Міністерство інфраструктури України
Державна служба зв'язку
Одеська національна академія зв'язку ім. О.С.Попова

Шаповаленко В. А., Буката Л. М., Трофименко О. Г.

Чисельні методи та моделювання на ЕОМ

МОДУЛЬ 2

Чисельні методи моделювання об'єктів

Частина 2

Методичні вказівки для лабораторних та практичних занять

для студентів напрямів бакалаврської підготовки:
"***Автоматизація та комп'ютерно-інтегровані технології***",
"***Телекомунікації***", "***Мережі та системи поштового зв'язку***",
"***Системи технічного захисту інформації***"

Одеса 2011

УДК 004.43

План НМВ 2011 р.

Шаповаленко В. А. Чисельні методи моделювання об’єктів: метод. вказівки для лаб. та практ. занять. Модуль 2. / Шаповаленко В. А., Буката Л. М., Трофименко О. Г. – Одеса: ВЦ ОНАЗ, 2011. – Ч. 2. – 72 с.

Містить короткі теоретичні відомості про методи розв’язування інженерних і наукових задач з використанням чисельних методів. У даній частині вказівок подано методи й алгоритми розв’язування задач апроксимації, інтерполяції, оптимізації та розв’язку систем диференційних рівнянь і приклади їхнього програмування алгоритмічною мовою С++ і в математичних пакетах Mathcad та MATLAB. Згідно з навчальним планом другого модуля у вказівках подано індивідуальні завдання для виконання лабораторних робіт та комплексного завдання.

Призначено для набуття теоретичних та практичних знань студентами академії, які вивчають дисципліни “Чисельні методи та моделювання на ЕОМ”, “Основи математичного моделювання”, “Програмування інженерних задач”. Буде корисним для студентів при закріпленні лекційного матеріалу та підготовці до практичних і лабораторних занять. Також стане у пригоді аспірантам та науковим співробітникам при розв’язуванні поширених наукових задач з використанням чисельних методів та математичних пакетів.

СХВАЛЕНО
на засіданні кафедри
інформаційних технологій
і рекомендовано до друку.
Протокол № 9 від 8 квітня 2010 р.

ЗАТВЕРДЖЕНО
методичною радою
академії зв’язку
Протокол № 8 від 11.02.2011 р.

ПЕРЕДМОВА

Дисципліна **“**Чисельні методи та моделювання на ЕОМ” викладається в семестрах 2.3 і 2.4 студентам другого курсу, які навчаються за напрямом бакалаврської підготовки “Автоматизація та комп’ютерно-інтегровані технології”, призначена для формування знань та навичок з використання комп’ютерів у професійній та повсякденній діяльності, а саме: програмування та розв’язування поширених інженерних задач автоматизації технологічних процесів з використовуванням чисельних методів та математичних пакетів. Методи та алгоритми розв’язування інженерних задач, подані у вказівках, також можуть використовувати студенти, які вивчають дисципліну “Програмування інженерних задач” (напрям “Мережі та системи поштового зв’язку”), “Чисельні методи” (напрям “Системи технічного захисту інформації”) та “Основи математичного моделювання” (напрям “Телекомунікації”).

Мета дисципліни – формування у студентів знань та навичок у таких областях:

- моделювання технологічних об’єктів;
- математичні пакети;
- чисельне диференціювання та інтегрування;
- розв’язування нелінійних рівнянь;
- розв’язування систем алгебраїчних рівнянь;
- чисельні методи розв’язування диференційних рівнянь та систем;
- інтерполяція та апроксимація функцій;
- чисельні методи розв’язування задач одновимірної оптимізації функцій;
- розв’язування задач багатовимірної оптимізації.

Програма курсу складається з двох модулів:

Модуль 1 Чисельне обчислення функцій, характеристик матриць і розв’язування нелінійних рівнянь та систем рівнянь.

Модуль 2 Чисельні методи моделювання об’єктів.

Кожна із запропонованих до виконання лабораторних робіт у другому модулі має індивідуальне завдання. Метою завдань є набуття практичних навичок розв’язування інженерних задач з використанням чисельних методів, математичних пакетів Mathcad і MATLAB та закріплення знань і вмінь програмувати алгоритмічною мовою C++.

Кожне лабораторне завдання містить 30 індивідуальних варіантів. Студент сам чи то за вказівкою викладача обирає варіант завдання згідно до номера студента у списку групи.

До виконання лабораторної роботи допускається студент, який самостійно підготував протокол лабораторної роботи згідно до нижченаведених вимог. Після виконання лабораторної роботи студент повинен занести до протоколу результати обчислень, графіки функцій та зробити відповідні висновки. Правильність здобутих результатів перевіряє викладач.

Перелік знань та умінь, яких повинен набути студент у процесі вивчення матеріалу модуля 2

Знання:

- поняття моделювання об’єктів;
- елементи програмування у математичному пакеті MATLAB;
- інтерполяція та апроксимація функцій для моделювання;
- обчислювальні методи розв’язування звичайних диференційних рівнянь та їхніх систем;
- система математичного моделювання Simulink;
- чисельні методи пошуку екстремуму функції.

Вміння:

- інтерполювати функцію за допомогою C++ та Mathcad;
- апроксимувати функцію за допомогою C++ та Mathcad;
- розв’язувати диференційні рівняння у C++ та Mathcad;
- обчислювати функції у середовищі MATLAB;
- складати найпростіші моделі у Simulink;
- знаходити оптимальні значення функції.

Перелік лабораторних робіт

Лабораторна робота № 1. Обчислення функцій та побудова графіків у середовищі Matlab. Елементи моделювання в Simulink.

Лабораторна робота № 2. Розв’язування диференційних рівнянь та систем за допомогою програмних засобів (C++, Mathcad).

Лабораторна робота № 3. Інтерполяція та апроксимація функцій за допомогою C++ та Mathcad.

Лабораторна робота № 4. Розв’язування задач пошуку екстремуму.

Вимоги щодо оформлення протоколу лабораторної роботи

1 Лабораторні роботи оформлюються в окремому зошиті.

2 Для кожної лабораторної роботи слід записати тему та мету.

3 Для *кожної задачі* лабораторної роботи слід записати такі розділи:

а) умову задачі за індивідуальним варіантом;

б) опис розв’язування задачі на комп’ютері;

в) результати обчислень на комп’ютері (після виконання програми на комп’ютері);

г) аналіз результатів та висновки.

4 Наприкінці роботи треба написати своє прізвище, поставити підпис і дату виконання роботи.

1 ЕЛЕМЕНТИ ПРОГРАМУВАННЯ MATLAB

1.1 Призначення й особливості програмування у MATLAB

Більшість математичних систем створювалися зважаючи на те, що користувач буде розв'язувати свої задачі, практично не займаючись програмуванням. Однак, з самого початку було зрозуміло, що подібний підхід має недоліки й, загалом кажучи, є порочний. Адже значна кількість задач потребує розвинених засобів програмування, які спрощують запис алгоритмів задач та інколи відкривають нові методи створення алгоритмів.

З одного боку MATLAB містить величезну кількість операторів та функцій для розв'язання різноманітних практичних задач, що знімає потребу написання доволі складних програм. Приміром, це функції обігу та транспонування матриць, обчислення значень похідних та інтегралів тощо. Кількість таких функцій з урахуванням пакетів розширення системи вже сягає багатьох тисяч і безупинно збільшується.

Але з іншого боку, з моменту свого створення, система MATLAB розроблялась як потужна математико-орієнтована мова програмування високого рівня. Ця важлива перевага системи свідчить про можливості її застосування для розв'язування нових, найбільш складних математичних задач.

Система MATLAB має вхідну мову, що нагадує Бейсик (з домішкою Фортрану та Паскаля). Запис програм у системі є традиційний і звичний для більшості користувачів комп'ютерів. Крім того, система надає можливість редагувати програми за допомогою будь-якого звичного для користувача текстового редактора. Вона має і свій власний редактор з відлагоджувачем. Відмова від притаманної для системи Mathcad переваги – записування задач у вигляді формул – компенсується помітним збільшенням швидкості обчислень – за інших однакових умов вона майже на порядок є вище, аніж у системи Mathcad. А це є суттєва перевага! Мова системи MATLAB у частині програмування математичних обчислень набагато багатше будь-якої універсальної мови програмування високого рівня. Вона реалізує майже всі відомі засоби програмування, у тому числі об'єктно-орієнтоване й (засобами Simulink) візуальне програмування. Це надає досвідченим програмістам неосяжні можливості для самовиразу.

Головне вікно MATLAB наведено на рис. 1.1. Воно складається (за замовченням − default) з наступних частин: рядок команд меню (File, Edit, View, Window та Help), рядок піктограм, вікна Command Window, Command History, Workspace, Current Directory.

Вікно Command Window призначено для роботи у командному режимі − виконуються тільки одна команда чи оператор за форматом:

>> команда (оператор)<Enter>

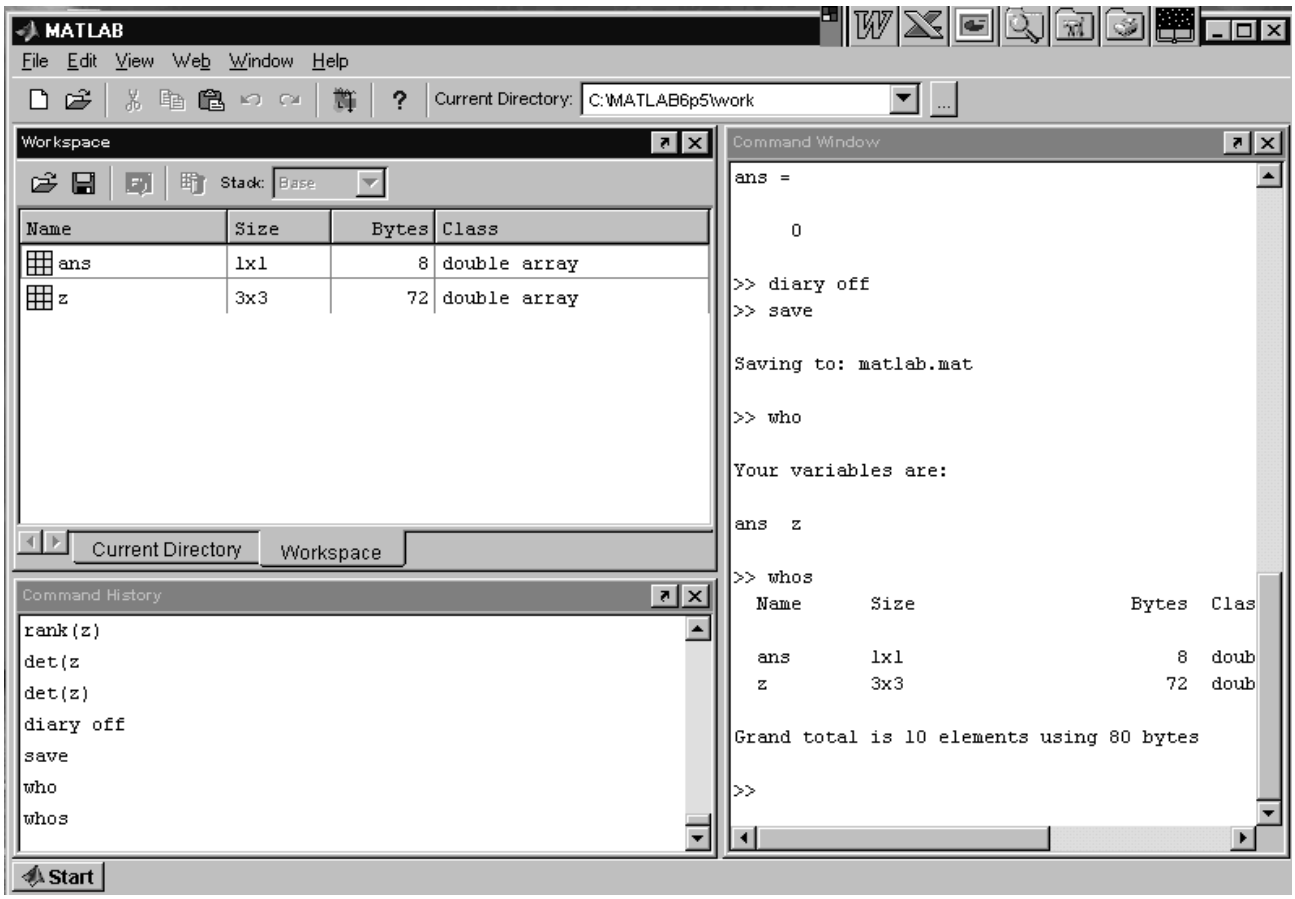


Рисунок 1.1 − Вікно системи MATLAB

Символи >> означають запрошення MATLAB до введення команди. Після натискання на клавішу <Enter> відразу ж виконується команда або оператор, наприклад:

```
>> who
Your variables are:
ans    z
```

В наведеному прикладі після команди who у наступному рядку виведено імена змінних оперативної пам'яті `ans` та `z` (рис.1.1). Декілька операторів у командному ряду:

```
>>x=2; y=x+7
```

Виконують присвоєння значення 2 змінній x та обчислення значення змінної y за формулою y=x+7; після натискання на клавішу <Enter> отримаємо результат y = 9.

Вікно Command History запам'ятовує та відображає перелік команд, які було введено. Ці команди, після виокремлення, можна копіювати у вікно Command Window та в m-файл. *Вікно Workspace* містить інформацію про змінні в оперативній пам'яті: Name − ім'я змінної, `Size` − розміри матриці, `Bytes` − кількість байтів, `Class` − тип змінної. *Вікно Current Directory* містить список файлів робочої теки MATLAB.

Наведемо деякі корисні команди *керування вікном командного режиму*:

- ✓ clc – очищує вікно Command Window і розміщує курсор у його лівому верхньому куті;
- ✓ clear − очищує оперативну пам'ять (Workspace) від змінних;
- ✓ who − виводить список змінних в оперативній пам'яті;
- ✓ save *<file_name>* − зберегти значення змінних з оперативної пам'яті в файл;
- ✓ load *<file_name>* − завантажити значення змінних з файла в оперативну пам'ять;
- ✓ diary on/off − включити / виключити режим запису протоколу роботи у вікні Command Window до файла;
- ✓ type *<file_name>* − виведення на екран файла зі змістом протоколу, який записано командою diary;
- ✓ home – повертає курсор до лівого верхнього кута вікна;
- ✓ echo *<file_name>* on – вмикає режим виведення на екран тексту Script-файла (файла-сценарія);
- ✓ echo *<file_name>* off – вимикає режим виведення на екран тексту Script-файла;
- ✓ echo *<fi1e_name>* – змінює режим виведення на протилежний;
- ✓ echo on all – вмикає режим виведення на екран тексту усіх m-файлів;
- ✓ echo off all – вимикає режим виведення на екран тексту усіх m-файлів;
- ✓ more on – вмикає режим посторінкового виведення (корисний при перегляданні великих m-файлів);

Починаючи з версії MATLAB 6.0 обидві команди clc та home діють аналогічно – очищують екран і розміщують курсор у лівому верхньому куті вікна командного режиму роботи. Команди echo дозволяють вмикати чи вимикати відображення тексту m-файлів за кожного звертання до них. Як правило, відображення тексту файлів сильно захаращує екран і доволі часто не є необхідним. При великих розмірах файлів початок їхнього тексту (листинга) втікає далеко за межі області переглядання (поточного вікна командного режиму). Тому для переглядання довгих листингів файлів доцільно вмикати посторінковий режим виведення командою more on.

Повідомлення про помилки й виправлення помилок

MATLAB діагностує введені команди і вирази та виводить відповідні повідомлення про помилки чи попередження. Розглянемо декілька прикладів помилкових виразів:

```
» 1 / 0          % ділення на нуль
    Warning: Divide by zero
    ans = Inf

» sqr(2)         % корінь квадратний – sqrt, а не sqr
    ??? Undefined function or variable 'sqr'.
```

Для громіздких виразів зручно користуватись редактором. Для цього достатньо після виразу натиснути клавішу *Tab*. Система виведе підказку, проаналізувавши введені символи. Якщо варіантів декілька, клавішу *Tab* доведеться натискати ще раз. Із запропонованих системою операторів обираємо потрібний і натискаємо клавішу ENTER.

Іноді у ході виведення результатів обчислень виводиться скорочення NaN (від слів **N**ot **a** **N**umber – не число). Воно означає невизначеність, наприклад, виду 0/0 чи Inf/Inf, де Inf – системна змінна зі значенням машинної нескінченості (число 10^{308}).

Взагалі кажучи, у MATLAB слід відрізняти попередження про помилку від повідомлення про неї. Попередження (зазвичай після слова Warning) не зупиняють обчислень, а лише попереджують користувача про те, що діагностована помилка здатна вплинути на хід обчислень. Повідомлення про помилку (після знаків ???) зупиняє обчислення.

1.2 Основні об’єкти MATLAB

Математичні вирази

Центральним поняттям усіх математичних систем є математичний вираз. Він задає те, що має бути обчислено у чисельному (іноді символьному) вигляді. Приклади простих математичних виразів:

```
2+3
2.301*sin(x)
4+exp(3)/5
sqrt(y)/2
sin(pi/2)
```

Математичні вирази будуються на основі чисел, констант, змінних, операторів, функцій та різних спецзнаків. Коротко пояснимо суть цих понять.

Дійсні та комплексні числа

Числа можуть бути цілими і дійсними з фіксованою чи рухомою крапкою. Можливим є подання чисел в експоненціальній формі зі зазначенням мантиси, символа “e” і порядка числа. Приклади подання чисел:

```
0
2
-3
2.301   0.00001   123.456e-24
-234.456e10
```

Комплексні числа містять дійсну та уявну частини. Уявна частина має множник i чи j, який означає корінь квадратний з –1 $\left(\sqrt{-1}\right)$:

```
3i
```

```
2j
2+3i
-3.141i
-123.456+2.7e-3i
```

Спеціальна функція real(z) повертає дійну частину комплексного числа, а функція imag(z) – уявну. Для здобуття модуля комплексного числа використовують функцію abs(z), а для обчислення фази – angle(Z):

```
» i
    ans = 0 +1.0000i
» j
    ans = 0 + 1.0000i
» z=2+3i
    z = 2.0000 + 3.0000i
» abs(z)
    ans = 3.6056
» real(z)
    ans = 2
» imag(z)
    ans = 3
» angle(z)
    ans = 0.9828
```

Взагалі кажучи, у MATLAB не прийнято поділяти числа на цілі і дійсні, короткі й довгі тощо, як це прийнято у більшості мов програмування, хоча задавати числа у таких формах можна.

Константи і системні змінні

Константа – це завчасно визначене числове чи символьне значення, подане унікальним ім'ям. Числа (приміром 1, –2 та 1.23) є безіменними *числовими константами.* Інші види констант у MATLAB називають *системними змінними,* оскільки, з одного боку, вони задаються системою при її завантаженні, а з іншого – можуть перевизначатися.

Основні *системні змінні*:

pi – число $\pi = 3.1416$;

i чи j – уявна одиниця $\left(\sqrt{-1}\right)$;

eps – похибка обчислень над числами з рухомою крапкою $eps = 2^{-52}$, що приблизно складає 2.22e–16;

Inf – подання машинної додатної нескінченності, складає 10^{308}. При операціях ділення на нуль, яке призводить до результатів надто великих, їх буде подано у вигляді числа з рухомою крапкою;

ans – результат виконання останньої операції. Змінна ans створюється автоматично, коли не визначено вихідні аргументи якогось оператора;

NaN – зазначає нечисловий характер даних (скорочення від Not a Number – не число);

realmax – найбільше число у форматі з рухомою крапкою (2^{1023});

realmin – найменше нормалізоване додатне число у форматі з рухомою крапкою (2^{-1022}).

Приклади використання системних змінних:

```
» 2*pi
    ans = 6.2832
» cos([0:2*pi])
    ans = 1.0000 0.5403 −0.4161 −0.9900 −0.6536
    0.2837 0.9602
» realmin
    ans = 2.2251e-308
» realmax
    ans = 1.7977e+308
» 1/0
    Warning: Divide by zero
    ans = Inf
» 0/0
    Warning: Divide by zero
    ans = NaN
```

Як зазначалось, системні змінні можуть перевизначатися. Так системній змінній eps можна надати інше значення, наприклад: eps=0.0001. А змінні i та j можна використовувати як індекси у циклах for. Однак це може спричинити плутанину, якщо всередині циклу користувач задасть вирази з комплексними числами. Щоби уникнути цього, можна використовувати як індекси I та J замість i та j.

Текстові коментарі

Оскільки MATLAB використовується для достатньо складних обчислень, важливе значення має наочність їхнього описання. Вона досягається, зокрема, за допомогою текстових коментарів. *Текстові коментарі* позначаються символом “%”, наприклад:

```
% Bit is factorial function
```

Зауважимо, що можна, але не бажано вводити неангломовні коментарі, оскільки це інколи може зробити програму недієздатною.

Змінні і надання їм значень

Змінні – це іменовані об’єкти для зберігання числових, символьних, векторних чи матричних даних. Для надання змінним значень використовують операцію присвоювання “=”. Типи змінних заздалегідь не декларуються. Вони визначаються виразом, значення якого присвоюється змінній. Так, якщо цей вираз – вектор чи матриця, то змінна буде векторною чи матричною.

Ім'я змінної (її ідентифікатор) має бути унікальним, тобто не збігатися з іменами інших змінних, функцій та процедур системи, та може містити до 31 символів: літер, цифр і символу підкреслення "_", але починатися має лише з літери. Для символьних змінних їхні значення записуються в апострофах, наприклад: s = 'Demo'.

Оператори і функції

Оператор – це спеціальне позначення певної операції над даними – операндами. Найпростішими арифметичними операторами є знаки суми "+", віднімання "–", множення "*" і ділення "/". Оператори використовуються разом з операндами. Приміром, у виразі 2 + 3 знак "+" є оператором додавання, а числа 2 і 3 – операндами.

Зауважимо, що більшість операторів у MATLAB відносяться до матричних операцій, що може спричинити чималі непорозуміння. Приміром, оператори множення "*" та ділення "/" обчислюють добуток і частку від ділення двох багатовимірних масивів, векторів чи матриць. Існує низка спеціальних операторів, приміром, оператор "\" означає ділення справа наліво, а оператори "**.***" та "**./**" означають відповідно поелементне множення та поелементне ділення масивів.

Пояснимо сказане на прикладах операцій з векторами:

```
» V1 = [2   4   6   8]
    V1 = 2 4 6 8
» V2 = [1   2   3   4]
    V2 = 1 2 3 4
 » V1/ V2
    ans = 2
» V1.*V2
    ans = 2   8   18   32
» V1./ V2
    ans = 2   2   2   2
```

Повний список операторів можна побачити, виконавши команду » help ops.

Функції можуть бути *вбудованими* (внутрішніми) і *зовнішніми*, які містять свої визначення в m-файлах. Вбудовані функції, наприклад, sin(x), exp(y), зберігаються у відкомпільованому ядрі системи MATLAB, а тому вони виконуються максимально швидко.

Зі списком елементарних функцій можна ознайомитись, виконавши команду help elfun, а зі списком спеціальних функцій – за допомогою команди help specfun.

*Оператор послідовності "***:***"* (двокрапка) використовують для формування упорядкованих числових послідовностей, наприклад, при створенні векторів або для значень абсциси при побудові графіків.

Формат оператора:

початкове_значення **:** *крок* **:** *кінцеве_значення*

Ця конструкція утворює зростаючу послідовність чисел, яка розпочинається від початкового значення, збільшується з заданим кроком і завершується кінцевим значенням. Якщо крок не задано, то він набуває значення 1. Якщо кінцеве значення задано меншим за початкове, – виведеться повідомлення про помилку. Приклади використання:

```
» 1 : 5
ans =
    1   2   3   4   5
» i = 0 : 2 : 10
    i = 0   2   4   6   8   10
» j = 10 : –2 : 2
    j = 10   8   6   4   2
» V = 0 : pi/2 : 2*pi
    V = 0   1.5708   3.1416   4.7124   6.2832
» X = 1 : –.2 : 0
    X = 1.0000   0.8000   0.6000   0.4000   0.2000   0
» 5 : 2
    ans = Empty matrix: 1 – by – 0
```

Як зазначалось, належність MATLAB до матричних систем вносить корективи у призначення операторів і за невмілого їхнього використання спричиняє казуси. Розглянемо характерний приклад:

```
» x = 0 : 5
    x = 0 1 2 3 4 5
» cos(x)
ans = 1.0000 0.5403 –0.4161 –0.9900 –0.6536 0.2837
» sin(x)/x
    ans = –0.0862
```

Обчислення масиву косинусів тут відбулося коректно. А ось обчислення масиву значень функції sin(x)/x дає несподіваний, на перший погляд, ефект – замість масиву з шести елементів обчислено лише одне значення! Причина "парадокса" полягає в тому, що оператор "/" обчислює відношення двох матриць, векторів чи то багатовимірних масивів. Якщо вони одного розміру, то результат буде одним числом, що в такому разі і видала система. Щоб дійсно здобути вектор значень sin(x)/x, треба використовувати спеціальний оператор поелементного ділення масивів – "./":

```
» sin(x)./x
Warning: Divide by zero.
ans = NaN   0.8415   0.4546   0.0470   –0.1892   –0.1918
```

Проте, і тут без особливостей не обійшлось. Оскільки, для x = 0 значення sin(x)/x утворює невизначеність при спробі ділення на 0, MATLAB виводить відповідне попередження і символьну константу NaN.

Спеціальні символи

Охарактеризуємо спеціальні символи MATLAB:

: двокрапка – формування підвекторів і підматриць з векторів і матриць. Цей оператор – один з найбільш використовуваних операторів у системі MATLAB. Він має такі правила для створення векторів:
j:k – те саме, що й [j, j+1, ..., k]; j:k – пустий вектор за умови j>k;
j:i:k – те саме, що й [j, j+i, j+2i ..... k];
j:i:k – пустий вектор за умов i>0 та j>k чи якщо i<0 та j<k, де 1, j, k – скалярні величини.
За допомогою цього оператора можна вибирати рядки, стовпці та елементи з векторів, матриць і багатовимірних масивів:
A(:, j) – це j-тий стовпець з A; A(i, :) – це i-тий рядок з A;
A(:, :) – еквівалент двовимірного масиву;
A(j : k) – це A(j), A(j+l),...,A(k);
A(: j : k) – це A(:, j), A(:, j+l), ..., A(:, k);
A(:, :, k) – це k-та сторінка тривимірного масиву A;
A(i, j, k, :) – вектор, вибраний з чотиривимірного масиву A з елементами: A(1, j, k, 1), A(i, j, k, 2), A(i, j, k, 3) тощо;
A(:) – виводить усі елементи масиву A у стовпчик;

( ) круглі дужки – застосовують, коли задають порядок виконання операцій в арифметичних виразах, зазначають послідовність аргументів функції чи коли записують індекси елемента вектора чи матриці;

[ ] квадратні дужки – використовують при формуванні векторів і матриць, наприклад:
[6.9 9.64 sqrt(–1)] – вектор з трьох елементів;
[6.9 9.64 i] – такий самий вектор;
[1+j 2–j 3] та [1 +j 2 –j 3] – різні вектори: перший містить три елементи, а другий – п’ять;
[11 12 13; 21 22 23] – матриця розміром 2 x 3. Крапка з комою відокремлює перший рядок від другого;
A = [ ] – зберігає пусту матрицю в A;
A(m, :) = [ ] – видаляє рядок m з матриці A;
A(:, n) = [ ] – видаляє стовпець n з матриці A.

{ } фігурні дужки – застосовують для формування комірок масивів. Наприклад, {magic(3) 6.9 'hello'} – масив комірок з трьома елементами;

. по-перше, крапка використовується для відокремлення цілої та дійсної частин десяткових чисел, наприклад: 314/100, 3.14 и .314e1 – одне й те саме число. По-друге, символ крапки (.) використовується для виокремлення полів структур, наприклад: A.(field) та A(i).field, де A – структура, означає виокремлення поля структури з ім’ям field;

.. батьківський каталог – перехід по дереву каталогів на один рівень уверх;

... продовження рядка – три чи більше крапок наприкінці рядка зазначають його продовження;

, кома, розділювач – використовується для розмежування індексів елементів матриці й аргументів функції, а також для розмежування операторів мови MATLAB. При розмежуванні операторів у рядку, кома може замінюватись на крапку з комою (**;**) з метою заборони виведення на екран результату обчислень;

; крапка з комою – використовується всередині круглих дужок для розмежування рядків матриць, а також наприкінці операторів для заборони виведення на екран результату обчислень;

% коментар – використовується для зазначення логічного кінця рядка. Текст після знака процента сприймається як коментар та ігнорується (на жаль, за винятком російськомовних коментарів, які частенько призводять до помилкових команд);

! знак оклику є ознакою введення команди операційної системи. Рядок, який іде за ним, сприймається як команда операційної системи;

= присвоювання значень в арифметичних виразах;

' одинарна лапка, апостроф – текст у таких лапках інтерпретується як вектор символів з компонентами, яві є ASCII-кодами символів. Лапка усередині рядка задається двома лапками, наприклад:

```
» a = 'Hello''my friend'
  a = Hello'my friend;
```

' транспонування з комплексним сполученням – транспонування матриць, приміром A' – транспонована матриця A. Для комплексних матриць транспонування виконується комплексним сполученням. Рядки транспонованої матриці відповідають стовпцям початкової матриці;

.' транспонування – транспонування масиву, приміром A.' – транспонований масив A. Для комплексних масивів операція сполучення не виконується;

[.] горизонтальна конкатенація. Так, [A.B] – горизонтальна конкатенація (зчіплювання) матриць A та B. A та B повинні мати однакову кількість рядків. [A B] діє аналогічно. Горизонтальна конкатенація може бути застосована для будь-якої кількості матриць у межах одних дужок: [A,B,C]. Горизонтальна і вертикальна конкатенації можуть використовуватись одночасно: [A,B:C];

[:] вертикальна конкатенація. Так, [A:B] – вертикальна конкатенація (зчіплювання) матриць A та B. A та B повинні мати однакову кількість стовпців. Вертикальна конкатенація може бути застосована для будь-якої кількості матриць у межах одних дужок: [A:B:C]. Горизонтальна і вертикальна конкатенації можуть використовуватись одночасно: [A:B,C];

( ),{ } присвоювання підмасиву. Наведемо декілька прикладів:
A(i)=B – присвоює значення елементів масиву B елементам масиву A, які визначаються вектором індексів i. Масив B повинен мати таку саму розмірність, що і масив i, або може бути скаляром;
A(i,j)=B – присвоює значення масиву B елементам прямокутної під-

матриці A, які визначаються векторами індексів i та j. Масив B повинен мати length(i) рядків і length(j) стовпців;
A{i}=B – присвоює визначеній значенням скаляром I заданій комірці масиву A копію масиву B.

Формати числових даних у MATLAB

За замовчуванням MATLAB видає числові результати у нормалізованій *формі* з чотирма цифрами після десяткової крапки й однією до неї. Багатьох така форма подання не завжди влаштовує. Тому при роботі з числовими даними можна задавати різноманітні формати подання чисел. Однак у будь-якому разі усі обчислення здійснюються з подвійною точністю.

Формати чисел:

short — короткий формат, чотири знаки після крапки, наприклад: 2.1386;
long — довгий формат, 15 цифр після крапки, наприклад: 2.1385765798124346);
short e — короткий формат в експоненціальній формі, 5 цифр мантиси й 3 цифри порядку: 2.1386e–003 (0.0021386);
long e — довгий формат в експоненціальній формі, 15 цифр мантиси й 3 цифри порядку: 2.1386e–003 (0.0021386);
short g, long g — універсальні формати. Наприклад, формат short g: 0.000013564 (1.3564e–005), 25.648 (2.5648e–001)
hex — подання чисел у шістнадцятковій формі;
bank — подання для грошових одиниць.

Для того, щоби змінити формат, використовується оператор format, наприклад:

```
» format short
» format long
```

Для ілюстрації різних форматів розглянемо вектор з двох елементів-чисел:

```
x = [4/3   1.2345e-6]
```

У різних форматах їхнє подання будуть мати такий вигляд:

format short	1.3333	0.0000
format short e	1.3333E+000	1.2345E-006
format long	1.333333333333338	0.000001234500000
format long e	1.333333333333338E+000	1.234500000000000E-006
format bank	1.33	0.00

Зазначення формату впливає лише на формі виведення чисел. Обчислення насправді здійснюються у форматі подвійної точності, а введення чисел можливе у будь-якому зручному для користувача вигляді.

Інтервальний тип даних (вектори)

Інтервальний тип даних у MATLAB схожий на такий самий тип даних у Mathcad, але відрізняється синтаксисом:

Ім'я_змінної = *початкове_значення* **:** *крок* **:** *кінцеве_значення*

Наприклад, створимо у MATLAB вектор x зі значеннями від 1 до 4 з кроком 0.5:

```
»x = 1 : 0.5 : 4
```

Після виконання цієї команди здобудемо вектор x = [1 1.5 2 2.5 3 3.5 4].

Зауважимо, що якщо крок змінення інтервальної змінної дорівнює 1, то його можна не зазначати, наприклад команда

```
»n = 1 : 10
```

створить вектор n = [1 2 3 4 5 6 7 8 9 10].

1.3 Основи редагування і відлагодження m-файлів

MATLAB дозволяє записувати послідовність дій програми в окремих m-файлах. Підготовлений і записаний на диску m-файл стає частиною системи, і його можна викликати як з командного рядка, так і з іншого m-файла. При створенні m-файли проходять синтаксичний контроль за допомогою вбудованого в систему MATLAB редактора / відлагоджувача m-файлів.

Існує два різновиди m-файлів: файли-функції та файли-сценарії:

✓ *файли-функції*, що мають вхідні параметри, список яких зазначається у круглих дужках. Застосовувані у файлі-функції змінні є локальними змінними;

✓ *файл-сценарій* чи Script-файл є простим записом послідовності команд та операторів для розрахунків.

Для запуску Script-файла на виконання з командного рядка MATLAB достатньо записати його ім’я в цьому рядку:

>> ім’я_файла.m

Для того щоби створити файл слід виконати команди з меню *File / New*, після чого відкриється вікно створення й редагування текстів програм, в якому послідовно записують і виконують програмні команди (рис. 1.2).

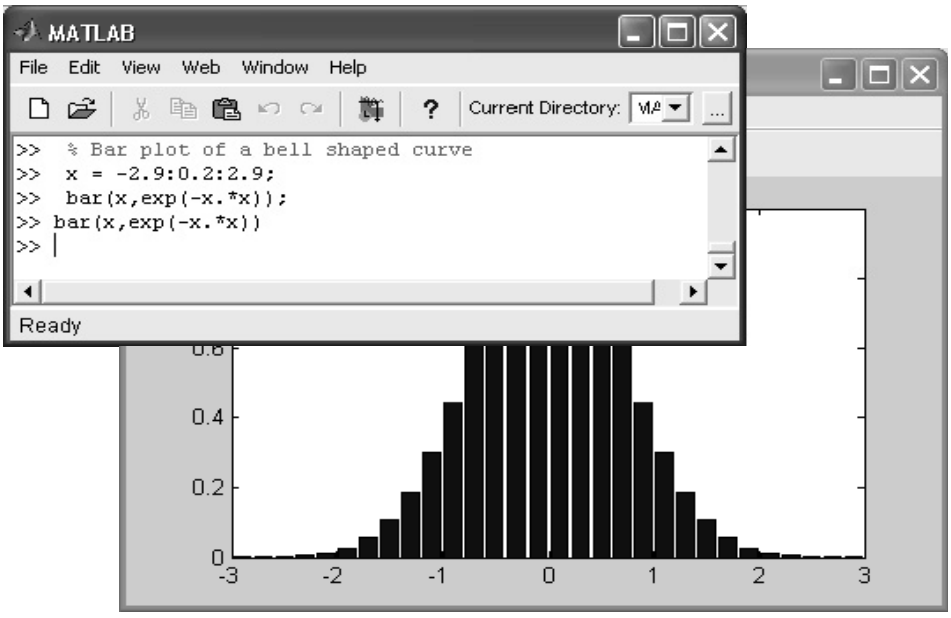


Рисунок 1.2 − Приклад виконання файла-сценарія з командного рядка

Редактор/відлагоджувач m-файлів виконує синтаксичний контроль програмного коду в міру введення тексту. При цьому використовуються такі кольорові виокремлення:

- ✓ ключові слова мови програмування – синій колір;
- ✓ оператори, константи і змінні – чорний колір;
- ✓ коментарі після знаку % – зелений колір;
- ✓ символьні змінні (в апострофах) – зелений колір;
- ✓ синтаксичні помилки – червоний колір.

Завдяки кольоровим виокремленням вірогідність синтаксичних помилок знижується. Однак далеко не всі помилки діагностуються. Помилки, пов'язані з неправильним застосування операторів чи функцій (приміром, застосування оператора "–" замість "+" чи функції cos(x) замість sin(x) тощо), не здатна виявити жодна система програмування. Усунення такого роду помилок (їх називають семантичними) – справа користувача, що відлагоджує свої алгоритми і програми.

1.4 Візуалізація і графічні засоби

Останнім часом творці математичних систем приділяють величезну увагу візуалізації розв'язування математичних задач. Простіше кажучи, це означає, що постановка й опис розв'язуваної задачі та результати розв'язання мають бути ґранично зрозумілими не лише тим, хто розв'язує задачі, а й тим, хто надалі їх вивчає чи просто переглядає. Значну роль у візуалізації розв'язування математичних задач відіграє графічне подання результатів, причому як кінцевих, так і проміжних. Візуалізація результатів обчислень досягається застосуванням потужних засобів графіки, у тому числі анімаційної, а також використанням засобів символьної математики.

Графічні засоби Handle Graphics (дескрипторна чи описова графіка) дозволяють створювати повноцінні об'єкти графіки високого дозволу, як геометричного, так і колірного. Можливості цієї графіки підтримуються засобами об'єктно-орієнтованого програмування системи MATLAB. Реалізується, до того ж з підвищеною швидкістю, побудова графіків практично усіх відомих у науці й техніці типів. Графіки виводяться окремо від текстів в окремих вікнах. На одному графіку можна вивести декілька кривих, які відрізняються кольорами і виглядом маркерів (кружки, хрестики, прямокутники тощо). Графіки можна виводити до одного чи декількох вікон.

Побудова графіків відрізками прямих

Функції однієї змінної y(x) знаходять широке застосування у практиці математичних та інших розрахунків, а також у комп'ютерному математичному моделюванні. Для відображення таких функцій використовуються графіки у декартовій (прямокутній) системі координат. При цьому, зазвичай, будуються дві осі – горизонтальна X та вертикальна Y – і задаються координати x та y, які визначають вузлові точки функції y(x). Ці точки з'єднуються одна з одною від-

різками прямих, тобто при будові графіка здійснюється лінійна інтерполяція для проміжних точок. Оскільки MATLAB – матрична система, сукупність точок y(x) задається векторами X та Y однакового розміру.

Команда plot призначена для створення графіків функцій у декартовій системі координат. Розглянемо можливі параметри цієї команди на прикладах.

1) plot(X, Y) – будує графік функції y(x), координати точок (x, y) беруться з векторів однакового розміру Y та X. Якщо X чи Y – матриця, то будується сімейство графіків за даними колонок матриці. Наведений нижче приклад ілюструє побудову графіків двох функцій – sin(x) та cos(x), значення яких містяться у матриці Y, а значення аргумента x зберігаються у векторі X:

```
» x = [0 1 2 3 4 5];
» Y = [sin(x)  cos(x)];
» plot(x, Y)
```

На рис. 1.3 наведено графік функцій з цього приклада. На ньому чітко видно, що графіки складаються з відрізків.

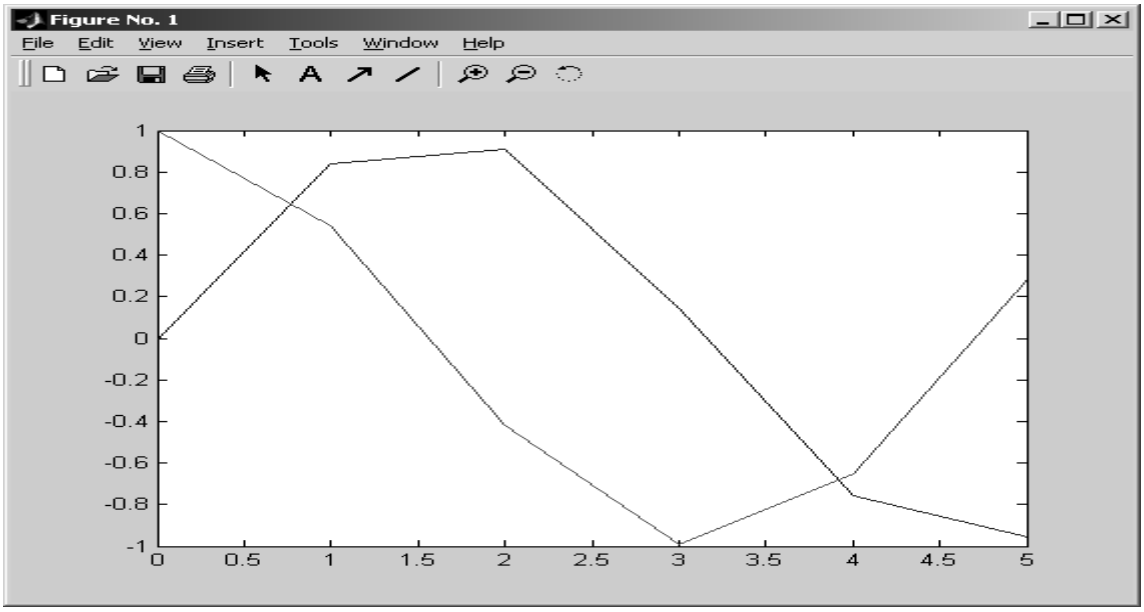


Рисунок 1.3 − Графіки двох функцій у декартовой системі координат

Якщо є потреба у відображенні функції у вигляді гладкої кривої, слід збільшити кількість вузлових точок, наприклад, задати x = 0 : 0.01 : 5 (рис. 1.4).

2) plot(y) – будує графік y(n), де значення аргумента n це індекси елементів вектора y, а вісь абсцис являє собою відповідного елемента. Якщо y містить комплексні елементи, то виконується команда plot(real(y), imag(y)). У решті випадків уявна частина даних ігнорується. Приклад використання команди plot(y):

```
» x = 2*pi : 0.02*pi : 2*pi;
» y = sin(x) + i*cos(3*x);
» plot(y)
```

Відповідний графік наведено на рис. 1.5.

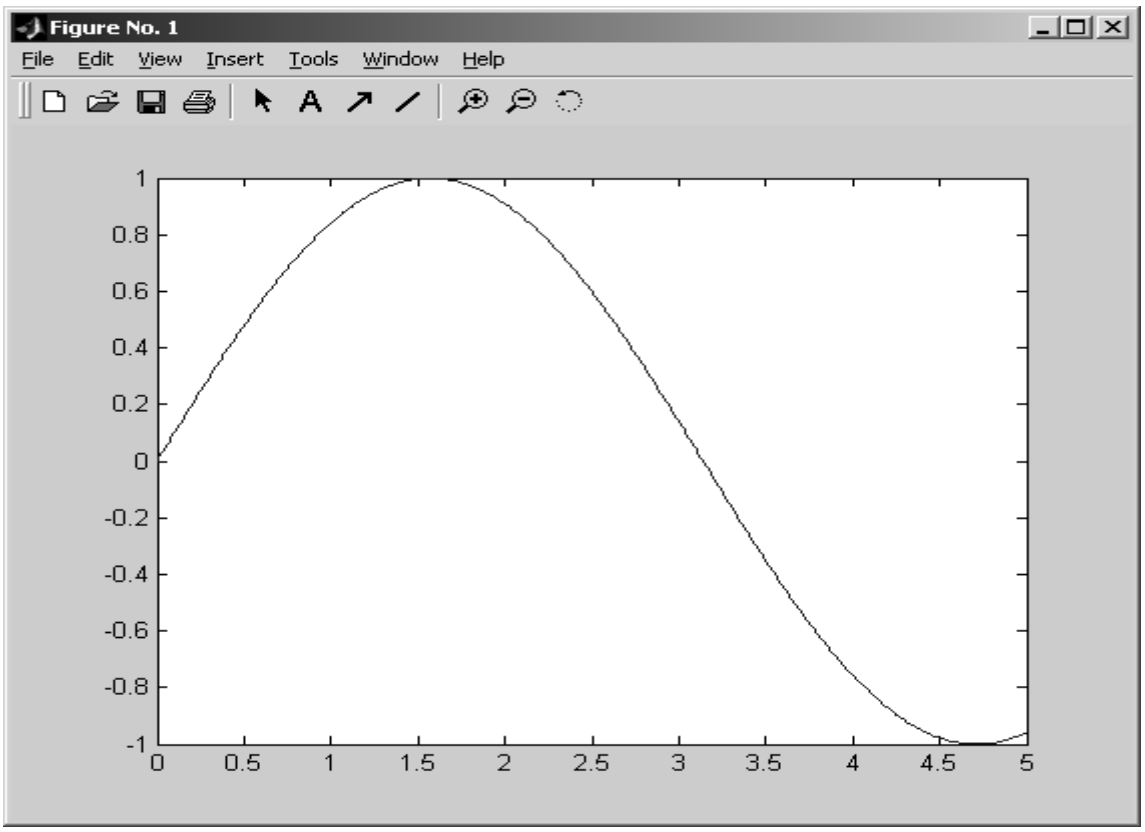


Рисунок 1.4 − Графік неперервної функції

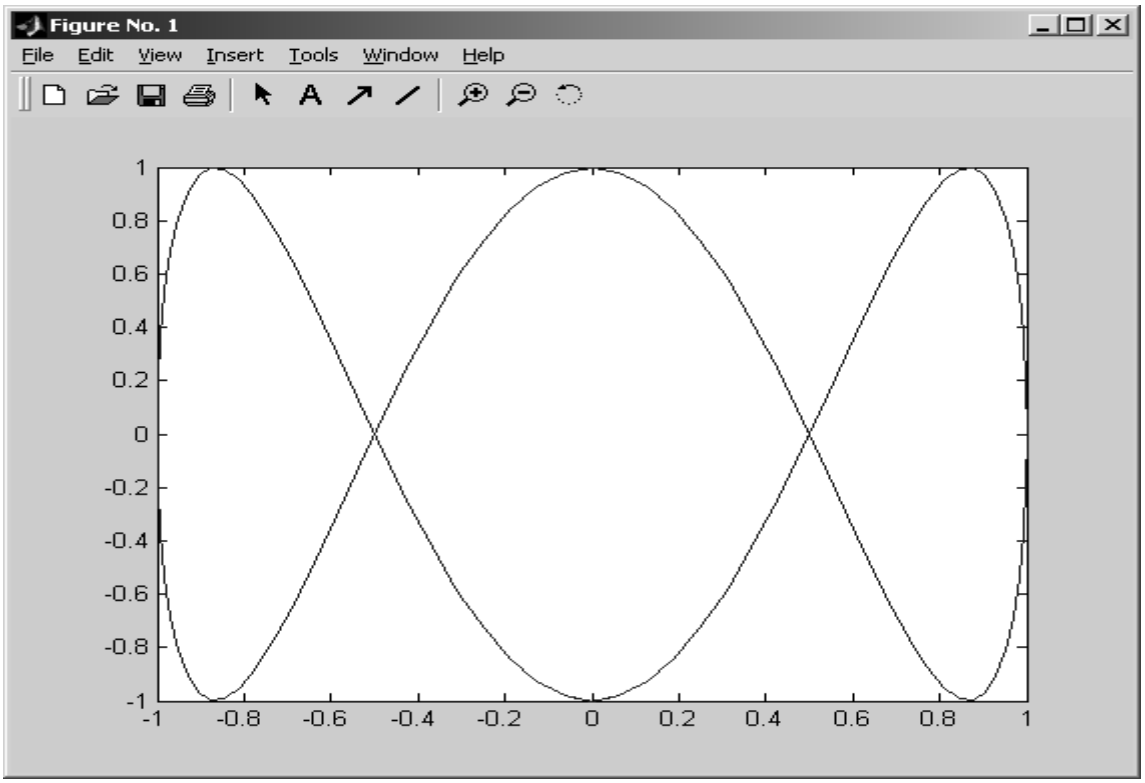


Рисунок 1.5 − Графік функції, що зображує вектор y з комплексними елементами

3) plot(X, Y, S) – є аналогічна до команди plot(X, Y), проте додатково задається тип лінії графіка значенням константи S (рядок символів):

Символ	Колір лінії
y	жовтий
m	фіолетовий
c	блакитний
r	червоний
g	зелений
b	синій
w	білий
k	чорний

Символ	Тип маркера
.	точка ●
o	кружок ○
x	хрестик ×
+	плюс +
*	зірочка ✶
s	квадрат ■
D	ромб ◆
V	трикутник ▼
A	трикутник ▲
<	трикутник ◀
>	трикутник ▶
P	п’ятикутник ⬟
H	шестикутник ⬢

Символ	Тип лінії
-	суцільна
;	подвійний пунктир
-.	пунктирна
--	штрихова

Отже, за допомогою набору з певних символів третім параметром команди plot можна змінювати тип і колір лінії, задавати вигляд маркера для вузлових точок.

4) plot (X1, Y1, S1, X2, Y2, S2, X3, Y3, S3, ...) – ця команда будує на одному графіку декілька ліній, поданих даними вигляду (X, Y, S), де X та Y – вектори чи матриці, а S – рядок певних символів. За допомогою такої конструкції можна побудувати графік функції лінією, колір якої відрізняється від кольору маркерів. Приміром, щоб побудувати графік функції лінією синього кольору з червоними маркерами, спочатку треба побудувати графік з маркерів червоного кольору (без лінії), а потім – графік лінії синього кольору (без маркерів). За відсутності вказівок на кольори ліній та маркерів, вони обираються автоматично з таблиці кольорів (за винятком білого). Якщо ліній на одному графіку є більше шести, то кольори будуть повторюватись. Для монохромних систем лінії задають різними типами. Розглянемо приклад побудови графіків трьох функцій з різним стилем подання кожної з них (рис. 1.6):

```
» x = –2*pi : 0.1*pi : 2*pi;
» y1 = sin(x);
» y2 = sin(x).^2;
» y3 = sin(x).^3;
» plot(x, y1, '-m', x, y2, '-.+r', x, y3, '--ok')
```

Як видно з рисунку, графік функції y1 будується суцільною фіолетовою лінією, графік y2 будується пунктирною лінією із маркерами у вигляді знака “плюс” червоного кольору, а графік y3 будується штриховою лінією з кружками чорного кольору. Слід враховувати, що при роздруковуванні на чорно-білих принтерах на папері кольорові лінії можуть зображуватись дуже схожими між собою градаціями сірих кольорів.

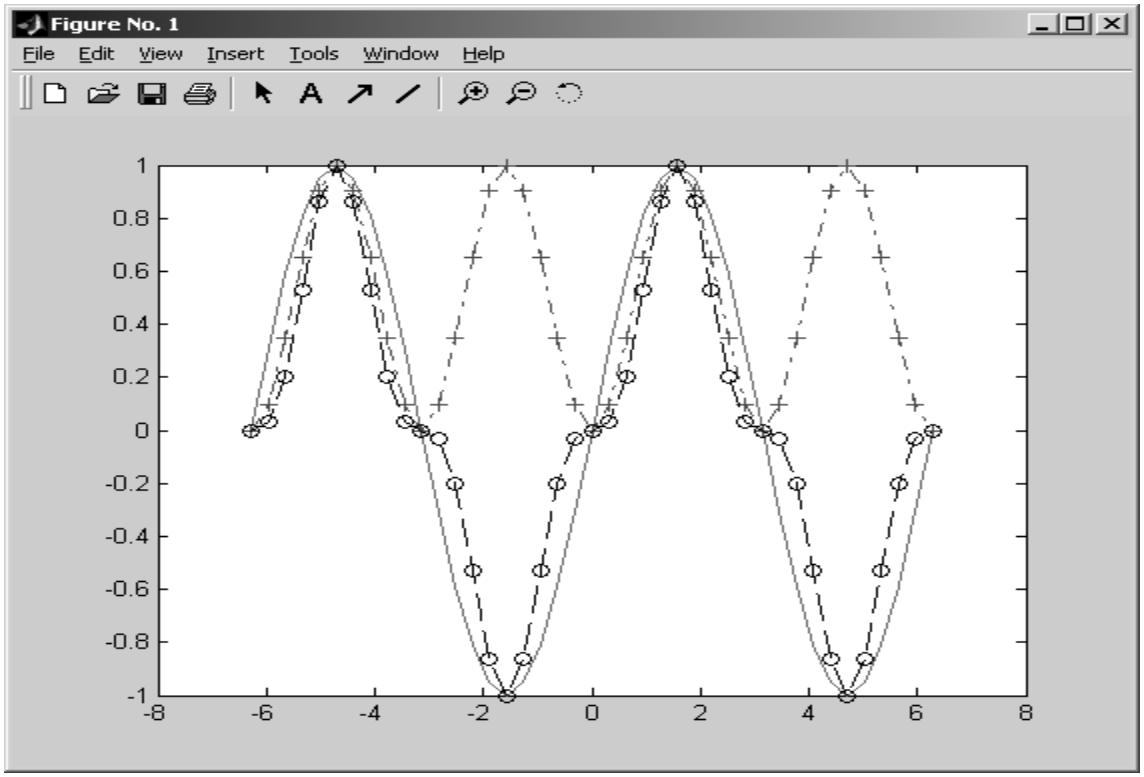


Рисунок 1.6 − Вигляд трьох функцій на одному графіку з різними стилями ліній

1.5 Статистичні функції MATLAB

1) sum(x) – обчислює суму елементів стовпців матриці, (якщо x – вектор, обчислюється сума елементів вектора $\sum_{i=1}^{n} x_i$). Наприклад, задамо значення матриці A та обчислимо суми елементів рядків S і суми елементів стовпців S1:

```
»A = [ 1 2 3;  4 5 6;  7 8 9 ]
   A=[1  2  3
      4  5  6
      7  8  9]
»S = sum(A)       %  вектор сум елементів стовпців
      S  =  [12  15  18]
»S1= sum(A')      %  вектор сум елементів рядків
      S1  =  [6  15  24]
```

2) mean(x) – обчислює середні значення елементів стовпців матриці, а якщо x – вектор з *n* елементів, то обчислюється $m_x = \frac{1}{n}\sum_{i=1}^{n} x_i$. Наприклад:

```
» MA=mean(A)
      MA  =  [4  5  6]
» MS=mean(S)
      S=  15
```

3) min(x) – обчислює мінімальне значення вектора чи елементів стовпців матриці, наприклад для матриці A :

```
» Mn= min (A)
  Mn = [1 2 3]
```

4) max(x) – максимальне значення вектора чи елементів стовпців матриці, наприклад для матриці A:

```
» Mx= max(A)
  Mx= [7 8 9]
```

5) std(x) – стандартні (середньоквадратичні) відхилення значень елементів стовпців матриці. Якщо x – вектор з n елементів, то стандартне відхилення (δ_s) та середньоквадратичне (δ_k) відхилення значень елементів стовпців обчислюються за формулами:

$$\delta_s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - m_x)^2} \quad ,\delta_k = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - m_x)^2} \; .$$

Наприклад, обчислимо стандартне відхилення (δ_s) для вектора S:

```
» Ssv= std (S)
  Ssv = 3
```

Зауважимо, що для обчислення середньоквадратичного відхилення (δ_s) функцію std записують з додатковим параметром (1), наприклад:

```
» Ssv= std (S,1)
  Ssv=2.4495
```

6) sort(y) – функція *сортування за зростанням*. Наприклад:

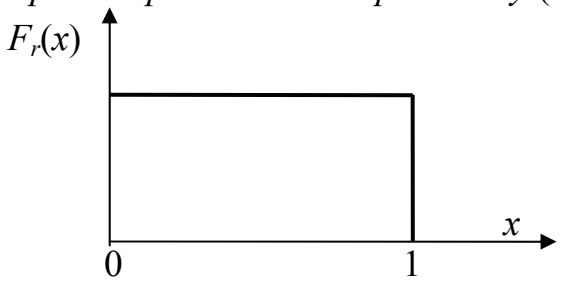
```
»y =  [3 1 8 4 5];
»w = sort(y)
      w = [1 3 4 5 8]
```

Для *сортування за спаданням* запишемо:

```
»b = − sort(-y)
      b = [8 5 4 3 1]
```

1.6 Формування випадкових чисел

1) x = rand(n) формує матрицю x розміром $n \times n$ випадкових чисел від −1 до 1 з *рівномірним законом розподілу* (на рисунку − функція розподілу $F_r(x)$)



Варіанти використання цієї функції:

rand(n, m) – формує матрицю випадкових чисел з n рядків та m стовпців;

rand(n, 1) – створює стовпець з n випадкових чисел;

rand(1, n) – створює рядок з n випадкових чисел.

Можна змінювати початкове значення, з якого починається відлік випадкових чисел за допомогою функції rand('seek', k), де k – велике ціле непарне число. Наприклад, створимо стовпець зі 100 випадкових чисел із значеннями в проміжку [−1, 1] за рівномірним законом розподілу:

```
»rand('seek' ,10001); x = rand(100, 1);
```

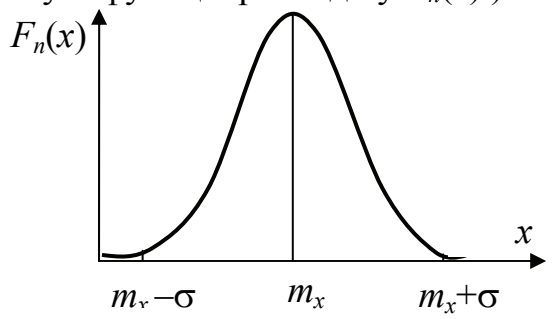
Для створення вектора випадкових чисел y з рівномірним розподілом зі значеннями в проміжку [a, b] необхідно записати команди:

```
»x = rand(n, 1); y=a+(b-a).*x;
```

2) randperm(n) – формує масив з *випадковою перестановкою n* цілих чисел:

```
»x = randperm(6)
      x = [5 3 1 4 2 6]
```

3) x =randn(n, m) – створює матрицю випадкових чисел за *нормальним законом розподілу* із математичним очікуванням нуль (0) і середньоквадратичним відхиленням одиниця (1). Вектор чисел за нормальним законом розподілу з заданими математичним очікуванням m_x і середньоквадратичним відхиленням σ (на рисунку − функція розподілу $F_n(x)$) обчислюється за формулою:

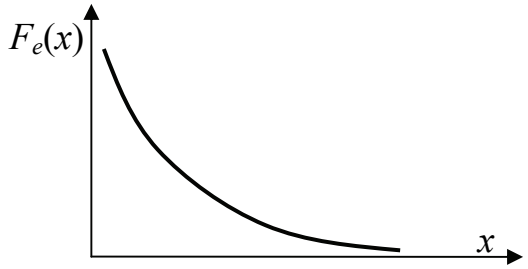


$$y = m_x + \sigma * x$$

Для створення вектора випадкових чисел y з нормальним законом розподілу із значеннями $m_x=mx$ та $\sigma=sx$ необхідно записати команди:

```
»x = randn(n, 1); y=mx+sx.*x;
```

4) вектор випадкових чисел *z* за *експоненціальним (Пуассона) законом* розподілу (на рисунку − функція експоненціального розподілу $F_e(x)$) із заданим середнім значенням m_x обчислюється за формулою:



$$z = -m_x \ln(1 - x),$$

де x − вектор випадкових чисел із значеннями в проміжку [−1, 1] за рівномірним законом розподілу. Команди формування вектору за експоненціальним законом розподілу із значенням $m_x=mx$:

```
»x = rand(n, 1); z = –mx.* ln(1–x);
```

Л а б о р а т о р н а р о б о т а № 1

Обчислення функцій та побудова графіків у середовищі MATLAB. Елементи моделювання в Simulink

1 ***Арифметичні обчислення у MATLAB*:**

- Обчисліть суму, добуток будь-яких двох чисел.
- Перегляньте список елементарних математичних функцій за допомогою команди help elfun).
- Обчисліть значення функції y = F(x) за заданими формулами та значеннями змінних згідно до варіанту індивідуального завдання 1, наведеного у табл. 1.
- Перегляньте список змінних робочої області.
- Збережіть результати попередніх обчислень, використовуючи команду save.
- Очистіть робочу область даних за допомогою команди clear.
- Завантажте значення змінних з файла до робочої області за допомогою команди load.

2 ***Обчислення з векторами й матрицями у MATLAB***:

- Задайте вектор-рядок K як послідовність чисел від 1 до 4 з кроком 1.
- Задайте вектор-стовпець P як послідовність із 4 чисел.
- Визначте вектор суми K та P, їхній скалярний добуток.
- Створіть матрицю M цілих чисел 4x4.
- Обчисліть добуток матриці M на вектор-стовпець P.

3 ***Апроксимація та побудова графіків функцій у MATLAB:***

- Побудуйте графіки функцій (в окремих вікнах)

$$y1 = \sin(x), \qquad y2 = \frac{1}{x},$$

де x змінюється від 0.1 до 1 із кроком 0.05.

- Знайдіть коефіцієнти апроксимуючого поліному $g(x)$ 2-го, 3-го та 5-го степенів для векторів X та Y:

X	–2.1	–1	0.5	1	2.1	3.1
Y	0.6	0.1	–0.5	0.1	0.7	0.9

- Побудуйте графіки заданої функції Y(X) та апроксимуючої функції g(x) (де x змінюється від –2.1 до 3.1 із кроком 0.05) в одній графічній площині.

Методичні поради

а) Інтервальний тип даних у MATLAB схожий на такий самий тип даних у Mathcad, проте має інший синтаксис:

Ім'я_змінної = початкове_значення **:** *крок* **:** *кінцеве_значення*

Наприклад, для створення у MATLAB вектора *x* зі значеннями від 1 до 4 з кроком 0.5 треба записати команду

```
x =1 : 0.5 : 4
```

б) Функція plot(x, y) будує графік функції y(x), де x та y – вектори однакового розміру.

в) Функція figure(N) будує графік у вікні з номером N. За замовчуванням графік будується у вікні figure(1).

г) Для визначення коефіцієнтів апроксимуючого поліному n-го степеня можна використовувати функцію з назвою polyfit, команда, звертання до якої у пакеті MATLAB має вигляд:

```
a = polyfit(X, Y, n)
```

де X та Y – вектори значень заданої функції, n – бажана степінь апроксимуючого поліному, $a = [a_n, \ldots, a_2, a_1, a_0]$ – вектор значень коефіцієнтів поліному $g(x) = a_n x^n + \ldots + a_2 x^2 + a_1 x + a_0$.

д) Для обчислення значень апроксимуючого поліному g(x) можна використовувати функцію з назвою polyval, команда, звертання до якої у пакеті MATLAB має вигляд:

```
y = polyval(a, x)
```

де x – вектор значень аргументу функції, за якими маємо обчислити значення апроксимуючого поліному, a – заданий вектор значень коефіцієнтів апроксимуючого поліному, y – вектор значень апроксимуючого поліному.

5 *Побудова моделей у Simulink.*

- Змоделюйте суму й добуток двох чисел.
- Змоделюйте обчислення значення cos(2).
- Змоделюйте обчислення функції e^x при $x = 2$ (використати блок Math Function).
- Змоделюйте інтегрування константи.
- Змоделюйте інтегрування пилкоподібного сигналу з наступним його диференціюванням. Вивести графіки всіх трьох сигналів.
- Змоделюйте обчислення значень поліному $x^3 - 5x + 1$ при значеннях $x = 1,2,3$.

Методична порада

Для виклику Simulink натисніть на зелено-синьо-рожеву піктограму  у головному вікні MATLAB або напишіть команду

```
Simulink
```

Далі виконайте команди *File / New / Model* і створіть схему моделі, розташувавши за допомогою миші у певній послідовності потрібні блоки. Для моделювання у Simulink використайте блоки вхідних (Sources) і вихідних (Sinks) сигналів, блоки операцій над сигналами (Math Function, Continuous і Math Operation). Для створення схеми моделі оберіть та перетягніть мишкою потрібні блоки і з'єднайте їх лініями. Запуск моделі виконується командою *Simulation / Start*. Приклади схем моделей показано на рис. 1.7 … 1.11.

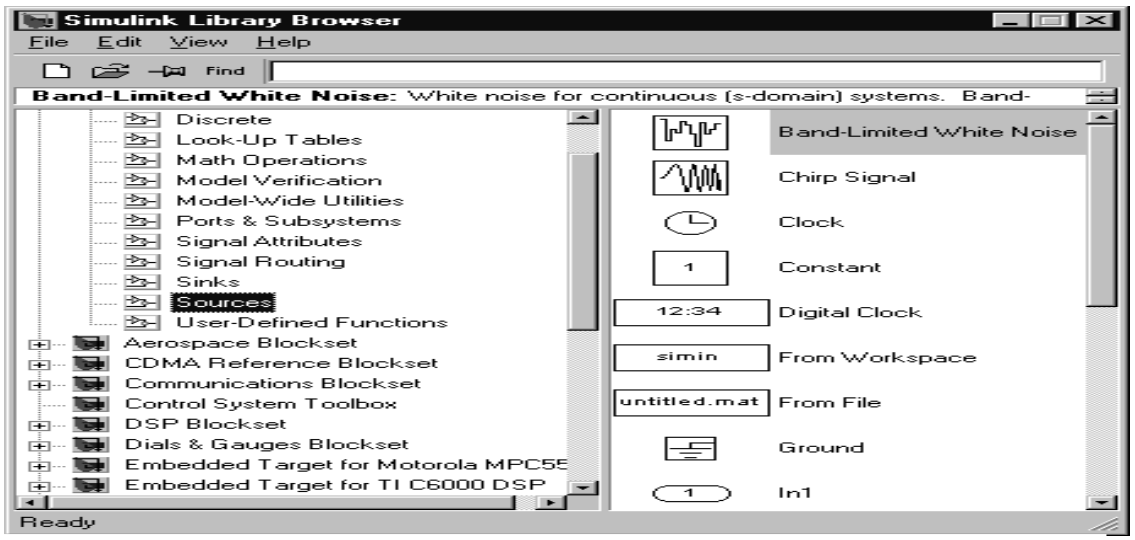


Рисунок 1.7 − Блоки вхідних сигналів у Simulink

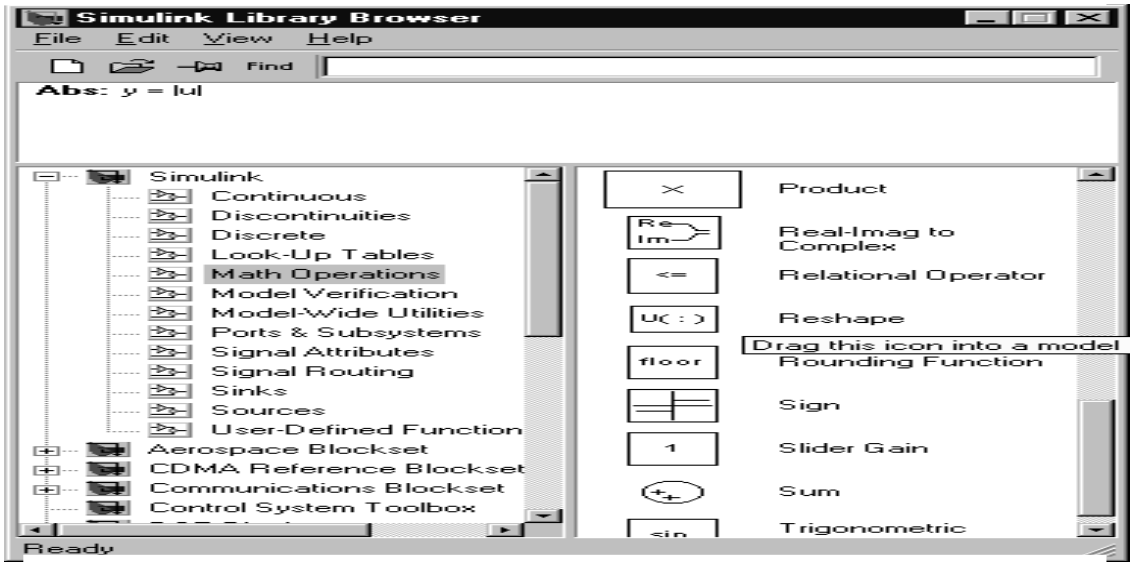


Рисунок 1.8 − Блоки математичних операцій у Simulink

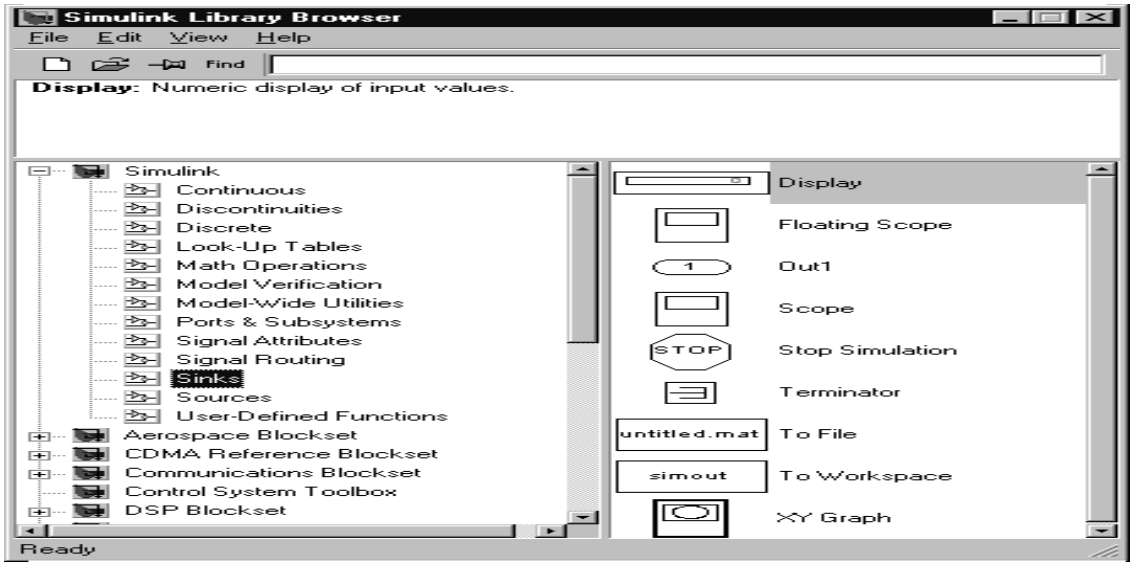


Рисунок 1.9 − Блоки вихідних сигналів у Simulink

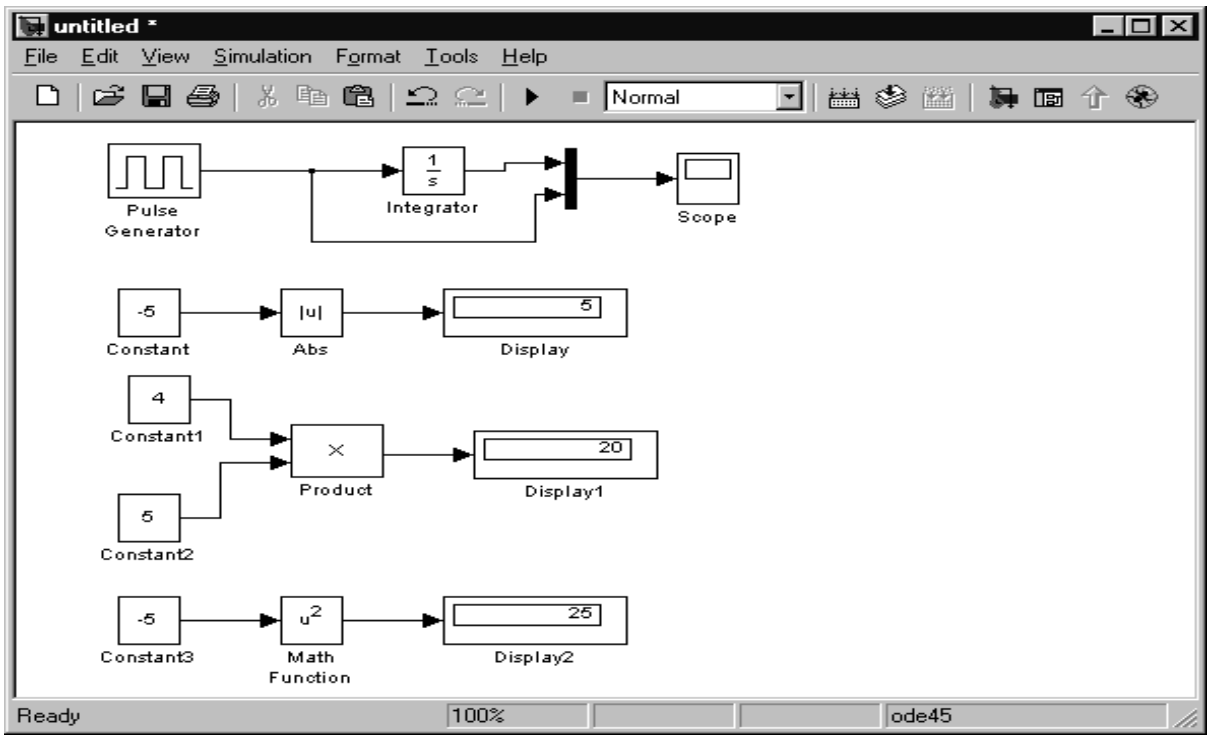


Рисунок 1.10 − Приклади обчислень у Simulink: інтегрованого прямокутного сигналу, обчислення модуля числа, добутку

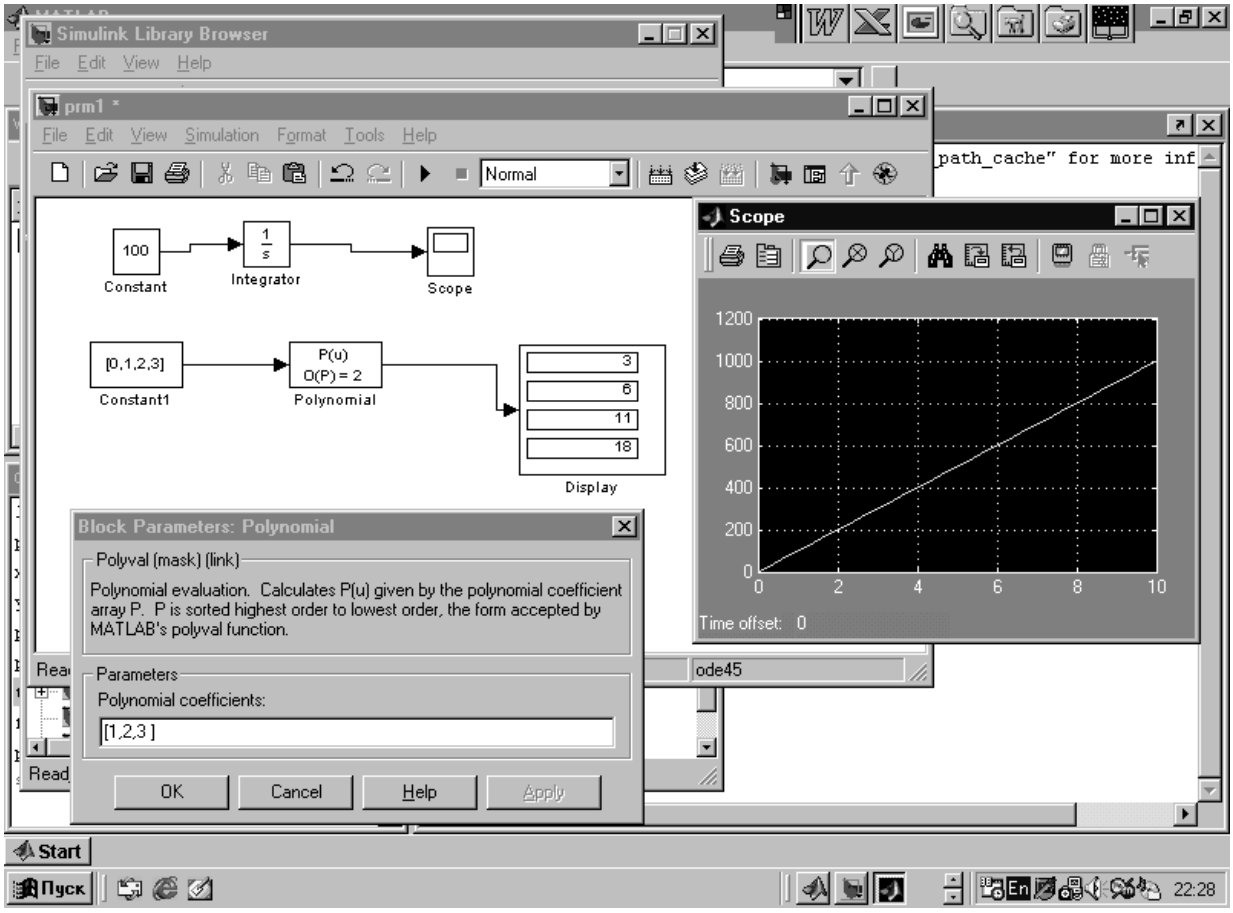


Рисунок 1.11 − Приклади обчислень у Simulink: інтегровані константи і обчислення значень поліному $x^2 + 2x + 3$ при значеннях $x = 0, 1, 2, 3$

2 РОЗВ'ЯЗАННЯ ДИФЕРЕНЦІЙНИХ РІВНЯНЬ ТА СИСТЕМ

2.1 Постановка задачі

Звичайні диференційні рівняння (ЗДР) широко використовуються для математичного моделювання процесів і явищ у різних галузях науки і техніки. Перехідні процеси у радіотехніці, кінетика хімічних реакцій, динаміка біологічних популяцій, рух космічних об'єктів, моделі економічного розвитку досліджуються за допомогою ЗДР.

Диференційне рівнянні n-го порядку має вигляд:

$$y^{(n)}=f\left(x, y, y', \ldots\ y^{(n-1)}\right)$$

$$\text{чи} \qquad \varphi(x, y, y', \ldots\ y^{(n)}) = 0. \tag{2.1}$$

де невідомими є функція $y(x)$ та її n похідних від x (y', ... $y^{(n)}$)

З теорії ЗДР відомо, що рівняння (2.1) є еквівалентне до *системи n рівнянь першого порядку*

$$\varphi_k\left(x, y_1, y_1', y_2, y_2', \ldots, y_n, y_n'\right) = 0 \tag{2.2}$$

де $k = 1, \ldots, n$.

Рівняння (2.1) і відповідна йому система (2.2) мають нескінченну множину розв'язків. Єдиний розв'язок виділяють за допомогою додаткових умов, які мають задовольняти рівнянню. Залежно від різновиду таких умов розглядають три типи задач, для яких доведено існування й одиничність розв'язку.

Перший тип – це задачі Коші чи задачі з початковими умовами. Для таких задач окрім рівняння (2.1) у деякій точці x_0 мають бути задані початкові умови, тобто значення функції $y(x)$ та її похідних

$$y(x_0) = y_0, \quad y'(x_0) = y_{10}, \ \ldots, \ y^{(n-1)}(x_0) = y_{n-1,0}.$$

Для системи ЗДР вигляду (2.2) початкові умови задаються у вигляді

$$y_1(x_0) = y_{10}, y_2(x_0) = y_{20}, \ \ldots\ , y_n(x_0) = y_{n0}. \tag{2.3}$$

До *другого типу задач* відносяться так звані граничні чи крайові задачі, для яких додаткові умови задаються у вигляді функціональних співвідношень між шуканими розв'язками. Кількість умов має збігатися з порядком n рівняння чи системи. Якщо розв'язування задачі визначено на інтервалі $x \in [x_0, x_k]$, то такі умови може бути задано як на межах, так і всередині інтервалу. Мінімальний порядок ЗДР, для яких може бути сформульовано граничну задачу, дорівнює двом.

Третій тип задач для ЗДР – це задачі на власні значення. Такі задачі відрізняються тим, що крім шуканих функцій $y(x)$ та їхніх похідних до рівняння входять додатково m невідомих параметрів $\lambda_1, \lambda_2, \ldots, \lambda_m$, які називаються власними значеннями. Для одиничності розв'язків на інтервалі $[x_0, x_k]$ треба задати

$m + n$ граничних умов. Як приклад, можна назвати задачі визначення власних частот, структури електромагнітних полів та механічних напруг у коливальних системах, задачі обчислення фазових коефіцієнтів (коефіцієнтів загасання) розподілу напруги полів хвильових процесів тощо.

До *чисельного розв'язування ЗДР* доводиться звертатися, коли не вдається здобути аналітичний розв'язок задачі через відомі функції. Хоча для деяких задач чисельні методи виявляються більш ефективними навіть за наявності аналітичних розв'язків. Більшість методів розв'язування ЗДУ засновано на задачі Коші, алгоритми й програми для якої буде далі розглянуто.

2.2 Методи Рунге – Кутти

Розглядання чисельних методів *розв'язування* диференційних рівнянь, розпочнемо з їхньої важливої категорії, відомої під загальною назвою методів Рунге – Кутти. Названі на честь німецьких математиків Карла Рунге і Мартіна Кутти, які відкрили ці методи.

Методи Рунге – Кутти мають такі *властивості*:

✓ Ці методи є одноступінчастими: щоби обчислити $y_{m+1}(x_{m+1})$, потрібна інформація про попередню точку (x_m, y_m), де m − номер ітерації обчислень.

✓ Ці методи майже збігаються з методом Тейлора аж до членів порядку h^P, де степінь p − є порядковий номер чи порядок методу (відрізняється для різних методів).

✓ Ці методи не вимагають обчислення похідних від $f(x, y)$, а вимагають обчислення самої функції.

Спочатку розглянемо *геометричну побудову* й виведемо деякі формули на основі геометричних аналогій, після чого підтвердимо здобуті результати аналітично (рис. 2.1).

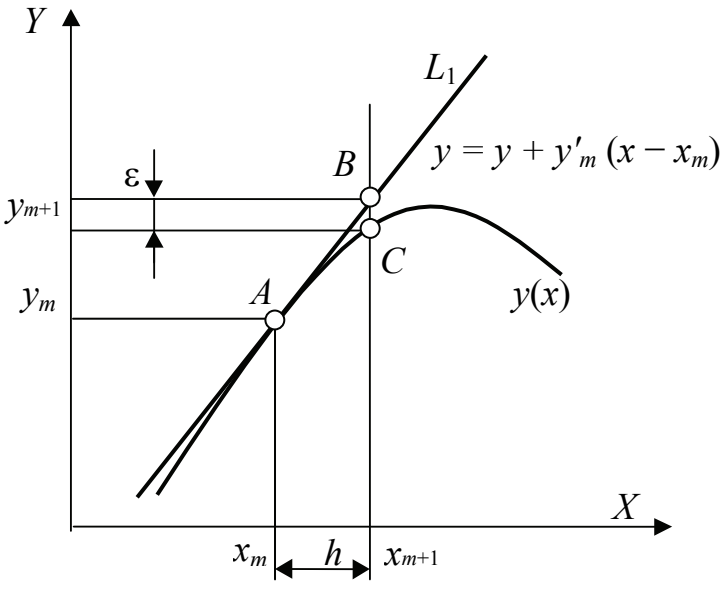


Рисунок 2.1 − Геометрична інтерпретація методу Рунге – Кутти першого порядку

Припустімо, що відомо точку A с координатами (x_m, y_m) на шуканій кривій. Тоді через цю точку можна провести пряму лінію з тангенсом кута нахилу $y'_m = f(x_m, y_m)$, яка пройде через точку A (x_m, y_m). На рис. 2.1 показано, що крива являє собою точний, але остаточно невідомий розв'язок рівняння, а пряму лінію L_1 побудовано так, як це тільки що описано. Тоді наступною точкою розв'язування можна вважати точку B, де пряма L_1 перетне ординату, проведену через точку $x_{m+1} = x_m + h$.

Рівняння прямої L_1 має вигляд:

$$y = y_m + y'_m(x - x_m), \tag{2.4}$$

оскільки за умовою $y' = f(x_m, y_m)$ і, крім того, $x_{m+1} = x_m + h$. Тоді рівняння (2.4) набуде вигляду

$$y_{m+1} = y_m + h*f(x_m, y_m) \tag{2.5}$$

Відхилення розв'язку в точці $x = x_{m+1}$ показано як відрізок ε. Вочевидь, віднайдене у такий спосіб наближене значення узгоджується з розкладанням у ряд Тейлора аж до членів порядку h, так що похибка розв'язку (відрізок BC на рис. 2.1) дорівнює $\varepsilon = k\,h^2$.

Формула (5) описує *метод Ейлера*. Це один з найпоширеніших і відомих методів чисельного інтегрування диференційних рівнянь. Зазначимо, що цей метод є методом Рунге – Кутти першого порядку.

Методи Рунге – Кутти другого та більших порядків можна вивести аналогічно до того, як це робилося при виведенні методу першого порядку. Не будемо тут відтворювати усі ці викладки, а обмежимося наведенням формул, які описують метод четвертого порядку, один з найуживаніших методів інтегрування диференційних рівнянь. Цей *класичний метод Рунге – Кутти* описується системою з п'яти співвідношень:

$$y_{m+1} = y_m + h/6\cdot(R_1 + 2R_2 + 2R_3 + R_4) \tag{2.6}$$

де

$$R_1 = f(x_m, y_m), \tag{2.7}$$
$$R_2 = f(x_m + h/2, y_m + R_1/2), \tag{2.8}$$
$$R_3 = f(x_m + h/2, y_m + R_2/2), \tag{2.9}$$
$$R_4 = f(x_m + h, y_m + R_3). \tag{2.10}$$

Похибка розв'язку для цього методу дорівнює $\varepsilon_t = kh^5$, так що формули (2.6 … 2.10) описують метод Рунге – Кутти четвертого порядку. Зауважимо, що при використанні цього методу функцію треба обчислювати чотири рази.

2.3 Реалізація програми розв'язування диференційного рівняння у C++ Builder

Завдання 1. Розв'язати у C++ Builder диференційне рівняння $y' = x+y-2$ при початкових умовах $y(0)=2$ на проміжку значень x від 0 до 1 методом Рунге − Кутти четвертого порядку.

Побудувати графіки наближеної та точної функцій розв'язку.

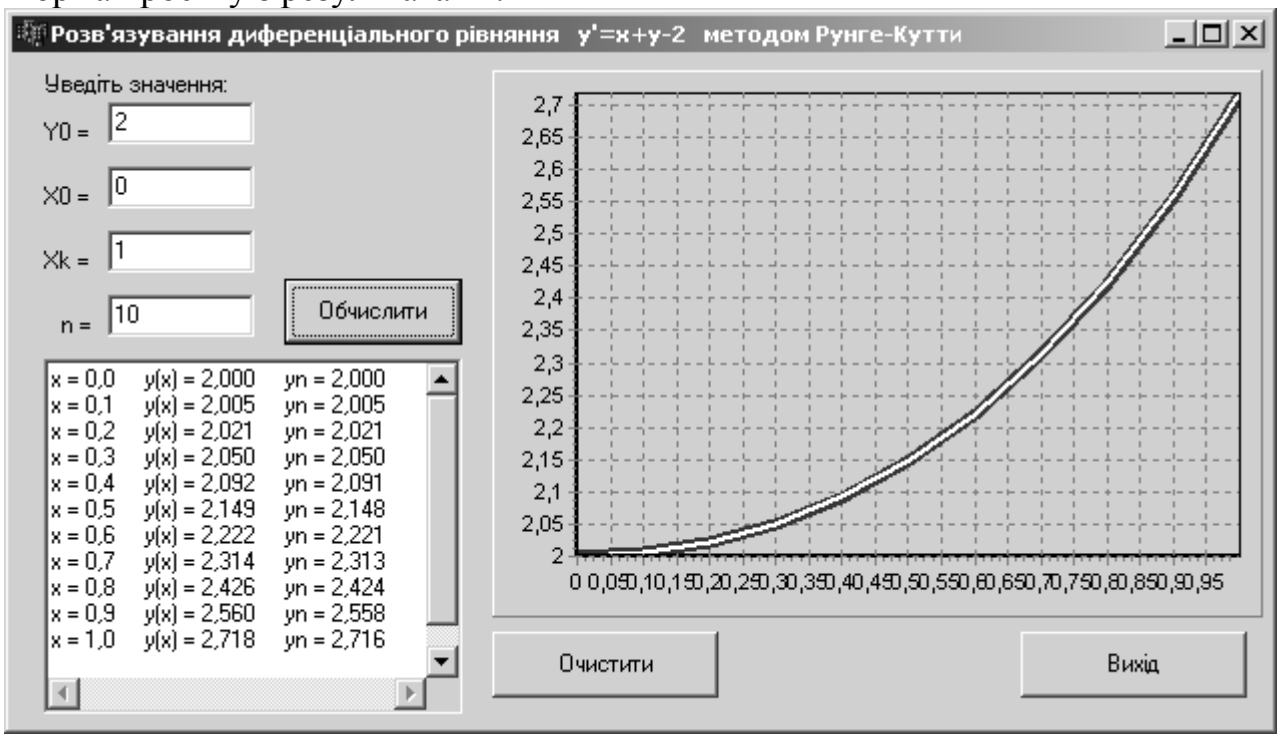
Для порівняння результатів знайдемо аналітично функцію розв'язку диференційного рівняння: $y(x) = e^x - x+1$.

Текст програми

```
……..
#include <math.h>
……
//---------------------------------------------------------------------------
double f(double x, double y)  // функція заданого рівняння
{   return x+y-2;
}

double yt(double x)  // функція точного розв'язку (для порівняння)
{   return exp(x)-x+1;
}

//---------------Кнопка «Обчислити»-------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   double x0, y0, xk, xi, yi, yr, yn, h, dy, R1, R2, R3, R4;
   int n;
   Memo1->Clear();    Series1->Clear();    Series2->Clear();
   y0 = StrToFloat(Edit1->Text);
   x0 = StrToFloat(Edit2->Text);
   xk = StrToFloat(Edit3->Text);
   n = StrToInt(Edit4->Text);  // кількість точок для обчислень
   h = (xk – x0) / (float)n;       // крок інтегрування
   xi = x0; yi = y0;
   do     { yr = yt(xi); // обчислення точки точного розв'язку рівняння
      Memo1->Lines->Add("x = "+FloatToStrF(xi, ffFixed, 4, 1)+
                        "   y(x) = "+FloatToStrF(yr, ffFixed, 7, 3)+
                        "     yn = "+FloatToStrF(yi, ffFixed, 7, 3));
      Series1->AddXY(xi, yi, " ", clRed);
      Series2->AddXY(xi, yr, "", clBlue);
      // обчислення точки наближеного розв'язку рівняння
      R1 = f(xi, yi);
      R2 = f(xi+h/2, yi+ R1/2);
      R3 = f(xi+h/2, yi+ R2/2);
      R4 = f(xi+h, yi+ R3);
      dy = h /6*( R1 + 2* R2 + 2* R3 + R4);
      yi += dy;     xi += h;
    }
     while (xi <= xk);
}
```

```
//---------- Кнопка «Очистити»----------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
   Edit4->Clear();  Memo1->Clear();
  Series1->Clear();
  Series2->Clear();
}

//---------- Кнопка «Вихід»----------
void __fastcall TForm1::Button3Click(TObject *Sender)
{    Close();    }
```

Форма проекту з результатами:



2.4 Розв’язування диференційного рівняння у Mathcad

Для розв’язування диференційного рівняння у Mathcad, треба використати функцію

Z=rkfixed(y, x0, xk, n, D),

де $y = [y_0\ y_1\ \dots\ y_m]$ − вектор початкових значень;

x0 – початкове значення незалежної змінної x;

xk – кінцеве значення незалежної змінної x;

n − кількість значень змінних x та y;

D(x, y) – вектор функцій правих частин диференційних рівнянь;

Z − матриця, нульовий стовпець якої − це вектор порядкових номерів точок розв’язку (від 0 до n), перший стовпець − значення вектора незалежної змінної

x , другий та подальші стопці − значення векторів розв'язку диференційного рівняння $y_0(x)$, $y_1(x)$ … $y_m(x)$.

Завдання 2. Розв'язати в Mathcad диференційне рівняння

$$y' = \frac{y}{x} + x^2$$

при початкових умовах $y(0)=0$ на проміжку значень x від 1 до 5 в 40 точках.

Лістинг розв'язку рівняння в Mathcad:

$y_0 := 0$

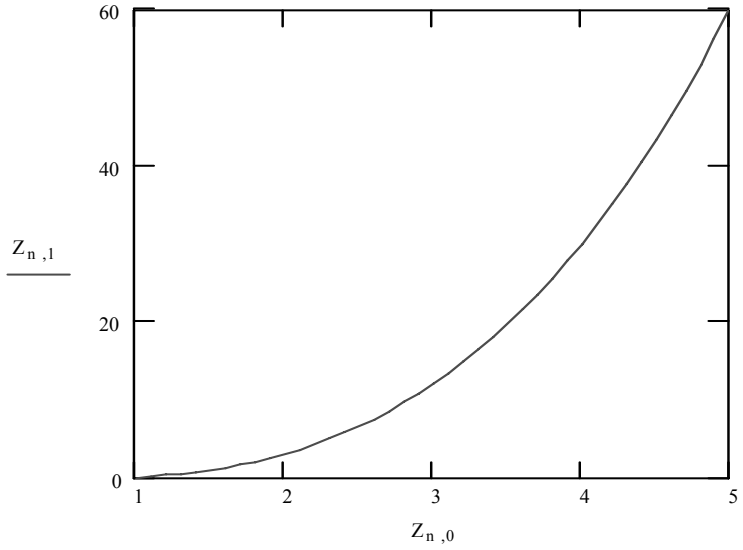
$D(x, y) := \frac{y_0}{x} + x^2$

$Z := \text{rkfixed}(y, 1, 5, 40, D)$

$Z =$

	0	1
0	1	0
1	1.1	0.115
2	1.2	0.264
3	1.3	0.448
4	1.4	0.672
5	1.5	0.937
6	1.6	1.248
7	1.7	1.606
8	1.8	2.016
9	1.9	2.479
10	2	3
11	2.1	3.58
12	2.2	4.224
13	2.3	4.933
14	2.4	5.712
15	2.5	6.562

$n := 0 .. 40$



Завдання 3. Розв'язати систему диференційних рівнянь

$$\begin{cases} x' = x + 2y \\ y' = 3x + y \end{cases}$$

при початкових умовах $x(0)=4$ $y(0)=6$ на проміжку значень x від 0 до 1в 40 точках. Побудувати графіки розв'язку системи рівнянь.

Лістинг розв'язку системи рівнянь в Mathcad:

Вектор початкових умов:

$$y := \begin{pmatrix} 4 \\ 6 \end{pmatrix}$$

Вектор функцій диференційних рівнянь:

$$D(x, y) := \begin{pmatrix} y_0 + 2 \cdot y_1 \\ 3 \cdot y_0 + y_1 \end{pmatrix}$$

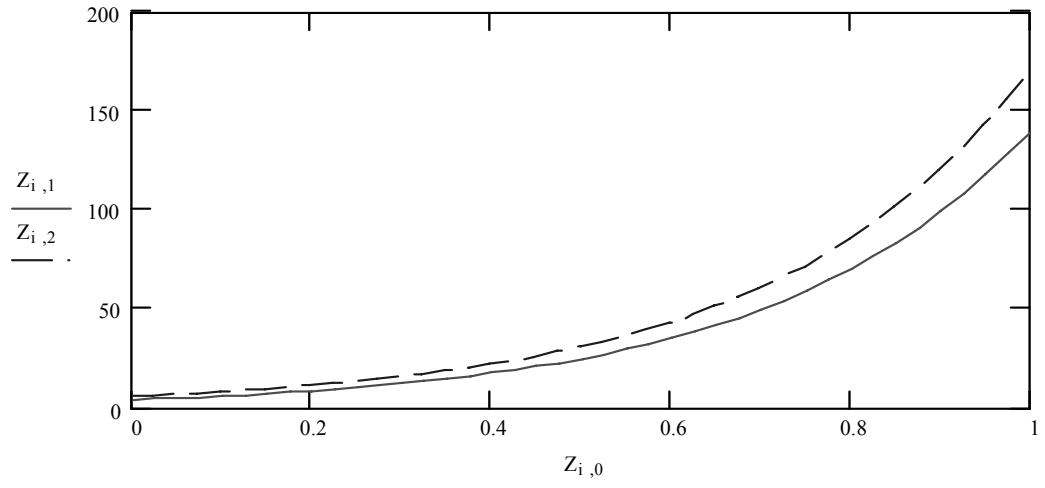
Обчислення розв'язку системи диференційних рівнянь:

$$Z := \text{rkfixed}(y, 0, 1, 40, D)$$

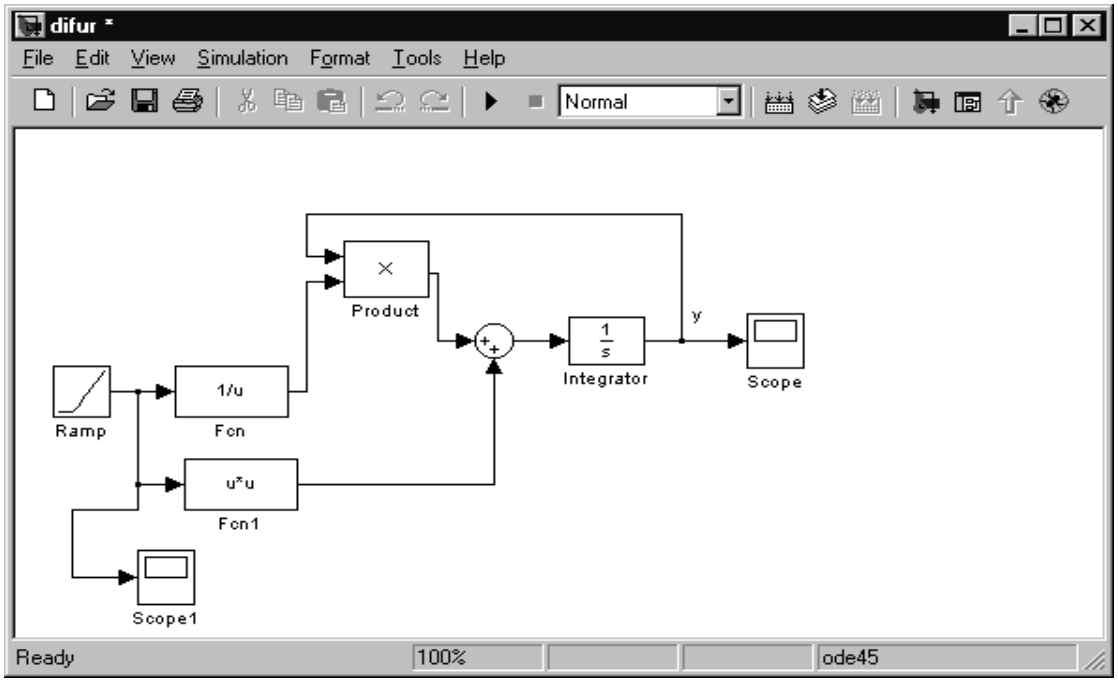
$Z =$

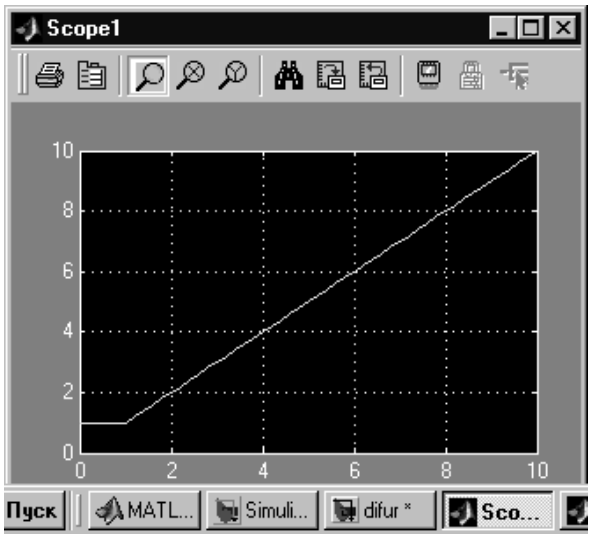
	0	1	2
0	0	4	6
1	0.025	4.417	6.471
2	0.05	4.869	6.987
3	0.075	5.36	7.552
4	0.1	5.893	8.17
5	0.125	6.473	8.846
6	0.15	7.103	9.586
7	0.175	7.788	10.393
8	0.2	8.534	11.276
9	0.225	9.345	12.239
10	0.25	10.227	13.292
11	0.275	11.187	14.441
12	0.3	12.233	15.695
13	0.325	13.371	17.064
14	0.35	14.611	18.557
15	0.375	15.961	20.187

$i := 0 .. 40$

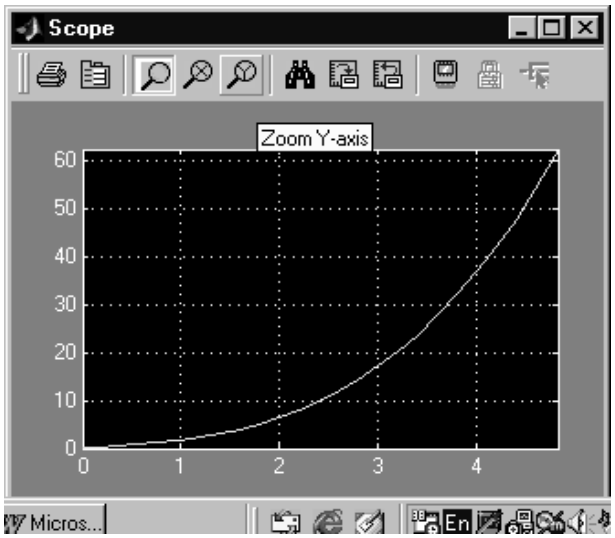


2.4 Розв'язування диференційного рівняння $y' = y/x + x^2$ у Simulink

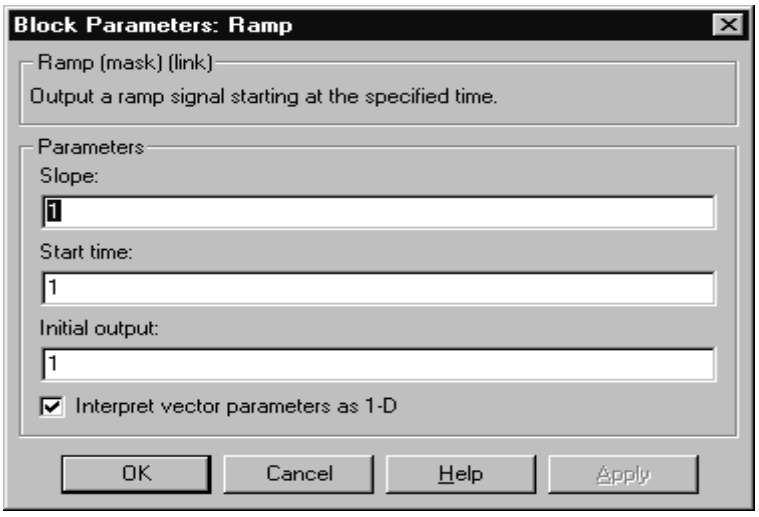




Вхідний сигнал x(t)



Вихідний сигнал y(t)



Параметри вхідного сигналу

Лабораторна робота № 2

Розв'язування диференційних рівнянь та систем за допомогою програмних засобів (C++, Mathcad)

Завдання

1 Подати короткий опис методів розв'язування диференційних рівнянь (Ейлера та Рунге – Кутти).

2 Обчислити таблицю наближених значень розв'язку диференційного рівняння $y' = f(x, y)$, що задовольняє початковим умовам $y(x_0) = y_0$ ($y(x_0) = 0$, якщо $x_0 = 0$) на проміжку [0, 1+a] з використанням математичного пакета Mathcad ($a = 0.1n$, n – номер варіанта). Крок змінювання значень аргументу $h = 0.05$. Індивідуальні варіанти завдань подано у табл. 2.1.

3 Побудувати графік розв'язку рівняння.

Таблиця 2.1− Індивідуальні завдання для розв'язування диференційних рівнянь

№ вар.	Диференційне рівняння	№ вар.	Диференційне рівняння
1	$y' = 1 + 0.2y \sin x - y^2$	**2**	$y' = \cos(x + y) + 0.5(x - y)$
3	$y' = \frac{\cos x}{x+1} - 0.5y^2$	**4**	$y' = (1 - y^2)\cos x + 0.6y$
5	$y' = 1 + 0.4\, y \sin x - 1.5y^2$	**6**	$y' = \frac{\cos y}{x+2} + 0.3y^2$
7	$y' = \cos(1.5x + y) + (x - y)$	**8**	$y' = 1 - \sin(x + y) + \frac{0.5y}{x+2}$
9	$y' = \frac{\cos y}{x+1.5} + 0.1y^2$	**10**	$y' = 0.6 \sin x - 1.25\, y^2 + 1$
11	$y' = \cos(2x + y) + 1.5(x - y)$	**12**	$y' = 1 - \frac{0.1y}{x+2} - \sin(2x + y)$
13	$y' = \frac{\cos y}{x+1.25} - 0.1y^2$	**14**	$y' = 1 + 0.8\, y \sin x - y^2$
15	$y' = \cos(1.5x + y) + 1.5(x - y)$	**16**	$y' = 1 - \sin(2x + y) + \frac{0.3y}{x+2}$
17	$y' = \frac{\cos y}{x+1.75} - 0.5y^2$	**18**	$y' = 1 + (1 - x)\sin y - (2 + x)y$
19	$y' = (0.8 + y^2)\cos x + 0.3y$	**20**	$y' = 1 + 2.2 \sin x + 1.5y$
21	$y' = \cos(x + y) + 0.75(x - y)$	**22**	$y' = 1 - \sin(1.25x + y) + \frac{0.5y}{x+2}$
23	$y' = \frac{\cos y}{x+2} - 0.3y^2$	**24**	$y' = 1 - \sin(1.75x + y) + \frac{0.1y}{x+2}$

Таблиця 2.1 (закінчення)

№ вар.	Диференційне рівняння	№ вар.	Диференційне рівняння
25	$y' = \frac{\cos y}{1.25 + x} - 0.5y^2$	**26**	$y' = \cos(1.5x + y) + 2.25(x - y)$
27	$y' = \frac{\cos y}{1.5 + x} - 1.25y^2$	**28**	$y' = 1 - (x - 1)\sin y + 2(x - y)$
29	$y' = 1 - \sin(0.75x - y) + \frac{1.75y}{x + 1}$	**30**	$y' = \cos(x - y) + \frac{1.25y}{1.5 + x}$

4 Обчислити таблицю наближених значень розв'язку диференційного рівняння $y' = f(x, y)$, що задовольняє початковим умовам $y(x_0) = y_0$ на проміжку [0, 1] з використанням алгоритмічної мови С++ методом:

а) Ейлера *для парних номерів* варіантів;

б) Рунге − Кутти *для непарних номерів* варіантів.

Побудувати графік функції. Крок змінювання значень аргументу $h = 0.05$. Індивідуальні варіанти диференційних рівнянь подано у табл. 2.1.

5 Розв'язати систему диференційних рівнянь на проміжку $0 \le t \le 2$ з кроком 0.05 у Mathcad

$$\begin{cases} x' = ax - 4y, \\ y' = x + ay, \end{cases}$$

де $a = 0.1n$, якщо n – номер варіанта, $x(0) = 2$ та $y(0) = 3$. Побудуйте графіки розв'язків системи диференційних рівнянь x(t) та $y(t)$.

3 НАБЛИЖЕННЯ ФУНКЦІЙ

3.1 Поняття апроксимації та інтерполяції

Апроксимація (лат. approximation – наближення) – наближене вираження одних математичних об'єктів іншими, простішими, наприклад, кривих ліній – ламаними, ірраціональних чисел — раціональними, неперервних функцій — многочленами і т. д.

Апроксимація функції $f(x)$ – це побудова такої функції $g(x)$, яка приблизно (у певному сенсі) дорівнює вихідній функції на розглянутому відрізку, тобто $g(x) \cong f(x)$ для $x \in [x_0, x_n]$. Тобто, апроксимацією називають наближення функції $f(x)$ більш простою функцією $g(x)$.

x	$f(x)$
x_0	f_0
x_1	f_1
…	…
x_n	f_n

Близькості цих функцій домагаються шляхом введення до апроксимуючої функції $g(x)$ вільних параметрів $a_0, a_1, \ldots, a_n$ та відповідним їхнім вибором. Наприклад, за таблицею значень вимірів (x_i, f_i) або обчислень $f(x)$ підібрати (побудувати) функціональну залежність – апроксимуючу функцію $g(x, a_0, a_1, \ldots, a_n)$, графік якої проходить порівняно близько щодо вузлів базової таблиці. Обрані значення x_i ($i = 0,1,\ldots, n$) називаються *вузлами таблиці*, частіше не рівновіддаленими.

Порядок наближення функції:

1) визначити клас шуканої наближеної функції;

2) визначити міру близькості вихідної функції та наближеної.

Наприклад, використовуючи методи найменших квадратів треба, щоби:

$$\Delta = \sqrt{\frac{1}{n+1}\sum_{i=0}^{n}\left(f(x_i) - g(x_i)\right)^2} = \min$$

А при *рівномірному наближенні*: $\max\left|f(x) - g(x)\right| < \varepsilon$.

У задачах теорії коливань, електродинаміки, твердотілої електроніки широко використовуються апроксимації функцій для описання фізичних параметрів середовищ, для задавання характеристик активних і пасивних елементів шляхом радіотехнічних ланцюгів і т.д. В обчислювальній математиці апроксимація функцій є основою для розроблення багатьох методів і алгоритмів.

Лінійна апроксимація функції використовує лінійну залежність, тобто будує пряму

$$y = a_0 + a_1 x,$$

яка проходить поблизу вузлів. Для її побудови треба віднайти два дійсних числа a_0 та a_1.

Інтерполяція – (від лат. interpolatio – перетворення) в обчислювальній математиці спосіб, за допомогою якого за таблицею, що містить певні числові дані, можна віднайти проміжні результати, яких нема безпосередньо у таблиці. Наприклад, визначення функції $f(x)$ для аргументів $x \in [x_0, x_n]$ за відомими значеннями $f(x_i)$, де $i = 0, 1, \ldots, n$. Якщо x лежить зовні інтервалу $[x_0, x_n]$, аналогічна процедура називається *екстраполяцією*. Найпростішою є лінійна інтерполяція, при якій приріст функції вважають пропорційним приросту аргументу.

3.2 Інтерполяція алгебраїчним поліномом

Нехай задано функцію $f(x)$: $f(x_0) = y_0$; $f(x_1) = y_1$; …; $f(x_n) = y_n$. Треба побудувати поліном

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$$

такий, щоби його значення $P_n(x_i) = y_i$, для $i = 1, n$ збігалося зі значеннями y_n у вузлах інтерполяції $x_0, x_1, \ldots, x$:

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + ... + a_n x_0^n = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + ... + a_n x_1^n = y_1 \\ ..... \\ a_0 + a_1 x_n + a_2 x_n^2 + ... + a_n x_n^n = y_n \end{cases}$$

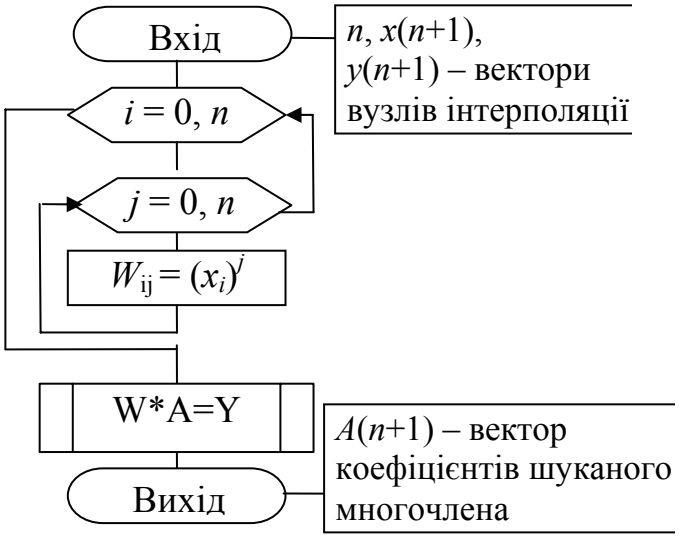
Система лінійних алгебраїчних рівнянь (СЛАР) щодо вільних параметрів a_i має розв'язок, оскільки визначник системи є відмінний від нуля, якщо серед вузлів x_i немає однакових. Визначник системи називають визначником Вандермонда.

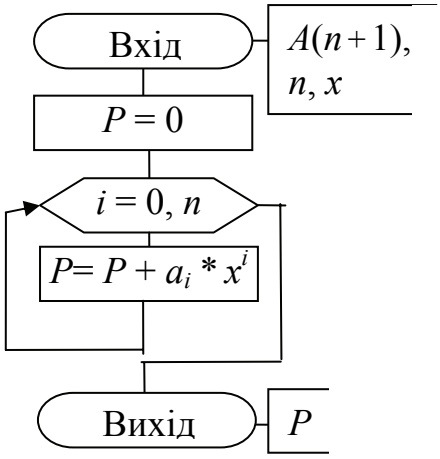
Визначник Вандермонда: $$W = \begin{vmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ . & . & . & & . \\ 1 & x_n & x_n^2 & & x_n^n \end{vmatrix}$$

Систему можна записати у матричному вигляді:

$$\mathbf{W} * \mathbf{A} = \mathbf{Y}.$$

Інтерполяція поліномами має такі переваги, як простота обчислень їхніх значень, диференціювання та інтегрування. Алгоритм для цієї задачі:



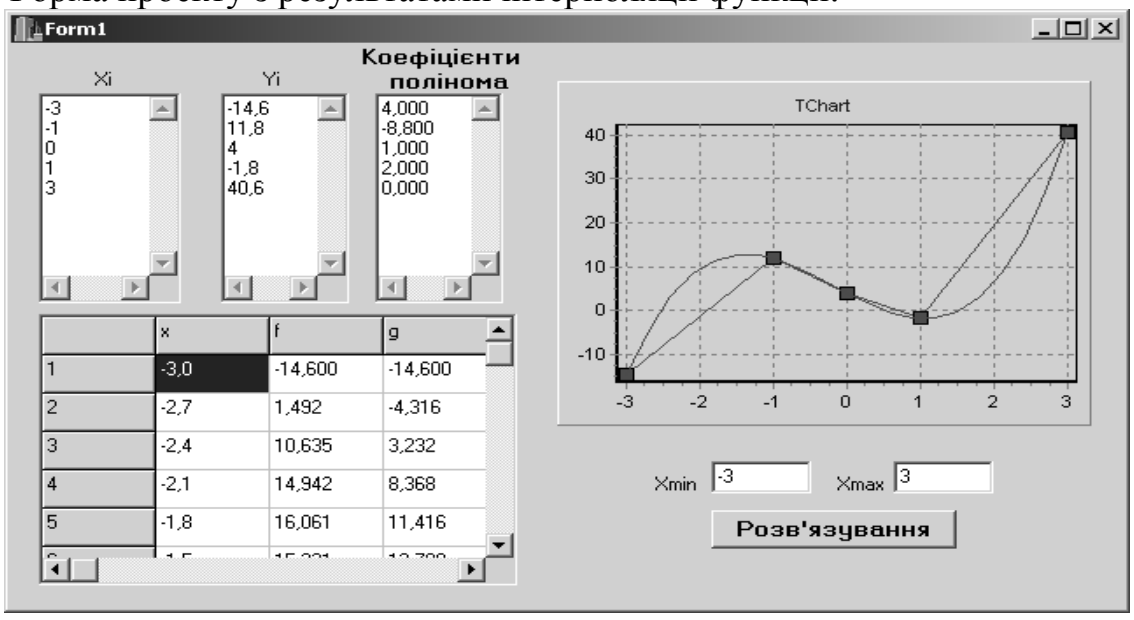


Текст програми у C++ Builder

```
#include <vcl.h>
#pragma hdrstop
// Підключення бібліотеки з методом Гауса
#include "c:\ :\Students\lib\gauss.cpp"
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{ }
//---------------------------------------------------------------------
// Параметри підпрограми розв'язування системи рівнянь з бібліотеки
int u;   float a[col+1][col+2],y[col+1];

// Опис вихідної функції
 float f(float x)
  {return 0.2*pow(x,5)+x*x-7*x+4;}

//------------ Кнопка «Розв'язування» ----------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{   int p;
 float x[20],z[20]; int i,j,n,n1;
   n=Memo1->Lines->Count;
// Ввод вектора X з Мемо
 for (i=0;i<n;i++)
    x[i]=StrToFloat(Memo1->Lines->Strings[i]);
    n1=Memo2->Lines->Count;
    if (n != n1)
   {ShowMessage("Проверьте количество введенных данных ! "); exit;}
   // ----- Введення массива Y з Мемо------------------
      for (i=0;i<n;i++)
 {    z[i]=StrToFloat(Memo2->Lines->Strings[i]);
// -----Побудова графіка по введеним точкам
      Series1->AddXY(x[i],z[i],"",clRed);
}
// ------Обчислення коефіцієнтів матриці----------------
      u=n;
     for (i=0;i<u;i++)
       { a[i+1][1]=1.0;
          for (j=1;j<u;j++)
           a[i+1][j+1]=a[i+1][j]*x[i];
            a[i+1][u+1]=z[i];       }
```

```
  p=gaussu(u); // звертання до підпрограми метода Гауса
// Виведення коефіцієнтів полінома
   for (i=0;i<u;i++)
   Memo3->Lines->Add(FloatToStrF(y[i+1],ffFixed,6,3));
//  Обчислення таблиці значень та побудова графіків
   float xbeg,xend,h,xt,yt,g,r;
    StringGrid1->Cells[1][0]="x";
    StringGrid1->Cells[2][0]="f";
    StringGrid1->Cells[3][0]="g";
      xbeg=x[0];   xend=x[n-1]; // Границі проміжку інтерполяції
      h=(float)(xend-xbeg)/20; // Крок  інтерполяції для 20 точок
       if (xbeg>xend ||h<=0)
     { ShowMessage("Невірно введено дані"); exit;}
      xt=xbeg;  i=1;
     do
    {       yt=f(xt);
    //Обчислення та виведення значень наближеної функції
    g=y[1];r=1;
    for (j=1;j<u;j++)
     {r*=xt;  g+= y[j+1]*r; }
    StringGrid1->Cells[0][i]=IntToStr(i) ;
    StringGrid1->Cells[1][i]=FloatToStrF(xt,ffFixed,5,1) ;
    StringGrid1->Cells[2][i]=FloatToStrF(yt,ffFixed,6,3);
    StringGrid1->Cells[3][i]=FloatToStrF(g,ffFixed,6,3);
    Series2->AddXY(xt, g,"",clGreen);
        xt+=h;  i++;    }
     while (xt<=xend);
}
```

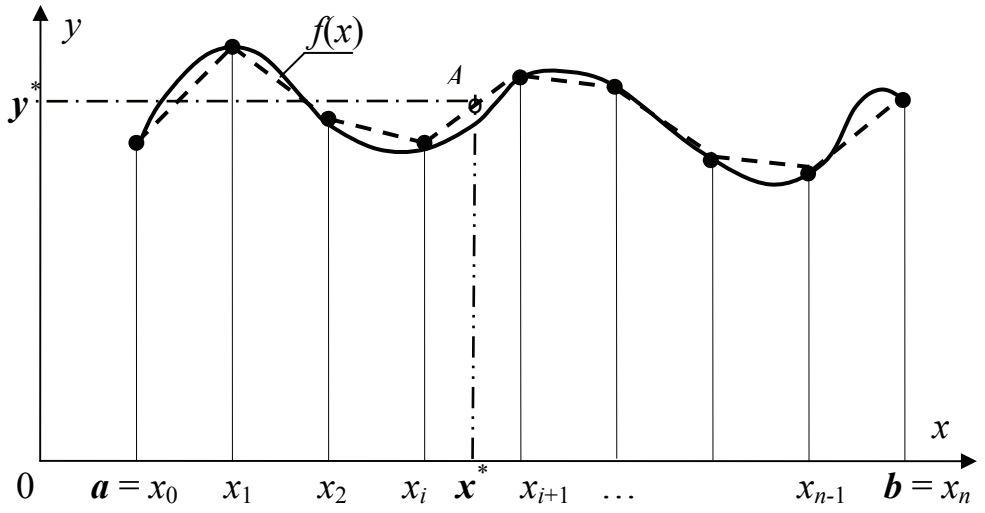
Форма проекту з результатами інтерполяції функції:



$$g(x) = 0 \cdot x^4 + 2x^3 + x^2 - 8.8x + 4 = 2x^3 + x^2 - 8.8x + 4$$

3.3 Лінійна інтерполяція

При лінійній інтерполяції задана функція *f(x)* (суцільна лінія на рис.) замінюється ламаною лінією (пунктирна лінія на рис.):



Формула лінійної інтерполяції для точки A (x^*, y^*):

$$y^* = y_i + (y_{i+1} - y_i)\frac{x^* - x_i}{x_{i+1} - x_i},$$

де x^* належить проміжку $[x_i, x_{i+1}]$, $i = 0, 1, \ldots, n$.

Розглянемо алгоритм лінійної інтерполяції на прикладі.

Завдання. Ввести з клавіатури парні набори значень x_i та y_i для функції $y = f(x)$ та обчислити для цієї функції значення лінійної інтерполяції на заданому проміжку змінної $x \in [x_0, x_n]$.

Текст програми

```
//---- Функція лінійної  інтерполяції --------------------------
double lin_interp (double *x, double *y, int n, double xp)
{ int i, j;  double p, yr;
  if(x[0] == xp) yr = y[0];
  for(i=0; i<n-1; i++)
   { if((x[i] < xp) && (xp <= x[i+1]))
      yr = y[i] + (y[i+1] – y[i]) * (xp – x[i]) / (x[i+1] – x[i]) ;   }
return yr;
}
//---------Кнопка “Обчислити”-----------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{ Memo3->Clear();    Series1->Clear();    Series2->Clear();
  double x[20], y[20];   int i, n, n1;
  n = Memo1->Lines->Count;      n1 = Memo2->Lines->Count;
  if(n != n1) { ShowMessage("Перевірте кількість введених даних!"); exit; }
     for(i=0; i<n; i++)
```

```
  { x[i] = StrToFloat(Memo1->Lines->Strings[i]);   // Введення масиву X
    y[i] = StrToFloat(Memo2->Lines->Strings[i]);   // Введення масиву Y
    Series1->AddXY(x[i], y[i], "", clRed);
  }
  double  xbeg, xend, h, xt, yt, yp;
  if(Edit1->Text == "" )
    {  ShowMessage("Введіть кількість контрольних точок!"); exit;}
  int kt = StrToInt(Edit1->Text);
  xbeg = x[0];
  xend = x[n-1];
  h = (double) (xend – xbeg) / (kt – 1);
  if(xbeg > xend  ||  h <= 0)
    {  ShowMessage("Неправильно Введені дані"); exit;  }
  xt = xbeg;
  do
    {  yp = lin_interp(x,y,n,xt);
       Memo3->Lines->Add("x=" + FloatToStrF(xt, ffFixed, 5, 1) +
                         + "   yp=" + FloatToStrF(yp, ffFixed, 6, 3));
       Series2->AddXY(xt, yp, "", clWhite);
       xt += h; }
      while (xt<=xend);
}
//--------- Кнопка "Очистити"-----------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{  Edit1->Clear();       Series1->Clear();   Series2->Clear();
   Memo1->Clear();     Memo2->Clear();   Memo3->Clear();
}
//--------- Кнопка "Вихід" -------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{  Close();    }
//---------------------------------------------------------------------------
```

Форма проекту з результатами:

3.4 Інтерполяційний многочлен Лагранжа

Задано функцію $f(x)$: $f(x_0) = y_0$, $f(x_1) = y_1$, …, $f(x_n) = y_n$. Треба побудувати многочлен $g(x)$, де x – довільне значення інтервалу $[x_0, x_n]$.

Формула многочлена Лагранжа має вигляд

$$g(x) = \sum_{i=0}^{n} y_i \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)},$$

або у скороченому вигляді

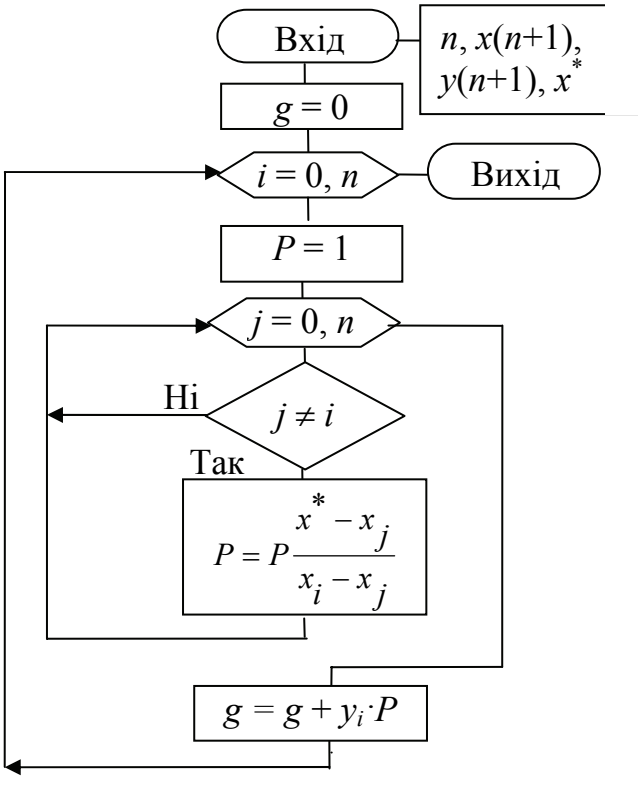
$$g(x) = \sum_{i=0}^{n} y_i \frac{\prod_{j=0, j\neq i}^{n}(x-x_j)}{\prod_{j=0, j\neq i}^{n}(x_i-x_j)} = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{(x-x_j)}{(x_i-x_j)}$$

Для обчислення полінома (многочлена) Лагранжа не треба попереднього визначення коефіцієнтів полінома шляхом розв'язування системи рівнянь, проте для кожного значення x^* поліном доводиться перераховувати знову. Тобто практичне застосування полінома Лагранжа виправдано лише у випадку, якщо інтерполяційна функція обчислюється у порівняно невеликій кількості значень x.

Приклад реалізації задачі інтерполяції поліномом Лагранжа у C++

Завдання. Ввести з клавіатури парні набори значень x_i та y_i для функції $y = f(x)$ й обчислити для цієї функції інтерполяційний многочлен Лагранжа. Обчислити наближені значення на цьому проміжку у заданій кількості точок.

Схема алгоритму інтерполяції цієї задачі поліномом Лагранжа.



Текст програми

```
//---------- Функція полінома Лагранжа----------
double lagrang(double *x, double *y, int n, double xp)
{ int i, j;   double p, yr;
  yr = 0;
  for(i=0; i<n; i++)
   {
     p = 1;
     for (j=0; j<n; j++)
       if(i != j) p *= (xp – x[j]) / (x[i] – x[j]) ;
     yr += y[i] * p;
   }
 return yr;
}

//---------- Кнопка "Обчислити"----------
void __fastcall TForm1::Button1Click(TObject *Sender)
{ double x[20], y[20];   int i, n, n1;
  Series1->Clear();      Series2->Clear();  Memo3->Clear();
  n=Memo1->Lines->Count;
  n1=Memo2->Lines->Count;
  if(n != n1)
   { ShowMessage("Перевірте кількість введених даних!");  exit; }
     for(i=0; i<n; i++)
      {  x[i] = StrToFloat(Memo1->Lines->Strings[i]); //  Введення масиву X
        y[i] = StrToFloat(Memo2->Lines->Strings[i]); //  Введення масиву Y
        Series1->AddXY(x[i], y[i], "", clRed);
      }
   double xbeg, xend, h, xt, yt, yp;
   if(Edit1->Text == "" )
     { ShowMessage("Введіть кількість контрольних точок!"); exit; }
   int  kt = StrToInt(Edit1->Text);
   xbeg = x[0];
   xend = x[n-1];
   h = (double) (xend – xbeg) / (kt – 1);
   if(xbeg>xend || h<=0) { ShowMessage("Неправильно введені дані"); exit; }
   xt = xbeg;
   do
     {  yp = lagrang(x, y, n, xt);
        Memo3->Lines->Add("x=" + FloatToStrF(xt, ffFixed, 5, 1) +
                          +"   yp=" + FloatToStrF(yp, ffFixed, 6, 3));
        Series2->AddXY(xt, yp, "", clGreen);
        xt += h;
    }   while (xt <= xend);
}
```

```
//-------------------- Кнопка "Очистити" --------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
   Edit1->Clear();     Series1->Clear();   Series2->Clear();
  Memo1->Clear();  Memo2->Clear();   Memo3->Clear();
}

//--------------------- Кнопка "Вихід"-------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
Close();
 }
```

Форма проекту з результатами:

Приклад виконання визначення наближеної функції у Mathcad

Лістинг виконання розрахунків:

Вихідні данні векторів

$$\mathbf{X} = \begin{pmatrix} -1.879 \\ -1.2 \\ -0.5 \\ 1.199 \\ 2.1 \\ 3.75 \end{pmatrix} \qquad \mathbf{Y} = \begin{pmatrix} -0.462 \\ 0 \\ 0.821 \\ 0.684 \\ 0.497 \\ 1.066 \end{pmatrix}$$

Інтерполяція алгебраїчним поліномом

$\mathbf{i} := 0 , 1 .. 5$

$\mathbf{j} := 0 , 1 .. 5$

$\mathbf{W_{i,j}} := (\mathbf{X_i})^{\mathbf{j}}$

$$\mathbf{W} = \begin{pmatrix} 1 & -1.887 & 3.559 & -6.715 & 12.668 & -23.9 \\ 1 & -1.2 & 1.44 & -1.728 & 2.074 & -2.488 \\ 1 & -0.5 & 0.25 & -0.125 & 0.063 & -0.031 \\ 1 & 1.239 & 1.534 & 1.9 & 2.353 & 2.915 \\ 1 & 2.4 & 5.76 & 13.824 & 33.178 & 79.626 \\ 1 & 3.8 & 14.44 & 54.872 & 208.514 & 792.352 \end{pmatrix}$$

$\mathbf{AP} := \mathbf{W}^{-1} \cdot \mathbf{Y}$

Наближена функція $\mathbf{YP(x)} := \sum_{\mathbf{i}=0}^{5} \mathbf{AP_i} \cdot (\mathbf{x})^{\mathbf{i}}$ $\qquad \mathbf{x} := \mathbf{X_0} , \mathbf{X_0} + 0.05 .. \mathbf{X_5}$

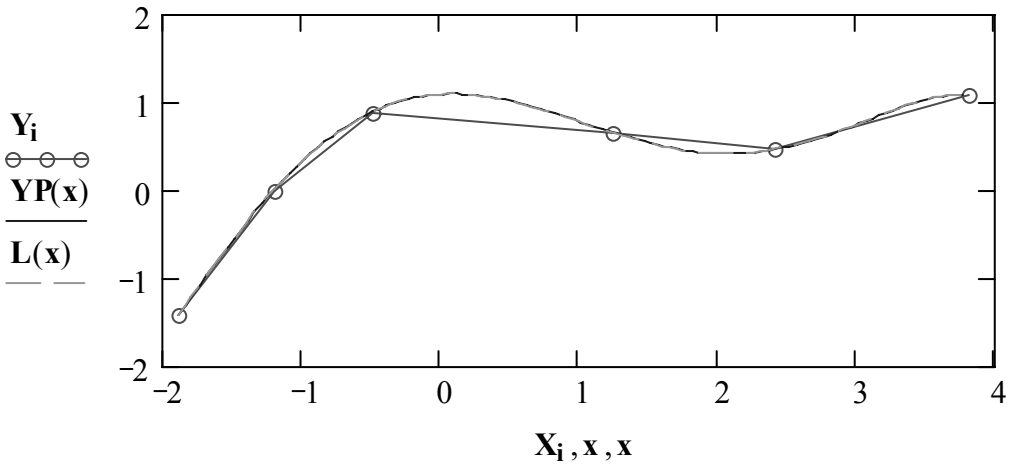
Коефіцієнти наближеної функції

$$\mathbf{AP} = \begin{pmatrix} 1.089 \\ 0.089 \\ -0.576 \\ 0.147 \\ 0.037 \\ -9.852 \times 10^{-3} \end{pmatrix}$$

Інтерполяція поліномом Лагранжа

$$\mathbf{L(x)} := \sum_{\mathbf{i}=0}^{5} \mathbf{Y_i} \cdot \prod_{\mathbf{j}=0}^{5} \mathbf{if}\left(\mathbf{i} = \mathbf{j} , 1 , \frac{\mathbf{x} - \mathbf{X_j}}{\mathbf{X_i} - \mathbf{X_j}}\right)$$

Графікі інтерполяційних функцій



Апроксимація поліномом степеню m

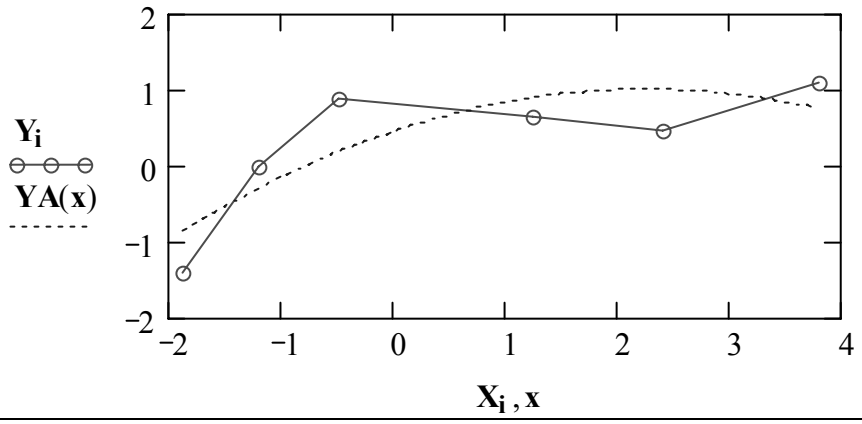
$m := 2 \qquad K := 0, 1 .. m \qquad L := 0, 1 .. m$

$$S_{K,L} := \sum_{i=0}^{5} (X_i)^{K+L} \qquad B_K := \sum_{i=0}^{5} Y_i \cdot (X_i)^K$$

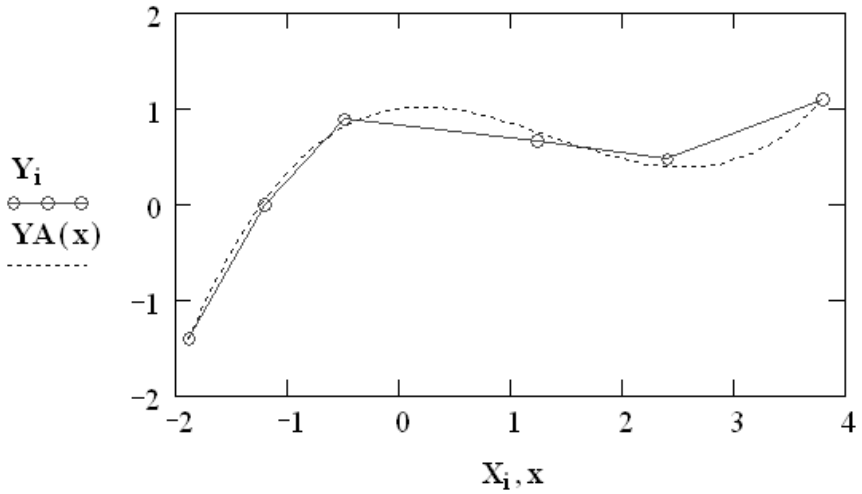
$$Ap := S^{-1} \cdot B \qquad Ap = \begin{pmatrix} 0.453 \\ 0.492 \\ -0.109 \end{pmatrix}$$

Наближена функція $$YA(x) := \sum_{i=0}^{m} Ap_i \cdot (x)^i \qquad x := X_0, X_0 + 0.05 .. X_5$$

Графік наближеної функціїї для m=2



Графік наближеної функції для m=3



Лабораторна робота №3

Інтерполяція та апроксимація функцій за допомогою C++ та Mathcad

1 Подати короткий опис методів наближення функцій.

2 Створити у Mathcad вектори вимірювань (X, Y) згідно до даних, які подано у табл. 3.1. У табл. позначено: i – номер елемента вектора, $a = 0.1N$, де N – номер індивідуального варіанта; X_i – вектор значень аргументу; Y_i – вектор значень функції для відповідних значень аргументу ($Y_i = f(X_i)$).

Таблиця 3.1 – Значення векторів X та Y для задачі наближення функцій

i	0	1	2	3	4	5
X_i	$-1.8 - 0.05\sqrt{a}$	–1.2	–0.5	$1 + 0.5\lg a$	$0.6 + 0.6a$	$3.5 + 0.1a$
Y_i	$1 - 0.12e^{a}$	0	$0.5 + 0.35 \ln a$	$1 - 0.2\sqrt{a}$	$0.5 + 0.25\cos\sqrt{a}$	$1.2 - 0.65\, e^{\sqrt{a}}$

3 Для заданих значень векторів (X_i, Y_i) обчислити алгоритмічною мовою C++ та у Mathcad наближену функцію з використанням:

а) інтерполяційного многочлена *Лагранжа для парних номерів* варіантів;

б) формул *лінійної інтерполяції для непарних номерів* варіантів.

Побудувати графіки значень $Y_i = f(X_i)$ та наближеної функції g(x), де x змінюється від X_0 до X_5 із кроком 0.02.

4 Для заданих значень векторів (X_i, Y_i) обчислити у Mathcad наближену функцію з використанням інтерполяційного алгебраїчного многочлена. Побудувати графіки значень $Y_i = f(X_i)$ та наближеної функції g(x), де x змінюється від X_0 до X_5 з кроком 0.01.

5 Для заданих значень векторів (X_i, Y_i) віднайти у Mathcad коефіцієнти апроксимуючих поліномів 2-го та 3-го ступенів. Побудувати графіки значень $Y_i = f(X_i)$ та наближеної функції g(x), де x змінюється від X_0 до X_5 із кроком 0.025. Обчислити та порівняти середньоквадратичне відхилення наближених функцій 2-го та 3-го степеню.

4 ЧИСЕЛЬНІ МЕТОДИ РОЗВ'ЯЗУВАННЯ ЗАДАЧІ ОПТИМІЗАЦІЇ

4.1 Постановка математичної задачі оптимізації

В достатньо загальному вигляді математичну задачу оптимізації можна сформулювати так: мінімізувати (максимізувати) цільову функцію з урахуванням обмежень на керуючі змінні. Тобто для заданої функції $F(x)$ треба знайти $\bar{x}^*$, яке є мінімумом (чи максимумом) цієї функції на інтервалі $[a, b]$ із заданою точністю ε, тобто знайти

$$\bar{x}^* = \min \{F(x)\}, \quad \bar{x}^* \in [a, b] \tag{4.1}$$

чи

$$\bar{x}^* = \max \{F(x)\}, \quad \bar{x}^* \in [a, b].$$

Функція $F(x)$ може мати кілька екстремальних (мінімальних і максимальних) значень.

Одновимірна оптимізація (пошук екстремумів функцій однієї змінної) є найпростішою і поширеною математичною задачею оптимізації, в якій цільова функція залежить від однієї змінної. Окрім того, до неї зводиться набагато більш складна задача – пошук екстремуму функції багатьох змінних.

4.2 Метод рівномірного пошуку екстремуму

Метод рівномірного пошуку мінімального значення функції заснований на тому, що змінній x присвоюється значення $x + \Delta x$ із кроком $\Delta x = \text{const}$ і обчислюються значення $F(x)$. Якщо $F(x_{n+1}) < F(x_n)$, змінній x надається новий приріст Δx. Як тільки-но $F(x_{n+1})$ стане більше за $F(x_n)$, пошук припиняється (на рис. 4.1 − $F(x_4) > F(x_3)$, тобто $x^* \in [x_3, x_4]$).

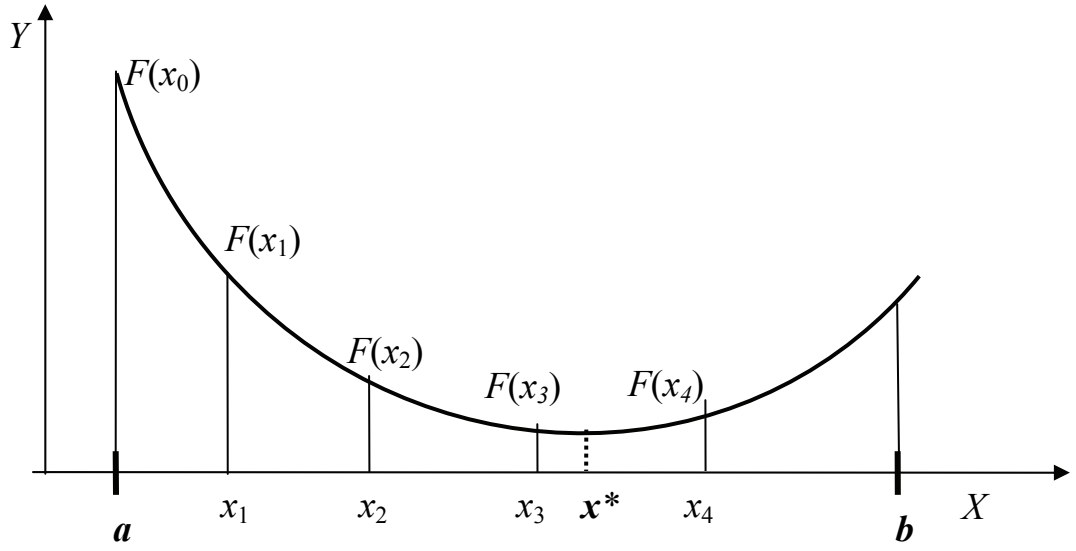


Рисунок 4.1 − Оптимізація методом рівномірного пошуку

При малій заданій похибці цей метод є неекономічний за витратами машинного часу, тому його застосовують тільки з відносно великим кроком

Δx для попереднього обрання проміжку пошуку екстремуму. Схему алгоритму методу рівномірного пошуку наведено на рис. 4.2.

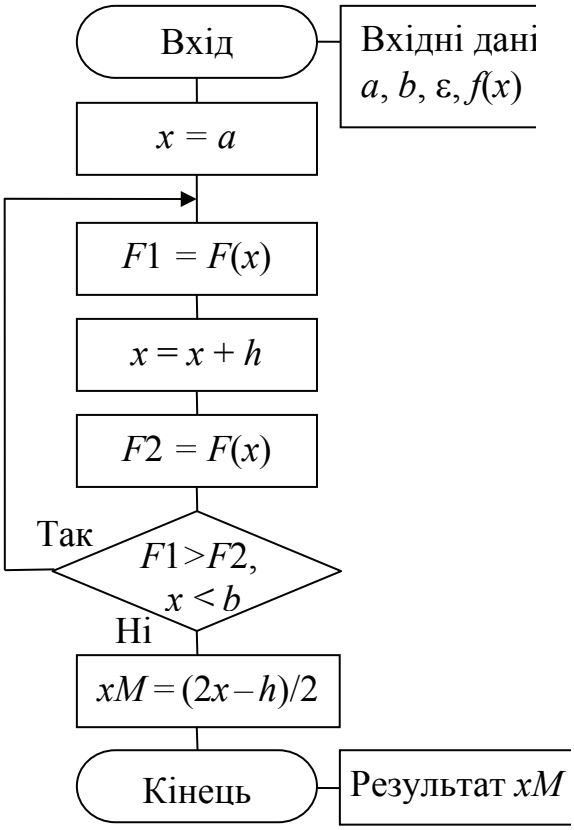


Рисунок 4.2 − Блок-схема методу рівномірного пошуку екстремуму

Текст програми відповідно до цієї блок-схеми разом з програмним кодом для інших методів подано нижче.

4.3 Метод бісекції

Метод бісекції, відомий також під назвами *метод ділення навпіл* чи *метод дихотомії*, – найпростіший, надійний, але порівняно повільний метод. Суть методу в тому, що інтервал ділиться навпіл, обраховуються значення функції зліва та справа від точки поділу на відстані похибки ε і в залежності від того, яке із значень функції більше змінюється границя інтервалу. Така процедура дозволяє виділити наполовину менший інтервал. Її повторюють доти, доки довжина інтервалу не стане меншою від заданої точності.

Цей метод є *методом прямого пошуку*. У ньому тільки обчислені значення цільової функції.

Запишемо докладний словесний алгоритм методу:

1) На кожному кроці процесу пошуку поділити відрізок [*a, b*] навпіл, тобто обчислити координату середини відрізка [*a, b*] за формулою

$$x = (a + b)/2.$$

2) Обчислити значення функції $F(x)$ в околі обчисленої точки $x \pm \varepsilon$ (рис. 4.3):

$$F1 = F(x - \varepsilon),$$
$$F2 = F(x + \varepsilon).$$

3) Порівняти $F1$ та $F2$ і відкинути одну з половинок відрізка $[a, b]$ (рис. 4.3):

а) *при пошуку мінімуму методом бісекції*: якщо $F1 < F2$, відкинути відрізок $[x, b]$, тоді $b = x$ (рис. 4.3,а), інакше – відкинути відрізок $[a, x]$, тоді $a = x$ (рис. 4.3,б);

б) *при пошуку максимуму*: якщо $F1 < F2$, відкинути відрізок $[a, x]$, тоді $a = x$, інакше – відкинути відрізок $[x, b]$, тоді $b = x$.

4) Ділення відрізка $[a, b]$ триває, допоки його довжина не стане меншою за задану точність ε, тобто $|b - a| \le \varepsilon$.

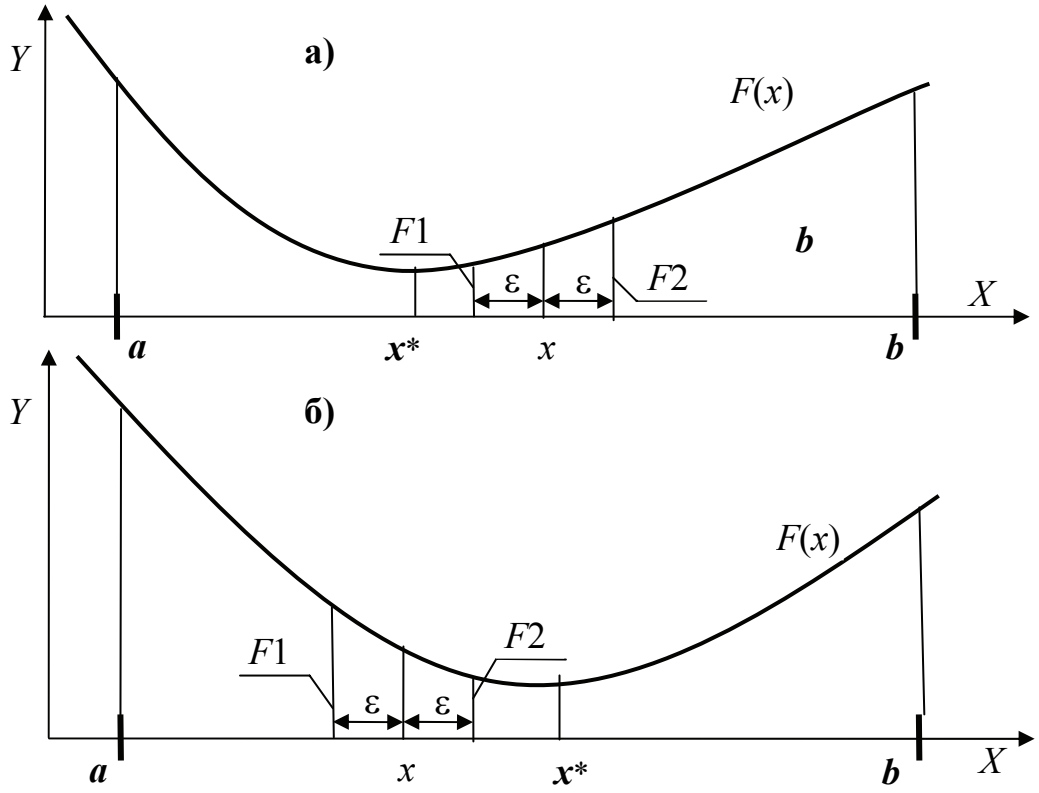


Рис. 4.3. Пошук екстремуму функції $F(x)$ методом бісекції (дихотомії)

Схему алгоритму *методу бісекції* подано на рис. 4.4,

У блоці виведення результатів x – координата точки, у якій функція $F(x)$ має мінімум (чи максимум), FM – значення функції $F(x)$ у цій точці.

Текст програми відповідно до цієї блок-схеми подано після п. 4.4, разом з програмним кодом для інших методів.

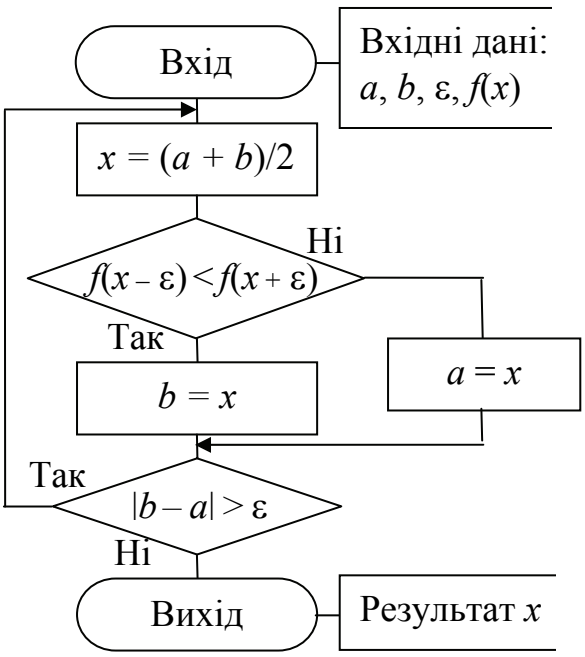


Рисунок 4.4 − Блок-схема функції пошуку екстремуму методом ділення навпіл

4.4 Метод “золотого перетину”

Розглянемо найбільш поширений метод одновимірної оптимізації – *метод “золотого перетину”*.

Точка золотого перетину x_1 поділяє відрізок [a, b] на дві частини так, що відношення більшої частини відрізка до цілого відрізка дорівнює відношенню меншої частини до більшого, тобто дорівнює так званому “золотому відношенню” (рис. 4.5):

$$\frac{b-x_1}{b-a}=\frac{x_1-a}{b-x_1}=\frac{\sqrt{5}-1}{2}\approx 0.618 \qquad (4.2)$$

На рис. 4.5 проілюстровано “золоте відношення” відрізків, де відрізки, позначені дугами догори, відповідають першому дробу рівняння (4.2), а відрізки, позначені дугами донизу, відповідають другому дробу рівняння. Точка x_2 є правою симетричною точкою золотого перетину на відрізку [a, b], якщо для неї виконується умова:

$$\frac{x_2-a}{b-a}=\frac{b-x_2}{x_2-a}.$$

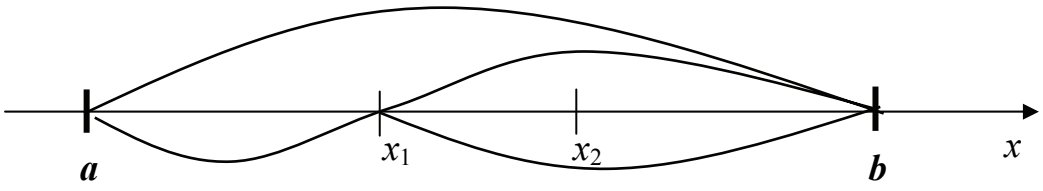


Рисунок 4.5 − Відношення відрізків за методом “золотого перетину”

Алгоритм пошуку екстремуму методом "золотого перетину" (рис. 4.6):

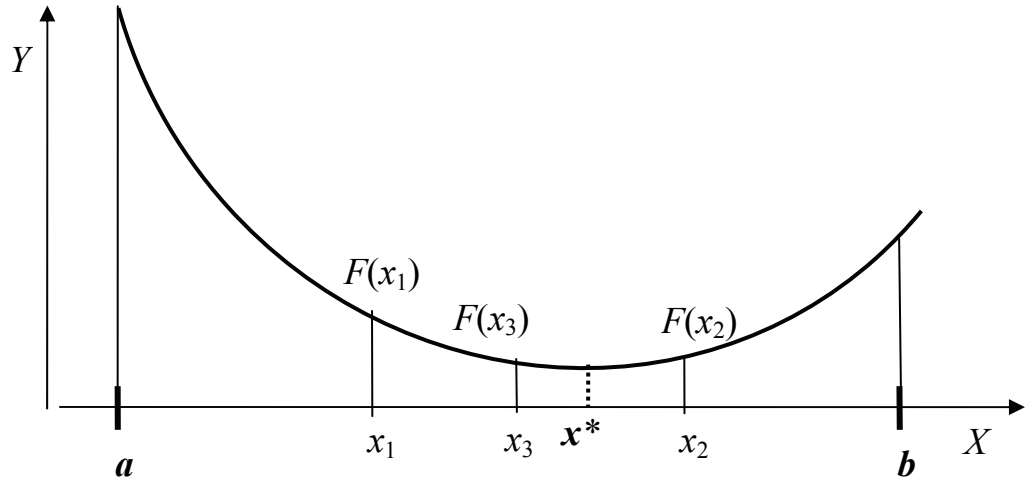


Рисунок 4.6 − Оптимізація методом "золотого перетину"

1) Обчислюємо значення симетричних точок золотого перетину

$$x_1 = a + (1 - \tau)\,(b - a), \quad x_2 = a + \tau\,(b - a),$$

де $\tau = 0.618$ коефіцієнт "золотого відношення".

2) Обчислюємо значення функції в точках золотого перетину

$$F1 = F(x_1), \qquad F2 = F(x_2).$$

3) Порівнюємо $F1$ і $F2$ і відкидаємо одну з половинок відрізка $[a, b]$.

а)При пошуку мінімуму: якщо $F1 < F2$, то відкидаємо відрізок $[x_2, b]$, тоді $b = x_2$, інакше відкидаємо відрізок $[a, x_1]$, тоді $a = x_1$ (див. рис. 4.6).

б)При пошуку максимуму: якщо $F1 < F2$, то відкидаємо відрізок $[a, x_1]$, тоді $a = x_1$. Інакше відкидаємо відрізок $[x_2, b]$, тоді $b = x_2$.

4) Для знайденого у п. 3 зменшеного відрізка обчислюємо симетричну точку золотого перетину (наприклад, $x_3 = x_1 + (1 - \tau)\,(b - x_1)$ – ліву точку золотого перетину відрізка $[x_1, b]$) і обчислюємо значення функції $F(x_3)$. Якщо ввести позначення $F1 = F2$, $F2 = F(x_3)$, то повернувшись до п. 3, можна продовжувати пошук екстремума.

5) Ділення відрізка $[a, b]$ триває, допоки його довжина не стане меншою за задану точність ε, тобто $|b - a| \le \varepsilon$.

Блок-схему алгоритму подано на рис. 4.7. На блок-схемі результат X_M − це координата точки, у якій функція $F(x)$ має мінімум (чи максимум); X_L та X_R – значення лівої та правої точок золотого перетину; $f(X_L)$ та $f(X_r)$ − значення функції $F(x)$ в точках золотого перетину.

Метод "золотого перетину" гарантує знаходження мінімуму у самих несприятливих умовах і вимагає удвічі менше обчислень значень функції $F(x)$ порівняно з методом бісекції, однак він має більш повільну збіжність, оскільки на кожному кроці довжина відрізка для пошуку мінімуму зменшується тільки в 1.7 рази.

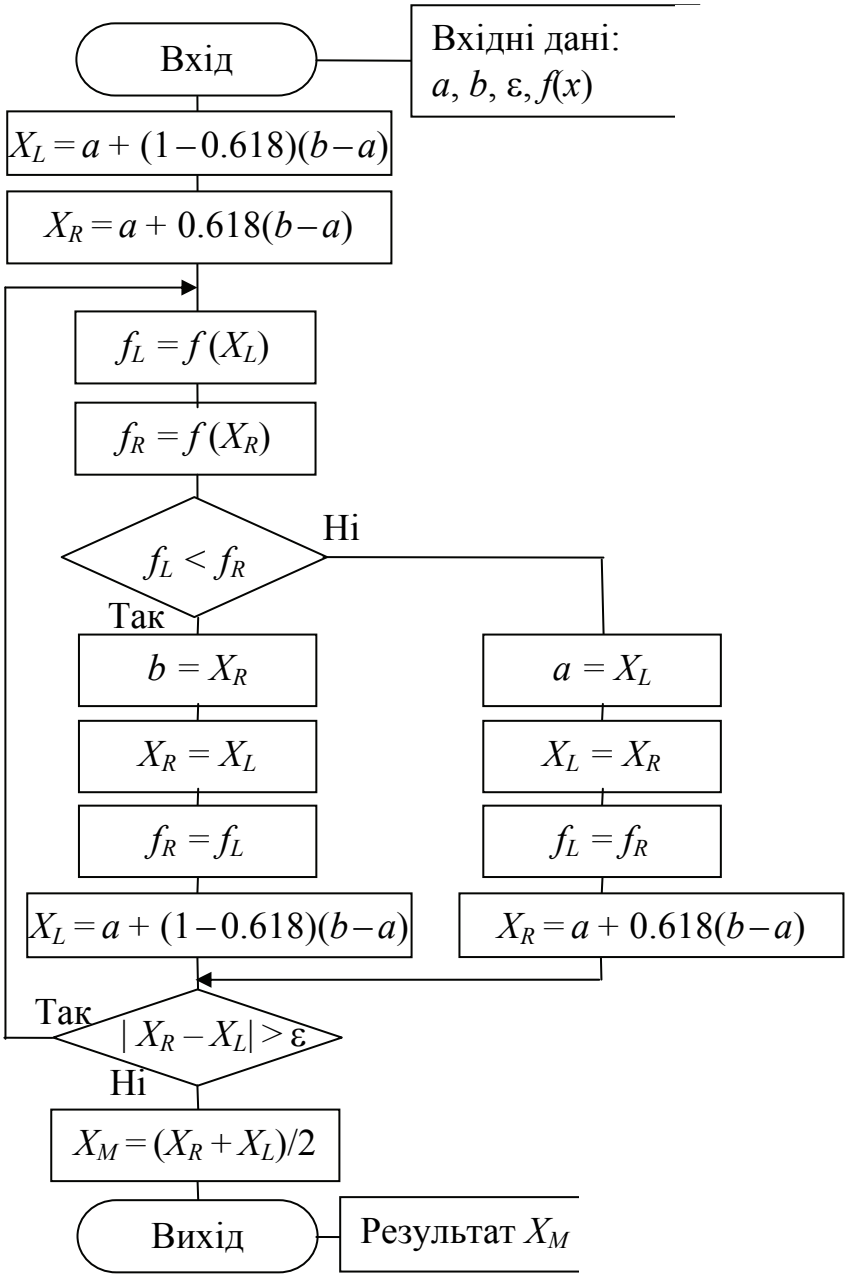


Рисунок 4.7 − Схема алгоритму методу “золотого перетину”

4.5 Реалізація програми одновимірної оптимізації у C++ Builder

Завдання. Знайти мінімальне значення функції $f(x) = x^3 + x^2 - 6x - 4$ на проміжку [–1, 3] з кроком 0.05 методами рівномірного пошуку (кроковий), бі-секції (дихотомії) та ”золотого перетину”.

У цій програмі для порівняння роботи методів наведемо програмний код для усіх трьох розглянутих методів розв’язування задачі оптимізації.

Текст програми

```
      ……….
#include "math.h" //Підключення бібліотеки математичних функцій C++
      ……….
float F(float x)          //  Функція для пошуку мінімуму
{ return pow(x, 3) + pow(x, 2) – 6*x – 4;}
//------------ Функція методу рівномірного пошуку ----------------------
double  ravnomern(double a, double b, double h)
{ double F1, F2, x = a;
  do
    {  F1 = F(x);
       x = x + h;
       F2 = F(x); }
    while (F1 > F2  &&  x <= b);
 double xM = (2*x - h)/2;
 return xM;
 }
//---------------- Кнопка «Рівномірний пошук» ----------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{ float a, b, h, x, xM;
  a = StrToFloat(Edit1->Text);
  b = StrToFloat(Edit2->Text);
  h = StrToFloat(Edit3->Text);
  do    // Побудова графіка функції F(x)
   {  Series1->AddXY(x, F(x), "", clRed);    x = x + h; }
    while (x <= b+h);
 // Звертання до функції методу рівномірного пошуку
  xM = ravnomern(a, b, h);
// Виведення результатів
  Edit4->Text = FloatToStr(xM);
  Edit5->Text = FloatToStr(F(xM));
}
//----------------Функція методу бісекції ---------------------------------
double  bis(double a, double b, double eps)
{ double x = a;
   do
    { x = (a+b)/2;
      if(F(x-eps)<F(x+eps)) b=x;  else a=x; }
while (fabs(b-a) > eps);
 double xM = (a+b)/2; return xM; }
//----------------Кнопка «Метод бісекції» ---------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{ float a, b, eps, xM;
  a = StrToFloat(Edit1->Text);
```

```
  b = StrToFloat(Edit2->Text);
  eps = StrToFloat(Edit3->Text);
  xM = bis(a, b, eps);
  Edit6->Text = FloatToStr(xM);
  Edit7->Text = FloatToStr(F(xM));
}
//------------------ Функція методу « золотого перетину» -----------------
double  zolotoy(double a, double b, double eps)
{ double  XL, XR, fL, fR;
  XL = a + (1 - 0.618)*(b-a);
  XR = a + 0.618*(b-a);
  fL = F(XL);   fR = F(XR);
  do
    {  if(fL < fR)
         { b=XR;  XR=XL;  fR=fL;  XL=a+(1–0.618)*(b–a);  fL = F(XL);  }
       else  { a=XL; XL=XR; fL=fR; XR = a + 0.618*(b-a);  fR=F(XR);  }
     }  while (fabs(b-a)>eps);
 double xM = (a+b)/2;
 return xM;
 }
//------------------ Кнопка «Метод  золотого перетину» -----------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{ float a, b, eps, xM;
  a = StrToFloat(Edit1->Text);    b = StrToFloat(Edit2->Text);
  eps = StrToFloat(Edit3->Text);
  xM = zolotoy(a, b, eps);
  Edit8->Text = FloatToStr(xM);
  Edit9->Text = FloatToStr(F(xM));
}
```

Форма проекту з результатами:

4.5 Приклад розв’язання задачі одновимірної оптимізації у Mathcad

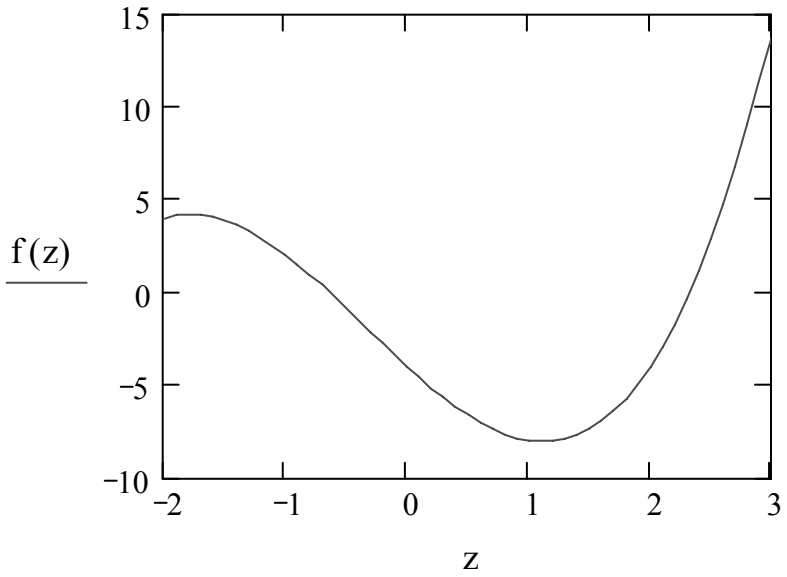
Лістинг пошуку мінімального значення функції за умовами завдання в п. 4.4:

Цільова функція

$$f(x) := x^2 - 6x - 4 + x^3$$

Побудова графіка функції

$$z := -2, -1.9 .. 3$$



Початкове значення для розв'язку

$$x := 0$$

Обмеження на аргумент

Given

$x > -2 \quad x < 3$

Обчислення мінімума

$$xmin := \text{Minimize}(f, x)$$

Розв'язок задачі оптимізації

$xmin = 1.12 \quad Fmin := f(xmin) \quad Fmin = -8.061$

4.6 Розв'язування задачі лінійного програмування (оптимізація з обмеженнями)

4.6.1 Постановка задачі

Знайти мінімум лінійної цільової функції

$$z = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$$

при виконанні наступних обмежень:

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \geq b_2 \\ \ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \geq b_m \end{array}\right\}$$

$$x_j \geq 0; \quad j = 1 \ldots n.$$

де c_i − коефіцієнти цільової функції (j = 1, 2, …,n), a_{ij} − коефіцієнти системи обмежень (i = 1, 2, …,m), b_i − значення вектору обмежень. Умова $x_i > 0$ свідчить про те, що значення вектору розв'язку задачі лінійного програмування не можуть бути від'ємними.

4.6.2 Приклад розв'язання задачі лінійного програмування в Mathcad

Завдання. Розв'язати в Mathcad задачу лінійного програмування (пошук максимального значення лінійної цільової функції з декількома змінними із заданою системою обмежень)

$$Z = 2x + 3y \;\Rightarrow\; \max,$$

при обмеженнях:

$$\begin{cases} -3x + 4y \leq 8 \\ x + 5y \geq 2 \\ 3x + 2y \leq 9 \end{cases}$$

$$x \geq 0, \qquad y \geq 0.$$

Лістинг розв'язання задачі лінійного програмування в Mathcad:

Цільова функція	Початкові значення розв'язку
$z(x, y) := 2 \cdot x + 3 \cdot y$	$x := 1 \qquad y := 1$

Закінчення лістингу розв'язання задачі лінійного програмування:

Обмеження

Given

$-3 \cdot x + 4 \cdot y \le 8$

$x + 5 \cdot y \ge 2$

$3 \cdot x + 2 \cdot y \le 9$

$x \ge 0 \qquad y \ge 0$

Обчислення максимума — Розв'язок

$xmax := Maximize(z, x, y) \qquad xmax = \begin{pmatrix} 1.111 \\ 2.833 \end{pmatrix}$

Максимальне значення цільової функції

$Fmax := z(xmax_0, xmax_1) \qquad Fmax = 10.722$

4.7 Транспортна задача лінійного програмування

4.7.1 Постановка задачі

Транспортна задача − це одна з типових задач оптимізації, де цільова функція та функції обмежень лінійні. Такі задачі мають назву задачі лінійного програмування. Транспортну задачу формулюють так. Маємо ***m*** **пунктів виробництва** A_1 *...,* A_m*,* які мають запаси однорідних продуктів у кількості a_1, ..., a_m одиниць. Маємо ***n*** **пунктів споживання** B_1*, ...,* B_n*,* потреби яких у цих продуктах складає b_1*, ...,* b_n одиниць. Відомі також **транспортні витрати** C_{ij}, перевезення одиниці продукту з пункту A_i в пункт B_j ($i = 1, ..., m; j = 1, ..., n$). Зазвичай повинно виконуватись рівняння

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j,$$

тобто загальний обсяг виробництва дорівнює загальному обсягу споживання. Необхідно **скласти такий план перевезень** (звідки, куди та скільки одиниць продукту везти), щоби забезпечити усі пункти споживання B_1*, ...,* B_n за рахунок реалізації усієї продукції, яку вироблено в пунктах A_1*, ...,* A_m, за мінімальною загальною вартістю усіх перевезень.

Запишемо математичну постановку задачі. Нехай x_{ij} **- кількість одиниць продукту**, який перевозять з пункту A_i в пункт B_j. Сумарні витрати на перевезення продуктів (Z) з усіх пунктів виробництва (A_i) до всіх пунктів споживання (B_j) запишемо формулою:

$$Z = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij} \cdot x_{ij}$$

Сумарна кількість продукту, який направлено від кожного пункту виробництва до всіх пунктів призначення, має дорівнювати запасу продукту в цьому пункті, тобто

$$\sum_{j=1}^{n} x_{ij} = a_i, \quad \text{де } i = 1,...,m.$$

Сумарна кількість вантажу, який перевозять до кожного пункту споживача з усіх пунктів відправлення, має дорівнювати потребам. Це умова повного задоволення потреб:

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad \text{де } j = 1,...,n.$$

Обсяги перевезень − невід'ємні числа, тому що перевезення з пунктів споживання до пунктів виробництва не можливі:

$$x_{ij} \geq 0,\ i = 1, ..., m;\ j = 1, ..., n.$$

Таким чином, транспортна задача − це визначення мінімальних сумарних витрат на перевезення вантажів при виконанні умов повного задоволення попиту та рівняння вивезеної кількості продукту до його запасів у пунктах відправлення.

4.7.2 Приклад розв'язання транспортної задачі у Mathcad

Завдання. На складах $A1$, $A2$, $A3$ зберігають $a1 = 100$, $a2 = 200$, $a3 = 120$ одиниць одного й того самого вантажу. Треба доставити його трьом споживачам $B1$, $B2$, $B3$, замовлення яких складають $b1 = 200$, $b2 = 110$, $b3 = 80$ одиниць вантажу відповідно. Вартість перевезення $C_{i,j}$ одиниці вантажу з i-го складу j-тому споживачеві наведено у таблиці:

	$b1 = 200$	$b2 = 110$	$b3 = 80$
$a1 = 100$	4	2	6
$a2 = 200$	7	5	3
$a3 = 120$	1	7	6

Необхідно знайти мінімальну вартість перевезень.

Для розв'язку цієї задачі, понад наявних n пунктів призначення $b1$, $b2$, $b3$ введемо ще один, фіктивний, пункт призначення $b4$, якому припишемо фіктивну заявку, значення якої дорівнює надлишку запасів над заявками. Вартість перевезень з усіх пунктів відправлення до фіктивного пункту призначення $b4$ вважатимемо нульовою. Введенням фіктивного пункту призначення $b4$ з його заявкою $b4$ ми зрівняли баланс транспортної задачі й тепер її можна розв'язувати

як звичайну транспортну задачу із правильним балансом (кількість перевезеного вантажу позначимо символами x_1, ..., x_n відповідно).

	$b1 = 200$	$b2 = 110$	$b3 = 80$	$b4 = 30$
$a1 = 100$	4 x_1	2 x_2	6 x_3	0 x_4
$a2 = 200$	7 x_5	5 x_6	3 x_7	0 x_8
$a3 = 120$	1 x_9	7 x_{10}	6 x_{11}	0 x_{12}

Подальші розрахунки виконаємо у Mathcad. Лістинг розв'язку задачі:

Задаємо початкові значення *x*:

ORIGIN := 1

i := 1..12

$x_i := 10$

Задаємо функцію загальної вартості перевезень:

$$F(x) := 4 \cdot x_1 + 2 \cdot x_2 + 6 \cdot x_3 + 0 \cdot x_4 + 7 \cdot x_5 + 5 \cdot x_6 + 3 \cdot x_7 + 0 \cdot x_8 + 1 \cdot x_9 + 7 \cdot x_{10} + 6 \cdot x_{11} + 0 \cdot x_{12}$$

Задаємо умови (обмеження):

Given

$$x_1 + x_2 + x_3 + x_4 = 100$$

$$x_5 + x_6 + x_7 + x_8 = 200$$

$$x_9 + x_{10} + x_{11} + x_{12} = 120$$

$$x_1 + x_5 + x_9 = 200$$

$$x_2 + x_6 + x_{10} = 110$$

$$x_3 + x_7 + x_{11} = 80$$

$$x_4 + x_8 + x_{12} = 30$$

$$x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \geq 0 \quad x_4 > 0$$

$$x_5 \geq 0 \quad x_6 \geq 0 \quad x_7 \geq 0 \quad x_8 \geq 0$$

$$x_9 \geq 0 \quad x_{10} \geq 0 \quad x_{11} \geq 0 \quad x_{12} \geq 0$$

Використовуючи функцію Minimize, знайдемо мінімальні значення x_1, ..., x_{12}.

Minimize(F, x) =

	1
1	0
2	100
3	0
4	0
5	80
6	10
7	80
8	30
9	120
10	0
11	0
12	0

Знайдемо мінімальну вартість перевезень:

$4 \cdot 0 + 2 \cdot 100 + 6 \cdot 0 + 0 \cdot 0 + 7 \cdot 80 + 5 \cdot 10 + 3 \cdot 80 + 0 \cdot 30 + 1 \cdot 120 + 7 \cdot 0 + 6 \cdot 0 + 0 \cdot 0 = 1.17 \times 10^3$

Результати заносимо у транспортну таблицю:

	$b1 = 200$	$b2 = 110$	$b3 = 80$
$a1 = 100$	4	2 $x_2 = 100$	6
$a2 = 200$	7 $x_5 = 80$	5 $x_6 = 10$	3 $x_7 = 80$
$a3 = 120$	1 $x_9 = 120$	7	6

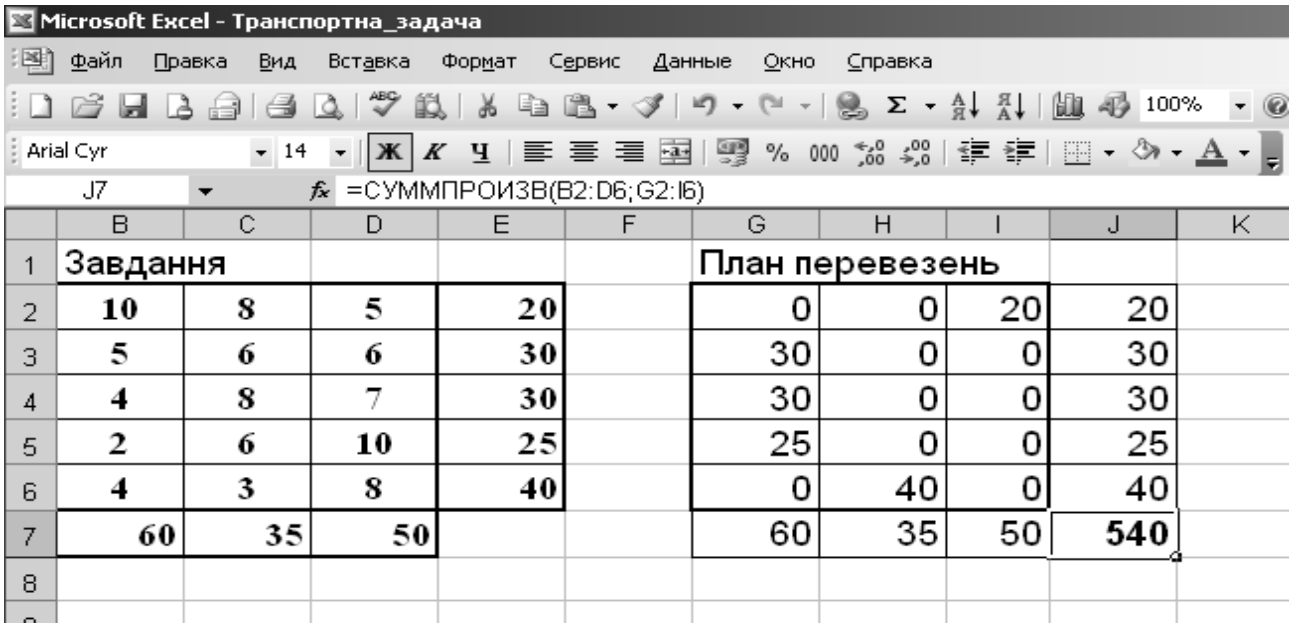
Мінімальна вартість перевезень**:** $F = 1170$.

4.7.3 Приклад розв'язання транспортної задачі в Excel

Завдання. Знайти мінімальну вартість перевезень вантажу від п'яти *Постачальників* до трьох *Пунктів* споживачів, за умов відомої кількості вантажу (запасів) у постачальників та потреб у вантажу в пунктах:

	Пункт 1	**Пункт 2**	**Пункт 3**	**Запаси**
Постачальник 1	10	8	5	20
Постачальник 2	5	6	6	30
Постачальник 3	4	8	7	30
Постачальник 4	2	6	10	25
Постачальник 5	4	3	8	40
Потреби	60	35	50	

Вікно Excel для розв’язання транспортної задачі (з результатами):



	B	C	D	E	F	G	H	I	J	K
1	Завдання					План перевезень				
2	10	8	5	20		0	0	20	20	
3	5	6	6	30		30	0	0	30	
4	4	8	7	30		30	0	0	30	
5	2	6	10	25		25	0	0	25	
6	4	3	8	40		0	40	0	40	
7	60	35	50			60	35	50	540	
8										

Пояснення.

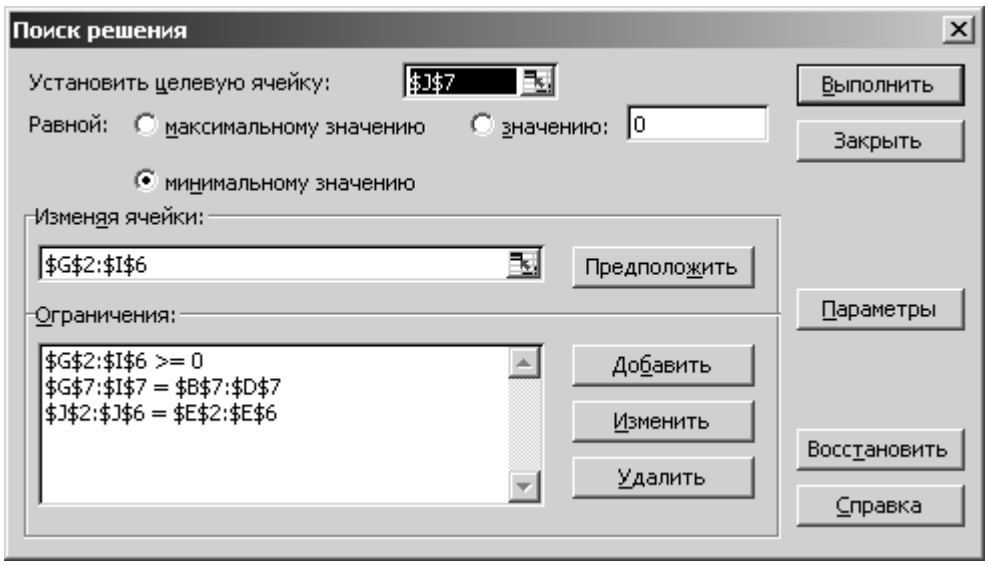
Значення запасів введено у комірки *E*2:*E*6, *значення потреб* − у комірки *B*7:*D*7, *значення вартостей перевезень* одиниці вантажу від постачальників до споживачів − у комірки *B*2:*D*6.

У комірках *J*2:*J*6 записуємо формулу для обчислення сум у рядках плану перевезень, у комірках *G*7:*I*7 записуємо *формулу для обчислення сум* у стовпчиках плану перевезень. Наприклад, у комірці *J*2 запишемо: = СУММ(*G*2:*I*2), у комірці *G*7 запишемо: = СУММ(*G*2: *G*6).

У комірку *J*7 записуємо *формулу вартості усіх перевезень вантажу*:

=СУММПРОИЗВ(*B*2:*D*6;*G*2:*I*6).

Далі відкриваємо через команду меню «Сервис» *вікно «Поиск решения»* та заповнюємо його наведеними адресами комірок («ячейка») та рівняннями обмежень («Ограничения»):



Після команди «Выполнить» одержимо у вікні Excel план перевезень у комірках *G*2:*I*6 та значення мінімальної вартості перевезень у комірці *J*7 (540).

Лабораторна робота № 4

Розв'зування задач пошуку екстремуму

1 ***Одновимірна оптимізація***

1.1 Визначити проміжки унімодальності (для мінімумів та максимумів) функції

$$f(x) = x^3 - x^2 - 4.4ax + 4a$$

на заданому проміжку [–4, 4]

1.2 Обчислити координати усіх точок екстремумів (локальних мінімумів та максимумів) у Mathcad та мовою С++ методом, відповідно до індивідуального завдання (табл. 4.1). Порівняти результати.

Таблиця 4.1 − Індивідуальні варіанти методів одновимірної оптимізації

Номери варіантів	**Методи оптимізації**
1, 5, 9, 10, 15, 16, 20, 22, 27, 28	Рівномірного пошуку
2, 4, 8, 11, 13, 18, 19, 23, 26, 30	Бісекції
3, 6, 7, 12, 14, 17, 21, 24, 25, 29	"Золотого перетину"

2 ***Задачі оптимізації з обмеженнями***

2.1 Розв'язати *графічно* і в *Mathcad* задачу лінійного програмування

$$Z = 2ax_1 + (1.5 + a)\, x_2 \;\Rightarrow\; \max,$$

при обмеженнях:

$$\begin{cases} -(a+2)x_1 + (5-a))x_2 \le 10 \\ x_1 + (3{,}5-a))x_2 \ge 3 \\ (a+1{,}8)x_1 + (4{,}1-a))x_2 \le 10 \\ \quad x_1 \ge 0, \quad x_2 \ge 0 \end{cases}$$

де $a = 0.1\,n$; n − номер індивідуального варіанта.

2.2 Знайти за допомогою *Excel* та *Mathcad* мінімальну вартість перевезень вантажу від постачальників у містах Київ, Львів, Одеса до споживачів у Сімферополі, Ужгороді, Вінниці, Луганську при відомих вартостях перевезення одиниці продукції, запасів продукції у постачальників і потребах споживачів у містах (у таблиці $a = 0.1n$, де n − номер індивідуального варіанта).

ПОСТА-ЧАЛЬНИКИ	СПОЖИВАЧІ				ЗАПАСИ
	Сімферополь	**Ужгород**	**Вінниця**	**Луганськ**	
Київ	$1 + a$	$7 - a$	$9 - a$	4	**$120 + 20a$**
Львів	1	2	6	5	**280**
Одеса	3	8	$1 + a$	2	**160**
ПОТРЕБИ	**$130 + 10a$**	**220**	**$60 + 10a$**	**150**	

2.3 При виробництві чотирьох видів кабелю виконується 5 груп технологічних операцій. Норми витрат на 1 км кабелю даного виду на кожній з груп операцій, прибуток від реалізації 1 км кожного виду кабелю, а також загальний фонд робочого часу, протягом якого можуть виконуватися ці операції, зазначено у таблиці:

Технологічна операція	**Норми витрат часу на обробку 1 км кабелю (годин)**				**Загальний фонд робочого часу (годин)**
	Марка 1	**Марка 2**	**Марка 3**	**Марка 4**	
Волочіння	$1.2 + a$	5	1.6	2.4	7 200
Накладення ізоляції	1	4	8	3	5 600
Скручування елементів у кабель	6.4	5.6	$6 + a$	$8 - a$	11 170
Оливування	$30 - a$	$1.2 + a$	1.8	2.4	3 600
Випробування і контроль	2.1	1.5	4	$3 + a$	4 200
Прибуток від реалізації 1 км кабелю	1.2	0.8	$1 + a$	1.3	

Визначити за допомогою *Excel* та *Mathcad* такий план випуску кабелю, при якому загальний прибуток від реалізації виготовленої продукції є максимальним. У таблиці використовується змінна $a = 0.1n$, де n − номер індивідуального варіанта.

5 КОМПЛЕКСНЕ ЗАВДАННЯ

“Дослідження функцій з використанням обчислювальних методів у Mathcad та мовою С++“

1 Записати програму обчислення таблиці значень та побудови графіка функції $f(x)$ у С++ та у Mathcad:

$$f(x) = a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0$$

Коефіцієнти a_i та інтервал табулювання $\lfloor x_{\min},\ x_{\max} \rfloor$ взяти з табл. 5.1 відповідно до номера варіанта індивідуального завдання. Крок табулювання – 0.2.

2 Визначити формулу наближеної функції $g(x)$ методом, обраним згідно з індивідуальним завданням (стовпчик 8 у табл. 5.1).

Обчислити значення наближеної функції $g(x)$ за обраною формулою в точках x_i та посередині поміж ними мовою С++ та у Mathcad. Побудувати графіки заданої та наближеної функцій в одній площині. Для побудови графіка наближеної функції у Mathcad обрати крок $h = 0.01$.

Обчислити середньоквадратичну похибку відхилення заданої та наближеної функцій

Зауваження. У якості таблиці точок функції для розрахунків обирати:

– у варіантах з методами інтерполяції в якості вихідних даних обрати по чотири пари значень x_i та $f(x_i)$ з обчисленої таблиці у п. 1 завдання

– у варіантах з методами апроксимації в якості вихідних даних обрати по шість пар значень x_i та $f(x_i)$ із обчисленої таблиці у п. 1 завдання.

Пари точок обирати граничні та рівномірно розташовані всередині заданого інтервалу.

3 Знайти інтервали унімодальності функції $f(x)$. Обчислити усі координати точок екстремумів функції $f(x)$ у Mathcad та мовою С++ на заданому інтервалі методом, відповідно індивідуального завдання (стовпчик 9 у табл. 5.1)

Таблиця 5.1− Варіанти індивідуальних завдань

№ з/п	a_0	a_1	a_2	a_3	a_4	**Проміжок** $[x_{\min},\ x_{\max}]$	**Метод наближення функції**	**Метод пошуку екстремуму**
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
1	0	2	–1	–7	5	[–1, 1]	лінійна інтерполяція	рівномірного пошуку
2	2	8	6	15	8	[–2, 2]	інтерполяція Лагранжа	бісекції
3	3	4	0	3	–5	[–2, 2]	інтерполяція алгебраїчним поліномом	“золотого перетину”
4	0	3	–2	10	–5	[–1.5, 3]	апроксимація поліномом 2-го степеня	“золотого перетину”

Продовження табл. 5.1

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
5	1	–2	–8	12	–3	[–1, 1.5]	апроксимація поліномом 3-го степеня	рівномірного пошуку
6	1	10	–7	21	–2	[–1, 1.5]	лінійна інтерполяція	бісекції
7	1	3	3	0	–5	[–1, 1.5]	інтерполяція Лагранжа	“золотого перетину”
8	1	3	–2	1	–5	[–1, 1.5]	інтерполяція алгебраїчним поліномом	“золотого перетину”
9	1	–3	8	0	–9	[–1.25, 1]	апроксимація поліномом 2-го степеня	рівномірного пошуку
10	1	2	–2	3	–3	[–1, 1.5]	апроксимація поліномом 3-го степеня	бісекції
11	0	1	0	–5	–3	[–2, 1]	лінійна інтерполяція	бісекції
12	1	2	–1	3	–4	[–1.2, 2]	інтерполяція Лагранжа	“золотого перетину”
13	0	3	0	–6	2	[–1, 2]	інтерполяція алгебраїчним поліномом	рівномірного пошуку
14	1	3	–4	5	–5	[–1, 1.5]	апроксимація поліномом 2-го степеня	бісекції
15	0	3	3	0	–15	[–1, 1.5]	апроксимація поліномом 3-го степеня	рівномірного пошуку
16	2	20	–11	19	–22	[–0.5, 1.5]	лінійна інтерполяція	бісекції
17	0	2	–1	–7	5	[–0.3, 1.5]	інтерполяція Лагранжа	золотого перетину
18	1	3	–4	–7	5	[–0.5, 1.5]	інтерполяція алгебраїчним поліномом	“золотого перетину”
19	1	2	–1	1	–1	[–1, 2]	апроксимація поліномом 2-го степеня	рівномірного пошуку
20	1	2	–3	–4	–5	[–1, 1]	апроксимація поліномом 3-го степеня	бісекції
21	0	1	–3	–4	1	[–1, 1]	лінійна інтерполяція	бісекції
22	1	1	6	–7	–4	[–1, 1]	інтерполяція Лагранжа	“золотого перетину”
23	0	1	–3	1	1	[–0.5, 1]	інтерполяція алгебраїчним поліномом	рівномірного пошуку
24	–2	2	–2	3	8	[–1.5, 1]	апроксимація поліномом 2-го степеня	бісекції
25	0	4	0	2	–3	[–1, 2]	апроксимація поліномом 3-го степеня	“золотого перетину”
26	0	4	–4	6	–6	[–0.5, 1.5]	лінійна інтерполяція	рівномірного пошуку

Закінчення табл. 5.1

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
27	0	5	0	2	–11	[–0.5, 1]	інтерполяція Лагранжа	бісекції
28	1	4	–3	2	–5	[–0.5, 1.3]	інтерполяція алгебраїчним поліномом	бісекції
29	0	1	3	0	–1	[–2, 2]	апроксимація поліномом 2-го степеня	“золотого перетину”
30	1	2	–7	6	–4	[–1, 1.5]	апроксимація поліномом 3-го степеня	рівномірного пошуку
31	1	0	–12	1	–4	[–1, 1]	лінійна інтерполяція	бісекції
32	0	3	1	2	–3	[–1, 2]	інтерполяція Лагранжа	рівномірного пошуку

4 *Оптимізація з обмеженнями (задача лінійного програмування).*

При виготовленні тари для бандеролей та посилок використовують три види ресурсів, запаси яких обмеженні. Норми витрат ресурсів на кожний виріб, ціна одиниці виробу та обмеження на запаси ресурсів подано нижче (a=0,1·n+2, де n – номер варіанту):

Ресурси	**Норми витрат на 1 од. виробу**		**Запаси ресурсу**
	Бандеролі	**Посилки**	
Картон	2+a	8–a	360
Трудові витрати	6–a	4	192
Електроенергія	2	3+a	180
Прибуток за 1 од. виробу	12	16	

Скласти план випуску виробів (кількість бандеролей та посилок), що забезпечує максимальний прибуток за вартістю. Завдання виконати графічно, в Mathcad і Excel.

5 *Оптимізація з обмеженнями (Транспортна задача).*

Розв’язати в Excel задачу перевезення вантажу з вузлів зв'язку на пункти сортування. У відділеннях зв'язку $ВЗ_1$, $ВЗ_2$, $ВЗ_3$, $ВЗ_4$, $ВЗ_5$, $ВЗ_6$ накопичено вантажі відповідно в об’ємах: **20 +10*n*, 30, 30 + 10*n*, 40, 25, 15** (центнерів), де ***n*** − номер індивідуального варіанта. Пункти сортування $ПС_1$, $ПС_2$, $ПС_3$ можуть прийняти вантажі у об’ємах, що становлять відповідно: **60+10*n*, 50+10*n*** і **50** (центнерів). Відомі вартості перевезень одного центнера вантажу з кожного відділення зв'язку в кожний пункт сортування (матриця вартостей):

	$ПС_1$	$ПС_2$	$ПС_3$
$ВЗ_1$	10	8	5
$ВЗ_2$	5	6	6
$ВЗ_3$	4	8	7
$ВЗ_4$	11	4	5
$ВЗ_5$	2	6	10
$ВЗ_6$	4	3	8

Визначити за допомогою Excel та Mathcad такий план перевезень, щоб його сумарна вартість була мінімальна, й при цьому вантаж був би вивезений із всіх відділень, а кожний пункт сортування одержав необхідну кількість вантажу.

Вимоги щодо оформлення комплексного завдання

1) Комплексне завдання оформлюється в окремому зошиті чи на аркушах формату А4, які підшиті у швидкозшивач. Комплексне завдання пишуть від руки, але графіки результатів обчислень, які виконано на комп'ютері, можна роздрукувати та вклеїти (чи вставити) як рисунки. Сторінки роботи мають бути пронумеровані.

2) Слід заповнювати лише один бік аркуша із залишенням полів для зауважень викладача.

3) Для *кожної задачі* комплексного завдання треба записати такі розділи:

а) умова задачі з індивідуальним завданням;

б) короткі теоретичні відомості;

в) опис розв'язування задачі на комп'ютері;

г) результати обчислень на комп'ютері (після виконання програми на комп'ютері);

д) аналіз результатів та висновки.

4) Схеми алгоритмів програм слід виконувати олівцем під лінійку у відповідності до ЄСПД.

5) Наприкінці роботи треба подати список використаної літератури, поставити підпис та дату виконання роботи.

ЛІТЕРАТУРА

1 **Дьяконов В. П.** Mathcad 8 PRO в математике, физике и Internet / В. П. Дьяконов, Н. В. Авраменко. – М. : Нолидж, 1999. – 512 с.

2 **Єщенко А. І.** Основи програмування в математичному пакеті Mathcad / А. І. Єщенко, І. А. Єщенко. – Одеса: УДАЗ, 2000. – 285 с.

3 **Краскевич В. Е.** Численные методы в инженерных исследованиях / Краскевич В. Е. , Зеленский К. Х., Гречко В. И. − К. : Вища школа, 1986. – 263 с.

4 **Крячков А. В.** Программирование на С++. Практикум: учеб. пособие для вузов / Крячков А. В., Сухинина И. В., Томшин В. К. − М. : Горячая линия – Телеком, 2000. – 344 с.

5 **Леонов Ю. Г.** Программирование инженерных задач: метод. пособие / Леонов Ю. Г., Силкина Н. В. , Шпинова О. Д. – Одесса : ОНАС, 2002. – 68 с.

6 **Шаповаленко В. А.** Чисельні методи та моделювання на ЕОМ: навч. посіб. Модуль 1 / Шаповаленко В. А., Буката Л. М., Трофименко О. Г. – Одеса : ВЦ ОНАЗ, 2010. – Ч. 1. – С. 95.

7 **Шаповаленко В. А.** Чисельні методи моделювання об’єктів: метод. вказівки для лаб. та практ. занять. Модуль 1 / Шаповаленко В. А., Буката Л. М., Трофименко О. Г. – Одеса : ВЦ ОНАЗ, 2010. – Ч. 2. – 85 с.

8 **Поддубный Г. В.** Численные методы и их применение в инженерных расчетах: учебн. пособие / Г. В. Поддубный, Л. И. Соколов. – Одесса : ОЭИС, 1981. – Ч. 1. – 85 с.

9 **Поддубный Г. В.** Численные методы и их применение в инженерных расчетах: учебн. пособие / Г. В. Поддубный, Л. И. Соколов. – Одесса : ОЭИС, 1983. – Ч. 2. – 85 с.

10 **Фельдман Л. П.** Чисельні методи в інформатиці / Фельдман Л. П., Петренко А. І., Дмитрієва О. А. – К. : BHV, 2006. – 480 с.

11 **Мэтьюз Д.Г.** Численные методы. Использование MATLAB; пер. с англ. / Д. Г. Мэтьюз, К. Д. Финк. – М. : Вильямс, 2001. – 720 с.

12 **Дьяконов В. П**. MATLAB 6/6.1/6.5+Simulink 4/5. Основы применения – М. : Солон-Пресс, 2002. – 768 с.

ЗМІСТ

Здано в набір 15.03.2011 Підписано до друку 28.04.2011
Формат 60x90/16 Зам. № 4511
Наклад 500 прим. Обсяг 4,5 друк. арк.
Віддруковано на видавничому устаткуванні фірми RISO
у друкарні редакційно-видавничого центру ОНАЗ ім. О.С. Попова
м. Одеса, вул. Ковалевського, 5
Тел. 720-78-94
ОНАЗ, 2011