Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики

Лабораторна робота №2

Чисельні методи в інформатиці

"Розв'язок СЛАР прямим та ітераційними методами"

Варіант №8

Виконав студент групи ІПС-31
Тесленко Назар Олександрович

Київ - 2025

Постановка задачі:

Варіант №8

Розв'язати СЛАР наступними методами:

- Методом Гаусса розв'язати систему рівнянь, знайти визначник та обернену матрицю.

7	2	3	0		X1		20
0	3	2	6	*	X2	=	36
2	5	1	0		X3		15
0	1	4	2		X4		22

- Методом квадратного кореня розв'язати систему рівнянь, знайти визначник та число обумовленості, норму обрати самостійно

1	2	0		X1		5
2	2	3	x	X2	=	15
0	3	2		X3		12

- Методом Зейделя розв'язати систему рівнянь

4	0	1	0		X1		12
0	3	0	2	x	X2	=	19
1	0	5	1		X3		27
0	2	1	4		X4		30

Теоретичний опис та обґрунтування:

<u>Метод Гаусса</u>

Прямий метод вирішення СЛАР типу $Ax = b$

Розглядаємо даний алгоритм з вибором головного елементу по стовпцях. Ведучим елементом матрціі А обирається максимальний по модулю елемент стовпця, що розглядається:

$$a_{lk} = \max_i \left|a_{ik}^{(k-1)}\right|, \ i = \overline{k, n}$$

де:

$a_{ik}^{(k-1)}$ - елемент матриці після (k-1)-го кроку виключення

k - номер поточного стовпця

i - індекс рядка, який перебирається

l - номер рядка у якому знайдено максимальний за модулем елемент

Вводимо матрицю перестановок P_k яка ініціалізується одиничною матрицею, для перестановки рядків k та l:

$$\widehat{A}_k = P_k A_{k-1}$$

Для занулення елементів під головною діагоналлю використовується матрицю M. M матриця зберігає множники, що використовуються для виключенян елментів під головною діагоналлю під час прямого ходу. Кожен елемент цієї матриці обчислюється за формулою:

$$m_{ik} = \frac{a_{ik}}{a_{kk}}, \ i = \overline{k + 1, n}$$

$$m_{kk} = \frac{1}{a_{kk}}$$ - для елементів головної діагоналі

За допомгою прямого ходу:

$$M_n P_n \dots M_1 P_1 Ax = M_n P_n \dots M_1 P_1 b$$

зводимо систему до верхньої трукутньої матриці.

Для знаходження розв’язку застосовуємо зворотні хід Гаусса:

$$x_n = \frac{a_{n,n+1}^{(n)}}{a_{n,n}}$$

$$x_i = (a_{i,n+1}^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j) / a_{i,i}^{(i)}, \; i = \overline{n-1, 1}$$

Пошук визначника:

$\det A = (-1)^p a_{11}^{(1)} * a_{22}^{(2)} * ... * a_{nn}^{(n)}$, де p - кількість перестановок

Знаходження оберненої матриці:

Під час прямого ходу методу Гауса матриця A послідовно перетворюється до верхньотрикутної форми за допомогою матричних множників M_i та перестановочних матриць P_i , які відповідають перестановкам рядків при виборі головного елемента. Ті самі перетворення одночасно виконуються над одиничною матрицею E, у результаті чого вона поступово переходить у матрицю A^{-1}

Після прямого ходу отримуємо систему:

$$U * X = E'$$

Під час зворотнього ходу система розв'язується постовпчиково:

$$U * x_i = e'_i$$

Метод квадратного кореня

Прямий метод для розв'язування СЛАР типу Ax=b

Необхідна умова застосування: матриця A симетрична $A = A^T$

Матрицю A представимо у вигляді: $A = S^T DS$

Матриця S - верхня трикутна матриця

Матриця D - діагональна матриця для збереження знаку ведучих елементів

Формули заповнення матриці S:

$$s_{ii} = \sqrt{\left| a_{ii} - \sum_{p=1}^{i-1} s_{pi}^{\ 2} d_{pp} \right|},\ i = \overline{1, n}$$

$$s_{ij} = \frac{a_{ij} - \sum_{p=1}^{i-1} (s_{pi} d_{pp} s_{pj})}{d_{ii} s_{ii}},\ i = \overline{2, n - 1},\ j = \overline{i + 1, n}$$

Формули заповнення матриці D:

$$d_{ii} = sgn\left| a_{ii} - \sum_{p=1}^{i-1} s_{pi}^{\ 2} d_{pp} \right|$$

Подалше рівняння зводиться до розв’язку двох СЛАР з трикутними матрицями. З першої системи знаходять y:

$$S^T D y = b$$

A з другої - x:

$$Sx = y$$

Пошук визначника:

$$\det A = \prod_{k=1}^{n} d_{kk} s_{kk}^{\ 2}$$

<u>Метод Зейделя</u>

Ітераційни метод для розв’язання СЛАР типу Ax=b. Розв’язок знаходимо із заданою точністю **ε**. Початкове наближення обираємо довільним чином. Ітераційний процес має вигляд:

$$x_i^{k+1} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{k+1} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{k} + \frac{b_i}{a_{ii}}$$

Умова зупинки:

$$\left| x^{k+1} - x^{k} \right| \le \varepsilon$$

Достатні умови збіжності:

- Якщо $\forall i$: $i = \overline{1, n}$ виконується нерівність

$$\left|a_{ii}\right| \geq \sum_{j=1, j\neq i}^{n} \left|a_{ij}\right|$$

то ітераційний процес методу Зейделя збігається з лінійною швидкістю.

- Якщо $A = A^T > 0$, то ітераційни процес методу Зейделя збігається з лівнійною швидкістю.

Необхідні і досстатні умови збіжності:

- Для $\forall x^0$ ітераційний процес методу Зейделя збігається тоді і тільки тоді, коли $|\lambda| < 1$, де λ - корені нелійнійного рівняння

Хід роботи

Мова реалізації: Python

Метод Гаусса

7	2	3	0		X1		20
0	3	2	6	*	X2	=	36
2	5	1	0		X3		15
0	1	4	2		X4		22

Ініціалізуємо матрицю та вектор:

```
--------------------------------------------------
Initial matrix A and vector b
   7.000    2.000    3.000    0.000 |   20.000
   0.000    3.000    2.000    6.000 |   36.000
   2.000    5.000    1.000    0.000 |   15.000
   0.000    1.000    4.000    2.000 |   22.000
--------------------------------------------------
```

Прямим ходом за допомогою матриць P_i (перестановки) і M_i (елементарні перетворення) зводимо A до трикутної форми, одночасно перетворюючи одиничну матрицю для знаходження оберненої.

```
Working on column 0

Matrix M1:
   0.143    0.000    0.000    0.000
   0.000    1.000    0.000    0.000
  -0.286    0.000    1.000    0.000
   0.000    0.000    0.000    1.000

Matrix after elimination step:

------------------------------
M1P1A = M1P1b
------------------------------
   1.000    0.286    0.429    0.000 |    2.857
   0.000    3.000    2.000    6.000 |   36.000
   0.000    4.429    0.143    0.000 |    9.286
   0.000    1.000    4.000    2.000 |   22.000
------------------------------
```

```
Working on column 1
   1.000    0.286    0.429    0.000 |    2.857
   0.000    3.000    2.000    6.000 |   36.000
   0.000    4.429    0.143    0.000 |    9.286
   0.000    1.000    4.000    2.000 |   22.000

----------------------------
P2M1P1A = P2M1P1b
----------------------------

Matrix P2
   1.000    0.000    0.000    0.000
   0.000    0.000    1.000    0.000
   0.000    1.000    0.000    0.000
   0.000    0.000    0.000    1.000

   1.000    0.286    0.429    0.000 |    2.857
   0.000    4.429    0.143    0.000 |    9.286
   0.000    3.000    2.000    6.000 |   36.000
   0.000    1.000    4.000    2.000 |   22.000
```

```
Matrix M2:
   1.000    0.000    0.000    0.000
   0.000    0.226    0.000    0.000
   0.000   -0.677    1.000    0.000
   0.000   -0.226    0.000    1.000

Matrix after elimination step:

------------------------------
M2P2M1P1A = M2P2M1P1b
------------------------------
   1.000    0.286    0.429    0.000 |    2.857
   0.000    1.000    0.032    0.000 |    2.097
   0.000    0.000    1.903    6.000 |   29.710
   0.000    0.000    3.968    2.000 |   19.903
------------------------------
```

....

```
Working on column 3

Matrix M4:
   1.000    0.000    0.000    0.000
   0.000    1.000    0.000    0.000
   0.000    0.000    1.000    0.000
   0.000    0.000    0.000    0.198

Matrix after elimination step:

------------------------------
M4P4M3P3M2P2M1P1A = M4P4M3P3M2P2M1P1b
------------------------------
   1.000    0.286    0.429    0.000 |    2.857
   0.000    1.000    0.032    0.000 |    2.097
   0.000    0.000    1.000    0.504 |    5.016
   0.000    0.000    0.000    1.000 |    4.000
------------------------------
```

Після отриманого результату, починаємо зворотній хід для коренів СЛАР:

```
------------------------------
Back substitution: x[3] = 4.000 - 0.000 = 4.000
Back substitution: x[2] = 5.016 - 2.016 = 3.000
Back substitution: x[1] = 2.097 - 0.097 = 2.000
Back substitution: x[0] = 2.857 - 1.857 = 1.000
-----------------------------
Solution vector x:
   1.000    2.000    3.000    4.000
-----------------------------
```

Знаходження детермінаннту матриці:

```
---------------------------
Calculating det A:
det A: (-1)^2 * 7.0 * 4.428571428571429 * 3.967741935483871 * 5.040650406504065
Det A:620.0
Calculating det A with NumPy: 620.0
```

Перевіримо мануально коректність знайдених коренів:

7 * 1	+2*2	+3*3	+0		20
0	+3*2	+2*3	+6*4	=	36
2*1	+5*2	+1*3	+0		15
0	+1*2	+4*3	+2*4		22

Пошук оберненої матриці:

```
Inverted A:

   0.161    0.042   -0.065   -0.126
  -0.065    0.003    0.226   -0.010
   0.000   -0.100    0.000    0.300
   0.032    0.198   -0.113   -0.095

A*A^-1=E. The inveted A matrix is correct
```

У результаті отримали вектор, заданий за умовою - корені знайдені правильно.

Метод квадратного кореня

$$\begin{matrix} 1 & 2 & 0 \\ 2 & 2 & 3 \\ 0 & 3 & 2 \end{matrix} \times \begin{matrix} X1 \\ X2 \\ X3 \end{matrix} = \begin{matrix} 5 \\ 15 \\ 12 \end{matrix}$$

Ініціалізуємо дані:

```
==================================================
SQUARE ROOT METHOD
==================================================

Initial matrix A:
   1.000    2.000    0.000
   2.000    2.000    3.000
   0.000    3.000    2.000

Vector b:
   5.000   15.000   12.000
```

Перевіряємо достатню умову: $A = A^T$

```
#Check sufficient condition
# A=A^T
if(np.all(A!=np.transpose(A))):
    print("Sufficient condition is not satisfied!")
    return
print("\nThe sufficient condition is satisified: A=A^T \n")
```

```
The sufficient condition is satisified: A=A^T
```

Обраховуємо матриці S та D:

```
Matrix S:
   1.000     2.000     0.000
   0.000     1.414    -2.121
   0.000     0.000     2.550

Vector D:
   1.000    -1.000     1.000
```

Обраховуємо дві СЛАР:

- $S^T D y = b$

```
# Forward substitution: S^T*D*y = b
y = np.zeros(n)
for i in range(n):
    sum_sdy = 0
    for j in range(i):
        sum_sdy += S[j, i] * D[j] * y[j]
    y[i] = (b[i] - sum_sdy) / (S[i, i] * D[i])
```

- $Sx = y$

```
# Backward substitution: S*x = y
x = np.zeros(n)
for i in range(n-1, -1, -1):
    sum_sx = 0
    for j in range(i+1, n):
        sum_sx += S[i, j] * x[j]
    x[i] = (y[i] - sum_sx) / S[i, i]
```

Результат:

```
Solution vector x:
   1.000     2.000     3.000
```

Перевіримо коректність знайдених коренів:

1*1	+2*2	+0		5
2*1	+2*2	+3*3	=	15
0	+3*2	+2*3		12

Результати відповідають вектору b, отже корені вірні.

Знайдемо визначник:

$$\det A = \prod_{k=1}^{n} d_{kk} s_{kk}^{\ \ 2}$$

```
Calculating det A:
det A: 1.000*1.000^2 * -1.000*1.414^2 * 1.000*2.550^2

Det A:-13.0
Calculating det A with NumPy: -13.0
```

Знайдемо число обумовленості:

Число обумовленості cond(A) матриці характеризує чутливість розв’язку системи Ax=b до похибок у даних A та b.

Обчислюємо за формулою:

$$cond(A) = ||A|| * \left|\left|A^{-1}\right|\right|$$

Обираємо inf-норму - найбільша сума по рядках матриці

$$||A||_{inf} = \max_{i} \sum_{j} \left|a_{ij}\right|$$

```
norm_A = np.linalg.norm(A, ord=np.inf)
norm_A_inv = np.linalg.norm(np.linalg.inv(A), ord=np.inf)
cond_A = norm_A * norm_A_inv
print(f"Condition number: {cond_A:.3f}")
return x,detA
```

```
Condition number: 8.077
```

Метод Зейделя

4	0	1	0		X1		12
0	3	0	2	x	X2	=	19
1	0	5	1		X3		27
0	2	1	4		X4		30

Ініціалізуємо вхідні дані:

```
=====================================================
SEIDELS METHOD
=====================================================

Initial matrix A:
   4.000    0.000    1.000    0.000
   0.000    3.000    0.000    2.000
   1.000    0.000    5.000    1.000
   0.000    2.000    1.000    4.000

Vector b:
  12.000   19.000   27.000   30.000
```

Перевіримо достатні умови для збіжності даного методу:

- Якщо $\forall i\colon\ i = \overline{1, n}$ виконується нерівність

$$\left|a_{ii}\right| \geq \sum_{j=1, j\neq i}^{n} \left|a_{ij}\right|$$

то ітераційний процес методу Зейделя збігається з лінійною швидкістю.

- Якщо $A = A^T > 0$, то ітераційни процес методу Зейделя збігається з лівнійною швидкістю.

```
----------------------------------------
Check the first condition:
Row 0: 4.0>=1.0
Row 1: 3.0>=2.0
Row 2: 5.0>=2.0
Row 3: 4.0>=3.0
First condition is satisfied!
Second condition is satisfied!
----------------------------------------
```

Достатні умови виконуються отже починаємо ітераційний процес:

```
Iterations:
Iter 1: [3.      6.33333 4.8     3.13333]
Iter 2: [1.8     4.24444 4.41333 4.27444]
Iter 3: [1.89667 3.4837  4.16578 4.7167 ]
Iter 4: [1.95856 3.18886 4.06495 4.88933]
Iter 5: [1.98376 3.07378 4.02538 4.95676]
Iter 6: [1.99365 3.02882 4.00992 4.98311]
Iter 7: [1.99752 3.01126 4.00387 4.9934 ]
Iter 8: [1.99903 3.0044  4.00151 4.99742]
Iter 9: [1.99962 3.00172 4.00059 4.99899]
Iter 10: [1.99985 3.00067 4.00023 4.99961]
Iter 11: [1.99994 3.00026 4.00009 4.99985]
Iter 12: [1.99998 3.0001  4.00004 4.99994]
```

Задана точність $\varepsilon = 1e - 4$

Результуючий вектор:

```
Final solution vector x:
   2.000    3.000    4.000    5.000
Number of iterations: 13
```

Перевіримо коректність знеайдених коренів:

4*2	+0	+1*3	+0		12
0	+3*3	+0	+2*5	=	19
1*2	+0	+5*4	+1*5		27
0	+2*3	+1*4	+4*5		30

Знайдені корені задовільняють систему, отже алгоритм виконується вірно.

[Github code]