вивести числові дані й назви стовпців:

```
>> [A,variables]=xlsread('Exbook1.xls','Лист1','B1:D6') % Діапазон без 1-го стовпця
A =
    0.5000    0.1700   -0.7500
    0.5400    0.2100   -0.6800
    0.4800    0.1700   -0.7000
    0.4900    0.1800   -0.7100
    0.5100    0.2200   -0.6950
variables =
'X1' 'X2' 'X3'.
```

### **File/Open** Відкрити файл засобами Windows

Це традиційний для **Windows** спосіб.

У вікні **MatLab** обираємо пункт меню: File / Open. З'являється вікно **Open**, де знаходимо ім'я потрібного файлу.

Залежно від того, яке розширення у файлі, розкриється те або інше вікно, де треба виконати відповідні дії. В результаті зміст файлу потрапляє на вкладку *Workspace*, після чого може бути скопійований в командне вікно (*Command Windows*) як змінна.

*Приклад:*

Файл `sof1.dat` знаходиться в папці `work`. Після його відкриття з'являється вікно `Import Wizard` (рис. 21.8), де міститься вся інформація про змінні, що записані у файлі.

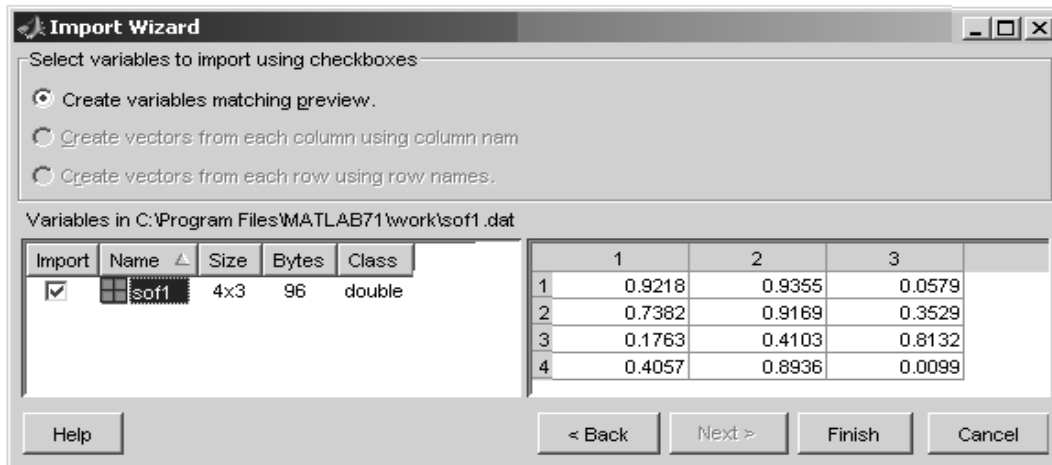


Рис. 21.8. Вікно `Import Wizard`

Виділивши ім'я потрібної змінної, на панелі праворуч можна переглянути її дані. Далі натискаємо кнопку `Finish`, і обрана змінна переноситься в робочу область `MatLab`, що можна побачити на вкладці `Workspace` (рис. 21.9):

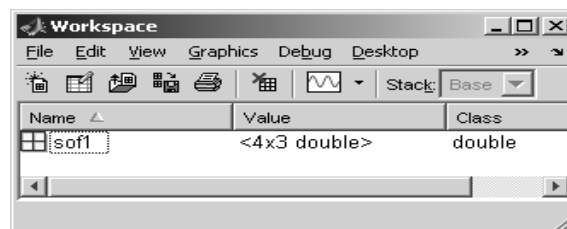


Рис. 21.9. Вкладка `Workspace`

Перевіряємо правильність імпорту:

```
>> sof1
sof1 =
    0.9218    0.9355    0.0579
    0.7382    0.9169    0.3529
    0.1763    0.4103    0.8132
    0.4057    0.8936    0.0099
```

## **File / Import Data** Імпорт даних у робочу область **Workspace**

Найпростіший шлях імпорту даних у середовище **MatLab** – це використовувати Майстер імпорту. Вам не потрібно знати формат даних. Ви просто обираєте файл даних, який перебуває в будь-якій директорії **MatLab**, і Майстер імпорту обробляє зміст файла автоматично.

*Приклади:*

1. Імпорт файлу `mdt1.txt`.

В меню **File** обираємо пункт **Import Data**, після чого відкривається перше вікно Майстра імпорту, зображене на рис. 21.10.

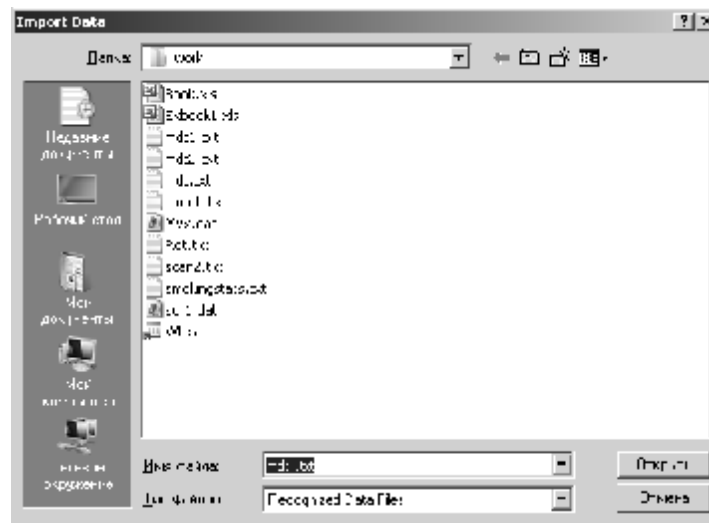


Рис. 21.10. Вікно **Import Data** з виділенням файлом `mdt1.txt`

Знаходимо на диску потрібний файл і відкриваємо його кнопкою **Открыть**. З'являється друге вікно Майстра імпорту – вікно **Import Wizard** (рис. 21.11), у якому дані файла показані у вигляді форматованої таблиці (ліворуч) і окремо числові і текстові змінні (праворуч). У верхній частині панелі є перемикачі, що дозволяють вибрати тип роздільника даних в імпортованому файлі. Допускається імпорт файлів, дані в яких розділені комою (**Comma**), пропусками (**Space**), крапкою з комою (**Semicolon**), символами табуляції (**Tab**) і деякими іншими (**Other**). Даний файл читається лише за вибору роздільника **Tab**.

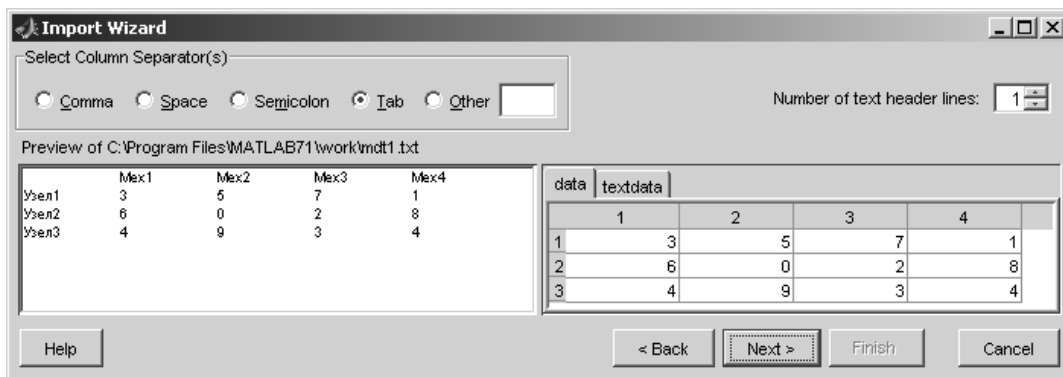


Рис. 21.11. Вікно Import Wizard

Натискаємо кнопку **Next**. З'являється наступне вікно (рис. 21.12) з переглядом змінних, які будуть створені в **MatLab**. Натискаємо кнопку **Finish** і відмічені прапорцями  змінні будуть імпортовані в робочу область **MatLab**.

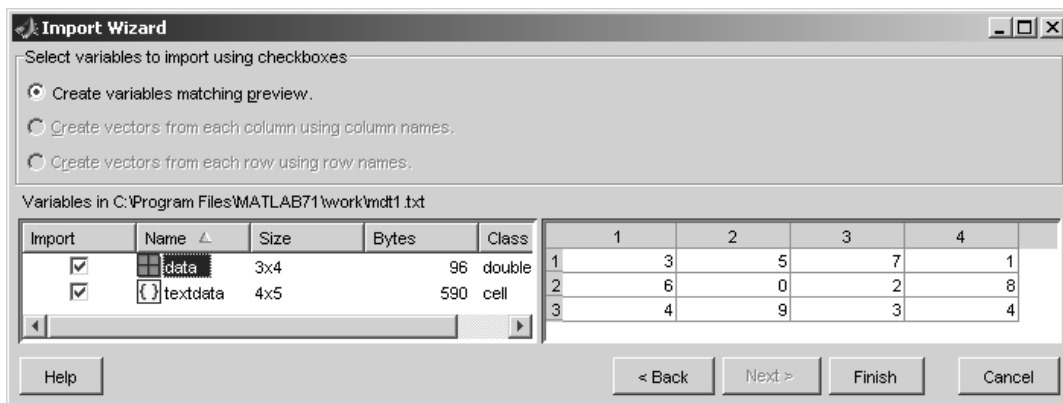


Рис. 21.12. Третє вікно Import Wizard

Перевіряємо:

```
>> data
data =

    3     5     7     1
    6     0     2     8
    4     9     3     4

>> textdata
textdata =

    'Mex1' 'Mex2' 'Mex3' 'Mex4'
'Узел1'   []      []      []      []
'Узел2'   []      []      []      []
'Узел3'   []      []      []      []
```

2. Імпорт файла sat2.dat.

Імпорт dat-файлів нічим не відрізняється від імпорту txt-файлів. У директорії C:\Program Files\MATLAB71\toolbox\stats знаходиться файл sat2.dat. Дані цього файла розділені комою, і Майстер імпорту визначив це (рис. 21.13). Для імпорту в середовище **MatLab** Майстер імпорту представив дані у вигляді двох змінних: числову з іменем *data* і текстову з іменем *texdata*. У вікні Import Wizard (праворуч) можна переглянути обидві змінні (рис. 21.13 і 21.14).

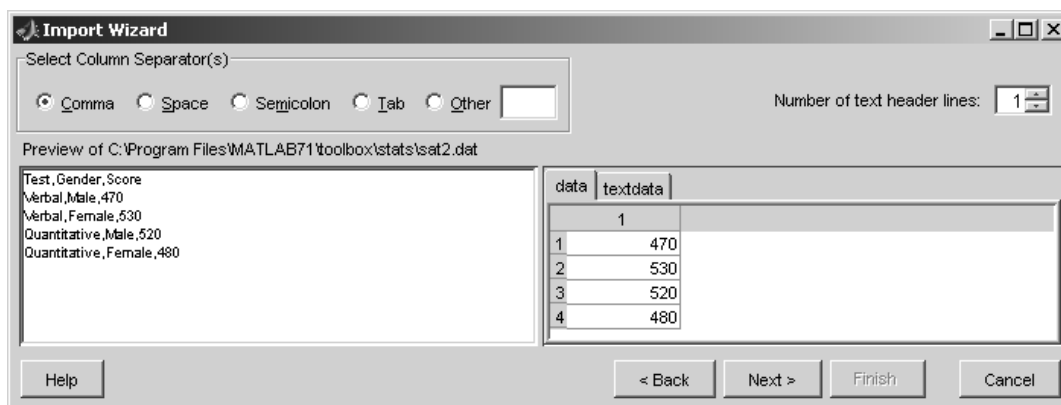


Рис. 21.13. Числова частина файла sat2.dat

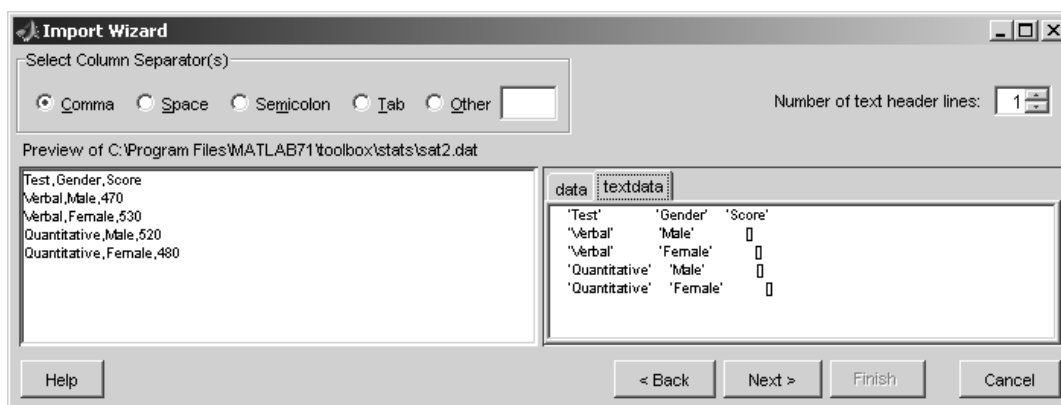


Рис. 21.14. Текстова частина файла sat2.dat

Натискаємо кнопки **Next** і **Finish** (у наступному вікні), після чого змінні будуть імпортовані в робочу область **MatLab**.

### 3. Імпорт файла Exbook1.xls.

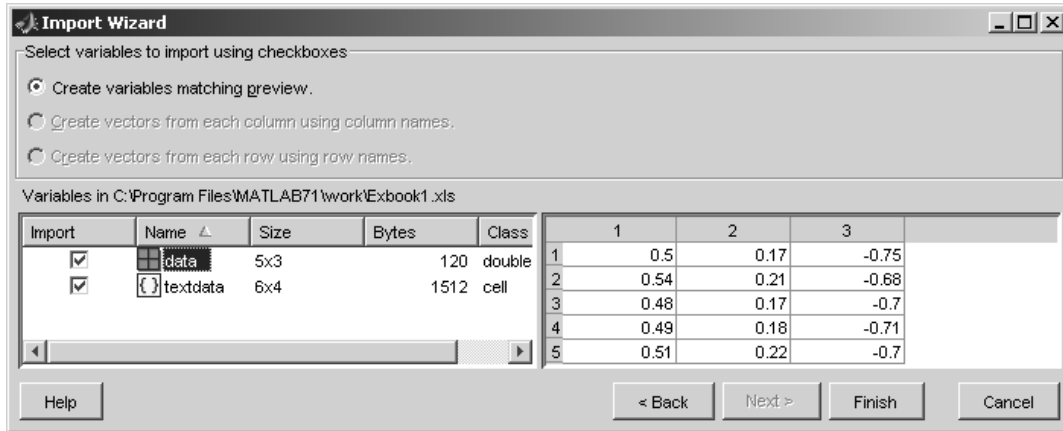


Рис. 21.15. Імпорт xls-файла

Імпорт xls-файлів майже нічим не відрізняється від імпорту txt і dat-файлів. Досить знайти потрібний xls-файл, як відразу відкривається фінальне вікно Майстра імпорту (рис. 21.15).

### 4. Імпорт файла discrim.mat.

Файли з такими розширеннями є бінарними. Майстер відразу пропонує розбити файл на 6 змінних: 3 числових типу double і три символні типу char, це як показано на рис 21.16.

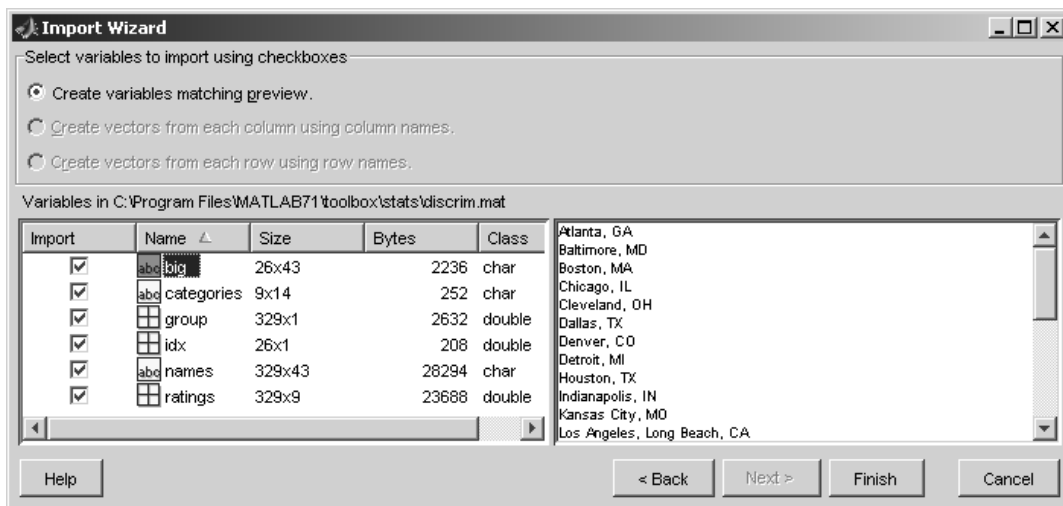


Рис. 21.16. Імпорт файла discrim.mat

Якщо ми погоджуємося, то всі вони (після натискання кнопки **Finish**) будуть імпортовані в робочу область **MatLab**. Для перевірки переглянемо змінну *ratings* (нижче наведені перші 5 рядків з 329):

```
>> ratings
```

ratings =

521	6200	237	923	4031	2757	996	1405	7633
575	8138	1656	886	4883	2438	5564	2632	4350
468	7339	618	970	2531	2560	237	859	5250
476	7908	1431	610	6883	3399	4655	1617	5864
659	8393	1853	1483	6558	3026	4496	2612	5727

## **UIIMPORT** Відкриває Майстер імпорту

Використовуючи цю функцію можна відкрити Майстер імпорту з командного вікна.

*Синтаксис:*

```
uiimport  
uiimport('filename')  
uiimport('-file')  
uiimport('-pastespecial')  
S = uiimport(...)
```

*Опис:*

Функція `uiimport` відкриває Майстер імпорту (рис. 21.17) в трохи іншому вигляді: з кнопкою **Browse** (Огляд) і перемикачем на Clipboard (Буфер обміну), що дає можливість завантаження даних як з файла, так і з буфера обміну.



Рис. 21.17. Вікно Майстра імпорту

Функція `uiimport('filename')` виводить вікно Майстра імпорту, відкриваючи файл, зазначений в аргументі. Ім'я файла треба вводити як аргумент повністю.

Функція `uiimport('-pastespecial')` відкриває вікно Майстра імпорту, читаючи при цьому файл даних, що втримується в буфері обміну

## 21.2. Форматування даних

Після введення даних іноді потрібне їх форматування, наприклад, виділення окремих рядків чи стовпців матриці (або багатовимірного масиву), підсумовування, визначення найбільших і найменших значень, видалення частини множини тощо. Це можна зробити шляхом програмування відповідних циклів, але **MatLab** дозволяє майже завжди обійтися без циклів. Наприклад, якщо потрібно скопіювати матрицю  $A$  в матрицю  $B$ , то замість подвійного циклу досить просто ввести команду `>> B=A`. Ці ж принципи застосовуються й за інших маніпуляцій з даними, наприклад, з вибірками. Нехай  $x$  – матриця чисел або символів розміром  $5 \times 8$ . Тоді: `>> y = x(1,:)` – виділяє перший рядок (результат  $y$  – вектор-рядок  $1 \times 8$ ); `>> y = x(:,3)` – виділяє третій стовпець (результат – вектор-стовпець  $5 \times 1$ ); `>> y = x(1:3,:)` – виділяє рядки з першого по третій ( $y$  – матриця  $3 \times 8$ ); `>> y = x(1:2,[1 3 6])` – виділяє 1-й і 2-й рядки й 1-й, 3-й і 6-й стовпці (результат  $y$  – матриця  $2 \times 3$ ); можна поміняти порядок рядків чи стовпців: `>> y = x(1:2,[3 1 6])` – виділяє 1-й і 2-й рядки й 3-й, 1-й і 6-й стовпці, тобто при цьому міняє місцями 1-й і 3-й стовпці (результат  $y$  – матриця  $2 \times 3$ ).

Аналогічно вирішується задача для багатовимірних масивів.

Приведемо інші корисні функції для маніпуляцій з матрицями.

`cat(dim,A,B)` – зчеплення масивів  $A$ ,  $B$  уздовж розмірності  $dim$ . Якщо  $dim = 1$ , масиви  $A$ ,  $B$  розташовуються у стовпчик; якщо  $dim = 2$ , масиви  $A$ ,  $B$  розташовуються поряд (у рядок); якщо  $dim = 3$ , створюється тривимірний масив. Відповідні розміри  $A$ ,  $B$  повинні співпадати.

`fliplr(M)` – дзеркальне відбиття матриці  $M$  навколо вертикальної осі (елементи кожного рядка записуються у зворотному порядку);

`flipud(M)` – дзеркальне відбиття матриці  $M$  навколо горизонтальної осі (елементи кожного стовпця записуються у зворотному порядку);

`flipdim` – дзеркальне відбиття матриці або багатовимірного масиву навколо заданої розмірності: `flipdim(M,1) ~ fliplr(M)`; `flipdim(M,2) ~ flipud(M)`;

`rot90(M)` – обертання матриці  $M$  на кут  $90^\circ$ ;

**x(:)** – запис елементів масиву будь-якої розмірності у вигляді вектора – стовпця. При цьому швидше всього міняється індекс для першої розмірності, потім для другої і так далі;

**reshape(M,p,q)** – переформатування масиву  $M$  в масив іншої розмірності  $p \times q$ . Елементи в новий масив заносяться в порядку  $x(:)$

**repmat(M,k,dim)** – копіювання масиву  $M$   $k$  раз уздовж розмірності  $dim$ ;

**A=diag(a)** – створення діагональної матриці з елементами вектора  $a$  на діагоналі; **a=diag(A)** – виділення діагоналі матриці  $A$  у вектор  $a$ ;

**eye(n)** – створення квадратної одиничної матриці; **eye(n,m)** – створення прямокутної матриці з одиницями на діагоналі;

**ones(n)** – створення квадратної матриці з одиниць;

**ones(n,m)** – створення прямокутної матриці з одиниць;

**zeros(n)** – створення квадратної матриці з нулів;

**zeros(n,m)** – створення прямокутної матриці з нулів;

**min** – знаходить мінімальний елемент вектора. Може застосовуватися до матриць і багатовимірних масивів, де знаходить мінімальний елемент за першою розмірністю; **[Amin,I]=min(A)** – повертає також номер  $I$  першого мінімального елемента (для кожного стовпця матриці  $A$ );

**max** – те ж саме для максимального значення;

**sort** – сортує елементи вектора в порядку зростання або убутання. Може застосовуватися до матриць, де провадить сортування кожного стовпця незалежно від інших;

**sortrows(A)** – сортує рядки матриці; **sortrows(A,col)** – сортує рядки матриці  $A$  за елементами стовпця  $col$  (за замовченням  $col = 1$ ); якщо  $col$  – від'ємне, сортування провадиться за убутанням елементів  $col$ ;

**union** – поєднує елементи, відбирає неповторювані й сортує їх. Повертає також номери відібраних елементів;

**intersect** – знаходить загальні елементи, вибирає неповторювані й сортує їх. Повертає також номери відібраних елементів;

**setdiff** – видаляє з однієї множини елементи іншої; з того, що залишилися, вибирає неповторювані елементи і сортує їх. Повертає також номери відібраних елементів.

**setxor** – вибирає з кожної із двох множин елементи, які не є для них спільними. З отриманої множини вибирає неповторювані елементи і сортує їх. Повертає також номери відібраних елементів;

**ismember** – повертає 1 (*true*) або 0 (*false*) залежно від того, чи є елементи однієї множини елементами іншої. Повертає також номери вхідних елементів;

**all** – повертає 1, якщо всі елементи вектора дійсні і ненульові;

**any** – повертає 1, якщо хоча б один елемент дійсний ненульовий.

### 21.3. Збереження інформації

Заключним етапом обробки даних є звичайне збереження текстової, бінарної й графічної інформації для її подальшого використання. Для цього маємо наступні функції:

<b>SAVE</b>	Збереження файла .....	492
<b>TBLWRITE</b>	Запис табульованих даних .....	494
<b>CASEWRITE</b>	Запис у текстовий файл за рядками .....	495
<b>XLSWRITE</b>	Запис даних в xls-файли .....	497
<b>FWRITE</b>	Запис у бінарні файли .....	498
<b>FPRINTF</b>	Збереження текстової інформації .....	501

#### **SAVE** Збереження файла

*Синтаксис:*

**save ім'я файла [змінна] [змінна] ...**

**save ім'я файла [змінна] [змінна] ... -ascii**

*Опис:*

Командою **save x** створюється файл із розширенням `mat`. Змінні в таких файлах зберігаються у бінарному кодуванні. Спроба перегляду цих файлів у будь-якому текстовому редакторі не дає ніякої зрозумілої інформації про змінні і їх значення. Змінні треба завантажувати в `Workspace`.

Команда **save** із ключем `-ascii` використовується для запису різно-рідної інформації в один текстовий `txt`-файл. Файл записується в поточну папку. Для виводу більшої кількості знаків можна вказати ще один ключ – **double**.

*Приклад:*

Створимо в командному вікні дві матриці різного розміру й два текстові рядки на різних мовах. Запишемо все це в текстовий файл.

```
>> A=hadamard(4);
>> B=eye(3);
>> s='save file'; % Це текст англ. мовою
>> k='Харківський національний університет';
>> save x.txt A B s k -ascii % У файлі x.txt чотири змінні: A, B, s, k
```

Спробуємо вивести файл `x.txt` в командне вікно командою **type x.txt**:

```
>> type x.txt
1.0000000e+000 1.0000000e+000 1.0000000e+000 1.0000000e+000
1.0000000e+000 -1.0000000e+000 1.0000000e+000 -1.0000000e+000
1.0000000e+000 1.0000000e+000 -1.0000000e+000 -1.0000000e+000
1.0000000e+000 -1.0000000e+000 -1.0000000e+000 1.0000000e+000
1.0000000e+000 0.0000000e+000 0.0000000e+000
0.0000000e+000 1.0000000e+000 0.0000000e+000
0.0000000e+000 0.0000000e+000 1.0000000e+000
1.1500000e+002 9.7000000e+001 1.1800000e+002 1.0100000e+002
3.2000000e+001 1.0200000e+002 1.0500000e+002 1.0800000e+002 1.0100000e+002
1.0610000e+003 1.0720000e+003 1.0880000e+003 1.1000000e+003
1.0820000e+003 1.0860000e+003 1.0740000e+003 1.0890000e+003 1.0820000e+003
1.0800000e+003 1.0810000e+003 3.2000000e+001 1.0850000e+003 1.0720000e+003
1.0940000e+003 1.0800000e+003 1.0860000e+003 1.0850000e+003 1.0720000e+003
1.0830000e+003 1.1000000e+003 1.0850000e+003 1.0990000e+003 1.0810000e+003
3.2000000e+001 1.0910000e+003 1.0850000e+003 1.0800000e+003 1.0740000e+003
1.0770000e+003 1.0880000e+003 1.0890000e+003 1.0800000e+003 1.0900000e+003
1.0770000e+003 1.0900000e+003
```

Лише перші сім рядків введеної інформації зрозумілі – це матриця Адамара  $A$  ( $4 \times 4$ ) і одинична матриця  $B$  ( $3 \times 3$ ). За умови використання

цієї команди всі символічні змінні були перетворені в масив дійсних чисел, причому для символів кирилиці використано кодування Unicode.

Застосуємо іншу функцію введення даних з текстового файла:

```
>> x=textread('x.txt'); % читаємо весь файл
```

Далі читаємо кожну змінну:

```
>> A=x(1:4,1:4) % виводимо змінну A – матрицю Адамара
```

```
A =  
 1  1  1  1  
 1 -1  1 -1  
 1  1 -1 -1  
 1 -1 -1  1
```

```
>> B=x(5:7,1:3) % виводимо одиничну матрицю B
```

```
B =  
 1  0  0  
 0  1  0  
 0  0  1
```

```
>> s=char(x(8,:)) % виводимо символічну змінну s
```

```
s =  
save file
```

```
>> k=char(x(9,:)) % виводимо символічну змінну k
```

```
k =  
Харківський національний університет
```

## **TBLWRITE** Запис табульованих даних

За допомогою цієї функції в текстовий файл записується таблиця із заголовками рядків і стовпців

*Синтаксис:*

```
tblwrite(data,varnames,casenames)  
tblwrite(data,varnames,casenames,filename)  
tblwrite(data,varnames,casenames,filename,delimiter)
```

*Опис:*

Функція **tblwrite(data,varnames,casenames)** відкриває стандартне діалогове вікно запису у файл, у якому можна вибрати ім'я файла для запису табульованих даних. Матриця числових даних повинна бути в аргументі *data*, заголовки полів – у матриці символів *varnames* (за рядками, більш короткі рядки доповнюються пропусками), а імена записів – у матриці символів *casenames* (також за рядками, більш короткі рядки доповнюються пропусками).

Функція **tblwrite(data,varnames,casenames,filename)** в аргументі *filename* визначається ім'я файла (текстовий рядок). Файл записується в поточну папку. При необхідності можна вказати повний шлях.

Функція **tblwrite(data,varnames,casenames,filename,delimiter)** в аргументі *delimiter* дозволяє задати символ-роздільник полів у кожному запису. Цей аргумент повинен бути текстовим рядком. Його можливі значення перелічені в процесі розгляду функції **tblread**.

*Приклад:*

```
>> data=[30 70; 50 50]; % числові дані
>> vnames=char('Курці','Некурці'); % назви стовпців
>> cnames=char('Юнаки','Дівчата'); % назви рядків
>> tblwrite(data,vnames,cnames,'smokingstats.txt') % запис у txt-файл
```

Прочитаємо запис:

```
>> type smokingstats.txt % друкуємо зміст файла
      Курці   Некурці
Юнаки   30     70
Дівчата 50     50
```

Таку таблицю зручно читати у будь-якому текстовому редакторі й можна ввести в **MatLab** функцією **tblread**.

### **CASEWRITE** Запис у текстовий файл за рядками

Цією функцією можна записати в текстовий файл послідовність текстових рядків однакової довжини

*Синтаксис:*

```
casewrite(strmat)
casewrite(strmat,filename)
```

*Опис:*

Функція **casewrite(strmat)** відкриває стандартне діалогове вікно запису у файл, у якому можна вибрати ім'я файла для запису в нього матриці символів *strmat*.

Функція **casewrite(strmat,filename)** в аргументі *filename* визначає ім'я файла (текстовий рядок). Файл записується в поточну папку. Якщо є потреба, можна вказати повний шлях до файла.

Приклад:

```
>> strmat=char('Січень','Лютий','Березень','Квітень','Травень','Червень', ...  
'Липень','Август','Вересень','Жовтень','Листопад','Грудень'); % набираємо  
>> casewrite(strmat,'month.txt'); % записуємо у файл month.txt.
```

Отриманий файл можна прочитати функцією **casewrite**. Якщо застосувати її прямо, то одержимо недекодований результат.

```
>> dow=caseread('month.txt') % читаємо файл  
dow =  
□□□□□□  
□□□□□□□  
□□□□  
□□□□□□  
□□□  
□□□□  
□□□□  
□□□□□□  
□□□□□□□□  
□□□□□□□  
□□□□□□  
□□□□□□□
```

Змінну **dow** треба ще декодувати з використанням декодера **native2unicode**:

```
>> [m,n]=size(dow) % число рядків і символів у рядку  
m =  
    12  
n =  
     8  
>> for k=1:m, % друкуємо декодовані рядки  
    fprintf('%s\n',native2unicode(dow(k,:)));  
end  
Січень  
Лютий  
Березень  
Квітень  
Травень  
Червень  
Липень  
Серпень  
Вересень  
Жовтень  
Листопад  
Грудень
```

Розмір  $n = 8$  визначається числом символів найбільш довгого за кількістю букв місяця.

## **XLSWRITE** Запис даних в xls-файли

За допомогою цієї команди можна записати дані у форматі **MS Excel**.

*Синтаксис:*

```
xlswrite('filename', M)  
xlswrite('filename', M, sheet)  
xlswrite('filename', M, sheet, 'range')
```

*Опис:*

Функція **xlswrite('filename', M)** записує матрицю  $M$  у робочу книгу **Excel** із ім'ям filename.xls. Вхідна матриця  $M$  розміром  $m \times n$  може бути цифровою, або символною, або масивом символних рядків, де  $m < 65535$  і  $n < 256$ . Матриця даних записується на перший робочий лист, починаючи із клітинки A1.

Функція **xlswrite('filename', M, sheet)** записує матрицю  $M$  на робочий лист, зазначений в аргументі *sheet*. Він повинен бути або позитивним двозначним числом (наприклад, 03 або 14), що вказує на номер робочого листа, або ім'ям листа, узятого в лапки.

Функція **xlswrite('filename', M, sheet, 'range')** записує змінну  $M$  як масив даних з ім'ям filename.xls в робочий лист *sheet* **Excel**, в діапазон клітинок *'range'*. Діапазон *'range'* можна задавати повністю як адреси двох клітинок через двокрапку (що указують на два протилежні кути прямокутної області), а можна вказувати адресу лише лівого верхнього кута області даних.

*Приклад:*

```
>> x=[0.5 0.17 -0.75; 0.54 0.21 -0.68; 0.48 0.17 -0.7;...  
-0.49 0.18 0.71; -0.52 0.33 -0.694]; % матриця чисел розміром 5x3  
>> variables={'Перша','Друга','Третя'}; % заголовки стовпців  
>> points=num2str([1;2;3;4;5]); % заголовки рядків  
>> xlswrite('Book.xls',x,'b2:d6') % записуємо числовий масив в діапазон b2:d6  
>> xlswrite('Book.xls',variables,'b1:d1') % записуємо заголовки стовпців в b1:d1  
>> xlswrite('Book.xls',points,'a2:a6') % записуємо заголовки рядків в a2:a6
```

У робочій директорії з'являється файл Book.xls. Відкриємо його для перевірки:

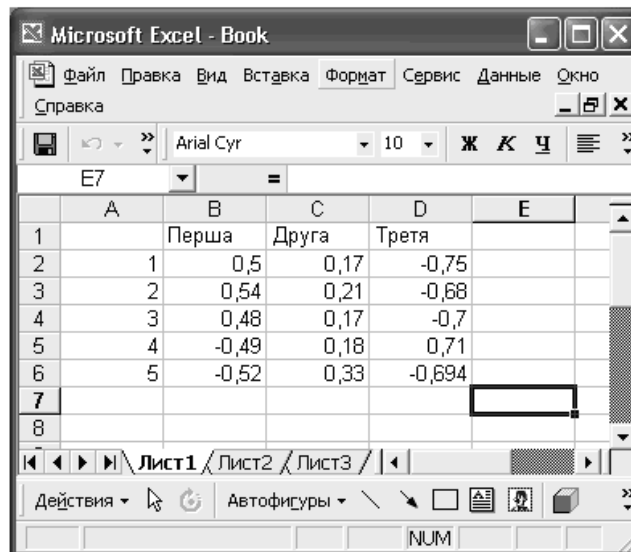


Рис.21.18. Відкриття файла Book.xls

### **FWRITE** Запис у бінарні файли

Дані можна записувати не лише в текстові, але й у бінарні файли. Такі файли більш ошадливі, але їх не можна прочитати звичайними текстовими редакторами. Для запису бінарних даних використовуються функції:

- fwrite** – записати у файл;
- fseek** – установити позицію;
- frewind** – установити початкову позицію;
- ftell** – повернути поточну позицію;
- fopen** – відкрити файл;
- fclose** – закрити файл.

*Синтаксис:*

**count = fwrite(fid,A,precision)**  
**count= fwrite(fid,A,precision,skip)**

*Опис:*

Функція **count = fwrite(fid,A,precision)** записує елементи матриці *A* в зазначений (*fid*) файл. Попередньо файл має бути відкритим функцією **fopen**, яка повертає *fid* – числовий ідентифікатор відкритого файла. Функція **fwrite** записує дані у файл за стовпцями і повертає *count* – кількість записаних елементів. Аргумент *precision* задає тип даних, що записуються, для виділення для них пам'яті. Можливі значення *precision*

наведено в табл. 21.2 при опису функції **fread**. Якщо цей аргумент не зазначений, то за замовчуванням використовується *'uchar'* – символ без знака, 8 біт.

Функція **count = fwrite(fid,A,precision,skip)** включає ще необов'язковий аргумент *skip*, який указує на число байтів, що пропускається перед тим, як зробити запис. За наявності цього аргументу функція пропускає фіксовану кількість байтів перед першим записом, далі знову пропускає ту ж кількість байтів перед черговим записом, й так доти, поки не буде записана вся матриця *A*.

#### Приклади:

1. Запишемо в бінарний файл вектор-рядок і матрицю.

```
>> a=[1;2;3]; B=[4 5 6; 7 8 9]; %вектор і матриця
>> fid=fopen('Wtest.mat','wb') % відкриємо бінарний файл для запису
fid = % ідентифікатор файла
     3
>> fwrite(fid,a,'float64'); % записуємо вектор a
>> fwrite(fid,B,'float64'); % записуємо матрицю B
>> fclose(fid) % закриваємо файл
ans =
     0
```

У поточному каталозі з'явився файл *Wtest.mat* розміром 72 байта, оскільки ми записали 9 дійсних чисел з подвійною точністю й кожному з них виділяється по 8 байтів ( $9 \times 8 = 72$ ). Якщо спробувати прочитати цей файл у текстовому редакторі, наприклад, *AkelPad*, то одержимо "абракадабру", показану на рис. 21.19.



Рис. 21.19. Відкриття бінарного файла в текстовому редакторі

Це типове зображення, коли бінарні файли проглядаються редакторами, орієнтованими на текстові файли. Збереження інформації в бінарних файлах носить досить потайливий характер, самі файли за обсягом мінімальні. Проте навіть у текстовому редакторі, рахуючи від

початку рядка до її кінця, можна визначити, що всього у файл Wtest.mat записано 72 байта інформації (включаючи пропуски).

Система **MatLab** дозволяє прочитати такий файл за допомогою функції **fread**, яка докладно була описана вище. Для прикладу приведемо результати застосування.

```
>> fid=fopen('Wtest.mat','rb');
>> [a,count]=fread(fid,[1,3],'float64')
% читаємо вектор a
a =
    1    2    3
count =
     3
% вектор містить три цифри
```

```
>> [a,count]=fread(fid,[2,3],'float64')
% читаємо матрицю B
a =
     4     5     6
     7     8     9
count =
     6
% матриця містить 6 цифр
```

## 2. Перепишемо файл **MS Excel** у бінарний файл Myx.dat:

```
>> x=xlsread('Book.xls') % Переписали дані xls-файла в змінну x
x =
    1.0000    0.5000    0.1700   -0.7500
    2.0000    0.5400    0.2100   -0.6800
    3.0000    0.4800    0.1700   -0.7000
    4.0000   -0.4900    0.1800    0.7100
    5.0000   -0.5200    0.3300   -0.6940
```

```
>> fid=fopen('Myx.dat','w'); % Відкрили бінарний файл для запису (з ключем 'w')
>> c=fwrite(fid,x,'double'); % Запис у бінарний файл
>> fclose(fid); % Закрили файл
>> fprintf('Кількість записаних чисел=%d\n',c) % Перевірка
Кількість записаних чисел=20
```

Записано 20 чисел – елементів матриці 5×4 в бінарний файл.

Організувати його перегляд можна в такий спосіб:

```
>> fid=fopen('Myx.dat'); % відкрили файл для читання (без ключів)
>> y=fread(fid,[5,4],'double') % читаємо матрицю даних розміром 5×4
>> fclose(fid); % закрили файл
y =
    1.0000    0.5000    0.1700   -0.7500
    2.0000    0.5400    0.2100   -0.6800
    3.0000    0.4800    0.1700   -0.7000
    4.0000   -0.4900    0.1800    0.7100
    5.0000   -0.5200    0.3300   -0.6940
```

Переглянемо тип змінної y:

```
>> whos y
Name      Size      Bytes   Class
y         5x4       160     double array
Grand total is 20 elements using 160 bytes
```

Записано 20 елементів матриці в бінарний файл об'ємом в 160 байт по 8 байт на елемент.

## **FPRINTF** Збереження текстової інформації

Будь-яку текстову інформацію можна записати у файл за допомогою цієї функції.

*Синтаксис:*

```
fprintf(fid,'текст')  
count = fprintf(fid, format, A,...)
```

*Опис:*

Функція **fprintf(fid,'текст')** записує (додає) рядки в текстовий файл із ідентифікатором *fid*, що повертається функцією **fopen** за умови відкриття файла. Наступне застосування функції записує новий текст у ті самі рядки, а не в нові.

У функції **count = fprintf(fid,format,A,...)** аргумент *format* містить специфікатори формату й керуючі символи, які визначають вид запису значень змінних зі списку, заданого третім аргументом. У списку може бути одна або кілька змінних, у тому числі й масивів, наприклад, матриць. Матриці, за замовчуванням, записуються в один загальний стовпець один за одним стовпців матриці.

Специфікатори формату й керуючі символи у файл не записуються. Вони керують процесом запису в файл значень змінних, зазначених у списку аргументів функції один за одним, і задають вирівнювання, кількість значущих розрядів, ширину поля, і інші аспекти формату, наприклад, містять знаки зміни регістру.

Керуючі символи (насправді це набір із двох символів, першим з яких є зворотний слеш \) форматують текст, наприклад, здійснюють табуляцію, перехід на новий рядок або сторінку.

Керуючі символи і їх опис представлено в таблиці 21.4.

## Керуючі символи

Символ	Опис	Символ	Опис
\b	Пропуск	\t	Горизонтальна табуляція
\f	Формат чисел	\\	Запис слеша
\n	Перехід на новий рядок	% %	Запис знака відсотка
\r	Повернення каретки	\"	Запис двох лапок

Наприклад, керуючий символ \n означає перехід на новий рядок.

Тепер розглянемо специфікатори формату. Вони починаються зі знака % і містять обов'язкові і необов'язкові елементи, як це показано на рис. 21.20. Їх можливі значення наведено в табл. 21.5.

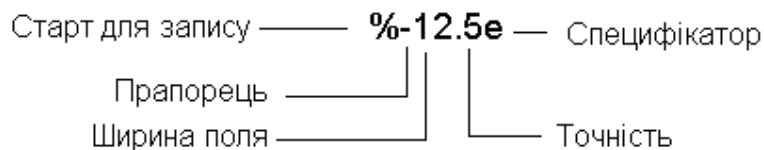


Рис. 21.20. Порядок уведення специфікаторів

Між символом % і буквою можуть бути додаткові числові параметри й поділяюча їх крапка, що контролюють ширину поля й точність запису чисел. Наприклад, запис %6.2f означає, що дійсне число записується набором символів у шести позиціях, з яких два приділяється для запису дробової частини числа.

## Специфікатори формату

Специфікатор	Опис	Специфікатор	Опис
% c	Один символ	% G	Більш компактний запис, ніж %E і без 0 цілих (.2817)
% d або %i	Десяткова система зі знаком	% o	Восьмерична система без знака
% e	Показова форма з використанням нижнього регістру як в 3.1415e+00)	% s	Рядок символів
% E	Показова форма з використанням верхнього регістру як в 3.1415E+00)	% u	Десяткова система без знака
% f	Дійсне число як текстовий рядок	% x	Шістнадцятерична система з використанням нижнього регістру (a-f)

<b>% g</b>	Більш компактний запис, ніж %e й без 0 цілих (.2817)	<b>% X</b>	Шістнадцятерична система з використанням верхнього регістру (A-F)
------------	--	------------	---

До необов'язкових символів відносяться прапорці, що керують вирівнюванням тексту (табл. 21.6).

Таблиця 21.6

### Прапорці

Знак	Опис	Приклад
Мінус (-)	Вирівнювання по лівому краю	% - 5.2d
Плюс(+)	Завжди друкує символ знака (+ або -)	% + 5.2d
Знак пропуску ( )	Уводить пропуск перед величиною	% 5.2d

*Приклад.*

1. Не записуючи у файл, виведемо на екран текстову інформацію.

```
>> fprintf('Ми вчимося у Харківському національному економічному
університеті.\n')
```

Ми вчимося у Харківському національному економічному університеті.

2. Виведемо на екран текстову й цифрову інформацію.

```
>> fprintf('Одиничне коло має довжину %e радіан.\n',2*pi)
```

Одиничне коло має довжину 6.283185e+000 радіан.

3. Поміняємо специфікатор з %e на %f

```
>> fprintf('Одиничне коло має довжину %f радіан.\n',2*pi)
```

Одиничне коло має довжину 6.283185 радіан.

4. У цьому прикладі складається таблиця значень функції  $y = \cos x$  у проміжку  $[0; \pi/2]$  з кроком  $\pi/24$  і записується в текстовий файл Cosin1.txt.

```
>> x=0:pi/24:pi/2;
>> A=[x;cos(x)];
>> fid = fopen('Cosin1.txt','wt');
>> fprintf(fid,'Таблиця значень\nфункції y=cosx\n');
>> fprintf(fid,'-----\n');
>> fprintf(fid,'| x | y |\n');
>> fprintf(fid,'-----\n');
>> fprintf(fid,' %5.3f %5.3f\n',A);
>> fclose(fid)
```

У поточному каталозі з'явився цей файл і його зміст у текстовому редакторі наведено на рис. 21.21. Праворуч від рис. 21.21 представлені

дані файла Cosin1.txt, прочитані функцією **type** в командному вікні **MatLab** .

5. У прикладі генеруються три вибірки, елементи яких розподілені за стандартним нормальним законом. Вони записуються в текстовий файл tnorm3.txt, який зберігається в поточному каталозі.

```
>> clear all % очищаємо командне вікно й робочу область
>> F=fopen('tnorm3.txt','w'); % відкриваємо файл для запису
>> A=normrnd(0,1,5,3); % генеруємо 3 вибірки по 5 елементів кожна
>> disp(A) % перевіряємо, що буде записане
-0.3999  0.6686 -1.6041
 0.6900  1.1908  0.2573
 0.8156 -1.2025 -1.0565
 0.7119 -0.0198  1.4151
 1.2902 -0.1567 -0.8051
>> fprintf(F,'%7.4f %7.4f %7.4f\n',A); % записуємо форматовані дані
>> fclose(F) % закриваємо файл
ans =
 0
```

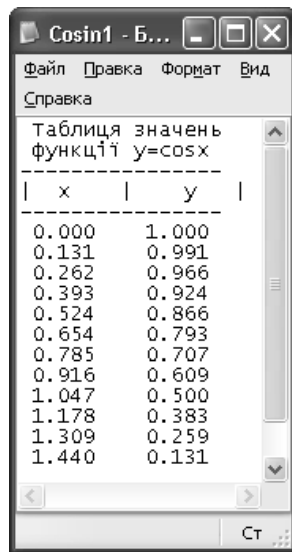
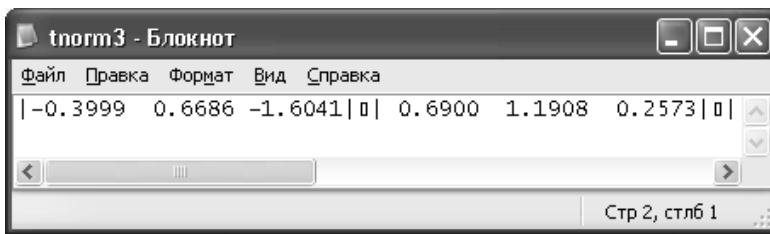


Рис.21.21. **Зміст файла Cosin1.txt, відкритого в текстовому редакторі**

```
>> type('Cosin1.txt')
% Таблиця значень
  функції y=cosx
-----
| x | y |
-----
0.000  1.000
0.131  0.991
0.262  0.966
0.393  0.924
0.524  0.866
0.654  0.793
0.785  0.707
0.916  0.609
1.047  0.500
1.178  0.383
1.309  0.259
1.440  0.131
1.571  0.000
```

Зверніть увагу на те, що у прикладі 5 записується транспонована матриця даних за рядками, тому що в текстовому редакторі вона записується за замовчуванням в один стовпець із роздільниками (рис. 21.22). Праворуч від рис. 21.22 представлені дані цього файла, прочитані функцією **type**.



```
>> type('tnorm3.txt')
|-0.3999 0.6686 -1.6041|
| 0.6900 1.1908 0.2573|
| 0.8156 -1.2025 -1.0565|
| 0.7119 -0.0198 1.4151|
| 1.2902 -0.1567 -0.8051|
```

Рис.21.22. Фрагмент файлу tnorm3.txt  
в текстовому редакторі

6. Маємо наступну таблицю (вона була записана у файлі Book.xls MS Excel ).

	Перша	Друга	Третя
1	0.50	0.17	-0.75
2	0.54	0.21	-0.68
3	0.48	0.17	-0.70
4	-0.49	0.18	0.71
5	-0.52	0.33	-0.69

Перепишемо цю таблицю в текстовий файл Rot.txt:

```
>> x=[0.5 0.17 -0.75; 0.54 0.21 -0.68; 0.48 0.17 -0.7;...
-0.49 0.18 0.71; -0.52 0.33 -0.694]; % уводимо матрицю чисел
>> variables={'Перша','Друга','Третя'}; % заголовки стовпців
>> points=num2str([1;2;3;4;5]); % заголовки рядків
>> s=[sprintf('%s',variables{:}),sprintf('\n')];
>> s=[s sprintf('%2.0f %4.2f %4.2f %4.2fn',[str2num(points) x]']);
>> fid=fopen('Rot.txt','w'); % відкрили файл для запису
>> fprintf(fid,'%s',s); % записали мультирядок
>> fclose(fid); % закрили файл
>> type('Rot.txt') % виводимо файл у командне вікно
Перша Друга Третя
1 0.50 0.17 -0.75
2 0.54 0.21 -0.68
3 0.48 0.17 -0.70
4 -0.49 0.18 0.71
5 -0.52 0.33 -0.69
```

# Використана література

## Література по MatLab

1. Алексеев Е. Р. MatLab 7. Самоучитель / Е. Р. Алексеев, О. В. Чеснокова. – М.: НТ Пресс, 2006. – 520 с.
2. Ануфриев И. Самоучитель MatLab 5.3/6.x (с дискетой). – СПб.: БХВ-Петербург, 2002. – 400 с.
3. Ануфриев И. MatLab 7.0. В подлиннике / И. Ануфриев, А. Смирнов, Е. Смирнова. – СПб.: БХВ-Петербург, 2005. – 1087 с.
4. Дьяконов В. MatLab 6/6.1/6.5 + Simulink 4/5 в математике и моделировании. – М.: СОЛОН-Пресс, 2003. – 576 с.
5. Дьяконов В. MatLab 6/6.1/6.5 + Simulink 4/5. Основы применения. Полное руководство пользователя – 2-е изд. – М.: СОЛОН-Пресс, 2004. – 376 с.
6. Дьяконов В. MatLab 6/6.1/6.5 + Simulink 4/5. Основы применения. Полное руководство пользователя – М.: Солон-Пресс, 2002. – 324 с.
7. Дьяконов В. Математические пакеты расширения MatLab. Специальный справочник / В. Дьяконов, В. Круглов. – СПб.: Питер, 2001. – 400 с.
8. Иглин С. П. Математические расчеты на базе MatLab. – М.: ВНУ Санкт-Петербург, 2005. – 649 с.
9. Иглин С. П. Теория вероятностей и математическая статистика на базе MatLab. – Харьков: НТУ "ХПИ", 2006. – 612 с.
10. Кривилев А. Основы компьютерной математики с использованием системы MatLab. – М.: Лекс-Книга, 2005. – 484 с.
11. Курбатова Е. А. MatLab 7. Самоучитель. – М.: Вильямс, 2006. – 256 с.
12. Мартынов Н. Введение в MatLab 6. – М.: Кудиц-образ, 2002. – 348 с.
13. Поршнева С. В. MatLab 7. Основы работы и программирования. Учебник. – М.: Бином. Лаборатория знаний, 2006. – 320 с.
14. Потёмкин В. Г. MatLab 6: Среда проектирования инженерных приложений. – М.: Диалог-МИФИ, 2003. – 320 с.

15. Потёмкин В. Г. Система MatLab 5 для студентов. – М.: Диалог-МИФИ, 1998 – 314 с.
16. Потёмкин В. Г. Система MatLab. Справочное пособие. – М.: Диалог-МИФИ, 1998 – 350 с.

### **Література з математичної статистики**

17. Айвазян С. А. Прикладная статистика. Исследование зависимостей / С. А. Айвазян, И. С. Енюков, Л. Д. Мешалкин. – М.: Финансы и статистика, 1985. – 487 с.
18. Айвазян С. А. Прикладная статистика: Классификация и снижение размерности / С. А. Айвазян, В. М. Бухштабер, И. С. Енюков, Л. Д. Мешалкин. – М.: Финансы и статистика, 1989. – 607 с.
19. Айвазян С. А. Прикладная статистика и основы эконометрики / С. А. Айвазян, В. С. Мхитарян. – М.: Юнити, 1998, 1022 с.
20. Бард Й. Нелинейное оценивание параметров. – М.: Статистика, 1979. – 380 с.
21. Болч Б. У. Многомерные статистические методы для экономики / Б. У. Болч, К. Д. Хуань. – М.: Статистика, 1979. – 317 с.
22. Бородич С. А. Эконометрика. Серия: Экономическое образование, 2001. – 408 с.
23. Вапник В. Н. Восстановление зависимостей по эмпирическим данным. – М.: Наука, 1979. – 447 с.
24. Вероятность и математическая статистика: Энциклопедия / Гл. ред. Ю. В. Прохоров. – М.: Большая Российская энциклопедия, 1999. – 910 с.
25. Гайдышев И. Анализ и обработка данных. Специальный справочник. – СПб.: Питер, 2001. – 752 с.
26. Демиденко Е. З. Линейная и нелинейная регрессия. – М.: Финансы и статистика, 1981. – 302 с.
27. Джонстон Д. Эконометрические методы. – М.: Статистика, 1980. – 444 с.
28. Доугерти К. Введение в эконометрику. – М.: ИНФРА-М, 1997.–402 с.
29. Дрейпер Н. Прикладной регрессионный анализ: В 2-х кн. / Н. Дрейпер, Г. Смит. – М: Финансы и статистика, 1986. – 366 с.

30. Дюк В. А. Обработка данных на ПК в примерах. – СПб: Питер, 1997. – 210 с.
31. Дюран Б. Кластерный анализ / Б. Дюран, П. Оделл. – М.: Статистика, 1977. – 128 с.
32. Єгоршин О. О. Методи багатовимірного статистичного аналізу: Навч. посібник / О. О. Єгоршин, А. М. Зосімов, В. С. Пономаренко. – К.: ІЗМН, 1998. – 208 с.
33. Егоршин А. А. Корреляционно–регрессионный анализ. Курс лекций и лабораторных работ / А. А. Егоршин, Л. М. Малярец. – Харьков: Основа, 1998. – 208 с.
34. Егоршин А. А. Практикум по эконометрии в Excel. Учебное пособие для экономических вузов / А. А. Егоршин, Л. М. Малярец. – Харьков: ИД "ИНЖЭК", 2005. – 100 с.
35. Ермаков С. М. Математическая теория оптимального эксперимента / С. М. Ермаков, А. А. Жиглявский. – М.: Наука, 1982. – 319 с.
36. Катышев П. К. Сборник задач к начальному курсу эконометрики / П.К. Катышев, А.А. Пересецкий. – М.: Дело, 1999. – 207 с.
37. Кокс Д. Р. Анализ данных типа времени жизни / Д. Р. Кокс, Д. Оукс. – М.: Финансы и статистика, 1988. – 186 с.
38. Кремер Н. Ш. Эконометрика / Н. Ш. Кремер, Б. А. Путко. – М.: ЮНИТИ, 2002, – 311 с.
39. Лимер Э. Статистический анализ неэкспериментальных данных. – М.: Финансы и статистика, 1983. – 320 с.
40. Лоули Д. Факторный анализ как статистический метод / Д. Лоули, А. Максвелл. – М.: Мир, 1967. – 144 с.
41. Магнус Я. Р. Эконометрика. Начальный курс. Учебное пособие / Я. Р. Магнус, П. К. Катышев, А. А. Пересецкий. – 3-е, изд. испр. и доп. – М.: Дело, 2000. – 248 с.
42. Маленво Э. Статистические методы эконометрии. – М.: Статистика, вып. 1, 1975. – 424 с.; вып. 2, 1976. – 325 с.
43. Многомерный статистический анализ в экономике / Под ред. В. Н. Тамашевича. – М.: Юнити-Дана, 1999. – 598 с.
44. Мостеллер Ф. Анализ данных и регрессия / Ф. Мостеллер, Дж. Тьюки. – В 2-х вып. – М.: Финансы и статистика, 1982. – 624 с.

45. Песаран М. Динамическая регрессия: теория и алгоритмы / М. Песаран, Л. Слейтер. – М.: Финансы и статистика, 1984. – 310 с.
46. Розин Б. Б. Теория распознавания образов в экономических исследованиях. – М.: Статистика, 1973. – 126 с.
47. Себер Дж. Линейный регрессионный анализ. – М.: Мир, 1980. – 456с.
48. Справочник по прикладной статистике. – М.: Финансы и статистика, 1990. – 526 с.
49. Хардле В. Прикладная непараметрическая регрессия. – М.: Мир, 1993. – 526 с.
50. Харман Г. Современный факторный анализ. – М.: Статистика, 1972. – 346 с.
51. Хьюбер П. Робастность в статистике. – М.: Мир, 1984. – 304 с.
52. Шеффе Г. Дисперсионный анализ. – М.: Наука, 1980. – 511 с.
53. Экономическая статистика. Эконометрика: Методические материалы по экономическим дисциплинам для преподавателей средних школ и вузов. / Под ред. Л. С. Гребнева, О. И. Образцова, О. В. Назарова, Г. Г. Канторович. – М.: Высшая школа экономики, 2000. – 112 с.
54. Эфрон Б. Нетрадиционные методы многомерного статистического анализа: Сборник статей. – М.: Финансы и статистика, 1988. – 263 с.

### **Анотовані інтернет-ресурси**

55. <http://inftech.webservis.ru/it/database/datamining> – Сайт Санкт-Петербурзького інституту інформатики і автоматизації РАН. Детально розглянуті питання інтелектуального аналізу даних.
56. <http://www.amstat.org> – Сайт американської статистичної асоціації. Наведені керивництва і статті з використання статистичних методів обробки інформації для аналізу експериментальних даних.
57. <http://www.alexbar.narod.ru> – На сайті розміщені матеріали за обробки експериментальної інформації.
58. <http://www.math.rsu.ru/mexmat/kvm/MME> – На сайті є описи універсальних математичних пакетів і прикладних програм.
59. [infoscope.forth.ru](http://infoscope.forth.ru); <http://algolist.manual.ru/maths> – На цих сайтах є довідники основних статистичних розподілів.

60. <http://www.nag.co.uk> – Сайт NAG's Statistical software. Містить опис основних статистичних процедур и алгоритмів.

61. <http://www.exponenta.ru> – Математичний російський портал.

# Додаток А

## Додаткові статистичні функції

Повні тексти цих функцій (оригінальні m-файли) знаходяться за адресою

<http://www.mathworks.com/matlabcentral/fileexchange/loadcategory.do> у категорії Statistics and Propability

Нижче наведені тексти зі скороченими коментарями. Ці m-файли (або оригінальні m-файли) треба скопіювати в директорію work **MatLab** (C:\Program Files\MATLAB71\work).

### chi2test.m

```
function [chi2, critical] = chi2test (data, n, alpha, dist, a1, a2, a3);
% Copyright 2004 Leonardo Salomone, Carleton University, Ottawa, Canada
if nargin < 4, error('Not enough input arguments'); end
nsamples = length(data);
if nsamples
+вай      es < 20,
    error('Sample data too small, chi-square test not recommended');
elseif nsamples < 50,
    if n < 5,          error('Number of intervals too small'); end
    if n > 10,         error('Number of intervals too large'); end
elseif nsamples < 100,
    if n < 10,        error('Number of intervals too small'); end
    if n > 20,        error('Number of intervals too large'); end
else
    if n < sqrt(nsamples), error('Number of intervals too small'); end
    if n > nsamples/5,   error('Number of intervals too large'); end
end;
switch (7 - nargin)
    case 2
        prob = inline(sprintf('cdf('%s', b, %10.10g) - cdf('%s', a,
%10.10g)', dist, a1, dist, a1), 'a', 'b');
        invcdf = inline(sprintf('icdf('%s', x, %10.10g)', dist, a1),
'x');
    case 1
        prob = inline(sprintf('cdf('%s', b, %10.10g, %10.10g) -
cdf('%s', a, %10.10g, %10.10g)', dist, a1, a2, dist, a1, a2), 'a', 'b');
        invcdf = inline(sprintf('icdf('%s', x, %10.10g, %10.10g)', dist,
a1, a2), 'x');
    case 0
        prob = inline(sprintf('cdf('%s', b, %10.10g, %10.10g, %10.10g) -
cdf('%s', a, %10.10g, %10.10g, %10.10g)', dist, a1, a2, a3, dist, a1, a2, a3),
'a', 'b');
        invcdf = inline(sprintf('icdf('%s', x, %10.10g, %10.10g,
%10.10g)', dist, a1, a2, a3), 'x');
    otherwise
        return;
```

```

end;
% find out the bin edges, for equal probabilities of continuous distributions, using the inverse CDF
pi = (1/n) .* [0:n];
intvls = invcdf(pi);
switch (dist)
case 'exp'
    intvls(end) = inf;
end;
o_freq = histc(data, intvls);
o_freq = o_freq(1:end-1);
e_freq = prob(intvls(1),intvls(2)) .* ones(1,n);
e_freq = length(data) .* e_freq;
chi2bins = ((o_freq - e_freq).^2)./e_freq;
chi2 = sum(chi2bins);
df = n - (nargin - 3);
critical = chi2inv(1-alpha, df);

```

## Btest.m

```

function [Btest] = Btest(X,alpha)
if nargin < 2,
    alpha = 0.05;
end
k=max(X(:,2));
fprintf('The number of samples are:%2i\n\n', k);
n=[];s2=[];
indice=X(:,2);
for i=1:k
    Xe=find(indice==i);
    eval(['X' num2str(i) '=X(Xe,1);']);
    eval(['n' num2str(i) '=length(X' num2str(i) ') ;']);
    eval(['s2' num2str(i) '=(std(X' num2str(i) ')^2) ;']);
    eval(['xn= n' num2str(i) ' ;']);
    eval(['xs2= s2' num2str(i) ' ;']);
    n=[n;xn];s2=[s2;xs2];
end
fprintf('-----\n');
disp(' Sample      Size      Variance')
fprintf('-----\n');
for i=1:k
    fprintf('      %d      %2i      %.4f\n',i,n(i),s2(i))
end
fprintf('-----\n');
disp(' ')
%Bartlett's Procedure.
deno=sum(n)-k;
suma=0;
for i=1:k

```

```

        eval(['suma =suma + (n' num2str(i) '-1)*s2' num2str(i) ';']);
    end
    Sp=suma/deno;
    Falta=0;
    for i=1:k
        eval(['Falta =Falta + (n' num2str(i) '-1)*log(s2' num2str(i) ');']);
    end
    B=((sum(n)-k)*log(Sp))-Falta;
    sumal=0;
    for i=1:k
        eval(['sumal=sumal + 1/(n' num2str(i) '-1);']);
    end
    suma2=0;
    for i=1:k
        eval(['suma2=suma2 + 1/((n' num2str(i) '-1)^2);']);
    end
    C=1+((1/(3*(k-1)))*(sumal-(1/deno)));
    X2=B/C; %Chi-squared-statistic.
    v=k-1; %degrees of freedom.
    F=X2/v; %F-statistic.
    P = 1 - chi2cdf(X2,v); %probability associated to the Chi-squared-
statistic.
    df=v;
    df1=v;df2=sum(n)-k-1;
    fprintf('Bartlett's Test for Equality of Variances X2=%3.4f, df=%2i,
F=%7.4f, df1=%2i, df2=%2i\n', X2,df,F,df1,df2);
    fprintf('Probability associated to the Chi-squared statistic = %3.4f\n',
P);
    if P >= alpha;
        fprintf('The associated probability for the Chi-squared test is equal or
larger than% 3.2f\n', alpha);
        fprintf('So, the assumption of homoscedasticity was met.\n');
    else
        fprintf('The associated probability for the Chi-squared test is smaller
than% 3.2f\n', alpha);
        fprintf('So, the assumption of homoscedasticity was not met.\n');
    end
end

```

## cochcdf.m

```

function y=cochcdf(X,k,v)
% Використовується як підпрограма в функції Cochtest
% Created by A. Trujillo-Ortiz and R. Hernandez-Walls
% Facultad de Ciencias Marinas, Universidad Autonoma de Baja California
% Apdo. Postal 453, Ensenada, Baja California, Mexico.atrujo@uabc.mx
% April 8, 2002.
alpha=0.05;
if nargin < 3,
    error('Requires three input arguments.');
```

```

end
[errorcode X k v] = distchck(3,X,k,v);
if errorcode > 0
    error('Requires non-scalar arguments to match in size.');
```

```

end
x=linspace(.000001,X,1000001);
DX=x(2)-x(1);
y=1/beta(.5*v,.5*v*(k-1));
y=y*(x.^((.5*v)-1));
y=y.*((1-x).^((.5*v*(k-1))-1));
N=length(x);
y=1-(k*(1-(DX.*(y(1)+y(N) + 4*sum(y(2:2:N-1))+2*sum(y(3:2:N-2)))/3.0)));
if y <= 0;
    y = abs(y);
else
    y;
end;
```

## Cochtest.m

```

function [Cochtest]=Cochtest(X,alpha)
% Created by A. Trujillo-Ortiz and R. Hernandez-Walls
% Facultad de Ciencias Marinas, Universidad Autonoma de Baja California
% Apdo. Postal 453, Ensenada, Baja California, Mexico. atrujo@uabc.mx
% April 21, 2003.
if nargin < 2,
    alpha = 0.05;
end
k=max(X(:,2));
fprintf('The number of samples are:%2i\n\n', k);
n=[];n2=[];s2=[];
indice=X(:,2);
for i=1:k
    Xe=find(indice==i);
    eval(['X' num2str(i) '=X(Xe,1);']);
    eval(['n' num2str(i) '=length(X' num2str(i) ') ;']);
    eval(['n2' num2str(i) '=(length(X' num2str(i) ')^2);']);
    eval(['s2' num2str(i) '=(std(X' num2str(i) ')^2) ;']);
    eval(['xn= n' num2str(i) ';']);
    eval(['xn2= n2' num2str(i) ';']);
    eval(['xs2= s2' num2str(i) ';']);
    n=[n;xn];n2=[n2;xn2];s2=[s2;xs2];
end
disp(' Sample      Size      Variance')
for i=1:k
    fprintf('   %d      %2i      %.4f\n',i,n(i),s2(i))
end
disp(' ')
%Cochran's test procedure.
```

```

C=max(s2)/sum(s2); %Cochran's C statistic.
v=k-1;
no=(1/v)*(sum(n)-(sum(n2)/sum(n))); %procedure for equal or unequal sample
sizes.
v=round(no)-1; %sample degrees of freedom.
P=1-cochcdf(C,k,v); %probability associated to the Cochran's C statistic.
fprintf('Cochran''s Test for Equality of Variances C = %3.4f\n', C);
fprintf('Probability associated to the Cochran''s statistic = %3.4f\n',
P);
if P >= alpha;
    fprintf('The associated probability for the Cochran'' test is equal or
larger than% 3.2f\n', alpha);
    fprintf('So, the assumption of homoscedasticity was met.\n');
else
    fprintf('The associated probability for the Cochran''s test is smaller
than% 3.2f\n', alpha);
    fprintf('So, the assumption of homoscedasticity was not met.\n');
end

```

## Додаток Б

### Перелік статистичних функцій частини 2

ADDEDVARPLOT . 193	EXEL LINK ..... 478	LILLIETEST ..... 31
ANOVA1 ..... 77	FACTORAN ..... 310	LINKAGE ..... 264
ANOVA2 ..... 85	FF2N ..... 403	LOAD ..... 464
ANOVAN ..... 92	File / Import Data ... 485	LSCOV ..... 206
AOCTOOL ..... 132	File/Open ..... 483	LSQNONNTG ..... 209
BARTTEST ..... 324	FOPEN ..... 470	MAHAL ..... 342
BBDESIGN ..... 418	FOPEN, FGETL .... 477	MANOVA1 ..... 127
BIPLOT ..... 322	FPRINTF ..... 501	MANOVA CLUSTER 130
BTEST ..... 69	FRACFACT ..... 407	MDSCALE ..... 336
CANDEXCH ..... 424	FREAD ..... 472	MULTCOMPARE .... 106
CANDGEN ..... 421	FRIDMAN ..... 123	NLINFIT ..... 237
CANONCORR ..... 327	FULLFACT ..... 404	NLINTOOL ..... 244
CAPABLE ..... 381	FWRITE ..... 498	NLPARCI ..... 248
CAPAPLOT ..... 384	GLMFIT ..... 212	NLPREDCI ..... 251
CASEREAD ..... 466	GLMVAL ..... 223	PCACOV ..... 304
CASEWRITE ..... 495	HADAMARD ..... 412	PCARES ..... 299
CCDESIGN ..... 415	HMMDECODE ..... 459	PDIST ..... 257
CHI2TEST ..... 6	HMMESTIMATE .... 453	POLYCONF ..... 157
CLASSIFY ..... 330	HMMGENERATE .. 447	POLYFIT ..... 150
CLUSTER ..... 273	HMMTRAIN ..... 456	POLYVAL ..... 153
CLUSTERDATA ... 277	HMMVITERBI ..... 450	PRINCOMP ..... 296
CMDSCALE ..... 333	INCONSISTENT ... 281	PROCRUSTES ..... 340
COCHTEST ..... 72	JBTEST ..... 25	RANKSUM ..... 53
COPHENET ..... 280	KMEANS ..... 283	RCOPLOT ..... 181
CORDEXCH ..... 429	KRUSKALWALLIS 119	REGRESS ..... 160
DAUGMENT ..... 432	KSTEST ..... 11	REGSTATS ..... 182
DCOVARY ..... 435	KSTEST2 ..... 19	RIDGE ..... 197
DENDROGRAM ... 270	LEVERAGE ..... 190	ROBUSTFIT ..... 201
DUMMYVAR ..... 146	LHSDESIGN ..... 438	ROTATEFACTORS 306
EWMAPLOT ..... 386	LHSNORM ..... 441	ROWEXCH ..... 426

RSTOOL .....	174	TBLREAD .....	465	TTEST2 .....	48
SAVE .....	492	TBLWRITE .....	494	TYPE .....	470
SCHART .....	392	TEXTREAD .....	467	UIIMPORT .....	489
SIGNRANK .....	58	TREEDISP .....	358	X2FX .....	173
SIGNTEST .....	64	TREEFIT .....	344	XBARPLOT .....	395
SILHOUETTE .....	289	TREEPRUNE .....	363	XLSREAD .....	481
SQUAREFORM ....	292	TREETEST .....	369	XLSWRITE .....	497
STEPWISE .....	167	TREEVAL .....	374	ZTEST .....	37
STEPWISEFIT .....	171	TTEST .....	42		

## Зміст частини 2

<b>Вступ</b> .....	<b>1</b>
<b>10. Перевірка статистичних гіпотез про згоду розподілу експериментальним даним</b> .....	<b>6</b>
<b>CHI2TEST</b> $\chi^2$ -критерій згоди Пірсона .....	6
<b>KSTEST</b> Тест Колмогорова-Смірнова на непротириччя розподілу значень випадкової величини заданому закону .....	11
<b>KSTEST2</b> Тест Колмогорова-Смірнова для порівняння законів розподілу двох сукупностей значень випадкових величин .....	19
<b>JBTEST</b> Тест Яркі-Бера на непротириччя розподілу значень випадкової величини нормальному закону .....	25
<b>LILLIETEST</b> Тест Лільєфорса на непротириччя розподілу значень випадкової величини нормальному закону .....	31
<b>11. Перевірка статистичних гіпотез (порівняння вибірових характеристик)</b> .....	<b>37</b>
<b>ZTEST</b> Перевірка параметричної гіпотези про значення середнього за відомої дисперсії нормальної сукупності .....	37
<b>TTEST</b> Перевірка параметричної гіпотези про значення середнього за невідомої дисперсії нормальної сукупності .....	42
<b>TTEST2</b> Перевірка параметричної гіпотези про різницю значень середніх двох сукупностей .....	48
<b>RANKSUM</b> Ранговий тест Вілкоксона на рівність медіан двох незалежних сукупностей .....	53
<b>SIGNRANK</b> Знаковий тест Вілкоксона .....	58
<b>SIGNTEST</b> Знаковий тест .....	64
<b>BTEST</b> Критерій Бартлета для порівняння дисперсій .....	69
<b>COCHTEST</b> Критерій Кокрена для порівняння дисперсій .....	72
<b>12. Дисперсійний, коваріаційний і регресійний аналізи</b> .....	<b>75</b>
<b>12.1. Дисперсійний аналіз</b> .....	<b>76</b>
<b>ANOVA1</b> Однофакторний дисперсійний аналіз (ANOVA) .....	77
<b>ANOVA2</b> Двофакторний дисперсійний аналіз ....//.....	85
<b>ANOVAN</b> Багатофакторний дисперсійний аналіз (MANOVA) .....	92
<b>MULTCOMPARE</b> Перевірка параметричних гіпотез за парного порівняння групових середніх .....	106
<b>KRUSKALWALLIS</b> Непараметричний тест Краскала-Уоліса .....	119
<b>FRIDMAN</b> Тест Фрідмана .....	123
<b>MANOVA1</b> Однофакторний багатовимірний дисперсійний аналіз .....	127
<b>MANOVACLUSTER</b> Дендрограма результатів однофакторного багатовимірного дисперсійного аналізу .....	130
<b>12.2. Коваріаційний аналіз</b> .....	<b>132</b>

<b>AOCTOOL</b>	Інтерактивний графічний коваріаційний аналіз .....	132
<b>DUMMYVAR</b>	Формування матриці індикаторних змінних .....	146
<b>12.3. Регресійний аналіз</b>	.....	<b>150</b>
<b>POLYFIT</b>	Розрахунок коефіцієнтів однофакторної регресійної поліноміальної моделі довільного порядку .....	150
<b>POLYVAL</b>	Розрахунок значень однофакторної регресійної поліноміальної моделі довільного порядку .....	153
<b>POLYCONF</b>	Розрахунок значень залежної змінної і довірчих інтервалів для однофакторної регресійної поліноміальної моделі довільного порядку .....	157
<b>REGRESS</b>	Множинна лінійна регресія .....	160
<b>STEPWISE</b>	Покрокова регресія в інтерактивному режимі .....	167
<b>STEPWISEFIT</b>	Послідовна регресія .....	171
<b>X2FX</b>	Перетворення матриці факторів у матрицю експерименту .....	173
<b>RSTOOL</b>	Візуалізація багатовимірної поверхні відгуку .....	174
<b>12.4. Діагностика регресійних моделей</b>	.....	<b>180</b>
<b>RCOPLLOT</b>	Графік залишкових похибок .....	181
<b>REGSTATS</b>	Розрахунок параметрів і діагностика множинної регресійної моделі .....	182
<b>LEVERAGE</b>	Розрахунок коефіцієнтів впливу окремих спостережень на параметри поліноміальної регресійної моделі .....	190
<b>ADDEDVARPLOT</b>	Графік регресії з доданою змінною .....	193
<b>12.5. Спеціальні процедури регресійного аналізу</b>	.....	<b>197</b>
<b>RIDGE</b>	Ридж-регресія .....	197
<b>ROBUSTFIT</b>	Робастна лінійна регресія .....	201
<b>LSCOV</b>	Метод найменших квадратів із заданою коваріаційною матрицею .....	206
<b>LSQNONNTG</b>	Метод найменших квадратів з невід'ємними коефіцієнтами .....	209
<b>12.6. Узагальнені лінійні моделі</b>	.....	<b>212</b>
<b>GLMFIT</b>	Добір узагальненої лінійної моделі .....	212
<b>GLMVAL</b>	Розрахунок значень залежної змінної за узагальненою регресійною моделлю .....	223
<b>13. Нелінійні моделі</b>	.....	<b>237</b>
<b>NLINFIT</b>	Нелінійна регресія за методом найменших квадратів з використанням ітераційної процедури Гауса-Ньютона .....	237
<b>NLINTOOL</b>	Інтерактивний нелінійний регресійний аналіз .....	244
<b>NLPARC</b>	Розрахунок інтервальних оцінок коефіцієнтів нелінійного рівняння регресії .....	248

<b>NLPREDCI</b> Розрахунок довірчих інтервалів для нелінійної регресійної моделі .....	251
<b>14. Функції кластерного аналізу .....</b>	<b>256</b>
<b>PDIST</b> Розрахунок парних відстаней між об'єктами вихідної множини даних .....	257
<b>LINKAGE</b> Формування ієрархічного дерева бінарних кластерів .....	264
<b>DENDROGRAM</b> Графік дендрограми ієрархічного дерева кластерів .....	270
<b>CLUSTER</b> Кластеризація об'єктів за результативним параметром функції linkage .....	273
<b>CLUSTERDATA</b> Побудова кластерів за даними .....	277
<b>COPHENET</b> Коефіцієнт кофенетичної (якісної) кореляції .....	280
<b>INCONSISTENT</b> Коефіцієнти несумісності .....	281
<b>KMEANS</b> Кластеризація об'єктів за внутрішньогруповими середніми .....	283
<b>SILHOUETTE</b> Силуети кластерів .....	289
<b>SQUAREFORM</b> Перетворення вектора відстаней в матрицю .....	292
<b>15. Функції зниження розмірності задачі .....</b>	<b>294</b>
<b>PRINCOMP</b> Аналіз головних компонент за матрицею даних .....	296
<b>PCARES</b> Розрахунок залишків (нев'язок) після аналізу головних компонент .....	299
<b>PCACOV</b> Аналіз головних компонент за коваріаційною матрицею .....	304
<b>ROTATEFACTORS</b> Обертання для факторного аналізу .....	306
<b>FACTORAN</b> Загальний факторний аналіз .....	310
<b>BIPLOT</b> Діаграма факторних навантажень і об'єктів .....	322
<b>BARTTEST</b> Тест Бартлета для оцінки розмірності простору головних компонент .....	324
<b>16. Функції аналізу багатовимірних випадкових величин .....</b>	<b>327</b>
<b>CANONCORR</b> Канонічний кореляційний аналіз .....	327
<b>CLASSIFY</b> Лінійний дискримінантний аналіз .....	330
<b>CMDSCALE</b> Класичне багатовимірне шкалювання .....	333
<b>MDSCALE</b> Метричне й неметричне багатовимірне шкалювання .....	336
<b>PROCRUSTES</b> Прокрустів аналіз .....	340
<b>MAHAL</b> Відстані Махаланобіса .....	342
<b>17. Функції нелінійного регресійного аналізу на підставі графа можливих рішень .....</b>	<b>344</b>

<b>TREEFIT</b>	Розрахунок параметрів ієрархічної нелінійної регресійної моделі або бінарного дерева класифікації спостережень .....	344
<b>TREEDISP</b>	Графічне представлення ієрархічної нелінійної регресійної моделі або бінарного дерева класифікації спостережень .....	358
<b>TREEPRUNE</b>	Розрахунок параметрів скороченого бінарного дерева рішень ієрархічної регресійної моделі .....	363
<b>TREETEST</b>	Визначення похибок ієрархічної нелінійної регресійної моделі .....	369
<b>TREEVAL</b>	Розрахунок залежної змінної за ієрархічною нелінійною регресійною моделлю .....	374
<b>18. Керування статистичними процесами .....</b>		<b>380</b>
<b>CAPABLE</b>	Розрахунок індексів відтворюваності .....	381
<b>CAPAPLOT</b>	Графік відтворюваності процесу .....	384
<b>EWMAPLOT</b>	Контрольна карта експоненціально зваженого ковзного середнього (EWMA контрольна карта) .....	386
<b>SCHART</b>	Контрольна карта середніх квадратичних відхилень (s-контрольна карта) .....	392
<b>XBARPLOT</b>	Контрольна карта середніх арифметичних значень ( $\bar{X}$ -контрольна карта) .....	395
<b>19. Планування експерименту .....</b>		<b>402</b>
<b>19.1. Повні факторні експерименти (ПФЕ) .....</b>		<b>403</b>
<b>FF2N</b>	Генерування матриці повного факторного експерименту для факторів, що варіюють на 2-х рівнях .....	403
<b>FULLFACT</b>	Генерування матриці повного факторного експерименту для довільного числа факторів й кількості їх рівнів .....	404
<b>19.2. Дробові факторні експерименти (ДФЕ) .....</b>		<b>407</b>
<b>FRACFACT</b>	Генерування матриці дробового факторного експерименту для довільного числа факторів з двома рівнями варіювання .....	407
<b>HADAMARD</b>	Генерування матриці Адамара .....	412
<b>19.3. Композиційні плани експерименту .....</b>		<b>414</b>
<b>CCDESIGN</b>	Генерування матриці центрального композиційного плану .....	415
<b>BBDESIGN</b>	Генерування матриці плану Бокса-Бенкена .....	418
<b>19.4. D-оптимальні плани експерименту .....</b>		<b>420</b>
<b>CANDGEN</b>	Генерування початкової множини точок у факторному просторі для D-оптимального плану .....	421
<b>CANDEXCH</b>	Генерування D-оптимального плану з початкової множини точок у факторному просторі на підставі алгоритму перестановки рядків .....	424
<b>ROWEXCH</b>	Генерування матриці D-оптимального плану на підставі алгоритму перестановки рядків .....	426

<b>CORDEXCH</b>	Генерування матриці D-оптимального плану на підставі алгоритму зміни координат .....	429
<b>DAUGMENT</b>	Доповнення матриці значень факторів до D-оптимального плану .....	432
<b>DCOVARY</b>	Генерування D-оптимального блокового плану .....	435
<b>9.5. Латинські квадрати, куби, гіперкуби</b>		<b>438</b>
<b>LHSDESIGN</b>	Функція генерування вибірки чисел латинського гіперкуба .....	438
<b>LHSNORM</b>	Функція генерування вибірки чисел за нормально розподіленим латинським гіперкубом .....	441
<b>20. Марківські моделі</b>		<b>444</b>
<b>HMMGENERATE</b>	Генерування схованої марківської послідовності ...	447
<b>HMMVITERBI</b>	Відновлення послідовності станів .....	450
<b>HMMESTIMATE</b>	Оцінка параметрів моделі за відомої послідовності станів .....	453
<b>HMMTRAIN</b>	Оцінка параметрів моделі за невідомої послідовності станів .....	456
<b>HMMDECODE</b>	Апостеріорні ймовірності станів .....	459
<b>21. Функції введення, форматування й збереження інформації</b>		<b>463</b>
<b>21.1. Введення даних</b>		<b>463</b>
<b>LOAD</b>	Введення числових даних з txt-файла .....	464
<b>TBLREAD</b>	Введення даних з форматованої таблиці .....	465
<b>CASEREAD</b>	Введення символічної інформації .....	466
<b>TEXTREAD</b>	Універсальна функція введення даних .....	467
<b>TYPE</b>	Функція читання даних з файлів різних форматів .....	470
<b>FOPEN</b>	Відкривання файла .....	470
<b>FREAD</b>	Читання бінарних файлів .....	472
<b>FOPEN, FGETL</b>	Читання даних за рядками .....	477
<b>EXEL LINK</b>	Зв'язок MATLAB з EXCEL .....	478
<b>XLSREAD</b>	Введення даних з xls-файла Excel .....	481
<b>File/Open</b>	Відкрити файл засобами Windows .....	483
<b>File / Import Data</b>	Імпорт даних у робочу область Workspace .....	485
<b>UIIMPORT</b>	Відкриває Майстер імпорту .....	489
<b>21.2. Форматування даних</b>		<b>490</b>
<b>21.3. Збереження інформації</b>		<b>492</b>
<b>SAVE</b>	Збереження файла .....	492
<b>TBLWRITE</b>	Запис табульованих даних .....	494
<b>CASEWRITE</b>	Запис у текстовий файл за рядками .....	495
<b>XLSWRITE</b>	Запис даних в xls-файли .....	497

<b>FWRITE</b>	Запис у бінарні файли .....	498
<b>FPRINTF</b>	Збереження текстової інформації .....	501
<b>Використана література</b>	.....	<b>506</b>
Література по MatLab	.....	506
Література з математичної статистики	.....	507
Анотовані інтернет-ресурси	.....	509
<b>Додаток А</b>	.....	<b>511</b>
<b>Додаток Б</b>	.....	<b>516</b>
<b>Зміст частини 2</b>	.....	<b>516</b>

НАВЧАЛЬНЕ ВИДАННЯ

**Єгоршин Олександр Олександрович  
Малярець Людмила Михайлівна  
Сінкевич Борис Володимирович**

# **ДОВІДНИК З МАТЕМАТИЧНОЇ СТАТИСТИКИ З ПРИКЛАДАМИ ОБЧИСЛЕНЬ У MATLAB**

**Навчально-практичний посібник**

**Частина 2**

Відповідальний за випуск **Малярець Л. М.**

Відповідальний редактор **Єдова Л. М.**

Редактор **Шелепова Т. Ф.**

Коректор **Гергеша А. В.**

План 2009 р. Поз. № 31-П.

Підп. до друку 4.05.2010 Формат 60 × 90 1/16. Папір MultiCopy. Друк Riso.

Ум.-друк. арк. 31,75. Обл.-вид. арк. 39,69. Тираж 400 прим. Зам. № 321

---

Видавець і виготівник — видавництво ХНЕУ, 61001, м. Харків, пр. Леніна, 9а

*Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи*

*Дк № 481 від 13.06.2001 р.*