МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

МОДЕЛЮВАННЯ ОПТИКО-ЕЛЕКТРОННИХ ПРИЛАДІВ

ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для студентів, які навчаються за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології»

Київ

КПІ ім. Ігоря Сікорського

2020

Рецензенти: *Аврутов Вадим Вікторович, канд. техн. наук, доцент*

Поздняков Дмитро Вікторович, канд. техн. наук

Відповідальний редактор *Колобродов Валентин Георгійович, доктор техн. наук, проф.*

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол №8 від 09.04.2020 р.) за поданням Вченої ради приладобудівного факультету (протокол № 2/20 від24.02.2020 р.)

Електронне мережне навчальне видання

Кравченко Ігор Володимирович

МОДЕЛЮВАННЯ ОПТИКО-ЕЛЕКТРОННИХ ПРИЛАДІВ

ПРАКТИКУМ

Моделювання оптико-електронних приладів: Практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності «Автоматизація та комп'ютерно-інтегровані технології» / І. В. Кравченко; КПІ ім. Ігоря Сікорського . – Електронні текстові дані. – Київ : КПІ ім. Ігоря Сікорського, 2020. – 103с.

Розглянуто технологію, методики, особливості застосування чисельних методів в системах комп'ютерної математики (СКМ) «MATHCAD», «MATLAB» для моделювання оптико-електронних приладів. Містить теоретичний матеріал, приклади та комплекс завдань для комп'ютерного практикуму та самостійної роботи. Розглянуто питання розв'язання алгебраїчних та нелінійних рівнянь, систем лінійних алгебраїчних та систем нелінійних рівнянь, проведення глобальної та ковзаючої інтерполяції, апроксимації, створення діалогової графічної оболонки для проведення комп'ютерних експериментів в СКМ «MATLAB». Може бути корисний студентам та фахівцям технічних спеціальностей для набуття навичок застосування інформаційних технологій у вигляді сучасних СКМ в учбовій та науково-технічній практиці.

Зміст

ПЕРЕДМОВА

Навчальний посібник призначено для забезпечення інформаційними та методичними матеріалами кредитного модуля "Моделювання оптико – електронних систем" дисципліни "Комп'ютерне моделювання процесів та систем".

В посібнику викладено необхідні відомості щодо чисельних методів розв'язання рівнянь, систем рівнянь, одномірної інтерполяції та апроксимації. Детально розглянуті можливості, особливості застосування, технології розв'язання математичних та технічних задач в системах комп'ютерної математики (СКМ) «MathCAD», «Matlab». Матеріал забезпечений практичними прикладами та комплексом завдань для самостійної роботи.

Системам комп'ютерної математики присвячено значна кількість джерел. Більшість є довідниками з описом можливостей конкретних систем. Деякі з джерел мають технічну, найчастіше механічну спрямованість. Кількість навчальної літератури значно менша. Серед вітчизняних видань україномовними навчальними є джерела [1, 2, 3].

Кредитний модуль "Моделювання оптико-електронних приладів" викладається на третьому курсі навчання в обсязі 4 кредити.

В практикумах 1, 2 навчального посібника розглядаються питання створення графічної оболонки для реалізації діалогу при проведенні комп'ютерного експерименту в СКМ «Matlab».

Розв'язанню алгебраїчних та нелінійних рівнянь, систем лінійних алгебраїчних та систем нелінійних рівнянь, засобам, можливостям, особливостям застосування, методиці розв'язання цих завдань в СКМ «MathCAD» присвячено практикум 3, «Matlab» – практикум 4.

Практикуми 5, 6 навчального посібника присвячено обробці табличних даних, регресійному аналізу. Розглядаються засоби, можливості, особливості застосування, методики розв'язання задач глобальної поліноміальної, ковзаючої сплайн інтерполяції, апроксимації в СКМ «MathCAD», «Matlab».

Практикуми 7, 8 містять приклади типових задач, які вирішуються при розробленні оптико-електронних приладів, та технології їхнього розв’язання в СКМ «MathCAD» та «Matlab».

Приклади та завдання в посібнику базуються на версії «MathCAD 15» «Matlab 7X». Види вікон діалогу та синтаксис команд в інших версіях пакетів можуть дещо відрізнятись від наведених у даному посібнику.

Мета навчального посібника – допомога студентам в самостійному вивченні відповідних розділів навчальної дисципліни, в інформаційному та методичному забезпеченні комп’ютерного практикуму, в набутті навичок застосування інформаційних технологій у вигляді сучасної СКМ в учбовій та науково-технічній практиці. Посібник може використовуватись для самостійної роботи та дистанційного навчання.

Практикум 1. Дескрипторна графіка Matlab. Низькорівневе програмування

Метою практикуму є вивчення принципів побудови графічних об’єктів із завданими властивостями та можливостями керування обчислювальним експериментом, набуття навичок в створенні керуючих елементів засобами низькорівневої графіки СКМ «Matlab».

1.1. Теоретичні відомості

Найбільш сучасним та корисним є проведення моделювання поведінки об’єкта в діалоговому режимі (рис. 1.1).

СКМ «Matlab» має вбудовані можливості для розробки таких вікон.

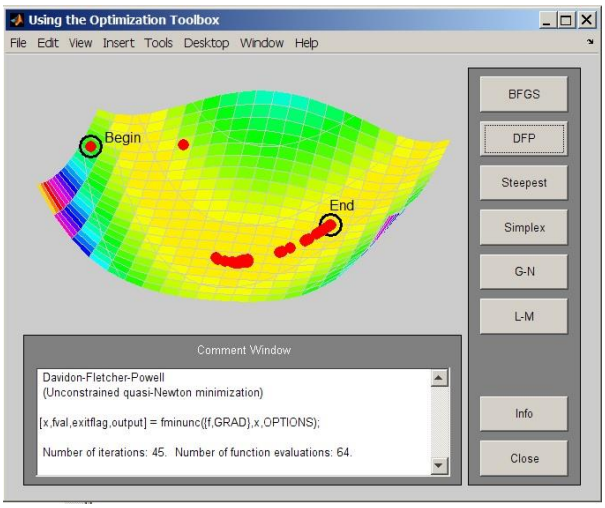


Рис. 1.1 – Вигляд графічного вікна з елементами керування.

Розробляння подібних застосунків базується на об’єктній моделі програмування в Matlab та включає наступні основні етапи:

- опис розташування потрібних елементів інтерфейсу в графічному вікні;
- визначення дій (команд Matlab), які будуть виконуватися при зверненні користувача до цих об’єктів, наприклад, при натисканні кнопки.

Кінцевим результатом є застосунок Matlab з графічним інтерфейсом (**GUI**). Такі програми можуть розміщуватися в кількох файлах, запуск яких проводиться написом імені основного модуля в командному рядку Matlab.

Розробляння аплікацій може вестись засобами вбудованої низькорівненвої дескрипторної (**handle**) графіки «вручну» або з використанням вбудованого середовища **GUIDE**. Середовище **GUIDE** призначено для візуальної розробки зовнішнього виду діалогових вікон аплікацій.

Низькорівненва дескрипторна графіка

Діалогові графічні вікна побудовані на основі ієрархічних графічних об'єктів (рис. 1.2).

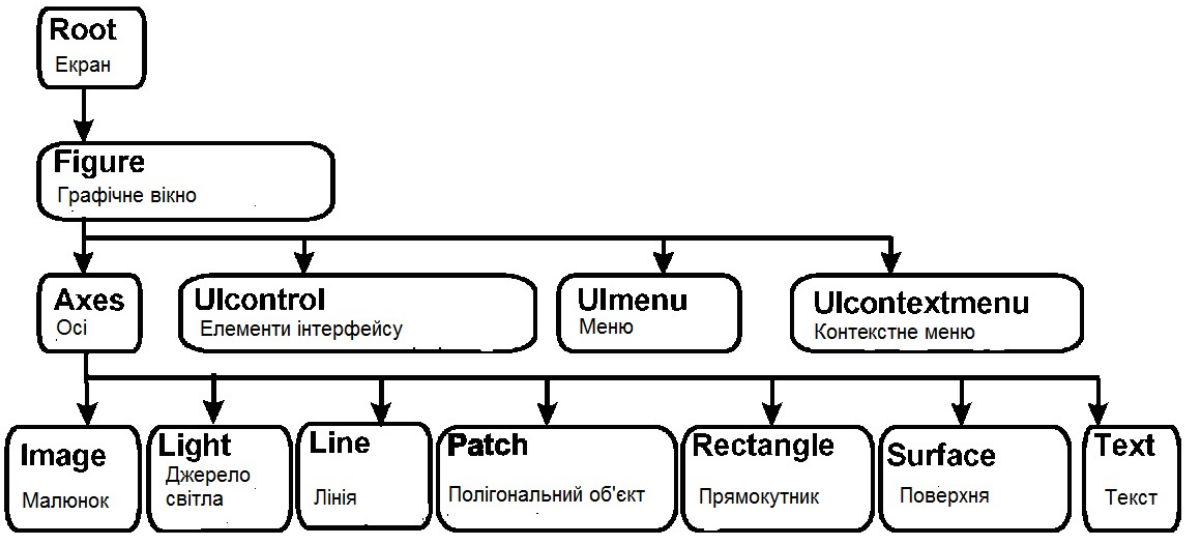


Рис. 1.2 – Ієрархія графічних об'єктів Matlab

Графічні об'єкти є об'єктно-орієнтованими базовими елементами системи керування графіки в Matlab. Вони описуються властивостями (**properties**) та створюють дерево структурної ієрархії: **Root, Figure, Uicontrol, Text, Edit, Checkbox,Listbox, Frame,Popupmenu, Pushbutton, Togglebuttoon, Radiobutton, Slider, Uipanel,Uibuttongroup, Uimenu, Uicontextmenu,Axes, Image, Line, Surface, Light.**

Кожен графічний об'єкт має свій унікальний ідентифікатор **handle** (покажчик), який Matlab присвоює об'єкту при створенні.

Ідентифікатор об'єкта **root** завжди дорівнює нулю. Ідентифікатор об'єкта **figure** – це ціле число, яке за замовчанням відображається в заголовку вікна. Ідентифікаторами інших об'єктів є числа з плаваючою точкою.

Для доступу до поточних об'єктів можна використовувати наступні вбудовані покажчики:

gcf – поточне вікно.

gca – поточні осі (поле графіка).

gco – поточний графічний об'єкт.

gcbo – графічний об'єкт, функція обробки якого виконується в даний час.

Поточним є останній створений чи обраний мишею об'єкт.

Об'єкт **Root** немає предків, а його потомками є графічні вікна – об'єкти **Figure**. Об'єкт **Root** автоматично створюється при запуску Matlab та призначений для установки загальних властивостей. Покажчик на **Root** завжди дорівнює нулю. Властивість **units** об'єкта **Root** задає одиниці виміру, за умовчанням використовуються пікселі. Властивість **Pointer** повертає покажчик на графічне вікно, над яким знаходиться курсор та нуль, коли курсор знаходиться зовні графічних вікон.

Об'єкт **Figure** представляє собою графічне вікно. Графічне вікно має кілька потомків: осі (**Axes**), елементи інтерфейсу (**uicontrol**), меню вікна (**uimenu**), контекстні меню (**uicontextmenu**). Значення **off** властивості **Resize** дозволяє заборонити зміну розмірів вікна, а **on** – дозволити. Властивість **ResizeFcn** змінюється при зміні розмірів вікна користувачем.

Вікна поділяються на два типи: звичайні, між якими можна переключатися, та модальні (діалогові), які потребують завершення роботи з даним вікном. Тип вікна визначається властивістю **WindowStyle** (**normal** за замовчанням чи **modal**).

Властивості **currentAxes** та **Currentobj** визначають покажчики на поточні осі та об'єкти, які розміщені в області вікна.

Властивість **Currentcharacter** повертає символ, який задав користувач з клавіатури.

Доступ до властивостей об'єктів **Figure** за замовчанням проводиться тільки з рівня предку **Figure**, тобто об'єкта **Root**.

Всі об'єкти для візуалізації даних (**Image, Light, Line, Patch, Rectangle, Surface, Text**) є потомками **Axes**.

Властивість **callbackobject** містить покажчик на об'єкт, дія якого обробляється (властивість **callback**) на даний момент. Покажчик на поточне графічне вікно є значенням властивості **CurrentFigure**. Положення покажчика курсору зберігається в векторі з двох елементів, який є значенням **PointerLocation**.

Створення об'єктів

Виклик функції з іменем об'єкту створює цей об'єкт. Наприклад, функція **text** створює об'єкт **text**, функція **figure** створює об'єкт **figure** і т.д.

Більшість вбудованих графічних команд автоматично створює необхідні об'єкти та надає доступ до визначення властивостей у вигляді пари аргументів: «властивість», «значення властивості».

Для можливості подальшого доступу до об'єктів доцільно при їхньому створенні використовувати не просто команду:

```
plot(x,y)
text(0.5, 0.5, 'omega'),
```

а рядок визначення змінної для зберігання ідентифікатора:

```
hf=figure   %покажчик на вікно
ha=axes     %покажчик на поле графіка
hp=plot(x,y) %покажчик на лінію графіка
ht=text(0.5, 0.5, 'omega')  % покажчик на текст.
```

За необхідності за допомогою ідентифікатора зручно програмно змінювати властивості об'єктів.

Для видалення об’єкта служить функція **delete**, з аргументом ідентифікатором об’єкту. Наприклад, **delete(gca)** видалить поточні осі (**axes**), а разом з ними і всі дочірні об’єкти.

Властивості об’єктів

Всі об’єкти описуються властивостями, які визначають, в якому вигляді об’єкти виводяться на екран (табл. 1.1). Matlab надає два механізми для опису властивостей. Властивості можна визначити під час створення об’єкту або змінити чи надати властивості вже існуючому об’єкту.

Команда

Set(покажчик об’єкту, 'PropertyName',PropertyValue)

дозволяє задати значення будь якої властивості графічного об’єкту. Перший аргумент – покажчик об’єкта, а другий та третій – пара: властивість, значення.

Частина назв властивостей закінчується словом **Mode**, наприклад **YTickMode**. Такі властивості можуть мати два значення – **'auto'** (встановлюється за умовчанням) та **'manual'**. Режим **'auto'** забезпечує автоматичний підбір значення відповідної властивості. Визначення значення вектора в **YTick** призводить до зміни значення **YTickMode** з **'auto'** на **'manual'**. Скинути властивості **YTick** можна установкою **YTickMode** в **'auto'**. Вищенаведене вірно для всіх властивостей, які закінчуються на ‘**Mode’**.

Таблиця 1.1. Властивості об’єкта осі

Властивість	Опис	Значення
Box	Рамка навколо поля графіка	on / off
Color	Колір фона осей	Колір може бути задано чисельним значенням RGB, скороченою чи повною назвою: RGB Short Long [1 1 0] y yellow

		[1 0 1] m magenta [0 1 1] c cyan [1 0 0] r red [0 1 0] g green [0 0 1] b blue [1 1 1] w white [0 0 0] k black
FontAngle	Нахил шрифту осей	normal / italic
FontName	Назва шрифту	Рядок з назвою шрифту
Fontsize	Розмір шрифту	Ціле число
FontWeight	Товщина шрифту	normal, bold, light, demi
GridLineStyle	Стиль ліній сітки	-/ - -/ :/ -/ none

Таблиця 1.1. Продовження

Властивість	**Опис**	**Значення**
LineWidth	Товщина ліній осей	Значення в пунктах (1/72 дюйма)
visible	Видимість осей	on /off
Властивості осі X (Y)		
x(y)color	Колір осі	Див. **color**
X(Y)Dir	Напрям осі	normal/ reverse
X(Y)Grid	Перпендикулярність сітки	on/ off
X(Y)AxisLocation	Розташування осі	top / bottom (right / left для осі Y)
X(Y)Lim	Межі змін	Вектор з двох компонентів, що є межами, [-1.5 2.3]
X(Y)Scale	Тип осі	linear / log
X(Y)Tick	Координати міток осі	Вектор з координатами розмітки, наприклад[0 13 5]
X(Y)TickLabel	Мітки осі	Вектор комірок з назвами розмітки (число комірок

		дорівнює довжині вектора з координатами розмітки), наприклад ['zero'; 'one'; 'three'; 'five']

Команда **Get(покажчик об'єкта <,'PropertyName'>)** дозволяє переглянути значення властивостей графічного об'єкта. Аргумент – покажчик об'єкта. Для перегляду значення окремої властивості слугує другий параметр.

Наприклад, наступний код виводить графік залежності температури шляхом створення трьох об'єктів із заміною деяких з їх властивостей за замовчанням (рис. 1.3):

```
days = ['Su';'Mo';'Tu';'We';'Th';'Fr';'Sa'  ];
temp = [21.1 22.2 19.4 23.3 23.5 21.1 20.0];
f= figure;
 a=axes('Ylim',[16 26],'Xtick,1:7,'XtickLabel',days )
 h=line( l:7,   temp)
```

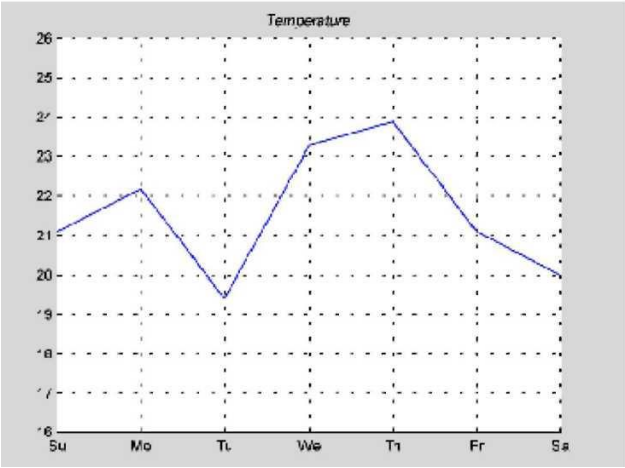


Рис. 1.3 – Графік температурної залежності

Пояснення

days – масив символів, який містить дні тижня, **temp** – чисельний масив температур. Графічне вікно будується функцією **figure** без аргументів, тобто зі значеннями за умовчанням. Осі існують всередині **figure** та мають заданий діапазон по осі **y** та задані мітки по **x**. Лінії

існують всередині осей та мають задані значення для даних по **x** та **y**. Три ідентифікатори **f, a, h** можуть бути застосовані для подальшого використання.

За допомогою команд **uicontrol, uimenu, uicontextmenu** створюються керуючи об'єкти та задаються їхні властивості. Властивості, які не вказано в команді, беруться за умовчанням.

Команда **<h>=uimenu('PropName',PropVal,...)** створює меню, задає його властивості та повертає покажчик **h** на нього.

Команда

<h>=uicontrol('Parent',<parent>,'PropName',PropVal,...)

створює елемент завданого типу, задає його властивості та повертає покажчик **h** на нього. Тип елемента визначається властивістю **Style**.

Команда **<h>=uicontextmenu('PropName',PropVal,...)** створює контекстне меню (викликається правою кнопкою миші, спливає безпосередньо над елементом), задає його властивості та повертає покажчик **h** на нього.

Наприклад, команда

```
uicontrol('Style','edit','String','hello')
```

створює поле введення знизу ліворуч у поточному вікні

Наступні рядки:

```
fig1 = figure;
fig2 = figure;
uicontrol('Parent',fig1,'Style','edit',...
'String','hello');
```

створюють два вікна та у першому створюють поле введення зі значенням у ньому 'hello' (рис. 1.4).

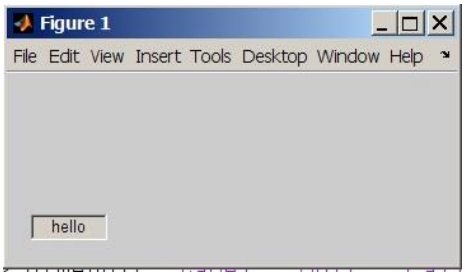


Рис. 1.4 – Вигляд вікна з елементом уведення

Наступні рядки:

```
f = uimenu('Label','Workspace');
uimenu(f,'Label','New Figure','Callback','figure');
uimenu(f,'Label','Save','Callback','save');
uimenu(f,'Label','Quit','Callback','exit',...
'Separator','on','Accelerator','Q');
```

створюють меню '**Workspace**' з пунктами **'New Figure'**, '**Save**', '**Quit**' (рис. 1.5).

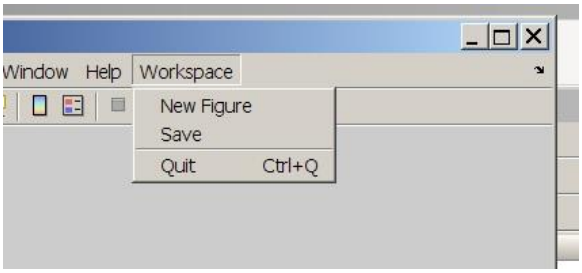


Рис. 1.5 – Вигляд меню

Наступні рядки створюють контекстне меню. Меню спливає при натисканні правої кнопки над лінією та має пункти **'dashed'**, **'dotted'**, **'solid'** (рис. 1.6).

```
cmenu = uicontextmenu;
%Define the line and associate it with the context menu
hline = plot(1:10, 'UIContextMenu', cmenu);
% Define callbacks for context menu items
cb1 = ['set(hline, ''LineStyle'', ''--'')'];
cb2 = ['set(hline, ''LineStyle'', '':'')'];
cb3 = ['set(hline, ''LineStyle'', ''-'')'];
% Define the context menu items
item1=uimenu(cmenu,'Label', 'dashed', 'Callback', cb1);
item2=uimenu(cmenu,'Label', 'dotted', 'Callback', cb2);
item3=uimenu(cmenu, 'Label', 'solid', 'Callback', cb3);
```

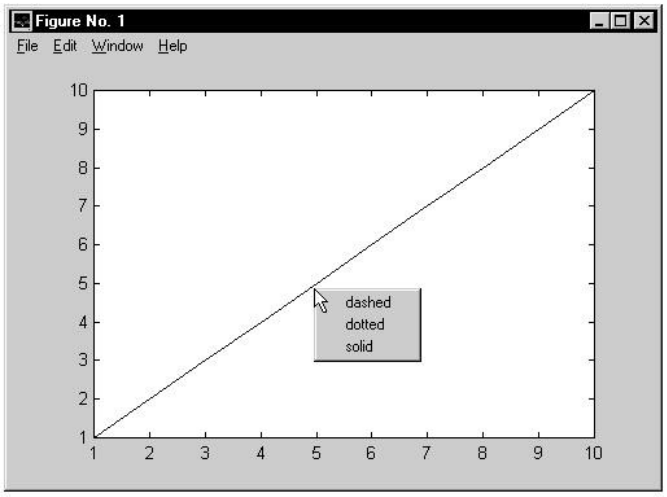


Рис. 1.6 – Вигляд контекстного меню

При описі елементів керування (**uicontrol**) першим параметром вказується дескриптор батьківського вікна, а далі в довільному порядку перераховуються пари: ім’я властивості та значення.

Наприклад, наступні рядки створюють кнопку, при натисканні на яку буде виконуватись функція **fun** (файл fun.m):

```
hBut=uicontrol(hFig,'Style','pushbutton','String',…
'Кнопка', 'Position', [20 20 30 40],'Callback', 'fun');
```

Ключове слово **Style** визначає тип елемента: кнопка **pushbutton**, властивість **String** задає напис на кнопці, властивість **Position** – вектор-рядок з чотирьох чисел: перші два числа встановлюють положення кнопки відносно лівого нижнього кута графічного вікна, третє число задає ширину кнопки, а четверте – її висоту. Властивість **Callback** визначає ім’я функції **fun**, яка буде викликана при натисканні кнопки. Файл **fun.m** повинен бути створений раніше.

Таблиця 1. 2. Основні властивості елементів керування

Властивість	**Зміст**	**Значення**
BackgroundColor	Колір фона	Вектор RGB чи службове слово: [1 0 0] / ‘r’ / ‘red’
BusyAction	Тип обробки переривання користувача при виконанні	cancel / queue

	callback функції	
ButtonDownFcn	Ім’я **callback** функції, що виконується коли елемент відключений	Рядок
CData	RGB матриця з описом рисунка користувача на графічному елементі	Тримірна матриця, елемент – число в діапазоні [0 ..1]
Callback	Ім’я **callback** функції	Рядок
CreateFcn	Ім’я **callback** функції, яку потрібно виконати при створенні елемента	Рядок

Таблиця 1.2. Продовження

Властивість	**Зміст**	**Значення**
DeleteFcn	Ім’я **callback** функції, яку потрібно виконати при видаленні елемента	Рядок
Extent	Розміри елемента	Вектор: [0 0 ширина висота]
FontAngle	Вид шрифту	normal/ italic / oblique
FontName	Ім’я шрифту	Рядок
Enable	Режим доступності елемента	on – активний, inactive – неактивний, графічно не змінений, off – вимкнений и має в сірий колір
FontSize	Розмір шрифту	Число
FontUnits	Одиниці виміру розмірів шрифту	Points(1/72inch)/normaliz / in /cm / pixels
ForegroundColor	Колір шрифту	Див. **Color**
HorizontalAlignment	Горизонтальне центрування	left / center/ right
ListboxTop	Довжина списку, коли кількість позицій більше	Число

	висоти елемента	
Max	Число. **Check boxes, Toggle buttons, Radio buttons** – значення властивості **Value** при обранні елемента.	
Min	Число. **Check boxes, Toggle buttons, Radio buttons** – значення властивості **Value** коли елемент не обраний.	
Parent	Покажчик на батьківський об’єкт	Покажчик
Position	Положення та розміри елемента	Вектор [left bottom width height]

Таблиця 1.2. Продовження

Властивість	**Зміст**	**Значення**
SliderStep	Величина зсуву повзунка	Вектор, два значення в діапазоні 0 ..1 [поле повзунка кнопка]
String	Назва елемента або пункти списків, що відображуються на екрані	Рядки багаторядкового тексту розділяються символами `\n '
Style	Тип об’єкта	pushbutton / togglebutton / radiobutton / checkbox / edit/ text / slider / frame / listbox / popupmenu
Tag	Ім’я об’єкта	Рядок
TooltipString	Спливаюча підказка	Рядок
UIContextMenu	Контекстне меню об’єкта	Покажчик
Units	Одиниці виміру розмірів об’єкта	pixels / normalized (0..1) /| inches / centimeters / points(1/72 inch) / characters
UserData	Дані користувача. (Визначення	Матриця

	– **set**, показ – **get**)	
Visible	Видимість об’єкта	on /off
Value	Поточне значення об’єкта	Число чи вектор Check boxes: Max/ Min , List boxes: вектор номерів обраних пунктів Pop-up menu: номер обраного пункту Radio buttons: Max/ Min Sliders: положення повзунка Toggle buttons: Max/ Min

Діалогові вікна (вікна повідомлень)

Зручність роботи в значній мірі визначається вікнами повідомлень про можливі наслідки дій користувача. Matlab надає можливість використання стандартних діалогових вікон Windows. Діалогові вікна можуть мати три наперед визначені кнопки ‘Yes’, ‘No’ и ‘Cancel’.

Вікно підтвердження.

Деякі дії доцільно підтверджувати. Наприклад, користувач може випадково натиснути кнопку Очистить, яка призначена для очистки осей. Перед виконанням дії слід вивести діалогове вікно, в якому користувач повинен підтвердити необхідність дій.

Вікно підтвердження створюється функцією

```
questdlg('string-question', <'window label'>,...
<'button name'>,<'button name'>,<'default ...
 button name'>)
```

Перший аргумент – рядок повідомлення, другий аргумент – опціональний заголовок вікна, наступні аргументи – опціональні імена кнопок, останній аргумент – ім’я кнопки за замовчанням. Результат – рядок з іменем натиснутої кнопки.

Наприклад, команда

```
s=questdlg('Закрыть окно приложения?','Подтверждение ...
закрытия', 'Да', 'Нет', 'Нет');
```

виведе вікно (рис. 1.7).

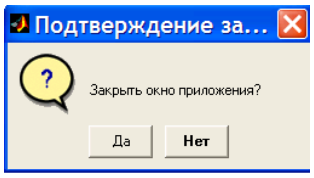


Рис. 1. 7 – Вікно підтвердження

Обробку дій користувача можна організувати по назві натиснутої користувачем кнопки:

```
switch s
    case 'Да'  delete(gcf)
    case 'Нет'  return
end
```

Вікна відкриття файлу та запису в файл

Діалогове вікна роботи з файлом створюється функцією

```
[fname, pname] = uigetfile(<'*.dat'>, <'title'>)
```

Опціональний аргумент '*.dat' – фільтр розширення файлів.

Функція виводить діалогове вікно з іменем **'title'** та повертає в першому результаті рядок **fname** імені, а в другому **pname** – шлях до файлу, обраному користувачем. Вихідні результати дорівнюють нулю, якщо користувач відмінив відкриття чи при відкритті файлу виникла помилка.

Функція **uiputfile** призначена для виведення діалогового вікна збереження файлу. Вхідні аргументи та вихідні результати **uiputfile** є такими ж, як у функції **uigetfile**.

Вікно повідомлення про помилку.

Для створення діалогового вікна з повідомленням про помилку призначена функція

```
errordlg (<'errorstring'>, <'window title'>)
```

Аргументами функції є рядки з текстом та заголовком вікна.

Обробка дій користувача

При виборі елемента керування виконується функція (m-файл), ім'я якої указано в властивості **Callback** елемента.

Завдання дій елемента керування можливо:

- написанням унікальних функцій оброблення для кожного елемента,
- використанням однієї функції для спільного оброблення.

В останньому випадку в функцію оброблення слід передавати в якості аргументу ознаку, яка визначить який елемент викликає функцію. В якості ознаки може бути використано ім'я елемента (властивість **Tag**), покажчик елемента чи ознака, яка призначена користувачем. В функції оброблення необхідно передбачити виконання перенаправлення дій в залежності від значення аргументу-ознаки, наприклад функціями **if**, **switch**.

Для визначення дескриптора елемента може використовуватися функція

```
h = findobj('PropertyName',PropertyValue,...)
```

В якості аргументів задаються властивість елемента та її значення.

Наприклад, рядком

```
H_Edit=findobj('Style','edit')
```

в змінній **H_Edit** будуть збережені всі покажчики на об'єкти типу **edit**.

Рядком **H_MEdit=findobj('Tag','My_edit')**

в змінній **H_MEdit** буде збережено дескриптор об'єкта з іменем 'My_edit'.

Для введення користувачем функціональних залежностей в поле **edit** зручно використовувати функцію

```
inline(expr,arg1,arg2, ...)
```

Функція перетворює текстовий аргумент **expr** в функцію Matlab, значення якої може бути розраховано. Опціональні аргументи **arg** – рядки, що визначають аргументи функції.

Наприклад, запис **g=inline('sin(x)','x')** створює функцію **g(x) = sin(x)**, яка може бути викликана як будь-яка функція пакета.

```
> g(1)
ans =    0.8415
```

Структура модуля

Модуль повинен вміщувати описи всіх графічних об’єктів та всі описи дій, як графічних елементів, так і дій з даними та по алгоритмах користувача.

Описи дій (**`callback`**) графічних елементів оформлюються у вигляді підпрограм-функцій користувача (**`function`**). Функція може бути одна, спільна для всіх елементів. Функцій може бути декілька з описом в кожній з них дії одного елемента.

Опис вікна **`GUI`** можна розмістити в головному скрипт-файлі модуля. В такому випадку скрипт вийде громіздким, що ускладнює орієнтацію в тексті модуля. Рекомендується виносити опис вікна в окремий скрипт-файл.

Версії Matlab, вищі за 7-му, дозволяють застосовувати підпрограми-функції без параметрів. Їх також можна використовувати як у вигляді основного модуля, так і у вигляді опису графічного вікна. Особливістю використання підпрограм-функцій є локальність всіх даних в функції. За необхідності використання спільних структур даних, в головному модулі та підпрограмі-функції слід ввести «глобальні» змінні у всіх модулях командою **`global`**.

1.2. Приклади та завдання до виконання

Приклад.

Створити вікно введення даних та виведення графіка, керування виглядом для функції **`y(x)=sin(x)`** у відповідності до рис. 1. 9 засобами дескрипторної графіки.

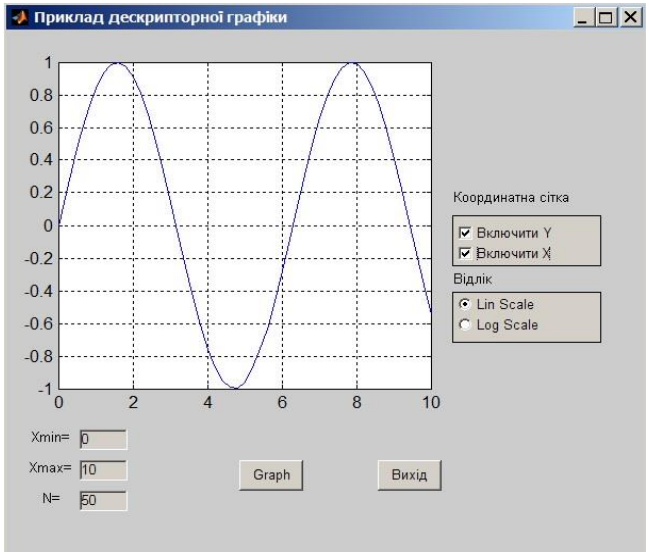


Рис. 1. 9 – Вікно прикладу

РОЗВ’ЯЗАННЯ

Файл-опис вікна:

```
function fig =priklad1_1()
global xmin xmax step N
%створення вікна з дескриптором f1
f1=figure('Color',[0.8   0.8   0.8],'MenuBar','none',...
'Position',[250  100  600  550],  'Name',  'Приклад  ...
дескрипторноїграфіки',     'NumberTitle','off','Tag',...
'Fig1');

% створення поля графіки
a1 = axes('Parent',f1, 'Units','pixels','Position', ...
[50 200 350 320], 'Tag','Axes1', 'XColor',[0 0 0], ...
'YColor',[0 0 0]);
xlabel('X');
ylabel('Y');
title('Графік', 'FontName','Arial');

% кнопка малювання
a1 = uicontrol('Parent',f1, 'Callback',...
'callfun(''press_Run'');','Position',[220 100 ...
60   30],'String','Graph','Tag','button_Run','Style',...
'pushbutton');
```

```
% кнопка вихід
a1 = uicontrol('Parent',f1, 'Callback',...
'callfun(''press_Quit'');', 'Position',[350 100 ...
60 30],'String','Вихід', 'Tag','button_Quit');

%кнопки сітки в рамці
a1=uicontrol('Parent',f1, 'Position', [420 320 140 ...
50],'Style','frame', 'Tag','Frame1');
a1 = uicontrol('Parent',f1, ...
'Callback','callfun(''press_GridX'');', 'Position',...
[425 325 100 15], 'String','Включити X', 'Style',  ...
'checkbox','Tag','box_GridX' );
a1 = uicontrol('Parent',f1, 'Callback',...
'callfun(''press_GridY'');','Position',[425 345 100 ...
15],'String','Включити    Y','Style','checkbox',    ...
'Tag','box_GridY');
a1 = uicontrol('Parent',f1,'HorizontalAlignment',...
'left','Position',[420 380 190 15],'String',...
'Координатна сітка', 'Style','text',  'Tag',...
'StaticText1','BackgroundColor',[0.8 0.8 0.8]);

% кнопки масштабу в рамці
a1 = uicontrol('Parent',f1,'Position',[420 245 140 ...
50],'Style','frame', 'Tag','Frame2');
a1 = uicontrol('Parent',f1, 'Callback',...
'callfun(''press_Lin'');', 'Position',[425 275 90 ...
15], 'String','Lin Scale', 'Style','radiobutton', ...
'Tag', 'Radio_Lin', 'Value',1);
a1 = uicontrol('Parent',f1,'Callback',...
'callfun(''press_Log'');', 'Position',[425 255 90 ...
15], 'String','Log Scale', 'Style','radiobutton', ...
'Tag','Radio_Log', 'Value',0);
a1 = uicontrol('Parent',f1,'HorizontalAlignment',...
'left','Position',[420 300 90 15],'String','Відлік',...
'Style','text', 'BackgroundColor',[0.8 0.8 0.8]);

% поля уведення даних з поясненнями
```

```
a1 = uicontrol('Parent',f1, 'Callback',...
'callfun(''Edit_xmin'');','HorizontalAlignment',...
'left','Position',[70     140     45     20],     ...
'Style','edit', 'Tag','Edit_xmin');
a1 = uicontrol('Parent',f1, 'Position',[20 140 45 ...
20],     'Style','text',     'string','Xmin=',     ...
'BackgroundColor',[0.8 0.8 0.8]);
a1 = uicontrol('Parent',f1, 'Callback',...
'callfun(''Edit_xmax'');', 'HorizontalAlignment',...
'left', 'Position',[70 110 45 20], 'Style',...
'edit', 'Tag','Edit_xmax');
a1 = uicontrol('Parent',f1, 'Position',[20 110 45 ...
20],Style','text',         'string','Xmax=',         ...
'BackgroundColor',[0.8 0.8 0.8]);
a1 = uicontrol('Parent',f1,'Callback',...
'callfun(''Edit_N'');','HorizontalAlignment','left',...
'Position',[70  80  45  20], 'Style','edit',  'Tag',...
'Edit_N');
a1= uicontrol('Parent',f1, 'Position',[20 80 45 20],...
'Style','text', 'string','N=','BackgroundColor',...
 [0.8 0.8 0.8]);
```

Функція обробки дій (**callback**):

```
function [x,y,t,T]=callfun(event)
global xmin xmax step N;
switch event
case 'Edit_xmin'
   xmin=str2num(get(gcbo,'String'));
case 'Edit_xmax'
   xmax=str2num(get(gcbo,'String'));
case 'Edit_N'
   N=str2num(get(gcbo,'String'));
   step=(xmax-xmin)/N;
case 'press_GridX'
   if get(gcbo,'Value')
      set(gca,'Xgrid', 'on')
   else
```

```
      set(gca,'Xgrid', 'off')
   end

case 'press_GridY'
   if get(gcbo,'Value')
      set(gca,'Ygrid', 'on')
   else
      set(gca,'Ygrid', 'off')
   end
case 'press_Lin'
      h=findobj('Tag','Radio_Log');
   set(h,'Value',0);
case 'press_Log'
     h=findobj('Tag','Radio_Lin');
   set(h,'Value',0);
case 'press_Run'
    HStyle_Edit=findobj('Style','edit');
    if max(strcmp(get(HStyle_Edit,'String'),''))
       errordlg('Не всі дані задано','Помилка!');
    else
        h=findobj('Tag','Radio_Lin');
       if get(h, 'Value')
          x=xmin:step:xmax;
          plot(x,sin(x));
       else
          x=xmin:step:xmax;
          semilogx(x,sin(x));
       end
   end
case 'press_Quit'
   button=questdlg('Завершити роботу ?', ...
          'i1','Так','Ні','Ні');
   if strcmp(button,'Так')
      clear all
      close all
      clc
   end
end
```

Пояснення

В прикладі:

– модуль складається з двох функцій користувача. Головна функція **`priklad1_1`** містить описи графічних елементів. Для передавання даних між функціями частина змінних зроблена глобальними.

– не використовуються покажчики об’єктів в явному виді, крім об’єкта «вікно». Для визначення об’єкта задіяні системні змінні **`gca`**, **`gcbo`** та функція **`findobj`**,

– використана одна функція обробки **`callfun`**, яка є спільною для всіх елементів. Для ідентифікації джерела інформації використано признак користувача як аргумент функції обробки,

– не передбачені початкові дані. Для визначення того, що всі дані введено з допомогою функції **`findobj`** визначається масив посилань на елементи введення типа **`edit`**, з допомогою функції **`get`** зчитується зміст елементів в масив, з допомогою функції **`strcmp`** визначається чи введено якусь інформацію в поля (шляхом порівняння зчитаної інформації з ''), з допомогою функції **`max`** визначається значення максимального елемента в масиві (результат, який не дорівнює 0, показує, що не у всі елементи введено дані).

Завдання 1.1.

Написати скрипт для прикладу 1.1. Оформлення визначити функцією. Параметри передавати у створені об’єкти у вигляді змінних, які попередньо визначити. Звертатися до елементів по індивідуальних покажчиках. Графік намалювати червоною лінією товщиною 2.

Завдання 1.2.

Виконати приклад 1.2 з використанням окремих функцій обробки дій (callback).

Практикум 2. Дескрипторна графіка Matlab. Середовище GUIDE

Метою практикуму є вивчення принципів побудови графічних об'єктів із завданими властивостями та можливостями керування обчислювальним експериментом. Набуття навичок в створенні керуючих елементів засобами редактора **GUIDE**.

2.1. Теоретичні відомості

Для облегшення процесу створення графічних діалогових вікон в Matlab існує середовище **GUIDE** (рис. 2.1).

Застосунок, що розроблена в середовищі **GUIDE** зберігається в двох файлах. Один – з розширеннями «fig», другий – «m». Перший вміщує інформацію про розташованих у вікні об'єктах, а другий є М-файлом з основною функцією та функціями обробки (Callback).

Додавання елемента інтерфейсу в редакторі призводить до автоматичного створення відповідної функції обробки.

Виклик редактора **GUIDE** проводиться командою **guide**.

У вікні редактора знаходяться (рис. 2.1): рядок меню; панель інструментів керування аплікацією; заготовка вікна з нанесеною сіткою; вертикальна та горизонтальна лінійки; панель інструментів для вставляння елементів інтерфейсу у вікно аплікації.

Користувач може використовувати (рис. 2.2): кнопки, перемикачі, кнопки-перемикачі, поля вибору, поля текстового введення, списки, випадаючи списки, текстові поля, декоративні рамки (групові панелі), повзунки, графіки.

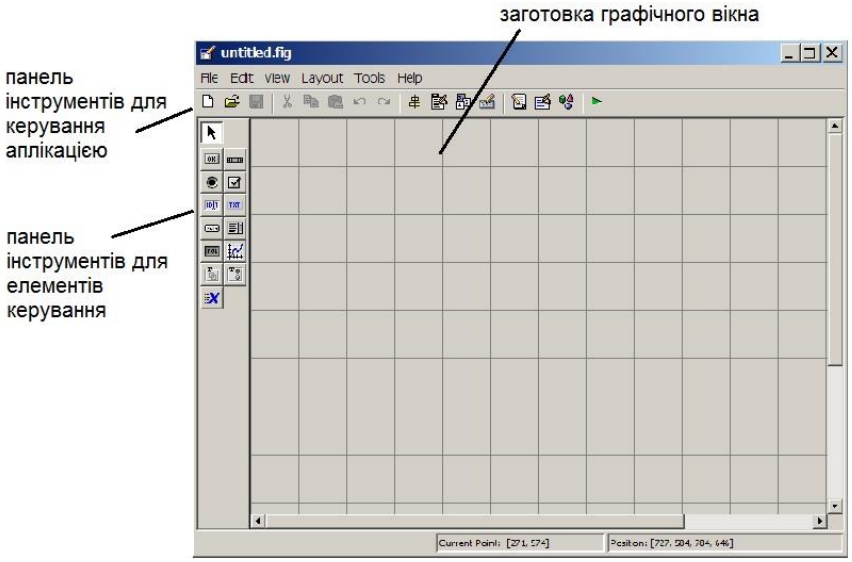


Рис. 2. 1 – Редактор діалогових вікон **GUIDE**

Рис. 2. 2 – Панель інструментів для додавання елементів інтерфейсу

Створення елементів керування в діалоговому вікні

Для того, щоб розташувати елементи інтерфейсу треба клацнути мишею відповідну кнопку на панелі інструментів. Повторне клацання в потрібному місці заготовки вікна розташує елемент в вікні. Інший спосіб складається з «перетаскування» з утриманням лівою кнопкою миші кнопки з панелі інструментів в потрібне місце заготовки вікна.

Розмір, положення доданих об'єктів змінюються за допомогою миші, редактора властивостей об'єктів (*Property Inspector*), опції "*Align Objects*" пункту "*Tool*" меню чи панелі інструментів (Рис. 2. 3).

Будь-який об’єкт можна видалити з вікна за допомогою клавіші “*Delete*” на клавіатурі або опції “*Clear*” меню.

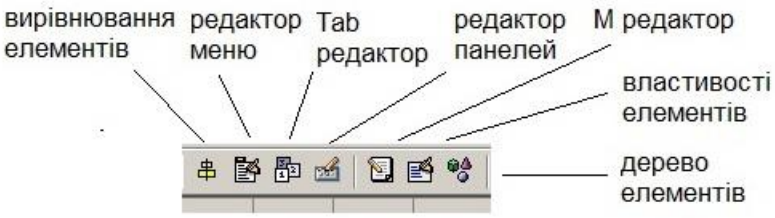


Рис. 2.3 – Панель інструментів завдання властивостей

Редактор містить прості засоби вирівнювання об’єктів – сітку та лінійку. На заготовці вікна нанесені сітка, вертикальна та горизонтальна лінійки. Лінійка дозволяє розміщувати елементи інтерфейсу в позиції з будь-якими координатами в пікселях. Координати відраховуються від лівого нижнього кута заготовки вікна. Сітка редактора дозволяє зробити зсув об’єктів дискретним. Пункт *Grid* and *Rulers* меню *Tools* виводить діалогове вікно *Grid and Rulers*, з відповідними налаштуваннями(рис. 2.4).

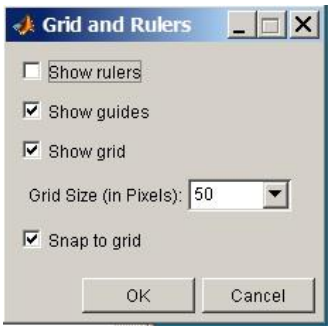


Рис. 2.4 – Діалогове вікно Grid and Rulers

Мітки *Show rulers* та *Show grid* керують відображенням лінійок та сітки в вікні редактора, а випадаючий список *Grid Size* задає розмір комірок сітки. Мінімально допустимий розмір десять пікселів дозволяє достатньо точно розташувати елементи в вікні аплікацій. Перемикач *Snap to grid* керує прив’язкою переміщень до ліній сітки.

Прив’язка дозволяє розташувати об’єкт та змінити його розміри тільки за умови проходження меж об’єкта по лініях сітки. Вибір мілкого кроку

сітки разом з прив'язкою дозволяє швидко оформити вікно. Плавно змінювати положення об'єкта можна за допомогою клавіш зі стрілками.

Вирівнювання об'єктів в ряд по вертикалі чи горизонталі потребує попереднього визначення допоміжної лінії. Для цього можна додати горизонтальну чи вертикальну лінію вирівнювання. Слід навести курсор миші на відповідну лінійку (курсор змінює форму на двосторонню стрілку) та, тримаючи ліву кнопку миші, витягнути на поле синю лінію. Пересування об'єктів на лінію вирівнювання викликає автоматичне розташування їх на даній лінії. Самі лінії вирівнювання можна прибрати перетягнувши їх мишею назад на лінійку. Перемикач *Show guides* вікна *Grid and Rules* керує відображенням ліній вирівнювання.

Прив'язка діє навіть при виключених перемикачах *Show rulers, Show guides, Show grid.*

Програмування подій

Структура функції, яка генерується редактором **GUIDE** складніше, ніж за «ручної» розробки. В ній закладено додаткові дані та структури.

Функції обробки мають наступний формат:

function varargout=name_Callback(h,eventdata,handles, varargin),

де **name** – ім'я графічного елемента, **eventdata** – не використовується, **handles** – структура даних, в якій зберігаються покажчики графічних елементів, **varargin** – змінна, яку можна використати для передачі в функцію додаткових даних.

В структурі **handles** можна зберігати також дані користувача. Доступ до покажчика проводиться по імені. Наприклад, **get(handles.slider1,'Value');**

За допомогою структури **handles** можна організувати обмін даними між функціями. Функція, яка передає дані, повинна містити запис даних в нове поле та збереження структури командою **guidata**.

Наприклад, можна зберегти масив в полі **datl** структури **handles** , а потім використати його в іншій функції:

```
handles.datl =  [1.2 3.2 0.1];
guidata(gcbo, handles)
```

```
max(handles.datl)
```

Обробка списків

Обробка **callback** випадаючого списку складається з визначення вибору користувача та відповідній зміні функції обробки. Властивість **value** містить номер обраного рядка (рядки списку нумеруються з одиниці).

В списках може бути виділено кілька елементів. Властивість **value** містить в такому випадку вектор номерів обраних елементів. Вибір кількох елементів визначається значеннями властивостей **Max** и **Min**. Якщо різниця **Max – Min** більше одиниці, то можна виділяти кілька рядків.

Застосування редактора **GUIDE**:

- суттєво зменшує обсяг дій при створенні опису **GUI**, особливо в операціях визначення положення та вирівнювання елементів;
- знижує вимоги до кваліфікації користувача як програміста;
- полегшує передавання даних між модулями;
- не дозволяє редагувати графічні елементи без застосування редактора;
- обмежує міграцію застосунків між версіями Matlab.

Застосування «ручного» програмування:

- збільшує обсяг дій при створенні опису **GUI**;
- вимагає підвищеної кваліфікації користувача як програміста;
- ускладнює передавання даних між модулями;
- забезпечує вільний вибір типів підпрограм застосунку;
- дозволяє редагувати графічні елементи без застосування редактора;
- не обмежує міграцію застосунків між версіями Matlab.

2.1. Завдання до виконання

Завдання 2.1.

Виконати приклад 1.2 з використанням середовища **GUIDE**.

Завдання 2.2.

Розробити застосунок в середовищі **GUIDE** для побудови графіка функції «прямокутника» (завдання 2.3 [3]) в діапазоні x=[-2..2] з елементами введення амплітуди, зсуву, тривалості імпульсу, кількості точок. керування вікном. Передбачити початкові вихідні дані U=1, N=20, t=1, to=0.

Завдання 2.3.

Розробити **GUI** скрипт-файл для моделювання коефіцієнта передачі амплітуди та фази фотоелектричного ланцюга в напівлогарифмічному масштабі один під другим в одному вікні (рис. 2.5) згідно наступного виразу

$$K(\omega) = \frac{n}{1 + a_1 + a_2 - \omega^2 \frac{\tau_1 \tau_2}{a_1} + j\omega(\tau_1 + \tau_2) + \frac{1}{j\omega\tau_3}}$$

де n – коефіцієнт передачі трансформатора; R_f – опір фотоприймача; R_t – опір навантаження; C_t – ємність; L_1 – індуктивність первинної обмотки; L_2 – індуктивність вторинної обмотки трансформатора.

$$a_1 = \frac{R_f}{R_t} \quad a_2 = \frac{L_1}{L_2} \quad \tau_1 = C_t R_f \qquad \tau_2 = \frac{L_1}{R_t} \qquad \tau_3 = \frac{L_2}{R_f} \quad n = \frac{L_2}{L_1}$$

Графіки амплітудної та фазової характеристик намалювати в "напівлогармічному" масштабі. Аргументи графіків доцільно формувати командою **logspace**.

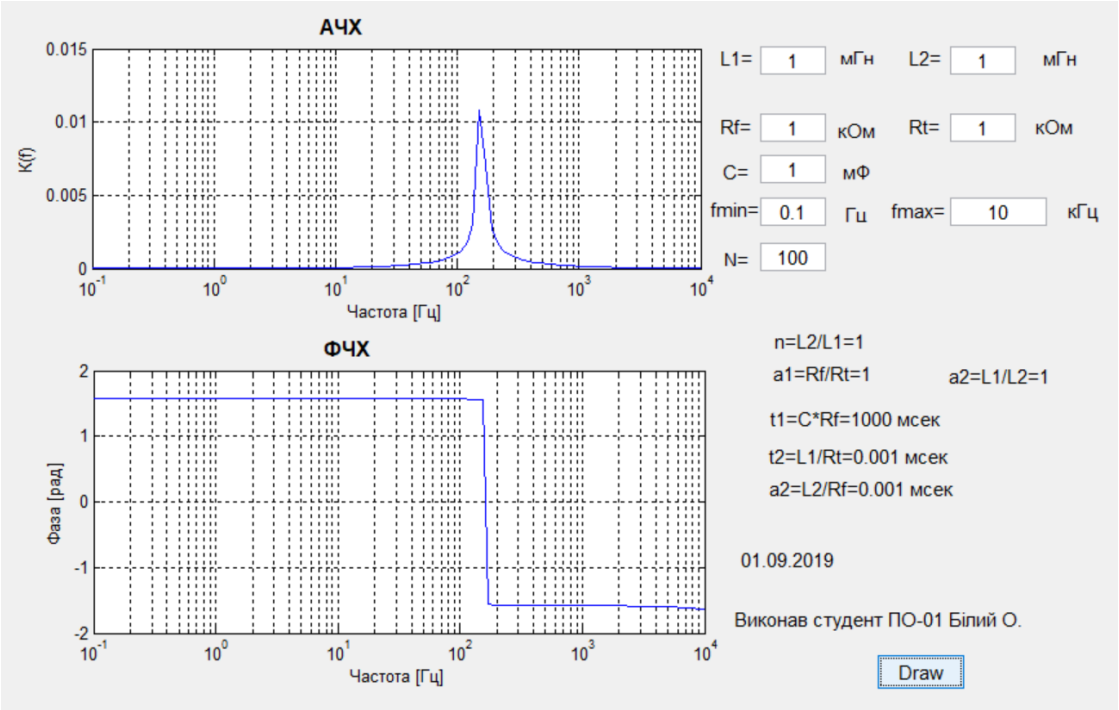


Рис. 2.5 – Вигляд вікна завдання 2.3

Практикум 3. Розв’язання рівнянь та систем рівнянь засобами MathCAD

Метою практикуму є вивчення методів, особливостей розв’язання рівнянь та систем рівнянь в СКМ. Набуття навичок в застосуванні засобів розв’язання алгебраїчних та нелінійних рівнянь, систем рівнянь в СКМ «MathCAD».

3.1. Теоретичні відомості

Чисельні засоби

Функція **polyroots** призначеня для розв'язання алгебраїчних рівнянь та може використовувати метод *Лагера* та метод *супроводжуючої матриці*. Вибір методу проводиться через контекстне меню функції (рис. 3.1).

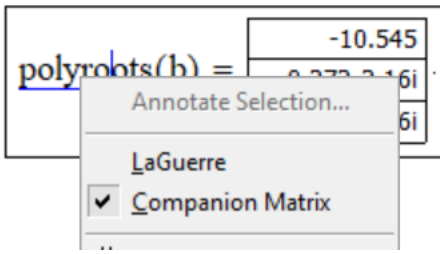


Рис. 3.1 – Контекстне меню функції **polyroots**

Функція **polyroots** має єдиний параметр – вектор-стовпець коефіцієнтів алгебраїчного рівняння. Коефіцієнти мають розташовуватися в порядку збільшення ступеню змінної рівняння. Тобто починатися з елемента, що має нижчий ступінь. Функція повертає всі корені рівняння у вигляді вектора стовпцю. Точність результатів не залежить від значення змінної **TOL**.

Наприклад, для алгебраїчного рівняння

$$2x^3 + 20x^2 - 2x + 100 = 0 \qquad (3.1)$$

застосування функції може мати наступний вигляд:

$$\text{polyroots}\begin{pmatrix}100\\-2\\20\\2\end{pmatrix} = \begin{pmatrix}-10.545\\0.272-2.16i\\0.272+2.16i\end{pmatrix} \qquad b := \begin{pmatrix}100\\-2\\20\\2\end{pmatrix} \qquad \text{polyroots}(b) = \begin{pmatrix}-10.545\\0.272-2.16i\\0.272+2.16i\end{pmatrix}$$

Функція **root** слугує для знаходження кореня рівняння чисельними методами. Обмежень на вид рівняння немає.

Функція **root(f(x),x, <a, b>)** здійснює чисельне розв'язання рівнянь методом *січних (Мюлера)* або *дихотомії (Брента).*

Для застосування методу січних в функції **root** використовуються два аргументи: вираз рівняння у вигляді імені функції користувача чи безпосередньо виразу функції рівняння та змінна, відносно якої необхідно розв'язання рівняння. Змінна повинна мати чисельне значення початку пошуку. Наприклад:

$z := 5 \quad y(x) := \sqrt[3]{x - 5}$

$\text{root}\left(\sqrt[3]{z - 5}, z\right) = 5$

$\text{root}(y(z), z) = 5$

Метод дихотомії поребує додаткового визначення після імені змінної рівняння двох праметрів – початку та кінця діапазону пошуку кореня. Наприклад:

$z := 5 \quad y(x) := \sqrt[3]{x - 5} \quad a := 1 \quad b := 7$

$\text{root}\left(\sqrt[3]{z - 5}, z, 1, 7\right) = 5$

$\text{root}(y(z), z, a, b) = 5$

Функція **root** може застосовуватися до рівняння з параметрами. Згачення параметра повинно бути попередньо визначено явно у вигляді чисельного значення. Наприклад,

$a := 2 \qquad y(x) := a \cdot x^3 + 20 \cdot x^2 - 2 \cdot x + 100 \quad z := 5$

$\text{root}(y(z), z) = -10.545$

.

Параметр рівняння може бути введено аргументом функції рівняння:

$ya(a, x) := a \cdot x^3 + 20 \cdot x^2 - 2 \cdot x + 100$

$\text{root}(ya(1, z), z) = -20.34$

.

Функція **root** може бути використана всередині функції користувача. Наприклад:

$$y(a,x) := x - a$$

$$b(f,p,z) := \text{root}(f(p,z),z) \qquad z := 4 \qquad b(y,0.6,z) = 0.6$$

В функції **root** корінь визначається з точністю, значення якої зберігається в системній змінній **TOL**.

Використання функції **root** в режимі метода січних може привести до неочікуваного або невірного результату.

Наприклад, для рівняння (3.1) графік (рис. 3.2) показує наявність одного дійсного кореня в околі значення -10.5.

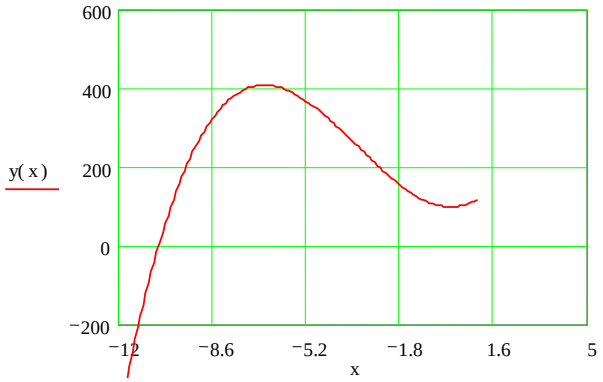


Рис. 3.2 – Графік функції (3.1)

Розв'язання з початковим наближенням -5 видає саме цей корінь – 10.545. Розв'язання з початковим наближенням 0 видає комплексне значення. Це значення є коренем рівняння, але не є дійсним числом.

$$y(x) := 2\cdot x^3 + 20\cdot x^2 - 2\cdot x + 100 \qquad z := -0$$

$$\text{root}(y(z),z) = 0.272 + 2.16i$$

Для рівняння $x^2 + 0.0001 = 0$ коренями є комплексні числа ±0.01i. Застосування функції **root** зі значенням **TOL=0.001** (за замовчанням) дає дійсне значення, яке не можна суворо вважати коренем рівняння:

$y2(x) := x^2 + 0.0001$

$x2 := root(y2(z), z) \quad x2 = 0.018$

$y2(x2) = 4.225 \times 10^{-4}$

.

При цьому формальний критерій співвідношення значення функції та похибки виконано. Правильне значення кореню з'являється тільки за умови зменшення точності нижче за значення вільного члена рівняння:

$y2(x) := x^2 + 0.0001 \quad TOL := 10^{-4}$

$x2 := root(y2(z), z) \quad x2 = -0.01i$

$y2(x2) = 0$

Спосіб зменшення значення похибки можна вважати універсальним.

Для рівняння, яке має декілька коренів, початкове значення в точці зі значенням похідної, що наближене до нуля, може привести не до найближчого очікуваного значення. Чим ближче до нуля буде похідна, тим більшим буде знайдений корінь.

Для дослідження залежності винайдених коренів від початкового значення визначимо початкове значення ранжованою змінною в околі значення π/2 та викличимо функцію для всіх початкових значень:

$x := \frac{\pi}{2} - 0.1, \frac{\pi}{2} - 0.08 .. \frac{\pi}{2} + 0.1 \qquad y(z) := root(\sin(z), z)$

x =	$\frac{y(x)}{\pi}$ =	sin(y(x)) =
1.4708	-2.0000	$-2.107 \cdot 10^{-14}$
1.4908	-2.0000	$1.357 \cdot 10^{-9}$
1.5108	-4.0000	$2.137 \cdot 10^{-13}$
1.5308	9.0000	$1.102 \cdot 10^{-15}$
1.5508	$9.1040 \cdot 10^{3}$	$-1.74 \cdot 10^{-12}$
1.5708	$-7.2900 \cdot 10^{4}$	$1.329 \cdot 10^{-11}$
1.5908	12.0000	$5.636 \cdot 10^{-15}$
1.6108	74.0000	$-1.957 \cdot 10^{-15}$
1.6308	58.0000	$-8.811 \cdot 10^{-13}$
1.6508	3.0000	0
1.6708	4.0000	0

З результатів видно, що корінь в обраному діапазоні початкових точок віддалений від очікуваного значення від 2-х до 10000 періодів.

Використання ранжованих змінних разом з функцією **root** в версіях пакета до *MathCAD Prime* теж потребує обережності. Наприклад, для первинного та модифікованого рівнянь (3.1) проведемо аналіз залежності знайденого кореня від початкової точки. Початкову точку визначимо у вигляді ранжованої змінної в діапазоні від -1 до 0.2 з кроком 0.1:

start := −1 stop := 0.2 dx := 0.1

zz := start , start + dx .. stop

zz =	root(y(zz) ,zz)	root(s(zz) ,zz)
-1	-10.545	-10.545
-0.9	-10.545	-10.545
-0.8	-10.545	-10.545
-0.7	-10.545	-10.545
-0.6	-10.545	-10.545
-0.5	-10.545	-10.545
-0.4	-10.545	-10.545
-0.3	-10.545	-10.545
-0.2	-10.545	-10.545
-0.1	-10.545	-10.545
0	-10.545	0
0.1	0.272-2.16i	0.1
0.2	-10.545	-10.545

z0 := 0 z01 := 0.1

root(s(z01) ,z01) = 0.1

root(y(z01) ,z01) = 0.272 − 2.16i

root(s(z0) ,z0) = 0.272 + 2.16i

root(y(z0) ,z0) = 0.272 + 2.16i

З результатів видно, що функція **root** для первинного рівняння змінює корінь з -10.545 на 0.272-2.26i при початковій точці 0.1. Перевірка без ранжованої змінної показує, що відповідь змінюється на комплексну 0.272+2.26i ще й в початковій точці 0. Обидві відповіді є коренями рівняння. Для модифікованого рівняння видно зміну відповіді в точці 0 на 0 та в точці 0.1 на 0.1. Обидві відповіді є невірними. Перевірка без ранжованої змінної показує, що в точці 0 відповіддю є комплексний корінь 0.272+2.26i, а не 0.

Послідовним застосуванням функції **root** можна знайти всі корені алгебраїчного рівняння. Для цього слід використати представлення рівняння у вигляді добутку

$$f(x) = (x - x_1)(x - x_2)...(x - x_n) \ ,$$

де x_i – корені рівняння, n – ступінь полінома.

Перший нуль шукається для функції $h(x)=f(x)$ ступеню n, подальші – для функції ступеню n-1

$$h(x) = \frac{h(x)}{x - x_i},$$

де x_i – корінь рівняння, який знайдено на попередньому кроці.

Засоби розв'язання систем рівнянь

Система лінійних рівнянь має наступний вигляд:

$$\begin{aligned} &a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1, \\ &a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = b_2, \\ &\ldots \\ &a_{n,1}x_1 + a_{n,2}x_n + \ldots + a_{n,n}x_n = b_n. \end{aligned}$$

де $a_{1,1}$, $a_{1,2}$, …, $a_{n,n}$ – коефіцієнти лівої частини системи, які можна поєднати в матрицю A, x_1, x_2, …, x_n – невідомі, які можна поєднати в вектор X, b_1, b_2, …, b_n – вільні члени: вектор B. Система може бути представлена в матричному вигляді $A \cdot X = B$, де A – матриця коефіцієнтів, X – шуканий вектор невідомих, B – вектор вільних членів. Тоді розв'язання системи може бути записано через зворотну матрицю A як

$$\mathbf{X = A^{-1} \cdot B}.$$

Матричний метод розв'язання системи лінійних рівнянь (СЛР) може застосовуватися тільки до визначених систем, в яких кількість рівнянь співпадає з кількістю невідомих.

Функція **`lsolve`** призначена для розв'язання СЛАР різновидом *метода Гауса* – методом *трикутних LU матриць.*

Точність розв'язання СЛР не залежить від значення змінної TOL та знижується зі збільшенням розмірності системи.

В разі отримання невірної відповіді слід перевірити коректність СЛР.

Аргументами функції **`lsolve(A,B)`** є матриця A коефіцієнтів системи та вектор-стовпець B правих частин системи.

Наприклад:

$$A := \begin{pmatrix} 2.3 & 6.7 & 9.8 \\ 24 & 42 & 1 \\ 6 & 0 & 12 \end{pmatrix} \qquad B := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$x1 := A^{-1} \cdot B = \begin{pmatrix} 0.272 \\ -0.111 \\ 0.114 \end{pmatrix} \qquad x01 := \text{lsolve}(A, B) = \begin{pmatrix} 0.272 \\ -0.111 \\ 0.114 \end{pmatrix} \qquad A \cdot x01 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$A := \begin{pmatrix} 2.3 & 9.8 \\ 24 & 1 \\ 6 & 12 \end{pmatrix} \quad x2 := A^{-1} \cdot B = \quad x02 := \text{lsolve}(A, B) = \begin{pmatrix} 0.08 \\ 0.159 \end{pmatrix} \quad A \cdot x02 = \begin{pmatrix} 1.742 \\ 2.082 \\ 2.388 \end{pmatrix}$$

$$A := \begin{pmatrix} 2.3 & 6.7 & 2.8 & 9.8 \\ 24 & 42 & 7 & 1 \\ 6 & 0 & 8 & 12 \end{pmatrix} \qquad x3 := A^{-1} \cdot B = \qquad x03 := \text{lsolve}(A, B) = \begin{pmatrix} 0.156 \\ -0.069 \\ 0.155 \\ 0.068 \end{pmatrix}$$

$$A \cdot x03 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad +$$

Для визначених систем матричний метод та функція забезпечують вірне розв’язання. Для систем з неквадратною матрицею матричний метод не дозволяє знайти розв’язання.

Навіть для визначених систем методи без попередження можуть видати невірну відповідь. Наприклад:

$$A := \begin{pmatrix} 1 & 2 & -3.2 & 3.4 \\ 2 & 1 & -0.2 & -0.4 \\ 3 & 1.4 & -0.3 & -0.6 \\ -5 & -10 & 16 & -17 \end{pmatrix} \qquad B := \begin{pmatrix} 1.2 \\ 1.8 \\ 2.7 \\ 6 \end{pmatrix}$$

$$|A| = 0 \qquad \text{lsolve}(A, B) = \begin{pmatrix} -1.708 \times 10^{15} \\ 26.457 \\ -6.276 \times 10^{15} \\ -5.404 \times 10^{15} \end{pmatrix} \qquad A^{-1} \cdot B = \begin{pmatrix} -1.708 \times 10^{15} \\ 24 \\ -6.276 \times 10^{15} \\ -5.404 \times 10^{15} \end{pmatrix}$$

Обидва методи видали однакові відповіді. Обидві не є вірними, бо система немає коренів, так як визначник матриці дорівнює нулю.

Для розв'язання систем трансцендентних рівнянь різговидом методу *Ньютона - методом Левенберга-Маркардта.* призначено обчислювальний блок **given/find(x₁ <,x₂,...>)/minerr(x₁ <,x₂,...>),** між котрими записується необхідна система. В якості аргументів зазначаються імена змінних, відносно яких потребується провести розв'язання.

Обов'язковим є попереднє визначення первинних значень змінних, для яких проводиться обрахунок системи.

Блок заходить ТІЛЬКИ ДІЙСНІ значення розв'язання системи.

Блоки не можуть бути вкладеними один в один.

Всередині блока не може застосовуватися оператор прирівнювання (:=), ранжована змінна.

Рівняння системи записуються через знак логічного «жирного» рівняння (Ctrl + "=") «=».

Якщо в системі замість знаків логічного рівняння стоять знаки не рівняння (<, >), то блоковий оператор розв'яже систему нерівностей. Відповідь буде значенням координат однієї точки, яка задовольняє умовам системи. Для отримання координат іншої точки треба змінити початкові умови.

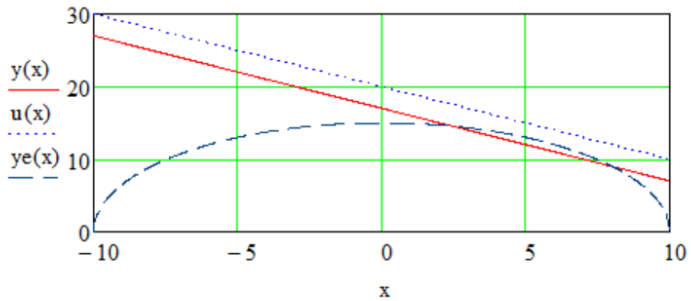
Наприклад, дослідимо точки перетину еліпса та двох прямих

$$\frac{x^2}{4}+\frac{y^2}{9}=25 \quad y(x)=17-x \quad u(x)=20-x\,.$$

Визначимо функції користувача та побудуємо графік.

$$f(x,y):=\frac{x^2}{4}+\frac{y^2}{9}-25 \qquad y(x):=17-x \qquad u(x):=20-x$$

$$ye(x):=3\cdot\sqrt{25-\frac{x^2}{4}}$$



З графічного аналізу видно, що пряма y(x) має дві точки перетину, в околі x=2.5 та x=8. Друга пряма u(x) немає точок перетину з еліпсом.

Визначимо початкові значення та застосуємо блок **given-find**:

$x := 4 \quad y := 10$

Given $\quad f(x,y) = 0 \quad y + x = 17 \quad z := \text{Find}(x,y) = \begin{pmatrix} 8 \\ 9 \end{pmatrix}$

Given $\quad f(x,y) = 0 \quad y + x = 20 \quad z := \text{Find}(x,y) =$

Для початкових значень x=2, y=10 блок **given-find** визначив корінь для першої системи та не знайшов корінь для другої.

Для знаходження другої точки перетину слід або змінити початкові значення, або додати логічну умову всередину блоку. Це може бути область допустимих значень коренів:

$x := 4 \quad y := 10$

Given $\quad f(x,y) = 0 \quad y + x = 17 \quad x < 5 \quad z := \text{Find}(x,y) = \begin{pmatrix} 2.462 \\ 14.538 \end{pmatrix}$

$x := 2$

Given $\quad f(x,y) = 0 \quad y + x = 17 \quad z := \text{Find}(x,y) = \begin{pmatrix} 2.462 \\ 14.538 \end{pmatrix}$

В разі необхідності знайдення не точних, а наближених значень коренів, слід застосувати в блоці з **given** функцію **minerr**.

Функція **minerr** відноситься до функцій оптимізації. Заходить найкраще наближення за допомогою МНК.

Знайденікорені слід ОБОВ'ЯЗКОВО перевіряти. Для цього призначено вбудована змінна **ERR**, яка містить середньоквадратичне значення (RMSE) похибки.

Наприклад, для функції u(x) застосування **minerr** дає:

$x := 2$

Given $\quad f(x,y) = 0 \quad y + x = 20 \quad z := \text{Minerr}(x,y) = \begin{pmatrix} 5.616 \\ 12.635 \end{pmatrix}$

$\text{ERR} = 1.857$

Застосування **minerr** може допомогти, коли значення параметрів системи не дають застосувати функцію **find**. Наприклад, рівняння параболоїду може бути записано як

$$kx^2 + y^2 = 0 \ .$$

При невеликих значеннях коефіцієнта **k** функції **find** достатньо для визначення коренів. Для великих значень коефіцієнта к функція **find** не справляється з розв'язанням, а функція **minerr** дає відповідь:

$k := 10^6$ $\quad x := 1$ $\quad y := 1$

given $\quad k \cdot x^2 + y^2 = 0$ $\quad$ find(x ,y) =

Given $\quad k \cdot x^2 + y^2 = 0$ $\quad$ minerr(x ,y) = $\begin{pmatrix} -4.912 \times 10^{-4} \\ 7.091 \times 10^{-4} \end{pmatrix}$

Блок **given** може застосовуватися також для розв'язання одного рівняння.

Можуть трапитися випадки, коли при невдалому початковому наближенні блок **given** відповіді не дає.

Наприклад, для рівняння (3.1) застосування блока дає

$y(x) := 2 \cdot x^3 + 20 \cdot x^2 - 2 \cdot x + 100$

$z := -5$ given $\quad y(z) = 0 \quad$ find(z) =

$z := -10$ Given $\quad y(z) = 0 \quad$ Find(z) = −10.545

Символьні засоби розв'язання рівнянь та систем рівнянь

Символьними перетвореннями можна отримати розв'язання в аналітичному або в чисельному вигляді. Рівняння можуть бути параметричними.

Отриманий символьний результат може мати неочікуваний вигляд. В такому випадку можуть потребуватися додаткових дій для надання йому потрібного вигляду.

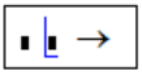
Іноді символьна відповідь відображається у вигляді спеціальних математичних функцій, деякі з яких можуть бути оброблені в пакеті тільки символьно.

Для отримання символьної відповіді достатньо застосувати оператор символьного виведення результату «→». Вводиться оператор кнопкою з символьної панелі або комбінацією клавіш “Ctrl+.”.

Деякі символьні перетворення потребують ведення ключового слова.

Ключові слова вводяться:

- послідовно кнопкою шаблона дії з ключовим словом символьної панелі



з наступним заповненням першого маркера вихідними даними, другого – написом ключового слова дії, введенням шаблона комбінацією клавіш “Ctrl+>” з наступним заповненням маркерів шаблона,

- натисканням клавіші відповідної дії символьної панелі з наступним введенням в поле маркера вихідних даних.

Оператори символьних перетворень можуть застосовуватися послідовно. Корисним є послідовне застосування основного оператора дії з уточненням спрощення **simplify**, отримання чисельного результату **float**, умови до типу результату **assume**. Для отримання додаткового кроку послідовності слід після введення першого символьного оператора натиснути одночасно «Ctrl+Shift+.» або просто після натискання відповідної кнопки першого символьного оператора з символьної панелі натиснути другу.

Обчислювальний блок **given-find** в комбінації з оператором символьного виведення результату «→» може застосовуватися для символьного застосування для будь яких рівнянь та систем рівнянь.

Наприклад,

given

$$x^2 + y^2 = 1$$

$$\text{find}(y) \rightarrow \left(\sqrt{x-1}\cdot\sqrt{x+1}\cdot i \quad -\sqrt{x-1}\cdot\sqrt{x+1}\cdot i\right)$$

Недоліком подібного способу є те, що неможливо додати умови розв'язання або визначити послідовність символьних дій.

Функція символьного розв'язання

▪ solve →

призначена для символьного розв'язання рівнянь та систем рівнянь.

В поле маркера оператора **solve** вводиться вираз рівняння або вектор функцій системи рівнянь. Для функцій, які записані в нормальному вигляді, прирівнювання до нуля ставити не потрібно. В інших випадках запис має містити знак «жирного» рівняння та значення правої частини рівняння. За необхідності після ключового слова через кому позначаються імена змінних, відносно яких потрібно знайти розв'язок.

Наприклад,

$$x^3 + 11x^2 + 3x - 135 = 0 \text{ solve} \rightarrow \begin{pmatrix} -9 \\ -5 \\ 3 \end{pmatrix} \quad x - a + 5 \text{ solve},x \rightarrow a - 5$$

$$\begin{pmatrix} x^2 + y^2 = 1 \\ y - x^2 = 1 \end{pmatrix} \text{solve},x,y \rightarrow \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ i\cdot 3^{\frac{1}{2}} & -2 \\ -i\cdot 3^{\frac{1}{2}} & -2 \end{pmatrix}$$

Оператор **solve** для шостої та вищих ступенів видає чисельне розв'язання.

Для тригонометричних рівнянь оператор **solve** за замовчанням видає один корінь. Наприклад,

$$\sin(a\cdot x) \text{ solve},x \rightarrow 0$$

Блок **find** видає корені системи рівнянь у вигляді вектора-рядка, оператор **solve** – у вигляді вектора-стовпця.

Не завжди оператори символьного розв'язання дають відповідь за один крок. Для досягнення мети слід спробувати кілька варіантів дій.

Наприклад, теоретичне розв’язання рівняння $x^2+y^2=1$ відносно y має наступний вигляд

$$y = \pm\sqrt{1 - x^2}$$

Безпосереднє символьне розв’язання виглядає як

$$x^2 + y^2 = 1 \text{ solve, y} \rightarrow \begin{pmatrix} \sqrt{x - 1}\cdot\sqrt{x + 1}\cdot i \\ -\sqrt{x - 1}\cdot\sqrt{x + 1}\cdot i \end{pmatrix}$$

.

Відповідь вірна, але запис незручний із використанням уявної одиниці. Спроба додаткового кроку спрощення не змінює вигляд відповіді:

$$x^2 + y^2 = 1 \left| \begin{matrix} \text{solve, y} \\ \text{simplify} \end{matrix} \right. \rightarrow \begin{pmatrix} \sqrt{x - 1}\cdot\sqrt{x + 1}\cdot i \\ -\sqrt{x - 1}\cdot\sqrt{x + 1}\cdot i \end{pmatrix}$$

Тільки встановлення додаткового модифікатора дає звичний вигляд

$$x^2 + y^2 = 1 \left| \begin{matrix} \text{solve, y, real} \\ \text{simplify} \end{matrix} \right. \rightarrow \begin{pmatrix} \sqrt{1 - x^2} & 0 \\ -\sqrt{1 - x^2} & 0 \end{pmatrix}$$

.

Для рівняння (3.1) символьне розв’язання виглядає дуже громіздким та незручним. Ані спрощення, ані модифікатори вигляд відповіді не змінюють. До звичного результату призводить тільки застосування виведення в чисельному вигляді

$$2\cdot x^3 + 20\cdot x^2 - 2\cdot x + 100 \left| \begin{matrix} \text{solve, x, real} \\ \text{float} \end{matrix} \right. \rightarrow \begin{pmatrix} -10.544528449924184116 & 0 \\ 0.27226422496209205808 - 2.1604786330404814207i & 0 \\ 0.27226422496209205808 + 2.1604786330404814207i & 0 \end{pmatrix}$$

В літературі рекомендується прості рівняння та системи розв’язувати символьно, більш складні – чисельно.

3.2. Завдання до виконання

Завдання 3.1.

а) Дослідити залежність знаходження значення кореня рівняння (3.1) функцією **root** від початкової точки в діапазоні від -1 до 0.2 з кроком 0.2 визначивши початкову точку вектором.

б) Розв'язати первинне та модифіковане рівняння (3.1) методом дихотомії. Дослідити вплив початкових точок та масштабного коефіцієнта на пошук кореня.

Завдання 3.2.

Порівняти точність розв'язання алгебраїчного рівняння $\mathbf{15x^4-6x^3+4x^2-12x-10=0}$ методом Лагера, супроводжуючої матриці, функцією **root.** Еталонні корені рівняння

1.1709543213058003802

-0.11333921276773893441+1.0164632705947297274i

-0.54427589577032251142

-0.11333921276773893441-1.0164632705947297274i

Завдання 3.3

Порівняти точність градієнтних методів функції **find** для системи, яка має корені [3 2 1].

$$\begin{cases} x^2 + y^2 - z^2 = 12 \\ x + y + z = 6 \\ xyz = 6 \end{cases}$$

Завдання 3.4

Отримати координати точок перетину

- параболи $y=x^2$ та прямої $y=8+3x$
- кола $(x+1)^2+(y+1)^2=5.5$ та прямої $x+y=0.95$

Завдання 3.5

- Отримати символьне розв'язання алгебраїчного параметричного рівняння $ax^4-x^3+a^2x-a=0$. Визначити значення при параметрі a=1. Порівняти зі значеннями чисельного методу.
- Отримати символьне розв'язання рівняння

$$\frac{1}{\sqrt{x+2}+x}=\sqrt{x+2}+1\ .$$

Порівняти зі значеннями чисельного методу.

Завдання 3.6

Отримати розв'язання лінійної системи алгебраїчних рівнянь

$$\begin{cases}2x-3y+z=1.5\\1.5y-x+3z=6\\7z+5y-1.5x=7\end{cases}$$

матричним методом, UL-методом, функцією **find**, символьним методом. Порівняти результати, якщо точне значення

(-3.935064935064935065 -2.480519480519480520 1.928571428571428571).

Завдання 3.7

Отримати розв'язання системи параметричних рівнянь. Визначити значення при параметрах a=1, b=2 c=3.

$$\begin{cases}x+y+z=0\\cx+ay+bz=0\\(x+b)^2+(y+c)^2+(z+a)^2=a^2+b^2+c^2\end{cases}$$

Завдання 3.8

Отримати чисельне значення розв'язку рівняння

$$\frac{2x+10}{4}=\frac{8}{2^{x-2}}$$

чисельними та символьним методами.

Практикум 4. Розв'язання рівнянь та систем рівнянь засобами Matlab

Метою практикуму є вивчення методів, особливостей розв'язання рівнянь та систем рівнянь в СКМ. Набуття навичок в застосуванні засобів розв'язання алгебраїчних та нелінійних рівнянь, систем рівнянь в СКМ «Matlab».

4.1. Теоретичні відомості

Чисельні засоби

Функція **fzero** призначена для чисельного знаходження коренів нелінійних рівнянь за алгоритмом *Декера*, який є комбінацією *методів бісекції, січних та зворотної інтерполяції.*

Функція **roots** призначена для чисельного знаходження коренів алгебраїних рівнянь методом *власних значень супроводжуючої матриці*.

Функція
[x,fval,exitflag,output]=fzero(fun,x0,<[x1x2],opt,...
P1,P2,...>) повертає уточнене значення **x**, при якому досягається нуль функції **fun**. Процес пошуку кореня складається з двох етапів. На першому етапі в околі точки **x0** визначається діапазон, в якому знаходиться корінь. На другому етапі проводиться уточнення значення кореню всередині цього діапазону.

fun – функція рівняння. Є обов'язковим аргументом. **x0** – вихідне значення початку пошуку. Може бути числом або іменем наперед визначеної змінної. Обов'язкове поле. **[x1 x2]** – опціональний вектор інтервалу пошуку кореня. Вводиться замість значення x0. Скорочує час пошуку кореня. **opt** – опціональна структура з додатковими умовами застосування функції.

Функція **fun** при викликах може задаватися:

1) покажчиком на в m-файл функцію користувача, в якому описано рівняння. Наприклад,

```
function ff = test(v)
```

```
ff = sin(pi*v).
```

Виклик функції:

```
x= fzero (@test,0,1)  %@test– покажчик на функцію
```

2) рядком з іменем функції з m-файлу функції користувача. Наприклад,

```
x= fzero ('test',0,1)  або    x= fzero ('test(x)',0,1)
```

Формула, яка обрана в апострофи, може містити тільки змінну «x». Використання інших імен викликає помилку.

3) за допомогою «inline» функції. Наприклад,

```
f = inline('1./((x - 0.3)^2 + 0.01) - 6'); % опис
```

Виклик: **z=fzero(f,1)**

4) посилання на анонімну функцію:

```
ms=@(x)3*cos(x)
fzero(@ms,1)
```

5) сама анонімна функція: **fzero(@(x)sin(x*x), 1);**

Структура **opt** визначає умови проведення обчислювальних дій. Керування структурою **opt** здійснюється командою **optimset**. Структура в першу чергу призначена для керування функціями оптимізації з додатку **Optimization Toolbox** та містить біля 50-ти полів.

Аргументи структури **opt** задаються попарно:

```
options = optimset(...,назва поля,  значення,...).
```

Функція **fzero** використовує наступні поля структури:

Display – 'off' без виведення інформації про хід пошуку/ **'iter'–** виведення інформації по кожній ітерації / **'final'–** виведення результуючою інформації/ **'notify'** (default) – виведення інформації в разі виникнення проблем.

FunValCheck – **'on'** здійснення перевірки рівняння на комплексні та невизначені значення / **'off'** (the default).

OutputFcn – посилання на функцію користувача, яка буде виконуватися на кожній ітерації пошуку кореня.

PlotFcns – масив комірок посилань (cell array) які виводять на екран зображення процесу пошуку кореня. За замовчанням – порожній масив []. Визначені значення елементів масиву: **@optimplotx** – графік точок, **@optimplotfval** – графік значень функції рівняння на етапі уточнення кореня.

TolX – визначення точності по x. За замовчанням розв'язання проводиться з точністю 10^{-6}.

P1,P2,… – опціональні додаткові параметри (до десяти), які потрібно передати в функцію рівняння **fun(x,P1,P2,…).**

fval – опціональне значення функції рівняння в точці **x**.

exitflag – чисельний код результату дій: 1 – знайдено корінь, -1 – алгоритм перервано, -3 – під час дії отримано невизначене (0/0) чи нескінченне значення, -4 – виникло комплексне значення при діях всередині інтервалу **[x1 x2]** , -5 – точка розриву, -6 – не виявлено зміни знаку функції.

output – структура з додатковою інформацією в наступних полях:

algorithm – 'bisection, interpolation',

funcCount – кількість обрахувань функції,

intervaliterations – кількість ітерацій для знаходження інтервалу пошуку,

iterations – кількість ітерацій пошуку кореня.

Наприклад, звернення до функції

```
>> [x f exi opt]=fzero(inline('x^3-1'),2)
```

повертає корінь x, значення функції f, прапор exi, структуру пояснень значеннями кроків ітерацій, кількістю обчислень функції, застосованим алгоритмом – opt:

```
x =      1
f =      0
exi =      1
```

```
    opt =
        intervaliterations: 10
                iterations: 8
                 funcCount: 28
                 algorithm: 'bisection, interpolation'
 message: 'Zero found in the interval [0.72, 2.9051]'
```

Наприклад, варіант з використанням рядка з іменем функції користувача:

```
function [ y ] = mzero(a, x )
y = a/((x - 0.3)^2 + 0.01)-6;
end
fzero('mzero(1,x)',1)
```

варіант з використанням анонімної функції та функцією користувача:

```
a=1;
fzero(@(x) mzero(a,x,1)
```

варіант з вкладеною функцією:

```
function y = myzero(x,a,x0)
y = fzero(@(x) mzro(a,x),x0);
function y = mzro(x,a) %вкладена функція
y = a/((x - 0.3)^2 + 0.01)-6;
end % end mzro
 end  % end myzero
```

Використання **fzero** потребує попереднього аналізу поведінки функції рівняння.

Наприклад, рівняння, яке має комплексні корені в MathCAD розв'язується простим викликом функції **root**. Виклик функції **fzero** скоріш за все видасть повідомлення про помилку.

```
Exiting fzero: aborting search for an interval containing a sign change
    because NaN or Inf function value encountered during search.
(Function value at -1.7162e+154 is Inf.)
Check function or try again with a different starting value.
```

Розв’язання може бути отримане тільки викликом функції з аргументом діапазону пошуку з комплексними значеннями такими, щоб функція мала на кінцях діапазону дійсне значення.

В Matlab існує багато засобів для роботи з СЛАР. Даними для цих засобів слугують матриця коефіцієнтів рівняння А та вектор вільних членів В.

Найбільш простим є застомування "/" правого ділення. Запис **X=A/B** дає розв’язок СЛАР **XA=B**, де В та Х – вектори-рядки ;

Якщо **A** є квадратною матрицею, то **X=A/B** розв’язує СЛАР *методом Гауса*.

Matlab має спеціалізовані засоби для розв’язання особливих СЛАР. Засоби базуються на *методі найменших квадратів*.

Точні методи

• **X =lscov(A,B,V)** – розв’язання СЛАР типу **A*X=B+e**, де вектор **e** має коваріаційну матрицю **V**. Матриця **A** повинна мати розмір **m-X-n**, де **m>n**. Критерієм мінімізації слугує вираз **(AX-b)'*inv(V)*(AX-b).**

Розв’язок має вигляд **X= inv(A'* inv(V)*A)*A'*inv(V)*B**. Інвертування коваріаційної матриці не проводиться. Опціонально можна отримати в векторі результату крім Х значення стандартної похибки

```
mse=B'*(inv(V)-inv(V)*A*inv(A'*inv(V)*A)*A'*inv(V))...
*B./(m-n)
dX = sqrt(diag(inv(A'*inv(V)*A)*mse))
```

• **lsqr (A, B, tol)** – розв’язання системи СЛАР з розрідженою матрицею **A.** Критерієм для функцій **lsqr, bicg, pcg, cgs** слугує вираз **norm(B-Ax)/norm(B) < tol** (за замовчанням 1e-6).

• **bicg(A, B)** – розв’язує СЛАР розміром n-X-n *двонаправленим методом спряжених градієнтів*. Функція є швидшою за **lsqr**. Різновидом функції **bicg** є функція **bicgstab**, яка реалізує *стійкий двонапрвлений метод спряжених градієнтів.*

Для випадку, коли необхідно провести розв’язання для однієї n-X-n матриці А з декількома варіантами вектора В можна скористатися

розкладанням Холецького функцією **[R,p,S] = chol(A).** Для розрідженої матриці A повертається матриця перестановок S. R – верхня трикутна матриці така, що **R'*R=S'*A*S**, для p=0. Для інших значень p, R є верхньою трикутною матрицею розміру q X n. Для діапазону вигляду L перші q рядків та q стовпців добутку **R'*R= S'*A*S**. Застосування дає виграш приблизно в n разів. Система **A*x=B** переводиться в систему **R'*R*x=B** з розв'язком **x=R\(R'\B).**

Ітераційні методи

• **pcg(A,B)** розв'язує СЛАР для квадратної додатно визначеної матриці A *методом спряжених градієнтів*.

• **cgs(A,B)** реалізує *квадратичний метод спряжених градієнтів*.

Безпосередньо для розв'язання системи нелінійних рівнянь $F(\vec{x}) = 0$ призначено функцію **fsolve**.

Можливо застосовування функцій **fminsearch**, **fminunc**, **lsqnonlin** з додатку **Optimization toolbox** Matlab.

Синтаксис всіх наведених функцій має однаковий вигляд:

[x,<fval>,<exitflag>,<output>,<jacobian>]=

MLABFUN(fun,x0,<options>,...),

де **x** – вектор коренів системи, **fun** – функція опису системи (цільової функції), **x0** – вектор початкових значень. Точка початку пошуку. **options** – опціональна структура опису умов роботи функції **fsolve.**

Функція **fun** приймає аргумент – вектор значень x, повертає результат – вектор значень рівнянь системи. В разі застосування методів розв'язання з використанням аналітично визначеного Якобіану, повинна містити розрахунок останнього. Використання аналітично визначеного Якобіану проводиться записом опції в параметрі **options=optimset('Jacobian','on').** В такому випадку функція опису системи повинна видавати результат у вигляді вектора з двох елементів. Перший елемент – вектор значень рівнянь, другий – матриця значень Якобіану.

Додатково до полів, які використовуються функцією **fzero**, для **fsolve** доступні наступні поля:

Algorithm – визначає метод розв'язання: *'trust-region-dogleg' «собачої ноги, погоні»* (за замовчанням), *'trust-region-reflective'* (*модифікований ньютонівський з відбиттям*), *'levenberg-marquardt'* (*модифікований ньютонівський – Левенберга-Маркуардта*). За замовчанням параметр останнього метода $\lambda = 0.01$. Для зміни, наприклад, на 0.5 потрібно визначити комірку **{'levenberg-marquardt',0.05}.** Тільки останній метод може бути застосований для систем з розміром не n-X-n.

DerivativeCheck – перевірка значень чисельних похідних: 'on'/ 'off' (за замовчанням).

Diagnostics – виведення службової інформації про поведінку рівняння: 'on'/ 'off' (за замовчанням).

DiffMaxChange – максимальне допустиме значення градієнта (Inf за замовчанням).

DiffMinChange – мінімальне допустиме за амплітудою значення градієнта (0 за замовчанням).

FinDiffRelStep – вектор кроку в чисельній похідній. Для центральної похідної **delta = v.*max(abs(x),TypicalX).** Для однобічної похідної **delta= v.*sign(x).*max(abs(x),TypicalX).**

FinDiffType – тип чисельної похідної: **'forward'** (однобічна вперед, за замовчанням), **'central'** (центральна).

Jacobian – спосіб обчислення матриці Якобі: 'on' – аналітичне в функції користувача, 'off' (чисельно, за замовчанням).

MaxFunEvals – максимальна кількість обчислень функції. За замовчанням 100*кількість змінних.

MaxIter – максимальна кількість ітерацій. За замовчанням 400.

PlotFcns – додатково доступні наступні графіки: @**optimplotfunccount** – кількість обчислень функції, @**optimplotresnorm** – норма (модуль) нев'язки, @**optimplotstepsize** – крок.

TolFun – порогове значення функції рівняння. За замовчанням 1e-6.

TypicalX – *Trust-region-dogleg* алгоритм присвоює це значення елементам головної діагоналі масштабуючої матриці. За замовчанням дорівняю 1.

fval – опціональний результат. Вектор значень рівнянь системи в точці **x**.

exitflag – опціональний результат. Код ознаки закінчення дії функції:

1 Значення градієнта менше точності;

2 Крок X менше точності;

3 Крок зміни цільової функції менше точності;

4 Зсув в визначеному напрямку менший за точність.

0 Досягнуто максимальну кількість ітерацій;

-1 Алгоритм завершено по значенню функції;

-2 Алгоритм намагається визначити точку, яка не є коренем.

-3 Радіус області замалий або параметр регуляції завеликий.

Для *ньютонівського метода з відбиттям* додатково доступні поля:

JacobMult – посилання на функцію множення якобіану.

JacobPattern – матриця розряження якобіану для чисельного диференціювання.

MaxPCGIter – максимальна кількість ітерацій спряжіння градієнтів.

TolPCG – порогове значення спряжіння градієнтів. За замовчанням 0.1.

output – опціональний результат. Структура з описом ходу розв’язання системи.

output.algorithm – використаний алгоритм;

output.funcCount – кількість оцінок функції;

output.iterations – кількість ітерацій.

jacobian – опціональний результат. Матриця Якобі перших похідних рівнянь системи.

Наприклад, розв'язання системи

$$2x_1 - x_2 - e^{-x_1} = 0$$
$$2x_2 - x_1 - e^{-x_2} = 0$$

з параметрами за замовчанням (метод – «dogleg», точність по функції та кореню – 1.e-6) з початковою точкою (-5, 5) може виглядати наступним чином:

```
function F=myfun11(x)
F=[2*x(1)-x(2)-exp(-x(1));-x(1)+2*x(2)-exp(-x(2))];
```

виклик:

```
[x,fval]=fsolve(@myfun, [-5; 5])
```

відповідь:

```
Equation solved.fsolve completed because the vector
of function values is near  zeroas measured by the
default value of the function tolerance, and the
problem appears regular as measured by the gradient.
x =     0.5671     0.5671
fval =   1.0e-006 *        -0.4059       -0.4059
```

Символьні засоби

Функція **solve** видає символьну відповідь при можливості та чисельну, якщо не може знайти аналітичний розв'язок.

```
S = solve(expr<,var,Name,Value>)
```

expr – вираз символьного типу або рядок. В разі відсутності знаку рівняння в виразі, вважається , що рівняння записано в нормальному вигляді. Для систем рівнянь кожне записується через кому.

Наприклад,

```
solve('x+1=0')   solve('x+1')
```

a=sym(‘x+1’); solve(a); solve('x+y=10','x*y=2')

S – результат символьного типу. В разі напису одного імені – змінна чи структура. В разі напису вектором, кожен елемент вектора стає змінною символьного типу. Наприклад,

для запису **s=solve(‘x-1+y=0’)** змінна s приймає значення **1-y,**

для запису **s= solve('x+y=10','x*y=2')** s стає структурою з полями **s.x=(√23+5 5-√23)** та **s.y=(5-√23 √23+5),**

для запису **[s a]=solve('x+y=10','x*y=2')** результатом є два вектори символьного типу **s=[√23+5 5-√23]** та **a=[5-√23 √23+5]**

var – ім’я змінної символьного (визначеного функціями **sym**, **syms**, **symvar**) або текстового типу (забране в апострофи). За замовчанням розв’язування йде відносно змінної «x» для рівнянь та «x, y» – для систем рівнянь. Функція **solve** перед своїм виконанням викликає функцію **symvar**, яка автоматично визначає змінні в виразі. В разі необхідності змінну, відносно якої потрібно знайти розв’язання, можна визначити вручну безпосередньо. Наприклад,

>> solve('x+y+a') ans = - a - y

>> solve('x+y+a','a') ans =- x - y

Name, Value – послідовність пар імені властивості та значення властивості для керування процесом обчислення. Імена забираються в апострофи.

MaxDegree – максимальний ступінь алгебраїчних рівнянь, для яких визначається символьна відповідь (1 - 5). За замовчанням 3.

IgnoreAnalyticConstraints – ігнорування математичних обмежень (**false/true**). Наприклад, діапазону значень під квадратним коренем. За замовчанням **false.**

Наприклад, розв'язання логарифмічного рівняння проводиться без спрощення виразу

```
solve(log(x*y) - log(y) - 5, 'x')
ans = exp(log(y) + 5)/y
```

Ігнорування обмежень дозволяє спростити відповідь:

```
solve(log(x*y)-log(y)- 5, x,...
'IgnoreAnalyticConstraints', true)
ans =exp(5)
```

IgnoreProperties – ігнорування сумісності виразу з типом змінної (**false/true**). За замовчанням **false.**

Наприклад, для додатної змінної функція видасть тільки один додатний корінь

```
syms x positive;
solve(x^2 + 5*x - 6, x)
ans = 1
```

Ігнорування типу змінної дозволяє отримати обидва корені:

```
solve(x^2 + 5*x - 6, x, 'IgnoreProperties', true)
ans =   1  -6
```

PrincipalValue – керування кількістю відповідей (**false/true**). За замовчанням **false.** Значення **True** призводить до виведення тільки одного значення.

Real – виведення тільки дійсних відповідей (**false/true**). За замовчанням **false.**

4.2. Завдання до виконання

Завдання 4.1. Дослідити вплив початкової точки для функцій **`fzero`**, **`fsolve`**

- в діапазоні від -1 до 0.2 з кроком 0.1 на розв'язок рівняння (3.1).
- в діапазоні від $\frac{\pi}{2}-0.1$ до $\frac{\pi}{2}+0.1$ з кроком 0.02 для рівняння sin(x)=0

Результат оформити у вигляді таблиці з рядками: «x0=XXX x=XXX». Порівняти результати, отримані в MathCAD та Matlab.

Завдання 4.2. Розв'язати рівняння $x^2+0.0001=0$ та $1/x=0$ функцією **`fzero`**. Розв'язати рівняння (3.1) функцією **`root`**.

Завдання 4.3. Порівняти результати розв'язання СЛАР (стор.44) засобами роботи з СЛАР Matlab (/, \, lscov, lsqr, bicg, chol, pcg)

Завдання 4.4. Знайти точки перетину еліпса та двох прямих

$$\frac{x^2}{4}+\frac{y^2}{9}=25 \quad y(x)=17-x \quad u(x)=20-x\,.$$

Завдання 4.5. Порівняти точність та швидкість методів функції **`fsolve`** *'trust-region-dogleg' «собачої ноги, погоні»* (за замовчанням), 'trust-region-reflective' (*модифікований ньютонівський з відбиттям*), 'levenberg-marquardt' (*модифікований ньютонівський – Левенберга-Маркуардта*) для системи, яка має корені [3 2 1].

$$\begin{cases} x^2+y^2-z^2=12 \\ x+y+z=6 \\ xyz=6 \end{cases}$$

Завдання 4.6. Розв'язати символьно рівняння (3.1), систему завдання 3.5.

Практикум 5. Обробка табличних даних засобами MathCAD

Метою практикуму є вивчення методів, особливостей проведення глобальної поліноміальної інтерполяції, локальної поліноміальної та сплайн інтерполяції, поліноміальної та функціональної апроксимації. Набуття навичок в застосуванні засобів розв'язання цих задач в СКМ «MathCAD».

5.1. Теоретичні відомості

При розрахунках ОЕП часто припадає зіштовхуватись з необхідністю опису заданих за допомогою таблиці даних аналітичною функцією.

Для розв'язання такого роду задач використовуються методи регресійного аналізу.

В залежності від критерія, за яким співвідносяться значення аналітичної функції та вихідної сітчастої функції-таблиці, розрізняють дії інтерполяції та апроксимації.

Для інтерполяції виконується умова Лагранжа, згідно якої значення інтерполяційної функції та сітчастої функції в вузлах сітки співпадають. Тобто нев'язка у вигляді абсолютної похибки між значеннями вихідної та інтерполюючої функцій в вузлах сітки дорівнює нулю. При цьому умова не торкається значень між вузлами сітки.

Для апроксимації використовується критерій мінімізації норми вектора нев'язок в вузлах сітки. На практиці найбільш зручним вважається друга норма (евклідова) – сума квадратів значень нев'язок в вузлах сітки.

Інтерполяція

В разі, коли розраховується одна інтерполююча функція для всіх значень сітки, інтерполяція називається глобальною. Коли розраховується кілька інтерполюючих функцій, по одній кожного з виділених піддіапазонів значень сітки, говорять про локальну або «ковзаючу» інтерполяцію.

Функціональна інтерполяція, коли в якості наближаючої функції використовується інша залежність (логарифмічна, показова, тощо), призводить до необхідності складання та розв'язання системи нелінійних рівнянь.

Знаходження коефіцієнтів a_i інтерполюючого поліному зводиться до розв'язання СЛАР:

$$a_0 x_0^{\ 0} + a_1 x_0^{\ 1} + a_2 x_0^{\ 2} + \ldots + a_{N-1} x_0^{\ N-1} = y_0$$

$$a_0x_1^0+a_1x_1^1+a_2x_1^2+...+a_{N-1}x_1^{N-1}=y_1$$

$$\cdots$$

$$a_0x_{N-1}^0+a_1x_{N-1}^1+a_2x_{N-1}^2+...+a_{N-1}x_{N-1}^{N-1}=y_{N-1}$$

Або у матричному вигляді

$$X\vec{a}=\vec{y} \quad ,$$

де X – матриця Вандермонде значень координат вузлів сітки, y – вектор значень сітчастої функції, x – вектор аргументів сітчастої функції, a – вектор шуканих коефіцієнтів полінома.

Пряме розв'язання завдання інтерполяції має наступний алгоритм:

1. Визначення векторів аргументу «x» та табличної функції «y» в N вузлах **x=(x0,x1...xN-1) y=(y0,y1...yN-1).**

2. Визначення матриці коефіцієнтів рівнянь системи через вектор аргументів «x».

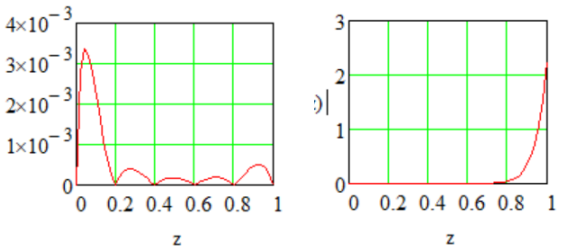
3. Знаходження вектора коефіцієнтів «a» розв'язанням СЛАР.

4. Розрахунок значення інтерполяційного полінома в визначеній точці z за виразом (5.1).

В MathCAD для розв'язання СЛАР призначено матричний метод з обертанням матриці X ($X\vec{a}=\vec{y} \quad \vec{a}=X^{-1}\cdot\vec{y}$) та функція **lsolve**.

В випадках, коли сітчаста функція визначена на великій кількості точок, інтерполяційний поліном стає занадто громіздким. Порядок СЛАР глобальної інтерполяція, похибки та час розв'язання стають завеликими навіть для сучасних СКМ.

Наприклад, для функції $y(x)=\sin(x)e^{-2\sqrt{x}}$ в діапазоні x від 0 до 1 в 6-ти точках абсолютна похибка інтерполяції з прямим розв'язанням СЛАР в матричному вигляді з обертанням матриці становить ~0.001 (рис. 5.1б), а для 26-ти точок – вже ~2 (рис. 5.1б)



а) 6 вузлів б) 26 вузлів

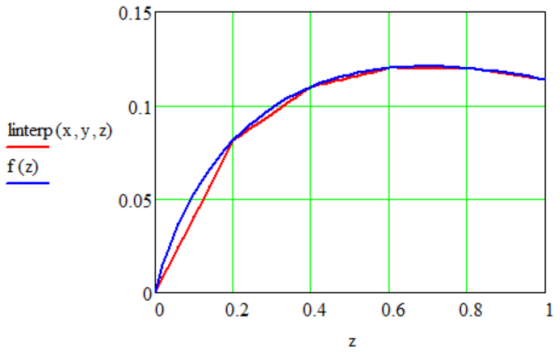
Рис. 5.1 – Похибки поліноміальної інтерполяції

Кускова лінійна інтерполяція в MathCAD реалізується функцією **`linterp(vx,vy,x)`**, коли між сусідніми вузлами сітки проводиться пряма. Аргумент «**x**» може бути числом або вектором. Повертається лінійно наближене значення функції в точках аргументу.

Наприклад, результат лінійної інтерполяції для вищенаведеної функції може мати вигляд, який показано на рис. 5.2. Середньоквадратична похибка для лінійної кускової інтерполяції складає 0.0044, а для глобальної інтерполяції 0.00098. Відносні похибки складають 3.6% та 0.8%, відповідно.

Основний недолік звичайної «ковзаючої» інтерполяції: наявність зламів в вузлах сітки , – долає сплайн інтерполяція.

Сплайн інтерполяція – «ковзаюча» лінійна, квадратична, кубічна інтерполяція з додатковими умовами щодо поведінки першої та другої похідних інтерполюючої функції. За рахунок виконання додаткових умов вона забезпечує нерозривність та плавність проходження функції.



$$\text{norm}(a) := \sqrt{\sum_{i=0}^{\text{last}(a)} \frac{(a_i)^2}{\text{length}(a)}}$$

$$lf := \text{linterp}(x, y, z)$$

$$\text{norm}(lf - f(z)) = 4.397 \times 10^{-3}$$

$$\text{norm}(fi(a, z, N) - f(z)) = 9.821 \times 10^{-4}$$

$$\frac{\text{norm}(lf - f(z))}{\text{norm}(fi(a, z, N) - f(z))} = 4.477$$

$$\frac{\text{norm}(lf - f(z))}{\max(f(z))} = 0.036 \qquad \frac{\text{norm}(fi(a, z, N) - f(z))}{\max(f(z))} = 0.008$$

Рис. 5.2 – Лінійна інтерполяція

Основною функцією розрахунку значення функції в визначеній точці «x» по відомим табличним даним «vx, vy» в MathCAD є функція

interp(vs,vx,vy,x) ,

де **vs** – допоміжний вектор коефіцієнтів, **vx, vy** – вектори аргументів та табличної функції відповідно. Для двомірних функції **vx** – матриця розміром Nx2, елементами якої є координати точок діагоналі сітки, **vy** – матриця розміром NxN зі значеннями сітчастої сітки в вузлах, **x** – точка/ вектор, в якій необхідно обрахувати значення.

Значення вектора **vs** визначає тип наближення. Для визначення вектора призначені наступні вбудовані функції:

lspline(vx,vy) – для лінійного сплайна,

pspline(vx,vy) – для квадратичного сплайна,

cspline(vx, vy) – для кубічного сплайна.

bspline(vx, vy, u, n) – розраховує допоміжний вектор коефіцієнтів **vs**. Відмінність полягає в тому, що стикування сплайнів проводиться не в вузлах сітки, а в точках, які визначає користувач і в векторі **u** довжиною N-1. Параметр **n** визначає: 1 – лінійний сплайн, 2 – квадратичний, 3 – кубічний. Розмірність вектора u повинна бути на 1 (2, 3) меншою за розмірність векторів **vx, vy. u1≥vx1, uN≤vxN**.

Апроксимація

Для розрахунку значень по табличних вихідних даних великої розмірності використовується апроксимація. Апроксимуюча функція не співпадає з сітчастою в вузлах сітки. Вона проводиться таким чином, що друга норма (евклідова) – сума квадратів значень нев'язок в вузлах сітки є мінімальною.

Використовуються як функціональна, так і поліноміальна апроксимації. В апроксимуючому поліномі максимальний ступінь M на відміну від інтерполяції не M=N-1, а M≤N-1.

В [4] для вибору ступеню полінома рекомендовано критерій, згідно якому ступінь поліному визначається порядком кінцевої різниці, яка є сталою для сітчастої функції.

Знаходження коефіцієнтів апроксимуючого поліному

$$f(z)=\sum_{i=0}^{M} a_i z^i$$

за критерієм мінімізації середньоквадратичного відхилення y(x) від дійсних значень зводиться до розв'язання наступної СЛАР:

$$\begin{aligned}
&a_0\sum xi^0 + a_1\sum xi^1 + \ldots + a_N\sum xi^{M-1} = \sum yixi^0\\
&a_0\sum x^1 + a_1\sum xi^2 + \ldots + a_N\sum xi^M = \sum yixi^1\\
&a_0\sum xi^{M-1} + a_1\sum xi^M + \ldots + a_N\sum xi^{2(M-1)} = \sum yix^{M-1},
\end{aligned}$$

де N – число пар точок, M – ступінь поліному апроксимації.

Побудова поліному апроксимації має наступний алгоритм:

1. Визначення векторів аргументу «x» та функції «y» в N вузлах.
2. Визначення матриці X коефіцієнтів СЛАР.
3. Знаходження коефіцієнтів поліному розв'язанням СЛАР.

В разі застосування функціональної апроксимації $f(\vec{a},\vec{x})$ складається система нелінійних рівнянь

$$\sum_{i=1}^{N}(y_i(\vec{x})-f(\vec{a},\vec{x}))\cdot\frac{df(\vec{a},\vec{x})}{d\vec{a}}=\overrightarrow{y(\vec{x})}$$

Наприклад, при наявності 10-ти пар точок сітчастої функції для лінійної залежності **y(x)=ax+b** система рівнянь з двома коефіцієнтами набуває наступного вигляду:

$$\begin{cases}\sum_{i=1}^{10}(y_i-(ax_i+b))\frac{d(ax+b)}{da}=y_1\\ \sum_{i=1}^{10}(y_i-(ax_i+b))\frac{d(ax+b)}{db}=y_2\end{cases}\quad\begin{cases}\sum_{i=1}^{10}(y_i-(ax_i+b))x_i=y_1\\ \sum_{i=1}^{10}y_i-(ax_i+b)=y_2\end{cases}$$

Для поліноміальної апроксимації в MathCAD також застосовуєтья функція **interp(vs,vx,vy,x,N).** Вектор допоміжних коефіцієнтів **vs** розраховується функцією **regress(vx, vy, N)**, де **N** – ступінь апроксимуючого поліному. В результуючому векторі **VS** починаючи з четвертого елемента знаходяться коефіцієнти «$\mathbf{a_i}$» апроксимуючої функції наближення. Технологія застосування функцій аналогічна проведенню сплайн інтерполяції.

Апроксимація лінійною комбінацією функцій **$f_i(x)$** записується в вигляді багаточлену:

$$y(x) = a_0 f_0(x) + a_1 f_1(x) + ... + a_N f_N(x) \ ,$$

В MathCAD наближення лінійною комбінацією проводить функція **linfit(vx,vy,F)**, де

vx,vy – вектори вузлів та значень сітчастої функції,

F – вектор-стовпець функцій **$f_i(x)$**.

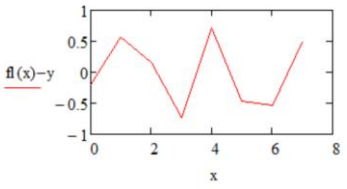
Наприклад, для сітчастої функції з вузлами 0, 1, 2, 3, 4, 5, 6, 7 та значеннями 4, 5, 9, 20, 33, 52, 73, 96, відповідно, апроксимація виразом

$$y(x) = ax^2 + be^{-x} + c\frac{x}{1+x} + d$$

розв'язання може виглядати наступним чином

$$F(x) := \begin{pmatrix} x^2 & e^{-x} & \frac{x}{1+x} & 1 \end{pmatrix}^T \qquad f := \text{linfit}(x, y, F) \qquad f = \begin{pmatrix} 1.729 \\ 73.913 \\ 93.51 \\ -70.113 \end{pmatrix}$$

$$fl(x) := f_0 \cdot x^2 + f_1 \cdot \exp(-x) + f_2 \cdot \frac{x}{1+x} + f_3$$



$$\frac{\text{norm}(fl(x) - y)}{\min(y)} = 0.131$$

Функціональна апроксимація в MathCAD проводиться функцією

y:=genfit (vx,vy,vg,F),

де **vx, vy** – вектори вузлів та значень сітчастої функції; **y** – результат. Вектор значень коефіцієнтів апроксимуючої функції; **vg** – вектор початкового наближення для всіх коефіцієнтів; **F** – ім'я функції користувача, яка повертає вектор стовпець з самої функції апроксимації та її часткових похідних по всіх коефіцієнтах.

Примітка. Для спрощення передбачена можливість не тільки аналітичного визначення елементів F вручну користувачем, а й використання функцій аналітичного диференціювання (оператор Δ Ctrl+Shift+G) та **stack**.

Наприклад, для функції апроксимації $y(x) = a + bx^c$ записи можуть мати наступний вигляд:

із визначенням коефіцієнтів функції через елементами вектора

$$f(x,a) := a_0 + a_1 \cdot x^{a_2}$$

$$FF(x,a) := \begin{pmatrix} a_0 + a_1 \cdot x^{a_2} \\ 1 \\ x^{a_2} \\ a_1 \cdot x^{a_2} \cdot \ln(x) \end{pmatrix}$$

$$FF(x,a) := \text{stack}\left(f(x,a), \nabla_a f(x,a)\right) \rightarrow \begin{pmatrix} x^{a_2} \cdot a_1 + a_0 \\ 1 \\ x^{a_2} \\ x^{a_2} \cdot \ln(x) \cdot a_1 \end{pmatrix}$$

або

із визначенням коефіцієнтів функції змінними

$$f(x,a,b,c) := a + b \cdot x^c$$

$$FF(x,a,b,c) := \begin{pmatrix} a + b \cdot x^c \\ 1 \\ x^b \\ a \cdot x^b \cdot \ln(x) \end{pmatrix}$$

$$FF(x,a,b,c) := \text{stack}\left(f(x,a,b,c), \nabla_{a,b,c} f(x,a,b,c)\right) \rightarrow \begin{pmatrix} a + b \cdot x^c \\ 1 \\ x^c \\ b \cdot x^c \cdot \ln(x) \end{pmatrix}$$

або

В 15-й версії пакету допускається використання замість імені вектора F просто ім’я функції апроксимації у вигляді функції користувача.

Наприклад, $\text{genfit}(x, y, x0, f)$ або $\text{genfit}(x, y, x0, FF)$

Для найбільш розповсюджених функцій маються вбудовані засоби, деякі з яких наведено нижче

ступенева **pwrfit(vx,vy,vg),**

логарифмічна **lnfit(vx, vy), logfit(vx,vy,vg),**

показова **expfit(vx,vy,[vg]),**

гармонійна **sinfit(vx,vy,vg)**

Наприклад, для сітчастої функції з вузлами 0, 1, 2, 3, 4, 5 та значеннями 1, 1.959, 2.683, 2.995, 2.2817, 2.197, відповідно, апроксимація виразом $y(x) = a + b\sin(cx)$ розв’язання може виглядати наступним чином:

$$fs(x,a,b,c) := a\cdot\sin(b\cdot x) + c \qquad FF(x,a,b,c) := \begin{pmatrix} c + a\cdot\sin(b\cdot x) \\ \sin(b\cdot x) \\ a\cdot x\cdot\cos(b\cdot x) \\ 1 \end{pmatrix}$$

$$genfit\left[XS, YS, \begin{pmatrix}1\\1\\1\end{pmatrix}, fs\right] = \begin{pmatrix}2\\0.5\\1\end{pmatrix} \qquad genfit\left[XS, YS, \begin{pmatrix}1\\1\\1\end{pmatrix}, FF\right] = \begin{pmatrix}2\\0.5\\1\end{pmatrix}$$

Примітка. Результат функції є вкрай чутливим до початкової точки. Невдале початкове значення може привести до невірної відповіді, або взагалі до неможливості отримати результат. Наприклад:

$$genfit\left[XS, YS, \begin{pmatrix}2\\1\\1\end{pmatrix}, fs\right] = \begin{pmatrix}-0.454\\1.401\\2.312\end{pmatrix} \qquad genfit\left[XS, YS, \begin{pmatrix}-1\\-1\\0\end{pmatrix}, FF\right] =$$

5.2 Завдання до виконання

Завдання 5.1.

1. Провести пряме розв'язання задачі поліноміальної глобальної інтерполяції в MathCAD для

$$x = 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\pi \ , y=sin(x)$$

Визначення матриці X провести ранжованою змінною/ циклом та векторно, розв'язання СЛАР – матрично та вбудованою функцією.

2. Побудувати графіки в MathCAD похибки від кількості точок N=6, 20, 40 для із застосуванням функції **`lsolve`** та матричного розв'язання СЛАР.

3. Побудувати в MathCAD графіки вихідної та інтерполюючих функцій та нев'язок поліноміальної інтерполяції в діапазоні x = 0 .. π з кроком 0.1 прямою формулою з коефіцієнтами матричного метода, прямою формулою з коефіцієнтами з функції **`regress`**.

Завдання 5.2.

Побудувати в MathCAD графіки абсолютної похибки апроксимації функції завдання 5.1 поліномами ступеню 2, 3, 4 .

Завдання 5.3.

1. Побудувати в MathCAD графіки та таблиці середньоквадратичної похибки функцій лінійного, квадратичного та кубічного сплайнів для x=(0.3..1.2) з кроком 0.1 для опису функції по точках x =0.3, 0.5, 0.9, 1.2.

x	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2
y	0.10	0.21	0.37	0.57	0.78	0.94	1.0	0.78	0.37	0.1

Завдання 5.4.

Для сітчастої функції з вузлами 1, 2, 3, 4, 5 та значеннями 0.29 0.44 0.55 0.62 0.67 0.7, відповідно, провести функціональну апроксимацію виразами $y(x) = \frac{x}{a + bx}$ та $y(x) = ax^b$.

Практикум 6. Обробка табличних даних засобами Matlab

Метою практикуму є вивчення методів, особливостей проведення глобальної поліноміальної інтерполяції, локальної поліноміальної та сплайн інтерполяції, поліноміальної та функціональної апроксимації. Набуття навичок в застосуванні засобів розв'язання цих задач в СКМ «Matlab».

6.1. Теоретичні відомості

В Matlab для розв'язання СЛАР інтерполяції призначено матричний метод ($X\vec{a} = \vec{y} \quad \vec{a} = X / \vec{y}$). Для того, щоб підготувати матрицю Вандермонде X з вектора x слугує функція **vander(x).**

Вбудовані функції інтерполяції в Matlab зосереджено в папці **toolbox\matlab\polyfun**.

Всі вбудовані функції орієнтовані на роботу з векторами та матрицями. Тобто вихідна функція повинна задаватися у вигляді вектора (матриці) в точках аргументу – вектора-стовбця. В разі, коли аргумент не задано, вважається, що аргументом є вектор x=[1 .. N], де N співпадає з довжиною вектора вихідної функції.

Для одномірної інтерполяції призначено функції **interp1q**, **interp1**, **interpft**, **spline**, для багатомірної – **interpn**, **interp2**, **interp3**.

Всі інтерполяційні функції Matlab реалізовують кускову («ковзаючу») інтерполяцію по відомих значеннях векторів сітчастої функції **X** та **Y**. Аргумент **Y** може задаватися звертанням до функції користувача. Методи **'nearest'**(найближчий), **'linear'**(лінійний) – використовують дві найближчі точки, **'spline'** **(**кубічний сплайн), **'pchip','cubic'(**кусковий кубічний поліном Ерміта) **–** чотири.

Функція **yi=interp1q(<x>,Y,xi)** повертає лінійно інтерпольовані значення вектора одновимірної функції в точках, які визначені у векторі стовбці **xi**.

Функція

interp1(<x>,Y, xi,<method>,<'extrap'>,<extrapval>,<'pp'>)

є більш універсальною. Для методів, окрім **'nearest'** , **'linear'**, можливо провести екстраполяцію за межами інтервалу х опцією **'extrap'** та задати фіксовані значення екстраполяції **extrapval**. Опція **'spline'** викликає функцію **spline**, опції **'pchip'**, **'cubic'** – функцію **pchip**.

Функція **y=spline(x,y,xi)** виконує одномірну кубічну сплайн інтерполяцію в точках, які визначені у векторі стовбці **xi**.

Для обробки періодичних даних призначена функція **y=interpft(x,n),** яка виконує одновимірну інтерполяцію з використанням перетворення Фур'є. Вона повертає вектор **y** довжиною **n,** по значенням вектора періодичної функції **x** довжиною **m (m<=n).**

Апроксимація

Функція **[P<,S,M>]=polyfit(X,Y,N)** призначена для знаходження коефіцієнтів глобальної апроксимації ступеневим поліномом **P** ступеню **N** по вихідних векторах **X,Y**.

Коефіцієнти полінома заносяться в результуючий вектор **P** в порядку зниження ступеню. Опціональний параметр результату **S** є структурою, яка містить наступні поля: коефіцієнт розкладання матриці Вандермонде R, ступінь свободи df, норму нев'язки normr. Структура використовується функцією **polyval**. Опціональний параметр **M** є двоелементним вектором, що містить середнє значення та СКВ аргументу X. Функція проводить глобальний регресійний аналіз по всіх **N** точках вихідних даних. В разі, коли ступінь поліному **N** на одиницю менша розміру векторів аргументів, результатом є квазіінтерполяційний поліном. В інших випадках – апроксимаційний.

Функція **[Y<,DELTA>]=polyval(P,X<,S>)** розраховує значення ступеневого поліному з коефіцієнтами з вектора **P** в точці (точках) **X**. В разі наявності опційного аргументу **S**, можна отримати опційний результуючий параметр **DELTA** – СКВ результату.

В Matlab відсутні засоби функціональної апроксимації. Для її проведення рекомендується використовувати лінеаризовані моделі з функцією **polyfit** для полінома 1-го ступеню або засоби з додатку оптимізації **Optimization Toolbox** на зразок **lsqcurvefit**, **lsqnonlin**, **fminsearch**.

6.2. Basic fitting

В Matlab в вікні графіки **figure** з пункту **Tools** меню відкривається додаток інтерактивного регресійного аналізу **Basic Fitting** – базові різновиди наближення (рис. 6.1).

Додаток **Basic Fitting** дозволяє провести інтерполяцію кубічними сплайнами (**spline interpolat**), апроксимацію ступеневими поліномами з ступенем від 1 до 10 (**linear, quadratic, cubic, nth degree polynomial**), інтерполяцію поліномами Ерміта (**shape-preserving**).

При обранні з списку **Check to display** на панелі **Plots fits** типу наближення в графічному вікні відображається відповідні графіки з поясненнями та формулами функцій наближення, графіки нев'язок (**residuals**).

При обранні з списку на панелі **Numerical results** виду наближення в полі **Coefficients and norm of residuals:** відображається формульна залежність, чисельні значення коефіцієнтів та нев'язки. Значення для типу наближення, які обираються зі списку **Fit**, є незалежними від тих, графіки яких зображуються в вікні **Figure** при обранні зі списку **Check to display**.

Введення значення x в поле панелі **Find** та натискання кнопки **Evaluate** виводить в таблицю чисельне значення наближаючої функції для введеного аргументу.

Select data – вибір набору даних з тих, для яких зображені графіки.

Center and scale x data – центрування даних до нульового середнього значення та нормування до одиничного стандартного відхилення.

Check to display fits on figure – список типів наближення з якого проводиться вибір поточного чи поточних.

Show equations – керує виведенням формульних залежностей на графіки.

Plot residuals – виводить графік нев'язок.

`Show norm of residuals` – керує виведенням чисельного значення норми нев'язки на графік нев'язок.

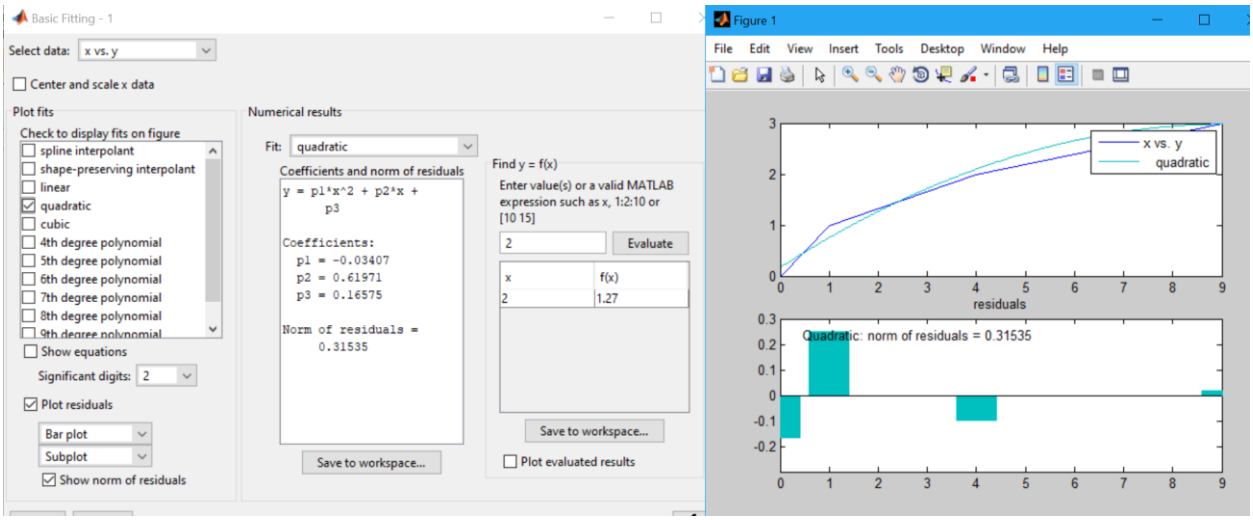


Рисунок 6.1 – Додаток **`Basic Fitting`**

6.3 Приклади та завдання до виконання

Приклад 6.1.

Розробити комп'ютерну модель для дослідження методів одномірної інтерполяції. Модель повинна зчитувати функцію для формування даних, значення чотирьох точок аргументу, точки інтерполяції. Розраховувати та виводити на екран значення функції в точках аргументів, інтерполяційні значення за вбудованими лінійним та кубічним методами, лінійним та квадратичним методами із розв'язанням системи лінійних рівнянь та безпосереднім розрахунком. Для всіх методів розраховувати та виводити час розрахунку та абсолютну похибку інтерполяції (рис. 6.2).

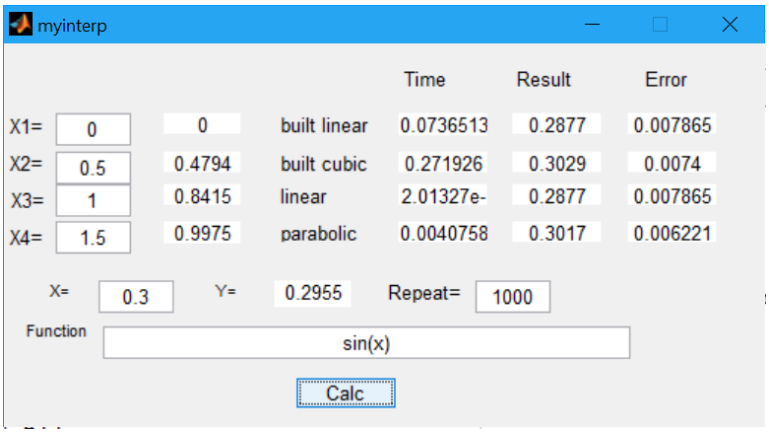


Рисунок 6.2 – Вигляд вікна моделі

РОЗВ'ЯЗАННЯ

Щоб забезпечити працездатність моделі безпосередньо після запуску, визначимо початкові дані: аргументи функції – вектор **`[0 0.5 1 1.5]`**, функція – **`sin(x)`**, точка інтерполяції – **`0.3`**, кількість повторів – **`1000`**. Повторення дій введено для того, щоб компенсувати велику швидкодію сучасних комп'ютерів. Без повторення неможливо проаналізувати час інтерполяції.

Для скорочення витрат оформлення елементів проведемо в середовищі **`GUIDE`**. При цьому відпадає необхідність вводити глобальні змінні. Значення будуть передаватися через вбудовану структуру **`handles`**.

У вікні розташовані: поля введення опорних значень аргументів x1, x2, x3, x4, точки інтерполяції **`p`**, функції інтерполяції, кількості повторень, поля виведення результатів значень функції в чотирьох опорних точках, точці інтерполяції, значення вбудованої лінійної, кубічної інтерполяції, лінійної та параболічної інтерполяції, часу розрахунку та абсолютної похибки кожного виду інтерполяції.

Передача цих та інших даних між підпрограмами в структурі GUI введено поля **`x`** – вектор точок аргументу, **`y`** – вектор значень функції, **`point`** – точка інтерполяції, **`q`** – базова функція, **`n`** – кількість повторів. Ініціюються поля в стандартному методі **`OpeningFcn`**, який описує дії застосунка перед його виведенням на екран. Визначені поля записуються в структуру стандартною функцією **`guidata`**.

Вбудована лінійна інтерполяція виконується функцією **interp1** з ключем **linear**. Вбудована кубічна інтерполяція – **interp1** з ключем **cubic.**

Для лінійної інтерполяції використано вираз

$$f(p) = y_1 + (y_2 - y_1) \cdot \frac{p - x_1}{x_2 - x_1}$$

Значення параболічної інтерполяції розраховується в два етапи. На першому по вихідних даних визначаються коефіцієнти A_1, A_2, A_3 параболічного поліному [5].

$$A_1 = \frac{y_1 \cdot (x_3 - x_2) + y_2 \cdot (x_1 - x_3) + y_3 \cdot (x_2 - x_1)}{d}$$

$$A_2 = \frac{y_1 \cdot (x_2^2 - x_3^2) + y_2 \cdot (x_3^2 - x_1^2) + y_3 \cdot (x_1^2 - x_2^2)}{d}$$

$$A_1 = \frac{y_1 \cdot (x_3 - x_2) \cdot x_3 \cdot x_2 + y_2 \cdot (x_1 - x_3) \cdot x_1 \cdot x_3 + y_3 \cdot (x_2 - x_1) \cdot x_2 \cdot x_1}{d}$$

де $d=(x_1-x_2)\bullet(x_2-x_3)(x_3-x_1)$.

На другому етапі розраховується сам поліном

$$f(p) = \sum_{i=1}^{3} A_i \cdot p^{i-1}$$

Дії компонентів

Лістинг функції керування наведено на рис. 6.3.

Поля введення даних зчитують текстовий рядок, перетворюють його на чисельне значення та записують у відповідне поле структури **handles**. Для зміни первинної функції використовується **inline** функція Matlab, яка виконує дію, прописану в текстовому рядку свого аргументу.

Для того, щоб не повторювати однакові рядки коду в зворотних функціях обробки дій полів введення, застосовано окрему функцію користувача **myinit**. Функція виводить результати значень функції в базових точках у відповідні поля вікна після кожної зміни введених даних. Викликається в кожній зворотній функції полів даних.

Розрахунки по виразах проводяться в зворотній функції кнопки **Calcbutton**. Для кожного методу запускається секундомір **tic**, **n** разів в

циклі проводиться інтерполяція, результати переводяться в текстовий формат та виводяться у відповідне поле виведення.

```
function varargout = myinterp(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
             'gui_OpeningFcn', @myinterp_OpeningFcn, ...
             'gui_OutputFcn',  @myinterp_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code


function myinterp_OpeningFcn(hObject, eventdata, ...
handles, varargin)
 handles.output = hObject;
 handles.q=inline('sin(x)');
 handles.x=[0 0.5 1 1.5]; handles.n=1000;handles.point=0.3;
 handles.y=handles.q(handles.x);
 guidata(hObject, handles);
 myinit(hObject, eventdata, handles);

function pushbutton1_Callback(hObject, eventdata, handles)
d=handles.q(handles.point);
 tic;
 for i=1:handles.n
   tmp=interp1(handles.x,handles.y,handles.point,'linear');
 end
 t=toc;
 set(handles.blintimetext,'String',t);
 set(handles.blinrestext,'String',num2str(tmp,4));
 set(handles.blinerrortext,'String',...
 num2str(abs(tmp-d),4));
 tic;
```

```
 for i=1:handles.n
    tmp=interp1(handles.x,handles.y,handles.point,...
   'cubic');
 end
 t=toc;
 set(handles.bqtimetext,'String',t);
 set(handles.bqrestext,'String',num2str(tmp,4));
 set(handles.bqerrortext,'String',...
 num2str(abs(tmp-d),4));
 tic;
 for i=1:handles.n
    tmp=handles.y(1)+(handles.y(2)-handles.y(1))...
    *(handles.point-handles.x(1))/(handles.x(2)-...
    handles.x(1));
 end
 t=toc;
 set(handles.lintimetext,'String',t);
 set(handles.linrestext,'String',num2str(tmp,4));
 set(handles.linerrortext,'String',...
 num2str(abs(tmp-d),4));
 tic;
 for i=1:handles.n
    tmp=quadinterp(handles.x,handles.y,handles.point);
 end
 t=toc;
 set(handles.partimetext,'String',t);
 set(handles.parrestext,'String',num2str(tmp,4));
 set(handles.quaderrortext,'String',...
 num2str(abs(tmp-d),4));

function Fedit_Callback(hObject, eventdata, handles)
  handles.q=inline(get(hObject,'String'));
  guidata(hObject,handles);
  myinit(hObject, eventdata, handles);

function x1edit_Callback(hObject, eventdata, handles)
 handles.x(1)=str2num(get(hObject,'String'));
 handles.y=handles.q(handles.x); guidata(hObject,handles);
 myinit(hObject, eventdata, handles);

function X2edit_Callback(hObject, eventdata, handles)
 handles.x(2)=str2num(get(hObject,'String'));
 handles.y=handles.q(handles.x); guidata(hObject,handles);
 myinit(hObject, eventdata, handles)';

function X3edit_Callback(hObject, eventdata, handles)
 handles.x(3)=str2num(get(hObject,'String'));
```

```
 handles.y=handles.q(handles.x); guidata(hObject,handles);
 myinit(hObject, eventdata, handles);

function Xedit_Callback(hObject, eventdata, handles)
 handles.point=str2num(get(hObject,'String'));
 guidata(hObject,handles);
 myinit(hObject, eventdata, handles);

function Nedit_Callback(hObject, eventdata, handles)
 handles.n=str2int(get(hObject,'String'));
 guidata(hObject,handles);

function myinit(hObject, eventdata, handles)
 handles.y=handles.q(handles.x);
 guidata(hObject,handles);
 set(handles.realtext,'string',...
 num2str(handles.q(handles.point),4));
 set(handles.y1text,'string',...
 num2str(handles.q(handles.x(1)),4));
 set(handles.y2text,'string',...
 num2str(handles.q(handles.x(2)),4));
 set(handles.y3text,'string',...
 num2str(handles.q(handles.x(3)),4));
 set(handles.y4text,'string',...
 num2str(handles.q(handles.x(4)),4));

function X4edit_Callback(hObject, eventdata, handles)
 handles.x(4)=str2num(get(hObject,'String'));
 handles.y=handles.q(handles.x);
 guidata(hObject,handles);
 myinit(hObject, eventdata, handles);



function z=quadinterp(x,y,a)
 d=(x(1)-x(2))*(x(2)-x(3))*(x(3)-x(1));
 CC(1)=(y(1)*(x(3)-x(2))+y(2)*(x(1)-x(3))+...
 y(3)*(x(2)-x(1)))/d;
 CC(2)=(y(1)*(x(2)^2-x(3)^2)+y(2)*(x(3)^2-...
 x(1)^2)+y(3)*(x(1)^2-x(2)^2))/d;
 CC(3)=(y(1)*(x(3)-x(2))*x(3)*x(2)+y(2)*(x(1)-...
 x(3))*x(1)*x(3)+y(3)*(x(2)-x(1))*x(2)*x(1))/d;
 z=CC(1)*a^2+CC(2)*a+CC(3);
```

Рис. 6.3 – Лістинг моделі прикладу 6.1

Завдання 6.1.

1. Провести пряме розв'язання задачі поліноміальної глобальної інтерполяції в Matlab для

$$x = 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\pi \ , y=sin(x)$$

Визначення матриці X провести ранжованою змінною/ циклом та векторно, розв'язання СЛАР – матрично та вбудованою функцією. Порівняти результати з завданням 5.1.

Завдання 6.2.

Побудувати в Matlab графіки абсолютної похибки апроксимації функції завдання 3.1 поліномами ступеню 2, 3, 4 . Порівняти результати з завданням 5.2.

Завдання 5.3.

1. Побудувати в Matlab графіки середньоквадратичної похибки інтерполяції всіма методами функції **`interp1`**, функціями **`spline`**, **`interpq`** для x=(0.3..1.2) з кроком 0.1 для опису функції по точках x =0.3, 0.5, 0.9, 1.2.

x	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2
y	0.10	0.21	0.37	0.57	0.78	0.94	1.0	0.78	0.37	0.1

Завдання 6.4.

Для сітчастої функції з вузлами 1, 2, 3, 4, 5 та значеннями 0.29 0.44 0.55 0.62 0.67 0.7, відповідно, провести функціональну апроксимацію виразами $y(x) = \frac{x}{a + bx}$ та $y(x) = ax^b$. Порівняти точність наближення в MathCAD та Matlab.

Практикум 7. Моделювання елементів оптико-електронних приладів методами геометричної оптики

Мета практикуму – ознайомлення студентів з методиками розв'язання типових оптичних задач засобами комп'ютерної математики. Набуття студентами практичних навичок у застосуванні чисельних методів засобами СКМ в задачах геометричної оптики та радіометрії.

7.1. Теоретичні відомості

Більшість завдань геометричної оптики вирішуються із застосуванням рекурентних формул висот h та кутів σ в тому чи іншому вигляді. Базою є таблиці конструктивних даних оптичної системи (ОС): радіусів, показників заломлення, відстаней між поверхнями («товщин»), – так званих «r, d, n» таблиць. Наприклад, для тонкої оптичної системи формули ходу променів мають наступний вигляд:

$$\sigma_{i+1} := \frac{n_{i+1} - n_i}{n_{i+1} \cdot r_i} \cdot h_i + \frac{n_i \cdot \sigma_i}{n_{i+1}} \quad , \qquad h_{i+1} := h_i - d_i \cdot \sigma_{i+1}$$

В формулах на і-му кроці використовується значення висоти та кута, які були отримані на попередньому кроці. Тому формули називаються рекурентними.

За даними розрахунку можна визначити кардинальні параметри системи, аберацийні суми третього порядку і т. і..

$$f' = \frac{h_1}{\sigma_{\max}} \ , \ S'_f = \frac{h_{\max}}{\sigma_{\max}},$$

де h_1 – висота на першій поверхні ОС, h_{max} – висота на останній поверхні ОС, σ_{max} – кут після останньої поверхні ОС.

Для отримання значень висоти та кута променю на поверхні слід

1. Визначити вихідні значення висоти h та кута σ на початковій поверхні (в площині об'єктів) .
2. За формулами висот та кутів послідовно для кожної поверхні провести розрахунки.

ОБЧИСЛЕННЯ ПО РЕКУРЕНТНИХ ФОРМУЛАХ РАНЖОВАНИМИ ЗМІННИМИ В СКМ MATHCAD НЕ ДАЄ ВІРНОЇ ВІДПОВІДІ.

В MathCAD обчислення можна проводити вручну почергово для кожної поверхні, застосувати програмування або використати матричний підхід. В Matlab обчислення реалізуються в командному режимі почергово для кожної поверхні або програмуванням функцій користувача та скрипт-функцій.

Матрична модель ОС [6]

Рівняння для вхідної висоти h, вихідної висоти h’ та відповідних кутів σ, σ’ записуються наступним чином

$$\sigma' := \frac{n}{n'}\sigma + \frac{n'-n}{n'\cdot r}\cdot h$$
$$h' = h$$

Якщо параметри вихідного променю в матричному вигляді записати через двокомпонентний вектор

$$I = \begin{bmatrix} h \\ \sigma \end{bmatrix},$$

то формули для знаходження параметрів вихідного променю після взаємодії з поверхнею як елементів двокомпонентного вектора набудуть вигляду множення « рефракційної матриці поверхні» на вектор вхідного променю:

$$I' = \begin{bmatrix} h' \\ \sigma' \end{bmatrix} = \begin{cases} h \\ \frac{n}{n'}\sigma + \frac{n'-n}{n'r}h \end{cases} = \begin{bmatrix} 1 & 0 \\ \frac{n'-n}{n'r} & \frac{n}{n'} \end{bmatrix} \cdot \begin{bmatrix} h \\ \sigma \end{bmatrix} = R \cdot I \ ,$$

де рефракційна матриця поверхні $R = \begin{bmatrix} 1 & 0 \\ \frac{n'-n}{n'r} & \frac{n}{n'} \end{bmatrix}$

Для доведення променю до наступної поверхні h’’ в матричному вигляді записати формулу висот

$$I'' = \begin{bmatrix} h'' \\ \sigma'' \end{bmatrix} = \begin{cases} h' - d\sigma' \\ \sigma' \end{cases} = \begin{bmatrix} 1 & -d \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} h' \\ \sigma' \end{bmatrix} = T \cdot I',$$

де матриця переведення між поверхнями $T = \begin{bmatrix} 1 & -d \\ 0 & 1 \end{bmatrix}$.

Дія довільної ОС описується в матричному вигляді як добуток рефракційної матриці останньої поверхні та N-кратного попарного множення рефракційних матриць та матриць доведення до відповідних матриць

$$RS = R_N \prod_{i=1}^{N-1} T_i R_i .$$

Наприклад, дія лінзи може бути записана наступним чином

$$I' = \begin{bmatrix} h' \\ \sigma' \end{bmatrix} = R_2 T_{12} R_1 \cdot \begin{bmatrix} h \\ \sigma \end{bmatrix} = RS \cdot I$$

7.1. Приклади та завдання до виконання

Приклад 7.1. Визначити фокусну відстань та накреслити хід променів в ОС, яка задана конструктивними параметрами r, d ,n.

Засоби Matlab

Для розрахунку ходу променю крізь ОС засобами програмування використовується функцію користувача "**hod(u,v,r,d,n)**". Функція приймає в якості параметрів початкові значення на першій поверхні кута **u** та висоти **v,** масивів радіусів **r**, товщин **d** та показників заломлення **n** ОС. За формулами кутів та висот формує та повертає вектори кутів **t** та висот **h** на поверхнях ОС.

Опис функції:

```
function [ t h ] = hod(u,v,r,d,n )
t(1)=u;
h(1)=v;
for i=1:length(d)
    t(i+1)=n(i)/n(i+1)*t(i)+(n(i+1)...
    -n(i))/n(i+1)/r(i)*h(i);
    h(i+1)=h(i)-d(i)*t(i+1);
end
k=length(r);
 t(k+1)=n(k)/n(k+1)*t(k)+(n(k+1)...
  -n(k))/n(k+1)/r(k)*h(k);
end
```

Виклик

```
>> r=[10 -10];
>> d=[1];
>> n=[1 1.5163 1];
[a b]=hod(0,1,r,d,n)
a =         0    0.0340    0.1015
b =    1.0000    0.9660
```

Результатом є вектор **a,** який містить значення кутів, та вектор **b**, який містить значення висот.

Для розрахунку ходу променю крізь ОС матричним методом використано функцію користувача "**hod_matr(N,r,d,n)**". Функція приймає в якості параметрів значення кількості поверхонь ОС **N,** масивів радіусів **r**, товщин **d** та показників заломлення **n** ОС. Формує та повертає матрицю перетворення ОС – **rs**.

На відміну від MathCAD, в Matlab неможливо програмно визначити вектор, елементами якого є матриці. Тому функція не зберігає масиви матриць рефракції та переведення для кожної поверхні, а тільки тимчасово зберігає їх в проміжних змінних.

Опис функції:

```
function [ rs ] = hod_matr( N,r,d,n )
rs=0;
for i=1:N-1
    T_tmp=[1 -d(i);0 1];
    R_tmp=[1 0; (n(i+1)-n(i))/n(i+1)/r(i)  n(i)...
           /n(i+1)];
    rs=T_tmp*R_tmp;
end
R_tmp=[1 0; (n(N+1)-n(N))/n(N+1)/r(N)  n(N)...
      /n(N+1)];
rs=R_tmp*rs;
end
```

Виклик:

```
>> R=hod_matr(2,r,d,n)
R =
    0.9660   -0.6595
    0.1015    0.9660
>> I=[1;0];
>> R*I
ans =
    0.9660
    0.1015
```

Результатом є вектор. Перший елемент вектора містить значення висоти на останній поверхні, другий – кута.

Для підготування даних для зображення ходу променю крізь ОС використано функцію користувача "**hod_fig(l_1,l_2, u, v ,d)**". Функція приймає в якості параметрів значення відстані від площини предметів до ОС **l_1,** значення відстані від ОС до площини аналізу **l_2**, масивів кутів **u**, висот **v** та товщин **d** ОС. Формує та повертає вектори відстаней **l** та відповідних висот **h**.

Опис функції:

```
function [ l h] = hod_fig( l_1,l_2, u, v ,d)
l(1)=0; l(2)=l_1;
N=length(d);
for i=1:N
    l(i+2)=l(i+1)+d(i);
end
l(N+3)=l(N+2)+l_2;
h=[v(1) v v(N+1)-l_2*u(N+2)];
end
```

Виклик:

```
>> [a b]=hod_fig(1,1,a,b,d)
>> plot(a,b)
```

Результат функції наведено на рис.7.1.

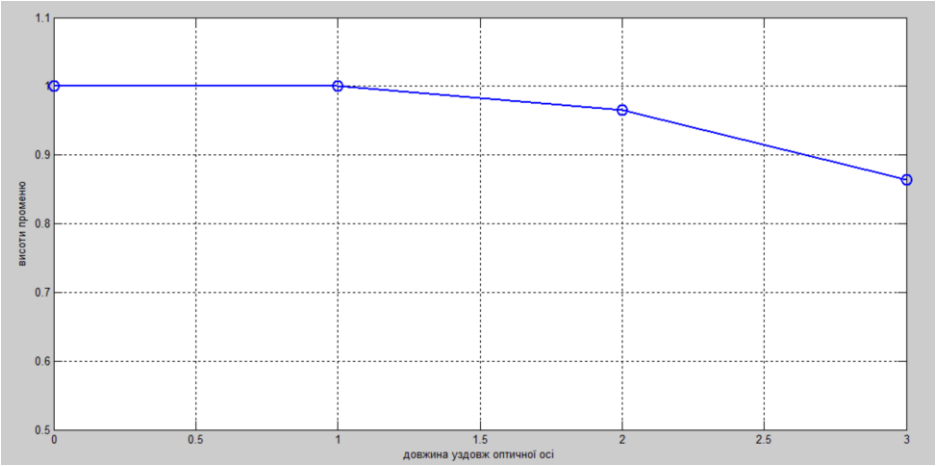


Рисунок 7.1 – результат функції **hod_fig**

Засоби MathCAD

Розрахунки вручну послідовно для кожної поверхні

$$\sigma_1 := \frac{n_1 - n_0}{n_1 \cdot r_0} \qquad h_1 := h_0 - d_0 \cdot \sigma_1 \qquad \sigma_2 := \frac{n_2 - n_1}{n_2 \cdot r_1} \cdot h_1 + \frac{n_1 \cdot \sigma_1}{n_2}$$

$$f := \frac{h_0}{\sigma_{last(\sigma)}} = 9.852 \qquad sf := \frac{h_{last(h)}}{\sigma_{last(\sigma)}} = 9.517$$

$$h = \begin{pmatrix} 1 \\ 0.966 \end{pmatrix} \qquad \sigma = \begin{pmatrix} 0 \\ 0.034 \\ 0.102 \end{pmatrix}$$

Визначення вектору параметрів вхідного променю

$$Iinp := \begin{pmatrix} h_0 \\ \sigma_0 \end{pmatrix} \qquad N := last(r)$$

Визначення рефракційних матриць R та матриці переведення T

$$j := 0 .. N$$

$$R_j := \begin{pmatrix} 1 & 0 \\ \frac{1}{r_j} \cdot \frac{n_{j+1} - n_j}{n_{j+1}} & \frac{n_j}{n_{j+1}} \end{pmatrix} \qquad T_0 := \begin{pmatrix} 1 & -d_0 \\ 0 & 1 \end{pmatrix}$$

Визначення матриці перетворення лінзи

$$RD(N) := R_N \cdot \prod_{i=0}^{N-1} (T_i \cdot R_i)$$

Розрахунок параметрів вихідного променю

$$Iout := RD(N) \cdot Iinp = \begin{pmatrix} 0.966 \\ 0.1015 \end{pmatrix}$$

Програмування реалізовано через зовнішню функцію користувача “hod”. В документ функція вставляється як зовнішнє посилання для скорочення обсягу. Для універсальності всі потрібні дані в функцію передаються як параметри: початкові значення кута “u” та висоти “v”, вектори радіусів “r”, товщин “d” та показників заломлення “n”. Функція за стандартним алгоритмом обчислює та повертає два вектори: вектор кутів на поверхнях ОС “t” та вектор висот на поверхнях – “h”:

$$\text{hod}(u,v,r,d,n) := \left| \begin{array}{l} t_0 \leftarrow u \\ h_0 \leftarrow v \\ \text{for } i \in 0..\text{last}(d) \\ \quad \left| \begin{array}{l} t_{i+1} \leftarrow \dfrac{n_i \cdot t_i}{n_{i+1}} + h_i \cdot \dfrac{n_{i+1} - n_i}{n_{i+1} \cdot r_i} \\ h_{i+1} \leftarrow h_i - d_i \cdot t_{i+1} \end{array} \right. \\ k \leftarrow \text{last}(r) \\ t_{k+1} \leftarrow n_k \cdot \dfrac{t_k}{n_{k+1}} + h_k \cdot \dfrac{n_{k+1} - n_k}{n_{k+1} \cdot r_k} \\ (t \;\; h) \end{array} \right.$$

Reference:D:\for_students\kmod\lab\hod.xmcd(R)

$$(u \;\; v) := \text{hod}(\sigma_0, h_0, r, d, n)$$

$$u = \begin{pmatrix} 0 \\ 0.034 \\ 0.102 \end{pmatrix} \qquad v = \begin{pmatrix} 1 \\ 0.966 \end{pmatrix}$$

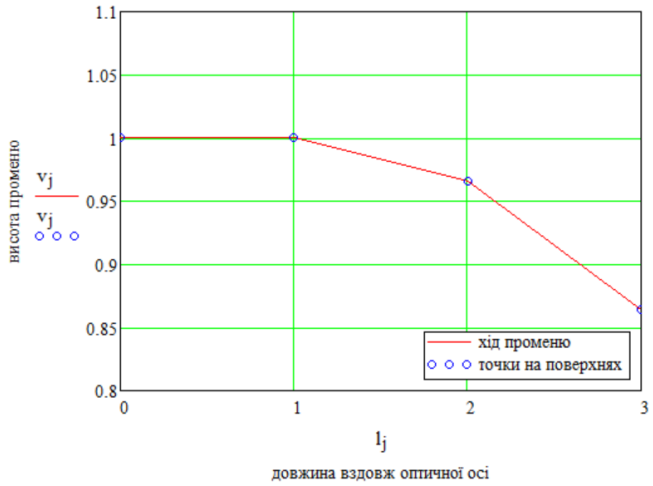
$$\text{fokus} := \frac{v_0}{u_{\text{last}(u)}} = 9.852 \qquad Sf := \frac{v_{\text{last}(v)}}{u_{\text{last}(u)}} = 9.517$$

Як видно з наведеного вище лістингу, результати всіх способів співпадають.

Для отримання графічного зображення ходу променів треба на основі вектора значень осьових відстаней між суміжними поверхнями “d” та інформації про відстані від площини предметів до першої поверхні та від останньої поверхні до поверхні аналізу створити вектор відстаней “l” , і-тий елемент якого буде дорівнювати відстані між площиною предметів та і-тою поверхнею. Цей вектор має кількість елементів на 2 більшу за N (кількість поверхонь). Графік будується як параметрична залежність вектора висот від вектора відстаней. Щоб вирівняти значення в відповідних точках векторів до вектора висот слід додати в кінець ще одне значення – висоти в площині аналізу.

$l_0 := 0 \quad l_1 := 1 \quad l_2 := l_1 + d_0 \quad d_1 := 1 \quad l_3 := l_2 + d_1$

$v_{N+1} := v_N - u_{N+1} \cdot d_N \quad v := \text{stack}(h_0, v) \quad j := 0..N + 2$



Завдання 7.1. Розрахувати в MathCAD та Matlab кардинальні параметри ОС та візуалізувати хід променю для ОС з наступними параметрами:

r	-20	20	-20	20	
d	1	10	1		
n	1	1.5163	1	1.5163	1

Завдання 7.2. Отримати за допомогою символьних операцій вираз для матриці перетворень оптичної системи з прикладу 7.1.

Завдання 7.3. Побудувати залежність положення зображення в повітрі для оптичної системи з f'=50 мм від положення предмета a=(-100..0) мм з кроком 1mm за формулою відрізків. Пояснити отриманий графік.

$$\frac{f'}{a'} + \frac{f}{a} = 1$$

Практикум 8. Радіометричні моделі елементів оптико-електронних приладів

Мета практикуму – ознайомлення студентів з методиками розв'язання типових оптичних задач засобами комп'ютерної математики. Набуття студентами практичних навичок у застосуванні чисельних методів засобами СКМ в задачах радіометрії.

8.1. Теоретичні відомості

Для задач визначення апертурних втрат, ефективності ОС з точки зору збирання променевої енергії, випромінення, яке потрапляє на вхідну зіницю, розрахунків реальної чутливості фотоприймачів, електричних сигналів з фотоприймачів типовим вирішення задач інтегрування та наближення табличних функцій.

Розрахунок променевого потоку від джерела потребує знаходження значення інтегрального виразу для тілесних кутів

$$F = Fo\int_0^{\Omega} \overset{0}{F}(\omega)d\omega ,$$

або для плоского апертурного кута в разі осьової симетрії індикатриси випромінення

$$F = 2\pi Fo\int_0^{\alpha} \overset{0}{F}(\alpha)\sin(\alpha)d\alpha$$

де F – променевий потік на ОС, Fo – паспортна осьова сила випромінення випромінювача, $\overset{0}{F}(\omega)$ – відносна сила випромінення для тілесного кута ω, Ω – граничне значення тілесного кута апертури, $\overset{0}{F}(\alpha)$ – відносна сила випромінення для апертурного кута α, α – граничне значення тілесного кута апертури.

Примітка. Для проведення розрахунків необхідно мати значення Fo, $\overset{0}{F}(\alpha)$ *або* $\overset{0}{F}(\omega)$. *Виробники не тільки не завжди наводять їх в паспортах виробів, але й наводять суперечливі значення.*

Наприклад, відомий виробник "OSRAM" наводить в паспорті на випромінювач IRL81 значення потоку F=20мВт, осьової сили випромінення

Fo=25мВт/ср та графік відносної сили випромінення, який наведено на рисунку 8.1.

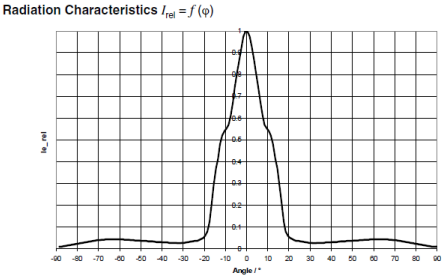


Рисунок 8.1 – Індикатриса IRL81

Поліноміальна апроксимація дев'ятого ступеню в діапазоні 0^0 ... 80^0 забезпечує відносну похибку менше 1%.

Очікуваним є отримання значення повного потоку відповідно до паспортного F=20мВт.

$$F=2\pi Fo\cdot\int_0^{80^0} F(\alpha)\sin(\alpha)d\alpha = 2\cdot 6.28\cdot 25\cdot 0.046 = 7.17 мВт$$

Значення НЕ ЗБІГАЮТЬСЯ.

В разі неможливості гарантування реальності наведених в паспортах даних, рекомендується використовувати значення загального променевого потоку та розподілу сили випромінення. В такому випадку осьову силу випромінення слід розраховувати за виразом

$$Fo=\frac{F}{2\pi\int_0^{\alpha} F(\alpha)\sin(\alpha)d\alpha}$$

Для наведеного прикладу осьова сила випромінення становитиме

$$Fo=\frac{F}{2\pi\int_0^{\alpha} F(\alpha)\sin(\alpha)d\alpha}=\frac{20}{2\pi 0.046}=69.19\frac{мВт}{ср}$$

В MathCAD інтегрування та диференціювання проводиться в один клік введення звичних символів з панелі розширених операторів. Matlab потребує програмування з використанням вбудованих функцій. Для чисельного обчислення однократного визначеного інтеграла можуть використовуватися функції **`trapz, quad, quadl`**.

Функція **trapz(<X>,Y< dim >)** обчислює інтеграл від вектора або матриці **Y** методом трапецій. Опціональний вектор **X** визначає значення аргументу, в яких буде обчислюватися інтеграл. В разі відсутності **X** вважається, що крок інтегрування дорівнює одиниці. Опціональний параметр **dim** визначає рядки або стовпці матриці **Y**, для яких буде проводитися інтегрування.

Наприклад, виклик розрахує інтеграл методом трапецій по чотирьох точках з кроком 1.

```
>> y=[1,2,3,4];
>> trapz(y)
ans = 7.5000
```

Виклик розрахує інтеграл для функції cos(x) для аргументу X в діапазоні від 0 до π/2 з кроком π/70.

```
>> X=0:pi/70:pi/2; Y=cos(X); Z = trapz(Y)
Z = 22.2780
```

Функція потребує додаткового аналізу точності результатів інтегрування, тому не рекомендується для вжитку.

Функція **quad(fun,a,b,<tol,trace,P1,P2,…>)** обчислює інтеграл від функції fun методом парабол Сімпсона. Параметри **a, b** визначають межі інтегрування. Опціональними параметрами є **tol,trace,P1,P2**. Параметр **tol** визначає точність інтегрування, яка за замовчанням встановлена 10^{-6}. Параметр trace дозволяє вивести графік про хід інтегрування в разі присвоєння ненульового значення. Параметри **P1,P2** є параметрами користувача, які можна передати до підінтегральної функції.

Підінтегральна функція визначається текстовим рядком або посиланням на функцію користувача. Функція повинна бути записана із використанням векторних поелементних дій.

Функція **quadl** проводить інтегрування методом Гауса-Лобатто. Синтаксис функції є аналогічним синтаксису функції **quad**.

Наприклад, виклик з текстовим рядком:

```
>> quad('x.*exp(-x.^0.8)+0.2',0,8)
ans =3.1604
```

виклик з посиланням на функцію користувача:

```
function y=mfun(x)
y=x.*exp(-x.^0.8)+0.2;

>> q=quad(@mfun,0,8)
q =3.1604
```

виклик **inline** з функцією:

```
» f=inline(('x.*exp(-x.^0.8)+0.2',xt')
Inline function:
f(x) ='x*exp(-x^0.8)+0.2
≫ quad(f,0,8)
ans =3.1604
```

Для інтегрування можуть також бути використані засоби символьних обчислень з додатків **Mupad** та **Symbolic Toolbox**, функції **quadgk** (Обчислення методом Гауса-Кронрода), **quadv** (Обчислення методом Сімпсона для векторних функцій скалярного аргументу).

8.2. Приклади та завдання до виконання

Приклад 8.1. Визначити осьову силу випромінення та аналітичний вираз індикатриси у вигляді поліному для випромінювача CQY36n з потужністю F=10мВт (рис. 8.1, табл. 8.1).

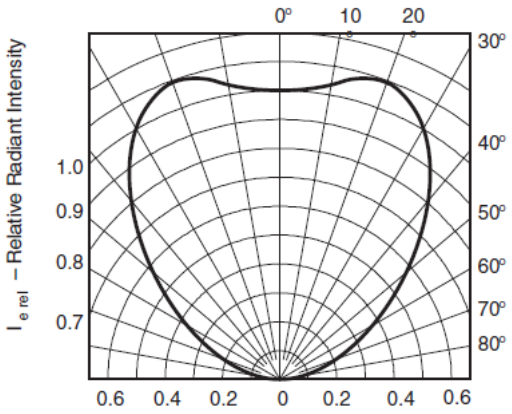


Рисунок 8.1 – Індиктриса CQY36n

Таблиця 8.1. Індикатриса CQY36n

α[град]	0	10	20	30	40	50	60	70	80
F(α)	1.000	1.010	1.093	1.000	0.806	0.572	0.373	0.222	0.105

Засоби Mathcad

В таблиці даних 9 пар точок, тому максимальний ступінь поліному – 8.

Для знаходження коефіцієнтів поліному найпростіше застосувати службову функцію **`regress`**, яка готує дані для вбудованої функції інтерполяції. В векторі результатів функції з четвертого елементу розташовані шукані коефіцієнти, які можна використати в аналітичному запису у вигляді функції користувача **`f3(x)`.** Наведений нижче фрагмент обраховується для третього ступеню поліному.

$$imax := last(x_{cq36}) = 8 \qquad N := 3 \quad vs3 := regress(x_{cq36}, y_{cq36}, N)$$

$$f3(x) := \sum_{i=0}^{N} \left(vs3_{3+i} \cdot x^{i}\right)$$

Середньоквадратичні похибки апроксимації для ступенів поліному 3, 4, 5, 6 складають

$$\frac{\left|y_{cq36} - \overrightarrow{f3(x_{cq36})}\right|}{imax} = 0.012 \qquad \frac{\left|y_{cq36} - \overrightarrow{f4(x_{cq36})}\right|}{imax} = 0.01$$

$$\frac{\left|y_{cq36} - \overrightarrow{f5(x_{cq36})}\right|}{imax} = 3.57 \times 10^{-3} \qquad \frac{\left|y_{cq36} - \overrightarrow{f6(x_{cq36})}\right|}{imax} = 1.324 \times 10^{-3}$$

Найбільш цікавими є поліноми 5-го та 6-го ступеню. Графічний аналіз (рис.8.2) показує, що в області оптичної осі поліном 5-го ступеню є ближчим до вихідних даних.

$xmax := x_{cq36_{imax}}$ $i := 0 .. imax$ $z := 0 .. xmax$

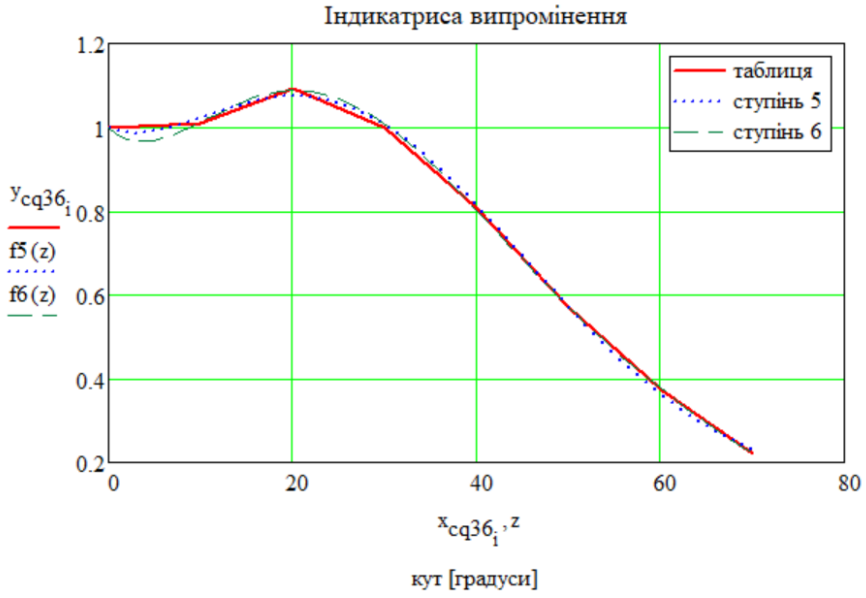


Рисунок 8.2 – Графіки наближення

Подальше інтегрування дозволяє визначити силу випромінення

$$Int := 2\cdot\pi\cdot\int_0^{xmax\cdot\frac{\pi}{180}} f5\left(x\cdot\frac{180}{\pi}\right)\cdot\sin(x)\,dx = 2.872 \qquad P_{cq36} := 10mW$$

$$Io := \frac{P_{cq36}}{Int} = 3.482\cdot\frac{mW}{sr}$$

Засоби Matlab

Для знаходження коефіцієнтів поліноміальної апроксимації з подальшим розрахунком значення апроксимаційної функції слушно застосувати функції **`polyfit`** та **`polyval`**.

```
x=0:10:80;
y=[1 1.01 1.093 1 0.806 0.572 0.373 0.222 0.105];
s=polyfit(x,y,5);
f=polyval(s,x);
```

Для визначення середньоквадратичної похибки можна використати прямий вираз

```
sum((y-f).^2)^(1/2)/(length(x)-1)
ans =  3.5696e-003
```

або вбудовану функцію розрахунку норми вектора

```
norm(y-f,2)/(length(x)-1)
```

```
ans =  3.5696e-003
```

Отримані відповіді співпадають з результатами MathCAD.

Для інтегрування визначимо підінтегральну функцію користувача. Функція повинна мати тільки один аргумент. Щоб не вводити додаткових анонімних функцій забезпечимо доступ до значень вектора s через глобальні змінні:

```
function [ y ] = mint( x )
global s
y=polyval(s,x.*180/pi).*sin(x);
end
```

Виклик:

```
global s
int=2*pi*quad(@mint,0,x(length(x))*pi/180)
Io=10/int
Io =  3.4818e+000
```

Відповіді співпадають.

Приклад 8.2. Розрахувати залежність потоку та коефіцієнта ефективності ОС від діаметра вхідної зіниці D=(10мм...50мм) з кроком 10мм для випромінювача АЛ107Б, який знаходиться на відстані l=1м від ОС.

Потік на зіниці визначається через силу світла інтегруванням виразу

$$F_{\Omega} = Io\int_{0}^{A} F(\alpha)\sin(\alpha)d\alpha$$

Для знаходження потоку потрібно визначення осьової сили випромінення Io, аналітичного виразу індикатриси випромінення F(α) та значення апертурного кута A.

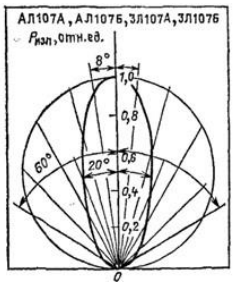
$$A = arctg(\frac{D}{2l}) \quad I_0 = \frac{F}{\int_{0}^{\pi/2} F(\alpha)\sin(\alpha)d\alpha}$$

Коефіцієнт ефективності ОС k визначається як відношення потоку, який збирається ОС до загального потоку випромінювача

$$k = \frac{F_\Omega}{F_\Sigma} = \frac{\int_0^A F(\alpha)\sin(\alpha)d\alpha}{\int_0^{\pi/2} F(\alpha)\sin(\alpha)d\alpha}$$

Паспортні дані АЛ107Б. Променевий потік F=10мВт, відносна щільність потоку

α[град]	0	8	15	17	20	30	45	60
F(α)	1.00	0.91	0.70	0.62	0.51	0.33	0.14	0.089



РОЗВ'ЯЗАННЯ

Індикатриса є гладкою і може бути наближена ступеневою косинусною залежністю $F(\alpha) = \cos(\alpha)^n$ або гаусовою залежністю $F(\alpha) = e^{-\frac{\alpha^2}{n}}$.

Засоби MathCAD

Для функціональної апроксимації призначена функція **genfit**. В якості її аргументу задається ім'я функції користувача **fc** або ім'я вектора похідних функції апроксимації **FC**.

$$fc(a,x) := \left(\cos\left(x\cdot\frac{\pi}{180}\right)\right)^a \qquad FC(a,x) := \text{stack}\left(fc(a,x), \nabla_a fc(a,x)\right)$$

$$a := \text{genfit}\left(x_{al}, y_{al}, 9.3, fc\right) = 14.524$$

$$a := \text{genfit}\left(x_{al}, y_{al}, 9.3, FC\right) = 13.645$$

Застосування функції приводить до насторожуючих результатів. Відповіді з викликом у вигляді самої функції апроксимації та вектора похідних не збігаються.

Для перевірки застосуємо пряме розв'язання системи рівнянь функціональної апроксимації з мінімізацією середньоквадратичного відхилення апроксимуючої функції від вихідної.

Пряме застосування блоку **given-find** для рівняння в векторному вигляді відповіді не дає

$ac := 10$

Given

$$\left| \overrightarrow{y_{al} - \cos\left(\frac{\pi}{180} \cdot x_{al}\right)^a} \right| = 0$$

Find(a) =

.

Рівняння повинно бути переписано в явному вигляді

$a := 20$

Given

$$\sum_{i=0}^{imax} \left[\left(y_{al_i} - \cos\left(x_{al_i} \cdot \frac{\pi}{180}\right)^a \right) \cdot \cos\left(x_{al_i} \cdot \frac{\pi}{180}\right)^a \cdot \ln\left(\cos\left(x_{al_i} \cdot \frac{\pi}{180}\right)\right) \right] = 0$$

Find(a) = 9.183

Відповідь отримано. Значення відрізняється від отриманих раніше.

Застосування функції **minerr** дозволяє розв'язати рівняння в векторному вигляді

$a := 10$

Given

$$\left| \overrightarrow{y_{al} - \cos\left(\frac{\pi}{180} \cdot x_{al}\right)^a} \right| = 0$$

Minerr(a) = 9.183

Відповіді функцій **given-find** та **minerr** співпадають.

Середньоквадратична похибка для отриманих значень становить відповідно

$$\frac{\left| \overrightarrow{y_{al} - fc(14.571, x_{al})} \right|}{imax} = 0.045 \qquad \frac{\left| \overrightarrow{y_{al} - fc(9.186, x_{al})} \right|}{imax} = 0.023$$

$$\frac{\left| \overrightarrow{y_{al} - fc(13.728, x_{al})} \right|}{imax} = 0.041$$

ВІДПОВІДІ ФУНКЦІЇ НЕВІРНІ. Косинусна залежність має похибку 0.023.

Застосування функції **genfit** для гаусової апроксимації взагалі не дає відповіді

$$fe(ac, x) := e^{-\left(\frac{x}{ac}\right)^2}$$

$$genfit(x_{al}, y_{al}, 100, fe) = \blacksquare$$

Пряме розв’язання рівняння в векторному вигляді

$$ac := 10$$

Given

$$\left| y_{al} - \left[e^{-\left(\frac{\overrightarrow{x_{al}}}{ac}\right)^2} \right] \right| = 0 \qquad \frac{\left| y_{al} - \overrightarrow{fe\left(26.456, x_{al}\right)} \right|}{imax} = 0.021$$

$$Minerr(ac) = 26.456$$

дозволяє отримати відповідь, яка забезпечує похибку наближення 0.021.

Для подальших розрахунків будемо використовувати гаусово наближення.

Інтегрування за замовчанням виконується методом Ромберга з точністю 0.001. Змінити метод інтегрування можна в контекстному меню оператора інтегрування (рис.8.3). Змінити точність перевизначенням змінної TOL.

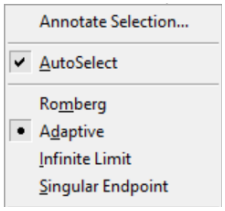


Рисунок 8.3 – Контестне меню інтегрування

$$D := 10mm, 20mm .. 50mm \qquad l := 1m \qquad P := 10mW$$

$$F(x) := \exp\left[-\left(\frac{x}{26.456}\right)^2\right] \qquad Io := \frac{P}{\int_0^{90} F(x)\sin(x \cdot deg)\, dx} = 1.696 \cdot \frac{mW}{sr}$$

$$FD(d, l) := Io \cdot \left(\int_0^{atan\left(\frac{d}{2 \cdot l}\right)} F\left(x \cdot \frac{180}{\pi}\right) \cdot \sin(x)\, dx \right)$$

D =		FD(D, l) =		$\frac{FD(D,l)}{P}$ =		$atan\left(\frac{D}{2 \cdot l}\right)$ =	
10	·mm	$2.12 \cdot 10^{-5}$	·mW	$2.12 \cdot 10^{-6}$		0.286	·deg
20		$8.479 \cdot 10^{-5}$		$8.479 \cdot 10^{-6}$		0.573	
30		$1.907 \cdot 10^{-4}$		$1.907 \cdot 10^{-5}$		0.859	
40		$3.388 \cdot 10^{-4}$		$3.388 \cdot 10^{-5}$		1.146	
50		$5.29 \cdot 10^{-4}$		$5.29 \cdot 10^{-5}$		1.432	

Одиниці тілесного кута є позасистемними, тому в пакеті не автоматично не виводяться, потрібно додавати вручну.

З результатів видно, що навіть для зіниці в 50мм апертурний кут становить всього 3 градуси. Це дає коефіцієнт корисної дії ОС 0.005%

Засоби Matlab

В стандартній бібліотеці пакета відсутні засоби функціональної апроксимації. Наявні функції з додатку **Optimization Toolbox** є засобами оптимізації та не входять до матеріалів дисципліни. Тому задачу знаходження значень коефіцієнтів ступеневої косинусною та гаусової апроксимаційних функцій доведеться вирішувати шляхом прямого розв'язання рівняння апроксимації. З функцій стандартної комплектації корисними можуть бути функція розв'язання рівнянь **fzero** та функція розв'язання систем рівнянь **fsolve**.

Для обох функцій аргументом є посилання на функцію рівняння. Функція рівняння повинна залежати тільки від коефіцієнтів апроксимаційної залежності. Рівняння функціональної апроксимації з мінімізацією середньоквадратичного відхилення апроксимуючої функції від вихідної має в якості аргументів значення табличної функції. Для спрощення запису аналогічно підходу з інтегруванням визначимо ці вектори глобальними.

```
function [ yr ] = funur(a )
global xr y
yr=norm(y-cos(xr).^a,2); end
```

Скрипт визначення значення коефіцієнта косинусної апроксимації:

```
clear
global xr y
x=[0 8 15 17 20 30 45 60];
y=[1 0.91 0.70 0.62 0.51 0.33 0.14 0.089];
xr=pi/180*x;
fzero(@funur,1)
fsolve(@funur,1)
a=fsolve(@funur,1);
er=norm(y-cos(xr).^a,2)/(length(x)-1);
fprintf('a= %g errror=%g',a,er)
```

Функція **fzero** не знайшла значення, бо умовою її використання є гарантований перетин осі абсцис. Апроксимаційна функція має мінімум, який наближається до осі, але не перетинає її.

```
    Exiting fzero: aborting search for an interval
containing a sign change because NaN or Inf function
value encountered during search.
    (Function value at -1309.72 is Inf.)
    Check function or try again with a different
starting value.
    ans =    NaN
```

Функція **fsolve** видала попередження про відсутність кореня рівняння, але визначила точку, в якій апроксимаційна функція має мінімум, який максимально наближається до осі.

```
    No solution found.
    fsolve stopped because the problem appears regular
as measured by the gradient,but the vector of function
values is not near zero as measured by the default
value of the function tolerance.
    <stopping criteria details>

    a= 9.18274 error=0.0234597
```

Значення співпадають з результатами MathCAD.

Знаходження коефіцієнта гаусової апроксимації проводиться заміною функції рівняння:

```
function [ yr ] = funur(a )
global xr y
%yr=norm(y-cos(xr).^a,2);
yr=norm(y-exp(-(xr/a).^2),2);
end
```

Скрипт визначення значення коефіцієнта гаусової апроксимації:

```
clc
clear
global xr y
xr=[0 8 15 17 20 30 45 60];
y=[1 0.91 0.70 0.62 0.51 0.33 0.14 0.089];
a=fsolve(@funur,10);
er=norm(y-exp(-(xr/a).^2),2)/(length(xr)-1);
```

```
fprintf('a= %g errror=%g',a,er)
```

Відповідь співпадає з результатами MathCAD.

a= 26.4555 errror=0.0213389.

Пакет не має розмірностей змінних, тому всі вихідні дані приводяться до однієї розмірності. Функції інтегрування не сприймають вектори в якості аргументів, тому для розрахунку в діапазоні діаметрів зіниць введено цикл.

Підінтегральну функцію описано як функцію користувача **mintal:**

```
function [ y ] = mintal( x )
y=exp(-((x.*180./pi).^2/26.456^2)).*sin(x);
end
```

Скрипт для визначення сили випромінення, коефіцієнта ефективності ОС:

```
clc
D=10:10:50;
l=1000;
p=10;
io=p/quad(@mintal,0,pi/2);
a=atan(D./2./l);

for i=1:length(D)
    F(i)=io*quad(@mintal,0,a(i));
    fprintf('%g   %6.3g   %6.3g   %6.3g\n',D(i),F(i),...
F(i)/p,a(i)*180/pi)
end
```

Результат з виведенням по рядках значення діаметра зіниці в мм, потоку на зіниці в мВт, коефіцієнта ОС та апертурного кута в градусах:

```
10 0.0012 0.00012   0.29
20 0.0049 0.00049   0.57
30  0.011 0.0011   0.86
40  0.019 0.0019    1.1
50   0.03  0.003    1.4
```

Завдання 8.1. Розрахувати силу випромінення та індикатрису з поліноміальною, ступеневою косінусною, гаусовою апроксимаціями для випромінювача HDSL4251 з променевим потоком F=100 мВт.

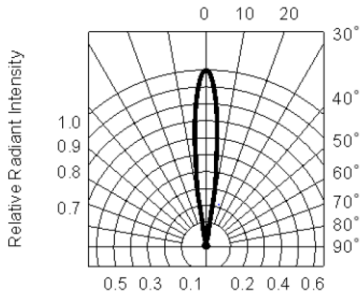


FIG.7 RADIATION DIAGRAM

α[град]	0	2	5	7	10
Io	1.000	0.932	0.751	0.449	0.117

Література

1. Струтинський В.Б. Математичнс моделювання процесів та систем механіки: Підручник.- Житомир: ЖІТІ, 2001. - 612 с.
2. Лазарєв Ю. Ф., Моделювання на ЕОМ. Навчальний посібник. - К.: Політехніка, 2007. - 290 с.
3. Інформаційні технології: Системи комп'ютерної математики [Електронний ресурс] : навч. посіб. для студ. спеціальності «Автоматизація та комп'ютерно-інтегровані технології» / І. В. Кравченко, В. І. Микитенко; КПІ ім. Ігоря Сікорського . - Електронні текстові дані (1 файл: 5,57 Мбайт). - Київ : КПІ ім. Ігоря Сікорського, 2018. - 243с.
4. Половко А. М., Бутусов П. Н., MATLAB для студента. - СПб.: БХВ-Петербург, 2005. - 320 с.
5. Банди Б. Методы оптимизации. Вводный курс: Пер. с англ. -М.: Радио и связь, 1988. - 128с.
6. K. D. Moller, Optics Learning by Computing, with Examples Using Mathcad, Matlab, Mathematica and Maple. Second Edition.-Springler, 2007.- 459p.