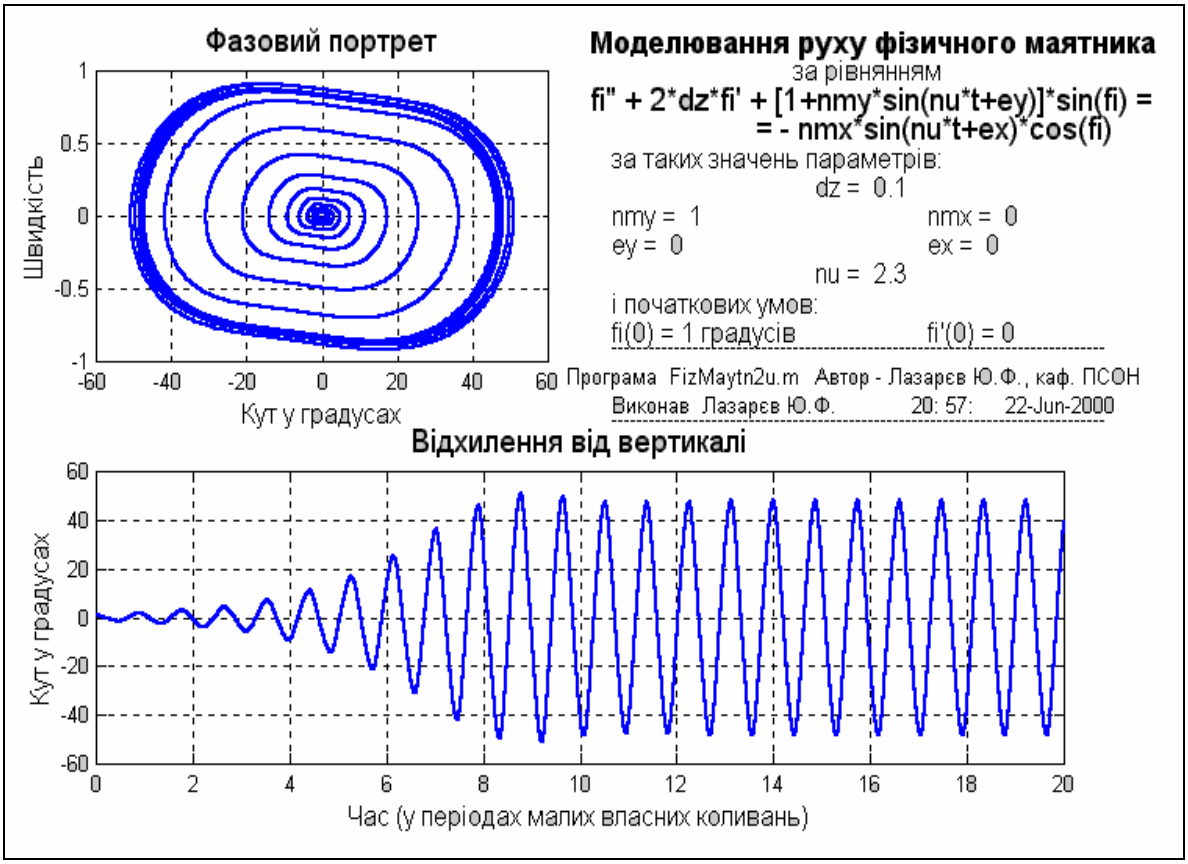
Ю. Ф. Лазарєв

MATLAB і моделювання динамічних систем

Навчальний посібник

Глава 2. Програмування у Matlab



Київ – 2009

УДК 681.3(0.75)
Л17

Лазарєв Ю. Ф.
Л17 MATLAB і моделювання динамічних систем. Навчальний посібник. Глава 2. Програмування у Matlab. – Київ: НТУУ "КПІ", 2009. – 60 с.

Зміст

2. Програмування у MatLAB ***4***

2.1. Оператори керування обчислювальним процесом **7**

2.1.1. Оператор умовного переходу 7

2.1.2. Оператор переключення 8

2.1.3. Оператори циклу 9

2.1.4. Завдання 11

2.1.5. Запитання 13

2.2. Створення найпростіших файл-функцій (процедур) **14**

2.2.1. Загальні вимоги до побудови 14

2.2.2. Типове оформлення процедури-функції 16

2.2.3. Запитання 17

2.3. Створення Script-файлів **18**

2.3.1. Основні особливості Script-файлів 18

2.3.2. Введення й виведення інформації у діалоговому режимі 19

2.3.3. Організація повторювання дій 21

2.3.4. Організація змінювання даних у діалоговому режимі 22

2.3.5. Типова структура й оформлення Script-файлу 24

2.4. Графічне оформлення результатів **26**

2.4.1. Загальні вимоги до подання графічної інформації 26

2.4.2. Розбиття графічного вікна на підвікна 27

2.4.3. Виведення тексту в графічне вікно (підвікно) 29

2.5. Функції функцій **33**

2.5.1. Стандартні функції від функцій Matlab 33

2.5.2. Завдання 35

2.5.3. Запитання 36

2.6. Створення функцій від функцій **37**

2.6.1. Процедура feval 37

2.6.2. Приклади створення процедур від функцій 38

2.6.3. Завдання 43

2.7. Приклад створення складної програми **50**

2.7.1. Програма моделювання руху маятника 50

2.7.2. Завдання 59

2.8. Література **60**

2. Програмування у MatLAB

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 правильну їхню роботу через неминучі помилки при набиранні команд. Тому складні, із перериваннями, складними переходами по певних умовах, із часто повторюваними однотипними діями обчислення, які, до того ж, необхідно проводити неодноразово при змінених первинних даних, потребують їхнього спеціального оформлення у виді записаних на диску файлів, тобто у виді програм. Перевага програм у тому, що, унаслідок того, що вони зафіксовані у виді записаних файлів, стає можливим багаторазове звернення до тих самих операторів і до програми в цілому. Це дозволяє спростити процес налагоджування програми, зробити процес обчислень більш наочним і прозорим, а завдячуючи цьому - різко зменшити можливість появи принципових помилок при розробці програм. Крім того, у програмах виникає можливість автоматизувати також і процес змінювання значень первісних параметрів у діалоговому режимі.

Створення програми в середовищі Matlab здійснюється за допомогою вбудованого редактора. Вікно цього вбудованого редактора виникає на екрані, якщо перед цим використано команду "M-file" із поділу *New* або обрано назву одного з існуючих М-файлів при виклику команди *Open M-file* із меню ***File*** командного вікна. У першому випадку вікно текстового редактора є порожнім (рис. 2.1), у другому - у ньому міститься текст викликаного М-файлу. В обох випадках вікно текстового редактора готове для введення нового тексту або коригування існуючого.

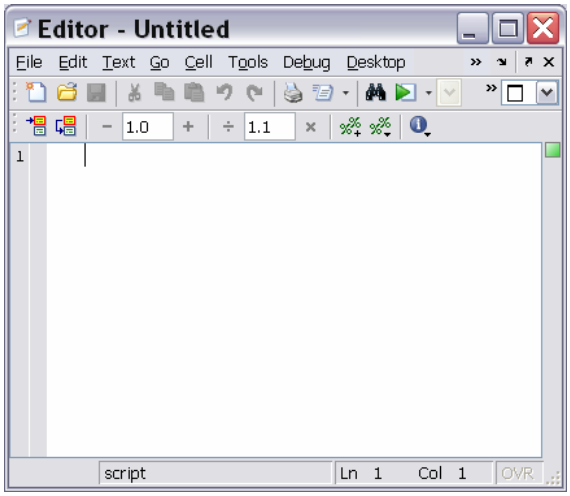


Рис. 2.1. Вікно вбудованого текстового редактора Matlab

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 MatLAB 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 "a" і "A" можуть використовуватися в одній програмі для позначення різних величин.

Програми мовою MatLAB мають два різновиди - так званий *Script-файл* (*файл-сценарій*, або *керуюча програма*) і *файл-функція (процедура)*. Обидва різ-

новиди матимуть розширення ***.m*** при записі тексту файлу на диск, тобто їх не можна розрізнити по виду ймення.

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 виді 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. Головні правила написання текстів М-файлів були наведені раніше (п. 1. 2. 8).

2.1. 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 хибою мови MatLAB і утруднює організацію повернення обчислювального процесу до будь-якого попереднього або наступного оператора програми.

Усі оператори циклу й умовного переходу побудовані в MatLAB у виді 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 ñëîâà ***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.

2.1.1. Оператор умовного переходу

Конструкція оператора переходу за умовою в загальному виді є такою:

```
if <умова>
        <оператори1>
  else
        <оператори2>
end
```

Працює оператор у такий спосіб. 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 вид:

```
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.

2.1.2. 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.***

2.1.3. 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***, який раніше застосовувався. 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 виді 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 початкове 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.

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
```

У такий спосіб можна забезпечити виведення інформації у вигляді таблиць.

2.1.4. Завдання

Завдання 2.1.

1. Відповідно до таблиці 2.1 виконати:

- обчислення точних (по стандартних функціях MatLAB) значень відповідної функції у діапазоні змінювання аргументу від x_1 до x_2 в m рівновіддалених точках цього діапазону, включаючи його межі;

- обчислення по зазначених степеневих рядах наближених значень функції у тих же точках, обмежуючись r першими членами ряду;

- розрахунок похибки наближеного визначення функції у кожній точці, порівнюючи наближене значення з точним, і побудову графіка залежності похибки від аргументу;

- обчислення наближених значень функції у тих же точках із відносною похибкою не вище ε=0.001; побудову графіка отриманих відносних похибок.

Таблиця 2.1.

Вар.	x1	x2	m	r	f(x)	Наближена формула
1	0.2	5	20	4	sin(x)	$\sum(-1)^k \frac{x^{2k-1}}{(2k-1)!}$
2	1	10	30	5	cos(x)	$1-\sum(-1)^k \frac{x^{2k}}{(2k)!}$
3	0.3	3	40	5	exp(x)	$1+\sum\frac{x^k}{k!}$
4	0.4	4	50	4	ln(1+x)	$\sum(-1)^{k-1}\frac{x^k}{k}$
5	0.5	5	30	3	ln(x)	$\sum\frac{a^k}{k}$, де $a=\frac{x-1}{x}$
6	0.6	6	40	4	ln(x)	$\sum(-1)^k\frac{a^k}{k}$, де $a=x-1$
7	0.7	7	50	5	ln(x)	$2\sum\frac{a^{2k-1}}{2k-1}$, де $a=\frac{x-1}{x+1}$
8	0.8	8	45	6	ln(x+a)	$\ln(a)+2\sum\frac{b^{2k-1}}{2k-1}$, де $b=\frac{x}{2a+x}$

9	1.1	11	40	3	ctg(x)	$\frac{1}{x}+2x\sum\frac{1}{x^2-k^2\pi^2}$
10	1.2	12	50	4	cosec(x)	$\sum\frac{1}{(x-k\pi)^2}$
11	1.3	13	50	5	cosec(x)	$\frac{1}{x}+2x\sum\frac{(-1)^k}{x^2+k^2\pi^2}$
12	1.4	14	60	6	arctg(x)	$\sum(-1)^{k-1}\frac{x^{2k-1}}{2k-1}$
13	1.5	15	45	5	arctg(x)	$\frac{\pi}{2}+\sum(-1)^k\frac{1}{(2k-1)x^{2k-1}}$
14	1.6	16	40	4	ln(x)	$\sum(-1)^{k-1}\frac{a^k}{r}$, де $a=x-1$
15	0.9	9	50	6	sin(x)	$x\prod(1-\frac{x^2}{(k\pi)^2})$
16	1	10	50	4	cos(x)	$\prod(1-\frac{x^2}{(2k-1)^2\pi^2})$
17	0.6	5	50	3	ln(x)	$\sum\frac{a^k}{k}$, де $a=(x-1)/x$
18	-0. 9	0.9	45	4	arcctg(x)	$\frac{\pi}{2}-\sum(-1)^k\frac{x^{2k-1}}{(2k-1)}$
19	1	20	50	5	sh(x)	$\sum\frac{x^{2k-1}}{(2k-1)!}$
20	1	20	50	5	ch(x)	$1+\sum\frac{x^{2k}}{(2k)!}$
21	-0. 9	0.9	50	5	arcth(x)	$\sum\frac{x^{2k-1}}{2k-1}$
22	1	20	50	5	arcth(x)	$\sum\frac{1}{(2k-1)x^{2k-1}}$
23	-0. 8	0.8	50	4	arcsin(x0	$x+\sum\frac{1\cdot3\cdot5...(2k-1)\cdot x^{2k+1}}{2\cdot4\cdot6....(2k)\cdot(2k-1)}$
24	-0. 8	0.8	50	4	arccos(x 0	$\frac{\pi}{2}-\{\ x+\sum\frac{1\cdot3\cdot5...(2k-1)\cdot x^{2k+1}}{2\cdot4\cdot6....(2k)\cdot(2k-1)}\ \}$
25	-5	5	50	6	exp(x)	$1+\sum\frac{x^k}{k!}$

Завдання 2.2. Обчисліть значення функції із *завдання 2.1* при значеннях аргументу в діапазоні від 0.1 до 100, що утворять *геометричну прогресію* зі знаменником, рівному квадратному кореню з 10, і виведіть у командне вікно таблицю результатів обчислень.

Завдання 2.3. Обчисліть значення модуля ЧПФ і її аргументу (у градусах) із *завдання 1.7* при значеннях аргументу в діапазоні від 0.1 до 100, утворять *геометричну прогресію* зі знаменником, рівним квадратному кореню з 10, і виведіть у командне вікно таблицю результатів обчислень.

2.1.5. Запитання

1. Які засоби керування перебігом обчислювального процесу передбачені в мові MatLAB?
2. Як можна організувати обчислення за циклом мовою MatLAB?
3. Як організувати виведення таблиці результатів обчислень у командне вікно MatLAB?
4. Як здійснити складні (багатооператорні) обчислення в режимі калькулятора?

2.2. Створення найпростіших файл-функцій (процедур)

2.2.1. Загальні вимоги до побудови

Як було зазначено раніше, файл-функція (процедура) має починатися з рядка заголовка виду

function [<ПКВ>] = <ім'я процедури>(<ПВВ>).

Якщо перелік кінцевих (вихідних) величин (ПКВ) містить тільки один об'єкт (у загальному випадку - матрицю), то файл-функція є звичайною функцію (однієї або кількох змінних). Фактично навіть у цьому найпростішому випадку файл-функція є вже процедурою у звичайному розумінні інших мов програмування, якщо вихідна величина є вектором або матрицею. Перший рядок у цьому випадку має вид:

function <ім'я змінної> = <ім'я процедури>(<ПВВ>).

Якщо ж у результаті виконання файл-функції мають бути визначені (обчислені) кілька об'єктів (типу матриць), така файл-функція є вже більш складним об'єктом, який у програмуванні зазвичай називається або процедурою (у мові Паскаль), або підпрограмою. Загальний вид першого рядка в цьому випадку стає таким:

function [y1, y2, ... , yN] = <ім'я процедури>(<ПВВ>),

тобто перелік вихідних величин y1, y2, ... , yN має бути поданий як вектор-рядок з елементами y1, y2, ... , yN (кожний з яких може бути матрицею).

У найпростішому випадку функції однієї змінної заголовок набуває вид:

function y = ***func***(x),

де ***func*** **-** ім'я функції (M-файлу).

Як приклад розглянемо складання M-файлу для функції

$$y = f_1(x) = d^3 \cdot ctg(x) \cdot \sqrt{sin^4(x) - cos^4(x)} \quad .$$

Для цього треба активізувати поділ "File" головного меню командного вікна MatLAB, вибрати в підменю, що виникло на екрані, розділ "New", а потім - команду "M-file". На екрані з'явиться вікно текстового редактора. У ньому потрібно набрати такий текст:

```
function y = F1(x,d)
 %        Процедура,  яка обчислює значення функції
 %          y = (d^3)*ctg(x)*sqrt(sin(x)^4-cos(x)^4).
 %                  Звернення   y = F1(x,d).
y = (d^3)*cot(x). *sqrt(sin(x). ^4-cos(x). ^4);
```

Після цього необхідно зберегти цей текст у файлі під ім'ям "F1.m". Необхідний M-файл створений. Тепер можна користуватися цією функцією при розрахунках. Так, якщо ввести команду

```
» y = F1(1, 0.1)
```

то одержимо результат

```
y =  4. 1421e-004.
```

Варто зауважити, що аналогічно можна одержати одразу вектор усіх значень зазначеної функції при різних значеннях аргументу, якщо останні зібрати в деякий вектор. Так, якщо сформувати вектор

» zet= 0:0. 3:1. 8;

і звернутися до тієї ж процедури

» my = F1(zet,1),

те одержимо:

```
Warning: Divide by zero
my =
  Columns 1 through 4
     Na + Inf i 0 + 2. 9369i        0 + 0. 8799i   0. 3783
  Columns 5 through 7
   0. 3339           0. 0706           -0. 2209
```

Примітки.

1. *Можливість використання сформованої процедури як для окремих чисел, так і для векторів і матриць обумовлена застосуванням у запису відповідного М-файлу замість звичайних знаків арифметичних дій їхніх аналогів із попередньою крапкою.*

2. Щоб уникнути виведення на екран небажаних проміжних результатів, *необхідно в тексті процедури усі обчислювальні оператори завершувати символом* " ; ".

3. Як показують наведені приклади, *імена змінних, зазначені в заголовку файл-функції можуть бути будь-якими* (збігатися чи ні з іменами, використовуваними при зверненні до цієї файл-функції), тобто мають формальний характер. Важливіше за все лише те, щоб структура звернення цілком відповідала структурі заголовка в запису тексту М-файлу і щоб змінні в цьому зверненні мали той тип і розмір, що й в заголовку М-файлу.

Щоб одержати інформацію про створену процедуру, достатньо набрати в командному вікні команду:

» help f1,

і в командному вікні з'явиться

```
Процедура, яка  обчислює значення функції
 y = (d^3)*ctg(x)*sqrt(sin(x)^4-cos(x)^4).
        Звернення   y = F1(x,d).
```

Інший приклад. Побудуємо графік двох функцій:

y1 = 200 sin(x)/x; ***y2 = x2.***

Для цього створимо М-файл, що обчислює значення цих функцій:

```
function y = myfun(x)
 % Обчислення двох функцій
 %  y(1) = 200 sin(x)/x,          y(2) = x2.
y(:,1) = 200*sin(x) . / x;
y(:,2) = x . ^ 2;
```

Тепер побудуємо графіки цих функцій:

```
» fplot('myfun', [-20 20], 50, 2), grid
» set(gcf,'color','white'); title('Графік  функції  "MYFUN"')
```

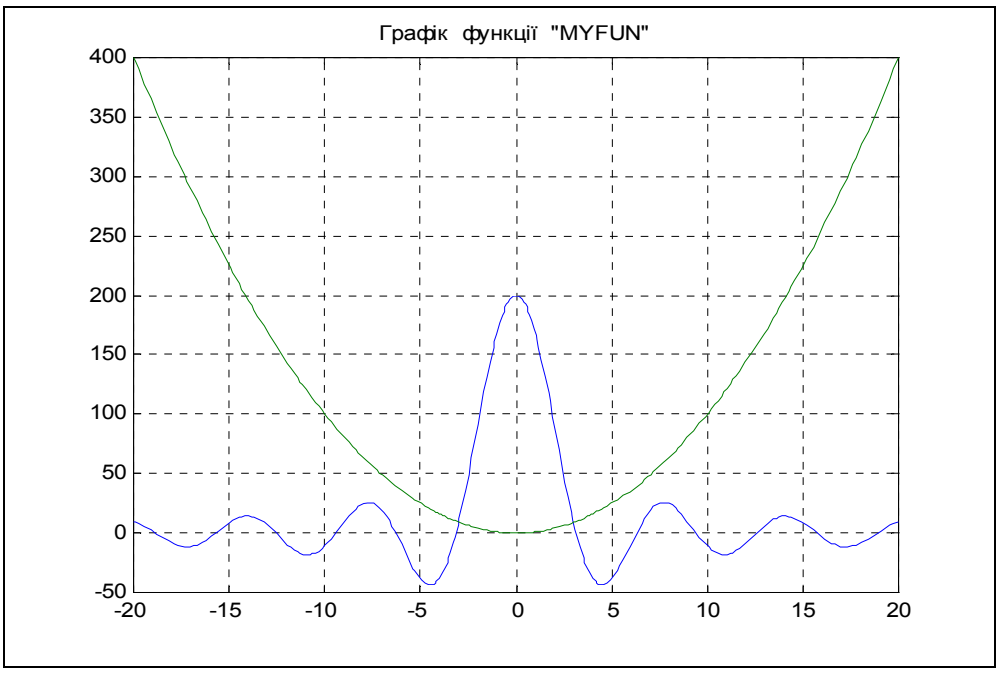


Рис. 2.2. Побудова графіків власних функцій

Результат зображений на рис. 2.2.

Третій приклад - створення файл-функції, що обчислює значення функції

$$y(t) = k1+k2*t+k3*\sin(k4*t+k5).$$

У цьому випадку зручно об'єднати сукупність коефіцієнтів ***k*** у єдиний вектор К:

К = [k1 k2 k3 k4 k5]

і створити такий М-файл:

```
function y = dvob(x, K)
 % Обчислення функції
 % y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5)),
 % де К - вектор із п'яти елементів
%
  %  Використовується для визначення поточних значень
 %        параметрів руху рухомого об'єкта
  y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5));
```

Тоді розрахунок, наприклад, 11-ти значень цієї функції можна здійснити так

```
» K = ones(1,5);
» t = 0:1:10;
» fi = dvob(t, K)
```

fi = 1. 8415 2. 9093 3. 1411 3. 2432 4. 0411 5. 7206 7. 6570 8. 9894 9. 4560 10. 0000

2.2.2. Типове оформлення процедури-функції

Рекомендується оформляти М-файл процедури-функції за такою схемою:

```
function [<Вихід>] = <ім'я функції>(<Вхід>)
  % <Стисле пояснення, що робить процедура>
  % Вхідні змінні
   %<Детальне пояснення про зміст, типи і розміри
   %  кожній із змінних, перерахованих у переліку  <Вхід>
  % Вихідні змінні
```

```
%  <Детальне пояснення про зміст, типи і розміри
%  кожної із змінних переліку  <Вихід>
% і величини, що використані у процедурі як глобальні>
%  Використання інших функцій і процедур
%<Розділ заповнюється, якщо процедура містить звернення
%  до інших процедур, крім умонтованих>
        < П о р о ж н і й   р я д о к >
% Автор : <Вказується автор процедури, дата створення кінцевого варіанта
% процедури й організація, у якій створена програма>
< Т е к с т       п р о ц е д у р и >
```

Тут позначено: <Вихід> - перелік вихідних змінних процедури, <Вхід> - перелік вхідних змінних, розділених комами.

Примітка. При використанні команди ***help*** <ім'я процедури> у командне вікно виводяться рядки коментарю *до першого порожнього рядка.*

2.2.3. Запитання

1. Для чого створюються програми в середовищі MatLAB?
2. Чим відрізняються файли-функції від Script-файлів? Яка сфера застосування кожного з цих видів файлів?
3. Як створити М-файл процедури або функції?
4. Які основні правила написання текстів М-файлів у мові MatLAB?

2.3. Створення Script-файлів

2.3.1. Основні особливості Script-файлів

Перелікуємо головні особливості Script-файлів:

■ Script-файли є блоками операторів і команд, що незалежно (самостійно) виконуються;

■ усі змінні, які використовуються в них, утворюють так називаний *робочий простір*, що є загальним для всіх Script-файлів, що виконуються; із цього випливає, що при виконанні кількох Script-файлів імена змінних у них мають бути узгоджені, щоб одне ім'я означало в кожному з них той самий об'єкт обчислень;

■ в них немає заголовка, тобто першого рядка визначеного виду й призначення;

■ звернення до них не потребує вказівки ніяких імен змінних: усі змінні формуються в результаті виконання програми або сформовані раніше й існують у робочому просторі.

Необхідно відзначити, що *робочий простір Script-файлів є недосяжним для файл-функцій*, які використовуються в ньому. У файл-функціях неможливо використовувати значення, що одержують змінні в Script-файлі, обходячи заголовок файл-функції (через те, що всі змінні файл-функції є *локальними*). Єдиною можливістю зробити так, щоб усередині файл-функції деяка змінна робочого простору могла бути застосованою, зберігши своє значення й ім'я, є спеціальне оголошення цієї змінної у Script-файлі як *глобальної* за допомогою службового слова ***global***. Крім того, аналогічний запис має бути розташований й у тексті М-файлу тієї файл-функції, яка буде використовувати значення відповідної змінної Script-файлу.

Наприклад, можна перебудувати файл-функції першого й третього прикладів із попереднього розділу, уводячи коефіцієнти відповідних функцій як глобальні змінні:

```
function y = dvob1(x)
 % Обчислення функції
 % y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5)),
 % де ДО - глобальний вектор із п'ятьох елементів
 %   Застосовується для визначення поточних значень
 %       параметрів руху рухомого об'єкта
global K
  y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5));
```

Щоб використати нову файл-функцію ***dvob1*** у Script-файлі, в останньому перед зверненням до цієї функції має бути записаний рядок

```
global K
```

і визначений вектор-рядок К із п'ятьох елементів (задані їхні значення).

Якщо в одному рядку оголошуються *кілька змінних* як глобальні, вони *повинні відокремлюватися прогалинами* (*не комами!*).

2.3.2. Введення й виведення інформації у діалоговому режимі

Для забезпечення взаємодії з користувачем у процесі виконання М-файлу в системі MatLAB призначені такі команди:

disp, sprintf, input, menu, keyboard, pause.

Команда ***disp*** здійснює виведення значень зазначеної змінної або зазначеного тексту в командне вікно. Звернення до неї має вид:

disp (<змінна або текст в апострофах>).

Особливістю цієї команди є те, що аргумент у неї може бути тільки один. Тому неможливо без спеціальних заходів здійснити виведення кількох змінних і, особливо, об'єднання тексту з чисельними значеннями деяких змінних, що часто є необхідним для зручного подання інформації.

Для усунення цієї хиби використовують декілька засобів.

Щоб вивести значення кількох змінних у єдиний рядок (це необхідно при створенні таблиць даних), потрібно створити єдиний об'єкт, що містив би всі ці значення. Це можна зробити, об'єднавши відповідні змінні у вектор, користуючись операцією створення вектора-рядка:

x = [x1 x2 ... xN].

Тоді виведення значень кількох перемінних в один рядок буде мати вид:

disp ([x1 x2 ... xN]).

Наведемо приклад:

```
» x1=1.24; x2=-3. 45;    x3=5.76; x4=-8. 07;
» disp([x1 x2 x3 x4])
   1. 2400   -3. 4500    5. 7600   -8. 0700.
```

Аналогічно можна об'єднувати декілька текстових змінних, наприклад:

```
» x1=' psi  ';   x2='  fi   ';    x3='  teta  '; x4='   w1  ';
» disp([x1 x2 x3 x4])
 psi    fi     teta     w1
```

Значно складніше об'єднати в один рядок текст і значення змінних, що також часто є необхідним. Труднощі виникають тому, що текстові і числові змінні не можуть бути об'єднані, бо є даними різних типів. Одним із шляхів подолання цієї перешкоди є переклад числового значення змінної у символьну (текстову) форму. Це можливо, якщо скористатися функцією ***num2str***, яка здійснює таке перетворення. Запис

y = ***num2str****(x)*

перетворить числове значення змінної "x" на її текстове подання, наприклад:

```
» x = -9. 30876e-15
x = -9. 3088e-015
» y = num2str(x)
y = -9. 309e-015
```

При цьому *форма символьного подання визначається встановленим режимом виведення чисел на екран* (Numeric Format).

Якщо T - текстова змінна, або деякий текст, а X - числова змінна, то виведення їх в однім рядку можна забезпечити зверненням

disp ([T num2str(X)]).

Розглянемо приклад:

```
x = -9. 3088e-015
```

```
» T = 'Значення параметра дорівнює ';
» disp([T x])
Значення параметра дорівнює
» disp([T num2str(x)])
Значення параметра дорівнює -9. 309e-015
```

Як випливає з цього приклада, "механічне" об'єднання текстової й числової змінних не приводить до бажаного результату, а використання функції ***num2str*** - приводить.

Іншим засобом досягнення того ж результату є використання функції ***sprintf.*** Звернення до неї має вид:

Y = ***sprintf*** (‘<текст1> **%g** <текст2>’, X).

У результаті утворюється текстовий рядок Y, що складається з тексту, зазначеного в <текст1>, і значення числової змінної X в відповідності з форматом %g, причому текст із фрагмента <текст2> розташовується після значення змінної X. Цю функцію можна використовувати в команді ***disp*** у виді:

disp (***sprintf*** (‘<текст> %g', X)).

Приклад:

```
» disp(sprintf('Параметр1 =  %g ',x))
Параметр1 =  -9. 30876e-015
```

Уведення інформації із клавіатури в діалоговому режимі можна здійснити за допомогою функції ***input***. Звернення до неї виду:

x = ***input***(‘<запрошення>’)

приводить до таких дій комп‘ютера. Виконання операторів програми припиняється. Комп‘ютер переходить у режим очікування закінчення введення інформації із клавіатури. Після закінчення введення із клавіатури (яке визначається натисканням клавіші <Enter>) уведена інформація запам'ятовується в програмі під ім'ям "x", і виконання програми продовжується.

Зручним інструментом вибору деякої з альтернатив майбутніх обчислювальних дій є функція ***menu*** MatLAB, яка утворює вікно меню користувача. Функція ***menu*** має такий формат звернення:

```
k=menu(‘Заголовок меню’,’Альтернатива1','Альтернатива2', 'Альтернатива n').
```

Таке звернення приводить до появи на екрані дисплея меню, зображеного на рис. 2.3.

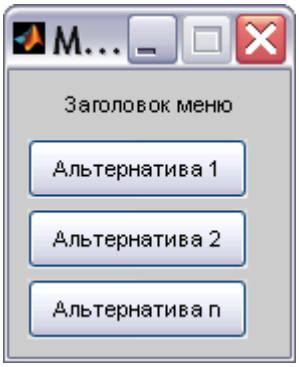


Рис. 2.3. Вид меню

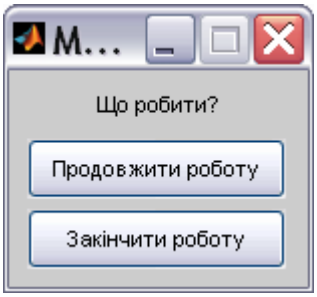


Рис. 2.4. Меню продовження роботи

Виконання програми тимчасово припиняється. Система очікує відповіді у виді натискання лівої клавіші миші після встановлення курсору на одну із зображених "кнопок" меню з альтернативами. Після правильної відповіді вихідному параметрові "k" присвоюється значення номера обраної альтернативи (1, 2 або 3). У загальному випадку число альтернатив може бути до 32.

Тепер, у залежності від отриманого значення цього параметра, можна побудувати процес розгалуження обчислень, наприклад, вибору параметра, значення якого потрібно змінити.

Команда ***pause*** тимчасово припиняє виконання програми доти, поки користувач не натисне будь-яку клавішу клавіатури. Якщо після назви команди зазначити в скобках деяке додатне ціле число ***n***, то затримка виконання програми буде здійснена протягом ***n*** секунд.

Якщо в тексті М-файлу зустрічається команда ***keyboard,*** то при виконанні програми виконання М-файлу припиняється, і керування передається клавіатурі дисплея. Цей спеціальний режим роботи супроводжується появою у командному вікні MatLAB нового виду запрошення до дій

k>>.

У цьому режимі користувач може здійснити будь-які дії, перевірити або змінити дані. При цьому йому доступні всі команди й процедури системи MatLAB. Для завершення роботи в цьому режимі необхідно набрати команду ***return.*** Тоді система продовжить роботу програми з оператора, що є наступним після оператора ***keyboard.***

2.3.3. Організація повторювання дій

Однією з основних задач створення самостійної програми є забезпечення повертання до початку програми з метою продовження її виконання при нових значеннях первісних даних.

Нехай основні оператори створеної програми розташовані в Script-файлі з ім'ям "ScrFil_yadro. m". Тоді схема забезпечення повертання до початку виконання цього Script-файлу може бути, приміром, такою:

```
flag =0;
while flag == 0
      ScrFil_yadro
      kon=0;
      kon=input('Закінчити роботу-<3>, продовжити - <Enter>');
      if kon==3,
            flag=3;
      end
end
```

У цьому випадку Script-файл "ScrFil_yadro" буде повторно виконуватися доти, поки на питання **'Закінчити роботу-<3>, продовжити - <Enter>'** не буде введено із клавіатури відповідь "3". Якщо ж відповідь буде саме такою, цикл закінчиться й будуть виконуватися наступні за цим циклом оператори. Природно,

що змінна ***flag*** не повинна змінювати своє значення в Script-файлі "ScrFil_yadro".

Можна також із тією ж метою використовувати механізм створення меню. У цьому випадку програму можна подати, приміром, так:

```
k=1;
while k==1
  ScrFile_Yadro
  k = menu(' Що робити ? ',' Продовжити роботу ', ' Закінчити роботу ');
end
```

Тоді, після першого виконання Script-файлу "ScrFil_Yadro" на екрані з'явиться вікно меню, зображене на рис. 2.4, і, при натисканні кнопки першої альтернативи значення k залишиться рівним одиниці, цикл повториться, а при натисканні другої кнопки k стане рівним 2, цикл закінчиться і програма перейде до закінчення роботи.

2.3.4. Організація змінювання даних у діалоговому режимі

Повторення дій, що містяться в ядрі "ScrFil_yadro", має сенс тільки у випадку, коли на початку цього ядра забезпечене виконання дій по змінюванню деяких із первинних величин. MatLAB містить ряд зручних засобів, які дозволяють здійснювати змінювання даних у діалоговому режимі з використанням стандартних меню-вікон користувача.

Організацію діалогового змінювання даних розглянемо на прикладі деяких 5 параметрів, які назвемо *Параметр1*, *Параметр2*, ... , *Параметр5*. Нехай їхні позначення як змінних у програмі такі: x1, x2, ... , x5. Тоді меню вибору параметра для змінювання його значення повинно містити 6 альтернатив: 5 із них призначені для вибору одного із зазначених параметрів, а остання альтернатива повинна надати можливість виходу з меню, коли значення всіх параметрів установлені.

Тому варіант оформлення такого меню може бути, наприклад, наступним:

```
k = menu('   Що змінити ?  ', ' Параметр1 ', ' Параметр2 ', ...
         '  Параметр3  ', '  Параметр4 ', '  Параметр5  ', '  Нічого не змінювати '),
```

що приведе до появи вікна, поданого на рис. 2.5.

Легко помітити хиби такого оформлення вікна меню. Щоб прийняти рішення, значення якого саме параметра варто змінити і як, користувач має мати перед очима не тільки перелік параметрів, які можна змінити, але й поточні значення цих параметрів. Тому на кожній кнопці меню має міститися також інформація про поточне значення відповідного параметра. Це можна зробити, використовуючи раніше згадану функцію ***sprintf***, наприклад, у такий спосіб:

```
x1=-1.89;  x2=239.78;  x3=-2.56e-3;    x4=7.28e-15;    x5=1.023e-32;
k = menu( '   Що  змінювати ?   ', ...
sprintf (' Параметр1    x1 = %g', x1),...
```

```
sprintf (' Параметр2    x2 = %g', x2),...
sprintf (' Параметр3    x3 = %g', x3),...
sprintf (' Параметр4    x4 =  %g', x4),...
sprintf (' Параметр5    x5 =  %g', x5),...
 '  Нічого не змінювати  ')
```

Результат наведено на рис. 2.6.

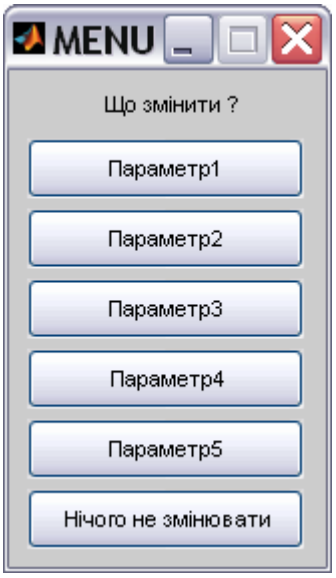


Рис. 2.5. Варіант 1 меню

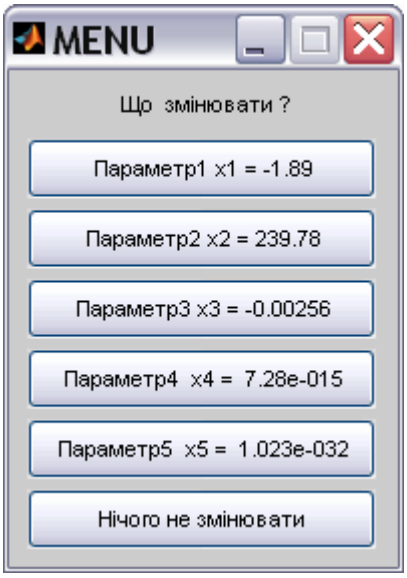


Рис. 2.6. Варіант 2 меню

Меню дозволяє обрати параметр, який потрібно змінити, проте не забезпечує самого змінювання обраного параметра. Це змінювання повинно бути здійснено за допомогою введення нового значення із клавіатури, приміром, так:

x = ***input***([***sprintf*** ('Поточне значення x =%g',x), 'Нове значення x = ']).

Наприклад, якщо ввести команди

```
» x = 3. 02e-2;
» x=input( [sprintf('Поточне значення  x = %g',x),   ' Нове  значення x = ']
```

то в командному вікні з'явиться напис:

```
Поточне значення  x = 0. 0302   Нове значення  x =
```

Виконання програми припиниться. ПК буде очікувати закінчення введення інформації із клавіатури. Якщо тепер набрати на клавіатурі "0.073" і натиснути клавішу <Enter>, то в командному вікні з'явиться напис:

```
Поточне значення  x = 0. 0302   Нове значення  x = 0. 073
x =    0. 0730
```

Щоб запобігти повторному виведенню на екран уведеного значення, *необхідно рядок із функцією* ***input*** *завершити символом* " ; ".

Тепер слід організувати вибір різних видів такого типу операторів відповідно до окремих обраних параметрів. Для цього можна використовувати оператор умовного переходу, наприклад, так:

```
if k==1,
        x1 = input( [sprintf( 'Поточне значення x1  = %g', x1) ...
                                  '  Нове значення  x1= ']);
   elseif k==2,
        x2 = input( [sprintf('Поточне значення  x2  = %g', x2) ...
                                  '  Нове значення  x2= ']);
   elseif k==3,
        x3 = input( [sprintf('Поточне значення  x3  = %g', x3) ...
                                  '  Нове значення  x3= ']);
```

```
    elseif k==4
        x4 = input( [sprintf('Поточне значення  x4  = %g', x4) ...
                                    '  Нове значення  x4= ']);
    elseif k==5
        x5 = input( [sprintf('Поточне значення  x5  = %g', x5) ...
                                    '  Нове значення  x5= ']);
end
```

Щоб можна було проконтролювати правильність введення нових значень, забезпечити можливість їхнього коригування і послідовного змінювання всіх бажаних параметрів, потрібно, щоб після введення нового значення будь-якого параметра на екрані знову виникало те саме вікно меню, але вже зі скоригованими значеннями. При цьому кінець роботи з меню має настати тільки за умови вибору останньої альтернативи меню "Нічого не змінювати", що відповідає значенню k, рівному 6.

Тому попередні оператори варто замкнути в цикл:

```
k=1;
while k<6
k = menu( ' Що змінити ?   ', ...
sprintf (' Параметр1 x1 = %g', x1),...
sprintf (' Параметр2 x2 = %g', x2),...
sprintf (' Параметр3 x3 = %g', x3),...
sprintf (' Параметр4 x4 = %g', x4),...
sprintf (' Параметр5 x5 = %g', x5), ' Нічого не змінювати ');
    if k==1,
            x1 = input( [sprintf('Поточне значення  x1  = %g', x1) ...
                                    '  Нове значення  x1= ']);
        elseif   k==2,
            x2 = input( [sprintf('Поточне значення  x2  = %g', x2) ...
                                    '  Нове значення  x2= ']);
        elseif   k==3,
            x3 = input( [sprintf('Поточне значення  x3  = %g', x3) ...
                                    '  Нове значення  x3= ']);
        elseif    k==4
            x4 = input( [sprintf('Поточне значення  x4  = %g', x4) ...
                                    '  Нове значення  x4= ']);
        elseif   k==5
            x5 = input( [sprintf('Поточне значення  x5  = %g', x5) ...
                                    '   Нове значення  x5= ']);
    end
end
```

У такий спосіб організується досить зручне діалогове змінювання значень параметрів.

Якщо вхідних параметрів, значення яких потрібно змінювати, досить багато, варто об'єднати такі параметри в компактні групи (бажано - за якоюсь загальною властивістю, яка відрізняє визначену групу від інших) і аналогічним чином забезпечити діалогове змінювання, використовуючи окреме меню для кожної групи. Очевидно, при цьому необхідно попередньо забезпечити вибір однієї з цих груп параметрів через додаткове вікно меню.

2.3.5. Типова структура й оформлення Script-файлу

При написанні тексту програми у виді Script-файлу необхідно брати до уваги наступне.

1. Зручніше оформляти весь процес діалогового змінювання значень параметрів у виді окремого Script-файлу, приміром, за ім'ям "ScrFil_Menu", де під скороченням "ScrFil" розуміється ім'я основного (збірного) Script-файлу.

2. Через те що вже на самому початку роботи із програмою в меню вибору змінюваного параметра, яке з'являється на екрані, мають виводитися деякі значення параметрів, перед головним циклом програми, який забезпечує повертання до початку обчислень, необхідно розмістити частину програми, яка б задавала первісні значення всіх параметрів. Крім того, на початку роботи програми дуже зручно вивести на екран стислу інформацію про призначення програми, більш детальну інформацію про досліджувану математичну модель з указівкою місця в ній і змісту усіх первинних параметрів, а також первісні ("ушиті") значення всіх параметрів цієї моделі. Це бажано також оформити у виді окремого Script-файлу, наприклад, під ім'ям "ScrFil_Zastavka".

3. При завершенні роботи програми зазвичай виникає потреба дещо упорядкувати робочий простір, наприклад, очистити його від уведених глобальних змінних (вони, залишаючись у робочому просторі, перешкоджають правильній роботі іншої, наступної програми, яка може мати зовсім інші глобальні змінні, або такі ж за ім'ям, але зовсім інші за типом, змістом і значенням), закрити відкриті програмою графічні вікна (фігури) і т.д. Цю завершальну частину теж можна оформити як окремий Script-файл, наприклад, назвавши його "ScrFil_Kin".

У цілому типова схема оформлення Script-файлу окремої програми може бути подана в такому виді:

```
%    <Позначення Script-файлу (ScrFil.m)>
% <Текст коментарю з описом призначення програми>
      < Порожній рядок >
%    Автор < Прізвище І.  Б., дата створення, організація>

 ScrFil_Zastavka
k = menu(' Що робити ? ','Продовжити роботу ', ' Закінчити роботу   ');
if k==1,
          while k==1
             ScrFil_Menu
             ScrFile_Yadro
              k = menu(' Що  робити ? ',' Продовжити роботу  ', ' Закінчити роботу   ');
         end
end
ScrFil_Kin
```

2.4. Графічне оформлення результатів

2.4.1. Загальні вимоги до подання графічної інформації

Обчислювальна програма, утворювана інженером-розробником, призначена, по-більшості, для дослідження поводження розроблювального устрою за різноманітних умов його експлуатації, при різних значеннях його конструктивних параметрів або для розрахунку певних параметрів його поводження. Інформація, одержувана в результаті виконання обчислювальної інженерної програми, як правило, має форму деякого ряду чисел, кожний з яких відповідає певному значенню деякого параметра (аргументу). Таку інформацію зручніше усього узагальнювати й подавати в графічній формі.

Вимоги до оформлення інженерної графічної інформації відрізняються від вимог до звичайних графіків у математиці. Зазвичай на основі поданої графічної інформації користувач-інженер має мати змогу прийняти якесь інженерне рішення про вибір значень певних конструктивних параметрів, що характеризують досліджуваний процес або технічний устрій, із метою, щоб прогнозоване поводження технічного устрою задовольняло певні задані умови. Тому інженерні графіки мають бути, як говорять, "читабельними", тобто мати такий вид, щоб з них легко було "відлічувати" значення функції при будь-яких значеннях аргументу і навпаки з відносною похибкою в декілька відсотків. Це стає можливим, якщо координатна сітка графіків відповідає певним цілим числам якогось десяткового розряду. Як уже раніше відзначалося, *графіки, побудовані системою MatLAB, цілком відповідають цим вимогам*.

Крім цього, інженерна графічна інформація має супроводжуватися достатньо детальним описом, із якого має стати зрозуміло, який об'єкт і за якою математичною моделлю досліджується, наведені числові значення параметрів досліджуваного об'єкта і математичної моделі. Не є зайвим і вказівка імені програми, за допомогою якої отримана ця графічна інформація, а також зведень про автора програми й дослідника, щоб користувачеві було зрозуміло, до кого і куди треба звертатися для наведення довідок про отриману інформацію.

Задачею інженерної програми часто є порівняння декількох функцій, отриманих при різних сполученнях конструктивних параметрів або параметрів зовнішніх дій. Таке порівняння зручніше і наочніше проводити, якщо згадані функції подані у виді графіків.

При цьому потрібно звернути увагу на наступне:

1) якщо потрібно порівнювати графіки функцій одного аргументу, діапазони змінювання яких не надто відрізняються один від одного (не більш ніж на десятковий порядок, тобто не більш ніж у десять разів), порівняння зручніше усього здійснювати по графіках цих функцій, побудованих в одному графічному полі (тобто у загальних координатних осях); у цьому випадку *варто виводити графіки за допомогою однієї функції* ***plot***;

2) якщо за тих самих умов діапазони змінювання функцій значно різняться, можна запропонувати два підходи:

- якщо всі порівнювані функції є значеннями величин однакової фізичної природи і ці значення усі додатні, графіки варто виводити також в одне графічне поле, але *в логарифмічному масштабі* (тобто використовувати процедуру ***semilogy***)*;*

- *за умови, що усі функції мають різну фізичну природу*, але аргумент у них загальний і змінюється в однім діапазоні, графіки *потрібно будувати в одному графічному вікні (фігурі), але в різних графічних полях* (користуючись для цього декількома окремими зверненнями до функції ***plot*** у різних підвікнах графічного вікна, що забезпечується застосуванням процедури ***subplot***); при цьому зручно *розміщувати окремі графіки один під одним таким чином, щоб однакові значення аргументу у всіх графіках розташовувалися на одній вертикалі;*

3) крім згаданих раніше написів у кожному графічному полі, закінчене графічне оформлення будь-якого графічного вікна (фігури) обов'язково повинно містити *додаткову текстову інформацію* такого вмісту:

- стисле повідомлення про об'єкт дослідження;
- математична модель, закладена основу здійснених у програмі обчислень з указівкою імен параметрів і змінних;
- інформація про використані значення параметрів;
- інформація про отримані значення деяких обчислених інтегральних параметрів;
- інформація про ім'я М-файлу використаної програми, виконавця роботи й дату проведення обчислювального експерименту;
- інформація про автора використаної програми й організації, де він працює.

Остання вимога ставить перед необхідністю відокремлювати (за допомогою тієї ж процедури ***subplot***) у кожній фігурі місце для виведення зазначеної текстової інформації.

2.4.2. Розбиття графічного вікна на підвікна

Як випливає зі сказаного, при створенні закінченого графічного інженерного документа в системі MatLAB необхідно використовувати процедуру ***subplot***. Загальне призначення й застосування функції ***subplot*** описані в поділі. 1.5.3.

Розглянемо, як забезпечити бажане розбиття всього графічного вікна на окремі поля графіків і текстове підвікно.

Нехай потрібно розбити усе поле графічного вікна так, щоб верхня третина вікна утворила поле виведення тексту, а нижні дві третини утворили єдине поле виведення графіків. Це можна здійснити таким чином:

- перед виведенням текстової інформації у графічне вікно треба встановити команду ***subplot***(3,1,1), яка показує, що весь графічний екран, розділений на три однакові частини по вертикалі, а для наступного виведення буде використана верхня з цих трьох частин (рис. 2.7);

Рис. 2.7. Схема 1 розташування підвікон у графічному вікні

- виведенню графіків у графічне вікно має передувати команда ***subplot***(3,1,[2 3]), відповідно до якої графічне вікно розділяється, як і раніше, на три частини по вертикалі, але тепер для виведення графічної інформації буде використаний простір, що об'єднує друге й третє зі створених підвікон (полів) (зверніть увагу, що *підвікна об'єднуються таким самим чином, як елементи вектора у вектор-рядок*).

Рис. 2.8. Схема 2 розташування підвікон у графічному вікні

Щоб створити три окремих поля графіків один під іншим на трьох чвертях екрана по горизонталі, а текстову інформацію розмістити в останній чверті по горизонталі, то це можна зробити (рис. 2.8) так:

1) поділити весь простір фігури на 12 частин - на 3 частини по вертикалі і на 4 частини по горизонталі; при цьому підвікна будуть розташовані так, як показано на рис. 2.7;
2) щоб організувати виведення графіків у перше графічне підвікно, треба попередньо ввести команду ***subplot***(3,4,[1 2 3]), що об'єднує підвікна sp1, sp2 і sp3 у єдине графічне підвікно;
3) аналогічно, виведенню графіків у друге графічне підвікно повинно передувати звернення до команди ***subplot***(3,4,[5 6 7]), а виведенню графіків у третє графічне підвікно - ***subplot***(3,4,[9 10 11]);
4) нарешті, до оформлення тексту можна приступити після звернення ***subplot***(3,4,[4 8 12]).

2.4.3. Виведення тексту в графічне вікно (підвікно)

Якщо по черзі сформувати підвікна, приміром, відповідно до останньої схеми, не здійснюючи ніяких операцій по виведенню графіків або тексту:

```
» subplot(3,4,[5 6 7])
» subplot(3,4,1:3)
» subplot(3,4,9:11)
» subplot(3,4,[4 8 12]),
```

у вікні фігури з'явиться зображення, подане на рис. 2.9.

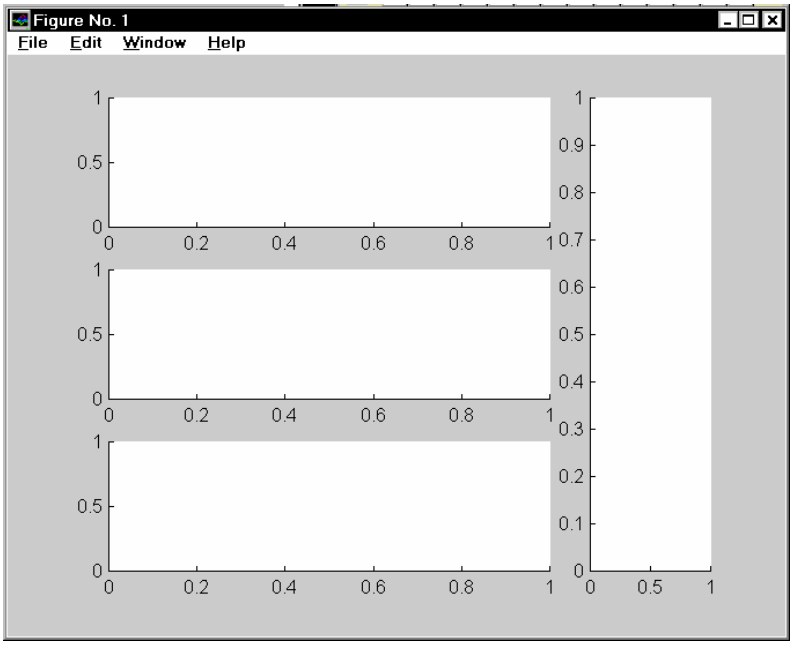


Рис. 2.9. Вид графічного вікна після розбиття його на підвікна

З його розгляду випливає:

1) після звернення до процедури ***subplot*** у відповідному підвікні з'являється зображення осей координат із позначенням поділків по осях;

2) початковий діапазон змінювання координат по обох осях підвікна завжди встановлюється за замовчуванням від 0 до 1;

3) поле виведення графіків займає не весь простір відповідного підвікна , - залишається деяке місце навколо поля графіка для виведення заголовка графіка, написів по осях координат та ін.

Тому для виведення тексту в одне з підвікон потрібно спочатку очистити це підвікно від зображення осей координат і написів на них. Це робиться за допомогою команди

axis('off').

Наприклад, якщо цю команду ввести після попередніх команд, у вікні фігури зникне зображення координатних осей останнього підвікна (рис. 2.10).

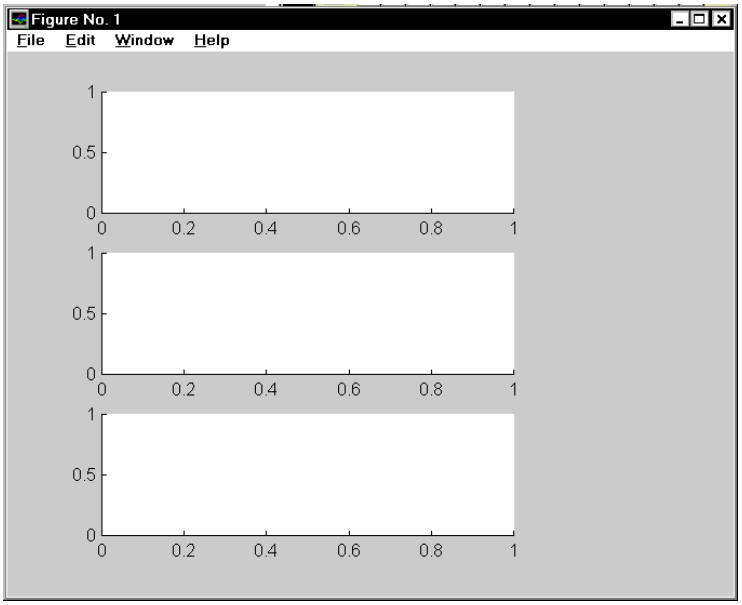


Рис. 2.10. Вид графічного вікна після команди ***axis('off').***

Тепер можна починати виведення текста в це підвікно.

Основною функцією, яка забезпечує виведення тексту в графічне вікно, є функція ***text.*** Узагальнена форма звернення до неї має вид:

h = **text** (x, y, '<текст>','FontName','<назва шрифту>', 'FontSize',<розмір шрифту в пикселах>).

Вона здійснює виведення зазначеного тексту зазначеним шрифтом зазначеного розміру, починаючи з точки підвікна з координатами x і y відповідного поля графіка підвікна. При цьому координати x і y вимірюються в одиницях величин, які відкладаються уздовж відповідних осей графіка підвікна. Через те що, як ми переконалися, діапазон змінювання цих координат дорівнює [0...1], то для того, щоб помістити початок тексту в точку усередині поля графіка, необхідно, щоб його координати x і y були в цьому діапазоні.

Проте можна використовувати і дещо ширший діапазон, враховуючи те, що поле підвікна більше поля його графіка.

Розглянемо приклад текстового оформлення. Нехай частина програми має вид:

```
subplot(3,4,1:3);subplot(3,4,5:7);   subplot(3,4,9:11); subplot(3,4,[4;8;12]);
axis('off');
 % Процедура виведення даних в текстове поле графічного вікна
 D1 =[2 1 300 1 50];
 D2 = [ 0.1 0.02 -0.03 0 1 4 -1.5 2 0.1 -0.15 0 0];
```

```
 D5 = [0.001 0.01 15 16];
sprogram = 'vsp1'; sname = 'Лазарев Ю.Ф.';
h1=text(-0.2,1,'Исходные параметры:','FontSize',12);
h1=text(0,0.95,'Гиротахометров','FontSize',10);
h1=text(0.2,0.9,sprintf(' = %g ',D1(3)),'FontSize',10);
h1=text(-0.2,0.85,sprintf(' = %g ',D1(4)),'FontSize',10);
h1=text(0.6,0.85,sprintf(' = %g ',D1(5)),'FontSize',10);
h1=text(-0.2,0.8,sprintf(' = %g ',D1(1)),'FontSize',10);
h1=text(0.6,0.8,sprintf('J2 = %g ',D1(2)),'FontSize',10);
h1=text(0,0.75,'Внешних воздействий','FontSize',10);
h1=text(-0.2,0.7,sprintf('pst0 = %g ',D2(1)),'FontSize',10);
h1=text(0.6,0.7,sprintf(' tet0 = %g ',D2(2)),'FontSize',10);
h1=text(0.2,0.66,sprintf(' fit0 = %g ',D2(3)),'FontSize',10);
h1=text(-0.2,0.62,sprintf('psm = %g ',D2(4)),'FontSize',10);
h1=text(0.6,0.62,sprintf(' tem = %g ',D2(5)),'FontSize',10);
h1=text(0.2,0.58,sprintf(' fim = %g ',D2(6)),'FontSize',10);
h1=text(-0.2,0.54,sprintf('omps = %g ',D2(7)),'FontSize',10);
h1=text(0.6,0.54,sprintf(' omte = %g ',D2(8)),'FontSize',10);
h1=text(0.2,0.5,sprintf('omfi = %g ',D2(9)),'FontSize',10);
h1=text(-0. 2,0.46,sprintf('eps = %g ',D2(10)),'FontSize',10);
h1=text(0.6,0.46,sprintf(' ete = %g ',D2(11)),'FontSize',10);
h1=text(0.2,0.42,sprintf('efi=%g ',D2(12)),'FontSize',10);
h1=text(0,0.35,'Интегрирования','FontSize',10,'FontUnderline','on');
h1=text(0,0.3,sprintf('h = %g ',D5(1)),'FontSize',10);
h1=text(0,0.25,sprintf('hpr = %g ',D5(2)),'FontSize',10);
h1=text(0,0.2,sprintf('t = %g ',D5(3)),'FontSize',10);
h1=text(0,0.15,sprintf('tfinal = %g ',D5(4)),'FontSize',10);
h1=text(-0.3,0.12,'-----------------------------------------------------','FontSize',10);
tm=fix(clock); Tv=tm(4:6);
h1=text(-0.2,0.08,['Программа ' sprogram],'FontSize',10);
h1=text(-0.3,0.04,['Расчеты провел ' sname],'FontSize',10);
h1=text(-0.3,0,[sprintf(' %g :',Tv) ' ' date],'FontSize',10);
h1=text(-0.3,-0.04,'-----------------------------------------------------','FontSize',10);
h1=text(-0. 3,-0. 08,'Ukraine,   KPI,   cath. PSON','FontSize',10);
```

Виконання цієї програми приводить до появи у вікні фігури зображення, поданого на рис. 2.11.

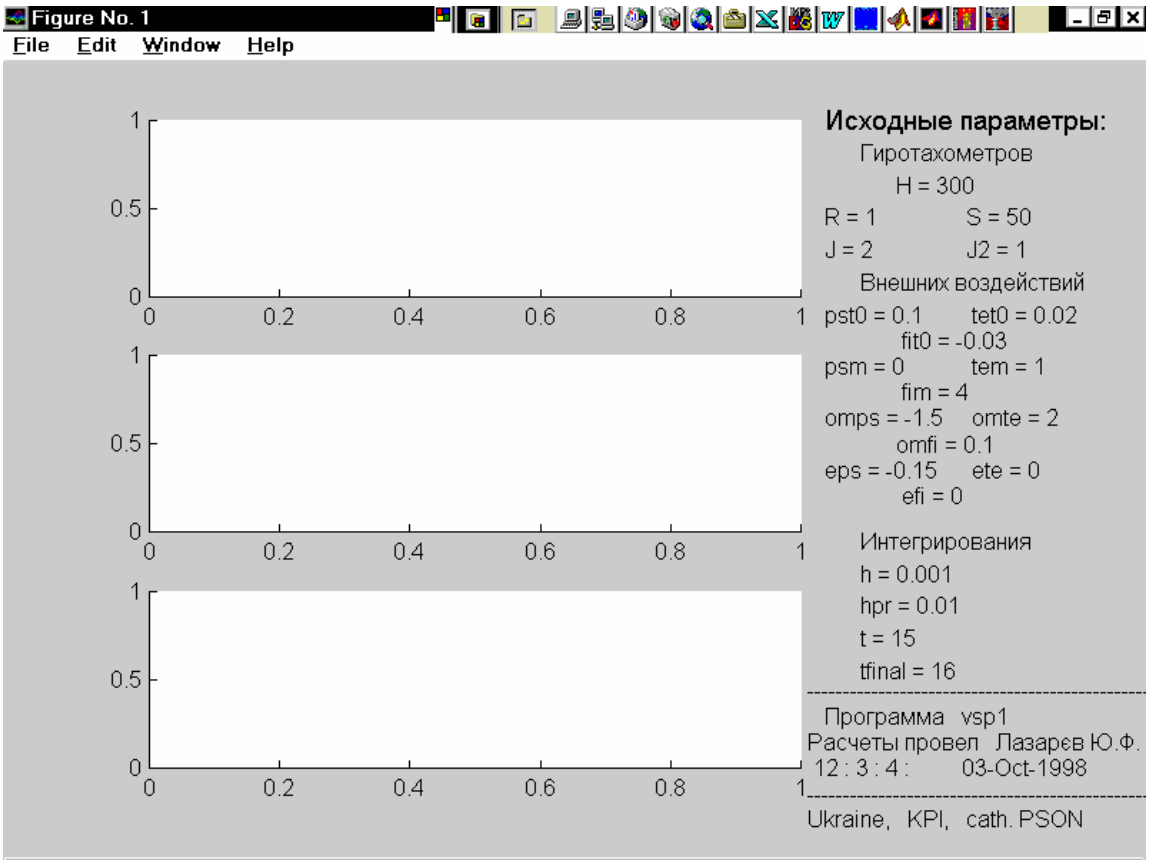


Рис. 2.10. Вид графічного вікна з оформленим текстовим підвікном1

2.5. Функції функцій

Деякі важливі універсальні процедури в MatLAB використовують як змінний параметр ім'я функції, із яким вони оперують, і тому потребують при зверненні до них указівки імені М-файлу, у якому записаний текст програми обчислення деякої іншої процедури (функції). Такі процедури називають ***функціями функцій***.

Щоб скористатися такою функцією від функції, необхідно, щоб користувач попередньо створив М-файл, у якому обчислювалося б значення потрібної функції за відомим значенням її аргументу.

2.5.1. Стандартні функції від функцій Matlab

Перелікуємо деякі зі стандартних функцій від функцій, передбачених у MatLAB.

Обчислення інтеграла методом квадратур здійснюється процедурою

[I, cnt] = ***quad***(‘<ім'я функції>’, a,b*)*.

Тут a і b - нижня й верхня межа змінювання аргументу функції; I - отримане значення інтеграла; cnt - кількість звернень до обчислення функції, поданої М-файлом із назвою, указаним у <ім'я функції>. Функція ***quad*** використовує квадратурні формули Ньютона-Котеса четвертого порядку.

Аналогічна процедура ***quad8*** використовує більш точні формули 8-го порядку.

Інтегрування звичайних диференціальних рівнянь здійснюють функції ***ode23*** і ***ode45***. Вони можуть застосовуватися як для розв'язування простих диференціальних рівнянь, так і для моделювання складних динамічних систем, тобто систем, поводження яких можна описати сукупністю звичайних диференціальних рівнянь.

Відомо, що будь-яка система звичайних диференціальних рівнянь (ЗДР) може бути подана як система рівнянь 1-го порядку у формі Коші:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y},t),$$

де $\boldsymbol{y}$ - вектор змінних стану (фазових змінних системи); t - аргумент (зазвичай - час); $\boldsymbol{f}$ - нелінійна вектор-функція від змінних стану $\boldsymbol{y}$ і аргументу t.

Звернення до процедур чисельного інтегрування ЗДР має вид:

[t, y] = ***ode23*** (‘<ім'я функції>’, tspan, y0, options)

[t, y] = ***ode45*** (‘<ім'я функції>’, tspan, y0, options),

де використані параметри мають такий зміст: <ім'я функції> - рядок символів, що є ім'ям М-файла, у якому обчислюється вектор-функція ***f***(y,t), тобто *праві частини системи ЗДР*; y0 - вектор початкових значень змінних стану; t - масив кінцевих значень аргументу, що відповідають крокам інтегрування; y - матриця проінтегрованих значень фазових змінних, в якій кожний стовпчик відповідає одній зі змінних стану, а рядок містить значення змінних стану, що відповіда-

ють певному кроку інтегрування; tspan - вектор-рядок [t0 tfinal], що містить два значення: t0 - початкове значення аргументу t; tfinal - кінцеве значення аргументу; ortions - рядок із параметрів, що визначають значення припустимої відносної й абсолютної похибки інтегрування.

Параметр ortions можна не вказувати. Тоді за замовчуванням припустима відносна похибка інтегрування приймається рівною $1 \cdot 10^{-3}$, абсолютна (по кожній із змінних стану) - $1 \cdot 10^{-6}$. Якщо ж по якихось параметрах ці значення не влаштовують користувача, треба перед зверненням до процедури чисельного інтегрування встановити нові значення припустимих похибок за допомогою процедури ***odeset*** у такий спосіб:

options = ***odeset*** ('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]).

Параметр RelTol визначає відносну похибку чисельного інтегрування по усіх фазових змінних одночасно, а AbsTol є вектором-рядком, що складається з абсолютних припустимих похибок чисельного інтегрування по кожній з фазових змінних.

Функція ***ode23*** здійснює інтегрування чисельним методом Рунге-Кутта 2-го порядку, а за допомогою методу 3-го порядку контролює відносні й абсолютні похибки інтегрування на кожному кроці і змінює величину кроку інтегрування так, щоб забезпечити задані межі похибок інтегрування.

Для функції ***ode45*** основним методом інтегрування є метод Рунге-Кутта 4-го порядку, а величина кроку контролюється методом 5-го порядку.

Обчислення мінімумів і коренів функції здійснюється такими функціями MatLAB:

fmin - відшукування мінімуму функції одного аргументу;

fmins - відшукування мінімуму функції кількох аргументів;

fzero - відшукування нулів (коренів) функції одного аргументу.

Звернення до першої з них у загальному випадку має такий вид:

Xmin = ***fmin*** (‘<ім'я функції>’, X1, X2).

Результатом цього звернення буде значення Xmin аргументу функції, яке відповідає локальному мінімуму в інтервалі X1<X<X2 функції, заданої М-файлом із зазначеним ім'ям.

Як приклад розглянемо відшукання значення числа π як значення локального мінімуму функції ***y = cos(x)*** на відрізку [3, 4]:

```
» Xmin = fmin('cos',3,4)
Xmin =  3. 1416e+000
```

Звернення до другої процедури повинно мати форму:

Xmin = ***fmins*** (‘<ім'я функції>’, X0),

при цьому X є вектором аргументів, а X0 означає початкове (первинне) значення цього вектора, в околі якого відшукується найближчий локальний мінімум функції, заданої М-файлом із зазначеним ім'ям. Функція ***fmins*** знаходить вектор аргументів Xmin, який відповідає знайденому локальному мінімуму.

Звернення до функції ***fzero*** повинно мати вид:

z = ***fzero*** (‘<ім'я функції>’, x0, tol, trace).

Тут позначено: x0 - початкове значення аргументу, в околі якого відшукується дійсний корінь функції, значення якої обчислюються в М-файлі із заданим ім'ям; tol - задана відносна похибка обчислення кореня; trace - позначення необхідності виводити на екран проміжні результати; z - значення шуканого кореня.

Побудова графіків функції однієї змінної може бути здійснена за допомогою процедури ***fplot***. Відмінність її від процедури ***plot*** у тому, що для побудови графіка функції немає необхідності в попередньому обчисленні значення функції й аргументу. Звернення до неї має вид:

fplot (‘<ім'я функції>’, [<інтервал>], n),

де <інтервал> - це вектор-рядок із двох чисел, що задають, відповідно, нижню й верхню межі змінювання аргументу; <ім'я функції> - ім'я М-файла з текстом процедури обчислення значення бажаної функції за заданим значенням її аргументу; n - бажане число частин розбиття зазначеного інтервалу. Якщо останню величину не задати, за замовчуванням інтервал розбивається на 25 частин. Хоча кількість частин "n", на які розбито інтервал змінювання аргументу, визначено, проте насправді кількість значень вектора "x" може бути значно більшою за рахунок того, що функція ***fplot*** проводить обчислення з додатковим обмеженням, щоб збільшення кута нахилу графіка функції на кожному кроці не перевищувало 10 градусів. Якщо ж воно виявилося більшим, здійснюється роздрібнення кроку змінювання аргументу, але не більше ніж у 20 разів. Останні два числа (10 і 20) можуть бути змінені користувачем, для цього при зверненні слід додати ці нові значення в заголовок процедури в зазначеному порядку.

Якщо звернутися до цієї процедури так:

[x, Y] = ***fplot*** (‘<ім'я функції>’, [<інтервал>], n),

то графік зазначеної функції не відображується на екрані (у графічному вікні). Замість цього обчислюється вектор "x" аргументів і вектор (або матриця) Y відповідних значень зазначеної функції. Щоб при зверненні останнього виду побудувати графік, необхідно зробити це у подальшому за допомогою процедури ***plot***(x, Y)***.***

2.5.2. Завдання

Завдання 2.4. Створіть М-файл, що обчислює функцію із *завдання 1.5.* обудуйте графік цієї функції за допомогою процедури ***fplot*** у межах, визначених у завданні 1.5. Обчисліть інтеграл від функції у тих же межах, використовуючи процедури ***quad*** і ***quad8***. Знайдіть точку локального мінімуму і локальний мінімум функції і найближчий корінь (нуль).

Завдання 2.5. Знайдіть точку локального мінімуму і локальний мінімум функції двох змінних, прийнявши за початкову точку із заданими координатами (таблиця 2.2). Попередньо створіть відповідну файл-функцію.

Таблиця 2.2

Варіант	x_o	y_o	$f(x,y)$
1	0	1	$e^{x+y}+(x-y)^2-2x-2y$
2	0.7	-1. 2	$(x-y)^2-\cos(x-y-1)$
3	1.5	-0. 5	$e^{x+y}-2x-2y-\cos(x-y-1)$
4	0.5	1.5	$e^{x+y}+4x^2-3x-3y$
5	0	1	$4x^2+\ln(x+y)+\frac{1}{x+y}$
6	1.2	0.7	$2^{x+y}-2x-2y+2(x-y)^2$
7	0	-0. 9	$e^{x-y}+2x+2y+(x+y)^2$
8	0.8	1.3	$(x-y)^2-\cos(x+y-1)$
9	1.5	0.5	$e^{x-y}-2x+2y-\cos(x+y-1)$
10	0.5	-1. 5	$e^{x-y}-3x+3y+4x^2$
11	0	-1	$4x^2+\ln(x-y)+\frac{1}{x-y}$
12	1.2	-0. 8	$2^{x-y}-2x+2y+2(x+y)^2$

2.5.3. Запитання

1. Що розуміється під поняттям "функція функцій"?
2. Які найбільш вживані стандартні функції функцій є в MatLAB?

2.6. Створення функцій від функцій

Деякі алгоритми є загальними для усіх функцій певного типу. Тому їхня програмна реалізація є єдиною для усіх функцій цього типу, але потребує використання алгоритму обчислення конкретної функції. Останній алгоритм може бути зафіксований у виді певного файлу-функції. Щоб перший, більш загальний алгоритм був пристосований для будь-якої функції, потрібно, щоб ім'я цієї функції було деякої змінною, яка могла б набувати визначеного значення (текстового імені файл-функції) тільки при зверненні до основного алгоритму. Такі функції від функцій уже розглядалися в розділі 2.1. До них належать процедури:

- обчислень інтеграла від функції, що потребують вказівки імені М-файлу, що містить обчислення значення подінтегральної функції;
- чисельного інтегрування диференціальних рівнянь, використання яких потребує вказівки імені М-файлу, у якому обчислюються праві частини рівнянь у формі Коші;
- алгоритмів чисельного обчислення коренів нелінійних алгебричних рівнянь (нулів функцій), які потребують указівки файл-функції, нуль якого відшукується;
- алгоритмів пошуку мінімуму функції, яку, у свою чергу, треба задавати відповідним М-файлом і т.п.

На практиці досить часто виникає необхідність створювати власні процедури такого типу. MatLAB надає такі можливості.

2.6.1. Процедура feval

У MatLAB будь-яка функція (процедура), наприклад, за ім'ям FUN1, може бути виконана не тільки за допомогою звичайного звернення виду:

[y1,y2,... ,yk] = **FUN1**(x1,x2,... ,xn),

а і за допомогою спеціальної процедури ***feval*** у такий спосіб:

[y1,y2,... ,yk] = ***feval***(‘FUN1',x1,x2,... ,xn),

де ім'я функції FUN1 є вже однією із вхідних змінних - текстовою змінною - і тому поміщається в апострофи.

Перевагою виклику функції у другій формі є те, що таке звернення не змінює своєї форми при змінюванні йменння функції, наприклад, на FUN2. Це дозволяє уніфікувати звернення до усіх функцій певного типу, тобто таких, що мають однакову кількість вхідних і вихідних параметрів визначеного типу. При цьому ім'я функції (а, виходить, і сама функція, що використовується) може бути довільним і змінюватися при повторних зверненнях.

Через те, що при виклику функції за допомогою процедури ***feval*** ім'я функції розглядається як один із вхідних параметрів процедури, останнє (ім'я функції) можна використовувати як змінну й оформляти в М-файлі звернення до неї, не знаючи ще конкретного імені функції.

2.6.2. Приклади створення процедур від функцій

Процедура методу чисельного інтегрування Рунге-Кутта 4-го порядку

Нехай задано систему звичайних диференціальних рівнянь (ЗДР) у формі Коші:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{Z}(\boldsymbol{y},t),$$

де $\boldsymbol{y}$ - вектор змінних стану системи; t - аргумент (час); $\boldsymbol{Z}$ - вектор заданих нелінійних функцій, який, власне, і визначає конкретну систему ЗДР.

Якщо значення вектора змінних $\boldsymbol{y}$ стану в момент часу t є відомим, то загальна формула, за якою може бути знайдений вектор ***yout*** значень змінних стану системи в момент часу $tout = t + h$ (h - крок інтегрування), має вид:

$\boldsymbol{yout} = \boldsymbol{y} + h*\boldsymbol{F}(\boldsymbol{y},t)$.

Функція $\boldsymbol{F}(\boldsymbol{y},t)$ пов'язана з вектором $\boldsymbol{Z}$ і може набувати різного виду в залежності від обраного методу чисельного інтегрування. Для методу Рунге-Кутта 4-го порядку виберемо таку її форму:

$$\mathbf{F} = (\mathbf{k_1} + 3\cdot\mathbf{k_2} + 3\cdot\mathbf{k_3} + \mathbf{k_4})/8,$$

де $\mathbf{k_1} = \mathbf{Z}(\mathbf{y}, t);$

$\mathbf{k}_2 = \mathbf{Z}(\mathbf{y} + h\cdot\mathbf{k_1}/3, \quad t + h/3);$

$\mathbf{k}_3 = \mathbf{Z}(\mathbf{y} + h\cdot\mathbf{k}_2 - h\cdot\mathbf{k_1}/3, \quad t + 2h/3);$

$\mathbf{k}_4 = \mathbf{Z}(\mathbf{y} + h\cdot\mathbf{k}_3 - h\cdot\mathbf{k}_2 - h\cdot\mathbf{k_1}, \quad t + h).$

Створимо М-файл процедури, яка здійснює ці обчислення, назвавши його "rko43":

```
function  [tout,yout] = rko43(Zpfun,h,t,y)
 %RKO43  Інтегрування ЗДР методом Рунге-Кутта 4-го порядку,
 %       праві частини яких задані процедурою Zpfun.
 %   Вхідні змінні:
 %   Zpfun - рядок символів, що містить ім'я процедури
 %               обчислення правих частин ЗДР
 %    Звернення: z = fun(t,y), де Zpfun = 'fun'
 %                          де t - поточний момент часу
 %                               y    - вектор поточних значень змінних стану
 %                               z    - обчислені значення похідних z(i) = dy(i)/dt.
 %   h    - крок інтегрування
 %   t    - попередній момент часу
 %   y    - попереднє значення вектора змінних стану.
 %   Вихідні змінні:
 %   tout - новий момент часу
 %   yout -  обчислене значення вектора  y через крок інтегрування

 % Ю.Ф.Лазарєв, каф. ПСОН, К П І, Україна
  %       Розрахунок проміжних значень похідних
   k1 = feval(Zpfun, t, y);
   k2 = feval(Zpfun, t+h/3, y+h/3*k1);
   k3 = feval(Zpfun, t+2*h/3, y+h*k2-h/3*k1);
   k4 = feval(Zpfun, t+h, y+h*(k3+k1-k2));
 %       Розрахунок нових значень вектора змінних стану
     tout = t + h;
     yout = y + h*(k1 + 3*k2 + 3*k3 + k4)/8;
 %       Кінець процедури RKO43
```

Зверніть увагу на такі обставини:

- звернення до процедури обчислення правих частин не є конкретизованим; ім'я цієї процедури належить до вхідних змінних процедури інтегрування і має бути конкретизовано лише при зверненні до останньої;

- проміжні змінні k є векторами-рядками (те ж стосується і змінних 'y', а також змінних 'z', які обчислюються в процедурі правих частин).

Процедура правих частин ЗДР маятника

Розглянемо процес створення процедури обчислення правих частин ЗДР на прикладі рівняння маятника, точка підвісу якого поступально переміщується з часом за гармонічним законом:

$$J \cdot \ddot{\varphi} + R \cdot \dot{\varphi} + mgl \cdot (1 + n_{my} \cdot \sin(\omega \cdot t + \varepsilon_y)) \cdot \sin\varphi =$$

$$= -mgl \cdot n_{mx} \cdot \sin(\omega \cdot t + \varepsilon_x) \cdot \cos\varphi,$$

де: J - момент інерції маятника; R - коефіцієнт демпфірування; mgl - опорний маятниковий момент маятника; nmy - амплітуда віброперевантаження точки підвісу маятника у вертикальному напрямку; nmx - амплітуда віброперевантаження в горизонтальному напрямку; φ - кут відхилення маятника від вертикалі; ω - частота коливань точки підвісу; ε_x , ε_y - початкові фази коливань (по прискореннях) точки підвісу в горизонтальному й вертикальному напрямках.

Щоб скласти М-файл процедури обчислення правих частин заданої системи ЗДР, насамперед треба привести первинну систему ЗДР до форми Коші. Для цього введемо позначення:

$y1 = \varphi$; $\quad y2 = \dot{\varphi}$.

Тоді рівняння маятника можна подати у виді сукупності двох диференціальних рівнянь 1-го порядку:

$$\frac{dy1}{dt} = y2;$$

$$\frac{dy2}{dt} = \{-mgl \cdot n_{mx} \cdot \sin(\omega \cdot t + \varepsilon_x) \cdot \cos(y1) - R \cdot y2 -$$

$$- mgl \cdot [1 + n_{my} \cdot \sin(\omega \cdot t + \varepsilon_y)] \cdot \sin(y1)\} / J$$

Порівнюючи отриману систему з загальною формою рівнянь Коші, можна зробити висновок, що

$$z1 = y2;$$

$$z2 = \{-mgl \cdot n_{mx} \cdot \sin(\omega \cdot t + \varepsilon_x) \cdot \cos(y1) - R \cdot y2 -$$

$$- mgl \cdot [1 + n_{my} \cdot \sin(\omega \cdot t + \varepsilon_y)] \cdot \sin(y1)\} / J$$

Саме обчислення цих двох функцій і повинно відбуватися в процедурі правих частин. Назвемо майбутню процедуру fm0. Вихідною змінною у ній буде вектор z = [z1 z2], а вхідними - момент часу t і вектор y = [y1 y2].

Деякою складністю є те, що постійні коефіцієнти в правих частинах не можна передати в процедуру через її заголовок. Тому об'єднаємо їх у вектор

коефіцієнтів k = [J,R,mgl, nmy,nmx, om,ey,ex] і віднесемо цей вектор до категорії глобальних

```
global K.
```

Тоді М-файл буде мати вид:

```
function z = FM0(t,y);
 % Процедура правих частин рівняння Фізичного Маятника.
 % Здійснює розрахунок вектора “z” похідних
 % від вектора "y" змінних стану по формулах:
 %          z(1)=y(2);
 %          z(2)=(-mgl*nmx*sin(om*t+ex)*cos(y(1))-R*y(2)-...
 %              mgl*(1+nmy*sin(om*t+ey))*sin(y(1)))/J,
 % Коефіцієнти передаються в процедуру через глобальний вектор
 %              K=[J,R,mgl,nmy,nmx,om,ey,ex]
 % Ю.Ф.Лазарєв 5-10-1998р.
global K
z(1) = y(2);
z(2) = (-K(3)*K(5)*sin(K(6)*t+K(8))*cos(y(1)) - K(2)*y(2) -...
  K(3)*(1+K(4)*sin(K(6)*t+K(7)))*sin(y(1)))/K(1);
 % Кінець процедури FM0
```

При використанні цієї процедури варто пам'ятати, що в тексті програми попередньо має бути оголошений глобальний вектор **K** за допомогою службового слова ***global***, а потім визначені усе 8 його елементів.

Цю процедуру можна дещо ускладнити, групуючи разом обчислення всіх зовнішніх моментів сил, крім моменту сил тяжіння, і оформлюючи їх як окрему процедуру. Для цього спочатку перетворимо початкове рівняння, записуючи його у виді:

$$\varphi'' + \sin\varphi = S(\tau, \varphi, \varphi'),$$

де штрих - позначення похідної за безрозмірним часом $\tau = \omega_0 \cdot t$,

$$\omega_0 = \sqrt{\frac{mgl}{J}},$$

а як $S(\tau, \varphi, \varphi')$ позначено деяку задану функцію безрозмірного часу, кута повороту маятника і його безрозмірної швидкості

$$\varphi' = \frac{d\varphi}{d\tau}.$$

В аналізованому випадку ця функція набуває такого виду:

$$S(\tau, \varphi, \varphi') = -2 \cdot \zeta \cdot \varphi' -$$

$$-[n_{mx} \cdot \sin(\nu \cdot \tau + \varepsilon_x) \cdot \cos\varphi + n_{my} \cdot \sin(\nu \cdot \tau + \varepsilon_y) \cdot \sin\varphi],$$

причому безрозмірні величини ζ і ν визначаються виразами:

$$\zeta = \frac{R}{2 \cdot \sqrt{mgl \cdot J}}; \qquad \nu = \frac{\omega}{\omega_0}.$$

Така безрозмірна форма подання рівнянь є кращою і зручною, тому що дозволяє скоротити кількість параметрів (у нашому випадку замість трьох розмірних параметрів J, R і mgl залишився один - ζ), а також подавати розв'язки рівняння в більш загальній формі.

Винесення обчислення моментів зовнішніх дій в окрему обчислювальну процедуру дозволяє також зробити процедуру правих частин рівняння маятника більш загальною, якщо звернення до процедури обчислення моментів здійснювати теж через функцію ***feval***.

Створимо процедуру ***MomFm1***, яка буде обчислювати моменти сил, що діють на маятник:

```
function m = MomFM1(t,y);
 % Обчислення Моментів сил, що діють на Фізичний Маятник.
 % Здійснює розрахунок моменту "m" сил
 % за формулою:
 %            m =-2*dz*y(2) - (nmx*sin(nu*t+ex)*cos(y(1)) +...
 %                  + nmy*sin(nu*t+ey)*sin(y(1)),
 % Коефіцієнти передаються в процедуру через глобальний вектор
 %            KM1=[dz,nmy,nmx,nu,ey,ex]
 % Ю.Ф.Лазарєв 5-10-1998р.
global KM1
m= -2*KM1(1)*y(2)- (KM1(3)*sin(KM1(4)*t+KM1(6))*cos(y(1)) +...
  KM1(2)*sin(KM1(4)*t+KM1(5))*sin(y(1)));
 % Кінець процедури MomFM1
```

Тепер слід перебудувати процедуру правих частин. Назвемо цей варіант FM1:

```
function z = FM1(mpfun,t,y);
 % Процедура правих частин рівняння Фізичного Маятника.
 % Здійснює розрахунок вектора “z” похідних
 %    векторів "y" змінних стану по формулах:
 %               z(1)=y(2);
 %               z(2)= - sin(y(1)) +S(t,y),
 % де
  % вхідні параметри:
 %            mpfun - ім'я процедури S(t,y)
 %                        mpfun = 'S';
 %               t - поточний момент часу;
 %                y - поточне значення вектора змінних стану;
 % вихідні параметри:
 %               z - вектор значень похідних від змінних стана.

 % Ю.Ф.Лазарєв 5-10-1998р.
z(1) = y(2);
z(2) = - sin(y(1)) + feval(mpfun,t,y);
 % Кінець процедури FM1
```

Через те що вид звернення до процедури правих частин змінився (додано нову вхідну змінну - ім'я процедури обчислення моментів), необхідно також перебудувати і процедуру чисельного методу. Назвемо її RKO43m:

```
function  [tout,yout] = rko43m(Zpfun,Mpfun,h,t,y)
 %RKO43m Інтегрування ОДУ методом Рунге-Кутта 4-го порядку,
 % праві частини яких задані процедурами Zpfun і Mpfun.
 % Вхідні параметри:
 %          Zpfun - рядок символів, що містить ім'я процедури
 %                        обчислення правих частин ЗДР.
 %                    Звернення: z = fun(Mpfun,t,y),
 %                де Zpfun = 'fun',
```

```
%                   Mpfun - рядок з ім'ям процедури, до якого
%                           звертається процедура fun;
%                   t - поточний момент часу
%                       y - вектор поточних значень змінних стана
%                       z - обчислені значення похідних z(i) = dy(i)/dt.
%               h - крок інтегрування
%           t - попередній момент часу
%           y - попереднє значення вектори змінних стану.
% Вихідні параметри:
%           tout - новий момент часу
%           yout - нове значення вектора змінних стану
%                   через крок інтегрування
% Розрахунок проміжних значень похідних
  k1 = feval(Zpfun, Mpfun,t, y);
  k2 = feval(Zpfun, Mpfun, t+h/3, y+h/3*k1);
  k3 = feval(Zpfun, Mpf un, t+2*h/3, y+h*k2-h/3*k1);
  k4 = feval(Zpfun, Mpfun, t+h, y+h*(k3+k1-k2));
%            Розрахунок нових значень вектора змінних стану
    tout = t + h;
    yout = y + h*(k1 + 3*k2 + 3*k3 + k4)/8;
%  Кінець процедури RKO43m
```

Така форма подання процедури обчислення правих частин диференціальних рівнянь є незручною, бо, по-перше, процедуру виду FM1 не можна використовувати при інтегруванні процедурами MatLAB ***ode23*** і ***ode45*** (останні потребують, щоб у процедурі правих частин було тільки два вхідні параметри, а в процедурі FM1 їх три), а, по-друге, така форма вимагає необхідність створення нових М-файлів методів чисельного інтегрування.

Можна цього уникнути, переробляючи ім'я додаткової функції Mpfun на глобальну змінну. Тоді процедура правих частин може бути записана так:

```
function z = FM2(t,y);
 % Процедура правих частин рівняння Фізичного Маятника.
  % Здійснює розрахунок вектора “z”
  % похідних вектору "y" змінних стану по формулах:
 %                   z(1)=y(2);
 %                       z(2)= - sin(y(1)) +S(t,y),
 % Вхідні параметри:
 %      mpfun - ім'я процедури S(t,y) - передається як глобальна змінна
  %                       mpfun = 'S';
 %      t - поточний момент часу;
 %      y - поточне значення вектора змінних стану;
 % Вихідні параметри:
 %      z - вектор значень похідних від змінних стану.

 % Ю.Ф.Лазарєв 7-10-1998р.
%
global MPFUN
z(1) = y(2);
z(2) = - sin(y(1)) + feval(MPFUN,t,y);
 % Кінець процедури FM2
```

Тепер процедура FM2 має тільки два вхідні параметри, що передаються через заголовок, і може бути використана будь-якою процедурою чисельного методу інтегрування, у тому числі - процедурами ***ode23*** і ***ode45***. Необхідно лише пам'ятати, що в основній програмі змінний MPFUN треба присвоїти деяке символьне значення (ім'я функції, яка буде використана в процедурі правих

частин), і вона має бути оголошеною як глобальна. Наприклад, якщо буде використана раніше створена процедура MomFun1, у Script-файлі має бути присутнім рядки

```
global MPFUN
MPFUN = 'MomFm1';
```

2.6.3. Завдання

Завдання 2.1 - 2.13. Створіть М-файл методу чисельного інтегрування диференціальних рівнянь відповідно до формул, наведених у таблицях 2.3 та 2.4.

Таблиця 2.3. Методи Рунге-Кутта

$$y_{m+1} = y_m + h\,F(t_m; y_m)$$

N% вар	Формула методу	Допоміжні величини	Назва методу
1	F=k1	$k1=Z(t_m;y_m)$	Ейлера
2	F=(k1+k2)/2	$k1=Z(t_m;y_m)$; $k2=Z(t_m+h;y_m+hk1)$	модифікований Ейлера
3	$F=Z(t_m+h/2; y_m+hk1/2)$	$k1=Z(t_m;y_m)$	
4	F=(k1+4k2+k3)/6	$k1=Z(t_m;y_m)$; $k2=Z(t_m+h/2;y_m+hk1/2)$; $k3=Z(t_m+h;y_m+h(2k2-k1))$	Хойне
5	F=(k1+3k3)/4	$k1=Z(t_m;y_m)$; $k2=Z(t_m+h/3;y_m+hk1/3)$; $k3=Z(t_m+2h/3;y_m+2hk2/3)$	
6	F=(k1+2k2+2k3+ +k4)/6	$k1=Z(t_m;y_m)$; $k2=Z(t_m+h/2;y_m+hk1/2)$; $k3=Z(t_m+h/2;y_m+hk2/2)$; $k4=Z(t_m+h;y_m+hk3)$	Рунге-Кутта
7	F=(k1+3k2+3k3+ +k4)/8	$k1=Z(t_m;y_m)$; $k2=Z(t_m+h/3;y_m+hk1/3)$; $k3=Z(t_m+2h/3;y_m+h(k2-k1/3))$; $k4=Z(t_m+h;y_m+h(k1--k2+k3))$	

Таблиця 2.4. Багатокрокові методи

N% вар	Формула прогнозу	Формула корекції	Назва методу
8	$y_{m+1}=y_{m-1}+2h(t_m;y_m)$	$y_{m+1}=y_m+h[Z(t_{m+1};y^*_{m+1})+Z(t_m;y_m)]/2$	
9	$y_{m+1}=y_m+h[Z(t_m;y_m)-Z(t_{m-1};y_{m-1})]/2$	$y_{m+1}=y_m+h[Z(t_{m+1};y^*_{m+1})+Z(t_m;y_m)]/2$	
10	$y_{m+1}=y_m+h[23Z(t_m;y_m)-16Z(t_{m-1};y_{m-1})+5Z(t_{m-2};y_{m-2}]/12$	$y_{m+1}=y_m+h[5Z(t_{m+1};y^*_{m+1})+8Z(t_m;y_m)-Z(t_{m-1};y_{m-1})]/12$	
11	$y_{m+1}=y_m+h[55Z(t_m;y_m)-59Z(t_{m-1};y_{m-1})+37Z(t_{m-2};y_{m-2})-9Z(t_{m-3};y_{m-3})]/24$	$y_{m+1}=y_m+h[9Z(t_{m+1};y^*_{m+1})+19Z(t_m;y_m)-5Z(t_{m-1};y_{m-1})+Z(t_{m-2};y_{m-2})]/24$	Адамса-Башфорта
12	$y_{m+1}=y_{m-3}+4h[2Z(t_m;y_m)-Z(t_{m-1};y_{m-1})+2Z(t_{m-2};y_{m-2})]/3$	$y_{m+1}=y_{m-1}+h[Z(t_{m+1};y^*_{m+1})+4Z(t_m;y_m)+Z(t_{m-1};y_{m-1})]/3$	Мілна
13	$y_{m+1}=y_{m-3}+4h[2Z(t_m;y_m)-Z(t_{m-1};y_{m-1})+2Z(t_{m-2};y_{m-2})]/3$	$y_{m+1}=\{9y_m-y_{m-2}+3h[Z(t_{m+1};y^*_{m+1})+2Z(t_m;y_m)-Z(t_{m-1};y_{m-1})]\}/8$	Хеммінга

Завдання 2.14. Створіть М-файл процедури правих частин диференціальних рівнянь руху двоступеневого гіроскопічного компаса:

$$J_1\ddot{\beta}+[H-J_2(\omega_3\cos\varphi\cos\beta+u_N(t)\cos\beta-u_E(t)\sin\beta)]*$$

$$*(\omega_3\cos\varphi\sin\beta+u_E(t)\cos\beta+u_N(t)\sin\beta)=0,$$

де J_1, J_2 - моменти інерції гірокомпаса; H - його власний кінетичний момент; β - кут відхилення головної осі гірокомпаса від площини географічного меридіана місця ; φ - географічна широта місця об'єкта, на якому встановлений гірокомпас;

$$u_N(t)=u_{Nm}\sin(\omega t+\varepsilon_N),$$

$$u_E(t)=u_{Em}\sin(\omega t+\varepsilon_E)$$

- відповідно північна і східна складові кутової швидкості повороту основи, на якій встановлений гірокомпас.

Завдання 2.15. Створіть процедуру правих частин диференціальних рівнянь, що описують динаміку обсягів популяцій $x_1(t)$ хижаків і $x_2(t)$ жертв, і відомих як модель Вольтерра:

$$\dot{x}_1 = -a_{11}\, x_1 + a_{12}\, x_1\, x_2;$$

$$\dot{x}_2 = a_{22}\, x_1 - a_{21}\, x_1\, x_2.$$

Завдання 2.16. Створіть процедуру правих частин диференціального рівняння кутового руху торпеди в горизонтальній площині, яка керується нелінійним виконуючим елементом:

$$J\frac{d^2\psi}{dt^2} + R\frac{d\psi}{dt} + kF(\psi) = 0,$$

Процедура має передбачати можливість використання кількох істотно нелінійних законів керування, зокрема, релейного із зонами нечутливості і гістерезисом:

якщо $\dot{x} > 0$, то $F(x) = \begin{cases} -c & \text{при } x < -b_1, \\ 0 & \text{при } -b_1 < x < b_2, \\ +c & \text{при } x > b_2; \end{cases}$

якщо ж $\dot{x} < 0$, то $F(x) = \begin{cases} +c & \text{при } x > b_1, \\ 0 & \text{при } -b_2 < x < b_1, \\ -c & \text{при } x < b_2. \end{cases}$

Завдання 2.17. Створіть процедуру правих частин диференціального рівняння кутового руху штучного супутника Землі, керованого за логічним законом:

$$J\frac{d\omega}{dt} = k\,\Phi(\omega,\varphi);$$

$$\frac{d\varphi}{dt} = \omega,$$

де ω - кутова швидкість руху супутника; J - його момент інерції; $\Phi(\omega,\varphi)$ - задана логічна нелінійна функція, яку визначимо за допомогою такої таблиці:

Значення функції $\Phi(\omega,\varphi)$

Знак ω	Знак φ		
	-	0	+
-	+1	0	0
0	+1	0	-1
+	0	0	-1

Завдання 2.18. Створіть процедуру правих частин диференціальних рівнянь руху дзиґи зі сферичним підп'ятником, встановленим у конічній лунці:

$$\begin{cases} \dfrac{dK_\xi}{dt} = M_{mp\xi}, \\ \dfrac{dK_\eta}{dt} = mgl\sin\delta_1\cos\delta_2 + M_{mp\eta}, \\ \dfrac{dK_\varsigma}{dt} = mgl\sin\delta_2 + M_{mp\varsigma}, \\ J_e\dot{\delta}_1\cos\delta_1 = K_\eta - K_\xi\cos\delta_1\sin\delta_2 + K_\varsigma\sin\delta_1\sin\delta_2, \\ J_e\dot{\delta}_2 = K_\xi\sin\delta_1 + K_\varsigma\cos\delta_1, \end{cases}$$

де J_e - екваторіальний момент інерції дзиґи щодо точки опори; H - кінетичний момент дзиги; δ_1, δ_2 - кути відхилення осі дзиґи від вертикалі; mgl - опорний маятниковий момент дзиґи ; $M_{mp\xi}, M_{mp\eta}, M_{mp\varsigma}$ - складові моменту сил тертя підп'ятника дзиґи; $K_\xi, K_\eta, K_\varsigma$ - проекції кінетичного моменту дзиґи на нерухомі осі. Моменти сил тертя $M_{mp\xi}$, $M_{mp\eta}$, $M_{mp\varsigma}$ можна подати наступними залежностями:

$$M_{mp\xi} = -C\cos^2\alpha\cos\delta_1\cos\delta_2;$$

$$M_{mp\eta} = -C\{\sin\delta_2[1-(\sin^2\alpha)/2] + \cos\delta_2\sin\delta_1(\sin^2\alpha)/2\};$$

$$M_{mp\varsigma} = C\cos\delta_2\sin\delta_1[1-(\sin^2\alpha)/2],$$

де

$$C = k\frac{R}{l}mgl\frac{1}{B\cos\alpha}sign(H),$$

а

$$B = \sqrt{1-\cos^2\alpha\cos^2\delta_1\cos^2\delta_2 - \sin^2\alpha(\sin^2\delta_2 + \sin^2\delta_1\cos^2\delta_2)/2}\ .$$

Вище використані позначення: k - коефіцієнт тертя матеріалу підп'ятника й матеріалу опори; R - радіус сфери підп'ятника; α - кут між утворюючою конуса лунки і площиною горизонту.

Взаємозв'язок проекцій $K_\xi, K_\eta, K_\varsigma$ із власним кінетичним моментом H і іншими кінематичними величинами визначається співвідношеннями:

$$K_\xi = H\cos\delta_2\cos\delta_1 - J_e\dot{\delta}_1\sin\delta_2\cos\delta_2\cos\delta_1 + J_e\dot{\delta}_2\sin\delta_1;$$

$$K_\eta = H\sin\delta_2 + J_e\dot{\delta}_1\cos^2\delta_2;$$

$$K_\varsigma = -H\cos\delta_2\sin\delta_1 + J_e\dot{\delta}_1\sin\delta_2\cos\delta_2\sin\delta_1 + J_e\dot{\delta}_2\cos\delta_1.$$

Завдання 2.19. Створіть процедуру правих частин диференціальних рівнянь гіроскопа в кардановому підвісі (ГКП), установленого на нерухомій основі:

$$\begin{cases} (J_1 + J_2\cos^2\beta)\,\ddot{\alpha} - 2J_2\dot{\alpha}\dot{\beta}\sin\beta\cos\beta + H\dot{\beta}\cos\beta = \\ = -f_2\dot{\alpha} + N_0 + N_m\sin(\omega t + \varepsilon_N) - [R_0 + R_m\sin(\omega t + \varepsilon_R)]\sin\beta, \\ J_3\ddot{\beta} + J_2\dot{\alpha}\sin\beta\cos\beta - H\dot{\alpha}\cos\beta = -f_2\dot{\beta} + L_0 + \\ \qquad + L_m\sin(\omega t + \varepsilon_L), \\ \dfrac{dH}{dt} = R_0 + R_m\sin(\omega t + \varepsilon_R), \end{cases}$$

де $J_1 = J_{2X} + J_{1Z}$; $J_2 = J_{1X} + J_e - J_{1Z}$; $J_3 = J_{1Y} + J_e$; J_{2X} - момент інерції зовнішньої рамки карданового підвісу відносно зовнішньої осі підвісу; J_{1X}, J_{1Y}, J_{1Z} - моменти інерції внутрішньої рамки відносно зазначених осей; J_e - екваторіальний момент інерції ротора гіроскопа; α, β - кути повороту головної осі ГКП навколо зовнішньої і внутрішньої осей підвісу; H - власний кінетичний момент ГКП; f_1, f_2 - коефіцієнти в'язкого тертя вдовж внутрішньої і зовнішньої осей підвісу; N_0, L_0, R_0 - сталі складові моментів зовнішніх сил, спрямованих по зовнішній, внутрішній осях підвісу й головній осі гіроскопа відповідно; N_m, L_m, R_m - амплітуди гармонійних складових моментів сил, що діють по відповідних осях; ω - частота змінювання гармонічних складових моментів сил; $\varepsilon_N, \varepsilon_L, \varepsilon_R$ - початкові фази гармонічних складових моментів сил.

Завдання 2.20. Створіть процедуру правих частин диференціального рівняння гіроскопічного тахометра (ГТ), встановленого на обертовій основі:

$$J_1\ddot{x} + f\dot{x} + cx = -c[u_{Z1} - (J_1 / H)\dot{u}_{Y1} + (J_2 / H)u_{X1}u_{Z1}] + M_0 c / H,$$

де x - вихідний сигнал ГТ; H - власний кінетичний момент ГТ; J_1, J_2 - моменти інерції ГТ; c - кутова жорсткість пружного зв'язку ГТ з основою; f - коефіцієнт кутового демпфірування; u_{X1}, u_{Y1}, u_{Z1} - проекції кутової швидкості основи на осі, пов'язані з ГТ. Останні пов'язані із проекціями на осі, пов'язані з основою, співвідношеннями:

$$u_{X1} = u_X\cos\beta - u_Z\sin\beta;$$

$$u_{Z1} = u_Z\cos\beta + u_X\sin\beta;$$

$$u_{Y1} = u_Y,$$

де $\beta = -\dfrac{H}{c}x.$

Проекції кутової швидкості основи на осі, пов'язані з тією ж основою вважати такими, що змінюються з часом за законами:

$$u_X = u_{X0} + u_{Xm}\sin(\omega t + \varepsilon_X);$$

$$u_Y = u_{Y0} + u_{Ym}\sin(\omega t + \varepsilon_Y);$$

$$u_Z = u_{Z0} + u_{Zm}\sin(\omega t + \varepsilon_Z).$$

Завдання 2.21. Слідкуюча система складається із задавального елемента, який задає ϑ_1 кут, на який має повернутися вихідний вал слідкуючої системи (вал електродвигуна), формуючого елемента (сельсина), який порівнює цей кут

із кутом повороту ϑ_2 вихідного вала електричного двигуна і формує електричний сигнал, пропорційний синусу різниці цих двох кутів:

$u_1 = U_{1m} \sin(\vartheta_1 - \vartheta_2)$.

Цей сигнал підсумовується із сигналом тахогенератора на валі двигуна:

$u_2 = u_1 - u_k; \quad u_k = k_Д \ \omega_Д$.

Сигнал u_2 подається на підсилювальний устрій, що є трипозиційним реле з гістерезисом. Останнє формує напругу $u_Д$ у відповідності із залежністю:

$$u_Д = f(u_2) = \begin{cases} z_b \ sign(u_2) & при \quad |u_2| > x_b; \\ 0 & при \quad |u_2| < x_a. \end{cases}$$

Ця напруга подається на електродвигун для керування обертанням його вихідного вала. Обертальний рух вала двигуна описується диференціальними рівняннями:

$$\begin{cases} T_Д \dfrac{d\omega_Д}{dt} + \omega_Д = k_Д \ u_Д; \\ \dfrac{d\vartheta_2}{dt} = \omega_Д. \end{cases}$$

Створіть у формі М-файлу процедуру обчислення правих частин диференціальних рівнянь слідкуючої системи, вважаючи вихідними величинами кут $\vartheta = \vartheta_1 - \vartheta_2$ неузгодженості й швидкість його зміни.

Завдання 2.22. Скласти процедуру відшукання точних розв'язків системи лінійних однорідних диференціальних рівнянь по заданій матриці $\boldsymbol{A}$ системи ДР у формі Коші:

$$\frac{d\mathbf{y}}{dt} = \mathbf{A} \cdot \mathbf{y}$$

і заданому вектору **y0** початкових умов.

Завдання 2.23. Створіть процедуру правих частин диференціальних рівнянь прямування в просторі трьох гравітуючих матеріальних точок (задача трьох тіл у небесній механіці)

$$\begin{cases} \dfrac{d^2\mathbf{R_2}}{dt^2} = -\gamma \left(\dfrac{m_1}{R_{21}^3}\mathbf{R_{21}} - \dfrac{m_3}{R_{32}^3}\mathbf{R_{32}}\right), \\ \dfrac{d^2\mathbf{R_3}}{dt^2} = -\gamma \left(\dfrac{m_2}{R_{32}^3}\mathbf{R_{32}} - \dfrac{m_1}{R_{13}^3}\mathbf{R_{13}}\right), \\ \mathbf{R_1} = -\dfrac{1}{m_1}(m_2\mathbf{R_2} + m_3\mathbf{R_3}), \end{cases}$$

де m_1, m_2, m_3 - маси гравітуючих матеріальних тіл; $\mathbf{R_1}, \mathbf{R_2}, \mathbf{R_3}$ - радіуси-вектори матеріальних точок щодо їхнього загального центру мас; $\mathbf{R_{21}} = \mathbf{R_2} - \mathbf{R_1}$; $\mathbf{R_{32}} = \mathbf{R_3} - \mathbf{R_2}$; $\mathbf{R_{13}} = \mathbf{R_1} - \mathbf{R_3}$ - радіуси-вектори точок одна щодо одної; R_{21}, R_{32}, R_{13} - поточні відстані між точками; γ - гравітаційна стала.

Завдання 2.24. Скласти процедуру правих частин диференціального рівняння лампового генератора:

$\ddot{q} + \omega^2\, q = (\chi - \lambda\, q^2)\, \dot{q}$.

Це так називане рівняння Ван-дер-Поля.

Завдання 2.25. Скласти процедуру правих частин диференціального рівняння руху маятника на рухомій основі з урахуванням моменту сил сухого тертя уздовж осі маятника:

$$J\,\ddot{\varphi} + R\,\dot{\psi} + mgl\sin\varphi = M_{m\text{p}} + M_0 + M_m \sin(\omega t),$$

де $\psi = \varphi - \vartheta(t)$, $\vartheta(t)$ - поточний кут повороту основи навколо осі маятника; M_0 - постійна складова моменту зовнішніх сил, що діють на маятник; M_m - амплітуда гармонійної складової моменту зовнішніх сил; ω - частота змінювання моменту сил; $M_{m\text{p}}$ - момент сил сухого тертя по осі маятника.

Вважати $M_{m\text{p}} = -M\ sign(\dot{\psi})$, де

$$sign(x) = \begin{cases} +1, & \text{якщо} \quad x > 0, \\ 0, & \text{якщо} \quad x = 0, \\ -1, & \text{якщо} \quad x < 0, \end{cases}$$

а також

$\vartheta(t) = \vartheta_0 + \dot{\vartheta}_0\, t + \vartheta_m \sin(\omega t + \varepsilon)$.

Завдання 2.26. Скласти процедуру відшукування точних часткових розв'язків системи лінійних неоднорідних диференціальних рівнянь по заданій матриці $\boldsymbol{A}$ системи ДР у формі Коші і матриці $\boldsymbol{B}$ коефіцієнтів вхідних дій:

$$\frac{d\mathbf{y}}{dt} = \mathbf{A}\,\mathbf{y} + \mathbf{B}\,\mathbf{x},$$

де вектор $\boldsymbol{x}$ вхідних дій взяти у виді

$\mathbf{x} = \mathbf{B_o} + \mathbf{B_S} \sin(\omega t) + \mathbf{B_C} \cos(\omega t)$.

2.7. Приклад створення складної програми

Раніше були розглянуті основні перешкоди, що стоять на шляху створення складних програм, і засоби їхнього подолання. Тепер, з огляду на це, спробуємо скласти й випробувати в роботі одну з досить складних комплексних програм.

2.7.1. Програма моделювання руху маятника

Постановка задачі. Нехай потрібно створити програму, що дозволила б моделювати рух фізичного маятника з точкою підвісу, що вібрує, шляхом чисельного інтегрування диференціального рівняння цього руху.

Диференціальне рівняння руху маятника для цієї задачі можна прийняти таким:

$$J \cdot \ddot{\varphi} + R \cdot \dot{\varphi} + mgl \cdot (1 + n_{my} \cdot \sin(\omega \cdot t + \varepsilon_y)) \cdot \sin\varphi =$$

$$= -mgl \cdot n_{mx} \cdot \sin(\omega \cdot t + \varepsilon_x) \cdot \cos\varphi,$$

де J - момент інерції маятника; R - коефіцієнт демпфірування; mgl - опорний маятниковий момент маятника; nmy - амплітуда віброперевантаження точки підвісу маятника у вертикальному напрямку; nmx - амплітуда віброперевантаження в горизонтальному напрямку; φ - кут відхилення маятника від вертикалі; ω - частота коливань точки підвісу; ε_x , ε_y - початкові фази коливань (по прискореннях) точки підвісу в горизонтальному і вертикальному напрямках.

Потрібно створити таку програму, яка дозволяла б обчисляти закон змінювання кута відхилення маятника від вертикалі з часом при довільних, встановлюваних користувачем значеннях усіх вищезазначених параметрів маятника і поступального руху основи, а також за довільних початкових умов.

Обчислення будемо здійснювати шляхом чисельного інтегрування за допомогою стандартної процедури ***ode45***.

Перетворення рівняння. Для підготування диференціальних рівнянь до чисельного інтегрування насамперед необхідно *привести ці рівняння до нормальної форми Коші*. Бажано також подати їх у безрозмірній формі.

Для поданого рівняння це було зроблено в попередньому розділі.

Запис М-файлу процедури правих частин. Наступним кроком підготовки програми є складання і запис на диск тексту процедури обчислення правих частин отриманої системи ДР у формі Коші.

Ця процедура була створена в попередньому розділі й остаточно вона має вид:

Файл FM2 . m

```
function z = FM2(t,y);
 % Процедура правих частин рівняння Фізичного Маятника.
 % Здійснює розрахунок вектора "z" похідних вектора
 %       "y" змінних стану по формулах:
 %              z(1)=y(2);
```

```
%                z(2)=-sin(y(1)) +S(t,y),
% де
% Вхідні параметри:
 %                t - поточний момент часу;
 %           y - поточне значення вектора змінних стану;
%                MPFUN - ім'я процедури S(t,y) - глобальна змінна
%                                MPFUN = 'S';
% Вихідні параметри:
 %           z - вектор значень похідних від змінних стану
.
 % Ю.Ф.Лазарєв 7-10-1998р.
%
global MPFUN
z(1) = y(2);
z(2) = - sin(y(1))+ feval(MPFUN,t,y);
z =z';
 %  Кінець процедури FM2
```

Примітка. Зверніть увагу на незначну, але істотну відмінність наведеної процедури від аналогічної процедури попереднього розділу - наявність наприкінці процедури операції транспонування вектора похідних. Це обумовлено тим, що *процедура **ode45** потребує, щоб вектор похідних був обов'язково стовпцем*.

Як додаткову процедуру, що використовується в процедурі FM2, виберемо раніше створену процедуру MomFM1, яку запишемо у файл MomFM1.

Файл MomFM1 . m

```
function m = MomFM1(t,y);
 % Обчислення Моментів Сил, що діють на Фізичний Маятник.
 % Здійснює розрахунок моменту "m" сил
 % по формулі:
 % m =-2*dz*y(2) - (nmx*sin(nu*t+ex)*cos(y(1)) +...
 % + nmy*sin(nu*t+ey)*sin(y(1)),
 % Коефіцієнти передаються в процедуру через
 % глобальний вектор KM1=[dz,nmy,nmx,nu,ey,ex]
 % Ю.Ф.Лазарєв 5-10-1998р.
global KM1
m = -2*KM1(1)*y(2)- KM1(3)*sin(KM1(4)*t+KM1(6))*cos(y(1)) -...
  KM1(2)*sin(KM1(4)*t+KM1(5))*sin(y(1));
 % Кінець процедури MomFM1
```

Очевидно, що у викличному Script-файлі треба передбачити оголошення імені додаткового файлу MomFM1 як глобальної змінної MPFUN, а також забезпечити оголошення глобальної змінної по імені KM1 і завдання значень цього числового масиву з п'ятьох елементів.

Створення керуючого (головного) Script-файлу. Головний файл створимо відповідно до рекомендацій поділу. 2.4.5 :

Файл FizMayatn2 . m

```
 % FizMayatn2
 % Керуюча програма дослідження руху фізичного маятника,
 % встановленого на поступально вібруючій основі

 %  Лазарєв Ю.Ф., кафедра ПСОН, КПІ, Україна, 7-10-1998р.
```

```
%
FizMayatn2_Zastavka
k = menu(' Що робити ?   ',' Продовжити роботу ', ' Закінчити роботу ');
if k==1,
     while k==1
          FizMayatn2_Menu
          FizMayatn2_Yadro
          k =menu(' Що робити ?   ',' Продовжити роботу ', ' Закінчити роботу ');
     end
end
clear global
clear
 % Кінець FizMayatn2
```

Як бачимо, програма викликає три додаткових Script-файли - FizMayatn2_Zastavka, FizMayatn2_Menu і FizMayatn2_Yadro. Тому потрібно створити ще ці три М-файли.

Створення Script-файла заставки. Як зазначалося, цей файл повинний містити оператори виведення на екран інформації про основні особливості математичної моделі, реалізованої у програмі, і введення первинних значень параметрів цієї моделі. Нижче приведений текст М-файла FizMayatn2_Zastavka.

Файл FizMayatn2_Zastavka . m

```
 % FizMayatn2_Zastavka
 % Частина (виведення заставки на екран)
 % програми FizMayatn2
 % Ю.Ф.Лазарєв, кафедра ПСОН, КПІ, Україна, 24-09-1998р.
%
 % Введення "вшитих" значень
sprogram = 'FizMaytn2.m';
sname ='Лазарєв Ю.Ф.';
KM1 = [0 0 0 0 0 0];
MPFUN = 'MomFm1';
global KM1 MPFUN
tfinal =2*pi*5;
fi0 =pi/180; fit0 = 0;
clc
 disp( [' Це програма, що здійснює інтегрування рівняння ';...
  ' Фізичного Маятника при поступальній вібрації точки підвісу ';...
  ' у формі ';...
  ' fi" + sin(fi) = - 2*dz*fi" - ';...
  ' - nmy*sin(nu*t+ey)*sin(fi) -nmx*sin(nu*t+ex)*cos(fi)';...
  'де fi - кут відхилення маятника від вертикалі, ';...
  ' dz - відносний коефіцієнт загасання, ';...
  ' nu - відносна частота вібрації точки підвісу, ';...
  ' nmy,nmx - амплітуди віброперевантаження у вертикальному ';...
  ' і горизонтальному напрямках відповідно, ';...
  ' ey,ex - початкові фази коливань у вертикальному ';...
  ' і горизонтальному напрямках відповідно, ';...
  ' KM1 = [dz,nmy,nmx,nu,ey,ex] - матриця коефіцієнтів '])
 % Кінець FizMayatn2_Zastavka
```

У ньому здійснюється присвоювання первісних ("вшитіх") значень усім параметрам заданого диференційного рівняння, а також параметрам чисельного інтегрування - початковим умовам руху маятника й тривалості процесу інтегрування. Частина цих параметрів об'єднується в єдиний глобальний вектор

KM1. Одночасно змінній MPFUN, що буде використовуватися при інтегруванні, присвоюється значення 'MomFm1'.

Створення файлу меню. Зміст файлу меню FizMayatn2_Menu наведено нижче.

Файл FizMayatn2_Menu . m

```
% FizMayatn2_Menu
% Частина (здійснююча діалогове змінювання даних)
% програми FizMayatn2
% Ю.Ф.Лазарєв, кафедра ПСОН, КПІ, Україна, 24-09-1998р.
k=1;
while k<10
  disp(' ')
  disp(' Зараз установлено ')
  disp([sprintf(' Початковий ріг (градуси) = %g', fi0*180/pi),...
  sprintf(' Початкова швидкість = %g', fit0)])
  disp(sprintf(' Число періодів = %g', tfinal/2/pi))
  KM1
 % KM1=[dz,nmy,nmx,nu,ey,ex]
  k = menu( ' Що змінювати ?   ', ...
  sprintf(' Відносний к-нт згасання = %g', KM1(1)),...
  sprintf(' Перевантаж.(вертикаль) = %g', KM1(2)),...
  sprintf(' Перевантаж.(горизонталь) = %g', KM1(3)),...
  sprintf(' Відносну частоту = %g', KM1(4)),...
  sprintf(' Фазу (вертикаль) = %g', KM1(5)),...
  sprintf(' Фазу (горизонталь) = %g', KM1(6)),...
  sprintf(' Початковий кут (градуси) = %g', fi0*180/pi),...
  sprintf(' Початкову швидкість = %g', fit0),...
  sprintf(' Кількість періодів = %g', tfinal/2/pi),...
   '  Нічого не змінювати  ');
  disp(' ')
  if k<7,KM1(k) = input( ['Зараз KM1(',num2str(k),sprintf(') =  %g', KM1(k)),...
              '   Введіть нове значення  = ']);
  elseif k==7, fi0 = input([sprintf('Зараз fi0 =  %g градусів', fi0*180/pi),...
              '   Введіть нове значення  = ']);
      fi0 = fi0*pi/180;
          elseif k==8, fit0 = input([sprintf('Зараз fit0 =  %g', fit0),...
              '   Введіть нове значення  = ']);
  elseif k==9,tfinal=input([sprintf('Зараз кількість періодів = %g', tfinal/2/pi),...
              ' Введіть нове значення = ']);
      tfinal = tfinal*2*pi;
  end
end % FizMayatn2_Menu
```

Файл здійснює організацію діалогового введення-змінювання значень параметрів фізичного маятника, руху основи й параметрів чисельного інтегрування у відповідності зі схемою, описаною в поділі 2.4.4.

Створення файлу ядра програми. Основні дії по організації процесу чисельного інтегрування й виведенню графіків зосереджені у файлі по імені FizMayatn2_Yadro:

Файл FizMayatn2_Yadro . m

```
% FizMayatn2_Yadro
% Частина (здійснююча основні обчислення)
 % програми FizMayatn2
```

```
 % Ю.Ф.Лазарєв, кафедра ПСОН, КПІ, Україна, 7-10-1998р.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1. Підготовка початкових умов
%---------------------------------
t = 0; tf = tfinal;       y0 =[fi0 fit0];
options = odeset('RelTol',1e-8,'AbsTol',[1e-10 1e-10]);
%---------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2.  Організація циклу інтегрування
%---------------------------------
 [t,y] = ode45('FM2',[0 tf],y0,options);
%---------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3. Виведення графіків
subplot(2,1,2);
plot(t/2/pi,y(:,1)*180/pi);grid;
title('Відхилення від вертикалі','FontSize',14);
xlabel('Час (у періодах малих власних коливань)','FontSize',12);
ylabel('Кут у градусах','FontSize',12);
subplot(2,4,1:2);
plot(y(:,1)*180/pi,y(:,2));grid;
title('Фазова траєкторія','FontSize',14);
xlabel('Кут у градусах','FontSize',12);
ylabel('Швидкість','FontSize',12);
%_____________________
% Виведення текстової інформації у графічне вікно
subplot(2,4,3:4);     axis('off');
h1=text(0,1.1,'Моделювання руху фізичного маятника', 'FontSize', 14, 'FontWeight',
'Bold');
h1=text(0.4, 1,'за рівнянням','FontSize',12);
h1=text(0,0.9,'fi'' + 2*dz*fi'' + [1+nmy*sin(nu*t+ey)]*sin(fi) =','FontSize',14);
h1=text(0.55,0.8,' = - nmx*sin(nu*t+ex)*cos(fi)','FontSize',14);
h1=text(0,0.7,'за таких значень параметрів:','FontSize',12);
h1=text(0.45,0.6,sprintf('dz = %g',KM1(1)),'FontSize',12);
h1=text(0,0.5,sprintf('nmy = %g',KM1(2)),'FontSize',12);
h1=text(0.7,0.5,sprintf('nmx = %g',KM1(3)),'FontSize',12);
h1=text(0,0.4,sprintf('ey = %g',KM1(5)),'FontSize',12);
h1=text(0.7,0.4,sprintf('ex = %g',KM1(6)),'FontSize',12);
h1=text(0.45,0.3,sprintf('nu = %g',KM1(4)),'FontSize',12);
h1=text(0,0.2,'і початкових розумів:','FontSize',12);
h1=text(0,0.1,[sprintf('fi(0) = %g',fi0*180/pi),' градусів'],'FontSize',12);
h1=text(0.7,0.1,sprintf('fi''(0) = %g',fit0),'FontSize',12);
h1=text(0,0.05,);----------------------------------------------------------------------------');
h1=text(0,-0.2,);----------------------------------------------------------------------------');
h1=text(-0.05,-0.05,['Програма ',sprogram]);
h1=text(0.55,-0.05,'Автор - Лазарєв Ю.Ф., каф. ПСОН');
h1=text(0,-0.15,['Виконав ',sname]);
tm=fix(clock);    Tv=tm(4:5);
h1=text(0.65,-0. 15,[sprintf(' %g:',Tv),'     ',date]);
 % Кінець файла FizMayatn2_Yadro
```

Як бачимо, основні операції включають три головних поділи - уведення початкових умов, організації циклу інтегрування й організації оформлення графічного вікна виведення.

Налагоджування програми. Налагодження програми складається із запуску головного М-файлу FizMayatn2, перевірки правильності функціювання всіх частин програми, внесення коректив у тексти використовуваних М-файлів доти, поки всі запрограмовані дії не будуть задовольняти задані вимоги.

Крім того, сюди відносяться і дії по перевірці "адекватності", тобто відповідності одержуваних програмою результатів окремим апріорно відомим випадкам поводження досліджуваної системи. Очевидно, для такої перевірки потрібно підібрати декілька сукупностей значень параметрів системи, при яких поводження системи є цілком відомим із попередніх теоретичних або експериментальних досліджень. Якщо отримані програмою результати цілком узгодяться з відомими, програма є адекватною прийнятій математичній моделі.

У приведеному тексті програми "вшиті" початкові значення параметрів відповідають вільному рухові маятника без впливу тертя. За таких умов рух маятника є незатухаючими коливаннями відносно положення вертикалі. Тому, якщо програма працює вірно, на графіках повинні спостерігатися саме такі коливання маятника. Результат роботи створеної програми при цих умовах поданий на рис. 2.12. Як очевидно, у цьому відношенні програма є адекватною прийнятій математичній моделі.

Проведення досліджень. Створена програма тепер може бути використана для моделювання й дослідження різноманітних нелінійних ефектів, які спостерігаються у фізичного маятника при поступальній вібрації точки його підвісу. На рис. 2.13 - 2.18 продемонстровані деякі можливості створеної програми.

На рис. 2.13 приведені параметричні коливання маятника, що можуть виникати при вібрації точки підвісу у вертикальному напрямку. З рисунка очевидно, що в цьому випадку коливання маятника щодо вертикалі спочатку збільшуються по амплітуді, а потім усталюються, причому частота сталих коливань удвічі менше за частоту вібрації основи і складає приблизно 1,15.

Випрямний ефект маятника проілюстрований на рис. 2.14. З нього наочно видно, що одночасна вібрація основи у вертикальному й горизонтальному напрямках призводить до відхилення середнього положення маятника від вертикалі на кут приблизно -5 градусів.

Рис. 2.15 ілюструє стаціонарні коливання маятника щодо верхнього положення рівноваги, які можуть спостерігатися при інтенсивній вертикальній вібрації. Ці коливання при наявності тертя загасають, як показано на рис. 2.16, і маятник "застигає" у верхньому положенні.

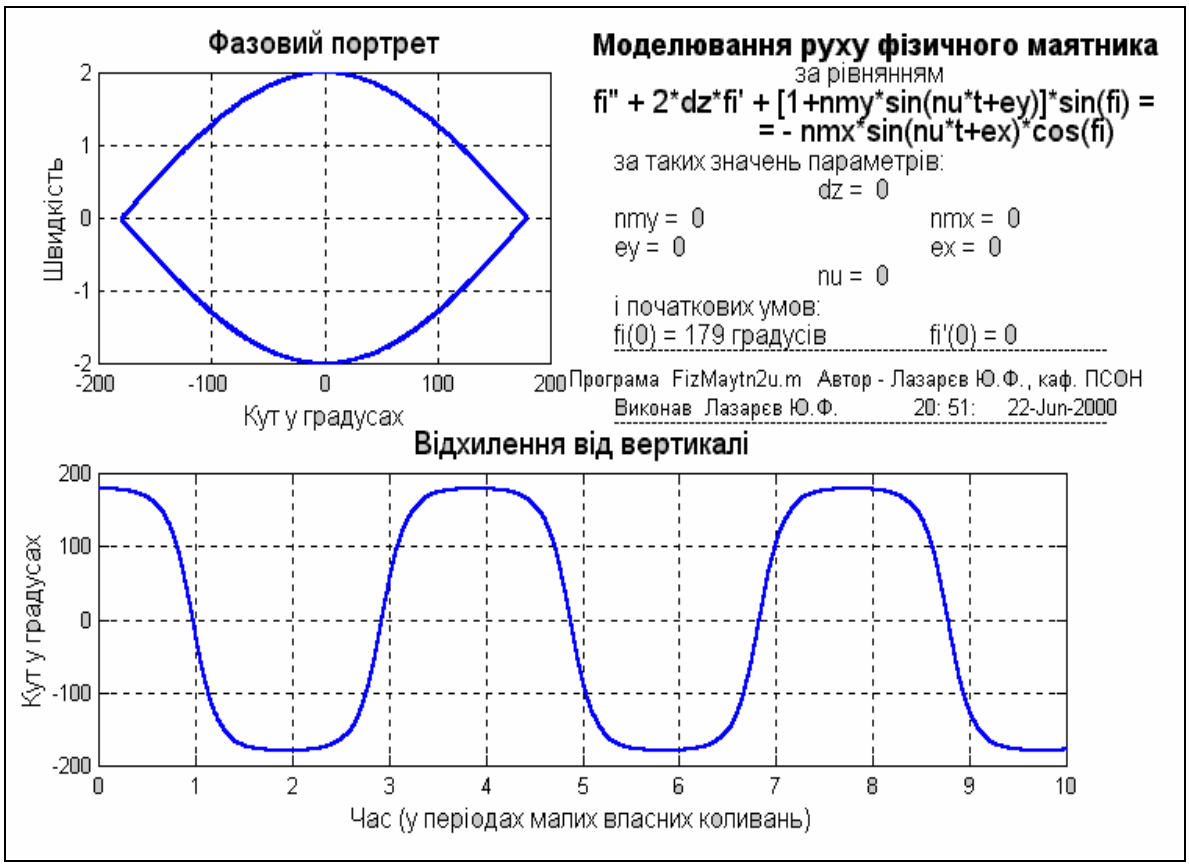


Рис. 2.12. Власні незгасаючи коливання з великою амплітудою

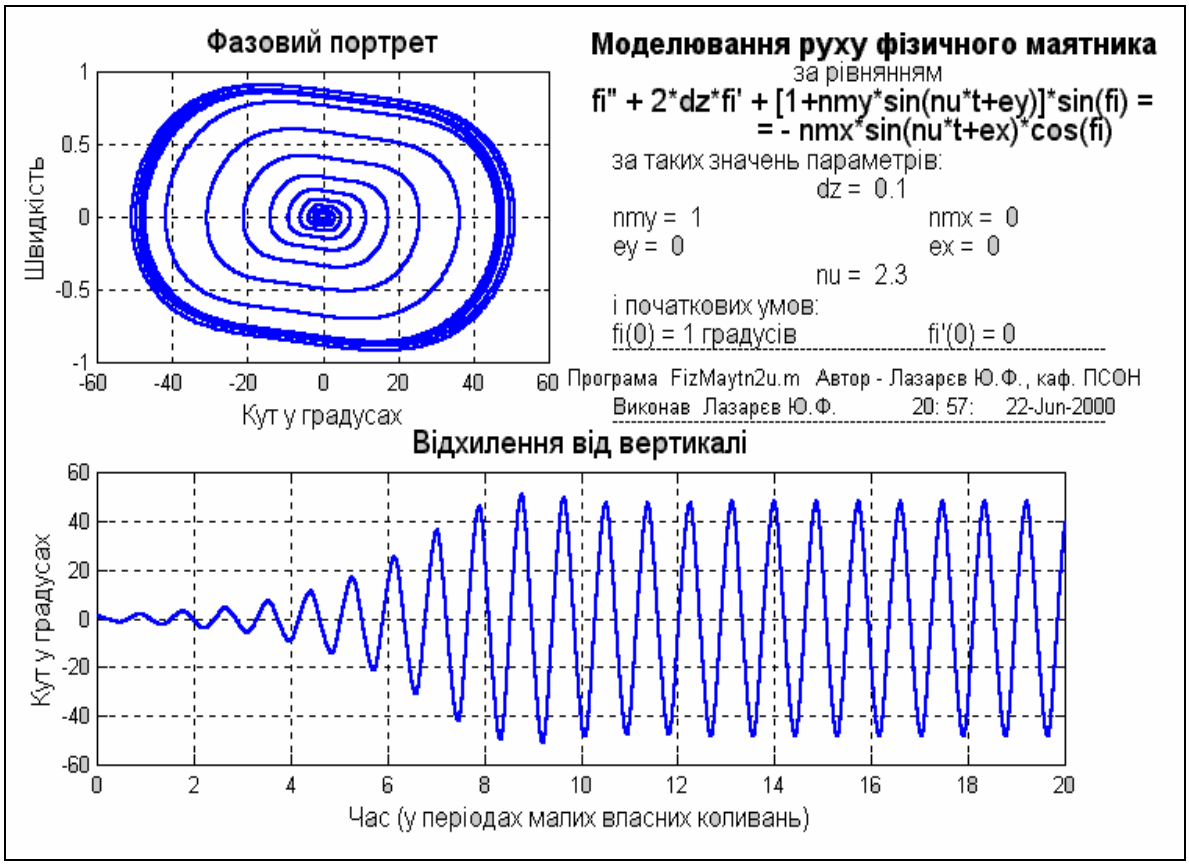


Рис. 2.13. Збудження параметричних коливань

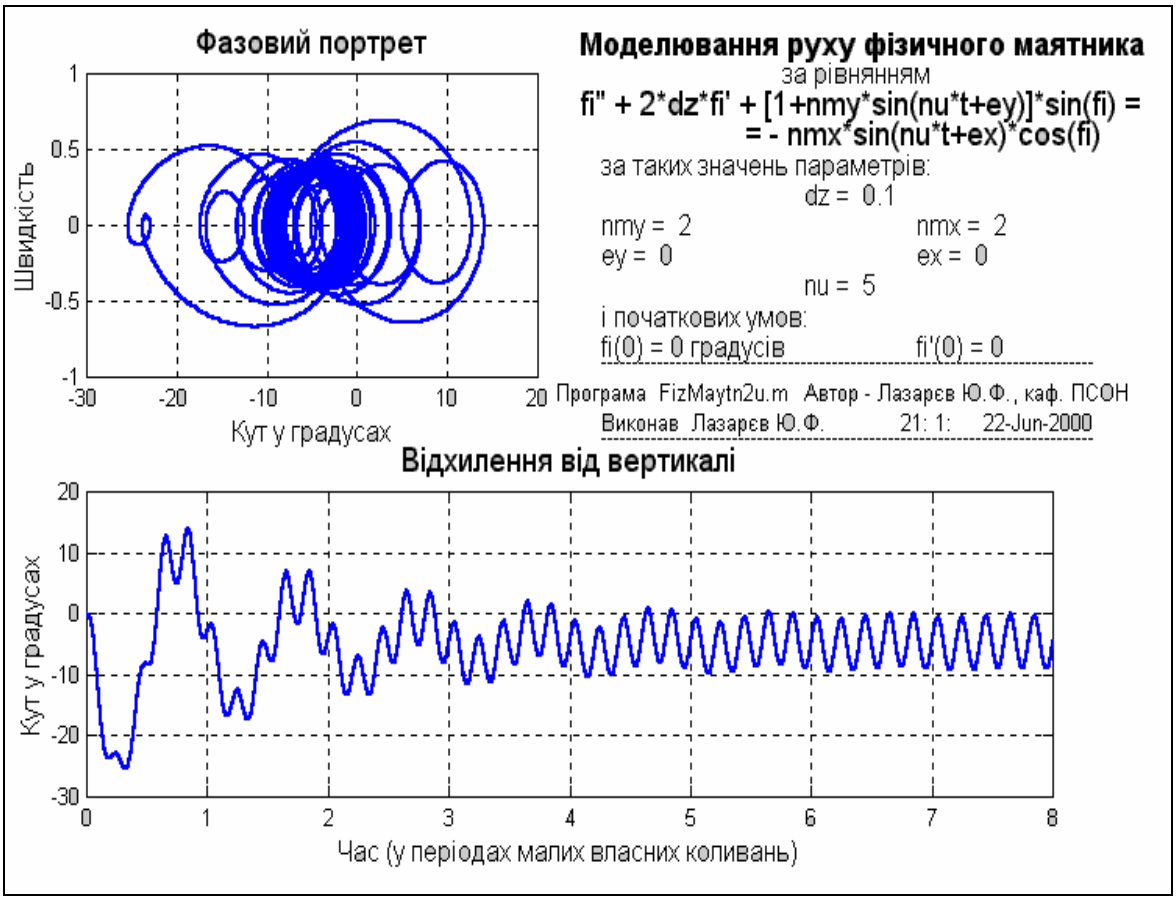


Рис. 2.14. Випрямний ефект при навскісній вібрації точки підвісу

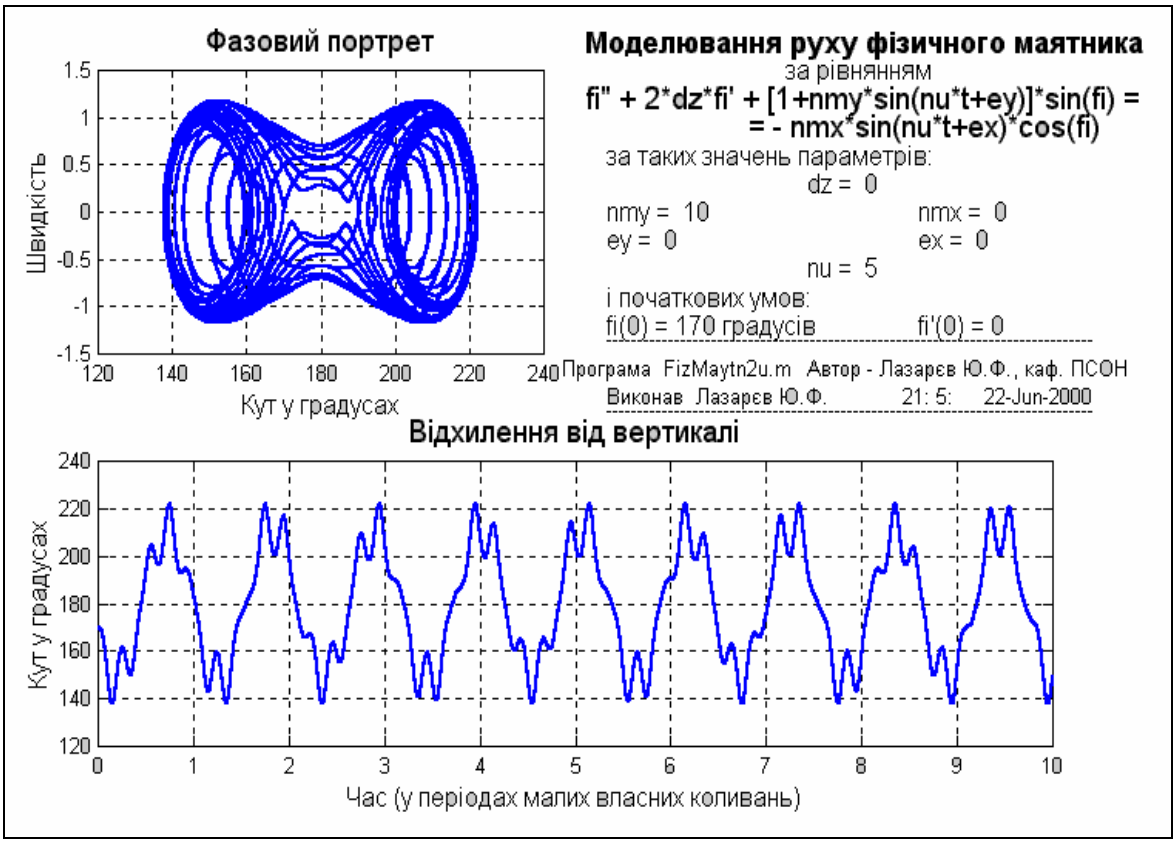


Рис. 2.15. Стійкість горішнього положення рівноваги

Рис. 2.16. Стійкість горішнього положення рівноваги при демпфіруванні

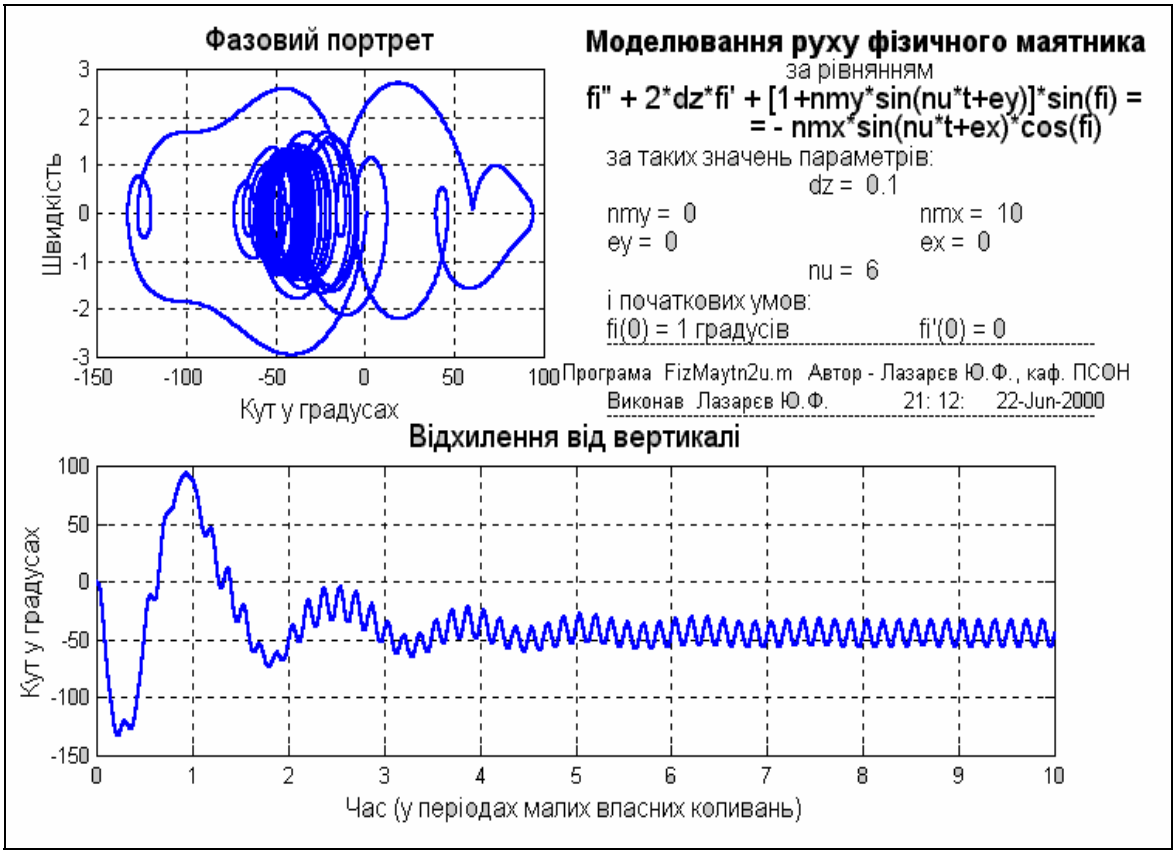


Рис. 2.17. Випрямний ефект при горизонтальній вібрації точки підвісу

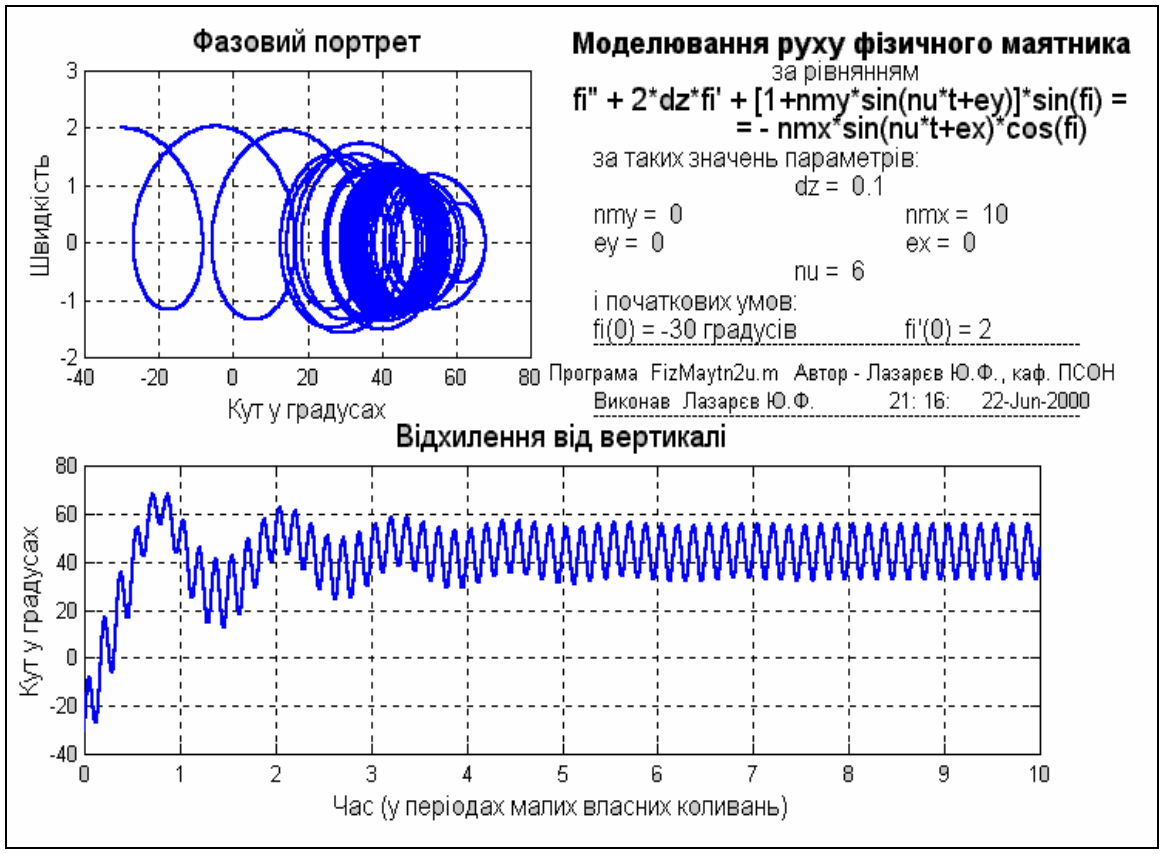


Рис. 2.18. Випрямний ефект при горизонтальній вібрації точки підвісу за інших початкових умов

Нарешті, рис. 2.16 і 2.18 демонструють можливість значних відхилень середнього положення маятника від вертикалі і при суто горизонтальній вібрації
основи. Відповідно до них це відхилення сягає величини понад 40^o при прийнятих значеннях параметрів вібрації, причому напрямок відхилення залежить від початкових умов руху маятника.

2.7.2. Завдання

За умов завдань 2.14 - 2.26 (п. 2.6.3) скласти програми чисельного інтегрування відповідних диференціальних рівнянь, які забезпечували б чисельне моделювання поводження заданої системи при заданих умовах і довільних, установлюваних користувачем значень початкових параметрів. При цьому забезпечити:

а) діалогове введення-змінювання-виведення на екран значень усіх початкових параметрів;

б) виведення графіків змінювання у часі найважливіших вихідних величин;

в) оформлення графіків відповідно до зазначених раніше вимог.

2.8. Література

1. Лазарєв Ю. Ф. Початки програмування у середовищі MatLab. Навч. посібник. – К.: Корнійчук, 1999. – 160 с.

2. Лазарев Ю. Ф. MatLAB 5.x. – К.: Издат. группа BHV, 2000. - 384 с.

3. Лазарев Ю. Ф. Моделирование процессов и систем в MATLAB. Учебный курс. – СПб.: Питер; Киев: Издат. группа BHV, 2005. – 512 с.

4. Лазарєв Ю. Ф. Моделювання на ЕОМ. Навч. посібник. – К.: Корнійчук, 2007. = 290 с.