Лабораторна робота №9

**ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ОБРОБКИ ВИМІРЮВАЛЬНОЇ ІНФОРМАЦІЇ ТА СПЕЦІАЛЬНІ ТИПИ ГРАФІКІВ У РОБОЧОМУ СЕРЕДОВИЩІ MATLAB**

**Мета:***проведення досліджень щодо створення М-файлів у робочому середовищі програми Matlab, що задають візуалізацію результатів обробки вимірювальної інформації та спеціальні типи графіків.*

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

1. **Додавання формул на графіки**

Досить часто необхідним є розміщення формули в заголовку графіку або поруч із вертикальною віссю. Для розміщення підпису біля вертикальної осі використовується команда **zlabel**.

Використання в аргументах команд деяких математичних позначень в форматі ТеХ дозволяє додавати формули на графік.

Якщо формула не поміщається в один рядок при наборі команди, то слід використати оператор три крапки. Наприклад, для відображення формули в заголовку функції:



слід набрати наступну команду:



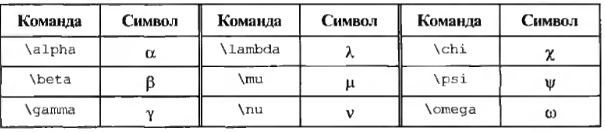
Правила набору формул та зміна властивостей шрифтів наведені в таблиці 10.1

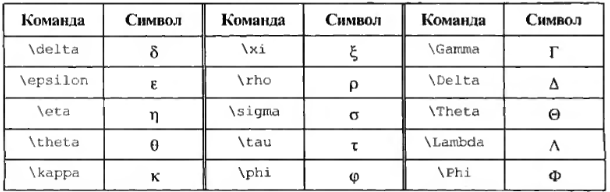
Таблиця 10.1 – Правила набору формул та зміна властивостей шрифтів

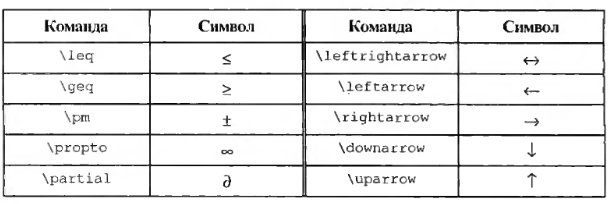
|  |  |  |
| --- | --- | --- |
| Що вимагається? | Команда ТеХ | Результат |
| Виділення ***курсивом*** одного символу або тексту | {\itx} | *x* |
| 1.2{\itP} | 1.2*P* |
| {\itГіперболічний} синус | *Гіперболічний* синус |
| Виділення жирним шрифтом одного символу або тексту | Шаблон матриці {\bfM} | Шаблон матриці **М** |
| {\bfАЧХ} фільтра | **АЧХ** фільтра |
| Набір символу або тексту жирним курсивом | Вектори {\bf\itx} та {\bf\ity} | Вектори ***x*** та ***y*** |
| {\bf\itОптимальна} крива | ***Оптимальна*** крива |
| Зміна шрифта та його розміру | {\fontname{arial}\fontsize{14}z-функція} | Z-функція |
| Степінь, верхній індекс | x^{2} | x2 |
| {\itx}^{2.5} | *x*2.5 |
| {\ite}^{\it-x} | *e-x* |
| Нижній індекс | f\_{5} | f5 |
|  | f\_{\itxx} | f*xx* |

Можливе використання грецьких літер та спеціальних символів, наприклад title(‘Залежність при a=\pi’) призводить до заголовку «Залежність при а=π». В таблиці 10.2 приведено команди ТеХ для вставки деяких прописних і рядкових грецьких літер і спеціальних символів.

Таблиця 10.2 – Грецькі літери та спеціальні символи



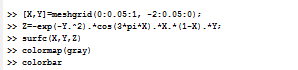




Так, розглянемо приклад побудови функції:



Введемо наступні команди в у вікні Command Window:





В результаті отримуємо графік функції з вказаними підписами осей та заголовком:

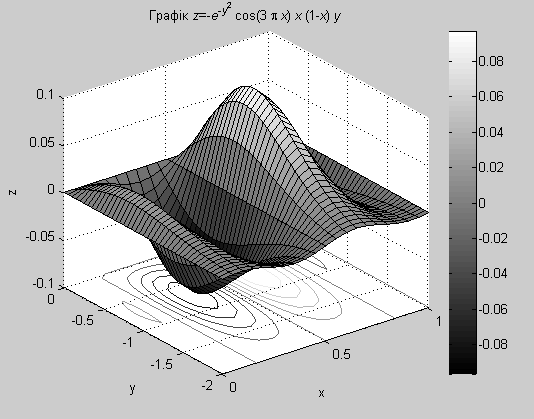
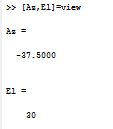


Рисунок 10.1

1. **Анімаційні графіки**

Для отримання повної інформації про поверхню та огляду її зі всіх сторін необхідно можна змінювати положення спостерігача. Положення спостерігача характеризується двома кутами: азимутом (Az) та кутом піднесення (El). Азимут вираховується від осі, що протилежна ***y***, а кут піднесення від площини ***xy***. Зміна положення спостерігача відбувається за допомогою функції **view**. Аргументами **view** є азимут та кут піднесення, що обраховується в градусах. По замовчуванню:

Az=–37,50, El=300



Задаємо параметри Az=1350, El=450



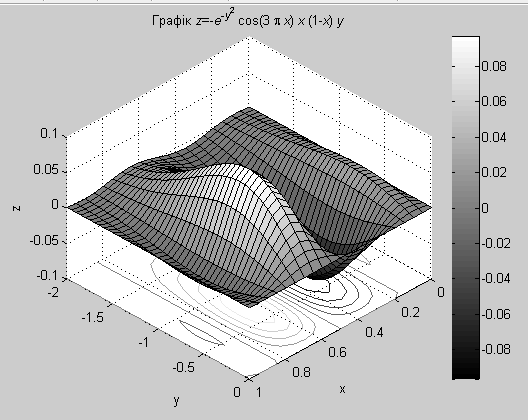


Рисунок 10.2 – Графік з точки зору спостерігача Az=1350, El=450

При *view(0,0)*

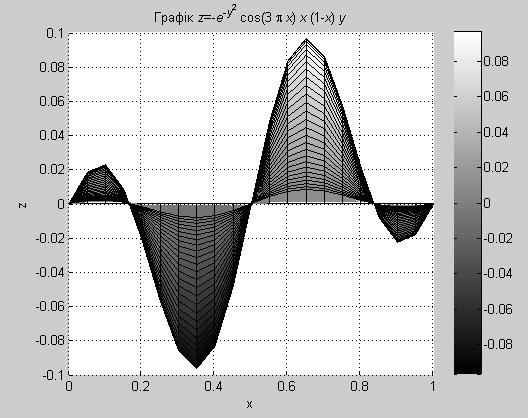


Рисунок 10.3 **–** Графік з точки зору спостерігача Az=00, El=00

Для побудови анімаційних графіків використовується функції comet та comet3. Так наприклад, побудуємо траєкторію руху точки протягом 10 секунд, координати яких змінюються по закону:



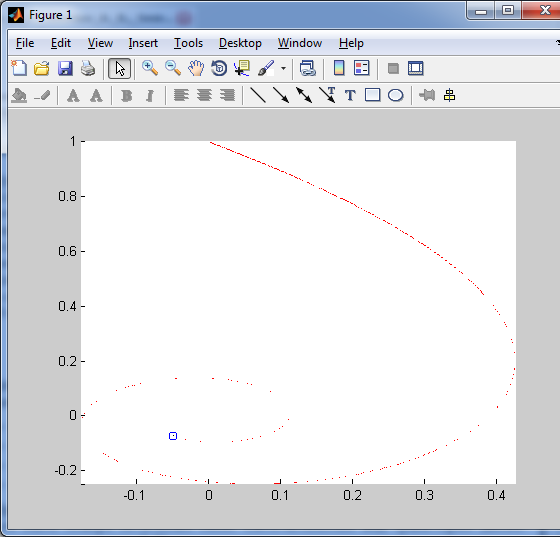
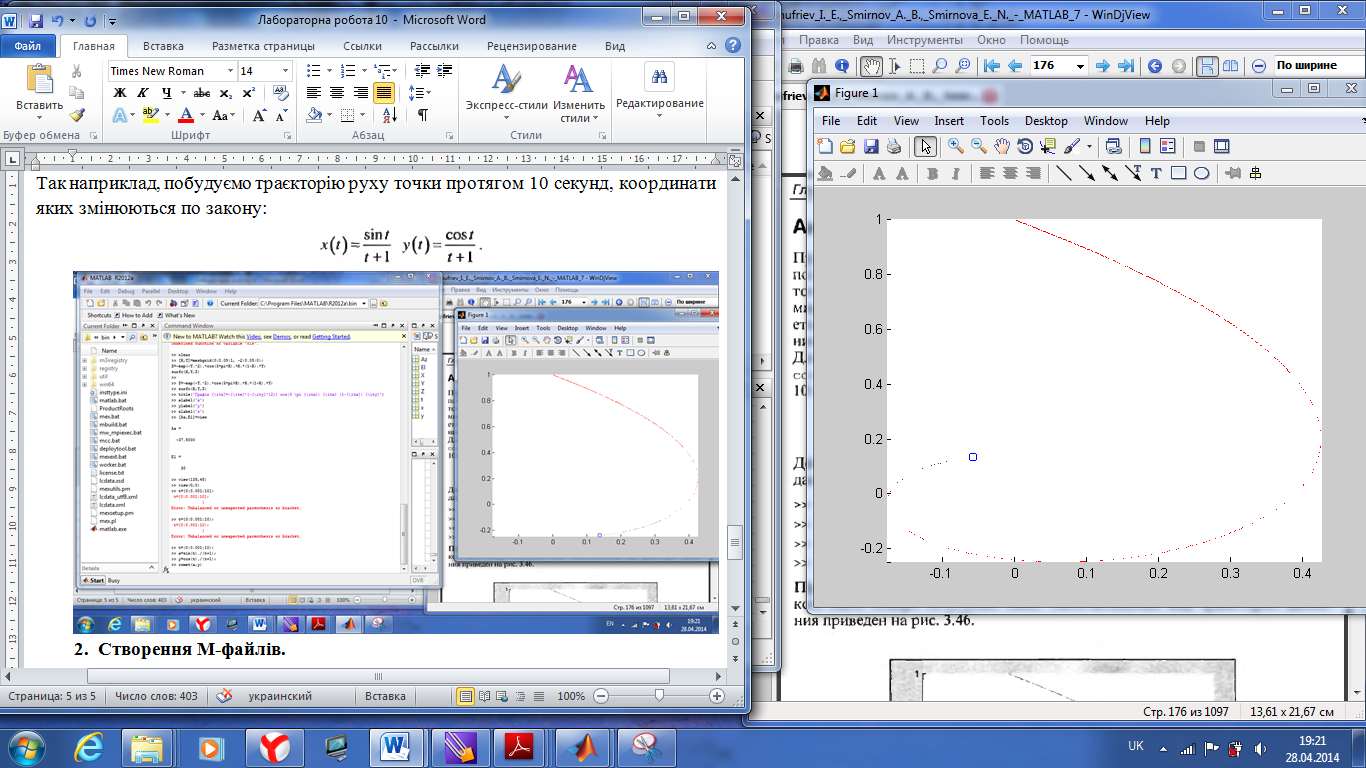
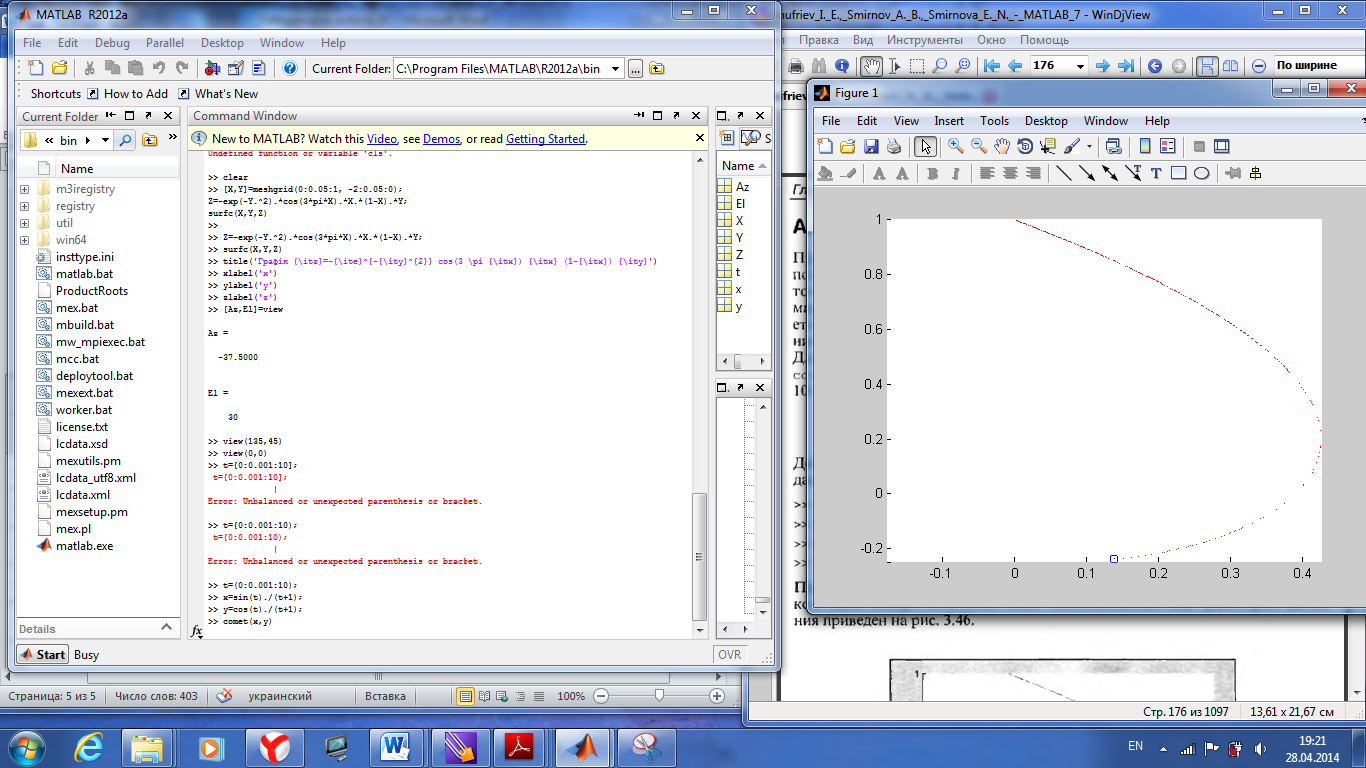


Рисунок 10.4 – Вигляд траєкторії руху точки

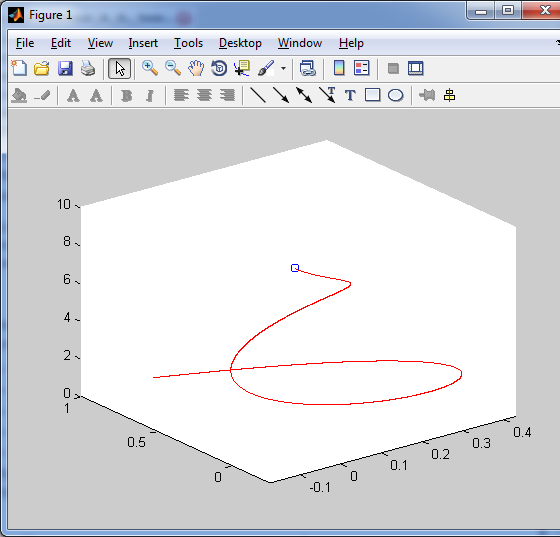


Рисунок 10.5 – Кінцевий вигляд траєкторії руху (comet3)

1. **Візуалізація векторних полів**

Функції **compass**, **feather** та **quiver** графічно представляють сукупність двомірних векторів. В якості прикладу побудуємо залежність вектора швидкості тіла, що кинуте під кутом до горизонталі, від часу:

, ,

де , – проекції вектора початкової швидкості; m – маса; g – прискорення вільного падіння; t – час. Вектори повинні виходити із точок, що належать траєкторії руху тіла. Траєкторія руху описується законом:

Приймаємо та тривалість польоту складає дві секунди до падіння тіла на поверхню і прослідкуємо за зміною його швидкості через кожні 0,2 секунди.







Вигляд графіку представлено на рис. 10.6.

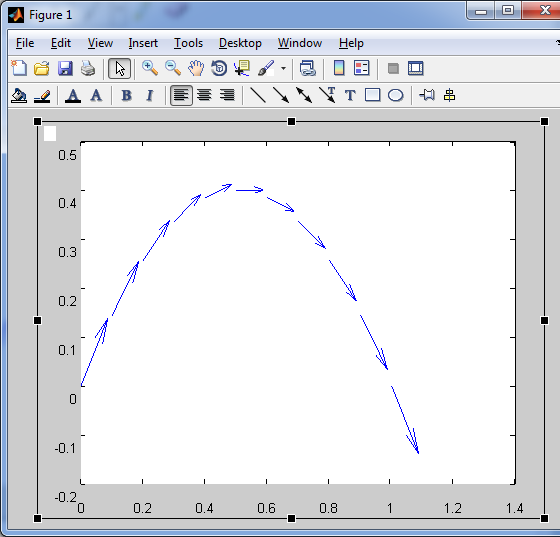


Рисунок 10.6

Функції compass та feather реалізує дещо інші способи графічного відображення векторних полів.

>> compass(ux, uy)

>> feather(ux, uy)

1. **Створення М-файлів.**

Робота в режимі калькулятора в середовищі Matlab, незважаючи на досить значні можливості, має істотні незручності. Неможливо повторити всі попередні обчислення й дії при нових значеннях початкових даних без повторного набирання всіх попередніх операторів. Не можна повернутися назад і повторити деякі дії, або за деякою умовою перейти до виконання іншої послідовності операторів. І взагалі, якщо кількість операторів є значною, стає проблемою налагодити їх правильну роботу через неминучі помилки при набиранні команд.

Якщо користувачу необхідно багаторазово повторювати обчислення або реалізувати складні обчислювальні алгоритми при розробці власних додатків, є доцільним використання M-файлів і зберігати власні додатки як М-файл з унікальним ім’ям у папку, яка задовольняє користувача. За замовчуванням при виконанні програми М-файл запишеться в поточну папку та йому присвоюється ім’я Untitled.m

При виконанні простих обчислюваних операцій, що не потребують багатократного повторення однотипних обчислюваних процедур доцільно створювати та використовувати скрипт-файли, що являють собою послідовність команд Matlab. Ці програми не мають, як правило, вхідних і вихідних аргументів. Тоді як файл-функції виконують необхідні дії з вхідними аргументами та повертають результат у вихідний аргумент. Основні відмінності представлені у таблиці 10.3.

Таблиця 10.3 – Типи М-файлів у Matlab

|  |  |
| --- | --- |
| Файл-скрипт | Файл-функція |
| Не використовує вхідні і вихідні аргументи | Використовують вхідні та вихідні аргументи |
| Оперують даними з робочої області | По замовчуванню, внутрішні змінні функції є локальними |
| Призначені для автоматизації послідовності простих обчислень, які потрібно багаторазово використовувати | Призначені для розширення можливостей Matlab, дозволяючи створювати бібліотеки функції користувачів та пакети прикладних програм. |

Запис тексту програми (М-файлу) мовою Matlab має підпорядковуватися наступним правилам.

1. Зазвичай кожний оператор записується в окремому рядку тексту програми. Ознакою кінця оператора є символ (він не виникає у вікні) повернення каретки й переходу на наступний рядок, який вводиться в програму при натисканні клавіші Enter, тобто при переході при записі тексту програми на наступний рядок.

2. Можна розміщувати кілька операторів в одному рядку. Тоді попередній оператор у тому ж рядку має закінчуватися символом ' ; ' або ' , '.

3. Можна довгий оператор записувати в декілька рядків. При цьому попередній рядок оператора має завершуватися трьома крапками (' ... ').

4. Якщо черговий оператор не закінчується символом ' ; ', результат його дії при виконанні програми буде виведений у командне вікно. Щоб запобігти виведенню на екран результатів дії оператора програми, запис цього оператора в тексті програми має закінчуватися символом ' ; '.

5. Рядок програми, що починається із символу ' % ', не виконується. Цей рядок сприймається системою Matlab як коментар. Тому для введення коментарю в будь-яке місце тексту програми достатньо почати відповідний рядок із символу ' % '.

6. Рядки коментарю, які передують першому виконуваному (тобто такому, що не є коментарем) оператору програми, сприймаються системою Matlab як опис програми. Саме ці рядки виводяться в командне вікно, якщо в ньому набрано команду help <ім'я файла> .

7. У програмах мовою Matlab відсутній символ закінчення тексту програми.

8. У мові Matlab змінні не описуються і не оголошуються. Будь-яке нове ім'я, що зустрічається в тексті програми при її виконанні, сприймається системою Matlab як ім'я матриці. Розмір цієї матриці встановлюється при введенні значень її елементів або визначається діями по встановленню значень її елементів, описаними у попередньому операторі або процедурі. Ця особливість робить мову Matlab дуже простою у вжитку і привабливою. У мові Мatlab неможливо використання вхідної матриці або змінної, у якій попередньо не введені або обчислені значення її елементів (а значить - і визначені розміри цієї матриці). У противному випадку при виконанні програми Matlab з'явиться повідомлення про помилку - "Змінна не визначена".

9. Імена змінних можуть містити лише букви латинського алфавіту або цифри і мають починатися з букви. Загальна кількість символів в імені може сягати 30. В іменах змінних можуть використовуватися як великі, так і малі букви. Особливістю мови Matlab є те, що великі й малі букви в іменах розрізнюються системою. Наприклад, символи "а" і "А" можуть використовуватися в одній програмі для позначення різних величин.

Script-файл (файл-сценарій, або керуюча програма) та файл-функція (процедура) матимуть розширення .m при записі тексту файлу на диск, тобто їх не можна розрізнити по типу.

За допомогою Script-файлів оформлюють основні програми, що керують із початку до кінця організацією усього обчислювального процесу, і окремі частини основних програм (вони можуть бути записані у виді окремих Script-файлів). Як файл-функції оформляють окремі процедури й функції (тобто такі частини програми, які розраховані на неодноразове використання Script-файлами або другими процедурами при змінених значеннях вхідних параметрів і не можуть бути виконані без попереднього завдання значень деяких змінних, які називають вхідними).

Головною зовнішньою відмінністю цих двох видів файлів є те, що файл-функції мають перший рядок виду:

function <ПКВ> = <ім'я процедури >(<ПВВ>),

де позначено ПКВ - Перелік Кінцевих Величин, ПВВ - Перелік Вхідних Величин.

Script-файли такого рядка не мають.

Принципова ж відмінність полягає в зовсім різному сприйнятті системою імен змінних у цих двох видах файлів. У файл-функціях усі імена змінних усередині файлу, а також імена змінних, зазначені в заголовку (ПКВ і ПВВ), сприймаються як локальні, тобто усі значення цих змінних після завершення роботи процедури зникають, і область оперативної пам'яті, що була відведена під запис значень цих змінних, звільняється для запису в її значень інших змінних.

У Script-файлах усі використовувані змінні утворюють так званий “робочий простір” (Work Space). Значення й зміст їх зберігаються не тільки протягом часу роботи програми, але й протягом усього сеансу роботи із системою, а, виходить, і при переході від виконання одного Script-файлу до іншого. Інакше кажучи, робочий простір є єдиним для всіх Script-файлів, що викликаються в поточному сеансі роботи із системою. Саме завдячуючи цьому будь-який довгий Script-файл можна розбити на декілька фрагментів, оформити кожний з них у вигляді окремого Script-файлу, а в головному Script-файлі замість відповідного фрагменту записати оператор виклику Script-файлу, який подає цей фрагмент.

Цим забезпечується компактне й наочне подання навіть досить складної програми. За винятком зазначених відмінностей, файл-функції і Script-файли оформляються однаково.

1. **Оператори керування обчислювальним процесом**

Загалом кажучи, оператори керування необхідні, головним чином, для організації обчислювального процесу, який записується у виді деякого тексту програми на мові програмування високого рівня. При цьому до операторів керування обчислювальним процесом звичайно відносять оператори безумовного переходу, умовних переходів (розгалуження обчислювального процесу) і оператори організації циклічних процесів. Проте система MatLAB побудована таким чином, що ці оператори можуть бути використані і при роботі MatLAB у режимі калькулятора.

*У мові MatLAB відсутній оператор безумовного переходу і, відповідно до цього, немає поняття мітки.* Ця обставина є недоліком мови MatLAB і ускладнює організацію повернення обчислювального процесу до будь-якого попереднього або наступного оператора програми.

Усі оператори циклу й умовного переходу побудовані в MatLAB у вигляді складного оператора, що починається з одного зі службових слів ***if, while, switch*** або ***for*** і закінчується службовим словом ***end***. Оператори усередині між цими словами сприймаються системою як частини одного складного оператора. Тому натискання клавіші <Enter> для переходу до наступного рядка не призводить у цьому випадку до виконання цих операторів. Виконання операторів починається лише тоді, коли введена "завершувальна дужка" складного оператора у виді ***end*** , а потім натиснуто клавішу <Enter>. Якщо декілька складних операторів такого типу вкладені один в інший, обчислення починаються лише тоді, коли записаний кінець ***end*** найбільш охоплюючого (зовнішнього) складного оператора. З цього випливає можливість здійснення навіть у режимі калькулятора досить складних і об'ємних (що складаються з багатьох рядків і операторів) обчислень, якщо вони охоплені складним оператором.

**5.1. Оператор умовного переходу**

Конструкція оператора переходу за умовою в загальному вигляді є такою:

***if*** <умова>

<оператори1>

***else***

<оператори2>

***end***

Працює оператор таким чином. Спочатку перевіряється, чи виконується зазначена умова. Якщо її виконано, програма виконує сукупність операторів, що записана в поділі <оператори1>. Якщо умову не виконано, виконується послідовність операторів поділу <оператори2>.

Скорочена форма умовного оператора має вигляд:

***if*** <умова>

<оператори>

***end***

Дія оператора в цьому випадку аналогічно, за винятком того, що при невиконанні заданої умови виконується оператор, наступний за оператором ***end***.

Легко помітити хиби цього оператора, що випливають із відсутності оператора безумовного переходу: *усі частини програми, що виконуються в залежності від умови, повинні розміщатися усередині операторних дужок* ***if*** *і* ***end****.*

Як умова використовується вирази типу:

<ім'я змінної1> <операція порівнювання> <ім'я змінної2>

*Операції порівнювання в мові MatLAB* можуть бути такими:

< - менше;

> - більше;

<= - менше або дорівнює;

>= - більше або дорівнює;

= = - дорівнює;

~ = - не дорівнює.

Умова може бути складеною, тобто складатися з кількох простих умов, що об'єднуються знаками логічних операцій. Знаками логічних операцій у мові MatLAB є:

& - логічна операція “І” (“AND”);

| - логічна операція “АБО” (“OR”);

~ - логічна операція “НІ” (“NOT”).

Логічна операція “Виняткове АБО” може бути реалізована за допомогою функції ***xor***(А,У), де А і В - деякі умови.

Є припустимою ще одна конструкція оператора умовного переходу:

***if*** <умова1>

<оператори1>

***elseif*** <умова2>

<оператори2>

***elseif*** <умова3>

<оператори3>

. . .

***else***

<оператори>

***end***

Оператор ***elseif*** виконується тоді, коли <умова1> не виконана. При цьому спочатку перевіряється <умова2>. Якщо її виконано, виконуються <оператори2>, якщо ж ні, <оператори2> ігноруються, і відбувається перехід до наступного оператора ***elseif***, тобто до перевірки виконання <умови3>. Аналогічно, при виконанні її виконуються <операторы3>, у противному випадку відбувається перехід до наступного оператора ***elseif***. Якщо жодну з умов в операторах ***elseif*** не виконано, виконуються <оператори>, що містяться за оператором ***else***.

У такий спосіб може бути забезпечене розгалуження програми по кількох напрямках.

***5.2. Оператор переключення***

Оператор переключення має таку структуру:

***switch*** <вираз, скаляр або рядок символів>

***case*** <значення1>

<оператори1>

***case*** <значення2>

<оператори2>

***. . .***

***otherwise***

<оператори>

***end***

Він здійснює розгалужування обчислень у залежності від значень деякої змінної або виразу, порівнюючи значення, отримане в результаті обчислення виразу в рядку ***switch,*** із значеннями, зазначеними в рядках із словом ***case.*** Відповідна група операторів ***case*** виконується, якщо значення виразу збігається зі значенням, зазначеним у відповідному рядку ***case.*** Якщо значення виразу не збігається з жодним із значень у групах ***case***, виконуються оператори, що наслідують ***otherwise.***

***5.3. Оператори циклу***

У мові MatLAB є два різновиди операторів циклу - *умовний* і *арифметичний*.

*Оператор циклу з передумовою* має вид:

***while*** <умова>

<оператори>

***end***

Оператори усередині циклу виконуються лише в тому випадку, якщо є виконаною умова, записана після слова ***while***. При цьому серед операторів усередині циклу обов'язково повинні бути такі, що змінюють значення однієї зі змінних, зазначених в умові циклу.

Наведемо приклад обчислення значення синуса при 21 значенні аргументу від 0.2 до 4 із кроком 0.2:

**» i = 1;**

**» while i <= 20**

**x = i/5;**

**si = sin(x);**

**disp([x,si])**

**i = i+1;**

**end**

0.2000 0.1987

0.4000 0.3894

0.6000 0.5646

0.8000 0.7174

1.0000 0.8415

1.2000 0.9320

1.4000 0.9854

1.6000 0.9996

1.8000 0.9738

2.0000 0.9093

2.2000 0.8085

2.4000 0.6755

2.6000 0.5155

2.8000 0.3350

3.0000 0.1411

3.2000 -0.0584

3.4000 -0.2555

3.6000 -0.4425

3. 8000 -0. 6119

4.0000 -0. 7568

***Примітка.*** Зверніть увагу на те, якими засобами в зазначеному прикладі забезпечено виведення на екран значень кількох змінних одним рядком. Для цього використовується оператор ***disp***. Але, відповідно до правил застосування цього оператора, у ньому повинний бути тільки один аргумент (текст, змінна або матриця). Щоб обминути цю перешкоду, потрібно декілька числових змінних об'єднати в єдиний об'єкт - вектор-рядок, а останнє легко виконується за допомогою звичайної операції формування вектора-рядка з окремих елементів

[ x1, x2, ... , x].

Таким чином, за допомогою оператора виду:

***disp***([x1, x2, ... , x])

можна забезпечити виведення результатів обчислень у вигляді таблиці даних.

*Арифметичний оператор циклу* має вид:

***for*** <ім'я> = <ПЗ> : <К> : <КЗ>

<оператори>

***end,***

де <ім'я> – ім'я керуючої змінної циклу– "лічильника" циклу; <ПЗ> – задане по-

чаткове значення цієї змінної; <К> – значення кроку, із яким вона повинна змінюватися; <КЗ> – кінцеве значення змінної циклу. У цьому випадку <оператори> усередині циклу виконуються кілька разів (кожного разу при новому значенні керуючої змінної) доти, поки значення керуючої змінної не вийде за межі інтервалу між <ПЗ> і <КЗ>. Якщо параметр <К> не зазначений, за замовчуванням його значення приймається рівним одиниці.



Рисунок 10.7 – Блок-схема циклу for

В даному операторі допустиме значення змінної можна задавати і вектором.

Щоб достроково вийти з циклу (наприклад, при виконанні деякої умови) застосовують оператор ***break***. Якщо програма стикається з цим оператором, виконання циклу достроково припиняється, і починає виконуватися оператор,

наступний за словом ***end*** циклу.

Для приклада використаємо попереднє завдання:

**» a = [' i ',' x ',' sin(x) '];**

**» for i = 1:20**

**x = i/5;**

**si = sin(x);**

**if i==1**

**disp(a)**

**end**

**disp([i,x,si])**

**end**

В результаті виходить

i x sin(x)

1.0000 0.2000 0.1987

2.0000 0.4000 0.3894

3.0000 0.6000 0.5646

4.0000 0.8000 0.7174

5.0000 1.0000 0.8415

6.0000 1.2000 0.9320

7.0000 1.4000 0.9854

8.0000 1.6000 0.9996

9.0000 1.8000 0.9738

10.0000 2.0000 0.9093

11.0000 2.2000 0.8085

12.0000 2.4000 0.6755

13.0000 2.6000 0.5155

14.0000 2.8000 0.3350

15.0000 3.0000 0.1411

16.0000 3.2000 -0.0584

17.0000 3.4000 -0.2555

18.0000 3.6000 -0.4425

19. 0000 3.8000 -0. 6119

20.0000 4. 0000 -0. 7568

Ще один варіант

**for i=1:100**

**операторы**

**if a(i) == 0**

**break**

**end**

**операторы**

**end**



Рисунок 10.8 – Ілюстрація використання оператору ***break***

У такий спосіб можна забезпечити виведення інформації у вигляді таблиць.

1. **Створення власних функцій**

Шаблон опису функції виглядає наступним чином:

**function результат = имя ( аргументы )**

**операторы**

Перший рядок є обов’язковим, що вказує на те, що даний файл є файл-функцією. Після неї розміщується тіло функції – оператори. Ім’я функції обов’язково повинно співпадати з ім’ям файлу.

Наприклад:

**function y=minf(x, n)**

**y=x(1) ;**

**for i=1:n**

**if x(i)<y**

**y=x(i) ;**

**end**

**end**

Останнє значення, що присвоєне змінній результату **y** буде передано у вищестоящу програму або в робочу область самого Matlab. Якщо із функції необхідно вираховувати декілька значень (або масивів), то вони перераховуються у прямокутних скобках:

**function [y, k]=minf(x, n)**

**y=x(1) ;**

**k=1;**

**for i=1:n**

**if x(i)<y**

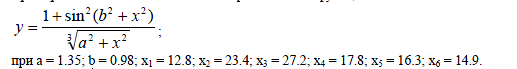
**y=x(i) ;**

**k=i ;**

**end**

**end**

Приклад. Знайти значення функції та вивести результат обчислень на екран.



Файл-програма представлена на рисунку 10.9



Рисунок 10.9 – Вирішення задачі за допомогою оператора циклу

Результат буде представлено у командному вікні у вигляді:



Проведемо вирішення даної задачі без використання оператору циклу. Лістинг програми представлено на рисунку 10.10



Рисунок 10.10 – Вирішення задачі без використання оператору циклу

Результат:



Проведемо вирішення даної задачі з побудовою файл-функції. Створюємо файл-функцію test1.m:

function [y] = test1(a,b,x)

y =(1+sin(b^2+x.^2).^2)./(a^2+x.^2).^(1/3)

На наступному етапі створюємо скрипт-файл в якому визначаються значення параметрів функції. В даному файлі також записується рядок з викликом зазначеної функції, рис.10.11:



Рисунок 10.11 – Виклик функції test1.m

В результаті отримаємо наступні значення функції:



Побудуємо графік розглянутої функції за допомогою скрипт-файлу, рис.10.12:



Рисунок 10.12 – Лістинг програми для побудови графіку функції

LineWidth – ширина лінії в точках (points), де 1 точка рівна 1/72 дюйма.

MarkerEdgeColor – колір маркера або колір границь маркера для маркеров замкнутої форми (кружків, квадратів, ромбів та ін.).

MarkerFaceColor – колір поверхні, що заповнені маркерами.

MarkerSize –размір маркера в одиницях точки.

Результат роботи даного лістингу представлено на рисунку 10.13



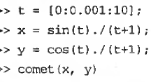
Рисунок 10.13 – Графічна візуалізація результатів обчислень

**ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ**

Запустіть програму MATLAB.

1. Побудуйте графік функції , що розглянутий у п.1. та представте графік функції як показано на рис.10.1 – 10.3.
2. Побудувати анімаційний графік, що відображає траєкторію руху точки протягом вказаного часу, координати яких змінюються по закону:





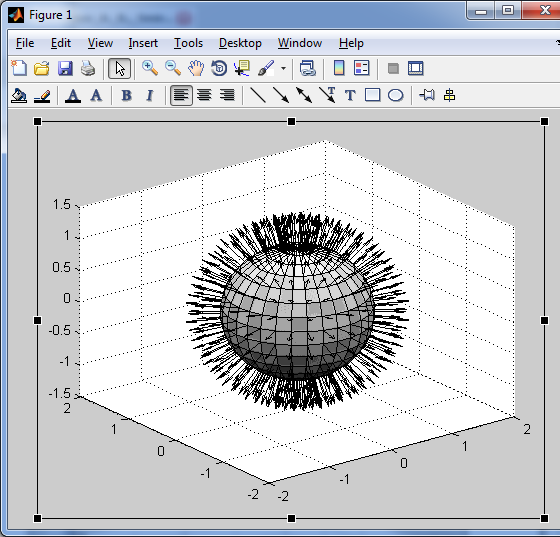
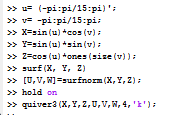
відповідно до варіантів, що представлені у таблиці:

|  |  |
| --- | --- |
| **Варіант** | **Функція** |
|  | t=[0:0.5:50] |
|  | t=[1:0.25:30] |
|  | t=[0.5:0.4:40] |
|  | t=[1:1:60] |
|  | t=[0:0.2:20] |
|  | t=[0:0.3:30] |
|  | t=[0:0.4:40] |
|  | t=[0:0.5:50] |
|  | t=[0:0.6:60] |
|  | t=[0:0.7:70] |
|  | t=[0:0.25:25] |
|  | t=[0:0.35:35] |
|  | t=[0:0.3:35] |
|  | t=[0:0.1:25] |
|  | t=[0:0.15:30] |
|  | t=[0:0.25:60] |
|  | t=[0:0.22:45] |
|  | t=[0:0.14:36] |
|  | t=[0:0.15:45] |
|  | t=[0:0.25:25] |
|  | t=[0:0.15:30] |
|  | t=[0:0.15:20] |
|  | t=[0:0,35:50] |
|  | t=[0:0.45:50] |
|  | t=[0:0.17:50] |
|  | t=[0:0.18:50] |
|  | t=[0:0.18:64] |
|  | t=[0:0.5:42] |
|  | t=[0:0.5:60] |
|  | t=[0:0.5:55] |

1. Задайте наступні параметри функції згідно варіантів таблиці та представте графік отриманої функції

|  |  |
| --- | --- |
| **Варіант** | **Функція** |
|  | quiver (x, y, ux, uy, 1.5) |
|  | quiver (x, y, ux, uy, 0) |
|  | quiver (x, y, ux, uy, ‘or--’) |
|  | quiver (x, y, ux, uy, 2.1 ‘\*k:’) |
|  | quiver (x, y, ux, uy, 1) |
|  | quiver (x, y, ux, uy, 0.3) |
|  | quiver (x, y, ux, uy, 1.2) |
|  | quiver (x, y, ux, uy, 2.5) |
|  | quiver (x, y, ux, uy, 1.7) |
|  | quiver (x, y, ux, uy, 1.9) |
|  | quiver (x, y, ux, uy, 1.6) |
|  | quiver (x, y, ux, uy, 0.5) |
|  | quiver (x, y, ux, uy, 0.9) |
|  | quiver (x, y, ux, uy, 1.1) |
|  | quiver (x, y, ux, uy, 0.7) |
|  | quiver (x, y, ux, uy, 0.2) |
|  | quiver (x, y, ux, uy, 0.4) |
|  | quiver (x, y, ux, uy, 0.8) |
|  | quiver (x, y, ux, uy, 1.4) |
|  | quiver (x, y, ux, uy, 1.3) |
|  | quiver (x, y, ux, uy, 1.8) |
|  | quiver (x, y, ux, uy, 2.2) |
|  | quiver (x, y, ux, uy, 2.6) |
|  | quiver (x, y, ux, uy, 2.4) |
|  | quiver (x, y, ux, uy, 2.8) |
|  | quiver (x, y, ux, uy, 2.9) |
|  | quiver (x, y, ux, uy, 3.1) |
|  | quiver (x, y, ux, uy, 3.3) |
|  | quiver (x, y, ux, uy, 3.7) |
|  | quiver (x, y, ux, uy, 4.0) |

1. Дослідити роботу функцій compass(ux, uy), feather(ux, uy), та представити графіки.
2. Дослідіть роботу функції **quiver3** та побудуватинаступний графік:



Покрутіть графік та отримайте ще два вигляди даного графіку.

1. Скласти програму за допомогою файлу-функції та скрипт-файлу для знайдення значень функції *yi*  для заданих значень аргументу *xi* та параметрів *a* та *b.* Результати обчислень вивести у командне вікно та побудувати графік залежності функції *yi* от аргумента *xi (yi = f(xi)).* Варіанти завдань представлено у таблиці:

**

**

*З 11 по 20 варіанти значення, наведені в таблиці множать на 1,2.*

*З 21 по 30 варіанти значення, наведені в таблиці множать на 1,5.*

1. Оформити звіт та зробити висновки.