Лабораторна робота № 7.

Тема: Розв'язання нелінійних рівнянь та систем нелінійних рівнянь у системі **MATLAB**.

Мета роботи: ознайомитись з деякими методами розв'язування нелінійних рівнянь та систем нелінійних рівнянь.

Теоретичний мінімум

Нелінійні рівняння можна розділити на два класи — алгебраїчні та трансцендентні [24].

Визначимо дві групи методів розв'язання нелінійних рівнянь:

- точні методи;
- ітераційні методи.

Точні методи дозволяють записати корені у вигляді деякого скінченого співвідношення (формули). В курсі алгебри з такими методами ми зустрічалися при розв'язуванні тригонометричних, логарифмічних, показникових, а також алгебраїчних нелінійних рівнянь.

Вправа 1. Розв'язування нелінійних рівнянь на основі методу бісекції.

Більшість трансцендентних рівнянь, також як і систем цих рівнянь, не мають аналітичних розв'язків. Доведено, що не можна побудувати формулу, за якою можна знайти корені довільного алгебраїчного рівняння степені вище ніж четверта. Крім того, в деяких випадках рівняння містять коефіцієнти, які відомі лише наближено, і, отже, сама задача про точне визначення коренів рівняння втрачає сенс. Для розв'язування таких задач використовуються ітераційні методи із заданим ступенем точності.

Нагадаємо, що всяке значення $x = \xi$, що обертає функцію $f(x)$ в нуль, тобто $f(\xi) = 0$, називається коренем рівняння

$$f(x) = 0 \qquad (1)$$

або нулем функції $f(x)$.

Ітераційний процес полягає в послідовному уточненні початкового наближення $x(0)$ можливого значення кореня рівняння (1). Кожний такий крок називається ітерацією. В результаті ітерацій знаходиться послідовність наближених значень кореня $x(1), x(2), ..., x(n)$. Якщо ці значення із збільшенням числа ітерацій n наближаються до точного значення кореня, то кажуть, що ітераційний процес збігається.

Передбачаємо, що функція $f(x)$ в рівнянні (1) задовільняє наступним вимогам:

1) функція $f(x)$ неперервна на відрізку $[a,b]$ разом з своїми похідними 1-го і 2-го порядку;

2) значення функції $f(x)$ на кінцях відрізку мають різні знаки, тобто $f(a)*f(b)<0$;

3) перші та другі похідні функції, $f'(x)$ та $f''(x)$, зберігають певний знак на всьому відрізку $[a, b]$.

Умови 1) і 2) гарантують, що на інтервалі $[a,b]$ знаходиться хоча б один корінь, а з умови 3) витікає, що функція $f(x)$ на даному інтервалі є монотоною і тому корінь рівняння (1) буде єдиним.

Задача знаходження кореня рівняння $f(x)=0$ ітераційним методом вирішується в два етапи:

1. Визначаються корені, тобто відшукується наближене значення кореня або відрізка, що його містить;
2. Уточнюється наближене значення коренів, тобто наближені величини коренів доводяться до заданої степені точності.

Передбачимо, що наближене значення кореня $\bar{x}$ необхідно отримати з точністю ε, тобто знайти таке значення $\bar{x}$, щоб для точного значення кореня x мала місце наступна нерівність: $|x-\bar{x}|<\varepsilon$.

Процес визначення коренів починається зі встановлення знаків функції $f(x)$ в граничних точках області її існування: $x=a$ та $x=b$.

Вирішення рівняння (1) ітераційним методом передбачає встановлення відповідей на питання: чи має рівняння (1) корені і скільки їх існує, а також виявлення процедури знаходження значень вказаних коренів з потрібною точністю.

Для наближеного знаходження коренів рівняння (1) можна застосовувати метод бісекції.

Нехай $[a,b]$ — відрізок, що містить шуканий корінь. Допустимо, що функція $f(x)$ неперервна на відрізку $[a,b]$ і на кінцях відрізка приймає значення різних знаків: $f(a)\cdot f(b)<0$. Алгоритм методу бісекції полягає в побудові послідовності вкладених відрізків, на кінцях яких функція приймає значення різних знаків. Кожен наступний відрізок отримують в результаті ділення попереднього відрізку навпіл.

Нехай на k-ому кроці даного алгоритму знайдений такий відрізок $[a^{(k)}, b^{(k)}]$, що $f(a^{(k)})\cdot f(b^{(k)})<0$. Знайдемо середину цього відрізку $c^{(k)}=\dfrac{a^{(k)}+b^{(k)}}{2}$.

Якщо $f(c^{(k)})=0$, то $c^{(k)}$ буде шуканим коренем.

Якщо ця умова не виконується, то з двох частин відрізку вибираємо ту, на кінцях якої функція $f(x)$ має протилежні знаки. Вибираємо $[a^{(k+1)}, c^{(k)}]$,

коли $f(a^{(k)}) * f(c^{(k)}) < 0$, де $a^{(k+1)} = a^{(k)}$, $b^{(k+1)} = c^{(k)}$, або відрізок $[c^{(k)}, b^{(k+1)}]$, коли $f(c^{(k)}) * f(b^{(k)}) < 0$, де $a^{(k+1)} = c^{(k)}$, $b^{(k+1)} = b^{(k)}$.

Наближені значення коренів можуть бути встановлені з фізичного формування задачі, з процедури розв'язування аналогічної задачі при інших початкових даних або за допомогою графічного аналізу, що застосовують для визначення наближених коренів.

Приклад 1. Знайти корені рівняння $f(x) = x^3 - 10x - 10 = 0$.

Множину існування функції $f(x)$ розбиваємо на підмножини:

$$(-\infty, -4],\ (-4, -2],\ (-2, 0],\ (0, 2],\ (2, 4],\ (4, +\infty).$$

Встановимо знаки функції $f(x)$ в граничних точках підмножин у випадках, коли $x \to -\infty$ та $x \to +\infty$:

x	$-\infty$	–4	–2	0	2	4	$+\infty$
$f(x)$	–	–	+	–	–	+	+

З даної таблиці витікає, що корені функції можуть міститися в наступних відрізках [–4, –2], [–2, 0] і [2, 4].

Зауважимо, що дійсні корені рівняння (1) — це точки перетину графіка функції $f(x)$ з віссю абсцис. Процедура встановлення координат коренів передбачає побудову графіка функції $f(x)$, визначення точок перетину графіка функції $f(x)$ з віссю Ох або визначення на вісі Ох відрізків, що містять один нуль функції.

Тепер розглянемо розв'язання рівняння $f(x) = 0$: $x^3 - 10x - 10 = 0$ за допомогою команд системи **MATLAB**. Для завдання інформації про функцію $f(x)$ використовуємо команду **inline**. Локалізуємо корні рівняння:

```
>> f = inline('x.^3–10.*x–10');
```

Побудуємо графік функції на відрізку [–10,10]:

```
>> x=linspace(–10,10,100);
>> plot(x,f(x));grid
```

Визначаємо відрізки, що вміщують можливі нулі функції і будуємо графікі функції $f(x)$ послідовно на визначених відрізках [–4, –2], [–2, 0] і [2, 4]. Визначаємо корінь на відрізку [–4, –2]:

```
>> x=linspace(–4,–2,100);
>> plot(x,f(x));grid
```

Визначаємо корінь на більш меньшому відрізку [–2.6,–2.4]:

```
>> x=linspace(–2.6,–2.4,1000); plot(x,f(x));grid
```

Шуканий корінь відрізку [–4, –2]:

```
x1=–2.425.
```

Робимо перевірку знайденого кореня:

```
>> f(–2.425)
ans =
  –0.0105
```

У такій же спосіб знайдемо інший корінь. Визначаємо корінь на відрізку [–2, 0]:

```
>> f = inline('x.^3–10.*x–10');
x=linspace(–4,–2,100);
plot(x,f(x));grid
>> x=linspace(–2,0,100);
>> plot(x,f(x));grid
>> x=linspace(–1.2,–1,100);
>> plot(x,f(x));grid
>> x=linspace(–1.16,–1.15,100);
>> plot(x,f(x));grid;
>> f(–1.155)
```

Перевіримо знайдений корень:

```
ans =
   0.0092
```

Знаходження третього кореня. Визначаємо корінь на відрізку [2, 4]:

```
>> x=linspace(2,4,100);
>> plot(x,f(x));grid
>> x=linspace(3.4,3.6,100);
>> plot(x,f(x));grid
```

Перевіряємо знайдений корінь:

```
>> f(3.5771)
ans =
  2.9962e–004
```

Побудову графіків часто вдається спростити, якщо замінити ліву частину рівняння (1) рівнозначною функцією $f(x) \equiv f_1(x) + f_2(x)$. Тоді рівняння (1) набуває вигляду

$$f_1(x) + f_2(x) = 0,$$

де функції $f_1(x)$ і $f_2(x)$ – значно простіші, ніж функція $f(x)$. Якщо побудувати графіки функцій $y = f_1(x)$ і $y = -f_2(x)$, то шукані корені визначаються як абсциси точок перетину вказаних графіків.

Приклад 2. Треба визначити корені рівняння: $x \lg x = 1$.

Рівняння зручно переписати у вигляді рівності: $\lg x = \frac{1}{x}$. Звідси ясно, що корені рівняння можуть бути знайдени як абсциси точок перетину логарифмічної кривої $y = \lg x$ і гіперболи $y = \frac{1}{x}$. Після побудови графіків, можна наближено встановити координату точки перетину, тобто координату шуканого кореня вихідного рівняння $x \lg x = 1$, $x \approx 1.76$ або визначити відрізок [1.75, 1.78], який містить в собі цей корінь.

Реалізуємо цю процедуру знаходження кореня за допомогою команд системи **MATLAB**:

```
>> x=1.5:.01:2;
>> f1=log(x);f2=1./x;plot(x,f1,x,f2);grid
```

Локалізуємо корінь:

```
>> x=1.75:.001:1.78;
>> f1=log(x);f2=1./x;plot(x,f1,x,f2);grid
```

Приклад 3. Знайти корені рівняння $f(x)=0$, де $f(x)=\dfrac{x^2-6}{x+6}$.

Пошук кореня за процедурою з застосуванням команд системи **MATLAB:**

```
>> x=– 5:1:5; f1=x.^2–6;f2=x+6;
>> plot(x,f1,x,f2);grid
```

Приклад 4. Знайти корені рівняння $x^2+6=e^{-x}$.

Пошук кореня за процедурою з застосуванням команд системи **MATLAB:**

```
>> x=–3:.1:–1; f1=exp(–x);f2=x.^2+6;
>> plot(x,f1,x,f2);grid
```

Вправа 2. Розв’язування нелінійних рівнянь за допомогою пошуку нулів функції.

Для розв’язання рівнянь виду $f(x)=0$ (тобто для знаходження нулів функції) існує команда **fzero**, яка дозволяє знаходити наближене значення кореня за заданим початковим значенням [44]:

- **z=fzero(fun ',x0),** структура команди дозволяє знаходить нуль функції в околі точки **x0**;
- **z=fzero(fun,x0,tol),** структура команди **fzero** повертає результат з відносною похибкою **tol**, величина якої задається користувачем (у випадку замовчування **tol=eps**);
- **z=fzero(fun,x0,tol,trace)** структура команди **fzero** дозволює видавати на екран поточні послідовні значення процедури пошуку нуля функції.

За допомогою цієї команди **fzero** реалізуються метод ділення відрізка навпіл, метод січних та зворотньої квадратичної інтерполяції.

Параметр **fun** команди **fzero** може бути заданий одним з наступних способів:

- як формула з невідомим **x**, яка ув’язнена в апострофи;
- як ім'я **m**-файлу (у апострофах і без розширення);
- як показник на функцію (наприклад **@fun**_name);

Формула, ув’язнена в апострофи, розглядається як незалежна змінна і може містити тільки **x**. Використання незалежної змінної з іншим ім'ям приведе до появи інформації про помилку.

Параметр **x0** команди **fzero** може бути заданий одним з двох способів:

• у вигляді вектора **[a b]**, що задає інтервал **(a<b)**, на кінцях якого параметр **fun** має різні знаки, що гарантує знаходження, принаймні одного кореня на цьому інтервалі;

• у вигляді скалярного значення, в околі якого передбачається знаходження шуканого кореня; в цьому випадку команда **fzero** сама намагається знайти відрізок з центром в заданій точці **x0**, на кінцях якого параметр **fun** має різні знаки.

Щоб полегшити роботу по вибору початкового наближення, рекомендується побудувати графік функції **y=fun(x).**

Команда **fzero** може повертати ще два вихідних параметра **e_flag, inform: [x,f,e_flag,inform] = fzero(fun,x0).**

Додатні значення **e_flag** (зазвичай це l) означає, що вдалося знайти інтервал, на кінцях якого параметр **fun** має різні знаки. Якщо такий інтервал не виявлений, то **e_flag= –1**.

Структура **inform** містить три поля з іменами **iterations, funcCount** і **algorithm**.

У першому полі записується кількість ітерацій, що були виконані під час пошуку шуканого кореня; у другому полі — кількість звернень до параметру **fun**, в третьому полі — найменування алгоритму, що був використаний для знаходження кореня.

Приклад 1. Знайти корінь рівняння $5xe^{-x} + 5\cos x = 0$ на відрізку $[-1, 1]$.

Застосуємо команду **fzero** з додатковими параметрами виходу:

```
>> [x,f,e_flag,inform]=fzero('5.*x.*exp(–x)+5*cos(x)',[–1,1])
x =
  –0.5178
f =
    0
e_flag =
    1
inform =
   iterations: 11
    funcCount: 11
    algorithm: 'bisection, interpolation'
```

Приклад 2. Знайти корені рівняння $x^3 - 10x - 10 = 0$.

Скористаємось командою **fplot** для побудови графіку вказанної функції на відрізку [–4 4]:

```
>> fplot('x.^3–10.*x–10',[–4 4]);
>> grid
```

```
>> x1=fzero('x.^3–10.*x–10', –2.5)
x1 =
  –2.4236
```
Обчислення точності знайденого рішення:
```
>> disp(eps)
  2.2204e–016
```
Задана машинна точність:
```
>> f(x1–eps)*f(x1+eps)
ans =
     0
>> [x2,f]=fzero('x.^3–10.*x-10',-1.5)
x2 =
  –1.1535
f =
  1.7764e-015
>> [x3,f,e_flag,inform]=fzero('x.^3–10.*x–10',3.5)
x3 =
    3.5771
f =
–1.4211e–014
e_flag =
     1
inform =
    iterations: 8
     funcCount: 8
     algorithm: 'bisection, interpolation'
```

Зауважимо, що точність знаходження коренів у системі **MATLAB** достатньо висока. Користувачеві надається можливість формувати різні умови припинення ітераційного процесу – згідно точності обчислення координати **x**, по модулю значення параметра **fun**, згідно кількості звернень до параметру **fun** і т.д.

Умова виявлення інтервалу, на кінцях якого функція приймає значення різних знаків, є принциповим для алгоритму, який використовується командою **fzero**. Навіть у таких тривіальних рівняннях, як $x^4 = 0$, зазвичай не вдається знайти шуканий розв’язок:

```
>> [x,f,e_flag,inform]=fzero('x.^4',[–1,1])
??? Error using ==> fzero
The function values at the interval endpoints must differ in sign.
```

Відмова щодо виконання команди відбулася через те, що на обох кінцях заданого інтервалу функція приймає додатні значення. Якщо початкове наближення "випадково" співпадає з коренем, то команда **fzero** повертає це випадкове значення (яке може збігатись з розв’язком):

```
>> [x,f]=fzero('x.^4',0)
x =
     0
f =
     0
```

Проте зміна стартової точки знову приводить алгоритм до невизначеності. Спроба розширити інтервал з центром в точці **x=1** приводить до появи нескінченно великих значень функції, але інтервал, на якому відбулася би зміна знаку, так може і не бути виявлений (звернить увагу на значення вихідного параметра **e_flag**):

```
>> [x,f,e_flag]=fzero('x.^4',1)
Exiting fzero: aborting search for an interval containing a sign change
   because NaN or Inf function value encountered during search
(Function value at –1.482139e+077 is Inf)
Check function or try again with a different starting value.
x =
   NaN
f =
   NaN
e_flag =
   –1
```

Існує ще один спосіб вирішення рівнянь – це графічне знаходження коренів рівняння за допомогою команди **ginput**.

Приклад 3. Розв’язати нелінійне рівняння $x\sin x^2 = 0$.

Процедура знаходження коренів визначається наступними командами:

```
>> x=0:.01:3;
>> f=x.*sin(x.^2);
>> plot(x,[f;0*f]); grid
>> ginput
```

Після виконання цих команд з’являється графічне вікно з двома рухомими лініями, які перетинаються під кутом 90 градусів. За допомогою цих ліній можна визначити точки перетину. Як тільки усі точки були визначені, треба ініціювати на клавіатурі клавішу **ENTER**, після чого наближено обчислюються шукані корені:

```
ans =
   2.5127   –0.0088
   1.7730   –0.0088
   0.0035   –0.0088
```

Вправа 3 Розв'язання системи нелінійних рівнянь за допомогою команди **fsolve.**

Формально задача пошуку розв'язку системи нелінійних рівнянь

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ ..... \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases}$$

може бути представлена в еквівалентній форми, як задача пошуку кореня одного рівняння, де $f(x) = (f(x)_1, f_2(x), ..., f_n(x))$ і $x = (x_1, x_2, ..., x_n)$.

Команда **fsolve** призначена для розв'язання систем нелінійних рівнянь виду $f(x) = 0$, де x — вектор невідомих, а f — функція, значенням якої є вектор або матриця. Алгоритм роботи команди **fsolve** використовує початкове значення x_0 і базується на мінімізації суми квадратів складових функції $f(x) = (f(x)_1, f_2(x), ..., f_n(x))$ методами Гауса-Ньютона і Льовенберга-Марквардта [46]. У простому випадку структура звернення до команди **fsolve** має вигляд**: X = fsolve(F,X0).**

Команду **fsolve** можна використовувати і як альтернативу функції **fzero** при знаходженні кореня нелінійного рівняння, наприклад, $\cos x = 0$:

```
>> x=fsolve(@cos,1)
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.
x =
   1.5708
```

Другий параметр команди **fsolve** може бути заданий у вигляді вектора початкових значень, і для кожного компоненту цього вектора будуть знайдені наближені розв'язки. Наприклад:

```
>> x=fsolve(@cos,[1 2 3 4 5],optimset('fsolve'))
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.
x =
   1.5708    1.5708    1.5708    4.7124    4.7124
```

На відміну від **fzero** команда **fsolve** може знайти наближений розв'язок для вище розглянутого рівняння $x^4 = 0$:

```
>> x=fsolve('x.^4',1,optimset('Display','off'))
x =
   0.1001
```

У випадку розв'язування рівняння $tgx = 0$ команда **fsolve** може знайти наближений розв'язок і не схибити в околі точки розриву функції:

```
>> x=fsolve(@tan,1,optimset('Display','off'))
x =
  2.3205e–010
```

Сутність параметра **'Display'** та **'off'** полягає в викресленні "зайвих" повідомлень.

Головним призначенням команди **fsolve** є розв'язок систем нелінійних рівнянь.

Приклад 1. Розв'язати систему рівнянь: $\begin{cases} x_1 = -x_2 + \sin(\pi x_1) \\ x_1 = \ \ x_2 + \cos(\pi x_2) \end{cases}$.

Випишимо **m**-файл **funsc**, значення якого сформуємо у вигляді вектора-стовпця:

```
function y = funsc(x)
y=[x(1)+x(2)–sin(pi*x(1)); x(1)–x(2)–cos(pi*x(1))];
```

Використовуємо команду **fsolve.** Зауважимо, що кожне звернення до цієї команди передбачає завдання різних початкових значень **(xl, x2)**. Оскільки команда **fsolve** подібно команді **fzero** повертає і вектор-стовпець значень функцій в знайденій точці, то можна сформувати два вихідних параметра, щоб оцінити точність розв'язання. Звернемо увагу на те, що координати початкової точки теж представлені у вигляді вектора-стовпця:

```
>> [x,f] = fsolve(@funsc,[0.2;1],optimset('Display','off'))
x =
   0.5000
   0.5000
f =
  1.0e–007 *
   0.2139
  –0.0000
```

Результати подальших звернень до команди **fsolve** з різними початковими значеннями (x1, x2) наведені в таблиці 8.1:

Таблиця 8.1 – Результати обчислень

№ п/п	x0[x1;x2]		Розв'язок системи		Значення функції (з поточною точністю)	
	x1	x2	x1	x2	y1	y2
1	0.2	1	0.5	0.5	0.2139e-007	0
2	1	0.5	0.5	0.5	0.4694e-008	0
3	1	–0.2	0.5	0.5	0.9686e-012	–0.0012e-0.12
4	–0.2	0	–0.5	–0.5	–0.4223e-007	0.0001e-007
5	–0.5	–1	–0.5	–0.5	0	–0.6123e-016
6	–1	–2	–0.5	–0.5	–0.1604e-006	0.0001e-006

Розібратися з виписаними ситуаціями допоможуть графіки функцій, які отримані шляхом розв’язування рівнянь вихідної системи відносно відповідних змінних:

```
>>axes('Xlim', [–1 1.5] , 'Ylim', [–1 1.5]);axis equal; grid on; hold on
x1=–1:0.1:1.5; y=sin(pi*x1)–x1; plot(x1,y,'g-');
x2=–1:0.1:1.5; y=cos(pi*x2)+x2; plot(y,x2,'r:');
xlabel('x1'); ylabel('x2');
title('Розв’язок нелінійної системи','FontName','Courier');
```

Розв’язок системи – точки перетинання двох графіків.

Серед вихідних параметрів команди **fsolve** можуть з’явитися такі ж за змістом параметри, які дозволяють використовувати структуру команди **fzero**: **[x,f,exitflag,output] = fsolve(fun,xO).**

Значення параметру **exitflag=1** свідчить про те, що розв’язок системи знайдений. При **exitfiag=0** розв’язок не був знайдений, оскільки ітераційний процес був перерваний у зв'язку з досягненням максимальної кількості ітерацій або звернень до параметру **fun**. При **exitfiag= –1** досягнутий мінімум не є розв’язком системи.

У структурі **output** з'являються додаткові поля з іменами:

- **stepsize**, в якому видається величина кроку при завершенні пошуку;
- **firstorderopt**, в якому видається точність, досягнута згідно значенням функції.

Приклад 2. Розв’язати систему приклада 1 $\begin{cases} x_1 = -x_2 + \sin(\pi x_1) \\ x_1 = \ x_2 + \cos(\pi x_2) \end{cases}$ з додатковіими вихідними параметрами e_flag,inform.

Процедура розв’язання використовує наступні команди:

```
>> [x,f,e_flag,inform] = ...
fsolve(@funsc,[1;0.5],optimset('Display','off'))
x =
    0.5000
    0.5000
f =
  1.0e–008 *
    0.4694
   –0.0000
e_flag =
     1
inform =
       iterations: 5
        funcCount: 15
        algorithm: 'trust-region dogleg'
    firstorderopt: 4.6948e–009
```