# Электромагнитная совместимость:

## ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ ЭЛЕКТРОСТАТИКИ

С.П. Куксенко

# 2 МЕТОДЫ РЕШЕНИЯ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ)

## 2.1 Общие сведения

### 2.1.1 Постановка задачи

В вычислительной линейной алгебре выделяют 4 основные задачи: решение СЛАУ; вычисление определителей; нахождение обратных матриц; определение собственных значений и собственных векторов. При этом решение СЛАУ называют первой основной задачей. Эффективность способа решения СЛАУ вида

$$\mathbf{Ax} = \mathbf{b} \tag{2.1}$$

во многом зависит от структуры матрицы **A**: размера, обусловленности, симметричности, заполненности (т.е. соотношения между числом ненулевых и нулевых элементов), специфики расположения ненулевых элементов в матрице и т. д. Будем полагать, что матрица **A** задана и является невырожденной. Известно, что в этом случае решение системы существует, единственно и устойчиво по входным данным, т. е. рассматриваемая задача корректна.

Пусть $\mathbf{x}^* = \left(x_1^*, x_2^*, ..., x_N^*\right)^T$ – приближенное решение СЛАУ ($T$ – символ транспонирования). Будем стремиться к получению решения, для которого погрешность $\mathbf{e} = \mathbf{x} - \mathbf{x}^*$ мала. Заметим, что качество полученного решения далеко не всегда характеризуется тем, насколько мала погрешность $\mathbf{x} - \mathbf{x}^*$. Иногда вполне удовлетворительным является критерий малости невязки $\mathbf{r} = \mathbf{b} - \mathbf{Ax}^*$. Так, вектор **r** показывает, насколько отличается правая часть системы от левой, если подставить в нее приближенное решение. Заметим также, что $\mathbf{r} = \mathbf{Ax} - \mathbf{Ax}^* = \mathbf{A}(\mathbf{x} - \mathbf{x}^*)$, поэтому погрешность и невязка связаны равенством

$$\mathbf{e} = \mathbf{x} - \mathbf{x}^* = \mathbf{A}^{-1}\mathbf{r}. \tag{2.2}$$

### 2.1.2 Нормы векторов

Решением СЛАУ является вектор $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$, который будем рассматривать как элемент векторного пространства $\boldsymbol{C}^N$. Приближенное решение $\mathbf{x}^* = \left(x_1^*, x_2^*, ..., x_N^*\right)^T$ и погрешность $\mathbf{e} = \mathbf{x} - \mathbf{x}^* = \left(x_1 - x_1^*, x_2 - x_2^*, ..., x_N - x_N^*\right)^T$ также являются элементами пространства $\boldsymbol{C}^N$. Для того чтобы анализировать методы решения СЛАУ, необходимо уметь количественно оценивать «величины» векторов $\mathbf{x}^*$ и $\mathbf{x} - \mathbf{x}^*$, а также векторов $\mathbf{b}$ и $\mathbf{b} - \mathbf{b}^*$, где $\mathbf{b}^* = \left(b_1^*, b_2^*, ..., b_N^*\right)^T$ – вектор приближенно заданных правых частей. Удобной для этой цели количественной характеристикой является широко используемое понятие нормы вектора.

Говорят, что в пространстве $\boldsymbol{C}^N$ задана норма, если каждому вектору $\mathbf{x}$ из $\boldsymbol{C}^N$ сопоставлено вещественное число $||\mathbf{x}||$, называемое нормой вектора $\mathbf{x}$ и обладающее следующими свойствами:

1) $||\mathbf{x}|| \geq 0$, причем $||\mathbf{x}|| = 0$ тогда и только тогда, когда $\mathbf{x} = 0$;

2) $||\alpha\mathbf{x}|| = |\alpha| \, ||\mathbf{x}||$ для любого вектора $\mathbf{x}$ и любого числа $\alpha$;

3) $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$ для любых векторов $\mathbf{x}$ и $\mathbf{y}$.

Заметим, что такими же свойствами обладает обычная геометрическая длина вектора в трехмерном пространстве. Свойство 3 в этом случае следует из правила сложения векторов и из того известного факта, что сумма длин двух сторон треугольника всегда больше длины третьей стороны.

Существует множество различных способов введения норм. В вычислительных методах наиболее употребительными являются следующие три нормы:

$\left\|\mathbf{x}\right\|_1 = \sum_{i=1}^{N} |\, x_i \,|$ – первая (манхэттенская, октоэдрическая) норма;

$\left\|\mathbf{x}\right\|_2 = \left( \sum_{i=1}^{N} |\, x_i \,|^2 \right)^{1/2}$ – вторая (евклидова норма, сферическая) норма;

$\|\mathbf{x}\|_\infty = \max\limits_{1\le i\le N} |x_i|$ – бесконечная (равномерная, чебышевская, кубическая) норма.

В определенном смысле эти нормы эквивалентны, так как каждая из них оценивается любой из двух других с точностью до множителя.

**Пример 2.1**

Найти нормы $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_2$, $\|\mathbf{x}\|_\infty$ для вектора $\mathbf{x} = (0.12, -0.15, 0.16)^T$.

*Решение*

По приведенным выше формулам имеем

$$\|\mathbf{x}\|_1 = 0.12 + 0.15 + 0.16 = 0.43,$$
$$\|\mathbf{x}\|_2 = (0.12^2 + 0.15^2 + 0.16^2)^{1/2} = 0.25,$$
$$\|\mathbf{x}\|_\infty = \max\{0.12, 0.15, 0.16\} = 0.16.$$

### 2.1.3 Скалярное произведение векторов

Скалярным произведением векторов $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ и $\mathbf{y} = (y_1, y_2, \ldots, y_N)^T$ называется величина $\mathbf{x}^T\mathbf{y}$ или

$$(\mathbf{x}, \mathbf{y}) = x_1 y_1 + \ldots + x_N y_N = \sum_{i=1}^{N} x_i y_i.$$

Нетрудно заметить, что $\|\mathbf{x}\|_2 = (\mathbf{x}, \mathbf{x})^{1/2}$. Когда векторы $\mathbf{x}$ и $\mathbf{y}$ имеют комплексные компоненты, скалярное произведение записывается в виде

$$(\mathbf{x}, \mathbf{y}) = x_1 \overline{y}_1 + \ldots + x_N \overline{y}_N = \sum_{i=1}^{N} x_i \overline{y}_i,$$

где черта означает комплексное сопряжение.

### 2.1.4 Абсолютная и относительная погрешности векторов

Далее будем всюду считать, что в пространстве $N$-мерных векторов $\mathbf{C}^N$ введена и фиксирована некоторая норма $\|\mathbf{x}\|$. В этом случае в качестве меры степени близости векторов $\mathbf{x}$ и $\mathbf{x}^*$ естественно использовать величину $\|\mathbf{x} - \mathbf{x}^*\|$, являющуюся аналогом

расстояния между точками $\mathbf{x}$ и $\mathbf{x}^*$. Введем абсолютную и относительную погрешности вектора $\mathbf{x}^*$ с помощью формул

$$\Delta\left(\mathbf{x}^*\right) = \left\| \mathbf{x} - \mathbf{x}^* \right\|,$$

$$\delta\left(\mathbf{x}^*\right) = \left\| \mathbf{x} - \mathbf{x}^* \right\| \Big/ \left\| \mathbf{x} \right\|.$$

Выбор той или иной нормы в практических задачах диктуется тем, какие требования предъявляются к точности решения. Выбор нормы $\|\mathbf{x}\|_1$ фактически отвечает случаю, когда малой должна быть суммарная абсолютная погрешность в компонентах решения; выбор нормы $\|\mathbf{x}\|_2$ соответствует критерию малости среднеквадратичной погрешности, а принятие в качестве нормы $\|\mathbf{x}\|_\infty$ означает, что малой должна быть максимальная из абсолютных погрешностей в компонентах решения.

### 2.1.5 Сходимость по норме

Пусть $\mathbf{x}^{(m)}$ $(m = 1, 2, \ldots, \infty)$ – последовательность векторов $\mathbf{x}^{(m)} = (x_1^{(m)}, x_2^{(m)}, \ldots, x_N^{(m)})^T$. Говорят, что последовательность векторов $\mathbf{x}^{(m)}$ сходится к вектору $\mathbf{x}$ при $m \to \infty$ ($\mathbf{x}^{(m)} \to \mathbf{x}$ при $m \to \infty$), если $\Delta(\mathbf{x}^{(m)}) = \|\mathbf{x}^{(m)} - \mathbf{x}\| \to 0$ при $m \to \infty$.

Сам факт наличия или отсутствия сходимости $\mathbf{x}^{(m)}$ к $\mathbf{x}$ при $m \to \infty$ в конечномерных пространствах не зависит от выбора нормы. Известно, что из сходимости последовательности по одной из норм следует сходимость этой последовательности по любой другой норме. Более того, $\mathbf{x}^{(m)} \to \mathbf{x}$ при $m \to \infty$ тогда и только тогда, когда для всех $i = 1, 2, \ldots, N$ имеем $\mathbf{x}_i^{(m)} \to \mathbf{x}_i$ при $m \to \infty$, т. е. сходимость по норме в пространстве $\boldsymbol{C}^N$ эквивалентна покомпонентной (покоординатной) сходимости.

### 2.1.6 Нормы матриц

Величина

$$\left\|\mathbf{A}\right\| = \max_{\mathbf{x}\neq 0} \frac{\left\| \mathbf{A}\mathbf{x} \right\|}{\left\| \mathbf{x} \right\|} \tag{2.3}$$

называется нормой матицы $\mathbf{A}$, подчиненной норме векторов, в $\boldsymbol{C}^N$. Понятие матрицы ввел Джеймс Джозеф Сильвестр в 1850 г.

Заметим, что множество всех квадратных матриц размером $N \times N$ является векторным пространством. Можно показать, что введенная в этом пространстве норма обладает следующими свойствами, аналогичными свойствам нормы вектора:

1) $||\mathbf{A}|| \geq 0$, причем $||\mathbf{A}|| = 0$ тогда и только тогда, когда $\mathbf{A} = 0$;

2) $||\alpha\mathbf{A}|| = |\alpha| \, ||\mathbf{A}||$ для любой матрицы $\mathbf{A}$ и любого числа $\alpha$;

3) $||\mathbf{A} + \mathbf{B}|| \leq ||\mathbf{A}|| + ||\mathbf{B}||$ для любых матриц $\mathbf{A}$ и $\mathbf{B}$.

Дополнительно к этому верны следующие свойства:

4) $||\mathbf{A} \cdot \mathbf{B}|| \leq ||\mathbf{A}|| \cdot ||\mathbf{B}||$ для любых матриц $\mathbf{A}$ и $\mathbf{B}$;

5) для любой матрицы $\mathbf{A}$ и любого вектора $\mathbf{x}$ справедливо неравенство

$$||\mathbf{Ax}|| \leq ||\mathbf{A}|| \cdot ||\mathbf{x}||. \tag{2.4}$$

Как следует из определения (2.3), каждой из векторных норм $||\mathbf{x}||$ соответствует своя подчиненная норма матрицы $\mathbf{A}$. Известно, в частности, что нормам $||\mathbf{x}||_1$, $||\mathbf{x}||_2$ и $||\mathbf{x}||_\infty$ подчинены нормы $||\mathbf{A}||_1$, $||\mathbf{A}||_2$ и $||\mathbf{A}||_\infty$, вычисляемые по формулам

$$\| \mathbf{A} \|_1 = \max_{1 \leq j \leq N} \sum_{i=1}^{N} | a_{ij} |, \tag{2.5}$$

$$\| \mathbf{A} \|_2 = \max_{1 \leq j \leq N} \sqrt{\lambda_j \left( \mathbf{A}^T \mathbf{A} \right)}, \tag{2.6}$$

где $\lambda_j(\mathbf{A}^T\mathbf{A})$ – собственные числа матрицы $\mathbf{A}^T\mathbf{A}$;

$$\| \mathbf{A} \|_\infty = \max_{1 \leq i \leq N} \sum_{j=1}^{N} | a_{ij} |. \tag{2.7}$$

Нормы $||\mathbf{A}||_1$ и $||\mathbf{A}||_\infty$ вычисляются просто. Для вычисления $||\mathbf{A}||_1$ нужно найти сумму модулей элементов каждого из столбцов матрицы $\mathbf{A}$, а затем выбрать максимальную из этих сумм. Для получения значения $||\mathbf{A}||_\infty$ нужно аналогичным образом поступить со строками матрицы $\mathbf{A}$. Как правило, вычислить значение нормы $||\mathbf{A}||_2$ бывает трудно, так как для этого следует искать собственные числа $\lambda_j$. Для оценки величины $||\mathbf{A}||_2$ можно, например, использовать неравенство

$$\| \mathbf{A} \|_2 \leq \| \mathbf{A} \|_E,$$

где $\|\mathbf{A}\|_E = \left( \sum_{i,j=1}^{N} | a_{ij} |^2 \right)^{1/2}$ – величина, называемая евклидовой нормой матрицы **A**. Ее также называют нормой Фробениуса.

**Пример 2.2**

Найти нормы $||\mathbf{A}||_1$, $||\mathbf{A}||_2$, $||\mathbf{A}||_\infty$ для матрицы $\mathbf{A} = \begin{pmatrix} 0.1 & -0.4 & 0 \\ 0.2 & 1 & -0.3 \\ 0 & 0.1 & 0.3 \end{pmatrix}$.

*Решение*

По приведенным выше формулам имеем

$$\|\mathbf{A}\|_1 = \max\{0.1 + 0.2 + 0;\ 0.4 + 1 + 0.1;\ 0 + 0.3 + 0.3\} = 1.5;$$

$$\|\mathbf{A}\|_\infty = \max\{0.1 + 0.4 + 0;\ 0.2 + 1 + 0.3;\ 0 + 0.1 + 0.3\} = 1.5;$$

$$\|\mathbf{A}\|_2 \le \|\mathbf{A}\|_E = \left( \sum_{i,j=1}^{3} | a_{ij} |^2 \right)^{1/2} \approx 1.18.$$

### 2.1.7 Обусловленность задачи решения СЛАУ

Известно, что решения различных СЛАУ обладают разной чувствительностью к погрешностям входных данных. Так, задача вычисления решения **x** уравнений **Ax** = **b** может быть как хорошо, так и плохо обусловленной.

Пусть элементы матрицы **A** из (2.1) считаются заданными точно, а правая часть – приближенно. Тогда для погрешности приближенного решения системы справедлива оценка

$$\Delta(\mathbf{x}^*) \le ||\mathbf{A}^{-1}|| \cdot ||\mathbf{r}||, \qquad (2.8)$$

где $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}^*$ – невязка, отвечающая $\mathbf{x}^*$. Для доказательства достаточно взять норму левой и правой частей равенства (2.2) и воспользоваться свойством (2.4).

Пусть $\mathbf{x}^*$ – точное решение системы $\mathbf{A}\mathbf{x}^* = \mathbf{b}^*$, в которой правая часть $\mathbf{b}^*$ является приближением к **b**. Тогда верны следующие оценки абсолютной и относительной погрешностей:

$$\Delta(\mathbf{x}^*) \le \nu_\Delta \Delta(\mathbf{b}^*); \qquad (2.9)$$

$$\delta(\mathbf{x}^*) \le \nu_\delta \delta(\mathbf{b}^*), \qquad (2.10)$$

где $\nu_\Delta = \|\mathbf{A}^{-1}\|$, $\nu_\delta(\mathbf{x}) = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{r}\| / \|\mathbf{x}\| = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{Ax}\| / \|\mathbf{x}\|$.

В рассматриваемом случае $\mathbf{r} = \mathbf{b} - \mathbf{Ax}^* = \mathbf{b} - \mathbf{b}^*$ и оценка (2.8) принимает вид (2.9). Разделив обе части этого неравенства на $\|\mathbf{x}\|$ и записав его в виде

$$\Delta(\mathbf{x}^*) / \|\mathbf{x}\| \le (\|\mathbf{A}^{-1}\| \cdot \|\mathbf{b}\| / \|\mathbf{x}\|) \cdot (\Delta(\mathbf{x}^*) / \|\mathbf{b}\|),$$

приходим к оценке (2.10).

Сделаем несколько замечаний относительно числа обусловленности СЛАУ.

– Величина $\nu_\Delta = \|\mathbf{A}^{-1}\|$ для задачи $\mathbf{Ax} = \mathbf{b}$ играет роль абсолютного числа обусловленности.

– Величина $\nu_\delta(\mathbf{x}) = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{r}\| / \|\mathbf{x}\|$ называется естественным числом обусловленности. Она зависит от конкретного решения $\mathbf{x}$ и характеризует коэффициент возможного возрастания относительной погрешности этого решения, вызванного погрешностью задания правой части. Это означает, что $\nu_\delta(\mathbf{x})$ для задачи решения системы $\mathbf{Ax} = \mathbf{b}$ играет роль относительного числа обусловленности.

– Полученные оценки (2.9) и (2.10) точны в том смысле, что для системы $\mathbf{Ax} = \mathbf{b}$ с произвольной невырожденной правой частью $\mathbf{b} \neq 0$ найдется сколь угодно близкий к $\mathbf{b}$ приближенно заданный вектор $\mathbf{b}^* \neq \mathbf{b}$, для которого эти неравенства превращаются в равенства.

Вычислим максимальное значение естественного числа обусловленности, используя определение нормы матрицы:

$$\max_{\mathbf{x}\neq 0} \nu_\delta(\mathbf{x}) = \frac{\| \mathbf{A}^{-1} \| \cdot \| \mathbf{Ax} \|}{\| \mathbf{x} \|} = \| \mathbf{A}^{-1} \| \cdot \| \mathbf{A} \| . \qquad (2.11)$$

Полученную величину называют числом обусловленности матрицы $\mathbf{A}$ и обозначают $\mathrm{cond}(\mathbf{A})$. Таким образом,

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\|. \qquad (2.12)$$

Отметим, что впервые этот термин предложен А. Тьюрингом в 1948 г.

Следствием оценок (2.9) и (2.10) является оценка

$$\delta(\mathbf{x}^*) \le \mathrm{cond}(\mathbf{A}) \cdot \delta(\mathbf{b}^*). \qquad (2.13)$$

Оценка (2.13) точна в том смысле, что для системы $\mathbf{Ax} = \mathbf{b}$ с произвольной невырожденной матрицей $\mathbf{A}$ найдутся правая часть $\mathbf{b} \neq 0$ (и отвечающее этой правой части решение $\mathbf{x}$) и сколь угодно близкий к $\mathbf{b}$ приближенно заданный вектор $\mathbf{b}^* \neq \mathbf{b}$ такие, что это неравенство превращается в равенство.

Величина $\mathrm{cond}(\mathbf{A})$ является широко используемой количественной мерой обусловленности системы $\mathbf{Ax} = \mathbf{b}$. В частности, систему и матрицу $\mathbf{A}$ принято называть плохо обусловленными, если $\mathrm{cond}(\mathbf{A}) >> 1$. В силу последнего замечания и оценки (2.13) для такой системы существуют решения, обладающие чрезвычайно высокой чувствительностью к малым погрешностям задания входного данного $\mathbf{b}$. Тем не менее заметим, что для всякого решения $\mathbf{x}$ коэффициент $\nu_\delta(\mathbf{x})$ роста относительной погрешности достигает значений, близких к максимально возможному значению $\mathrm{cond}(\mathbf{A})$.

Приведем часто используемые, свойства числа обусловленности.

– Для единичной матрицы $\mathrm{cond}(\mathbf{E}) = 1$.

– Справедливо неравенство $\mathrm{cond}(\mathbf{A}) \geq 1$. Так, из равенства $\mathbf{E} = \mathbf{A}\mathbf{A}^{-1}$, свойства 4 норм матриц и равенства $||\mathbf{E}|| = 1$ следует, что $1 = ||\mathbf{E}|| \leq ||\mathbf{A}^{-1}|| \cdot ||\mathbf{A}|| = \mathrm{cond}(\mathbf{A})$.

– Число обусловленности матрицы $\mathbf{A}$ не меняется при умножении матрицы на произвольное число $\alpha \neq 0$. Заметим, что $(\alpha\mathbf{A})^{-1} = \alpha^{-1}\mathbf{A}^{-1}$. Поэтому $\mathrm{cond}(\alpha\mathbf{A}) = ||\alpha\mathbf{A}|| \cdot ||(\alpha\mathbf{A})^{-1}|| = |\alpha| \cdot ||\mathbf{A}|| \cdot |\alpha|^{-1} \cdot ||\mathbf{A}^{-1}|| = \mathrm{cond}(\mathbf{A})$.

Величина $\mathrm{cond}(\mathbf{A})$ зависит, вообще говоря, от выбора нормы векторов в пространстве $\boldsymbol{C}^N$. Фактически, это есть зависимость максимального коэффициента роста погрешности от способа измерения величины входных данных и решения.

**Пример 2.3**

Найти $\mathrm{cond}_\infty(\mathbf{A})$ для матрицы $\mathbf{A} = \begin{pmatrix} 1.03 & 0.991 \\ 0.991 & 0.943 \end{pmatrix}$.

*Решение*

Сначала найдем обратную матрицу: $\mathbf{A}^{-1} \approx \begin{pmatrix} -87.4 & 91.8 \\ 91.8 & -95.4 \end{pmatrix}$.

Тогда $\mathrm{cond}_\infty(\mathbf{A}) = \|\mathbf{A}\|_\infty \cdot \|\mathbf{A}^{-1}\|_\infty \approx 2.021 \cdot 187.2 \approx 378$. Если входные данные для СЛАУ с матрицей **A** содержат относительную погрешность порядка 0,1–1 %, то систему можно расценивать как плохо обусловленную.

**Пример 2.4**

Рассмотрим систему уравнений

$$1.03x_1 + 0.991x_2 = 2.51,$$
$$0.991x_1 + 0.943x_2 = 2.41,$$

с матрицей из примера 2.3. Ее решением является $x_1 = 1.981$, $x_2 = 0.4735$. Правая часть системы известна с точностью до 0.005, если считать, что числа 2.51 и 2.41 получены округлением «истинных» значений при вводе в память трехзначного десятичного компьютера. Как влияет погрешность во входных данных такого уровня на погрешность решения?

*Решение*

Возмутим каждый из компонентов вектора $\mathbf{b} = (2.51, 2.41)^T$ на 0.005, взяв $\mathbf{b}^* = (2.505, 2.415)^T$. Решением системы, отвечающим $\mathbf{b}^*$, является теперь $x_1^* = 2.877$, $x_2^* = -0.4629$. Таким образом, решение оказалось полностью искаженным. Относительная погрешность правой части $\delta(\mathbf{b}^*) = \|\mathbf{b} - \mathbf{b}^*\|_\infty / \|\mathbf{b}\|_\infty = 0.005 / 2.51 \approx 0.2$ % привела к относительной погрешности решения $\delta(\mathbf{x}^*) = \|\mathbf{x} - \mathbf{x}^*\|_\infty / \|\mathbf{x}\|_\infty \approx 0.9364 / 1.981 \approx 47.3$ %. Следовательно, погрешность возросла примерно в 237 раз.

Можно ли внести в правую часть системы такую погрешность, чтобы получить существенно большее, чем 237, значение коэффициента роста погрешности? Вычислим естественное число обусловленности, являющееся максимальным значением рассматриваемого коэффициента, отвечающим решению $x_1 = 1.981$, $x_2 = 0.4735$, и получим

$$\nu_\delta(\mathbf{x}) = \|\mathbf{A}^{-1}\|_\infty \cdot \|\mathbf{b}\|_\infty / \|\mathbf{x}\|_\infty \approx 187.2 \cdot 2.51 / 1.981 \approx 237.$$

Таким образом, на поставленный вопрос следует ответить отрицательно.

### 2.1.8 Масштабирование

Перед началом решения СЛАУ целесообразно масштабировать систему так, чтобы ее коэффициенты были величинами

порядка единицы. Существуют два естественных способа масштабирования системы (2.1). Первый заключается в умножении каждого из уравнений на некоторый масштабирующий множитель $t_i$. Второй состоит в умножении на масштабирующий множитель $\alpha_j$ каждого $j$-го столбца матрицы, что соответствует замене переменных $x'_j = \alpha_j^{-1} x_j$ (фактически это замена единиц измерения). В реальных ситуациях чаще всего масштабирование может быть выполнено без существенных трудностей. Однако подчеркнем, что в общем случае удовлетворительного способа масштабирования пока не найдено. На практике масштабирование обычно производят с помощью деления каждого уравнения на его наибольший по модулю коэффициент. Это вполне удовлетворительный способ для большинства реально встречающихся задач.

### 2.1.9 Форматы хранения матриц

Плотной (полной) считается матрица размером $N \times M$, содержащая $NM$ ненулевых элементов. На практике если в матрице очень мало нулевых элементов по сравнению с их общим количеством, то она тоже считается плотной. Для хранения плотных матриц используются стандартные двухмерные массивы.

Термин «разреженная матрица» имеет достаточно много определений. Наиболее употребительным сейчас является следующее. Разреженной является матрица, для которой существуют специальные приемы, позволяющие извлечь выгоды из большого числа ее нулевых элементов и их расположения. Упомянутая выгода, как правило, заключается в уменьшении требуемой компьютерной памяти (для хранения матрицы по сравнению со стандартным хранением плотных матриц) и количества математических операций с этой матрицей. Исторически извлечение выгоды из наличия разреженности было направлено на решение СЛАУ. Так, в 1950-х гг. началось широкое использование разреженности при решении СЛАУ.

К настоящему времени разработано достаточно большое количество форматов (схем) хранения общих разреженных матриц, например из двух векторов, Кнута, Рейнболдта и Местеньи, Лар-

кума, Шермана, разреженные строчный (CSR) и столбцовый (CSC), разреженный неравномерный диагональный, разреженный блочный строчный, координатный, модифицированный разреженный строчный и др. Также для хранения ленточных матриц разработан сжатый диагональный формат, а для симметричных – симметричный скайлайн-формат. Они отличаются степенью сжатия матрицы и требуемыми затратами машинного времени при их использовании. Для каждого класса задач целесообразен выбор оптимального формата, обеспечивающего экономию машинной памяти. При этом стоит помнить, что увеличение коэффициента сжатия, как правило, ведет к увеличению затрат времени.

Пожалуй, самыми распространенными на данный момент являются форматы CSR, CSC и координатный. Например, в GNU Octave (далее Octave), используется формат CSC, также известный как формат Harwell-Boeing. Указанные форматы предъявляют минимальные требования к памяти и эффективны для реализации базовых операций с разреженными матрицами при решении СЛАУ как прямыми, так и итерационными методами и т. д. Так, в формате CSR используются 3 массива (**aelem** – ненулевые элементы, **jptr** – индексы столбцов, **iptr** – указатели на ненулевые элементы, с которых начинается очередная строка). Формат CSC реализует хранение элементов матрицы по столбцам. При этом также используются 3 массива. Координатный формат использует также 3 массива: для хранения значений матричных элементов (**AA**), их индексов строк (**JR**) и столбцов (**JC**). Достоинством координатного формата является то, что данные, хранящиеся с его помощью, могут быть легко преобразованы в другие форматы. Поэтому он часто используется как стандартный входной формат в пакетах прикладных программ.

### 2.1.10 Методы решения СЛАУ

Все методы решения СЛАУ можно разбить на два класса: прямые и итерационные. Прямыми называются методы, которые приводят к решению за конечное число арифметических операций. Если операции реализуются точно, то и решение будет точным (в связи с чем к классу прямых методов применяют еще

название «точные методы»). Итерационными являются методы, в которых точное решение может быть получено лишь в результате бесконечного повторения, как правило, единообразных действий.

Уделим основное внимание задаче вычисления вектора **x**, являющегося решением СЛАУ (2.1). Будем полагать, что матрица **A** задана и является невырожденной ($\det\mathbf{A} \neq 0$). Известно, что в этом случае решение системы существует, единственно и устойчиво по входным данным, т. е. рассматриваемая задача корректна.

На точность и время решения СЛАУ прямым методам сильное влияние оказывает обусловленность матрицы. Чем выше число обусловленности матрицы, тем хуже она обусловлена. С ростом числа обусловленности растет погрешность решения из-за представления чисел с плавающей запятой конечным числом разрядов. Одним из важных следствий этого является невозможность получения корректного решения СЛАУ методом Гаусса при чрезмерном росте $\mathrm{cond}(\mathbf{A})$ матриц больших порядков и малой разрядности представления чисел.

## 2.2 Прямые методы решения СЛАУ

### 2.2.1 Метод исключения Гаусса

Задача численного решения СЛАУ вида (2.1) имеет продолжительную историю. Так, до н.э. в Китае были изданы «Девять книг о математическом искусстве», где метод, который теперь принято называть методом исключения Гаусса или просто методом Гаусса, был представлен в «натуральной» форме. Метод имеет долгую и интересную историю, а к его становлению приложило немало усилий большое количество ученых и инженеров. Этот метод известен в различных вариантах, которые алгебраически тождественны, уже более 2000 лет. Варианты отличаются характером хранения матриц, порядком исключения, способами предупреждения больших погрешностей округления и тем, как уточняются вычисленные решения. Имеются также варианты, специально приспособленные для систем с симметричными положительно определенными матрицами, которые хранятся в при-

мерно вдвое меньшем объеме. Вычисления с помощью метода Гаусса включают два основных этапа, называемых прямым и обратным ходом. Прямой ход метода Гаусса состоит в последовательном исключении неизвестных из системы (2.1) для преобразования ее к эквивалентной системе с верхней треугольной матрицей. Вычисление значений неизвестных производится на этапе обратного хода [17].

На практике широкое распространение получила версия метода, основанная на LU-разложении матрица **A**. Рассмотрим ее подробнее.

Пусть $\mathbf{A} = \left(a_{ij}\right)_{i,j=1}^{N}$ – данная $N \times N$-матрица, а $\mathbf{L} = \left(l_{ij}\right)_{i,j=1}^{N}$ и $\mathbf{U} = \left(u_{ij}\right)_{i,j=1}^{N}$ – соответственно нижняя (левая) и верхняя (правая) треугольные матрицы. Справедливо следующее утверждение. Если все главные миноры квадратной матрицы **A** отличны от нуля, то существуют такие нижняя **L** и верхняя **U** треугольные матрицы, что $\mathbf{A} = \mathbf{LU}$ (главными минорами матрицы $\mathbf{A} = \left(a_{ij}\right)_{i,j=1}^{N}$ называются определители подматриц $\mathbf{A}_k = \left(a_{ij}\right)_{i,j=1}^{k}$, где $k = 1, 2, \ldots, N-1$). Если элементы диагонали одной из матриц, **L** или **U**, фиксированы (ненулевые), то такое разложение единственно.

Реализация LU-разложения с фиксированием диагонали верхней треугольной матрицы ($u_{ij} = 1$ при $i = j$) называется методом Краута. Рассмотрим часто используемое на практике разложение матриц при фиксировании диагонали нижней треугольной матрицы ($l_{ij} = 1$ при $i = j$) – метод Дулитла. Находят $l_{ij}$ при $i > j$ ($l_{ij} = 0$ при $i < j$) и $u_{ij}$ при $i \le j$ ($u_{ij} = 0$ при $i > j$) такие, чтобы

$$\begin{pmatrix} 1 & 0 & \ldots & 0 \\ l_{21} & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ l_{N1} & l_{N1} & \ldots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \ldots & u_{1N} \\ 0 & u_{22} & \ldots & u_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & u_{NN} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1N} \\ 0 & a_{22} & \ldots & a_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & a_{NN} \end{pmatrix}.$$

Выполнив перемножение матриц, на основе поэлементного приравнивания левых и правых частей приходим к $N \times N$-матрице уравнений

$$\begin{array}{llll} u_{11} = a_{11}, & u_{12} = a_{12}, & \ldots & u_{1N} = a_{1N}, \\ l_{21}u_{11} = a_{21}, & l_{21}u_{12} + u_{22} = a_{22}, & \ldots & l_{21}u_{1N} + u_{2N} = a_{2N}, \\ \ldots & \ldots & \ldots & \ldots \\ l_{N1}u_{11} = a_{N1}, & l_{N1}u_{12} + l_{N2}u_{22} = a_{N2}, & \ldots & l_{N1}u_{1N} + \ldots + u_{NN} = a_{NN}, \end{array}$$

относительно $N \times N$-матрицы неизвестных

$$\begin{array}{llll} u_{11}, & u_{12}, & \ldots & u_{1N}, \\ l_{21}, & lu_{22}, & \ldots & u_{2N}, \\ \ldots & \ldots & \ldots & \ldots \\ l_{N1}, & l_{N2}, & \ldots & u_{NN}. \end{array} \qquad (2.14)$$

Видно, что все отличные от 0 и 1 элементы матриц **L** и **U** могут быть однозначно вычислены с помощью всего двух формул:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}, \text{ где } i \le j, \qquad (2.15)$$

$$l_{ij} = \frac{1}{u_{jj}}\left( a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \right), \text{ где } i > j. \qquad (2.16)$$

Из приведенных преобразований следует, что реализовать LU-разложение по данным формулам можно различными методами, например построчным вычислением, т. е. пока не вычислена $i$-я строка матриц **L** и **U**, алгоритм не модернизирует $(i + 1)$-ю строку.

При практическом выполнении разложения (факторизации) матрицы **A** нужно иметь в виду следующие два обстоятельства.

Во-первых, организация вычислений по формулам (2.15)–(2.16) должна предусматривать переключение счета с одной формулы на другую. Это удобно делать, ориентируясь на матрицу неизвестных (2.14) (ее, кстати, можно интерпретировать как $N^2$-мерный массив для компактного хранения LU-разложения в памяти компьютера), а именно: первая строка матрицы (2.14) вычисляется по формуле (2.15) при $i = 1$, $j = 1, 2, \ldots, N$; первый столбец

(2.14) (без первого элемента) – по формуле (2.16) при $j = 1$, $i = 2,\ldots, N$, и т. д.

Во-вторых, препятствием для осуществимости описанного процесса LU-разложения матрицы **A** может оказаться равенство нулю диагональных элементов матрицы **U**, поскольку на них выполняется деление в формуле (2.16). Отсюда следует требование, накладываемое на главные миноры. Для определенных классов матриц требования о разложении заведомо выполняются. Это относится, например, к матрицам с преобладанием диагональных элементов.

Далее приведена построчная схема LU-разложения (так называемая *ikj*-версия) без выбора ведущего элемента, которая является, пожалуй, самой предпочтительной для программной реализации. Можно показать, что перестановка трех циклов дает шесть возможных версий разложения.

**Алгоритм *ikj*-версии LU-разложения без выбора ведущего элемента (результат хранится на месте исходной матрицы)**

Для $i = 2, \ldots, N$
    Для $k = 1, \ldots, i - 1$
        $a_{ik} = a_{ik} / a_{kk}$
        Для $j = k + 1, \ldots, N$
            $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$
        Увеличить $j$
    Увеличить $k$
Увеличить $i$

Ниже приведены функции на языке Octave, реализующие разложение по этому алгоритму (листинг 2.1). Общие сведения по разработке программ на языке Octave изложены в приложении А.

```
function A = ikj (A)
 n = size(A,1) ;
for i=1:n
  for k=1:i-1
    A(i,k) = A(i,k)/A(k,k);
    A(i,k+1:n) = A(i,k+1:n) - A(i,k)*A(k,k+1:n);
  end
end
%L = diag(ones(n,1)) + tril(A,-1);
%U = triu(A);
```

Листинг 2.1 – Функция LU-разложения на языке Octave

Этот алгоритм позволяет пересчитать $i$-ю строку матрицы **A** в $i$-е строки матриц **L** и **U**. Каждая из строк 1, 2,…, $j-1$ участвует в определении $j$-х строк матриц **L** и **U**, но сами они больше не модернизируются.

Вообще, существует более десяти вариантов LU-разложения. Иногда данное представление матрицы в виде произведения матриц называют LR-разложением.

**Пример 2.5**

Найти LU-разложение для матрицы $\mathbf{A} = \begin{pmatrix} 10 & 6 & 2 & 0 \\ 5 & 1 & -2 & 4 \\ 3 & 5 & 1 & 4 \\ 0 & 6 & -2 & 2 \end{pmatrix}$.

*Решение*

Воспользовавшись формулами (2.15)–(2.16), получим

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.3 & -1.6 & 1 & 0 \\ 0 & -3 & 2.5 & 1 \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} 10 & 6 & 2 & 0 \\ 0 & -2 & -3 & 4 \\ 0 & 0 & -4.4 & 5.4 \\ 0 & 0 & 0 & 0.5 \end{pmatrix}.$$

Если матрица **A** исходной системы **Ax** = **b** разложена в произведение треугольных матриц **L** и **U**, то вместо **Ax** = **b** можно записать

$$\mathbf{LUx} = \mathbf{b}.$$

Введя вектор вспомогательных переменных **y**, последнее выражение можно переписать в виде системы

$$\begin{cases} \mathbf{Ly} = \mathbf{b}, \\ \mathbf{Ux} = \mathbf{y}. \end{cases}$$

Таким образом, решение данной системы с квадратной матрицей коэффициентов свелось к последовательному решению двух систем с треугольными матрицами коэффициентов.

Очевидно, все $y_i$ могут быть найдены из системы **Ly** = **b** при $i = 1, 2, \ldots, N$ по формуле (прямой ход)

$$y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k. \qquad (2.17)$$

Значения неизвестных $x_i$ находятся из системы $\mathbf{Ux} = \mathbf{y}$ в обратном порядке, т.е. при $i = N, N-1,\ldots, 1$, по формуле (обратный ход)

$$x_i = \frac{1}{u_{ii}}\left( y_i - \sum_{k=i+1}^{N} u_{ik} x_k \right). \qquad (2.18)$$

Итак, решение СЛАУ посредством LU-факторизации сводится к организации вычислений по четырем формулам: совокупности формул (2.15), (2.16) для получения матрицы $\mathbf{L} + \mathbf{U} - \mathbf{I}$ (2.14) ненулевых и неединичных элементов матриц для $\mathbf{L}$ и $\mathbf{U}$; формулы (2.17) для получения вектора свободных членов треугольной системы $\mathbf{Ux} = \mathbf{y}$; формулы (2.18), генерирующей решение исходной системы $\mathbf{Ax} = \mathbf{b}$.

Для обращения матрицы $\mathbf{A}$ с помощью LU-разложения можно $N$-кратно использовать формулы (2.17) и (2.18) для получения столбцов матрицы $\mathbf{A}^{-1}$; при этом в качестве $b_i$ в формуле (2.17) должны фигурировать только 0 и 1: для нахождения первого столбца полагают $b_1 = 1$, $b_2 = 0$, $b_3 = 0$, …, $b_N = 0$; для второго – $b_1 = 0$, $b_2 = 1$, $b_3 = 0$, …, $b_N = 0$ и т. д.

В результате после $N$ шагов на месте исходной матрицы $\mathbf{A}$ находится ее обратная матрица $\mathbf{A}^{-1}$.

### 2.2.2 Метод прогонки

Метод прогонки предложен в начале 1950-х гг. независимо несколькими авторами, в том числе советскими. Это эффективный алгоритм решения СЛАУ с трехдиагональными матрицами вида

$$\begin{aligned}
b_1x_1 &+ c_1x_2 &&&&&&&= d_1,\\
a_2x_1 &+ b_2x_2 &+ c_2x_3 &&&&&&= d_2,\\
&\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots &&&&&&&\\
&a_ix_{i-1} + b_ix_i + c_ix_{i+1} &&&&&&= d_i,\\
&\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots &&&&&&&\\
&a_{N-1}x_{N-2} + b_{N-1}x_{N-1} + c_{N-1}x_N &&&&&&= d_{N-1},\\
&a_Nx_{N-1} + b_Nx_N &&&&&&= d_N.
\end{aligned} \qquad (2.19)$$

Системы такого вида часто возникают при решении различных задач математической физики.

Преобразуем первое уравнение (2.19) к виду

$$x_1 = \alpha_1 x_2 + \beta_1, \qquad (2.20)$$

где $\alpha_1 = -c_1 / b_1$; $\beta_1 = d_1 / b_1$.

Подставим выражение для $x_1$ во второе уравнение системы:

$$a_2(\alpha_1 x_2 + \beta_1) + b_2 x_2 + c_2 x_3 = d_2.$$

Преобразуем это уравнение к виду

$$x_2 = \alpha_2 x_3 + \beta_2, \qquad (2.21)$$

где $\alpha_2 = -c_2 / (b_2 + a_2\alpha_1)$; $\beta_2 = (d_2 - a_2\beta_1) / (b_2 + a_2\alpha_1)$. Это выражение подставим в третье уравнение системы и т. д. На $i$-м шаге ($1 < i < N$) $i$-е уравнение системы преобразуется к виду

$$x_i = \alpha_i x_i + 1 + \beta i, \qquad (2.22)$$

где $\alpha_i = -c_i / (b_i + a_i\alpha_{i-1})$; $\beta_i = (d_i - a_i\beta_{i-1}) / (b_i + a_i\alpha_{i-1})$.

На $N$-м шаге подстановка в последнее уравнение выражения $x_{N-1} = \alpha_N x_{N+1} + \beta_{N-1}$ дает

$$a_N(\alpha_{N-1} x_N + \beta_{N-1}) + b_N x_N = d_N,$$

откуда можно определить

$$x_N = \beta_N = \left(d_N - a_N\beta_{N-1}\right)\big/\left(b_N + a_N\alpha_{N-1}\right).$$

Значения остальных неизвестных $x_i$ для $i = N - 1, N - 2, \ldots, 1$ теперь легко вычисляются по формуле (2.22).

Сделанные преобразования позволяют организовать вычисления методом прогонки в два этапа.

Прямой ход (прямая прогонка) состоит в вычислении прогоночных коэффициентов $\alpha_i$ ($1 \le i < N$) и $\beta_i$ ($1 \le i < N$). При $i = 1$ коэффициенты вычисляют по формулам

$$\alpha_1 = -c_1 / \gamma_1, \quad \beta_1 = d_1 / \gamma_1, \quad \gamma_1 = b_1, \qquad (2.23)$$

а при $i = 2, 3, \ldots, N - 1$ – по рекуррентным формулам

$$\alpha_i = -c_i / \gamma_i, \quad \beta_i = (d_i - a_i\beta_{i-1}) / \gamma_i, \quad \gamma_i = b_i + a_i\alpha_{i-1}. \qquad (2.24)$$

При $i = N$ прямая прогонка завершается вычислениями

$$\beta_N = \left(d_N - a_N\beta_{N-1}\right)\big/\gamma_N, \quad \gamma_N = b_N + a_N\alpha_{N-1}. \qquad (2.25)$$

Обратный ход метода прогонки (обратная прогонка) дает значения неизвестных. Сначала полагают $x_N = \beta_N$. Затем значения остальных неизвестных вычисляют по формуле

$$x_i = \alpha_i x_{i+1} + \beta_i, \quad i = N-1, N-2, \ldots, 1. \tag{2.26}$$

Вычисления выполняют в порядке убывания значений $i$ от $N-1$ до 1.

**Пример 2.6**

С помощью метода прогонки решить СЛАУ

$$\begin{aligned} 5x_1 - x_2 &= 2.0, \\ 2x_1 + 4.6x_2 - x_3 &= 3.3, \\ 2x_2 + 3.6x_3 - 0.8x_4 &= 2.6, \\ 3x_3 + 4.4x_4 &= 7.2. \end{aligned}$$

*Решение*

Прямой ход. Согласно формулам (2.23)–(2.25) получим:

$$\gamma_1 = b_1 = 5, \ \alpha_1 = -c_1 / \gamma_1 = 1 / 5 = 0.2, \ \beta_1 = d_1 / \gamma_1 = 2.0 / 5 = 0.4;$$

$$\gamma_2 = b_2 + a_2\alpha_1 = 4.6 + 2 \cdot 0.2 = 5, \ \alpha_2 = -c_2 / \gamma_2 = 1 / 5 = 0.2;$$

$$\beta_2 = (d_2 - a_2\beta_1) / \gamma_2 = (3.3 - 2 \cdot 0.4) / 5 = 0.5;$$

$$\gamma_3 = b_3 + a_3\alpha_2 = 3.6 + 2 \cdot 0.2 = 4, \ \alpha_3 = -c_3 / \gamma_3 = 0.8 / 4 = 0.2;$$

$$\beta_3 = (d_3 - a_3\beta_2) / \gamma_3 = (2.6 - 2 \cdot 0.5) / 4 = 0.4;$$

$$\gamma_4 = b_4 + a_4\alpha_3 = 4.4 + 3 \cdot 0.2 = 5;$$

$$\beta_4 = (d_4 - a_4\beta_3) / \gamma_4 = (7.2 - 3 \cdot 0.4) /5 = 1.2.$$

Обратный ход. Полагаем $x_4 = \beta_4 = 1.2$. Далее находим:

$$x_3 = \alpha_3 x_4 + \beta_3 = 0.2 \cdot 1.2 + 0.4 = 0.64;$$

$$x_2 = \alpha_2 x_3 + \beta_2 = 0.2 \cdot 0.64 + 0.5 = 0.628;$$

$$x_1 = \alpha_1 x_2 + \beta_1 = 0.2 \cdot 0.628 + 0.4 = 0.5256.$$

Итак, найденное решение: $x_1 = 0.5256$, $x_2 = 0.628$, $x_3 = 0.64$, $x_4 = 1.2$.

Непосредственный подсчет показывает, что для реализации вычислений по формулам (2.23)–(2.26) требуется примерно $8N$ арифметических операций, тогда как в методе Гаусса это число составляет примерно $(2/3)N^3$. Важно и то, что трехдиагональная

структура матрицы системы позволяет использовать для ее хранения $3N - 2$ машинных слова.

Таким образом, при одной и той же производительности и оперативной памяти компьютера метод прогонки позволяет решать системы гораздо большей размерности, чем стандартный метод Гаусса для систем уравнений с плотной (заполненной) матрицей.

О других прямых методах можно узнать, например, из [18].

### 2.2.3 Многократное решение СЛАУ

Часто необходимы многовариантный анализ или оптимизация рассматриваемого объекта (явления) в диапазоне изменения его параметров (изменение частоты воздействующего сигнала, учет частотной зависимости или разброса параметров структуры) с целью получения набора параметров, используемых для дальнейшего моделирования. Нередко такой анализ в САПР выполняется в интерактивном режиме, что соответствует изменению параметров математической модели. В этом случае временные затраты возрастают из-за необходимости решения (многократного) последовательности СЛАУ вида

$$\mathbf{A}_k\mathbf{X}_k = \mathbf{B}_k, \quad k = 1, 2, \ldots, m, \tag{2.27}$$

где $\mathbf{B} = [\mathbf{b}_1| \ \ldots \ |\mathbf{b}_s]$, $s << N$.

При неизменной матрице и нескольких правых частях используется LU-разложение. Если изменения (вариации) в матрицах СЛАУ незначительны и структурированы (меняется только малое количество строк и столбцов), а правая часть постоянна, то применим метод на основе формулы Шермана – Моррисона – Вудбери (метод окаймления [18]), а также схожие с ним, основанные на корректировке обратной матрицы. Вообще, формула Шермана – Моррисона – Вудбери использовалась и используется для исправления элементов обратной матрицы из-за ошибок округления и неточности задания исходных данных. Для СЛАУ с ленточной матрицей также разработаны эффективные алгоритмы. К сожалению, данные методы неприменимы для решения СЛАУ (2.27) при неструктурированных различиях в матрицах.

В практических приложениях при использовании метода моментов обусловленность матрицы СЛАУ ухудшается с приближением частоты воздействия к частотам собственных колебаний (внутренним резонансам) исследуемой структуры. Матрица таких структур на частотах собственных колебаний становится вырожденной, и ее определитель равен или близок к нулю. Поскольку частота воздействия выбирается дискретно, то существует вероятность такого совпадения. Решение на этой частоте может быть вовсе не получено, а на близких частотах оказывается неточным или некорректным, например показывает наличие мнимых резонансов. Для решения данной проблемы разработано несколько подходов различной степени кардинальности [19].

Предположим, что после того как было найдено решение $\mathbf{x}$ системы $\mathbf{Ax} = \mathbf{b}$, возникла необходимость решить систему $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{b}$ с матрицей, отличающейся от матрицы $\mathbf{A}$ несколькими элементами. Например, могло выясниться, что заданные ранее элементы содержали грубые ошибки. Возможно также, что решаемая задача такова, что в ней элементы матрицы последовательно меняются, а правая часть остается неизменной.

Разумеется, решение $\tilde{\mathbf{x}}$ можно найти, решив систему снова и проигнорировав полученную на предыдущем этапе информацию. Однако в рассматриваемом случае можно найти поправку $\Delta\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{x}$ к найденному ранее решению, используя всего $O(N^2)$ операций.

Пусть $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{u}\upsilon^T$, где $\mathbf{u}$ и $\upsilon$ – векторы размерностью $N$. Тогда справедлива формула Шермана – Моррисона – Вудбери [20]

$$\tilde{\mathbf{A}}^{-1} = \mathbf{A}^{-1} - \alpha\left(\mathbf{A}^{-1}\mathbf{u}\right)\left(\upsilon^T\mathbf{A}^{-1}\right),$$

где $\alpha = 1\Big/\left(1 - \upsilon^T\mathbf{A}^{-1}\mathbf{u}\right)$. Для ее применения необходимо выполнить следующую последовательность действий:

– решить СЛАУ $\mathbf{Ay} = \mathbf{u}$ относительно $\mathbf{y}$;

– решить СЛАУ $\mathbf{A}^T\mathbf{z} = \upsilon$ относительно $\mathbf{z}$;

– вычислить $\alpha = 1 / (1 + \upsilon^T\mathbf{y})$, $\beta = \mathbf{z}^T\mathbf{b}$ и $\Delta\mathbf{x} = \alpha\beta\mathbf{y}$;

– вычислить $\tilde{\mathbf{x}} = \mathbf{x} - \Delta\mathbf{x}$.

Суммарное число операций будет действительно составлять $O(N^2)$, если для решения систем $\mathbf{Ay} = \mathbf{u}$ и $\mathbf{A}^T\mathbf{z} = \upsilon$ применить обратную подстановку с использованием LU-разложения матрицы $\mathbf{A}$, найденного ранее на этапе решения $\mathbf{Ax} = \mathbf{b}$.

Когда матрица $\tilde{\mathbf{A}}$ отличается от матрицы $\mathbf{A}$ только одним элементом $\tilde{a}_{ij} = a_{ij} + \Delta a_{ij}$, можно положить $\mathbf{u} = \Delta a_{ij}\mathbf{e}_i$ и $\upsilon = \mathbf{e}_j$. Тогда последовательность действий примет вид:

– решить СЛАУ $\mathbf{Ay} = \Delta a_{ij}\mathbf{e}_i$ относительно $\mathbf{y}$;

– решить СЛАУ $\mathbf{A}^T\mathbf{z} = \mathbf{e}_j$ относительно $\mathbf{z}$;

– вычислить $\alpha = 1 / (1 - y_j)$, $\beta = \mathbf{z}^T\mathbf{b}$ и $\Delta\mathbf{x} = \alpha\beta\mathbf{y}$;

– вычислить $\tilde{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x}$.

Основываясь на формуле Шермана – Моррисона, можно указать способ пересчета LU-разложения матрицы [21].

**Пример 2.7**

Дано

$$\mathbf{A}_1\mathbf{x}_1 = \mathbf{b},\ \mathbf{A}_2\mathbf{x}_2 = \mathbf{b} \text{ и } \mathbf{A}_2 = \mathbf{A}_1 + \mathbf{u}\upsilon^T,$$

где $\mathbf{A}_1 = \begin{pmatrix} 3 & 2 & 2 \\ 1 & 3 & 1 \\ 5 & 3 & 4 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 3 \\ -1 \\ 6 \end{pmatrix}, \mathbf{u} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \upsilon^T = [2\ 3\ 7]$. Найти $\mathbf{x}_2$, используя формулу Шермана – Моррисона – Вудбери.

*Решение*

Результатом LU-разложения матрицы $\mathbf{A}_1$ будет

$$\mathbf{L}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0.333 & 1 & 0 \\ 1.667 & -0{,}143 & 1 \end{pmatrix},\ \mathbf{U}_1 = \begin{pmatrix} 3 & 2 & 2 \\ 0 & 2.333 & 0.333 \\ 0 & 0 & 0.714 \end{pmatrix},$$

а решением – $\mathbf{x}_1 = (1, -1, 1)^T$. Далее, решив системы $\mathbf{A}_1\mathbf{y} = \mathbf{u}$ и $\mathbf{A}_1{}^T\mathbf{z} = \upsilon$, где $\upsilon = \mathbf{A}^T = (\mathbf{LU})^T = \mathbf{U}^T\mathbf{L}^T$, получим $\mathbf{y} = (-0.4, 0.4, 0.2)^T$ и $\mathbf{z} = (-12.6, 1.8, 7.6)^T$.

Тогда $\upsilon^T\mathbf{y} = 1.8$, $\alpha = 0.357$ и $\beta = 6$. В результате $\Delta\mathbf{x} = (-0.857, 0.857, 0.429)^T$ и $\mathbf{x}_2 = (1.857, -1.857, 0.571)^T$.

## 2.3 Итерационные методы решения СЛАУ

### 2.3.1 Особенности итерационных методов

Развитие вычислительной техники и вызванный этим процессом переход ко все более сложным моделям привели к необходимости решения больших СЛАУ. Точные методы понятны и просты для программной реализации, однако их вычислительные затраты серьезно ограничивают круг рассматриваемых проблем. Поэтому итерационные методы получили широкое распространение при решении различных прикладных и научных задач.

Одной из главных особенностей итерационных методов является то, что получаемая погрешность решения из-за конечного числа разрядов много меньше, чем в методе Гаусса, так как она не накапливается, а определяется только последней итерацией и не зависит от их числа. Поэтому решение с заданной точностью при росте числа обусловленности матрицы достигается просто увеличением числа итераций. Очевидно, что снижение числа обусловленности матрицы позволяет уменьшить время решения с требуемой точностью за счет уменьшения числа итераций. Также можно уменьшить количество разрядов представления чисел с плавающей запятой для снижения вычислительных затрат. Такая опциональность для выбора пользователя реализована, например, в FEKO [5]. Применение итерационных методов является эффективным способом решения СЛАУ с плохо обусловленной матрицей. Кроме того, итерационные методы приемлемы и при хорошо обусловленной матрице. Так, метод моментов дает СЛАУ с плотной матрицей, для решения которой традиционно используются точные методы, например метод Гаусса. Однако их вычислительные затраты $\sim N^3$ ($N$ – порядок матрицы), что существенно ограничивает круг рассматриваемых задач даже при использовании высокоскоростных компьютеров. Практика же диктует необходимость решения сложных задач с постоянно увеличивающимися порядками СЛАУ. Поскольку в основе каждой итерации лежит умножение матрицы на вектор, то затраты итерационных методов (без предобусловливания) $\sim N_{it} \cdot N^2$, где $N^2$ – вычислительные

затраты на итерацию; $N_{it}$ – число итераций, необходимых для сходимости. Если по грубой эмпирической оценке $N_{it} \approx N^{0,5}$, тогда выигрыш по сравнению с методом Гаусса также $\approx N^{0,5}$. Однако ряд разработанных математических подходов к ускорению процедуры умножения матрицы на вектор может уменьшить вычислительные затраты с $\sim N^2$ до $\sim N \log_2 N$, позволяя получить значительный дополнительный выигрыш.

Итерационные методы имеют долгую и интересную историю развития, как и прямые, и первые из них также связаны с именем К. Ф. Гаусса. Ставшие уже классическими методы Гаусса – Зейделя, Якоби, Ричардсона, релаксации и другие, основанные на расщеплении матрицы СЛАУ, редко применяются на практике из-за их медленной или неустойчивой сходимости и невозможности применения предобусловливания. Исключение составляют их блочные или параллельные версии. В целом эти методы являются отличным началом для освещения общих принципов построения итерационного процесса в учебных дисциплинах по численным методам и формированию предобусловливателей.

Наиболее эффективными и устойчивыми среди итерационных методов являются так называемые проекционные методы, и особенно тот их класс, который связан с проектированием на подпространства Крылова. Эти методы обладают рядом достоинств: они устойчивы, допускают эффективное распараллеливание и работу с предобусловливателями разных типов. В 1906 г. в своих первых «Лекциях о приближенных вычислениях» (неоднократно переиздававшихся) Алексей Николаевич Крылов заложил принципы вычислительной математики и по праву считается ее основателем.

Исторически итерационные методы разрабатывались для решения разреженных СЛАУ высокого порядка. Главный источник таких СЛАУ – сеточные методы (конечных разностей, конечных элементов и др.) решения многомерных краевых задач.

Первые итерационные методы основывались на циклическом покомпонентном изменении вектора решения, осуществляемом таким образом, чтобы обнулить соответствующий коэффициент вектора невязки и тем самым уменьшить его норму. Подобная ме-

тодика уточнения решения получила название «релаксация». Хотя в настоящее время такие методы в их классической формулировке уже практически не применяются, существуют определенные классы задач, для которых разработаны их модификации, хорошо себя зарекомендовавшие. Кроме того, как будет показано далее, эти методы могут быть применены не в качестве самостоятельного средства решения СЛАУ, а для предобусловливания.

### 2.3.2 Методы Якоби и Гаусса – Зейделя

Пусть матрица **A** системы **Ax** = **b** такова, что ее главная диагональ не содержит нулевых элементов. Представим ее в виде разности

$$\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}, \qquad (2.28)$$

где матрица **D** содержит диагональные элементы матрицы **A**; матрица **E** – только поддиагональные; матрица **F** – только наддиагональные. Тогда система **Ax** = **b** может быть записана в виде

$$\mathbf{Dx} - \mathbf{Ex} - \mathbf{Fx} = \mathbf{b}.$$

Если имеется приближение $\mathbf{x}_k$ к точному решению СЛАУ $\mathbf{x}_*$, то при $\mathbf{x}_k \neq \mathbf{x}_*$ это соотношение не выполняется. Однако, если в выражении

$$\mathbf{Dx}_k - \mathbf{Ex}_k - \mathbf{Fx}_k = \mathbf{b} \qquad (2.29)$$

одно или два из вхождений вектора $\mathbf{x}_k$ заменить на $\mathbf{x}_{k+1}$ и потребовать, чтобы равенство имело место, можно получить некоторую вычислительную схему для уточнения решения.

Наиболее простой с точки зрения объема вычислений вариант получается при замене в уравнении (2.29) $\mathbf{Dx}_k$ на $\mathbf{Dx}_{k+1}$. При этом получается схема

$$\mathbf{x}_{k+1} = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})\mathbf{x}_k + \mathbf{D}^{-1}\mathbf{b}, \qquad (2.30)$$

известная как метод Якоби (еще называемая методом Гаусса – Якоби). Метод известен с 1840-х гг.

Выражение (2.30) в скалярной форме имеет вид

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=k+1}^{N} a_{ij} x_j^{(k)} \right), \; i = 1,\ldots,N, \qquad (2.31)$$

откуда хорошо видна основная идея метода: на ($k$+1)-й итерации $i$-й компонент вектора решения изменяется по сравнению с $k$-й итерацией так, чтобы $i$-й компонент вектора невязки $\mathbf{r}_{k+1}$ стал нулевым (при условии отсутствия изменений в других компонентах вектора $\mathbf{x}$). Далее приведем алгоритм метода Якоби.

### Алгоритм метода Якоби

Выбрать произвольное начальное приближение $\mathbf{x}^{(0)}$
Для $k = 1, 2, \ldots$
    Для $i = 1, \ldots, N$
        $\tilde{x}_i = 0$
        Для $j = 1, \ldots, i-1, i+1, \ldots, N$
            $\tilde{x}_i = \tilde{x}_i + a_{ij} x_j^{(k-1)}$
        Увеличить $j$
        $\tilde{x}_i = (b_i - \tilde{x}_i) / a_{ii}$
    Увеличить $i$
    $\mathbf{x}^{(k)} = \tilde{\mathbf{x}}$
    Проверить сходимость и продолжить при необходимости
Увеличить $k$

### Пример 2.8

С помощью метода Якоби решить СЛАУ $\begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 11 \end{pmatrix}$.

*Решение*

Пусть $\mathbf{x}^{(0)} = (1, 1)^T$, тогда получим следующий результат.

| | | |
|---|---|---|
| Первая итерация | $x_1^{(1)} = (b_1 - a_{12}x_2^{(0)})/a_{11} =$ $= (4 - 1\cdot 1) / 2 = 1.5$ | $x_2^{(1)} = (b_2 - a_{21}x_1^{(0)})/a_{22} =$ $= (11 - 3\cdot 1) / 4 = 2$ |
| Вторая итерация | $x_1^{(2)} = (b_1 - a_{12}x_2^{(1)})/a_{11} =$ $= (4 - 1\cdot 2) / 2 = 1$ | $x_2^{(2)} = (b_2 - a_{21}x_1^{(1)})/a_{22} =$ $= (11 - 3\cdot 1.5) / 4 = 1.625$ |
| Третья итерация | $x_1^{(3)} = (b_1 - a_{12}x_2^{(2)})/a_{11} =$ $= (4 - 1\cdot 1.625) / 2 = 1.1875$ | $x_2^{(3)} = (b_2 - a_{21}x_1^{(2)})/a_{22} =$ $= (11 - 3\cdot 1) / 4 = 2$ |
| Четвертая итерация | $x_1^{(4)} = (b_1 - a_{12}x_2^{(3)})/a_{11} =$ $= (4 - 1\cdot 2) / 2 = 1$ | $x_2^{(4)} = (b_2 - a_{21}x_1^{(3)})/a_{22} =$ $= (11 - 3\cdot 1.1875) / 4 \approx 1.9063$ |
| Пятая итерация | $x_1^{(5)} = (b_1 - a_{12}x_2^{(4)})/a_{11} =$ $= (4 - 1\cdot 1.9063) / 2 \approx 1.0469$ | $x_2^{(5)} = (b_2 - a_{21}x_1^{(4)})/a_{22} =$ $= (11 - 3\cdot 1) / 4 = 2$ |
| Шестая итерация | $x_1^{(6)} = (b_1 - a_{12}x_2^{(5)})/a_{11} =$ $= (4 - 1\cdot 2) / 2 = 1$ | $x_2^{(6)} = (b_2 - a_{21}x_1^{(5)})/a_{22} =$ $= (11 - 3\cdot 1.0469) / 4 \approx 1.9648$ |

| | | |
|---|---|---|
| Седьмая итерация | $x_1^{(7)} = (b_1 - a_{12}x_2^{(6)})/a_{11} =$ $= (4 - 1\cdot 1.9648) / 2 \approx 1.0176$ | $x_2^{(7)} = (b_2 - a_{21}x_1^{(6)})/a_{22} =$ $= (11 - 3\cdot 1) / 4 = 2$ |
| Восьмая итерация | $x_1^{(8)} = (b_1 - a_{12}x_2^{(7)})/a_{11} =$ $= (4 - 1\cdot 2) / 2 = 1$ | $x_2^{(8)} = (b_2 - a_{21}x_1^{(7)})/a_{22} =$ $= (11 - 3\cdot 1.0176) / 4 \approx 1.9868$ |
| Девятая итерация | $x_1^{(9)} = (b_1 - a_{12}x_2^{(8)})/a_{11} =$ $= (4 - 1\cdot 1.9868) / 2 \approx 1.0061$ | $x_2^{(9)} = (b_2 - a_{21}x_1^{(8)})/a_{22} =$ $= (11 - 3\cdot 1)/4 = 2$ |
| Десятая итерация | $x_1^{(10)} = (b_1 - a_{12}x_2^{(9)})/a_{11} =$ $= (4 - 1\cdot 2) / 2 = 1$ | $x_2^{(10)} = (b_2 - a_{21}x_1^{(9)})/a_{22} =$ $= (11 - 3\cdot 1.0061) / 4 \approx 1.9954$ |

Недостатком схемы (2.30)–(2.31) является то, что при нахождении компонента $x_i^{(k+1)}$ никак не используется информация о пересчитанных компонентах $x_1^{(k+1)},\ldots, x_{i-1}^{(k+1)}$. Исправить этот недостаток можно, переписав выражение (2.31) в виде

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=k+1}^{N} a_{ij}x_j^{(k)} \right), \quad i = 1,\ldots,N, \qquad (2.32)$$

что в векторной форме эквивалентно

$$\mathbf{x}_{k+1} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{F}\mathbf{x}_k + (\mathbf{D} - \mathbf{E})^{-1}\mathbf{b}. \qquad (2.33)$$

Такая схема называется методом Гаусса – Зейделя (иногда его называют просто методом Зейделя), который известен с 1870-х гг. Приведем алгоритм метода Гаусса – Зейделя.

### Алгоритм метода Гаусса – Зейделя

Выбрать произвольное начальное приближение $\mathbf{x}^{(0)}$
Для $k = 1, 2, \ldots$
  Для $i = 1, \ldots, N$
    $\sigma = 0$
    Для $j = 1, \ldots, i - 1$
      $\sigma = \sigma + a_{ij}x_j^{(k)}$
    Увеличить $j$
    Для $j = i + 1, \ldots, N$
      $\sigma = \sigma + a_{ij}x_j^{(k-1)}$
    Увеличить $j$
    $x_i^{(k)} = (b_i - \sigma) / a_{ii}$
  Увеличить $i$
  Проверить сходимость и продолжить при необходимости
Увеличить $k$

**Пример 2.9**

С помощью метода Якоби решить СЛАУ из примера 2.8.

*Решение*

Пусть $\mathbf{x}^{(0)} = (1, 1)^T$, тогда получим следующий результат.

| | | |
|---|---|---|
| Первая итерация | $x_1^{(1)} = (b_1 - a_{12}x_2^{(0)})/a_{11} =$ $= (4 - 1\cdot 1) / 2 = 1.5$ | $x_2^{(1)} = (b_2 - a_{21}x_1^{(1)})/a_{22} =$ $= (11 - 3\cdot 3 / 2) / 4 = 1.625$ |
| Вторая итерация | $x_1^{(2)} = (b_1 - a_{12}x_2^{(1)})/a_{11} =$ $= (4 - 1\cdot 1.625) / 2 = 1.1875$ | $x_2^{(2)} = (b_2 - a_{21}x_1^{(2)})/a_{22} =$ $= (11 - 3\cdot 1.1875)/4 \approx 1.9063$ |
| Третья итерация | $x_1^{(3)} = (b_1 - a_{12}x_2^{(2)})/a_{11} =$ $= (4 - 1\cdot 1.9063) / 2 \approx 1.0469$ | $x_2^{(3)} = (b_2 - a_{21}x_1^{(3)})/a_{22} =$ $= (11 - 3\cdot 1.0469) / 4 \approx 1.9648$ |
| Четвертая итерация | $x_1^{(4)} = (b_1 - a_{12}x_2^{(3)})/a_{11} =$ $= (4 - 1\cdot 1.9648) / 2 \approx 1.0176$ | $x_2^{(4)} = (b_2 - a_{21}x_1^{(4)})/a_{22} =$ $= (11 - 3\cdot 1.0176) / 4 \approx 1.9868$ |
| Пятая итерация | $x_1^{(5)} = (b_1 - a_{12}x_2^{(4)})/a_{11} =$ $= (4 - 1\cdot 1.9868) / 2 \approx 1.0061$ | $x_2^{(5)} = (b_2 - a_{21}x_1^{(5)})/a_{22} =$ $= (11 - 3\cdot 1.0061) / 4 \approx 1.9954$ |
| Шестая итерация | $x_1^{(6)} = (b_1 - a_{12}x_2^{(5)})/a_{11} =$ $= (4 - 1\cdot 1.9954) / 2 \approx 1.0023$ | $x_2^{(6)} = (b_2 - a_{21}x_1^{(6)})/a_{22} =$ $= (11 - 3\cdot 1.0023) / 4 \approx 1.9983$ |
| Седьмая итерация | $x_1^{(7)} = (b_1 - a_{12}x_2^{(6)})/a_{11} =$ $= (4 - 1\cdot 1.9983) / 2 \approx 1.0009$ | $x_2^{(7)} = (b_2 - a_{21}x_1^{(7)})/a_{22} =$ $= (11 - 3\cdot 1.0009) / 4 \approx 1.9994$ |

Выражение (2.32) получается из (2.29) заменой $\mathbf{x}_k$ на $\mathbf{x}_{k+1}$ при матрицах **D** и **E**. Если вместо **D** и **E** взять **D** и **F**, то получится похожая схема

$$\mathbf{x}_{k+1} = (\mathbf{D} - \mathbf{F})^{-1}\mathbf{E}\mathbf{x}_k + (\mathbf{D} - \mathbf{F})^{-1}\mathbf{b}, \qquad (2.34)$$

которая называется обратным методом Гаусса – Зейделя.

Еще одной модификацией является симметричный метод Гаусса – Зейделя, который заключается в циклическом чередовании формул (2.33) и (2.34) на соседних итерациях.

Заметим, что выражения (2.30), (2.33) и (2.34), могут быть записаны в виде

$$\mathbf{K}\mathbf{x}_{k+1} = \mathbf{R}\mathbf{x}_k + \mathbf{b}, \qquad (2.35)$$

где матрицы **K** и **R** связаны соотношением

$$\mathbf{A} = \mathbf{K} - \mathbf{R}. \qquad (2.36)$$

Такое представление матрицы **A** называется расщеплением, а методы вида (2.35) – методами, основанными на расщеплении. Очевидно, что матрица **K** должна быть невырожденной и легко обратимой (т.е. число затрачиваемых арифметических операций должно быть как можно меньше).

### 2.3.3 Релаксационные методы

Скорость сходимости методов, основанных на расщеплении, непосредственно связана со спектральным радиусом матрицы (максимальное по абсолютной величине собственное число) $\mathbf{K}^{-1}\mathbf{R}$; с другой стороны, выбор **K** ограничен требованием легкой обратимости. Одним из распространенных способов улучшения сходимости является введение дополнительного параметра. Пусть $\omega$ – некоторое вещественное число. Рассмотрим вместо системы $\mathbf{Ax} = \mathbf{b}$ масштабированную систему

$$\omega\mathbf{Ax} = \omega\mathbf{b}, \tag{2.37}$$

и вместо выражения (2.28) воспользуемся представлением

$$\omega\mathbf{A} = (\mathbf{D} - \omega\mathbf{E}) - \big(\omega\mathbf{F} + (1-\omega)\mathbf{D}\big), \tag{2.38}$$

где матрицы **D**, **E**, **F** имеют тот же смысл, что и в (2.28).

Тогда на основании соотношений (2.37) и (2.38) можно построить итерационную схему, похожую на метод Гаусса – Зейделя:

$$(\mathbf{D} - \omega\mathbf{E})\mathbf{x}_{k+1} = (\omega\mathbf{F} + (1-\omega)\mathbf{D})\mathbf{x}_k + \omega\mathbf{b}. \tag{2.39}$$

Эта схема называется методом последовательной верхней релаксации (SOR). Для нее

$$\mathbf{K}_{\mathrm{SOR}}(\omega) = \mathbf{D} - \omega\mathbf{E}\ ,$$

$$\mathbf{R}_{\mathrm{SOR}}(\omega) = \omega\mathbf{F} + (1-\omega)\mathbf{D}.$$

Выбор параметра $\omega$, минимизирующего спектральный радиус, является, вообще говоря, достаточно сложной проблемой. Но для многих классов матриц такая задача исследована и оптимальные значения известны. Приведем алгоритм метода последовательной верхней релаксации.

## Алгоритм метода последовательной верхней релаксации (SOR)

Выбрать произвольное начальное приближение $\mathbf{x}^{(0)}$
Для $k = 1, 2, \ldots$
    Для $i = 1, \ldots, N$
        $\sigma = 0$
        Для $j = 1, \ldots, i - 1$
            $\sigma = \sigma + a_{ij} x_j^{(k)}$
        Увеличить $j$
        Для $j = i + 1, \ldots, N$
            $\sigma = \sigma + a_{ij} x_j^{(k-1)}$
        Увеличить $j$
        $\sigma = (b_i - \sigma) / a_{ii}$
        $x_i^{(k)} = x_i^{(k-1)} + \omega(\sigma - x_i^{(k-1)})$
    Увеличить $i$
    Проверить сходимость и продолжить при необходимости
Увеличить $k$

Выражение (2.38) остается тождеством, если в нем поменять местами матрицы **E** и **F**. Такая перестановка дает обратный метод последовательной верхней релаксации:

$$(\mathbf{D} - \omega\mathbf{F})\mathbf{x}_{k+1} = [\omega\mathbf{E} + (1 - \omega)\mathbf{D}]\mathbf{x}_k + \omega\mathbf{b}.$$

Последовательное применение прямого и обратного методов SOR является симметричным методом последовательной верхней релаксации (SSOR):

$$(\mathbf{D} - \omega\mathbf{E})\mathbf{x}_{k+1/2} = [\omega\mathbf{F} + (1 - \omega)\mathbf{D}]\mathbf{x}_k + \omega\mathbf{b};$$

$$(\mathbf{D} - \omega\mathbf{F})\mathbf{x}_{k+1} = [\omega\mathbf{E} + (1 - \omega)\mathbf{D}]\mathbf{x}_{k+1/2} + \omega\mathbf{b}.$$

Приведем алгоритм метода симметричной последовательной верхней релаксации.

## Алгоритм метода симметричной последовательной верхней релаксации (SSOR)

Выбрать произвольное начальное приближение $\mathbf{x}^{(0)}$
$\mathbf{x}^{(1/2)} = \mathbf{x}^{(0)}$
Для $k = 1, 2, \ldots$
    Для $i = 1, \ldots, N$
        $\sigma = 0$
        Для $j = 1, \ldots, i - 1$

$\sigma = \sigma + a_{ij} x_j^{(k-1/2)}$

Увеличить $j$

Для $j = i + 1, \ldots, N$

$\sigma = \sigma + a_{ij} x_j^{(k-1)}$

Увеличить $j$

$\sigma = (b_i - \sigma) / a_{ii}$

$x_i^{(k-1/2)} = x_i^{(k-1)} + \omega(\sigma - x_i^{(k-1)})$

Увеличить $i$

Для $i = N, N - 1, \ldots, 1$

$\sigma = 0$

Для $j = 1, \ldots, i - 1$

$\sigma = \sigma + a_{ij} x_j^{(k-1/2)}$

Увеличить $j$

Для $j = i + 1, \ldots, N$

$\sigma = \sigma + a_{ij} x_j^{(k)}$

Увеличить $j$

$x_i^{(k)} = x_i^{(k-1/2)} + \omega(\sigma - x_i^{(k-1/2)})$

Проверить сходимость и продолжить при необходимости

Увеличить $k$

### 2.3.4 Методы крыловского типа

Рассмотрим систему (2.1) и сформируем для нее следующую задачу. Пусть заданы два подпространства $K \subset R^N$ и $L \subset R^N$. Требуется найти такой вектор $\mathbf{x} \in K$, который обеспечит решение системы (2.1), оптимальное относительно подпространства $L$, т. е. будет выполняться условие Петрова – Галеркина

$$\forall\, \mathbf{l} \in L\colon (\mathbf{Ax}, \mathbf{l}) = (\mathbf{b}, \mathbf{l})$$

или

$$\forall \mathbf{l} \in L\colon (\mathbf{r}_\mathbf{x}, \mathbf{l}) = 0 \Rightarrow \mathbf{r}_\mathbf{x} = \mathbf{b} - \mathbf{Ax} \perp L. \qquad (2.40)$$

Такая задача называется задачей проектирования $\mathbf{x}$ на подпространство $K$ ортогонально к подпространству $L$. В более общей постановке задача формулируется следующим образом. Пусть известно некоторое приближение $\mathbf{x}_0$ к точному решению $\mathbf{x}_*$ системы (2.1) и требуется уточнить его поправку $\boldsymbol{\delta}_\mathbf{x} \in K$ таким образом, чтобы $\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \boldsymbol{\delta}_\mathbf{x}) \perp L$. Тогда условие (2.40) примет вид

$$\forall\, \mathbf{l} \in L\colon (\mathbf{r}_{\mathbf{x}_0 + \boldsymbol{\delta}_x}, \mathbf{l}) = ((\mathbf{b} - \mathbf{Ax}_0) - \mathbf{A}\boldsymbol{\delta}_\mathrm{x}, \mathbf{l}) = (\mathbf{r}_0 - \mathbf{A}\boldsymbol{\delta}_\mathbf{x}, \mathbf{l}) = 0.$$

Введем в подпространствах $K$ и $L$ базисы $\{\boldsymbol{\nu}_j\}_{j=1}^{m}$ и $\{\boldsymbol{\omega}_j\}_{j=1}^{m}$, где $\dim K = \dim L = m$. Тогда последнее выражение будет справедливо при

$$\forall j\ (1 \le j \le m)\colon (\mathbf{r}_0 - \mathbf{A}\boldsymbol{\delta}_{\mathbf{x}}, \omega_j) = 0. \qquad (2.41)$$

Введя для базисов матричные обозначения $\mathbf{V} = [\boldsymbol{\nu}_1| \ldots |\boldsymbol{\nu}_m]$ и $\mathbf{W} = [\boldsymbol{\omega}_1| \ldots |\boldsymbol{\omega}_m]$, можно записать $\boldsymbol{\delta}_{\mathbf{x}} = \mathbf{V}\mathbf{y}$, где $\mathbf{y} \subset R^m$ – вектор коэффициентов. Тогда выражение (2.41) преобразуется к виду

$$\mathbf{W}^T(\mathbf{r}_0 - \mathbf{A}\mathbf{V}\mathbf{y}) = 0 \Rightarrow\ \mathbf{y} = (\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}_0. \qquad (2.42)$$

С учетом этого решение системы (2.1) должно уточняться в соответствии с формулой

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{V}\mathbf{y} = \mathbf{x}_0 + \mathbf{V}(\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}_0,$$

из которой вытекает важное требование для организации вычислений: произведение $\mathbf{W}^T\mathbf{A}\mathbf{V}$ должно быть или малой размерности, или легко обращаться. Из выражения (2.42) также вытекает соотношение

$$\mathbf{V}\mathbf{y} = \mathbf{A}^{-1}\mathbf{r}_0 = \mathbf{A}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_0) = \mathbf{x}_* - \mathbf{x}_0 = \boldsymbol{\delta}_{\mathbf{x}},$$

где $\mathbf{x}_*$ – точное решение СЛАУ (2.1). Таким образом, $\mathbf{V}\mathbf{y}$ – проекция разности между точным решением и начальным приближением на подпространство $K$.

В качестве подпространства $K$ часто выбирают подпространства Крылова – линейные пространства размерностью $m$, порожденные вектором $\boldsymbol{\nu}$ и матрицей $\mathbf{A}$ (линейная оболочка):

$$K_m(\boldsymbol{\nu}, \mathbf{A}) = \operatorname{span}\{\boldsymbol{\nu}, \mathbf{A}\boldsymbol{\nu}, \mathbf{A}^2\boldsymbol{\nu}, \ldots, \mathbf{A}^{m-1}\boldsymbol{\nu}\}.$$

При этом в качестве вектора $\boldsymbol{\nu}$ выбирается невязка начального приближения $\mathbf{r}_0$. Тогда конкретный выбор подпространства $L$ и способа построения базисов подпространств (биортогонализация Ланцоша, $A$-биортогонализация, ортогонализация Арнольди) полностью определяет вычислительную схему метода.

Пусть пространства $K$ и $L$ связаны соотношением $L = \mathbf{A}K$, причем в качестве $K$ используем подпространство Крылова $K_m(\boldsymbol{\nu}_1, \mathbf{A})$, где $\boldsymbol{\nu}_1 = \mathbf{r}_0 / \beta$, $\beta = \|\mathbf{r}_0\|_2$, а для построения ортонормированного базиса – ортогонализацию Арнольди. Рассмотрим задачу

минимизации функционала $\mathbf{\Phi}(\mathbf{x}) = \|\mathbf{r}_x\|_2^2$ (эквивалентную задаче проектирования (2.42)), так как любой вектор **x** из пространства $(\mathbf{x}_0 + K_m)$ может быть записан в виде

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y},$$

где **y** – вектор размера $m$. Таким образом, задачу минимизации функционала можно переписать как

$$\mathbf{y}_m = \arg\min_{\mathrm{y}} \|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_m\mathbf{y})\|_2^2 .$$

Вычислительная схема, построенная с использованием таких предпосылок, называется методом обобщенных минимальных невязок (GMRES). На практике чаще используется версия этого метода с рестартами, сокращенно GMRES($m$).

В отличие от ортогонализации Арнольди, биортогонализация Ланцоша использует для построения базиса экономичные трехчленные формулы. Так, выбрав в качестве пространства $K$ – $K_m(\mathbf{r}_0, \mathbf{A})$, а в качестве $L$ – $L_m(\underline{\mathbf{r}}_0, \mathbf{A}^T)$, где вектор $\underline{\mathbf{r}}_0$ выбирается из условия $(\mathbf{r}_0, \underline{\mathbf{r}}_0) \neq 0$, строят метод бисопряженных градиентов (BiCG). Достаточно часто для него характерна неустойчивость решения и осциллирующее поведение нормы невязки. Более того, итерационный процесс может полностью оборваться без возможности его дальнейшего продления. К тому же метод BiCG плохо поддается реализации на многопроцессорных вычислительных системах с распределенной памятью за счет использования операций с транспонированной матрицей. Эти проблемы привели к разработке целого класса методов, в которых операция с транспонированной матрицей не используется.

Алгебраически это достигается за счет изменения специальным образом полинома $\mathrm{p}_m$, которому удовлетворяет последовательность невязок в методах, использующих подпространства Крылова. Из ряда методов, свободных от транспонирования, в настоящее время широко применяется стабилизированный метод бисопряженных градиентов (BiCGStab), использующий соотношение $\mathrm{r}_m = \mathrm{p}_m(\mathbf{A})\mathrm{q}_m(\mathbf{A})\mathbf{r}^{(0)}$, где $\mathrm{q}_m$ – специальным образом строящийся полином, такой, что произведение $\mathrm{p}_m\mathrm{q}_m$ не содержит нечетных степеней. Еще один метод этого семейства – квадратичный

метод сопряженных градиентов (CGS). Данный метод строится на основе соотношения $r_m = p_m(\mathbf{A}^T)\underline{\mathbf{r}}_0$.

Если матрица СЛАУ симметричная и положительно определенная, то метод BiCG имеет более простой вид. Метод, который получается за счет упрощений, вносимых симметричностью, называется методом сопряженных градиентов (CG). Следует заметить, что метод бисопряженных градиентов исторически появился как обобщение CG на несимметричный случай. Существуют и другие методы. Одни из них появились самостоятельно, а другие – как модификации уже известных.

В таблице 2.1 приведен перечень итерационных методов крыловского типа (1950–2014 гг.) для решения СЛАУ с одной правой частью [22]. При решении электромагнитных задач широкое распространение нашли методы BiCGStab и GMRES($m$). Стоит отметить, что данный перечень неполный, но еще раз подтверждает тот факт, что развитие итерационных методов актуально.

Таблица 2.1 – Итерационные методы крыловского типа

| Год | Разработчик(и) | Метод |
|---|---|---|
| 1950 | Lanczos | Lanczos |
| 1951 | Arnoldi | Arnoldi |
| 1952 | Hestenes, Stiefel | CG, CGNR |
| 1952 | Lanczos | Lanczos (CG) |
| 1955 | Craig | CGNE |
| 1975 | Paige, Saunders | MINRES |
| 1975 | Paige, Saunders | SYMMLQ |
| 1975 | Fletcher | BiCG |
| 1976 | Concus, Golub | CGW |
| 1977 | Vinsome | ORTHOMIN |
| 1977 | Meijerink, Van der Vorst | ICCG |
| 1978 | Widlund | CGW |
| 1980 | Jea, Young | ORTHODIR |
| 1980 | Wesseling, Sonneveld | IDR |
| 1981 | Saad | FOM |
| 1982 | Paige, Saunders | LSQR |
| 1983 | Eisenstat и др. | GCR |
| 1986 | Saad, Schultz | GMRES |
| 1989 | Sonneveld | CGS |
| 1990 | Van der Vorst, Melissen | COCG |

Окончание таблицы 2.1

| Год | Разработчик(и) | Метод |
|---|---|---|
| 1991 | Freund and Nachtigal | QMR |
| 1992 | Van der Vorst | BiCGStab |
| 1993 | Gutknecht | BiCGStab2 |
| 1993 | Sleijpen, Fokkema | BiCGStab(*l*) |
| 1994 | Freund | TFQMR |
| 1994 | Weiss | GMERR |
| 1994 | Chan и др. | QMR-BiCGStab |
| 1994 | Freund, Nachtigal | SQMR |
| 1995 | Kasenally, Ebrahim | GMBACK |
| 1996 | Fokkema и др. | CGS2 |
| 1997 | Pocock, Walker | CBICG |
| 1997 | Zhang | GPBi-CG |
| 1999 | de Sturler | GCROT |
| 1999 | Sadok | CMHR |
| 2001 | Szyld, Vogel | FQMR |
| 2007 | Sogabe, Zhang | COCR |
| 2008 | Sonneveld, van Gijzen | IDR(*s*) |
| 2008 | Ильин | BiCR (A-BiCG, A-CGS, A-BiCGStab), SCR |
| 2009 | Jing и др. | BiCOR, BiCORSTAB |
| 2010 | Abe, Sleijpen | BiCR |
| 2010 | Tanio, Sugihara | GBi-CGSTAB |
| 2010 | Hicken, Zingg | GCROT(*m*, *k*) |
| 2011 | Carpentieri и др. | BiCOR, CORS |
| 2013 | Zhao и др. | GPBiCOR |
| 2013 | Zhao, Huang | BiCORSTAB2 |
| 2013 | Hajarian | QMRCGSTAB |
| 2014 | Sun и др. | QMRCORSTAB |
| 2014 | Zhang | GCORS |

Для примера приведем алгоритмы методов BiCGStab и CGS с предобусловливанием.

### Алгоритм метода BiCGStab

Вычислить предобусловливатель $\mathbf{M}$
Выбрать начальное приближение $\mathbf{x}_0$
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
Выбрать вектор $\tilde{\mathbf{r}}$, удовлетворяющий условию $(\mathbf{r}_0, \tilde{\mathbf{r}}) \neq 0$ (например, $\tilde{\mathbf{r}} = \mathbf{r}_0$)

Для $i = 1, 2, \ldots$ до сходимости или до $N_{it}^{\max}$

$\rho_{i-1} = (\tilde{\mathbf{r}}, \mathbf{r}_{i-1})$

Если $\rho_{i-1} = 0$

то метод не может решить данную систему

Если $i = 1$

$\mathrm{p}_i = \mathrm{r}_{i-1}$

Иначе

$\beta_{i-1} = (\rho_{i-1} / \rho_{i-2})\ (\alpha_{i-1} / \omega_{i-1})$

$\mathbf{p}_i = \mathbf{r}_{i-1} + \beta_{i-1}(\mathbf{p}_{i-1} - \omega_{i-1}\, \mathbf{v}_{i-1})$

Решить СЛАУ $\mathbf{M}\tilde{\mathbf{p}} = \mathbf{p}_i$ относительно $\tilde{\mathbf{p}}$

$\mathbf{v}_i = \mathbf{A}\tilde{\mathbf{p}}$

$\alpha_i = \rho_{i-1} / (\tilde{\mathbf{r}}, \mathbf{v}_i)$

$\mathbf{s} = \mathbf{r}_{i-1} - \alpha_i\, \mathbf{v}_i$

Если $\|\mathbf{s}\|_2 / \|\mathbf{r}_0\|_2 \le Tol$

то КОНЕЦ ($\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i \tilde{\mathbf{p}}$ – полученное решение)

Решить СЛАУ $\mathbf{M}\tilde{\mathbf{s}} = \mathbf{s}_i$ относительно $\tilde{\mathbf{s}}$

$\mathbf{t} = \mathbf{A}\,\tilde{\mathbf{s}}$

$\omega_i = (\mathbf{t}, \mathbf{s}) / (\mathbf{t}, \mathbf{t})$

$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i \tilde{\mathbf{p}} + \omega_i\, \tilde{\mathbf{s}}$

$\mathbf{r}_i = \mathbf{s} - \omega_i\, \mathbf{t}$

Если $\|\mathbf{r}\|_2 / \|\mathbf{r}_0\|_2 \le Tol$

то КОНЕЦ ($\mathbf{x}^{(i)}$ – полученное решение)

Увеличить $i$

## Алгоритм метода CGS

Вычислить предобусловливатель $\mathbf{M}$

Выбрать начальное приближение $\mathbf{x}_0$

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

Выбрать вектор $\tilde{\mathbf{r}}$, удовлетворяющий условию $(\mathbf{r}_0, \tilde{\mathbf{r}}) \neq 0$ (например, $\tilde{\mathbf{r}} = \mathbf{r}_0$)

Для $i = 1, 2, \ldots$ до сходимости или до $N_{it}^{\max}$

$\rho_{i-1} = (\tilde{\mathbf{r}}, \mathbf{r}_{i-1})$

Если $\rho_{i-1} = 0$

то метод не может решить данную систему

Если $i = 1$

$\mathbf{u}_1 = \mathbf{r}_0$

$\mathbf{p}_1 = \mathbf{u}_1$

**Иначе**

$\beta_{i-1} = (\rho_{i-1} / \rho_{i-2})$

$\mathbf{u}_i = \mathbf{r}_{i-1} + \beta_{i-1}\mathbf{q}_{i-1}$

$\mathbf{p}_i = \mathbf{u}_i + \beta_{i-1}(\mathbf{q}_{i-1} + \beta_{i-1}\mathbf{p}_{i-1})$

Решить СЛАУ $\mathbf{M}\tilde{\mathbf{p}} = \mathbf{p}_i$ относительно $\tilde{\mathbf{p}}$

$\tilde{\mathbf{v}} = \mathbf{A}\tilde{\mathbf{p}}$

$\alpha_i = \rho_{i-1} / (\tilde{\mathbf{r}}, \tilde{\mathbf{v}})$

$\mathbf{q}_i = \mathbf{u}_i - \alpha_i \tilde{\mathbf{v}}$

Решить СЛАУ $\mathbf{M}\tilde{\mathbf{u}} = \mathbf{u}_i + \mathbf{q}_i$ относительно $\tilde{\mathbf{u}}$

$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i \tilde{\mathbf{u}}$

$\tilde{\mathbf{q}} = \mathbf{A}\tilde{\mathbf{u}}$

$\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_i \tilde{\mathbf{q}}$

Если $\|\mathbf{r}\|_2 / \|\mathbf{r}_0\|_2 \le Tol$

то КОНЕЦ ($\mathbf{x}_i$ – полученное решение)

Увеличить $i$

### 2.3.5 Предобусловливание

Для ускорения процесса итерационного решения разреженных СЛАУ часто используется предобусловливание, которое позволяет улучшить обусловленность результирующей матрицы и тем самым уменьшить число итераций. Так, с ростом cond(**A**) обусловленность ухудшается и для ряда проблем сходимость может оказаться очень медленной, поэтому итерационный процесс может стагнировать или даже оборваться. Термин «предобусловливание» (preconditioning), по всей видимости, был впервые использован Форсайтом в 1955 г. при рецензировании работы Ланцоша. (В [18] данный подход назван «подготовка».) В современной трактовке этот термин впервые использован в работе [23]. Позже предобусловливание было применено и при решении СЛАУ с плотной матрицей. При этом для решения таких систем, как правило, используют метод исключения Гаусса и его модификации.

Поясним суть данного подхода. Пусть **M** – некоторая невырожденная квадратная матрица порядка *N*.

Тогда, домножив систему (2.1) на матрицу $\mathbf{M}^{-1}$, получим эквивалентную систему

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}, \qquad (2.43)$$

имеющую то же точное решение $\mathbf{x}_*$. Хотя выражения (2.1) и (2.43) алгебраически эквивалентны, спектральные характеристики матрицы $\mathbf{M}^{-1}\mathbf{A}$ отличаются от характеристик исходной матрицы **A**,

что ведет к изменению скорости сходимости методов для системы (2.1) в конечной арифметике. Процесс перехода от (2.1) к (2.43) и называется предобусловливанием, а матрица $\mathbf{M}^{-1}$ – предобусловливателем (предобусловливающая матрица). При этом матрица $\mathbf{M}$ должны быть близка к матрице $\mathbf{A}$ (сформирована из нее) и легко вычислима и обратима. Невязка ($\underline{\mathbf{r}}$) системы (2.43) связана с невязкой ($\mathbf{r}$) исходной системы (2.1) соотношением $\mathbf{M}\underline{\mathbf{r}} = \mathbf{r}$, которое справедливо и для других матрично-векторных произведений, что позволяет вместо явного перехода от (2.1) к (2.43) вводить в схемы методов корректирующие шаги (матрично-векторное умножение).

Описанное выше предобусловливание принято называть левым, так как домножение на предобусловливатель выполняется слева. Другой метод основан на переходе к системе

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{y} = \mathbf{b}, \tag{2.44}$$

решение которой $\mathbf{y}_*$ связано с точным решением $\mathbf{x}_*$ исходной СЛАУ (2.1) соотношением

$$\mathbf{x}_* = \mathbf{M}^{-1}\mathbf{y}_*. \tag{2.45}$$

Предобусловливание (2.44) реализуется путем двойных умножений вида $\mathbf{z} = \mathbf{A}\left(\mathbf{M}^{-1}\mathbf{q}\right)$. Кроме того, при достижении требуемой точности осуществляется пересчет решения в соответствии с выражением (2.45). Такая схема предобусловливания называется правой.

Последний возможный вариант – двухстороннее предобусловливание, являющееся компромиссным относительно левого и правого. В этом случае имеет место представление $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$, тогда решение вычисляется в виде $\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{z} = \mathbf{M}_1^{-1}\mathbf{b}$ и $\mathbf{x} = \mathbf{M}_2^{-1}\mathbf{z}$.

Способы предобусловливания можно разбить на два вида: явные и неявные. Для представления системы (2.1) с явным предобусловливанием удобно воспользоваться записью (2.43). Как указано выше, предобусловливание может быть введено в схему метода без необходимости явного вычисления матричного произведения. Так, явное предобусловливание требует нахождения матрицы $\mathbf{M}^{-1}$ и умножения матрицы предобусловливания на вектор в

каждой итерации. Для неявного необходимо решать СЛАУ с матрицей **M** в каждой итерации. Наиболее простым является предобусловливание Якоби. Оно заключается в том, что диагональные элементы матриц **M** и **A** совпадают, а внедиагональные элементы матрицы **M** полагаются равными нулю. Рассмотрим эти два вида предобусловливания.

Неявно предобусловленную систему (2.1) удобно представить в виде уравнения

$$\mathbf{MAx} = \mathbf{Mb}. \qquad (2.46)$$

Наиболее работоспособные и часто используемые методы неявного предобусловливания основаны на LU-разложении. Если матрица, подвергающаяся разложению, является разреженной, то результирующие матрицы **L** и **U** являются более плотными. Поэтому применяют неполную факторизацию (неполное разложение). Идея создания таких методов принадлежит Булееву Н.И., который разрабатывал их с 1950-х гг. Впоследствии эти методы неоднократно переоткрывались зарубежными учеными. Неявное предобусловливание намного чаще используется при решении СЛАУ с плотной матрицей. Ограничимся кратким рассмотрением этих методов.

С учетом требования близости матриц положим, что **M** = **A**, и представим ее в виде

$$\mathbf{M} = \mathbf{LU} + \mathbf{R},$$

где **L** и **U** – нижне- и верхнетреугольные матрицы соответственно; **R** – матрица ошибки. Тогда приближенное представление $\mathbf{M} \approx \mathbf{LU}$ и есть неполное LU-разложение матрицы **A**, или коротко ILU. Самое простое ILU(0)-разложение заключается в применении LU-разложения к матрице **A**, но если $a_{ij} = 0$, то сразу полагается $m_{ij}$ ($l_{ij}$ или $u_{ij}$) равным нулю. Если обозначить $NZ(\mathbf{A})$ структуру (портрет) разреженности матрицы **A**, а $NZ(\mathbf{M})$ – матрицы **M**, то очевидно, что данный способ приведет к тому, что их структуры (портреты) совпадут. Чтобы подчеркнуть, что в данной постановке задачи в структуру не вводятся новые ненулевые элементы, такую факторизацию часто называют факторизацией с нулевым заполнением (zero fill-in) или просто ILU(0). Это позволяет

оптимально использовать оперативную память, когда матрица **A** хранится с помощью специальных форматов, учитывающих только ненулевые элементы. Тогда алгоритм (*ikj*-версия) этого разложения можно представить в следующем виде.

**Алгоритм ILU(0)-разложения**

Для $i = 2, 3, \ldots, N$
  Для $k = 1, 2, \ldots, i-1$
    Если $(i, k) \in NZ(\mathrm{A})$
      $m_{ik} = m_{ik} / m_{kk}$
    Для $j = k+1, 2, \ldots, N$ и
      Если $(i, j) \in NZ(\mathrm{A})$
        $m_{ij} = m_{ij} - m_{ik} \cdot m_{kj}$
    Увеличить $j$
  Увеличить $k$
Увеличить $i$

Поскольку полное разложение приводит к заполнению структуры матрицы без каких-либо ограничений (факторизация с бесконечным заполнением), а ILU(0)-разложение – к нулевому заполнению, очевидно, что еще одним вариантом может служить разложение с неким уровнем заполнения. Таким способом является ILU($p$)-разложение. Здесь каждому ненулевому элементу матрицы первоначально присваивается нулевой уровень заполнения, в противном случае уровень принимается равным бесконечности. В ходе вычислений ($m_{ij} = m_{ij} - m_{ik}\, m_{kj}$) после нахождения элемента в позиции $(i, j)$ необходимо провести корректировку его уровня заполнения согласно правилу

$$lev_{ij} = \min\left\{lev_{ij}, lev_{ik} + lev_{kj} + 1\right\}.$$

Отметим, что элементы, имеющие до начала вычислений нулевой уровень, никогда его не меняют. Эти элементы определяют структуру разреженности, совпадающую с исходной. Элементы, получившие во время вычислений уровень $\le p$ (задаваемого значения), расширяют эту структуру. Таким образом, при $p = \infty$ данное разложение вырождается в полное, а если $p = 0$ – в ILU(0).

Другой стратегией расширения структуры разреженности является использование постфильтрации, осуществляемой в два эта-

па и позволяющей контролировать уплотнение структуры (портрета) матрицы. Данное разложение называется неполным LU-разложением с упорождиванием (отбрасыванием), или ILUT.

**Алгоритм ILUT-разложения**

Для $i = 1, \ldots, N$
  $\mathbf{w} = \boldsymbol{a}_{i*}$
  Для $k = 1,\ldots, i-1$ и $w_k \neq 0$
    $w_k = w_k / u_{kk}$
    Применить правило обнуления к $w_k$
    Для $j = k+1,\ldots, N$
      $\mathbf{w} = \mathbf{w} - w_k \cdot \boldsymbol{u}_{k*}$
    Увеличить $j$
  Увеличить $k$
  Применить правило обнуления к строке **w**
  $l_{ij} = w_j$, для $j = 1, \ldots, i-1$
  $u_{ij} = w_j$, для $j = i, \ldots, N$
Увеличить $i$

Приведем правила обнуления, основанные на постфильтрации элементов.

1. В строке 5 алгоритма ILUT-разложения элемент $w_k$ обнуляется, если его значение меньше, чем евклидова норма преобразуемой строки, умноженная на задаваемый допуск обнуления $\tau$.

2. В строке 10 первоначально происходит обнуление элементов строки **w** аналогично правилу пункта 1. На втором этапе происходит сохранение по $p$ элементов в **L** и **U** частях преобразуемой строки в сочетании с диагональными элементами, которые никогда не обнуляются.

Таким образом, пункт 2 позволяет контролировать количество элементов в $i$-й строке. По-существу, параметр $p$ позволяет контролировать затраты машинной памяти, а параметр $\tau$ – вычислительные затраты. Созданы различные модификации пункта 2. Одной из них является сохранение $nu(i)+p$ элементов в **U** части преобразуемой строки и $nl(i)+p$ элементов в **L** части, где $nu(i)$ и $nl(i)$ – число ненулевых элементов в **U** и **L** частях $i$-й строки матрицы **A** соответственно.

Существуют и другие алгоритмы неполного LU-разложения, например разложение ILUC, основанное не на *ikj*-версии LU-разложения, а на так называемой краутовской версии и постфильтрации, описанной в ILUT.

Данный вид предобусловливания исходит из нахождения матрицы $\mathbf{M}^{-1}$. Из него по работоспособности выделяются способы, основанные на приближенном обращении (approximate inverse). Их удобно разделить на три группы.

Способы первой группы появились первыми и базируются на нахождении матрицы **M**, минимизирующей норму Фробениуса матрицы невязки $\mathbf{I} - \mathbf{AM}$. Существует две стратегии вычисления обратной матрицы: первая основана на вычислении обратной матрицы «глобально»; вторая основана на постолбцовом вычислении обратной матрицы. Среди способов этой группы выделяются: SPAI – разреженный приближенный обратный (sparse approximate inverse), использующий постолбцовую стратегию в сочетании с QR-разложением и итерационным уточнением; SPMR – метод минимальных невязок с самопредобусловливанием (Self-preconditioned Minimal Residual), использующий постолбцовую стратегию, реализуемую с помощью итерационного метода минимальных невязок с предобусловливанием.

Вторую группу образуют способы, формирующие матрицу предобусловливания с помощью факторизованных приближенных обратных матриц. Эти способы основаны на неполном обращении треугольных матриц **L** и **U**, т. е. неполном нахождении обратной матрицы. Так, если существует разложение $\mathbf{A} = \mathbf{LDU}$, где **L** и **U** – нижне- и верхнетреугольные матрицы с единичными диагоналями, а **D** – диагональная матрица, то матрица $\mathbf{A}^{-1}$ может быть найдена с помощью разложения $\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{D}^{-1}\mathbf{L}^{-1} = \mathbf{Z}\mathbf{D}^{-1}\mathbf{W}^{T}$, где $\mathbf{Z} = \mathbf{U}^{-1}$ и $\mathbf{W} = \mathbf{L}^{-T}$ – верхнетреугольные матрицы с единичной диагональю. Факторизованные приближенные обратные предобусловливатели могут быть получены вычислением с помощью разреживания матриц $\underline{\mathbf{Z}} \approx \mathbf{Z}$, $\underline{\mathbf{W}} \approx \mathbf{W}$ и $\underline{\mathbf{D}} \approx \mathbf{D}$, тогда $\mathbf{M} = \underline{\mathbf{Z}}\,\underline{\mathbf{D}}^{-1}\underline{\mathbf{W}}^{T} \approx \mathbf{A}^{-1}$. Имеется несколько подходов для вычисления приближенного обращения. Эти методы не требуют нахождения треугольных матриц **L** и **U**, т. е. предобусловливатель вычисляется непосредственно из мат-

рицы **A**. К данному классу относятся методы FSAI, AINV и SAINV.

В третью группу входят методы, вычисляющие обратную матрицу, основываясь на двухшаговом процессе. Первоначально производится неполное LU-разложение (используя стандартные способы, описанные выше), а затем полученные при неполном разложении матрицы приближенно обращаются, например путем решения $2N$ треугольных систем.

### 2.3.6 Предфильтрация

Как было показано выше, для улучшения сходимости итерационного процесса решения разреженных СЛАУ применяется предобусловливание, основанное на использовании структуры (портрета) разреженности матрицы **A**. Таким образом, для формирования предобусловливателя при решении СЛАУ с плотной матрицей сначала необходимо определить или задать некую структуру разреженности матрицы **A**. Процесс, позволяющий сформировать разреженную матрицу $\mathbf{A}_S$ из плотной матрицы **A**, называют предфильтрацией в качестве альтернативы постфильтрации как, например, в методе ILUT. Далее из матрицы $\mathbf{A}_S$ формируется предобусловливатель **M**.

Одним из простейших способов (правил) является выбор заранее известной структуры, например ленточной или блочно-диагональной, т. е. элементы матрицы $\mathbf{A}_S$ совпадают с ленточной (блочно-диагональной) частью матрицы **A** и равны нулю вне данной структуры. Такая предфильтрация характеризуется шириной ленты (размером блока) и, таким образом, основана на позиции элемента в матрице (структурная предфильтрация), а не на его значении. Результаты ее использования при решении плотных СЛАУ можно найти, например, в [24].

В работе [25] приведено несколько методов, специально приспособленных для построения матрицы $\mathbf{M}^{-1}$. Одним из них является выбор $k$ максимальных элементов в каждом столбце, так что в результате в матрице будет $kN$ элементов, не равных нулю. Данный способ связан с многократным поиском, поэтому он

характеризуется значительными временными затратами на предфильтрацию.

Чаще используется динамическое определение структуры разреженности, например с помощью некоторого порога (алгебраическая предфильтрация). Этот метод заключается в задании (нахождении) определенного порога, с помощью которого происходит обнуление малозначащих элементов путем сравнения модуля каждого элемента с данным (полученным) значением.

Порог обнуления в свою очередь может быть получен поразному. Первый способ основан на малости значений матрицы относительно заданного порога ($\varepsilon$):

$$a_{ij}^{s} = a_{ij}, \text{ если } \left|a_{ij}\right| > \varepsilon, \;\; i, j = 1, 2, \ldots, N \;\text{ или }\; i = j. \qquad (2.47)$$

Он является простейшим с точки зрения вычислений и требует наименьшего времени на формирование разреженной матрицы $\mathbf{A}^{s}$, поскольку порог обнуления задается непосредственно, без его вычисления.

Второй способ обнуления основан на нормировке всей матрицы с помощью максимального элемента:

$$a_{ij}^{s} = a_{ij}, \text{ если } \left|a_{ij} \Big/ a^{\max}\right| > \varepsilon, \;\; i, j = 1, 2, \ldots, N \text{ или } i = j, \quad (2.48)$$

где $a^{\max}$ – максимальный элемент матрицы. Понятно, что при использовании данного способа достаточно много времени требуется на поиск наибольшего элемента в матрице.

Способ был предложен для применения явного предобусловливания при решении СЛАУ с плотной матрицей. Первоначально происходило обнуление малозначащих элементов исходной матрицы, т. е. плотную матрицу приводили к разреженному виду. Далее на основе полученной структуры разреженности матрицы формировался предобусловливатель. Таким образом решение СЛАУ с плотной матрицей сводилось к решению разреженной системы итерационным методом с предобусловливанием. Впоследствии, уйдя от разреживания исходной матрицы, исследователи пришли к идее использовать данный подход для формирования предобусловливателя, т. е. при предфильтрации. Он является, пожалуй, самым распространенным.

Следующий способ обнуляет (игнорирует) элементы с помощью бесконечной нормы матрицы:

$$a_{ij}^{s} = a_{ij}, \text{ если } |a_{ij}| > \varepsilon = \|\mathbf{A}\|_{\infty}\tau/N,\ i, j = 1, 2, \ldots, N \text{ или } i = j, \quad (2.49)$$

где $\| \mathbf{A} \|_{\infty} = \max_i \sum_{j=1}^{N} | a_{ij} |$; $\tau$ – допуск обнуления. Очевидно, что значительные временные затраты при использовании данного способа связаны с поиском наибольшей суммы строки.

Способ использовался следующим образом. Первоначально система с плотной матрицей преобразовывалась с помощью вейвлетов, после чего она оставалась достаточно плотной, но основная часть ее элементов имела небольшие абсолютные значения. Обнуление элементов по описанному выше правилу приводило к получению разреженной матрицы СЛАУ. Таким образом, была продемонстрирована возможность приведения СЛАУ с плотной матрицей к разреженному виду и тем самым применения эффективных итерационных алгоритмов, ориентированных на разреженные системы.

В последующем этот подход использовали при решении задачи излучения и он оказался эффективным. Вычисления методом моментов производились на примере излучения цилиндра. Было установлено, что выбор $\tau = 0{,}1–0{,}2$ является оптимальным для минимизации времени решения СЛАУ. Поскольку такое упрощение, дающее ускорение, не всегда обосновано и приемлемо для некоторых классов задач, то исследователи стали использовать это правило обнуления как способ определения структуры разреженности матрицы предобусловливания, т. е. как предфильтрацию.

Еще один способ выбора структуры разреженности основан на нахождении наибольшего элемента в строке

$$a_{ij}^{s} = a_{ij}, \text{ если } | a_{ij} / a_i^{\max} | > \varepsilon,\ i, j = 1, 2, \ldots, N \text{ или } i = j, \quad (2.50)$$

где $a_i^{\max}$ – максимальный элемент в $i$-й строке. Поскольку способ основан на нахождении максимального элемента в каждой строке, порог обнуления является разным для каждой из строк. После того как максимальный элемент найден, происходит процесс, подобный описанному при использовании подхода (2.48), но вместо

всей матрицы сканируется ее строка. Очевидно, что при использовании данного метода значительные затраты времени связаны с поиском максимальных элементов строк.

Последний способ был предложен для решения задач магнитостатики [26]:

$$a_{ij}^{S} = a_{ij}, \text{ если } a_{ij} > \varepsilon = \tau \min\left(|\, a_{ij}\,|, |\, a_{jj}\,|\right),\ i, j = 1, 2, \ldots, N \text{ или } i = j. \tag{2.51}$$

К сожалению, рекомендаций по выбору допуска обнуления в [26] не приведено. Результаты вычислений представлены только при фиксированном значении $\tau$, при котором плотность матрицы $\mathbf{A}_S$ составляет 5 %.

Как видно из рассмотренных способов, с течением времени исследователи пытались найти предфильтрацию, адаптивную к различным задачам. Существуют и другие способы предфильтрации. Одни позволяют представить исходную матрицу суммой нескольких разреженных матриц посредством специальных подходов и в дальнейшем использовать структуру одной из них в качестве структуры матрицы предобусловливания. Другие строятся на основе геометрических или топологических особенностей исследуемой задачи.

Отметим, что алгебраическая предфильтрация может сочетаться как со структурной, так и с динамической. Приведенные способы алгебраической предфильтрации могут быть использованы и при постфильтрации. Таким образом, данный вид предфильтрации является, пожалуй, самым универсальным. Однако трудно сказать, какой способ предпочтительнее с точки зрения как минимизации времени решения СЛАУ, так и стабильности допуска/порога обнуления.

### 2.3.7 Многократное решение СЛАУ

Помимо упомянутых ранее достоинств подпространств Крылова, они позволяют строить эффективные итерационные методы для решения СЛАУ с несколькими правыми частями и неизменной матрицей $\mathbf{AX} = \mathbf{B}$, где $\mathbf{X}$ и $\mathbf{B}$ – матрицы размером $N \times m$, $m << N$. Это уравнение является частным случаем СЛАУ (2.27).

Очевидно, что для решения таких СЛАУ самым простым способом (с точки зрения реализации) будет последовательное решение СЛАУ с каждой правой частью по отдельности. Однако данный подход характеризуется наибольшими вычислительными затратами. Поэтому для решения таких систем разработаны разные блочные версии итерационных методов. Так, известны методы, использующие подпространства вида $Km(\mathbf{R}_0, \mathbf{A})$, где $\mathbf{R}_0$ – обобщенная невязка начального приближения. Назовем только некоторые из них [22]: Bl-CG и Bl-BiCG, Bl-GMRES, Bl-QMR, Bl-BiCGStab, Bl-LSQR, Bl-IDR($s$), Bl-GCROT($m$, $k$), Bl-CMHR. Именно эти методы считаются наиболее подходящими для решения СЛАУ с плотной матрицей. Их особенностью является то, что все правые части должны быть одновременно доступны до начала вычислений. Другой подход основан на выборе опорной (seed) СЛАУ, построении подпространств Крылова для нее и последующем решении остальных систем путем проектирования их невязок на подпространство Крылова. На его базе разработаны методы Seed-GMRES, Seed-EGCR, Seed-QMR, Seed-MINRES, Seed-MEGCR, Seed-BiCGStab и др. (здесь для единообразного представления использованы отличающиеся от оригинальных аббревиатуры названий методов). Этот подход считается эффективным, если правые части близки между собой. Еще один подход (глобальный) основан на формировании и решении тензоризованных систем, например Gl-GMRES и Gl-FOM, Gl-BiCG и Gl-BiCGStab, Gl-CMRH, Gl-SCD, Gl-CGS, Gl-BiCR. Последний способ, разработанный недавно, использует перестановку внутренних циклов (loop-interchanging) [27], что, по сути, является альтернативой последовательному раздельному вычислению с каждой правой частью и позволяет несколько снизить вычислительную сложность одной итерации. Представителем этих методов является Li-BiCG.

Особым случаем является решение последовательности (2.27), когда изменяется матрица, а правая часть или постоянна, или изменяется несущественно. Подобное многократное решение СЛАУ возникает во многих научных и инженерных приложениях: при восстановлении изображений, рекурсивном вычислении

наименьших квадратов, оптимизации, решении нелинейных уравнений, в прикладной статистике и т. д.

Еще раз отметим, что блочные методы применимы, только если все правые части доступны до вычислений. Однако на практике это не всегда выполнимо. Так, часто «новые» матрица и правая часть формируются с использованием предыдущего решения СЛАУ. Для решения последовательности (2.27) с разреженными матрицами (для не электромагнитных задач) разработано несколько подходов, в основном связанных с построением эффективного предобусловливателя. При этом подавляющее число исследователей рассматривает задачу решения последовательности «сдвинутых» (shifted) СЛАУ, т. е. когда от матрицы к матрице изменяются только диагональные элементы:

$$\mathbf{A}_k = \mathbf{A} + \mathrm{diag}\left(\delta_1^k, ..., \delta_N^k\right),\ \delta_i^k \geq 0,\ \ i = 1, \ldots, N,\ k = 1, 2, \ldots, m.$$

Один из подходов основан на перевычислении предобусловливателя для каждой отдельно взятой СЛАУ. Другой подход использует «замороженный» (frozen) предобусловливатель, вычисленный из матрицы первой СЛАУ последовательности с одной правой частью, для решения остальных систем. В [28] рассмотрено использование такого предобусловливателя в сочетании с фиксированным числом (шагов по времени $p$) СЛАУ, после решения которых происходит переформирование предобусловливателя. Определение оптимального значения $p$ с точки зрения минимизации временных затрат на моделирование выполнено эмпирически и находится в диапазоне от 5 до 10. При этом число шагов по времени не выбиралось априори, а определялось динамически во время моделирования до достижения стационарности решения. Очевидно, что при $p = 1$ данный подход эквивалентен первому подходу.

Основная мысль следующего подхода достаточно проста. Повторное использование одного и того же предобусловливателя часто приводит к медленной сходимости итерационного процесса, а повторное вычисление предобусловливателя для каждой новой системы является вычислительно затратным. Поэтому очевидно, что для этих крайних случаев существуют промежуточные аль-

тернативы. Так, необходимо найти возможность обновления предобусловливателя с меньшими затратами, чем на его повторное вычисление. Тогда можно ожидать, что полученный предобусловливатель, хоть и будет менее эффективным, чем заново вычисленный, с точки зрения числа итераций, но общая сложность окажется значительно сниженной. В [29] сформулирован общий подход к вычислению «идеального» предобусловливателя для решения последовательности СЛАУ, не ограничивающийся «сдвинутыми» системами. Для ясности изложения рассмотрим последовательность СЛАУ, где $\mathbf{Ax} = \mathbf{b}$ обозначим первую из них, а $\mathbf{A}^{+}\mathbf{x}^{+} = \mathbf{b}^{+}$ – одну из последующих. В качестве предобусловливателя используем $\mathbf{M} = \mathbf{LDU}$. Очевидно, что матрица разницы (вариации) будет $\underline{\mathbf{A}} = \mathbf{A} - \mathbf{A}^{+}$. В [29] предложено вычисление приближения для «идеального» предобусловливателя вида $\mathbf{M}^{+} = \mathbf{M} - \underline{\mathbf{A}}$, которое является очень затратным. Поэтому для практических вычислений вместо $\mathbf{M} - \underline{\mathbf{A}}$ рассмотрены приближения, основанные на малости $\|\mathbf{L} - \mathbf{I}\|$ и $\|\mathbf{U} - \mathbf{I}\|$, что справедливо, если матрица $\mathbf{A}$ строго диагональная. Такие последовательности СЛАУ часто встречаются при решении нелинейных уравнений. Тогда если матрица $\mathbf{M} - \underline{\mathbf{A}}$ неособенная, то она может быть представлена в виде

$$\mathbf{M} - \underline{\mathbf{A}} = \mathbf{L}(\mathbf{DU} - \mathbf{L}^{-1}\underline{\mathbf{A}}) \approx \mathbf{L}(\mathbf{DU} - \underline{\mathbf{A}}) \approx \mathbf{L}(\mathbf{DU} - \mathrm{triu}(\underline{\mathbf{A}}))$$

или

$$\mathbf{M} - \underline{\mathbf{A}} = (\mathbf{LD} - \underline{\mathbf{A}}\mathbf{U}^{-1})\mathbf{U} \approx (\mathbf{LD} - \underline{\mathbf{A}})\mathbf{U} \approx (\mathbf{LD} - \mathrm{tril}(\underline{\mathbf{A}}))\mathbf{U}.$$

Выбор первого или второго варианта обновления осуществляется с помощью простой оценки малости $\|\mathbf{L} - \mathbf{I}\|$ и $\|\mathbf{U} - \mathbf{I}\|$. Так, если значение $\|\mathbf{L} - \mathbf{I}\|$ меньше, чем $\|\mathbf{U} - \mathbf{I}\|$, то более точным является первый вариант, а в противном случае – второй. Такие обновления называются структурированными (structured update). Указанные требования справедливы при решении нелинейных уравнений методами ньютоновского и бройденовского типа. Упрощенным вариантом обновлений является

$$\mathbf{M} - \underline{\mathbf{A}} \approx \mathrm{diag}(\mathbf{DU} - \underline{\mathbf{A}}).$$

При этом подход в целом является обобщением для обновления симметричных матриц. Известен более общий случай за счет использования трансформаций Гаусса – Жордана. Он разработан

специально для задач, когда вариации существенны по всей матрице **A**, а не только по ее треугольным частям. При этом обновления происходят или периодически, или перед началом решения текущей системы.

Еще один подход сочетает в себе идеи предыдущих. Он состоит в том, что задается диапазон СЛАУ, на котором используется вычисленный для первой СЛАУ из диапазона «замороженный» предобусловливатель. Количество требуемых для ее решения итераций фиксируется ($iter_0$). Если для решения последующей СЛАУ требуемое количество итераций превышает значение $iter_0 + j$, где $j$ – заданный порог, то происходит обновление предобусловливателя.

Последний подход основан на адаптивном использовании информации об уже построенных подпространствах Крылова (recycling) при решении предыдущих систем для обновления предобусловливателя [30]. Отметим, что аналогичная идея использована при разработке итерационных методов, ориентированных на решение «сдвинутых» систем [31].

## Контрольные вопросы и задания

1. Вычислить $(\mathbf{x}, \mathbf{x})$ и $\|\mathbf{x}\|_2$ для вектора $\mathbf{x} = (3, 4, -5)^T$.

2. Вычислить скалярное произведение векторов $\mathbf{x} = (3, 4, -5)^T$ и $\mathbf{y} = (1, 2, i)^T$.

3. Как оценивается обусловленность задачи решения СЛАУ?

4. Найти $\mathrm{cond}_2(\mathbf{A})$ для матрицы $\mathbf{A} = \begin{pmatrix} 5.001 & 5.1 \\ 7.1 & 7.001 \end{pmatrix}$.

5. Найти LU-разложение матрицы $\mathbf{A} = \begin{pmatrix} 4 & 3 & 4 & 6 \\ 2 & 1 & 2 & 4 \\ 3 & 2 & 1 & -1 \\ 5 & 3 & -2 & 2 \end{pmatrix}$.

6. Чему пропорциональны вычислительные затраты прямых и итерационных методов решения СЛАУ?

7. В чем отличие методов Якоби и Гаусса – Зейделя?

8. Разработать программу на языке Octave для обращения матриц с помощью LU-разложения.

9. Разработать программу на языке Octave для решения СЛАУ методом Гаусса – Зейделя.

10. Разработать программу на языке Octave для решения СЛАУ методом BiCGStab.

11. Разработать программу на языке Octave для решения двух СЛАУ, матрицы которых отличаются только элементами одной строки.