Методы решения систем линейных алгебраических уравнений

Постановка задачи

Необходимо решить уравнение $A\mathbf{x} - \mathbf{b} = 0$, где

$A \in \mathbb{R}_{n \times n}$ – квадратная вещественная матрица

$\mathrm{rank}(A) = n$ – матрица также невырожденная

Частные случаи

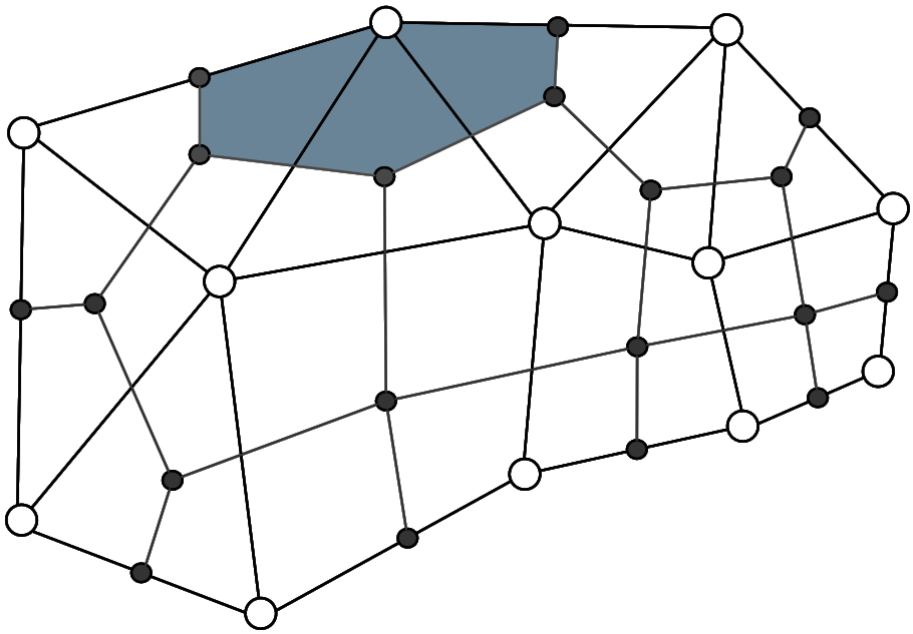
- Симметричная матрица $A^{\mathrm{T}} = A$
- Положительно определенная матрица: $\mathbf{x}^{\mathbf{T}} A\mathbf{x} > 0, \forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0$

Задачи, в которых появляются СЛАУ

Моделирование жидкостей

$$\underbrace{\frac{\partial}{\partial t}(\rho\phi)}_{unsteady\ term} + \underbrace{\nabla \cdot (\rho \mathbf{v} \phi)}_{convection\ term} = \underbrace{\nabla \cdot \left(\Gamma^{\phi} \nabla \phi\right)}_{diffusion\ term} + \underbrace{Q^{\phi}}_{source\ term}$$

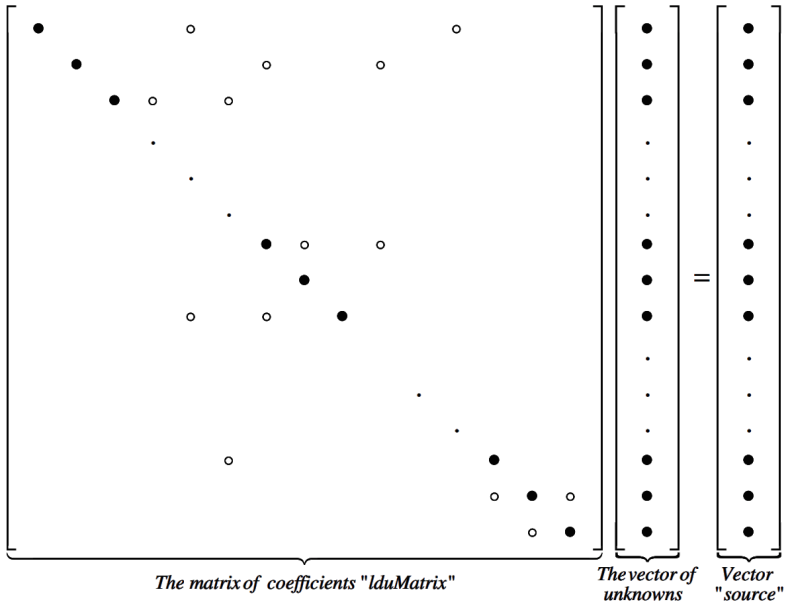
Это математическое гидродинамическое уравнение, которое должно выполняться в любой точке некоторого пространства, где течет некоторая идеальная математическая жидкость. Чтобы с такими уравнениями работать на компьютере, пространство разбивается:



В таком случае речь идет уже о выполнении не этих уравнений в каждой точке, а о выполнении некоторой дискретизированной формы этого уравнения в каждой вершине этой сетки, или в каждом объеме.

В случае сетки, которая приведена на рисунке выше, неизвестными будут числовые характеристики жидкости в вершинах сетки (например, давление). Уравнений должно быть столько же, сколько неизвестных, и уравниваются другие числовые характеристики: потоки, проходящие между границами некоторых контрольных «подобъемов».

В итоге, получается система линейных уравнений, которую нужно решить на каждом шаге симуляции.



Матрица квадратная, причем n равно количеству вершин в сетке, или количеству ячеек (иногда удвоенному или утроенному их количеству). Элементы матрицы численно характеризуют значения переменных в одних регионах сетки на поток жидкости в других регионах. Матрица получается разреженной, потому что давление жидкости в одном месте практически не влияет на поток жидкости в другом месте.

Разреженная матрица характеризуется большим количеством нулей, для них есть специализированные методы хранения и работы. Стоит отметить, что в этой задаче матрица не симметричная

Решение систем нелинейных уравнений

$$\mathbf{f}(\mathbf{x}) = 0, \quad \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$$

Функцию можно разложить в ряд Тейлора в окрестности точки $\mathbf{x}$:

$$\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \nabla\mathbf{f}(\mathbf{x})\Delta\mathbf{x} + O(||\Delta\mathbf{x}||^2)$$

Здесь $\nabla\mathbf{f}(\mathbf{x})$ - градиент функции:

$$\nabla\mathbf{f} = \begin{bmatrix} \dfrac{\partial\mathbf{f}}{\partial x_1} & \cdots & \dfrac{\partial\mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \cdots & \dfrac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Поскольку функция n-значная и $\mathbf{x}$ – вектор длины n, то матрица получается квадратной. Эту матрицу еще называют *якобиан*.

Метод Ньютона для функции многих переменных

$$\nabla \mathbf{f}(\mathbf{x}^{(n)})\ (\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}) + \mathbf{f}(\mathbf{x}^{(n)}) = \mathbf{0}$$

Задача - получить $\mathbf{x}^{(n+1)}$ - приближение корня на следующем шаге. Полученное уравнение как раз представлено в формате:

$A\mathbf{x} + \mathbf{b} = 0$

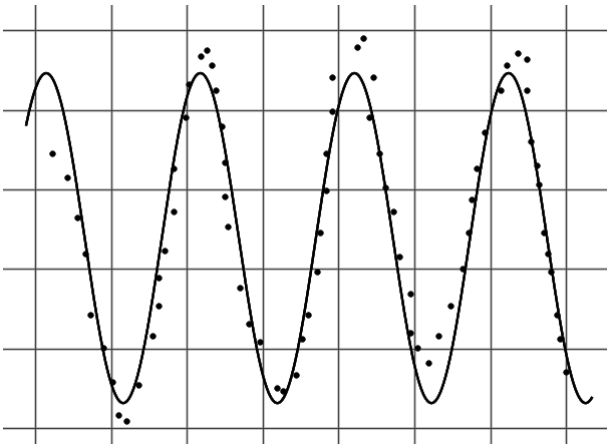
Несложно заметить, что если $n = 1$, то уравнение приходит методу Ньютона функции от одной переменной:

$$x^{(n+1)} = x^{(n)} - \frac{f(x)}{f'(x)}$$

В случае, если якобиан у функции везде одинаковый (то есть функция является линейной), метод Ньютона для функции многих переменных может сойтись за один шаг.

Нелинейный метод наименьших квадратов

Рассмотрим пример нахождения корня нелинейной функции, который проистекает из необходимости найти минимум другой нелинейной функции. Допустим, мы строим модель какого-то процесса или явления, и необходимо подогнать параметры модели с тем, чтобы сократить по возможности невязки между экспериментальными данными (точки на графике) и модельными (кривая).



Невязки - разности значений модели и наблюдений в точках, а минимизируется сумма квадратов этих разностей.

$$\begin{aligned} \arg\min_{\mathbf{x}} S(\mathbf{x}) &= ? \\ S(\mathbf{x}) &= \sum_{j=1}^{m} r_j^2(\mathbf{x}) \end{aligned}$$

Необходимо найти такой вектор параметров модели $\mathbf{x}$, при котором сумма квадратов невязок будет минимальна. Мы считаем, что этот минимум один. Если функция имеет минимум и она гладкая, то в этой точке ее градиент равен 0.

$$\nabla S(\mathbf{x}) = \mathbf{0} \Rightarrow f_i(\mathbf{x}) = \sum_{j=1}^{m} \frac{\partial r_j(\mathbf{x})}{\partial x_i} r_j(\mathbf{x}) = 0, \quad i = 1..n$$

Таким образом, получается система n уравнений, которую необходимо решить. Для удобства введем матрицу A:

$$A = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix}$$

Тогда получаем матричную форму записи:

$$A^{\mathrm{T}}\mathbf{r} = 0$$

Стоит заметить, что матрица не квадратная: как правило, число наблюдений больше, чем число параметров.

Чтобы применить метод Ньютона, надо посчитать градиент функции f:

$$\frac{\partial f_i(\mathbf{x})}{\partial x_k} = \frac{\partial r_i(\mathbf{x})}{\partial x_i}\frac{\partial r_i(\mathbf{x})}{\partial x_k} + r_k\frac{\partial^2 r_k(\mathbf{x})}{\partial x_i \partial x_k}$$

Здесь можно отбросить второй член суммы, и тогда аппроксимация якобиана:

$$\nabla \mathbf{f} \approx A^{\mathrm{T}}A$$

Остается записать уравнение метода Ньютона:

$$A^{\mathrm{T}}A\left(\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}\right) + A^{\mathrm{T}}\mathbf{r}(\mathbf{x}^{(n)}) = \mathbf{0}$$

Классификация методов решения СЛАУ

- Прямые методы
  - LU-разложение (A не обязательно квадратная)
  - QR-разложение (A не обязательно квадратная)
  - LDL-разложение (A симметричная)
  - Разложение Холецкого (A симметричная, положительно определенная)
  - Мультифронтальное LU-разложение
  - Диагональные методы
    - Метод прогонки (A трехдиагональная)
    - SPIKE (A ленточная)
- Итеративные методы
  - Методы неподвижной точки
    - Метод Гаусса-Зейделя (основан на принципе сжимающего отображения)
    - Метод релаксации (обобщение метода Гаусса-Зейделя)
  - Методы, основанные на подпространствах Крылова
    - Метод сопряженных градиентов (A симметричная, положительно определенная)
    - Стабилизированный метод бисопряженных градиентов

Прямые методы отличаются тем, что выполняют фиксированное количество операций и выдают результат с максимальной доступной для данного метода точностью

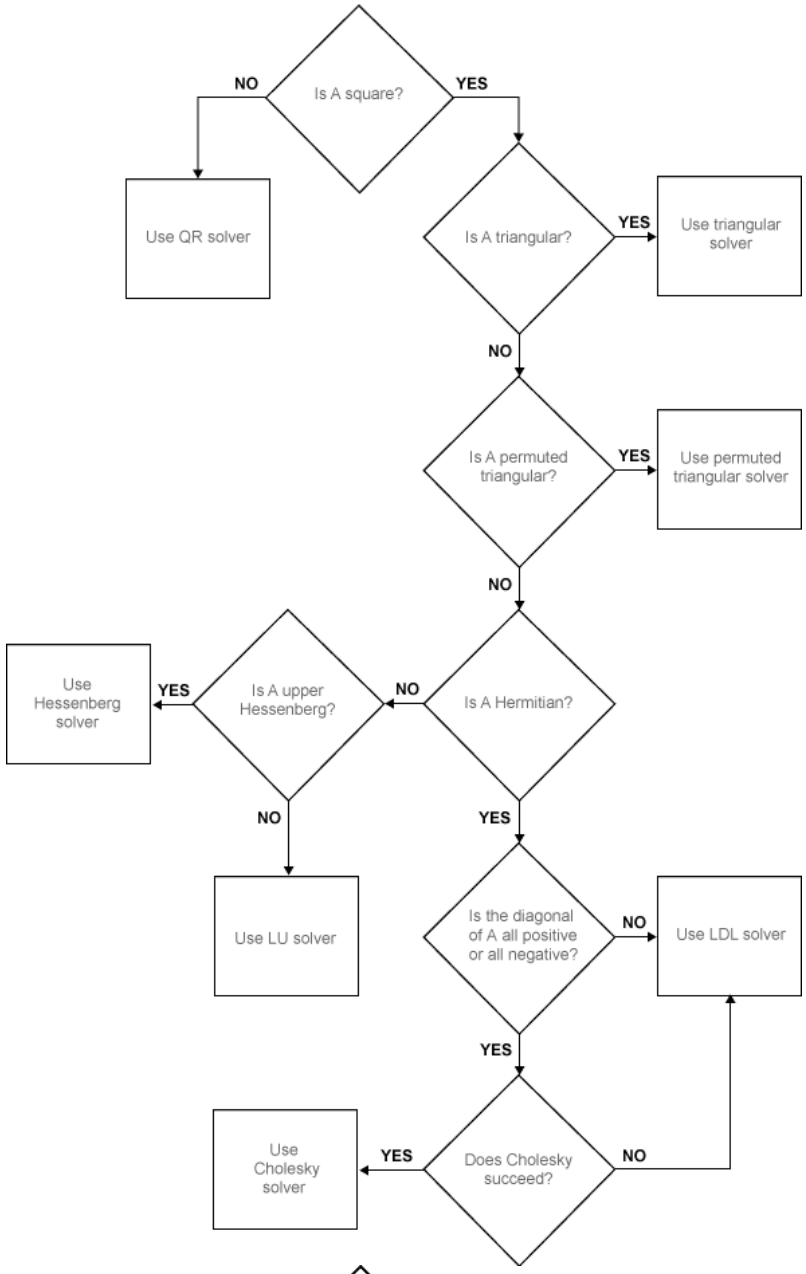
Итеративные методы не имеют фиксированного количества времени выполнения, потому что оно зависит от количества итераций. С каждой итерацией значение x все ближе и ближе к искомому x^*.

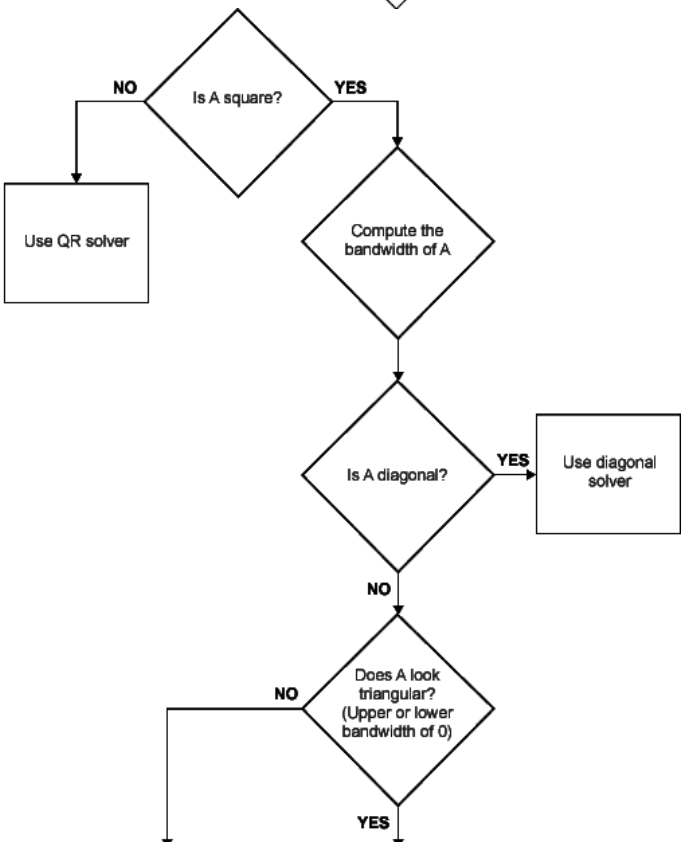
Итеративные методы зачастую более эффективны и часто применяются для решения систем с очень большим числом неизвестных (особенно если их миллион или больше)

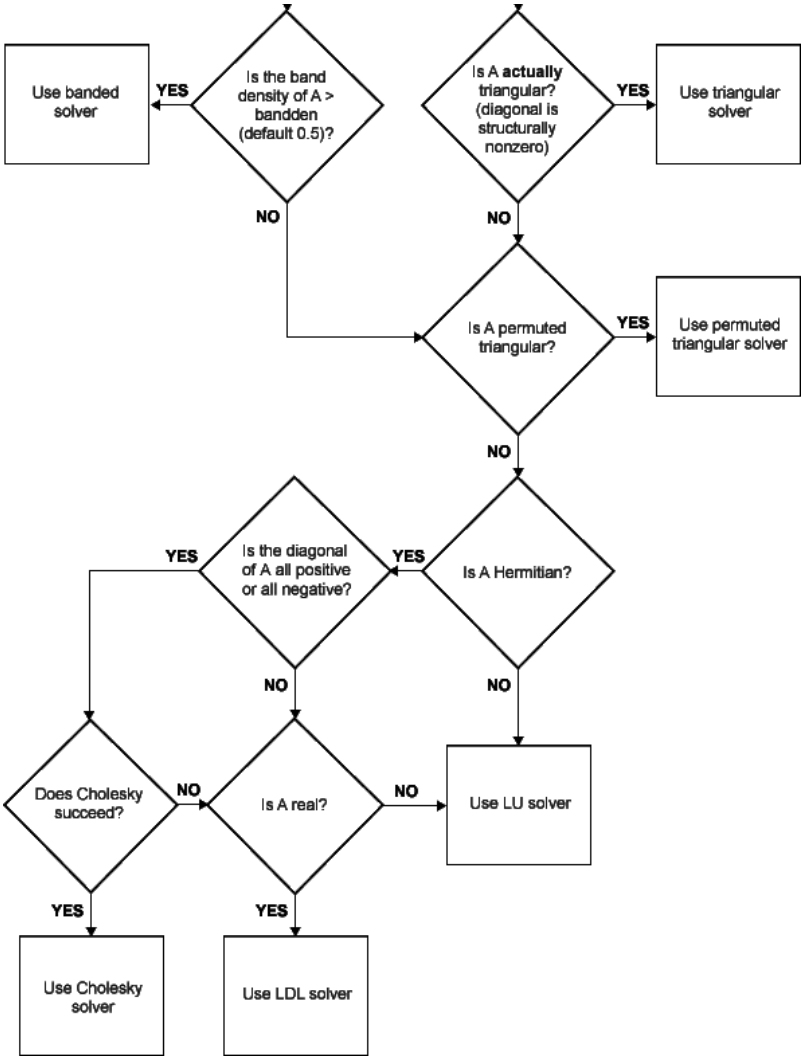
Иллюстрация многообразия применяемых методов

Предположим, пользователь пишет в Mathlab: `x = A \ b`

Документация представляет собой разветвленную систему проверок, причем из двух деревьев.


Is A square?
NO
YES
Use QR solver
Is A triangular?
YES
Use triangular solver
NO
Is A permuted triangular?
YES
Use permuted triangular solver
NO
Is A Hermitian?
NO
Is A upper Hessenberg?
YES
Use Hessenberg solver
NO
Use LU solver
YES
Is the diagonal of A all positive or all negative?
NO
Use LDL solver
YES
Does Cholesky succeed?
NO
YES
Use Cholesky solver


Is A square?
NO
YES
Use QR solver
Compute the bandwidth of A
Is A diagonal?
YES
Use diagonal solver
NO
Does A look triangular? (Upper or lower bandwidth of 0)
NO
YES



> Источник изображений: Solve systems of linear equations Ax = B for x - MATLAB mldivide \ (mathworks.com)

Первое дерево работает в случае, если `x` – неразреженная матрица, а во втором случае – если разреженная.

В каждом случае проверяется:

- Является ли матрица верхней треугольной / нижней треугольной
- Является ли диагональной / ленточной
- Является ли симметричной

И так далее...

И в зависимости от того, какая это матрица, выбирается тот или иной специализированный метод.

LU-разложение

Метод работает в два этапа.

На первом этапе матрица A раскладывается следующим образом:

$$A = LU = \begin{bmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ l_{n1} & \dots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & u_{nn} \end{bmatrix}$$

На втором этапе мы разбиваем задачу решения исходной системы на два последовательных этапа:

$$LU\mathbf{x} = \mathbf{b} \;\rightarrow\; L\mathbf{y} = \mathbf{b},\; U\mathbf{x} = \mathbf{y}$$

Такое представление полезно тем, что системы по отдельности решаются проще.

- Решение первой системы

$$\begin{bmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ l_{n1} & \dots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\begin{aligned} y_1 &= b_1 \\ y_2 &= b_2 - y_1 l_{21} \\ &\vdots \\ y_n &= b_n - \textstyle\sum_{i=1}^{n-1} y_i l_{ni} \end{aligned}$$

- Решение второй системы

$$\begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\begin{aligned} x_1 &= (y_1 - \textstyle\sum_{i=2}^{n} x_i u_{1i})/u_{11} \\ &\vdots \\ x_{n-1} &= (y_{n-1} - x_n u_{n-1,n})/u_{n-1,n-1} \\ x_n &= y_n/u_{nn} \end{aligned}$$

Как происходит само LU-разложение?

Алгоритм итеративный. Используем нотацию не совсем математическую: далее знак $\leftarrow$ означает присвоение переменной значения.

Начало: $L \leftarrow I,\; U \leftarrow A$

Шаг i:

$$u_{jk} \leftarrow u_{jk} - u_{ik}u_{ji}/u_{ii}, \quad j = (i+1)..n,\; k = i..n$$

$$l_{ji} \leftarrow u_{ji}/u_{ii}, \quad j = (i+1)..n$$

$$\begin{bmatrix} 1 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ l_{i1} & \dots & 1 & \dots & 0 \\ l_{(i+1)1} & \dots & u_{(i+1)i}/u_{ii} & \dots & 0 \\ \vdots & & \vdots & & \vdots \\ l_{n1} & \dots & u_{ni}/u_{ii} & \dots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & \dots & u_{1i} & \dots & u_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & u_{ii} & \dots & u_{in} \\ 0 & \dots & u_{(i+1)i} & \dots & u_{(i+1)n} \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & u_{ni} & \dots & u_{nn} \end{bmatrix} \begin{matrix} \\ \\ \\ \times - u_{(i+1)i}/u_{ii} \\ \vdots \\ \times - u_{ni}/u_{ii} \end{matrix}$$

На практике при реализации матрицы L и U можно сложить в одну матрицу (так как единицы на диагонали матрицы L будут всегда, их необязательно держать в памяти). Более того, матрицы можно хранить в том же месте, где хранилась матрица A.

Схема с замещением, шаг i:

$$\begin{array}{lcll} a_{jk} & \leftarrow & a_{jk} - a_{ik}a_{ji}/a_{ii}, & j = (i+1)..n,\ k = (i+1)..n \\ a_{ji} & \leftarrow & a_{ji}/a_{ii}, & j = (i+1)..n \end{array}$$

Также в алгоритме содержится деление. Во-первых, знаменатели могут быть равны 0. Для гарантированной работы алгоритма матрица A должна иметь невырожденные главные миноры.

LU-разложение с перестановками (LUP)

Пусть есть некоторая перестановка σ:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}$$

Тогда матрица этой перестановки P_σ:

$$P_\sigma = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

- PA - (умножение слева) перестановка строк
- AP - (умножение справа) перестановка столбцов

LU-разложение с частичным выбором: $PA = LU$ (работает с любой невырожденной A)

$$\begin{bmatrix} u_{11} & \dots & u_{1i} & \dots & u_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & u_{ii} & \dots & u_{in} \\ 0 & \dots & u_{(i+1)i} & \dots & u_{(i+1)n} \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & u_{ni} & \dots & u_{nn} \end{bmatrix}$$

Если $u_{ii} = 0$, то можно переставить местами строки. В таком случае меняются столбцы в матрице P, чтобы инвариант сохранялся. Изначально матрица P является единичной матрицей. Таким образом решается проблема деления на 0.

Итак, решим систему уравнений с разложением $PA = LU$. Все так же разбиваем систему на две:

$$PA\mathbf{x} = P\mathbf{b} \rightarrow L\mathbf{y} = P\mathbf{b},\ U\mathbf{x} = \mathbf{y}$$

LU-разложение с полным выбором: $PAQ = LU$

- P отвечает за перестановку строк
- Q отвечает за перестановку столбцов

$$PAQQ^T\mathbf{x} = P\mathbf{b} \rightarrow L\mathbf{y} = P\mathbf{b},\ U\mathbf{z} = \mathbf{y},\ \mathbf{x} = Q^T\mathbf{z}$$

Отметим, что матрица перестановки ортогональна, поэтому $QQ^T = I$

Разложение Холецкого

Существует для симметричных положительно определенных матриц. Для таких матриц существует разложение вида

$$A = LL^T, \quad L = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & & \ddots & \vdots \\ l_{n1} & \dots & l_{n,n-1} & l_{nn} \end{bmatrix}$$

Если такое разложение существует, то можно представить исходную систему в виде двух:

$$A\mathbf{x} = \mathbf{b} \to L\mathbf{y} = \mathbf{b}, \; L^T\mathbf{x} = \mathbf{y}$$

Так как матрица L – нижняя треугольная, то обе системы решаются последовательным исключением неизвестных.

Алгоритм разложения проще, чем алгоритмы LU разложения, основан на рекурсивных отношениях, которые связывают A и L:

$$\begin{aligned} l_{11} &= \sqrt{a_{11}} \\ l_{j1} &= a_{j1}/l_{11}, \quad j = 2..n \\ l_{ii} &= \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \quad i = 2..n \\ l_{ji} &= \left(a_{ji} - \sum_{p=1}^{i-1} l_{ip}l_{jp}\right)/l_{ii}, \quad i = 2..(n-1), \; j = (i+1)..n \end{aligned}$$

Так как матрица положительно определенная, то элементы под корнем не могут быть отрицательными.

Алгоритм быстрее, чем LU (асимптотически за то же кубическое время, но число операций меньше), численно устойчив, перестановки не требуются.

В методе наименьших квадратов также используются симметричные положительно определенные матрицы, в связи с чем разложение Холецкого часто используется в задачах минимизации, оптимизации, подгонки параметров.

Метод Гаусса-Зейделя

Если предыдущие методы были прямые, то метод Гаусса-Зейделя является итеративным.

Матрица A выражается как сумма нижней треугольной и строго верхней треугольной:

$$A = L + U = \begin{bmatrix} a_{11} & 0 & \dots & 0 & 0 \\ a_{21} & a_{22} & \dots & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n-1} & 0 \\ a_{n,1} & a_{n2} & \dots & a_{n,n-1} & a_{nn} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & \dots & a_{1,n-1} & a_{1n} \\ 0 & 0 & \dots & a_{2,n-1} & a_{2n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1,n} \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

Система уравнений приводится к виду:

$$A\mathbf{x} = \mathbf{b} \to L\mathbf{x} = \mathbf{b} - U\mathbf{x}$$

Пусть есть начальное приближение $x^{(0)}$, тогда итеративная схема выглядит следующим образом:

$$
\begin{aligned}
L\mathbf{x}^{(1)} &= \mathbf{b} - U\mathbf{x}^{(0)} \\
L\mathbf{x}^{(2)} &= \mathbf{b} - U\mathbf{x}^{(1)} \\
L\mathbf{x}^{(3)} &= \mathbf{b} - U\mathbf{x}^{(2)} \\
&\dots
\end{aligned}
$$

Последовательность обязательно сходится, когда отображение является сжимающим, то есть:

$$||L^{-1}U|| < 1$$

Каждый шаг метода Гаусса-Зейделя делает количество операций, пропорциональное n^2. Таким образом, если шагов не очень много, метод будет работать эффективнее вышерассмотренных прямых методов.

Метод сопряженных градиентов

Итеративный метод, условие – матрица A симметричная положительно определенная.

В методе задача рассматривается не как задача уравнения, а как задача минимизации квадратичной формы:

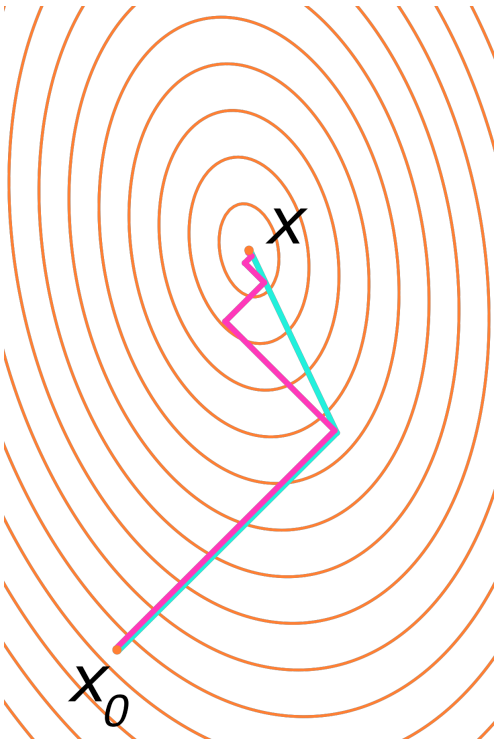
$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$$

f имеет единственный локальный минимум (при симм. полож. опред. A)

$$\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$$

Общая идея метода

Рассмотрим ход итерации в методе сопряженных градиентов как спуск некоторой «воронки» от точки x_0 к минимуму x.



Вычисляется градиент в x_0:

$$\mathbf{r}^{(0)} = A\mathbf{x}^{(0)} - \mathbf{b}$$

Чтобы идти к минимуму функции нужно двигаться в противоположную от него сторону:

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \alpha_0 \mathbf{r}^{(0)}$$

Здесь α_0 – это некоторый шаг, определяющий расстояние, которое нужно пройти в противоположную сторону от градиента. Метод должен двигаться в эту сторону до тех пор, пока функция не начнет возрастать. Выбор такого шага – решение минимизации функции относительно α_0.

$$\begin{aligned} f(\mathbf{x}^{(1)}) &= \tfrac{1}{2}(\mathbf{x}^{(0)} - \alpha_0 \mathbf{r}^{(0)})^T A(\mathbf{x}^{(0)} - \alpha_0 \mathbf{r}^{(0)}) - (\mathbf{x}^{(0)} - \alpha_0 \mathbf{r}^{(0)})^T \mathbf{b} \\ &= \tfrac{1}{2}\left(\mathbf{x^{(0)}}^T A \mathbf{x}^{(0)} - 2\alpha_0 \mathbf{r^{(0)}}^T A \mathbf{x}^{(0)} + \alpha_0^2 \mathbf{r^{(0)}}^T A \mathbf{r}^{(0)}\right) - \mathbf{x^{(0)}}^T \mathbf{b} + \alpha_0 \mathbf{r^{(0)}}^T \mathbf{b} \end{aligned}$$

$$\frac{\mathrm{d}f}{\mathrm{d}\alpha_0} = \alpha_0 \mathbf{r^{(0)}}^T A \mathbf{r}^{(0)} - \mathbf{r^{(0)}}^T (A\mathbf{x}^{(0)} - \mathrm{b})$$

$$\alpha_0 = \frac{\mathbf{r^{(0)}}^T (A\mathbf{x}^{(0)} - \mathrm{b})}{\mathbf{r^{(0)}}^T A \mathbf{r}^{(0)}}$$

Таким образом за один шаг можно существенно приблизиться к минимуму за счет оптимального выбора α_0. Если такую же схему применять на дальнейших шагах, то получился бы метод градиентного спуска (розовый на графике), но метод сопряженных градиентов (бирюзовый на графике) работает иначе и работает гораздо быстрее.

Шаги метода

На первом шаге задается вектор $\mathbf{p}^{(0)}$:

$$\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$$

На каждом шаге вычисляется невязка:

$$\mathbf{r}^{(i)} = A\mathbf{x}^{(i)} - \mathbf{b}$$

Направление движения определяется вектором $\mathbf{p^{(i)}}$, который может существенно отличаться от обратного направления градиенту в текущей точке:

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \alpha_i \mathbf{p}^{(i)}$$

Меняется и вычисление α_i:

$$\alpha_i = \frac{\mathbf{r^{(i)}}^T (A\mathbf{x}^{(i)} - \mathrm{b})}{\mathbf{p^{(i)}}^T A \mathbf{p}^{(i)}} = \frac{\mathbf{r^{(i)}}^T \mathbf{r}^{(i)}}{\mathbf{p^{(i)}}^T A \mathbf{p}^{(i)}}$$

Как вычисляется направление спуска? Для этого применяются подпространства Крылова - это линейная оболочка всех невязок с 0-го шага до i-го:

$$\mathcal{K}_i = \mathrm{span}(\mathbf{r}^{(0)}, \ldots, \mathbf{r}^{(i)})$$

Очередной $x^{(i+1)}$ ищется по всему подпространству, то есть:

$$\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)} \in \mathcal{K}_i$$

Для базиса подпространства будут использоваться $\mathbf{p}^{(i)}$. При этом необходимо задать условия:

$$\mathbf{p^{(i+1)}}^T A \mathbf{p}^{(j)} = 0, \quad j \in [0, i]$$

Итоговая формула для $\mathbf{p}^{(i+1)}$:

$$\mathbf{p}^{(i+1)} = \mathbf{r}^{(i+1)} + \frac{\mathbf{r^{(i+1)}}^T \mathbf{r}^{(i+1)}}{\mathbf{r^{(i)}}^T \mathbf{r}^{(i)}} \mathbf{p}^{(i)}$$

Подпространство Крылова ограничено размером вектора $\mathbf{x}$, то есть алгоритм не будет делать больше чем n шагов, а как правило, делает намного меньше. В системе, где $n = 1000000$, метод может сойтись за, например, 10 или 20 шагов. Это делает его очень применимым в реальных задачах с большими матрицами.

В алгоритме всего два деления на шаг, что делает метод удобным для реализации работы с разряженными матрицами.

Другие методы

- QR-разложение
- SVD-разложение
- LDL-разложение – родственно разложению Холецкого, предназначено для симметричных матриц (нет условия положительной определенности)
- Блочные алгоритмы – работают с матрицами как с наборами квадратных блоков, что актуально для параллельных вычислений
- Алгоритмы для разреженных матриц
- Мультифронтальные LU (MUMPS, UMFPack) – хорошо работает для разреженных матриц, в отличие от обычного LU
- BiCG, BiCGSTAB – модификации метода сопряженных градиентов, которые работают с любыми невырожденными матрицами
- Предобуславливатели