МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА «ЕЛЕКТРИЧНА ІНЖЕНЕРІЯ»

Методичні вказівки до виконання лабораторних робіт з дисципліни «Сучасні пакети прикладних програм»

Красноармійськ - 2015

УДК №
ББК №

Методичні вказівки до лабораторних робіт з дисципліни «Сучасні пакети прикладних програм» для студентів (освітньо-кваліфікаційного рівня бакалавр) Колларов О.Ю. – Красноармійськ: ДонНТУ, 2015 рік, 110 сторінок.

Методичні вказівки містять теми лабораторних робіт, алгоритми розрахунків і методичні вказівки до їх виконання. Основна мета - допомогти студентам закріпити теоретичний матеріал курсу.

Укладач: _____ Колларов О.Ю., в. о. завідувача кафедри «Електричної інженерії», к.т.н., доцент кафедри

Рецензент: _____ Власенко М.М., доц. каф. «Електричної інженерії», к.т.н., доцент.

Відповідальний за випуск: _____ Колларов О.Ю., в. о. завідувача кафедри «Електричної інженерії», к.т.н., доцент кафедри.

Затверджено навчально-методичним відділом ДонНТУ,
протокол № 3 від 03.11.2015р.

Розглянуто на засіданні кафедри «Електричної інженерії»,
протокол № 2 від 07.10.2015р.

ЗМІСТ

1. Програмний пакет Mathematica

Лабораторна робота №1

Основні арифметичні операції, стандартні функції, константи, матриці, вектори.

Мета: ознайомитися з роботою в обчислювальному пакеті MATHEMATICA, навчитися проводити найпростіші обчислення.

1.1 Основні теоретичні положення

Wolfram Mathematica є пакетом символьної математики. Величезна кількість закладених розроблювачами функцій, а також відкрите середовище, що дозволяє доповнювати пакет своїми власними розширеннями роблять його можливості воістину безмежними. Mathematica має високу швидкість і практично не обмежену точність обчислень, що дозволяє їй працювати як на дуже потужних комп'ютерах, так і не дуже сильних персональних комп'ютерах. На основі ядра пакета є Web-сервер, що дозволяє користуватися її можливостями необмеженому числу людей.

Часто основними конкурентами пакета називають Maple, MathCAD й MatLab. Якщо з першим складно посперечатися, то щодо MathCAD й MatLab можна. Справа в тому, що ці два пакети займають зовсім іншу нішу, ніж Mathematica. Обидва при обчисленні використовують чисельні алгоритми, а не символьні. Символьні обчислення є слабко розвиненими (у порівнянні с пакетами символьних обчислень) доповненнями.

Mathematica виконує наступні стандартні арифметичні операції:

Таблиця 1.1 – Стандартні математичні операції Mathematica

«+» - додавання	«/» - ділення
«-» - вирахування	«^» - піднесення в ступінь
«*» - множення	«!» - факторіал

Операції порівняння чисел мають наступні позначення:

Таблиця 1.2 – Операції порівняння чисел Mathematica

«<» - менше	«>=» - не менше
«>» - більше	«==» - дорівнює
«<=» - не більше	«!=» - не дорівнює

При обробці вираз арифметичні операції виконуються в стандартному порядку: піднесення в ступінь (a^b), присвоєння знака (-a=(-1)*a), множення й ділення (виконується ліворуч праворуч a*b/c=(a*b)/c), додавання й вирахування, перевірка виконання умов (a==b, a<b). При записі вираз знак множення «*» може бути замінений пробілом. Для зміни порядку виконання операцій використовуються круглі дужки.

*In[1]:=1+2*3*
Out[1]=7

або

In[2]:=1+2 3
Out[2]=7

Деякі функції, наприклад Random, не вимагають аргументу. Однак вони також повинні викликатися із квадратними дужками: Random. У

Mathematica назви всіх убудованих функцій починаються із заголовної букви.

Для визначення списків і матриць використовуються фігурні дужки: {}. Список або вектор являє собою кілька об'єктів, укладених у фігурні дужки й розділених комами. Приклад:

In[4]:={x, x^2, x^3}
Out[4]={x, x2, x3}

Матриця являє собою вкладений список або список списків. Наприклад, матриця 2x2 представляється у вигляді:

In[5]:={ {1, 2}, {3, 4} }
Out[5]= {{1, 2}, {3, 4}}

Для подання матриці у звичайному виді використається функція MatrixForm.

In[6]:=MatrixForm[{{1, 2}, {3, 4}}]
Out[6]//MatrixForm=
$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Для виділення елементів списку використовуються подвійні квадратні дужки. Наприклад, для вектора v = {1, 2, 3} позначення v [[i]] відповідає і-му елементу списку за умови, що і - ціле число. Відлік елементів списку починається з 1. Приклади:

In[7]:=v = {11, 23, 5}
Out[7]= {11, 23, 5}
In[8]:=v[[3]]

Out[8]= 5

Розглянемо матрицю 2x3:

In[9]:=m= {{1, 15, 28}, {56, 4, 79}}

Out[9]={{1, 15, 28}, {56, 4, 79}}

In [10]:= MatrixForm [m]

Out [10]// MatrixForm = $\begin{pmatrix} 1 & 15 & 28 \\ 56 & 4 & 79 \end{pmatrix}$

Позначення m[[i]] відповідає і-й рядку матриці m, а m[[i, j]]- j-му елементу і-й рядка.

In[11]:=m[[i]]

Out[11]=(1, 15, 28)

In[12]:=m[[2, 3]]

Out[12]=79

Точні й наближені обчислення

У Mathematica використовуються чотири типи чисельних даних:

Таблиця 1.3 – Типи чисельних даних

Тип	Опис	Приклад
Integer	Ціле число	5
Real	Дійсне число типу nn.mm (містить десяткову крапку)	4.6
Rational	Раціональне число типу $\frac{}{}$	$\frac{2}{3}$
Complex	Комплексне число виду x+iy, де x и y – цілі, раціональні й дійсні числа	3+2.5I

Цілі й раціональні числа вважаються заданими точно. Працюючи з ними, Mathematica видає точний результат, навіть якщо виходить дуже довге вираз. Приклади:

In [15]:=2/ 4 + 24/144

Out [15]= $\frac{2}{3}$

Число, що містить десяткову крапку, розглядається як наближене. Якщо вираз містить наближені числа, то й результат обчислень буде наближеним.

In[16]:=1/2 + 2.4/144

Out[16]=0.516667

За замовчуванням Mathematica видає наближений результат з точністю шість значущих цифр. Кількість значущих цифр, використовуваних при обчисленнях, залежить від типу комп'ютера й визначається за допомогою функції Precision. Так, для останнього результату маємо:

In[17]:= Precision [%]

Out[17]= 16

Для перетворення точного результату в наближений служить функція N. Ця функція дозволяє одержати результат з будь-якою точністю. Як приклад обчислимо число π с точністю 10 значущих цифр.

In[18]= N[2 / 3]

Out[18]= 0.666667

In[19]:=[Pi, 10]

Out[19]= 3.1415926535

Mathematica прагне представляти числа як можна більш точно й видає результат у такій же формі, у якому минулому представлені уведені дані. Наприклад, ірраціональне число √17 можна представити у вигляді десяткового дробу. Однак Mathematica робить це тільки тоді, коли надходить відповідна команда, або коли число √17 задане приблизно, тобто містить десяткову крапку. Приклади:

In[18]:=17^(1/2)

Out[18]= √17

In [19]:= 17.^(1/2)

Out [19]= 4 .12311

Існує кілька функцій, що дозволяють перетворити наближений результат у точний:

Таблиця 1.4 – Функції перетворення чисел

Тип	Опис	Приклад
Rationalize[x]	Перетворить наближене число в раціональне	Rationalize[0.3] –> $\frac{3}{10}$
Ceiling[x]	Найменше ціле, не менше x	Ceiling[2.3] –> 3
Floor[x]	Найбільше ціле не більше x	Floor[3.6] –> 3
Round[x]	Найближче до x ціле (округлення)	Round[5.3] –> 5
Chop[x]	Заміна близьких до нуля чисел точним нулем	Chop[1.10^-11] –> 0

За замовчуванням максимальне число, що функція Chop заміняє нулем, рівняється 10^{-11}. Однак ця функція може викликатися із двома аргументами, причому другий аргумент визначає граничне число, починаючи з якого числа заміняються нулем.

In [20]:= *Chop[1.10^-4, 10^-5]*

Out [20]= *0*

In[21]:=Chop [1.10^-4, 10^-5]

Out [21]= *0. 0001*

Mathematica знає найбільше часто використовувані математичні постійні й розглядає їх як числа, що задані точно за допомогою функції N постійні можуть бути обчислені з будь-якою точністю. Позначення деяких постійних наведені нижче в таблиці:

Таблиця 1.5 – Математичні постійні

Постійна	Позначення	Назва	Чисельне значення
П	Pi	Відношення довжини окружності до її діаметра	≈3. 141592
E	E	Підстава натуральних логарифмів	≈2. 71828
i або *j*	I	Мнима одиниця	≈√-1
π/180	Degree	Множник для переведення градусів у радіани	≈0. 017453
	GoldenRatio	Золоте відношення	≈1. 61803
Γ	EulerGamma	Постійна Ейлера	≈0. 577216
∞	Infinity	Нескінченність	

Для генерації рівномірно розподілених псевдоскалярних чисел служить функція Random. Якщо ця функція не містить аргументу, то вона генерує дійсні числа з відрізка [0, 1].

In [22]:= Random []

Out[22]= 0.0581936

Як аргументи функції Random можна вказати тип числа й інтервал, у якому перебувають числа, що генеруються. Так, у наступному прикладі генеруються комплексні числа, дійсна частина яких лежить на відрізку [1, 5], а мнима - на відрізку [2, 8].

In[23]:= Random[Complex, {1+21, 5 + 81}]
Out[23]= 1.56434 + 7 .24709 I

Вбудовані математичні функції

Mathematica використовує велику кількість убудованих математичних функцій. Якщо для функції немає стандартної абревіатури, у якості її імені використається відповідне англійське слово. У кожному разі назва вбудованої функції обов'язково пишеться із заголовної букви. Якщо назва функції складається з декількох слів, слова пишуться одне за іншим без пробілів, причому кожне слово пишеться із заголовної букви. Аргументи функцій полягають у квадратні дужки.

Таблиця 1.6 – Математичні функції пакету Mathematica

Sqrt[x]	Квадратний корінь ($\sqrt{x}$)
Exp[x]	Експонента (e^x)
Log[x]	Натуральний логарифм (ln x)
Log[b, x]	Логарифм заосновою b ($\log_b x$)
Sin[x], Cos[x], Tan[x], Cot[x], Sec[x], Csc[x]	Тригонометричні функції
ArcSin[x], ArcCos[x], ArcTan[x], ArcCot[x], ArcSec[x], ArcCsc[x]	Зворотні тригонометричні функції
Sinh[x], Cosh[x], Tanh[x], Coth[x], Sec[x], Csch[x]	Гіперболічні функції

ArcSinh[x], ArcCosh[x], ArcTanh[x], ArcCoth[x], ArcSech[x], ArcCsch[x]	Зворотні гіперболічні функції
Abs[z]	Модуль комплексного числа |z|
Arg[z]	Аргумент φ комплексного числа
Re[z], Im[z]	Дійсна й мнима частини комплексного числа
Mod[n, m]	Остача від ділення n на m
Min[x, y, …]	Мінімальне число із множини [x, y, …]
Max[x, y, …]	Максимальне число із множини [x, y, …]
Zeta[z]	Дзета-функція Римана
Gamma[z]	Гамма-функція Ейлера

Лабораторна робота №2
Чисельне й аналітичне розв’язання звичайних рівнянь та їхніх систем

Мета: здобуття практичних навичок розв’язання звичайних рівнянь та їхніх систем за допомогою прикладного пакету Mathematica

1.1 Основні теоретичні положення

Розв’язання рівнянь

Mathematica має значну кількість потужних засобів для розв’язання багатьох видів рівнянь.

Для розв’язання рівнянь застосовується функція Solve. Варто пам'ятати, що при записі рівняння використовується подвійний знак рівності «==», а не одинарний «=»:

In[1]:= Solve[ArcSin[x]==0,x]

Out[1]= {{x→0}}

Результат виводиться у вигляді правил (-a) (Rule), вкладених у список. Зовнішній список містить всі розв’язання, а кожен внутрішній містить один корінь. Дане рівняння має три корені:

In[2]:= Solve[x^3+3x==0,x]

Out[2]= {{x→0},{x→-i√3},{x→i√3}}

Для розв’язання системи рівнянь, використайте список як перший аргумент:

In[3]:= Solve[{x^2-1==0,x^2-3x+2==0},x]

Out[3]= {{x→1}}

Для даного приклада існують одночасно два розв’язання системи рівнянь; кожен набір рішень укладений у власний список:

```
In[4]:= Solve[{x-y==0,x^2+y^2==1},{x,y}]
Out[4]= {{x→-1/Sqrt[2],y→-1/Sqrt[2]},{x→1/Sqrt[2],y→1/Sqrt[2]}}
```

Тут розв’язання дається шляхом вираз однієї змінної через іншу:

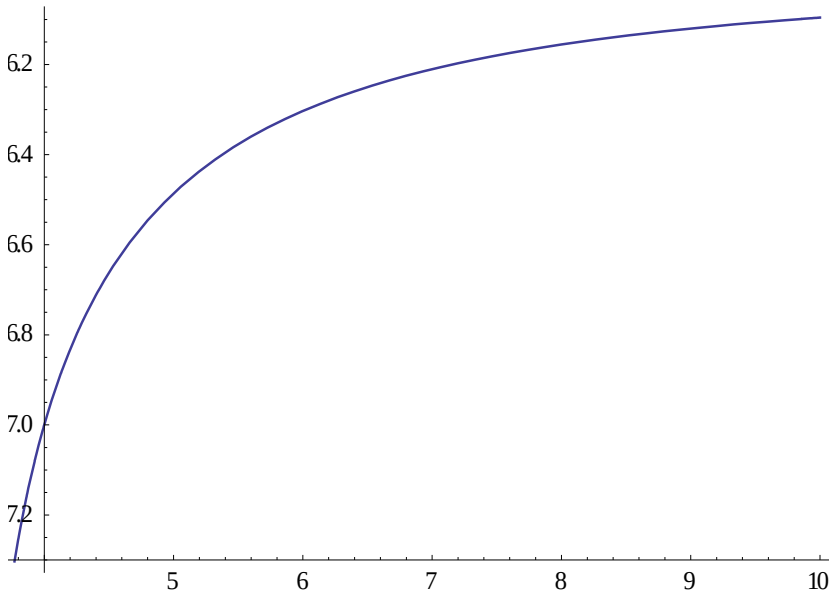
```
In[5]:= sol=Solve[x^2+yx+3==0,x]
Out[5]= {{x→1/2 (-y-Sqrt[-12+y^2])},{x→1/2 (-y+Sqrt[-12+y^2])}}
```

Щоб виділити й вивести конкретний корінь із загального списку рішень (у цьому прикладі виведений тільки перший корінь), використайте [[...]] (коротку форму запису функції Part) і комбінацію символів /. (коротка форма запису функції ReplaceAll) для застосування правила:

```
In[6]:= x/.sol[[1]]
Out[6]= 1/2 (-y-Sqrt[-12+y^2])
```

Як приклад, побудуємо графік вираз x^2 - y^2 залежно від значень змінної y, резонно думаючи, що x привласнено перше розв’язання згаданого вище рівняння:

```
In[7]:= Plot[x^2-y^2/.sol[[1]],{y,Sqrt[12],10}]
Out[7]=
```



Систему рівнянь із декількома змінними Ви можете вирішити для декількох або для всіх змінних, варіюючи список у другому аргументі:

In[2]:= Solve[{x^2+yx+3==0,x+y==1},{x,y}]

Out[2]= {{x→-3,y→4}}

Якщо система недовизначена, Mathematica дасть відповідь, виражена через вільну змінну:

In[3]:= Solve[{x^2+yx+z==0,x+y==1},{x,y}]

Out[3]= {{x→-z,y→1+z}}

Функція Solve знаходить так звані "загальні" розв’язання рівнянь. Це ті розв’язання, які не залежать від змінних, не зазначених у другому аргументі. Наприклад:

In[10]:= Solve[xy==0,x]

Out[10]= {{x→0}}

Природа множника y не має в цьому випадку значення: цікавлять

тільки значення змінної x , при яких рівняння звертається у вірну рівність. Але є й інше розв'язання, що залежить від y, а саме: рівність 0 змінної y. Додавання y у другий аргумент команди змушує показати це розв'язання:

In[4]:= Solve[xy==0,{x,y}]

Solve::svars: Equations may not give solutions for all "solve" variables.

Out[4]= {{x→0},{y→0}}

Є й інші випадки, коли Solve не знаходить всі розв'язання. Наприклад:

In[5]:= Solve[Cos[x]==1/2,x]

Out[5]= {{x→ConditionalExpression[-π/3+2πC[1],C[1]∈Integers]},

{x→ConditionalExpression[π/3+2πC[1],C[1]∈Integers]}}

Ви також можете вирішувати ірраціональні рівняння:

In[11]:= Solve[1/x+1/Sqrt[x]==1,x]

Out[11]= {{x→1/2 (3+√5)}}

Зверніть увагу, що в ірраціональних рівняннях Solve відкидає сторонні коріння. Щоб переглянути всі корені, включаючи сторонні, встановіть для параметра VerifySolutions опцію False:

In[16]:= Solve[1/x+1/Sqrt[x]==1,x,VerifySolutions→False]

Out[16]= {{x→1/2 (3-√5)},{x→1/2 (3+√5)}}

Так робиться перевірка розв'язання:

In[17]:= 1/x+1/Sqrt[x]==1/. %//FullSimplify

Out[17]= {False, True}

Рівняння можна вирішувати й за допомогою функції Reduce:

In[13]:= Reduce[x^3+3x==0,x]
Out[13]= x==0||x==-i√3||x==i√3

Відображення результату для Reduce відрізняється від відображення результату для Solve: функція Reduce видає логічне вираз, еквівалентне вихідному рівнянню, тому вона ніколи не опускає розв'язання:

In[14]:= Reduce[xy==0,x]
Out[14]= x==0||y==0
In[15]:= Reduce[Cos[x]==1/2,x]
Out[15]= C[1]∈Integers&&(x==-π/3+2πC[1]||x==π/3+2πC[1])

Mathematica також дозволяє знаходити чисельні розв'язання рівнянь.

Наприклад, Ви можете застосувати функцію N до результату функції Solve для того, щоб одержати наближене чисельне значення символьного розв'язання:

In[18]:= nsoll=Solve[x^2+2x-7==0,x]
Out[18]= {{x→-1-2√2},{x→-1+2√2}}
In[19]:= N[nsoll]
Out[19]={{x→-3.8284271247461903},{x→1.8284271247461903}}

Крім цього, Ви можете одержати чисельне розв'язання прямо, скориставшись функцією NSolve, що очевидно швидше комбінації N й Solve:

In[20]:= NSolve[x^2+2x-7==0,x]

Out[20]= {{x→-3.8284271247461903},{x→1.82842712474619}

Застосуємо NSolve для розв'язання більш складного поліноміального рівняння:

In[21]:=poly=3+3x-7x^2-x^3+2x^4+3x^7-3x^8-x^9+x^10

In[22]:= NSolve[poly==0,x]

Out[22]=

{{x→-1.7320508075688772},

{x→-0.8686883014299477-0.5852818552099751i},

{x→-0.8686883014299477+0.5852818552099751i},

{x→-0.496292071389336 4},

{x→0.07635557774869899-1.1409461420499265i},

{x→0.07635557774869899+1.1409461420499265i},{x→1.},

{x→1.0404787593759168-0.5673495826845649i},

{x→1.0404787593759168+0.5673495826845649i},

{x→1.7320508075688772}}

Функція NSolve використається й для чисельного розв'язання систем рівнянь. Використайте такий же самий синтаксис, що й для Solve:

In[23]:=NSolve[{x+y==2,x-3y+z==3,x-y+z==0},{x,y,z}]

Out[23]= {{x→3.5,y→-1.5,z>?5.}}

Якщо рівняння містять тільки лінійні функції або багаточлени, то можна використати NSolve для одержання наближених чисельних результатів для всіх рішень. Однак, для рівнянь, що містять більш складні функції, не існує єдиного підходу для знаходження всіх рішень, навіть чисельних. У таких випадках, для пошуку рішень доцільно скористатися

функцією FindRoot.

Використаємо FindRoot для пошуку чисельних рішень для x, починаючи з 1:

```
In[24]:= FindRoot[3Cos[x]==Log[x],{x,1}]
Out[24]= {x→1.447258617277903}
```

2.2 Завдання до виконання лабораторної роботи

Згідно з номером варіанту розв'язати наступні рівняння та системи рівнянь

Таблиця 2.1 – Рівняння та системи рівнянь

№	**Завдання**	
1	$x^4-x^3+5*x^2=0$	$\begin{cases} x+2y+4z=8 \\ 3x-6y+9z=3 \\ 7x+7y-3z=14 \end{cases}$
2	$x^3-x^2-5=0$	$\begin{cases} x-y+5z=3 \\ 3x-2y-7z=1 \\ 3x+4y-z=-1 \end{cases}$
3	$2x^2-5x=7$	$\begin{cases} x+2y-z=-3 \\ 2x+3y+z=-1 \\ x-y-z=3 \end{cases}$
4	$log_{x^2}\frac{4}{(7-x)}=5$	$\begin{cases} 2x+3y=5 \\ 4x+6y+z=7 \\ 5y-4z=12 \end{cases}$
5	$lg\,x=x-5$	$\begin{cases} 3x-4y-4z=5 \\ 6x-8y-2z=10 \\ x+y-5z=-4 \end{cases}$
6	$\frac{x-1}{x^2}=4$	$\begin{cases} x+3y-z=0 \\ 2x-y+3z=1 \\ 7x+7y+3z=2 \end{cases}$
7	$(x+4)^2=1$	$\begin{cases} x+3y-4z=5 \\ 2x-3y+6z=11 \\ 8x-3y+10z=21 \end{cases}$
8	$\frac{x^2-1}{x+2}=4x$	$\begin{cases} 3x+y+z=-2 \\ 5x-y-z=10 \\ x-y+5z=-12 \end{cases}$

9	$tg\ (x^2-1)=0.5$	$\begin{cases} x-2y+z=3 \\ 2x-4y+2z=5 \\ 3x-6y+3z=9 \end{cases}$
10	$\sqrt{x-1}=x$	$\begin{cases} x-4y+3z=3 \\ 2x-8y+6z=10 \\ 3x-12y+9z=15 \end{cases}$

2.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити наступні пункти:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи
3. Стислий зміст роботи
4. Результати роботи

2.4 Контрольні запитання

1. Які функції для розв’язання рівнянь Ви знаєте?
2. Назвіть синтаксис цих функцій
3. Яким чином розв’язується система рівнянь?
4. У чому полягає відмінність аналітичного розв’язання від чисельного?

Лабораторна робота №3

Розв’язання диференціальних рівнянь і систем

Мета: здобуття практичних навичок розв’язання диференціальних рівнянь та їхніх систем за допомогою прикладного пакету Mathematica

1.1 Основні теоретичні положення

Функції Mathematica для розв’язання диференціальних рівнянь можуть застосовуватися до багатьох різних класів диференціальних рівнянь, автоматично вибираючи відповідні алгоритми, без необхідності їхньої попередньої обробки користувачем.

Використаємо DSolve для розв’язання диференціального рівняння y`(x)=x для y(x) з незалежною змінною x:

```
In[2]:= DSolve[y'[x]==x,y[x],x]
Out[2]= {{y[x]→x^2/2+C[1]}}
```

Розв’язання, що дає функція DSolve, є вкладеним списком із правил. Зовнішній список містить у собі всі розв’язання, а кожен менший список є приватним розв’язанням.

Якщо Ви бажаєте використати розв’язання як функцію, задайте спочатку правило визначальне ім'я для розв’язання, у цьому випадку solution:

```
In[3]:= solution=DSolve[y'' [x]==x,y[x],x]
Out[3]= {{y[x]→x^2/2+C[1]}}
```

Тепер, використаємо функцію Part для здобуття першої частини

розв’язання, використовуючи коротку форму цієї функції solution[[1]]. Замінимо y[x], використовуючи /. (коротку форму запису для ReplaceAll), і застосуємо = для визначення функції f[x]:

```
In[4]:= f[x_] = y[x] /. solution[[1]]
Out[4]= -Cos[x] + C[1]
```

Тепер, f[x] обчислюється як будь-яка звичайна функція:

```
In[5]:= f[4]
Out[5]= 8 + C[1]
```

Для завдання початкових умов, помістить рівняння й початкові умови (y(0)=1 й y'(0)=1) у список:

```
In[6]:= DSolve[{y''[x] == y[x], y[0] == 1, y'[0] == 1}, y[x], x]
Out[6]= {{y[x] → e^x}}
```

Якщо заданих початкових умов недостатньо, будуть виведені константи C[n]:

```
In[10]:= DSolve[y''[x] == y[x], y[x], x]
Out[10]= {{y[x] → e^x C[1] + e^-x C[2]}}
```

Щоб указати для яких функцій повинне бути знайдене розв’язання, використайте другий список:

```
In[7]:=
DSolve[{y'[x] == z[x], z'[x] == -y[x], y[0] == 0, z[0] == 1}, {y[x], z[x]}, x]
Out[7]= {{y[x] → Sin[x], z[x] → Cos[x]}}
```

У наведеному прикладі, розв’язання є не елементарними функціями:

In[8]:=

DSolve[{x'[s]==Cos[t[s]],y'[s]==Sin[t[s]],t'[s]==s,x[0]==0,y[0]==0,
y[0]==0,t[0]==0},{x[s],y[s],t[s]},s]

Out[8]=$\{\{t[s]\to\frac{s^2}{2},x[s]\to\sqrt{\pi}FresnelC\left[\frac{s}{\sqrt{\pi}}\right],y[s]\to\sqrt{\pi}FresnelS\left[\frac{s}{\sqrt{\pi}}\right]\}\}$

Ви можете використати DSolve, /., Table й Plot разом, щоб відобразити графічно розв’язання недовизначених диференціальних рівнянь при різних значеннях константи.

По-перше, вирішимо диференціальне рівняння за допомогою DSolve і задамо результату ім'я solution:

In[11]:= solution=DSolve[{y''[x]==y[x],y'[0]==0},y[x],x]

Out[11]= $\{\{y[x]\to e^{-x}(1+e^{2x})C[1]\}\}$

Використаємо =, /. і Part для визначення функції g[x] з використанням solution:

In[12]:= g[x_]=y[x]/.solution[[1]]

Out[12]= $e^{-x}(1+e^{2x})C[1]$

Визначимо список функцій t[x] для цілих значень C[2] від 1 до 10:

In[15]:= t[x_]=Table[g[x]/.C[1]→j,{j,1,10}]

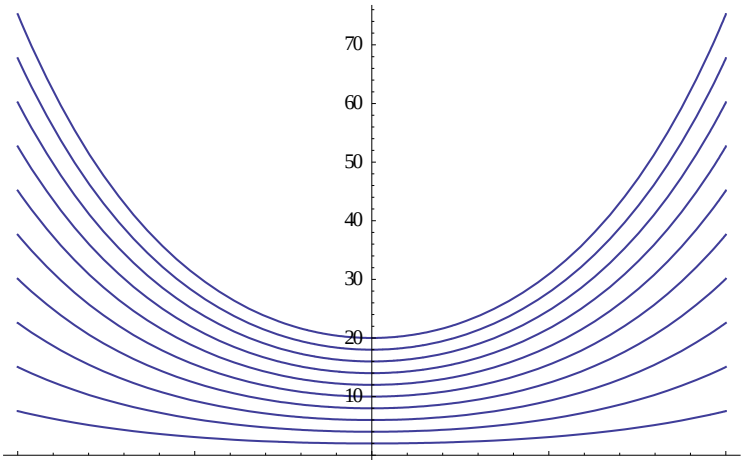
Out[15]=

$\{e^{-x}(1+e^{2x}),2e^{-x}(1+e^{2x}),3e^{-x}(1+e^{2x}),4e^{-x}(1+e^{2x}),5e^{-x}(1+e^{2x}),6e^{-x}(1+e^{2x}),$
$7e^{-x}(1+e^{2x}),8e^{-x}(1+e^{2x}),9e^{-x}(1+e^{2x}),10e^{-x}(1+e^{2x})\}$

Застосуємо функцію Plot для побудови графіків функцій зі списку в діапазоні від -2<x<2:

In[16]:= Plot[{t[x], x}, {x, -2, 2}]

Out[16]=



3.2 Завдання до лабораторної роботи

Згідно з номером варіанту розвязати наступні диференціальні рівняння та їхні системи

Таблиця 3.1 – Рівняння та системи дифференційних рівняннь

№	Завдання	
1	$tg\ x=\sqrt{x^2+3}$	$\begin{cases} x'=-2x+4y \\ y'=-x+3y \end{cases}$
2	$x^2y'=\frac{1}{\cos y}$	$\begin{cases} 2x''=-6x+2y \\ y''=2x-2y \end{cases}$
3	$y'=x^3y^2+2x^3y$	$\begin{cases} x'=-2y \\ y'=0.5x \end{cases}$
4	$xy^2=2xy'+3$	$\begin{cases} x'=-y \\ y'=1.01x-0.2y \end{cases}$
5	$(x+1)y'=y^2\sin x+xy^2$	$\begin{cases} x''-5x+4y=0 \\ y''+4x-5y=0 \end{cases}$
6	$(x^2+e^x)y'=xy+y^2\cos x$	$\begin{cases} x''=(1-y)x \\ y'=(1-x)y \end{cases}$

7	$y' = \frac{2x+4y}{x-3y}$	$\begin{cases} x' = -2y \\ y' = 2x \end{cases}$
8	$y' = tg\, ln\, sin \frac{y}{x}$	$\begin{cases} x' = 0.5y \\ y' = -8x \end{cases}$
9	$y' = cos \frac{y}{x} + \frac{x^2}{y^2}$	$\begin{cases} x' = 10y \\ y' = -10x \end{cases}$
10	$y' = \frac{\sqrt{x^4 - 2y^4}}{\sqrt[3]{x^6 + xy^5 + 2x^4y^2}}$	$\begin{cases} x' = -y \\ y' = 10x - 7y \end{cases}$

3.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити наступні пункти:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи
3. Стислий зміст роботи
4. Результати роботи

3.4 Контрольні запитання

1. Які функції для розв’язання диференціальних рівнянь Ви знаєте?
2. Назвіть синтаксис цих функцій
3. Яким чином розв’язується система диференціальних рівнянь?
4. У чому полягає відмінність аналітичного розв’язання від чисельного?

Лабораторна робота №4
Мова програмування Mathematica

Мета: оволодіти навичками програмування за допомогою прикладного пакету Mathematica

4.1 Основні теоретичні положення

Крім того, Mathematica це мова функціонального програмування. Mathematica допускає відкладені обчислення за допомогою оператора визначення «:=».

Можна сказати, що система Mathematica написана мовою Mathematica, хоча деякі функції, що особливо стосуються лінійної алгебри, з метою оптимізації були написані мовою C.

Функціональне програмування

Приклад коду, що послідовно 5 разів застосовує функцію f:

NestList[f, x, 5]

Результатом буде список з аргументу x, одного, двох, трьох і так далі до п'яти застосувань функції f до цього аргументу, усього 6 елементів списку:

{x, f[x], f[f[x]], f[f[f[x]]], f[f[f[f[x]]]], f[f[f[f[f[x]]]]]}

Приклад 3-х кратного застосування конкретної функції Sin[x + 1] має вигляд:

NestList[Sin[# + 1] &, x, 3]

Результатом буде:

{x, Sin[1 + x], Sin[1 + Sin[1 + x]], Sin[1 + Sin[1 + Sin[1 + x]]]}

Процедурне програмування

Mathematica також підтримує процедурний стиль програмування:

```
For[i = 1; t = x, i^2 < 10, i++,
t = t^2 + i;
Print[[t]]
```

Крапка з комою відокремлює різні послідовні команди. Для здійснення стандартних циклів є функції Do, While, For. Умовні вираз здійснюються за допомогою функцій If, Which, Switch. Таким чином, будучи споконвічно функціональною мовою програмування Mathematica, проте, має функції, які дозволяють писати код дуже близький до стандартних процедурних мов програмування [3].

Програмування за допомогою завдання правил

У системі Mathematica також можна задавати правила роботи з тими або іншими виразами. Таким чином, можна визначати об'єкти подібно тому як це звичайно робиться в Mathematica, задаючи їхньої властивості за допомогою правил:

f[x_ + y_] := f[x] + f[y];

f[a + b + c]

Результат:

f[a] + f[b] + f[c]

Об’єктно-оріентоване програмування

Mathematica дозволяє також явно задавати визначення, пов'язані з певним об'єктом, реалізуючи тим самим можливість використання об’єктно-оріентованого стилю програмування:

h /: f[h[x_], x_] := hf[x];

h /: p_[h[x_]] := ph[f, x];

h /: h[x_] + h[y_] := hsum[x, y];

Обчислюючи (з даними визначеннями):

h[a] + h[b] + p[h[r]] + h[h[x]]

результатом буде:

hsum[a, b] + ph[f, r] + ph[f, x]

4.2 Завдання до виконання лабораторної роботи

Створити масив даних, що повинен містити не менше 5 різних додатних та 5 різних від’ємних чисел, щонайменше 2 нулі. Далі згідно з номером варіанту сформувати наступний масив:

Таблиця 4.1 – Завдання на програмування

№ варіанту	Завдання

1	Тільки з від’ємних чисел за зростанням
2	Тільки з додатних чисел, більших за 3
3	Тільки з від’ємних чисел за зменшенням
4	Тільки з додатних чисел за зростанням
5	Тільки з додатних чисел за зменшенням
6	Тільки з від’ємних чисел більших за -1
7	Відмінних від нуля від’ємних чисел
8	Відмінних від нуля додатних чисел
9	Виключити нулі з масиву
10	Залишити у масиві один нуль та всі додатні числа

4.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи
3. Стислий зміст роботи
4. Результати роботи

4.4 Контрольні запитання

1. Які можливості програмування надає програмний пакет?
2. Назвіть та охарактеризуйте оператори циклів
3. Яким чином формується цикл програми?

Програмний пакет Matlab

Лабораторна робота №1

Основні арифметичні операції, стандартні функції, константи, матриці, вектори.

Мета: ознайомитися з роботою в обчислювальному пакеті Matlab, навчитися проводити найпростіші обчислення.

1.1 Основні теоретичні положення

MATLAB – пакет прикладних програм для розв’язання задач технічних обчислень й однойменнамова програмування.

Мова MATLAB є високорівневим язиком програмування, що включає засновані наматрицяхструктури даних, широкий спектр функцій, інтегроване середовище розробки, об’єктно-оріентовані можливості й інтерфейси до програм, що написані на інших мовах програмування.

Програми, що написані на MATLAB, бувають двох типів - функції й скріпти. Функції мають вхідні й вихідні аргументи, а також власний робочий простір для зберігання проміжних результатів обчислень і змінних. Скріпти же використають загальний робочий простір. Як скріпти, так і функції не компілюються в машинний код і зберігаються у вигляді текстових файлів. Існує також можливість зберігати так називані pre-parsed програми - функції й скріпти, оброблені у вид, зручний для машинного виконання. У загальному випадку такі програми виконуються швидше звичайних, особливо якщо функція містить команди побудови графіків.

Числа

MATLAB використає прийняту десяткову систему числення, з необов'язковою десятковою крапкою й знаками плюс-мінус для чисел.

Наукова система числення використає букву **e** для визначення множника ступеня десяти. Уявні числа використають i або j як суфікс.

Всі числа для зберігання використають формат long, певний стандартом плаваючої крапки IEE. Числа із плаваючою крапкою мають обмежену точність - приблизно 16 значущих цифр й обмеженим діапазоном - приблизно від 10^{-308} до 10^{308} (Комп'ютер VAX використає інший формат чисел із плаваючою крапкою, але їхня точність і діапазон приблизно ті ж).

Оператори

Вираз використають звичайні арифметичні операції й правила старшинства:

Таблиця 1.1 – Арифметичні операції Matlab

+	додавання
-	вирахування
*	множення
/	Ділення
\	ліве ділення
^	Ступінь
‘	комплексно сполучене транспонування
()	визначення порядку обчислення

Функції

MATLAB надає велику кількість елементарних математичних функцій, таких як abs, sqrt, exp, sin. Обчислення квадратного кореня або логарифма негативного числа не є помилкою: у цьому випадку результатом є відповідне комплексне число.

Щоб вивести список всіх елементарних математичних функцій, наберіть:

help elfun

Для висновку більше складних математичних і матричних функцій, наберіть

help specfun

help elmat

Деякі функції, такі як *sqrt* й *sin,* –вбудовані. Вони є частиною **MATLAB**, тому вони дуже ефективні, але їхні обчислювальні деталі важко доступні. У той час як інші функції, такі як *gamma* й *sink,*реалізовані в М-файлах. Тому ви можете легко побачити їхній код й, якщо буде потреба, навіть модифікувати його.

Таблиця 1.2 – Константи пакету Matlab

Pi	3. 14159265...
I	мнима одиниця
J	те ж саме, що й і
Eps	відносна точність числа із плаваючою крапкою
Realmin	найменше число із плаваючою крапкою
Realmax	найбільше число із плаваючою крапкою
Inf	нескінченність
Na	не число

Матриці

Скаляри, вектори й матриці

В MatLab можна використовувати скаляри, вектори й матриці. Для уведення скаляра досить приписати його значення якийсь змінні, наприклад

>>p=2

ans

p=2

MatLab розрізняє заголовні й прописні букви, так що p й P - це різні змінні. Для уведення масивів (векторів або матриць) їхні елементи беруть у квадратні дужки. Так для уведення вектора-рядка розміром 1x3, використається наступна команда, у якій елементи рядка відокремлюються пробілами або комами.

>>x=[2 8 7]

ans

x= 2 8 7

При уведенні вектора-стовпця елементи розділяють крапкою з коми. Наприклад:

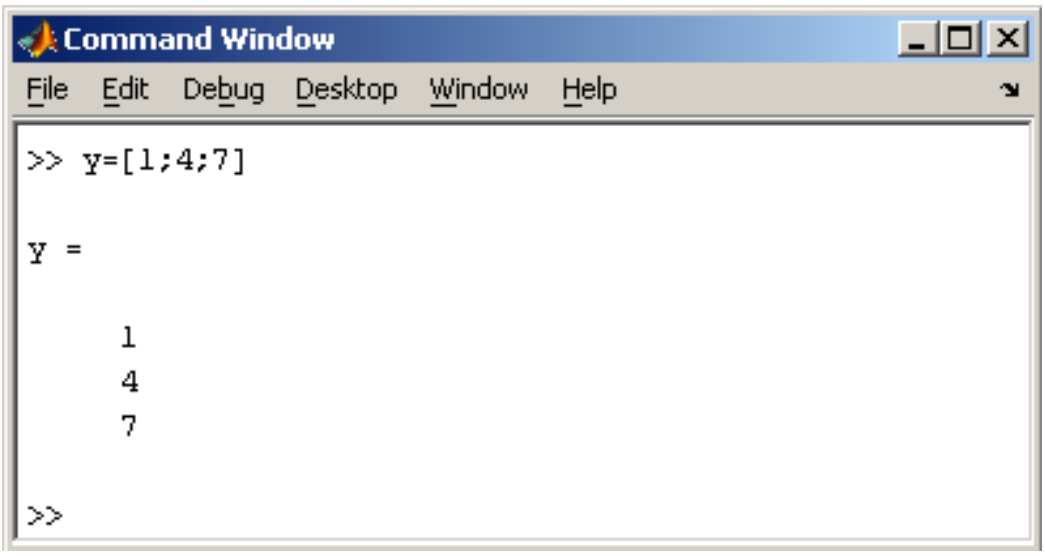


Рисунок 1.1 – Введення вектора-стовпця

Уводити невеликі за розміром матриці зручно прямо з командного рядка. При уведенні матрицю можна розглядати як вектор-стовпець, кожен елемент якого є вектором-рядком.

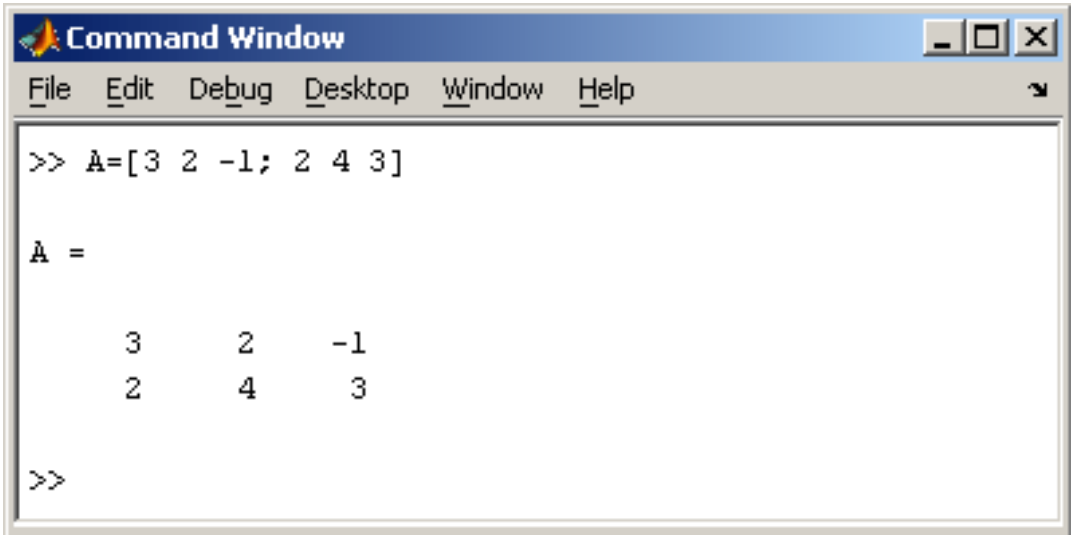


Рисунок 1.2 – Введення матриці

Доступ до елементів

Доступ до елементів матриць здійснюється за допомогою двох індексів - номерів рядка й стовпця, ув'язнених у круглі дужки, наприклад команда B(2,3) видасть елемент другого рядка й третього стовпця матриці. Для виділення з матриці стовпця або рядка треба в якості одного з індексів використати номер стовпця або рядка матриці, а інший індекс замінити двокрапкою. Наприклад, запишемо другий рядок матриці A у вектор z

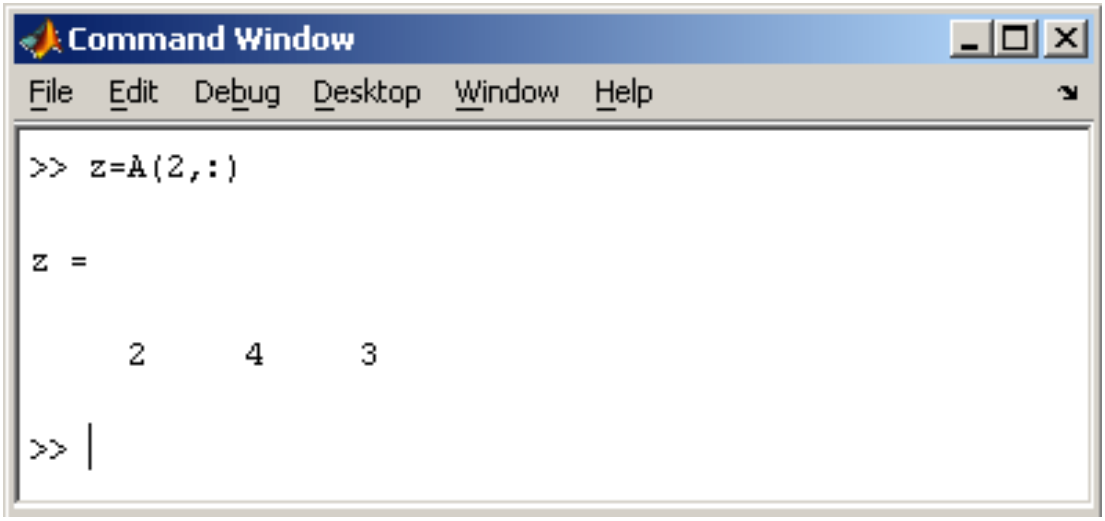


Рисунок 1.3 – Виділення рядка

Також можна здійснювати виділення блоків матриць за допомогою двокрапки. Наприклад, виділимо з матриці P блок відзначений кольорами

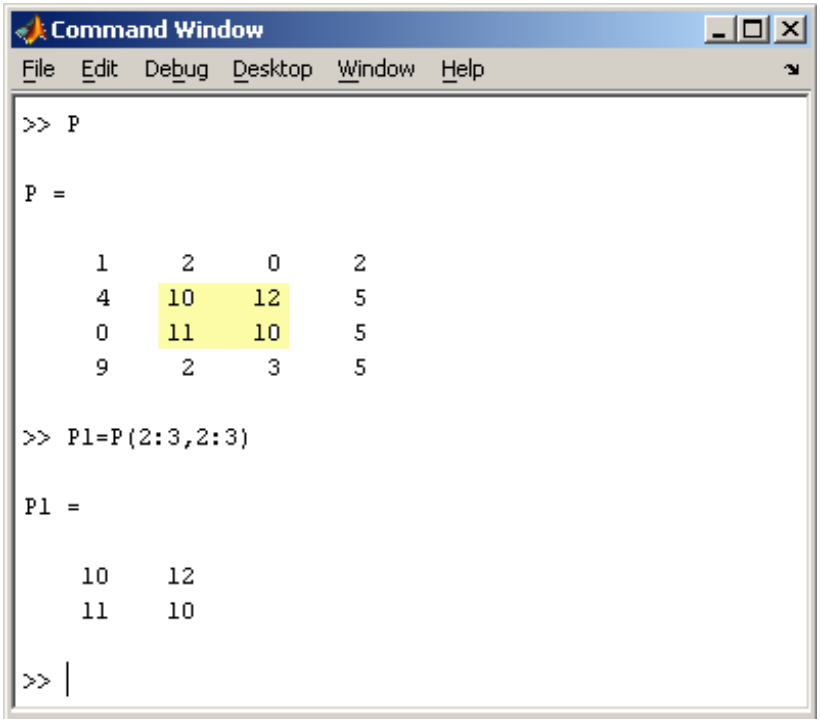


Рисунок 1.4 – Виділення блоку матриці

Якщо необхідно подивитися змінні робітничі середовища, у командному рядку необхідно набрати команду whos.

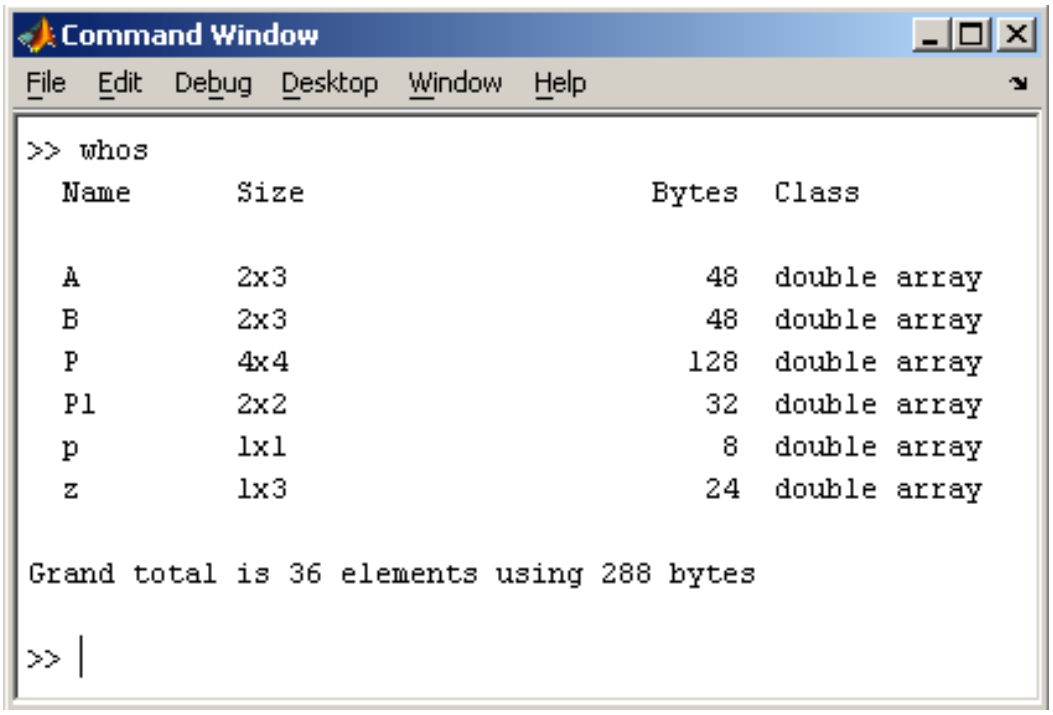


Рисунок 1.5 – Вікно Command Window, функція whos

Основні матричні операції

При використанні матричних операцій варто пам'ятати, що для додавання або вирахування матриці повинні бути одного розміру, а при перемножуванні число стовпців першої матриці повинно рівнятися числу рядків другої матриці. Додавання й вирахування матриць, так само як чисел і векторів, здійснюється за допомогою знаків плюс і мінус

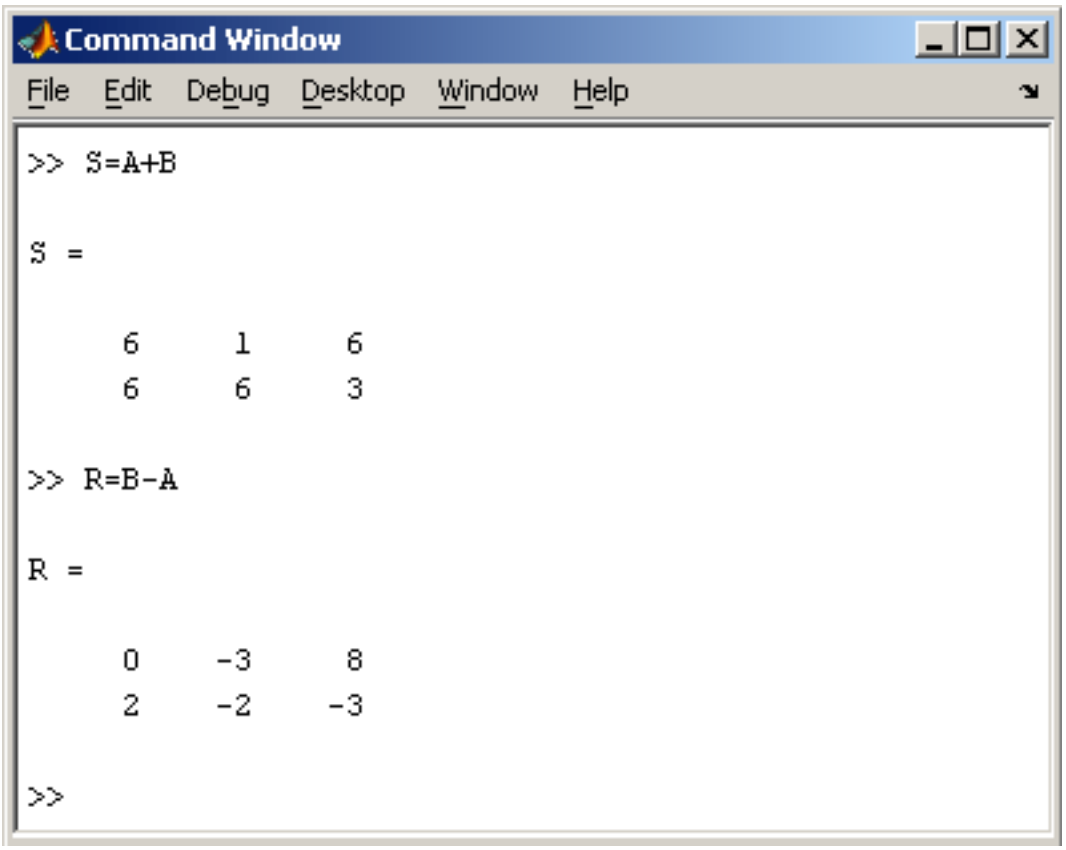


Рисунок 1.6 – Арифметичні дії над матрицями

а множення — знаком зірочка *. Уведемо матрицю розміром 3x2

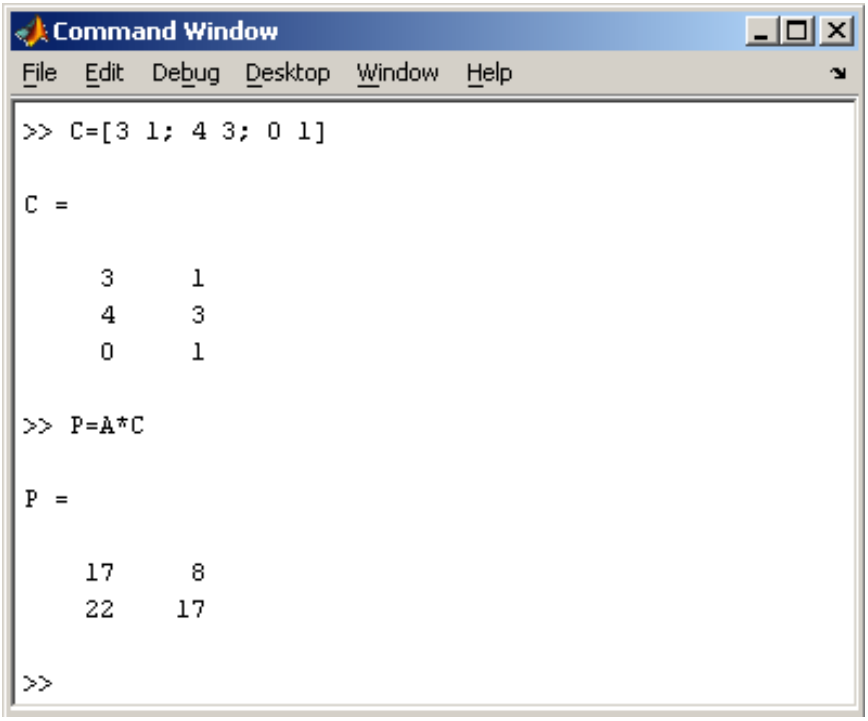


Рисунок 1.7 – Арифметичні дії над матрицями

Множення матриці на число теж здійснюється за допомогою зірочки, причому множити на число можна як праворуч, так і ліворуч. Піднесення квадратної матриці в цілий ступінь виробляється з використанням оператора ^

Створення матриць спеціального виду

Заповнення прямокутної матриці нулями виробляється убудованою функцією zeros

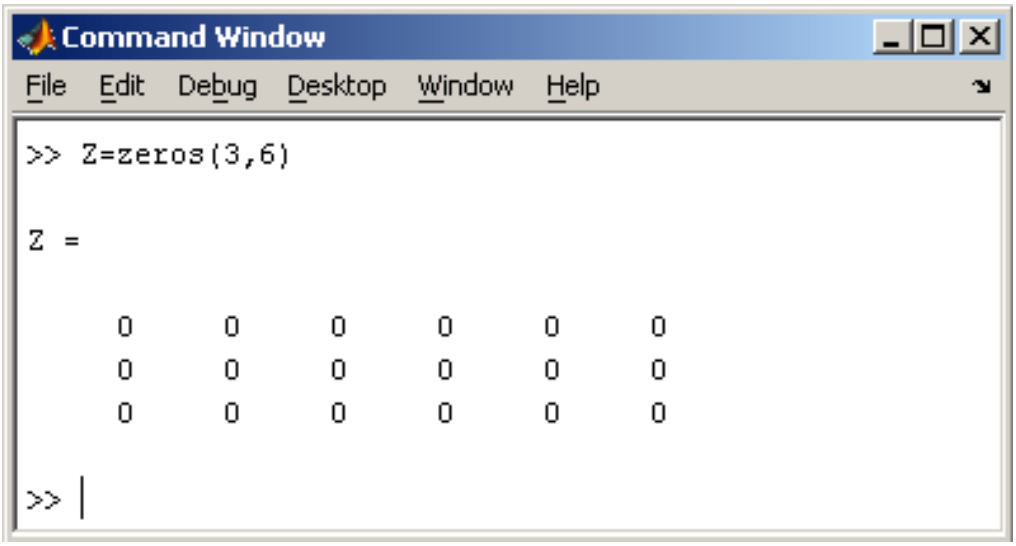


Рисунок 1.8 – Нульова матриця

Одинична матриця створюється за допомогою функції eye

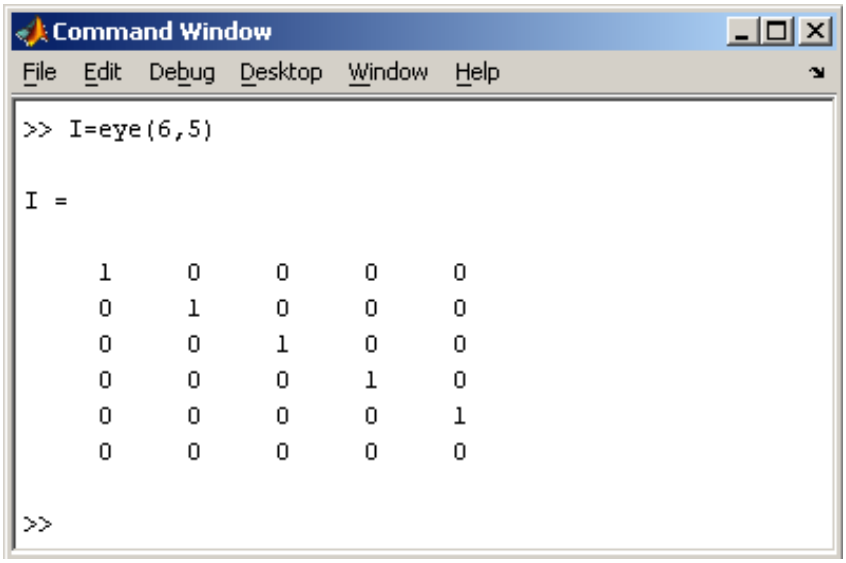


Рисунок 1.9 – Одинична матриця

Матриця, що складається з одиниць, утвориться в результаті виклику функції ones

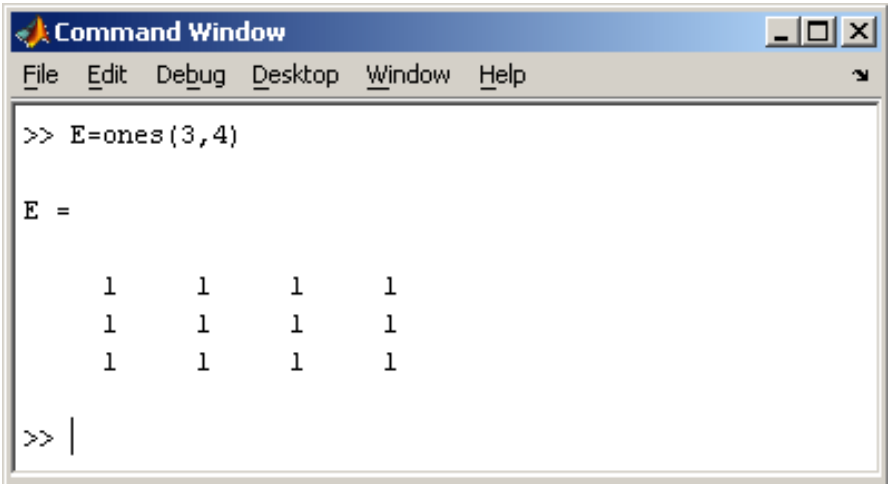


Рисунок 1.10 – Матриця з одиниць

MatLab надає можливість заповнення матриць випадковими числами. Результатом функції rand є матриця чисел, рівномірно розподілених між нулем й одиницею, а функції randn - матриця чисел, розподілених за нормальним законом з нульовим середнім й одиничною дисперсією.

Функція *diag* формує діагональну матрицю з вектора, розташовуючи елемент по діагоналі.

MatLab містить багато різних функцій для роботи з матрицями. Так, наприклад, транспонування матриці виробляється за допомогою апострофа '

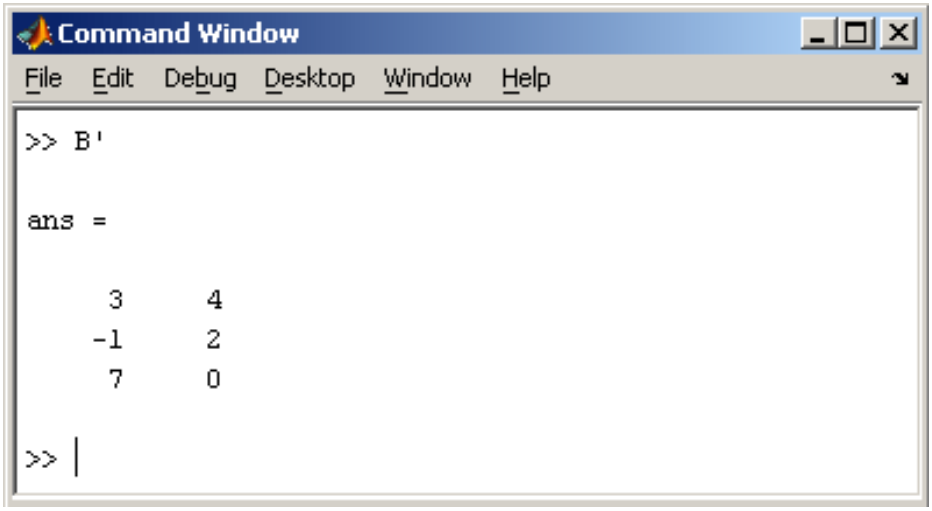


Рисунок 1.11 – Транспонування матриці

Знаходження зворотної матриці проводиться за допомогою функції inv для квадратних матриць. Псевдозворотню матрицю можна знайти за допомогою функції pinv.

Більш докладно про обробку матричних даних можна дізнатися, якщо вивести список всіх убудованих функцій обробки даних командою help datafun, а потім подивитися інформацію про потрібну функцію, наприклад help max.

Створення графіка

MatLab має широкі можливості для графічного зображення векторів і матриць, а також для створення коментарів і друкування графіків. Дамо опис декількох важливих графічних функцій.

Функція plot має різні форми, пов'язані із вхідними параметрами, наприклад plot(y) створює кусочно-лінійний графік залежності елементів y від їхніх індексів. Якщо як аргументи задані два вектори, то plot(x,y) створить графік залежності y від x. Наприклад, для побудови графіка функції sin в інтервалі від 0 до 2?, зробимо наступне

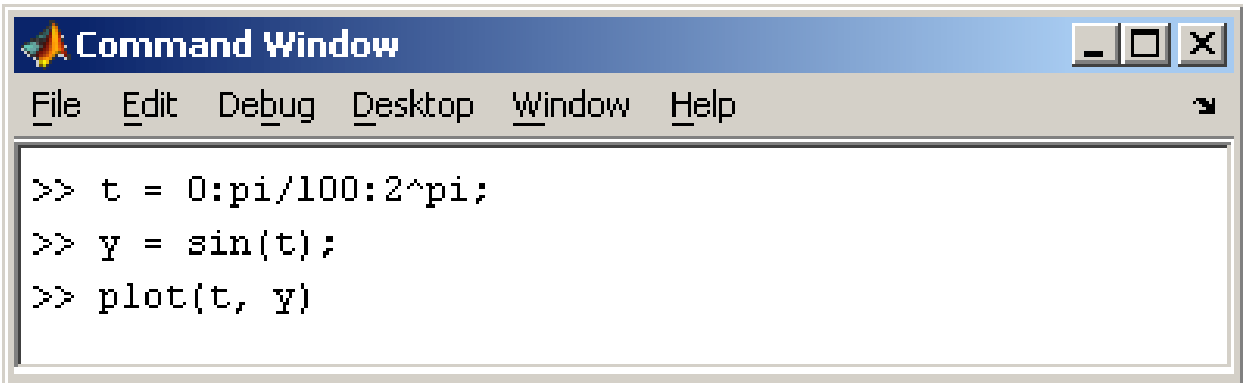


Рисунок 1.12 – Функція plot

Програма побудувала графік залежності, що відображається у вікні Figure 1

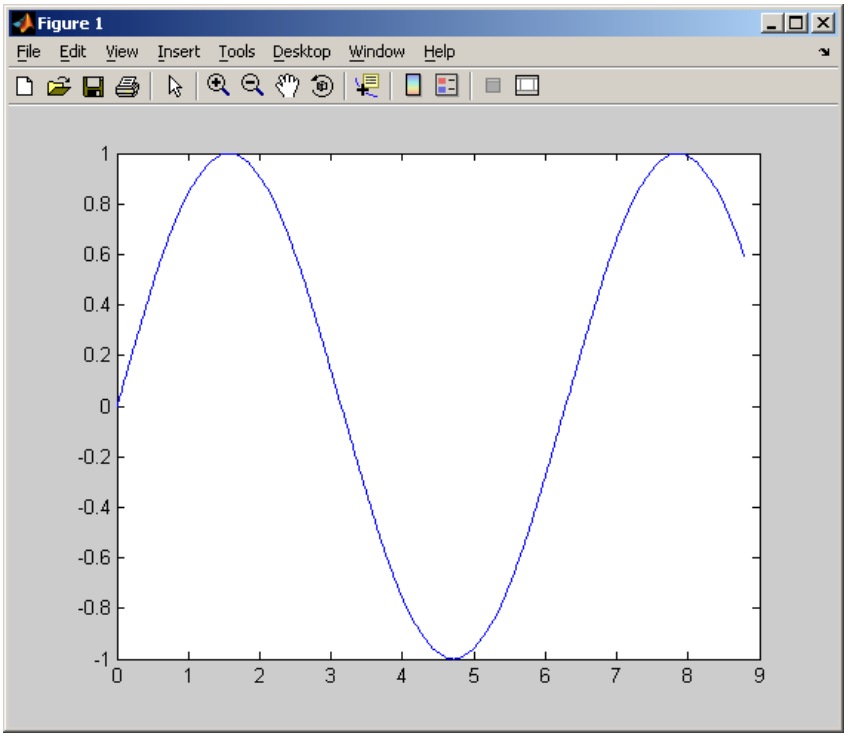


Рисунок 1.13 – Результат функції plot – графік figure(1)

MatLab автоматично привласнює кожному графікові свої кольори (крім випадків, коли це робить користувач), що дозволяє розрізняти набори даних.

Команда hold on дозволяє додавати криві на існуючий графік. Функція subplot дозволяє виводити множину графіків в одному вікні

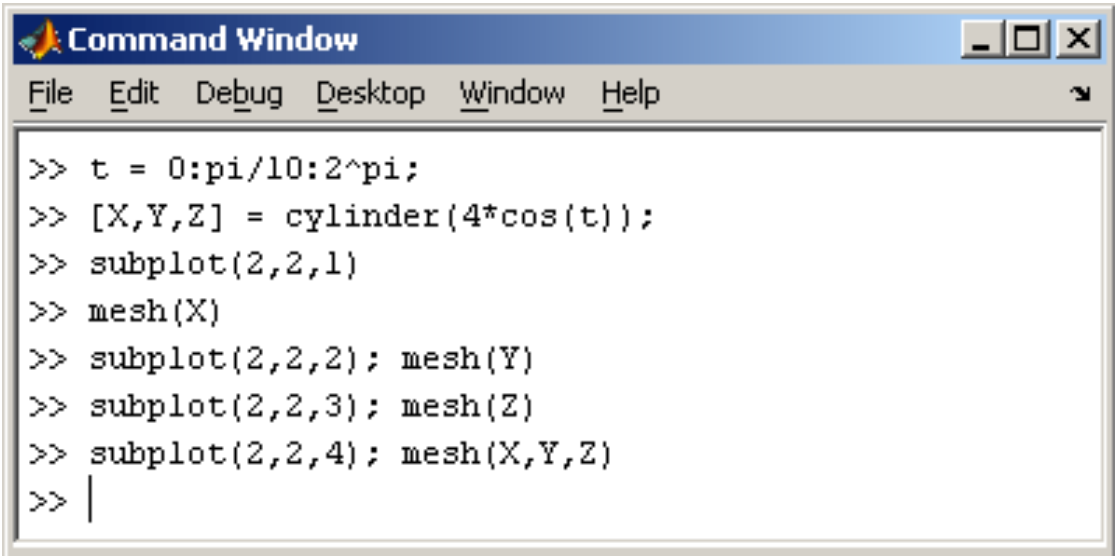


Рисунок 1.14 – Функція subplot

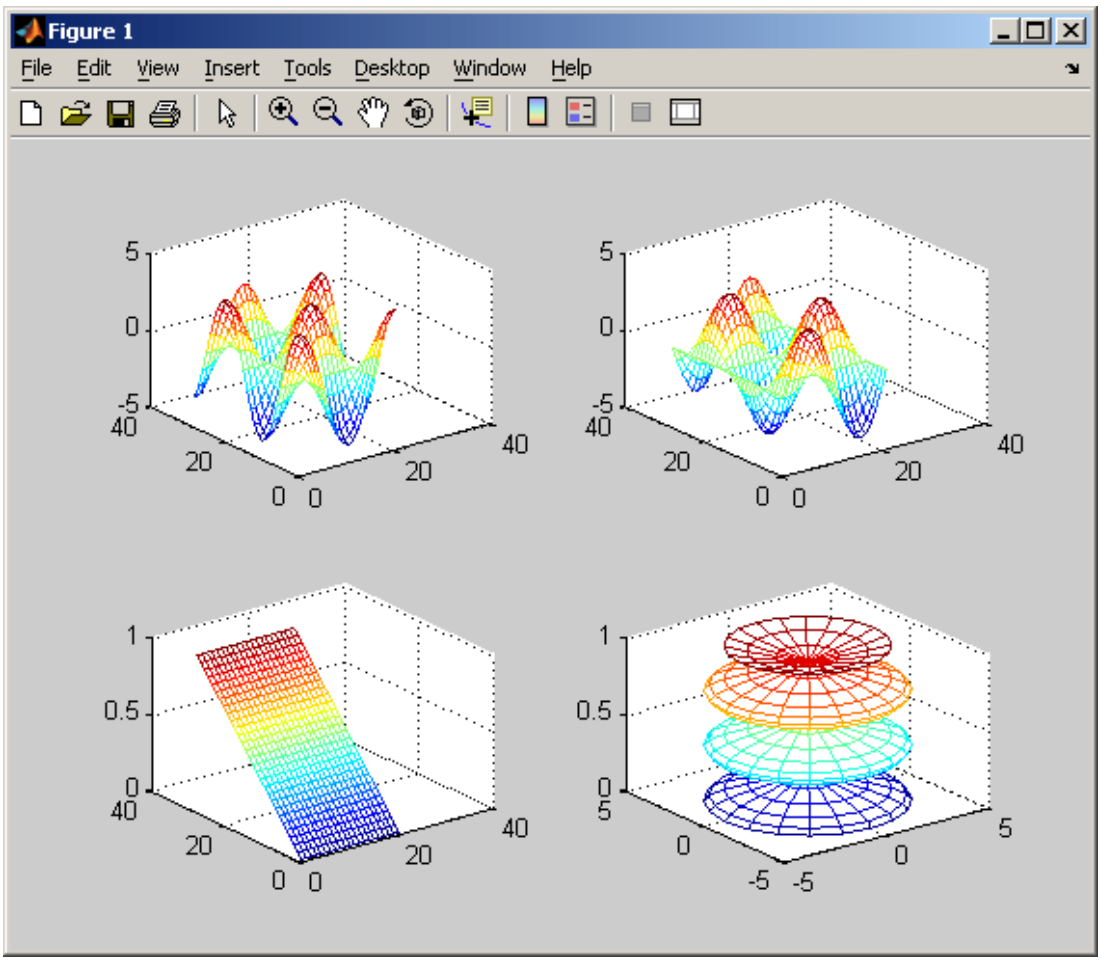


Рисунок 1.15 – Результат функції subplot

Лабораторна робота №2

Чисельне й аналітичне розв'язання звичайних рівнянь та їхніх систем

Мета: здобуття практичних навичок розв'язання звичайних рівнянь та їхніх систем за допомогою прикладного пакету Matlab

2.1 Основні теоретичні положення

Розв'язання алгебраїчних рівнянь і систем- solve

Для розв'язання систем алгебраїчних рівнянь й одиночних рівнянь служить функція**solve**:

solve (expr1, expr2,..., expr, var1, var2,..., var) – повертає значення змінних *var*, при яких дотримуються рівності, задані виразми *expr*. Якщо у виразх не використовуються знаки рівності, то покладається *expr=0.*

Розберемо докладніше Matlab розв'язання нелінійних рівнянь, наприклад квадратного рівняння x^2- 2x - 4 = 0

```
syms x;
solve ('x^2 - 2*x -4=0')
ans =5^(1/2) + 1
1 - 5^(1/2)
```

ви можете визначити більш ніж одне рівняння.

Наприклад:

```
[x, y] = solve ('x^2 - y = 2', 'y - 2*x =5')
x =  1+2*2^(1/2)
1-2*2^(1/2)
y =  7+4*2^(1/2)
7-4*2^(1/2)
```

Функція solve дозволяє вирішувати рівняння, представлені в аналітичному виді.

Вирішити рівняння

$ax^2+bx+c=0.$

Розв’язання:

```
>> S=solve('a*x^2+b*x+c=0',x)
S =
[1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

Це відомі вираз корінь $x_{1,2}$= квадратні рівняння $ax^2+bx+c=0$ через його коефіцієнти *a, b, c* (зворотна теорема Вієта). Точно також можна виразити за допомогою радикалів розв’язання кубічного рівняння $ax^3+bx^2+cx+d=0$, хоча ці вираз досить складні.

Розв’язання будь-якого трансцендентного рівняння, у тому числі й тригонометричного, досить складна й серйозна проблема для систем аналітичних обчислень [4]. Іноді просте трансцендентне рівняння може й не вирішуватися в MATLAB. Коли MATLAB не може знайти жодного розв’язання, то функція solve повертає порожню послідовність*[empty sym]*. Це означає, що або розв’язання не існує, або MATLAB не вдалося його знайти. У таких випадках, можливо, системі варто допомогти, зробивши які те нестандартні перетворення рівняння або системи, привівши їх до іншого виду.

Нехай необхідно вирішити систему рівнянь:

$$\begin{cases} ax+\frac{b}{y}=2, \\ \frac{b}{x}+ay=2ab \end{cases}$$

clear all; % очищення пам'яті

syms x y a b; % позначимо змінні

*[x,y] = solve(a*x+b/y-2, b/x+a*y-2*a*b) % уведемо команду для пошуку розв'язання системи рівнянь*

2.2 Завдання до виконання лабораторної роботи

Згідно з номером варіанту розв'язати наступні рівняння та системи рівнянь

Таблиця 2.1 – Рівняння та системи рівнянь

№	**Завдання**	
1	$x^4-x^3+5*x^2=0$	$\begin{cases} x+2y+4z=8 \\ 3x-6y+9z=3 \\ 7x+7y-3z=14 \end{cases}$
2	$x^3-x^2-5=0$	$\begin{cases} x-y+5z=3 \\ 3x-2y-7z=1 \\ 3x+4y-z=-1 \end{cases}$
3	$2x^2-5x=7$	$\begin{cases} x+2y-z=-3 \\ 2x+3y+z=-1 \\ x-y-z=3 \end{cases}$
4	$log_{x^2}\frac{4}{(7-x)}=5$	$\begin{cases} 2x+3y=5 \\ 4x+6y+z=7 \\ 5y-4z=12 \end{cases}$
5	$lg\,x=x-5$	$\begin{cases} 3x-4y-4z=5 \\ 6x-8y-2z=10 \\ x+y-5z=-4 \end{cases}$
6	$\frac{x-1}{x^2}=4$	$\begin{cases} x+3y-z=0 \\ 2x-y+3z=1 \\ 7x+7y+3z=2 \end{cases}$

7	$(x+4)^2=1$	$\begin{cases} x+3y-4z=5 \\ 2x-3y+6z=11 \\ 8x-3y+10z=21 \end{cases}$
8	$\frac{x^2-1}{x+2}=4x$	$\begin{cases} 3x+y+z=-2 \\ 5x-y-z=10 \\ x-y+5z=-12 \end{cases}$
9	$tg\ (x^2-1)=0.5$	$\begin{cases} x-2y+z=3 \\ 2x-4y+2z=5 \\ 3x-6y+3z=9 \end{cases}$
10	$\sqrt{x-1}=x$	$\begin{cases} x-4y+3z=3 \\ 2x-8y+6z=10 \\ 3x-12y+9z=15 \end{cases}$

2.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити наступні пункти:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи
3. Стислий зміст роботи
4. Результати роботи

2.4 Контрольні запитання

1. Які функції для розв’язання рівнянь Ви знаєте?
2. Назвіть синтаксис цих функцій
3. Яким чином розв’язується система рівнянь?
4. У чому полягає відмінність аналітичного розв’язання від чисельного?

Лабораторна робота №3
Розв’язання диференціальних рівнянь і систем

Мета: здобуття практичних навичок розв’язання диференціальних рівнянь та їхніх систем за допомогою прикладного пакету Matlab

3.1 Основні теоретичні положення

У деяких випадках їх можна вирішити аналітично, але частіше доводиться вирішувати чисельно. Нижче розглянемо, як це можна зробити.

Розв’язання ОДУ за допомогою функції *dsolve*

Функція *dsolve* може знаходити як загальне розв’язання ОДУ, так і частка розв’язання для заданих початкових або граничних умов. При цьому слід дотримуватися певних обмежень щодо форми запису рівняння (або системи рівнянь). Так, якщо невідома функція позначена символічної змінної y, то її похідні варто позначати як *D[n]y***,** де y квадратних дужках зазначений порядок похідної. Таким чином, похідна повинна позначатися в символьному вираженні *Dy***,** друга похідна — D2y і т.д. Функція *dsolve* може викликатися з різним набором параметрів, залежно від типу розв'язуваної задачі й порядку системи рівнянь. Деталі можна знайти в документації, ми ж обмежимося невеликим прикладом.

Нехай необхідно вирішити задачу Коші виду

$$y'' + 2y' + y = x \sin 2x, \quad y(0) = 1, y'(0) = -1. \qquad (3.1)$$

Опишемо необхідні для цієї дії.

```
eq = ’D2y + 2*Dy + y - x*sin(2*x)’; % Описуємо рівняння
x0 = 0; % Початкові умови
```

```
y0 = 1;
y10 = -1;
ic1 = sprintf('y(%d) = %d', x0, y0); % Перетворимо їх у символьні вираз
ic2 = sprintf('Dy(%d) = %d', x0, y10); % у символьні вираз
fprintf('Диференціальне рівняння: \n%s=0\n', eq);
fprintf('Початкова умова: \n%s\n%s\n', ic1, ic2);
Sol = dsolve(eq, ic1, ic2, 'x'); % Вирішуємо початкову задачу
if isempty(Sol) % Якщо рішень ні,то...
disp('Розв'язання не знайдені); % друкуємо відповідь
else % У противному випадку ...
n = length(Sol);
fprintf('Знайдене розв'язання : %d\n', n)
for i = 1:n %Друкуємо все розв'язання
fprintf('Розв'язання %d: \ny=%s\n', i, char(Sol(i)));
end
end
```

Чисельне розв'язання рівнянь, заданих символьним виразом

Чисельне розв'язання ОДУ здійснюється в MATLAB цілим набором різних функцій, які реалізують самі різні методи розв'язання. Рівняння реалізується у вигляді функції, що обчислює праву частину й цю функцію передається як аргумент так називаному «вирішувачу» ОДУ ***(ODE** solver)***.** Крім того, указується ряд параметрів і початкова умова для задачі Коші.

Для того, щоб продемонструвати всю цю процедуру, продовжимо попередній приклад і вирішимо розглянуту задачу Коші чисельно, скориставшись «вирішувачем» *ode45***,** що реалізує метод Рунге-Кутта 4-5 порядку точності. Помітимо, що чисельні методи розв'язання в більшості

випадків вимагають подання рівняння (системи рівнянь) у вигляді системи першого порядку. Це зажадає попередніх перетворень.

Код, наведений нижче, варто розуміти як продовження попереднього приклада.

```
D2y = solve(eq, 'D2y'); % Вирішуємо відносно D2y
 % Робимо заміну змінних
f2 = subs(D2y, {sym('y'), sym('Dy')}, {sym('y(1)'), sym('y(2)')});
 %Починаємо створювати вміст майбутньої функції
s{1} = 'function dydx = MyEquation(x)'; % Заголовок функції
s{2} = 'dydx = zeros(2,1);'; % Заготівля результату
s{3} = 'dydx(1) = y(2);'; % Перше рівняння системи
s{4} = ['dydx(2) = ' char(f2) ';']; % Друге рівняння системи
filename = fullfile(pwd, 'MyEquation.m'); % Визначаємо повне ім'я файлу
fid = fopen(filename, 'w'); % Відкриваємо файл
fprintf(fid, '%s\n', s{:}); % Записуємо вміст
fclose(fid); % Закриваємо файл
 % Вирішуємо рівняння чисельно
xfinish = 5; % Задаємо кінцеву крапку
xr = linspace(x0, xfinish, 20); % Створюємо масив абцис
yinit = [y0; y10]; % Задаємо початкові умови
[xx, YY] = ode45('MyEquation', xr, yinit); % Вирішуємо ОДУ
 % Малюємо спочатку розв'язання,отримане аналітично за допомогою dsolve
xp1 = linspace(xo, xfinish); % Створюємо масив координат
yp1 = subs(Sol(1), sym('x'), xp1); % для графічного розв'язання
plot(xp1, yp1, 'b'); % Малюємо графік
hold on; % Заморожуємо полотно графіків
 % Малюємо графік чисельного розв'язання
```

plot(xx, YY(:,1), 'r'); % Малюємо чисельне розв'язання по крапках

hold off; % Розморожуємо полотно графіка

title('Diff'); % Задаємо заголовок

xlabel('\itx'); % Назва осів

ylabel('\ity'); %

xlim([x0, xfinish]); % Межі по осі абцис

delete(filename); % Видаляємо непотрібний файл

Система диференціальних рівнянь

$$\begin{cases} \frac{d}{dx}y_1 = y_2 \\ \frac{d}{dx}y_2 = \left(\frac{y_1}{x} - y_2\right)\frac{1}{x} - y_1 \end{cases}$$

y1(0) = 0.1; y2(0) = 0.5

У М-файлі з ім'ям lr3.m пишемо:

function dy=lr3(x,y)

dy=zeros(2,1);

dy(1)=y(2);

dy(2)=((y(1)/x)-y(2))(1/x)-y(1);*

end

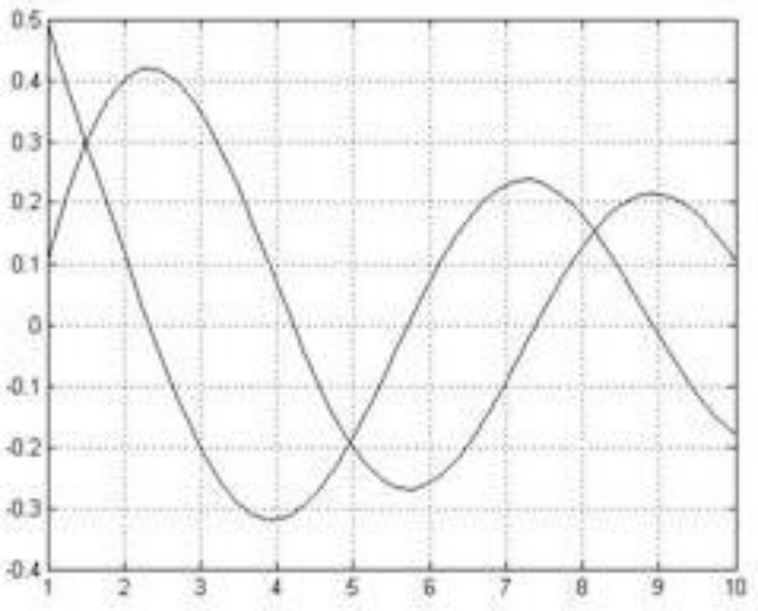
Потім у командному вікні викликаємо функцію ode45:

[x,y]=ode45(@pr8,[1 10], [0.1 0.5]);

plot(x,y,'-k')

grid;

Результатом буде графік:



3.2 Завдання до лабораторної роботи

Згідно з номером варіанту розвязати наступні диференціальні рівняння та їхні системи

Таблиця 3.1 – Рівняння та системи дифференційних рівняннь

№	Завдання	
1	$tg\,x=\sqrt{x^2+3}$	$\begin{cases} x'=-2x+4y \\ y'=-x+3y \end{cases}$
2	$x^2y'=\frac{1}{\cos y}$	$\begin{cases} 2x''=-6x+2y \\ y''=2x-2y \end{cases}$
3	$y'=x^3y^2+2x^3y$	$\begin{cases} x'=-2y \\ y'=0.5x \end{cases}$
4	$xy^2=2xy'+3$	$\begin{cases} x'=-y \\ y'=1.01x-0.2y \end{cases}$
5	$(x+1)y'=y^2\sin x+xy^2$	$\begin{cases} x''-5x+4y=0 \\ y''+4x-5y=0 \end{cases}$
6	$(x^2+e^x)y'=xy+y^2\cos x$	$\begin{cases} x''=(1-y)x \\ y'=(1-x)y \end{cases}$
7	$y'=\frac{2x+4y}{x-3y}$	$\begin{cases} x'=-2y \\ y'=2x \end{cases}$

8	$y' = tg\, ln\, sin \frac{y}{x}$	$\begin{cases} x' = 0.5y \\ y' = -8x \end{cases}$
9	$y' = cos \frac{y}{x} + \frac{x^2}{y^2}$	$\begin{cases} x' = 10y \\ y' = -10x \end{cases}$
10	$y' = \frac{\sqrt{x^4 - 2y^4}}{\sqrt[3]{x^6 + xy^5 + 2x^4y^2}}$	$\begin{cases} x' = -y \\ y' = 10x - 7y \end{cases}$

3.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити наступні пункти:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи
3. Стислий зміст роботи
4. Результати роботи

3.4 Контрольні запитання

1. Які функції для розв’язання диференціальних рівнянь Ви знаєте?
2. Назвіть синтаксис цих функцій
3. Яким чином розв’язується система диференціальних рівнянь?
4. У чому полягає відмінність аналітичного розв’язання від чисельного?

Лабораторна робота №4
Мова програмування Matlab

Мета: оволодіти навичками програмування за допомогою прикладного пакету Matlab

4.1 Основні теоретичні положення

Часто при організації циклу потрібно перебирати значення лічильника в заданому діапазоні значень і із заданим кроком зміни. Наприклад, щоб перебрати елементи вектора (масиву), потрібно організувати лічильник від 1 до N із кроком 1, де N - число елементів вектора. Щоб обчислити суму ряду, також задається лічильник від a до b з необхідним кроком зміни step. І так далі. У зв'язку з тим, що подібні задачі часто зустрічаються в практиці програмування, для їхньої реалізації був запропонований свій оператор циклу for, що дозволяє простіше й наочніше реалізовувати цикл із лічильником.

Синтаксис оператора циклу for має такий вигляд:

for <лічильник>=<початкове значення>:<крок>:<кінцеве значення>

<оператори циклу>

end

Найпростіший спосіб створити цикл - це використати вираз for. Нижче показаний простий приклад, де обчислюється й відображається

10! = 10 * 9 * 8 ... * 2 * 1

Цикл в MATLAB починається з вираз for і закінчується виразом end. Команда між цими виразами виконується в цілому дев'ять разів, по одному разі для кожного значення n від 2 до 10. Для переривання проміжного висновку усередині циклу ми використали крапку з коми. Щоб побачити

кінцевий результат, необхідно ввести f після завершення циклу. Якщо не використати крапку з коми, програма MATLAB буде відображати кожне проміжне значення 2!, 3!, і т.д.

```
f=1;
for n=2:10
f=f*n;
end
f=3628800
```

Умовний оператор if

Для того щоб мати можливість реалізувати логіку в програмі використовуються умовні оператори. Умоглядно ці оператори можна представити у вигляді вузлових пунктів, досягаючи яким програма робить вибір по якому з можливих напрямків рухатися далі. Наприклад, потрібно визначити, чи містить деяка змінна arg позитивне або негативне число й вивести відповідне повідомлення на екран. Для цього можна скористатися оператором if (якщо), що і виконує подібні перевірки.

У найпростішому випадку синтаксис даного оператора if має вигляд:

```
if    <вираз>
<оператори>
End
```

Нижче представлений приклад реалізації функції sign(), що повертає +1, якщо число більше нуля, -1 - якщо число менше нуля й 0, якщо число дорівнює нулю:

```
function my sign
x = 5;
if     x > 0
disp(1);
end
if     x < 0
disp(-1);
end
if     x == 0
disp(0);
end
```

Аналіз наведеного приклада показує, що всі ці три умови є взаємовиключними, тобто при спрацьовуванні одного з них немає необхідності перевіряти інші. Реалізація саме такої логіки дозволить збільшити швидкість виконання програми. Цього можна домогтися шляхом використання конструкції

```
if     <вираз>
<оператори1> % виконуються, якщо умова істина
else
<оператори2> % виконуються, якщо умова не істина
end
```

Тоді наведений вище приклад можна записати в такий спосіб:

```
function my_sign
x= 5;
if     x > 0
disp(1);
```

```
else
if     x < 0
disp(-1);
else
disp(0);
end
end
```

У даній програмі спочатку виконується перевірка на позитивність змінної x, і якщо це так, то на екран виводиться значення 1, а всі інші умови ігноруються. Якщо ж перша умова виявилася помилковою, то виконання програми переходить по else (інакше) на другу умову, де виконується перевірка змінної x на заперечність, і у випадку істинності умови, на екран виводиться значення -1. Якщо обоє умови виявилися помилковими, то виводиться значення 0.

Для реалізації складених умов в MatLab використовуються логічні оператори:

& -логічне И

| - логічне АБО

~ - логічне НЕ

Розглянемо приклад використання складених умов. Нехай потрібно перевірити влучення змінної x у діапазон від 0 до 2. Програма запишеться в такий спосіб:

```
function my_if
x = 1;
if     x >= 0 & x <= 2
disp('x належить діапазону від 0 до 2');
else
disp('x  не належить діапазону від 0 до 2');
```

end

Пріоритет логічних операцій наступний:

НІ (~) – найвищий пріоритет;

І (&) – середній пріоритет;

АБО (|) – найнижчий пріоритет.

Оператор циклу while

У найпростішому випадку цикл у програмі організується за допомогою оператора while, що має наступний синтаксис:

while <умова>
<оператори>
end

Тут <умова> означає умовне вираз подібне тому, що застосовується в операторі if, і цикл while працює доти, поки ця умова істинно.

Варто звернути увагу на те, що якщо умова буде помилковим до початку виконання циклу, то оператори, що входять у цикл, не будуть виконані жодного разу.

Приведемо приклад роботи циклу while для підрахунку суми ряду$\sum_{i=1}^{20} i$:

function sum_i
S = 0; *% початкове значення суми*
i=1; *% лічильник суми*
while *i <= 20* *% цикл (працює поки i <= 20)*

```
S=S+i;      % підраховується сума
i=i+1;      % збільшується лічильник на 1
end                 % кінець циклу
disp(S);            % відображення суми 210 на екрані
```

Умовний оператор switch

У деяких задачах програмування потрібно виконувати перевірку на рівність деякої змінної константним значенням. Наприклад, потрібно перетворити малі букви в заголовні. У цьому випадку необхідно зробити перевірку поточного символу з усіма можливими буквами алфавіту й при рівності з однієї з них, замінити її на заголовну. Для розв'язання таких задач зручніше користуватися умовним оператором switch, що має наступний синтаксис:

```
switch expr
case case_expr,
<оператори1>
case {case_expr1, case_expr2, case_expr3,...}
<оператори2>
...
otherwise,
<оператори>
end
```

Тут expr - змінна, значення якої перевіряється на рівність тим або іншим константам; case_expr - константи, з яким зрівнюється значення змінної; otherwise - ключове слово, для виконання операторів, при всіх помилкових умовах.

Приведемо приклад роботи даного оператора для перетворення малих букв латинського алфавіту в заголовні.

```
function upper_symbol
ch='c';
switch ch
case 'a', ch='A';
case 'b', ch='B';
case 'c', ch='C';
case 'd', ch='D';
case 'e', ch='E';
...
case 'z', ch='Z';
end
disp(ch);
```

У даній програмі задається символьна змінна ch зі значенням c. Потім, за допомогою оператора switch перевіряється її значення з усіма можливими малими буквами латинського алфавіту від a до z. Як тільки одне з умов спрацювало, оператор switch завершує свою роботу й виконання програми переходить на функцію disp(), що відображає значення змінної ch на екран.

4.2 Завдання до виконання лабораторної роботи

Створити масив даних, що повинен містити не менше 5 різних додатних та 5 різних від'ємних чисел, щонайменше 2 нулі. Далі згідно з номером варіанту сформувати наступний масив:

Таблиця 4.1 – Завдання на програмування

№ варіанту	Завдання
1	Тільки з від'ємних чисел за зростанням

2	Тільки з додатних чисел, більших за 3
3	Тільки з від’ємних чисел за зменшенням
4	Тільки з додатних чисел за зростанням
5	Тільки з додатних чисел за зменшенням
6	Тільки з від’ємних чисел більших за -1
7	Відмінних від нуля від’ємних чисел
8	Відмінних від нуля додатних чисел
9	Виключити нулі з масиву
10	Залишити у масиві один нуль та всі додатні числа

4.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити:

3 Титульний аркуш з назвою лабораторної роботи та номером варіанту

4 Мету роботи

5 Стислий зміст роботи

6 Результати роботи

4.4 Контрольні запитання

4. Які можливості програмування надає програмний пакет?
5. Назвіть та охарактеризуйте оператори циклів
6. Яким чином формується цикл програми?

Програмний пакет MATHCAD

Лабораторна робота №1

Основні арифметичні операції, стандартні функції, константи, матриці, вектори.

Мета: ознайомитися з роботою в обчислювальному пакеті MATHCAD, навчитися проводити найпростіші обчислення.

1.1 Основні теоретичні положення

Стандартні функції

Mathcad містить понад 200 убудовані функції. На стандартній панелі натисніть кнопку f(x). Ви побачите список всіх убудованих функцій. Перегляньте групи функцій (лівий список). Клацнувши мишею на кожній із груп функцій, ви побачите праворуч перелік функцій, що входять у цю групу. Познайомитися з іншими функціями можна нажавши кнопку f(x) на стандартній панелі.

Почнемо із двох груп: Log and Exponential (Логарифмічні й експонентні) і Trigonometric (Тригонометричні). Придивитеся до написання цих функцій, що не завжди збігається зі звичним математичним записом. Назви функцій можна вводити зі стандартної панелі з розкритого вікна функцій f(x), виділивши назву функції й нажавши кнопку Insert (Вставити), або набравши ім'я функції на клавіатурі в точності так, як воно записано у вікні функцій.

Уведена із клавіатури латинська буква e усередині математичного вираз означає підставу натурального логарифма e - 2,718. Це значення можна скасувати й привласнити e інше значення, використавши знак локального присвоювання :=, наприклад: e:=2.

Знак нескінченності ∞ можна вставити з математичної панелі Calculs (зі знаком інтеграла).

Часто використовуване у виразах число π можна набрати з математичної панелі Calculator (Калькулятор), де є кнопка π, або нажати аналогічну кнопку панелі грецьких букв.

Функції користувача

Зручність й ефективність розрахунків в Mathcad визначається насамперед можливістю й легкістю створення функцій користувача. При багаторазовому використанні того самого вираз без функцій користувача просто не обійтися.

Вид функції користувача:

Функції користувача мають наступний вигляд: ліворуч назва функції (з параметрами в дужках),праворуч, після оператора присвоювання :=, що обчислює вираз.

Змінні величини, що входять у праву частину, повинні бути записані в параметри після імені функції. Всі величини із правої частини, що не входять у параметри лівої частини, повинні бути задані чисельно зліва й вище функції користувача. У противному випадку Mathcad указує на помилку, офарблюючи не задану величину в червоні кольори. При виділенні функції щигликом миші з'являється текст повідомлення про помилку **This variable is not definited above** (Ця змінна не визначена раніше).

Форматування чисел

В Mathcad на результат розрахунку вплинути не можна, але можна змінити формат висновку чисел. Mathcad обчислює всі вираз з точністю 20 знаків, але виводить на екран не всі значущі цифри [7].

Установивши покажчик миші на потрібному чисельному результаті розрахунку, зробіть подвійного щиглика лівою кнопкою миші. Відкриється вікно форматування чисел **Result Format** (Формат результату), відкрите на пункті **Number Format** (Формат чисел). У цьому вікні можна вибрати наступні формати:

- **General** (Загальний) - прийнятий за замовчуванням. Числа відображаються з порядком. Кількість знаків перед комою визначається в пункті **Exponential threshold** (Поріг експоненти).
- **Scientific** (Науковий) - числа відображаються тільки з порядком: $1{,}22 \cdot 10^5$.
- **Decimal** (Десятковий) - десяткове подання чисел із плаваючої коми: 12,2564.
- **Engendering** (Інженерний) - числа відображаються тільки з порядком, кратним 3:$1{,}22 \cdot 10^6$.
- **Fraction** (Дріб) - у вигляді правильного або неправильного дробу

У дробовому форматі можна вибрати рівень точності (Level of accuracy) і змішані числа (Use fixed number).

Крім виду формату можна змінювати кількість знаків після коми (Number of decimal pieces) і поріг порядку (Exponential threshold). При перевищенні порога число відображається з порядком. Mathcad автоматично округляє числа до нуля, якщо вони менше встановленого порога.

Побудова двовимірного графіка функції

Для побудови плоского графіка функції треба:

- установити хрестоподібний курсор у те місце, де треба побудувати графік;

- на математичній панелі клацнути мишею на кнопці Graph Toolbar > X-Y Plot (Плоский графік);
- у шаблоні, що з'явився на місці курсору, плоского графіка ввести на осі абсцис ім'я аргументу, на осі ординат - ім'я функції;
- клацнути мишею поза шаблоном графіка. Графік побудований для заданого діапазону зміни аргументу.

Якщо діапазон значень аргументу не заданий, за замовчуванням графік буде побудований у діапазоні значень аргументу від -10 до 10.

Щоб на одному шаблоні розмістити трохи графіків, треба, набравши на осі ординат ім'я першої функції, нажати клавішу коми (куточок курсору при цьому обов'язково повинен перебувати наприкінці імені функції). У місці, що з'явилося, уведення (чорному квадратику) впишіть ім'я другої функції й т.д.

Якщо дві функції мають різні аргументи, наприклад, $f_1(x)$і $f_2(y)$, то на осі ординат потрібно ввести (через кому) імена обох функцій, а на осі абсцис (також через кому) - імена обох аргументів, х и в. Тоді перший графік буде побудований для першої функції по першому аргументі, а другий графік - для другої функції по другому аргументі.

Якщо функцій уведено небагато, а аргументів - 2, то графік першої функції будується по першому аргументі, а графіки інших функцій - по другому.

Якщо ввести на осях ординат й абсцис імена двох функцій одного аргументу, то буде побудований параметричний графік функції.

Для побудови тривимірного графіка

- наберіть ім'я функції двох змінних, знак присвоєння значення := і вираз функції;
- установите курсор у те місце, де ви хочете побудувати графіка;

- у математичній панелі клацніть мишею на кнопці **Graph Toolbar** (Панель графіків), що зображує графік, потім на **Surface Plot** (Графік поверхні). На місці курсору з'явиться шаблон тривимірного графіка;
- у єдиному полі уведення шаблона графіка введіть ім'я функції;
- клацніть мишею поза областю шаблона. Графік побудований.

Крім описаного способу прискореної побудови графіка поверхні по функції двох змінних існує й часто застосовується інший спосіб створення графіка поверхні — з використанням масиву чисельних значень функції. Для побудови 3-D-графіки потрібно:

- за допомогою дискретних змінних увести значення обох аргументів заданої функції;
- увести масив, елементами якого є значення функції, обчислені при заданих значеннях аргументів;
- установити курсор у те місце, де ви бажаєте побудувати графік;
- у математичній панелі клацнути мишею на кнопці, що зображує графік, і вибрати тривимірний графік. На місці курсору з'явиться шаблон тривимірного графіка;
- у єдиному полі уведення шаблона графіка ввести ім'я функції;
- клацнути мишею поза областю шаблона. Графік побудований.

Звертання до функції:

find(x, y, z...), де x,y,z — невідомі. Кількість невідомих повинне бути дорівнює кількості рівнянь. Повертає значення невідомих x, в, z, що обертає рівняння в тотожності.

Функція find може вирішувати й одне рівняння з один невідомим як окремий випадок системи рівнянь. Для системи з декількох рівнянь функція find виводить розв’язання у вигляді вектора.

Робота з векторами й матрицями

Перевага роботи з Mathcad особливо відчутно при роботі з масивами (векторами й матрицями). Як правило, ця робота складається в багаторазовому повторенні однотипних розрахунків з усіма елементами матриці.

Є три способи створення масиву чисел:

1. Заповнення шаблона матриці, що містить порожні місця уведення чисел, що підходить для уведення невеликих масивів (не більше 100 елементів).

2. Використання дискретної змінної. Цей метод підходить, коли є явна формула для обчислення елементів масиву.

3. Зчитування даних з файлів.

Розглянемо докладно перший, найбільш простий метод створення матриці:

- Встановіть курсор у те місце, де треба створити матрицю.
- Клацніть на кнопці математичної панелі **Vector and Matrix Toolbar** (Вектори й матриці), а в панелі, що з'явилася, інструментів клацніть на **кнопці Matrix or** Vector (Матриця або вектор) , як показано на мал. 4.1. Можна в головному меню Mathcad вибрати команду **Insert ► Matrix** (Вставка ► Матриця) або нажати комбінацію клавіш **Ctrl+m.**
- У діалоговому вікні, що з'явилося, впишіть число **рядків (Rows)** і число **стовпців (Columns)** , а потім клацніть на **кнопці ОК.** На місці курсору з'явиться шаблон матриці.
- У кожне місце уведення впишіть число, буквену константу або функцію. Перехід від одного місця уведення до іншого здійснюється клавішами зі стрілками. Можна також клацнути мишею на потрібнім місці уведення, але це не так зручно, як переміщення за допомогою клавіш.

Лабораторна робота №2

Чисельне й аналітичне розв’язання звичайних рівнянь та їхніх систем

Мета: здобуття практичних навичок розв’язання звичайних рівнянь та їхніх систем за допомогою прикладного пакету Mathcad

2.1 Основні теоретичні положення

Розв’язання рівнянь

Mathcad дозволяє вирішити будь-яке алгебраїчне, а також багато диференціальних й інтегральних рівнянь.

Для приклада візьмемо квадратне рівняння й знайдемо його розв’язання спочатку символьним, потім чисельним способом.

Символьне розв’язання

Для символьного розв’язання рівняння потрібно:

- набрати розв'язуване рівняння й синій куточок курсору виділити змінну, щодо якої потрібно вирішити рівняння;
- у головному меню вибрати команду Symbolics > Variable > Solve (Символьні обчислення > Змінна > Вирішити).

Недолік використання меню Symbolics полягає в тім, що знайдене розв’язання не перераховується автоматично при зміні вираз або вхідних у нього величин і не бере участь у наступних розрахунках.

Достоїнством використання меню Symbolics є те, що раніше прийняті чисельні значення величин не враховуються в символьних розрахунках.[8]

Якщо виділене вираз не має символьного розв’язання (а більшість рівнянь не має символьного розв’язання), то Mathcad повідомляє про помилку: «No solution was found» («Розв’язання не знайдене»).

$$a \cdot x^2 + b \cdot x + c = 0$$

$$\begin{pmatrix} -\dfrac{\dfrac{b}{2} + \dfrac{\sqrt{b^2 - 4 \cdot a \cdot c}}{2}}{a} \\ -\dfrac{\dfrac{b}{2} - \dfrac{\sqrt{b^2 - 4 \cdot a \cdot c}}{2}}{a} \end{pmatrix}$$

В Mathcad є й інші можливості символьного розв’язання рівнянь, наприклад, з використанням функції root.

Чисельне розв’язання (функція root, solve)

Розглянемо одну з ряду функцій, що дозволяють вирішувати алгебраїчні рівняння, - функцію root.

Звертання до функції:

root(f(x), x),

де f(x) = 0. Повертає значення x, при якому функція f(x) = 0.

Функція root вирішує рівняння ітераційним методом січних і тому вимагає завдання перед собою початкових значень. Крім того, функція root, виконуючи обчислення методом спуска, знаходить і виводить тільки один корінь, найближчий до початкового наближення.

Перш ніж вирішувати рівняння, бажано побудувати графік функції f(x). На графіку видно, чи перетинає крива f(x) вісь абсцис, тобто чи має дійсні коріння.[8] Якщо крапки перетинання кривій з віссю є, потрібно вибирати початкове наближення ближче до значення кореня. Якщо корінь

трохи, для знаходження кожного кореня треба задавати своє початкове наближення.

$$f(x) := x^2 - 2x - 4 \text{ solve}, x \rightarrow \begin{pmatrix} \sqrt{5} + 1 \\ 1 - \sqrt{5} \end{pmatrix}$$

В Mathcad системи рівнянь вирішуються за допомогою обчислювального блоку Given-find. Тому що системи рівнянь вирішуються ітераційним методом, перед розв'язанням необхідно задати початкові наближення для всіх невідомих.

Щоб вирішити систему алгебраїчних рівнянь, потрібно:

- задати початкові наближення для всіх невідомих, що входять у систему;
- надрукувати ключове слово Given (Дане). Переконаєтеся, що при печатці ви не перебуваєте в текстовій області. Якщо нажати клавішу пробілу, то математичне вираз стає текстовою областю й слово Given перестає сприйматися як ключове;
- увести рівняння й нерівності, що входять у систему, лівіше й нижче ключового слова Given. Між лівою й правою частинами рівняння повинен стояти знак рівності. Це не знак присвоєння значення, а знак логічної рівності. Для його уведення використайте комбінацію клавіш Ctrl+= або виберіть його на панелі Boolean;
- уведіть будь-яке вираз, що містить функцію find. При друкуванні слів Given й find можна використати будь-який шрифт, прописні й малі літери.

Given

$x^2 - y = 2$

$y - 2 \cdot x = 5$

$$\text{Find}(x,y) \rightarrow \begin{pmatrix} 2\cdot\sqrt{2} + 1 & 1 - 2\cdot\sqrt{2} \\ 4\cdot\sqrt{2} + 7 & 7 - 4\cdot\sqrt{2} \end{pmatrix}$$

Given

$$a \cdot x + \frac{b}{y} = 2$$

$$\frac{b}{x} + a \cdot y = 2 \cdot a \cdot b$$

$$\text{Find}(x,y) \rightarrow \begin{pmatrix} \frac{1}{a} \\ b \end{pmatrix}$$

2.2 Завдання до виконання лабораторної роботи

Згідно з номером варіанту розв’язати наступні рівняння та системи рівнянь

Таблиця 2.1 – Рівняння та системи рівняннь

№	**Завдання**	
1	$x^4-x^3+5*x^2=0$	$\begin{cases} x+2y+4z=8 \\ 3x-6y+9z=3 \\ 7x+7y-3z=14 \end{cases}$
2	$x^3-x^2-5=0$	$\begin{cases} x-y+5z=3 \\ 3x-2y-7z=1 \\ 3x+4y-z=-1 \end{cases}$
3	$2x^2-5x=7$	$\begin{cases} x+2y-z=-3 \\ 2x+3y+z=-1 \\ x-y-z=3 \end{cases}$
4	$log_{x^2}\frac{4}{(7-x)}=5$	$\begin{cases} 2x+3y=5 \\ 4x+6y+z=7 \\ 5y-4z=12 \end{cases}$
5	$lg\,x=x-5$	$\begin{cases} 3x-4y-4z=5 \\ 6x-8y-2z=10 \\ x+y-5z=-4 \end{cases}$
6	$\frac{x-1}{x^2}=4$	$\begin{cases} x+3y-z=0 \\ 2x-y+3z=1 \\ 7x+7y+3z=2 \end{cases}$
7	$(x+4)^2=1$	$\begin{cases} x+3y-4z=5 \\ 2x-3y+6z=11 \\ 8x-3y+10z=21 \end{cases}$
8	$\frac{x^2-1}{x+2}=4x$	$\begin{cases} 3x+y+z=-2 \\ 5x-y-z=10 \\ x-y+5z=-12 \end{cases}$
9	$tg\ (x^2-1)=0.5$	$\begin{cases} x-2y+z=3 \\ 2x-4y+2z=5 \\ 3x-6y+3z=9 \end{cases}$
10	$\sqrt{x-1}=x$	$\begin{cases} x-4y+3z=3 \\ 2x-8y+6z=10 \\ 3x-12y+9z=15 \end{cases}$

2.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити наступні пункти:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи
3. Стислий зміст роботи
4. Результати роботи

2.4 Контрольні запитання

1. Які функції для розв’язання рівнянь Ви знаєте?
2. Назвіть синтаксис цих функцій
3. Яким чином розв’язується система рівнянь?
4. У чому полягає відмінність аналітичного розв’язання від чисельного?

Лабораторна робота №3

Розв’язання диференціальних рівнянь і систем

Мета: здобуття практичних навичок розв’язання диференціальних рівнянь та їхніх систем за допомогою прикладного пакету Mathcad

3.1 Основні теоретичні положення

Диференціальне рівняння називається звичайним (ОДУ), якщо в нього входять похідні тільки за однією змінною. У противному випадку говорять про рівняння в частинних похідних. Mathcad надає більші можливості для розв’язання ОДУ й дуже обмежені - для розв’язання рівнянь у частинних похідних.[8]

Mathcad вирішує ОДУ двох типів:

- задачі Коші - ОДУ з початковими умовами, у яких задаються значення функції і її похідних у початковій крапці інтервалу інтегрування;
- крайові задачі - ОДУ із граничними умовами, де задаються значення функції і її похідних на початку й наприкінці інтервалу інтегрування.

Для чисельного інтегрування одну ОДУ (так само як і системи ОДУ) можна використати обчислювальний блок Given-Odesolve

Застосування функції Odesolve вимагає запису обчислювального блоку, що складає із трьох частин:

- ключового слова Given (Дане);
- диференціального рівняння й початкових або граничних умов до нього;
- функції Odesolve(x, xk,n)(розв’язання ОДУ), де x - ім'я змінної, щодо якої вирішується рівняння; xk - кінець інтервалу інтегрування (початок інтервалу інтегрування зазначено раніше, у початкових умовах);

n - необов’язковий внутрішній параметр, що визначає число інтервалів, використовуваних для інтерполяції функції розв’язання рівняння.

Given

$$\frac{d}{dt}y(t) = y(t) - y(t)^2$$

$$y(0) = 0.1$$

$$y := Odesolve(t,10)$$

Вбудовані функції для розв’язання систем ОДУ

В Mathcad є три убудовані функції, які дозволяють вирішувати поставлену задачу Коші різними чисельними методами.

- rkfixed(y0, t0, t1, M, D) - метод Рунге-Кутта з фіксованим кроком,
- Rkadapt(y0, t0, t1, M, D) - метод Рунге-Кутта зі змінним кроком;
- Buistoer(y0, t0, t1, M, D) - метод Булирша-Штера;
- y0 - вектор початкових значень у крапці to розміру NXI;
- t0 - початкова точка розрахунку,
- t1 - кінцева точка розрахунку,
- M - число кроків, на яких чисельний метод знаходить розв’язання;
- D - векторна функція розміру NXI двох аргументів - скалярного t і векторного в При цьому в - шукана векторна функція аргументу t того ж розміру NXI.

Дотримуйтеся регістра першої букви розглянутих функцій, оскільки це впливає на вибір алгоритму рахунку, на відміну від багатьох інших убудованих функцій Mathcad, наприклад Find=fmd

Кожна з наведених функцій видає розв’язання у вигляді матриці розміру (M+1)x(N+1). У її лівому стовпці перебувають значення аргументу t,

що ділять інтервал на рівномірні кроки, а в інших N стовпцях - значення шуканих функцій y0(t) ,y1(t), .. ,yN-1(t), розраховані для цих значень аргументу Оскільки всього крапок (крім початкової) м, те рядків у матриці розв'язання буде всього M+1

$$D(t,y) := \begin{pmatrix} y_1 \\ -y_0 - 0.1\,y_1 \end{pmatrix}$$

$$y0 := \begin{pmatrix} 0.1 \\ 0 \end{pmatrix}$$

$$M := 100$$

$$u := \text{rkfixed}(y0, 0, 50, M, D)$$

3.2 Завдання до лабораторної роботи

Згідно з номером варіанту розвязати наступні диференціальні рівняння та їхні системи

Таблиця 3.1 – Рівняння та системи дифференційних рівняннь

№	**Завдання**	
1	$tg\,x=\sqrt{x^2+3}$	$\begin{cases} x'=-2x+4y \\ y'=-x+3y \end{cases}$
2	$x^2y'=\frac{1}{\cos y}$	$\begin{cases} 2x''=-6x+2y \\ y''=2x-2y \end{cases}$
3	$y'=x^3y^2+2x^3y$	$\begin{cases} x'=-2y \\ y'=0.5x \end{cases}$
4	$xy^2=2xy'+3$	$\begin{cases} x'=-y \\ y'=1.01x-0.2y \end{cases}$
5	$(x+1)y'=y^2\sin x+xy^2$	$\begin{cases} x''-5x+4y=0 \\ y''+4x-5y=0 \end{cases}$
6	$(x^2+e^x)y'=xy+y^2\cos x$	$\begin{cases} x''=(1-y)x \\ y'=(1-x)y \end{cases}$
7	$y'=\frac{2x+4y}{x-3y}$	$\begin{cases} x'=-2y \\ y'=2x \end{cases}$
8	$y'=tg\ln\sin\frac{y}{x}$	$\begin{cases} x'=0.5y \\ y'=-8x \end{cases}$

9	$y' = cos\frac{y}{x} + \frac{x^2}{y^2}$	$\begin{cases} x' = 10y \\ y' = -10x \end{cases}$
10	$y' = \frac{\sqrt{x^4 - 2y^4}}{\sqrt[3]{x^6 + xy^5 + 2x^4y^2}}$	$\begin{cases} x' = -y \\ y' = 10x - 7y \end{cases}$

3.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити наступні пункти:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи
3. Стислий зміст роботи
4. Результати роботи

3.4 Контрольні запитання

1. Які функції для розв’язання диференціальних рівнянь Ви знаєте?
2. Назвіть синтаксис цих функцій
3. Яким чином розв’язується система диференціальних рівнянь?
4. У чому полягає відмінність аналітичного розв’язання від чисельного?

Лабораторна робота №4
Мова програмування Mathcad

Мета: оволодіти навичками програмування за допомогою прикладного пакету Mathcad

4.1 Основні теоретичні положення

Створення програм

Програма Mathcad – це окремий випадок вираз Mathcad. Подібно будь-якому вираженню, програма повертає значення, якщо за нею треба знак рівності. Звичайне вираз Mathcad складається з одного рядка. Вираз-програма містить багато рядків. Фактично це складений вираз.

Вираз-програма складається з назви виразу, що випливає за ним знака присвоєння значення й необхідних виражень у правій частині, записаних у стовпчик й об'єднаних ліворуч вертикальною рисою.

Уведення рядків у програму

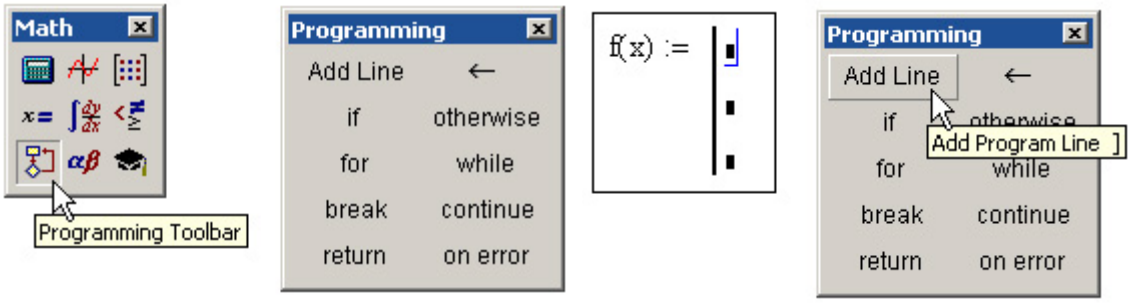
Для створення програми треба:

- увести ім'я вираз-програми;
- увести оператор присвоювання (:=);
- клацнути на кнопці панелі програмування **Add Line** (Додати рядок) стільки разів, скільки рядків повинна містити програма;
- у місця, що з'явилися, уведення ввести потрібні оператори, зайві місця уведення видалити.

Щоб створити відсутні місця уведення, треба поставити синій куточок курсору в кінець рядка, після якої ввести новий рядок. Клавішею пробілу варто виділити повністю весь рядок і нажати кнопку **Add Line.** При цьому можливі два варіанти:

- Якщо синій куточок курсору перебуває на початку рядка, то після натискання **Add Line** місце уведення з'явиться вище цього рядка.

- Якщо синій куточок курсору перебуває наприкінці рядка, то після натискання **Add Line** місце уведення з'явиться нижче цього рядка



Умовний оператор if

Дія умовного оператора складається із двох частин. Спочатку перевіряється умова, записане праворуч від оператора **if.** Якщо воно вірно, виконується вираз, що коштує ліворуч від **if,** якщо не вірно, відбувається перехід до наступного рядка програми.

Щоб вставити умовний оператор у програму, потрібно:

- у створюваній програмі встановити курсор на вільне місце уведення, де повинен з'явитися умовний оператор;
- на панелі програмування **(Programming Toolbar)** клацнути на кнопці **if.** У програмі з'явиться шаблон оператора із двома місцями уведення;
- у праве місце уведення ввести умову. Користуйтеся при цьому логічними операторами, уводячи їх з панелі **Boolean**;
- ліворуч від оператора **if** увести вираз, що повинне виконуватися, якщо умова вірно;
- якщо при виконанні умови повинне виконуватися відразу кілька виразів, треба мати кілька місць уведення. Установите курсор на місце уведення ліворуч від **if** і натисніть **Add Line** стільки разів, скільки рядків треба ввести. Зверніть увагу на те, що при цьому змінюється вид умовного

оператора. Стовпчик місць уведення з'являється не ліворуч, а під оператором **if**

Якщо умова не вірно, відбувається перехід до наступного рядка програми. Вона може містити нову умову або бути звичайним виразм.

Часто зустрічається умова із двома варіантами дії: **якщо** те **інакше ....** Для слова «інакше» на панелі **Programming** є оператор **otherwise** (Інакше), що вводиться так само, як **if.**

На практиці краще вводити умова в трохи іншому порядку:

- На вільне місце уведення в програмі вводять вираз, виконуване, якщо умова вірно.
- Виділяють курсором вираз так, щоб куточок курсору був наприкінці вираз, і на панелі програмування натискають if.
- Праворуч від оператора if уводять умову.
- На наступному вільному місці уведення (на наступному рядку) пишуть вираз, виконуване «інакше», якщо умова не виконується.
- Виділяють курсором вираз так, щоб куточок курсору був наприкінці вираз, і на панелі програмування натискають otherwise.

$$f(x) := \begin{cases} \text{"negative"} & \text{if } x < 0 \\ \text{"positive"} & \text{if } x > 0 \\ \text{"zero"} & \text{otherwise} \end{cases}$$

$f(1) = \text{"positive"}$

$f(-1) = \text{"negative"}$

$f(0) = \text{"zero"}$

Оператори циклу

Найважливішим елементом програмування, крім умовного оператора, є оператор циклу. У звичайному Mathcad-документі використання дискретної змінної фактично аналогічно застосуванню оператора циклу, використовуваного для обчислення одного виразу.

Mathcad обчислює вираз зверху вниз і переходить до наступного виразу, лише завершивши всі обчислення попереднього вираз, оскільки повернутися до нього вже не зможе. Якщо ж у кожному циклі повинне бути обчислене кілька виразів, необхідно становити програму.

Панель програмування Mathcad містить два операторів циклу, for й while:

- Якщо число виконань циклу заздалегідь відомо, використається оператор for.
- Якщо цикл повинен завершитися по виконанні деякої умови й момент виконання цієї умови невідомий, використається оператор while.

Оператор while

Цикл while виконується доти, поки залишається щирим умова циклу, тому немає необхідності знати число обчислень заздалегідь. Важливо тільки, щоб де-небудь усередині циклу або в іншій виконуваній ділянці програми був присутній оператор, що робить умова циклу помилковим. У противному випадку цикл буде виконуватися нескінченно.

Якщо виконувана програма зациклилася, її можна зупинити, нажавши клавішу **Esc.**

Щоб записати цикл while, виконайте наступні дії:

- Встановіть курсор на вільне місце уведення в програмі (праворуч від вертикальної риси).
- На панелі програмування натисніть кнопку while. З'явиться шаблон із двома місцями уведення.
- Праворуч від слова «while» уведіть умову виконання циклу. Звичайно це логічне вираз.
- У поле, що залишилося, уведення (унизу під словом «while») уведіть вираз, що обчислюється у циклі.

- Якщо в циклі треба обчислювати кілька виражень, то спочатку встановите курсор на місце уведення й натисніть кнопку Add Line (або клавішу]) стільки разів, скільки рядків буде містити цикл. Потім заповните всі місця уведення, увівши потрібні вираз. Видалите зайві місця уведення.
- Виявивши заголовок циклу while, Mathcad перевіряє умова циклу. Якщо воно щиро, то Mathcad виконує тіло циклу й знову перевіряє умова. Якщо воно помилково, Mathcad закінчує виконання циклу.

```
x := | z ← 0
     | while z < 10
     |   z ← z + 1

x = 10
```

Оператор for

У циклі for число повторень циклу визначається змінною, що задається на початку циклу. Розглянемо створення такого циклу:

- Установите курсор на вільне місце уведення в програмі (праворуч від вертикальної риси).
- На панелі програмування натисніть кнопку for Loop (Цикл for). З'явиться шаблон із трьома місцями уведення.
- Праворуч від слова «for» уведіть ім'я змінної циклу. Після знака є введіть діапазон зміни змінної циклу так само, як це робиться за допомогою дискретної змінної. Змінної циклу може бути ряд чисел, або вектор, або список скалярів, діапазонів, векторів, розділених коми (мал. 9.6).
- У поле, що залишилося, уведення (унизу, під словом «for») уведіть вираз, що обчислюється в циклі.
- Якщо в циклі треба обчислювати кілька виражень, то спочатку встановите курсор на місце уведення й натисніть кнопку Add Line (або

клавішу]) стільки разів, скільки рядків буде містити цикл. Потім заповните всі місця уведення, увівши потрібні вираз. Видалите зайві місця уведення.

```
x := | z ← 0                x := | z ← 0
     | for i ∈ 0..5               | for i ∈ (1  2  3)
     |   z ← z + i               |   z ← z + i

x = 15                     x = 6
```

Оператори break, continue, return

Ці оператори використовуються для керування роботою циклів і всієї програми в цілому:

- continue повертає розрахунок до початку циклу;
- break забезпечує вихід із циклу й продовження роботи програми;
- return забезпечує вихід із програми.

```
x := | z ← 0
     | for i ∈ 0..5
     |   | z ← z + i
     |   | (break) if i = 2

x = 3
```

4.2 Завдання до виконання лабораторної роботи

Створити масив даних, що повинен містити не менше 5 різних додатних та 5 різних від’ємних чисел, щонайменше 2 нулі. Далі згідно з номером варіанту сформувати наступний масив:

Таблиця 4.1 – Завдання на програмування

№ варіанту	Завдання
1	Тільки з від’ємних чисел за зростанням
2	Тільки з додатних чисел, більших за 3
3	Тільки з від’ємних чисел за зменшенням

4	Тільки з додатних чисел за зростанням
5	Тільки з додатних чисел за зменшенням
6	Тільки з від’ємних чисел більших за -1
7	Відмінних від нуля від’ємних чисел
8	Відмінних від нуля додатних чисел
9	Виключити нулі з масиву
10	Залишити у масиві один нуль та всі додатні числа

4.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи
3. Стислий зміст роботи
4. Результати роботи

4.4 Контрольні запитання

1. Які можливості програмування надає програмний пакет?
2. Назвіть та охарактеризуйте оператори циклів
3. Яким чином формується цикл програми?

Програмний пакет Maple

Лабораторна робота №1

Основні арифметичні операції, стандартні функції, константи, матриці, вектори.

Мета: ознайомитися з роботою в обчислювальному пакеті Maple, навчитися проводити найпростіші обчислення.

1.1 Основні теоретичні положення

Числа

Maple працює із числами наступного типу:

- цілими десятковими (0, 1, 123, -456 і т.д.),
- раціональними у вигляді відношення цілих чисел (7/9, -123/127),
- радикалами,
- речовинними з мантисою й порядком (1.23E5, 123.456E-10)
- комплексними (2+3*I)

Цілі числа задаються у вигляді послідовності цифр від 0 до 9.

Maple може працювати із цілими числами довільної величини, кількість цифр практично обмежено числом 228. Більші числа, які не містяться на одному рядку області висновку, Maple переносить на наступний рядок, використовуючи символ зворотного слеша (\).

Крім стандартних арифметичних операцій, до яких выдносяться додавання, вирахування, множення, ділення й обчислення факторіала (!), Maple пропонує досить великий набір команд, що дозволяє виконати дії, специфічні при обробці цілих чисел.

Одержати список всіх команд для роботи із цілими числами можна, набравши команду ?integer:

Приведемо деякі із цих команд:

- ifactor розкладання на прості множники

- iquo обчислення частки при операції цілого ділення
- irem обчислення остачі при операції цілого ділення
- igcd знаходження найбільшого загального дільника двох цілих чисел
- isprime перевірка, чи є ціле число простим

Звичайні дроби задаються за допомогою операції ділення двох цілих чисел.

```
> 4+8/2;
                    8
> evalf(7/4,25);
1.750000000000\
   000000000000
```

Maple автоматично робить скорочення дробів. Над звичайними дробами можна виконувати всі основні арифметичні операції. Якщо при завданні дробу її знаменник скорочується, то така «дріб» трактується програмою Maple як ціле число. Для перетворення звичайного дробу в десяткову служить команда evalf( ). Другий параметр цієї команди задає число значущих цифр [8]. Помітимо, що десяткове подання всього лише апроксимація точної величини, представленою звичайним дробом, тобто дріб й її десяткове подання не є ідентичними об'єктами Maple.

Радикали задаються як результат піднесення в дробовий ступінь цілих або дробових чисел, або обчислення з них же квадратного кореня функцією sqrt( ), або кореня n-ої ступеня функцією surd(число, n).

```
> (3/4)^(2/3);
                  (2/3)  (1/3)
                 3      4
> sqrt(66/9);
                    1/2
                   1/3 66
> surd(75/3,4);
                    1/2
                   5
```

В Maple піднесення в ступінь задається символом (^) або послідовністю двох зірочок (**). При завданні радикалів також виробляються всілякі спрощення, пов'язані з винесенням з-під знака радикала максимально можливої величини.

Обчислення із цілими, дробами й радикалами є абсолютно точними, тому що при роботі із цими об'єктами Maple не робить ніяких округлень на відміну від чисел із плаваючою крапкою.

Числа із плаваючою крапкою задаються у вигляді цілої й дробової частин, розділених десятковою крапкою. Їх можна представити також, використовуючи так називану експонентну форму запису (для вказівки порядку застосовується символ e або E).

```
> 3^5*0.1;
                  24.3
> 3^5*(1/10);
                  243
                  ---
                   10
> 2e1+4/5+sqrt(3)*4/5*0.1+
  surd(5,3);
                                  1
                                  -
                                  3
  20.80000000 + .08000000000 sqrt(3) + 5
```

Якщо у вираженні присутнє число із плаваючою крапкою, то результатом обчислення такого "змішаного" вираз буде число із плаваючою крапкою. Якщо у вираженні присутнє радикал, то він обчислюється точно, а коефіцієнт при ньому обчислюється або точно, або у вигляді числа із плаваючою крапкою залежно від виду співмножників.

Комплексні числа. Для мнимої одиниці в Maple використається константа I. Завдання комплексного числа нічим не відрізняється від його звичайного завдання в математику (наприклад 2+3*I або d+k*I).

```
> (2/5+3*I)+(4+1/2*I);
```

$$\frac{22}{5}+\frac{7}{2}I$$

```
> (2/5+3*I)+(4.+1/2*I);
```

$$4.400000000+3.500000000\,I$$

```
> Re(2+3*I);
```

$$2$$

```
> conjugate(2+3*I);
```

$$2-3\,I$$

```
> argument(%);
```

$$-\arctan\left(\frac{3}{2}\right)$$

Якщо хоча б одна із частин комплексного числа (дійсна або мнима) обчислюються у вигляді числа із плаваючою крапкою, то й результат буде таким же.

Деякі команди для роботи з комплексними числами:

- Re( ) виділення дійсної частини комплексного числа
- Im( ) виділення мнимої частини комплексного числа
- argument( ) обчислення аргументу комплексного числа
- conjugate( ) побудова комплексно-комплексно-сполученого числа

Константи

Maple містить цілий ряд визначених іменованих констант - таких, до значень яких можна звертатися по імені. Частина цих констант не може бути змінена. До них відносяться:

Таблиця 1.1 – Основні константи Maple

Позначення	**Розшифровка**
False	- логічне значення "не істинно";
Gamma	- константа Эйлера (0.5772156649..);
Infinity	- позитивна нескінченність;
True	- логічне значення "істинно";

Catalan	- константа Каталана (0.915965594..);
FAIL	- спеціальна константа, використається як третє значення при обчисленні функцій тризначної логіки;
I	- мнима одиниця;
Pi	- константа p=3.141..

Число e задається як exp(1)

```
> evalf(Pi);
                3.141592654
> Digits:=30;
                Digits := 30
> evalf(Pi);
    3.14159265358979323846264338328
```

Константи, значення яких можуть бути перевизначені:

- Digits-задає число значущих цифр для чисел із плаваючою крапкою (за замовчуванням 10);
- Order -визначає кількість членів у розкладанні функції в ряд Тейлора (за замовчуванням 6);

Подивитися всі константи, певні в Maple, можна, виконавши команду: ?ininame. Крім перерахованих на сторінці Справки констант всі змінні, імена яких починаються з_Env , за замовчуванням є системними константами Maple.

Масиви

Масив є подальшим розвитком концепції списку. Якщо список можна мислити як перенумеровану послідовність, індекси якої можуть приймати тільки позитивні цілі значення (нумерація обов'язково починається з одиниці), то в масиві кожен елемент також пов'язаний з індексом, однак не обмежений однією розмірністю (масив може мати багато

розмірностей, кожну зі своїм індексом), причому індекси можуть приймати поряд із цілими позитивними й негативні значення, і нуль.

Створення масивів.. Створити масив в Maple можна декількома способами. Найпростіший з них - використання функції array( ) стандартної бібліотеки. Синтаксис створення масиву наступний:

array(границі, список, опції);

Параметр границі представляє діапазон(ы) зміни індексу(ов) масиву (для багатомірного масиву діапазони задаються через кому).

```
> A:=array(1..2,identity);
```

$$A := \mathrm{array}(identity, 1 .. 2, [\,])$$

```
> eval(A);
```

$$[1, 1]$$

```
> M:=array(-1..0,-2..-1,
  [[a,a],[b,b]]);
```

$$M := \mathrm{array}(-1 .. 0, -2 .. -1, [\,(-1, -2) = a,\ (-1, -1) = a,\ (0, -2) = b,\ (0, -1) = b\,])$$

Список задає значення елементів масиву, причому для двовимірного масиву елементами списку є списки, тобто цей параметр є списком списків.

Параметр опції використається для завдання масивів спеціального виду. Цей параметр може приймати наступні значення: symmetric, antisymmetric, sparse, identity, diagonal , для завдання , відповідно, симетричних, антисиметричних, розріджених, одиничних і діагональних масивів.

Всі параметри цієї функції є необов'язковими, однак, або параметр границі, або список значень повинен обов'язково бути присутнім.

```
> V:=vector(3,[ff,gg,hh^2]);
```

$$V := [ff, gg, hh^2]$$

```
> M:=matrix(2,2,[x,x^2,2*x,3*
  z]);
```

$$M := \begin{bmatrix} x & x^2 \\ 2x & 3z \end{bmatrix}$$

Для завдання вектора й матриці можна використати відповідно команди vector( ) і matrx( ). Синтаксис цих команд:

vector(n,[елемент1, елемент2, ...]);

matrx(n, m, [елемент1, елемент2, ...])

Тут цілі величини n й m задають розмірності вектора й матриці, а значення їхніх елементів задаються у вигляді простого списку.

```
> M1:=array(1..4);
```

$$M1 := \text{array}(1..4, [\,])$$

```
> M1[1]:=x;M1[2]:=y;
  M1[3]:=z;M1[4]:=t;
```

$$M1_1 := x$$
$$M1_2 := y$$
$$M1_3 := z$$
$$M1_4 := t$$

```
> M1;
```

$$M1$$

```
> eval(M1);
```

$$[x, y, z, t]$$

```
> M2:=array(1..4,[a,b,c,d]);
```

$$M2 := [a, b, c, d]$$

```
> M2;
```

$$M2$$

```
> print(M2);
```

$$[a, b, c, d]$$

Значення елементів вектора або матриці не обов'язково задавати при створенні цих об'єктів. Можна пізніше за допомогою індексного посилання на елементи вектора або матриці (у квадратних дужках) привласнити їм значення.

Для відображення вмісту вектора або матриці використається команда print(V) або eval(V), тому що змінні, утримуючі складні об'єкти, обчислюється не повністю, а тільки до свого імені.

Для відображення результатів ОБЧИСЛЕНЬ із матрицями й векторами служить команда evalm(V). Ця команда відображає результат обчислень на рівні елементів. Над матрицями й векторами (без підключення спеціальних пакетів linalg й LinearAlgebra) можливі будь-які дії, задані операторами додавання (+), вирахування (-), некомутативного множення (&*), ділення (/) і піднесення в ступінь (^).

Пакети linalg й LinearAlgebra містять множину функцій для структурних перетворень матриць і векторів, команд виконання матричних операцій.

Лабораторна робота №2

Чисельне й аналітичне розв’язання звичайних рівнянь та систем

Мета: здобуття практичних навичок розв’язання звичайних рівнянь та їхніх систем за допомогою прикладного пакету Maple

2.1 Основні теоретичні положення

Способи завдання функцій. Заміна змінних

В Maple є кілька способів подання функції.

Спосіб 1. Визначення функції за допомогою оператора присвоювання :=: якомусь вираженню привласнюється ім'я, наприклад:

> *f:=sin(x)+cos(x);*

Якщо задати конкретне значення змінної x, то вийде значення функції f для цього x. Наприклад, якщо продовжити попередній приклад й обчислити значення f при , те варто записати:

> *x:=Pi/4;* $\rightarrow x:=\frac{\pi}{4}$

> *f;* $>\sqrt{2}$

Після виконання цих команд змінна x має задане значення .

Щоб назовсім не привласнювати змінної конкретного значення, зручніше використати команду підстановки subs({x1=a1, x2=a2,..., },f), де у фігурних дужках указуються змінні xi й їхні нові значення ai (i=1,2,...), які варто підставити у функцію f . Наприклад:

> *f:=x*exp(-t);* $\rightarrow f:=xe^{(-t)}$

> *subs({x=2,t=1},f);* $\rightarrow 2e^{(-1)}$

Всі обчислення в Maple за замовчуванням виробляються символьно, тобто результат буде містити в явному виді ірраціональні константи, такі як, і інші. Щоб одержати наближене значення у вигляді числа із плаваючою комою, варто використати команду evalf(expr,t), де expr - вираз, t - точність, виражена в числах після коми. Наприклад, у продовження попереднього приклада, обчислимо отримане значення функції приблизно:

> evalf(%); > .7357588824

Тут використаний символ (%) для виклику попередньої команди.

Спосіб 2. Визначення функції за допомогою функціонального оператора, що ставить у відповідність набору змінних (x1,x2,...) одне або кілька виражень (f1,f2,...). Наприклад, визначення функції двох змінних за допомогою функціонального оператора виглядає в такий спосіб:

> f:=(x,y)->sin(x+y); → f:=sin(x+y)

Звертання до цієї функції здійснюється найбільш звичним у математику способом, коли в дужках замість аргументів функції вказуються конкретні значення змінних. У продовження попереднього приклада обчислюється значення функції:

> f(Pi/2,0); > 1

Спосіб 3. За допомогою команди unapply(expr,x1,x2,...), де expr - вираз, x1,x2,... - набір змінних, від яких воно залежить, можна перетворити вираз expr у функціональний оператор. Наприклад:

> f:=unapply(x^2+y^2,x,y); →f:=(x,y) – >$x^2 + y^2$

> f(-7,5); > 74

Змінні

Як треба із самої назви, змінні - це об'єкти, значення яких можуть мінятися по ходу виконання документа. Поки ми розглядаємо лише глобальні змінні, доступні для модифікації значень у будь-якому місці документа. Тип змінної в системі Maple визначається привласненим їй значенням - це можуть бути цілочисельні (integer), раціональні (rational), речовинні (real), комплексні (complex) або рядкові (string) змінні й т.д. Змінні можуть також бути символьного типу (їхнім значенням є математичне вираз) або типу списку (див. далі). Для явної вказівки типу змінних використається конструкція:

name::type де name - ім'я (ідентифікатор) змінної, type - тип змінної, наприклад цілочисельний (integer), речовинний із плаваючою крапкою (float), з ненегативним значенням (nonneg), комплексний (complex) і т.д.

Універсальні графічні команди зібрані в пакеті plots (їх можна поділити на команди двовимірної й просторової графіки), а в подпакеті statplots пакета stats перебувають спеціальні команди відображення статистичних даних. Команди побудови графіків чисельних рішень звичайних диференціальних рівнянь DEplot() і рівнянь у частинних похідних PDEplot() можна знайти, відповідно, у пакетах DEtools й PDEtools.

- plot() - призначена для побудови графіків функцій одна змінної (двовимірна графіка);
- plot3d() - будує тривимірні графічні відображення поверхонь і просторових кривих.

Розв’язання звичайних рівнянь.

Для розв’язання рівнянь в *Maple* існує універсальна команда **solve(eq,x)**, де **eq** – рівняння, **x** – змінна, щодо якої рівняння треба вирішити. У результаті виконання цієї команди в рядку висновку з'явиться вираз, що є розв’язанням даного рівняння. Наприклад:

*> solve(a*x+b=c,x);*

Якщо рівняння має кілька рішень, які вам знадобляться для подальших розрахунків, то команді solve варто привласнити яке-небудь ім'я name. Звертання до якому-небудь k-ому розв’язання даного рівняння виробляється вказівкою його імені з номером розв’язання **k** у квадратних дужках: name[k]. Наприклад:

> x:=solve(x^2-a=0,x);
x := − √a,√ a
> x[1];
− √a
> x[2];
√a
> x[1]+x[2];
0

Чисельне розв’язання рівнянь.

Для чисельного розв’язання рівнянь, у тих випадках, коли трансцендентні рівняння не мають аналітичних рішень, використається спеціальна команда **fsolve(eq,x)**, параметри якої такі ж, як і команди **solve**. Наприклад:

> x:=fsolve(cos(x)=x,x);
x:=.7390851332

2.2 Завдання до виконання лабораторної роботи

Згідно з номером варіанту розв’язати наступні рівняння та системи рівнянь

Таблиця 2.1 – Рівняння та системи рівнянь

№	Завдання	
1	$x^4-x^3+5*x^2=0$	$\begin{cases} x+2y+4z=8 \\ 3x-6y+9z=3 \\ 7x+7y-3z=14 \end{cases}$
2	$x^3-x^2-5=0$	$\begin{cases} x-y+5z=3 \\ 3x-2y-7z=1 \\ 3x+4y-z=-1 \end{cases}$
3	$2x^2-5x=7$	$\begin{cases} x+2y-z=-3 \\ 2x+3y+z=-1 \\ x-y-z=3 \end{cases}$
4	$log_{x^2}\frac{4}{(7-x)}=5$	$\begin{cases} 2x+3y=5 \\ 4x+6y+z=7 \\ 5y-4z=12 \end{cases}$
5	$lg\,x=x-5$	$\begin{cases} 3x-4y-4z=5 \\ 6x-8y-2z=10 \\ x+y-5z=-4 \end{cases}$
6	$\frac{x-1}{x^2}=4$	$\begin{cases} x+3y-z=0 \\ 2x-y+3z=1 \\ 7x+7y+3z=2 \end{cases}$
7	$(x+4)^2=1$	$\begin{cases} x+3y-4z=5 \\ 2x-3y+6z=11 \\ 8x-3y+10z=21 \end{cases}$
8	$\frac{x^2-1}{x+2}=4x$	$\begin{cases} 3x+y+z=-2 \\ 5x-y-z=10 \\ x-y+5z=-12 \end{cases}$
9	$tg\ (x^2-1)=0.5$	$\begin{cases} x-2y+z=3 \\ 2x-4y+2z=5 \\ 3x-6y+3z=9 \end{cases}$
10	$\sqrt{x-1}=x$	$\begin{cases} x-4y+3z=3 \\ 2x-8y+6z=10 \\ 3x-12y+9z=15 \end{cases}$

2.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити наступні пункти:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи

3. Стислий зміст роботи
4. Результати роботи

2.4 Контрольні запитання

1. Які функції для розв’язання рівнянь Ви знаєте?
2. Назвіть синтаксис цих функцій
3. Яким чином розв’язується система рівнянь?
4. У чому полягає відмінність аналітичного розв’язання від чисельного?

Лабораторна робота №3

Розв’язання диференціальних рівнянь і систем

Мета: здобуття практичних навичок розв’язання диференціальних рівнянь та їхніх систем за допомогою прикладного пакету Maple

3.1 Основні теоретичні положення

Основні можливості системи Maple.

Розв’язання диференціальних рівнянь всіляких типів - одне з достоїнств системи Maple. Для цієї мети можуть бути використані функції ядра системи (dsolve, pdsolve, pdesolve), інструментальні пакети DEtools (розв’язання звичайних **диференціальних**. рівнянь) і PDEtools (розв’язання **диференціальних**. рівнянь у частинних похідних). Крім цього існує множина функцій, що дозволяють класифікувати рівняння, робити заміни в **диференціальних**. рівняннях, здійснювати перетворення рішень, визуалізувати отримані розв’язання (будувати графіки різних типів).

Звичайні диференціальні рівняння.

Для розв’язання звичайних диференціальних рівнянь і системи простих диференціальних рівнянь використається наступна функція ядра системи:

dsolve(deqn)
dsolve({deqn,In}, var)
dsolve({deqns,Inc},{vars})
dsolve({deqns,In},{vars}, option)

де deqn(s)-одне диференціальне рівняння або система з диференціальних рівнянь першого порядку, Inc початкові умови,var(s) -

змінна або змінні, щодо яких шукається розв’язання,option-необов'язковий параметр, що вказує на метод розв’язання.

Параметр option задає один з методів розв’язання:

- exact - аналітичне розв’язання (прийняте за замовчуванням);
- explicit - розв’язання в явному виді;
- laplace - розв’язання через перетворення Лапласа;
- series - розв’язання у вигляді ряду з порядком, що вказують значенням змінної Order (указується до звертання до функції dsolve);
- numeric - розв’язання в чисельному виді.

```
> ode1 :=
  diff(y(x),x)-y(x)^2+y(x)*sin(x)-cos(x);
```

$$ode1 := \left(\frac{\partial}{\partial x} y(x)\right) - y(x)^2 + y(x)\sin(x) - \cos(x)$$

```
> ans1 := dsolve(ode1);
```

$$ans1 := y(x) = \sin(x) - \frac{\_C1\, e^{(-\cos(x))}}{\_C1 \int e^{(-\cos(x))}\, dx + 1}$$

```
> restart;
```

```
> ode2:=(D@@2)(y)(x)+y(x);
```

$$ode2 := (D^{(2)})(y)(x) + y(x)$$

```
> dsolve(ode2,y(x));
```

$$y(x) = \_C1\cos(x) + \_C2\sin(x)$$

```
> assign(%);y(x);
```

$$\_C1\cos(x) + \_C2\sin(x)$$

```
> g(x):=diff(y(x),x);
```

$$g(x) := -\_C1\sin(x) + \_C2\cos(x)$$

Для розв’язання задачі Коші в In треба задати початкові умови, а при рішенні крайових задач - крайові умови. Якщо Maple здатна знайти розв’язання при числі початкових або крайових умов менше порядку системи, то в рішенні будуть з'являтися невизначені константи виду _C1,_C2 і т.д. Вони ж можуть бути при аналітичному рішенні системи, коли початкові умови не задані. Якщо розв’язання знайдене в неявному виді, то в ньому з'явиться параметр _T. Для присвоєння отриманого розв’язання шуканої функції використається команда assign.

```
> de:=m*diff(y(x),x$2)-k*diff(y(x),x);
  yx0:=y(0)=0,D(y)(0)=1;
  # Определение диф.уравнения и нач.усл.
```

$$de := m\,y'' - k\,y'$$

$$yx0 := y(0) = 0, D(y)(0) = 1$$

```
> dsolve({de,yx0},y(x));
```

$$y = -\frac{m}{k} + \frac{m\,e^{\left(\frac{k\,x}{m}\right)}}{k}$$

```
> de2:=m*diff(y(x),x$2)-k*diff(y(x),x);
  yx2:=y(0)=0,diff(y(b),b)=1;
  yy:=dsolve({de2,yx2},y(x));
```

$$de2 := m\,y'' - k\,y'$$

$$yx2 := y(0) = 0, y_b = 1$$

$$yy := y = -\frac{m}{k\,e^{\left(\frac{k\,b}{m}\right)}} + \frac{m\,e^{\left(\frac{k\,x}{m}\right)}}{k\,e^{\left(\frac{k\,b}{m}\right)}}$$

```
> subs(y(x)=yy,de2):
  simplify(%);
```

$$m\,y'' - k\,y' = 0$$

Похідні при записі диференціальних рівнянь і початкових/граничних умов можуть задаватися функцією diff або оператором D.

Якщо умова на похідну задається в довільній точці x=b, то для його завдання може бути використаний оператор diff: наприклад, diff(y(b),b)=1. Умови на похідні в конкретній точці повинні бути записані за допомогою диференціального оператора D: D(y)(0)=5, D(D(y)(4))=7.

При рішенні систем диференціальних рівнянь самі рівняння й початкові (граничні) умови вказуються у фігурних дужках. Множина шуканих функцій також указується у фігурних дужках.

Тут для виділення окремих функцій розв'язання системи використається функція підстановки subs. Далі можна побудувати графіки рішень, використовуючи функцію plot.

```
> Order:=8:
  FS:=dsolve({sys,InC},funcs,series);
```

$$FS := \{z(x) = 1 + x^2 - \frac{1}{2}x^3 + \frac{7}{24}x^4 - \frac{13}{120}x^5 + \frac{3}{80}x^6 - \frac{53}{5040}x^7 + O(x^8),$$

$$y(x) = 2\,x - \frac{3}{2}x^2 + \frac{7}{6}x^3 - \frac{13}{24}x^4 + \frac{9}{40}x^5 - \frac{53}{720}x^6 + \frac{107}{5040}x^7 + O(x^8)\}$$

```
>
```

```
> FF:=dsolve({sys,InC},funcs,laplace);
```

$$FF := \{y(x) = -\frac{5}{6}e^{(-2\,x)} + \frac{1}{3}e^{x} + \frac{1}{2},$$

$$z(x) = \frac{1}{3}e^{x} + \frac{5}{12}e^{(-2\,x)} + \frac{1}{2}x + \frac{1}{4}\}$$

```
> F=FF:evalb(%);
```

true

```
> evalb(F=FS);
```

false

При використанні опції laplase для розв’язання системи ми одержимо ту ж відповідь:

При використанні опції series одержуване у вигляді ряду розв’язання відрізняється від явного розв’язання й розв’язання із застосуванням перетворення Лапласа.

Для перевірки знайденого розв’язання може бути використана функція odetest (sol,ODE), де в sol - знайдене розв’язання , ODE - вихідне диференціальне рівняння

Для класифікації диференціальних рівнянь в Maple існує функція odeadvisor:

odeadvisor(ODE)

odeadvisor(ODE, y(x), [type1, type2, ...], help)

```
> with(DEtools):
> ODE :=
  x*diff(y(x),x)+a*y(x)^2-y(x)+s*x^2;
```

$$ODE := x\left(\frac{\partial}{\partial x}y(x)\right) + a\,y(x)^2 - y(x) + s\,x^2$$

```
> odeadvisor(ODE);
```

$$[[\_homogeneous, class\ D], \_rational, \_Riccati]$$

ODE- звичайне диференціальне рівняння

y(x)- шукана функція

type1, type2,... - опція, підмножина типів класифікації

help - опція, указує чи треба виводити довідку із приводу методів розв’язання даного виду **диференціальних**. рівнянь (виводиться в окремих вікнах).

Чисельне розв’язання диференціальних рівнянь.

Для розв’язання диференціальних рівнянь у чисельному виді використається функція dsolve з опцією numeric або type=numeric. При цьому розв’язання повертається у вигляді спеціальної процедури, за

замовчуванням реалізуючий широко відомий метод розв'язання диференціальних рівнянь Рунге-Кутта-Фельберга порядку 4 й 5 (залежно від умов адаптації розв'язання до швидкості його зміни). Ця процедура називається rkf45 і символічно виводиться (без тіла) при спробі розв'язання заданої системи диференціальних рівнянь.

У список параметрів функції dsolve можна явно включити вказівку на метод розв'язання, наприклад, опція method=dverk78 задає розв'язання методом Рунге-Кутта порядку 7 або 8.

```
sys1:=diff(y(x),x)=2*z(x)*sin(x)-y(x)-x,
diff(z(x),x)=y(x);

funcs:={y(x),z(x)};
InC:=y(0)=0,z(0)=1;
```

$$sys1 := \frac{\partial}{\partial x} y(x) = 2\,z(x)\sin(x) - y(x) - x,\ \frac{\partial}{\partial x} z(x) = y(x)$$

$$funcs := \{y(x), z(x)\}$$

$$InC := y(0) = 0, z(0) = 1$$

```
> FNUM:=dsolve({sys1,InC},funcs,numeric,
  output=listprocedure);
```

$$FNUM := [x = (\mathbf{proc}(x) \ \ldots\ \mathbf{end\ proc}),\ y(x) = (\mathbf{proc}(x) \ \ldots\ \mathbf{end\ proc}),\ z(x) = (\mathbf{proc}(x) \ \ldots\ \mathbf{end\ proc})]$$

```
> Y1:=subs(FNUM,y(x));Z1:=subs(FNUM,z(x));
```

$$Y1 := \mathbf{proc}(x) \ \ldots\ \mathbf{end\ proc}$$

$$Z1 := \mathbf{proc}(x) \ \ldots\ \mathbf{end\ proc}$$

```
> plot({Y1,Z1},0..6,color=[blue,green],
  thickness=2);
```

У ряді випадків мати явне розв'язання диференціальних рівнянь недоцільно. Для неявного їхнього подання в Maple уведена спеціальна структура DESol:

DESol(expr,vars), expr - вихідна система рівнянь або одне рівняння, vars - заданий список змінних (або одна змінна).

Структура DESol утворить деякий об'єкт, що нагадує RootOf. Із цим об'єктом можна звертатися як з функцією: інтегрувати, диференціювати, одержувати розкладання в ряд, використати в обчисленнях.

```
> dde:=DESol(D(y)-y,y,{y(0)=1});
```

$$dde := \mathrm{DESol}(\{D(y) - y\}, \{y\}, \{y(0) = 1\})$$

```
> series(dde(x),x,8);
```

$$y(0) + y(0)\,x + \frac{1}{2}y(0)\,x^2 + \frac{1}{6}y(0)\,x^3 + \frac{1}{24}y(0)\,x^4 + \frac{1}{120}y(0)\,x^5 + \frac{1}{720}y(0)\,x^6 + \frac{1}{5040}y(0)\,x^7 + O(x^8)$$

```
> D(dde)-dde;
```

$$0$$

3.2 Завдання до лабораторної роботи

Згідно з номером варіанту розвязати наступні диференціальні рівняння та їхні системи

Таблиця 3.1 – Рівняння та системи дифференційних рівняннь

№	**Завдання**	
1	$tg\ x=\sqrt{x^2+3}$	$\begin{cases} x'=-2x+4y \\ y'=-x+3y \end{cases}$
2	$x^2y'=\frac{1}{\cos y}$	$\begin{cases} 2x''=-6x+2y \\ y''=2x-2y \end{cases}$
3	$y'=x^3y^2+2x^3y$	$\begin{cases} x'=-2y \\ y'=0.5x \end{cases}$
4	$xy^2=2xy'+3$	$\begin{cases} x'=-y \\ y'=1.01x-0.2y \end{cases}$
5	$(x+1)y'=y^2\sin x+xy^2$	$\begin{cases} x''-5x+4y=0 \\ y''+4x-5y=0 \end{cases}$
6	$(x^2+e^x)y'=xy+y^2\cos x$	$\begin{cases} x''=(1-y)x \\ y'=(1-x)y \end{cases}$
7	$y'=\frac{2x+4y}{x-3y}$	$\begin{cases} x'=-2y \\ y'=2x \end{cases}$
8	$y'=tg\ln\sin\frac{y}{x}$	$\begin{cases} x'=0.5y \\ y'=-8x \end{cases}$
9	$y'=\cos\frac{y}{x}+\frac{x^2}{y^2}$	$\begin{cases} x'=10y \\ y'=-10x \end{cases}$
10	$y'=\frac{\sqrt{x^4-2y^4}}{\sqrt[3]{x^6+xy^5+2x^4y^2}}$	$\begin{cases} x'=-y \\ y'=10x-7y \end{cases}$

3.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити наступні пункти:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи

3. Стислий зміст роботи
4. Результати роботи

3.4 Контрольні запитання

1. Які функції для розв’язання диференціальних рівнянь Ви знаєте?
2. Назвіть синтаксис цих функцій
3. Яким чином розв’язується система диференціальних рівнянь?
4. У чому полягає відмінність аналітичного розв’язання від чисельного?

Лабораторна робота №4
Мова програмування Maple

Мета: оволодіти навичками програмування за допомогою прикладного пакету Maple

4.1 Основні теоретичні положення

Цикли

Найчастіше необхідно циклічне повторення виконання вираз задане число раз або доти, поки виконується певна умова. Maple має узагальнену конструкцію циклу, що задається в такий спосіб:

| for <name>| |from <exprl>| |to <expr3>| |by <expr2>| (while <expr4>| do Statement sequence> od;

Тут name - ім'я керуючої змінної циклу, exprl, expr2 й exprз - вираз, що задають початкове значення, кінцеве значення й крок зміни змінної name, expr4 - вираз, що задає умова, поки цикл (набір об'єктів між словами do й od) буде виконуватися. [10]

У ході виконання циклу керуюча змінна міняється від значення exprl до значення expr2 із кроком, заданим exprз. Якщо блок by <expr2> відсутній, то керуюча змінна буде мінятися із кроком +1 при exprгкexpr2. Це наочно пояснює наступний приклад:

> for i front 1 to 5 do printd) od; 1 2 3 4 5

У ньому виводяться значення змінної і у ході виконання циклу. Неважко помітити, що вона дійсно змінюється від значення 1 до значення 5

із кроком +1. Наступний приклад показує, що границі зміни керуючої змінної можна задати арифметичними виразами:

> for i from 7/(2+5) to 2+3 do printd) od: 1 2 3 4 5

А ще один приклад показує завдання циклу, у якого змінна циклу міняється від значення 1 до 10 із кроком 2:

> for i from 1 to 10 by 2 do printd) od: 1 3 5 7 9

У цьому випадку виводяться непарні числа від 1 до 9. Крок може бути й негативним:

> for i from 9 to 1 by -2 do print(i) od: 9 7 5 3 1

Слід зазначити, що якщо exprl>expr2 задати свідомо нездійсненну умову, наприклад, expr1>expr2 і позитивне значення кроку, то цикл виконуватися не буде. Цикл можна перервати за допомогою додаткового блоку while <expr4>. Цикл із таким блоком виконується до кінця або доти, поки умова expr4 істинно.

> for i from 1 to 10 by 2 while i<6 do print(i) od: 1 3 5

Таким чином, конструкція циклу в Maple-мові програмування увібрала в себе основні конструкції циклів for й while. Є ще одна, більше специфічна конструкція циклу:

|for <name>| |in <exprl>| |whi1e <expr2>| do statement sequence> od:

Тут exprl задає список значень, які буде приймати керуюча змінна name. Цикл буде виконуватися, поки не буде вичерпані список і поки виконується умова, задане виразм expr2. Наступний приклад ілюструє сказане:

> for i in [1.2,5,-1.7.12] do print(i) od; 1 2 5 -1 7
12 > for i in [1,2.5.-1,7.12] while i>0 do print(i) od: 1 2 5

На закінчення відзначимо, що можливо спрощену приватну конструкцію циклу типу while:

while expr do statseq od:

Тут вираз statseq виконуються, поки виконується логічна умова expr. Приклад такого циклу:

> n:=1: n:=1 .
*> while n<16 do n:»2*n od;*
n:=2 n := 4 n := 8 n :=16

У цьому прикладі йде подвоєння числа n з початковим значенням n = 1 доти, поки значення n менше 16.

Елементи порівняння

Ми вже зустрічалися з оператором присвоювання «:=» раніше, що використається для присвоювання значення деякому імені. Цей розділ пояснює розходження між оператором присвоювання позначуваним «:=» (символ двокрапки супроводжуваний знайомій рівності =), і оператором рівняння, позначуваним знаком рівності =.

Варто зробити ще кілька зауважень щодо оператора присвоювання. При використанні оператора присвоювання Maple пам'ятає тільки останнє привласнене значення для будь-який змінної. Якщо Ви привласните змінної x значення 5, а потім - значення 75, то запам'ятається тільки останнє присвоювання. Ви можете перевизначити будь-яке використане вами ім'я для команди або іншого об'єкта, однак Maple не дозволить вам використати ім'я для змінної, якщо воно використається як ім'я однієї з убудованих команд, функцій або констант Maple. Імена цих об'єктів захищені й ви одержите повідомлення про помилку. Крім того, можна, використовуючи команду protect(ім'я) захистити будь-яке уведене вами ім'я.

Оператор рівняння (знак рівності «=»), на відміну від розглянутого вище оператора присвоювання, - просто математичні вираз, що зв'язує між собою деякі змінні й значення. Рівняння не привласнюють явних значень змінним, які вони містять.

Наприклад:

```
> x = y + 3;
> x;
> y;
```

Змінним x й y нічого не привласнюється. Оператор «=» найчастіше вживається або в параметрі команди Maple або у висновку результату. Дуже корисне сімейство команд, що використають оператор «=» - команди розв’язання рівнянь різного виду:

- solve призначена для аналітичного розв’язання лінійних і нелінійних рівнянь, нерівностей і систем;
- fsolve призначена для чисельного розв’язання лінійних і нелінійних рівнянь, нерівностей і систем;
- dsolve вирішує набір звичайних диференціальних рівнянь;
- rsolve вирішує набір рекурсивних рівнянь.

Приведемо приклади.

> sols:= solve({ x+y=3, x-y=1 }, { x, y }); → sols:={y=1, x=2}

> x;

> y;

Отримане розв’язання - набір рівнянь для визначення змінних. Якщо є багато рішень, вони все будуть отримані. У той же час змінним x й y у вищезгаданому прикладі значення рішень не привласнюються. Для присвоювання рішень вихідним змінним потрібно використати команду assign , що у рівнянні (або наборі рівнянь) заміняє кожен оператор = на оператор :=.

> assign(sols);

> x; → 2

> y; → 1

> x:='x'; → x:=x

> y:='y'; → y:=y

Інше часте використання знака рівності - в операторах булевих (логічних) виражень. Коли необхідно з'ясувати характер залежності між значеннями двох змінних, оператор «=» може використатися для перевірки рівності. Інші булеві оператори порівняння : <, <=, <>, and, or, not. Команда evalb перевіряє, чи є булево співвідношення щирим або помилковим.

Наприклад

> evalb(sin(Pi/3) = sqrt(3)/2); → true

> evalb(exp(1.)^2 > 10); → false

> evalb (isprime (7) and isprime (131)); → true

Іноді буває потрібним пропустити певне значення змінної циклу. Для цього використається оператор next (наступний). Наведений нижче приклад ілюструє застосування оператора next у складі вираз if-fi для виключення висновку значення i = -2:

> for 1 in [1,2.3.-2.4] do if i--2 then next else print(i) fi od: 1 2.3 4

Інший оператор - break - перериває виконання фрагмента програми (або циклу). Його дія пояснює злегка модифікований попередній приклад:

> for i in [1.2.3.-2,4] do if i=2 then break else print(i) fi od: 1 2 3

У цьому випадку при значенні i = -2 відбулося повне припинення виконання циклу. Тому наступне значення 4 змінної z привласнено не було й це значення на печатку не потрапило.

Любою з операторів quit, done або stop забезпечує також переривання виконання поточної програми (зокрема, циклу), але при цьому вікно поточного документа закривається.

4.2 Завдання до виконання лабораторної роботи

Створити масив даних, що повинен містити не менше 5 різних додатних та 5 різних від'ємних чисел, щонайменше 2 нулі. Далі згідно з номером варіанту сформувати наступний масив:

Таблиця 4.1 – Завдання на програмування

№ варіанту	Завдання
1	Тільки з від'ємних чисел за зростанням
2	Тільки з додатних чисел, більших за 3
3	Тільки з від'ємних чисел за зменшенням
4	Тільки з додатних чисел за зростанням
5	Тільки з додатних чисел за зменшенням

6	Тільки з від’ємних чисел більших за -1
7	Відмінних від нуля від’ємних чисел
8	Відмінних від нуля додатних чисел
9	Виключити нулі з масиву
10	Залишити у масиві один нуль та всі додатні числа

4.3 Зміст звіту

Після виконання лабораторної роботи необхідно скласти звіт, який має містити:

1. Титульний аркуш з назвою лабораторної роботи та номером варіанту
2. Мету роботи
3. Стислий зміст роботи
4. Результати роботи

4.4 Контрольні запитання

1. Які можливості програмування надає програмний пакет?
2. Назвіть та охарактеризуйте оператори циклів
3. Яким чином формується цикл програми?

Список використаної літератури

1. В.З. Аладьев, В.К. Бойко, Е.А. Ровба «Программирование в пакетах Maple и Mathematica: Сравнительный аспект» / Монография / Гродно: Гродненский Госуниверситет, 2011, 517 с.
2. В.З. Аладьев, Д.С. Гринь «Расширение функциональной среды системы Mathematica» / Монография / Херсон: Олди-Плюс, 2012, 552 с.
3. Я. К. Шмидский Mathematica 5. Самоучитель. М.: Диалектика. 2004.
4. Курбатова Е.А. MATLAB 7. Самоучитель. Издательство: Вильямс. Год издания: 2005г. 256 стр.
5. Алексеев Е.Р., Чеснокова О.В. MATLAB 7. Самоучитель. ISBN: 5-477-00283-2. Издательство "НТ Пресс" 2006г. 464 стр.
6. Гандер В., Гржебичек И. Решение задач в научных вычислениях с применением Maple и MATLAB. ISBN: 985-6642-06-X. Издательство «Вассамедина» 2005г. 520 стр.
7. В.Ф. Очков. Mathcad 14 для студентов и инженеров. С.-Пб.: БХВ-Петербург, 2007.
8. Е. Р. Алексеев, О. В. Чеснокова. Решение задач вычислительной математики в пакетах Mathcad 12, MATLAB 7, Maple 9. М: НТ Пресс, 2006, 496с.
9. М. Н. Кирсанов. "Практика программирования в системе Maple" М.: Издательский дом МЭИ, 2011, 208с.
10. М.Семененко. Введение в математическое моделирование. М.:Солон-Р, 2002.