§4. Решение систем линейных алгебраических уравнений.

Рассмотрим систему n линейных алгебраических уравнений с n неизвестными (СЛАУ) вида

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n = b_n \end{cases}$$

или $AX = B$, где a_{ij} – коэффициенты системы, $A = (a_{ij})$ – матрица системы, X – столбец неизвестных x_j, B – столбец свободных членов b_i, $i = \overline{1, n}$, $j = \overline{1, n}$.

Будем также предполагать, что определитель главной матрицы системы A отличен от нуля. Случаи, когда число неизвестных не совпадает с числом уравнений или когда $\det A = 0$, можно свести к этому.

Рассмотрим два вида методов решения систем линейных уравнений:

1) **прямые** (или **точные**) – методы, которые в предположении, что вычисления ведутся без округлений, позволяют за конечное число шагов получить точное решение системы или установить ее неразрешимость;

2) **итерационные** – методы, позволяющие получить приближенное решение системы за конечное число шагов, количество которых зависит от требуемой точности приближения.

Эффективность методов решения СЛАУ во многом зависит от структуры и свойств матрицы системы, одним из которых является ее *обусловленность*.

Нормы векторов и матриц. Число обусловленности матрицы. Погрешность решения СЛАУ прямыми методами.

Наиболее часто используются следующие нормы векторов и согласованные с ними нормы матриц.

1. ***Первая норма вектора*** $\vec{x} = (x_1, x_2, \ldots, x_n)$ (1-норма, норма-сумма, октаэдрическая норма):

$$\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i| .$$

Первая норма матрицы $A=\left(a_{ij}\right)$ (1-норма, норма-сумма, столбцовая норма):

$$\left\|A\right\|_1=\max_{1\le j\le n}\sum_{i=1}^{n}\left|a_{ij}\right|.$$

Здесь суммируются взятые по модулю элементы каждого столбца, а затем выбирается наибольшее из полученных чисел.

2. ***Норма-максимум вектора*** $\vec{x}=(x_1,\,x_2,\,...,\,x_n)$ (∞-норма, кубическая норма):

$$\left\|\vec{x}\right\|_\infty=\max_{1\le i\le n}\left|x_i\right|.$$

Норма-максимум матрицы $A=\left(a_{ij}\right)$ (∞-норма, строчная норма):

$$\left\|A\right\|_\infty=\max_{1\le i\le n}\sum_{j=1}^{n}\left|a_{ij}\right|.$$

Здесь суммируются взятые по модулю элементы каждой строки, а затем выбирается наибольшее из полученных чисел.

3. ***Евклидова норма вектора*** $\vec{x}=(x_1,\,x_2,\,...,\,x_n)$ (сферическая норма):

$$\left\|\vec{x}\right\|_2=\sqrt{\sum_{i=1}^{n}x_i^2}\,.$$

Евклидова норма матрицы A (2-норма, спектральная норма):

$$\left\|A\right\|_2=\sqrt{\lambda}\,,$$

где λ – наибольшее собственное значение матрицы A^TA (все собственные значения матриц такого вида являются неотрицательными действительными числами).

Числом обусловленности невырожденной матрицы A называется число, которое обозначается $cond(A)$, и вычисляется по формуле

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

Для любой матрицы A в указанных нормах справедливо неравенство $cond(A)\ge 1$.

Матрица A^{-1}, обратная к невырожденной матрице A называется ***устойчивой***, если малым изменениям элементов матрицы A соответствуют малые изменения элементов матрицы A^{-1}. В противном случае обратная матрица называется ***неустойчивой***.

Матрица A называется ***хорошо обусловленной***, если матрица A^{-1} устойчива, и ***плохо обусловленной,*** если обратная матрица неустойчива.

Хорошо обусловленными являются матрицы с малым числом обусловленности, плохо обусловленными – с большим $cond(A)$. Плохо обусловленными часто являются почти вырожденные матрицы, у которых определитель близок к нулю.

Пусть в системе линейных уравнений

$$AX = B$$

правая часть оказалась заданной неточно (получила возмущение) или в процессе вычислений возникли, например, ошибки округления. Тогда фактически решается возмущенная система

$$AX = B + \Delta B \quad \text{или} \quad (A + \Delta A)X = B + \Delta B.$$

Обозначим через X – решение точной системы $AX = B$, X^* – решение возмущенной системы. Решение X^* можно считать приближенным решением системы $AX = B$.

Оценить ***погрешность*** приближенного решения X^* можно при помощи двух векторов – ***вектора ошибки***

$$\Delta X = X - X^*$$

и ***вектора невязки***

$$R = AX^* - B.$$

Вектор невязки показывает, насколько левая часть системы $AX = B$, вычисленная на приближенном решении X^*, отличается от правой части системы.

Если матрица A линейной системы уравнений плохо обусловлена, то незначительные изменения ее коэффициентов или правых частей уравнений могут приводить к большим изменениям решения. Системы с большим числом обусловленности $cond(A)$ называются ***плохо обусловленными***.

Справедливо соотношение, связывающее предельную относительную погрешность решения $\delta_X = \|X^* - X\| / \|X^*\|$, число обусловленности матрицы системы $cond(A)$ и относительную погрешность правых частей системы $\delta_B = \|\Delta B\| / \|B + \Delta B\|$:

$$\delta_X = cond(A) \cdot \delta_B.$$

Если в системе $AX = B$ получила приращение не только правая часть B, но и матрица системы A, то при условии $\|A^{-1}\| \cdot \|\Delta A\| < 1$ верно

$$\delta_X = \frac{cond(A)}{1 - \delta_A \cdot cond(A)} \cdot (\delta_B + \delta_A),$$

где $\delta_A = \|\Delta A\| / \|A + \Delta A\|$ – относительная погрешность матрицы системы.

Прямые методы решения систем линейных уравнений.

Перечислим основные ***прямые*** (***точные***) методы решения линейных систем $AX = B$ с невырожденной матрицей A.

1. ***Матричный метод***. Решение получают по формуле $X = A^{-1}B$.

2. ***Метод Крамера***. Решение получают по формулам $x_i = \dfrac{d_i}{d}$, где $d = \det A$, d_i – определитель матрицы, полученной из матрицы A заменой в ней i-го столбца столбцом свободных членов B, $i = \overline{1, n}$.

3. ***Метод Гаусса*** **(*метод исключения неизвестных, схема единственного деления*)**. Решение этим методом состоит из двух этапов.

Сначала систему $AX = B$ с помощью элементарных преобразований над строками ее расширенной матрицы $(A|B)$ приводят к равносильной системе с верхней треугольной матрицей (***прямой ход*** метода Гаусса):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_{1,} \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)}, \\ a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n = b_3^{(2)}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{nn}^{(n-1)}x_n = b_n^{(n-1)}. \end{cases}$$

Для этого в первую очередь получают нули в первом столбце под элементом a_{11}, вычитая из i-й строки первую строку, умноженную на a_{i1}/a_{11} $(i = \overline{2, n})$. Если $a_{11} = 0$, то предварительно переставляют строки (уравнения системы) так, чтобы $a_{11} \neq 0$.

Затем получают нули во втором столбце под главной диагональю, т.е. в строках с третьей по n-ю, вычитая из i-й строки измененную вторую строку, умноженную на $a_{i2}^{(1)}/a_{22}^{(1)}$ $(i = \overline{3, n})$. И так далее. Элементы $a_{ii}^{(i-1)}$, на которые осуществляется деление, называются ***ведущими*** (или ***главными***) ***элементами*** метода Гаусса и не должны равняться нулю.

Обратный ход метода Гаусса заключается в последовательном определении неизвестных, начиная с x_n и заканчивая x_1 из системы, полученной на первом этапе:

$$x_n = b_n^{(n-1)} \Big/ a_{nn}^{(n-1)}, \quad x_i = \left(b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j \right) \Big/ a_{ii}^{(i-1)}, \quad i = \overline{n-1,1}.$$

Чтобы уменьшить влияние ошибок округлений и исключить деление на нуль рассматривают различные модификации метода Гаусса, например, метод Гаусса *с постолбцовым выбором главного элемента*, метод *оптимального исключения* и другие.

4. ***Метод прогонки*** решения систем с трехдиагональными матрицами. Этот метод является модификацией метода Гаусса и применяется к решению систем вида

$$\begin{cases} b_1 x_1 + c_1 x_2 = d_1, \\ a_2 x_1 + b_2 x_2 + c_2 x_3 = d_2, \\ \qquad a_3 x_2 + b_3 x_3 + c_3 x_4 = d_3, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \qquad\qquad a_{n-1} x_{n-2} + b_{n-1} x_{n-1} + c_{n-1} x_n = d_{n-1}, \\ \qquad\qquad\qquad a_n x_{n-1} + \ b_n x_n = d_n. \end{cases}$$

Матрица этой системы имеет ***трехдиагональный*** вид – ее ненулевые элементы расположены только на главной диагонали, над ней и под ней:

Система может быть записана в краткой форме:

$$\begin{cases} a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \quad i = \overline{1, n}, \\ a_1 = 0, \quad c_n = 0. \end{cases}$$

Метод прогонки состоит из двух этапов: первый – ***прямая прогонка***, второй – ***обратная прогонка***.

На первом этапе систему преобразуют так, чтобы ее ненулевые коэффициенты оставались только на главной диагонали и над ней, а именно, к виду

$$\begin{cases} x_i = L_i x_{i+1} + M_i, \quad i = \overline{1, n-1}, \\ x_n = M_n. \end{cases}$$

Числа L_i и M_i называются ***прогоночными коэффициентами*** и вычисляются по рекуррентным формулам:

$$L_1 = -\frac{c_1}{b_1}, \quad M_1 = \frac{d_1}{b_1}, , \quad L_i = -\frac{c_i}{b_i + a_i L_{i-1}}, \quad M_i = \frac{d_i - a_i M_{i-1}}{b_i + a_i L_{i-1}}, \quad i = \overline{2, n}.$$

На втором этапе (*обратная прогонка*) последовательно находят неизвестные, начиная с x_n, по формулам

$$x_n = M_n, \;\; x_i = L_i x_{i+1} + M_i, \quad i = \overline{1, n-1}.$$

Если выполняются неравенства

$$|b_i| \geq |a_i| + |c_i|, \;\; i = \overline{1, n},$$

причем хотя бы для одного значения i неравенство является строгим, $a_i \neq 0$, $c_i \neq 0$ (кроме $a_1 = 0$, $c_n = 0$ по условию), то знаменатели всех прогоночных коэффициентов не обращаются в нуль и система линейных уравнений имеет единственное решение.

Количество арифметических операций, которое нужно выполнить для получения решения системы n уравнений методом прогонки, пропорционально n.

Итерационные методы решения систем линейных уравнений.

Процесс решения системы линейных уравнений итерационными методами состоит в построении последовательности приближений к решению, сходящейся к нему. Существует большое количество итерационных методов, рассмотрим некоторые из них.

1. *Метод простой итерации*.

Рассмотрим систему n линейных алгебраических уравнений с n неизвестными $AX = B$ с невырожденной матрицей A. Преобразуем эту систему каким-нибудь способом (их бесконечно много) к виду

$$X = \Phi X + \beta,$$

где Φ – квадратная матрица порядка n, β – вектор-столбец.

Согласно методу простой итерации для начала вычислений задают начальное приближение $X^{(0)}$ (например, $X^{(0)} = \beta$), все следующие приближения к решению находят с помощью рекуррентной формулы

$$X^{(k)} = \Phi X^{(k-1)} + \beta, \quad k = 1, 2, \ldots$$

Для того чтобы метод простых итераций сходился к решению системы $X = \Phi X + \beta$ при любом начальном приближении $X^{(0)}$ ***необходимо и достаточно***, чтобы все собственные значения матрицы Φ были по модулю меньше единицы.

На практике это бывает трудно проверить, поэтому часто пользуются ***достаточными условиями сходимости***.

Если какая-нибудь норма матрицы Φ меньше единицы ($\|\Phi\| = q < 1$), то метод простых итераций сходится к единственному решению X^* системы $X = \Phi X + \beta$ при любом начальном приближении $X^{(0)}$. Причем для всех $k \in \mathbb{N}$ справедливы оценки погрешности:

1) $\left\|X^* - X^{(k)}\right\| \le \dfrac{q^k}{1-q}\left\|X^{(1)} - X^{(0)}\right\|$ (*априорная*);

2) $\left\|X^* - X^{(k)}\right\| \le \dfrac{q}{1-q}\left\|X^{(k)} - X^{(k-1)}\right\|$ (*апостериорная*).

Чем меньше норма матрицы Φ, тем быстрее сходится итерационный процесс.

Чтобы найти решение системы с точностью ε, то, как следует из апостериорной оценки погрешности, можно использовать следующее *условие окончания итерационного процесса*:

$$\left\|X^{(k)} - X^{(k-1)}\right\| < \frac{1-q}{q} \cdot \varepsilon, \quad \text{где} \quad q = \|\Phi\|.$$

2. ***Метод Якоби***.

Этот метод является частным случаем метода простой итерации.

Пусть все диагональные элементы матрицы A системы $AX = B$ отличны от нуля ($a_{ii} \neq 0$). Тогда, выражая диагональные элементы через остальные, получим систему вида $X = \Phi X + \beta$:

$$\begin{cases} x_1 = -\left(a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n - b_1\right)/a_{11}, \\ x_2 = -\left(a_{21}x_1 + a_{23}x_3 + \ldots + a_{2n}x_n - b_2\right)/a_{22}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = -\left(a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{n,n-1}x_{n-1} - b_n\right)/a_{nn} \end{cases}$$

или

$$x_i = -\frac{1}{a_{ii}}\left[\sum_{j=1,\, j\neq i}^{n} a_{ij}x_j - b_i\right], \quad i = \overline{1, n}.$$

В *методе Якоби* используется следующий алгоритм построения приближений:

$$x_i^{(k)} = -\frac{1}{a_{ii}}\left[\sum_{j=1,\ j\neq i}^{n} a_{ij}x_j^{(k-1)} - b_i\right],\quad i=\overline{1,n}.$$

Условия сходимости метода Якоби совпадают с условиями сходимости метода простой итерации. Отметим еще один вариант ***достаточного условия***:

если A – матрица с диагональным преобладанием, т.е. $|a_{ii}| > \sum_{j=1,\ j\neq i}^{n}|a_{ij}|$ для всех $i=\overline{1,n}$, то метод Якоби сходится при любом начальном приближении $X^{(0)}$.

3. ***Метод Зейделя.***

Метод Зейделя – это модификация метода простой итерации, в которой при k-й итерации для вычисления $x_i^{(k)}$ используются уже вычисленные на этом шаге новые значения $x_1^{(k)}$, $x_2^{(k)}$, …, $x_{i-1}^{(k)}$, а остальные берутся из предыдущего $(k-1)$-го шага. Это приводит, как правило, к ускорению сходимости.

Пусть система $AX = B$ приведена к виду $X = \Phi X + \beta$. Тогда расчетные формулы метода Зейделя имеют вид:

$$\begin{cases} x_1^{(k)} = \varphi_{11}x_1^{(k-1)} + \varphi_{12}x_2^{(k-1)} + \cdots + \varphi_{1n}x_n^{(k-1)} + \beta_1, \\ x_2^{(k)} = \varphi_{21}x_1^{(k)} + \varphi_{22}x_2^{(k-1)} + \cdots + \varphi_{2n}x_n^{(k-1)} + \beta_2, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_n^{(k)} = \varphi_{n1}x_1^{(k)} + \varphi_{n2}x_2^{(k)} + \cdots + \varphi_{nn}x_n^{(k-1)} + \beta_n, \end{cases}$$

или

$$x_i^{(k)} = \sum_{j=1,\ i\neq 1}^{i-1}\varphi_{ij}x_j^{(k)} + \sum_{j=i}^{n}\varphi_{ij}x_j^{(k-1)} + \beta_i,\quad i=\overline{1,n}.$$

Условия сходимости метода Зейделя совпадают с условиями сходимости метода простой итерации. Если система $AX = B$ приведена к виду $X = \Phi X + \beta$ так же, как в методе Якоби, то метод Зейделя сходится, если только матрица A удовлетворяет условию усиленного доминирования главной диагонали, т.е.

$$\sum_{j=1,\ j\neq i}^{n}|a_{ij}| \le \alpha\cdot|a_{ii}|,\quad 0<\alpha<1 \quad \text{для всех } i=\overline{1,n}.$$

Основные операции и функции пакета Mathematica для работы с векторами, матрицами и системами линейных алгебраических уравнений.

В пакете **Mathematica** любой набор элементов, заключенный в фигурные скобки, является списком. Вектор – список, матрицу заменяет список списков.

Ввести матрицу в программе **Mathematica** можно несколькими способами:

1) Непосредственно ввести с клавиатуры в виде списка списков, например $\{\{a, b\}, \{c, d\}\}$.
2) На главном меню выбрать **Insert** → **Table/Matrix** → **New...** В появившемся окне необходимо выбрать **Matrix** (**List of lists**) и ввести количество строк, столбцов, а затем нажать **OK** . Появится матрица, которую следует заполнить числами.
3) При помощи панели инструментов выбрать **Palettes** → **Basic Math Assistant** . Перейти на закладку **Advanced** и щелкнуть по кнопке **Matrix** для ввода матрицы 2×2. Дополнительная строка добавляется нажатием комбинации клавиш **Ctrl+Enter**, а столбец – **Ctrl+«,»** (запятая).

Array[*f*, *n*] – создает одномерный массив (вектор) длины *n* вида $\{f[1], f[2], ..., f[n]\}$.

Array[*f*, $\{n_1, n_2\}$] – создает двумерный массив (матрицу) размера $n_1 \times n_2$ с элементами $f[i, j]$, $i = \overline{1, n_1}$, $j = \overline{1, n_2}$.

A**[[*i*, *j*]]** – выбирает элемент a_{ij} матрицы A.

A**[[*i*]]** – выбирает *i*-ю строку матрицы A.

Append[*lst*,*x*] – создает новый список, добавляя элемент *x* в конец списка *lst*.

CharacteristicPolynomial[A , λ] – возвращает характеристический многочлен матрицы A относительно переменной λ.

ConstantArray[*b*, $\{m, n\}$] – задает постоянную матрицу размера $m\times n$ с элементами *b*.

Cross[v_1, v_2] или $v_1 \times v_2$ – находит векторное произведение векторов v_1 и v_2.

Dot[v_1, v_2] или $v_1.v_2$ – вычисляет скалярное произведение векторов v_1 и v_2.

Dot[*A*, *B*] или *A.B* – вычисляет произведение матриц A и B.

Det[A] – вычисляет определитель квадратной матрицы A.

***DiagonalMatrix*[*list*]** – создает диагональную матрицу, размещая на главной диагонали элементы списка *list*.

***Dimension*[A]** – определяет размерность матрицы A.

***Drop*[A, *i*]** – удаляет *i*-тую строку матрицы A.

***Drop*[A, {*j*}, {*k*}]** – удаляет *j*-й и *k*-й столбец матрицы A.

***Eigenvalues*[A]** – возвращает список собственных значений квадратной матрицы A.

***Eigenvectors*[A]** – возвращает список собственных векторов квадратной матрицы A.

***IdentityMatrix*[*n*]** – задает единичную матрицу *n*-го порядка.

***Inverse*[A]** – находит обратную матрицу A^{-1} для квадратной матрицы A.

Join[$list_1$, $list_2$, … **]** – объединяет списки $list_1$, $list_2$, … в единую цепочку.

Join[A, *row* **]** – добавляет к матрице A размера $m \times n$ строку *row*, которая должна содержать *n* элементов.

MatrixForm*[A]** или A//MatrixForm*** – выводит матрицу A в традиционной форме, а не в виде вложенного списка.

***MatrixPower*[A, *n*]** – возводит матрицу A в степень *n*.

***MatrixRank*[A]** – вычисляет ранг матрицы A.

***Minors*[A, *k*]** – возвращает список миноров порядка *k* матрицы A.

***Norm*[*expr, p*]** – возвращает *p*-норму вектора или матрицы *expr*.

Prepend[*lst,x*] – создает новый список, добавляя *x* в начало списка *lst*.

***RowReduce*[C]** – приводит матрицу C к ступенчатому виду, выполняя элементарные преобразования по строкам.

***Tranpose*[A]** – транспонирует матрицу A.

FindRoot $[\{eqn_1, eqn_2, ..., eqn_n\}, \{x_1, x_1^{(0)}\}, \{x_2, x_2^{(0)}\}, \ldots, \{x_n, x_n^{(0)}\}]$ – находит численное решение системы *n* уравнений $eqn_1, eqn_2, ..., eqn_n$ с *n* неизвестными x_1, x_2, … x_n, если задано начальное приближение $(x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})$.

***LinearSolve*[A, B]** – находит решение системы линейных алгебраических уравнений вида $AX = B$.

***NSolve*[{**$\{eqn_1, eqn_2, ...\}$**,**$\{x_1, x_2, ...\}$**}]** – находит численное решение заданной системы уравнений.

***NSolve*[**$\{eqn_1, eqn_2, ...\}$**,**$\{x_1, x_2, ...\}$**,** Reals**]** – находит численное решение заданной системы уравнений на множестве действительных чисел.

***Solve*[**$\{eqn_1, eqn_2, ...\}$**,**$\{x_1, x_2, ...\}$**]** – находит решение системы уравнений $eqn_1, eqn_2, ...$ относительно переменных $x_1, x_2, ...$

Примеры решения систем линейных уравнений и исследования погрешности их решений средствами пакета Mathematica.

Пример 4.1. Решить две системы линейных уравнений $AX = B$ и $AX = B + \Delta B$ (точную и возмущенную), где

$$A = \begin{pmatrix} 1 & 10 \\ 100 & 1001 \end{pmatrix},\ B = \begin{pmatrix} 11 \\ 1101 \end{pmatrix},\ \Delta B = \begin{pmatrix} 0{,}01 \\ 0 \end{pmatrix},\ X = \begin{pmatrix} x \\ y \end{pmatrix}.$$

Найти число обусловленности матрицы A в нормах $\|\cdot\|_1$ и $\|\cdot\|_\infty$. Найти предельную относительную погрешность возмущенной системы. Сравнить полученные решения, оценив их абсолютную и относительную погрешности.

Δ Введем данные матрицы в виде списков:

```
In[1]:= A = {{1, 10}, {100, 1001}}; B = {11, 1101}; dB = {0.01, 0}; X = {x, y};
```

Команда **MatrixForm** выдает результат в матричной форме:

```
In[2]:= A // MatrixForm
        матричная форма

Out[2]//MatrixForm=
        1    10
        100  1001
```

Запишем систему в матричной форме, операцию произведения матриц зададим с помощью точки (уравнение в системе **Mathematica** формируется двойным знаком равенства «==»):

```
In[3]:= A.X == B

Out[3]= {x + 10 y, 100 x + 1001 y} == {11, 1101}
```

Решим систему (или матричное уравнение) с помощью универсальной функции **Solve**:

```
In[4]:= sol = Solve[A.X == B, {x, y}]
        решить уравнения
Out[4]= {{x → 1, y → 1}}
```

Полученное решение является вложенным списком второго уровня. Понизим уровень, т.е. уберем внешние скобки, с помощью функции **Flatten**.

```
In[5]:= Flatten[sol]
        уплостить
Out[5]= {x → 1, y → 1}
```

Решение **X** данной системы сохраним в виде вектора **sol1** – списка с явными значениями переменных:

```
In[6]:= sol1 = X /. Flatten[sol]
        уплостить
Out[6]= {1, 1}
```

Чтобы использовать их отдельно, можно поступить следующим образом:

```
In[7]:= {x, y} = sol1
Out[7]= {1, 1}

In[8]:= x
Out[8]= 1

In[9]:= y
Out[9]= 1
```

Перейдем к решению возмущенной системы $AX = B + \Delta B$. Снова воспользуемся функцией **Solve**. Полученное решение сохраним в виде вектора **sol2.**

```
In[10]:= Clear[x, y]
         очистить

In[11]:= Solve[A.X == B + dB, {x, y}]
         решить уравнения
Out[11]= {{x → 11.01, y → 6.51911 x 10^-14}}
```

```
In[12]:= sol2 = Flatten[%]
         уплостить
Out[12]= {x → 11.01, y → 6.51911 x 10^-14}

In[13]:= sol2 = X /. sol2
Out[13]= {11.01, 6.51911 x 10^-14}
```

Решим возмущенную систему вторым способом, используя функцию **LinearSolve**, предназначенную и реализующую методы решения именно линейных систем:

```
In[14]:= sol21 = LinearSolve[A, B + dB]
         решить линейные уравнени:
Out[14]= {11.01, 0.}
```

Как видно, решение, полученное с помощью функции **Solve**, из-за погрешности ее методов, оказалось неточным. В дальнейших вычислениях будем использовать решение **sol21** возмущенной системы.

Для нахождения числа обусловленности системы найдем обратную матрицу с помощью функции **Inverse**:

```
In[15]:= A1 = Inverse[A]
         обратная мат
Out[15]= {{1001, -10}, {-100, 1}}
```

Вычислим нормы матриц, не используя встроенную функцию системы **Mathematica**.

По определению, норма-максимум матрицы A равна $\|A\|_\infty = \max\limits_{1\le i\le n} \sum\limits_{j=1}^{n} |a_{ij}|$. Для каждой из матриц ($A$ и $A^{-1} = A1$) сформируем вектор, координатами которого являются суммы взятых по модулю элементов каждой строки матрицы. Затем найдем наибольшую из координат, т.е. норму-максимум матриц A и A^{-1}.

```
In[16]:= t1 = Table[Sum[Abs[A[[i, j]]], {i, 1, 2}]
           таблица      абсолютное значение
Out[16]= {11, 1101}

         t2 = Table[Sum[Abs[A1[[i, j]]], {i, 1, 2}]
              таблица      абсолютное значение
Out[17]= {1011, 101}

In[18]:= norm1 = Max[t1]
            максимум
Out[18]= 1101

         norm2 = Max[t2]
            максимум
Out[19]= 1011
```

Число обусловленности матрицы A в норме-максимум равно:

```
In[20]:= condA = norm1 norm2
Out[20]= 1 113 111
```

Аналогичным образом найдем число обусловленности матрицы A в норме-сумме.

```
In[21]:= t11 = Table[Sum[Abs[A[[i, j]]], {i, 1, 2}], {j, 1, 2}]
          таблица  сумма  абсолютное значение

Out[21]= {101, 1011}

         t21 = Table[Sum[Abs[A1[[i, j]]], {i, 1, 2}], {j, 1, 2}]
          таблица  сумма  абсолютное значение

Out[22]= {1101, 11}

In[23]:= norm11 = Max[t11]
          максимум

Out[23]= 1011

         norm21 = Max[t21]
          максимум

Out[24]= 1101

In[25]:= condA1 = norm11 norm21

Out[25]= 1 113 111
```

Число обусловленности матрицы A равно 1113111, т.е. является величиной порядка 10^6. Матрица A плохо обусловлена.

Для анализа реакции решения на «возмущение» правой части системы найдем вектор ошибки $\Delta X = X - X^*$:

```
In[26]:= deltaX = sol21 - sol1

Out[26]= {10.01, -1.}
```

Абсолютную погрешность решения $\|\Delta X\| = \|X^* - X\|$ вычислим в норме-сумме, используя функцию **Norm**:

```
In[27]:= nd = Norm[deltaX, 1]
           норма

Out[27]= 11.01
```

Тогда относительная погрешность решения возмущенной системы $\delta_X = \|X^* - X\| / \|X^*\|$ в этой норме равна

```
In[28]:= nrd = nd / Norm[sol21, 1]

Out[28]= 1.
```

или 100%.

Вычислим прогнозируемую предельную относительную погрешность решения возмущенной системы $\delta_X = cond(A) \cdot \delta_B$, где $\delta_B = \|\Delta B\| / \|B + \Delta B\|$:

```
In[29]:= pr = condA1 * Norm[dB, 1] / Norm[B + dB, 1]

Out[29]= 10.0099
```

или 1001%.

По определению относительная погрешность решения не превосходит его предельную относительную погрешность. Это условие выполнено.

Пример 4.2. Решить методом Гаусса систему линейных уравнений $AX = B$, где $A = (a_{ij})$, $B = (b_i)$,

$$a_{ij} = \begin{cases} 6 - i - 2j, & i + j > 5, \\ 75 - 15i, & i + j = 5, \\ i + j - 6, & i + j > 5, \end{cases} \qquad b_i = \frac{i}{2} \cdot (i - 35) + 80, \qquad i = \overline{1, 4}, \ j = \overline{1, 4}.$$

Δ Введем матрицы A и B с помощью функции **Table** и условного оператора **If**.

```
In[1]:= A = Table[If[i + j > 5, i + j - 6,
                    табл··· условный оператор
                                If[i + j == 5, 75 - 15 i, 6 - i - 2 j]], {i, 4}, {j, 4}];
                                условный оператор

        B = Table[i/2 (i - 35) + 80, {i, 4}];
                    таблица значений

In[3]:= {MatrixForm[A], MatrixForm[B]}
          матричная форма  матричная форма

Out[3]= {{{3, 1, -1, 60}, {2, 0, 45, 0}, {1, 30, 0, 1}, {15, 0, 1, 2}}, {{63}, {47}, {32}, {18}}}
```

Организуем прямой ход метода Гаусса с постолбцовым выбором главного элемента. Найдем уравнение с максимальным по модулю элементом в первом столбце. Поменяем это уравнение местами с первым уравнением и исключим переменную x_1 из остальных уравнений. Аналогичным образом поступим со вторым столбцом, рассматривая новую систему без первого уравнения. И так далее.

Введем обозначения: n – количество уравнений системы и ее неизвестных, Coef – пока еще пустой список ведущих коэффициентов системы (именно на них производится деление на каждом шаге).

```
In[4]:= n = Length[B];
        длина
      Coef = List[];(* Список ведущих коэффициентов*)
             список
```

Для реализации прямого хода составим несколько вложенных циклов.

```
In[6]:= For[k = 1, k ≤ n - 1, k++,
        цикл ДЛЯ
  (* Выбрать уравнение p, в котором находится ведущий коэффициент A[p,k]*)
    t = Abs[A[[k, k]]]; p = k;
        абсолютное значение
    For[m = k + 1, m ≤ n, m++, If[Abs[A[[m, k]]] > t, p = m]];
    цикл ДЛЯ                   у··· абсолютное значение
    If[Abs[A[[p, k]]] < $MachineEpsilon, Return[{"Решение не найдено", MatrixForm[A]}]];
    у··· абсолютное значе··· машинная эпсилон вернуть управление матричная форма
  (* Если ведущий коэффициент по модулю меньше константы $MachineEpsilon,
  то закончить вычисления*)
    Coef = Append[Coef, A[[p, k]]]; (* Добавить ведущий коэффициент в список*)
           добавить в конец
    If[k < p, (* Переставить уравнение p и уравнение k*)
    условный оператор
    For[m = k, m ≤ n, m++,
    цикл ДЛЯ
      t = A[[k, m]]; A[[k, m]] = A[[p, m]]; A[[p, m]] = t;
      t = B[[k]]; B[[k]] = B[[p]]; B[[p]] = t];
    For[i = k + 1, i ≤ n, i++, (* Исключить неизвестное из уравнений k+1,...n*)
    цикл ДЛЯ
      t = A[[i, k]] / A[[k, k]]; A[[i, k]] = 0;
      For[j = k + 1, j ≤ n, j++,
      цикл ДЛЯ
        A[[i, j]] = A[[i, j]] - t × A[[k, j]]];
        B[[i]] = B[[i]] - t × B[[k]]]];
```

В результате прямого хода получена система с верхней треугольной матрицей.

```
In[7]:= {MatrixForm[A], MatrixForm[B]}
        матричная форма  матричная форма
```

$$\text{Out[7]=}\left\{\begin{pmatrix} 15 & 0 & 1 & 2 \\ 0 & 1 & -\frac{6}{5} & \frac{298}{5} \\ 0 & 0 & \frac{673}{15} & -\frac{4}{15} \\ 0 & 0 & 0 & -\frac{1282597}{673} \end{pmatrix}, \begin{pmatrix} 18 \\ \frac{297}{5} \\ \frac{223}{5} \\ -\frac{1282597}{673} \end{pmatrix}\right\}$$

Далее найдем решение системы, выполняя обратный ход метода Гаусса. Введем нулевой вектор X .

```
X = Table[0, {n}];
```
таблица значений

Из последнего уравнения найдем x_4. Затем из третьего уравнения – x_3. И так далее. Все вычисления организуем в цикле.

```
In[9]:= X[[n]] = B[[n]] / A[[n, n]];

In[10]:= For[i = n - 1, i ≥ 1, i--,
```
цикл ДЛЯ

```
  X[[i]] = (B[[i]] - ∑_{j=i+1}^{n} A[[i, j]] × X[[j]]) / A[[i, i]]];
```

Выведем решение X и вектор невязки $R = AX - B$.

```
In[11]:= {X, A.X - B}
Out[11]= {{1, 1, 1, 1}, {0, 0, 0, 0}}
```

Так как вектор невязки является нулевым, то получено точное решение $(1,1,1,1)$.