Лабораторна робота 12. Розв'язування систем нелінійних рівнянь

Зміст заняття: Розв'язування системи нелінійних рівнянь методом простих ітерацій і методом Ньютона.

Загальні відомості

Систему нелінійних рівнянь у загальному випадку можна зобразити у вигляді

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0, \\ f_2(x_1, x_2, ..., x_n) = 0, \\ ..... \\ f_n(x_1, x_2, ..., x_n) = 0, \end{cases} \quad (1)$$

тобто як n функцій f_i від n невідомих x_i, причому функції f_i не обов'язково лінійно залежать від змінних x_i.

Позначимо вектор змінних через $X = (x_1, x_2, ..., x_n)$,а вектор функцій через $F = (f_1, f_2, ..., f_n)$. Тоді систему (1) можна записати у формі

$$F(X) = 0. \quad (2)$$

Завдання полягає в тому, щоб знайти розв'язок цієї системи. Слід сказати, що на даний час не існує математичної теорії, яка дозволяла б у загальному вигляді розв'язати питання про існування та число розв'язків системи (2). Їх може не бути зовсім, може бути один, декілька або нескінченна множина. Крім того, важливою особливістю системи нелінійних рівнянь є те, що для їх розв'язування не можна використати прямі методи, зокрема, метод послідовного виключення невідомих. Усі розроблені методи є ітераційними, а найефективнішим і широко вживаним є метод Ньютона.

1. Метод простих ітерацій

Із вихідної системи (1) шляхом еквівалентних перетворень переходимо до системи виду:

$$\begin{cases} x_1 = \Phi_1(x_1, \ldots, x_n) \\ \ldots \\ x_n = \Phi_n(x_1, \ldots, x_n) \end{cases} \quad (3)$$

Ітераційний процес, який визначається формулами

$$\begin{cases} x_1^{k+1} = \Phi_1\left(x_1^{(k)}, \ldots, x_n^{(k)}\right) \\ \ldots \\ x_n^{k+1} = \Phi_n\left(x_1^{(k)}, \ldots, x_n^{(k)}\right) \end{cases}, \; k = 0,1,\ldots$$

можна почати, задавши початкове приближення $x^{(0)} = \left(x_1^{(0)}, \ldots, x_n^{(0)}\right)$.

Достатньою умовою збіжності ітераційного процесу є одна з двох умов:

$$\sum_{j=1}^{n}\left|\frac{\partial\Phi_i}{\partial x_j}\right|<1,\forall i=\overline{1,n} \quad \text{або} \quad \sum_{i=1}^{n}\left|\frac{\partial\Phi_i}{\partial x_j}\right|<1,\forall j=\overline{1,n}.$$

Розпишемо першу умову:

$$\left|\frac{\partial\Phi_1}{\partial x_1}\right|+\left|\frac{\partial\Phi_1}{\partial x_2}\right|+\ldots+\left|\frac{\partial\Phi_1}{\partial x_n}\right|<1 \quad \text{при } i=1$$

$$\left|\frac{\partial\Phi_n}{\partial x_1}\right|+\left|\frac{\partial\Phi_n}{\partial x_2}\right|+\ldots+\left|\frac{\partial\Phi_n}{\partial x_n}\right|<1 \quad \text{при } i=n.$$

Розпишемо другу умову:

$$\left|\frac{\partial\Phi_1}{\partial x_1}\right|+\left|\frac{\partial\Phi_2}{\partial x_1}\right|+\ldots+\left|\frac{\partial\Phi_n}{\partial x_1}\right|<1 \quad \text{при } j=1$$

$$\left|\frac{\partial\Phi_1}{\partial x_n}\right|+\left|\frac{\partial\Phi_2}{\partial x_n}\right|+\ldots+\left|\frac{\partial\Phi_n}{\partial x_n}\right|<1 \quad \text{при } j=n.$$

Ітераційний процес знаходження розв'язку системи (1) за методом простих ітерацій чи Ньютона є нескінченним. Тому для знаходження розв'язку з заданою точністю ε, як правило використовують наступну умову зупинки:

$$|X^{k+1}-X^k|\le\varepsilon$$

Приклад 12.1. Описати систему нелінійних рівнянь:

$$\begin{cases} f_1(x_1,x_2,x_3)=x_1+e^{x_1-1}+(x_2+x_3)^2-27=0 \\ f_2(x_1,x_2,x_3)=x_1\cdot e^{x_2-2}+x_3{}^2-10=0 \\ f_3(x_1,x_2,x_3)=x_3+\mathrm{Sin}(x_2-2)+x_2{}^2-7=0 \end{cases}.$$

Розв'язок цієї системи:

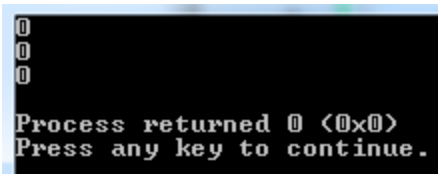
$$X=(x_1;x_2;x_3):\begin{cases}x_1=1.0\\x_2=2.0\\x_3=3.0\end{cases}$$

Програмна реалізація:

```
#include <iostream>
#include <cmath>

using namespace std;

float F(float X[], int n) //система рівнянь, n - номер рівняння
{
  switch (n)
  {
```

```
   case 0: return X[0] + exp(X[0]-1) + pow(X[1]+X[2],2) - 27; break;
   case 1: return X[0]*exp(X[1]-2) + pow(X[2],2) - 10; break;
   case 2: return X[2] + sin(X[1]-2) + pow(X[1],2) - 7; break;
  }
}

int main()
{
  int n = 3;
  float X[n]={1.0, 2.0, 3.0}; // розв'язок системи

  for (int i=0; i<n; i++)
  {
    cout<< F(X, i)<<endl; // виклик функції F
  }
  return 0;
}
```

Результат виконання програми:



Метод простих ітерацій є найбільш універсальним і простим для реалізації. Якщо система має великий порядок, то застосування даного методу, який має повільну швидкість збіжності, не рекомендується. В цьому випадку, використовують метод Ньютона, який має швидшу збіжність.

2. Метод Ньютона

Головна ідея методу Ньютона полягає у вилученні з рівнянь (1) лінійних частин, які є головними за малих приростів (збурень) аргументів. Такий крок дає змогу звести початкову задачу до розв'язання послідовності лінійних систем. Метод Ньютона для N рівнянь можна застосувати тільки тоді, коли можна обчислити всі частинні похідні функцій f_i за змінними x_i.

Позначимо через *Ja (X)* матрицю частинних похідних системи *F* по змінних X . Таку матрицю називають матрицею Якобі:

$$Ja\left(x^{(k)}\right)=\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}.$$

Її (i, j) –м елементом є значення похідної

$$\frac{\partial f_i}{\partial x_j}$$ у точці x_j.

З метою побудови формули ітераційного процесу Ньютона лінеаризують $F(X)$ в околі точки $X^{(0)}$, яку називають точкою початкового наближення. Лінеаризацію здійснюють, розкладаючи $F(X)$ в ряд Тейлора і враховуючи лише перші два члени розкладу:

$$F(X) \approx F\left(X^{(0)}\right) + Ja(X^{(0)}) \cdot (X - X^{(0)}).$$

З метою обчислення наступного наближення $X^{(1)}$ до розв'язку системи (1) розв'яжемо рівняння

$$F\left(X^{(0)}\right) + Ja\left(X^{(0)}\right) \cdot \left(X - X^{(0)}\right) = 0$$

щодо вектора X . Позначаючи його через $X^{(1)}$, одержимо:

$$X^{(1)} = X^{(0)} - Ja^{-1}\left(X^{(0)}\right) \cdot F\left(X^{(0)}\right),$$

де $Ja^{-1}\left(X^{(0)}\right)$ – обернена матриця Якобі, обчислена для значення $X^{(0)}$.

Узагальнюючи останній вираз , отримаємо формулу ітераційного методу Ньютона:

$$\boxed{X^{(k+1)} = X^{(k)} - Ja^{-1}\left(X^{(k)}\right) \cdot F\left(X^{(k)}\right).} \quad (4)$$

У формулі (4) символом k позначено номер ітерації. У дужки ми його взяли для того, щоб відрізнити номер від операції піднесення до степеня.

Ітераційний метод Ньютона у формі (4) для розв'язання систем (1) використовувати не доцільно, оскільки обертання матриці Якобі є процедурою, з погляду кількості арифметичних операцій, доволі громіздкою.

На практиці формулу (4) замінюють двома виразами, що дає змогу замінити процедуру обертання матриці Якобі процедурою Гауса для розв'язання СЛАР.

Введемо позначення:

$$Dx^{(k)} = Ja^{-1}\left(X^{(k)}\right) \cdot F\left(X^{(k)}\right)$$

Вектор $Dx^{(k)}$ називають вектором нев'язки. Домножуючи вираз зліва і справа на матрицю Якобі, отримаємо

$$Ja\left(X^{(k)}\right) \cdot Dx^{(k)} = F\left(X^{(k)}\right) \quad (5)$$

Вираз (5) є системою лінійних алгебричних рівнянь щодо вектора невідомих $Dx^{(k)}$. Розв'язавши цю систему методом Гауса, використаємо її розв'язок в ітераційному процесі Ньютона:

$$\boxed{X^{(k+1)} = X^{(k)} - Dx^{(k)}}.$$

Процес вважатимемо закінченим, якщо

$$\left\|X^{(k+1)} - X^{(k)}\right\| = \left\|Dx^{(k)}\right\| \leq \varepsilon$$

де ε – задана нами точність обчислень.

Алгоритм методу Ньютона

Сформулюємо кроки алгоритму (Рис.12.1) розв'язання СНР методом Ньютона:

1. Задаємо і вводимо значення таких параметрів:
   - N – розмір системи (1).
   - Eps – необхідна точність обчислень.
   - Константа Kmax – максимально допустима кількість ітерацій. Якщо за Kmax ітерацій розв'язок не знайдено, процес обчислення розв'язку припиняємо і друкуємо повідомлення про те, що за задану кількість ітерацій розв'язок із необхідною точністю знайти не вдалось.
2. Вводимо вектор початкових умов $X^{(0)}$.
3. Запускаємо ітераційний цикл за змінною k = 1..Kmax. У цьому циклі:

- обчислюємо вектор вільних членів функцією F;
- обчислюємо матрицю Якобі Ja функцією JAC;
- знаходимо вектор *Dx* , розв'язуючи систему (5) процедурою GAUSS;
- виконуємо ітерацію Ньютона: $X^{(k+1)} = X^{(k)} - Dx^{(k)}$, циклом по компонентах вектора *X*;
- у цьому самому циклі визначаємо норму вектора *Dx* , як значення модуля його максимальної компоненти: $Dxmax = \left\|Dx^{(k)}\right\| = \max_k\left(\left|Dx^{(k)}\right|\right)$;
- якщо *DxMax* ≤ *Eps*, розв'язок *X* вважаємо знайденим, друкуємо його і завершуємо роботу програми. Якщо ця умова не справджується – виконуємо наступну ітерацію, тобто повертаємо управління на початок нашого циклу.

4. Якщо за задане число Kmax ітерацій збіжності не досягнуто, друкуємо щодо цього відповідне повідомлення і завершуємо роботу програми.

Блок-схему наведеного вище алгоритму подано на рис. 12.1.

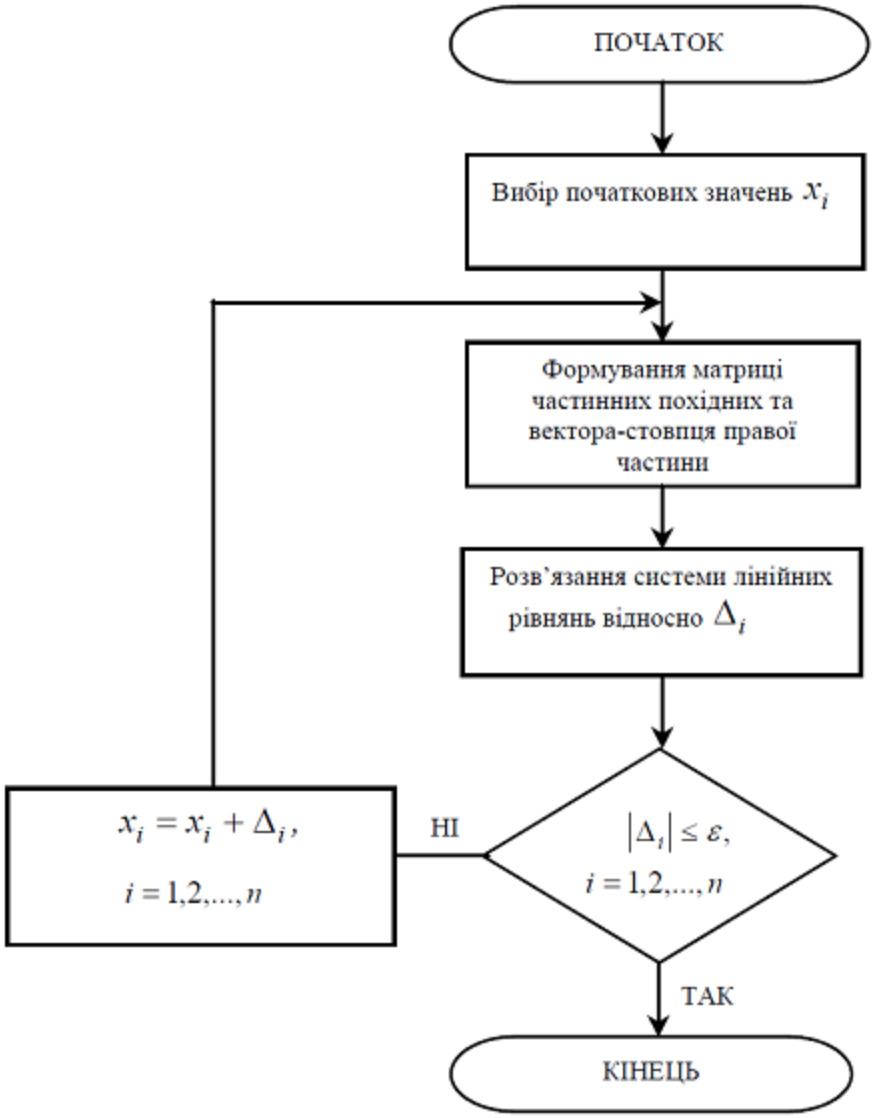


Рисунок 12.1 – Алгоритм методу Ньютона для систем нелінійних рівнянь

Під час реалізації нашого алгоритму виникає закономірне питання з приводу способу обчислення матриці Якобі. Якщо обчислення похідної функції вже в одновимірному випадку є непростою задачею, то для системи N рівнянь обчислення *Ja (X)* значно трудомісткіше, ніж обчислення *F(X)*. Звичайно, розв'язуючи автономно нашим алгоритмом нескладну СНР,

можна обчислити матрицю Якобі вручну, описуючи двовимірний масив і присвоюючи кожному його (i,j)-му елементові вирази відповідних похідних. Це буде надто громіздким і не універсальним процесом.

На практиці елементи матриці Якобі визначають чисельно, виходячи з означення похідної:

$$\frac{\partial f_1}{\partial x_1} = \frac{f_1(x_1+\delta, x_2, \dots, x_N) - f_1(x_1, x_2, \dots, x_N)}{\delta}$$

$$\frac{\partial f_1}{\partial x_2} = \frac{f_1(x_1, x_2+\delta, \dots, x_N) - f_1(x_1, x_2, \dots, x_N)}{\delta}$$

...

$$\frac{\partial f_2}{\partial x_1} = \frac{f_2(x_1+\delta, x_2, \dots, x_N) - f_2(x_1, x_2, \dots, x_N)}{\delta}$$

...

Тут δ – малий приріст змінної (збурення), який доцільно вибирати таким: $\delta = 0.0001$.

Приклад 12.2. Обчислення матриці Якобі системи трьох нелінійних рівнянь.

```
float JAC(float X[], int n, int m)
// n - номер рівняння у чисельнику, m - номер змінної у знаменнику
{ float d = 0.0001;
  float y0, y;
  float Jaij;
  y0 = F(X,n); // обчислення вектора функції F від незбуреного вектора X
  X[m] = X[m] + d; // збурення m-ої компоненти вектора X
  y = F(X,n); // обчислення вектора функції F від збуреного вектора X
  Jaij = (y-y0)/d; // обчислення частинної похідної df_n/dx_m
  X[m] = X[m] - d; // зняття збурення m-ої компоненти вектора X
  return Jaij;
}

int main()
{
    const int n = 3;
    float X[n]={0.75, 1.5, 4.0}; // початкове наближення

...

    //Формування матриці Якобі
    float Ja[n][n];
    for (int i=0; i<n; i++)
    {
         for (int j=0; j<n; j++)
         {
             Ja[i][j] = JAC(X, i, j);
         }
    }
...
```

Завдання до лабораторної роботи

Розв'язати систему рівнянь методом простих ітерацій та Ньютона з заданою точністю. Початкове наближення знайти графічно. Порівняти кількість ітерацій в різних методах.

Варіант №	Система
12.1	$\begin{cases} \sin(x+1) - y = 1.2; \\ 2x + \cos y = 2. \end{cases}$
12.2	$\begin{cases} \cos(x-1) + y = 0.5; \\ x - \cos y = 3. \end{cases}$
12.3	$\begin{cases} \sin x + 2y = 2; \\ x + \cos(y-1) = 0.7. \end{cases}$
12.4	$\begin{cases} \cos x + y = 1.5; \\ 2x - \sin(y-0.5) = 1. \end{cases}$
12.5	$\begin{cases} \sin(x+0.5) - y = 1; \\ \cos(y-2) + x = 0. \end{cases}$
12.6	$\begin{cases} \cos(x+0.5) + y = 0.8; \\ \sin y - 2x = 1.6. \end{cases}$
12.7	$\begin{cases} \sin(x-1) = 1.3 - y; \\ x - \sin(y+1) = 0.8. \end{cases}$
12.8	$\begin{cases} 2y - \cos(x+1) = 0; \\ x + \sin y = -0.4. \end{cases}$
12.9	$\begin{cases} \cos(x+0.5) - y = 2; \\ \sin y - 2x = 1. \end{cases}$
12.10	$\begin{cases} \sin(x+2) - y = 1.5; \\ x + \cos(y-2) = 0.5. \end{cases}$
12.11	$\begin{cases} \cos(y-1) + x = 0.5; \\ y - \cos x = 3. \end{cases}$
12.12	$\begin{cases} \sin y + 2x = 2; \\ \cos(x-1) + y = 0.7. \end{cases}$
12.13	$\begin{cases} \cos y + x = 1.5; \\ 2y - \sin(x-0.5) = 1. \end{cases}$
12.14	$\begin{cases} \cos(x+0.5) - y = 2; \\ \sin y - 2x = 1. \end{cases}$