Інститут спеціального зв’язку та захисту інформації
Національного технічного університету України
"Київський політехнічний інститут"

ЛАНДЕ Дмитро Володимирович
ЗУБОК Віталій Юрійович

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до практичних занять з навчальної дициплiни
«МАТЕМАТИЧНІ МЕТОДИ ДОСЛІДЖЕННЯ ОПЕРАЦІЙ»

Під загальною редакцією

доктора технічних наук Ланде Д.В.

Рекомендовано Вченою радою Інституту спеціального зв’язку та захисту інформації НТУУ «КПІ» для використання у навчальному процесі з підготовки фахівців за освітньо-кваліфікаційним рівнем бакалавр з напряму підготовки «Комп’ютерні науки»

Київ

Видавництво ІСЗЗІ НТУУ «КПІ»

2013

УДК 683.3.01:004.451.53
ББК 32.976я73
Л 93

Рецензенти: *Я.О. Калиновський*, д.т.н., с.н.с. (ІПРІ НАН України)
С.Д. Винничук, д.т.н., с.н.с. (ІПМЕ ім. Г.Є.Пухова НАН України)

Гриф надано Вченою радою
ІСЗЗІ НТУУ «КПІ»
(Протокол № 3 від 16.04.2013)

Л 93 Методичний посібник до практичних занять з навчальної дисципліни «Математичні методи дослідження операцій» / Ланде Д.В., Зубок В.Ю. / Під заг. ред. Д.В Ланде. – К.: ІСЗЗІ НТУУ «КПІ», 2013. – 90 с.
ISBN 978-966-96800-0-6

Коротко викладений матеріал, необхідний для проведення практичних занять, в рамках яких мають бути набутіі навички розв’язання задач дослідження операцій, зокрема, математичного програмування, в тому числі з застосування систем MATLAB і MathCAD.

Навчальне видання орієнтоване на курсантів, слухачів і студентів, які навчаються за напрямом підготовки «Комп’ютерні науки» і вивчають нормативну навчальну дисципліну «Математичні методи дослідження операцій». Також може бути корисне всім, хто хоче самостійно здобути практичні навички застосування сучасних засобів комп’ютерного моделювання.

УДК 683.3.01:004.451.53
ББК 32.976я73

ЗМІСТ

Передмова

Сучасні фахівці, які працюють у сфері комп'ютерних наук, обов'язково повинні мати теоретичні знання та практичні навички роботи із сучасними системами комп'ютерного моделювання (СКМ).

Тому при вивченні дисципліни «Математичні методи дослідження операцій», яка є нормативною з напряму підготовки «Комп'ютерні науки», велика увага надається питанням застосування СКМ, зокрема, при розв'язанні задач математичного програмування і вирішенні оптимізаційних задач.

Якщо на лекціях з даного курсу вивчаються теоретичні основи методів дослідженння операцій, які дозволяють знаходити оптимальні рішення цілеспрямовано, а не простим перебиранням варіантів, то практичні заняття організовані з орієнтацією на те, що (за словами О. С. Вентцель) «сучасні ЕОМ, як правило, оснащені підпрограмами для розв'язання задач лінійного програмування (цей випадок є частковим), отже особі, яка бажає їх розв'язувати, нема навіть особливої необхідності навчатися вирішенню таких задач «вручну» – праця вкрай неприємна та виснажлива».

В даній навчальній дисципліні практичним заняттям відведено значне місце, оскільки саме вони дозволяють отримувати необхідні практичні навички. На практичних заняттях у комп'ютерному класі повинні бути набуті практичні знання та навички рішення задач математичного програмування, теорії ігор, класифікації, теорії складних мереж у середовищах сучасних СКМ, зокрема, засобами систем MATLAB і MathCAD.

Це видання покликане, перед усім, підвищити ефективність практичних занять. Застосовувані комп'ютерні засоби не є предметом даного курсу, тому основні відомості щодо системи MATLAB викладено у окремому додатку, а відповідні посилання на деякі джерела надано у списку літератури.

Цей посібник розраховано на курсантів, слухачів та студентів бакалаврату за напрямом підготовки «Комп'ютерні науки» і які, відповідно, мають достатню підготовку в таких галузях математики, як математичний аналіз та диференціальні рівняння, основи вищої алгебри, та повинні засвоїти необхідні елементарні знання щодо мови програмування системи MATLAB і основних можливостей пакету MathCAD.

1. Початок роботи в системі MATLAB

Система MATLAB (MATrix LABoratoty) призначена для виконання наукових та інженерних розрахунків, з її допомогою розв'язуються задачі обчислювальної математики, математичної статистики, методів оптимізації та ін. До складу MATLAB входить кілька десятків спеціалізованих пакетів (тулбоксів -Toolbox) для обробки даних у різних галузях науки і техніки (методи оптимізації, статистика, нейронні мережі, цифрова обробка сигналів і т.п.). Основні можливості системи MATLAB, необхідні для засвоєння даного курсу, наведені в Додатку.

Завантаження системи MATLAB і робота у головному вікні

Завантажити MATLAB: кнопка **Пуск\Програми\MATLAB**.

Обчислення можна проводити прямо в командному вікні. Для цього у головному вікні після символу >> треба набрати відповідні команди. Результат буде збережено у змінній *ans*, що створюється автоматично. При виконанні великої кількості команд (наприклад, при реалізації певного алгоритму) обчислення можуть бути оформлені як певні програми (скрипти) у вигляді М-файлів.

Створення М-файлів

М-файли містять послідовності команд. Фактично М-файл являє собою код програми користувача, який є зручним для редагування і доповнення. Для створення М-файла слід зайти в меню File\New\M-file. В результаті відкривається текстовий редактор MATLAB, призначений для створення і редагування цих файлів. Ім'я файлу не повинне починатися з цифри, містити пропуски.

Використання довідникової системи

MATLAB містить велику кількість вбудованих функцій. Для того, щоб користуватися вбудованою функцією, необхідно знати, які параметри і в якій послідовності слід їй передавати. Довідку щодо вбудованої функції можна отримати набравши в командному вікні команду help і далі ім'я цієї вбудованої функції, наприклад, help sin. Виклик розширеної довідки здійснюється з головного меню MATLAB: Help\Product help (або натиснути клавішу F1). Пошук потрібної функції здійснюється з рядку пошуку по певному ключовому слову.

Робота з векторами і матрицями

Оголошення векторів і матриць може здійснюватися безпосередньо в середовищі MATLAB. Нижче наведено приклади.

Створення вектора:

```
>> V=[1 2 3 4 5];
```

Створення матриці:

```
>> M=[1 2 3 4 5; 10 20 30 40 50; 100 200 300 400 500]
```

Якщо після оголошення вектора (або матриці) поставити крапку з комою (;), то результат V (або M) буде обчислений, але значення змінних не будуть виводитися в командне вікно.

Доступ до окремого елемента вектора здійснюється шляхом визначення індексу в круглих дужках, наприклад: V(3) – третій елемент вектора V.

Доступ до окремого елементу матриці також здійснюється шляхом зазначення індексів, розділених комою, наприклад: M(2,3) – елемент другого рядка і третього стовпчика матриці M.

Доступ до стовпчика матриці здійснюється за допомогою використання символу ":" поряд з індексом, наприклад, M(:, 3) – доступ до третього стовпчика матриці M, M(2,:) – доступ до всього другого рядка матриці M.

Додання стовпчиків до матриці здійснюється за допомогою спеціального символу ";". Наприклад, додання рядка S=[1 2 3 4 5] до матриці M=[1 2 3 4 5; 1 2 3 4 5; 1 2 3 4 5] виконується за допомогою команди:

```
>> M=[M;S]
```

Для додання стовпчика до матриці застосовується кома замість крапки з комою в останньому операторові, наприклад:

```
>> M=[1 2 3 4 5; 1 2 3 4 5; 1 2 3 4 5];
>> S=[1; 2; 3];
>> M=[M,S]
```

Для видалення стовпчиків використовується послідовність символів "[ ]", наприклад для видалення усього третього стовпчика матриці M вводиться команда:

```
>> M(:,3)=[]
```

Видалення усього другого рядка матриці M:

```
>> M(2,:)=[]
```

Транспонування вектора і матриці здійснюється допомогою операції, яка визначається символом "’", наприклад:

```
>> b=[1 2 3 4];
>> s=b’
>> s
1
2
3
4
```

Арифметичні операції з векторами і матрицями

За умовчанням усі дії: +, –, *, /, ^ (піднесення до ступеня) виконуються над матрицями за правилами лінійної алгебри.

Для поелементного множення двох векторів необхідно використовувати оператор точка «.»:

```
>> d=a.*b;
```

У результаті множення вектора на вектор за правилами лінійної алгебри отримуємо число (другий вектор при цьому потрібно транспонувати):

```
>>a=[2 3 4 5];
>>b=[1 2 3 4];
>>c=a*(b')
```

Створення строкової змінної здійснюється з використанням одинарних лапок. Створення масиву строкових змінних здійсюється за допомогою масиву елементів, наприклад:

```
>> D = {'ein','zwei','drei'}
```

Робота з графікою

Основна команда для роботи з двомірною графікою – plot.

Наведемо приклад програми, що реалізує побудову графіка $f(x) = x\sin(x)$ (рис. 1).

Для цього оголосимо вектор-аргумент, наприклад:

```
>> x=0:0.1:20
```

Тут елементи вектора x набувають значень від 0 до 20 з кроком 0.1.

Після цього обчислюються значення цільової функції:

```
>> y=x.*sin(x)
```

Безпосередня побудова графіка:

```
>> plot(x,y)
```

Відображення сітки:

```
>> grid on
```

Заголовок графіка:

```
>> title ('Function sin(x)')
```

Підпис по осі x:

```
>> xlabel('Argument x')
```

Підпис по осі y:

```
>> ylabel('Function y')
```

Розглянемо побудову тривимірних графіків функцій на прикладі функції $f(x, y) = x^2 - y^2 + xy + 2$.

Визначення лівої межі для x:

```
>> Lx=-5;
```

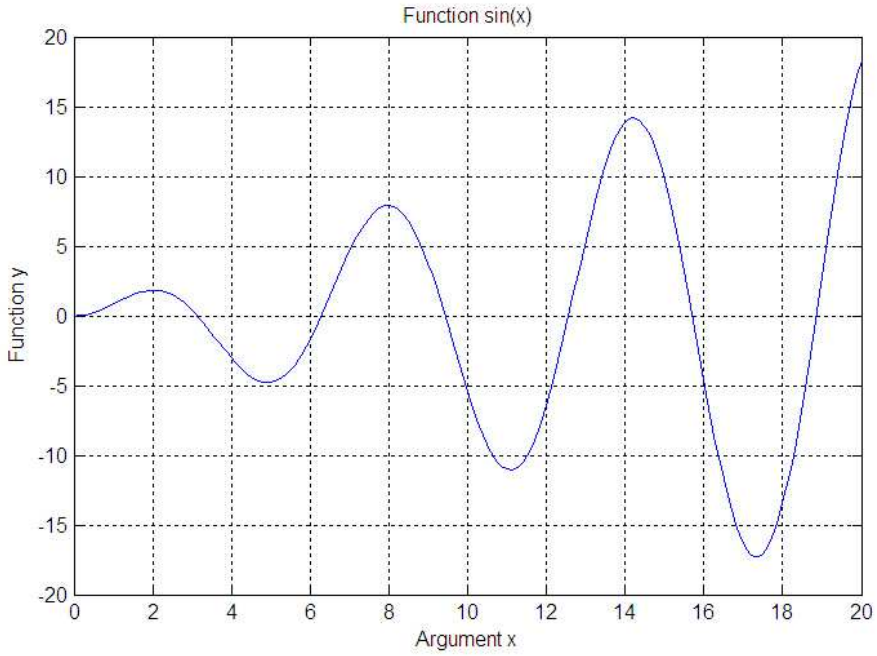


Рис. 1. Графік функції $f(x) = x\sin(x)$

Визначення правої межі для x:

```
>> Rx=5;
```

Визначення кроку по осі x:

```
>> stepx=0.05;
```

Визначення лівої межі для y:

```
>> Ly=-5;
```

Визначення правої межі для y:

```
>> Ry=5;
```

Визначення кроку по осі y:

```
>> stepy=0.05;
```

Дані для тривимірного графіка:

```
>> xs=Lx:stepx:Rx;
>> ys=Ly:stepy:Ry;
```

Створення сітки координат :

```
>>[X,Y] = meshgrid(xs,ys);
```

Обчислення функції:

```
>> Z=X.^2-Y.^2+X.*Y+2;
```

Побудова тривимірного графіка (рис. 2) :

```
>> surf(X,Y,Z)
```

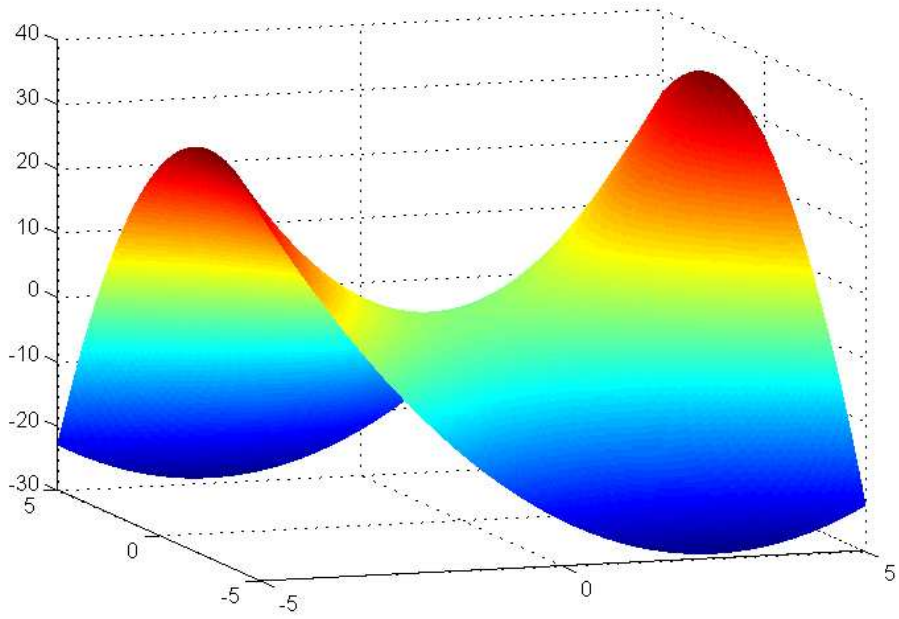


Рис. 2. Графік функції $f(x) = x^2 - y^2 + xy + 2$

Для збереження вікна з графіком у графічному форматі необхідно набрати в командному вікні код:

```
>> print -dtiff -r300 'c:/tmp/Graphic'
```

Тут – dtiff – формат файлу (tiff), – r300 – розподільна здатність (300 dpi), Graphic – ім’я файлу з рисунком, включаючи ім’я каталогу.

Контрольні запитання

1. Яке призначення системи MATLAB?
2. Як створити М-файл?
3. Як створити і заповнити значеннями матрицю розміром $m \times n$?
4. Як встановити діапазон зміни значень змінної-вектора?
5. Які команди MATLAB дозволяють вивести дво- і тривимірні графіки?

2. Задача лінійного програмування (функція linprog)

Необхідно мінімізувати значення виразу $f^T x$, тобто знайти $\min_x f^T x$ при обмеженнях:

$A \cdot x \le b;$

$Aeq \cdot x = beq;$

$lb \le x \le ub,$

де *f, x, b, beq, lb* и *ub* – вектори, а *A* і *Aeq* – матриці.

Синтаксис оператора **linprog**, що реалізує це завдання, наступний:

```
x = linprog(f,A,b,Aeq,beq)
x = linprog(f,A,b,Aeq,beq,lb,ub)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
[x,fval] = linprog(...)
[x,fval,exitflag] = linprog(...)
[x,fval,exitflag,output] = linprog(...)
[x,fval,exitflag,output,lambda] = linprog(...)
```

Оператор linprog вирішує завдання лінійного програмування. Приведемо деякі варіанти його використання:

- x = linprog(f, A, b) знаходить min f'*x за умови, що A*x <= b.
- x = linprog(f, A, b, Aeq, beq, lb, ub) визначає набір нижніх і верхніх меж для змінних x, так що рішення завжди знаходиться в діапазоні lb <= x <= ub. Якщо немає нерівностей, то встановлюється Aeq=[] і beq=[].
- [x, fval, exitflag, output, lambda] = linprog(...) повертає структуру lambda, поля якоївключають множники Лагранжа як рішення від x.

Розглянемо приклад: знайти таке x, що мінімізує функцію $f(x) = -5x_1 - 4x_2 - 6x_3$, при умові, що

$x_1 - x_2 + x_3 \le 20;$

$3x_1 + 2x_2 + 4x_3 \le 42;$

$3x_1 + 2x_2 \le 30;$

$0 \le x_1, 0 \le x_2, 0 \le x_3.$

Виконання в середовищі MATLAB :

```
>> f = [-5; -4; -6]
A =  [1 -1  1;  3  2  4; 3  2  0];
b = [20; 42; 30];
```

```
lb = zeros(3,1);
>>  [x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb);
```

Звідсі:

```
>> x
x =
    0.0000
    15.0000
    3.0000
```

Для графічного представлення приведемо рішення задачі $f(x) = -2x_1 + 7x_2$ при умові, що

$$2x_1 + x_2 \le 13;$$

$$-x_1 + 3x_2 \le 11;$$

$$-x_1 \le -1;$$

$$-2x_1 - 5x_2 \le -17.$$

$$0 \le x_1, 0 \le x_2.$$

У середовищі MATLAB вводимо команди:

```
>> f=[-2; 7]
f =
    -2
     7
>> A=[2 1; -1 3; -1 0; -2 -5]
A =
     2     1
    -1     3
    -1     0
    -2    -5
>> b=[13; 11; -1; -17]
b =
    13
    11
    -1
   -17
>> lb=zeros(2,1)
lb =
     0
     0
>> [x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb);
Optimization terminated.
>> x
x =
    6.0000
    1.0000
```

Отримуємо графічне рішення задачі у середовищі MATLAB (рис. 3):

```
>> x=0:0.5:8
>> y=13-2*x
>> plot(x,y)
```

```
>> hold on
>> y=(x+11)/3
>> plot(x,y)
>> y=(17-2*x)/5
>> plot(x,y)
>> y=2*x/7
>> plot(x,y,'--'))
>> y=0:0.005:12
>> x=1
>> plot(x,y)
```

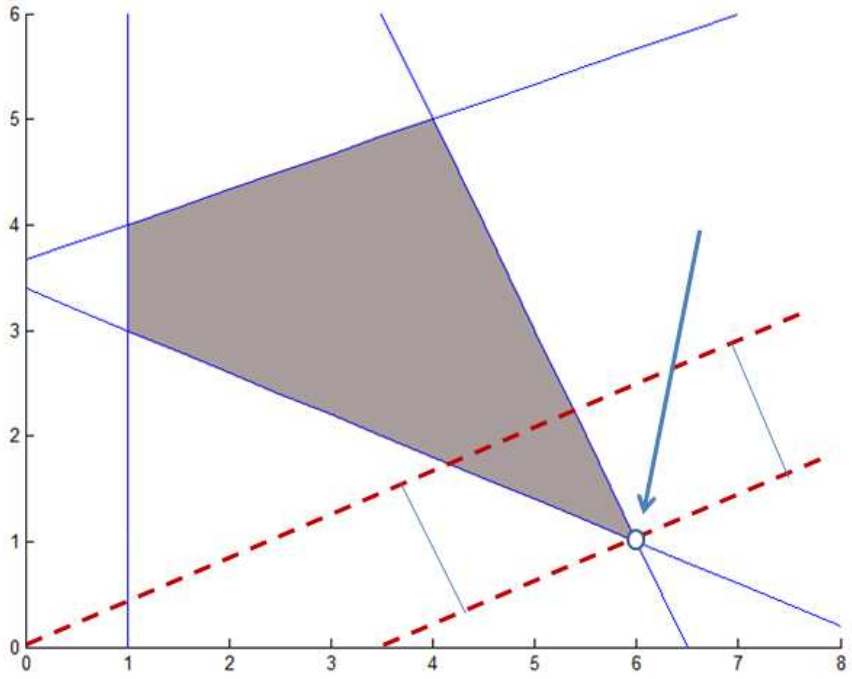


Рис. 3. Графічне рішення задачі лінійного програмування

Контрольні запитання

1. Як ознайомитись з описом можливостей пакету Optimization Toolbox системи MATLAB?

2. Яке призначення функції linprog?

3. Назвіть вхідні і вихідні параметри функції linprog?

4. Якою командою забезпечується накладання на одному екрані декількох графіків?

5. Які команди MATLAB дозволяють вивести дво- і тривимірні графіки?

3. Квадратична задача математичного програмування (функція quadprog)

Необхідно мінімізувати значення виразу $\frac{1}{2}x^T Hx + f^T x$, тобто знайти $\min_x \frac{1}{2}x^T Hx + f^T x$ при обмеженнях:

$$A \cdot x \le b;$$
$$Aeq \cdot x = beq;$$
$$lb \le x \le ub,$$

де *f, x, b, beq, lb* и *ub* – вектори, а *H, A* і *Aeq* – матриці.

Елементи синтаксису функції quadprog:

```
x = quadprog(H,f,A,b,Aeq,beq,lb,ub)
[x,fval,exitflag,output,lambda]=quadprog(H,f,A,b,Aeq,beq,lb,ub)
```

- x = quadprog(H, f, A, b) повертає вектор x, який мінімізує функцію 1/2*x'*H*x + f'*x при умові A*x <= b;
- x = quadprog(H,f,A,b,Aeq,beq,lb,ub) вирішує вказану вище задачу з додатковим виконанням обмежень типу рівності Aeq*x = beq, так що б рішення знаходилося в діапазоні lb <= x <= ub.

Вихідний аргументи exitflag описує вихідні умови:

- > 0 – ця функція сходиться до рішення по x;
- = 0 – перевищене максимальне число оцінок функції або ітерацій;
- < 0 – функція не сходиться до будь-якого рішення.

Як приклад розглянемо завдання оптимізації: знайти значення x, які мінімізують функцію

$$f(x) = \frac{1}{2}x_1^2 + x_2^2 - x_1 x_2 - 2x_1 - 6x_2$$

при умові, що

$$x_1 + x_2 \le 2;$$
$$-x_1 + 2x_2 \le 2;$$
$$2x_1 + x_2 \le 3;$$
$$0 \le x_1, 0 \le x_2.$$

Відзначимо, що це завдання в матичній формі запису має наступний вигляд:

$$f(x)=\frac{1}{2}x^T Hx+f^T x$$

де, зрозуміло,

$$f=\begin{bmatrix}-2\\-6\end{bmatrix},\quad x=\begin{bmatrix}x_1\\x_2\end{bmatrix}.$$

Значення матриці H обчислюється наступним чином:

$$\frac{1}{2}x_1^2+x_2^2-x_1x_2=\frac{1}{2}\begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}a & b\\c & d\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}=$$

$$=\frac{1}{2}\begin{bmatrix}x_1a+x_2c & x_1b+x_2d\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\frac{1}{2}\left(x_1^2a+x_2x_1c+x_1x_2b+x_2^2d\right).$$

Звідси отримуємо:

$$a=1;\quad d=2;\quad b+c=-2.$$

Вибираючи значення $b=-1,\ \ c=-1$, отримуємо матрицю:

$$H=\begin{bmatrix}1 & -1\\-1 & 2\end{bmatrix}.$$

Введемо ці коефіцієнти матриць в середовищі MATLAB:

```
>> H=[1 -1; -1 2]
H =
     1    -1
    -1     2
>> f=[-2 -6]
f =
    -2    -6
>> a=[1 1; -1 2; 2 1]
a =
     1     1
    -1     2
     2     1
>> b=[2 2 3]
b =
     2     2     3
>> lb=zeros(2,1)
lb =
     0
     0
>> [x,fval, exitflag, output, lambda]=quadprog(H,f,a,b,[],[],lb)
Optimization terminated.
>> x
x =
    0.6667
    1.3333
```

Для графічної ілюстрації побудуємо поверхню:

```
>>[X,Y] = meshgrid(0:0.05:2,0:0.05:3)
>>Z=0.5*X.^2+Y.^2-X.*Y-2*X-6*Y
>>surf(X,Y,Z)
```

Побудова площин, відповідних граничним умовам (рис. 4) виконується за допомогою введення команд:

```
>> hold on
>> [X,Z] = meshgrid(0:0.05:2,-15:0.1:5)
>> Y=3-2*X
>> surf(X,Y,Z)
>> Y=(2+X)/2
>> surf(X,Y,Z)
>> Y=2-X
>> surf(X,Y,Z)
```

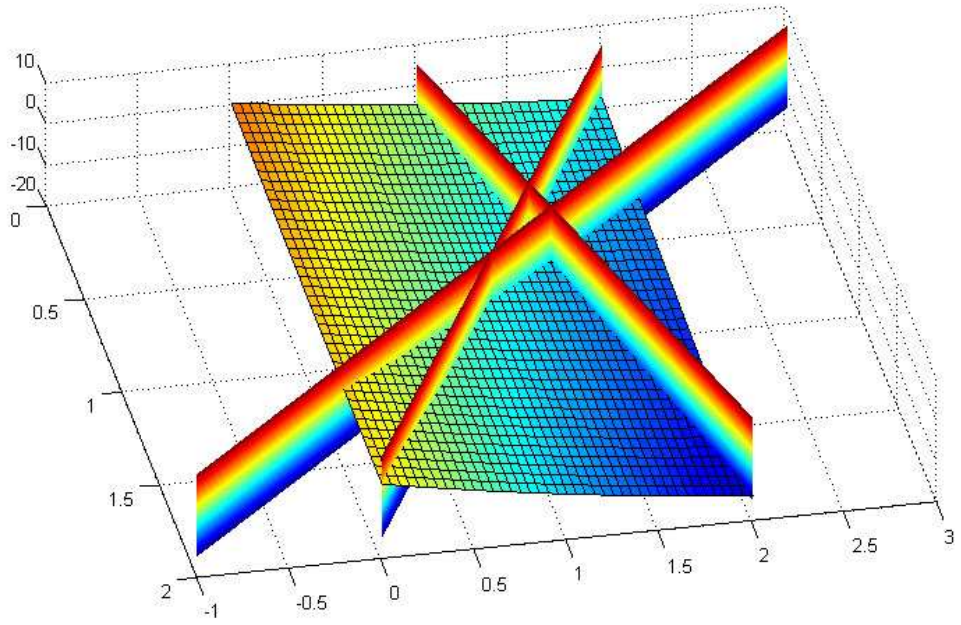


Рис. 4. Перетин поверхні з площинами, що відповідають граничним умовам

Контрольні запитання

1. Яке призначення функції quadprog?

2. Назвіть вхідні і вихідні параметри функції quadprog?

3. Як привести квадратичну функцію, екстремум якої необхідно визначити, до канонічного вигляду, придатного для застосування quadprog?

4. Якою командою визначається двохвимірна решітка даних для подальшого обрахунку поверхні? Який синтаксис цієї команди?

5. Які інтерактивні можливості перегляду і оформлення трьохвимірних графіків закладені у системі MATLAB?

4. Метод найменших квадратів (функція lsqlin)

Перевизначеною системою називається система рівнянь, в якій кількість рівнянь більше кількості невідомих.

Основне призначення функції **lsqlin**: знайти рішення методом найменших квадратів системи $C \cdot x = d$ (у тому числі і для перевизначеної) з лінійними обмеженнями: $A \cdot x \le b$ и $lb \le x \le ub$.

Знаходження мінімуму виразу $\frac{1}{2}\|Cx - d\|^2$, тобто $\min_x \frac{1}{2}\|Cx - d\|^2$ при обмеженнях:

$A \cdot x \le b$;

$Aeq \cdot x = beq$;

$lb \le x \le ub$,

де *C*, *A* і *Aeq* – матриці *d*, *b*, *beq*, *lb*, *ub* і *x* – вектори.

Приклад 1: Необхідно знайти мінімум виразу $x_1^2 + x_2^2 + x_3^2$ при обмеженнях:

$x_1 + 2x_2 + 4x_3 \le 7$;

$x_1 \ge 0, x_2 \ge 0, x_3 \ge 0.$

Рішення:

```
>> Aeq = [1 2 4];
beq = [7];
>> C=eye(3);
>> d=zeros(3,1);
>> [x, fval] =lsqlin(C, d, [], [], Aeq, beq)
x =
    0.3333
    0.6667
    1.3333
fval =
    2.3333
```

Для графічної ілюстрації завдання розглянемо більш простий випадок з двома змінними. Необхідно знайти мінімум виразу $x_1^2 + x_2^2$ при обмеженнях:

$x_1 + 2x_2 = 5$;

$x_1 \ge 0, x_2 \ge 0.$

Рішення за допомогою функції **lsqlin**:

```
>> Aeq=[1 2]
>> beq=[5]
>> C=eye(2)
>> d=zeros(2,1)
>> [x,fval]=lsqlin(C,d,[],[],Aeq,beq)
```

```
x =
    1.0000
    2.0000
```

Графічна ілюстрація завдання (рис. 5) реалізується шляхом введення команд:

```
>>[X,Y]=meshgrid(-3:0.05:3,-3:0.05:3)
>> Z=X.^2+Y.^2
>> surf(X,Y,Z)
>> X=-3:0.05:3
>> hold on
>>  [X,Z] = meshgrid(-3:0.05:3,-10:0.1:20)
>> Y=(5-X)/2
>> surf(X,Y,Z)
```

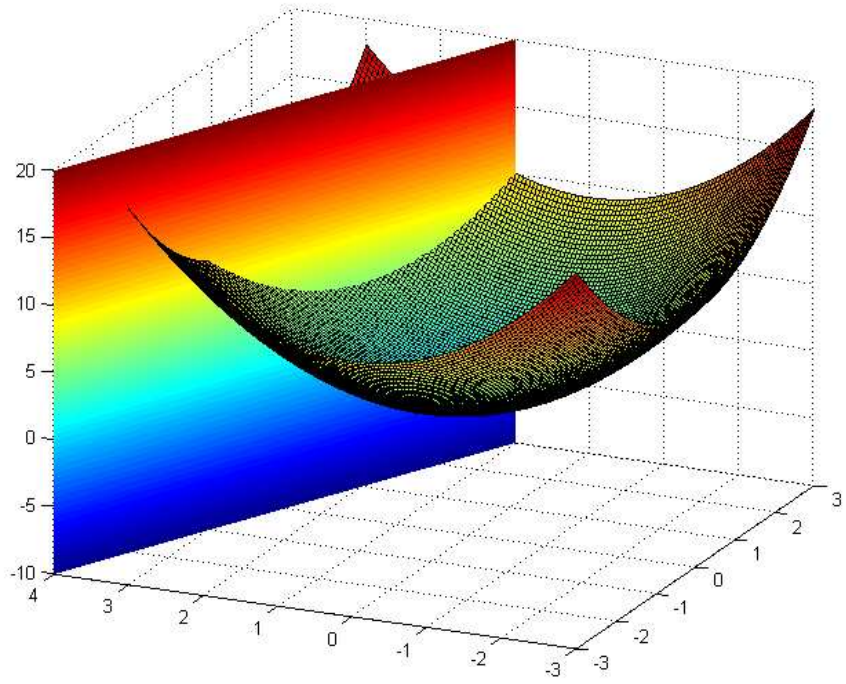


Рис. 5. Графічна ілюстрація завдання

Приклад 2: Необхідно знайти рішення для перевизначеної системи $C \cdot x = d$ при умові, що $A \cdot x \le b$ і $lb \le x \le ub$.

Перевизначена система:

$$0.9501x_1 + 0.7620x_2 + 0.6153x_3 = 0.0578$$

$$0.2311x_1 + 0.4564x_2 + 0.7919x_3 = 0.3528$$

$$0.6068x_1 + 0.0185x_2 + 0.9218x_3 = 0.8131$$

$$0.8912x_1 + 0.4447x_2 + 0.1762x_3 = 0.1388$$

Обмеження:

$$0.2027x_1 + 0.2721x_2 + 0.7467x_3 \le 0.5251;$$

$$0.1987x_1 + 0.1988x_2 + 0.4450x_3 \le 0.2026;$$

$$0.6037x_1 + 0.0152x_2 + 0.9318x_3 \le 0.6721;$$

$$-1 \le x_1 \le 2; \quad -1 \le x_2 \le 2; \quad -1 \le x_3 \le 2.$$

Коефіцієнти матриць, а також верхні і нижні обмеження визначаються таким чином:

```
>> C = [0.9501  0.7620 0.6153 ; 0.2311 0.4564  0.7919 ;
        0.6068  0.0185 0.9218 ; 0.8912  0.4447  0.1762];
>> d = [0.0578;    0.3528;    0.8131;    0.1388];
>> A = [0.2027 0.2721 0.7467; 0.1987 0.1988 0.4450;
        0.6037 0.0152 0.9318];
>> b = [0.5251; 0.2026; 0.6721];
>> lb = -ones(3,1);
>> ub = 2*ones(3,1);
```

Звернемося до програми методу найменших квадратів з лінійними обмеженнями:

```
[x,resnorm,residual,exitflag,output,lambda] = lsqlin(C,d,A,b,[
],[ ],lb,ub);
```

Увівши x і lambda.ineqlin отримаємо:

```
>> x
x =
    0.1958
   -0.5279
    0.6030
>> lambda.ineqlin
ans =
         0
         0
    0.1603
```

Ненульові елементи векторів в полях lambda вказують на активні обмеження при рішенні. Третє обмеження у вигляді нерівностей (у lambda.ineqlin) є активним (тобто рішення знаходиться на обмежувальних умовах).

Контрольні запитання

1. Яке призначення функції lsqlin?

2. Назвіть вхідні і вихідні параметри функції lsqlin?

3. Для чого застосовується метод найменших квадратів?

4. Як за допомогою функції lsqlin знайти рішення системи лінійних рівнянь при наявності обмежень?

5. Для чого застосовується функція eye у системі MATLAB?

5. Задача нелінійного програмування (метод множників Лагранжа)

Приклад 1. Нехай задано цільову функцію $F(x) = x_1 x_2$. Необхідно визначити набір $\{x_1, x_2\}$, при якому функція $F(x)$ досягає максимума за умови $2x_1 + x_2 = 8$.

Для рішення формуємо фіункцію Лагранжа:

$$\Phi(x, \Lambda) = x_1 x_2 - \lambda(2x_1 + x_2 - 8).$$

Беремо часткові похідні від $\Phi(x, \Lambda)$ по x_1, x_2, λ. Отримуємо систему рівнянь:

$$\begin{cases} \dfrac{\partial \Phi(x,\lambda)}{\partial x_1} = x_2 - 2\lambda = 0; \\ \dfrac{\partial \Phi(x,\lambda)}{\partial x_2} = x_1 - \lambda = 0; \\ \dfrac{\partial \Phi(x,\lambda)}{\partial \lambda} = 2x_1 + x_2 - 8 = 0. \end{cases}$$

Виражаючи x_1 і x_2 з перших двох рівнянь через λ отримуємо:

$$x_1 = \lambda; \quad x_2 = 2\lambda,$$

тобто

$$2x_1 + x_2 = 4\lambda = 8.$$

Отже:

$$x_1 = 2; \quad x_2 = 4; \quad F(x) = x_1 x_2 = 8.$$

Надамо графічну ілюстрацію цієї задачи у середовищі MATLAB (рис. 6)

Побудова поверхні $F(x) = x_1 x_2$:

```
>> [X,Y]=meshgrid(-8:0.1:8,-8:0.1:8)
>> Z=X.*Y
>> surf(X,Y,Z)
```

Побудова полощини – обмеження:

```
>> hold on
>> [X,Z]=meshgrid(-8:0.1:8,-60:0.1:60)
>> Y=8.-2*X
>> surf(X,Y,Z)
```

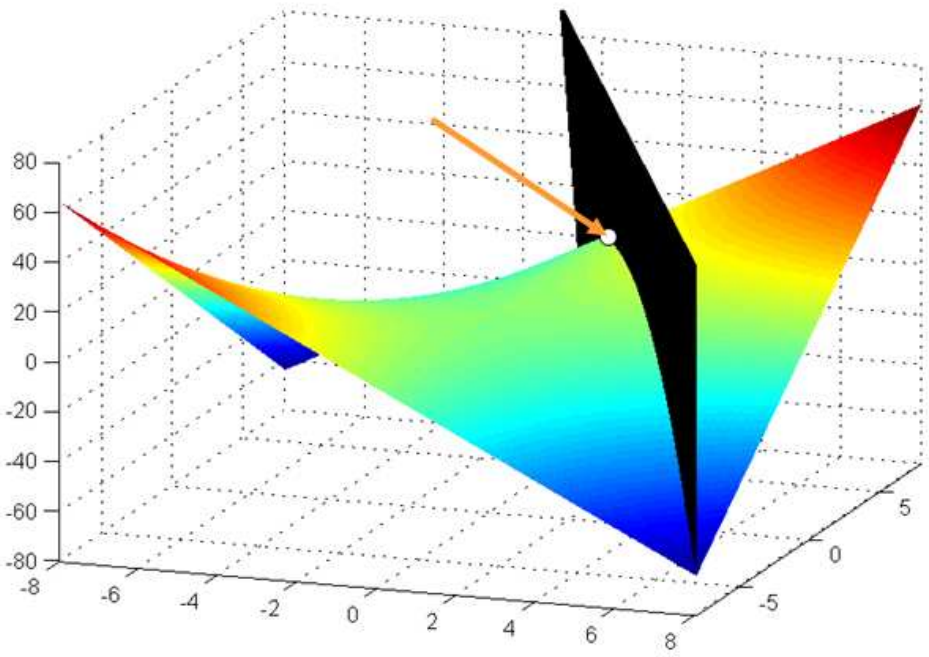


Рис. 6. Графічна ілюстрація рішення прикладу 1

Приклад 2. Нехай задано цільову функцію $F(x) = x_1 + x_2$. Необхідно визначити набір $\{x_1, x_2\}$, при якому функція $F(x)$ досягає максимуму за умови $x_1^2 + x_2^2 = 1$.

Рішення. Формуємо фіункцію Лагранжа:

$$\Phi(x, \Lambda) = x_1 + x_2 - \lambda\left(x_1^2 + x_2^2 - 1\right).$$

Беремо часткові похідні від $\Phi(x, \Lambda)$ по x_1, x_2, λ. Отримуємо систему рівнянь:

$$\begin{cases} \dfrac{\partial\Phi(x,\lambda)}{\partial x_1} = 1 - 2\lambda x_1 = 0; \\ \dfrac{\partial\Phi(x,\lambda)}{\partial x_2} = 1 - 2\lambda x_2 = 0; \\ \dfrac{\partial\Phi(x,\lambda)}{\partial \lambda} = -x_1^2 - x_2^2 + 1 = 0. \end{cases}$$

З першого і другого рівняння отримуємо:

$x_1 = x_2 = 1/2\lambda$, тобто $2x_1^2 = 2x_2^2 = 1$. Звідсі: $x_1 = x_2 = \sqrt{0,5}$. $F(x) = \sqrt{2}$.

Надамо графічну ілюстрацію цієї задачи у середовищі MATLAB (рис. 7).

Побудова поверхні обмеження $x_1^2 + x_2^2 = 1$:

```
>> [X,Z]=meshgrid(-1:0.05:1,-4:0.1:4)
```

```
>> Y=sqrt(1.-X.^2)
>> surf(X,Y,Z)
>> hold on
>> Y=-sqrt(1.-X.^2)
>> surf(X,Y,Z)
```

Побудова цільової поверхні:

```
>> [X,Y]=meshgrid(-1.5:0.1:1.5,-1.5:0.1:1.5)
>> Z=X+Y
>> surf(X,Y,Z)
```

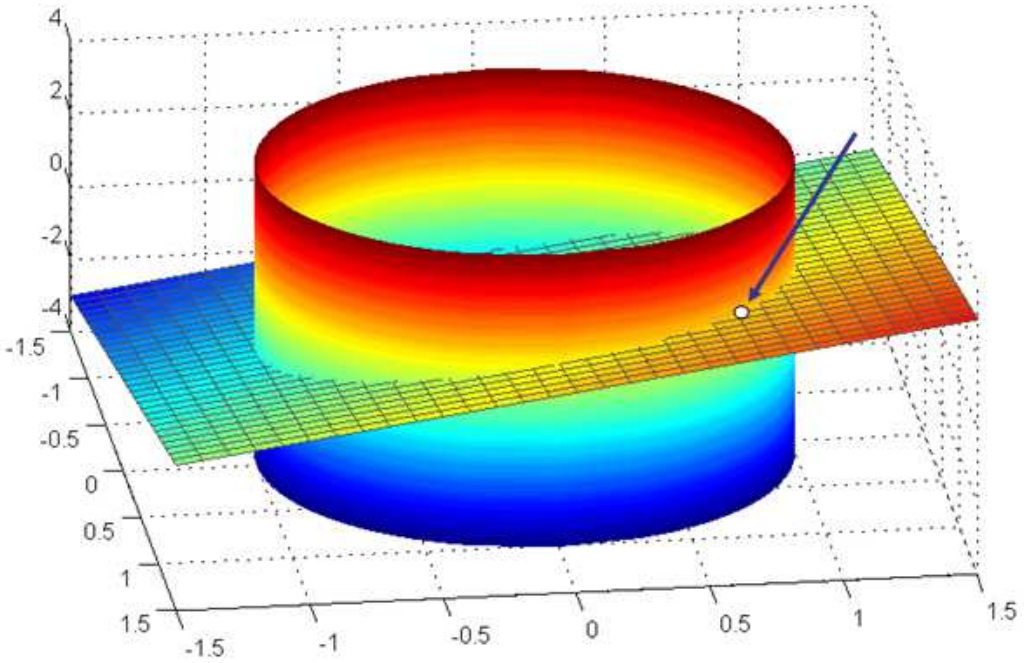


Рис. 7. Графічна ілюстрація рішення прикладу 2

Контрольні запитання

1. Яка основна ідея методу множників Лагранжа?

2. Які вимоги ставляться перед цільовою функцією і рівняннями обмежень для застосування методу множників Лагранжа?

3. Мінімізувати функцію $f(x) = x_1^2 + x_2^2$ при обмеженні $g(x) = 2x_1 + x_2 - 2 = 0$.

4. Для чого у другому прикладі функція sqrt застосовується два рази – один раз із знаком «+», а один раз – із знаком «–» ?

6. Задача цілочисельного програмування (функція bintprog)

Рішення задачі цілочисельного програмування, тобто $\underset{x}{\text{minimize}}\, f(x)$ при умові, що $A \cdot x \le b$ і $Aeq \cdot x = beq$, де *f, b* і *beq* є векторами, *A* і *Aeq* – матриці, а *x* – цілочисельний вектор рішення вектор рішення, тобто його компоненти повинні приймати значення 0 або 1.

Елементи синтаксису оператора **bintprog**:

x = bintprog(f)
x = bintprog(f, A, b)
x = bintprog(f, A, b, Aeq, beq)
[x, fval] = bintprog(...)
[x,fval, exitflag] = bintprog(...)
[x, fval, exitflag, output] = bintprog(...)

Особливості:

- x = bintprog(f) вирішує задачу цілочисельного програмування $\underset{x}{\text{minimize}}\, f' \cdot x$;
- x = bintprog(f, A, b) – задачу цілочисельного програмування $\underset{x}{\text{minimize}}\, f' \cdot x$ при умові, що $A \cdot x \le b$;
- x = bintprog(f, A, b, Aeq, beq) вирішує попередню задачу за додаткових умов типу рівності $Aeq \cdot x \le beq$.

Значення вихідних параметрів **exitflag** і **output** наведені у таблиці.

exitflag	Деяке ціле число, що відповідає причині зупинки алгоритму.	
	1	Функція зійшлася до рішення x
	0	Число ітерацій перевищило значення options.MaxIter
	-2	Це завдання не має рішення
	-4	Число перебраних вузлів перевищує значення options.MaxNodes
	-5	Час перебору перевищує значення options.MaxTime
	-6	Число ітерацій для деякого вузла при рішенні задачі перевищило значення options.MaxRLP
output	Структура з інформацією щодо результатів оптимізації.	

	iterations	Число виконаних ітерацій
	nodes	Число вузлів перебору
	time	Перевищення часу роботи алгоритму
	algorithm	Алгоритм, що використовувався
	message	Причина зупинки роботи алгоритму

У функції **bintprog** для вирішення завдання цілочисельного програмування використовується алгоритм лінійного програмування на основі методу гілок і меж. У цьому алгоритмі проводиться перебір оптимальних рішень задачі цілочисельного програмування. Алгоритм включає:

- перебір цілочисельних допустимих рішень;
- коригування найкращої цілочисельної допустимої точки по мірі просування по дереву пошуку;
- перевірку неможливості досягнення оптимальніших цілочисельних рішень.

Приклад 1. Необхідно мінімізувати функцію:

$$f(x) = -9x_1 - 5x_2 - 6x_3 - 4x_4$$

при наявності обмежень:

$$6x_1 + 3x_2 + 5x_3 + 2x_4 \le 9;$$

$$x_3 + x_4 \le 1;$$

$$-x_1 + x_3 \le 0;$$

$$-x_2 + x_4 \le 0,$$

де x_1, x_2, x_3 і x_4 є бінарними цілими. Запишемо обмеження у матричній формі:

$$\begin{bmatrix} 6 & 3 & 5 & 2 \\ 0 & 0 & 1 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \le \begin{bmatrix} 9 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

Виконаємо наступні команди:

```
>>f = [-9; -5; -6; -4];
>>A = [6 3 5 2; 0 0 1 1; -1 0 1 0; 0 -1 0 1];
>>b = [9; 1; 0; 0];
>> [x,fval, exitflag]  = bintprog(f,A,b);
Optimization terminated.
>> x'
ans =
```

```
     1     1     0     0
>> exitflag
exitflag =
     1
```

Для графічної ілюстрації розглянемо найпростіше завдання з двома змінними, а саме - мінімізацію виразу:

$$f(x) = -9x_1 - 5x_2$$

з обмеженнями:

$$6x_1 + 3x_2 \le 9;$$

$$-x_1 + x_2 \le 1.$$

У середовищі MATLAB вводимо команди:

```
>>f = [-9; -5];
>>A = [6 3 ; -1 1];
>>b = [9; 1];
>>f
f =
    -9
    -5
>> A
A =
     6     3
    -1     1
>> b
b =
     9
     1
>> [x,fval, exitflag]  = bintprog(f,A,b);
Optimization terminated.
>> x
x =
     1
     1
```

Графічне представлення завдання (рис. 8):

```
>>  [X,Y]=meshgrid(-3:1:3,-3:1:3)
>>  hold on
>> Z=-9*X-5*Y
>> surf(X,Y,Z)
>> [X,Z]=meshgrid(-3:1:3,-50:1:50)
>> Y=(9-6.*X)/3
>> surf(X,Y,Z)
>> Y=1+X
>> surf(X,Y,Z)
```

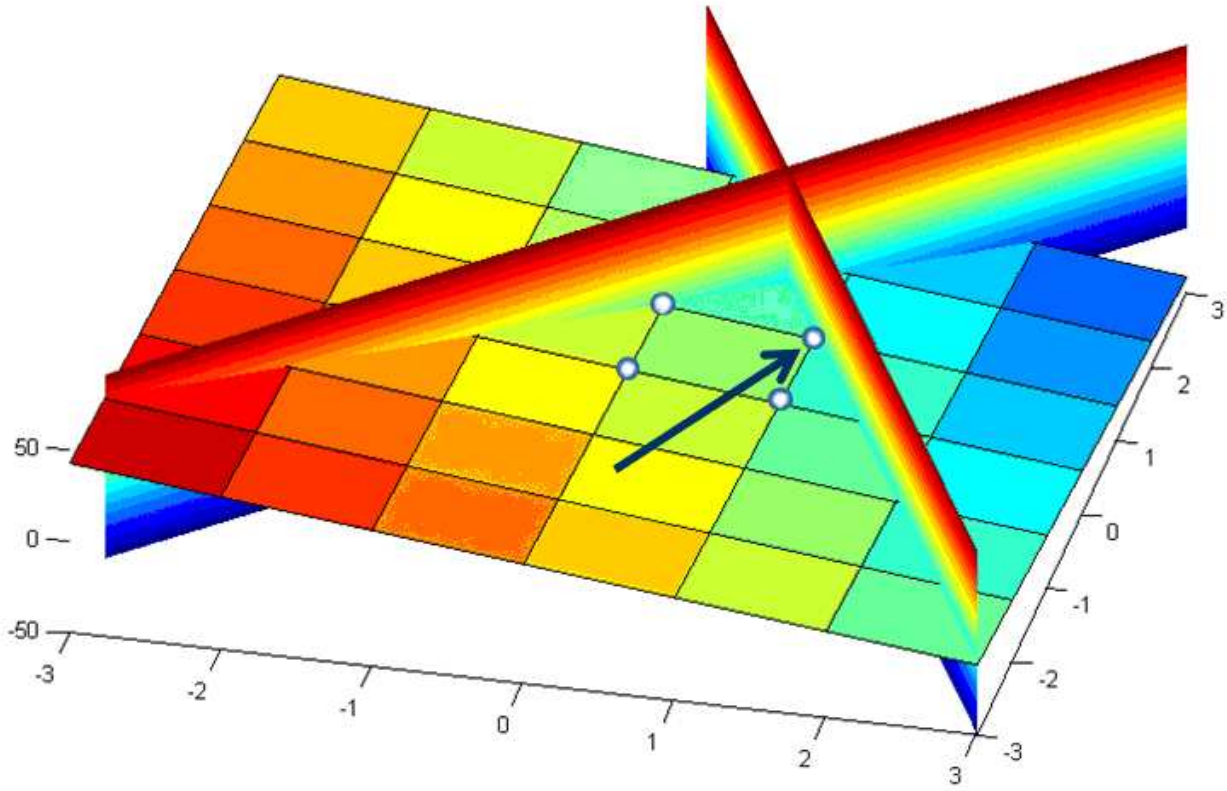


Рис. 8. Графічна ілюстрація спрощеного завдання цілочисельного програмування

Приклад 2. Варіант класичного завдання про рюкзак. Нехай задана деяка сума грошей, наприклад 43 коп. Треба видати її українськими монетами різних номіналів, якщо це можливо, а інакше необхідно видати різними монетами найбільшу можливу суму, яка є меншою за потрібну.

Необхідно мінімізувати функцію:

$$f(x) = 43 - 1x_1 - 2x_2 - 5x_3 - 10x_4 - 25x_5 - 50x_6$$

за наявності обмеження:

$$x_1 + 2x_2 + 5x_3 + 10x_4 + 25x_5 + 50x_6 \le 43$$

де x_1, x_2, x_3, x_4, x_5 і x_6 є бінарними цілими.

Очевидно, досить мінімізувати функцію

$$g(x) = -1x_1 - 2x_2 - 5x_3 - 10x_4 - 25x_5 - 50x_6$$

за тих самих умов.

Виконаємо наступні команди:

```
>> f = [-1; -2; -5; -10; -25; -50];
>> A = [1 2 5 10 25 50];
>> b=[43]
>> [x,fval, exitflag]  = bintprog(f,A,b);
Optimization terminated.
>> x'
ans =
     1     1     1     1     1     0
```

Перевірка: дійсно, 1+2+5+10+25=43.

Приклад 3. Задача про розбиття множини або «задача про каміння». Є купа каміння різної ваги. Необхідно сформувати з цих каменів дві нові купи так, щоб вага цих куп відрізнялася одна від одної якнайменше. Ця задача відноситься до так званних NP-повних задач.

Введемо вектор ваги каміння – *w*. Тоді вага купи буде дорівнювати *sum*(*w*). Компоненти вектора рішення будемо трактувати таким чином: x(i)=1 – камень відноситься до першої купи, x(i)=0 – до другої. Переформулюючи задачу маємо, що необхідно максимізувати функцію:

$$\sum w(i)x(i) \to \max$$

при

$$\sum w(i)x(i) \le 0.5sum.$$

Підготуємо вектор ваги: $w = [1\ \ 9\ \ 5\ \ 9\ \ 11]$.

Введемо наступні команди:

```
>> w = [1 9 5 9 11];
>> [x,fval, exitflag,output]  = bintprog(-w,w,0.5*sum(w))
```

Отримуємо результат:

```
x =
     1
     0
     1
     0
     1
fval =
   -17
exitflag =
     1
output =
          iterations: 19
```

Тобто у першу купу треба покласти три камені з номерами 1, 3 і 5, а у другу – камені 2 і 4. При цьому вага першої купи буде дорівнювати 17, другої – 18.

Контрольні запитання

1. Яке призначення функції bintprog?

2. Які значення приймають аргументи цільової функції задачі оптимізації для застосування bintprog?

3. Назвіть вхідні і вихідні параметри функції bintprog?

4. Що означає значення вихідного параметру exitflag = 1?

5. Перевірити, чи можливо рішення задачі, наведеної у прикладі 2 для чисел 45, 53, 62.

7. Задача нелінійного програмування (функція fmincon)

Функція **fmincon** призначена для розв'язування задачі нелінійного програмування вигляду:

$f(x) \to \min;$

$c(x) \le 0;$

$ceq(x) = 0;$

$A \cdot x \le b;$

$Aeq \cdot x = beq;$

$lb \le x \le ub,$

де $c(x)$, $ceq(x)$ – вектор-функції, компоненти яких, як і функція $f(x)$, можуть бути нелінійними, x, b, beq, lb, ub – вектори-стовпчики, A, Aeq – прямокутні матриці, і яка має такий синтаксис (наведемо окремі випадки):

```
x = fmincon(fun,x0,A,b);
x = fmincon(fun,x0,A,b,Aeq,beq);
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub);
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon);
[x,fval] = fmincon(...);
[x,fval,exitflag] = fmincon(...);
[x,fval,exitflag,output] = fmincon(...);
[x,fval,exitflag,output,lambda] = fmincon(...).
```

Приклад 1. За допомогою функції fmincon вирішити завдання нелінійного програмування:

$f(x) = x_1^3 - 6x_1^2 + 11x_1 + x_2 \to \min$

при обмеженнях:

$x_1^2 - x_2^2 \le 0;$

$x_1^2 + x_2^2 \ge 4;$

$x_1 \ge 0;$

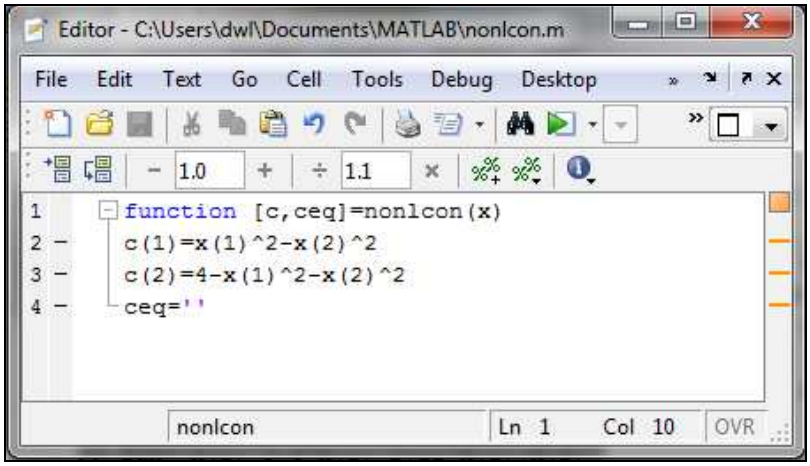
$x_2 \ge 0.$

Як початкові наближення покладемо:

a) $x^{(0)} = (0;1);$

b) $x^{(0)} = (8,4).$

Для того, щоб вирішити поставлену задачу, необхідно створити скрипт MATLAB – М-файл, наприклад, з ім'ям nonlcon, який описує систему функціональних обмежень:



Після цього необхідно ввести команди:

```
>> fun='x(1)^3-6*x(1)^2+11*x(1)+x(2)'
fun =
x(1)^3-6*x(1)^2+11*x(1)+x(2)
>>
[x,fval,exitflag,output]=fmincon(fun,[0;1],[],[],[],[],[0;0],[ ;
],@nonlcon)
x =
    0
    2.5000
fval =
    2.5000
exitflag =
    1
output =
iterations: 2
funcCount: 6
algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
>> [x,fval,exitflag,output]=
fmincon(fun,[8;4],[],[],[],[],[0;0],[ ; ],@nonlcon)
x =
     8
    10
fval =
   226
exitflag =
     1
output =
iterations: 2
```

Як бачимо, отримані результати для різних початкових наближень істотно різняться, тобто локальні екстремуми можуть не співпадати з глобальним екстремумом.

Для наочності розглянемо найпростіший приклад, одновимірну задачу.

Приклад 2. За допомогою функції fmincon вирішити завдання нелінійного програмування:

$$f(x) = -x \cdot \sin(x) \to \min$$

з обмеженнями:

$$x^2 - 9 \le 0;$$

$$-8 \le x \le 8.$$

Як початкові наближення покладемо:

$a)$ $x^{(0)} = 1;$

$b)$ $x^{(0)} = -1.$

Для того, щоб вирішити поставлене завдання, створюємо M -файл з ім'ям mfun, який описує обмеження:

```
1   function [c,ceq] = mfun(x)
2 -   c(1)=-x(1)^2-9;
3 -   ceq='';
4 -   end
```

Визначаємо цільову функцію:

```
>> fun='-x*sin(x)'
fun =
-x*sin(x)
```

Виклик функції fmincon в стартовій точці $x^{(0)} = 1$:

```
>>[x,fval,exitflag,output]=fmincon(fun,1,[],[],[],[],-8,8,@mfun)
x =
   2.0287
fval =
   -1.8197
exitflag =
     5
output =
       iterations: 5
```

Значення параметра exitflag, що більше за нуль, свідчить про те, що алгоритм рішення сходиться. Точка оптимуму – 2.0287.

Виклик функції fmincon в іншій стартовій точці $x^{(0)} = -1$:

```
>> [x,fval,exitflag,output]=fmincon(fun,-1,[],[],[],[],-8,8,
@mfun)
x =
   -2.0287
fval =
   -1.8197
exitflag =
     5
output =
```

```
    iterations: 5
```

Як і в минулому випадку, алгоритм рішення сходиться. Точка локального мінімуму в цьому випадку інша, симетрична першому рішенню відносно осі ординат: –2.0287.

І в даному випадку отримані результати для різних початкових наближень різні, тобто отримані локальні екстремуми не співпадають. Для випадку однієї змінної крива цільової функції і лінії обмежень є наочною конфігурацією, для графічного відображення якої (рис. 9) вводяться наступні команди у вікні Command Window системи MATLAB:

```
>> x=-8:0.05:8;
>> y=-x.*sin(x);
>> plot(x,y);
```

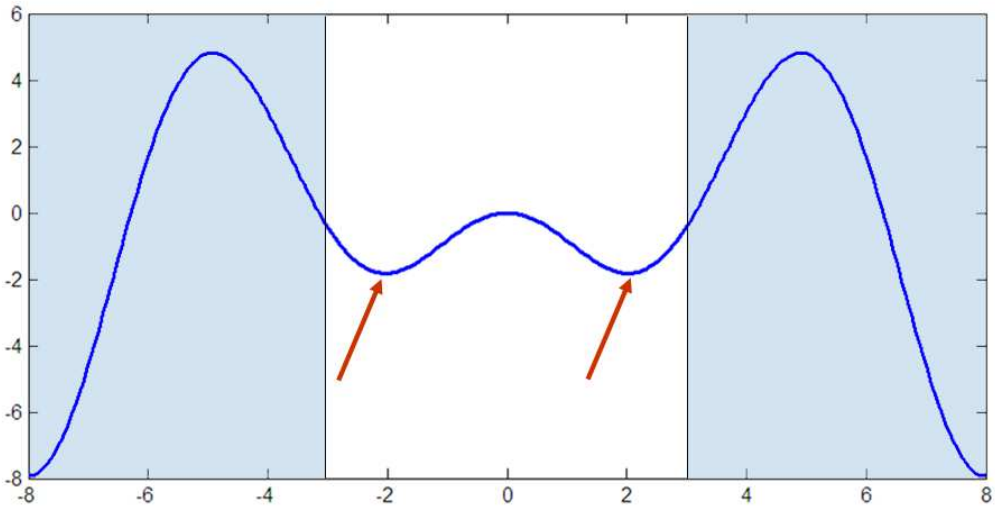


Рис. 9. Графічна ілюстрація цільової функції –x·sin(x) і ліній обмежень. Стрілками вказані локальні мінімуми

Контрольні запитання

1. Яке призначення функції fmincon?

2. Назвіть вхідні і вихідні параметри функції fmincon?

3. За допомогою чого при застосуванні функції fmincon описується система функціональних обмежень?

4. Яким вихідним параметром визначається кількість ітерацій при застосуванні функції fmincon для знаходження локального екстремуму?

8. Графічний інтерфейс користувача (Optimization Tool)

Графічний інтерфейс користувача тулбоксу оптимізації в системі MATLAB – Optimization Tool – викликається із вікна Command Window командою:

```
>> optimtool
```

Вирішення задач, що розглядалися на попередніх заняттях

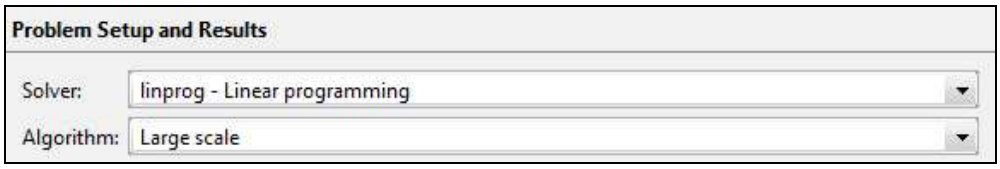
Приклад 1. Знаходження мінімуму виразу:

$$f(x) = -5x_1 - 4x_2 - 6x_3$$

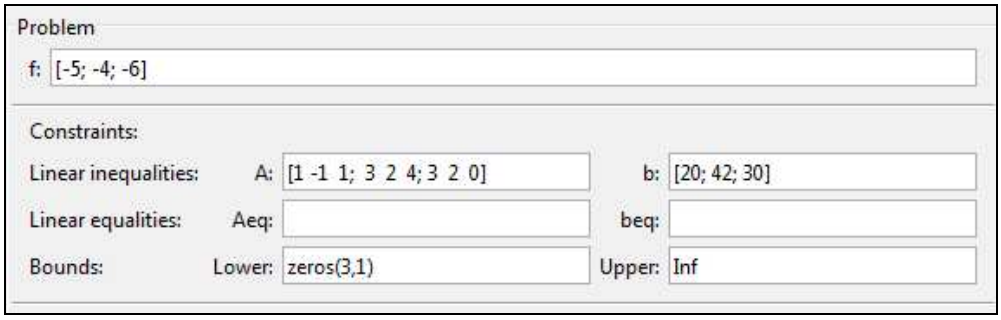
при умові, що

$$x_1 - x_2 + x_3 \le 20;$$
$$3x_1 + 2x_2 + 4x_3 \le 42;$$
$$3x_1 + 2x_2 \le 30;$$
$$0 \le x_1, 0 \le x_2, 0 \le x_3.$$

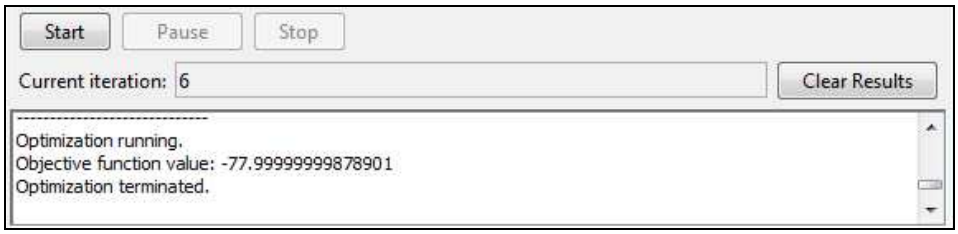
Вибір алгоритму **linprog**:



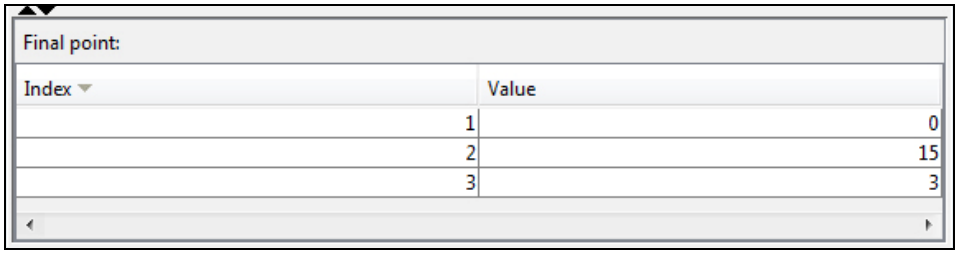
Визначення функції і параметрів:



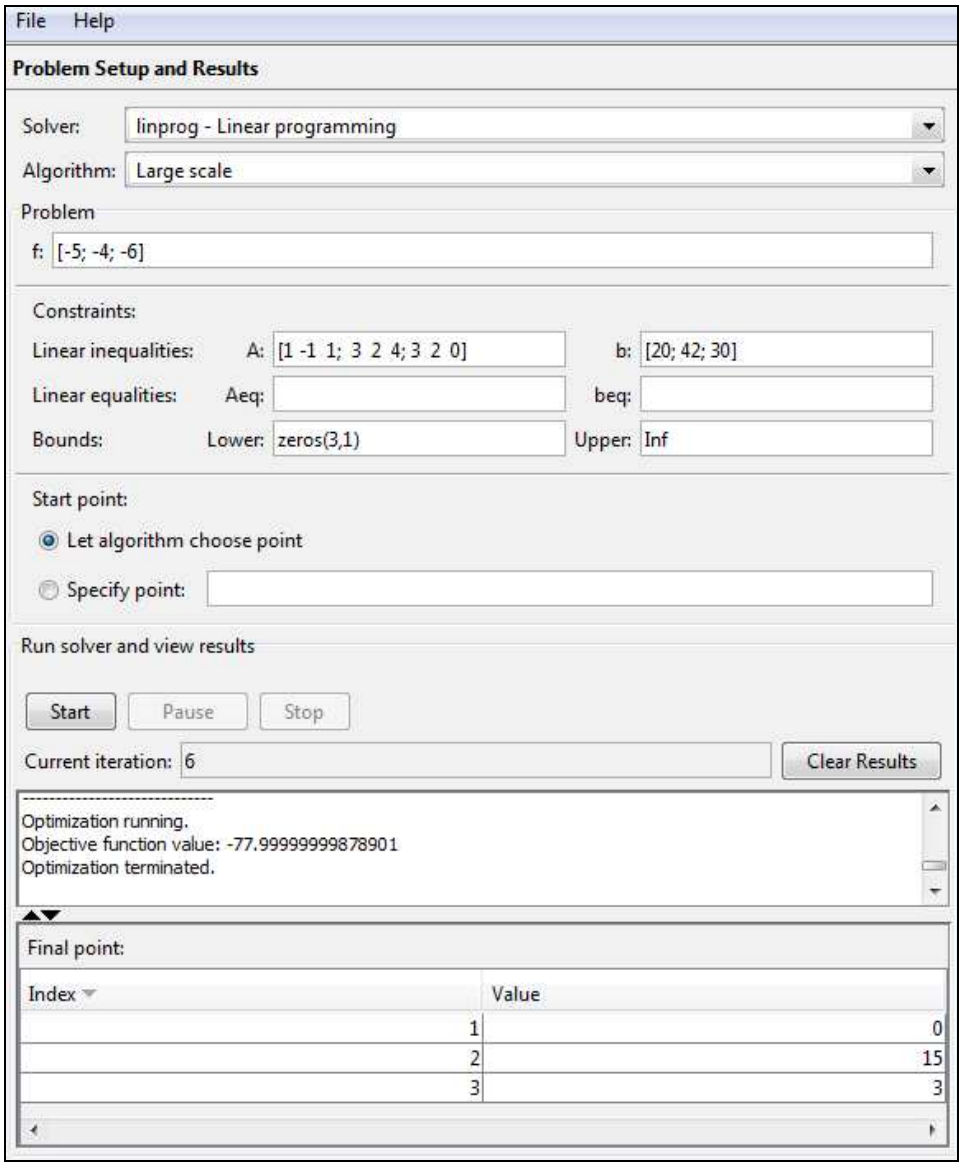
Отримання результатів:



Значення параметрів:



Загальний вигляд інтерфейсу:



Приклад 2. Пошук мінімума функції $f(x)=\frac{1}{2}x_1^2+x_2^2-x_1x_2-2x_1-6x_2$ при умові, що

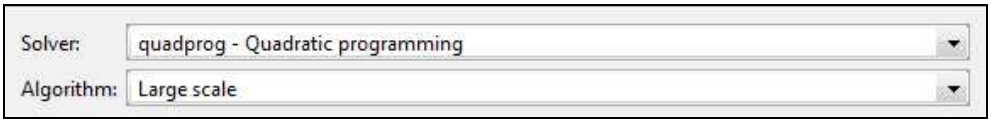
$x_1 + x_2 \le 2;$

$-x_1 + 2x_2 \le 2;$

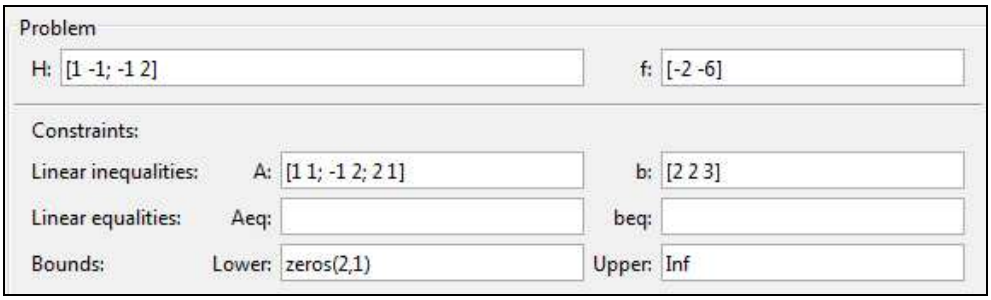
$2x_1 + x_2 \le 3;$

$0 \le x_1, 0 \le x_2.$

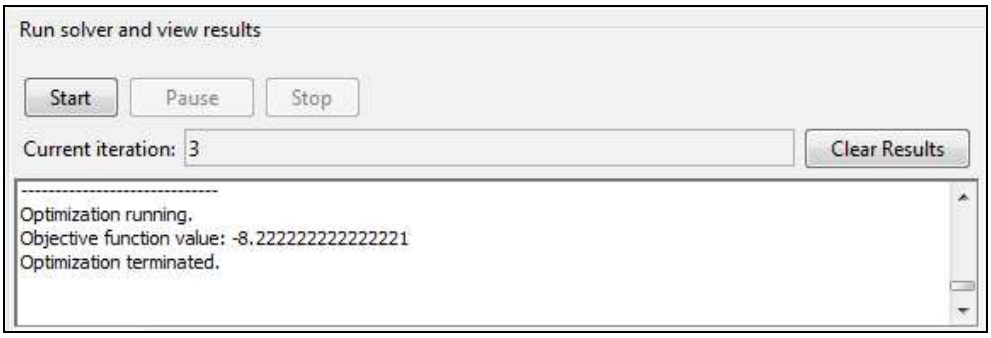
Вибір алгоритму **quadprog**:



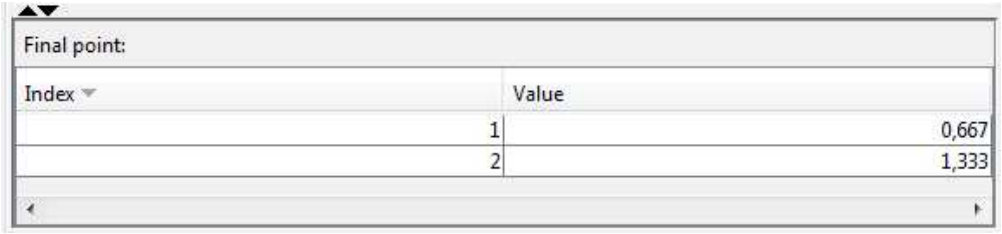
Визначення функції параметрів:



Отримання результатів:



Значення параметрів:

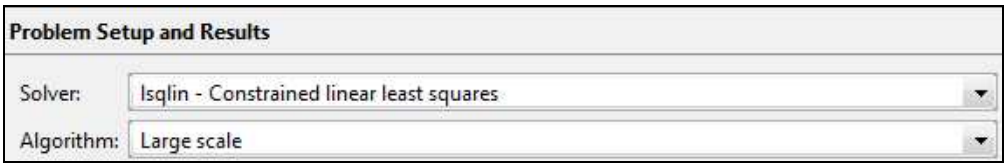


Приклад 3. Знахождення мінімуму виразу $x_1^2 + x_2^2 + x_3^2$ при обмеженнях:

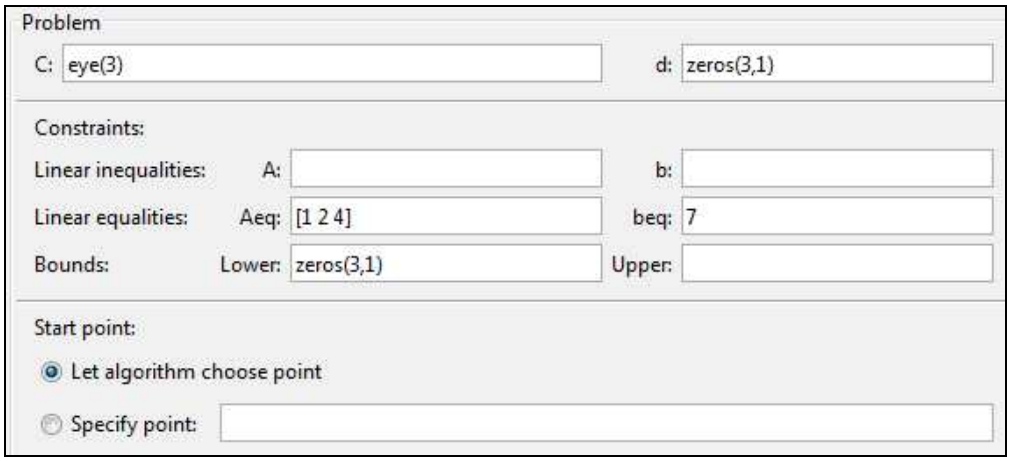
$x_1 + 2x_2 + 4x_3 \le 7;$

$x_1 \ge 0, x_2 \ge 0, x_3 \ge 0.$

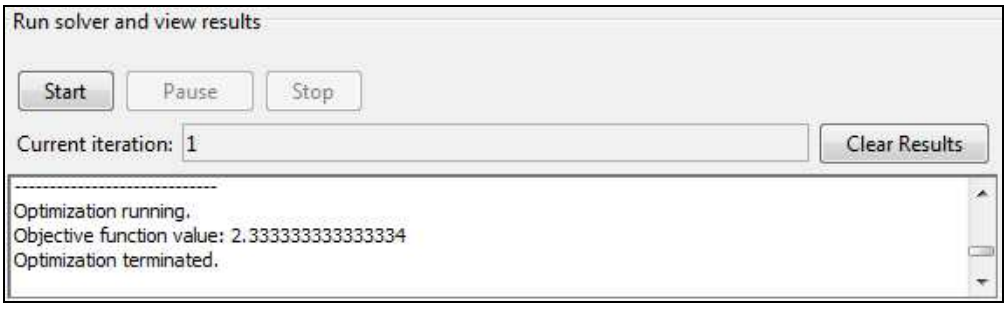
Вибір алгоритму **lsqlin**:



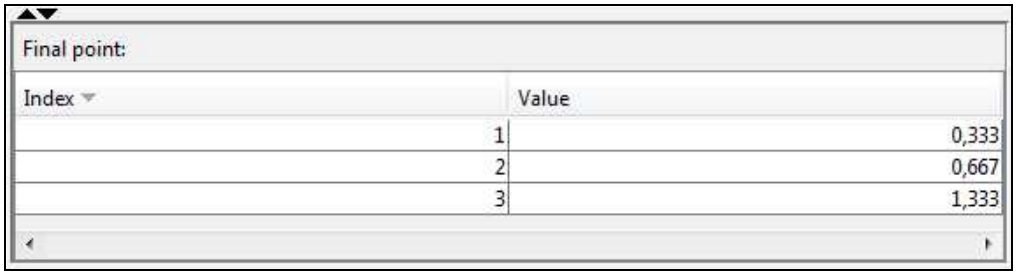
Визначення параметрів:



Отримання результатів:



Значення параметрів:



Приклад 4**.** Вирішити задачу цілочисельного програмування, тобто мінімізувати функцію $f(x) = -9x_1 - 5x_2 - 6x_3 - 4x_4$ при обмеженнях:

$$6x_1 + 3x_2 + 5x_3 + 2x_4 \le 9;$$

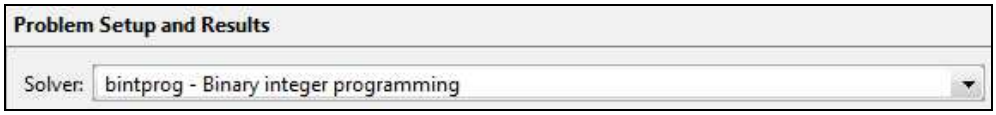
$$x_3 + x_4 \le 1;$$

$$-x_1 + x_3 \le 0;$$

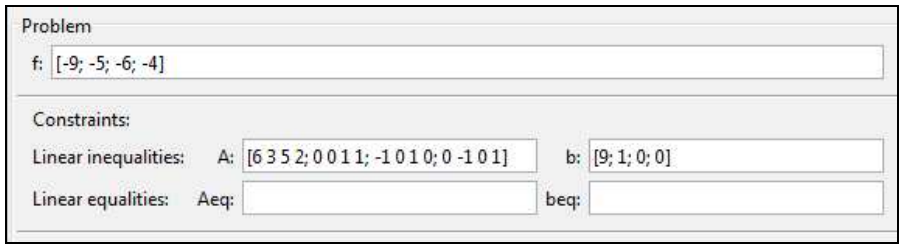
$$-x_2 + x_4 \le 0,$$

де x_1, x_2, x_3 і x_4 є бінарними цілими.

Вибір алгоритму **bintprog**:

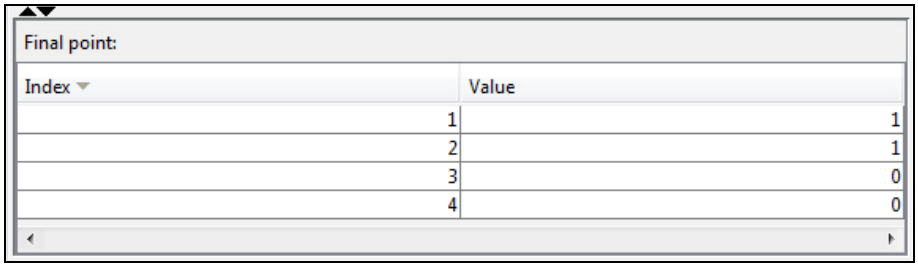


Визначення параметрів:



Получение результатов:

Значення параметрів:



Контрольні запитання

1. Яке призначення команди optimtool?

2. Як задаються параметри функції linprog у середовищі Optimization Tool?

3. Як задаються параметри функції quadprog у середовищі Optimization Tool?

4. Як задаються параметри функції lsqlin у середовищі Optimization Tool?

5. Як задаються параметри функції bintprog у середовищі Optimization Tool?

9. Задача класифікації (функції gscatter и classify)

Лінійний дискримінантний аналіз (ЛДА) – метод статистики і машинного навчання, що застосовується для знаходження лінійних комбінацій ознак, які розділяють два або більше класів об'єктів або подій. Отримана комбінація може бути використана як лінійний класифікатор. Алгоритм лінійного дискриминатного аналізу припускає, що межі між класами апроксимуються за допомогою лінійних функцій. Квадратичний дискримінантний аналіз (КДА) є природним узагальненням методу ЛДА. У разі ЛДА поверхня, що розділяє класи описується лінійними рівняннями, а у разі КДА – квадратичними рівняннями.

Для реалізації методів ЛДА і КДА в середовищі MATLAB використовують функції **gscatter** і **classify** з **Statistics Toolbox**.

Функція **gscatter** призначена для побудови графіка розподілу двох змінних, що групуються за значеннями третьої змінної.

Функція **classify** забезпечує класифікацію методама лінійного або квадратичного дискриминантого аналізу.

Деякі синтаксичні реалізації функції gscatter:

```
gscatter(x,y,group);
gscatter(x,y,group,clr,sym),
```

де gscatter(x,y,group) реалізує побудову графіка від x і y, згрупованих за group. x і y – вектори того ж розміру, що й group – вектор або масив рядків із значеннями категорій, що відповідають певним значенням x і y.

gscatter(x, y, group, clr, sym, siz) – це узагальнення попереднього виразу, що визначає крім того колір, тип маркерів і розмір для кожної групи. clr – масссив кольорів, відповідних функції plot, за умовчанням clr рівний 'bgrcmyk', sym – масив символів, також соответсвующих команді plot (за умовчанням – '.'), siz – вектор розмірів символів (за умовчанням – значення властивості 'DefaultLineMarkerSize').

Функція classify використовуватиметься в подальшому викладі в таких форматах:

```
class = classify(sample,training,group);
class = classify(sample,training,group,'type'),
```

де class = classify(sample, training, group) дозволяє класифікувати кожен рядок вхідних даних з sample за навчальною множиною (training), ставлячи їм у відповідність групу значень класифікатора (group). Тут sample і training - матриці з однаковим числом стовпців. Вихідний клас (class) вказує на групу, в якій кожен рядок визначений і має той самий тип, що і група (group).

Приклад: Необхідно побудувати лінійний і квадратичний класифікатори на підставі масиву точок, визначених координатами:

(1,3); (6,4); (3,7); (4,5);(7,7); (5,8); (7,3); (8,2); (3,6); (2,3); (6,0); (3,7); (0,5); (7,4); (5,3); (4,1); (3,6).

Попередня класифікація для цих точок задана вектором:

['a';'a';'a';'a';'b';'b';'b';'b';'c';'c';'c';'c';'c';'c';'c';'c';'c'].

Визначити якість класифікації.

Для вирішення завдання у вікні Command Window вводиться:

```
>> a=[1;6;3;4;7;5;7;8;3;2;6;3;0;7;5;4;3]
>> b=[3;4;7;5;7;8;3;2;6;3;0;7;5;4;3;1;6]
>> g=[a,b]
>> species=['a';'a';'a';'a';'b';'b';'b';'b';'c';'c';'c';'c';
'c';'c';'c';'c';'c']
```

після чого будуємо графік (рис. 10) за допомогою функції gscatter :

```
>> gscatter(g(:,1), g(:,2), species,'rgb','osd');
```

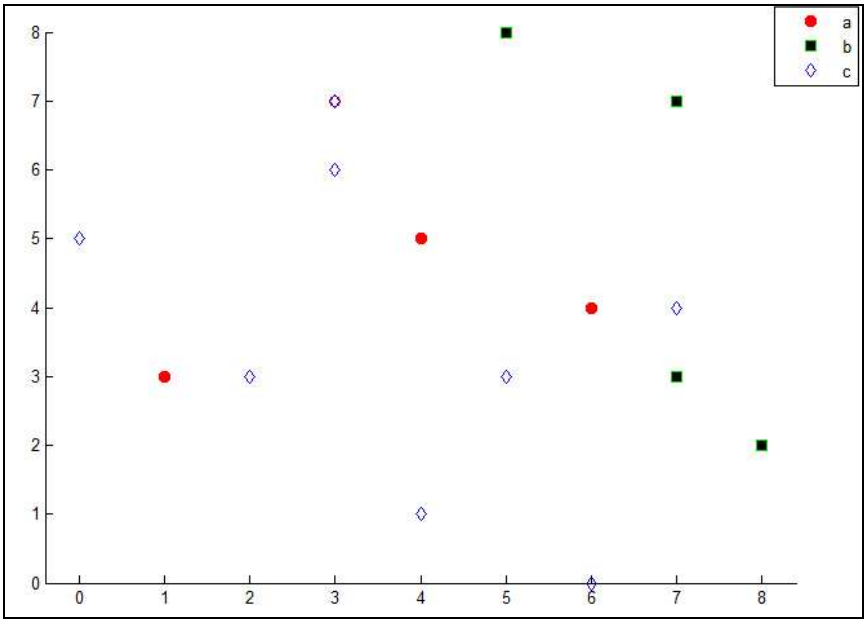


Рис. 10. Відображення результату тренувальної (попередньої) класифікації

Далі методом ЛДА будується класифікатор і за його допомогою виконується класифікація. Неспівпадаючі результати початкової класифікації і класифікації методом ЛДА називатимемо "помилками" і відмітимо закресленням (рис. 11).

Для виконання цього завдання у вікні Command Window вводяться команди:

```
>> hold on
>> linclass = classify(g(:,1:2),g(:,1:2),species);
>> bad=~strcmp(linclass,species);
```

```
>> plot(g(bad,1),g(bad,2),'kx');
```

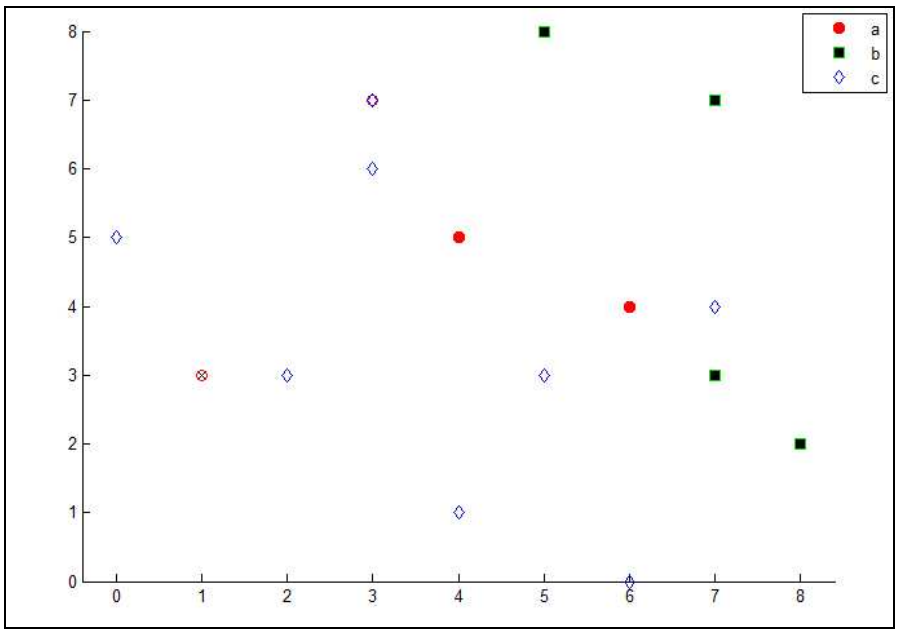


Рис. 11. Класифікація, уточнена методом ЛДА

Похибка класифікації обчислюється шляхом введення команди :

```
>> ldaResubErr = sum(bad) / 17
ldaResubErr =
    0.0588
```

Як бачимо, похибка всього одна – це точка з координатами (1,3).

Для наочної демонстрації ліній ЛДА-класифікатора (рис. 12) класифікуємо точки, розташовані на площині.

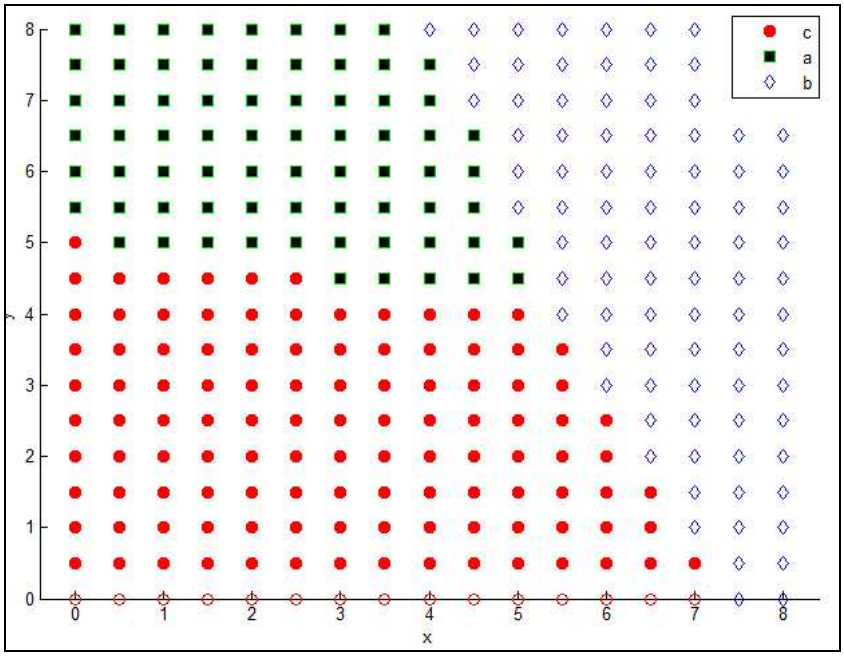


Рис. 12. Області ЛДА-класифікатора

Для цього введемо команди:

```
>> [x,y]=meshgrid(0:0.5:8,0:0.5:8)
>> x=x(:);
>> y=y(:);
>> j=classify([x y],g,species)
>> hold off
>>gscatter(x, y, j,'rgb','osd');
```

Далі для порівняння побудуємо класифікатор і виконаємо власну класифікацію за допомогою методу КДА, для чого введемо команди:

```
>> hold on
>> qdaClass =classify(g(:,1:2),g(:,1:2),species,'quadratic');
>> bad=~strcmp(linclass,species);
>> plot(g(bad,1),g(bad,2),'kx');
```

Похибка класифікації обчислюється шляхом введення команди:

```
>> qdaResubErr = sum(bad) / 17
qdaResubErr =
    0.0588
```

Як бачимо, для цього прикладу похибки у точці (1,3) у разі КДА і ЛДА співпадають.

Області КДА-класифікатора (рис. 13) будуються аналогічно, через введення команд:

```
>> [x,y]=meshgrid(0:0.5:8,0:0.5:8)
>> x=x(:);
>> y=y(:);
>> j=classify([x y],g,species, 'quadratic')
>> hold off
>> gscatter(x, y, j,'rgb','osd');
```

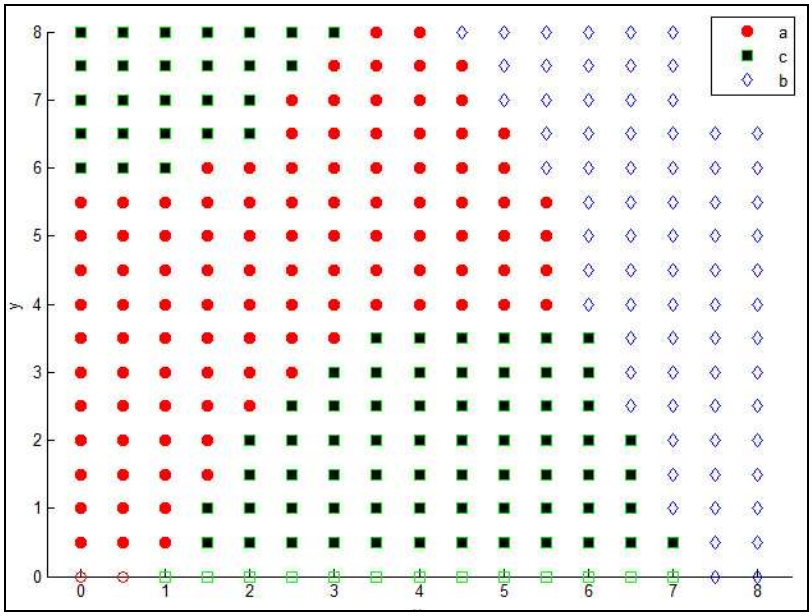


Рис. 13. Області КДА-класифікатора

Контрольні запитання

1. Які основні можливості пакету Statistics Toolbox?

2. Що таке лінійний і квадратичний дискримінантний аналіз? Яка різниця між ними?

3. Яке призначення і синтаксис функції gscatter?

4. Яке призначення і синтаксис функції classify?

5. Яким чином обраховується похибка класифікації?

10. Задача ієрархічного кластерного аналізу (Statistics Toolbox)

Кластерний аналіз – метод групування експериментальних даних у класи. Спостереження, що потрапили в один клас, у певному сенсі ближче одне до одного, ніж до спостережень з інших класів.

Кластеризація – розбиття множини даних на класи (кластери), смислові параметри яких заздалегідь невідомі. Кількість кластерів може бути довільною або фіксованою. Якщо класифікація допускає приписування даним визначених, відомих заздалегідь ознак, то кластеризація складніший процес, який допускає не лише приписування документам деяких ознак, але і виявлення самих цих ознак як основ формування класів. Мета методів кластеризації даних полягає в тому, щоб подібність даних, які потрапляють в кластер, була максимальною.

Для вивчення методів кластеризації, реалізованих в пакеті MATLAB, зупинимося на таких функціях з **Statistics Toolbox**, як **pdist, linkage, cluster, dendrogram**.

Функція **pdist** – розрахунок парних відстаней.

Розглянемо основні формати функції pdist :

Y = pdist(X)

Y = pdist(X,'metric')

Функція pdist(X) дозволяє розрахувати вектор відстаней Евкліда Y між парами об'єктів початкової великої кількості 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 багатовимірної випадкової величини – параметр Y є вектором, що містить значення парних відстаней між об'єктами. Число елементів Y дорівнює $\frac{m(m-1)}{2}$.

За допомогою функції **squareform** вектор Y можна конвертувати у квадратну матрицю. Елемент (i, j) отриманої матриці, при $i < j$, відповідатиме відстані між i-м і j-м об'єктами початкової множини даних.

У виразі Y = pdist(X', metric') вхідний параметр 'metric' визначає вид відстані між об'єктами початкової множини даних. 'metric' задається як строкова змінна.

Передбачені наступні види відстані між об'єктами:

Значення параметра 'metric'	**Метод розрахунку відстані між об'єктами**
'euclidean'	Відстань Евкліда. Значення за умовчанням
'minkowski'	Метрика Мінковського
'cosine'	Косинусна відстань. Визначається як одиниця мінус косинус від кута між об'єктами. Об'єкти в багатовимірному просторі розглядаються як вектори
'hamming'	Відстань Хэмминга. Визначається як відсоток відмінних координат від їх загального числа

```
>> X=normrnd(0,1,5,3)
X =
    0.1139    0.2944    0.8580
    1.0668   -1.3362    1.2540
    0.0593    0.7143   -1.5937
   -0.0956    1.6236   -1.4410
   -0.8323   -0.6918    0.5711
```

Евклідова відстань:

```
>> Y = pdist(X)
Y =
  Columns 1 through 8
    1.9297    2.4880    2.6638    1.3965    3.6509    4.1682
2.1185    0.9349
  Columns 9 through 10
    2.7311    3.1547
```

Квадратна (5x5) матриця зв'язків:

```
>> S = squareform(Y)
S =
    0         1.9297    2.4880    2.6638    1.3965
    1.9297         0    3.6509    4.1682    2.1185
    2.4880    3.6509         0    0.9349    2.7311
    2.6638    4.1682    0.9349         0    3.1547
    1.3965    2.1185    2.7311    3.1547         0
```

Функція linkage - формування ієрархічного дерева кластерів

Основні формати:

```
Z = linkage(Y)
Z = linkage(Y,'method')
```

Z = linkage(Y) – дозволяє сформувати ієрархічне дерево бінарних кластерів з використанням алгоритму "найближчого сусіда". Вхідний аргумент Y є вектором відстаней між парами об'єктів початкової множини даних. Число елементів вектора Y дорівнює $\frac{m(m-1)}{2}$, де m – кількість

об'єктів в початковій множині даних. Y може бути отриманий як вихідний параметр функції pdist.

Вихідний параметр Z є матрицею, що містить інформацію про дерево кластерів. Розмірність Z дорівнює $3(m-1)$.

Кінцеві вузли дерева кластерів є об'єктами початкової множини даних – спостережень багатовимірної випадкової величини Y, пронумерованих від 1 до *m*. Кінцеві вузли є одиничними кластерами. Вони поєднуються в кластери розміщеними вище вузлами дерева. Кожному наступному розміщеному вище вузлу дерева кластерів відповідає *i*-й рядок матриці Z. Йому ставиться у відповідність індекс $m+i$.

Стовпчики 1 і 2 матриці Z містять індекси об'єктів, пов'язаних в новий кластер. Кількість сформованих бінарних кластерів буде дорівнювати ($m-1$).

3-й стовчик матриці Z містить значення відстаней між парами об'єктів, об'єднаних в кластери.

Припустимо, що дерево кластерів містить 30 початкових вузлів. Якщо 10-й кластер був сформований об'єднанням 5-го і 7-го вузлів і відстань між ними дорівнює 1,5, тоді 10-й рядок матриці Z міститиме наступні значення Z(:, 10)=[5 7 1.5]. Цей кластер матиме індекс рівний 10+30=40.

Z = linkage(Y', method') вхідний аргумент 'method' дозволяє задати алгоритм кластеризації. Значення вхідного аргументу 'method' задається як тестовий рядок. Передбачені наступні алгоритми кластеризації:

Метод	Назва алгоритму
"single'	Алгоритм "найближчого сусіда". Базується на визначенні найменшої відстані між об'єктами в двох групах. Значення за умовчанням
"complete'	Алгоритм "далекого сусіда". Базується на визначенні найбільшої відстані між об'єктами у двох групах
"average'	Алгоритм "середнього зв'язку". Базується на розрахунку відстаней між усіма можливими парами об'єктів в кластерах
"centroid'	Центроїдний алгоритм, що використовує відстань за "центрам тяжіння" вибірок. Відстань між центроїдами кластерів
"ward'	Покроковий алгоритм. Відстань між кластерами, визначається за центроїдним алгоритмом. Алгоритм базується на збільшенні загальної внутрішньогрупової суми квадратів в результаті приєднання груп

Формування дерева кластерів базується на послідовному об'єднанні пар вузлів нижчого рівня у вузли вищого рівня.

Приклади формування ієрархічних дерев

Приклад 1. Формування ієрархічного дерева бінарних кластерів для тривимірної нормально розподіленої випадкової величини. Кількість об'єктів

у множині початкових даних дорівнює 5. Графічне представлення дерева бінарних кластерів виконується за допомогою функції **dendrogram** (рис. 14).

```
>> X=normrnd(0,1,5,3);
>> Y = pdist(X);
>> Z = linkage(Y)
   Z =
    3.0000    4.0000    0.9349
    1.0000    5.0000    1.3965
    2.0000    7.0000    1.9297
    6.0000    8.0000    2.4880

>> H = dendrogram(Z);
```

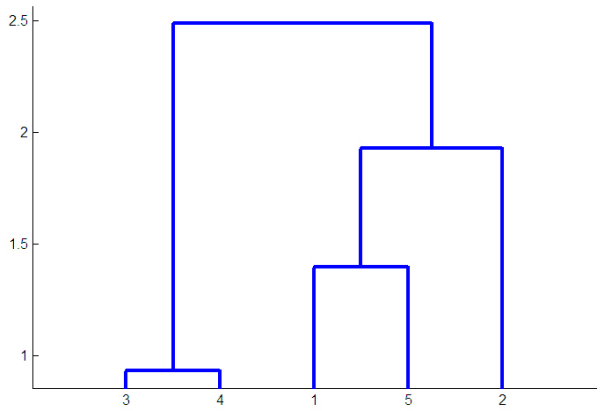


Рис. 14. Дерево бінарних кластерів

Приклад 2. Формування ієрархічного дерева бінарних кластерів для двомірної випадкової величини. Кількість об'єктів у множині початкових даних дорівнює 7. Порівнюються різні алгоритми кластеризації. Графічне представлення результатів кластеризації (рис. 15) виконується за допомогою функції **dendrogram**.

Початкові дані:

```
>> X = [3 1.7; 1 1; 2 3; 2 2.5; 1.2 1; 1.1 1.5; 3 1]
X =
    3.0000    1.7000
    1.0000    1.0000
    2.0000    3.0000
    2.0000    2.5000
    1.2000    1.0000
    1.1000    1.5000
    3.0000    1.0000
>> Y = pdist(X);
Y =
  Columns 1 through 11
    2.1190    1.6401    1.2806    1.9313    1.9105    0.7000
2.2361    1.8028    0.2000    0.5099    2.0000
  Columns 12 through 21
```

```
    0.5000    2.1541    1.7493    2.2361    1.7000    1.3454
1.8028    0.5099    1.8000    1.9647
```

Кластеризація за допомогою алгоритму "найближчого сусіда":

```
>> Z = linkage(Y, 'single')
Z =

    2.0000    5.0000    0.2000
    3.0000    4.0000    0.5000
    6.0000    8.0000    0.5099
    1.0000    7.0000    0.7000
    9.0000   11.0000    1.2806
   10.0000   12.0000    1.3454

>> dendrogram(Z);
```

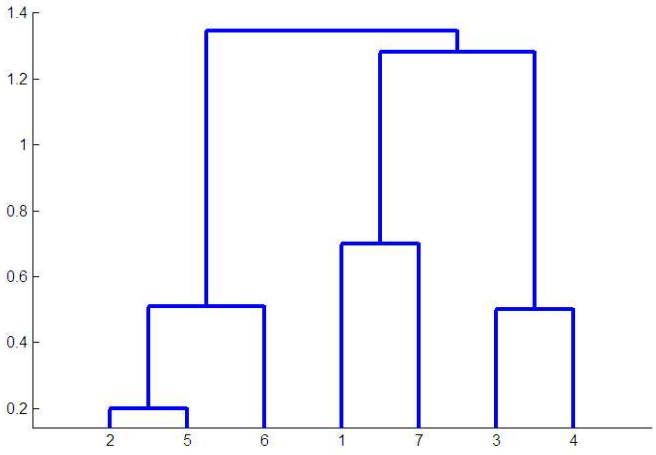
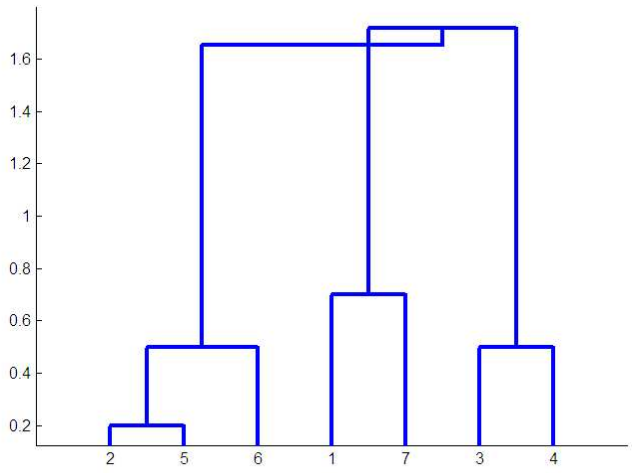


Рис. 15. Дерево бінарних кластерів для двовимірної випадкової величини

Кластеризація за допомогою центроїдного алгоритму (рис. 16)

```
>> Z = linkage(Y, 'centroid')
Z =
    2.0000    5.0000    0.2000
    6.0000    8.0000    0.5000
    3.0000    4.0000    0.5000
    1.0000    7.0000    0.7000
   10.0000   11.0000    1.7205
    9.0000   12.0000    1.6554

>> H = dendrogram(Z)
H =
  158.0043
  160.0043
  161.0043
  162.0043
  163.0043
```




```
164.0043
```

Рис. 16. Результати застосування центроїдного алгоритму

Функція cluster - кластеризація по виходу функції linkage

Розглядаються формати:

T = cluster(Z,'cutoff',c)

T = cluster(Z,'maxclust',n)

T = cluster(Z', cutoff ', c): функція призначена для розподілу на окремі кластери T ієрархічного дерева кластерів Z, отриманого за допомогою функції **linkage**. Вхідний параметр Z є матрицею розмірності $3(m-1)$, де m – кількість спостережень багатовимірної випадкової величини початкової множини даних. Вхідний параметр C – порогова величина призначена для розбиття ієрархічного дерева Z на кластери. Кластер формується шляхом виключення зв'язків ієрархічного дерева кластерів для яких значення коефіцієнтів несумісності менше величини C. Вихідний параметр T є вектором номерів кластерів, до яких віднесені об'єкти початкової множини даних. Число елементів вектора T дорівнює m.

T = cluster(Z', maxclust ', n): вхідний параметр n визначає максимальну кількість кластерів, на яку ділиться ієрархічне дерево кластерів Z.

Приклади застосування функції кластеризації об'єктів

Як початкові дані використовується 5-ти вимірна випадкова величина X. Розподіл X є композицією нормального закону і закону Вейбулла. Порівнюється розподіл об'єктів за кластерами для порогової величини, що дорівнює 0.1*(max(Z(:,3))), 0.3*(max(Z(:,3))), 0.4*(max(Z(:,3))). Графічне представлення ієрархічного дерева кластерів (рис. 17) виконується за допомогою функції **dendrogram**.

```
>> X=normrnd(0,1,15,5)+weibrnd(1,2,15,5);
>> Y = pdist(X);
>> Z = linkage(Y);
```

```
>> t1=0.1*(max(Z(:,3)));
>> t2=0.3*(max(Z(:,3)));
>> t3=0.4*(max(Z(:,3)));
>> dendrogram(Z);
```

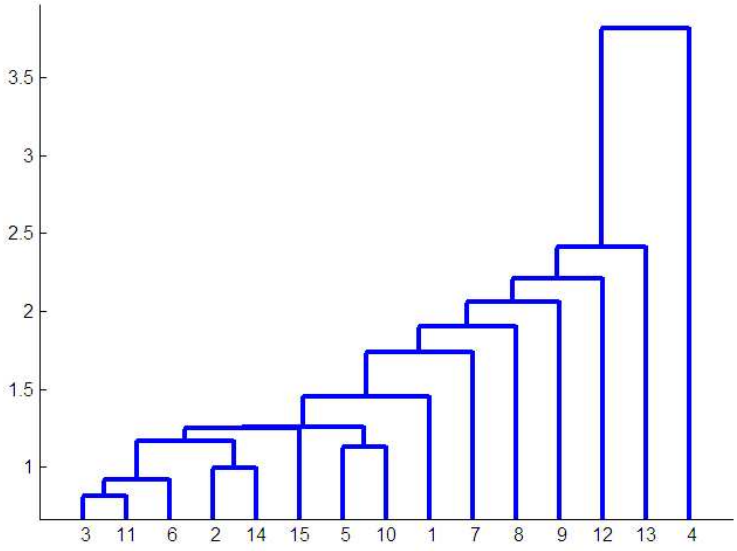


Рис. 17. Ієрархічне дерево кластерів

```
>> T1 = cluster(Z,'cutoff', t1);
>> T2 = cluster(Z,'cutoff', t2);
>> T3 = cluster(Z,'cutoff', t3);
>> cat(2,T1,T2,T3)
ans =
     4     1     1
    12     1     1
    11     1     1
    10     1     1
     3     1     1
     1     1     1
     5     1     1
     6     1     1
     7     1     1
     3     1     1
    11     1     1
     8     1     1
     9     1     1
    12     1     1
     2     1     1
```

Кластеризація об'єктів за вихідним параметром функції linkage

До початкових даних застосовується 10-ти вимірна випадкова величина X. Об'єм вибірки дорівнює 25 елементам. X розподілена за нормальним законом. Графічне представлення ієрархічного дерева кластерів (рис. 18) виконується за допомогою функції **dendrogram**. Максимальне число кластерів прийняте рівним 5.

```
>> X=normrnd(0,1,25,10);
>> Y = pdist(X);
```

```
>> Z = linkage(Y);
>> dendrogram(Z);
```

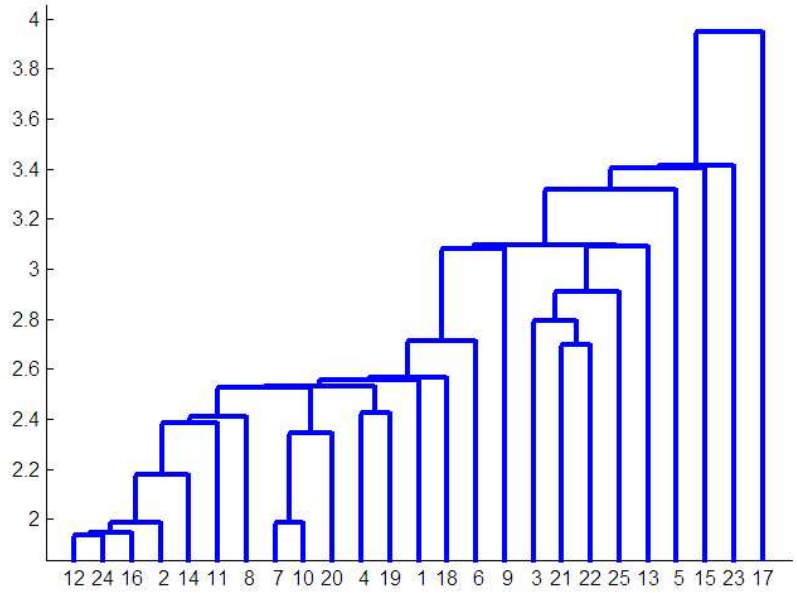


Рис. 18. Ієрархічне дерево кластерів

```
>> T = cluster(Z,'maxclust', 5);
>> T'
ans =
     2  2  2  2  1  2  2  2  2  2  2  2  2  2  3  2  5  2  2  2  2  2  4  2  2
```

Контрольні запитання

1. Надати визначення кластерного аналізу. Чим кластерний аналіз відрізняється від процесу класифікації?

2. Яке призначення і синтаксис функції pdist?

3. Яке призначення і синтаксис функції linkage?

4. Яке призначення і синтаксис функції cluster?

5. Яке призначення і синтаксис функції dendrogram?

11. Теорія ігор. Рішення матричних ігор

Класичним випадком, що розглядається у теорії ігор, є кінцева парна гра з нульовою сумою (антагоністична гра двох осіб). Розглянемо таку гру *G*, в якій беруть участь два гравці *A* і *B*, що мають протилежні інтереси: виграш одного дорівнює програшу іншого. Виграш гравця *A* дорівнює виграшу гравця *B* із зворотним знаком. Нехай у *A* є *m* можливих стратегій $A_1, A_2, ..., A_m$, а у супротивника – *n* можливих стратегій $B_1, B_2, ..., B_n$ (тобто має місце гра $m \times n$). Позначимо a_{ij} – виграш гравця A у разі, якщо він користується стратегією A_i, а супротивник – стратегією B_j.

Припустимо, що для кожної пари стратегій A_i, B_j виграш a_{ij} відомий. Тоді, в принципі, можна скласти матрицю, в якій перераховані стратегії гравців і відповідні виграші.

Змішані стратегії в теорії ігор є моделлю мінливої тактики, коли жоден з гравців не знає, як супротивник поведе себе в цій партії. Позначатимемо змішані стратегії гравців *A* і *B* відповідно $S_A = (p_1, p_2, ..., p_m)$, $S_B = (q_1, q_2, ..., q_n)$, де $p_1, p_2, ..., p_m$ (що складають у сумі одиницю) – ймовірності застосування гравцем *A* стратегій $A_1, A_2, ..., A_m$; $q_1, q_2, ..., q_n$ – ймовірності застосування гравцем *B* стратегій $B_1, B_2, ..., B_n$.

У окремому випадку, коли усі ймовірності, крім однієї, дорівнюють нулю, а ця одна – одиниці, змішана стратегія перетворюється на чисту.

Існує основна теорема теорії ігор: будь-яка кінцева гра двох осіб з нульовою сумою має принаймні одне рішення – пару оптимальних стратегій, в загальному випадку змішаних (S_A^*, S_B^*), і відповідну ціну *v*.

Пара оптимальних стратегій (S_A^*, S_B^*), що утворює рішення гри, має наступну властивість: якщо один з гравців дотримується своєї оптимальної стратегії, то іншому не може бути вигідно відступати від своєї. Ця пара стратегій утворює в грі деяке положення рівноваги: один гравець хоче обернути виграш в максимум, інший – в мінімум, кожен тягне у свій бік і, при розумній поведінці обох, встановлюється рівновага і сталий виграш *v*.

Розглянемо приклад гри без сідлової точки і приведемо (без доказу) її рішення. Гра полягає у наступному: два гравці *A* і *B* одночасно і не змовляючись показують один, два або три пальці. Виграш вирішує загальна кількість пальців: якщо вона парне, виграє *A* і отримує у *B* суму, рівну цьому числу; якщо непарна, то, навпаки *A* виплачує *B* суму, що дорівнює цьому числу. Як вести себе гравцям?

Складемо матрицю гри. В одній партії у кожного гравця три стратегії: показати один, два або три пальці. Матриця 3×3 задана в таблиці:

	B_1	B_2	B_3
A_1	2	-3	4
A_2	-3	4	-5
A_3	4	-5	6

Необхідно знайти рішення гри, тобто дві оптимальні змішані стратегії, що дають кожній стороні максимально можливий для неї середній виграш (мінімальний програш).

Покажемо, як визначається S_A^*.

Відомо, що якщо один з гравців (A) застосовує свою оптимальну стратегію, то інший (B) не може поліпшити своє положення, відступаючи від своєї. Змусимо супротивника (B) відступати від своєї оптимальної стратегії, користуючись чистими стратегіями $B_1, B_2, ..., B_n$ (тим часом A наполегливо тримається стратегії S_A^*). У будь-якому випадку наш виграш буде не менше, ніж v :

$$\begin{cases} a_{11}p_1 + a_{21}p_2 + ... + a_{m1}p_m \geq v; \\ a_{12}p_1 + a_{22}p_2 + ... + a_{m2}p_m \geq v; \\ ... \\ a_{1n}p_1 + a_{2n}p_2 + ... + a_{mn}p_m \geq v. \end{cases}$$

Розділимо нерівності на позитивну величину v і введемо позначення:

$$x_1 = \frac{p_1}{v}, \quad x_2 = \frac{p_2}{v}, ..., \quad x_m = \frac{p_m}{v}.$$

Тоді умови-обмеження мають вигляд:

$$\begin{cases} a_{11}x_1 + a_{21}x_2 + ... + a_{m1}x_m \geq 1; \\ a_{12}x_1 + a_{22}x_2 + ... + a_{m2}x_m \geq 1; \\ ... \\ a_{1n}x_1 + a_{2n}x_2 + ... + a_{mn}x_m \geq 1, \end{cases}$$

де $x_1, x_2, ..., x_m$ – ненегативні змінні.

З того, що $p_1 + p_2 + ... + p_m = 1$, змінні $x_1, x_2, ..., x_m$ задовольняють умові

$$x_1 + x_2 + ... + x_m = \frac{1}{v}.$$

Але v є не що інше, як гарантований виграш; природно, його необхідно хочемо зробити максимальним, і, відповідно, величину $1/v$ – мінімальною.

Таким чином, задача рішення гри звелося до завдання: знайти ненегативні значення змінних $x_1, x_2, ..., x_m$ такі, щоб вони задовольняли лінійним обмеженням-нерівностям і обертали у мінімум лінійну функцію цих змінних:

$$x_1 + x_2 + ... + x_m \to \min.$$

Абсолютно аналогічно, з тією різницею, що В прагне мінімізувати, а не максимізувати виграш, а значить, обернути не в мінімум, а в максимум величину $1/v$, а в обмеженнях-нерівностях знаків "≥" стоятимуть "≤". Пара завдань лінійного програмування, за якими знаходяться оптимальні стратегії (S_A^*, S_B^*), називається парою подвійних завдань лінійного програмування. Доведено (теорема Дж. фон Неймана), що максимум лінійної функції в одній з них дорівнює мінімуму лінійної функції в іншій.

Щоб вирішити гру за допомогою функції **linprog**, введемо значення матриці гри (матриці обмежень-нерівностей):

```
>> A=[2 -3 4; -3 4 -5; 4 -5 6]
```

Отримання рішення задачі для гравця *A*:

```
>>x=linprog(ones(3,1),-A',-ones(3,1),[],[],zeros(3,1))
>>y=linprog(-ones(3,1),A,ones(3,1),[],[],zeros(3,1))
>>x
x =
    1.3192
    2.6384
    1.3192
>>y
y =
    1.3192
    2.6384
    1.3192
```

Відповідно, змішані стратеги визначаються як:

$$p \sim x = [1/4 \;\; 1/2 \;\; 1/4], \;\; q \sim y = [1/4 \;\; 1/2 \;\; 1/4].$$

Розглянемо ще один приклад, карткову гру за такими правилами: кожному з двох учасників здається три карти – червоний туз, чорний туз і двійка, причому у *A* двійка червона, а у *B* – чорна. Цей розклад відомий обом учасникам. Кожен вибирає із своїх трьох карт одну і кладе її на стіл сорочкою догори. Потім карти відкривають і визначають результат гри. Якщо обидві карти одного кольору, виграє *A*, якщо різних – *B*. Сума виграшу/програшу визначається вагою карти, викладеної переможцем: якщо це туз – одне очко, якщо двійка – два очки. Якщо відкрилися дві двійки, оголошується нічия (платіж дорівнює нулю). Платіжна матриця цієї гри наступна:

$$A=\begin{pmatrix} 1 & -1 & -2 \\ -1 & 1 & 1 \\ 2 & -1 & 0 \end{pmatrix}$$

Щоб вирішити гру (пряму і двоїстну задачу) застосуємо функцію linprog, підготувавши для неї заздалегідь дані:

```
>> A = [1 -1 -2;-1 1 1;2 -1 0];
```

Пряма задача:

```
>> [x,fval]=linprog(ones (3,1),-A',-ones(3,1),[],[],zeros(3,1))
x =
     0.0000
     3.0000
     2.0000
fval =
     5.0000
```

Двоїстна задача:

```
[y, gval] = linprog(-ones (3,1) , A, ones (3,1) , [] , [], zeros
(3,1) )
y =
2.0000
3.0000
0.0000
gval =
-5.0000
```

Негативне значення gval викликане зміною знаку перед вектором коефіцієнтів цільової функції. Ціна гри: *v* = *i/fvai* = *i*/5.

Змішані стратегії:

p = *v*×*x* = [0 3/5 2/5], *q* = *v*×*y* = [2/5 3/5 0] .

Контрольні запитання

1. Пояснити зміст основної теореми теорії ігор.

2. Пояснити сенс подвійного застосування функції linprog для рішення матричних ігор.

3. Пояснити поняття пари двоїстих задач лінійного програмування.

12. Рішення задачі мінімаксу (функція fminimax)

Функція fminimax з пакету Optimization Toolbox забезпечує розв'язок задачі мінімаксу від декількох функцій $F_i(x)$, а саме знаходженння $\min_x \max_{\{F_i\}} \{F_i(x)\}$ при обмеженнях вигляду:

$c(x) \le 0;$

$ceq(x) = 0;$

$A \cdot x \le b;$

$Aeq \cdot x = beq;$

$lb \le x \le ub,$

де *x, b, beq, lb* і *ub* – вектори; *A* і *Aeq* – матриці, *c(x), ceq(x)* і *F(x)* – такі функції, що повертають вектори. *F(x), c(x)* і *ceq(x)* можуть бути нелінійними функціями.

Синтаксис:

```
x = fminimax(fun,x0)
x = fminimax(fun,x0,A,b)
x = fminimax(fun,x0,A,b,Aeq,beq)
x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub)
x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
[x,fval] = fminimax(...)
```

Функція fminimax мінімізує найбільші (у теорії ігор – найгірші варіанти) значення системи функцій декількох змінних, починаючи із стартової оцінки. Ці значення можуть бути з обмеженнями. У загальному сенсі, це завдання відноситься до проблеми мінімакса.

- x = fminimax(fun, x0) починає з точки x0 і знаходить рішення мінімакса від x для функції, описаної в fun.
- x = fminimax(fun, x0, A, b) вирішує задачу мінімакса з умовою лінійних нерівностей A*x <= b.
- x = fminimax(fun, x, A, b, Aeq, beq) вирішує задачу мінімакса з умовою лінійної рівності Aeq*x = beq. Встановлюється A=[] і b=[] у разі відсутності нерівностей.
- x = fminimax(fun, x, A, b, Aeq, beq, lb, ub) визначає набір нижньої і верхньої меж для розрахункових параметрів так, що рішення завжди знаходиться в діапазоні lb <= x <= ub.
- x = fminimax(fun, x0, A, b, Aeq, beq, lb, ub, nonlcon) накладає на завдання мінімакса обмеження типу нерівностей c(x) або рівності

ceq(x), що задаються в nonlcon. fminimax мінімізується таким чином, що c(x) <= 0 і ceq(x) = 0. Встановлюється lb=[] і/або ub=[] у разі відсутності меж.

- [x, fval] = fminimax(..) повертає значення цільової функції як рішення від x.
- [x, fval, maxfval] = fminimax(..) повертає максимальне значення функції як рішення від x.
- [x, fval, maxfval, exitflag] = fminimax(..) повертає значення exitflag, яке містить вихідні умови для fminimax.
- [x, fval, maxfval, exitflag, output] = fminimax(..) повертає структурний вихід з інформацією про оптимізацію.

Вхідні аргументи – деталі функціонування для fun і nonlcon:

Fun функція, що підлягає мінімізації	fun є деяка функція, яка приймає вектор x і повертає вектор F, як цільову функцію від x. Функція fun має бути визначена, описувач функції. *x = fminimax(@myfun,x0);* де myfun – така функція MATLAB, що *function F = myfun(x).* fun також може бути внутрішнім об’єктом: *x = fminimax(inline('sin(x.*x)'),x0);*
Nonlcon функція, що обчислює нелінійні обмеження у вигляді нерівностей c(x)<= 0 і нелінійні обмеження у вигляді рівностей ceq(x) = 0.	Функція nonlcon приймає вектор x і повертає два вектори c і ceq. Вектор c містить нелінійні нерівності x, а ceq містить нелінійну рівність при розрахунку від x. Функція nonlcon може бути визначена як описувач функції. *x = fminimax(@myfun,x0,A,b,Aeq,beq,lb,ub,@mycon),* де mycon – функція MATLAB, що визначена як *function [c,ceq] = mycon(x)* *c* = ... % обчислює нелінійні обмеження у вигляді нерівностей від x. ceq = … % обчислює нелінійні обмеження у вигляді рівності від x.

Розглянемо приклад. Необхідно знайти значення x, в яких буде знайдено мінімум серед максимальних значень $\left[f_1(x), f_2(x), f_3(x), f_4(x) \right]$, де:

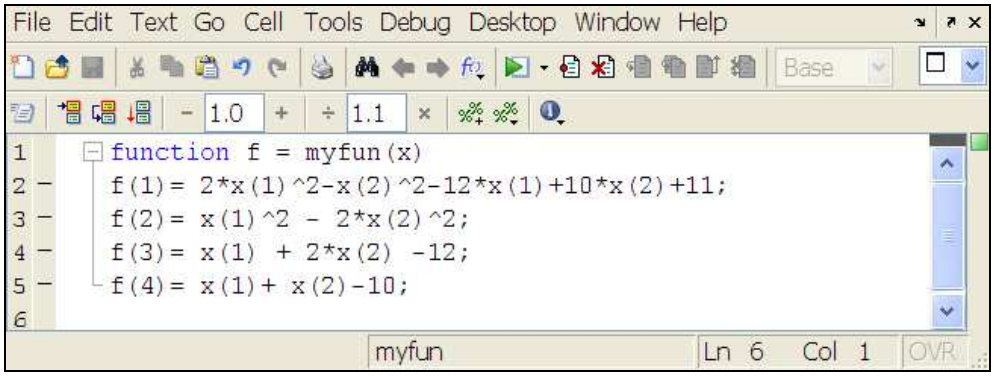
$$f_1(x) = 2x_1^2 - x_2^2 - 12x_1 + 10x_2 + 11;$$

$$f_2(x) = x_1^2 - 2x_2^2;$$

$$f_3(x) = x_1 + 2x_2 - 12;$$

$$f_4(x) = x_1 + x_2 - 10.$$

Запишемо М-файл, в якому розраховуються п'ять функцій від x:



Запуск функції оптимізації:

```
>> x0 = [1; -1];
>> [x,fval,no,no,output]  = fminimax(@myfun,x0)
```

Після 11-и ітерацій буде отримано рішення:

```
>> x
x =
   -3.0742
   -4.3582
>> output
output =
       iterations: 13
        algorithm: 'minimax SQP, Quasi-Newton, line_search'
>>fval
fval =
   -0.0092   -2.8536   -0.0000   -0.0000
```

Побудова графічної ілюстрації (рис. 19):

```
>> [x,y]=meshgrid(-5:0.2:5,-5:0.2:5);
>> z=2*x.^2-y.^2-12*x+10*y+11;
>> surf(x,y,z)
>> hold on
>> z=x.^2-2*y.^2;
>> surf(x,y,z)
>> z=x+2.*y-12;
>> surf(x,y,z)
>> z=x+y-10;
>> surf(x,y,z)
```

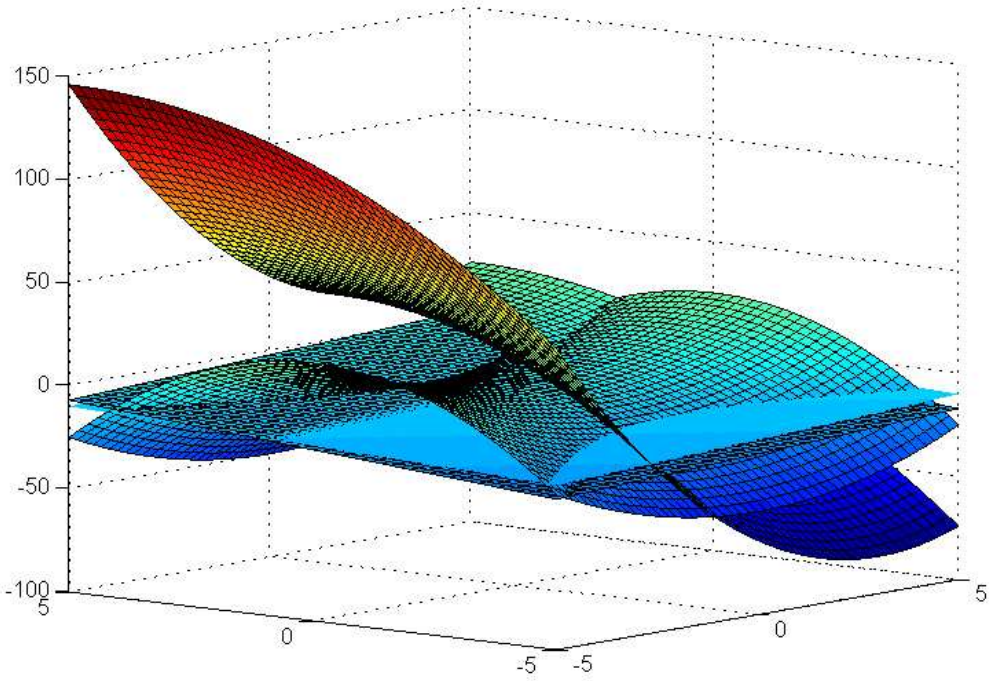


Рис. 19. Поверхні, що визначають завдання мінімакса

Контрольні запитання

1. Яке призначення функції fminimax з пакету Optimization Toolbox?

2. Який синтаксис функції fminimax, зміст вхідних і вихідних параметрів?

3. Для чого необхідно створювати функції, які зберігаються у М-файлах при застосуванні функції fminimax?

13. Генетичні алгоритми (Genetic Algorithm and Direct Search Toolbox)

Генетичні алгоритми – це метод рішення оптимізаційних задач, заснований на біологічних принципах природного відбору і еволюції. Генетичний алгоритм повторює кілька разів процедуру модифікації популяції (набору окремих рішень), прагнучи тим самим отримати нові набори рішень.

На кожному кроці генетичного алгоритму здійснюється імовірнісний добір деяких індивідуальних особливостей з поточного батьківського покоління і формується наступне дочірнє покоління. Через послідовний добір поколінь проходить "еволюція" – просування до оптимального рішення.

Генетичний алгоритм можна застосовувати для різноманітних завдань оптимізації, які не завжди вдало підходять для вирішення за допомогою стандартних оптимізаційних алгоритмів, і в першу чергу цей метод використовується при рішенні завдань, коли цільова функція є переривчастою, такою, що не диференціюється, стохастичною або у значні мірі нелінійною.

Усі функції цього тулбокса є M-файлами пакету MATLAB, складені з операторів MATLAB, і є реалізацією спеціалізованих алгоритмів оптимізації.

На кожному кроці для генерації наступного покоління в генетичному алгоритмі використовуються наступні три основних правила:

- правило відбору, яке добирає об'єкти, що мають назву батьків, які складають основу наступного покоління з поточного рішення;
- правило схрещування, за яким відбувається вибір з двох батьківських, формується дочірній об'єкт для наступного покоління;
- правило мутації, по якому на основі імовірнісних алгоритмів з батьківських об'єктів формуються дочірні.

Механізм роботи з генетичними алгоритмами в середовищі MATLAB може бути реалізовано двома способами:

1. Викликом функції генетичних алгоритмів через командний рядок.
2. Використанням комплекту Genetic Algorithm Tool

Обидва способи поставляються в числі стандартного набору функцій і модулів MATLAB.

Для того, щоб звернутися до генетичного алгоритму з командного рядка, слід у рамках наступного синтаксису викликати функцію генетичного алгоритму **ga:**

```
[x fval] = ga(@fitnessfun, nvars, options)
```

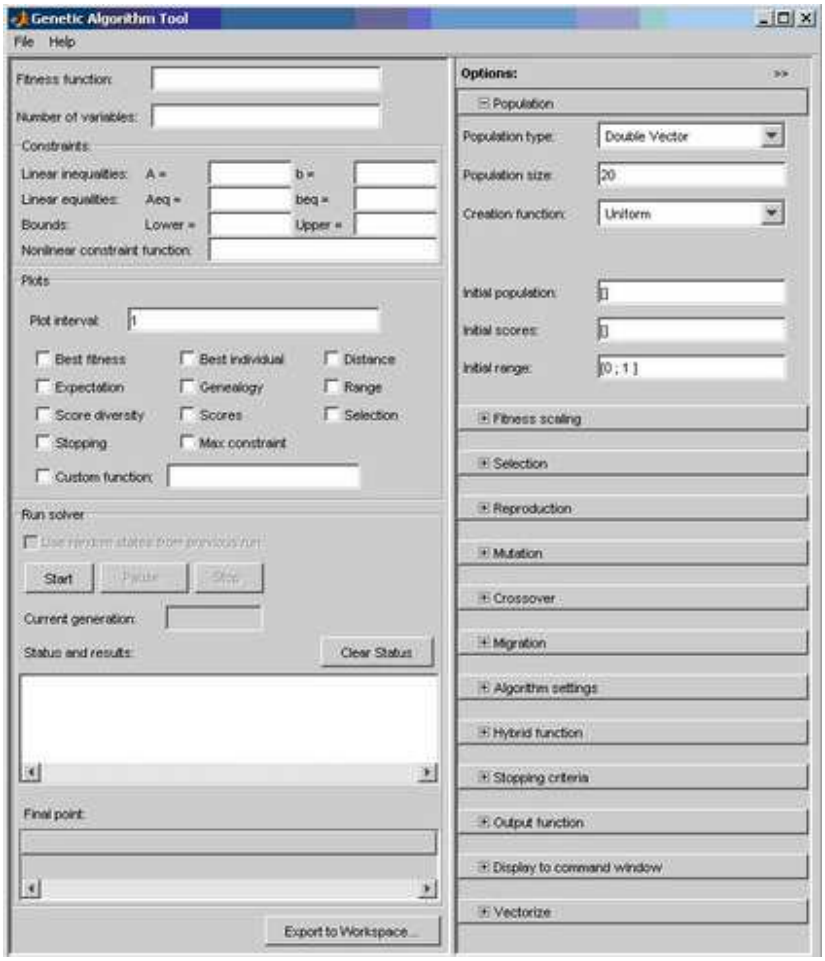
Значення параметрів:

- @fitnessfun – описувач функції придатності;
- nvars – число незалежних змінних функції придатності;
- options – структура, що включає опції генетичного алгоритму.

Отримані результати зберігаються в

- fval – кінцеве значення функції придатності;
- x – точка, в якій досягнуте кінцеве значення.

Для запуску пакету **Genetic Algorithm Tool** треба у командному рядку MATLAB виконати команду **gatool**. Після цього запуститься пакет генетичних алгоритмів і на екрані з'явиться основне вікно утиліти.



У полі **Fitness function** вказується функція, що оптимізується, у вигляді @fitnessfun, де fitnessfun.m – назва М-файлу, в якому заздалегідь слід описати функцію, що оптимізується.

Для виконання генетичного алгоритму слід клікнути мишкою на кнопку **Start**. Далі в панелі **Status and Results** здійснюється відображення результатів оптимізації.

Як приклад розглядається завдання знаходження оптимуму функції Растригіна, яка часто використовується для тестування генетичних алгоритмів.

У разі двох незалежних змінних функція Растригіна записується як

$$R(x) = 20 + x_1^2 + x_2^2 - 10\left(\cos 2\pi x_1 + \cos 2\pi x_2\right)$$

Для тривимірного відображення функції Растригіна (рис. 20) в середовищі MATLAB введемо команди:

```
>> [X,Y]=meshgrid(-4:0.05:4,-4:0.05:4)
>> Z=20+X.^2+Y.^2-10*(cos(2*3.1415*X)+ cos(2*3.1415*Y))
>> surfc(X,Y,Z)
```

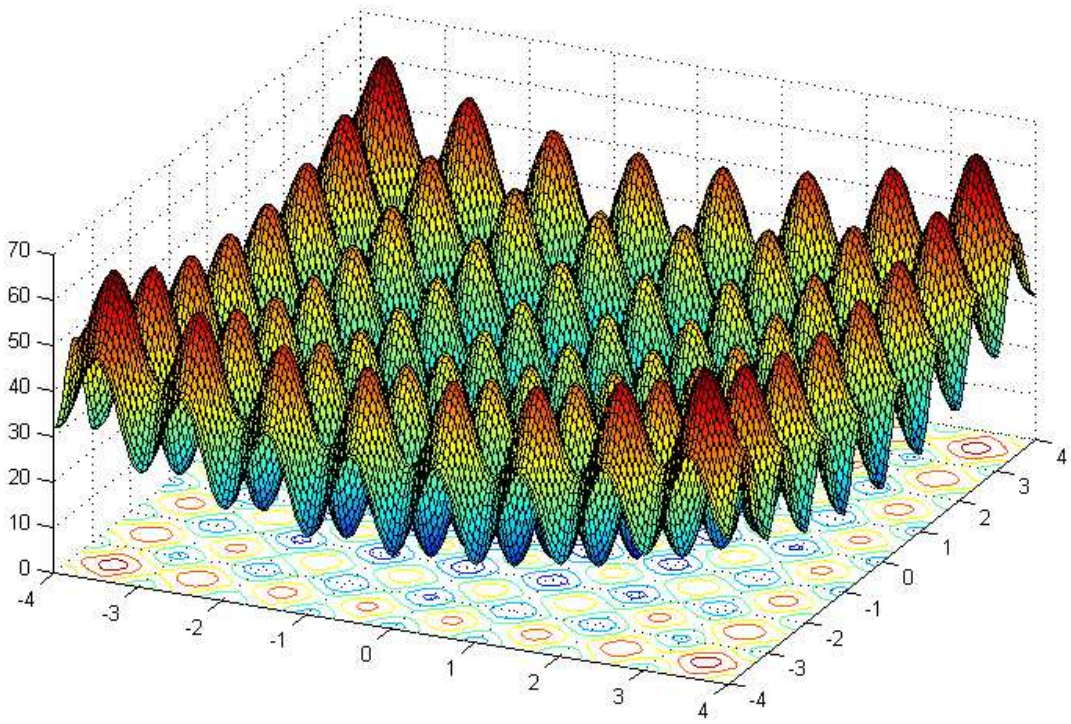


Рис. 20. Тривимірний графік функції Растригіна

Функція Растригіна часто використовується для тестування генетичного алгоритму, оскільки наявність великої кількості локальних мінімумів створює труднощі для використання стандартних методів пошуку глобального мінімуму на основі використання градієнтних методів.

Контурний графік функції Растрігина (рис. 21) ілюструє розташування альтернативних максимумів і мінімумів.

Слід зазначити, що до складу стандартних М -файлов MATLAB входить функція, що генерує функцію Растригіна. Для перегляду тексту цієї функції, досить з командного рядка ввести:

```
>> type rastriginsfcn.M
```

Отримаємо:

```
function scores = rastriginsfcn(pop)
```

```
%RASTRIGINSFCN Compute the "Rastrigin" function.
%   Copyright 2003-2004 The MathWorks, Inc.
%   $Revision: 1.1.6.1 $  $Date: 2009/08/29 08:28:19 $
    % pop = max(-5.12,min(5.12,pop));
    scores = 10.0 * size(pop,2) + sum(pop .^2 - 10.0 * cos(2 *
pi .* pop),2);
```

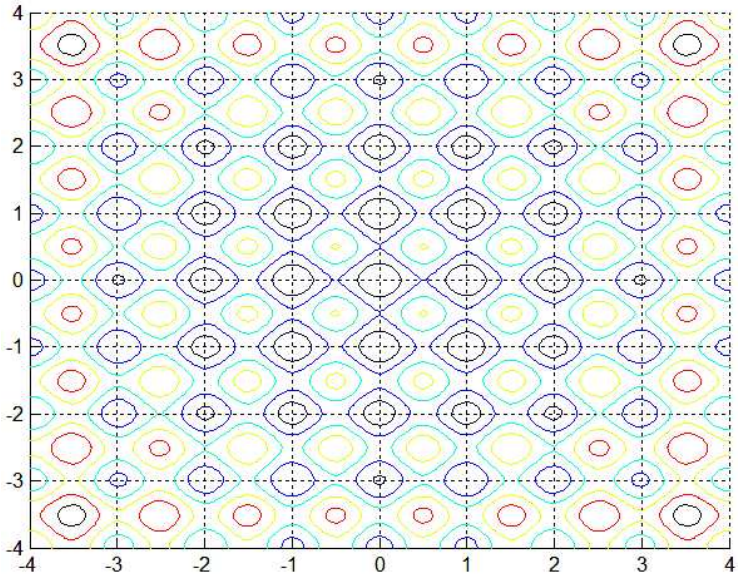


Рис. 21. Контурний графік функції Растригіна

Для знаходження мінімуму функції Растригіна слід виконати наступні кроки:

1. Ввести команду **gatool** в командному рядку, що активізує Genetic Algorithm Tool.
2. У пакеті Genetic Algorithm Tool виконати наступні дії:
    - У полі Fitness function, ввести @rastriginsfcn.
    - У полі Number of variables, ввести 2, число незалежних змінних для функції Растригіна.

   У поля **Fitness function** і **Number of variables** слід занести дані:

3. Клікнути мишкою на кнопку **Start** на панелі **Run solver**.

По мірі того, як відбувається виконання цього алгоритму, в полі **Current generation** відбувається відображення числа поточних генерацій. Можна час від часу припиняти алгоритм, натискаючи на кнопку **Pause**. Після того, як операція виконається, це ім’я кнопки заміниться на **Resume**. Для продовження роботи алгоритму з точки останову слід клікнути на кнопку **Resume**.

Після закінчення алгоритму на панелі **Status and results** відображується наступна інформація:

```
Function value: 0.0067749206244585025
```

Відмітимо, що значення, що відображується, знаходиться дуже близько до дійсного значення функції Растригина, рівного 0.

Таке ж рішення можна отримати, шляхом введення з командного рядка:

```
>> [x fval reason] = ga(@rastriginsfcn, 2)
```

Контрольні запитання

1. Поясниии призначення та принцип генетичних алгоритмів.

2. Яке призначення функції ga з пакету Genetic Algorithm and Direct Search Toolbox?

3. Який синтаксис функції ga, зміст вхідних і вихідних параметрів?

4. Для чого необхідно створювати функції, які зберігаються у М-файлах при застосуванні функції ga?

5. У чому відмінність графічних функцій surfc і surf?

14. Метод прямого пошуку (Genetic Algorithm and Direct Search Toolbox)

Метод прямого пошуку для вирішення завдань оптимізації – це такий метод, в якому не використовується ніяка інформація про градієнт цільової функції. В протилежність звичайним методам пошуку, в яких для пошуку самої точки мінімуму використовується інформація про градієнт цільової функції або про похідні різного порядку, в алгоритмі методу прямого пошуку аналізується певний набір точок навколо поточної точки. Причому вишукується така точка, в якій значення цільової функції менше, ніж значення в поточній точці.

Методи прямого пошуку для вирішення задач оптимізації можна використовувати тоді, коли відсутня будь-яка інформація щодо диференційованості цільової функції або для випадку переривчастої функції.

У алгоритмах безпосереднього пошуку здійснюється комп'ютеризований розрахунок такої послідовності точок, яка послідовно сходиться до точки оптимуму. На кожному кроці в цьому алгоритмі здійснюється пошук деякого набору точок – околу поточної точки – точки, яка є результатом розрахунку залежно від попереднього кроку вибраного алгоритму. У цьому алгоритмі такий окіл формується шляхом складання поточної точки з деяким скалярним множником з фіксованого набору векторів, що називається шаблоном. Якщо цей алгоритм виходить на деяку точку в околі, в якому відзначається покрашення цільової функції в порівнянні з поточною точкою, тоді ця точка приймає статус поточної точки для наступного кроку вибраного алгоритму.

Базовим є алгоритм безпосереднього пошуку (GPS), його модифікацією – алгоритм сіткового адаптивного пошуку (MADS). У алгоритмі GPS використовуються вектори з фіксованими напрямами, а в алгоритмі MADS для утворення початкового осередку використовується стохастичних відбір векторів.

Для запуску виконання команди в методі безпосереднього пошуку для завдань без обмежень за допомогою командного рядка слід виконати команду **patternsearch** з наступним синтаксисом:

```
[x fval] = patternsearch(@objfun, x0)
```

де

- @objfun – описувач цільової функції;
- x0 – стартова точка методу прямого пошуку.

Результати обчислень можна отримувати як:

- fval – кінцеве значення цільової функції;
- x – точка, в якій досягнуте кінцеве значення.

Розглянемо приклад використання методу безпосереднього пошуку для знаходження точки мінімуму заданої функції.

Як приклад використовується цільова функція ps_example, що включена у тулбокс Genetic Algorithms and Direct Search. Саму функцію можна переглянути за допомогою команди

```
>> type ps_example
```

На рис. 22 представлено графік цієї функції:

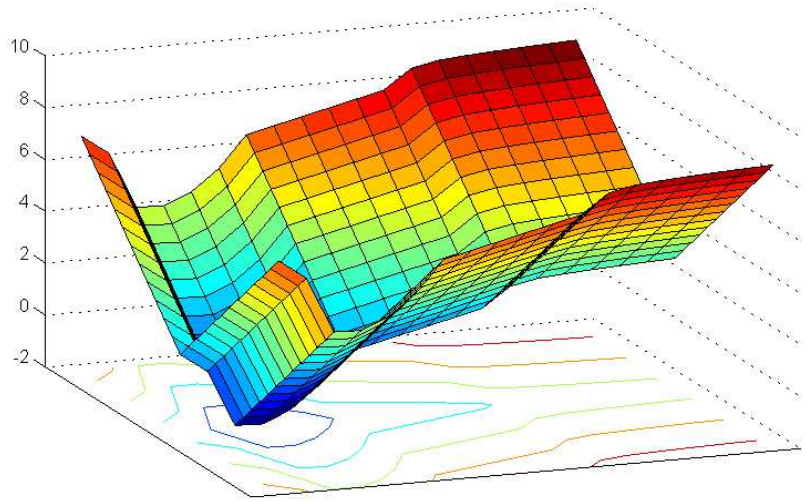


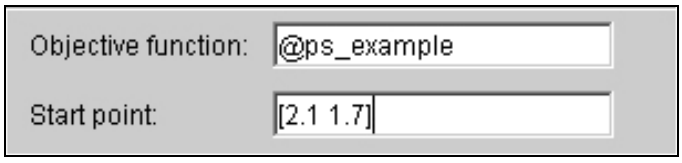
Рис. 22. Тривимірний графік функції ps_example

Для пошуку мінімуму функції ps_example слід виконати наступні команди:

1. Ввести

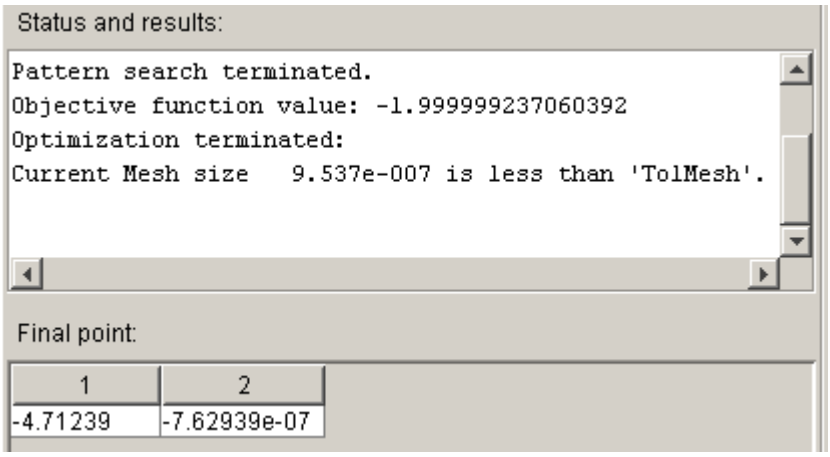
```
>> psearchtool
```

2. Відкрити інструментарій **Pattern Search**.
3. У полі **Objective function** інструментарію **Pattern Search Tool** ввести @ps_example.
4. У полі **Start point** набрати [2.1 1.7].



5. Поле панелі **Constraints** можна залишити без змін, оскільки обмеження, що накладаються, в цьому завданні відсутні.
6. Для виконання програми прямого пошуку слід виконати команду **Start**.

На панелі **Status and Results** будуть відображені результати виконання програми безпосереднього пошуку мінімуму для цієї функції.

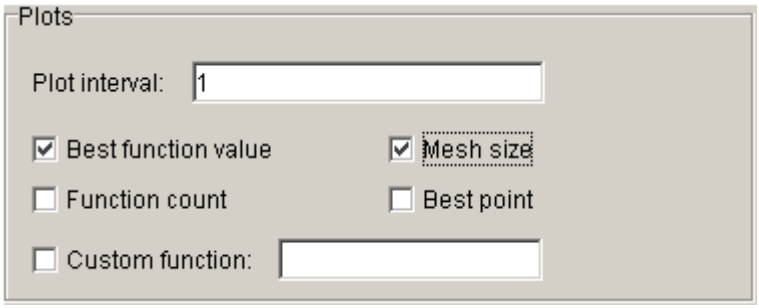


Мінімум функції приблизно рівний –2. На панелі **Final point** відображуються параметри знайденої точки мінімуму.

Для того, щоб переглянути характеристики результатів виконання програми безпосереднього пошуку можна побудувати графік найкращих значень функції і розмірів околу на кожній ітерації. Для цього на панелі **Plots** вводяться наступні параметри:

`Best function value`

`Mesh size`



Після цього за допомогою кнопки **Start** запускається програма безпосереднього пошуку. Після цього формуються графіки (рис. 23).

На верхньому графіку для кожної ітерації представлені значення цільової функції в найкращій точці. Як правило, значення цільової функції досить швидко покращуються на ранніх ітераціях і далі, по мірі наближення до оптимального значення, відбувається їх вирівнювання.

На нижньому графіку представлений розмір околу для кожної ітерації. Розмір околу збільшується після кожної успішної ітерації і зменшується після кожної неуспішної ітерації.

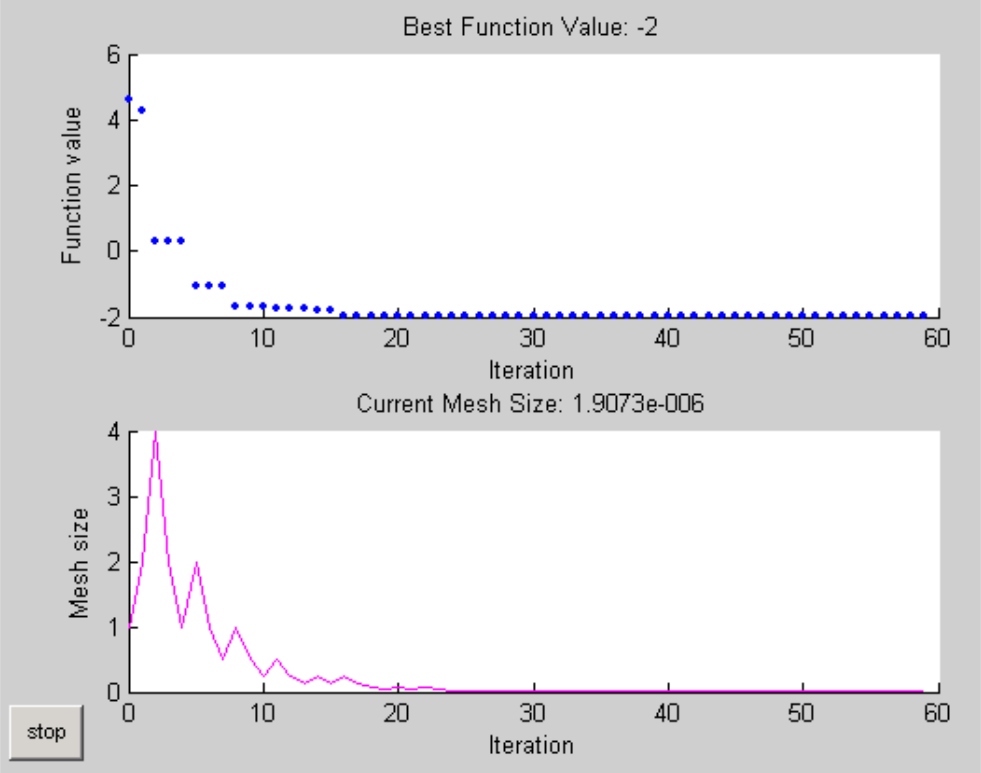


Рис. 23. Графік найкращих значень функції і розмірів околу на кожній ітерації

Контрольні запитання

1. Пояснии призначення та принцип алгоритмів прямого пошуку.

2. Яке призначення функції patternsearch з пакету Genetic Algorithm and Direct Search Toolbox?

3. Який синтаксис функції patternsearch, зміст вхідних і вихідних параметрів?

4. Для чого необхідно створювати функції, які зберігаються у М-файлах при застосуванні функції patternsearch?

5. У яке поле при використанні інструментарію psearchtool необхідно ввести назву М-файлу з цільовою функцією?

15. Основні відомості щодо системи MathCAD

Пакет MathCad фирми MathSoft, Inc. на даний час широко застосовується для рішення науково-технічних та інженерних задач.

Програма Mathcad дозволяє моделювати нуково-технічні розрахунки у формі, близькій до загальноприйнятного математичного вигляду.

Документ програми Mathcad зветься робочим листом. Він містить об’єкти: формули, графічні області і текстові блоки. У ході розрахунків формули і графічні області обробляються послідовно, зліва направо і зверху вниз, а текстові блоки ігноруються.

Блоки, у яких готуються операції, повинні розташовуватися перед блоками, що виконують відповідні операції.

Формули

Формули – це основні об’єкти робочого листа. За умовчанням новий об’єкт є формулою.

Щоб почати введення формули, треба встановити хрестоподібний курсор, розташований спочатку в лівому верхньому кутку робочого листа, в потрібне місце і почати введення. Переміщення курсору можливо як мишею, так і клавішамі управління курсором.

При цьому створюється область формули, в якій з'являється кутковий курсор, що охоплює поточний елемент формули, наприклад, ім'я змінної (функції) або число.

Якщо формула закінчується знаком "=", Mathcad відобразіть результат обчислення.

Будівельним блоком для побудови виразів є шаблон операції. Так, наприклад, при вводі бінарного оператора по другий бок знаку операції автоматично з’являється заповнювач у вигляді чорного прямокутника.

З виділеними кутовим курсором частинами виразів можна здійснювати стандартні операції: Вирізати/Копиювати/Вставити.

У це місце вводять черговий операнд. Шаблони можна вводити з клавіатури (що значно прискорює роботу) або за допомогою палітр операцій (панелей Інструментів, Toolbar).

Можна сконструювати дуже складні вирази, просто вкладаючи шаблони один в іншій.

Приклад. Обчислити значення виразу $\frac{2+3}{4\cdot\sqrt{7}}$.

№	Дія	На екрані
1	2+3	$2+3\|$
2	<пропуск>	$2+\underline{3}\|$
3	/	$\frac{2+3}{\blacksquare\|}$
4	4	$\frac{2+3}{4\|}$
5	*	$\frac{2+3}{4\cdot\blacksquare\|}$
6	\	$\frac{2+3}{4\cdot\sqrt{\blacksquare}\|}$
7	7	$\frac{2+3}{4\cdot\sqrt{7}\|}$
8	<пропуск>	$\frac{2+3}{4\cdot\sqrt{7}\|}$
9	<пропуск>	$\frac{2+3}{4\cdot\sqrt{7}\|}$
10	=	$\frac{2+3}{4\cdot\sqrt{7}}=0.472\ \blacksquare$

Необхідно звернути увагу на те, що знак множення набирається зірочкой (*), а відображається крапкою (·).

Числові константи

Числа у природній формі (123.45) записується як є, для розділення цілої і дрібної частини використовується крапка. Числа у експоненціальному (науковому) форматі (1.2345·102) вводяться наступним чином: 1.2345*10^2.

У системі доступні такі вбудовані константи:

Константа	**Значення**	**Спосіб введення**
∞	нескінченність	Ctrl+Shift+z
e	число e	e
π	число π	Ctrl+Shift+p
i	$\sqrt{-1}$	1i
j	$\sqrt{-1}$	1j
%	0.01	%
deg	множник для переведенняч у радіани	*deg*

Змінні

Імена змінних можуть складатися з латинських і грецьких букв, цифр, символів відсотка (%), апострофа (') і підкреслення (_), повинні починатися з букви.

Змінна може мати індекс, значення, що пояснює її. Для набору індексу слід натиснути точку (.). Наприклад, для введення імені змінною x_{min} треба натиснути клавіші <x><.><min> (після натиснення точки курсор введення зміститься вниз).

Для введення грецьких букв MathCad застосовується панель "Greek".

Інший спосіб – після введення латинського еквівалентц грецької букви (наприклад, "a" для "α"), натиснути Ctrl - G.

Перед використанням змінної їй необхідно присвоїти значення. Для присвоювання використовується позначення ": =" (вводиться натисканням двокрапки (":") або з панелі "Calculator").

Розлянемо приклади.

У MathCad	Дія
x:=3.62	Змінній x присвоюється значення 3.62
y:=x	Змінній y присвоюється значення змінної x
$y:=x^2 - 1$	Змінній y присвоюється значення виразу
y=	Знак "рівності" без двокрапки виводить результат

Текст

Текстовий блок сприймається системою MathCad як коментар. Він використовується тільки для пояснень і на порядок вичислення не впливає. Існує три способи вставки в документ текстового блоку:

1. Через меню Insert/Text Region;

2. За допомогою гарячої клавіші (") (подвійна лапка);

3. Система вважає, що користувач вводить текст, якщо при введенні натиснутий <Пропуск>.

Змінні-діапазони

Для виконання повторюваних обчислень в пакеті Mathcad використовуються змінні-діапазони (ранжирувані змінні, range).

Змінна-діапазон містить послідовність значень (арифметичну прогресію) і задається вказівкою першого, другого і граничного значень, наприклад:

```
x:=2,3..5
x=
2
3
4
5
```

Для введення символу "дві точки" ("..") використовується спеціальна клавіша в інтерфейсі системи, "крапка з комою" (";") або кнопка на панелі інструментів "Calculator". Якщо у виразі присутня змінна-діапазон, MathCad обчислює вираз стільки разів, скільки значень діапазон містить.

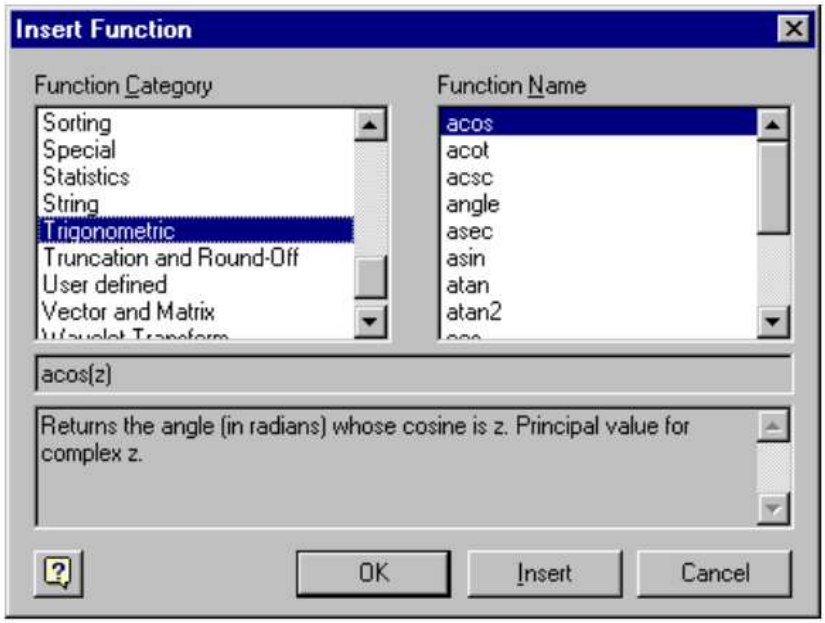
Функції у Mathcad

Система Mathcad охоплює декілька сотень вбудованих функцій; крім того, існує можливість створення власних (визначених користувачем) функцій. Є три способи введення стандартних функцій Mathcad:

1. Імена найбільш функцій, що найчастіше застосовуються (sin x, x і т. д.), можна вводити натисненням кнопок в панелі "Calculator".

2. Якщо написання імені функції відоме, можна ввести її з клавіатури, наприклад, sin(x).

3. Можна ввести функцію (наприклад, arccos(x)), скориставшись вікном вставки функції (меню Insert/ Function, поєднання клавіш Ctrl + E або кнопка на панелі інструментів).



Вікно вставки функцій дозволяє здійснити пошук функції за назвою, групі (категорії) і відображає коротку довідку по виділеній функції.

Категорія All містить всі функції (у алфавітному порядку).

Аргументом функції може бути як звичайна змінна, так і змінна-діапазон; у останньому випадку обчислення функції буде виконано для кожного значення із діапазону.

Ряд функцій приймає як аргументи вектори, матриці і навіть імена функцій.

Графіки у Mathcad

Графіки – це зручний засіб аналізу цифрової інформації. Система Mathcad надає можливість швидкої і зручної побудови графіків.

У системі є сім видів графіків:

1. Графік у декартовых координатах (поєднання клавіш Shift-2).
2. Графік у полярних координатах.
3. Поверхняь.
4. Линії рівня.
5. Об’ємна гістограма.
6. Об’ємний точковий графік.
7. Векторне поле.

Для побудови кожного типу графіка вставляється відповідний шаблон, потім поля шаблону заповнюються. Шаблони графіків можна вставити:

1. За допомогою меню "Insert / Graph".
2. З панелі інструментів "Graph".
3. За допомогою гарячих клавіш.

Приклад. Необхідно побудувати контурний графік, що задовольняє рівнянню:

$$f(x,y) = x^2 + y^2 + xy + 2$$

Для рішення цієї задачи (рис. 23) необхідно задати функцію:

$f(x,y):=x^2+y^2+x\cdot y+2,$

вибрати графічний режим із меню: Insert\Graph\Countor Plot, після чого ввести назву функції (у нашому випадку – *f*) у визначене місце під графіком.

Рішенння рівнянь

Рішення рівнянь і систем – це одна з найчастіших у інженерній практиці задач. З самого початку система MathCad була орієнтована на отримання чисельного результату, і для чисельного вирішення подібних завдань передбачено такі засоби.

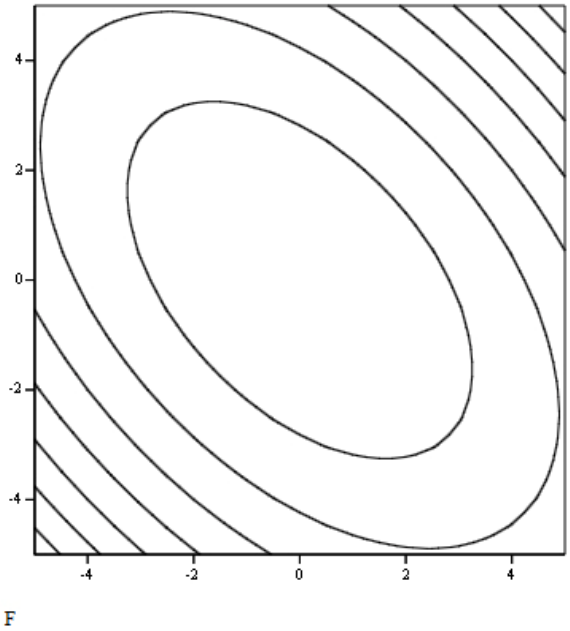


Рис. 23. Контурний графік функції $f(x, y) = x^2 + y^2 + xy + 2$

Функція root призначена для пошуку кореня функції однієї змінної (точніше, пошуку кореня за однією змінною).

У випадку декількох змінних одна змінна змінюється, значення інших повинні бути фіксовані. Форма запису:

```
root(f(x),x)
```

або

```
root(f(x),x, xmin, xmax)
```

Тут: f (x) – функція, нуль якої ми шукаємо (може бути виразом), x - змінна, по якій шукається корінь, [xmin, xmax] - інтервал пошуку. У разі, коли інтервал не заданий, необхідно встановити початкове значення для пошуку кореня.

Приклад. Необхідно знайти коріння функції cos(x) на інтервалі [3, 7].

а) пошук з початкової точки:

```
g(x):=cos(x)
x:=5
root(g(x),x)= 4.712
```

б) пошук на інтервалі

```
root(cos(x),3,7)=6.285
```

Контрольні запитання

1. Яке призначення системи MathCAD?

2. Що має бути виведено на екран, якщо за функцією з параметрами ввести знак "="?

3. Як визначити коріння аналітичної функції в системі MathCAD?

4. Як визначити змінні-діапазони у системі MathCAD?

5. Як за допомогою інтерфейса системи MathCAD вивести двох- і трохмірні графіки?

16. Рішення оптимізаційних задач з використанням системи MathCAD

Рішення оптимізаційних задач без умов обмежень

Для рішення оптимізаційних задач без умов обмежень застосовуються дві функції системи MathCAD:

- Maximize(f,<список параметрів>) – обчислення точки максимуму;
- Minimize(f,<список параметрів>) – обчислення точки мінімуму,

де f – им'я функціонала, , визначеного для звернення до функції, екстремум якої необхідно знайти; <список параметрів> – містить перелік (через кому) імен параметрів, відносно яких вирішується оптимізаційна задача.

Перед зверненням до функцій *Maximize, Minimize* (імена яких починаються з прописних літер) треба обов'язково задати початкові значення параметрів оптимізації.

Приклад 1. Нехай задано функціонал:

$$F(x, y) = x^2 + y^2 - \sin(x)\cos(y).$$

Необхідно визначити значення x, y при яких $F(x, y)$ досягає мінімального значення.

У середовищі системи MathCAD визначемо функціонал:

$$F(x, y) := x^2 + y^2 - \sin(x) \cdot \cos(y)$$

Вибір стартової точки:

$$x := 1$$
$$y := 1$$

Виклик функції мінімізації:

$$\begin{pmatrix} x \\ y \end{pmatrix} := Minimize(F, x, y)$$

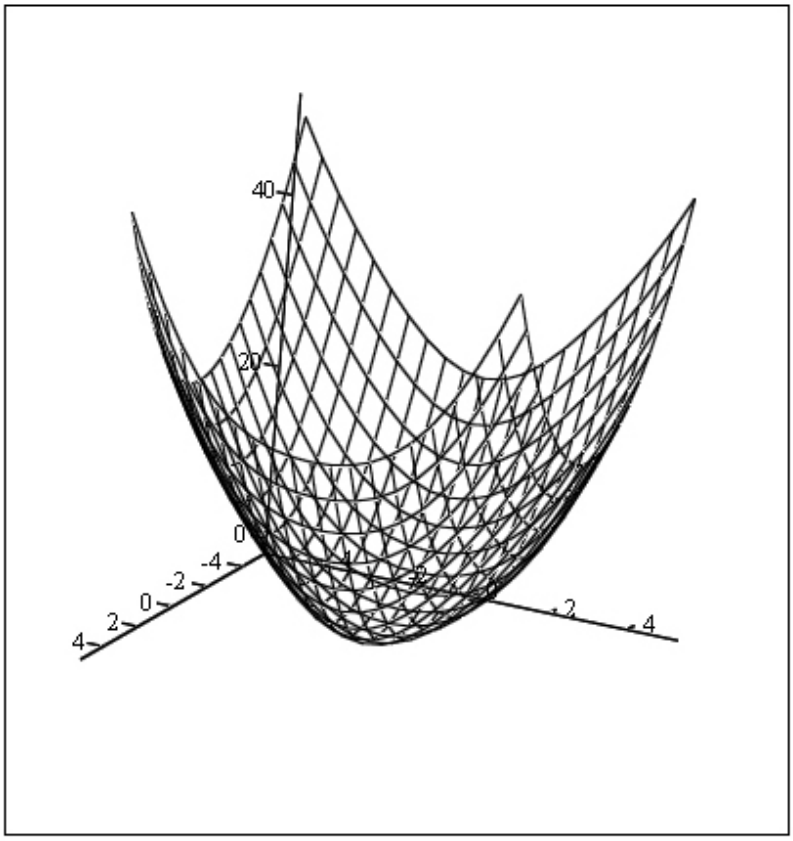
Отримуємо рішення:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0.45 \\ -2.878 \times 10^{-13} \end{pmatrix}$$

$$F(x, y) = -0.232$$

Для графічної ілюстрації цього прикладу (рис. 24) треба вибрати графічний режим (побудову тривимірної поверхні) шляхом вибору із меню:

Insert\Graph\Surface Plot, після чого ввести назву функції (у нашому випадку – F) у визначене місце під графіком.



F

Рис. 24. Графічна ілюстрація функції у середовищі системи MathCAD

Приклад 2. Нехай задано функціонал:

$$F(x, y) = e^{-4x^2-8y^2+2x+2y}$$

Необхідно визначити значення x,y, при яких F(x,y) досягає максимального значення.

В середовищі MathCAD визначемо:

$F(x, y) := \exp(-4 \cdot x^2 - 8 \cdot y^2 + 2 \cdot x + 2 \cdot y)$

Вибір стартової точки:

$x := 1$
$y := 1$

Виклик функції мінімізації:

$$\begin{pmatrix} x \\ y \end{pmatrix} := Minimize(F, x, y)$$

Рішення:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0.25 \\ 0.125 \end{pmatrix}$$

$$F(x, y) = 1.455$$

Графічну ілюстрацію цього прикладу наведено на рис. 25.

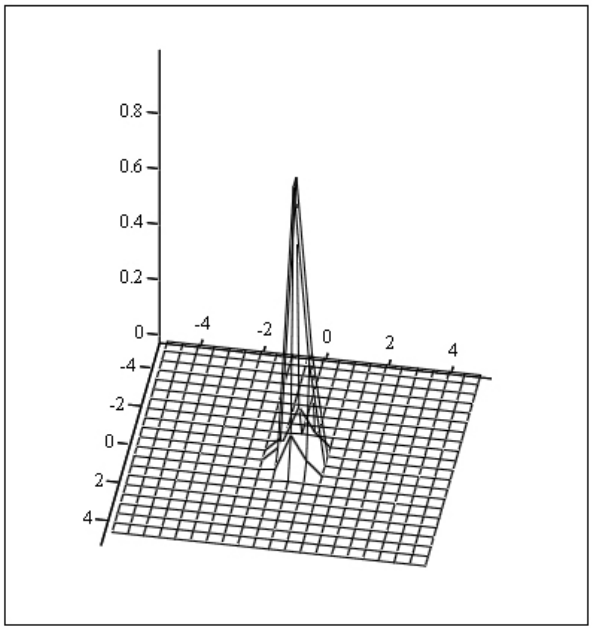


Рис. 25. Графічна ілюстрація функції (приклад 2)

Рішення оптимізаційних задач з умовами обмежень

Для рішення оптимізаційних задач з умовами обмежень у системі MathCAD застосовуються ті ж самі функції Maximize, Minimize, але вони входять вже до блоку рішення Given і перед ними розміщуються обмеження у вигляді рівностей або нерівностей, які визначають припустиму область значень параметрів оптимізації.

Приклад 3. Нехай задано функціонал: $F(x, y) := x^2 + y^2$ і обмеження у вигляді:

$x + 2y \geq 5;$

$x \geq 0;$

$y \geq 0.$

Визначити значення x, y, що забезпечують мінімальне значення функціонала і задовольняють нерівностям.

У середовищі MathCAD визначемо:

$F(x, y) := x^2 + y^2$

Вибір стартової точки:

$x := 1$
$y := 1$

Визначемо обмеження:

Given

$x + 2 \cdot y \geq 5$

$x \geq 0$

$y \geq 0$

Виклик функції мінімізації:

$$\begin{pmatrix} x \\ y \end{pmatrix} := Minimize(F, x, y)$$

Рішення:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$F(x, y) = 5$

Графічну ілюстрацію до цього прикладу наведено на рис. 5.

Приклад 4 (задача лінійного програмування). Цех малого підприємства повинен виготовити 50 виробів трьох типів $(x_1, x_2, ..., x_3)$, і не менше за 10 одиниць виробів кожного типу. На вироби йде 4, 3 и 2 кг металу, при його загальному запасі 160 кг, а також по 5, 10 і 2 кг пластмаси, при її загальному запасі у 350 кг. Прибуток від кожного виробу становить 4, 3 і 2 грн.

Визначити, скільки виробів кожного типу необхідно виготовити для отримання максимального прибутку у межах наявних запасів металу і пластмаси.

У середовищі MathCAD визначемо цільовий функціонал:

$F(x_1, x_2, x_3) := 4 \cdot x_1 + 3 \cdot x_2 + 2 \cdot x_3$

Вибір стартової точки:

$x_1 := 10$
$x_2 := 20$
$x_3 := 20$

Визначемо обмеження:

Given

$x_1 \geq 10$

$x_2 \geq 10$

$x_3 \geq 10$

$4 \cdot x_1 + 3 \cdot x_2 + 2 \cdot x_3 \leq 160$

$5 \cdot x_1 + 10 \cdot x_2 + 2 \cdot x_3 \leq 350$

$x_1 + x_2 + x_3 \leq 50$

$x_1 + x_2 + x_3 \geq 50$

Виклик функції мінімізації:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} := Maximize(F, x_1, x_2, x_3)$$

Рішення:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 24.2 \\ 11.6 \\ 14.2 \end{pmatrix}$$

Огрублюючи, можна дати таку відповідь:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 24 \\ 12 \\ 14 \end{pmatrix}$$

Отриманий прибуток дорівнює $4 \cdot x_1 + 3 \cdot x_2 + 2 \cdot x_3 = 160$ грн.

Контрольні запитання

1. Назвати дві основні функції рішення оптимізаційних задач у системі MathCAD. Навести їх синтаксис.

2. Знайти точки мінімуму і максімуму функціонала $\varphi(x, y, z) = (\cos(x \cdot y) + \cos(y \cdot z))\sin(x \cdot y \cdot z)$.

3. Як отримати графічне зображення функції двох змінних у вигляді поверхні у системі MathCAD?

4. Для чого при рішенні оптимізаційних задач у системі MathCAD використовується блок Given?

Література

1. *Вентцель Е.С.* Исследование операций: задачи, принципы, методология. – М.: Наука, Главная редакция физико-математической литературы, 1980. – 208 с.
2. *Кетков Ю. Л., Кетков А. Ю., Шульц М. М.* MATLAB 7. Программирование численных методов. – С-Пб.: БХВ-Петербург, 2005. – 742 с.
3. *Потемкин В.Г.* Вычисления в среде MATLAB. М.: Диалог-МИФИ, 2004. – 720 с.
4. *Конюшенко В.В.* Начало работы с MATLAB. – MathWorks, Inc., 1998. – 73 с.
5. *Триус Ю.В.* Розв’язування екстремальних задач за допомогою пакету Matlab 6.5 // Науковий часопис НПУ імені М. П. Драгоманова. – Серія 2. – Комп’ютерно-орієнтовані системи навчання: Зб. наук. праць. – К.: НПУ ім. М. П. Драгоманова, 2005. – №2(9). – С. 61–79.

Додаток. Короткі відомості щодо системи MATLAB

Система MATLAB (від MATrix LABoratory) розроблена фірмою The MathWorks, Inc. (США) і є інтерактивною системою для виконання інженерних і наукових розрахунків. Система MATLAB орієнтована на роботу з масивами даних і застосовується в таких областях, як теорія управління, цифрова обробка сигналів і зображень, дослідження операцій тощо. Наведена нижче інформація є необхідною для початку роботи в системі.

Оператори

Оператор – це спеціальне позначення для певної операції над даними – операндами. Найпростішими арифметичними операторами є знаки суми +, віднімання –, множення * і ділення /. Оператори використовуються разом з операндами. Наприклад, у виразі 2 + 3 знак + є оператором складання, а числа 2 і 3 – операндами. Більшість операторів у MATLAB відноситься до матричних операцій. Наприклад, оператори множення * і ділення / обчислюють добуток і частку від ділення двох багатовимірних масивів, векторів або матриць. Існує ряд спеціальних операторів, наприклад, оператор \, який означає ділення справа наліво, а оператори .* і ./ означають, відповідно, поелементне множення і поелементне ділення масивів.

Оператори системи MATLAB діляться на три категорії:

- арифметичні оператори, що дозволяють конструювати арифметичні вирази і виконувати числові обчислення.
- оператори відносин, що дозволяють порівнювати числові операнди.
- логічні оператори, що дозволяють будувати логічні вирази.

Арифметичні оператори:

+	– додавання
-	– віднімання
*	– множення матриць
.*	– поелементне множення масивів
^	– зведення матриці до степеня
.^	– поелементне зведення до степеня
\	– ліве ділення матриць
/	– праве ділення матриць
.\	– ліве ділення масивів
./	– праве ділення масивів

При роботі з масивом чисел встановлені такі приорітети операцій:

рівень 1:	(.^), (^)
рівень 2:	(+), (-)
рівень 3:	(.*), (./), (.\), (*), (/), (\)
рівень 4:	(+), (-)
рівень 5:	(:)

Усередині кожного рівня оператори мають рівний приорітет і обчислюються в порядку дотримання зліва направо. Заданий за умовчанням порядок приорітетів може бути змінений за допомогою круглих дужок.

Арифметичні оператори системи MATLAB працюють, як правило, з масивами однакового розміру. Для векторів і прямокутних масивів обидва операнди мають бути однакового розміру, за винятком єдиного випадку, коли один з них – скаляр. Якщо один з операндів скалярний, то в системі MATLAB він розширюється до розмірів другого операнда і задана операція застосовується до кожного елементу. Така операція називається розширенням скаляра.

Оператори відношення

У системі MATLAB визначені оператори відношення :

- < менше;
- <= менше або рівно;
- > більше;
- > = більше або рівно;
- == рівно тотожно;
- ~ = не рівно.

Оператори відношення виконують поелементне порівняння двох масивів рівної розмірності. Для векторів і прямокутних масивів, обидва операнди мають бути однакового розміру, за винятком випадку коли один з них скаляр. В цьому випадку MATLAB порівнює скаляр з кожним елементом іншого операнда. Позиції, де це співвідношення істинне, набувають значення 1, де помилкове – 0.

Логічні оператори

До складу логічних операторів системи MATLAB входять наступні оператори:

&	I
|	Або
~	Ні

Логічні оператори реалізують поелементне порівняння масивів однакової розмірності. Для векторів і прямокутних масивів обидва операнди мають бути однакового розміру, за винятком випадку, коли один з них скаляр. У останньому випадку MATLAB порівнює скаляр з кожним

елементом іншого операнда. Позиції, де це співвідношення істинне, набувають значення 1, де помилкове – 0.

Функції

Функції – це унікальні імена об'єктів, які виконують певні перетворення своїх аргументів і повертають результати цих перетворень. При цьому результат обчислення функції з одним вихідним параметром підставляється на місце її виклику, що дозволяє використовувати функції в математичних виразах, наприклад функцію sin в 2*sin(pi/2). Функції в загальному випадку мають список аргументів (параметрів), що поміщаються в круглі дужки.

Матриці

- ZEROS – формування масиву нулів
- ONES – формування масиву одиниць
- EYE – формування одиничної матриці
- RAND – формування масиву елементів, розподілених за рівномірним законом
- MESHGRID – формування вузлів двовимірної і тривимірної сіток
- ":" – формування векторів і підматриць

ZEROS – формування масива нулів

Y = zeros(n)

Y = zeros(m, n)

Y = zeros(size(A))

Функція Y = zeros(n) формує масив нулів розміру n × n

Функція Y = zeros(m, n) формує масив нулів розміру m × n

Функція Y = zeros(size(A)) формує масив нулів співвимірний з масивом A

ONES – формування масива одиниць

Y = ones(n)

Y = ones(m, n)

Y = ones(size(A))

Функція Y = ones(n) формує масив одиниць розміру n × n

Функція Y = ones(m, n) формує масив одиниць розміру m × n

Функція Y = ones(size(A)) формує масив одиниць співвимірний з масивом A

EYE – формування одиничної матриці

Y = eye(n)

Y = eye(m, n)

Y = eye(size(A))

Функція Y = ones(n) формує одиничну матрицю розміру n × n

Функція Y = ones(m, n) формує одиничну матрицю розміру m × n

Функція Y = ones(size(A)) формує одиничну матрицю співвимірну з матрицею A.

RAND – формування масива елементів, розподілених за рівномірним законом

X = rand(n)	rand
X = rand(m, n)	rand('seed')
X = rand(size(A))	rand('seed', x0)

Функція X = rand(n) формує масив розміру n × n, елементами якого є випадкові величини, розподілені за рівномірним законом в інтервалі (0, 1).

Функція X = rand(m, n) формує масив розміру m × n, елементами якого є випадкові величини, розподілені за рівномірним законом в інтервалі (0, 1).

Функція X = rand(size(A)) формує масив співвимірний з матрицею A, елементами якого є випадкові величини, розподілені за рівномірним законом в інтервалі (0, 1).

Функція rand без аргументів формує одне випадкове число, що підкоряється рівномірному закону розподілу в інтервалі (0, 1), який змінюється при кожному наступному виклику.

Функція rand('seed') повертає поточне значення бази (початкового значення) генератора випадкових чисел.

Функція rand('seed', x0) привласнює базі (початковому значенню) генератора випадкових чисел значення x0.

Алгоритм генерації рівномірно розподілених випадкових чисел заснований на лінійному конгруентном методі. Обчислення наступного випадкового числа реалізоване згідно із співвідношенням:

$$seed = (7^7 \text{ x } seed) \ (\text{mod}(2^{31} - 1).$$

MESHGRID – формування вузлів двовимірної і тривимірної сіток

[X, Y] = meshgrid(x, y)

[X, Y] = meshgrid(x)

[X, Y, Z] = meshgrid(x, y, z)

Функція [X, Y] = meshgrid(x, y) формує масиви X і Y, які визначають координати вузлів прямокутника, що задається векторами x і y. Цей прямокутник задає область визначення функції від двох змінних, яку можна побудувати у вигляді 3D-поверхности.

Функція [X, Y] = meshgrid(x) є скороченою формою запису функції [X, Y] = meshgrid(x, x).

Функція [X, Y, Z] = meshgrid(x, y, z) формує масиви X, Y і Z, які визначають координати вузлів паралелепіпеда, що задається векторами x, y і z.

«:» – формування векторів і підматриць

j : k	A(i1 : i2, j1 : j2)
j : i : k	A(n1 : n2)

Оператор ":" застосовується для формування векторів і матриць або для виділення з них підвекторів, підматриць, підблоків масиву.

Виделення підблоків

A(i1: i2, j1: j2) - виділення підблоку масиву A з рядками i1 : i2 і стовпцями j1 : j2.

A(i,:) - i -й рядок масиву A;

A(:, j) - j -й стовпець масиву A.

Оскільки в мові MATLAB елементи масиву впорядковані по стовпцях, то допустимі оператори вигляду A(n_1: n_2), які виділяють пронумеровані елементи з номера n_1 до номера n_2. Оператор A(:) записує усі елементи масиву A у вигляді стовпця.

Основні операції над масивами

- SUM, CUMSUM – підсумовування елементів масиву
- PROD, CUMPROD – добуток елементів масиву
- SORT – сортування елементів масиву за збільшенням
- MAX – визначення максимальних елементів масиву
- MIN – визначення мінімальних елементів масиву
- MEAN – визначення середніх значень елементів масиву
- STD – визначення стандартних відхилень елементів масиву

SUM, CUMSUM – сумування элементів масиву

```
sx = sum(X)
csx = cumsum(X)
```

Функція sx = sum(X) у разі одновимірного масиву повертає суму елементів масиву; у разі двовимірного масиву – це вектор-рядок, що містить суми елементів кожного стовпця.

Функція csx = cumsum(X), крім того, повертає усі проміжні результати підсумовування.

PROD, CUMPROD – добуток елементів масиву

px = prod(X)

cpx = cumprod(X)

Функція px = prod(X) у разі одновимірного масиву повертає добуток елементів масиву; у разі двовимірного масиву - це вектор-рядок, що містить добутки елементів кожного стовпця.

Функція cpx = cumprod(X), крім того, повертає усі проміжні результати.

SORT – сортування елементів масиву за зростанням

Y = sort(X)

[Y, I] = sort(X)

Функція Y = sort(X) у разі одновимірного масиву упорядковує елементи масиву за збільшенням; у разі двовимірного масиву відбувається впорядкування елементів кожного стовпця.

MAX – визначення максимальних елементів масиву

Y = max(X)

[Y, I] = max(X)

C = max(A, B)

Функція Y= max(X) у разі одновимірного масиву повертає найбільший елемент; у разі двовимірного масиву – вектор-рядок, що містить максимальні елементи кожного стовпця. Таким чином, max(max(X)) – це найбільший елемент масиву.

Функція [Y, I] = max(X) окрім самих максимальних елементів повертає вектор-рядок індексів цих елементів в цьому стовпці.

Функція C = max(A, B) повертає масив C тих же розмірів, які мають масиви A і B, кожен елемент якого максимальний з відповідних елементів цих масивів.

MIN – визначення мінімальних елементів масиву

Y = min(X)

[Y, I] = min(X)

C = min(A, B)

Функція Y = min(X) у разі одновимірного масиву повертає найменший елемент; у разі двовимірного масиву – вектор-рядок, що містить мінімальні елементи кожного стовпця. Таким чином, min(min(X)) – це найменший елемент масиву.

Функція [Y, I] = min(X) окрім самих мінімальних елементів повертає вектор-рядок індексів цих елементів в цьому стовпці.

Функція C = min(A, B) повертає масив C тих же розмірів, які мають масиви A і B, кожен елемент якого мінімальний з відповідних елементів цих масивів.

MEAN – визначення середніх значень елементів масиву

mx = mean(X)

Функція mx = mean(X) у разі одновимірного масиву повертає середнє арифметичне елементів масиву; у разі двовимірного масиву – вектор-рядок, що містить середнє арифметичне елементів кожного стовпця. Таким чином, mean(mean(X)) – це арифметичне середнє елементів масиву, що співпадає зі значенням mean(X(:)).

STD – визначення стандартних відхилень елементів масиву

sx = std(X)

Функція sx = std(X) у разі одновимірного масиву повертає стандартне відхилення елементів масиву; у разі двовимірного масиву – вектор-рядок, що містить стандартне відхилення елементів кожного стовпця.

Математичні функції

У системі MATLAB існує велика бібліотека математичних функцій. Аргументи функцій завжди вказуються в круглих дужках після імені функції і, якщо їх більше одного, розділяються комами. Розглянемо вбудовані математичні функції системи MATLAB, які застосовуються до чисел, скалярних змінних і до масивів.

- ABS – абсолютне значення
- SIGN – обчислення знаку числа
- CEIL, FIX, FLOOR, ROUND – функції округлення
- SQRT – квадратний корінь
- EXP – експоненціальна функція
- LOG – функція натурального логарифма
- SIN – функція синуса
- COS – функція косинуса
- TAN – функція тангенса

ABS – абсолютне значення

Y = abs(X)

Для масиву дійсних чисел X функція Y = abs(X) повертає масив Y абсолютних значень елементів X.

SIGN – обчислення знаку числа

S = sign(Z)

Для масивів дійсних чисел X функція S = sign(X) повертає масив S тих самих розмірів, в якому на місці позитивного числа стоїть 1, на місці нульового – 0, на місці негативного – (–1).

CEIL, FIX, FLOOR, ROUND – функції округлення

Y = ceil(X)

Y = fix(X)

Y = floor(X)

Y = round(X)

Для масивів дійсних чисел X :

- функція Y = ceil(X) повертає значення, округлені до найближчого цілого >=X;
- функція Y = fix(X) повертає значення з усіканням дробової частини числа;
- функція Y = floor(X) повертає значення, округлені до найближчого цілого<= X;
- функція Y = round(X) повертає значення, округлені до найближчого цілого.

SQRT – квадратний корінь

V = sqrt(Z)

Функція V = sqrt(Z) обчислює квадратне коріння елементів масиву Z. Для негативних і комплексних значень результат є комплексним числом.

EXP – експоненціальна функція

V = exp(Z)

Функція V = exp(Z) обчислює експоненти значень елементів масиву Z.

LOG – функція натурального логарифма

V = log(Z)

Функція V = log(Z) обчислює натуральний логарифм значень елементів масиву Z.

SIN – функція синусу

V= sin(Z)

Функція V = sin(Z) обчислює синус від значень елементів масиву Z.

COS – функція косинусу

V = cos(Z)

Функція V = cos(Z) обчислює косинус від значень елементів масиву Z.

TAN – функция тангенсу

V = tan(Z)

Функція V = tan(Z) обчислює тангенс від значень елементів масиву Z.

Графічні команди і функції

До складу системи MATLAB входить потужна графічна підсистема, яка підтримує як засоби візуалізації двовимірної і тривимірної графіки на екрані терміналу, так і засоби презентаційної графіки.

Основні графічні функції:

- PLOT – графік в лінійному масштабі
- LOGLOG – графік в логарифмічному масштабі
- PLOT3 – побудова ліній і точок в тривимірному просторі
- MESH, MESHC, MESHZ – тривимірна сітчаста поверхня
- SURF, SURFC – затінена сітчаста поверхня
- BAR – стовпчикові діаграми
- HIST – побудова гістограми

PLOT – графік у лінійному масштабі

plot(Y)

plot(X, Y)

plot(X, Y, S)

plot(X_1, Y_1, S_1, X_2, Y_2, S_2, ...)

Команда plot(Y) будує графік елементів одновимірного масиву Y залежно від номера елемента; якщо елементи масиву у комплексні, то будується графік plot(real(Y), imag(Y)). Якщо Y – двовимірний дійсний масив, то будуються графіки для стовпців; у разі комплексних елементів їх уявні частини ігноруються.

Команда plot(X, Y) відповідає побудові звичайної функції, коли одновимірний масив x відповідає значенням аргументу, а одновимірний масив Y – значенням функції. Якщо один з масивів X або Y, або обидва двовимірні, реалізуються наступні побудови:

- якщо масив Y двовимірний, а масив X одновимірний, то будуються графіки для стовпців масиву Y залежно від елементів вектора X;

- якщо двовимірним є масив X, а масив Y одновимірний, то будуються графіки стовпців масиву X залежно від елементів вектора Y;
- якщо обидва масиви X і Y двовимірні, то будуються залежності стовпців масиву Y від стовпців масиву X.

Команда plot(x, y, s) дозволяє виділити графік функції, вказавши спосіб відображення лінії, точок, колір ліній і точок за допомогою строкової змінної s, яка може включати до трьох символів з наступної таблиці:

Тип лінії		**Тип точки**		**Колір**	
Неперервна	-	Точка	.	Жовтий	y
Штрихова	--	Плюс	+	Фіолетовий	m
Подвійний пунктир	:	Зірочка	*	Блакитний	c
Штрих-пунктир	-.	Коло	o	Червоний	r
		Хрестик	x	Зелений	g
				Синій	b
				Білий	w
				Чорний	k

Якщо колір лінії не вказаний, він вибирається за умовчанням з шести перших кольорів, з жовтого до синього, повторюючись циклічно.

Команда plot(x_1, y_1, s_1, x_2, y_2, s_2, ...) дозволяє об'єднати на одному графіку декілька функцій $y_1(x_1)$, $y_2(x_2)$, …, визначивши для кожної з них свій спосіб відображення.

Звернення до команд plot вигляду plot(x, y, s1, x, y, s2) дозволяє для графіка y(x) визначити додаткові властивості, для визначення яких застосування однієї строкової змінної s1 недостатньо, наприклад, при завданні різних кольорів для лінії і для точок на ній.

LOGLOG – графік у логарифмічному масштабі

loglog(x, y)

loglog(x, y, s)

loglog(x1, y1, s1, x2, y2, s2, ...)

Команди loglog(...) рівносильні функціям plot, за винятком того, що вони використовують по обох осях логарифмічний масштаб замість лінійного.

PLOT3 – побудова ліній і точок у тривимірному просторі

plot3(x, y, z)

plot3(X, Y, Z)

plot3(x, y, z, s)

plot3(x1, y1, z1, s1, x2, y2, z2, s2, ...)

Команди plot3(...) є тривимірними аналогами функції plot(…).

Команда plot3(x, y, z), де x, y, z – одновимірні масиви однакового розміру, будує точки з координатами x(i), y(i), z(i) і сполучає їх прямими лініями.

Команда plot3(X, Y, Z), де X, Y, Z – двовимірні масиви однакового розміру, будує точки з координатами x(i,:), y(i,:), z(i,:) для кожного стовпчика і сполучає їх прямими лініями.

Команда plot3(x, y, z, s) дозволяє виділити графік функції z(x, y), вказавши спосіб відображення лінії, спосіб відображення точок, колір ліній і точок за допомогою строкової змінної s, яка може включати до трьох символів з тієї ж таблиці, що і для функції plot.

Команда plot3(x1, y1, z1, s1, x2, y2, z2, s2, ..) дозволяє об'єднати на одному графіку декілька функцій z1(x1, y1), z2(x2, y2), ..., визначивши для кожної з них свій спосіб відображення.

MESH, MESHC, MESHZ – сітчата поверхня у тривимірному просторі

mesh(X, Y, Z)	meshc(X, Y, Z)	meshz(X, Y, Z)
mesh(Z)	meshc(Z)	meshz(Z)

Команда mesh(X, Y, Z) дозволяє виводити на екран сітчасту поверхню для значень масиву Z, визначених на множині значень масивів X і Y. Колір вузлів пропорційний висоті поверхні.

Група команд meshc(...) додатково до тривимірних поверхонь будує проекцію ліній постійного рівня.

Група команд meshz(..) додатково до тривимірних поверхонь будує площину відліку на нульовому рівні, закриваючи поверхню, що лежить нижче за цей рівень.

SURF, SURFC – затінена сітчаста поверхня

surf(Z,C)	surfc(Z,C)
surf(X, Y, Z)	surfc(X, Y, Z)
surf(Z)	surfc(Z)

Команда surf(Z, C) використовує сітку, яка визначається одновимірними масивами x = 1, : n і y = 1 : m.

Команди surf(X, Y, Z), surf(x, y, Z), surf(Z) використовують як масив кольори C = Z, тобто колір в цьому випадку пропорційний висоті поверхні.

Група команд surfc(...) додатково до тривимірних затінених поверхонь будує проекцію ліній постійного рівня.

BAR – стовпчикові діаграми

bar(Y); bar(Y, ‘<тип лінії>‘)

bar(X,Y); bar(X,Y, ‘<тип лінії>‘)

Команда bar(Y) виводить графік елементів одновимірного масиву Y у вигляді столбцовой діаграми.

Команда bar(X, Y) виводить графік елементів масиву Y у вигляді стовпців в позиціях, визначуваних масивом X, елементи якого мають бути впорядковані в порядку зростання.

Якщо X і Y – двовимірні масиви однакових розмірів, то кожна діаграма визначається відповідною парою стовпців і вони надбудовуються одна над іншою.

Команди bar(Y, '<тип лінії>'), bar(X, Y, '<тип лінії>') дозволяють задати тип ліній, використовуваних для побудови стовпчикових діаграм, по аналогії з командою plot.

HIST – побудова гістограми

hist(y)

hist(y, n)

hist(y, x)

Команди hist(...) підраховують і відображують на графіці кількість елементів масиву y, значення яких потрапляють в заданий інтервал; для цього увесь діапазон значень y поділяється на n інтервалів (за умовчанням 10) і підраховується кількість елементів в кожному інтервалі.

Команда hist(y) виводить гістограму для 10 інтервалів.

Команда hist(y, n) виводить гістограму для n інтервалів.

Команда hist(y, x) виводить гістограму з урахуванням діапазону зміни змінної x.