Лабораторна робота № 3.

Тема: Графічна візуалізація даних у системі **MATLAB**.

Мета роботи: ознайомитись з прийомами побудови **2D** та **3D** графіків функцій від однієї або двох незалежних змінних у системі **MATLAB**.

Теоретичний мінімум

Пакет **MATLAB** підтримує як двувимірну, так і тривимірну графіку. Це дає можливість представляти (візуалізувати) у графічному вигляді явно, неявно та параметрично задані функції, багатовимірні функції і просто набори даних.

Вправа 1. Побудова простих графіків функцій. Стилі (види представлення) графіків.

Система **MATLAB** володіє широким набором засобів візуалізації для побудови графіків функцій однієї і двох змінних і відображення різних типів даних. Всі графіки виводяться в графічні вікна, що мають меню і панелями інструментів. Вид графіків визначається аргументами графічних команд і може бути змінений за допомогою інструментів графічного вікна. Важливо пам’ятати, що для побудови графіків функцій на деякій області зміни аргументів слід вміти обчислювати значення функції в точках області. Для отримання якисних графіків часто приходиться використовувати достатньо багато точок [33,с.53].

Приклад 1. Побудувати графік функції однієї змінної $f(x) = e^x \cos(x) - x^3$, де значення x вибираються з відрізку відрізку [–3, 3].

Перший крок полягає в виборі координат точок по осі абсцис. Задання вектора x, компоненти якого формуються за допомогою постійного кроку, використання двокрапки дозволяє просто вирішити цю задачу. Після цього необхідно покомпонентно обчислити значення $y(x)$ для кожного компонента вектора x і записати результат у вигляді вектора.

Тепер, для побудови графіка функції, слід використати яку-небудь з графічних функцій системи **MATLAB**. Достатньо універсальна графічна функція реалізується командою **plot**. В найпростішому випадку вона викликається з двома вхідними аргументами – парою **x** і **f** (тобто **plot** виводить залежність компонент одного вектора від компонентів іншого). Послідовність команд для побудови графіка виписана нижче:

```
> x=–3:0.05:3;
>> f=exp(x).*cos(x)–x.^3;
>> plot(x,f)
```

В результаті виконання наведених команд з'являється графічне вікно, що має назву **«Figure No.1»** і містить графік функції:

Приклад 2. Побудувати графік функції $f(x) = e^{-x}(\sin x + 0.1\sin(100\pi x))$, де значення аргументу x вибирається з відрізку $[0, 1]$.

Запрограмуємо **m**-файл **myfn** для обчислення функції $f(x)$. Шуканий **m**-файл, що містить файл-функцію, повинен починатися із заголовку, після якого записуються оператори **MATLAB**. Заголовок складається із слова **function**, після якого виписується перелік вихідних аргументів, імені файл-функції і перелік вхідних аргументів. Параметри в цих переліках розділяються комами. При формуванні функції використовуємо поелементні операції для того, щоб **m**-файл **myfn** був обрахований від відповідного вектора значень аргументу, а значення функції формувало вектор відповідних значень функції:

```
function y=myfn(x);
y=exp(–x).*(sin(x) +0.1*sin(100*pi*x));
```

Графік можна отримати двома способами.

Перший спосіб полягає в створенні вектора значень аргументу, наприклад, у вигляді дискретного проміжку з кроком 0.01, заповненні вектора значень функції і виклику команди **plot**:

```
>> x=0:0.01:1;
>> y=myfn(x);
>> plot(x,y)
```

В результаті отримуємо некоректний графік. Дійсно, при обчисленні значень функції на відрізку $[0, 1]$ з кроком 0.01 доданок $\sin(100\pi x)$ весь час перетворюється в нуль і команди **plot** будує графік іншої, а не шуканої функції $f(x)$. Невдалий вибір кроку часто приводить до втрати істотної інформації про поведінку функції.

У системі **MATLAB** існує вбудована команда **fplot** – аналог команди **plot**, але з автоматичним підбором кроку при побудові графіка. Першим вхідним параметром команди **fplot** є ім'я файл-функції (виділяється апострофами), а другим – вектор, елементи якого є границі відрізків: **fplot**('ім'я файл-функції' [a,b]).

Побудуйте тепер в новому вікні графік функції $f(x)$ за допомогою команди **fplot**:

```
>> figure
>> fplot('myfn'[0,1])
```

Розглянемо процедуру формування сітки, що використовується для побудови гріфіків функцій. Для генерації сітки передбачена команда **linspace**, що викликається від вхідного аргументу – вектору, який задає послідовність значень аргументу на осі **x**. Існує два типа запису функції:

- **x=linspace(x1,x2)** – запис формує лінійний масив розміром 1×100, початковим й кінцевим значенням якого є відповідно точки x1 та x2;
- **x=linspace(x1,x2,n)** – запис формує лінійний масив розміром 1×n, початковим й кінцевим значенням якого є відповідно точки x1 та x2.

Приклад 3. Побудувати графік функції $f = e^x \cos x - x^3$ за допомогою команди **linspace**:

```
>> x=linspace(–3,3,200);
>> f=exp(x).*cos(x)–x.^3;
>> plot(x,f)
```

Для зображення одновимірної кривої в тривимірному просторі використовується команда **plot3(x,y,z).**

Приклад 4. Побудувати криву, яка задана параметричним способом, у тривимірному просторі.

Для виконання завдання треба ввести послідновність команд:

```
>> t=0:.1:50;x=.1*t.*cos(t);y=0.2*t.*sin(t);z=0.2*t;plot3(x,y,z);grid on
```

Для більш якісної побудови графіка функції використовуються допоміжні параметри, що визначають: тип лінії, колір і маркери. Ці параметри визначаються значенням третього (після **x**, **f**) додаткового параметру функції **plot**. Цей параметр записується в апострофах, наприклад, виклик **plot(x,f,'go--')** приводить до побудови графіка зеленою штриховою лінією, яка розмічена круглими маркерами. Зверніть увагу, що перший елемент абсциси, вектора x, задає значення параметрів маркерів. Всього в додатковому аргументі може бути заповнено три позиції, що визначають вид кольору, тип маркерів і стиль лінії [33,с.60]. Позначення для них наведені в табл. 3.1. Послідовність запису параметрів, що визначають стиль графіка, може бути довільною і допустимо вказувати або один, або два параметри, наприклад, колір і тип маркерів.

Рекомендуємо проаналізувати результати виконання наступних команд:

```
>> plot(x,f,'m:'), plot(x,f,'k;'), plot(x,f,'x')
```

Команда **plot** має достатньо універсальний інтерфейс, вона, зокрема, дозволяє відображати графіки декількох функцій в одних координатних вісях.

Приклад 5. Вивести на екран два графіка: $f = e^x \cos x - x^3$ і $g(x) = e^{-x^2} \cos 4\pi x$ на відрізку [–1, 1].

Спочатку необхідно обчислити значення $g(x)$, а потім викликати команду **plot**, після якої в дужках вказуються через кому пари **x**, **f** та **x**, **g**, **i**, при бажанні, стиль кожної з ліній графіків функцій (див. табл. 3.1):

```
>> x= –1:0.005:1;
>> f=exp(x).*cos(x)-x.^3;
>> g=exp(–x.^2).*cos(4*pi*x);
>> plot(x,f,’g–‘,x,g,’k–‘)
```

Таблиця 3.1 – Стилі графків

Колір лінії	
Жовтий	Y
Фіолетовий	M
Блакитний	C
Червоний	R
Зелений	G
Синій	B
Білий	W
Чорний	K
Тип точки	
.	Точка
0	Коло
X	Хрест
+	Плюс
*	Зырочка
S	Квадрат
D	Ромб
V	Трикутник (вниз)
A	Трикутник (вверх)
<	Трикутник (вліво)
>	Трикутник (вправо)
P	Пятикутник
H	Шостикутник
Тип лінії	
-	Суцільна
;	Подвіцнийпунктир
-.	Штрих-пунктир
--	Штрихова

Допускається побудова довільного числа графіків функцій, стилі всіх ліній можуть бути різними. Крім того, області побудови кожної з функцій не обов'язково повинні співпадати, але тоді слід використовувати різні вектори для визначення значень аргументів і обчислювати значення функцій від відповідних векторів.

Приклад 6. Побудувати графік кусочно-заданої функції

$$y(x)=\begin{cases}\sin x & -4\pi < x \le -\pi \\ 3(x/\pi+1)^2 & -\pi < x \le 0 \\ 3e^{-x} & 0 < x \le 5\end{cases}$$

Для цього достатньо виконати наступну послідовність команд:

```
>> x1=[–4*pi:pi/10:–pi]; y1=sin(x1);
>> x2=[–pi:pi/30:0]; y2=3*(x2/pi+1).^2;
>> x3=[0:0.02:5]; y3=3*exp(–x3);
>> plot(x1,y1,x2,y2,x3,y3)
```

Зауважимо, що графіки гілок функції відображаються різними кольорами. Можна було поступити і поіншому, а саме: після формування аргументів **x1**, **y1**, **x2**, **y2**, **x3** і **y3** сформувати вектор **x** для значень аргументу **x** і вектор **y** для значень $y(x)$ і побудувати залежність **y** від **x**:

```
>> x=[x1 x2 x3];
>> y=[y1 y2 y3];
>> plot(x,y)
```

Приклад 7. Побудувати графіки трьох функцій: $\sin x$, $\cos x$, $\frac{\sin x}{x}$.

Перш за все відзначимо, що дані функції можуть бути позначені змінними, що не вказують явно на вид аргументу **y** як функціїї від **x**, тобто у вигляді **y(x)**:

```
>> y1=sin(x); y2=cos(x); y3=sin(x)/x;
```

Як булопоказано раніше, можна використовувати одну з можливих форм команди побудови графіків. Загальний вигляд команди для побудови графіку в одному вікні є наступним **plot(x1,y1,x2,y2,x3,y3)**, де **xl, x2**, **x3**„ – вектори значень аргументів функцій (в нашому випадку всі вони – **x**), а **y1, y2, y3**... – вектори значень функцій. В випадку, що розглядається, для побудови графіків вихідних функцій маємо:

```
>> x=[–3*pi:pi/20:3*pi];
>> y1=sin(x); y2=cos(x); y3=sin(x)./x;
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this
warning.)
>> plot(x,y1,x,y2,x,y3)
```

Треба звернути увагу на те, що у запису значень вектора **y3** присутня операція поелементного ділiння **«./»**. Якщо виконати команду матричного ділення без знака поелементного ділiння, то ніякого графіка не з’явиться взагалі. Не виключений навіть збій роботи програми. Причина цього казусу – при обчисленні функції **y3=sin(x)/x**, якщо **x** є масивом (вектором), і не можна використовувати оператор матричного ділення «**/»**.

Зверніть увагу, що хоча в цьому випадку команди системи **MATLAB** побудують графіки всіх трьох функцій, у вікні командного режиму з'явиться попередження про ділiння на нуль, коли **x=0**. Це вказує на те, що команда **plot** не «знає», що невизначеність **sin(x)/x=0/0** є усуненою і границя її , коли **x** прямує до нуля, дорівнює 1. Ця недоречність має місце практично в усіх системах для чисельних обчислень.

Прийміть до уваги, що система **MATLAB** має засоби для побудови графіків таких функцій, як **sin(x)/x**, для яких мають місцеусунені невизначеності. В таких випадках використовується інша графічна команда: **fplot('f(x)', [xmin xmax],n)**, де **f(x)** –ім’я функції, **[xmin xmax]** – інтервал

побудови графіку, **n** – кількість частин, на які розбиваєтьсяч інтервал (за замовчуванням n=25).

Ця команда дозволяє будувати функцію, задану в символьному вигляді, в інтервалі зміни аргументу **x** від значення **xmin** до значення **xmax** без фіксованого кроку зміни **x**. Не зважаючи на те, що в процесі обчислень попередження про помилку (розподіл на 0) виводяться, але графік будується правильно, при **x=0 sinx/x=l**. Переконаайтесь в правильності результату:

```
>> fplot('tan(x)./x',[–20 20]).
```

Відмітимо, що у системі **MATLAB** є функція **comet**, яка дозволяє простежити за рухом точки вздовш траєкторії параметрично заданої лінії. Виклик команди **comet(x,y)** (для трьохвимірної графікі – **comet3(x,y,t))** приводить до появи графічного вікна, на вісях якого зображується переміщення точки у вигляді руху комети з хвостом. Управління швидкістю руху здійснюється зміною кроку при визначенні вектора значень параметру.

Приклад 8. Побудувати переміщення точки, якщо її рух задається параметричним способом.

Для виконання поставленої задачі слід виконати наступні операції:

```
>> t = –10*pi:pi/500:10*pi; y=(sin(t).^3).*cos(t);
>> x=(cos(t).^3).*sin(t); comet3(x,y,t)
```

В системі **MATLAB** існують спеціальні графічнікоманди, які призначені для відображення графіків в логарифмічному і частково логарифмічному масштабах:

- **loglog** – використовується логарифмічний масштаб по обох вісях;
- **semilogx** – використовується логарифмічний масштаб тільки по вісі абсцис;
- **semilogy** – використовується логарифмічний масштаб тільки по вісі ординат.

Приклад 9. Побудувати графік функції $f(x) = \cos\ln x$ у логарифмічному масштабі по вісі Ox.

Звернемо увагу на те, що вхідні параметри графічної команди задаються так само, як і при використовуванні команди **plot.** Перелік необхідних команд наведений нижче:

```
>> x=0.1:0.001:1000;
>> y=cos(log(x));
>> semilogx(x,y);grid on
```

Якщо дві функції мають непорівнянні значення і їх відображення в одному вікні неможливо, то для порівняння цих функцій треба застосовувати команду **plotyy**. Команда **plotyy** викликається від двох пар вхідних параметрів (векторів) і приводить до появи двох ліній графіків, кожному з яких відповідають відповідні вісі координат.

Приклад 10. Побудувати графікі функцій $y_1(x) = x^3$, $y_2(x) = \sin 4x$. Завдякі використанню команди **plotyy** побудова графіків цих функцій зводиться запису стандарного набору команд:

```
>> x=0:0.1:4;
>> y1=x.^3;
>> y2=sin(4*x);
>> plotyy(x,y1,x,y2);
>> grid on
```

Вправа 3. Оформлення графіків функцій.

Для оформлення графіків в **MATLAB (**формування підписів**,** завдання координатної сітки, тощо**)** існують спеціальні команди і функції [33, с.61].

Сітка на координатній площині наноситься командою **grid on**, а знищується за допомогою команди **grid off**. Крок сітки у системі **MATLAB** відповідає цілим одиницям виміру, щоб полегшити сприймання графіка.

Заголовок розміщується в графічному вікні за допомогою команди **title**, вхідним параметром якої є рядок, укладений в апострофах: **title**('Результати експерименту').

За наявності декількох графіків вимагається розташувати легенду, яка формується командою **legend**. Написи легенди, укладені в апострофи, вказуються у вхідних параметрах команди **legend**, число їх повинне співпадати з числом ліній графіків. Крім того, останній додатковий вхідний параметр визначає положення легенди на малюнку:

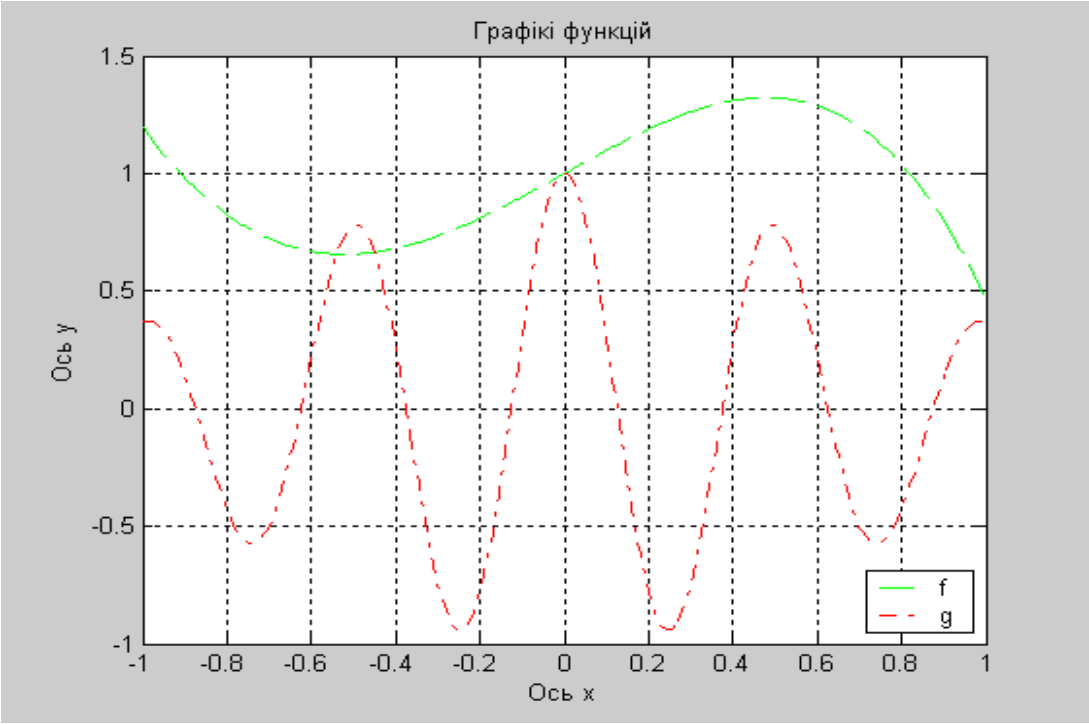
- –1 – вибирається положення легенди зовні малюнку в правому верхньому кутку графічного вікна;
- 0 – вибирається краще положення в межах малюнку так, щоб як можна меньше перекривалися самі графіки;
- 1 – вибирається положення легенди у верхньому правому кутку малюнку (це положення використовується за замовчуванням);
- 2 – вибирається положення легенди у верхньому лівому кутку малюнку;
- 3 – вибирається положення легенди у нижньому лівому кутку малюнку;
- 4 – вибирається положення легенди у нижньому правому кутку малюнку.

Команди **xlabel** і **ylabel** призначені для формування підписів до вісів, їх вхідні параметри також записуються в апострофах.

Приклад 1. Побудувати графікі функцій $f(x) = e^x \cos x = x^3$, $g(x) = e^{-x^2} \cos 4\pi x$ та оформити легенди графічного вікна.

Перелік необхідних операцій наведений нижче:

```
>> x=–1:0.005:1; f=exp(x).*cos(x)--x.^3;
>> g=exp(–x.^2).*cos(4*pi*x); plot(x,f,'g--',x,g,'r-.'); grid on;
>> title('Графікі функцій'); legend('f','g',4);
>> xlabel('Ось x'); ylabel('Ось y')
```



Мал. 3.1 – Графікі функцій $f(x) = e^x \cos x = x^3$ та $g(x) = e^{-x^2} \cos 4\pi x$

При застосуванні команд **plot, semilogx, semilogy, loglog** система автоматично вибирає такий масштаб, щоб у полі графіку розмістилися усі результати обчислень. Однако, користовувач може самостійно задавати режим масштабування. Для цього треба вибирати наступны команди:

- **axis ([xmin xmax ymin ymax])** – здійснює явне завдання діапазону змінних по вісям координат;
- **axis square** – здійснює завдання однакових діапазонів змінних по обох вісях координат;
- **axis equal** – здійснює завдання однакового масштабу по обох вісях координат;
- **axis auto** – здійснює повернення до масштабу, встановленому за замовчуванням;
- **zoom on** – здійснює включення режиму інтерактивної зміни масштабу для для поточного графіка, при якому масштаб може змінюватись за допомогою миши;
- **zoom off** – здійснює виключення режиму інтерактивної зміни масштабу.

Вправа 4. Візуалізація векторних та матричних даних. Побудова діаграм.

Розглянемо процедуру візуалізації векторних і матричних даних. Найпростіший спосіб відображення векторних даних полягає у використуванні команда **plot** з вектором у якості вхідного параметру. При цьому, графік, що виходить у вигляді ламаної лінії, символізує залежність значень елементів вектора від їх індексів. Другий додатковий параметр може визначати колір, стиль лінії і тип маркерів, наприклад: **plot(x,'ko').** Виклик команди **plot** від матриці приводить до побудови декількох графіків; їх число співпадає з числом стовпців матриці, а кожен з стовпців залежить від індексів, що вказують на ряд. Колір, стиль ліній і тип маркерів загальний для всіх ліній, визначається другим за порядком додатковимпараметром.

Зручним способом представлення матричних і векторних даних є спосіб представлення за допомогою різноманітних діаграми. Найпростішею є стовпцева діаграма, що будується за допомогою команди **bar**:

```
>> x=[0.8 2.3 2.5 2.9 1.8 0.5];
>> bar(x)
```

Додатковий числовий параметр **bar** вказує на ширину стовпців (за замовчанням він рівний 0.8), а у випадку, коли значення цього параметра більше одиниці, наприклад, **bar** (x,1.2), відбувається частакове перекриття простору стовпців. Розташування у вхідному параметрікоманди **bar** матриці приводить до побудови групової діаграми, число груп співпадає з числом рядків матриці, а в середині кожної групи відображаються в стовпчиках значення індексів рядків.

Кругові діаграми векторних даних будуються за допомогою комнади **pie**, яка має деякі особливості в порівнянні з командою **bar**. Розрізняються два випадки:

- якщо сума компонентів вектора більше або дорівнює одиниці, то виводиться повна кругова діаграма, площа кожного її сектора пропорційна величині елемента вектора;
- якщо сума компонентів вектора менше одиниці, то результатом буде неповна кругова діаграма, в якій площа кожного сектора пропорційна величині компонентів вектора, в припущенні що площа всього круга дорівнює одиниці.

Приклад 1. Переконайтесь в результатах дії нижчеперелікованих команд та зробить аналіз результатів:

```
>> pie([0.1 0.2 0.3])
>> pie([0.2 0.3 0.5])
>> pie([4 5 6])
```

Можна відділити деякі сектори від повного круга діаграми, для чого слід викликати команду **pie** з другим параметром – вектором тієї ж довжини, що і початковий. Ненульові компоненти другого вектора (параметра) відповідають секторам, що можуть бути відокремлюваними.

Приклад 2. Відділити від діаграми сектор, який відповідає найбільшому компоненту вектора **x**.

Для виконання поставленої задачі спочатку треба у вихідному масиві знайти найбільший за значенням компонент та сформувати нульовий масив, розмір якого співпадає з вихідним:

```
>> x=[0.3 2 1.4 0.5 0.9];
>> [k]=max(x);
>> v=zeros(size(x));
```

Після виконаних операцій будуємо діаграму з найбільшим сектором, який буде відділятися від інших:

```
>> v(k)=1;
>> pie(x,v)
```

Команди **bar** і **pie** мають аналоги:

- **barh** – команда здійснює побудову столбцевої діаграми з горизонтальним розташуванням стовпців;
- **bar3, pie3** – команда здійснює побудову об'ємних (тривимірних) діаграм.

Приклад 3. Сформувати кругову та столбцеву діаграми та підписати сектори.

Для виконання завдання у разі застосування команди **pie** у другому додатковому вхідному параметрі вказується інформація, яка повинна розташовуватися поряд з відповідними секторами:

```
>> pie([22 37 15 23],{'Понеділок','Вівторок','Середа','Четвер'})
>> pie3([2 4 3 5],[0 1 1 0],{'Понеділок','Вівторок','Середа','Четвер'})
```

При обробці великих масивів векторних даних часто вимагається отримати інформацію про те, яка частина даних знаходиться в тому або іншому інтервалі. Команда **hist** призначена для відображення даних гістограми і знаходження числа даних в інтервалах. Вхідним параметром команди **hist** є вектор з даними, а вихідним параметром – вектор, що містить кількість елементів (даних), які потрапили в кожний з інтервалів. За замовченням розглядається десять рівних інтервалів. Наприклад, застосування команди **hist(randn(1,5000))** приводить до появи на екрані гістограми даних, розподілених по нормальному закону, а **n=hist(randn(1,5000))** до заповнення вектора **n** з десятьома компонентами (у цьому випадку гістограма не будується). Число інтервалів указується у іншому додатковому параметрі команди **hist**. При виписуванні інтервалів можна використовувати в якості другого параметра не число, а вектор, що містить координати центрів інтервалів. Більш зручно задавати інтервали не координатами центрів, а границями інтервалів. В цьому випадку вимагається спочатку визначити кількість елементів в інтервалах за допомогою команди **histc**, а потім застосувати команду **bar** із спеціальним параметром **'histc'.**

Приклад 4. Побудувати гістограму розподілу компонент масиву **x**, який заповнений випадковими величинами, що розподілені по нормальному закону.

Перелік необхідних команд наведений нижче:

```
>> x=randn(1,10000);
>> int=[–2:0.5:2];
>> n=histc(x,int);
>> bar(int,n,'histc')
```

Вправа 5. Побудова графіків функції двох змінних.

Візуалізація (побудова графіків) функцій двох змінних в системі **MATLAB** може бути здійснена декількома способами, але всі вони припускають однотипні попередні дії [33,с.136].

Приклад 1. Побудувати графікі функцій двох змінних на прямокутній області завдання функції. Шуканий графік представляє собою поверхню, що описується функцією $z(x,y)=e^{-x}\cos(2\pi y)$, заданою на прямокутнику $x\in[-1,\ 1],\ y\in[0,\ 2]$.

Перший крок побудови графіку полягає в завданні сітки на прямокутнику, тобто сукупності точок (вузлів), в яких обчислюються значення функції. Для генерації сітки передбачена команда **meshgrid**, яка викликається за допомогою двох вхідних параметрів – векторів, що визначають точки на вісях Ox і Oy. Команда **meshgrid** повертає два вихідні параметри **X**, **Y**, які є матрицями:

```
>> [X,Y]=meshgrid(–1:0.1:1,0:0.1:2);
```

Матриця X формується з елементів першого вхідного параметру в команді **meshgrid** – вектора {–1:0.1:1}, а матриця **Y** в команді **meshgrid** – з елементів другого вхідного параметру – вектора {0:0.1:2}. Такі матриці необхідні на другому етапі заповнення матриці Z, кожний елемент якої є значенням функції z(x,y) в точках сітки. Очевидно, що використовування поелементних операцій при обчисленні функції z(x,y) приводить до шуканої матриці:

```
>> Z=exp(–X).*cos(2*pi*Y);
```

Для побудови графіка функції z(x,y) залишається викликати одну з графічнихкоманд, наприклад, команду **mesh**:

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

На екрані з'являється графічне вікно, що містить каркасну поверхню досліджуваної функції. Зверніть увагу, що колір поверхні відповідає значенню функції.

Команда **colorbar** приводить до відображення в графічному вікні стовпчика, що показує співвідношення між кольором і значенням функції z(x,y). Кольори палітри графіка можна змінювати за допомогою команди **colormap**, наприклад **colormap(gray)** відображає графік у відтінках сірого

кольору. Назви кольорів, які задають вид палітри, та операції зміни кольорів, їх відтінків, наведені нижче:

- **bone** – схожа на палітру **gray**, але з легким відтінком синього кольору;
- **colorcube** – задає тон кольору, який змінюється від темного до яскравого;
- **cool** – задає відтінки голубого і пурпурного кольорів;
- **copper** – задає відтінки мідного кольору;
- **hot** – задає плавну зміну тону кольору;
- **hsv** – задає плавну зміну тонів барв веселки;
- **jet** – задає плавну зміну тонів у послідовності: синій–блакитний–зелений–жовтий–червоний;
- **spring** – задає відтінки пурпурного і жовтого;
- **summer** – задає відтінки зеленого і жовтого;
- **winter** – задає відтінки синього і зеленого.

Приклад 2. Побудувати графік функції $z(x,y)=e^{-x}\cos(2\pi y)$ з використанням голубого та пурпурного відтінків палітри.

Для цього достатньо у командне вікно ввести наступні команди:

```
>> [X,Y]=meshgrid(–1:0.1:1,0:0.1:2);
>> Z=exp(–X).*cos(2*pi*Y);
>> colormap(cool); mesh(X,Y,Z)
```

У системі **MATLAB** сформований цілий набір графічних команд візуалізації графіків функцій двох змінних, серед яких:

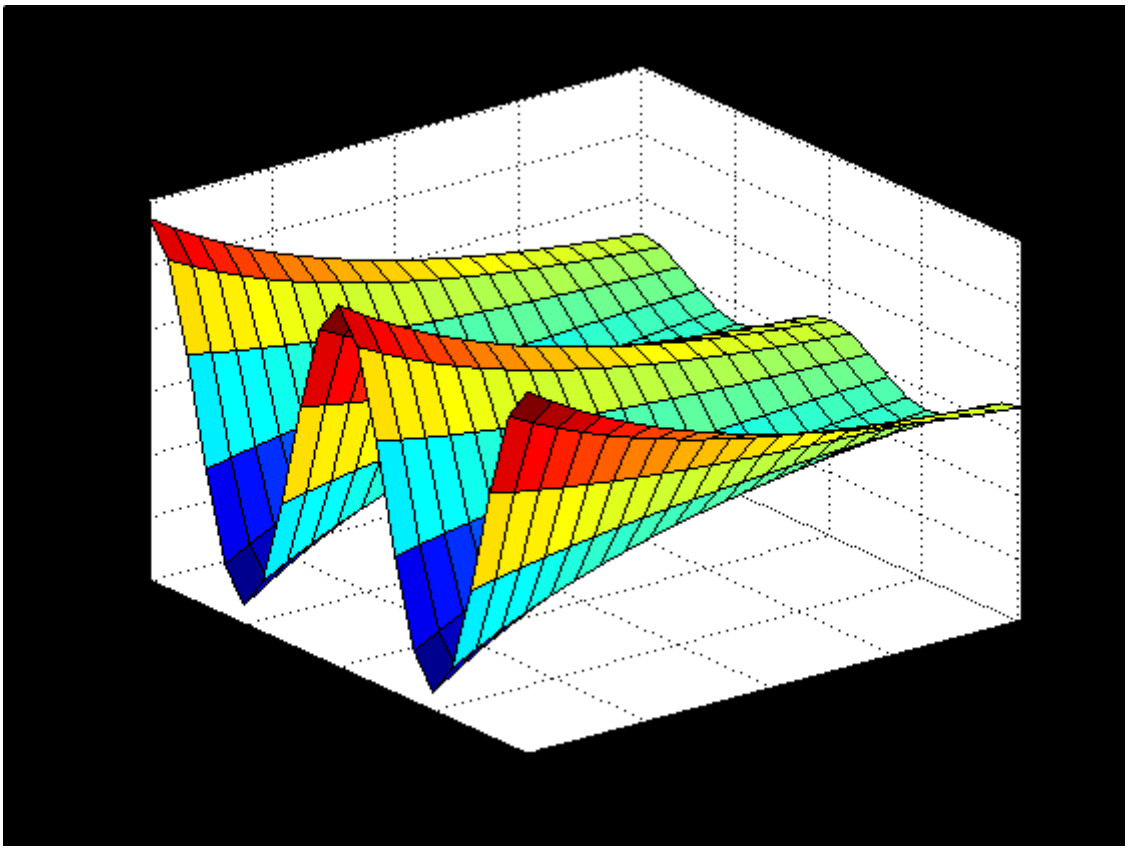
- **surf** – команда виконує заповнення кольором комірок каркасної поверхні;
- **meshc**, **surfc** – команди формують кольорові поверхні разом з лініями рівня на площині xOy;
- **contour** – команда формує площиний графік з лініями рівня;
- **contourf** – команда формує залитий кольором плоский графік з лініями рівня;
- **contour3** – команда відображає поверхню, яка сформована з ліній рівня;
- **surfl** – команда формує освітлену поверхню.

Всі перераховані команди допускають таку ж форму завдання вхідних параметрів, що і команда **mesh.**

Приклад 3. Побудувати поверхню, яка описується функцією $z(x,y)=e^{-x}\cos(2\pi y)$ з кольоровим заповненням каркасної поверхні.

Таке захавдання можна вирішується за допомогою наступних команд:

```
>> [X,Y]=meshgrid(–1:0.1:1,0:0.1:2);
>> Z=exp(–X).*cos(2*pi*Y);
>> surf(X,Y,Z)
```



Мал. 3.2 – Графік функції $z(x, y) = e^{-x} \cos(2\pi y)$

Рекомендуємо зупинитися більш детально на наступних питаннях. Перше питання: як змінювати установки, визначені за замовчуванням, при відображенні функцій лініями рівня за допомогою команд **contou** та **contourf**. Інформація про кількість ліній рівня в цих командах задається в четвертому додатковомупараметрі, наприклад:

```
>> Z=exp(–X).*cos(2*pi*Y); contourf(X,Y,Z,10)
```

Замість числа кількості ліній рівня можна вказати (за допомогою вектора) ті значення функції z(x,y), для яких вимагається побудувати лінії рівня:

```
>> Z=exp(–X).*cos(2*pi*Y);
>> contour(X,Y,Z,[–0.51 –0.25 –0.01 0.89])
```

Для формування підписів з відповідним значенням функції z(x,y) до кожної лінії рівня треба викликати команду **contour,** що повертає два вихідних параметра, перший з них – матриця з інформацією про положення ліній рівня, а другий – вектор з визначеними величинами значень функції на лінії. Отримані змінні слід використовувати як вхідні параметри команди **clabel**:

```
>> Z=exp(–X).*cos(2*pi*Y);
>> [CMatr, h] = contour(X, Y, Z);
>> clabel(CMatr, h)
```

де команда **contour(**X, Y, Z**)** малює лінії рівня для масиву Z з урахуванням координат, що виписани у векторах X та Y. Команда **contour**(X, Y, Z, n) малює **n** ліній для масиву даних Z (за замовчуванням отримуємо n=10).

Приклад 4. Побудувати лінії рівня (n=15) для функції $z(x, y) = e^{-x} \cos(2\pi y)$.

Побудова ліній відбувається завдяки використанню наступних команд:

```
>> [X,Y]=meshgrid(–1:0.1:1,0:0.1:2);
>> Z=exp(–X–Y)*cos(–X–Y);
>> [CMatr, h] = contour(X, Y, Z,15);
>> clabel(CMatr, h)
```

Для того щоб залити каркасні поверхні кольором, використовуються команди **sufr** і **surfc.** В результаті маємо постійний колір в межах кожного осередку. Застосування команди **shading interp**, що викликається після команд **surf** і **surfc**, дозволяє забезпечити плавну зміну кольорів в межах осередків і зникнення ліній сітки на поверхні. Якщо бажано ліквідувати сітку і зберегти постійний колір осередків, то достатньо використати команду **shading flat**, а команда **shading faceted** додасть графіку попередній вигляд.

Графічні команди за замовчунням розташовують поверхню так, що спостерігач бачить її частину під деяким кутом, а інша частина буде прихована від погляду. Положення спостерігача визначається двома кутами (кути задаються в градусах) : азимутом (**AZ**) і кутом піднесення (**EL**). Азимут відлічується від вісі, протилежної Oy, а кут піднесення – від площини xOy. Оглянути поверхню з усіх боків дозволяє команда **view**. Виклик команди **view** з двома вихідними параметрамиі без вхідних дає можливість визначити поточне положення спостерігача, наприклад:

```
>> [AZ, EL]=view
AZ =
     0
EL =
    90
```

Отримані значення система **MATLAB** використовує за замовчуванням при побудові тривимірних графіків. Для завдання положення спостерігача слід вказати азимут і кут піднесення (в градусах) як вхідних параметрів команди **view**, наприклад: **view(0,90)** розташовує спостеригача у верхній точці обзору графіка. Перед здійсненням поворотів графіку доцільно розставити позначення до вісей. Для цього використовуються такі ж самі команди **xlabel, ylabel** і **zlabel** для завдання підпису до вертикальної осі, що і у випадку 2D. Команда **view** допускає поширення варіантівзастосування:

- **view(3)** – здійснює повернення до стандартних установок;
- **view([x,y,z])** – задає переміщення спостерігача в точку з координатами x, y і z.

Освітлена поверхня будується за допомогою команда **surfl**, яка дозволяє отримати ясне уявлення про характер функції, що досліджується. Слід врахувати, що краще поєднувати виклик команди **surfl** з командою **shading interp** з аргументом, що визначає палітру кольорів. Відтінки палітри характеризуються наступними параметрами – **gray, copper, bone, winter** і т. д.

Оскількі поверхня має властивості розсіювання, віддзеркалення і поглинання світла, що витікає від деякого джерела, то часто необхідно задавати розташування цього джерела. Це здійснюється за допомогою четвертого додаткового параметру команди **surfl –** векторів з двома компонентами (азимут і кут піднесення джерела) або з трьох елементів (положення джерела світла в системі координат осей), наприклад:

```
>> Z=X.^2–Y.^2–exp(X.^2–Y.^2);
>> mesh(X,Y,Z)
>> surfl(X,Y,Z,[20 80])
```

або

```
>> [X,Y]=meshgrid([–3:0.1:3]);
>> Z=X.*exp(–X.^2–Y.^2);
>> surfl(X,Y,Z,[6 8 11])
```

Приклад 5. Побудувати графік функції $z(x,y) = x^2 - y^2 - e^{(x^2-y^2)}$ з використуванням команди згладжування кольорів **shading interp.**

Перелік операцій наведений нижче**:**

```
>> [x,y]=meshgrid(--1:0.01:1);
>> z=x.^2–y.^2–exp(x.^2–y.^2);
>> surfl(x,y,z);
>> colormap(jet);
>> shading interp
```

Вправа 6. Візуалізація графіків декількох функцій в одному графічному вікні.

Розглянемо задачу про побудову в одному вікні графіків декількох функцій. Перший виклик будь-якої графічної команди приводить до появи на екрані графічного вікна **Figure No. 1**, що містить осі з графіком. Проте, при подальших зверненнях до графічних команд попередній графік пропадає, а замість нього виводиться новий графік. Команда **figure** призначена для створення графічного вікна, якій не містить графіків. Якщо треба отримати декілька графіків в різних вікнах, то перед викликом графічних команд слід звертатися до команди **figure**. Графічні вікна при цьому нумеруються в наступної послідовності: **Figure No. 2**, **Figure No. 3** і т.д.

Кожне вікно має свої вісі, за наявності декількох пар вісей (в одному вікні або в різних вікнах) побудова графіків здійснюється в поточному вікні (у відповідних вісях). Остання створена пара вісей є поточною. Для того, щоб вибрати поточні вісі з тих, що є, достатньо виділити координатну площину,

що визначається даними вісями лівою кнопкою миші. Можлива і зворотна ситуація, коли в процесі роботи вимагається додавати графіки до вже існуючих вісей. В цій ситуації перед добавлення графіка слід виконати команду **hold on**. Для завершення такого режиму достатньо скористатися командою **hold off**.

В одному графічному вікні можна розташувати декілька вісей з своїми графіками. Команда **subplot** призначена для розбиття графічного вікна на частини і визначення поточної з них.

Припустимо, що вимагається вивести шість графіків у відповідних координатних вісях в одному графічному вікні (тобто треба розмістити два графіка по вертикалі і три графіка по горизонталі).

Приклад 1. Сформувати графічне вікно, у якому будуть виведені п'ять різних графіків функцій $z(x,y) = xe^{-x^2-y^2}$, $y(x) = \cos\ln x$, $y(x) = x^2 \sin x$, $z(x,y) = x^2 + y^2$, $z(x,y) = e^{-x}\cos(-x-y)$ і стовпчикової діаграми, що задається масивом елементів [1,2 0,3 2,8 0,9].

Для вирішення цієї задачи треба створити графічне вікно за допомогою команди **figure** і задати команду **subplot**:

```
>> subplot(2,3,1)
```

В лівому верхньому кутку вікна з'являються вісі. Перші два параметри в команді **subplot** вказують на загальне число пар вісей по вертикалі і по горизонталі, а останній параметр – визначає номер відповідних вісей. Нумерація малюнків графіків йде зліва направо, зверху вниз.

Використайте команди **subplot**(2,3,2), **subplot**(2,3,6) для створення решти вісей. Результат виконання будь-якої з графічних команд можна орієнтувати на відповідні вісі, що визначаються за допомогою команди **subplot**(2,3,k), наприклад:

```
>> subplot(2,3,1);
>> [X,Y]=meshgrid([–3:0.1:3]);
>> Z=X.*exp(–X.^2–Y.^2); mesh(X,Y,Z);
>> subplot(2,3,2);
>> bar([1.2 0.3 2.8 0.9]);
>> subplot(2,3,3);
>> x=0.001:0.0001:100; y=cos(log(x));
>> semilogx(x,y);grid on;
>> subplot(2,3,4); fplot('x.^2.*sin(x)',[0:10])
>> subplot(2,3,5);
>> [x,y]=meshgrid([–20:0.5:20]);
>> z=x.^2+y.^2; surf(x,y,z)
>> subplot(2,3,6);
>> [X,Y]=meshgrid(-1:0.1:1,0:0.1:2);
>> Z=exp(–X).*cos(–X–Y);
>> [CMatr, h] = contour(X, Y, Z,15);clabel(CMatr, h)
```

Вправа 8. Зберігання, екпорт та друк графіків функцій.

Графічне вікно у **MATLAB** зберігається так саме, як будь-який інший документ в програмах. Тобто, у операції меню **File** слід вибрати команду **Save** або **Save as** та зберігти файл звичайним способом. У цьому разі файл с графичним вікном отримує розширення **.fig** і за замовчуванням буде збережений у поточному робочому каталозі програми системи **MATLAB**. У цьому випадку вікно збережеться повністю, включаючи меню та панель інструментів.

Графікі, побудовані у системі **MATLAB**, можна експортувати в інші графічні формати, наприклад**: jpeg, tiff, bmp, gif.**

Процесс друку графіка у системі **MATLAB** включає наступні етапи:

- настроювання параметрів: встановлення параметрів сторінки у діалоговому вікні **Page Setup**;
- попередній перегляд графіку перед друком;
- вибір параметрів принтера у діалоговому вікні та друк документа.

Діалогове вікно **Page Setup** складається з чотирьох вкладок: **Size and Position** (розмір та розташування), **Paper** (аркуш), **Lines and Text** (лінії і текст), **Axes and Figure** (вісі та графік).

На вкладці **Size and Position** встановлюються розміри графіку та його положення на сторінці. Розміри та орієнтація аркуша вибираються на вкладинці **Paper** діалогового вікна **Page Setup**, встановлення чорно-білого або кольорового друку – на вкладинці **Lines and Text**, границі вісей та розмітки графіка – на вкладинці **Axes and Figure**.

Попередній перегляд графіку відбувається за допомогою команди **File/Print Preview** (попередній перегляд) графічного вікна. Вікно попереднього перегляду має своє меню, за допомогою якого можна визвати діалогове вікно для найстроювання параметрів принтера та сторінки, а також визвати діалогове вікно, в якому можна задати заголовок графіка і дату його друку.

Настроювання параметрів принтеру можна виконати в діалоговому вікні, які викликані за допомогою команди **Print Setup** (настроювання параметрів друку) з меню **File** графічного вікна. Для того щоб надрукувати графік, треба вибрати команду **File/Print** (друк) або команду **Print Figure** (друк зображення) панелі інструментів графічного вікна.