Тема 7.
Решение систем линейных алгебраических уравнений

Рассмотрим систему линейных алгебраических уравнений

$$\mathbf{Ax} = \mathbf{b}, \quad det(\mathbf{A}) \neq 0. \tag{1}$$

Так как матрица **A** неособенная ее единственным решением является вектор

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}. \tag{2}$$

До изложения известных методов решения системы (1), как это уже было, последуем полезному совету, данному американским ученым Р.В.Хеммингом читателям своей книги «Численные методы»: «*Прежде, чем решать задачу, подумай, что делать с ее решением*». Предлагается ответить на вопрос: «Как сильно изменяется решение (2) при малой вариации исходных данных (элементов матрицы **A** и вектора **b**)».

Плохая обусловленность матрицы

Численное решение линейных алгебраических систем подвержено влиянию нескольких источников ошибок. Два из них традиционны и очевидны: ограниченность разрядной сетки компьютера и погрешность представления исходных данных. Матрица и вместе с ней система (1) называются *плохо обусловленными,* если малым изменениям элементов **A** отвечают большие изменения элементов $\mathbf{A}^{-1}$ и, следовательно сильные изменения вектора решения (2). Получим количественную характеристику этого явления.

Первоначально будем считать, что матрица **A** известна точно, а вектор **b** – с некоторой погрешностью $\Delta\mathbf{b}$. Тогда система приобретет вид

$$\mathbf{A}\left(\mathbf{x} + \Delta\mathbf{x}\right) = \mathbf{b} + \Delta\mathbf{b}$$

или после вычитания (1) и обращения матрицы:

$$\mathbf{A}\Delta\mathbf{x} = \Delta\mathbf{b}; \qquad \Delta\mathbf{x} = \mathbf{A}^{-1}\Delta\mathbf{b}.$$

Далее при использовании любой нормы матрицы, согласованной с нормой вектора **x** , получаем

$$\left\|\Delta\mathbf{x}\right\| \le \left\|\mathbf{A}^{-1}\right\|\left\|\Delta\mathbf{b}\right\|, \qquad \left\|\mathbf{b}\right\| \le \left\|\mathbf{A}\right\|\left\|\mathbf{x}\right\|.$$

Перемножение этих двух неравенства в предположении, что $\mathbf{b} \neq \mathbf{0}$, и деление на $\left\|\mathbf{b}\right\| \cdot \left\|\mathbf{x}\right\|$ дает

$$\frac{\left\|\Delta\mathbf{x}\right\|}{\left\|\mathbf{x}\right\|} \le cond(\mathbf{A})\frac{\left\|\Delta\mathbf{b}\right\|}{\left\|\mathbf{b}\right\|}, \qquad cond(\mathbf{A}) = \left\|\mathbf{A}\right\|\left\|\mathbf{A}^{-1}\right\| \tag{3}$$

Число $cond(\mathbf{A}) = \left\|\mathbf{A}\right\|\left\|\mathbf{A}^{-1}\right\|$ будем называть *стандартным числом обусловленности*.

Вычисляя норму от обеих частей равенства $\mathbf{A}\cdot\mathbf{A}^{-1} = \mathbf{E}$, имеем $\|\mathbf{A}\|\cdot\|\mathbf{A}^{-1}\| \geq \mathbf{E}$, т.е. $cond(\mathbf{A}) \geq 1$. Равенство (3) допускает простую интерпретацию для практики. Число обусловленности матрицы **A** является верхней границей "усиления" относительной ошибки вектора **b**, т.е. относительное изменение вектора **b** влечет за собой относительное изменение в решении не более чем в *cond*(**A**) раз. Если величина *cond*(**A**) невелика, то говорят о хорошей обусловленности матрицы **A**, в противном случае – о плохой.

На практике плохая обусловленность часто сопровождается малой величиной определителя матрицы *det*(**A**). В связи с этим широко распространено заблуждение, что малость *det*(**A**) всегда сопровождается большим числом обусловленности *cond*(**A**). Однако, это не всегда так. Так, например, для матрицы $\mathbf{A} = \varepsilon\mathbf{E}$ ее определитель $\left(det(\mathbf{A}) = \varepsilon^m\right)$ может быть близок к нулю при больших размерностях m матрицы **A** и сравнительно не малых $\varepsilon < 1$. Однако матрица **A** не является плохо обусловленной и не вызывает проблем с решением (1).

Для иллюстрации рассмотрим систему (1) с параметрами

$$\mathbf{A} = \begin{pmatrix} 1.00 & 0.99 \\ 0.99 & 0.98 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 1.99 \\ 1.97 \end{pmatrix}, \tag{4}$$

точным решением которой являются; $x^{(1)} = 1$, $x^{(2)} = 1$. Небольшое изменение в исходных данных резко изменяет решение:

$$\mathbf{b} + \Delta\mathbf{b} = \begin{pmatrix} 1.989903 \\ 1.970106 \end{pmatrix},\ \Delta\mathbf{b} = \begin{pmatrix} -0.000097 \\ +0.000106 \end{pmatrix},\ \Delta\mathbf{x} = \begin{pmatrix} +2.0000 \\ -2.0203 \end{pmatrix}, \mathbf{x} + \Delta\mathbf{x} = \begin{pmatrix} +3.0000 \\ -1.0203 \end{pmatrix}$$

Непосредственное вычисление оценки из (1) дает *cond*(**A**)~ 40000.

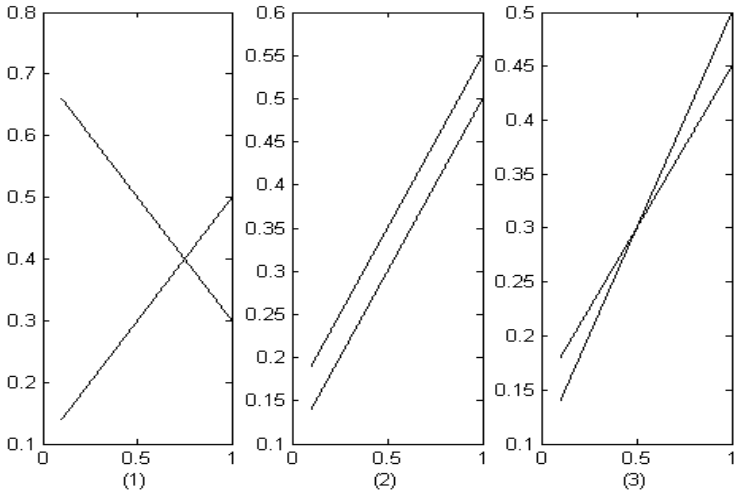

Рис. 1. Геометрическая иллюстрация решения линейной системы

Для этого простого случая есть хорошая геометрическая иллюстрация. Каждому уравнению системы соответствует прямая на плоскости, а точка пересечения этих прямых дает решение системы. Исходным данным (4)

качественно отвечает на рис.1 случай (3): прямые пересекаются под очень острым углом и малейшее изменение исходных коэффициентов **A** или **b** значительно влияет на расположение точки пересечения. Случай (1) демонстрирует хорошую обусловленность, а случай (2) - отсутствие решения при нулевом определителе.

Теперь рассмотрим ситуацию, когда вектор **b** известен точно, а коэффициенты матрицы **A** заданы с погрешностью $\Delta\mathbf{A}$:

$$(\mathbf{A}+\Delta\mathbf{A})(\mathbf{x}+\Delta\mathbf{x})=\mathbf{b}\,.$$

Вычитая из этой формулы равенство (1), получаем:

$$\mathbf{A}\Delta\mathbf{x}=-\Delta\mathbf{A}(\mathbf{x}+\Delta\mathbf{x}),$$

$$\|\Delta\mathbf{x}\|\le\|\mathbf{A}^{-1}\|\cdot\|\Delta\mathbf{A}\|\cdot\|\mathbf{x}+\Delta\mathbf{x}\|,$$

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}+\Delta x\|}\le cond(\mathbf{A})\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}\quad . \qquad (5)$$

И в этом случае *cond*(**A**) ограничивает сверху увеличение относительной .ошибки решения по сравнению с относительной ошибкой исходных данных. Если в примере (4) заменить элемент a_{22} на $a_{22}+\Delta a_{22}$ = 0.9802, то "возмущенное" решение не будет совпадать с исходным даже по знаку ($x^{(1`)}$ = 2.98, $x^{(2)}$= –1.00).

Существуют и другие количественные характеристики плохой обусловленности. Например, таким числом является величина, отражающая разброс спектра собственных значений **A**:

$$k(\mathbf{A})=\frac{|\lambda_k|_{\max}}{|\lambda_k|_{\min}}.$$

При этом легко показать, что $k(\mathbf{A})\le cond(\mathbf{A})$. Действительно, если λ_k собственные значения **A**, то $\frac{1}{\lambda_k}$ собственные значения $\mathbf{A}^{-1}$. Поэтому $\max\limits_k|\lambda_k|\le\|\mathbf{A}\|$, а также $\frac{1}{|\lambda_k|_{\min}}\le\|\mathbf{A}^{-1}\|$, и для плохо обусловленных матриц имеем:

$$cond(\mathbf{A})=\|\mathbf{A}\|\|\mathbf{A}^{-1}\|\ge k(\mathbf{A})=\frac{\max\limits_k|\lambda_k|}{\min\limits_k|\lambda_k|}\gg 1$$

Учитывая этот факт, интересно рассмотреть механизм влияния погрешности в задании элементов **A** на решение (1), анализируя собственные значения **A** и $\mathbf{A}^{-1}$. Предварительно обратимся к двум положениям.

Положение 1. Максимальные по модулю элементы матрицы **A** имеют величину по меньшей мере порядка максимальных по модулю собственных

значений матрицы $|\lambda_k|_{\max}$ (а, может быть, и значительно их превышают!). Действительно, это положение прямо следует из неравенства, связывающего собственные значения и норму матрицы: $|\lambda_k| \le \|\mathbf{A}\|$.

Положение 2. Пусть λ_k - собственные значения матрицы **A**. В "возмущенной" матрице $\mathbf{A}+\Delta\mathbf{A}$ собственные значения могут измениться на величину порядка элементов матрицы возмущений $\Delta\mathbf{A}$.

В качестве примера достаточно обратиться к матрице $\mathbf{A}+\Delta\mathbf{A}=\mathbf{A}+\varepsilon\mathbf{E}$, все собственные значения которой изменились на одну величину: $\Delta\lambda_k=\varepsilon$.

На практике *положение* 2 крайне часто имеет место, хотя можно специально таким образом выполнить достаточно большое возмущение, что в матрицах **A** и $\mathbf{A}+\Delta\mathbf{A}$ собственные значения не изменятся. Например:

$$\mathbf{A}=\begin{pmatrix}1 & -1\\ 2 & 4\end{pmatrix}, \qquad \mathbf{A}+\Delta\mathbf{A}=\begin{pmatrix}5 & 1\\ -6 & 0\end{pmatrix}, \qquad \lambda_1=2,\quad \lambda_2=3.$$

Пусть теперь предельная относительная погрешность задания элементов матрицы ***A*** равна δ. Тогда в соответствии с *положением* 1 максимальные по модулю элементы матрицы $\Delta\mathbf{A}$ по меньшей мере имеют величину порядка $\delta|\lambda_k|_{\max}$. В соответствии с *положением* 2 на эту величину могут измениться все собственные значения возмущенной матрицы.

Если величина δ мала, то максимальные по модулю собственные значения матрицы **A** изменятся незначительно. В то же время, если разброс между λ_k велик, минимальные по модулю собственные значения λ_k могут измениться весьма существенно. Но величина, обратная минимальному по модулю собственному значению матрицы **A**, является максимальным по модулю собственным значением для обратной матрицы $\mathbf{A}^{-1}$, элементы которой претерпят существенные изменения. Вместе с ними изменятся весьма сильно и компоненты вектора решения системы (1).

Описанный эффект будет проявляться тем сильнее, чем больше разброс собственных значений матрицы **A,** т.е., $k(\mathbf{A})=\dfrac{\max\limits_k|\lambda_k|}{\min\limits_k|\lambda_k|}\gg 1$.

Обратим внимание еще на один аспект интерпретации неравенств (3) и (5). Если значение *cond*(**A**) мало (близко к единице), то относительно малое изменение исходных данных *обязательно* приведет лишь к малому изменению решения. Если же значение *cond*(**A**) велико, то малые изменения исходных данных очень часто *могут* привести (но не обязательно приводят!) к большому изменению решения.

В зависимости от числа двоичных разрядов, представляющих вещественные числа в данном компьютере, существует "критическое" число обусловленности. Если, например, *cond*(**A**) = 10^6, то в решении может быть

потеряно шесть десятичных знаков. При длине мантиссы в семь десятичных разрядов это оказывается “вычислительной катастрофой”, в то время как при работе с двойной точностью (15 десятичных разрядов) проблем может не быть. Формализовать это можно следующим образом.

Точность машинной арифметики характеризуется посредством “машинного эпсилон”, т.е. наименьшего числа с плавающей точкой ε, такого, что $1 \oplus \varepsilon > 1$, где $\oplus$ - сложение на компьютере. При записи в оперативную память элементов матрицы **A** предельная относительная погрешность составляет не менее ε. Как следует из приведенных выше утверждений, только этого искажения исходной информации вполне достаточно, чтобы решение не имело ни одного верного знака, если число обусловленности удовлетворяет условию $cond(\mathbf{A}) > 1/\varepsilon$ (разумеется, здесь предполагается, что $cond(\mathbf{A})$ не является слишком завышенной мерой плохой обусловленности!). Последнее замечание не излишне, поскольку при решении системы (1) с диагональной матрицей **A** не возникает проблем, в то время как $cond(\mathbf{A})$ может быть очень большим:

$$\mathbf{A} = \begin{pmatrix} 10^4 & 0 \\ 0 & 10^{-4} \end{pmatrix}, \quad \mathbf{A}^{-1} = \begin{pmatrix} 10^{-4} & 0 \\ 0 & 10^4 \end{pmatrix}, \qquad cond(\mathbf{A}) = 10^8.$$

Все методы решения системы (1) делятся на два класса: *точные* методы и *итерационные* методы.

Метод Гаусса. LU – разложение матрицы. Программы *DECOMP* и *SOLVE*

Различают два больших класса методов решения систем (1): *точные* (или *прямые*) и *итерационные*. Точные методы за конечное число арифметических операций при отсутствии ошибок округления (что эквивалентно бесконечной разрядной сетке) дают точное решение задачи. В ходе применения итерационных методов рождается последовательность векторов, сходящаяся к решению.

В качестве наиболее популярного представителя методов первой группы рассмотрим метод Гаусса исключения неизвестных. Одна из его примитивных модификаций предполагает на первом шаге исключение $x^{(1)}$ с помощью первого уравнения из остальных уравнений. С этой целью первое уравнение умножается на $m_{k1} = -a_{k1}/a_{11}$ и складывается с k-м уравнением и т.д. На втором шаге с помощью преобразованного второго уравнения исключается $x^{(2)}$ из последующих уравнений. После исключения $x^{(n-1)}$ завершается так называемый *прямой ход* метода Гаусса, результатом которого является верхняя треугольная матрица. *Обратный ход* метода Гаусса (гораздо менее трудоемкий) сводится к последовательному получению неизвестных, начиная с последнего уравнения.

Алгоритм в таком виде нуждается в существенном замечании. Нельзя заранее предвидеть, что элемент, стоящий в левом верхнем углу обрабатываемой матрицы, всегда будет отличен от нуля. Если ситуация с нулевым элементом возникнет, то, чтобы избежать деления на нуль, необходимо переставить строки, сделав элемент в этой позиции (ведущий элемент) ненулевым. Более того, желательно избегать не только нулевых, но и относительно малых ведущих элементов. Подтвердим это примером.

$0.100 \cdot 10^{-3} \cdot x^{(1)} + 0.100 \cdot 10^{1} \cdot x^{(2)} = 0.100 \cdot 10^{1}$

$0.100 \cdot 10^{1} \cdot x^{(1)} + 0.100.10^{1} \cdot x^{(2)} = 0.200 \cdot 10^{1}$

Решение с малым ведущим элементом $a_{11}=0.100 \cdot 10^{-3}$ с шестью десятичными знаками таково: $x^{(1)} = 1.00010$, $x^{(2)} = 0.999900$, а с тремя знаками: $x^{(1)} = 0.000$, $x^{(2)} = 1.00$. Очевидно, что произошла "вычислительная катастрофа". Переставив уравнения (теперь $a_{11}=0.100 \cdot 10^{1}$), решим систему с тремя значащими цифрами: $x^{(1)}=1.00$, $x^{(2)}=1.00$. (Рекомендуется самостоятельно проделать вычисления, корректно округляя).

Наиболее известны следующие две стратегии выбора ведущего элемента.

Вариант 1 – *полный* выбор. Здесь на k-ом шаге в качестве ведущего берется наибольший по модулю элемент в неприведенной части матрицы. Затем строки и столбцы переставляются так, чтобы этот элемент поменялся местами с a_{kk}. В этом случае каждый раз осуществляется деление на максимальный элемент, но перестановка столбцов фактически сводится к перенумерации компонент вектора **x.**

Вариант 2 – *частичный* выбор. Здесь на k-ом шаге в качестве ведущего используют наибольший по модулю элемент первого столбца неприведенной части. Затем этот элемент меняют местами с a_{kk}, для чего переставляют только строки, избегая перенумерации компонент вектора **x.**

В ходе многочисленных машинных экспериментов установлено, что, как правило, частичный выбор лишь немного уступает полному в скорости роста ошибок округления. При этом полный выбор гораздо более трудоемок, требует перенумерации переменных при перестановке столбцов и увеличивает время получения решения.

С современной точки зрения метод Гаусса интерпретируется как разложение матрицы системы (1) в произведение двух треугольных матриц (**LU**–разложение). Этот факт отражает следующая теорема, приводимая без доказательства.

Теорема. Пусть $\mathbf{A}^{(k)}$- главные миноры квадратной матрицы **A** порядка $m \times m$ (k=1,2..., m–1). Предположим, что $det(\mathbf{A}^{(k)}) \neq 0$. Тогда существует единственная нижняя треугольная матрица $\mathbf{L} = (l_{ij})$, где $l_{11} = l_{22} = ... = l_{nn} = 1$, и единственная верхняя треугольная матрица $\mathbf{U} = (u_{ij})$, такие, что $\mathbf{L} \cdot \mathbf{U} = \mathbf{A}$. Более того, $\det(\mathbf{A}) = u_{11} \cdot u_{22} \cdot ... \cdot u_{nn}$.

Эта теорема позволяет представить решение (1) как решение двух систем с треугольными матрицами **L** и **U: Ly=b** и **Ux=y.** Решение первой системы с одновременным вычислением **L** и **U** соответствует прямому ходу метода Гаусса, а решение второй системы – обратному ходу. Технологию **LU**-разложения проиллюстрируем на примере системы четвертого порядка без выбора ведущего элемента. Пусть $m_{k1} = -a_{k1}/a_{11}$.(k=2,3,4). Первый шаг прямого хода эквивалентен умножению матрицы **A** и вектора **b** слева на матрицу $\mathbf{M}_1$:

$$\mathbf{M}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ m_{21} & 1 & 0 & 0 \\ m_{31} & 0 & 1 & 0 \\ m_{41} & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{A}_2 = \mathbf{M}_1\mathbf{A}, \qquad \mathbf{b}_2 = \mathbf{M}_1\mathbf{b}.$$

На втором шаге матрица $\mathbf{A}_2$ и вектор $\mathbf{b}_2$ умножаются на матрицу $\mathbf{M}_2$, а на третьем шаге матрица $\mathbf{A}_3 = \mathbf{M}_2\,\mathbf{A}_2$ и вектор $\mathbf{b}_3 = \mathbf{M}_2\mathbf{b}_2$ умножаются на матрицу $\mathbf{M}_3$ ($\mathbf{A}_4 = \mathbf{M}_3 \cdot \mathbf{M}_2 \cdot \mathbf{M}_1 \cdot \mathbf{A}$): $m_{k2} = -a_{k2}^{(2)} / a_{22}^{(2)}$, $m_{k3} = -a_{k3}^{(2)} / a_{33}^{(2)}$,

$$\mathbf{M}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & m_{32} & 1 & 0 \\ 0 & m_{33} & 0 & 1 \end{pmatrix}, \qquad \mathbf{M}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & m_{43} & 1 \end{pmatrix}$$

Согласно построению $\mathbf{A}_4$ есть верхняя треугольная матрица **U**:

$\mathbf{M} = \mathbf{M}_3\mathbf{M}_2\mathbf{M}_1$, $\mathbf{MA} = \mathbf{U}$, $\mathbf{L} = \mathbf{M}^{-1}$, $\mathbf{A} = \mathbf{LU}$, где

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -m_{21} & 1 & 0 & 0 \\ -m_{31} & -m_{32} & 1 & 0 \\ -m_{41} & -m_{42} & -m_{43} & 1 \end{pmatrix}.$$

Здесь учитывалось, что произведение однотипных треугольных матриц это треугольная матрица того же типа, а обратная матрица для треугольной (нижней или верхней) является треугольной того же типа.

Теперь сформулируем ряд условий, которым должна удовлетворять современная программа, реализующая метод Гаусса. Нужно предусмотреть: выбор ведущего элемента, эффективное решение нескольких систем уравнений (1) с одной и той же матрицей **A** и различными векторами **b**, оценку числа обусловленности. Реализованные в большинстве пакетов по линейной алгебре программы представляют собой набор из двух программ. В первой осуществляется **LU**–разложение, а во второй решаются две системы с треугольными матрицами **L** и **U** (**Ly=b** и **Ux=y**). Примером являются написанные на Фортране программы ***DECOMP*** и ***SOLVE*** из [14]. Программы имеют следующие параметры:

DECOMP (***NDIM, N, A, COND, IPVT, WORK***),

SOLVE(***NDIM, N, A, B, IPVT***),

где ***NDIM*** – объявленная в описании строчная размерность массива, в котором располагается матрица **A**;

N – порядок системы уравнений;

A – матрица, подвергающаяся разложению (по окончании работы программы на ее месте располагаются матрицы **L** и **U**),

COND – оценка числа обусловленности;

IPVT – вектор индексов ведущих элементов (размерность ***N***);

WORK– рабочий одномерный массив (размерность ***N***),

B – вектор правых частей системы (1), где по окончании работы программы ***SOLVE*** размещается вектор решения **x.**

Иногда функции ***DECOMP*** и ***SOLVE*** возлагаются на одну программу, имеющую в таком случае управляющий параметр. В ряде случаев предлагается совокупность разнообразных программ для решения систем (1) с различными матрицами (общего вида, ленточными, трехдиагональными, положительно определенными, симметрическими и пр.).

В заключение оценим число арифметических операций в методе Гаусса. На каждом шаге исключения мы встречаемся с операциями деления и умножения-вычитания. Возьмем за единицу измерения операцию именно такого типа. На k-ом шаге в одной строке выполняется одно деление и k умножений–вычитаний. Тогда для всех k–1 строк имеем: $(k+1)(k-1) = k^2-1$ операций. В прямом ходе Гаусса таких шагов m. В итоге получаем:

$$\sum_{k=1}^{m}\left(k^2-1\right)=\sum_{k=1}^{m}k^2-m=\frac{2m^3+3m^2-5m}{6}.$$

При больших значениях m хорошим приближением для числа операций будет $m^3/3$. Для обратного хода нужно на порядок меньше операций (одно деление и k-1 умножение-вычитание при вычислении $x^{(k)}$, что для всех компонент дает величину $\sum_{k=1}^{m}k=\frac{m^2+m}{2}$). Для сравнения: в формуле Крамера требуется выполнить $m!\left(m^2-1\right)$ операций. (Вычисление одного определителя m-ого порядка требует $m!(m-1)$ yмножений, а всего нужно вычислить $m+1$ определитель, значит, всего $m!\left(m^2-1\right)$**.** Если m=10, то для **LU**-разложения число операций $10^3/3$, а для формул Крамера $10!*99\cong 3*10^8$.)

Итерационные методы

Итерационные методы (еще одно название – *методы последовательных приближений*) дают возможность для системы (1) **Ax=b** строить последовательность векторов $\mathbf{x}_0$,$\mathbf{x}_1$, ...,$\mathbf{x}_n$, ... пределом которой должно быть точное решение **x*:**

$$\mathbf{x}^* = \lim_{n\to\infty}\mathbf{x}_n . \qquad (6)$$

На практике построение последовательности обрывается, как только достигается желаемая точность. Чаще всего для достаточно малого значения $\varepsilon>0$ контролируется выполнение оценки $\left|\mathbf{x}^* - \mathbf{x}_n\right| < \varepsilon$. Метод последовательных приближений может быть построен, например, по следующей схеме. Эквивалентными преобразованиями приведем систему (1) к виду

$$\mathbf{x} = \mathbf{C}\mathbf{x} + \mathbf{d}. \tag{7}$$

Под эквивалентными преобразованиями будем понимать преобразования, сохраняющие решение системы (т.е. решения (1) и (7) совпадают). Точное решение $\mathbf{x}^*$ системы (7) имеет вид

$$\mathbf{x}^* = (\mathbf{E} - \mathbf{C})^{-1}\mathbf{d} \tag{8}$$

Вместо (7) будем решать систему разностных уравнений (9)

$$\mathbf{x}_{n+1} = \mathbf{C}\mathbf{x}_n + \mathbf{d} \tag{9}$$

пошаговым методом. При этом необходимо решить целый ряд вопросов. Сходится ли итерационный процесс (9)? Если сходится, что является пределом последовательности, и какова скорость сходимости?

Ранее было показано, что решение системы (9) записывается в виде

$$\mathbf{x}_n = \mathbf{C}^n\mathbf{x}_0 + \left(\mathbf{E} - \mathbf{C}^n\right)(\mathbf{E} - \mathbf{C})^{-1}\mathbf{d} \tag{10}$$

Вычитая из (10) точное решение (8), получаем

$$\mathbf{x}_n - \mathbf{x}^* = \mathbf{C}^n\mathbf{x}_0 - \mathbf{C}^n(\mathbf{E} - \mathbf{C})^{-1}\mathbf{d} = \mathbf{C}^n\left(\mathbf{x}_0 - \mathbf{x}^*\right) \tag{11}$$

Чтобы обеспечить условие сходимости (6), все элементы матрицы $\mathbf{C}^n$ должны стремиться к нулю при $n \to \infty$. Для этого, в свою очередь, необходимо и достаточно, чтобы *все* собственные значения матрицы $\mathbf{C}$ были бы по модулю меньше единицы

$$|\lambda_k| < 1. \tag{12}$$

Поскольку нахождение всех собственных значений доставляет значительные трудности, вместо условия (12) можно использовать достаточное условие сходимости

$$\|\mathbf{C}\| < 1, \tag{13}$$

которое справедливо для любой канонической нормы.

Количество итераций по формуле (9) будет тем меньше, чем меньше по модулю собственные значения матрицы $\mathbf{C}$ и чем ближе к $\mathbf{x}^*$ выбрано начальное приближение $\mathbf{x}_0$. На практике при реализации на компьютере процесс (9) прерывается либо заданием максимального числа итераций, либо условием $\|\mathbf{x}_{n+1} - \mathbf{x}_n\| < \varepsilon$. Таким образом, основным неформальным моментом является такое приведение системы (1) к виду (7), чтобы выполнялось условие (12). В общем случае универсальный способ такого перехода с

малой трудоемкостью отсутствует, и поэтому часто используется специфика решаемой задачи. Рассмотрим следующий пример.

Пусть диагональные элементы матрицы $\mathbf{A}$ в (1) значительно превышают по модулю остальные элементы в соответствующих строках. Разделим каждое уравнение на соответствующий диагональный элемент и получим

$$\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{b}}, \qquad \mathbf{x} = \left(\mathbf{E} - \tilde{\mathbf{A}}\right)\mathbf{x} + \tilde{\mathbf{b}}.$$

На главной диагонали у матрицы $\tilde{\mathbf{A}}$ стоят единицы, а у матрицы $\left(\mathbf{E} - \tilde{\mathbf{A}}\right)$ расположены нули. Вне главной диагонали у обеих матриц находятся малые по модулю элементы, что позволяет, выбрав $\mathbf{C} = \mathbf{E} - \tilde{\mathbf{A}}$, легко обеспечить условие (13) и быструю сходимость итерационного процесса (9).

Остановимся на сравнении прямых и итерационных методов. Чаще всего преимущества итерационных методов сказываются в следующих ситуациях.

1) Имеется хорошее начальное приближение $\mathbf{x}_0$ к точному решению (1), что обеспечит сравнительно малое число итераций в (9).

2) Удалось получить матрицу $\mathbf{C}$ в (9) с весьма малыми по модулю собственными значениями, что гарантирует высокую скорость сходимости итерационного процесса.

3). Матрица $\mathbf{A}$ является разреженной, и в оперативной памяти компьютера хранится относительно небольшое число ее ненулевых элементов. Использование формулы (9) требует лишь написания специальной программы умножения разреженной матрицы на вектор, в то время как в процессе реализации точных методов и преобразования матрицы $\mathbf{A}$ может происходить заметное увеличение ненулевых элементов.

Один из способов хранения ненулевых элементов – связный список, где каждый элемент какой-то строки это запись со следующими полями: номер столбца, значение элемента, ссылка на следующий ненулевой элемент строки. Дополнительно строится вектор, где каждый элемент это ссылка на первый ненулевой элемент соответствующей строки. Преимущество итерационных методов в ряде случаев разреженных матриц над точными методами иллюстрирует следующий пример, где ненулевые элементы матрицы, обозначенные звездочкой, расположены следующим образом:

$$\begin{pmatrix} * & * & * & * & * & * & * & * \\ * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & 0 & * & 0 & 0 & 0 & 0 & 0 \\ * & 0 & 0 & * & 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 \\ * & 0 & 0 & 0 & 0 & * & 0 & 0 \\ * & 0 & 0 & 0 & 0 & 0 & * & 0 \\ * & 0 & 0 & 0 & 0 & 0 & 0 & * \end{pmatrix}.$$

Пусть элемент в позиции (1,1) самый большой по модулю. Деление на него и исключение $x^{(1)}$ с помощью первого уравнения в методе Гаусса приводит уже на первом шаге к полному заполнению матрицы и резкому возрастанию числа ненулевых элементов.