Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

О.В.Узунов

ОСНОВИ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ФІЗИЧНО РІЗНОРІДНИХ СИСТЕМ ТА ЇХ ЕЛЕМЕНТІВ

Лабораторний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «автоматизовані та роботизовані механічні системи»
спеціальностями G9 (131) «Прикладна механіка», G11 (133) «Галузеве машинобудування»,

Електронне мережеве навчальне видання

Київ
“КПІ ім. І. Сікорського”
2026

УДК 681.52:[004.896+004.94]
У34

Автор: *Узунов Олександр Васильович*, д-р техн. наук, проф.

Рецензент: Шевченко О.В., д-р техн. наук, проф.,
НН ММІ, НТУУ «КПІ ім..Ігоря Сікорського»

Відповідальний
редактор Коваль О.Д., к.т.н., доц.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 6 від 28.03.2024 р.)
за поданням вченої ради навчально-наукового механіко машинобудівного інституту
(протокол № 8 від 25.03.2024 р.)

Узунов О.В.

У34 Основи математичного моделювання фізично різнорідних систем та їх елементів [Електронний ресурс] : лаб. практикум : навч. посіб.для здобувачів ступеня бакалавра за освіт. програмою «автоматизовані та роботизовані механічні системи» спец. G9 (131) «Прикладна механіка», G11 (133) «Галузеве машинобудування»; КПІ ім. Ігоря Сікорського. – Електрон. текст. дані (1 файл). – Київ : КПІім. Ігоря Сікорського, 2026. – 118 с. Друге видання. Перероблене і доповнене.

Лабораторний практикум ґрунтується на застосуванні мов програмування С (MatLab) та Python, які представлені з орієнтацією на розробку комп'ютерних програм для виконання інженерних розрахунків та моделювання процесів в фізично різнорідних системах та їх елементах. Задачі математичного моделювання розглядаються в поєднанні з фізичним моделюванням, що створює основу для підвищення точності розроблених математичних моделей. Представлено комплект лабораторних робіт, який рекомендовано для використання в курсі «Основи математичного моделювання фізично різнорідних систем», Практикум призначений для здобувачів ступеня бакалавра за спеціальностями G9 (131) «Прикладна механіка», G9 (133) «Галузеве машинобудування», і буде також корисним студентам машинобудівних спеціальностей вищих учбових закладів

УДК 681.52:[004.896+004.94]

Реєстр. № НП 23/24-426. Обсяг 7.36 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

Зміст

ВСТУП

В межах лабораторного практикуму з дисципліни «Основи математичного моделювання фізично різнорідних систем» передбачено проведення лабораторних робіт. Роботи мають виконуватись як на комп'ютерах, так і на лабораторних стендах. При роботі на комп'ютерах вирішують задачі які спрямовані на розроблення програм для моделювання. Лабораторні стенди використовуються для фізичного моделювання і експериментального визначення характеристик елементів та систем.

Метою лабораторних робіт є: поглиблення розуміння принципів будови фізично різнорідних елементів та систем; отримання навичок та досвіду розроблення програм виконання розрахунків та моделювання; моделювання дії і дослідження фізичних елементів та систем; представлення дії фізичних елементів за допомогою математичних моделей.

Завдання для лабораторних робіт сформовані таким чином, щоб студенти, які не мають навичок і досвіду програмування, змогли від ознайомлення з процесом створення програм перейти до вирішення практичних задач.

Серед цих задач є такі, як візуальне представлення результатів; забезпечення взаємодії розробника або користувача з активованою програмою; розробка алгоритмів розрахунків та моделювання; керування ходом обчислювального процесу; використання стандартних процедур моделювання динамічних процесів. Частина завдань лабораторних робіт стосуються фізичного моделювання дії елементів та систем в цілому. Вони мають допомогти студентам зрозуміти зв'язок між теорією та практикою. В ході виконання робіт студенти мають навчитися проводити експерименти для визначення характеристик елементів, навчитися представляти характеристики математичними залежностями, а також перевіряти точність отриманих результатів. Такий підхід має підготувати студентів до вирішення більш складних та спеціальних задач моделювання.

Поступове ускладнення завдань лабораторних робіт передбачає також і самостійну роботу студентів. Вона полягає у попередній домашній підготовці до лабораторних занять, розвинені творчого підходу при розробці алгоритмів, програм, принципових схем стендів та методик проведення досліджень. Також студенти мають самостійно оформлювати протоколи виконаних робіт, В протоколах мають бути відображені тема роботи, основні відомості, постановка задачі, хід вирішення задач, отримані результати, їх аналіз та сформульовані висновки. Результати виконання кожної лабораторної роботи з відповідним протоколом мають бути захищені.

1. ЗАГАЛЬНІ ВІДОМОСТІ

1.1. Фізично різнорідні системи та задачі моделювання

Невід'ємною частиною підготовки фахівців з машинобудування є моделювання технічних систем. Під моделюванням зазвичай розуміють імітацію дії системи за допомогою її моделі в штучних умовах. Це може бути як фізичне так і математичне моделювання. У будь якому випадку вирішення задач моделювання стосується конкретного об'єкту, який, як правило, має певні особливості своєї будови і також особливості моделювання.

В подальшому, будемо розглядати об'єкти, які представляють собою фізично різнорідні технічні системи конкректного призначення. До фізично різнорідних систем відносять системи, які мають у своєму складі компоненти різної природи: - механічні, газові, рідинні, електричні, електронні та програмні. Складність системи, типи компонентів, їх кількість і зв'язки між ними залежать від функціонального призначення технічної системи. Для формування глибшого уявлення про такі системи треба визначити призначення окремих компонентів. На загальному рівні таких призначень є кілька і вони стросуються дій над потоками інформації і енергії, які проходять через систему. Вказані дії по відношенню до інформації або енергії є наступними: отримання; перетворення; розвітлення потоків; з'єднання потоків; перенесення або транспортування. Розглянемо кілька прикладів виконання дій компонентами різної природи. Наприклад, тверде тіло, як механічний компонент, у разі його підняття, накопичить певну кількість потенційної енергії. В цьому випадку можна стверджувати, що тверде тіло при його підніманні переносить енергію з початкового місця розташування до кінцевої позиції. Також, у разі прикладанні постійної сили до твердого тіла, воно перетворює перетворює потенційну енергію в кінетичну. Інший фізичний компонент – камера, при її наповнені газом або рідиною під тиском, змінює кінетичну енергію потоку рідини або газу на потенційну енергію тиску в середині камери. Також при витіканні рідини або газу з камери під дією внутрішнього тиску потенційна енергія буде перетворена на кінетичну і вона також буде перенесена на певну відстань по трубопроводу разом з носієм – рідиною або газом. Аналогічним чином, можна розглядати і електричні компоненти, наприклад, електричний конденсатор може перетворювати енергію потоку електронів в електричний потенціал і створювати можливість для її перенесення по електричним каналам.

Використання подібних компонентів в технічних системах потребує розрахунків або моделювання їх дії. В свою чергу, розрахунки і моделювання ґрунтується на фізичних законах, яким підпорядковується дія компонентів. Наприклад, дії механічних компонентів відповідають законам Ньютона і Гука, а дії рідинних компонентів відповідають законам Паскаля, Бернуллі, Ейлера та ін..

Для виконання певних функцій у складі системи компоненти різної природи можуть поєднуватись в модулі. Для прикладу, розглянемо модуль – гідравлічний циліндр, який комбінує в собі два типи фізично різнорідних компонентів (рис.1.1). В ньому задіяні гідравлічні та механічні компоненти – рідина та тверде тіло. При подаванні рідини під тиском в ліву порожнину гідроциліндру при нерухомому поршні в ній відбувається збільшення тиску. Тиск за законом Паскаля починає діяти на всі поверхні внутрішнього об'єму лівої порожнини циліндру. Результатом дії цього тиску на площу поршня є осьова сила, яка, відповідно до закону Ньютона, змушує поршень рухатись. При цьому рідина з правої порожнини буде витискатися до баку. Функцією такого модулю є перетворення гідравлічної енергії в механічну. Аналогічним чином

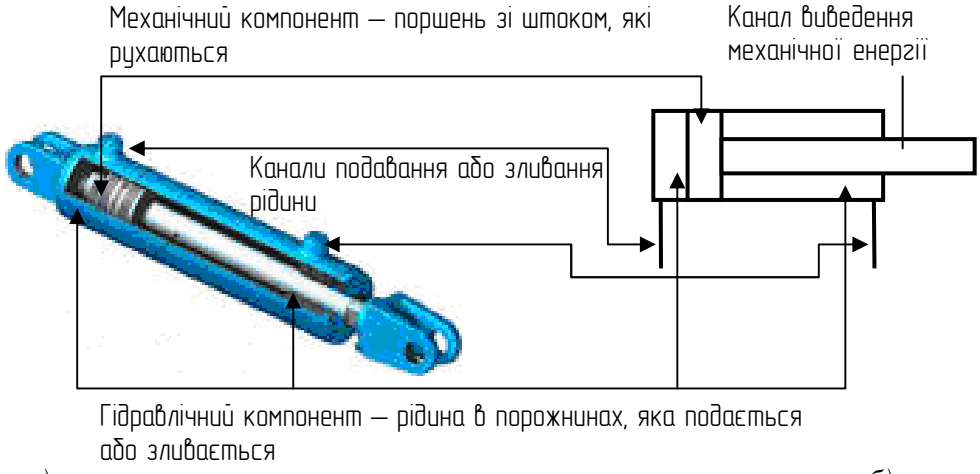


Рис.1.1. Гідравлічний циліндр як приклад фізично різнорідного модулю: а – зовнішній вигляд; б – принципове позначення на схемах

може бути розглянуто також взаємодію інших за природою компонентів. Наявність компонентів різної природи у складі технічної системи, наприклад, одночасно механічних, гідравлічних та електричних, призводить до необхідності враховувати і відповідний комплект фізичних законів. Це суттєво ускладнює вирішення задач моделювання.

Особливостями фізично різнорідних систем, які розглядаються в цьому курсі, є безперервність їх дії – за своїми функціональними ознаками це

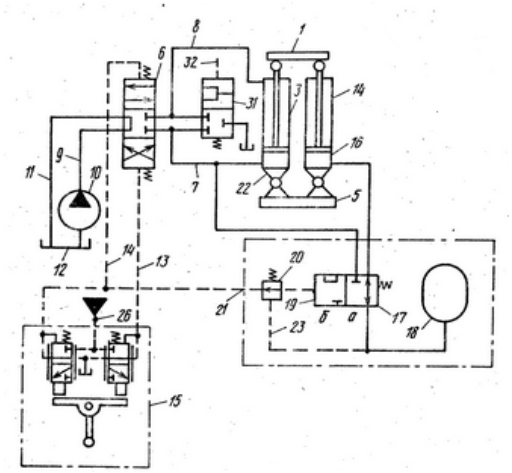
системи стабілізації, наприклад, рівня води в парогенераторі, або частоти обертання газової турбіни; системи програмного керування, наприклад, система катапультування пілота; системи слідкуючого типу, наприклад система керування рулем висоти літака. Складність проектування таких систем призводить до значних витрат часу та ресурсів. Математичне моделювання дозволяє суттєво скоротити ці витрати.

Одною з основних задач моделювання є визначення взаємозв'язку параметрів технічних систем з їх динамічними характеристиками. Дисципліна орієнтована на вирішення задач побудови математичних моделей і дослідження характеристик компонентів в залежності від їх параметрів та умов експлуатації. Також формується підґрунтя до розуміння основ побудови математичних моделей систем в цілому.

1.2. Приклади систем, проблеми та підхід до їх вирішення

Фізично різнорідні технічні системи широко використовують в промисловому обладнанні та машинах різного призначення. Наприклад, це екскаватори (рис.1.2 а), транспортні та військові літаки (рис.1.3).

а)



б)

Рис.1.2. Зовнішній вигляд екскаватора (а)[1] і схема його фізично різнорідної системи керування (б)[2]

Призначенням таких систем є забезпечення можливостей операторам або пілотам керувати машинами у всіх робочих та екстремальних режимах і можливих умовах експлуатації. Безпосередньо технічні системи забезпечують зв'язок між оператором (пілотом) та керуючими органами машин, або керування в автоматичному режимі. Для розуміння дії таких систем їх представляють за допомогою принципових схем, які містять в собі графічні

позначення фізично різнорідних елементів і відображають зв'язки між ними. Прикладами таких схем є система керування екскаватором (рис.1.2 б) або одна з систем керування літаком (рис.1.4).

В схемі (рис.1.2 б) представлені механічні та гідравлічні компоненти, а в схемі (рис.1.4) представлені також електричні, електронні та програмні компоненти.

а) б)

Рис.1.3. Вигляд транспортного літака 74ТК-200 ("Антонов-Канада")[3 (а) та військового літака F-16[4] (б), які оснащені фізично різнорідними слідкуючими системами керування рулевими поверхнями

Складність проектування таких систем обумовлена суттєвими відмінностями між компонентами різної природи, яка полягає різноманітності роочих процесів, як з точки зору фізичних законів, так і з точки зору їх часових характеристик. Це призводить до значних часових витрат на

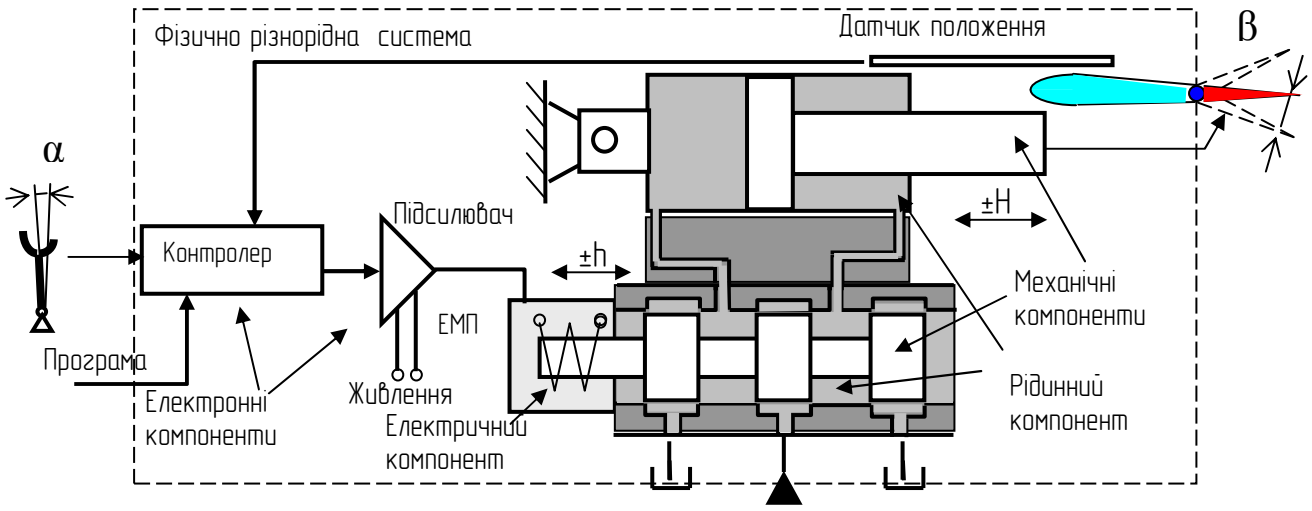


Рис. 1.4. Приклад схеми системи керування рулем літака, яка містить фізично різнорідні компоненти

узгодження вказаних процесів. Застосування методів математичного

моделювання дозволяє зменшити або виключити використання експериментального обладнання та фізичних зразків систем.

Для моделювання дії компонентів і систем в цілому, а також для виконання інженерних розрахунків широко використовують програмні засоби. Програмні засоби доволяють автоматизувати розрахунки та моделювання завдяки моливостям формувати завдання і передавати їх на виконання в операційне середовище комп'ютера.

1.3. Загальний процес розроблення програм моделювання

Задачами дисципліни є навчити студентів основам побудови математичних моделей фізично різнорідних систем та їх елементів, дослідженню процесів їх дії, а також основам побудови програм автоматизації розахунків. В свою чергу, задача побудови математичних моделей і автоматизації розрахунків має дві складові частини: 1 - математичне представлення моделі або розрахунків; 2 – забезпечення їх виконання в комп'ютерному середовищі. В цьому матеріалі увагу приділено саме другій складовій частині.

На загальному рівні технологія вирішення другої частини задачі виглядає наступним чином (рис.1.5) - розробник програм на основі заданого алгоритму розрахунку або алгоритму роботи моделі створює програму. Після цього програму передають в операційну систему комп'ютера і запускають. В цей момент запускається Обробник програм, який має

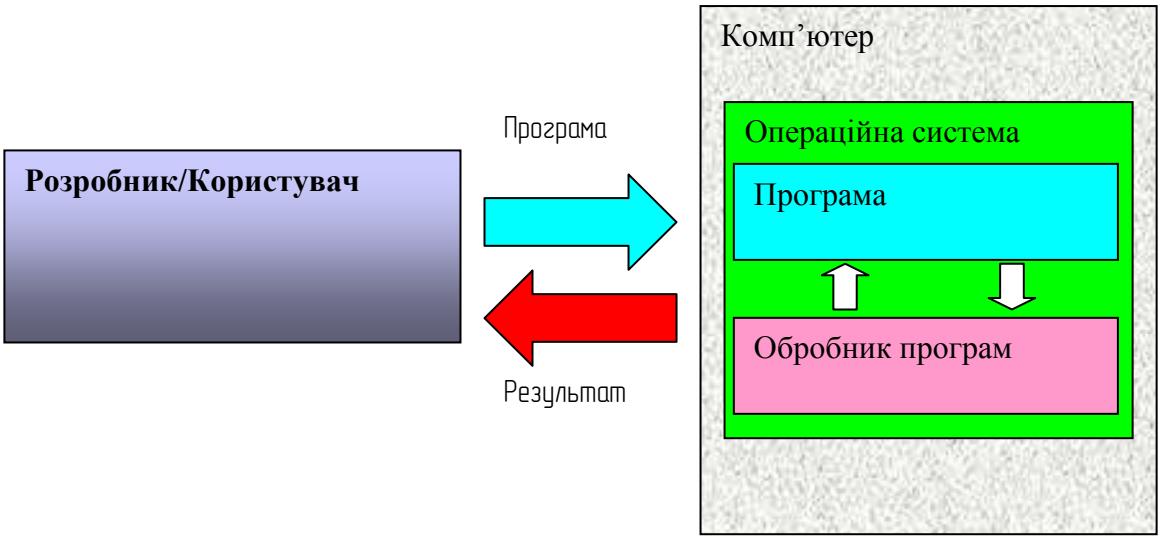


Рис.1.5. Загальна схема взаємодії Розробника програм з Обробником програм в операційному середовищі комп'ютера

виконувати задані в програмі команди. Функціонування програми в

операційній системі може супроводжуватись впливом на роботу програми в процесі її виконання. Це забезпечується комунікаціями між Розробником та Обробником. Коректність роботи програми і коректність результату залежать від взаєморозуміння між Розробником і Обробником програми. В основі взаєморозуміння лежить однаковість трактування команд мови програмування.

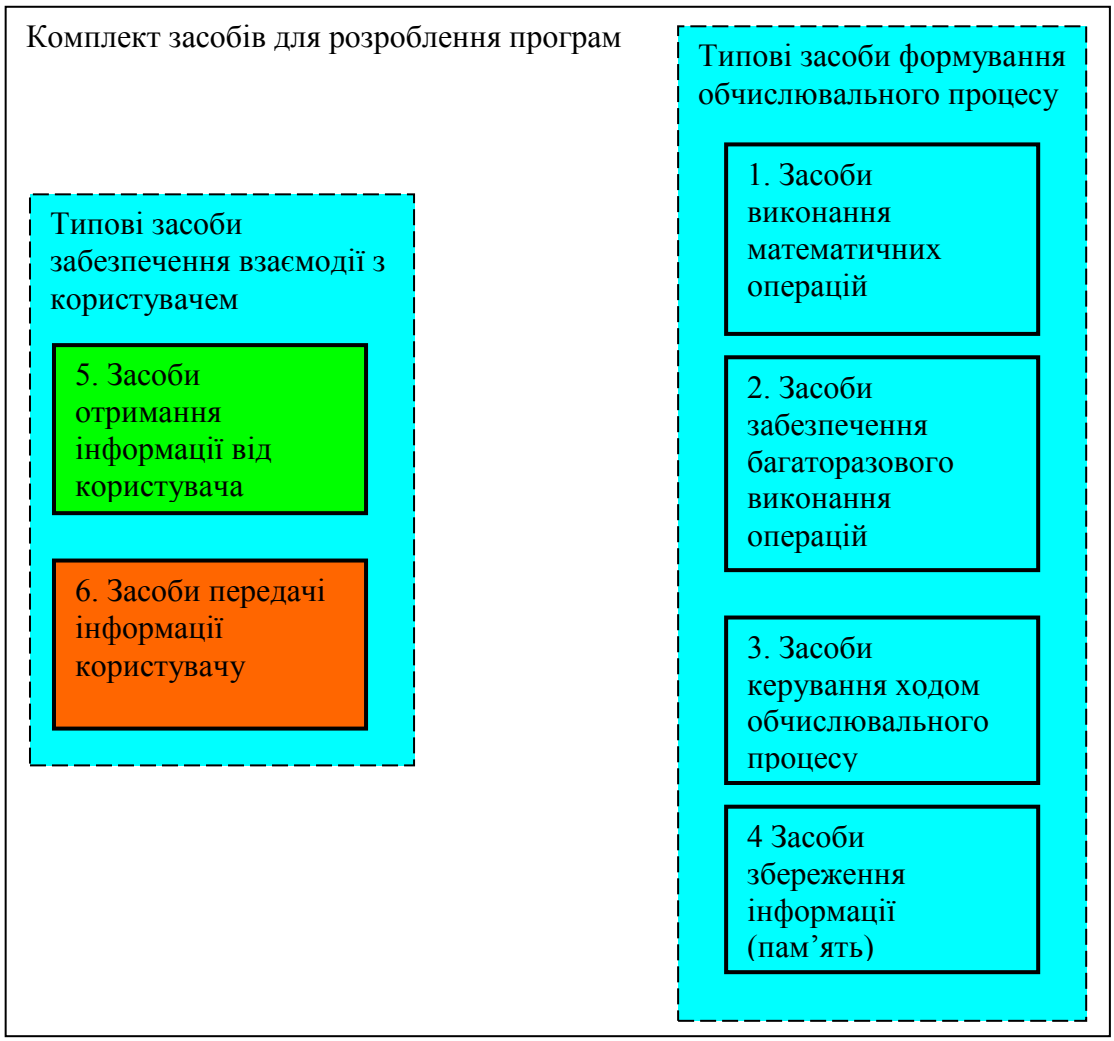


Рис. 1.6. Комплект типових засобів, які використовуються для розроблення програм інженерних розрахунків і моделювання

Мова програмування є інструментом для підготовки розрахунків та математичних моделей до роботи в комп'ютерному середовищі. Мов програмування існує значна кількість. Вони дозволяють вирішувати широке коло задач. Представлений нижче погляд на мову програмування, на наш погляд, є зручним для вирішення саме задач розробки програм для виконання розрахунків і моделювання технічних систем. Цей погляд полягає в наступному: мова програмування представляє комплект певних засобів побудови програм. Всі засоби розподілені у дві типові групи: - засоби

формування обчислювального процесу та засоби забезпечення взаємодії з користувачем (рис.1.6). Типовими засобами фрмування обчислювального процесу є засоби для: виконання математичних операцій; забезпечення багаторазового виконання операцій; керування ходом обчислювального процесу; збереження інформації або пам'ять. В свою чергу, типові засоби взаємодії з користувачем об’єднують засоби отримання інформації від користувача та засоби передачі інформації користувачеві.

Вибір потрібних засобів залежить від конкретної задачі. Це робиться наступним чином. Спочатку розробляється алгоритм, який має відобразити два аспекти - дії, які потрібні для розрахунку, та черговість їх виконання. Після цього, для визначених дій вибираються потрібні засоби іх виконання, і вони організуються в процес відповідно до алгоритму (рис.1.7). Засоби виконання дій реалізуються відповідними командами вибраної мови програмування. Алгоритм забезпечений командами представляє код програми.

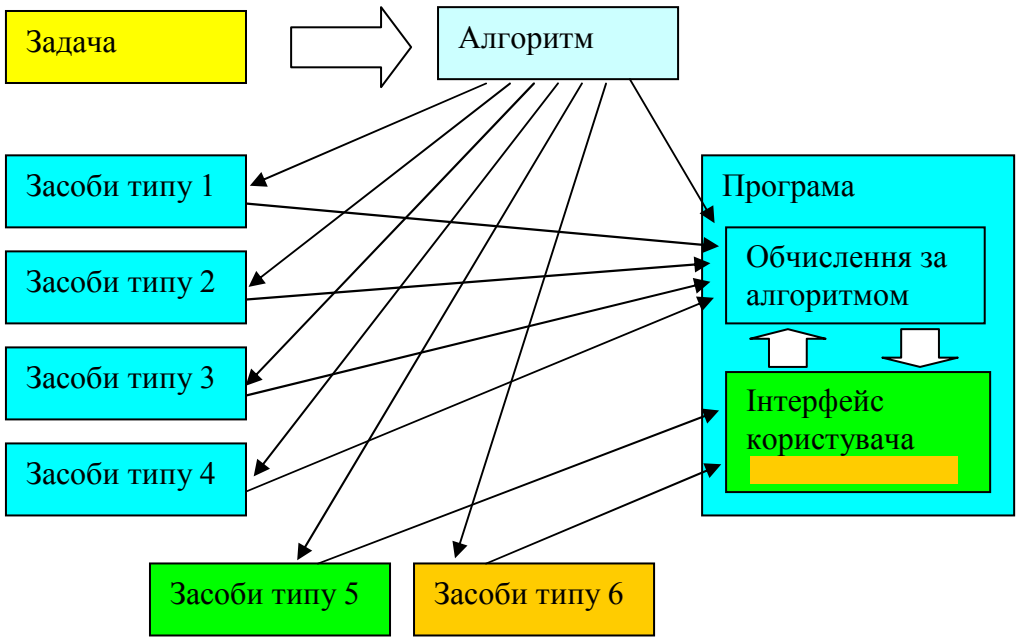


Рис. 1.7 Схема процесу розробки програм виконання інженерних розрахунків і моделювання

Вибір тієї або іншої мови залежить від вподобань розробника. Популярними мовами для вирішення інженерних задач і моделювання є мова програмування С, яка з деякими змінами покладена в основу пакету MatLab та мова Python.

2. ІНСТРУМЕНТ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ MATLAB

2.1. Засоби пакету MatLab для обчислень та моделювання [5]

2.1.1. Засоби формування обчислювального процесу

2.1.1.1. Засоби виконання обчислювальних операцій

Автоматизація виконання дій, по перше, потребує наявність засобу, що потенційно може ці дії виконувати, а по друге, потрібні також засоби, що дозволяють однозначно пояснити що, коли і як потрібно робити. Перший засіб умовно може бути названий «Обробником програм», а другий – «мовою спілкування» з Обробником.

Дії, які потрібно автоматизувати, за допомогою мови спілкування фіксуються у файлі у вигляді тексту. Цей файл передається для виконання обробнику програм. Обробник програм виконує дії, які передбачені у тексті програми. Такими діями можуть бути обчислювальні операції, повторення операцій, аналіз проміжних результатів та прийняття рішень про вибір і виконання наступних дій.

Обчислювальні операції мови програмування відповідають математичним операціям. Кожна математична операція може бути замовлена для виконання відповідним символом. Математичні операції застосовують в комбінації звичайних констант та змінних параметрів і позначаються наступним чином:

+ - додавання;
- - віднімання;
* - множення;
/ - розділення;
^n - взведення у ступінь n.

При необхідності застосування математичних операцій для змінних у векторному вигляді потрібно використовувати іншу групу математичних операцій, що позначаються додатковою точкою перед символом операції, наприклад, для взведення вступінь це виглядає наступним чином:

.^n - взведення векторної змінної у ступінь n.

2.1.1.2. Засоби забезпечення багаторазового виконання операцій

За потреби багаторазового використання однакових операцій в обумовленій черговості, застосовуються параметри та спеціальні ключові слова, що задають відповідні дії обробника програми.

У якості параметрів використовуються ідентифікатори. Ідентифікатори представляються у вигляді позначення, що записується як єдиний буквений символ, кілька символів (без пробілу поміж ними) або набору символів та цифр.

Приклад: x, x1, x3, tz та ін.

Ключові слова які передають обробнику програми завдання про багаторазове (циклічне) виконання потрібного переліку операцій застосовуються парами: перше ключове слово та заключне ключове слово. У тексті програми вони можуть додатково позначатися іншим кольором. Операції, що розташовані між ними обробник програм буде виконувати задану кількість разів.

Потрібна кількість обчислень вказується за допомогою запису, що слідує безпосередньо за першим ключовим словом. Запис складається з параметру циклу, якому присвоюється початкове значення, символу, що є роздільним, значення, що завдає крок прирощення параметру циклу, ще одного роздільного символу та кінцевого значення параметру циклу.

Приклад.

```
for x=0:1:10
    y=x^2;
end
```

Пояснення. Ключові слова **for** і **end**. Операція, що потребує багаторазового виконання y=x^2; Параметр циклу x. Початкове значення параметру 0. Крок прирощення параметру 1. Кінцеве значення 10. У разі виконання наведеного прикладу обробник програми виконає вказану операцію 11 разів.

Існує варіант застосування іншої пари ключових слів **while** та **end**, що також передає обробнику програм завдання на багаторазове виконання операцій.

Приклад.

```
x=0;
while x<13
```

```
y=x^2;
x=x+1;
end
```

Пояснення. Вказані ключові слова забезпечують багатократність виконання операції y=x^2.

Операції x=0; перед початком циклу та x=x+1; в середині циклу забезпечують отримання початкового значення та прирощення параметру циклу при кожному проході циклу.

2.1.1.3. Засоби керування ходом обчислювального процесу

Для керування ходом обчислювального процесу застосовуються відповідні ключові слова.

Такими словами є **if else elseif** та **end**. Є різні варіанти їх використання.

Перше ключове слово передбачає завдання умови для перевірки і опис операції, яка потребує виконання у разі відповідності умові. У разі не відповідності умові операцію буде пропущено.

Наприклад:

```
if x<6  y=x+2; end
```

Пояснення. Обробник програм під час виконання наведеного рядка програми перевіряє значення **x.** Якщо значення **x** менше 6, то виконується операція y=x+2, в іншому випадку операція пропускається.

Комбінація першого, другого і четвертого ключових слів дозволяє задати можливість виконання альтернативної операції.

Приклад:

```
if x<6  y=x+2;
  else   y=x-3; end
```

Пояснення. У разі виконання умови обробляється операція y=x+2, у протилежному випадку виконується операція y=x-3;

Використання комбінації ключових слів першого, третього і четвертого дозволяє продовжити перевірку умов.

Приклад:

```
if x<6  y=x+2;
  elseif   x>3   y=x-3; end
```

Пояснення. В наведеному прикладі у разі невиконання першої умови буде перевірятися друга умова. Якщо друга умова виконується обробляється операція y=x-3, у протилежному випадку вказана операція ігнорується.

Використання ключового слова **end** дозволяє обмежити блок операцій, які відносяться до попередньої умови і також забезпечити виконання кількох операцій послідовно для відповідних умов.

Приклад:

if x<6 y=x+2;
elseif x>3 y=x-3; z=x+16; **end**

Пояснення. У разі невиконання першої умови буде перевірятися друга умова. Якщо друга умова виконується обробляються операції y=x-2, і z=x+16, у протилежному випадку вказані операції ігноруються.

Ще одним засобом керування ходом обчислювалного процесу є оператор **switch,** який використовуют сумісно з ключовими **словами case, othervise** та **end.**

Нижче наведено фрагмент програми з використанням цього оператора

```
switch x
case :2
 y=x^2;
case 0
y=x^3;
case :3
y=x^4;
otherwise
y=x;
end
```

Пояснення. За ключовим словом **switch** обов'язково має слідувати параметр, яким в наведеному прикладі є «**x**», ключове слово **case** за своїм сенсом, означає «випадок». За цим ключовим словом слідує значення. В прикладі це значення є «2». Ключове слово **otherwise** означає «інший варіант». Ключове слово **end є** завершальним для оператора **switch.** Працює наведений фрагмент програми наступним чином. Якщо обробник програм перед входом в оператор **switch** має значення **«x»,** наприклад, 3, то в цому операторі він буде порівнювати це значення зі значеннями, які стоять поруч з кожним ключовим словом **case.** У разі неспівпадіння значень, блок операцій до наступного ключового слова буде пропущено. У разі спів падіння значень, блок операцій в цьому **case** буде виконано. В наданому прикладі, буде виконано лише **case** зі заначеням «3». Після цього обробник програм вийде з оператора **switch.** Результатом виконання програми буде значення y=81. Після цього обробник програм перейде до наступної частини програми, якщо вона є.

Для перевірки розуміння роботи цього оператора знайдіть значення y, якщо вхідний параметр має значення x=1.

2.1.1.4. Засоби збереження інформації

В процесі виконання програм є необхідність зберігати значення констант та проміжні значення змінних. Збереження даних відбувається «за кадром». Кожного разу при виконанні операції «=» присвоєння значення тому або іншому ідентифікатору, це значення зберігається в оперативній памяті комп'ютера. Значення буде змінено лише при присвоєні вказаному ідентифікатору іншого значення.

2.1.2. Забезпечення взаємодії з користувачем

2.1.2.1. Засоби отримання інформації від користувача

При розрахунках корисною буває можливість завдання користувачем конкретних значень змінних або параметрів безпосередньо в ході виконання програми. Отримання потрібної інформації від користувача забезпечується за допомогою спеціальних команд **input()** та **menu().**

Отримання числового значення

Команда **input()** здійснює запит числового значення. Після його завдання користувачем вказане значення може бути присвоєне необхідній змінній.

Приклад.

x =**input('x=');**

Пояснення. Виконання обробником програм вказаної команди призведе до виводу на екран комп'ютеру рядку:

x=

після чого комп'ютер буде знаходитися у стані очікування до моменту завдання значення користувачем. Після отримання значення, наприклад 13, воно буде присвоєно змінній **x.** В оперативній пам'яті комп'ютера змінна x буде зберігати значення 13.

Отримання інформації про прийняте рішення

Команда **menu()** дозволяє представити користувачу інформацію для прийняття рішення, яке може використовуватися, наприклад, для керуванням ходом обчислювального процесу.

Приклад.

z=menu(‘Заголовок’,’Кнопка_1’,’Кнопка_2’);

Пояснення. Вказаний запис призведе до виводу на екран комп’ютера віконця з трьома кнопками, на яких будуть відповідні записи, **Заголовок, Кнопка_1** та **Кнопка_2.** Після виводу меню, комп’ютер буде залишатися у стані очікування до надходження дії від користувача.

Користувач має можливість прийняти рішення про яке обробник програм дізнається по номеру натиснутої кнопки.

У разі натиснутої кнопки 2, змінній **z** буде присвоєно значення **2.** Вказане значення може бути використане в програмі для вибору наступних команд, відповідно до бажаного ходу обчислювального процесу.

Приклад фрагменту програми і результат її роботи представлено на рис.2.1.

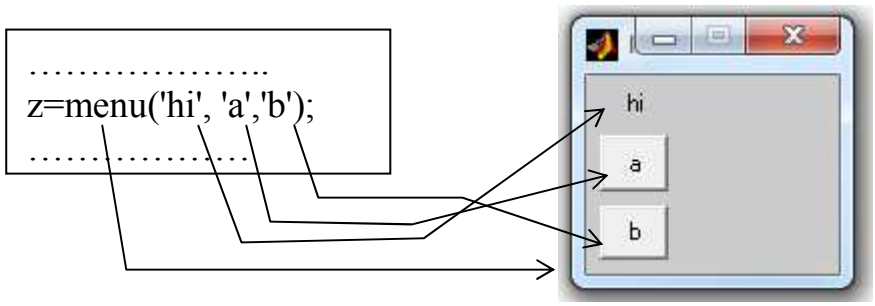


Рис. 2.1. Фрагмент програми (а) та відповідний вигляд вікна (б) при виведені меню в пакеті Матлаб

2.1.2.2. Засоби представлення інформації для користувача

Взаємодія з програмою може потребувати представлення обробником програм певної інформації для користувача.

Інформація може представлятися у вигляді текстових рядків, числових значень або графіків. Графіки у свою чергу можуть бути виведені у потрібне поле у вікні графічного виводу. Крім того, графіки можуть мати надпис та розмітку по осях координат. Для кожного випадку використовуються відповідні команди.

Вивід інформації у вигляді тексту

Для виводу тексту безпосередньо на екран комп’ютера може використовуватись команда :

disp(‘текст’)

Пояснення. В результаті виконання команди на екран комп'ютера буде виведено слово

текст

Для виводу текстового рядку в поле графічного вікна використовується команда

h=text(x,y,'Текст для виводу','FontSize',12);

Пояснення, **x,y –** кординати лівого нижнього кута поля виводу тексту, **'Текст для виводу' –** текст який буде виводитись у поле вікна, **'FontSize',12 –** завдає розмір шрифту.

Вивід числової інформації

Числова інформація виводиться у поле вікна аналогічно виводу текстової інформації, але перед виводом числова інформація має бути перетворена у текстову. Для цього використовується команда **num2str(x),** де **x –** змінна, що має числове значення.

Вивід інформації у вигляді графіків

Для представлення інформації у графічному вигляді можуть використовуватися команда **plot(x,y),** де **x,y** – координати точки.

Приклад.

plot(20,30);

Пояснення. Обробник програм виведе вікно з полем для графіку з розміткою координатних осей. На полі буде зображено точку з координатами **x=20,y=30.**

Виведення надпису для графіка

Виведення надпису для графіка забезпечується командою:

title('Графік залежності....');

Пояснення. У вікні над полем графіка буде виведено напис за аналогією з рис.2.2.

Очистка вікна та осей координат

axis('off'),

Пояснення. На полі графіку будуть видалені вісі.

Очистка вікна по за межами поля графіка

delete(gca);

Пояснення. Поле графіку буде очищене від попередньо побудованих графіків.

Зображення координатної сітки

grid

Пояснення. На поле графіку буде нанесено координатну сітку.

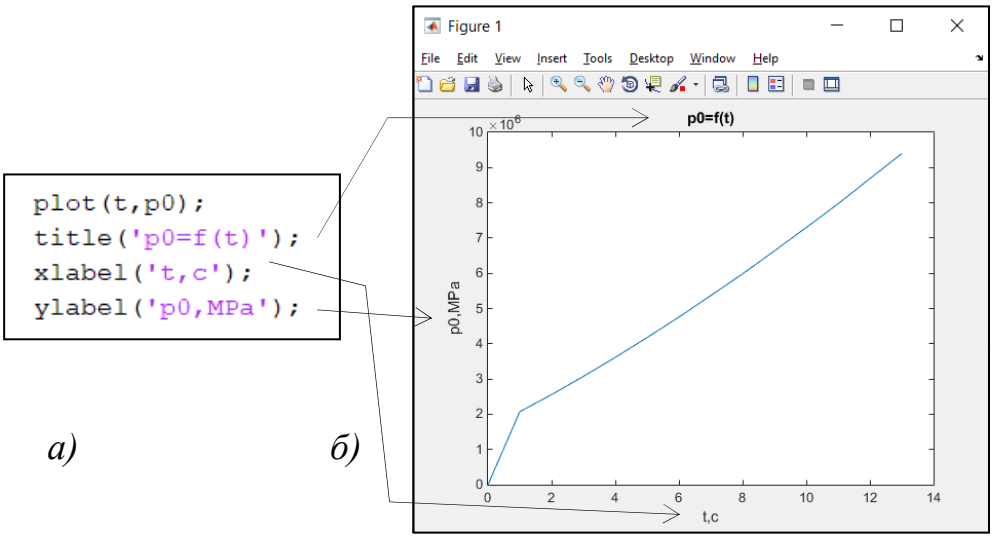


Рис. 2.2. Фрагмент програми (а) та відповідний вигляд вікна (б) при побудові графіків в пакеті Матлаб

Виведення поля графіка у потрібне місце вікна

subplot(2,3,6);

Пояснення. Поле вікна виводу розділюється на 2 рядки, 3 стовбчики, і передбачається вивід графіку у 6 віконце, яке розташоване у нижньому рядку праворуч (рис.2.3)

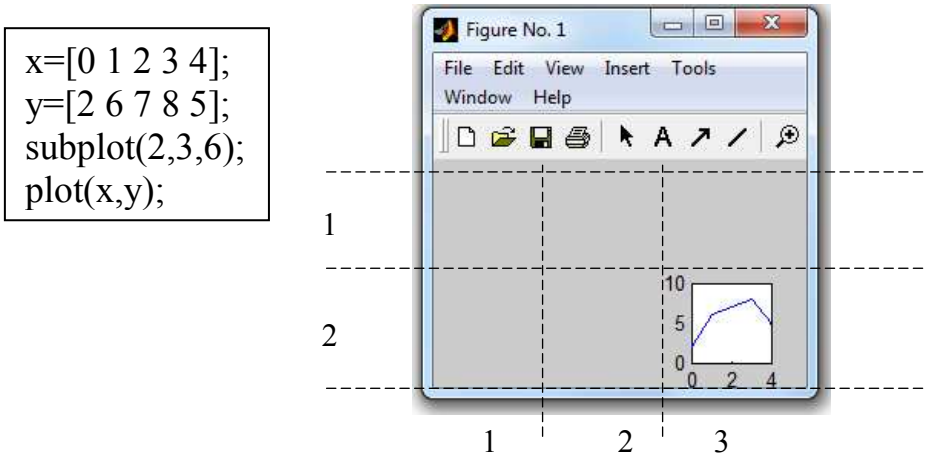


Рис. 2.3. Фрагмент програми і результат її роботи у вигляді вікна з виведеним графіком в шостому полі другого рядка третьої колонки.

Інший приклад
subplot(2,3,[4,5,6]),

Пояснення. Розділення поля вікна виводу на 2 рядки, 3 стовбчики і планування виводу графіку у 4,5 та 6 віконце одночасно, яке розташоване у нижньому рядку.

Збереження попередніх даних на полі графіка
hold on

Пояснення. При розташуванні цієї команди перед командою **plot()** попередня інформація, яку вже було виведено в поле графіку, буде збережена.

Представлення додаткової інформації для користувача

Малювання або видалення координатної сітки на полі графіку

grid on - малювання сітки;
grid off - видалення сітки.

Нанесення підписів по осях X та Y:

xlabel('Час')
ylabel('Переміщення')

Пояснення. Результат виконання команд такий, як наведено в прикладі (рис.2.2).

Виведення графіків з завданням параметрів для позначення ліній
plot (x, y, 'колір, тип лінії, тип точок');
Приклад.
plot (x, y, 'm -- *');

Пояснення. При виведені графіків для ліній буде використано фіолетовий колір, штрихова лінія, з позначенням точок символом «*».

Нижче наведені варіанти завдань для типів ліній, символів точок та кольорів.

Символи для завдання типу ліній графіків

-	неперервна
--	штрихова
:	пунктирна
-.	штрих-пунктирна

Символи для завдання типу точок на графіку

.	-	крапка
+	-	плюс
*	-	зірочка
o	-	коло
x	-	хрестик

Символи для завдання потрібного кольору лінії на графіку

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

Побудова тривимірних графіків

Для побудови тривимірних графіків в Матлаб можуть бути використані наступні три різні функції **plot3 (), mesh () і surf ().**

plot3 (X, Y, Z),
mesh(X,Y,Z).
surf(X,Y,Z).

Пояснення. Кожен з аргументів цих функцій **X,Y,Z** є одновимірним масивом, при цьому **X,Y –** є незалежними, а **Z** є масивом значень, які відповідають заданим **X,Y.** Для побудови графіків треба попередньо задати вказані масиви, а потім викликати функцію **meshrid**(**X,Y**) побудови області в координатах x**,y**. Після чого треба задати відповідний ним масив **Z**

і викликати одну з функцій **plot3(), mesh() і surf().**

Графік, побудований функцією **plot3()** представляється лініями в координатах z,y.

Графіки, побудовані функціями **mesh() і surf ()** представляються поверхнями у вигляді сітки зі значеннями z у її вузлах.

Підпис кожної вісі можна зробити відповідними функціями

xlabel('x'), ylabel('y'), zlabel('z').

Приклад. Побудувати графік функції z(x,y)=y^2-x^3 для інтервалів значень x та y $x \in [-5,5]$, $y \in [-5,5]$.

Рішення.

Формуємо сітку в площині x, y
[X,Y]=meshgrid(-5:0.4:5,-5:0.4:5);
Розраховуємо значення z для масиву Z
Z=Y.^2-X.^3;
Будуємо тримвимірний графік функції
mesh(X,Y,Z);
Підписуємо вісі
xlabel('x'); ylabel('y'); zlabel('z');
Отримуємо результат (рис.2.4)
У разі необхідності завдання потрібних масштабів по осях координат, використовуємо функцію
axis([xmin xmax ymin ymax zmin zmax])

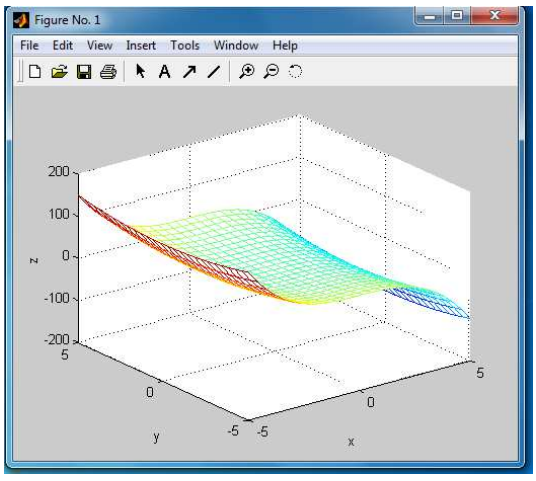


Рис. 2.4. Вигляд вікна в пакеті Матлаб при виведені тривимірних графіків

Представлення ілюстрацій у вікні програми

Для кращого розуміння користувачем постановки задачі або результатів роботи програми зручним засобом представлення інформації є команди виведення на екран різноманітних ілюстрацій. Наприклад, це може бути зовнішній вигляд об'єкту або його розрахункова схема з позначеннями основних параметрів.

Для представлення такої інформації потрібно попередньо її підготувати та зберегти на диску комп'ютера.

В тексті програми графічний файл треба прочитати та прив'язати отриману інформацію до ідентифікатору змінної. Після цього треба використати команду виведення на екран графічної інформації з посиланням на відповідний ідентифікатор. Наступний фрагмент програми дозволяє вивести на екран попередньо підготовлений графічний файл (рис.2.5).

```
%Вивід зображення варіант 1
img = imread('bubbles.jpg');
imshow(img)

%Вивід зображення варіант 2
 f = imread(`F:/Images/boat.jpg');
imshow(f)
```

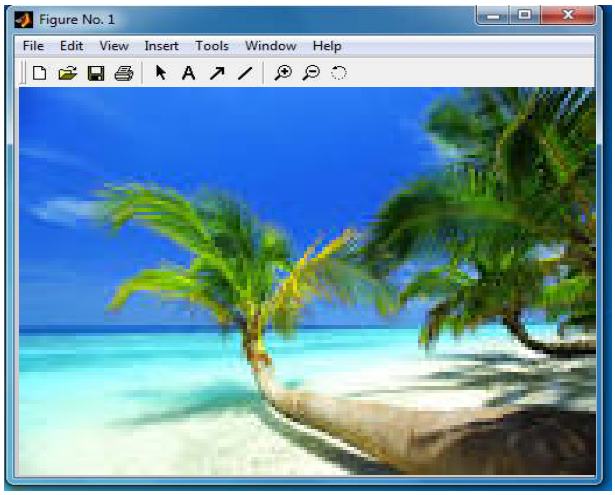


Рис. 2.5 Вигляд вікна програми при виведені графічної інформації

Пояснення. У фрагменті програми за варіантом 1 зчитування інформації відбувається з каталогу, який встановлено за замочуванням у програмному середовищі. У фрагменті програми за варіантом 2 зчитування інформації відбувається з каталогу, який в явному вигляді надається в програмі.

Розглянутий комплект засобів дозволяє задавати команди на виконання очислювальних операцій, команди багаторазового виконання операцій,

команди керування ходом обчислювального процесу, команди використання памяті, а також команди для забезпечення взаємодії користувача з обробником програм в ході її виконання. Наступною задачею є навчитися їх використовувати для розробки програм.

2.1.3. Побудова програм моделювання процесів

2.1.3.1.Формування алгоритму

Інженерні розрахунки або моделювання виконуються відповідно до наперед завданих методик. Методики регламентують початкові дані, потрібну послідовність дій, аналіз проміжних результатів та прийняття рішень що до вибору варіантів, у випадку можливих альтернативних дій, а також отримання кінцевих даних. Вказані дії та їх послідовність можуть бути відображені у вигляді алгоритму за допомогою графічних засобів, наприклад, як це покаано нижче (рис. 2.6).

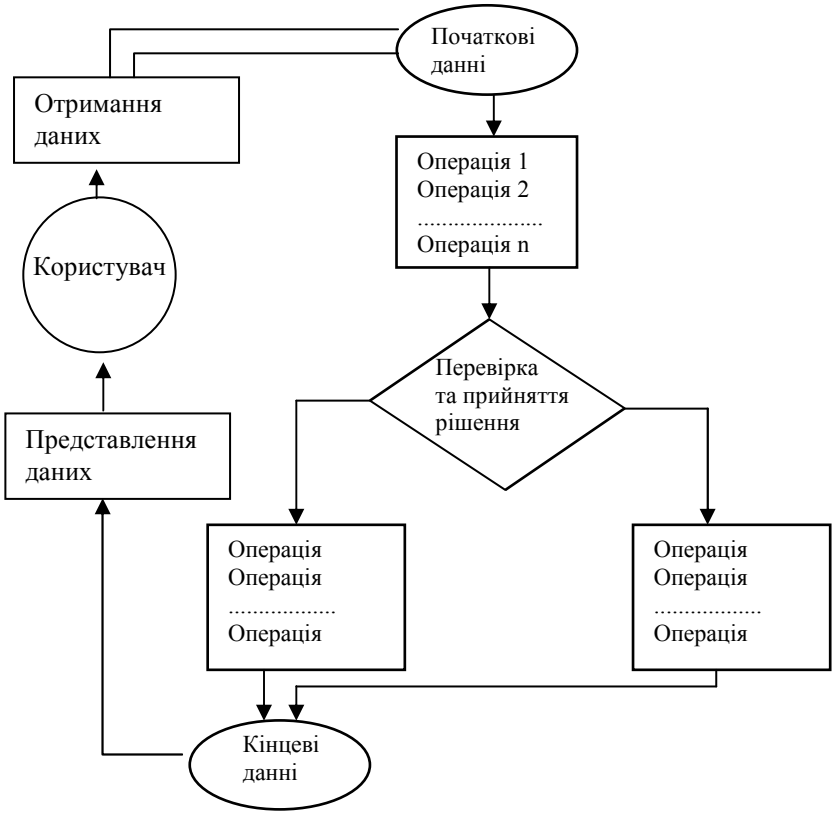


Рис.2.6. Приклад представлення алгоритму виконання розрахунків

Застосування графічних засобів представлення алгоритмів полегшує розуміння задачі, яка вирішується і дозволяє в подальшому використовувати різні мови програмування. Алгоритмічне представлення також дозволяє

розділяти складну задачу на більш прості і вирішувати їх командою розробників одночасно.

2.1.3.2. Перетворення алгоритму в програму

Перетворення алгоритму в програму можна розглядати як наповнення схеми алгоритму змістовними складовими.

Програму формують за допомогою засобів, що були наведені вище. При цьому, використовуються властивості обробника програм виконувати дії у суворій черговості згідно послідовності їх зчитування. Іншими словами, організація потрібної черговості виконання дій передається обробнику програм шляхом послідовного запису потрібних дій безпосередньо у програмі. Також, для забезпечення потрібної черговості використовують засоби керування ходом обчислювального процесу такі як **if else elseif** та **end**. Початкові данні отримуються від користувача, а кінцеві данні представляються користувачу також за допомогою відповідних засобів. У якості прикладу розглянемо формування програми на основі заданого алгоритму. Покажемо це для одного з його фрагментів. Заданий алгоритм представляє вирішення задачі дослідження поведінки графіку функції в залежності від значень коефіцієнтів (рис.2.7).

Програма має забезпечити можливість завдання коефіцієнтів користувачем, автоматизувати обчислення значень завданої функції, та представляти результати у графічній формі. Фрагментом алгоритму, який маємо перетворити в програму є блок реалізації перевірки та прийняття рішення (рис.2.7). Для перетворення цього фрагменту алгоритму в програму використовуємо засоби організації ходу обчислювального процесу такі як оператори «if», «else» та «end».

В результаті запису цих операторів з врахуванням потрібних операцій, які вказані в алгоритмі для кожної з ситуацій, отримуємо фрагмент програми:

if x<6
Операція 1;
Операція 2;
……..
Операція n;
end
else
Операція 1e;
Операція 2e;
……..

Операція me;

end

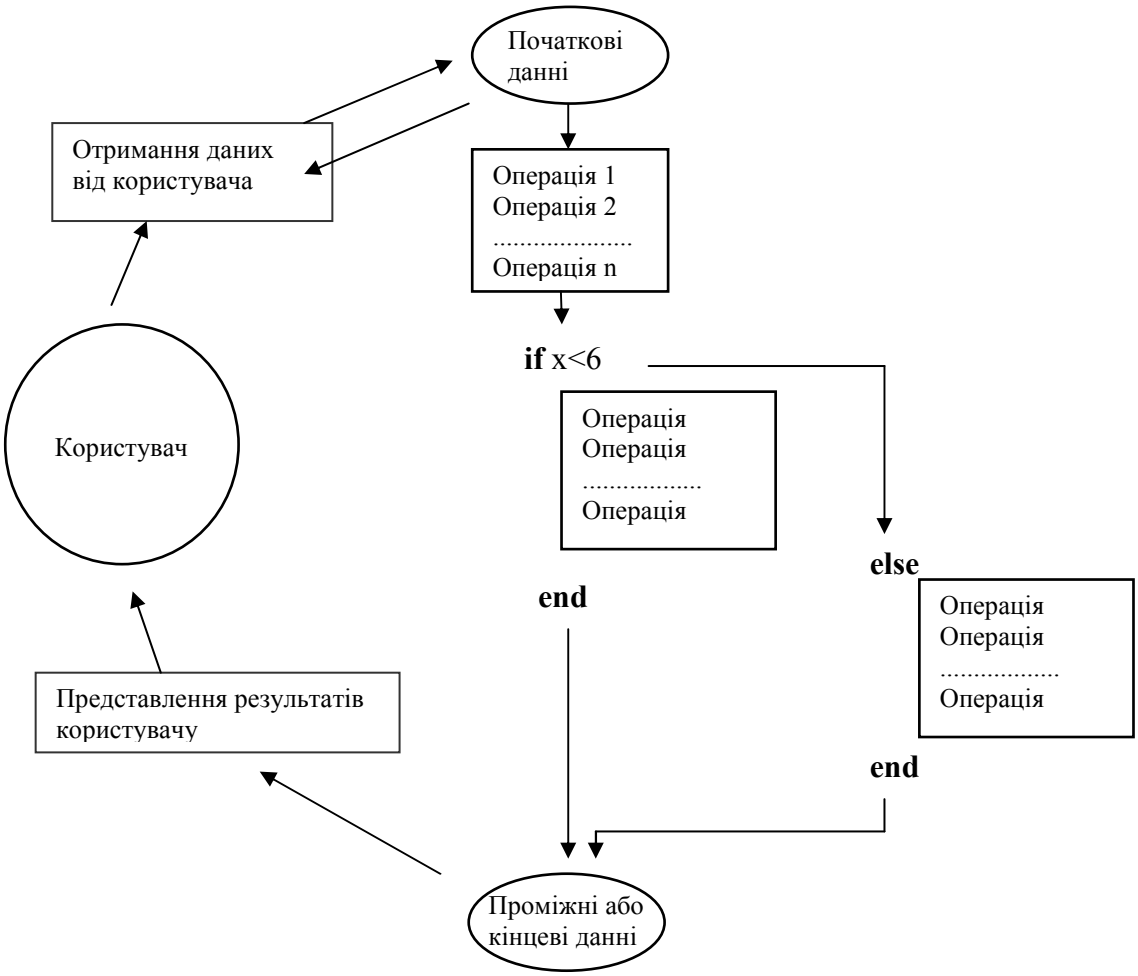


Рис.2.7. Алгоритм проведення розрахунків, що забезпечує інтерактивну взаємодію з користувачем

Аналогічним чином, кожен блок усього алгоритму має бути замінено відповідними засобами розробки програм з врахуванням ходу обчислювального процесу. Результатом є код програми, який структуровано відповідно до заданого алгоритму. Як правило, кожен блок має доповнюватись коментарями, які суттєво полегшують роботу з програмою при її налаштуванні та удосконаленні.

2.1.4. Засоби компонування програм

В залежності від задачі моделювання програми можуть мати різну складність. Для більш складних програм зручно використовувати засоби компонування програм. Вони дозволяють розподіляти розрахунки по локальних блоках, що має ряд позитивних наслідків: підвищується легкість

роботи з програмою; скорочується кількість рядків коду; з'являється можливість багаторазового використання локальних блоків з різними вхідними даними, та інше.

Для побудови коду таким чином використовують головну програму і програми, які використовуються головно програмою. В MatLab зазвичай один М-файл є головним, його іноді називають файлом-сценарієм, який керує ходом обчислень. Головний файл в своєму тілі може використовувати інші М-файли, які є зовнішні по відношенню до нього. Називають такі файли М-функціями. Звернення з головного файлу до М-файлів функцій виконується за допомогою їх імен. М-файл, що представляє собою функцію, повинен мати ряд формальних ознак. Такими ознаками є:

- ключове слово – **function;**
- символ, що позначає змінну яка повертається;
- ім'я функції зі списком значень;
- ключове слово **end,** яке закриває тіло функції.

Звернення до М-файлу функції з головного файлу виконується шляхом запису імені функції зі списком змінних, значення яких потрібні для її виконання.

Приклад запису М-файлу функції:

```
function y=F1(x,d)
y=x+d;
end
```

Пояснення. "x" та "d" є параметрами функції, "y" – змінна, яка буде отримувати та повертати результат виконання функції. При збереженні функції в файлі, його ім'я має бути таким самим, як і імя функції, але з розширенням «.m». Наведену вище функцію треба зберігати з ім'ям **F1.m**

Приклад звернення до файлу-функції F1 з головного файлу

Головний М-Файл

```
x=2:
d=16:
k=21;
y = F1(x,d);
z=y+k;
rez=num2str(z),
disp(rez)
```

Важливо. Треба враховувати, що в файлах функціях всі імена змінних, що знаходяться всередині М-файлу, а також змінні, визначені в заголовку є локальними і їх дія не розповсюджується на головний файл.

Всі змінні параметри, які використовуються у головному файлі-сценарії створюють певну робочу область, в якій їх значення зберігаються протягом всього сеансу роботи з програмою. Якщо в програмі одночасно використовуються кілька файлів-сценаріїв, то її робоча область є єдиною для всіх файлів-сценаріїв, що викликаються під час роботи програми.

2.1.5. Стандартні функції для вирішення спеціальних задач

Стандартними функціями є стандартні математичні операції, наприклад, sin(x), cos(x), sqrt(x) та ін., які можуть бути використані в середині обчислювальних блоків. Існують також стандартні функції вирішення систем диференційних рівнянь, якими, як правило, описується дія фізично різнорідних компонентів та систем в цілому. Для більш глибокого розуміння таких стандартних функцій розглянемо один з основних методів пошуку рішень систем диференційних рівнянь, які лежать в їх основі.

2.1.6. Чисельне рішення диференційних рівнянь [6,7]

Постановка задачі Коші.

Відомо диференційне рівняння вигляду

$y^{(n)} = f(t, y, y^1, ,....y^{(n-1)});$

з початковими умовами

$y(t_0) = y_0, y^1(t_0) = y_1,.....y^{(n-1)}(t_0) = f(t, y, y^1, ,....y^{(n-1)} = y_{n-1},$

Треба чисельним методом знайти функцію, яка буде представляти рішення цього рівняння.

Для чисельного вирішення, конкретне задане рівняння вищого порядку необхідно представити у вигляді системи рівнянь першого порядку

$y^1(t) = F(t, y), y(0) = y_0,$

де y – функція (якщо система складається з n – рівнянь, то y представлено вектором, а y0 – її початкові значення (у випадку системи рівнянь y0 також є вектором)

Для вирішення диференційного рівняння або системи рівнянь чисельним методом в пакеті MatLab можна використовувати стандартну функцію з ім'ям **ode45.**

Формат використання цієї функції наступний.

[T, Y] = ode45(odefun, [t0,tend], y0, options)

де `odefun` – ім'я функції, яка буде забезпечувати обчислення значень правих частин F(t,y) заданого рівняння або системи рівнянь у формі (1). Результатом виконання цієї функції буде значення або вектор значень у для кожного заданого значення незалежної змінної t. Функцію `odefun` треба задати і розраховувати перед викликом стандартної функції ode45.

Параметри `[t0,tend]` задають початкове і кінцеве значення діапазону змін незалежної змінної `t.`

`y0` – початкове значення або вектор функції, яка буде визначатись чисельно.

`Options` - необов'язковий аргумент, який може використовуватись для задання потрібної точності розрахунку. Його значення задане по умовчанню.

Результатом виконання функції `ode45` є масив T – значень незалежної змінної t для яких шукались значення функції і масив відповідних їм значень функції y.

Приклад використання стандартної функції ode45 пакету MatLab

Задано математичний опис дії тенічної системи, який представлено диференційним рівнянням

$$\frac{d^2h}{dt^2} + \frac{c}{m} \cdot h - g = 0.$$

1. Представляємо математичний опис у формі Коши

1.1. Записуємо змінні під знаком диференціала зліва по відношенню до знака рівності

$$\frac{d^2h}{dt^2} = -\frac{c}{m} \cdot h + g.$$

1.2. Записуємо рівняння у формі системи диференційних рівнянь першого ступеню

$$\begin{cases} \frac{dV}{dt} = -\frac{c}{m} \cdot h + g; \\ \frac{dh}{dt} = V. \end{cases}$$

Задаємо початкові умови

$$h(0) = 0; V(0) = 0.$$

2. Готуємо систему рівнянь для використання стандартної функції ***ode45***

2.1. Робимо заміну

$$y_1 = \mathrm{V}; \mathrm{y}_2 = h.$$

2.2. Пишемо програму моделювання

```
//Код програми
//Функція для розрахунку правих частин системи рівнянь
function F=ode_fnc(x,y)
F=[-c/m*y(2)+g; y(1)];

//Виклик стандартної функції чисельного методу
[T, Y] = ode45('ode_fnc', [0,10], [0,0])
```

2.3. Отримуємо результат моделювання

Формат виводу результату такий:

```
[T, Y] =
t0   y1(t0)   y2(t0)
```

t1 y1 (t1) y2(t1)

…….

tk y1(tk) y2(tk)

2.4. Додаємо команди виводу графіків

//Побудова графіків за отриманими результатами

plot(T,Y(:,1)); hold on;

plot(T,Y(:,2));

Приклад закінчено.

В залежності від версії пакету MatLab звернення до стандартної функції інтегрування можуть відрізнятися від наведеного в прикладі.

Ще один варіант зверненя наведено нижче

odefun=@(t,P0)E*q/V %Опис функції «odefun», яка задає

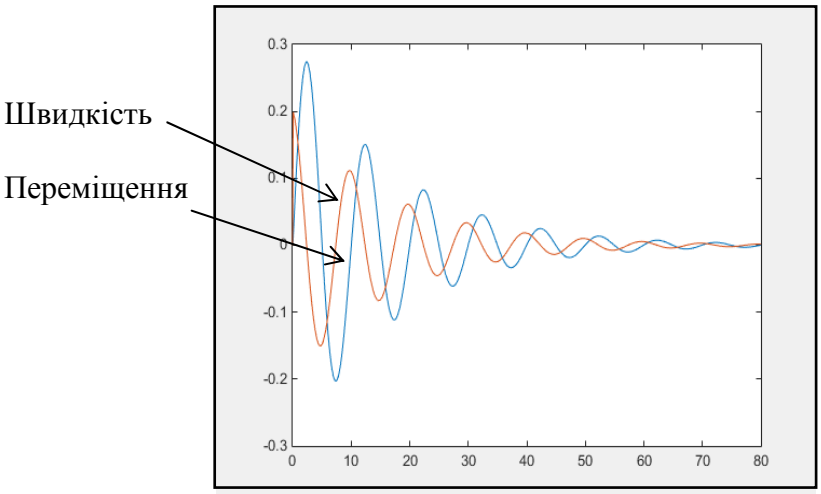
% праві частини рівняння у формі Коші

[t,P0]=ode45(odefun,t,P0); %Звернення до стандартної функції

%«ode45»

plot(t,P0) %Побудова графіку

Результатом вирішення таких систем рівнянь є процеси, які представлені змінами в часі певних фізичних параметрів, наприклад, швидкості та переміщення тягарця маятника після його примусового відхилення від вертикальної вісі і вільному коливанні (рис. 2.8)



\

Рис.2.8. Приклад вигляду графіків при моделюванні динамічних процесів в маятнику

Використання стандартної функції чисельного вирішення систем диференційних рівнянь дозволяє моделювати процеси в компонентах і системах і суттєво зменшує витрати часу на побудову математичних моделей. Однак при цьому залишаються ще питання про достовірність побудованих математичних моделей, відповіді на які знаходять експериментальним шляхом.

3. ІНСТРУМЕНТ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ PYTHON

3.1. Засоби пакету Python для обчислень та моделювання

Враховуючи, що студенти можуть вибирати інструмент для обчислень і моделювання за своїм бажанням, в цьому розділі наводимо основні засоби мови Python. Матеріал цієї частини представлено аналогічно до представлення засобів MatLab. Це дозволить легко орієнтуватись і швидко знаходити потрібний засіб для побудови програми моделювання.

Інтерфейс програмного середовища Python має традиційний вигляд та має лінійки меню для взаємодії (рис.3.1). Відкривши нове вікно для створення програми через File->New, можна приступати до написання коду. Запуск програми здійснюється натисканням на пункт Run і підтвердженням бажання зберегти код програми в поточному каталозі або в іншому каталозі, який уде треба вибрати.

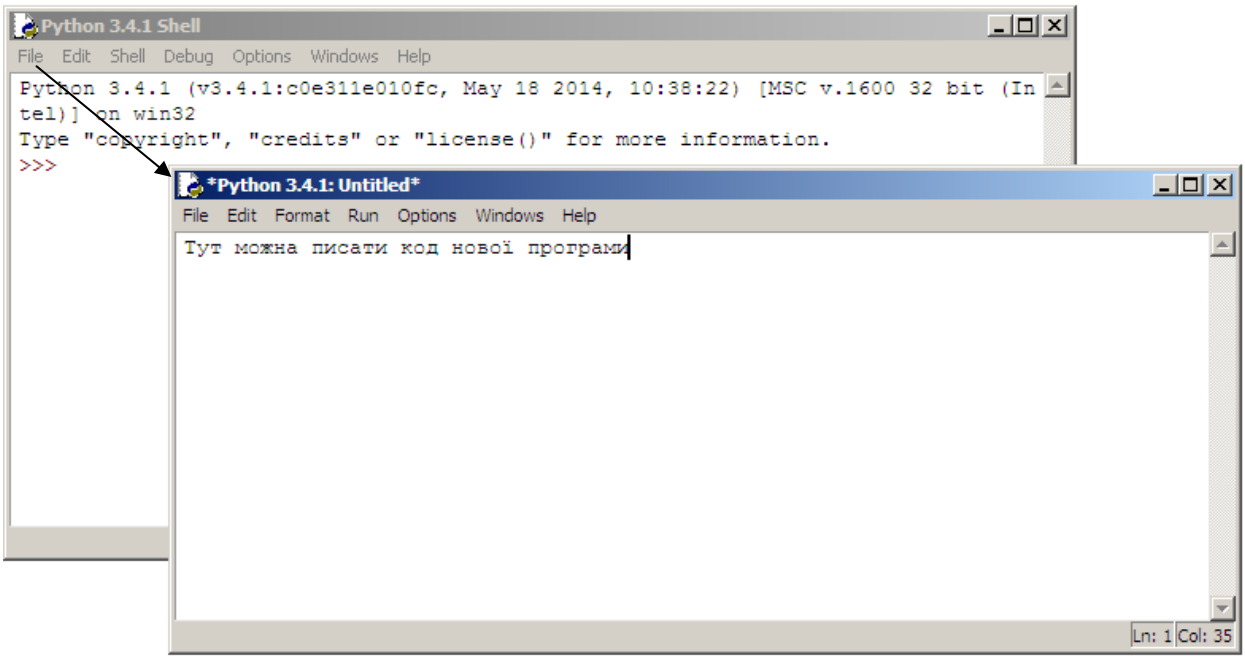


Рис. 3.1. Вигляд інтерфейсу оболонки середовища Python та вигляд вікна для написання коду програми

3.1.1.Засоби формування обчислювального процесу

Спочатку навчимось писати програми у новому вікні і познайомимось з типами змінних, які використовуються в ході обчислень.

Найпростіша програма мовою Python виглядає так

#Найпростіша програма мовою Python
print(“Hello”)

В результаті її виконання на екрані комп’ютера ми побачимо

>>> Hello

#Більш складна програма мовою Python
x=”Hello”
print(x)

Результат її виконання буде такий самий, як і в попередній програмі
>>> Hello

Однак роботи було виконано більше. В перший команді використовується змінна з іменем «x», яке за допомого символу «=», що означає операцію зв’язування, зв’язується з об’єктом “Hello”

Виконання другого рядку програми приводить до виводу на екран інформації, яка пов’язана зі змінною «x»

#Ще більш складна програма мовою Python
x=”Hello”
print(x)
x=1+2+3*2

В результаті виконання третьої команди цієї програми змінна «x» буде зв’язана з результатом обчислення, який буде мати значення 9. Наведений приклад демонструє, що особливістю мови є динамічне визначення типу даних, які використовуються. В наведеній програмі змінна «x» в першій команді була зв’язана з рядком ”Hello” , а в третій команді, ця сама змінна, була зв’язана з числом цілого типу 9. При цьому об’єкт ”Hello” було втрачено.

Таким чином, при роботі зі змінними треба розуміти наступне

- будь-яка змінна є посиланням;

- тип змінної визначається тим, на що вона посилається;
- тип змінної може довільно змінюватися по ходу виконання коду, коли змінна починає посилатися на інший об’єкт.
- попереднє значення змінно втрачається.

3.1.1.1 Засоби виконання обчислювальних операцій

Обчислення виконуються відповідно до потрібних формул, які можуть містити змінні, числа та математичні операції.

Нижче наведені арифметичні операції. Вони наведені в порядку, який відповідає пріоритетам їх виконання при використанні в математичних формулах

- x в ступеню y x**y
- x//y цілочислене розділення
- x%y визначає залишок від розділення
- ** – піднесення до степеня;
- –*x* – унарний мінус;
- /, // – звичайне ділення, ділення з округленням вниз;
- **%** – остача від ділення;
- – множення;
- – – віднімання;
- + – додавання.

Нижче наведено фрагмент програми, який ілюструє приклад команди виконання обчислень

x=7

z=3

y= x*2+z/3-z2**

Результатом виконання програми буде значення y, яке буде дорівнювати 41

Крім звичних обчислень в мові можуть використовуватись і такі операції, як множинне присвоювання.

x,y,z=1,2,5

Виконання цієї команди призведе до зв'язування імені кожної змінної з відповідним значенням. В результаті x буде мати значення 1, y буде мати значення 2, z буде мати значення 5.

Якщо a,b отримають значення, наприклад, в результаті виконання такої команди

a,b=2,3 #(a =2, b=3),

то в програмі допускається також кортежний обмін

a,b=b,a

в результаті кортежного обміну значення a буде дорівнювати 3, а значення b буде дорівнювати 2.

3.1.1.2. Засоби забезпечення багаторазового виконання операцій

Для забезпечення багаторазового виконання операцій використовують цикли. Одним з варіантів організації циклу є використання ключового слова **for.** За ним має слідувати змінна, яка є параметром циклу, далі має бути ключове слово **in**, і далі має бути задано спосіб змін параметру циклу і розділювач «**:**».

Приклад

```
for i in range(0,6,2):
   print ('Hello')
```

результат виконання програми буде наступним

Hello
Hello
Hello
Hello

Пояснення. Ключове слово **range**(0,6,2): - послідовно задає значення параметру циклу **i** від 0 до 6 кроком 2. Якщо крок зміни параметру циклу не задано, за замовчуванням він дорівнює 1.

Приклад

```
for n in range(5,8):
print(n)
```

результат:
5
6
7

Якщо початкове значення параметру циклу не задано, то, за замовченням, воно дорівнює 0, якщо і крок не задано, за замовчуванням він дорівнює 1.

Приклад

Програма
for n in range(3):
print(n)

результат:
0
1
2

Якщо способом завдання змін параметру циклу є набір значень в квадратних дужках, то саме ці значення і використовуються в порядку їх наведення.

Приклад

```
for i in [0, 1, 2, 3, 4]:
    print(i)
```

результат
0
1
2
3
4

Ще одним варіантом завдання циклів є використання ключового слова **while.** Особливістю використання цього циклу є обов'язкове завдання початкового значення параметру циклу та забезпечення його зміни в потрібному напрямку безпосередньо в самому циклі.

Приклад програми

```
i = 5
while i < 8:
  print ('Привіт')
  i = i+1
```

Результат:
Привіт
Привіт
Привіт

Допустимим є також використання циклів в середині циклів.

3.1.1.3. Засоби керування ходом обчислювального процесу

Для керування ходом обчислювального процесу використовуються умовні оператори

Використовують три варіанти форм таких операторів

Форма 1

if <умова>:
    <оператори>

Форма 2

if <умова>:
__<оператори 1>
else:
__<оператори 2>

Приклад

```
x = 7.5
if (x > -10) and (x < 5):
        y = 5*x - 10
else: y = 12-3*x
print(y)
```

Результат

```
>>>
-10.5
>>>
```

Пояснення. В рядку 1 програми наведена команда прив'язування змінної x до значення типу float. В рядку 2 перевіряються дві умови. Якщо обидві умови виконуються, то визначається значення y за формулою в рядку 3. Якщо не виконується хоч одна з умов обробник програм переходить на рядок 4 – інакше, і розраховує значення y за формулою в цьому рядку. Далі, обробник програм, виконує команду рядка 5 – друкує на екрані значення y/

Форма 3

if <умова1>: <оператори 1>
elif <умова2>: <оператори 2>
...
elif <умова N>: <оператори N>
else: <оператори>

Ця форма надає можливості виконувати кілька перевірок послідовно.

При використанні умовних операторів потрібно мати різні варіанти перевірок поточних значень змінних. Можна застосовувати наступні оператори відношення

- <
- <=
- >
- >=
- x== 10
- ~=
- !=
- |=

В умовних операторах можна також використовувати оператори завдання логічних відношень

Оператори логіки

and, or, not

Приклад

```
x = True
y = False
print('x and y —', x and y)
print('x or y —', x or y)
print('not x —', not x)
```

Результат

x and y — False
x or y — True
not x — False

В мові Python (версия 3.10. та далі) є також оператор **match**, який є аналогом оператора **switch**, який використовується в MatLab

В наведеному фрагменті програми оператор **match** запаковано в функцію **calc_rez**. При звернені до функції передається її параметр, з яким порівнюються задані випадки **case.**

```
def calc_rez(variant):
    match variant:
        case "first":
            print("Доброго ранку")
```

```
        case "second":
            print("Доброго дня")
        case "third":
            print("Доброго вечора")

        case _:
            print("Невідомий результат)
```

Приклад виклику функції

calc_rez(" ")

Результат

>>> Невідомий результат

Приклад виклику функції

calc_rez("second")

Результат

>>> Доброго дня

3.1.2 Забезпечення взаємодії з користувачем

3.1.2.1.Засоби отримання інформації від користувача

Для отримання інформації від користувача в процесі виконання програми можна використовувати команду **input().** При цьому для подальшого використання отриманої інформації її треба зберегти у відповідній змінній програмі

Приклад

x = float(input())

Пояснення. Після виконання цього рядку виконання екран комп'ютера буде виведено запит >>> і виконання програми буде зупинено в очікувані на відповідь користувача. Після отримання потрібної інформації змінна x буде прив'язана до значення типу float, яке буде зчитано з екрану комп'ютера після його введення.

Для інформування користувача, значення якої змінної потребує програма, рядок програми потрібно змінити наступним чином

x = float(input(‘x=”))

у цьому випадку на екран комп’ютера буде виведено запит

>>> x=

і програма перейде в стан очікування. Після введення користувачем потрібного значення, воно буде зчитано, прив’язано до змінної x в програмі і виконання програми буде продовжено.

Для отримання інформації від користувача в Python також можна використовувати елементи графічного інтерфейсу [9]. Враховуючи, що ці елементи знаходяться у відповідних бібліотечних файлах, їх треба підключити на початку програми.

Приклад програми для роботи з меню

```
import tkinter as tk            # Підключення бібліотеки  графіки tkinter

class Application(tk.Frame):   # Об’явлення класу Application з прив’язкою до
                                #бібліотеки tk
  def __init__(self, master=None):      # Функція ініціації класу
    tk.Frame.__init__(self, master)     #Ініціаліація фрейму(каркасу)
    self.pack()                         #Виведення фрейму на екран
    self.createWidgets()                #Створення віджету(вікна)

  def createWidgets(self):              #  Функція для створення вікна Wnd
    self.hi_there = tk.Button(self)  #  Команда для створення кнопки 1 у вікні
    self.hi_there["text"] = "Hello World\n(click me)"   # Команда для
                                                        #створення напису
    self.hi_there["command"] = self.say_hi      # Команда для реакції на
                                                #натискання
    self.hi_there.pack(side="top")              # Місце розташування у вікні

    self.QUIT_2 = tk.Button(self, text="Continue", fg="red",
                            command=self.say_hi2)      # Команда для створення
                                                       #  кнопки  2
    self.QUIT_2.pack(side="bottom")            # Місце розташування у вікні

    self.QUIT = tk.Button(self, text="QUIT", fg="red",
                            command=root.destroy)    # Команда для створення
```

```
                                                    # кнопки
    self.QUIT.pack(side="bottom")          # Місце розташування у вікні

  def say_hi(self):                        #  Функція для друку на екрані
    print("hi there, everyone!")           #   Команда для друку     тексту

root = tk.Tk()                             # Зв'язування  вікна з класом Tk

app = Application(master=root)             # Активація  коду
app.mainloop()                             # Активація головного циклу
```

В результаті виконання програми на екран буде виведено графічне вікно (рис.3.2) з можливістю введення інформації натисканням на кнопки і подальшого використання отриманої інформації в активній програмі.

Рис.3.2. Вигляд меню з кнопками, яке побудовано програмним шляхом

Нижче наведено приклад програми створення рядку запиту для отримання інформації від користувача та зчитування цієї інформації після її введення.

```
from tkinter import *    # Імпорт бібліотеки


def printer(event):  # Визначення функції для реакції на натискання клавіші
                     # вводу  Enter
    t=ent.get()      #  Присвоєння змінній t текстового значення з вікна
    print(t)         #   Друк значення t на екрані

root=Tk()            # Зв'язування  вікна з класом Tk

ent=Entry(root,width=20,bd=6)  # Виклик класу текстового поля вводу
ent.pack()                     #  Виведення вікна на екран
ent.bind('<Return>', printer)  # Привязування вікна вводу до клавіші Enter
                               # та  функції  реакції на її натискання
root.mainloop()                # Запуск головного циклу програми
```

Результат виконання програми (рис.3.3)

Рис.3.3. Вигляд графічного вікна для введення інформації

Вигляд вікна на екрані після введення числа як текстової змінної (рис.3.4) і виведене на екран значення після натискання клавіші Enter

Рис.3.4. Вигляд графічного вікна з введеною числовою інформацією, яка виводиться на екран, як текстова інформація

```
>>>
>>> 9
```

Вигляд вікна на екрані після введення числа як тексту (рис.3.5) і виведене на екран начення після натискання клавіші Enter

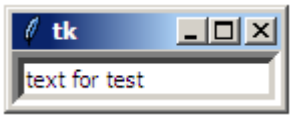


Рис.3.5. Вигляд графічного вікна з введеною текстовою інформацією, яка виводиться на екран, як текст

```
>>>
text for test
```

У разі, якщо потрібно мати більшу кількість вікон введення, то код програми може бути доповнено наступним чином

```
from tkinter import *

def printer(event):
    t=ent.get()
    print(t)

root=Tk()
ent=Entry(root,width=20,bd=6)
ent.pack()
ent2=Entry(root,width=20,bd=6)
```

```
ent2.pack()
ent.bind('<Return>', printer)
ent2.bind('<Return>', printer)
root.mainloop()
```

Результат виконання програми наступний (рис.3.6)

Рис.3.6. Вигляд графічного вікна з двома полями для введення текстової інформації

3.1.2.2. Засоби представлення інформації для користувача

Одним з засобів представлення інформації є команди виведення на екран текстових рядків або значень змінних з коментарями для пояснення їх приналежності до розрахунку в програмі. Приклад такого засобу було розглянуто вище, але нагадаємо його

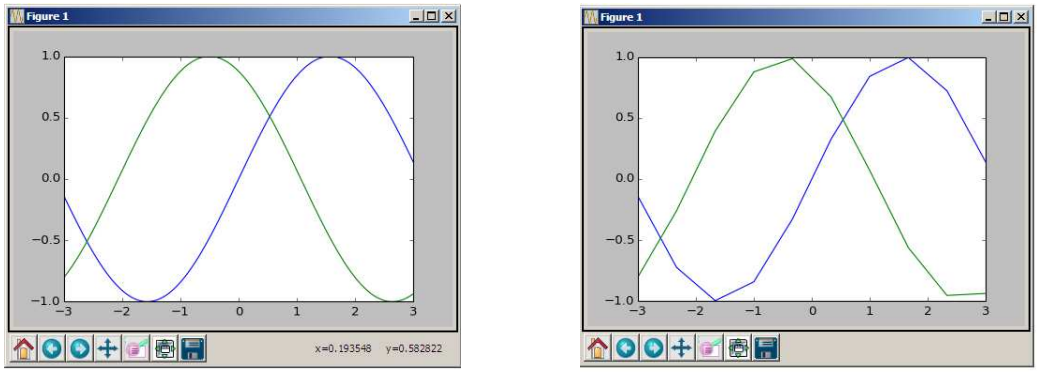
print(“Hello”)

Ще одним засобом є виведення графіків. Нижче наведено приклад коду програми, яка поетапно формує дані для графіків, будує координатну площину та відображає графіки.

```
import numpy as np      # іимпортимпортуємо потрібні модулі
import matplotlib.pyplot as plt
import scipy as sc
y1 = lambda x: np.sin(x)    # задаємо функції для моделювання.
                            #пояснення використання  lambda функції
                             # надано нижче
y2 = lambda x: np.cos(x+0.5)
fig = plt.subplots()   # створюємо координатну площину

x = np.linspace(-3, 3,100)    # Створюємо область даних для  x  (від -3 до
                              #  3 кількістю 100 точок)
plt.plot(x, y1(x))      # Малювання графіків
plt.plot(x, y2(x))
plt.show()                      # Відображення графіків
```

Результат виконання програми (рис.3.7)



а) б)

Рис.3.7. Вигляд графіків з різною кількістю розрахованих даних (а – для 100 точок; - для 10 точок)

3.1.3. Засоби компонування програм

Будь яку програму можна компонувати по різному. Наприклад, писати код по рядках, які будуть виконуватись послідовно з врахуванням потрібного логіки обчислювального процесу. Однак, якщо програма містить багато рядків, або в ньому є блоки, які потрібно виконувати кілька разів при різних початкових даних, то для скорочення кількості коду, зменшення часу на налаштування програми зручно використовувати такий засіб, як функція. Використання функцій в програмах вже частково було представлено при розгляді засобів отримання інформації від користувача. Функція це фрагмент коду, який певно мірою, представляє самостійну програму, дія якої зосереджена на виконанні конкретних локальних завдань в межах задачі основної програми. При використанні функцій головна програми має вирішувати глобальні задачі, а окремі функції, які вона буде викликати за потребою, будуть вирішувати різні локальні задачі. Компоновка такої програми має бути такою (рис.3.8)

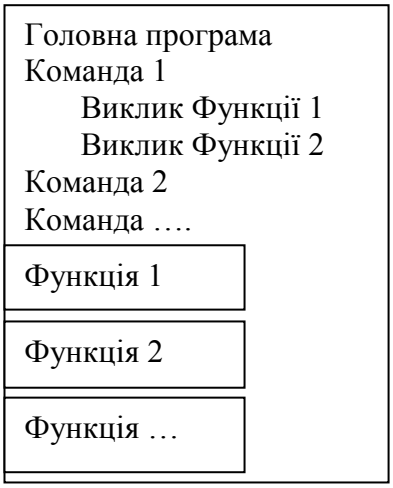


Рис.3.8. Приклад компоновки програми з використанням головної програми та функцій

Приклад написання функції

```
def vitannia(name):
  print('Hello, %s' % name)
```

Звертаємо увагу, що рядки коду, які відносяться до тіла функції мають бути зміщені відносно ключового слова **def.** Це є обов'язковим при написанні функцій, тому що слугує інформацією для обробника програм про блок команд відповідної функції.

Приклад виклику функції

```
vitannia('Петре')
```

Реультат
Hello, Петре.

Функції без імені

Є можливість також використовувати функції не задаючи їх імені. Такі функції позначають за допомогою ключового слова **lambda.**

Загальна форма представлення **lambda** функції є наступною

lambda аргумент1, ..., аргумент N**:** вираз, що треба робити з аргументами.

Приклад

```
rez=lambda x,y: x+y
```

Виклик

```
print(rez(5,4))
```

Результат

9

3.1.4. Стандартні функції для вирішення спеціальних задач

Стандартними функціями є стандартні математичні операції, наприклад, sin(x), cos(x), sqrt(x) та ін.. Вказані функції можуть бути використані в середині

обчислювальних блоків, однак потребують попереднього підключення бібліотеки. Для виклику конкретної функції потрібно використовувати відповідне звернення за наданими нижче іменами [8].

int(x) - перетворення змісту x в ціле число
float(x) - перетворення змісту x в дійсне (дробове) число
str(x) - перетворення змісту x в рядок
round(x) - повертає наближене ціле число до x
abs(x) - повертає значення модуля числа x
pow(a, b) піднесення числа a в ступінь b
min(x) повертає мінімальне значення елементу в ітерованому масиві x
max(x) повертає максимальне значення елементу в ітерованому масиві x

math.sin(x) сінус в радіанах
math.cos(x) косинус в радіанах
math.sqrt(x) корінь квадратний

Існують також стандартні функції для чисельного вирішення систем диференційних рівнянь, якими, як правило, описується дія фізично різнорідних компонентів та систем в цілому.

3.1.5. Чисельне рішення диференційних рівнянь

В мові Python для вирішення диференційних рівнянь використовується стандартна функція **odeint.**

Для використання цієї функції систему диференційних рівнянь потрібно представити у формі Коші. Це робиться наступним чином [6].

Якщо є диференційне рівняння вигляду

$$y^{(n)} = f(t, y, y^1, ,....y^{(n-1)});$$

з наступними початковими умовами

$$y(t_0) = y_0, y^1(t_0) = y_1,.....y^{(n-1)}(t_0) = f(t, y, y^1, ,....y^{(n-1)} = y_{n-1},$$

то задача полягає в пошуку чисельним методом функції, яка буде представляти рішення цього рівняння.

Для представлення заданого рівняння вищого порядку в формі Коші його необхідно перетворити у систему рівнянь першого порядку

$$y^1(t) = F(t, y), y(0) = y_0, \qquad (1)$$

де y – функція, рішення якої треба знайти, (якщо система складається з n – рівнянь, то y представлено вектором, а y0 – початкові значення функції (y випадку системи рівнянь y0 також є вектором)

Розглянемо більш детально використання стандартно функції **odeint** пакету Python.

Для звернення до цієї функції потрібно підготувати виначення аргументів та визначити за потребою інші опційні параметри.

Формат виклику функції є наступним

odeint(func, y0, t[,args=(), ...])

Аргумент **func** представляє ім'я Python функції двох змінних, першою з них є список y=[y1,y2,...,yn], а друга - t є незалежною змінною.

Функція **func(y,t)** забезпечує обчислення правих частин системи рівнянь (1) та повертає n значень функцій при заданих значеннях незалежного аргументу t.

Другий аргумент **y0** функции **odeint()** є масивом початкових значень при t=t0.

Третій аргумент **t** представляє масив значень часу для яких будуть обчислюватись рішення системи рівнянь.

Функция **odeint()** має також кілька необов'язкових опціональних аргументів (**[,args=(), ...]**)), наприклад, **rtol** , що задає відносну похибку результату, та **atol**, що задає абсолютну похибку результату. У разі відсутності цих аргументів їх значення задаються за замовчанням.

В результаті обчислень функція **odeint()** повертає масив значень t та y.

Приклад програми моделювання динамічних процесів наведено нижче

```
# Підключення бібліотек
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

 #Завдання констант F, m, V, b
F=0.1
b=0.2
m=0.1

#Опис функції def a
def a(V, t):
    a = (F - b * V) / m
```

```
        return a
#Опис функції def rp
def rp_fn(sol,t,F,b,m):
    x[0]=sol[1]
    x[1]=(F-b*sol[1])/m
    #x=[(F-b*x[0])/m,x[1]]
    return [x[0], x[1]]

#  Завдання початкових значень
x=[0,0.0]

# завдання точки початку, кроку та точки завершення розрахунку
t=np.linspace(0,2.5,300)

#Виклик функції odeint з параметрами (rp_fn,x,time, args=(F,b,m))
sol = odeint(rp_fn, [0.0, 0.0], t, args=(F,b,m))

#Підготовка, побудова та виведення графіку
plt.plot(t, sol[:, 0], 'b', label='x1')
plt.plot(t, sol[:, 1], 'g', label='x2')
plt.plot(t, a(sol[:, 1],t), 'r', label='x3')
plt.plot(np.array(t), a(sol[:, 1], t), "r-")

plt.legend(loc='best')
plt.xlabel('time')
plt.ylabel('x1(t), x2(t), x3(t)')
plt.grid()
plt.savefig('simulation.png')
plt.show()
```

В результаті виконання програми буде виведено вікно (рис.3.9) з графіками, що відображають прискорення, швидкість та переміщення рухомого об’єкту

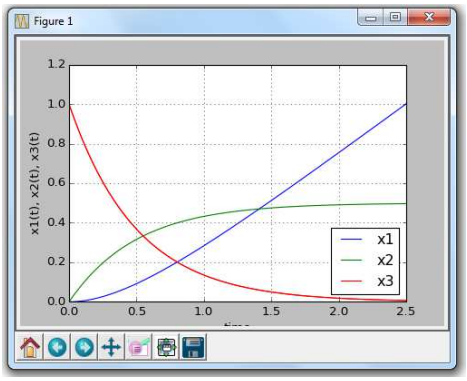


Рис.3.9. Вигляд вікна з графіками, що відображають динамічні процеси (x1 – крива переміщення; x2 – швидкість; x3 – прискорення.

4. ЗАСОБИ І ОСОБЛИВОСТІ ФІЗИЧНОГО МОДЕЛЮВАННЯ

4.1. Обладнання для фізичного моделювання

Вивчення курсу базується на виконанні лабораторних робіт з фізичним обладнанням. При цьому завданнями для студентів є з'ясування принципу дії фізично різнорідних елементів та систем, навчання користуватися типовими елементами, з'ясування їх статичних характеристик та створення математичних описів, що дозволяють моделювати дію елементів та систем. Для виконання лабораторних робіт студенти використовують спеціальні набори обладнання А1-А3, які містять електронні та електромеханічні елементи різного призначення, стенди для їх складання у функціональні компоненти, а також комплекти функціональних компонентів систем та вимірювальне обладнання (рис.4.1, рис. 4.2, рис.4.3),

Крім наведених наборів при виконанні лабораорни робіт використовуються також тестери та осцилографи.

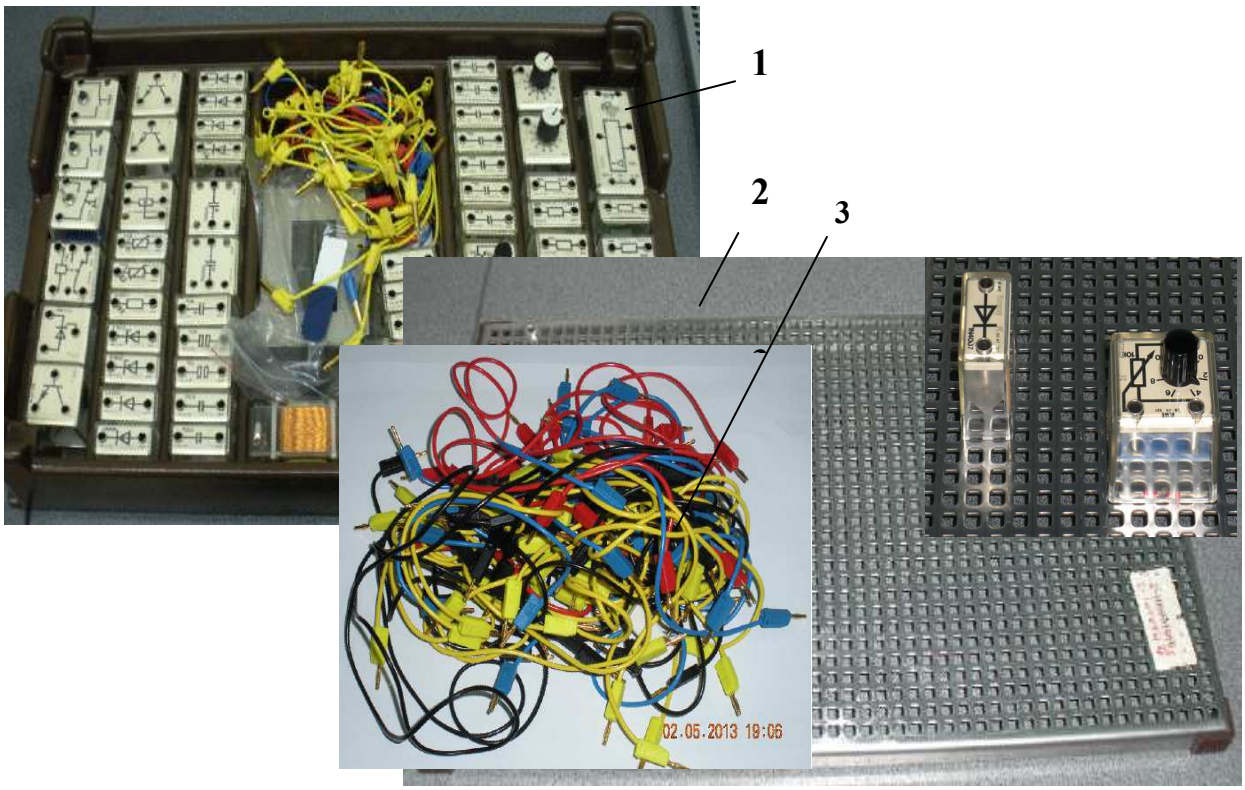


Рис.4.1. Комплект А1 елементів (1), панель для складання (2) та набір дротів (3) для з'єднання елементів

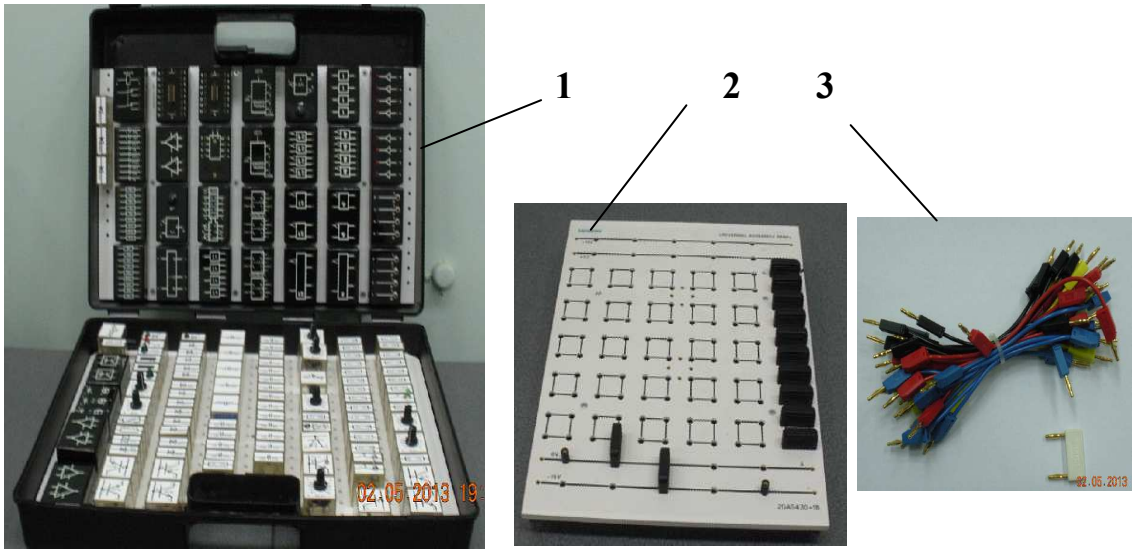


Рис.4.2. Комплект А2 елементів (1), панель для складання (2), набір дротів (3) для з’єднання елементів

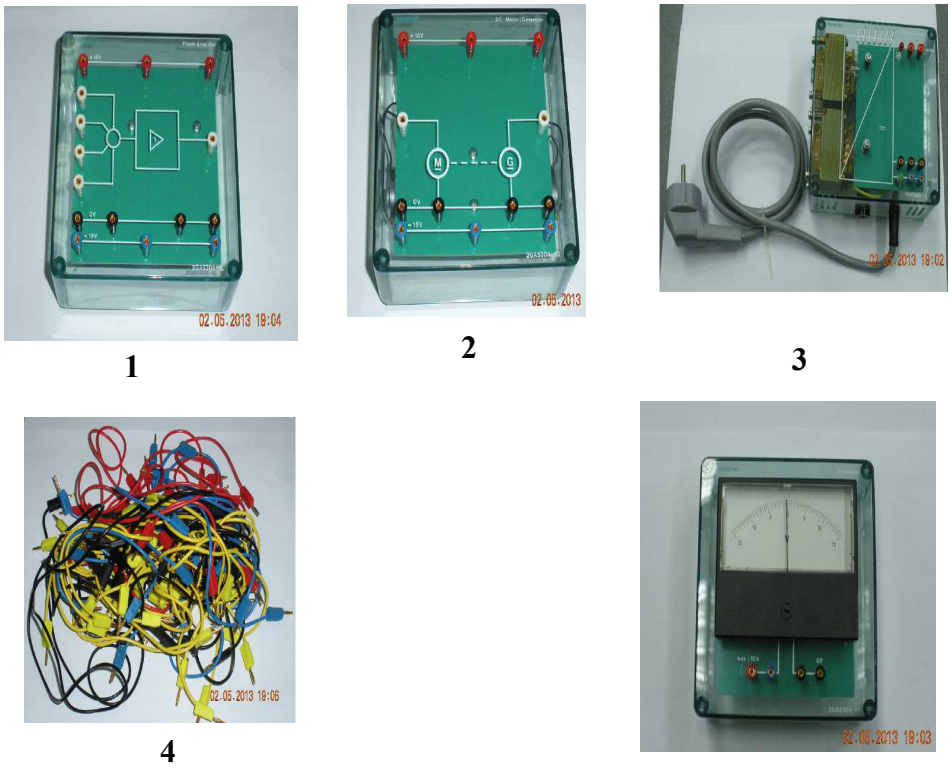


Рис.4.3. Комплект А3 компонентів: - електронний підсилювач з суматором (1), модуль електричний двигун-електричний генератор (2), модуль живлення (3), комплект дротів (4), вимірювальний прилад (5)

4.2. Фізичні експерименти та математичні моделі

На відміну від навичок та досвіду побудови програм за допомогою мов програмування фізичні експерименти потребують набуття інших навичок і досвіду. При цьому, важливим є розуміння і дотримання правил техніки безпеки.

Кожен студент перед початком роботи з обладнанням має бути ознайомлений з правилами техніки безпеки, зафіксувати це своїм підписом у відповідному журналі і ОБОВЯЗКОВО дотримуватись цих правил в ході виконання робіт.

Основним завданням при виконанні лабораторних робіт з фізичними компонентами систем є отримання навичок визначення їх характеристик та представлення дії компонентів в математичних моделях.

Завдання кожної лабораторної роботи виконуються в наступній черговості:

1. Зрозуміти принцип дії фізичного елементу заданого типу та описати його дію словами.
2. Визначити, яка фізична величина має подаватися на вхід елементу, а яка є результатом його роботи і є вихідною величиною.
3. Пригадати, що є статичною характеристикою, і сформулювати завдання визначення цієї характеристики саме для заданого фізичного елементу.
4. Розробити методику визначення статичної характеристики конкретного елементу з врахуванням умов його роботи.
5. Розробити форму таблиці для запису результатів досліджень.
6. Розробити принципову схему стенду для визначення статичних характеристик заданого елементу.
7. Підібрати обладнання та скласти експериментальний стенд.
8. Провести дослідження, фіксуючи результат в таблиці.
9. Побудувати статичну характеристику досліджуваного елементу.
10. Побудувати математичний опис дії досліджуваного елементу.
11. Змоделювати дію елементу для заданих вхідних сигналів та порівняти отримані результати з експериментальними даними.
12. Зробити висновки по результатах виконаної роботи.
13. Оформити протокол та захистити роботу.

Особливості роботи з електронними компонентами та вимірювальним обладнанням

В ході постановки експериментів і проведення досліджень необхідно навчитися представляти на схемах кожен елемент або компонент технічної

системи, коректно складати експериментальний стенд, задавати умови його роботи та виконувати вимірювання.

Для цього необхідно додержуватись наступних рекомендацій.

1. Перед постановкою експериментів потрібно на основі загальної схеми (рис.4.4) підготувати схему проведення досліджень.

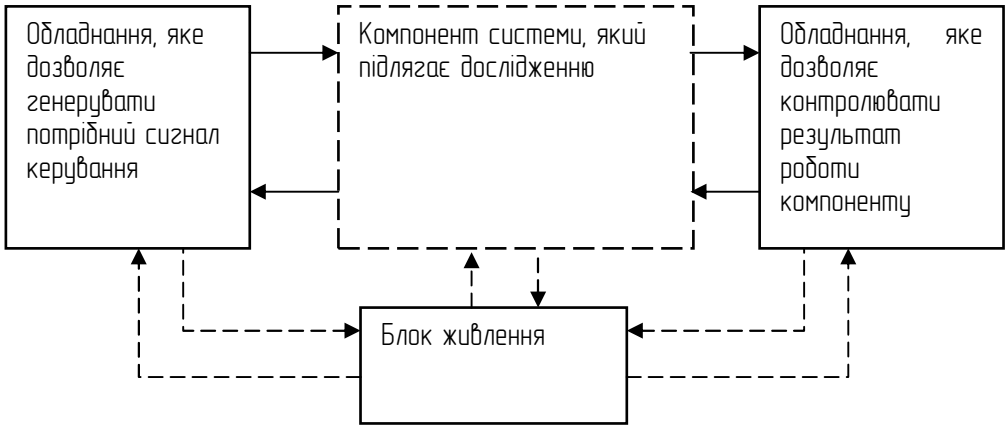


Рис. 4.4. Загальна схема для проведення досліджень компонентів фізично різнорідних систем

2.Вибрати обладнання для задання та вимірювання сигналів, а також джерело живлення. Сигнал керування може задаватися від генератора сигналів. Сигнал керування також може задаватися в ручному режимі за допомогою перемикача.

Обладнанням контролю результатів роботи можуть бути вимірювальні прилади: - амперметр, вольтметр, омметр, або комбінований пристрій - тестер, а також осцилограф.

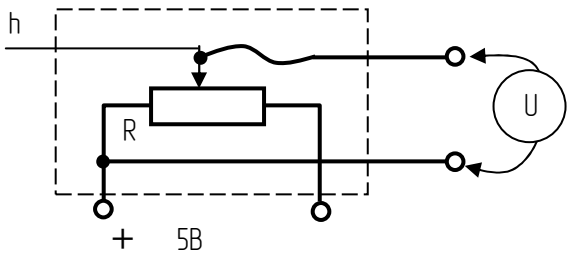
3. Скласти стенд, провести дослідження та обробити результати.

Для прикладу наведено схему дослідження інформаційного елементу, який дозволяє отримувати інформацію про механічне переміщення. При цьому, результатом роботи елементу є електрична напруга на його виході (рис.4.5.)

а) б) в)

Рис. 4.5. Загальне представлення компоненту (а), його зовнішній вигляд (б), позначення на принципових схемах (в)

Принципова схема дослідження інформаційного елементу (рис.4.6 а), таблиця для фіксації результатів (рис.4.6 б) і графічне представлення результату (рис.4.7 а) ілюструють етапи виконання роботи.



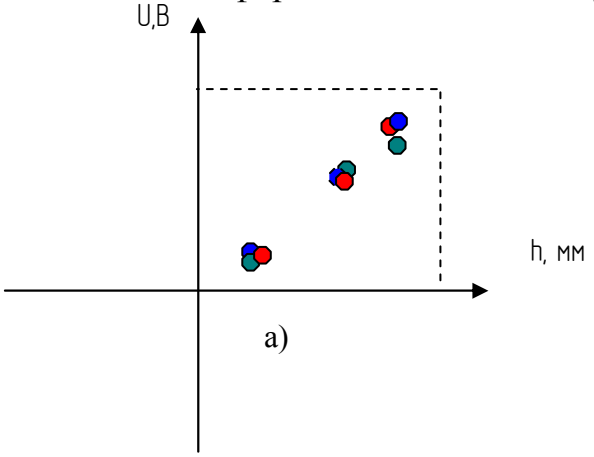
а)

№	h, мм	U1, В	U2, В	U3, В
1	0	0	0	0
2	6	5	5.1	4.9
3	3	2.5	2.48	2.51
4				
5				
6				

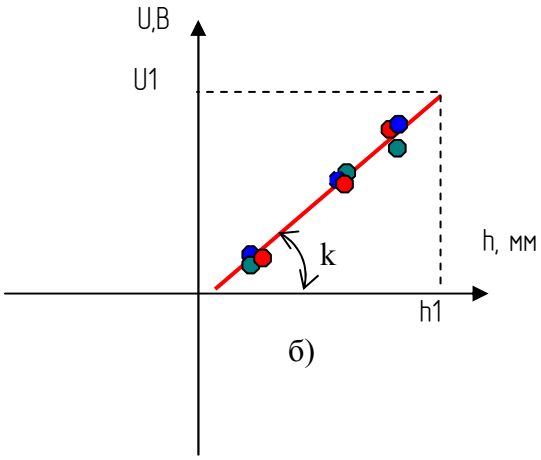
б)

Рис.4.6. Форми представлення етапів виконання лабораторної роботи (а- принципова схема стенду для дослідження, б – таблиця з результатами вимірювань

4. Наступним етапом виконання роботи є побудова математичного опису для імітації дії інформаційного елементу в комп'ютерному середовищі (рис.4.7 б).



а)



б)

Рис.4.7. Форми представлення результатів (а- графічне представлення результатів вимірів, б – апроксимація експериментальних результатів за допомогою математичної залежності

Математичний опис будують за шляхом апроксимації експериментальних даних за допомогою вибору відповідної математичної залежності. Вона має відповідати набору експериментальних даних з адаптованими до них коефіцієнтами. Наприклад, для результатів визначення статичної характеристики розглянутого вище інформаційного елементу такою є лінійна залежність

$$U = k \cdot h,$$

де $k -$ коефіцієнт, який визначається за формулою

$$k = tg(U_1 / h_1).$$

Аналогічним чином мають виконуватись експериментальні роботи, наведені в лабораторних завданнях.

5. ЗАВДАННЯ ДЛЯ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ

5.1. Лабораторна робота. Візуалізація математичних функцій

Мета: Отримання навичок автоматизації обчислювальних операцій та побудови графіків.

Основні відомості для виконання операцій у пакеті MatLAB.

Математичні операції пакету.

Корінь квадратний: **sqrt(x).**

Модуль: **abs(x).**

Додавання, віднімання, множення: **+,-,*.**

Ділення зліва на право: **./**

Ділення с права вліво: **.**

Взведення у ступінь: **.^**

Завдання циклу: **for** z=0 : 0.05 : 1.5 <оператори> **end**;

де 0 – початкове значення, 0.05 – шаг прирощення, 1.5- кінцеве значення.

Побудова та виведення на екран поля графіку:з трьома точками **plot(x,y,x1,y1,x2,y2);**

де x,y – координати точки 1, x1,y1- координати точки 2, x2,y2- координати точки 3.

Виведення надпису: **title('Графік функцій……');**

Ініціаліація та збереження значень в індексованій змінній

b(1)=20; b(2)=43; и т.д., де b – індексована змінна, 1, 2… - індекси; 20, 43 – значення.

Завдання на лабораторну роботу. Написати програму для вирішення системи рівнянь графічним способом.

Завдання

№ за списком	Система рівнянь
1.	$Y_1 = \sqrt{(25 - x^2)}; Y_2 = -3/4 \cdot x.$
2.	$Y_1 = \sqrt{(x+4)}; Y_2 = 2 \cdot \sqrt{(6-x)}.$
3.	$Y_1 = 7 \cdot \sqrt{(5 + 4x - x^2)}; Y_2 = 14 - 2 \cdot x.$

4.	$Y_1 = x^2; Y_2 = 7 \cdot \lvert x \rvert - 10.$
5.	$Y_1 = x^2; Y_2 = 3 \cdot \lvert x \rvert + 4.$
6.	$Y_1 = x^2; Y_2 = 7 \cdot \lvert x \rvert - 12.$
7.	$Y_1 = x^2; Y_2 = 2 \cdot x + 5 \cdot \lvert x \rvert + 7.$
8.	$Y_1 = \sqrt{(x-2)}; Y_2 = x - 3.$
9.	$Y_1 = \sqrt{(4 - x^2)}; Y_2 = -1 / x.$
10.	$Y_1 = \sqrt{(x^2 - 3 \cdot x - 10)}; Y_2 = 8 - x.$
11.	$Y_1 = x^3; Y_2 = 3 \cdot \lvert x \rvert + 4.$
12.	$Y_1 = x^2; Y_2 = -1 / x.$

Порядок захисту. Робота захищається шляхом демонстрації програми та відповідей на контрольні запитання.

Додаткові відомості. При налаштуванні програми буває зручним контроль проміжних результатів виконання окремих операторів. Для виведення таких результатів на екран в програмі після оператора потрібно прибрати символ «;».

5.2 Лабораторна робота. Розгалужена будова програми

Мета: Отримання навичок розробки розгалужених програм.

Основні відомості. Програма може складатися з кількох М-файлів, які взаємодіють один з одним у процесі обчислень. Зазвичай один із М-файлів є головним, іноді його називають файлом-сценарієм, який керує ходом обчислень. Головний файл в своєму тілі може використовувати інші М-файли, які є зовнішніми по відношенню до нього. Такі М-файли, як правило містять в собі частини програми, які можуть бути використані багатократно з різними вхідними параметрами. Називають такі М-файли М-функціями. Звернення з головного файлу до М-файлів функцій виконується за допомогою їх імен. Для

коректної роботи М-файл, що представляє собою функцію, повинен мати ряд формальних ознак. Такими ознаками є:

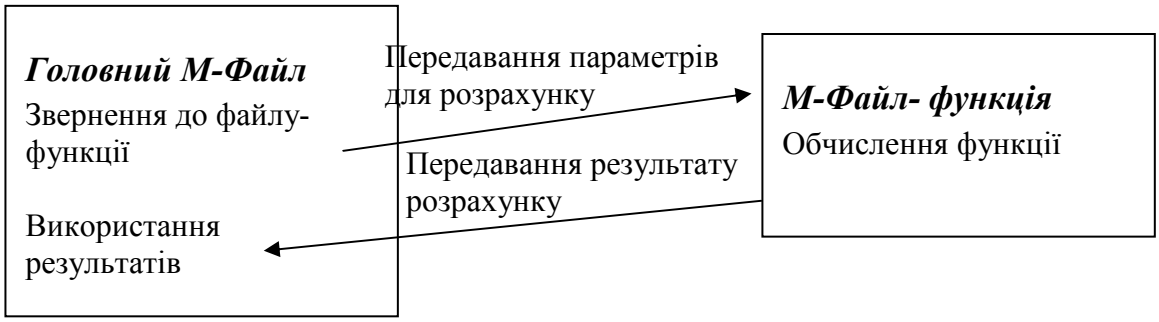
- ключове слово – **function;**
- символ, що позначає змінну яка повертається;
- ім'я функції зі списком значень.

Звернення до М-файлу функції з головного файлу виконується шляхом запису імені функції зі списком змінних, які потрібні для її виконання.

Приклад запису М-файлу функції:

```
%Приклад функції---------------------------------------
function y=F1(x,d)    %  "x" та  "d"  є параметрами функції, "y" – змінна, яка
y=x+d;                %  буде отримувати результат виконання функції
end   %Кінець
```

----------Приклад структури програми----------



---

*В коді символом «%» поначається коментар, який обробником програм пропускається.

Важливо. При збереженні функції в файлі, його ім'я має бути таким самим, як і імя функції, але з розширенням «.m».

Приклад. Для функції «**function** y=**F1**(x,d)» треба зберігати файл з ім'ям **F1.m**

Приклад звернення до файлу-функці F1 з головного файлу

Головний М-Файл

```
x=2:
d=16:
k=21;
y = F1(x,d);  %Так виглядає звернення до функції
%Нижче  може бути записана формула, яка використовує результат
% виконання функції
z=y+k;
```

При використанні функцій треба враховувати, що в файлах функціях всі імена змінних, що знаходяться всередині М-файлу, а також змінні, визначені в заголовку є локальними і їх дія не розповсюджується на головний файл.

Всі змінні, які використовуються у головному файлі-сценарії створюють певну робочу область, в якій їх значення зберігаються протягом всього сеансу роботи з системою. Якщо в програмі одночасно використовуютья кілька файлів-сценаріїв, то ї робоча область є єдиною для всіх файлів-сценаріїв, що викликаються під час роботи з системою.

Завдання на лабораторну роботу. Написати програму для вирішення системи рівнянь графічним способом з використанням М-файлів. При цьому програма має складатися з трьох частин: - М-файла головної програми; - М-фалу функції обчислення значень для першого графіку; - М-файлу функції обчислення значень для другого графіку. Система рівнянь береться з попередньої лабораторної роботи.

Порядок захисту. Робота захищається шляхом демонстрації програми та відповідей на контрольні запитання.

5.3. Лабораторна робота. Моделювання і дослідження поведінки математичної функції

Мета: Отримання навичок побудови математичних моделей з використанням діалогового режиму та графічної інтерпретації результатів.

Основні відомості. Засоби вводу та виводу інформації пакету MatLAB.

Вивід тексту на екран: **disp(‘текст’)**

Формування текстового рядку, який містить числове значення: **sprintf(‘параметр %g’, x)**

Запит на введення числового значення з клавіатури: **x=input(‘x=’)**

Організація діалогу з користувачем за допомогою меню: **z=menu(‘Заголовок’,’Кнопка_1’,’Кнопка_2’…..)**

Завдання паузи: **pause**

Очищення поля виводу у вікні: **delete(gca)**

Перетворення числового значення у рядкову змінну: **y=num2str(x), де** x – числове значення, y – рядкова змінна.

Формування полів виводу графічної та текстової інформації у вікні виводу**: subplot(m,n,p)**, де m – число полів по вертикалі, n- число полів по горизонталі, p- номера підвікон, що призначені для виводу інформації.

Приклад 1. **subplot(2,3,6)** – вікно буде розділено на 6- полів, по три поля на кожному з двух рівней. Вивід інформації буде відбуватися у праве нижнє поле.

Приклад 2. **subplot(2,3,[4,5,6]) –** Вивід інформації буде відбуватися у три поля, що розташовані знизу.

Виведення тексту в поле вікна: **h=text(x,y,’**Початкові данні:**’,’FontSize’,12)** , де x,y –координати для виводу тексту, **’FontSize’,12** – визначає розмір шрифту у пікселях.

Очищення вікна від зображення координатних вісей та надписів на них: **axis(‘off’)**

Завдання на лабораторну роботу. Для заданої функції написати програму для побудови графіків та виведення їх на екран. При цьому необхідно забезпечити можливість завдання параметрів функції у діалоговому режимі та виведення результатів в три різні поля вікна виводу. В перше поле вікна треба виводити графік функції при початкових значеннях параметрів, у друге поле вікна – графік функції для нових значень параметрів, що задаються користувачем, а у третє поле – текстової інформації, про тип та параметри заданої функції.

Завдання

№ за списком	Вид функції
1.	$Y = a + b \cdot x^m.$
2.	$Y = a + (1/b) \cdot x^m.$
3.	$Y = a + (b/c) \cdot x^m.$

4.	$Y = a + b / x^m.$
5.	$Y = (a + b \cdot x^m) \cdot c.$
6.	$Y = a + b \cdot \sin(F + \omega \cdot t).$
7.	$Y = a / x + b.$
8.	$Y = a \cdot (1 - \sin(\omega \cdot t).$
9.	$Y = b - c \cdot \sin(F + \omega \cdot t).$
10.	$Y = a + (1 / b) \cdot x^m.$
11	$Y = (a + b \cdot x^m) \cdot c.$
12.	$Y = A \cdot x^n - c.$

Порядок захисту. Робота захищається шляхом демонстрації програми та відповідей на контрольні запитання.

Рекомендації. Для написання програми треба використовувати операції з циклами та М-файли.

5.4. Лабораторна робота. Прогнозна модель дії об'єкту

Мета. Отримання навичок побудови математичних моделей на основі експериментальних даних з використанням діалогового режиму та графічної інтерпретації результатів.

Основні відомості. Математична модель на основі експериментальних даних може бути побудована шляхом їх апроксимації за допомогою поліномів виду: y(x)=A3*x^3+A2*x^2+A1*x+A0, де A3..A0- коефіцієнти поліному.

Спосіб завдання вектору прямою ініціалізацією: v= [1.5 -2.4 4.9e3];

Функція апроксимації даних, що були отримані експериментальним шляхом: polyfit(X,Y,n),

де X- вектор аргументів, Y – вектор експериментальних значень, n - ступінь апроксимуючого поліному.

Функція для обчислення значень поліному за завданими значеннями його аргументів: polyval(P,x),

де P – заданий вектор коефіцієнтів поліному, x- значення аргументу.

Завдання вектору шляхом його формування: x=0 : 0.05 : 1.5;

де 0 – початкове значення, 0.05 – крок прирощення, 1.5- кінцеве значення.

Вивід графіку: plot(x,y,x1,y1,x2,y2);

де x,y – координати точки 1, x1,y1- координати точки 2, x2,y2- координати точки 3.

Вивід надпису: title('Графік апроксимації результату експериментів');

Завдання на лабораторну роботу. Написати програму для апроксимації завданих експериментальних даних за допомогою поліномів 1-го, 2-го та 3-го ступеню. Забезпечити можливість вводу даних у діалоговому режимі. Вивести результати у вигляді графіків, порівняти їх та зробити висновки.

Завдання

№ за списком	**X**	1	2	3	4	5	6	7	8
1	***Y***	-1.1	0.2	0.4	0.8	0.7	0.6	0.4	0.1
2	***Y***	-3,1	0.8	0.9	0.7	0.3	0.1	0.15	0.01
3	***Y***	-1.1	0.2	0.4	0.5	0.8	0.3	0.25	0.2
4	***Y***	2.1	1.2	3.6	6.1	5.3	4.0	3.05	2.0
5	***Y***	-0.5	1.0	1.8	2.3	1,5	1.0	0.8	0.2
6	***Y***	-0.7	0.1	0.7	0.75	0.6	0.4	0.25	0.1
7	***Y***	-0.9	0.2	0.3	0.9	0.8	0.7	0.3	0.1
8	***Y***	-0.5	0.1	0.7	0.8	0.9	0.3	0.4	0.3
9	***Y***	-1.6	1.1	3.4	5.1	5.0	3,8	2.1	2.5
10	***Y***	-1.3	1.8	2.1	3.4	4.2	2.1	1.1	0.8

Порядок захисту. Робота захищається шляхом демонстрації програми та відповідей на контрольні запитання

Контрольні запитання. 1. Як впливає ступінь полінома на точність апроксимуючої функції? 2.Як користуватись прогнозною моделлю?

5.5. Лабораторна робота. Моделювання динамічного процесу руху тіла

Мета: Отримання навичок моделювання динамічних процесів при дії механічних елементів фізично різнорідних системв.

Постановка задачі. На рисунку (рис.1) представлена схема об'єкту моделювання. На тверде тіло, що лежить на плоскій поверхні, діє сила F. Результатом є рух тіла.

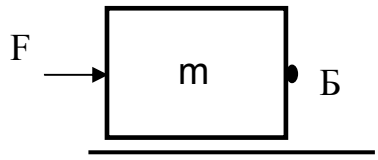


Рис.1.Схема об'єкту моделювання

Математичнй опис процесу дії об'єкту

Завдання на лабораторну роботу. Розробити математичний опис процесу руху тіла під дією зовнішньої сили. При цьому треба врахувати силу спротиву від в'язкого тертя тіла по поверхні руху (прийняти значення коефіцієнту в'язкого тертя b=10.0). Перетворити математичний опис в модель. Завдання 1. Змоделювати процес руху твердого тіла при різних законах зміни в часі зовнішньої сили (для т.Б, яка розташована на твердому тілі, спостерігати за змінами прискорення, швидкості та переміщення). Завдання 2. Визначити вплив величини маси тіла на час його руху при імпульсній зміні зовнішньої сили F. Результати представити у вигляді графіку залежності t=f(m), де t – час руху тіла, m—масса тіла.

Завдання

№ за списком	Значення сили F,N	Час дії сили, с (імпульс)
1.	**0.5**	**0.1**
2.	**3.5**	**0.05**
3.	**0.2.5**	**0.1**
4.	**6**	**0.05**
5.	**3**	**0.1**
6.	**4**	**0.05**
7.	**35**	**0.1**
8.	**30**	**0.05**
9.	**120**	**0.1**
10.	**63**	**0.05**
11.	**10**	**0.1**
12.	**45**	**0.05**

Графік, який ілюструє отримані результати

План виконання роботи. 1.Розробити математичний опис процесу руху тіла під дією зовнішньої сили. 2.Побудувати математичну модель. 3.Задати значення параметрів (m b, F(t)). 4. Виконати завдання І. Змоделювати процес для різних законах зміни F від часу. 5. Провести аналіз отриманих результатів. 6. Виконати завдання ІІ. Намалювати систему координат для побудови графіку t=f(m), На цій системі координат, до проведення експериментів, намалювати графік вказаної залежності відповідно до вашого розуміння. Після цього провести послідовно серію експериментів для різних значень мас тіла та однакових імпульсах сили F. В кожному експерименті визначати час руху маси та заносити отримані значення в таблицю. На основі отриманих даних побудувати графік t=f(m). Проаналізувати результат, порівняти його зі своїм попереднім розумінням та зробити висновки.

Порядок захисту. Робота захищається за наявності працюючої програми та оформленого протоколу шляхом відповідей на контрольні запитання.

Контрольні запитання. 1. Якими силами було нехтувано при побудові математичного опису? 2. Як визначається сила в'язкого тертя?

5.6. Лабораторна робота. Моделювання процесів у пружній системі

Мета: Отримання навичок моделювання динамічних процесів при взаємодії механічних елементів фізично різнорідних системв.

Постановка задачі. На рисунку (рис.1.) представлено схему об'єкту моделювання. На пружину у т. А діє сигнал X. Тіло масою m під дією сили, що виникає у пружині за рахунок її стиснення починає рухатись. Швидкість та положення точки Б, що розташована на тілі, залежить від сигналу X, жорсткості пружини C та величини маси m.

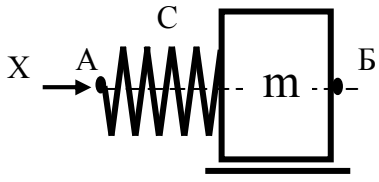


Рис.1. Схема об'єкту моделювання

Математичний опис работи об'єкту

Завдання на лабораторну роботу. Розробити математичний опис процесу руху тіла. Перетворити математичний опис у модель. Змоделювати процес руу тіла для двох різних типів вхідних сигналів.

1. Для ступінчатої зміни положення т. А (x=x0, де x- переміщення, x0- значення переміщення).

2. Для вхідного сигналу виду: x=x0*sin(ω*t), де x- переміщення, x0- максимальна амплітуда переміщення, ω - частота, t- час.

Результати представити у формі графіків: графік залежності швидкості т. Б від часу для при ступінчатого переміщення т. А на завдану величину; графік залежності максимальної швидкості руху т. Б, для різних частот коливань т. А.

Завдання

№ за списком	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
m, кг	5	3.5	2..5	0.6	0.3	4.6	0.5	0.7	10	1.5	7.0	0.8

План виконання роботи. 1.Розробити математичний опис процесу. 2. Провести моделювання процесу. 3. Результати роботи оформити у вигляді протоколу.

Результати моделювання

Порядок захисту. Робота захищається за наявності протоколу**.** Протокол має містити розрахункову схему об’єкту моделювання, опис моделі, значення параметрів для яких проведено дослідження, графіки залежності: графік залежності швидкості т. Б від часу для ступінчатого зміщення т. А; графік залежності максимальної швидкості руху т. Б, для різних частот коливань т. А.

Додаткові умови. Значення параметрів, що не вказані, обираються самостійно.

Контрольні запитання. 1. Поясніть термін власна частота коливань. 2. Від яких параметрів залежить коефіцієнт жосткості пружини?

5.7. Лабораторна робота. Моделювання процесів в гідравлічних компонентах систем

Мета: Отримання навичок моделювання динамічних процесів при взаємодії рідинних елементів фізично різнорідних системв.

Об'єкт моделювання. На рисунку (рис.1) представлена схема об'єкту моделювання. Камера підключена до джерела тиску **p,** який **є** більшим ніж тиск в середині камери. Відбувається процес додаткового наповнення камери рідиною. Результатом процесу є збільшення тиску в камері.

Математичний опис процесу*

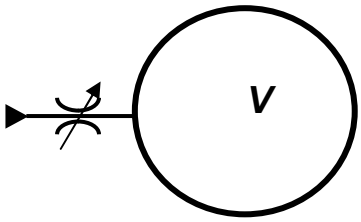


Рис.1. Схема об'єкту моделювання

Завдання. Розробити математичний опис процесу зміни тиску в середині камери; перетворити математичний опис в математичну модель. Змоделювати процес при різних параметрах об'єкту. Визначити вплив параметрів об'єкту та зовнішнього середовища на час зміни тиску в середині камери. Результати представити у формі графіків залежностей 1: p=f(t); 2: t1=f(V); 3: t1=f (d); де p – тиск в камері, t – час процесу, V – об'єм, d –діаметр отвору дроселя, t 1– час зміни тиску в камері.

Варіанти для індивідуального завдання.

№	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
p, МПа	2.0	5.0	10.0	12.0	15.0	17.0	20.0	25.0	30.0	35.0	40.0	45.0

Графики, що ілюструють отримані результати

План виконання роботи. 1.Розробити математичний опис процесу та трансформувати опис в модель. 2. За своїм розумінням до проведення експериментів намалювати графік: як змінюється в часі тиск при наповнені камери, 3. Задати значення параметрів. 4. Провести моделювання і за результатами побудувати графіки. 5. Провести аналіз отриманих результатів та зробити висновки. Порівняти отриманий графік процесу з графіком, який був намальований до проведення експериментів.

Порядок захисту роботи. Робота захищається при наявності робочої версії моделі та оформленого протоколу шляхом обґрунтованих відповідей на контрольні запитання.

Контрольні запитання. 1. Чи є вплив діаметра камери та діаметро отвора в дроселі однаковим на час її наповнення до однієї величини тиску? 2.Як врахувати деформацію камери під дією внутрішнього тиску?

**Додаткові відомості*

Для опису процесу наповнення камери використовувати наступні математичні залежності.

Визначення витрати через дросель

$q = \mu \cdot f \cdot \sqrt{2/\rho \cdot (p_2 - p_1)}$, де $\mu -$ коефіцієнт витрати (прийняти 0.6), $f -$ площа поперечного перерізу дроселя (треба задавати), $\rho -$ - щільність мінеральної олії, $p_1, p_2 -$ тиск на вході в росель і тиск на виоді дроселя відповідно.

Визначення змін тиску в камері при подаванні до неї витрати

$\frac{dp}{dt} = \frac{E}{V} q$, де

$p -$ тиск в камері, $E -$ адіабатичний модуль пружності робочої рідини, $V -$ об’єм камери, $q -$ витрата, яка потрапляє в камеру.

6. ЗАВДАННЯ ДЛЯ ФІЗИЧНОГО МОДЕЛЮВАННЯ

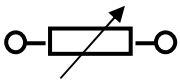
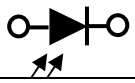
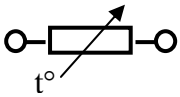
6.1. Лабораторна робота. Дослідження та математичний опис статичних характеристик інформаційних елементів

Мета. Ознайомлення з принципами дії інформаційних елементів фізично різнорідних систем та їх представленнями в математичних моделях.

Загальні відомості. Інформаційні елементи входять до складу автоматичних систем. Їх функцією є отримання інформації від зовнішнього середовища та від елементів системи. В залежності від типу інформаційного сигналу використовують різні принципи дії інформаційних елементів. Вони можуть будуватися наприклад, на осові електричних потенціометрів, світло діодів, терморезисторів та ін.. Кожен з таких елементів має свою статичну характеристику. Саме їх визначення та представлення в математичній формі є задачею лабораторної роботи

Лабораторне обладнання. Комплект експериментального обладнання А1 та засоби вимірювання – вольтметр, амперметр.

Безпека. Робота з електричними пристроями потребує виконання заходів безпеки. Підключення живлення до експериментального стенду виконується після ретельної перевірки правильності складання схеми та виключно в присутності викладача.

Тип елемента	Схема принципу дії елемента	Математичний опис дії елементу
Механічне переміщення-електрика		
Світло-електрика		
Температура-електрика		

Завдання та послідовність виконання роботи

1. Розробити схему визначення статичних характеристик для заданого типу інформаційного елементу.
2. Визначити, яка змінна визначає вхідну дію, та яка змінна визначає вихідний результат.
3. Розробити методику проведення дослідження статичної характеристики
4. Розробити форму таблиці для запису результатів.
5. Підібрати обладнання та скласти стенд відповідно до схеми.

6. Провести дослідження, фіксуючи результати в таблиці.
7. Побудувати статичну характеристику інформаційного елементу
8. Побудувати математичний опис статичної характеристики.
9. Скласти протокол лабораторної роботи.

Зміст протоколу. Протокол має містити принципову схему стенду з описом його роботи, опис виконання всіх пунктів плану і висновки за результатами практичної перевірки.

Контрольні запитання. 1. Поясніть принцип дії інформаційного елементу для отримання інформації про тиск. 2. Намалюйте графік його статичної характеристики.

6.2 Лабораторна робота. Дослідження та математичний опис статичних характеристик підсилюючих елементів

Мета. Ознайомлення з принципом дії підсилюючих елементів фізично різнорідних систем та їх представленнями в математичних моделях.

Загальні відомості. В технічних системах використовують елементи, які дозволяють підсилювати вхідний сигнал за потужністю (рис.1). При цьому зміни вихідного сигналу при зміні вхідного сигналу залежать від характеристик підсилюючого елементу. Характеристики елементу суттєво впливають на характеристики всієї системи, тоиу вони мають бути враховані при проектуванні систем.

Лабораторне обладнання. Комплект експериментального обладнання А2 та засоби вимірювання – вольтметр, амперметр.

Безпека. Робота з електричними пристроями потребує виконання заходів безпеки. Підключення живлення до експериментального стенду виконується після ретельної перевірки правильності складання схеми та виключно в присутності викладача.

В роботі має досліджуватись релейний підсилювач, принципову схему якого наведено на рис.2.

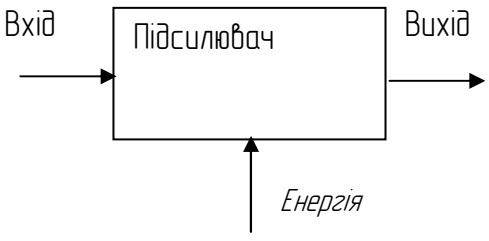


Рис.1. Загальна схема підсилюючого елементу

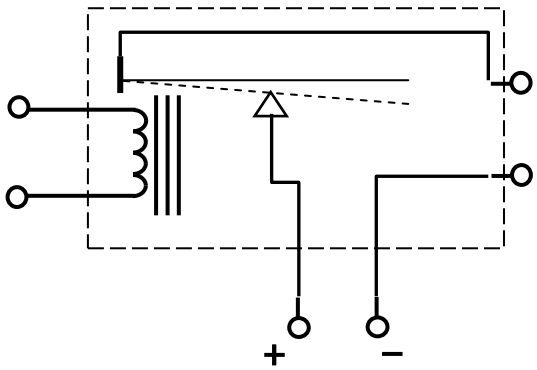


Рис.2. Принципова схема підсилювача релейного типу

Завдання та послідовність виконання роботи

1. Розробити схему стенду для визначення статичних характеристик заданого типу підсилюючого елементу.
2. Визначити, яка змінна визначає вхідну дію, та яка змінна визначає вихідний результат.
3. Розробити методику проведення дослідження статичної характеристики
4. Розробити форму таблиці для запису результатів.
5. Підібрати обладнання та скласти стенд відповідно до схеми.
6. Провести дослідження, фіксуючи результат в таблиці.
7. Побудувати статичну характеристику підсилюючого елементу.
8. Побудувати математичний опис статичної характеристик.
9. Скласти протокол лабораторної роботи.

Зміст протоколу. Протокол має містити принципову схему стенду з описом його роботи, опис виконання всіх пунктів плану і висновки за результатами практичної перевірки.

Контрольні запитання. 1. Поясніть принцип дії релейного підсидювача. 2. Поясніть графік його статичної характеристики.

6.3. Лабораторна робота. Розробка, перевірка та математичне представлення модулю узгодження

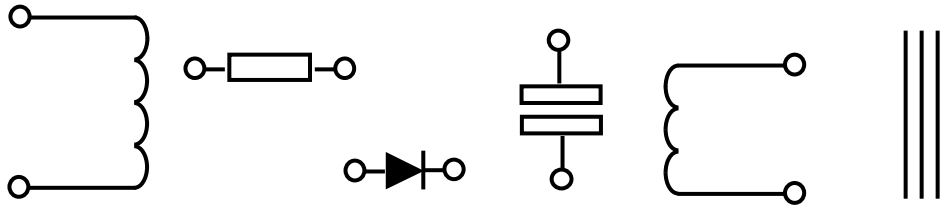
Мета. Ознайомлення з принципом дії узгоджуючих елементів фізично різнорідних систем та їх представленнями в математичних моделях.

Загальні відомості. Джерело електричної енергії 220 В змінного струму є в кожному приміщенні, але автоматичні прилади для свого живлення часто потребують іншої напруги та струму. Для їх узгодження використовують модулі на основі трансформаторів. Робота трансформатора грунтується на використанні ефекту утворення електромагнітного поля навколо провідників по яким тече змінний струм та ефекту виникнення електричного струму у провіднику у разі, якщо він знаходиться в межах дії електромагнітного поля (електромагнітна індукція). Коефіцієнт трансформації визначається як співвідношення кількості витків провідника у вихідній та вхідній обмотках. Для перетворення змінного струму в постійний (випрямлення) застосовують напівпровідникові діоди. Для стабілізації напруги використовують конденсатори (рис.1).

Лабораторне обладнання. Комплект експериментального обладнання А1 та засоби вимірювання – вольтметр, амперметр.

Безпека. Робота з електричними пристроями потребує виконання заходів безпеки. Підключення живлення до експериментального стенду виконується після ретельної перевірки правильності складання схеми та виключно в присутності викладача.

Завдання 1. 1. Розробити принципову схему модулю для



а) б) в) г) д) е)

Рис.1. Позначення елементів автоматичних систем. а – вхідна обмотка трансформатору, б – опір, в- діод, г- конденсатор, д – вихідна обмотка трансформатору, е – провідник електромагнітного поля (серцевина трансформатора)

узгодження напруги джерела (24В 50 Гц) з напругою, яка потрібна споживачеві (12В). При цьому необхідно гарантувати його безпечну роботу.

2. Провести розрахунки і визначити параметри елементів.

3. Перевірити робото спроможність модулю на стенді.

Завдання 2. Розробити принципову схему модулю з забезпеченням вихідного струму одного напрямку. Розрахувати та перевірити його роботу експериментально.

Завдання 3. Розробити принципову схему модулю з випрямленням вихідного струму та стабілізацією напруги. Розрахувати та перевірити його роботу експериментально.

Вимоги до модулю узгодження. 1. Передбачити можливість включення і виключення модулю в ручному режимі.

Хід виконання роботи. 1. Вибрати потрібні складові частини.

2. Розробити принципову схему модулю.

3. Перевірити логіку роботи.

4. Підібрати обладнання та скласти стенд відповідно до схеми.. Живлення **НЕ вмикати!**

5. Запросити викладача.

6. Перевірити робото спроможність модулю.

7. Скласти протокол лабораторної роботи.

Зміст протоколу. Протокол має містити принципову схему стенду з описом його роботи, опис виконання всіх пунктів плану і висновки за результатами практичної перевірки.

Контрольні запитання. 1. Поясніть причину використання елементів та модулів узгодження. 2. Надайте приклади їх принципів дії.

6.4. Лабораторна робота. Дослідження та математичний опис статичних характеристик обчислювальних елементів

Мета. Ознайомлення з принципом дії обчислювальних елементів фізично різнорідних систем та їх представленнями в математичних моделях.

Лабораторне обладнання. Комплект експериментального обладнання А2 та засоби вимірювання – вольтметр, амперметр.

Безпека. Робота з електричними пристроями потребує виконання заходів безпеки. Підключення живлення до експериментального стенду виконується після ретельної перевірки правильності складання схеми та виключно в присутності викладача.

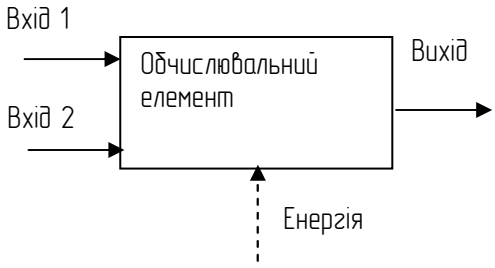


Рис.1.Загальна схема обчислювального елементу

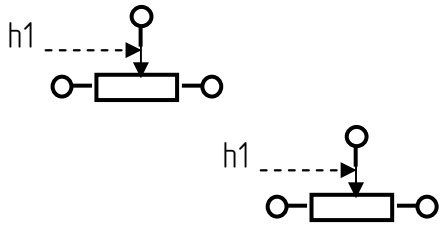


Рис.2. Заготовка до схеми обчислювального елементу на електричних потенціометрах

Завдання та послідовність виконання роботи

1. Розробити схему обчислювального елементу для алгебраїчного додавання механічних сигналів, використовуючи електричні потенціометри (рис.1,2).
2. Розробити схему визначення статичних характеристик для заданого типу обчислювального елементу.
3. Визначити, які змінні визначать вхідну дію, та яка змінна визначає вихідний результат.
4. Розробити методику проведення дослідження статичної характеристики.
5. Розробити форму таблиці для запису результатів.

6. Підібрати обладнання та скласти стенд відповідно до схеми.
7. Провести дослідження, фіксуючи результати в таблиці.
8. Побудувати статичну характеристику, що ілюструє роботу обчислювального елементу.
9. Побудувати математичний опис статичної характеристик.
10. Скласти протокол лабораторної роботи.

Зміст протоколу. Протокол має містити принципову схему стенду з описом його роботи, опис виконання всіх пунктів плану і висновки за результатами практичної перевірки.

Контрольні запитання. 1. Поясніть принцип дії обчислювального елементу. 2. Який принцип може бути використано для побудуви обчислювального елементу з більшою кількістю вхідних сигналів?.

6.5. Лабораторна робота. Дослідження та математичний опис статичних характеристик виконавчих елементів

Мета. Ознайомлення з принципом дії виконавчих елементів фізично різнорідних систем та їх представленнями в математичних моделях.

Загальні відомості. В тенічних системах використовують елементи, які дозволяють виконувати команди керування, наприклад рухати робочі органи об'єктів. Одним з таких елементів є електричний двигун (рис.1,2). На вхід двигуна подається електрична напруга, що призводить до обертання валу двигуна. Якщо вал двигуна поєднати з робочим органом, то він буде приводити його в рух і виконувати механічну роботу. Зміна вхідної напруги призводить до зміни частоти обертання валу двигуна.

Лабораторне обладнання. Комплект експериментального обладнання А3 та засоби вимірювання – вольтметр, амперметр.

Безпека. Робота з електричними пристроями потребує виконання заходів безпеки. Підключення живлення до експериментального стенду виконується після ретельної перевірки правильності складання схеми та виключно в присутності викладача.

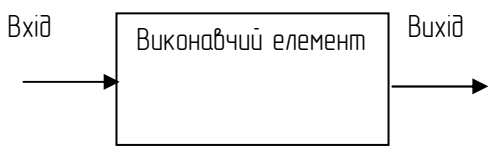


Рис.1.Загальна схема виконавчого елементу

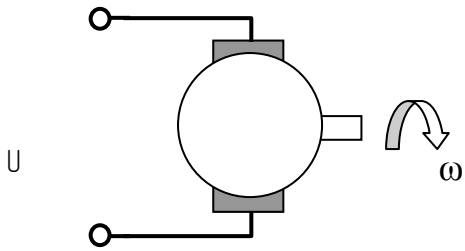


Рис.2.Графічне позначення електричного двигуна постійного струму

Завдання та послідовність виконання роботи

1. Розробити схему визначення статичних характеристик для заданого типу виконавчого елементу.
2. Визначити, яка змінна визначає вхідну дію, та яка змінна визначає вихідний результат.
3. Розробити методику проведення дослідження статичної характеристики
4. Розробити форму таблиці для запису результатів.
5. Підібрати обладнання та скласти стенд відповідно до схеми.
6. Провести дослідження, фіксуючи результат в таблиці.
7. Побудувати статичну характеристику виконавчого елементу.
8. Побудувати математичний опис статичної характеристик.
9. Скласти протокол лабораторної роботи.

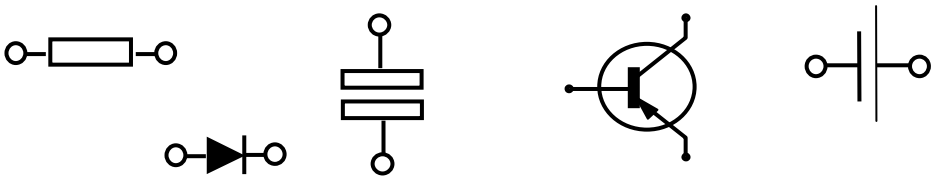
Зміст протоколу. Протокол має містити принципову схему стенду з описом його роботи, опис виконання всіх пунктів плану і висновки за результатами практичної перевірки.

Контрольні запитання. 1. Наведіть приклади виконавчих елементів з іншим принципом дії. 2.Від чого алежить частота обертання ротора двигуна?

6.6. Лабораторна робота. Розроблення, експериментальне дослідження та математичне представлення модулю підсилення

Мета. Ознайомлення з принципом дії модулів підсилення елементів фізично різнорідних систем та їх представленнями в математичних моделях.

Загальні відомості. Зазвичай потужність сигналів керування значно менше потужності, яка потрібна для виконання технологічних операцій. Для узгодження потужностей використовують підсилювачі (електричні, пневматичні, гідравлічні та ін.). Електричні підсилювачі можуть створюватись на основі напівпровідникових пристроїв – транзисторів (рис.1,2). Транзистор містить два напівпровідникові **p-n** переходи: емітерний та колекторний. Електричний опір переходу емітер-колектор залежить від потенціалу бази. Якщо база відключена, то опір переходу емітер-колектор є нескінченним.



а) *б)* *в)* *г)* *д)*

Рис.1. Графічні позначення елементів автоматичних систем. а – електричний опір, б – діод, в- конденсатор, г – транзистор зі структурою p-n-p, д – джерело постійного струму

Лабораторне обладнання. Комплект експериментального обладнання А1, А2 та засоби вимірювання – вольтметр, амперметр.

Безпека. Робота з електричними пристроями потребує виконання заходів безпеки. Підключення живлення до експериментального стенду виконується після ретельної перевірки правильності складання схеми та виключно в присутності викладача.

Завдання 1. 1. Розробити принципову схему модулю транзисторного підсилювача з використанням джерела постійного струму 12В.

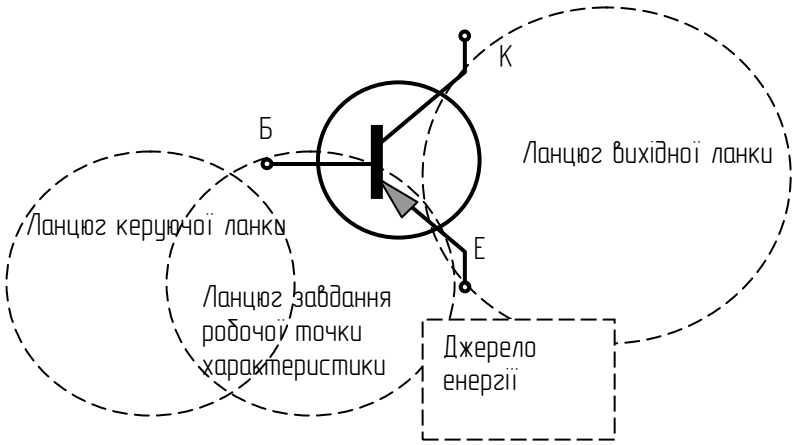


Рис.2. Схема циркуляції струму через транзистор для забезпечення підсилюючої функції

При цьому необхідно забезпечити можливість вимірювань струму та напруги у керуючий та вихідній ланках для налаштування та перевірки його функціонування.

2. Провести розрахунки і визначити параметри елементів.

3. Перевірити робото спроможність підсилювача на стенді.

4. Зробити висновки.

Вимоги до системи. 1. Передбачити можливість налаштування величини керуючого сигналу.

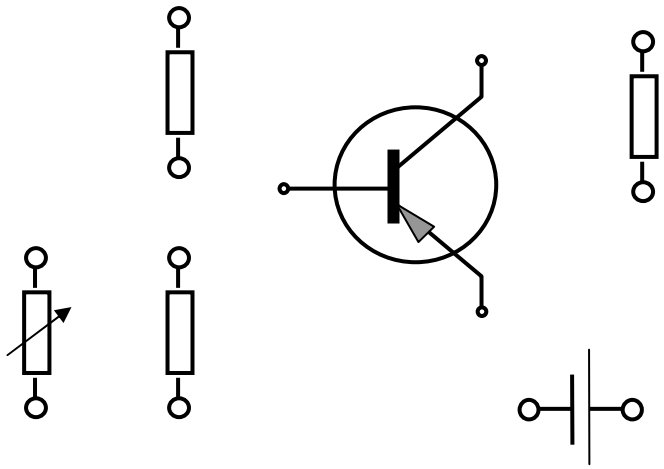


Рис.3. Схема транзисторного підсилювача та схема вимірювань для підтвердження виконання функції підсилення

Хід виконання роботи.

1. Вибрати потрібні елементи модулю на рис 3.
2. Розробити принципову схему модулю (рис.3).
3. Перевірити логіку роботи.
4. Підібрати обладнання та скласти стенд відповідно до схеми.. Живлення **НЕ вмикати!** Запросити викладача.
5. Виконати налаштування та перевірити робото спроможність модулю.
6. Розробити математичне представлення модулю підсилення.
7. Скласти протокол лабораторної роботи.

Зміст протоколу. Протокол має містити принципову схему стенду з описом його роботи, опис виконання всіх пунктів плану і висновки за результатами практичної перевірки.

Контрольні запитання. 1. Поясніть принцип роботи транзистора. 2. Якщо в схемі транзисторного підсилювача е регульований резистор, яке його призначення?

6.7 Лабораторна робота. Розробка, дослідження та представлення алгоритму дії системи керування електричним приводом

Мета. Ознайомлення з принципом дії систем дискретного керування та отримання навичок їх розробки та представленнями в математичних моделях.

Загальні відомості. В технічних об’єктах часто використовують системи ручного керування які забезпечують обертовий рух робочого органу з заданою частотою обертання. Наприклад система керування повітряним вентилятором аеродинамічної труби. При цьому буває необхідно

забезпечувати рух повітря як в прямому так і в зворотному напрямку. Частота обертання вентилятору задана. Така система складається з типових елементів (рис.1).

Лабораторне обладнання. Комплект експериментального обладнання А2, А3 та засоби вимірювання – вольтметр, амперметр.

Безпека. Робота з електричними пристроями потребує виконання заходів безпеки. Підключення живлення до експериментального стенду виконується після ретельної перевірки правильності складання схеми та виключно в присутності викладача.

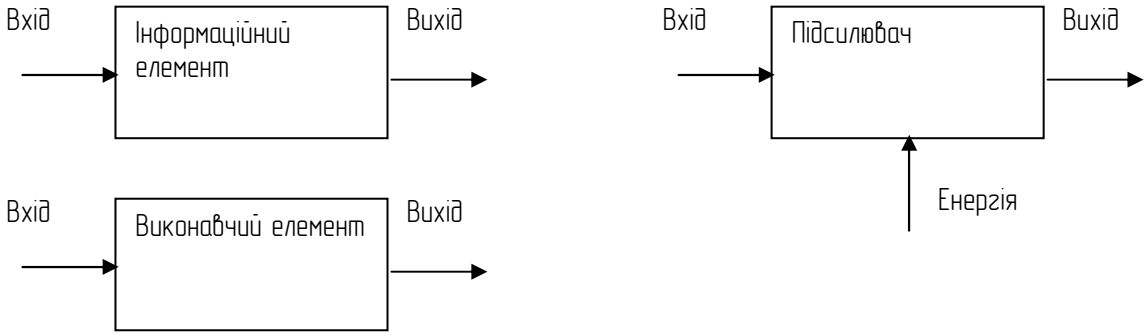


Рис. 1. Комплект типових елементів фізично різнорідної системи керування

Завдання та послідовність виконання роботи

1. Розробити структурну та принципову схеми системи ручного керування обертанням повітряного вентилятора аеродинамічної труби з забезпеченням заданої постійної частоти та можливістю реверсу. Для системи використати релейний підсилювач.
2. Визначити, яка змінна визначає вхідну дію, та яка змінна визначає вихідний результат.
3. Розробити методику проведення дослідження регулюючої характеристики системи (залежності частоти обертання валу електродвигуна від керуючого сигналу).
4. Розробити форму таблиці для запису результатів.
5. Підібрати обладнання та скласти стенд відповідно до схеми.. Живлення **НЕ вмикати!** Запросити викладача.
6. Провести дослідження, фіксуючи результат в таблиці.

7. Побудувати статичну характеристику фізично різнорідної системи.
8. Побудувати математичний опис статичної характеристики.
9. Скласти протокол лабораторної роботи.

Зміст протоколу. Протокол має містити принципову схему стенду з описом його роботи, опис виконання всіх пунктів плану і висновки за результатами практичної перевірки.

Контрольні запитання. 1. Деталізуйте алгоритм дії системи керування електричним приводом. 2.Поясніть принцип дії електричного двигуна постійного струму.

6.8. Лабораторна робота. Розробка, дослідження та представлення алгоритму дії системи автоматичного дискретного керування електричним приводом

Мета. Ознайомлення з принципом дії систем автоматичних дискретного керування та отримання навичок їх розробки та представленнями в математичних моделях.

Загальні відомості. В технічних об'єктах широко використовуються системи автоматичного керування. Наприклад система автоматичного керування вентилятором охолодження процесора комп'ютера. При цьому необхідно забезпечувати включення або відключення вентилятора при підвищенні температури вище заданої та його відключення при її знижені Частота обертання вентилятору задана. Система складається з типових елементів (рис.1).

Лабораторне обладнання. Комплект експериментального обладнання А2, А3 та засоби вимірювання – вольтметр, амперметр.

Безпека. Роота з електричними пристрями потребує виконання заходів безпеки. Підключення живлення до експериментального стенду виконується після ретельної перевірки правильності складання схеми та виключно в присутності викладача.

Завдання та послідовність виконання роботи

1. Розробити структурну та принципову схеми системи автоматичного керування вентилятором охолодження з забезпеченням заданої постійної частоти обертання.

2. Визначити, яка змінна визначає вхідну дію, та яка змінна визначає вихідний результат.
3. Розробити методику проведення дослідження регулюючої характеристики системи (залежності частоти обертання валу електродвигуна від керуючого сигналу).
4. Розробити форму таблиці для запису результатів.

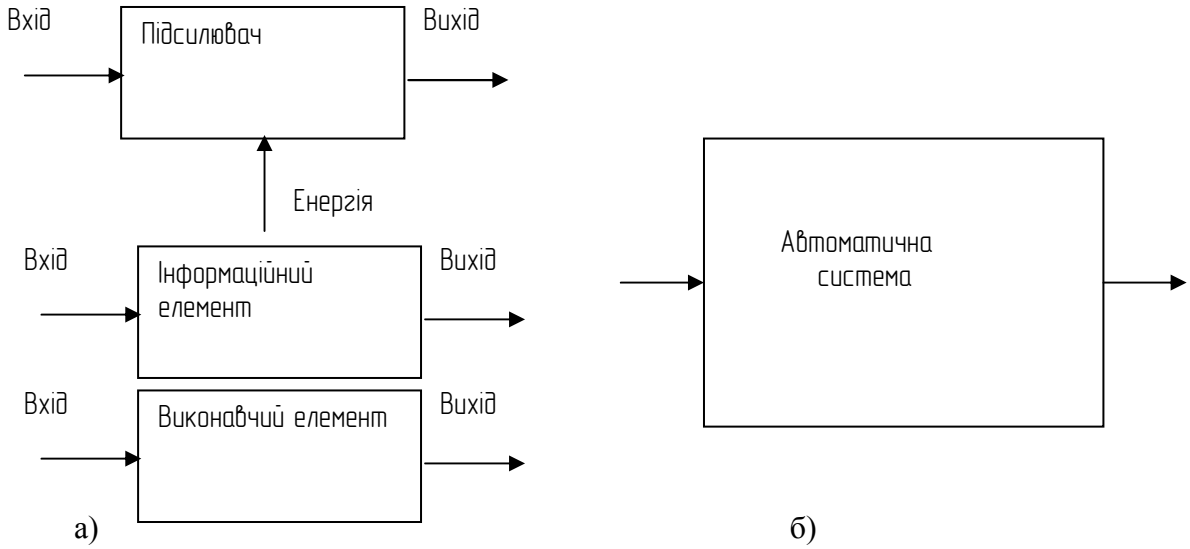


Рис. 1. Представлення типових елементів автоматичних систем (а) та автоматичної системи (б)

5. Підібрати обладнання та скласти стенд відповідно до схеми.. Живлення **НЕ вмикати!** Запросити викладача.
6. Провести дослідження, фіксуючи результат в таблиці.
7. Побудувати статичну характеристику системи.
8. Скласти протокол лабораторної роботи.

Зміст протоколу. Протокол має містити принципову схему стенду з описом його роботи, опис виконання всіх пунктів плану і висновки за результатами практичної перевірки.

Контрольні запитання. 1. Поясніть відмінності між дією системи керування електричним приводом та дією автоматичної системи дискретного керування. 2.Поясніть принцип дії принципової схеми розробленої системи керування.

Список використаних джерел інформації

1. Фото екскаватору. URL:https://zolochiv.net/try-porady-iaki-dopomozhut-vybraty-ekskavator/ (дата звернення: 15.03.2024).
2. Схема гідросистеми екскаватору. URL: https://patents.su/8-1745844-gidroprivod-strely-ehkskavatora.html (дата звернення: 19.03.2024).
3. Фото літака «Антонов-Канада» 74ТК-200. URL: https://www.epravda.com.ua/news/2021/07/26/676259/ (дата звернення: 11.03.2024).
4. Фото літака F-16. URL: https://www.epravda.com.ua/news/2021/07/26/676259/ (дата звернення: 19.03.2024).
5. Ю.Ф. Лазарев МатЛаб 5.х - :К.:Видавнича група BNV, 2000. –384 с. (Серія «Бібліотека студента») ISBN 566-522-06-7, ISBN 5-7315-0096-7.
6. Рішення диференціальних рівнянь чисельними методами в середовищі Matlab. URL: http://um.co.ua/6/6-7/6-7923.html (дата звернення: 18.03.2024).
7. MATLAB Help Center https://www.mathworks.com/help/matlab/functions.html
8. Путівник мовою програмування Python http://pythonguide.rozh2sch.org.ua/
9. Робота з графікою в Python: від основ до просунутих технік https://foxminded.ua/robota-z-hrafikoiu-python/

Рекомендована література

1. Яхно О.М., Узунов О.В., Луговський О.Ф., Ковалев В.А., Мовчанюк А.В., Коц І.І., Губарев О.П. Прикладна гідроаеромеханіка і механотроніка. Вінниця: ВНТУ, 2017. – 712с. id: 1683
2. Попович М.Г., Ковальчук О.В. Теорія автоматичного керування: Підручник. – 2-ге вид. перероб. І доп. – К.: Либідь, 2007. -656 с. ISBN 978-966-06-0447-6. URL:http://pdf.lib.vntu.edu.ua/books/Popovich_2007_656.pdf (дата звернення: 18.03.2024).
3. Ю.Ф. Лазарев МатЛаб 5.х - :К.:Видавнича група BNV, 2000. – 384 с.

(Серія «Бібліотека студента») ISBN 566-522-06-7, ISBN 5-7315-0096-7.
4. Назаренко І.І., Бернік І.М. Основи проектування і конструювання машин та обладнання виробництв. Навчальний посібник для вищих навчальних закладів. Видавництво «Аргар Медіа Груп». -К.:- 2013, - 544.
5. Томашевський, В.М. Моделювання систем/ В.М. Томашевський. - К. : Видавнича група BHV,2005.- 352с.
6. Струтинський, В.Б. Математичне моделювання процесів та систем механіки: Підручник/В.Б. Струтинський. - Житомир: ЖШТШ, 2001.- 612с.

7. ДОДАТКИ

В додатках А, Б та В наведено приклади вирішення задач автоматизації інженерних розрахунків та моделювання. Наведені програми були розроблені студентами в ході виконання розрахунково графічних робіт після прослуховування курсу лекцій з Основ математичного моделювання фізично різнорідних систем та виконання лабораторних робіт.

ДОДАТОК А
(довідковий)

РОЗРАХУНОК ПЛОЩІ ПОПЕРЕЧНОГО ПЕРЕРІЗУ, ОБЄМУ, МАСИ ТА ВАРТОСТІ СТАНДАРТНОГО ТА НЕСТАНДАРТНОГО ПРОФІЛІВ СОРТАМЕНТУ
(розробник програми Шевель Андрій, Гр.МА-02)

Призначення програми

Програма призначена для автоматизації розрахунків пов’язаних з визначенням площі поперечного перерізу, об’єму, маси та ціни заготовок нестандартного та стандартного сортаменту, а саме: труби круглого перерізу, швелера з прямими полками, рівнобокого кутика та двотавра з рівними полками..

Цільова група: студенти та працівники машинобудівних підприємств.

Користь від програми

Програма дозволяє автоматизувати, а відповідно і скоротити час для розрахунку площі поперечного перерізу, об’єму, маси та ціни заготовок нестандартного та стандартного сортаменту.

Визначення площі поперечного перерізу може бути використане у розрахунках на статичну та динамічну міцність заготовок.

Визначення об’єму та маси дозволить зорієнтуватися виробнику у визначенні розмірів упаковки. А також дозволить визначити масу, що може бути використана при транспортуванні виробів чи заготовок.

Визначення ціни заготовок дозволить зорієнтуватися користувача у економічному стані при купівлі сортаменту.

Постановка задачі

Вхідними даними в даній задачі були геометричні розміри прокатного сортаменту, а саме:

- труби круглого перерізу (внутрішній діаметр d, товщина стінки s та довжина ℓ)
- Швелера з прямими полками (висота h1 , товщина стійки s , ширина швелера b , товщина полки t та його довжина ℓ)
- Рівнобокого кутика (висота (є одночасно і шириною) b, товщина стінки t та довжина ℓ)
- Двотавр з прямими полками (висота h1 та товщина ніжки s , ширина полки b та товщина полки t та довжина прокату ℓ)

Отримані значення використовуються для розрахунку площі поперечного перетину, об'єму, маси та ціни цих профілів.

Самі значення розрахунків отримуємо підставивши вхідні дані у розрахункові формули, що будуть приведені далі.

Методика розрахунку

Розрахунок площі поперечного перетину профілів

1. Труба круглого перерізу

Розрахункова схема (див. рис. 1)

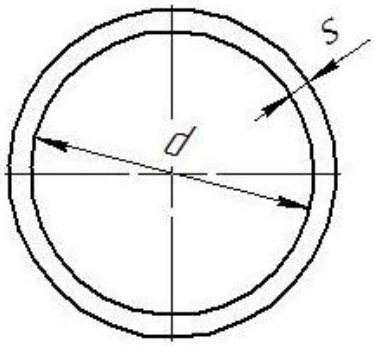


Рис.1. Розрахункова схема труби круглого перерізу

Значення площі поперечного перетину визначаємо за формулою:

$$S = \frac{\pi \cdot \left((d + 2s)^2 - d^2 \right)}{4}$$

2. Швелер з прямими полками

Розрахункова схема (див. рис. 2):

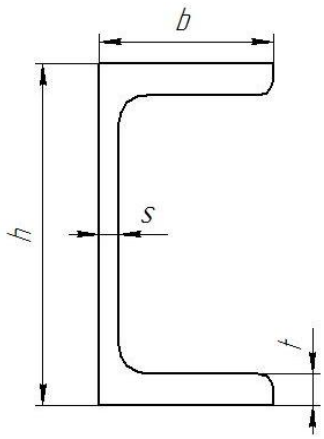


Рис.2. Розрахункова схема швелера з рівними полками

Значення площі поперечного перетину визначаємо за формулою:

$$S = (h1 \cdot s + 2 \cdot (b - s) \cdot t)$$

3. Рівнобокий кутик

Розрахункова схема (див. рис. 3)

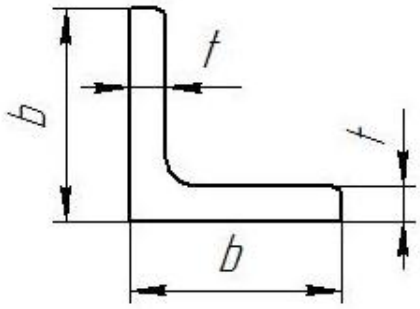


Рис.3. Розрахункова схема рівнобокого кутика

Значення площі поперечного перетину визначаємо за формулою:

$$S = (b \cdot t + (b - t) \cdot t)$$

4. Двотавр з прямими полками

Розрахункова схема (рис. 4)

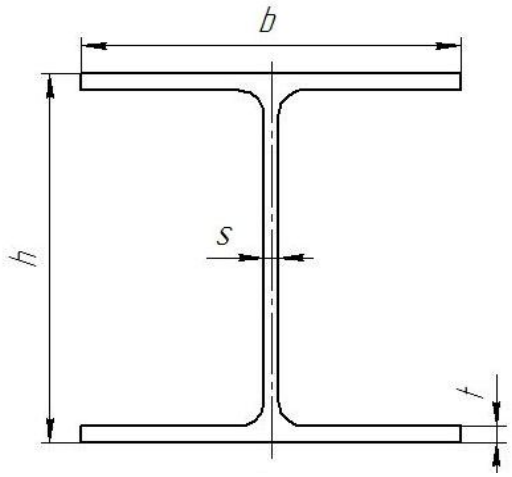


Рис .4. Розрахункова схема двотавра з прямими полками

Значення площі поперечного перетину визначаємо за формулою:

$$S = ((h1 - 2 \cdot t) \cdot s + 2 \cdot b \cdot t)$$

Розрахунок об'єму профілів

Для кожного з випадків об'єм профілів розраховуємо за формулою:

$$V = S \cdot \ell$$

де, ℓ - довжина профілю (заготовки).

Розрахунок маси

Для кожного з випадків маса профілів розраховуємо за формулою:

$$m = \rho \cdot V$$

де, ρ - густина матеріалу з якого виготовлено профіль (вибирається користувачем).

Розрахунок ціни

Для кожного з випадків ціна профілю розраховується за формулою:

$$C = p \cdot \ell$$

де, p - вартість одного погонного метру прокату (задається користувачем).

Наведемо приклад розрахунку маси швелера з прямими полками, що виготовлений із звичайної сталі:

1. Визначаємося з вибором розрахунку (вибираємо розрахунок маси заготовки).
2. Визначаємося з матеріалом з якого виготовлений профіль (вибираємо сталь звичайна)
3. Визначаємося з профілем заготовки (вибираємо швелер з прямими полками)
4. Задати геометричні розміри профілю:

4.1 Задати значення ширини швелера b;

4.2 Задати значення товщини полки t;

4.3 Задати значення висоти швелера h;

4.4 Задати значення товщини стійки швелера s;

4.5 Задати значення довжини профілю ℓ;

4.6 Розрахувати значення маси профілю за формулою:

$$m = \rho \cdot (h1 \cdot s + 2 \cdot (b - s) \cdot t) \cdot \ell$$

5. Вивести значення маси на екран.

Для інших варіантів розрахунків і профілів схема розрахунку буде подібною.

Алгоритм розрахунку

Наведемо алгоритм розрахунку для методики представленої у пункті 4.

Алгоритм розрахунку див. рис. 5.

В наведеному алгоритмі блок вибору типу розрахунку, матеріалу та профілю є складним. Він дозволяє змінювати як введені дані так і умови та типи розрахунків.

Код програми розрахунку площі поперечного перерізу, об'єму, маси та вартості стандартного та нестандартного профілів сортаменту

```
while i<3
Вибір типу розрахунку:
z=menu ('Вибір розрахунку','Розрахунок площі
поперечного перерізу заготовки','Розрахунок об"єму
заготовки','Розрахунок маси заготовки','Розрахунок
ціни заготовки','Вихід')
switch z
Розрахунок площі поперечного перетину профілів:
    case 1
        k5=menu ('Виберіть профіль','Труба
кругла','Швелер з прямими полками','Кутик
рівнобокий','Двотавр з прямими полками')
```

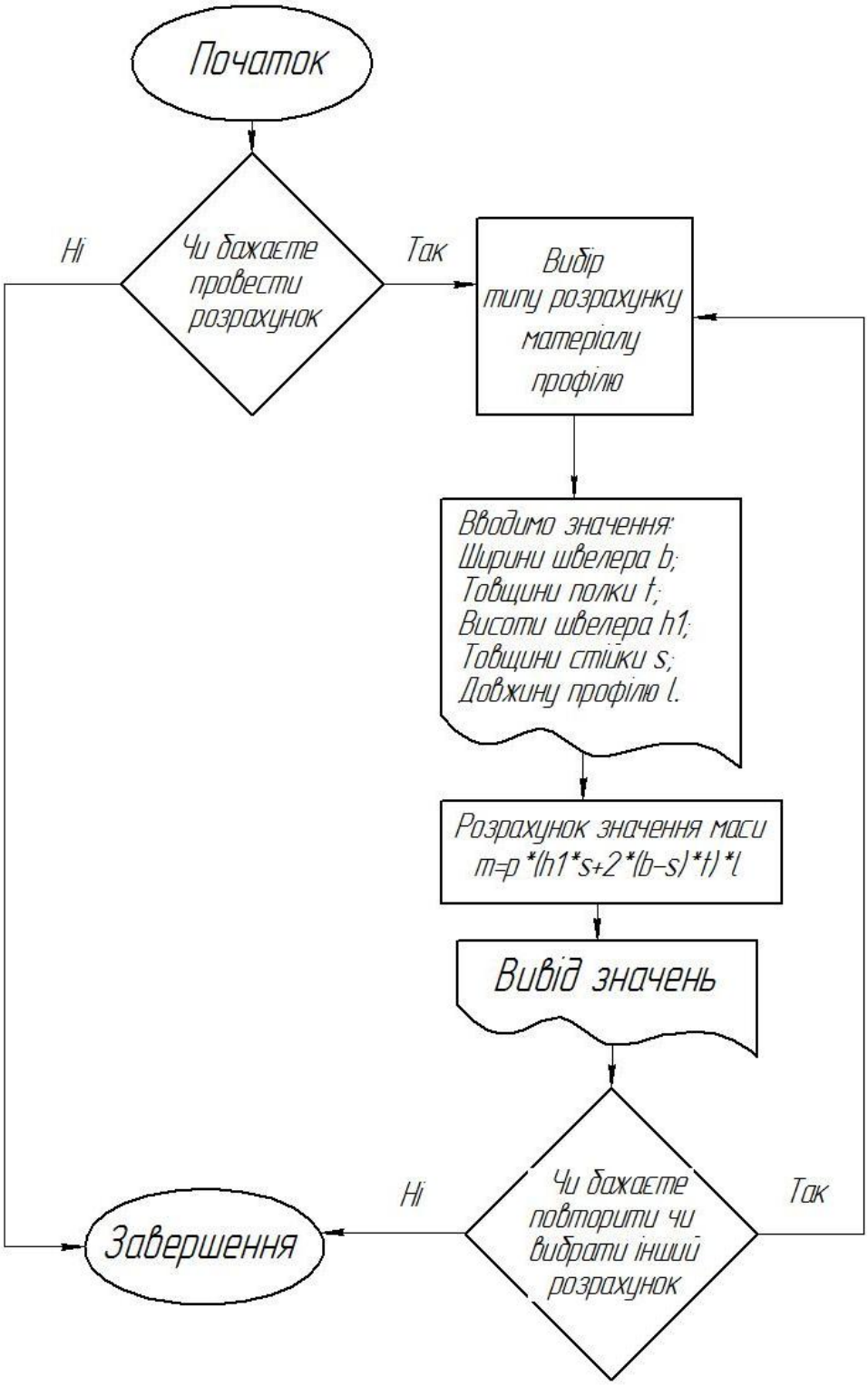


Рис.5. Схема алгоритму розрахунку площ поперечного перетину, об'єму, маси та ціни стандартного та нестандартного сортаменту

```
    switch k5
```

Розрахунок площі поперечного перетину труби круглого перерізу:

```
                case 1
```

```
                delete (gca)
                subplot (1,1,1)
                axis ('off')
        disp ('Введіть значення внутрішнього діаметра
    труби в мм')
        d=input ('d=')
        disp ('Введіть товщину стінки в мм')
        s=input ('s=')
          S=(3.14*((d+2*s)^2-d^2)/4)/100
        disp ('Площа поперечного перерізу труби кв.см')
     h=text (0,1,'Розрахунок площі поперечного перетину
труби','FontSize',15)
      h=text (0.1,0.5,'Внутрішній діаметр в
мм','FontSize',12)
      d1=num2str (d)
      h=text (1,0.5,d1,'FontSize',12)
      h=text (0.1,0.45,'Товщина стінки в
мм','FontSize',12)
      s1=num2str (s)
      h=text (1,0.45,s1,'FontSize',12)
      h=text (0.1,0.40,'Площа поперечного перетину
профілю в кв см','FontSize',12)
      S1=num2str (S)
      h=text (1,0.40,S1,'FontSize',12)
```

Розрахунок площі поперечного перетину швелера з прямими полками:

```
                case 2
               delete (gca)
               subplot (1,1,1)
               axis ('off')
        disp ('Введіть значення ширини швелера в мм')
        b=input ('b=')
        disp ('Введіть товщину полки в мм')
        t=input ('t=')
        disp ('Введіть висоту швелера в мм')
        h=input ('h=')
        disp ('Введіть товщину стійки в мм')
        s=input ('s=')
           S=(h*s+2*((b-s)*t))/100
        disp ('Площа поперечного перерізу швелера кв.см')
h=text (-0.1,1,'Розрахунок площі поперечного перерізу
швелера','FontSize',15)
       h=text (0.15,0.9,' з прямими
полками','FontSize',15)
```

```
   h=text (0.1,0.5,'Ширина швелера в мм','FontSize',12)
      b1=num2str (b)
      h=text (1,0.5,b1,'FontSize',12)
      h=text (0.1,0.45,'Товщина полки в
мм','FontSize',12)
      t1=num2str (t)
      h=text (1,0.45,t1,'FontSize',12)
      h=text (0.1,0.40,'Висота швелера в
мм','FontSize',12)
      h2=num2str (h1)
      h=text (1,0.40,h2,'FontSize',12)
      h=text (0.1,0.35,'Товщина стійки в
мм','FontSize',12)
      s1=num2str (s)
      h=text (1,0.35,s1,'FontSize',12)
       h=text (0.1,0.30,'Площа поперечного перерізу
профілю в кв см','FontSize',12)
      S1=num2str (S)
      h=text (1,0.30,S1,'FontSize',12)
```

Розрахунок площі поперечного перетину рівнобокого кутика:

```
                case 3
               delete (gca)
               subplot (1,1,1)
               axis ('off')
       disp ('Введіть значення ширини кутика в мм')
       b=input ('b=')
       disp ('Введіть товщину полки в мм')
       t=input ('t=')
             S=(b*t+(b-t)*t)/100
       disp ('Площа поперечного перерізу кутика кв.см')
    h=text (0.1,1,'Розрахунок площі поперечного
перетину','FontSize',15)
    h=text (0.3,0.9,' рівнобокого кутика','FontSize',15)
   h=text (0.1,0.5,'Ширина кутика в мм','FontSize',12)
      b1=num2str (b)
      h=text (1,0.5,b1,'FontSize',12)
      h=text (0.1,0.45,'Товщина полки в
мм','FontSize',12)
      t1=num2str (t)
      h=text (1,0.45,t1,'FontSize',12)
        h=text (0.1,0.40,'Площа поперечного перерізу
профілю в кв см','FontSize',12)
      S1=num2str (S)
```

```
      h=text (1,0.40,S1,'FontSize',12)
```

Розрахунок площі поперечного перетину двотавра з прямими полками:

```
                 case 4
                delete (gca)
                subplot (1,1,1)
                axis ('off')
        disp ('Введіть значення ширини двотавра в мм')
        b=input ('b=')
        disp ('Введіть товщину полки в мм')
        t=input ('t=')
        disp ('Введіть висоту двотавра в мм')
        h=input ('h=')
        disp ('Введіть товщину стійки в мм')
        s=input ('s=')
           S=(b*t*2+(h-2*t)*s)/100
        disp ('Площа поперечного перерізу двотавра
    кв.см')
        h=text (-0.1,1,'Розрахунок площі поперечного
  перерізу двотавра','FontSize',15)
       h=text (0.15,0.9,' з прямими
полками','FontSize',15)
   h=text (0.1,0.5,'Ширина двотавра в мм','FontSize',12)
      b1=num2str (b)
      h=text (1,0.5,b1,'FontSize',12)
      h=text (0.1,0.45,'Товщина полки в
мм','FontSize',12)
      t1=num2str (t)
      h=text (1,0.45,t1,'FontSize',12)
      h=text (0.1,0.40,'Висота двотавра в
мм','FontSize',12)
      h2=num2str (h1)
      h=text (1,0.40,h2,'FontSize',12)
      h=text (0.1,0.35,'Товщина стійки в
мм','FontSize',12)
      s1=num2str (s)
      h=text (1,0.35,s1,'FontSize',12)
       h=text (0.1,0.30,'Площа поперечного перерізу
профілю в кв см','FontSize',12)
      S1=num2str (S)
      h=text (1,0.30,S1,'FontSize',12)
            end
```

Розрахунок об’єму профілів:

```
        case 2
```

```
            k=menu ('Виберіть профіль','Труба
    кругла','Швелер з прямими полками','Кутик
    рівнобокий','Двотавр з прямими полками')
    switch k
```
Розрахунок об’єму труби круглого перерізу:
```
        case 1
              delete (gca)
              subplot (1,1,1)
              axis ('off')
       disp ('Введіть значення внутрішнього діаметра
    труби в мм')
       d=input ('d=')
       disp ('Введіть товщину стінки в мм')
       s=input ('s=')
       disp ('Введіть довжину труби в мм')
       l=input ('l=')
       V=(3.14*((d+2*s)^2-d^2)*l/4)/1000
       disp ('Об"єм труби куб.см')
      h=text (0,1,'Розрахунок об"єму
труби','FontSize',15)
      h=text (0.1,0.5,'Внутрішній діаметр в
мм','FontSize',12)
      d1=num2str (d)
      h=text (0.8,0.5,d1,'FontSize',12)
      h=text (0.1,0.45,'Товщина стінки в
мм','FontSize',12)
      s1=num2str (s)
      h=text (0.8,0.45,s1,'FontSize',12)
      h=text (0.1,0.40,'Довжина профіля в
мм','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.40,l1,'FontSize',12)
           h=text (0.1,0.35,'Об"єм профілю в куб
см','FontSize',12)
      V1=num2str (V)
      h=text (0.8,0.35,V1,'FontSize',12)
```
Розрахунок об’єму швелера з прямими полками:
```
        case 2
              delete (gca)
              subplot (1,1,1)
              axis ('off')
       disp ('Введіть значення ширини швелера в мм')
       b=input ('b=')
       disp ('Введіть товщину полки в мм')
```

```
        t=input ('t=')
        disp ('Введіть висоту швелера в мм')
        h=input ('h=')
        disp ('Введіть товщину стійки в мм')
        s=input ('s=')
        disp ('Введіть довжину швелера в мм')
        l=input ('l=')
        V=(h*s+2*((b-s)*t))*l/1000
        disp ('Об"єм швелера куб.см')
    h=text (0,1,'Розрахунок об"єму швелера з прямими
полками','FontSize',15)
      h=text (0.1,0.5,'Ширина швелера в
мм','FontSize',12)
      b1=num2str (b)
      h=text (0.8,0.5,b1,'FontSize',12)
      h=text (0.1,0.45,'Товщина полки в
мм','FontSize',12)
      t1=num2str (t)
      h=text (0.8,0.45,t1,'FontSize',12)
      h=text (0.1,0.40,'Висота швелера в
мм','FontSize',12)
      h2=num2str (h1)
      h=text (0.8,0.40,h2,'FontSize',12)
      h=text (0.1,0.35,'Товщина стійки в
мм','FontSize',12)
      s1=num2str (s)
      h=text (0.8,0.35,s1,'FontSize',12)
        h=text (0.1,0.30,'Довжина профіля в
мм','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.30,l1,'FontSize',12)
      h=text (0.1,0.25,'Об"єм профілю в куб
см','FontSize',12)
      V1=num2str (V)
      h=text (0.8,0.25,V1,'FontSize',12)
```

Розрахунок об’єму рівнобокого кутика:

```
        case 3
              delete (gca)
              subplot (1,1,1)
              axis ('off')
       disp ('Введіть значення ширини кутика в мм')
       b=input ('b=')
       disp ('Введіть товщину полки в мм')
       t=input ('t=')
```

```
          disp ('Введіть довжину кутика в мм')
        l=input ('l=')
        V=(b*t+(b-t)*t)*l/1000
        disp ('Об"єм кутика куб.см')
    h=text (0,1,'Розрахунок об"єму рівнобокого
кутика','FontSize',15)
      h=text (0.1,0.5,'Ширина кутика в
мм','FontSize',12)
      b1=num2str (b)
      h=text (0.8,0.5,b1,'FontSize',12)
      h=text (0.1,0.45,'Товщина полки в
мм','FontSize',12)
      t1=num2str (t)
      h=text (0.8,0.45,t1,'FontSize',12)
         h=text (0.1,0.40,'Довжина профіля в
мм','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.40,l1,'FontSize',12)
      h=text (0.1,0.35,'Об"єм профілю в куб
см','FontSize',12)
      V1=num2str (V)
      h=text (0.8,0.35,V1,'FontSize',12)
```

Розрахунок об’єму двотавра з прямими полками:

```
        case 4
              delete (gca)
              subplot (1,1,1)
              axis ('off')
       disp ('Введіть значення ширини двотавра в мм')
       b=input ('b=')
       disp ('Введіть товщину полки в мм')
       t=input ('t=')
       disp ('Введіть висоту двотавра в мм')
       h=input ('h=')
       disp ('Введіть товщину стійки в мм')
       s=input ('s=')
       disp ('Введіть довжину двотавра в мм')
       l=input ('l=');
       V=(b*t*2+(h-2*t)*s)*l/1000
       disp ('Об"єм двотавра куб.см')
    h=text (0,1,'Розрахунок об"єму двотавра з прямими
полками','FontSize',15)
      h=text (0.1,0.5,'Ширина двотавра в
мм','FontSize',12)
      b1=num2str (b)
```

```
      h=text (0.8,0.5,b1,'FontSize',12)
      h=text (0.1,0.45,'Товщина полки в
мм','FontSize',12)
      t1=num2str (t)
      h=text (0.8,0.45,t1,'FontSize',12)
      h=text (0.1,0.40,'Висота двотавра в
мм','FontSize',12)
      h2=num2str (h1)
      h=text (0.8,0.40,h2,'FontSize',12)
      h=text (0.1,0.35,'Товщина стійки в
мм','FontSize',12)
      s1=num2str (s)
      h=text (0.8,0.35,s1,'FontSize',12)
        h=text (0.1,0.30,'Довжина профіля в
мм','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.30,l1,'FontSize',12)
      h=text (0.1,0.25,'Об"єм профілю в куб
см','FontSize',12)
      V1=num2str (V)
      h=text (0.8,0.25,V1,'FontSize',12)
    end
```

Розрахунок маси профілів:

```
        case 3
```

Вибір матеріалу для розрахунку маси:

```
            k1=menu ('Виберіть матеріал','Сталь
    звичайна','Сталь нержавіюча','Алюміній')
            switch k1
                case 1
                    Ro=7.85
                case 2
                    Ro=7.7
                case 3
                    Ro=2.7
            end
            k2=menu ('Виберіть профіль','Труба
    кругла','Швелер з прямими полками','Кутик
    рівнобокий','Двотавр з прямими полками')
    switch k2
```

Розрахунок маси труби круглого перерізу:

```
        case 1
              delete (gca)
              subplot (1,1,1)
              axis ('off')
```

```
        disp ('Введіть значення внутрішнього діаметра
    труби в мм')
        d=input ('d=')
        disp ('Введіть товщину стінки в мм')
        s=input ('s=')
        disp ('Введіть довжину труби в мм')
        l=input ('l=')
        m=(3.14*((d+2*s)^2-d^2)*l*Ro/4)/(1000*1000)
        disp ('Маса труби кг')
     h=text (0,1,'Розрахунок маси труби','FontSize',15)
      h=text (0.1,0.5,'Внутрішній діаметр в
мм','FontSize',12)
      d1=num2str (d)
      h=text (0.8,0.5,d1,'FontSize',12)
      h=text (0.1,0.45,'Товщина стінки в
мм','FontSize',12)
      s1=num2str (s)
      h=text (0.8,0.45,s1,'FontSize',12)
      h=text (0.1,0.40,'Довжина профіля в
мм','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.40,l1,'FontSize',12)
      h=text (0.1,0.35,'Густина матеріалу в г/куб см
','FontSize',12)
      Ro1=num2str (Ro)
      h=text (0.8,0.35,Ro1,'FontSize',12)
      h=text (0.1,0.30,'Маса профілю в
кг','FontSize',12)
      m1=num2str (m)
      h=text (0.8,0.30,m1,'FontSize',12)
```

Розрахунок маси швелера з прямими полками:

```
       case 2
              delete (gca)
              subplot (1,1,1)
              axis ('off')
        disp ('Введіть значення ширини швелера в мм')
        b=input ('b=')
        disp ('Введіть товщину полки в мм')
        t=input ('t=')
        disp ('Введіть висоту швелера в мм')
        h=input ('h=')
        disp ('Введіть товщину стійки в мм')
        s=input ('s=')
        disp ('Введіть довжину швелера в мм')
```

```
        l=input ('l=')
        m=(h*s+2*((b-s)*t))*l*Ro/(1000*1000)
        disp ('Маса швелера кг')
     h=text (0,1,'Розрахунок маси швелера з прямими
полками','FontSize',15)
      h=text (0.1,0.5,'Ширина швелера в
мм','FontSize',12)
      b1=num2str (b)
      h=text (0.8,0.5,b1,'FontSize',12)
      h=text (0.1,0.45,'Товщина полки в
мм','FontSize',12)
      t1=num2str (t)
      h=text (0.8,0.45,t1,'FontSize',12)
      h=text (0.1,0.40,'Висота швелера в
мм','FontSize',12)
      h2=num2str (h1)
      h=text (0.8,0.40,h2,'FontSize',12)
      h=text (0.1,0.35,'Товщина стійки в
мм','FontSize',12)
      s1=num2str (s)
      h=text (0.8,0.35,s1,'FontSize',12)
        h=text (0.1,0.30,'Довжина профіля в
мм','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.30,l1,'FontSize',12)
      h=text (0.1,0.25,'Густина матеріалу в г/куб см
','FontSize',12)
      Ro1=num2str (Ro)
      h=text (0.8,0.25,Ro1,'FontSize',12)
      h=text (0.1,0.20,'Маса профілю в
кг','FontSize',12)
      m1=num2str (m)
      h=text (0.8,0.20,m1,'FontSize',12)
```

Розрахунок маси рівнобокого кутика:

```
        case 3
              delete (gca)
              subplot (1,1,1)
              axis ('off')
       disp ('Введіть значення ширини кутика в мм')
       b=input ('b=')
       disp ('Введіть товщину полки в мм')
       t=input ('t=')
          disp ('Введіть довжину кутика в мм')
       l=input ('l=')
```

```
        m=(b*t+(b-t)*t)*l*Ro/(1000*1000)
        disp ('Маса кутика кг')
       h=text (0,1,'Розрахунок маси рівнобокого
кутика','FontSize',15)
      h=text (0.1,0.5,'Ширина кутика в
мм','FontSize',12)
      b1=num2str (b)
      h=text (0.8,0.5,b1,'FontSize',12)
      h=text (0.1,0.45,'Товщина полки в
мм','FontSize',12)
      t1=num2str (t)
      h=text (0.8,0.45,t1,'FontSize',12)
         h=text (0.1,0.40,'Довжина профіля в
мм','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.40,l1,'FontSize',12)
      h=text (0.1,0.35,'Густина матеріалу в г/куб см
','FontSize',12)
      Ro1=num2str (Ro)
      h=text (0.8,0.35,Ro1,'FontSize',12)
      h=text (0.1,0.30,'Маса профілю в
кг','FontSize',12)
      m1=num2str (m)
      h=text (0.8,0.30,m1,'FontSize',12)
```

Розрахунок маси двотавра з прямими полками:

```
        case 4
              delete (gca)
              subplot (1,1,1)
              axis ('off')
       disp ('Введіть значення ширини двотавра в мм')
       b=input ('b=')
       disp ('Введіть товщину полки в мм')
       t=input ('t=')
       disp ('Введіть висоту двотавра в мм')
       h=input ('h=')
       disp ('Введіть товщину стійки в мм')
       s=input ('s=')
       disp ('Введіть довжину двотавра в мм')
       l=input ('l=')
       m=(b*t*2+(h-2*t)*s)*l*Ro/(1000*1000)
        disp ('Маса двотавра кг')
     h=text (0,1,'Розрахунок маси двотавра з прямими
полками','FontSize',15)
```

```
      h=text (0.1,0.5,'Ширина двотавра в
мм','FontSize',12)
      b1=num2str (b)
      h=text (0.8,0.5,b1,'FontSize',12)
      h=text (0.1,0.45,'Товщина полки в
мм','FontSize',12)
      t1=num2str (t)
      h=text (0.8,0.45,t1,'FontSize',12)
      h=text (0.1,0.40,'Висота двотавра в
мм','FontSize',12)
      h2=num2str (h1)
      h=text (0.8,0.40,h2,'FontSize',12)
      h=text (0.1,0.35,'Товщина стійки в
мм','FontSize',12)
      s1=num2str (s)
      h=text (0.8,0.35,s1,'FontSize',12)
        h=text (0.1,0.30,'Довжина профіля в
мм','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.30,l1,'FontSize',12)
      h=text (0.1,0.25,'Густина матеріалу в г/куб см
','FontSize',12)
      Ro1=num2str (Ro)
      h=text (0.8,0.25,Ro1,'FontSize',12)
      h=text (0.1,0.20,'Маса профілю в
кг','FontSize',12)
      m1=num2str (m)
      h=text (0.8,0.20,m1,'FontSize',12)
        end
```

Розрахунок ціни прокату:

```
        case 4
            k3=menu ('Виберіть профіль','Труба
    кругла','Швелер з прямими полками','Кутик
    рівнобокий','Двотавр з прямими полками')
    switch k3
```

Розрахунок ціни труби круглого перерізу:

```
        case 1
              delete (gca)
              subplot (1,1,1)
              axis ('off')
       disp ('Введіть довжину труби в м')
       l=input ('l=')
       disp ('Введіть ціну за погонний метр в грн')
       p=input ('p=')
```

```
        Cina=l*p
        disp ('Ціна сортаменту в грн')
      h=text (0,1,'Розрахунок ціни труби круглого
перерізу','FontSize',15)
      h=text (0.1,0.5,'Довжина профіля в
м','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.5,l1,'FontSize',12)
      h=text (0.1,0.45,'Ціна погонного метра
грн/м','FontSize',12)
      p1=num2str (p)
      h=text (0.8,0.45,p1,'FontSize',12)
      h=text (0.1,0.40,'Ціна профіля в
грн','FontSize',12)
      Cina1=num2str (Cina)
      h=text (0.8,0.40,Cina1,'FontSize',12)
```

Розрахунок ціни швелера з прямими полками:

```
        case 2
              delete (gca)
              subplot (1,1,1)
              axis ('off')
           disp ('Введіть довжину швелера в м')
       l=input ('l=')
       disp ('Введіть ціну за погонний метр в грн')
       p=input ('p=')
       Cina=l*p
       disp ('Ціна сортаменту в грн')
      h=text (0,1,'Розрахунок ціни швелера з прямими
полками','FontSize',15)
      h=text (0.1,0.5,'Довжина профіля в
м','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.5,l1,'FontSize',12)
      h=text (0.1,0.45,'Ціна погонного метра
грн/м','FontSize',12)
      p1=num2str (p)
      h=text (0.8,0.45,p1,'FontSize',12)
      h=text (0.1,0.40,'Ціна профіля в
грн','FontSize',12)
      Cina1=num2str (Cina)
      h=text (0.8,0.40,Cina1,'FontSize',12)
```

Розрахунок ціни рівнобокого кутика:

```
        case 3
              delete (gca)
```

```
                subplot (1,1,1)
                axis ('off')
             disp ('Введіть довжину кутика в м')
        l=input ('l=')
        disp ('Введіть ціну за погонний метр в грн')
        p=input ('p=')
        Cina=l*p
        disp ('Ціна сортаменту в грн')
      h=text (0,1,'Розрахунок ціни рівнобокого
кутика','FontSize',15)
      h=text (0.1,0.5,'Довжина профіля в
м','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.5,l1,'FontSize',12)
      h=text (0.1,0.45,'Ціна погонного метра
грн/м','FontSize',12)
      p1=num2str (p)
      h=text (0.8,0.45,p1,'FontSize',12)
      h=text (0.1,0.40,'Ціна профіля в
грн','FontSize',12)
      Cina1=num2str (Cina)
      h=text (0.8,0.40,Cina1,'FontSize',12)
```

Розрахунок ціни двотавра з прямими полками:

```
        case 4
                delete (gca)
                subplot (1,1,1)
                axis ('off')
             disp ('Введіть довжину двотавра в м')
        l=input ('l=')
        disp ('Введіть ціну за погонний метр в грн')
        p=input ('p=')
        Cina=l*p
        disp ('Ціна сортаменту в грн')
      h=text (0,1,'Розрахунок ціни двотавра з прямими
полками','FontSize',15)
      h=text (0.1,0.5,'Довжина профіля в
м','FontSize',12)
      l1=num2str (l)
      h=text (0.8,0.5,l1,'FontSize',12)
      h=text (0.1,0.45,'Ціна погонного метра
грн/м','FontSize',12)
      p1=num2str (p)
      h=text (0.8,0.45,p1,'FontSize',12)
```

```
      h=text (0.1,0.40,'Ціна профіля в
грн','FontSize',12)
      Cina1=num2str (Cina)
      h=text (0.8,0.40,Cina1,'FontSize',12)
    end
```

Вихід із програми:

```
        case 5
            delete (gcf)
            exit
    end
    end
```

2. Приклад розрахунку

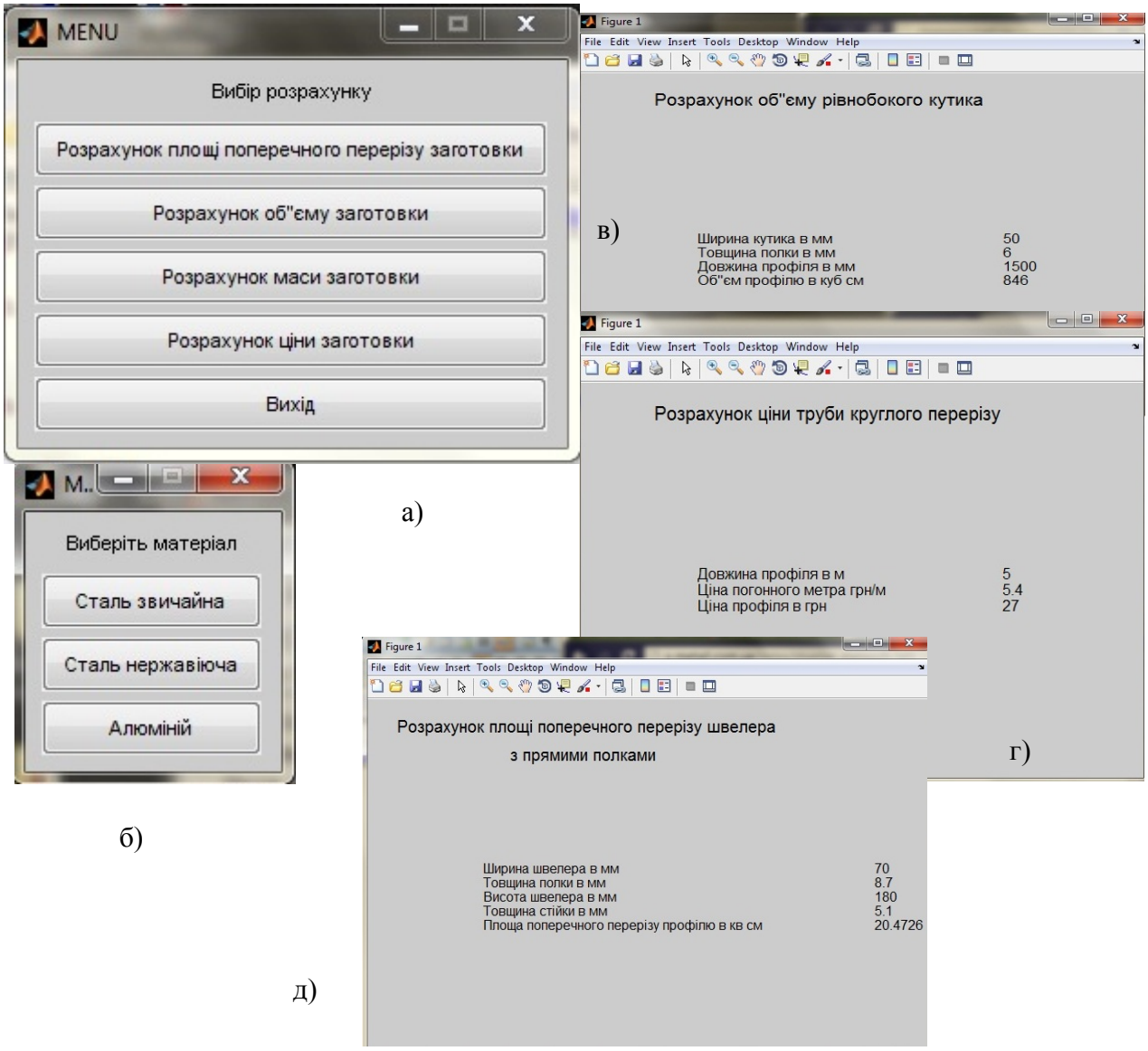


Рис. 6. 1.Вигляд вікон при роботі програми: а- головне вікно; б – вікно, що дозволяє вибрати матеріал; в – вікно з результатами розрахунку об’єму кутика; г – вікно з результатом розрахунку ціни; д – вікно з результатом розрахунку площі поперечного

Висновоки. За допомогою цього курсу, я навчився використовувати програмний пакет MATLAB для автоматизації інженерних розрахунків. Навчився складати на основі заданої методики алгоритми розрахунків, а також розробив програму, яка спрощує виконання розрахунків та дозволяє скоротити часові витрати.

ДОДАТОК Б
(довідковий)

РОЗРОБКА ПРОГРАМИ МОДЕЛЮВАННЯ ПОВЕДІНКИ ПОТОКУ РІДИНИ ПРИ ЗМІНІ ПАРАМЕТРІВ ТРУБОПРОВОДУ ЗА КРИТЕРІЄМ РЕЙНОЛЬДСА

(Розробник програми Глущенко Олег, група МА-02)

Призначення програми

Програма призначена для моделювання поведінки потоку рідини за допомогою автоматичного визначенням числа Рейнольдса.

Цільова група: студенти та працівники машинобудівних підприємств

Користь від програми

Програма дозволяє автоматизувати, а відповідно і скоротити час для розрахунку числа Рейнольдса.Також є можливість побудови графіків залежності числа Рейнольдса від (l) та (d).

Постановка задачі

Вхідними даними в даній задачі були діаметр,довжина прохідного пререрізу,коефіцієнт кін.в‘язкості,густина.

Самі значення розрахунків отримуємо підставивши вхідні дані у розрахункові формули, що будуть приведені далі.

Методика розрахунку

а)За формулою Дарсі

$$\mathrm{Re} = \frac{64}{\lambda}$$

б)За формулою Пуазеля

$$\mathrm{Re} = \frac{128\nu\rho l}{\Pi D^4}$$

в)

$$\mathrm{Re} = \frac{V * D}{\nu}$$

г) Для гнучких трубопроводів

$$\mathrm{Re} = \frac{68}{\lambda}$$

де λ-коефіцієнт кін.в’язкості

1. Визначаємося з вибором розрахунку (вибираємо формулу)
2. Задати геометричні розміри профілю:
3. Вивести значення числана екран.

Код програми моделювання поведінки потоку рідини при зміні параметрів трубопроводу за критерієм Рейнольдса

```
%Очищення оперативної памяті
clear;
%Підготовка текстової інформації для виведення у вікно програми
text1='                                        Самостійна робота                                                ';
text2='        з дисципліни "Основи автоматичного проектування"                ';
text3='                    Виконав студент 3го курсу гр.МА-02 ММІ                              ';
text4='Глущенко Олег';
text5={text1, text2, text3, text4};
m1=0, m2=0,m3=0,m4=0,m5=0;
f1=0,f2=0,f3=0,f4=0;
t1='',t2='', t3='',t4='',t5='',t6='';
%Запуск головного циклу програми
while m1~=6
%Виведення графічного інтерактивного меню на екран
   m1=menu('Виберіть формулу','Re=lamda/64','Re=128*nu*p*l/pi*d^4','Re=v*d/nu','Re=lamda/68','Про програму','Вихід')
%Блок реагування на команди користувача при натискання на кнопки інтерфейсу
%Якщо вибрали 1-у формулу, то
if m1==1
    m2=0;
    f1=0;
%Запуск внутрішнього циклу програми
    while m2~=3
%Блок реагування на команди користувача при натискання на кнопки інтерфейсу внутрішнього циклу
        m2=menu('Re=lamda/64','Ввести lamda','Графік','Головне меню')
        %Якщо вибрали ввести ламда, то
         if m2==1
           lamda1=inputdlg('Lamda','Dani',1,{'200'});
```

```
            Re1=str2num(lamda1{1})/64;
            f1=1;
          end
        %Якщо вибрали Графік, то
          if m2==2
        %Виведення графіку у вікно
             clf;
             hold all
            subplot(2,1,1)
            lamda=0:5:70000;
            Re=lamda/64;
            p1=plot(lamda,Re)
            title('Re=lamda/64');
            xlabel('lamda');
            ylabel('Re');
            grid on
            if f1==1
            subplot(2,1,2)
             axis off
          %Виведення текстової інформації у вікно
t1=text(0,1,strcat('lamda=',lamda1{1}),'fontsize',14);
t2=text(0,0.8,strcat('Re=',num2str(Re1)),'fontsize',14);
%Перевірка значень чисел Рейнольдса та виведення типу
режиму течії
            if (Re1>=2320)
                t3=text(0,0.6,'Turbulentnuy
potik','fontsize',14);
            end
            if (Re1<2320)
                t3=text(0,0.6,'Laminarnuy
potik','fontsize',14);
            end
           else
            warndlg('Введіть дані!');
           end
        end
    end
end
%Якщо вибрали 4-у формулу, то
if m1==4
    m5=0;
    f4=0;
%Запуск внутрішнього циклу програми
    while m5~=3
```

```
%Блок реагування на команди користувача при натискання
на кнопки інтерфейсу внутрішнього циклу
         m5=menu('Re=lamda/68','Ввести
lamda','Графік','Головне меню')
         %Якщо вибрали ввести ламда, то
        if m5==1
             lamda1=inputdlg('Lamda','Dani',1,{'200'});
            Re1=str2num(lamda1{1})/68
            f4=1;
        end
        %Якщо вибрали графік, то
         if m5==2
        %Виведення графіку у вікно
             clf;
            hold all
            subplot(2,1,1)
            lamda=0:5:70000;
            Re=lamda/68;
            p2=plot(lamda,Re)
            title('Re=lamda/68');
            xlabel('lamda');
            ylabel('Re');
            grid on
             if f4==1
             subplot(2,1,2)
              axis off
    %Виведення текстової інформації у вікно
t1=text(0,1,strcat('lamda=',lamda1{1}),'fontsize',14);
t2=text(0,0.8,strcat('Re=',num2str(Re1)),'fontsize',14);
             if (Re1>=2320)
                 t3=text(0,0.6,'Turbulentnuy
potik','fontsize',14);
             end
%Перевірка значень чисел Рейнольдса та виведення типу
режиму течії
             if Re1<2320
                 t3=text(0,0.6,'Laminarnuy
potik','fontsize',14);
             end
             else
            warndlg('Введіть дані!');
             end
         end
    end
```

```
end
%Якщо вибрали 2-у формулу, то
if  m1==2
    m3=0;
    f2=0;
%Запуск внутрішнього циклу програми
    while m3~=4
%Блок реагування на команди користувача при натискання на кнопки інтерфейсу внутрішнього циклу
        m3=menu('Re=128*nu*p*l/pi*d^4','Ввести дані','Графік Re(d)','Графік Re(l)','Головне меню')
        %Якщо вибрали введення данних, то
        if m3==1
data=inputdlg({'nu','p','l','d'},'Dani',1,{'120','1000','20','0.05'});
Re1=128*str2num(data{1})*str2num(data{2})*str2num(data{3})/pi*(str2num(data{4}))^4
            f2=1;
        end
        %якщо вибрали графік Re(d), то
        if m3==2
        %Виведення графіку у вікно
            clf;
             if f2==1
            hold all
            subplot(2,1,1)
            d=0:0.1:1;
Re=128*str2num(data{1})*str2num(data{2})*str2num(data{3})/pi*d.^4;
            p3=plot(d,Re)
            title('Re(d)=128*nu*p*l/pi*d^4');
            xlabel('d');
            ylabel('Re');
            grid on
             subplot(2,1,2)
             axis off
     %Виведення текстової інформації у вікно
t1=text(0,1,strcat('nu=',data{1}),'fontsize',14);
t2=text(0,0.8,strcat('p=',data{2}),'fontsize',14);
t3=text(0,0.6,strcat('l=',data{3}),'fontsize',14);
t4=text(0,0.4,strcat('d=',data{4}),'fontsize',14);
t5=text(0,0.2,strcat('Re=',num2str(Re1)),'fontsize',14);
%Перевірка значень чисел Рейнольдса та виведення типу режиму течії
```

```
if (Re1>=2320)
                t6=text(0,0,'Turbulentnuy
potik','fontsize',14);
            end
            if Re1<2320
                t6=text(0,0,'Laminarnuy
potik','fontsize',14);
            end
            else
         warndlg('Введіть дані!');
            end
        end
        %якщо вибрали графік Re(l), то
         if m3==3
%Виведення графіку у вікно
            clf;
             if f2==1
           hold all
           subplot(2,1,1)
           l=0:1:100;
Re=128*str2num(data{1})*str2num(data{2})*l/pi*(str2num(d
ata{4}))^4;
           p4=plot(l,Re)
           title('Re(l)=128*nu*p*l/pi*d^4');
           xlabel('l');
           ylabel('Re');
           grid on
            subplot(2,1,2)
            axis off
      %Виведення текстової інформації у вікно
t1=text(0,1,strcat('nu=',data{1}),'fontsize',14);
t2=text(0,0.8,strcat('p=',data{2}),'fontsize',14);
t3=text(0,0.6,strcat('l=',data{3}),'fontsize',14);
t4=text(0,0.4,strcat('d=',data{4}),'fontsize',14);
t5=text(0,0.2,strcat('Re=',num2str(Re1)),'fontsize',14);
%Перевірка значень чисел Рейнольдса та виведення типу
режиму течії
if (Re1>=2320)
                t6=text(0,0,'Turbulentnuy
potik','fontsize',14);
            end
 if Re1<2320
                t6=text(0,0,'Laminarnuy
potik','fontsize',14);
```

```
            end
            else
                warndlg('Введіть дані!');
            end
        end
    end
end
%Якщо вибрали 3-у формулу, то
if  m1==3
    m4=0;
    f3=0;
%Запуск внутрішнього циклу програми
    while m4~=4
%Блок реагування на команди користувача при натискання
на кнопки інтерфейсу внутрішнього циклу
        m4=menu('Re=v*d/nu','Ввести дані','Графік
Re(d)','Графік Re(v)','Головне меню')
        %Якщо вибрали ввести дані, то
       if m4==1
data=inputdlg({'v','d','nu'},'Dani',1,{'70','0.05','25.6
'});
Re1=str2num(data{1})*str2num(data{2})/str2num(data{3})
           f3=1;
       end
        %якщо вибрали графік Re(d), то
       if m4==2
%Виведення графіку у вікно
           clf;
           if f3==1
           hold all
           subplot(2,1,1)
           d=0:0.1:2;
           Re=str2num(data{1})*d/str2num(data{3});
           p5=plot(d,Re)
           title('Re(d)=Re=v*d/nu');
           xlabel('d');
           ylabel('Re');
           grid on
            subplot(2,1,2)
             axis off
    %Виведення текстової інформації у вікно
t1=text(0,1,strcat('v=',data{1}),'fontsize',14);
t2=text(0,0.8,strcat('s=',data{2}),'fontsize',14);
```

```
t3=text(0,0.6,strcat('nu=',data{3}),'fontsize',14);
t4=text(0,0.4,strcat('Re=',num2str(Re1)),'fontsize',14);
%Перевірка значень чисел Рейнольдса та виведення типу
режиму течії
if (Re1>=2320)
                t5=text(0,0.2,'Turbulentnuy
potik','fontsize',14);
            end
            if Re1<2320
                t5=text(0,0.2,'Laminarnuy
potik','fontsize',14);
            end
        else
         warndlg('Введіть дані!');
         end
        end
         %якщо вибрали графік Re(v), то
         if m4==3
%Виведення графіку у вікно
            clf;
             if f3==1
           hold all
           subplot(2,1,1)
           v=0:1:1000;
           Re=v*str2num(data{2})/str2num(data{3});
           p6=plot(v,Re)
           title('Re(v)=Re=v*d/nu');
           xlabel('v');
           ylabel('Re');
           grid on
            subplot(2,1,2)
            axis off
    %Виведення текстової інформації у вікно
t1=text(0,1,strcat('v=',data{1}),'fontsize',14);
t2=text(0,0.8,strcat('s=',data{2}),'fontsize',14);
t3=text(0,0.6,strcat('nu=',data{3}),'fontsize',14);
t4=text(0,0.4,strcat('Re=',num2str(Re1)),'fontsize',14);
            if (Re1>=2320)
                t5=text(0,0.2,'Turbulentnuy
potik','fontsize',14);
            end
%Перевірка значень чисел Рейнольдса та виведення типу
режиму течії
            if Re1<2320
```

```
                t5=text(0,0.2,'Laminarnuy
potik','fontsize',14);
            end
              else
                warndlg('Введіть дані!');
            end
        end
        end
end
    if m1==5
        msgbox(text5,'Про програму')
    end
end
```

6.Алгоритм розрахунку

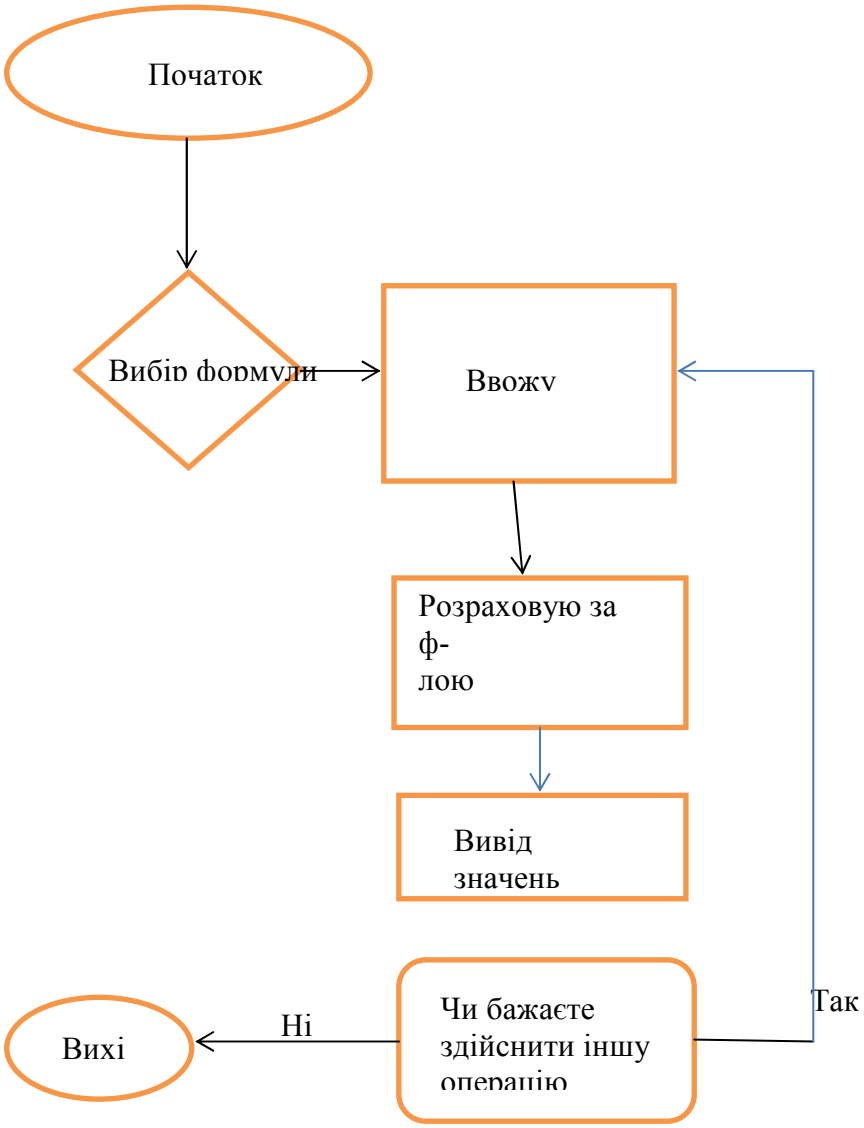
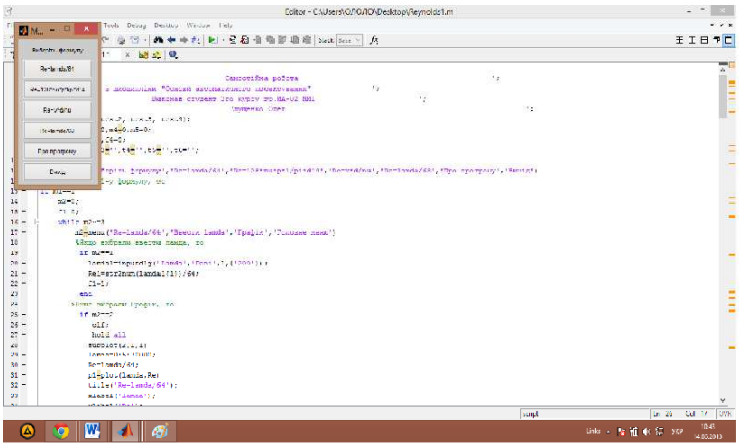
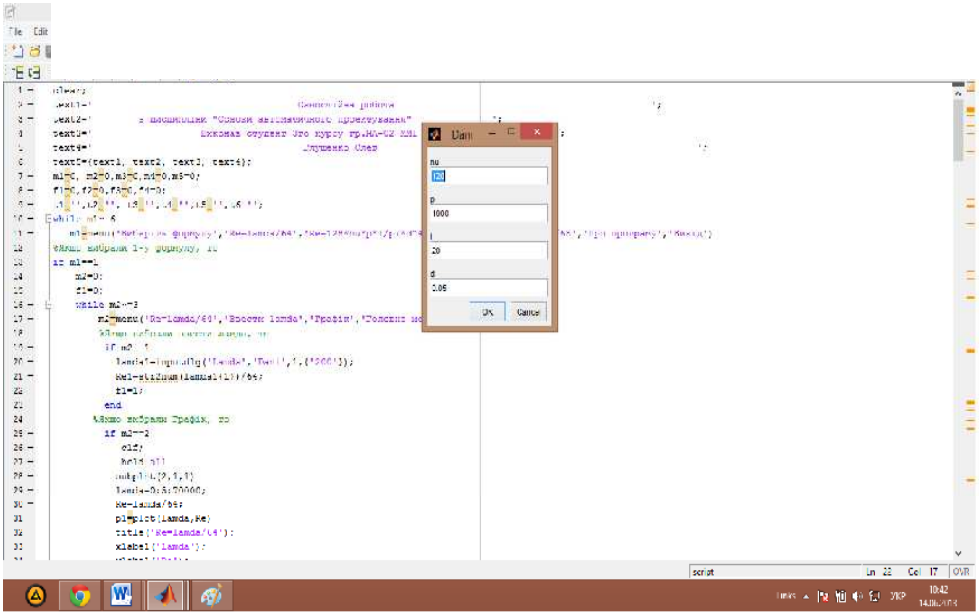


Рис.6.2. Алгоритм моделювання поведінки потоку рідини при зміні параметрів трубопроводу за критерієм Рейнольдса

Приклад розрахунку



а)



б)

Рис.6.3. Вигляд вікон вводу даних для моделювання поведінки потоку: а – меню вибору базової формули; б – меню завдання параметрів потоку.

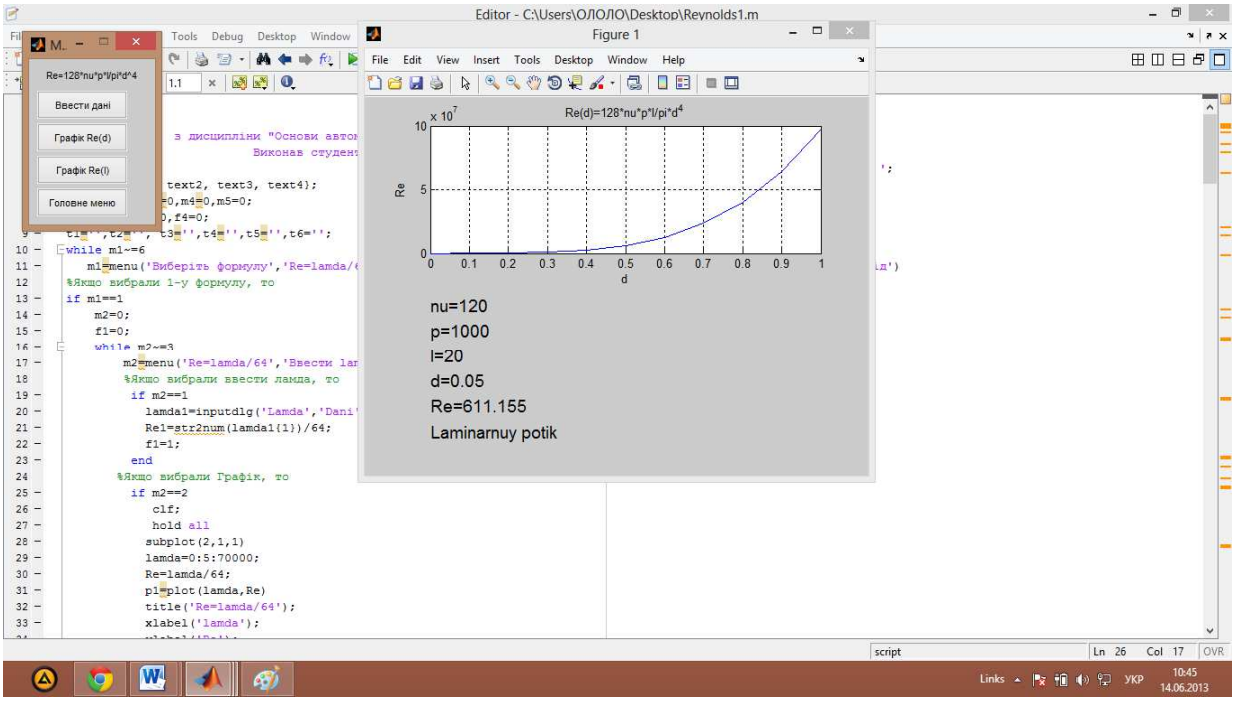


Рис 6.4. Вигляд вікон вводу результату моделювання поведінки потоку

Висновок. При вивченні курсу я навчився використовувати MATLAB у практичний цілях, навчився складати алгоритми розрахунків, а також розробив програму яка спрощує розрахунки числа Рейнольдса.

ДОДАТОК В
(довідковий)

МОДЕЛЮВАННЯ ДІЇ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ВИЗНАЧЕННЯ ОКРУГЛИХ ФОРМ НА ЗАДАНОМУ ЗОБРАЖЕНІ

(розробник програми Клюєв Василь, гр. МА-01)

Призначення програми.

Програма призначена для розпізнання округлих форм на зображенні (бульбашки, кола, отвори та інш.) із подальшим графічним виводом результатів аналізу та необхідного графіку.

Цільова група: наукова гілка спільноти (студенти, лаборанти, вчені, викладачі), а також дослідники та дизайнери.

1.Користь від програми.

Програма має досить широкий спектр застосування, від дизайнерських потреб до дослідження явищ кавітації. Так, людина, що досліджує це явище, маючи декілька зображень, на яких відображено у певні моменти часу кавітаційні бульбашки та їх утворення, зможе робити дослідження у цьому напрямку, завантаживши ці зображення у програму та проаналізувавши їх. Програма автоматично розпізнає та позначить на зображенні бульбашки, їх розміри і побудує графік.

Також програма несе в собі демонстраційні властивості потужності пакету MatLab Image Toolbox, що має користь при початковій стадії опанування цього програмного пакету студентами чи учнями. Це дозволить популяризувати програмування не лише серед студентів інженерних спеціальностей, а також серед інших верств населення.

2.Постановка задачі.

- Вхідними даними є зображення, на якому відображені округлі форми, що мають бути розпізнані автоматично. Критеріями для зображення є: чіткість, розмір не більш як 1600x1800px, округлість правильної форми (коловидної), відсутність відблисків або інших дефектів зображення. Формат зображння: jpg, png.

- Вихідні дані: проаналізоване зображення із нанесеними на ньому діаметрами округлостей і виводом графіку.

3.Методика роботи програми.

Використовуючи потенціал спеціальних операторів та функцій, що входять до пакету MatLab Image Toolbox ми маємо прості та водночас якісні процеси розпізнання форм на зображеннях, що не потребують поглиблених знань у програмуванні, мають початкові оптимізації, що позитивно впливає на швидкодії програми.

4.Алгоритм програми

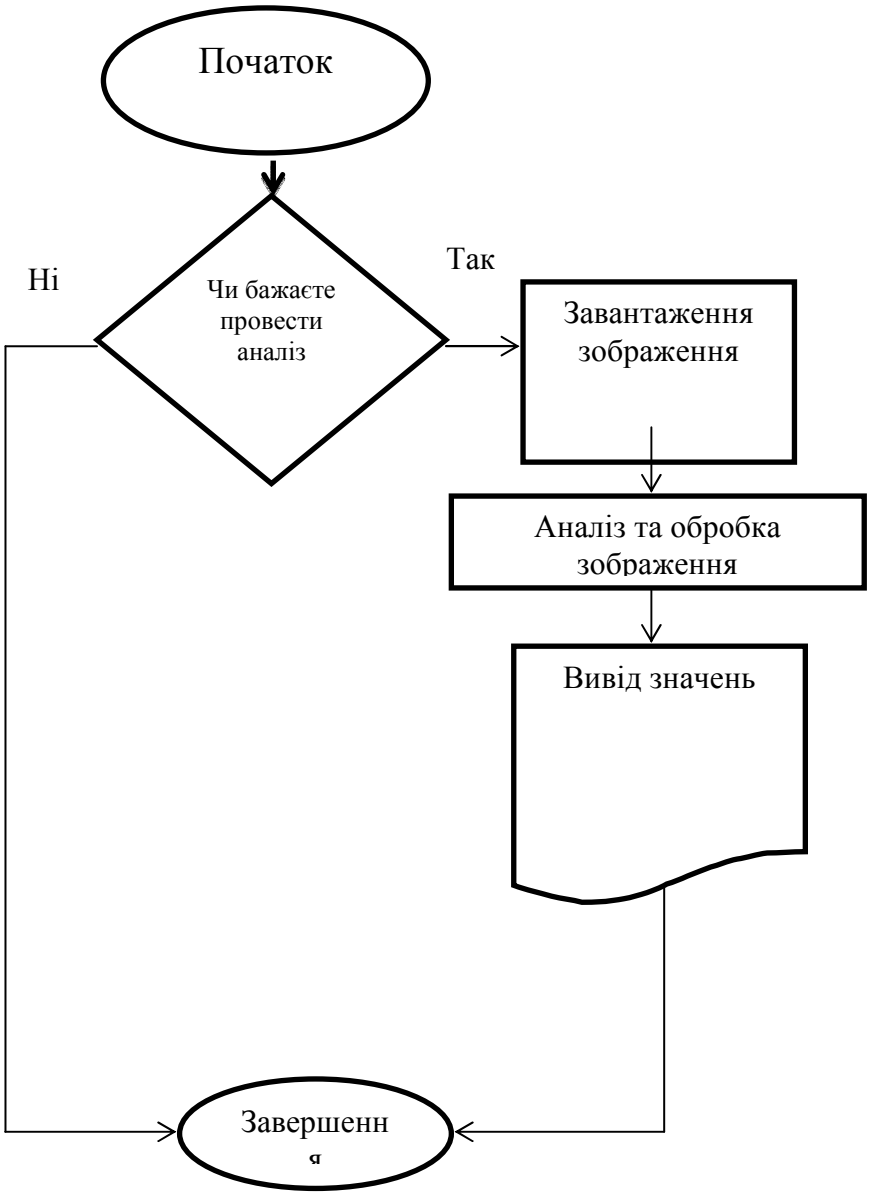


Рис.6.5. Алгоритм моделювання дії штучного інтелекту для визначення округлих форм на заданому зображенні

Код програми моделювання дії штучного інтелекту для визначення округлих форм на заданому зображенні

```
clear all;close all;clc;

img = imread('bubbles.jpg');
%відкриття зображення. У поле введіть свою назву
зображення
grayimg = rgb2gray(img);
%переведння зображення у відтінки сірого
grayimg = imadjust(grayimg);
%контрастування
bw = edge(grayimg,'canny', 0.15, 2);
%виділення меж методом "Канні". Також можливі методи
'sobel', 'prewitt', 'roberts'
bw = imfill(bw,'holes');
%заливання замкнутих областей
se = strel('disk',1);
%формування структурного елементу
bw = imopen(bw,se);
%морфологічне відкриття (позбавлення тонких ліній)

[B,L] = bwboundaries(bw);
%відслідковує зовнішні межі об`єкту та виводить 2
результата. B-структурний
%масив розміром Px1, P-кількість об`єктів на зображенні.
Елементи масиву B
%- матриці розміром Qx2, де Q - кількість пікселів, що
належать границі
%об`єкта. L-матриця, розмір котрої рівен розміру
зображення, маюча %невід`ємні числа, котрі відповідають
замкнутим областям.
stats = regionprops(L,'Centroid','EquivDiameter');
%утворює масив структур stats, поля елементів котрого
Centroid і
%EquivDiametr мають координати центру області і діаметр
області
figure, imshow(img)
hold on
%Виведення результатів
for k = 1:length(B)
 boundary = B{k};
 radius = stats(k).EquivDiameter/2;
 xc = stats(k).Centroid(1);
```

```
 yc = stats(k).Centroid(2);
 theta = 0:0.01:2*pi;
 Xfit = radius*cos(theta) + xc;
 Yfit = radius*sin(theta) + yc;
 plot(Xfit, Yfit, 'g');
 text(boundary(1,2)-15,boundary(1,1)+15,
num2str(radius,3),'Color','y',...
  'FontSize',8);
end
```

%виведення графіку

```
 radius2=zeros (length(B));
 xc2=zeros (length(B));
 yc2=zeros (length(B));
for k = 1:length(B)
 boundary = B{k};
 radius2(k) = stats(k).EquivDiameter/2;
 xc2(k) = stats(k).Centroid(1);
 yc2(k) = stats(k).Centroid(2);
 theta = 0:0.01:2*pi;
 Xfit2 = radius2(k).*cos(theta) + xc2(k);
 Yfit2 = radius2(k).*sin(theta) + yc2(k);


end
 figure;
plot(radius2);
 title('Radius');
```

7. Приклад аналізу

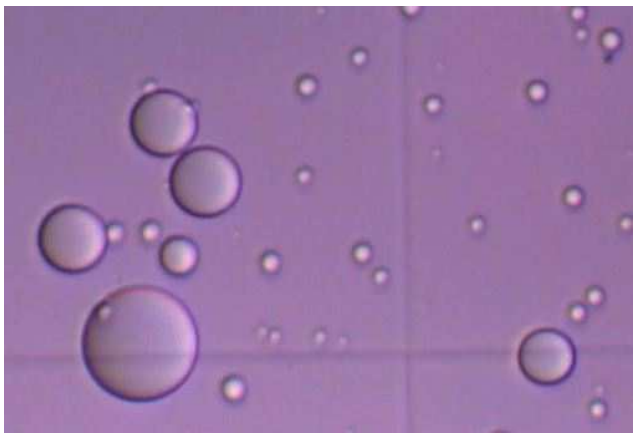
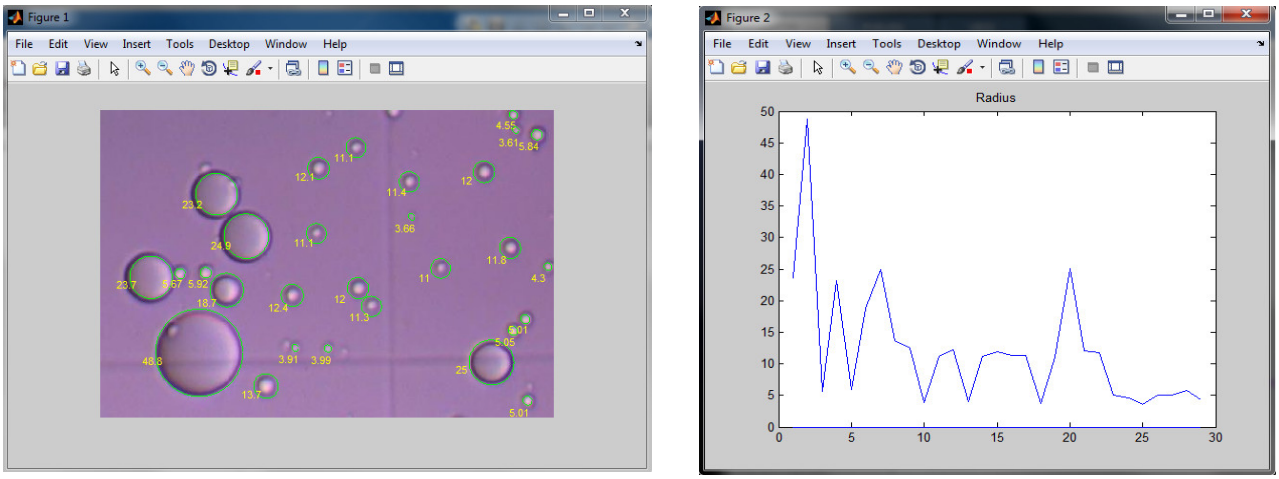


Рис.6.6. Приклад фото для аналізуваня



а) б)

Рис.6.7. Приклад результата роботи програми: а – оброблено фото з визначеними параметрами круглих форм; б – графік розподілення круглих форм за їх розмірами

Висновок. при вивченні цього курсу, я навчився використовувати пакет MATLAB у практичних цілях, відкрив для себе потужне доповнення Matlab Image Toolbox, навчився складати алгоритми та розв`язувати різного типу задачі.