

Решение систем линейных алгебраических уравнений.

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

[illegible]

или $AX = B$, где a_{ij} – коэффициенты системы, $A = (a_{ij})$ – матрица системы, X – столбец неизвестных x_i , 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 $\vec{x} = (x_1, x_2, \dots, x_n)$ (1-норма, норма-сумма, октаэдрическая норма):

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Для любой матрицы A в указанных нормах справедливо неравенство $\text{cond}(A) \geq 1$.

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

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

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

2. Метод Крамера. Решение получают по формулам $x_i = \frac{d_i}{d}$, где $d = \det A$,

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

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1, \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \dots + a_{2n}^{(1)}x_n = b_2^{(1)}, \\ a_{33}^{(2)}x_3 + \dots + a_{3n}^{(2)}x_n = b_3^{(2)}, \\ \dots\dots\dots \\ a_{nn}^{(n-1)}x_n = b_n^{(n-1)}. \end{array} \right.$$

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:

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

Чтобы уменьшить влияние ошибок округлений и исключить деление на нуль рассматривают различные модификации метода Гаусса, например, метод Гаусса *с постролбцовым 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

$$\left\{ \begin{array}{l} b_1x_1 + c_1x_2 = d_1, \\ a_2x_1 + b_2x_2 + c_2x_3 = d_2, \\ \qquad a_3x_2 + b_3x_3 + c_3x_4 = d_3, \\ \dots\dots\dots \\ \qquad \qquad a_{n-1}x_{n-2} + b_{n-1}x_{n-1} + c_{n-1}x_n = d_{n-1}, \\ \qquad \qquad \qquad a_nx_{n-1} + b_nx_n = d_n. \end{array} \right.$$

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:

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

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

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

Числа L_i и M_i называются *прогночными 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

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

Если выполняются неравенства

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

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

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

В *методе Якоби* используется следующий алгоритм построения

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

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 $x_1^{(k)}, x_2^{(k)}, \dots, x_{i-1}^{(k)}$, 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:

[illegible]

ИЛИ

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

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.

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

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).

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

$\text{Array}[f, \{n_1, n_2\}]$ – 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 .

$\text{Append}[lst, x]$ – создает новый список, добавляя элемент x в конец списка lst .

$\text{CharacteristicPolynomial}[A, \lambda]$ – возвращает характеристический многочлен матрицы A относительно переменной λ .

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

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

$\text{Dot}[v_1, v_2]$ или $v_1.v_2$ – вычисляет скалярное произведение векторов v_1 и v_2 .

$\text{Dot}[A, B]$ или $A.B$ – вычисляет произведение матриц A и B .

$\text{Det}[A]$ – 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⁾}, ..., {*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[$\{\{eqn_1, eqn_2, \dots\}, \{x_1, x_2, \dots\}\}$] – находит численное решение заданной системы уравнений.

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

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

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. Понижим 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 уравнения
```

```
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 $\|A\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$. 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]]], {j, 1, 2}], {i, 1, 2}
          |таблица|абсолютное значение
Out[16]= {11, 1101}

t2 = Table[Sum[Abs[A1[[i, j]]], {j, 1, 2}], {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]]], {j, 1, 2}]
           |таблица|abs[абсолютное значение]
```

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

```
Out[21]= {101, 1011}
```

```
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]= 1113111
```

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 $\delta_X = \text{cond}(A) \cdot \delta_B$, где $\delta_B = \|\Delta B\|/\|B + \Delta B\|$:

```
In[29]:= pr = condA1 *  $\frac{\text{Norm}[dB, 1]}{\text{Norm}[B + dB, 1]}$ 
```

```
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 $A = (a_{ij})$, $B = (b_i)$,

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

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[ $\frac{i}{2} (i - 35) + 80$ , {i, 4}];
    [таблица значений
```

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

```
Out[3]=  $\left\{ \begin{pmatrix} 3 & 1 & -1 & 60 \\ 2 & 0 & 45 & 0 \\ 1 & 30 & 0 & 1 \\ 15 & 0 & 1 & 2 \end{pmatrix}, \begin{pmatrix} 63 \\ 47 \\ 32 \\ 18 \end{pmatrix} \right\}$ 
```

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[8]:= 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 |y... |абсолютное значение
        If[Abs[A[[p, k]]] < $MachineEpsilon, Return[{"Решение не найдено", MatrixForm[A]}]];
        |y... |абсолютное значение |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

$$\text{Out[7]} = \left\{ \begin{pmatrix} 15 & 0 & 1 & 2 \\ 0 & 1 & -\frac{6}{5} & \frac{298}{5} \\ 0 & 0 & \frac{673}{15} & -\frac{4}{15} \\ 0 & 0 & 0 & -\frac{1202597}{673} \end{pmatrix}, \begin{pmatrix} 18 \\ \frac{297}{5} \\ \frac{223}{5} \\ -\frac{1202597}{673} \end{pmatrix} \right\}$$

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

$$X[[i]] = \frac{B[[i]] - \sum_{j=i+1}^n A[[i, j]] \times X[[j]]}{A[[i, i]]};$$

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 $(1, 1, 1, 1)$.