## ЛЕКЦІЯ 3: РОЗВ'ЯЗУВАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРИЧНИХ РІВНЯНЬ.

*Деякі поняття матричної алгебри. Представлення лінійної системи в матричній формі. Розв'язання систем лінійних рівнянь в матричній формі. Метод Крамера. Метод Гауса. Метод квадратних коренів. Матричний метод. Ітеративні методи. Практичні засоби розв'язування систем лінійних алгебричних рівнянь в пакетах матричної математики Scilab, MatLab.*

Способи розв'язування систем лінійних рівнянь в основному поділяють на дві групи: прямі методи та ітераційні методи. *Прямі (точні) методи* – це скінчені алгоритми для обчислення коренів системи, наприклад, методи Крамера, Гауса, головних елементів, квадратних коренів та ін. Вони забезпечують точний розв'язок в межах обчислювальних можливостей самих комп'ютерів. Ефективність прямих методів визначається за наперед відомою скінченною кількістю арифметичних операцій, необхідних для розв'язування системи. *Ітераційні методи* дозволяють одержувати корені системи із заданою точністю шляхом довільної кількості повторюваних збіжних процесів, наприклад методи простої ітерації, Зейделя, релаксації та ін.

### Представлення лінійної системи в матричній формі.

Нехай система $n$ лінійних алгебричних рівнянь з $n$ невідомими має вигляд

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = f_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = f_2 \\ \qquad\qquad \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = f_n \end{cases}. \qquad (1)$$

Якщо позначити через $A$ матрицю з коефіцієнтів системи (1)

$$A = \begin{Vmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{Vmatrix},$$

через $\vec{f}$ – вектор-стовпець $n$ вільних членів, а через $\vec{x}$ – вектор-стовпець $n$ невідомих (шуканий вектор):

$$\vec{f} = \begin{Vmatrix} f_1 \\ f_2 \\ \dots \\ f_n \end{Vmatrix}, \qquad \vec{x} = \begin{Vmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{Vmatrix},$$

то система лінійних алгебричних рівнянь (1) може бути записана у компактному вигляді матричного рівняння

$$A\vec{x} = \vec{f}\,. \qquad (2)$$

Слід також зазначити, що СЛАР (1), (2) має єдиний нетривіальний розв'язок лише за умови, що визначник $\triangle$ матриці $A$

$$\triangle = \det \mathbf{A} = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} \neq 0, \qquad (3)$$

**Теоретичні засади методів розв'язання систем лінійних рівнянь в матричній формі.**

***1. Метод оберненої матриці***

За умови (3) розв'язок СЛАР (2) має вигляд

$$\vec{x} = A^{-1}\vec{f}\,, \qquad (4)$$

де $A^{-1}$ – обернена до $A$ матриця, а вектор-стовпець (4) перетворює систему (2) у тотожність. Компоненти $x_k$ цього вектора називають коренями СЛАР. Така обернена матриця та корені можуть бути отримані різними шляхами. Один з найвідоміших точних методів це метод Крамера.

***2. Метод Крамера***

За умови (3) корені $x_k$ системи (2) можуть бути обчислені за *формулами Крамера*

$$x_k = \frac{\triangle_k}{\triangle} \quad (k = 1,2,...,n), \qquad (5)$$

де $\triangle_k$ – визначники матриць $A_k$ $(k = 1,2,...,n)$, які утворюються з матриці $A$ шляхом заміни у ній стовпчиків $\begin{Vmatrix} a_{1k} \\ a_{2k} \\ ... \\ a_{nk} \end{Vmatrix}$ вектором-стовпчиком $\vec{f}$.

Формули Крамера мають істотне теоретичне значення, але через великий обсяг обчислювальної роботи мало ефективні на практиці при чисельному розв'язуванні СЛАР. За цими формулами треба обчислювати значення $(n+1)$ визначників порядку *n,* для чого виконати значну кількість арифметичних операцій, що вимагає контролю над зростаючою кількістю похибок обчислення.

### *3. Метод Гауса*

Найпростішим методом розв'язування систем лінійних алгебраїчних рівнянь є *метод послідовного виключення змінних,* або *метод Гауса.* В основі цього методу є еквівалентні перетворення СЛАР, які полягають у тотожніх перетвореннях її рівнянь та їх перестановках. Є кілька модифікацій цього методу. Розглянемо *схему єдиного ділення,* за якою СЛАР розв'язують в два етапи. На першому етапі вихідну систему рівнянь зводять до рівносильної їй системи верхньої трикутної форми. Цей процес перетворення називають *прямим ходом.* На другому етапі, який називають *зворотним ходом,* із отриманої системи верхньої трикутної форми знаходять безпосередній розв'язок СЛАР.

Для одержання розв’язку системи (1) методом Ґауса перше її рівняння подається у вигляді

$$x_1 + c_{12}x_2 + c_{13}x_3 + ... + c_{1n}x_n = y_1, \qquad (6)$$

де $c_{ij} = \frac{a_{ij}}{a_{11}}$ $(j = \overline{2,n})$, $y_1 = \frac{f_1}{a_{11}}$, $a_{11} \neq 0$.

Якщо перемножити тепер по черзі рівняння (6) на коефіцієнти $a_{i1}$ біля $x_1$ у решті рівнянь

$$a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + \ldots + a_{in}x_n = f_i \quad (i = \overline{2,n}), \qquad (7)$$

а після цього відняти одержане рівняння від відповідного $i$ -го рівняння системи (1), то система набуде вигляду

$$\begin{cases} x_1 + c_{12}x_2 + c_{13}x_3 + \ldots + c_{1n}x_n = y_1, \\ \quad b_{22}x_2 + b_{23}x_3 + \ldots + b_{2n}x_n = g_2, \\ \quad \ldots\ldots\ldots\ldots\ldots\ldots, \\ \quad b_{n2}x_2 + b_{n3}x_3 + \ldots + b_{nn}x_n = g_n, \end{cases}$$

де $b_{ij} = a_{ij} - c_{ij}a_{i1}$, $g_i = f_i - y_1a_{i1}$ $(i, j = \overline{2,n})$.

Продовжуючи аналогічним чином процес виключення невідомих, можна отримати систему рівнянь

$$\begin{cases} x_1 + c_{12}x_2 + c_{13}x_3 + \ldots + c_{1n}x_n = y_1, \\ \qquad x_2 + c_{23}x_3 + \ldots + c_{2n}x_n = y_2, \\ \qquad\qquad x_3 + \ldots + c_{3n}x_n = y_3, \\ \qquad\qquad \ldots\ldots\ldots, \\ \qquad\qquad x_{n-1} + \ldots + c_{n-1\,n}x_n = y_{n-1}, \\ \qquad\qquad\qquad x_n = y_n, \end{cases} \qquad (8)$$

яка буде еквівалентною заданій системі (1).

Такий процес послідовного виключення невідомих, в результаті якого утворюється система рівнянь з трикутною матрицею

$$\mathbf{C} = \begin{Vmatrix} 1 & c_{12} & c_{13} & \ldots & c_{1n-1} & c_{1n} \\ 0 & 1 & c_{23} & \ldots & c_{2n-1} & c_{2n} \\ 0 & 0 & 1 & \ldots & c_{3n-1} & c_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 1 & c_{n-1n} \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{Vmatrix}$$

називається *прямим ходом методу Ґауса*.

Якщо користуватися системою (8) у зворотному порядку, починаючи з її останнього рівняння, то можна визначити по черзі всі невідомі системи $x_n, x_{n-1}, \ldots, x_1$.

Процес знаходження невідомих за допомогою системи (8) називається *зворотнім ходом методу Ґауса*. Формули зворотного ходу можна записати у загальному вигляді так:

$$x_i = \begin{cases} y_i, \ (i = n), \\ y_i - \sum_{j=i+1}^{n} c_{ij} x_j, \ i = (n-1, n-2, \ldots, 1). \end{cases}$$

Очевидно, що використовувати розглянутий варіант методу Ґауса можна лише у випадку, коли коефіцієнти $a_{11}, a_{22}, \ldots, a_{n-1n-1}$ біля $x_1, x_2, \ldots, x_{n-1}$, на які виконується операція ділення, відмінні від нуля.

Загалом коефіцієнт $a_{ij} \sim a_{kk} \ (i = j = k)$, за допомогою якого здійснюють визначення змінної на $k$-му кроці, називається провідним чи головним елементом на цьому кроці виключення невідомих. Якщо провідний елемент виявиться досить близьким до нуля, то розв'язок може значно відрізнятися від точного розв'язку, тобто у цьому випадку відбувається збільшення похибок обчислень. Така ситуація істотно обмежує область використання цього методу.

У таких випадках варто застосовувати метод Ґауса з вибором провідного елемента. Його основна ідея полягає в тому, що на кожному $k$-му кроці у рядку з номером $k$ і нижче може виключатися не невідома $x_k$, а та з невідомих, біля якої коефіцієнт $a_{kj}$ є найбільшим за модулем.

Найбільший за модулем коефіцієнт біля невідомої, яка виключається на даному кроці, називається *провідним (головним) елементом*.

Метод Ґауса з вибором головного елемента відрізняється від звичайного методу Ґауса лише тим, що на кожному кроці виключення невідомої відбувається додаткова перенумерація (фактично переставляння відповідних членів рівняння) відповідних змінних.

Наприклад, якщо в системі рівнянь

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = f_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = f_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = f_3, \end{cases}$$

коефіцієнт $a_{13}$ має максимальне за модулем значення серед коефіцієнтів першого рядка, то на першому кроці виключається невідома $x_3$, а не $x_1$, як би це було у звичайній схемі. Для цього дана система повинна бути записаною у вигляді

$$\begin{cases} a_{13}x_3 + a_{11}x_1 + a_{12}x_2 = f_1, \\ a_{23}x_3 + a_{21}x_1 + a_{22}x_2 = f_2, \\ a_{33}x_3 + a_{31}x_1 + a_{32}x_2 = f_2, \end{cases}$$

що дає можливість використовувати для неї звичайний метод Ґауса. На наступному кроці знову відбувається вибір головного елемента, перенумерація (переставляння) змінних і виключення нової невідомої.

***4. Метод квадратних коренів (точний)***

Цей метод використовують для знаходження розв'язку лінійної системи (2), в якій матриця $A$ є симетричною, тобто елементи симетричні відносно головної діагоналі: $a_{ij} = a_{ji}$. Відомо, що симетричну матрицю завжди можна подати у вигляді добутку двох взаємно транспонованих трикутних матриць

$$A = T'T. \qquad (10)$$

де

$$T = \begin{Vmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ 0 & t_{22} & \dots & t_{2n} \\ 0 & 0 & \dots & \dots \\ 0 & 0 & \dots & t_{nn} \end{Vmatrix}, \qquad T' = \begin{Vmatrix} t_{11} & 0 & \dots & 0 \\ t_{12} & t_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ t_{1n} & t_{2n} & \dots & t_{nn} \end{Vmatrix}. \qquad (11)$$

Якщо тепер перемножити матриці $T'$ та $T$, а потім прирівняти відповідні елементи матриць у рівності (10), то для знаходження $n(n+1)/2$

елементів $t_{ij}$ $\left(i=1,2,...,n;\ j=i,i+1,...,n\right)$ матриці $T$ отримаємо систему $n(n+1)/2$ рівнянь

$$\begin{cases} t_{1i}^2+t_{1i}^2+...+t_{ii}^2=a_{ii}, \\ t_{1i}t_{1j}+t_{2i}t_{2j}+...+t_{ii}t_{ij}=a_{ij} \end{cases} \qquad (i=1,2,...,n;\quad j-i+1,\ i+2,...,n;\ i<j).$$

З цієї системи знаходимо послідовно елементи матриці $T$ (і $T'$)

$$\begin{cases} t_{11}=\sqrt{a_{11}},\ t_{1j}=\dfrac{a_{1j}}{t_{1j}} \quad (j=2,3,...,n), \\ t_{ii}=\sqrt{a_{ii}-\sum\limits_{k=1}^{i-1}t_{ki}^2} \qquad (1<i\le n), \\ t_{ij}=a_{ij}-\sum\limits_{k=1}^{i-1}\dfrac{t_{ki}t_{kj}}{t_{ii}} \qquad (i<j), \\ t_{ij}=0 \qquad\qquad\qquad (i>j). \end{cases}$$

З рівності (10) випливає, що система (2) у випадку симетричності матриці рівносильна двом системам рівнянь з трикутними матрицями $T'y=f$ і $Tx=y$. Розв'язавши систему $T'y=f$ з нижньою трикутною матрицею $T'$ знайдемо

$$y_1=\frac{f_1}{t_{11}},\quad y_i=\frac{f_i-\sum\limits_{k=1}^{i-1}t_{ki}y_k}{t_{ii}} \quad (1<i\le n).$$

Після цього розв'язавши систему $Tx=y$ з верхньою матрицею $T$ знайдемо шуканий розв'язок системи (2)

$$x_n=\frac{y_n}{t_{nn}},\ x_i=\frac{y_i-\sum\limits_{k=i+1}^{n}t_{ik}x_k}{t_{ii}} \quad (1\le i<n).$$

### *5. Метод простої ітерації*

Порівняно з методом Ґауса ітераційні методи мають певні переваги лише для систем високого і дуже високого порядку $\left(n>10^3\right)$, особливо, з розрідженими матрицями, тобто такими, що мають багато нульових елементів. Крім того, за надто великої кількості невідомих під час

процедури розв’язування методом Гауса може виникнути критична маса похибок обчислення, що викличе сумнів у точності отриманого розв’язку (протиріччя: метод точний, але це теоретично, бо на практиці не завжди може бути реалізований). Ефективне застосування ітераційних методів вимагає певного досвіду. З цього погляду методи Ґауса є надійнішими, хоча не завжди дають можливість раціонально використовувати пам’ять комп’ютера і час, який затрачається на одержання розв’язку.

Розглянемо як приклад *метод простої ітерації.*

Нехай задано систему лінійних рівнянь (1), або в матричному вигляді (2). Нехай діагональні елементи $a_{ii}$ $(i=1,2,...,n)$ матриці $A$ відмінні від нуля. Тоді, розв'язавши перше рівняння системи (1) відносно $x_1$, друге - відносно $x_2$ і т.д., дістанемо систему

$$\begin{cases} x_1 = \alpha_{12}x_2 + \alpha_{13}x_3 + ... + \alpha_{1n}x_n + \beta_1 \\ x_2 = \alpha_{21}x_1 + \alpha_{23}x_3 + ... + \alpha_{2n}x_n + \beta_2 \\ x_3 = \alpha_{31}x_1 + \alpha_{32}x_2 + ... + \alpha_{3n}x_n + \beta_3 \\ ..... \\ x_n = \alpha_{n1}x_1 + \alpha_{n2}x_2 + ... + \alpha_{nn-1}x_{n-1} + \beta_n \end{cases} \qquad (11)$$

де

$$\alpha_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}}, & i \neq j \\ 0, & i = j \end{cases} \qquad \beta_i = \frac{f_i}{a_{ii}},$$

Якщо ввести до розгляду матриці

$$\alpha = \begin{Vmatrix} \alpha_{11} & \alpha_{12} & ... & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & ... & \alpha_{2n} \\ ... & ... & ... & ... \\ \alpha_{n1} & \alpha_{n2} & ... & \alpha_{nn} \end{Vmatrix}, \qquad \beta = \begin{Vmatrix} \beta_1 \\ \beta_2 \\ ... \\ \beta_n \end{Vmatrix}$$

то систему (11) можна записати у вигляді

$$x = \alpha x + \beta, \qquad (12)$$

який ще називають *нормальним.*

Далі процедура розв'язування має вигляд послідовних наближень. За початкове наближення візьмемо, наприклад, стовпець вільних членів, тобто

$x^{(0)} = \beta$. Тоді кожне наступне $k$-те наближення розв'язку СЛАР отримується виразом

$$x^{(k+1)} = \alpha x^{(k)} + \beta \quad (k = 0,1,2,....), \tag{13}$$

або в розгорнутому вигляді

$$x_i^{(0)} = \beta_i,$$

$$x_i^{(k+1)} = \sum_{j=1}^{n} a_{ij} x_i^{(k)} + \beta_i, \quad (i = 1,2,...,n;\ k = 0,1,2,....). \tag{14}$$

Якщо послідовність наближень $x^{(0)}, x^{(1)},..., x^{(k)},...$ є збіжною, тобто має границю $x^* = \lim\limits_{k\to\infty} x^{(k)}$, то ця границя і буде розв'язком системи (12). Справді, перейшовши до границі, коли $k \to \infty$, у рівності (13), маємо

$$\lim_{k\to\infty} x^{(k+1)} = \lim_{k\to\infty} \left(\alpha x^{(k)} + \beta\right) = \alpha \lim_{k\to\infty} x^{(k)} + \beta,$$

або $x^* = \alpha x^* + \beta$ – тотожність.

Таким чином, вектор $\vec{x}^* = \begin{Vmatrix} x_1^* \\ x_2^* \\ ... \\ x_n^* \end{Vmatrix}$,

є розв'язком системи (12), а отже, і системи (1). Зрозуміло, що на практиці виконувати безмежну кількість наближень неможливо навіть за наявності великих обчислювальних можливостей, бо починають з'являтися і накопичуватися похибки обчислень. Тому проводиться аналіз точності обчислень після кожного наступного $k$-того наближення шляхом порівняння відхилень отриманих значень розв'язку від значень, отриманих на попередньому $(k-1)$-ому кроці. Якщо ці відхилення з кожним наступним кроком зменшуються, то говорять, що процес збіжний і продовжують його до отримання результатів із потрібною кількістю вірних значущих цифр.

Метод *простої ітерації* є найпростішим з цілої низки методів послідовних наближень (методи Зейделя, релаксації, Халецького, …), кожен з яких має певні удосконалення, що переважно стосуються пришвидшення

збіжності, тобто зменшення кількості ітерацій для досягнення заданої точності.

## Практичні засади методів розв'язання систем лінійних рівнянь в пакеті матричних обчислень Scilab (Mathlab). Масиви і матриці

Для роботи з множиною даних зручно використовувати *масиви*. Наприклад, можна створити масив для зберігання числових або символьних даних. В цьому випадку замість створення змінної для зберігання кожного значення досить створити один масив, де кожному елементу буде присвоєний порядковий номер. Таким чином, масив - множинний тип даних, що складається з фіксованого числа елементів. Як і будь-якій іншій змінній, масиву має бути присвоєне ім'я.

Змінну, таку, що є просто списком даних, називають *одновимірним масивом,* або *вектором*. Для доступу до даних, що зберігаються в певному елементі масиву, необхідно вказати ім'я масиву і порядковий номер цього елементу, званий індексом.

Якщо виникає необхідність зберігання даних у вигляді таблиць, у форматі рядків і стовпців, то необхідно використовувати *двовимірні масиви (матриці).*

Для доступу до даних, що зберігаються в такому масиві, необхідно вказати ім'я масиву і два індекси: перший повинен відповідати номеру рядка, а другий - номеру стовпця, в яких зберігається необхідний елемент.

Значення нижньої межі індексації в Scilab дорівнює одиниці. Індекси можуть бути тільки натуральними числами.

### *1. Ввід і формування масивів*

Задавати одновимірний масив (вектор-рядок, вектор-стовпчик) в Scilab можна по-різному. Наприклад, якщо потрібно задати розбиття деякого проміжку з відомим кроком у вигляді вектора-рядка, то зручно скористатися засобом:

**name=Xn: dX:Xk**

де **name** - ім'я змінної, в яку буде записаний сформований масив, **Xn** - значення першого елементу масиву, **Xk** - значення останнього елементу масиву, **dX** - крок, за допомогою якого формується кожен наступний елемент масиву, тобто значення другого елементу складе **Xn+dX**, третього **Xn+ dX+dX** і так далі до **Xk**. Якщо параметр **dX** в конструкції відсутній, це означає, що за умовчанням він набуває значення, рівного одиниці, тобто кожен наступний елемент масиву дорівнює значенню попереднього плюс одиниця. Інший спосіб задати розбиття деякого проміжку, якщо відома кількість фрагментів розбиття – це використання функції linspace:

**name=linspace(Xn, Xk, N )**

де **N –** задана кількість частин, на які буде поділено проміжок **Xn, Xk.** Порівнюючи цей засіб з попереднім можна сказати, що крок розбиття **dX** дорівнює **(Xk-Xn)/N.**

Ще один спосіб завдання векторів і матриць в Scilab - це їх поелементне введення. Так, для визначення вектора-рядка слід ввести ім'я масиву, а потім, після знаку привласнення, в квадратних дужках через пробіл або кому перерахувати елементи масиву

**name=[x1 x2 ... xn]** чи **name=[x1, x2, ..., xn]**

Якщо потрібно сформувати вектор-стовпець, то як розділовий знак слід вжити крапку з комою

**name=[x1; x2; ... xn].**

Звернутися до елементу вектора можна, вказавши ім'я масиву і порядковий номер елементу в круглих дужках:

**name(індекс)**

Змінну, задану як масив, можна використовувати в арифметичних виразах і як аргумент математичних функцій.

Роглянемо декілька типових прикладів:

1) Задання одновимірного масиву (вектора-рядка)

```
-->Xn=-3.5; dX=1.5; Xk=4.5;
```

```
-->X=Xn:dX:Xk
-->X =
-3.5000 -2.0000 -0.5000 1.0000 2.5000 4.0000

-->A=0:5
-->A = 0 1 2 3 4 5
```

2) Поелементне задання одновимірного масиву (вектора-рядка, вектора-стовпця)

```
-->W=[1.1,2.3,-0.1,5.88]
-->W =
1.1000 2.3000 -0.1000 5.8800

-->X=[1;2;3]
-->X =
1
2
3
```

3) Обчислюється масив Y

```
-->Y=sin(X/2)
-->Y =
 -0.9840 -0.8415 -0.2474 0.4794 0.9490 0.9093
```

4) Звернення до елементу вектора

```
-->W=[1.1,2.3,-0.1,5.88];
-->W(1)+2*W(3)
ans =
 0.9000
```

Введення елементів матриці також здійснюється в квадратних дужках, при цьому елементи рядка відділяються один від одного пропуском або комою, а рядки розділяються між собою крапкою з комою:

**name=[x11, x12, .., x1n; x21, x22, .., x2n; ..; xm1, xm2, .., xmn;]**

Звернутися до елементу матриці можна, вказавши після імені матриці, в круглих дужках через кому, номер рядка і номер стовпця на перетині яких елемент розташований:

**name(індекс1, індекс2)**

Важливу роль при роботі з матрицями грає знак двокрапки. Вказуючи його замість індексу при зверненні до масиву, можна діставати доступ до груп його елементів.

Роглянемо декілька типових прикладів:

1) Задання матриці 3x3

```
-->A=[1 2 3;4 5 6;7 8 9]
-->A =
1 2 3
4 5 6
7 8 9
```

2) Звернення до елементів матриці A і дії над ними

```
-->A(1,2)^A(2,2)/A(3,3)
-->ans =
 3.5556
```

3) Виділити з матриці A другий стовпець

```
-->A(:,2)
-->ans =
2
5
8
```

4) Виділити з матриці A третій рядок

```
-->A(3,:)
-->ans =
7 8 9
```

5) Видалити з матриці A другий стовпець

```
-->A(:,2)=[]
```

```
-->A=
 1 3
 4 6
 7 9
```

6) Великі матриці можна формувати з кількох менших, розмірність яких узгоджена. Наприклад

```
-->A=[1, 2, 3, 4];
-->B=[0, 5, -1, 4];
-->C=[6, 2, -10, 5];
-->D=[A; B; C]
-->D=
  1 2  3   4
  0 5 -1   4
  6 2 -10 5
```

*Зауваження: Зверніть увагу, що наявність крапки з комою після введення команди Scilab відміняє вивід результату її виконання на дисплей.*

7) За потреби згенерувати специфічні матриці, як от одинична, діагональна, нульова в Scilab можна використати відповідні функції:

**eye(n, m)** – генерує одиничну діагональну матрицю відповідної розмірності:

```
-->eye(3,3)
 ans =
 1. 0. 0.
 0. 1. 0.
 0. 0. 1.
-->eye(5,1)
 ans =
 1.
 0.
 0.
```

0.

0.

**ones(n, m)** – формує повну одиничну матрицю:

-->m=3; n=2;

-->X=ones(n, m)

X =

1. 1. 1.

1. 1. 1.

**zeros(n, m)** – генерує нульову матрицю відповідної розмерності:

-->zeros(3,2)

ans =

0. 0.

0. 0.

0. 0.

**rand([n, m, p, …])** – генерує матрицю випадкових чисел, rand без аргументів генерує одне випадкове число

--> rand(3,3) //Квадратна матриця випадкових чисел

ans =

0.9501 0.4860 0.4565

0.2311 0.8913 0.0185

0.6068 0.7621 0.8214

--> rand  //Одне випадкове число

ans =

0.405

### 2.Дії над матрицями

Для роботи з матрицями і векторами в Scilab передбачені наступні операції:

| *Операція* | *Коментар* |
|---|---|
| + - додавання | Операції додавання і віднімання визначені згідно правил матричної алгебри, тобто для |

| | |
|---|---|
| - - віднімання | матриць однієї розмірності або векторів одного типу. |
| ' - транспонування | Якщо в деякій матриці замінити рядки відповідними стовпцями, то вийде транспонована матриця. |
| * - матричне множення<br>*. - множення на число або поелементне множення | Операція множення вектора на вектор визначена тільки для векторів однакового розміру, причому один з них має бути вектором-стовпцем, а другим вектором-рядком. Матричне множення виконується за правилом рядок на стовпець і допустиме, якщо кількість рядків в другій матриці співпадає з кількістю стовпців в першій. |
| ^ - піднесення до ступеня | Звести матрицю в n -у міру означає помножити її саму на себе n разів. При цьому цілочисельний показник міри може бути як додатнім, так і від'ємним. У першому випадку виконується алгоритм множення матриці на себе вказане число разів, в другому множиться на себе матриця, обернена до даної. |
| \ - ліве ділення | Операція лівого ділення може бути застосовна для розв'язування матричного рівняння виду A X = B, де X - невідомий вектор. |
| / - праве ділення | Операцію правого ділення використовують для розв'язування матричних рівнянь виду X A = B. |

Приклади матричних операцій :

```
-->x=[0.1 -2.2 %pi 0 -1];
 -->sin(x)
 ans =
    0.0998334  - 0.8084964    1.225D-16    0.  - 0.8414710
-->A=[1 2 0;-1 3 1;4 -2 5];
-->B=[-1 0 1;2 1 1;3 -1 -1];
-->//Обчислити (A'+B)^2-2A(0.5B'-A)
-->(A'+B)^2-2*A*(0.5*B'-A)
 ans =
   10.    8.    24.
   11.    20.   35.
```

```
   63.  - 30.    68.
-->//
-->//Розв’язати матричне рівняння  AX=B
-->X=A\B
X  =
 - 0.8857143  - 0.3428571    0.1428571
 - 0.0571429    0.1714286    0.4285714
   1.2857143    0.1428571  - 0.1428571
-->//Розвязати матричне рівняння  XA=B
-->X=B/A
 X  =
 - 0.7714286    0.5714286    0.0857143
   0.9428571  - 0.1428571    0.2285714
   1.4857143  - 1.2857143    0.0571429
-->//Перевірка
-->X*A-B
 ans  =
 10^(-15) *
   0.1110223         0.0555112         0.
   0.                0.                0.2220446
   0.4440892         0.                0.
```

Зверніть увагу на останню частину сценарію – Перевірку: обчислення виконано з точністю 15 вірних знаків, адже теоретично ми повинні отримати 0.

***3.Розв’язування систем лінійних рівнянь алгебри***

Нехай для прикладу маємо наступну СЛАР виду (2).

```
-->A=[2 1 -5 1;1 -3 0 -6;0 2 -1 2; 1 4 -7 6];  //матриця коефіцієнтів
-->f=[8;9;-5;0];                                //права частина
```

Тоді слід переконаттся, що визначник (3) відмінний від нуля. Це можна зробити різними шляхами:

а) з допомогою функції **det(A)**

```
-->D=det(A);                    //визначник A
 D  = 27
```

b) або посередньо – обчислюючи ранг матриці A з допомогою функції **rank(A).** Тут використовується наступний факт, що якщо ранг матриці рівний її розмірності то її визначник відмінний від нуля.

```
-->R=rank(A);                   //визначник A
 R  = 4
```

*Метод оберненої матриці*

Для використання методу оберненої матриці (4) скористаємося функцією **inv(A)**

```
-->x=inv(A)*f
 x  =
   3.
 - 4.
 - 1.
   1.
```

Метод оберненої матриці можна використати також з допомогою безпосередньої матричної операції, про що було згадано у попередніх прикладах матричних операцій

```
-->x=f/A
 x  =
   3.
 - 4.
 - 1.
   1.
```

та

```
-->x=A\f
```

```
 x  =
   3.
 - 4.
 - 1.
   1.
```

*Метод Крамера*

```
-->A1=A; A1(:,1)=f;          //Перша допоміжна матриця
-->A2=A; A2(:,2)=f;          //Друга допоміжна матриця
-->A3=A; A3(:,3)=f;          //Третя допоміжна матриця
-->A4=A; A4(:,4)=f;          //Четверта допоміжна матриця
-->D=det(A);                 //Головний визначник
-->//Допоміжні визначники
-->d(1)=det(A1); d(2)=det(A2); d(3)=det(A3); d(4)=det(A4);
-->x=d/D                              //Вектор розв'язків
 x  =
   3.
 - 4.
 - 1.
   1.
-->P=A*x-b                   //Перевірка
 P  =
   0.
   0.
 - 8.882D-16
   2.665D-15
```

***Метод Гауса***

Реалізація метод Гауса в Scilab (Mathlab) складається з двох етапів. Перший етап - це прямий хід, в результаті якого розширена матриця системи шляхом A-f (елементарних перетворень (перестановка рівнянь системи, множення рівнянь на число, відмінне від нуля, і додавання рівнянь)

приводиться до трикутного виду виду (8). На другому етапі (зворотний хід) трикутну матрицю перетворюють так, щоб в перших n стовпцях вийшла одинична матриця. Останній, **n+1** стовпець цієї матриці містить розв'язування системи лінійних рівнянь.

```
-->C=rref([A  f]);      //Зведення розширеної матриці до верхньої трикутної
-->//Визначення розмірності розширеної матриці
-->[n,m]=size(C);    //m – номер останнього стовпця матриці C
 -->x=C(:,m)          //останній стовпець C – розв'язок X
 x  =
    3.
  - 4.
  - 1.
    1.
```

***Метод ітерації***

За потреби розв'язати СЛАР великої розмірності (n > 50 ~ 100) краще використовувати ітераційний метод **gmres,** який однак добре спрацьовує і для нашого прикладу

```
-->[C, flag, err, iter, res] = gmres(A, f);
-->x=C(:,5))
 x  =
    3.
  - 4.
  - 1.
    1.
```

***Методи універсальні***

Середовища матричної математики Scilab (Mathlab) мають добре розвинену бібліотеку функцій універсального характеру, де метод

розв’язування вибирається автоматично з міркувань доцільності. Серед такич варто відзначити функцію **linsolve**

```
-->[x, kerA]=linsolve(A,-f)
 x  =
   3.
 - 4.
 - 1.
   1.
```

**Практичні завдання**

Розв’язати систему лінійних рівнянь алгебри трьома методами, зробити перевірку.

1. $\begin{cases} -x_1 - x_2 - 2x_3 - 3x_4 = 2 \\ 3x_1 - x_2 - x_3 - 2x_4 = -8 \\ 2x_1 + 3x_2 - x_3 - x_4 = -12 \\ x_1 + 2x_2 + 3x_3 - x_4 = 8 \end{cases}$

2. $\begin{cases} x_1 - 2x_2 + 3x_3 - 2x_4 = -6 \\ x_1 + x_2 - 2x_3 - 3x_4 = -8 \\ 3x_1 - 2x_2 - x_3 + 2x_4 = 4 \\ 2x_1 + 3x_2 + 2x_3 + x_4 = 8 \end{cases}$

3. $\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 5 \\ 2x_1 + x_2 + 2x_3 + 3x_4 = 1 \\ 3x_1 + 2x_2 + x_3 + 2x_4 = 1 \\ 4x_1 + 3x_2 + 2x_3 + x_4 = -5 \end{cases}$

4. $\begin{cases} 0.1x_1 + 0.5x_2 + 0.3x_3 - 0.4x_4 = 2 \\ 0.3x_1 + 0.1x_2 - 0.2x_3 = 0.9 \\ 0.5x_1 - 0.7x_2 + 1x_4 = -0.9 \\ 0.3x_2 - 0.5x_3 = 0.1 \end{cases}$

5. $\begin{cases} 10x_2 + 30x_3 + 40x_4 = -50 \\ 10x_1 + 20x_3 + 30x_4 = -40 \\ 30x_1 + 20x_2 - 50x_4 = 120 \\ 40x_1 + 30x_2 + 50x_3 = 50 \end{cases}$

6. $\begin{cases} 0.3x_1 + x_2 + 1.67x_3 - 2.3x_4 = 4 \\ 3x_1 + 5x_2 + 7x_3 - x_4 = 0 \\ 5x_1 + 7x_2 + x_3 - 3x_4 = 4 \\ 7x_1 + x_2 + 3x_3 - 5x_4 = 16 \end{cases}$

7. $\begin{cases} 2x_1 + x_2 + 5x_3 + x_4 = 8 \\ 0.333x_1 - x_2 - 2x_4 = 3 \\ 2x_2 + x_3 + 2x_4 = -5 \\ x_1 + 4x_2 + 7x_3 + 6x_4 = 0 \end{cases}$

8. $\begin{cases} -x_1 + x_2 + x_3 + x_4 = 12 \\ 2x_1 + x_2 + 2x_3 + 3x_4 = 13 \\ 1.5x_1 + x_2 + 0.5x_3 + x_4 = 7 \\ 4x_1 + 3x_2 + 2x_3 + x_4 = -15 \end{cases}$

Якщо можливо, обчислити матрицю, обернену до матриці D.

1. $D = 2(A^2 + B)(2B - A)$, где

$$A = \begin{pmatrix} 2 & 3 & -1 \\ 4 & 5 & 2 \\ -1 & 0 & 7 \end{pmatrix}, \qquad B = \begin{pmatrix} -1 & 0 & 5 \\ 0 & 1 & 3 \\ 2 & -2 & 4 \end{pmatrix}$$

2. $D = 3A - (A + 2B)B^2$, где

$$A = \begin{pmatrix} 4 & 5 & -2 \\ 3 & -1 & 0 \\ 4 & 2 & 7 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & 1 & -1 \\ 0 & 1 & 3 \\ 5 & 7 & 3 \end{pmatrix}$$

3. $D = 3A^2 - (A + 2B)B$, где

$$A = \begin{pmatrix} 4 & 5 & -2 \\ 3 & -1 & 0 \\ 4 & 2 & 7 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & 1 & -1 \\ 0 & 1 & 3 \\ 5 & 7 & 3 \end{pmatrix}$$

4. $D = (A - B^2 2)(2A + B^3)$, где

$$A = \begin{pmatrix} 5 & 2 & 0 \\ 10 & 4 & 1 \\ 7 & 3 & 2 \end{pmatrix}, \qquad B = \begin{pmatrix} 3 & 6 & -1 \\ -1 & -2 & 0 \\ 2 & 1 & 3 \end{pmatrix}$$

5. $D = 2(A - B)(A^2 + B)$, где

$$A = \begin{pmatrix} 5 & 1 & 7 \\ -10 & -2 & 1 \\ 0 & 1 & 2 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & 4 & 1 \\ 3 & 1 & 0 \\ 7 & 2 & 1 \end{pmatrix}$$

6. $D = (A - B)^2 A + 2B$, где

$$A = \begin{pmatrix} 5 & -1 & 3 \\ 0 & 2 & -1 \\ -2 & -1 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 3 & 7 & -2 \\ 1 & 1 & -2 \\ 0 & 1 & 3 \end{pmatrix}$$

7. $D = (A^2 - B^2)(A + B^2)$, где

$$A = \begin{pmatrix} 7 & 2 & 0 \\ -7 & -2 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 2 & 3 \\ 1 & 0 & -2 \\ 3 & 1 & 1 \end{pmatrix}$$

8. $D = 2(A - B)(A^2 + B)$, где

$$A = \begin{pmatrix} 5 & 1 & 7 \\ -10 & -2 & 1 \\ 0 & 1 & 2 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & 4 & 1 \\ 3 & 1 & 0 \\ 7 & 2 & 1 \end{pmatrix}$$

9. $D = 2A - (A^2 + B)B$, где

$$A = \begin{pmatrix} 1 & 4 & 2 \\ 2 & 1 & -2 \\ 0 & 1 & -1 \end{pmatrix}, \qquad B = \begin{pmatrix} 4 & 6 & -2 \\ 4 & 10 & 1 \\ 2 & 4 & -5 \end{pmatrix}$$

10. $D = 2(A - 0,5B) + A^3 B$, где

$$A = \begin{pmatrix} 5 & 3 & -1 \\ 2 & 0 & 4 \\ 3 & 5 & -1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 4 & 16 \\ -3 & -2 & 0 \\ 5 & 7 & 2 \end{pmatrix}$$

11. $D = (A - B)A^2 + 3B$, где

$$A = \begin{pmatrix} 3 & 2 & -5 \\ 4 & 2 & 0 \\ 1 & 1 & 2 \end{pmatrix}, \qquad B = \begin{pmatrix} -1 & 2 & 4 \\ 0 & 3 & 2 \\ -1 & -3 & 4 \end{pmatrix}$$

12. $D = 3(A^2 + B^2) - 2AB$, где

$$A = \begin{pmatrix} 4 & 2 & 1 \\ 3 & -2 & 0 \\ 0 & -1 & 2 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & 0 & 2 \\ 5 & -7 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

13. $D = 2A^3 + 3B(AB - 2A)$, где

$$A = \begin{pmatrix} 1 & -1 & 0 \\ 2 & 0 & -1 \\ 1 & 1 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 5 & 3 & 1 \\ -1 & 2 & 0 \\ -3 & 0 & 0 \end{pmatrix}$$

14. $D = A(A^2 - B) - 2(B + A)B$, где

$$A = \begin{pmatrix} 2 & 3 & 1 \\ -1 & 2 & 4 \\ 5 & 3 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & 7 & 13 \\ -1 & 0 & 5 \\ 5 & 13 & 21 \end{pmatrix}$$

15. $D = (2A - B)(3A + B) - 2A^2B$, где

$$A = \begin{pmatrix} 1 & 0 & 3 \\ -2 & 0 & 1 \\ -1 & 3 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 7 & 5 & 2 \\ 0 & 1 & 2 \\ -3 & -1 & -1 \end{pmatrix}$$