Лекции по вычислительным методам

Воеводин Анатолий Фёдорович

Тютюньков Вячеслав
Брусенцов Леонид

21 января 2006 г.

Оглавление

Сначала были рациональные числа, потом появилось земледелие и нужно было вычислять площадь и Герон придумал как это делать, в том числе алгоритм вычисления корня квадратного и иррационального числа, то есть всякие там e. Цивилизация развивалась, потом появились всякие механизмы, функции, комплексные числа, дифференциальные уравнения; а когда занялись воздухоплаванием, появились комплексные переменные, называемые там мнимыми числами; потом появились гипер-комплексные числа, связанные с космосом. Потом появились элементарные счёты, электронные приспособления для вычислений, потом механические компьютеры, потом электрические калькуляторы, называемые компьютерами; потом электронные электрические компьютеры. Потом появились электронные трансляторы Algon и тому подобные. Потом появились персональные компьютеры, выполняющие миллионы операций в секунду.

Иногда поставить натурные эксперименты невозможно или слишком дорого, так вот у нас проводят вычислительные эксперименты, моделирование. Сейчас уже все решаемые задачи до вас решены, остались только те, которые можно решить только численно, то есть с применением ЭВМ, надо написать программу.

Введение

Численные методы — такие, что могут быть реализованы на ЭВМ:

1. Создать метод решения (реализовать задачу);
2. Корректность:
   (а) Решение задачи этой существует;
   (б) Решение этой задачи единственно;
   (в) Непрерывная зависимость решения задачи от входных данных, то есть малым изменениям входных данных соответствуют малые изменения в решении.

Устойчивость — непрерывная зависимость решения от любых погрешностей при решении задач — более точная характеристика.

Первая погрешность: компьютер работает только с конечным представлением рациональных чисел, то есть $a = \alpha^p \cdot \cdot M$, где $M < 1$ — мантисса и $b = 0, m_1 m_2 \dots m_k$, причём в нормальной форме $m_1 \neq 0$, тогда это представление единственное, k — конечное число. Возникают ошибки округления, в частности при переводе из шестнадцатеричной в десятичную системы, например $\alpha = 16$, $\delta_{max} = 0,5 \cdot \alpha^{1-k}$, $k = 6$, то есть $\delta_{max} = 0,5 \cdot 16^{-5} \approx 10^{-8}/2$. Вы выполняете миллиард операций в секунду и возможно много ошибок, поэтому надо создать такие методы, чтобы ошибки были не велики.

Эти погрешности мы должны уметь вычислять, считать количество итераций.

Глава 1

Решение нелинейный уравнений (транциндентных)

1.1 Введение, постановка задачи. Основные сведения из математического анализа

Является одной из самых древних задач, решали в течении всей цивилизации, ещё Героном.

Нам дано уравнение $f(x) = 0$, надо найти корень уравнения x_* такой, что $f(x_*) \equiv 0$ — исходное уравнение. Как формулируется с точки зрения математики: введена непрерывная функция $f(x)$, которую рассматриваем для некоторых $x \in Q$, тогда решение уравнения сводится к отысканию нулей функции $f(x)$ в области Q или для некоторого $x \in [A, B]$; то есть значения аргумента, которые функцию $f(x)$ обращают в нуль.

Теорема 1.1.1 (Лагранжа)**.**

▷ Если функция $y(x)$ непрерывна на $[a, b]$ и имеет разные по знаку значения на $[a, b]$ (то есть например $f(a) \cdot f(b) < 0$), то на этом отрезке существует по крайней мере одна точка x_*, для которой $f(x_*) = 0$.

Если функция монотонна на отрезке $[a, b]$, то это значение единственно. Чтобы обеспечить единственность решения, мы должны выбрать отрезок, на котором она монотонна, для этого надо найти производную и она должна иметь один знак.

ОПР 1.1.2 (Гладкой функции)**.**
Функция называется гладкой*, если она непрерывна сама и её производная.*

Такие задачи решаются с помощью итерационных методов:

1. Условие, при котором решение существует и единственно.
2. Создаём итерационный метод, идея в том, что строится сначала нулевое приближение (a, b или середина отрезка, если на $[a, b]$ существует решение и единственно). Далее строится последовательность $x_0, x_1, x_2, \ldots, x_n$ значений такая, что $x_n \to x_*$, при $n \to \infty$ (говорят, сходится последовательность к решению задачи). Вся задача решения задачи сводится к построению такой последовательности, то есть сводится к уравнению $x = \varphi(x)$ и тогда зададим x_0, дальше $x_1 = \varphi(x_0)$, $x_2 = \varphi(x_1)$ и так далее, $x_n = \varphi(x_{n-1})$.

   Эта функция называется *производящей*, от неё зависит эффективность метода. Ясно, что $\varphi(x)$ не может быть какой попало.

Пример 1.1.3 (Решения задачи)**.**

▷ Пусть уравнение $x^2 - 2 = 0$, тогда можно выразить так: $x = 2/x$, сообразим, что ответ между 1 и 2, назначим $x_0 = 1$ и получим $x_1 = 2$, $x_2 = 2/2 = 1$, … называется *ложной сходимостью* (не улучшается).

Герон Александрийский построил $\varphi(x)$ таким способом:

$$x + x = \frac{2}{x} + x,$$

то есть добавил по x справа и слева и получил из этого

$$x = \frac{1}{2} \cdot \left(\frac{2}{x} + x \right),$$

берём тогда $x_0 = 1$, $x_1 = 3/2$, $x_2 = 17/12$, ... Но он, видимо, не так рассуждал, ему надо было вычислить $x = \sqrt{A}$, он выбирал приближения x_0 и брал A/x_0, пусть $A > 1$, тогда это значение больше, чем ответ, а он взял

$$\frac{1}{2} \cdot \left(\frac{A}{x_0} + x_0 \right).$$

$x_* = \varphi(x_*)$, а мы строим последовательность $x_n = \varphi(x_{n-1})$, рассмотрим величину погрешности $\varepsilon_n = x_n - x_*$, подставляем туда всё $\varepsilon_n = \varphi(x_{n-1}) - \varphi(x_*)$, пусть $\varphi(x)$ — гладкая, тогда она имеет непрерывную производную, если мы вспомним формулу конечных приращений, то это равно $\varphi'(\theta) \cdot (x_{n-1} - x_*)$ и мы получаем связь $|\varepsilon_n| \leqslant q \cdot |\varepsilon_{n-1}|$, где $q = \max_{a \leqslant x \leqslant b} |\varphi'(x)|$. Если $q < 1$, то ошибка $|\varepsilon_n| < |\varepsilon_{n-1}|$. Если мы обеспечим такое неравенство при построении, то при $\varepsilon_n \to 0$, при $n \to \infty$, будет падать как q^∞, то есть со скоростью геометрической прогрессии. Таким образом мы подобрались к требованиям для функции φ. 16.02.04

Всё было бы прекрасно, если бы всегда это было возможно, какие проблемы здесь возникают:

1. $f(x)$ может иметь такое значение x_*, что функция принимает всегда один знак, например $y(x) = (x-1)^2$ — всегда положителен, но ведь корень $x = 1$ имеется, по этой схеме, значит, мы этот корень не найдём. Всё потому, что корень кратный, здесь сама функция и её производная равны нулю.
2. Иногда в уравнение могут входить функции, которые непрерывные, но не гладкие, например $y = \sin|x|$ — производная терпит разрыв, другой пример $y = |x|$.

Поэтому здесь нужен какой-то новый критерий. Пусть $\pi \cdot \sin|x| + 3x = 0$, решение у него $x = -\pi/6$, будем брать $x = -\frac{\pi}{3} \cdot \sin|x| = \varphi(x)$, выберем хорошее приближение и всё сойдётся. То есть наша схема не универсальна, сформулируем теорему, которая включает и эти случаи.

1.2 Теорема о сходимости итерационных методов для решения нелинейных уравнений

1.2.1 Преамбула к теореме

Надо по-прежнему решить уравнение $f(x) = 0$ и мы это уравнение свели эквивалентными преобразованиями к $x = \varphi(x)$ и пусть $x \in [a, b]$ — отрезок, где мы ищем решение.

ОПР 1.2.1.1 (Условие Липшица)**.**
Функцию $\varphi(x)$ будем называть функцией, удовлетворяющей условию Липшица*, если $\forall x'$, $x'' \in [a, b]$ выполняется условие: $|\varphi(x') - \varphi(x'')| \leqslant L \cdot |x' - x''|$, где L называется* константой Липшица*, не зависит от x, но может зависеть от отрезка $[a, b]$.*

Теорема похожа на теорему о конечных приращениях, но не требуется существование производной. Можно взять $|\varphi(x') - \varphi(x'')| \leqslant M_1 \cdot |x' - x''|$, где $M_1 = \max_{x \in [a,b]} |\varphi'(x)| = L$, то есть гладкая функция всегда является Липшицевой.
Обозначим $\delta = (b-a)/2$, тогда заменим тождественным образом для удобства $[a, b]$ на область

$$Q\colon \{x_0 - \delta \leqslant x \leqslant x_0 + \delta\},$$

где $x_0 = (a+b)/2$, это называется *δ-окрестностью.*

Теорема 1.2.2 (Условие существования единственного корня)**.**

▷ Пусть

1. Дана $\varphi(x)$ — определена и непрерывна для $x \in Q$;
2. Функция $\varphi(x)$ удовлетворяет условиям Липшица с константой $q < 1$: $|\varphi(x') - \varphi(x'')| \leqslant L \cdot |x' - x''| - \forall x'$, $x'' \in Q$, $L = q < 1$;
3. Начальное приближение x_0 выбрано так, что $|x_0 - \varphi(x_0)| \leqslant (1-q) \cdot \delta$.

▷ Тогда

1. Существует единственный корень уравнения $x_* = \varphi_*(x)$;
2. Последовательность $x_n = \varphi(x_{n-1}) - \forall n = 1, 2, \ldots$ сходится к решению x_*, при $n \to \infty$, причём $|x_n - x_*| \leqslant \leqslant q^n \cdot \delta$ (это означает, что $\varepsilon_n \to 0$ при $n \to \infty$ со скоростью q).

Величина $F(x) = x - \varphi(x)$, называется ещё „невязка“, то есть не связалась, она должна быть меньше δ. Иногда, на практике, выбирают x_0, а потом выбирают окрестность — влево и вправо на $\delta/2$.

▷ **Доказательство**.

1. Первое, что мы должны: а имеет ли смысл её решать, то есть доказать существование. Наша цель найти на $x \in Q$, существует ли хотя бы один корень. Рассмотрим $F(x) = x - \varphi(x)$ — она везде определена, $F(x_0-\delta) = x_0-\delta-\varphi(x_0-\delta)$ — значение „невязки“ на левом конце; отниму и прибавлю $\varphi(x_0)$ и сгруппирую:

$$F(x_0 - \delta) = (x_0 - \varphi(x_0)) - \delta + (\varphi(x_0) - \varphi(x_0 - \delta))$$

и теперь попробуем оценить знак: согласно условию теоремы, функция удовлетворяет условию Липшица, то есть

$$-q \cdot |x_0 - (x_0 - \delta)| \leqslant \varphi(x_0) - \varphi(x_0 - \delta) \leqslant q \cdot |x_0 - (x_0 - \delta)| \Rightarrow -q\delta \leqslant \varphi(x_0) - \varphi(x_0 - \delta) \leqslant q\delta,$$

аналогично из условия 3:

$$-(1 - q) \cdot \delta \leqslant x_0 - \varphi(x_0) \leqslant (1 - q) \cdot \delta,$$

тогда

$$F(x_0 - \delta) \leqslant (1 - q) \cdot \delta - \delta + q\delta = 0,$$

то есть $F(x_0 - \delta) \leqslant 0$, а если равна нулю, то говорим, что $x_0 - \delta$ есть корень и доказательство закончено.

Если меньше нуля, то рассмотрим $F(x_0 + \delta) = x_0 + \delta + \varphi(x_0 + \delta)$, здесь аналогично

$$F(x_0 + \delta) = (x_0 - \varphi(x_0)) + \delta + (\varphi(x_0) - \varphi(x_0 + \delta))$$

и

$$-(1 - q) \cdot \delta \leqslant x_0 - \varphi(x_0) \leqslant (1 - q) \cdot \delta,$$

тогда

$$F(x_0 + \delta) \geqslant -(1 - q) \cdot \delta + \delta + (-q\delta) = 0,$$

если равно нулю, то правый конец является корнем.

Пусть концы не являются корнями, но тогда есть ещё корень, так как функция непрерывна и концы разных знаков, значит хотя бы один корень есть.

2. Единственность: пусть существует два решения: $x_*\big|\, x_* = \varphi(x_*)$ и $x_{**}\big|\, x_{**} = \varphi(x_{**})$, тогда вычтем: $x_* - x_{**} = \varphi(x_*) - \varphi(x_{**})$, тогда по условию Липшица

$$|x_* - x_{**}| \leqslant q \cdot |x_* - x_{**}| \Rightarrow |x_* - x_{**}| < |x_* - x_{**}|,$$

такого не бывает, то есть противоречие, остаётся одно предположение, что корень единственный.

3. Сходимость: ищем $x_n = \varphi(x_{n-1})$, первое, что докажем, что мы имеем право вычислить φ, то есть все $x_{n-1} \in Q$, то есть $|x_n - x_0| \leqslant \delta$, то есть что мы лежим в Q. Будем доказывать по индукции:
   а). $|x_0 - x_1| = |x_0 - \varphi(x_0)| < \delta \cdot (1 - q)$ — согласно условию теоремы.
   б). Пусть $x_n \in Q$, рассмотрим $x_{n+1} = \varphi(x_n)$ и рассмотрим

$$|x_{m+1} - x_m| = |\varphi(x_m) - \varphi(x_{m-1})| \leqslant q \cdot |x_m - x_{m-1}| = q \cdot |\varphi(x_{m-1}) - \varphi(x_{m-2})| \leqslant$$
$$\leqslant q^2 \cdot |x_{m-1} - x_{m-2}| \leqslant \ldots \leqslant q^m \cdot |x_1 - x_0|.$$

Рассмотрим ещё

$$|x_{m+1} - x_0| = |x_{m+1} - x_m + x_m - x_{m-1} + x_{m-1} + \ldots + x_1 - x_1 + x_0| = |(x_{m+1} - x_m) + \ldots + (x_1 - x_0)| \leqslant$$
$$\leqslant \Big|q^m \cdot |x_1 - x_0| + q^{m-1} \cdot |x_1 - x_0| + \ldots + |x_1 - x_0|\Big| = \frac{1 - q^{m+1}}{1 - q} \cdot |x_1 - x_0| \leqslant (1 - q^{m+1}) \cdot \delta < \delta,$$

значит все $x_m \in Q$, то есть те значения, которые мы нашли по правилу 1, они не выскакивают из отрезка, все условия теоремы обеспечивают нам выполнение процесса.

Но мы ещё не доказали сходимость, будем использовать критерий сходимости Больцано-Коши: $\forall p \geqslant 2$ целого нужно чтобы $|x_{m+p} - x_m| \to 0$ при $m + - \to \infty$. Рассмотрим:

$$|x_{m+p} - x_m| = |x_{m+p} - x_{m+p-1} + x_{m+p-1} + \ldots + x_{m+1} - x_m| \leqslant$$
$$\leqslant |q^{m+p-1} + q^{m+p-2} + \ldots + q^m| \cdot \delta = q^m \cdot \frac{1 - q^p}{1 - q} \cdot \delta \xrightarrow{m \to \infty} 0,$$

так как $q < 1$.

4. Рассмотрим:
$$|x_n - x_*| = |\varphi(x_n) - \varphi(x_*)| \leqslant q \cdot |x_{n-1} - x_*| \leqslant q^2 \cdot |x_{n-2} - x_*| \leqslant q^n \cdot \delta.$$

□

Следствие 1.2.2.1 (Для гладкой функции).

▷ Если функция $\varphi(x)$ — гладкая и $\max_{x\in Q} |\varphi'(x)| = q < 1$, то условия теоремы, сформулированной выше, выполняются.

У нас есть в компьютере ошибки округления, поэтому на практике всегда задаётся ε — точность, в которой должен быть результат. Остановить выполнение программы надо когда достигнута точность ε, как использовать для этого теорему?

1.2.3 Критерий остановки процесса

Пусть $\varepsilon_{\text{от}}$ — точность, с которой надо найти решение, тогда $|\varepsilon_n| = |x_n - x_*| \leqslant q^n \cdot \delta$, где δ — окрестность и относительная ошибка $|\varepsilon_{\text{от}}| = |\varepsilon_n|/\delta$, тогда согласно теореме это $\leqslant q^n \leqslant_{\text{(требуем)}} \varepsilon_{\text{от}}$, получим n — номер итерационного приближения, где надо остановить процесс. Тогда из
$$q^n \leqslant \varepsilon_{\text{от}} \Rightarrow \frac{1}{\varepsilon_{\text{от}}} \leqslant \left(\frac{1}{q}\right)^n \Rightarrow \ln\frac{1}{\varepsilon_{\text{от}}} \leqslant n \cdot \ln\frac{1}{q},$$
выбираем
$$n_0 = \left\lceil \ln\frac{1}{\varepsilon_{\text{от}}} \Big/ \ln\frac{1}{q} \right\rceil;$$
теперь рассмотрим по-другому, известно, что
$$|\varepsilon_n| \leqslant q \cdot |\varepsilon_{n-1}| = q \cdot |(x_{n-1} - x_n + x_n - x_*)| \leqslant q \cdot |x_n - x_{n-1}| + q \cdot |x_n - x_*| = q \cdot |x_n - x_{n-1}| + q \cdot |\varepsilon_n|$$
и тогда я получу оценку
$$|\varepsilon_n| \leqslant \frac{q}{1-q} \cdot |x_n - x_{n-1}|$$
— замечательный критерий, нужно проверить в компьютере, чтобы
$$|x_n - x_{n-1}| < \frac{\varepsilon}{q} \cdot (1-q),$$
то есть $|\varepsilon_n| \leqslant \varepsilon$ или
$$\frac{|x_n - x_{n-1}|}{|x_n|} \leqslant \frac{\varepsilon_{\text{от}}}{q} \cdot (1-q).$$
Остались только способы построить Q, а для этого надо уметь выбрать отрезок $[a, b]$.

1.3 Способы отделения корней

$f(x) = 0$, $y = f(x)$ — задача в какой-то области; иногда есть дополнительные условия, вроде положительности корней, минимальное или максимальное решение и тому подобное.

1. Способ графический, то есть обычно уравнение $f(x)$ представляют в виде $f_1(x) = f_2(x)$, мы рисуем обе функции $y_1 = f_1(x)$, $y_2 = f_2(x)$ и ищем их пересечение примерно по графику.

Пример 1.3.1 (Оценки по графику)**.**

▷ Пусть у нас уравнение $f(x) = x \cdot \sin x - 1 = 0$, $x = 0$ здесь не корень, так что делим на x: $\sin x = 1/x$, ищем положительные корни:
$$\sin\frac{\pi}{2} = 1,\ \frac{1}{\pi/2} = \frac{2}{\pi} < 1,$$
но кривые непрерывны на отрезке $[0, \pi/2]$, можно взять производную. Функция 1/x монотонна и лежит ниже, чем синус, значит они где-то пересекаются, то есть есть пересечение между нулём и $\pi/2$.

Но этот способ слишком интеллектуальный, а вы все погрузились в компьютеры, так что существует второй способ.

2. Сканированием (компьютерный), здесь совсем не надо думать. Пусть $y = f(x)$, $x \in [A, B]$, разобьём этот отрезок на N элементарных отрезков и сначала вычислим таблицу значений; разбиение наше с шагом $h = (B - A)/N$, то есть $x_i = A + i \cdot h$, $i = 0, 1, \ldots, N$, следовательно рассмотрим $f(x_{i-1}) \cdot f(x_i)$, $i = 1, 2, \ldots, N$, и рассмотрим $[x_{i-1}, x_i]$.

   Пример 1.3.2 (Метода сканирования)**.**

   ▷ Выберем $a = x_{i-1}$, $b = x_i$ и функцию $y = x^3 - x^2 - x + 1 = f(x) = 0$, подставим 2 и (-2), тогда получим $f(A) > 0$, а $f(B) < 0$, но эта функция не такая простая, она имеет два корня — 1 и (-1), обозначим $x_*^{(1)} = -1$, $x_*^{(2,\,i)} = 1$, то есть есть кратный корень. Поэтому при сканировании слева сразу получим корень, а справа и, допустим, в 0.8 и в 1.2 получим положительные маленькие значения, то есть мы можем корень прозевать, поэтому рассмотрим в подозрительной области $|f(x_{i-1})| \leqslant \varepsilon$, и для неё проверим $|f(x_i)| < \varepsilon$, если кратный корень, оба значения малы и это говорит, что может существовать корень.

   Но есть теорема в математическом анализе, что если точка x_* является p-кратным корнем, то $f(x)$ можно представить в виде $(x-x_*)^p \cdot g(x)$, причём $g(x_*) \neq 0$ и представление единственное. А это значит, что $f'(x) = p \cdot (x - x_*)^{p-1} \cdot g(x) + (x - x_*)^p \cdot g'(x)$, то есть при $f(x_*) = 0$, $f'(x_*)$ тоже обращается в нуль, то есть если корень кратности 2, то и производная тоже обратится в нуль. Поэтому рассмотрим $f'(x_i) \cdot f'(x_{i-1})$, если это $\leqslant 0$, то корень кратный и можно решать уравнение $f'(x) = 0$. Мы выбираем $a = x_{i-1}$, $b = x_i$ и решаем уравнение $f'(x) = 0$ и таким образом найдём кратный корень.

   02.03.05

3. Метод деления пополам (бисекция или дихотомия), где выбираем $a \leqslant x \leqslant b$, причём $f(a) \cdot f(b) < 0$, берём $c = (a + b)/2$ и смотрим $f(x)$, если $f(a) \cdot f(c) \leqslant 0$, то берём $a^{(1)} := a$, $b^{(1)} := c$, мы переприсваиваем значения и работаем уже на новом отрезке $[a, c]$; иначе берём $a^{(1)} := c$, $b^{(1)} := b$. То есть мы уточняем интервал, в котором лежит решение, но этот метод сходится очень медленно, но зато сходится всегда.

1.4 Способы конструирования итерационных методов

По-прежнему рассматриваем уравнение $f(x) = 0$ и мы нашли отрезок $a \leqslant x \leqslant b$, подозрительный на существование корня, нулевое приближение $x_0 = (a+b)/2$. Вместо уравнения $f(x) = 0$ строим уравнение $x = \varphi(x)$ и $x_{k+1} = \varphi(x_k)$. Главное требование на φ, чтобы $\varphi(x_k) \to x_*$ при $k \to \infty$, это тоже что $|\varphi'(x)| \leqslant q < 1$. Всё дело заключается в том, как построить эту φ.

1.4.1 Метод релаксации

Этот метод самый современный, имеет множество модификаций. Он, такой, самый общий метод, идея в том, что мы построили начальное приближение x_0, но $f(x_0) = r_0 \neq 0$ (невязка), тогда исправляем это x_0: $x_1 = x_0 + \tau_1 \cdot \cdot r_0 = x_0 + \tau_1 \cdot f(x_0)$, где τ_1 называется *релаксационный параметр* или просто *итерационный параметр.*

Если берём τ каждый раз разный, то способ называется *переменный метод релаксации*, иначе *метод релаксации с постоянным параметром* (простой итерации).

Но классический способ, когда $\tau_i = \tau$, но так не всегда сойдётся, так как может не удовлетворять условию теоремы.

$$x_{k+1} = x_k + \tau_{k+1} \cdot f(x_k) \tag{1.4.1}$$

— расчётная формула, для которой мы составили нашу программу. Но как выбрать τ_k? Здесь мы уже анализируем функцию.

1. Функция гладкая, то есть имеет непрерывную производную, считаем для определённости меньше нуля на $a \leqslant \leqslant x \leqslant b$, если же больше, то умножаем уравнение $f(x) = 0$ на (-1), тогда станет как нужно. Тогда согласно формуле (1.4.1), наша производная функция $x = \varphi(x) = x + \tau \cdot f(x)$, где τ тоже может зависеть от x, то есть может быть

   $$x = \varphi(x) = x + \tau(x) \cdot f(x). \tag{1.4.2}$$

   Если мы работаем по методу простой итерации и $\tau(x) = \tau = const$, тогда из формулы (1.4.2) получаем

   $$\varphi'(x) = 1 + \tau \cdot f'(x) = 1 - \tau \cdot |f'(x)|. \tag{1.4.3}$$

Главное, что $|\varphi'(x)| < 1$, тогда из соотношения (1.4.3) получаем

$$-1 < 1 - \tau \cdot |f'(x)| < 1 \tag{1.4.4}$$

и из него получаем условие на итерационный параметр

$$\tau < \frac{2}{|f'(x)|}. \tag{1.4.5}$$

Пусть мы можем оценить $\max_{a \leqslant x \leqslant b} |f'(x)| = M_1$ и берём загрублёное $\tau < 2/M_1$. При таком выборе τ мы всегда обеспечим, что производная < 1, то есть программа работает и итерации сходятся.

Возникает желание выбрать τ оптимальный, такой чтобы константа q была минимальной. Возникает так называемая задача „минимакс“, то есть ищем $q(\tau) = 1 - \tau \cdot |f'(x)|$ — зависит от τ и аргумента $z = |f'(x)|$. Пусть $m_1 = \min_{a \leqslant x \leqslant b} |f'(x)|$, тогда наш параметр $z\colon m_1 \leqslant z \leqslant M_1$ и мы получаем, что функция $q(\tau)$ — прямая с наклоном τ отрицательным, то есть график убывает, раз она линейная, то достигает по модулю максимальное значение на концах отрезка, а нам надо, чтобы

$$\min_{\tau} \left(\max_{m_1 \leqslant z \leqslant M_1} |q(\tau,\, z)| \right). \tag{1.4.6}$$

Если эта прямая проходит через этот отрезок, то $z_{\text{опт.}} = (M_1 + m_1)/2$, так как если максимум на левом конце и этот максимум больше, то получаем бóльшие значения, иначе наоборот. В этом случае чтобы $q(z_{\text{опт.}}) = 0$, оптимальное $\tau_{\text{опт.}} = 2/(M_1 + m_1)$, тогда $q_{\text{опт.}} = (M_1 - m_1)/(M_1 + m_1)$. Этот метод релаксации называется *методом простой релаксации с оптимальным параметром.*

2. Метод Ньютона (метод касательных). Это пример метода релаксации с переменным параметром, то есть это частный случай. Когда Ньютон изобретал, он, конечно, не знал метода релаксации.

   Выберем $f(x) = 0$, $x_0 = (a + b)/2$ и ищем $x_* \in [a,\, b]$, что $f(x_*) = 0$. Рассмотрим разность $|f(x_*) - f(x_0)|$ и пусть $f(x)$ — гладкая, тогда по теореме математического анализа о конечных приращениях

   $$|f(x_*) - f(x_0)| = f'(\theta) \cdot (x_* - x_0), \tag{1.4.7}$$

   где θ — существует, но как её выбрать, теорема не говорит. Но мы выбрали $f(x_*) = 0$, тогда я могу найти

   $$x_* = x_0 - \frac{f(x_0)}{f'(\theta)}, \tag{1.4.8}$$

   всё замечательно, но тут одна заковырка, а чему равна θ? Берём $\theta = x_0$, тогда получаем неточно

   $$\tilde{x}_* = x_0 - \frac{f(x_0)}{f'(x_0)} = x_1 \tag{1.4.9}$$

   и получаем первое приближение x_1.

   Теперь по аналогии второе и так далее, то есть

   $$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{1.4.10}$$

   — расчётная формула метода Ньютона, эту формулу и программируем.

   Можно записать в другой форме: $x_{k+1} = x_k + \theta_k \cdot f(x_k)$, где $\theta_k = -1/f'(x_k)$, то есть $\varphi(x) = x - f(x)/f'(x)$. Теперь начнём изучать сходимость метода Ньютона, значит нужно оценить

   $$\varphi'(x) = 1 - \frac{f'(x)}{f'(x)} + \frac{f(x)}{(f'(x))^2} \cdot f''(x) = \frac{f(x)}{(f'(x))^2} \cdot f''(x). \tag{1.4.11}$$

   Мы должны найти

   $$\max_{a \leqslant x \leqslant b} |f''(x)| = M_2 \text{ и } \min_{a \leqslant x \leqslant b} |f'(x)| = m_1, \tag{1.4.12}$$

   тогда из них получается, что

   $$|\varphi'(x)| \leqslant q = \frac{|f(x)| \cdot M_2}{m_1^2}, \tag{1.4.13}$$

единственное, что мы можем, выбрать — $a \leqslant x \leqslant b$, чтобы $f(x) < m_1^2/M_2$, то есть надо выбрать поаккуратнее отрезок.

Но вы скажете, зачем всё это нужно, вторая производная, не лучше ли метод релаксации? Но у метода Ньютона есть очень хорошее свойство, его сходимость быстрее, чем геометрическая за счёт того, что $|\varphi'(x_*)| = 0$. Рассмотрим рассчётную формулу:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},$$

вычтем по x_*, тогда получим погрешность

$$\varepsilon_{k+1} = \varepsilon_k - \frac{f(x_k)}{f'(x_k)},$$

отнимем $f(x_*) = 0$:

$$\varepsilon_{k+1} = \varepsilon_k - \frac{f(x_k) - f(x_*)}{f'(x_k)} \tag{1.4.14}$$

и разложим в ряд Тейлора:

$$f(x_*) = f(x_k) + f'(x_k) \cdot (x_* - x_k) + \frac{f''(\eta)}{2} \cdot (x_* - x_k)^2$$

— согласно остаточному члену в форму Лагранжа, где $\eta \in [x_*,\, x_k]$. Я подставлю это в формулу (1.4.14), тогда

$$\varepsilon_{k+1} = \varepsilon_k - \varepsilon_k + \frac{f''(\eta}{2f'(x_k)} \cdot \varepsilon_k^2,$$

то есть

$$\varepsilon_{k+1} = \frac{f''(\eta)}{2 \cdot f'(x_k)} \cdot \varepsilon_k^2,$$

таким образом ошибка у нас на порядок уменьшается, такая сходимость называется квадратичной и она даёт сходимость к нулю быстрее, чем геометрическая прогрессия.

Рассмотрим

$$|\varepsilon_1| \leqslant \frac{M_2}{2 \cdot m_1} \cdot \varepsilon_0^2,$$

где $|\varepsilon_0| = |x_0 - x_*| < \delta$, где δ — половина длины отрезка, тогда

$$|\varepsilon_1| \leqslant \frac{M_2}{2 \cdot m_1} \cdot |\varepsilon_0| \cdot |\varepsilon_0|.$$

Выберем наш отрезок таким образом, чтобы величина

$$\frac{M_2}{2 \cdot m_1} \cdot \delta = p < 1,$$

это можно достичь методом уточнения интервала, на каком-то шаге мы этого неравенства достигнем, если мы его достигли (всегда можем достичь, если обеспечить сходимость метода Ньютона), то получим

$$|\varepsilon_1| \leqslant p \cdot |\varepsilon_0|,\ |\varepsilon_2| \leqslant \frac{M_2}{2 \cdot m_1} \cdot |\varepsilon_1|^2, \tag{1.4.15}$$

а $|\varepsilon_1|^2$ из (1.4.15) даёт, что

$$|\varepsilon_2| \leqslant \frac{M_2}{2 \cdot m_1} \cdot p^2 \varepsilon_0^2 = p^3 \cdot |\varepsilon_0|,$$

то есть здесь степень уже 3. Если я возьму и оценю, $|\varepsilon_3| = p^7 \cdot |\varepsilon_0|$ и общая формула $|\varepsilon_{k+1}| \leqslant p^{2^k - 1} \cdot |\varepsilon_0|$, но ясно, что $p^{2^k-1} \to 0$ быстрее, чем геометрическая прогрессия, супер прогресс! Выгодно найти какое-то грубое приближение, а потом быстро свести методом Ньютона. Иногда этот метод называют методом Ньютона-Раввсона. Иногда ему дают геометрическую интерпретацию, смотрите рисунок.

$$x_1 = x_0 - \frac{1}{f'(x_0)} \cdot f(x_0),$$

на самом деле строится прямая через точку $(x_0,\, f(x_0))$, а тангенс угла наклона $\alpha = f'(x_0)$, прямая $(x - x_0) \cdot$ $\cdot\, f'(x_0) + f(x_0) = y(x)$ и получаем её пересечение с осью OX — точка x_1. И каждый раз итерация приближает.

3. Модификация метода Ньютона: видно, чем неудобен метод Ньютона, вычисление производной иногда сопряжено с большими вычислениями, решения проблемы:

   ▷ Иногда берут лишь первую производную, а дальше методом релаксации со стационарной переменной

   $$x_{k+1} = x_k - \frac{1}{f'(x_0)} \cdot f(x_k).$$

   При некотором условии этот метод сходится, но пользоваться нужно аккуратно.

   ▷ Не вычисляют вообще производную, берут приближённо. Вы знаете, как образуется производная: это предел

   $$\frac{f(x+\delta) - f(x)}{\delta} \to f'(x),$$

   при $\delta \to 0$. Эти соображения и вкладываются в модификацию метода Ньютона. На этой идее и построен метод Хорд (таможенного положения). Мы выбрали отрезок $a \leqslant x \leqslant b$ такой, что функция поменяла знак, тогда в качестве x_0 выберем один из концов отрезка, пусть $x_0 = a$, $x_1 = b$ и

   $$x_2 = x_0 - \frac{f(a)\cdot(b-a)}{f(b)-f(a)} = x_0 - \frac{f(x_0)\cdot(x_1-x_0)}{f(x_1)-f(x_0)},$$

   то есть заменим производную на приближенную производную, но сходимость окажется почти такая же.

   Так в приближении получается $f'(x_k) = \frac{f(x_k)-f(x_{k-1})}{x_k - x_{k-1}}$, но этот метод уже называется *двухшаговым* (многошаговым), так как приближается через два предыдущих значения, итого:

   $$x_{k+1} = x_k - \frac{f(x_k)\cdot(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}.$$

   Можно представить в виде $y = kx + b$:

   $$y - f(x_k) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \cdot (x - x_k)$$

   — уравнение прямой, проходящей через точки $f(a)$ и $f(b)$, действительно, подставляя $x_k = x$, получаем $y = f(x_k)$ и всё получается. Когда x обращается в нуль, получаем новое приближение. Ещё называется на западе „методом ложного положения“, так как на отрезке $a = x_1$, $b = x_2$ строим

   $$x_3 = x_2 - \frac{x_2 - x_1}{f(x_2) - f(x_1)} \cdot f(x_2),$$

   теперь вычисляем $f(x_3)$, если меньше ε, то завершаем вычисления. И теперь умножаем: $f(x_3)\cdot f(x_2)$, если больше нуля, то $x_2 := x_3$ (присваиваем) и включаем формулу для нового x_3, а иначе $x_1 := x_3$ и снова включаем формулу следующего приближения.

   Сходимость немного хуже метода Ньютона, но когда x_k близко становится к x_{k-1}, то становится очень близко. Здесь важно вовремя остановиться, так как машина не знает правило Лопиталя и нельзя делить на очень маленькие числа.

Мы рассмотрели наиболее популярные способы, конечно хорошо, если вы не забыли, как брать производную, тогда лучше использовать метод Ньютона.

1.4.2 Особенности отыскания корней полинома

Пусть есть $f(x) = 0$ и полином степени n:

$$P_n = a_0 + a_1 x^1 + a_2 x^2 + \ldots + a_n x^n$$

— вот как обычно вы записывали в алгебре. И пусть нам надо найти какой-то корень и сразу возникает проблема, так как даже для квадратного двучлена корень может быть не единственным, кратным и даже ко́мплексным (те, кто совсем крутые математики, говорят компле́ксные). Алгебра даёт нам число корней, их свойства, и это надо здесь привлекать. Я буду считать, что вы алгебру знаете и сможете выделить отрезок $[a, b]$.

Считаем, что мы выделили окрестность вещественного корня, пусть $x_* \in [a, b]$ — корень,

$$x_{k+1} = x_k - P_n(x_k)/P'_n(x_k)$$

— очень популярный метод Ньютона, но эта формула подразумевает, что мы должны вычислить полином, но так в каноническом виде вычислять нельзя, так как потратим порядка $n!$ операций. Поэтому для $x = x_k$ используется схема Горнера:

$$P_n(x) = a_0 + x \cdot (a_1 + x \cdot (a_2 + \ldots + x \cdot (a_{n-1} + x \cdot a_n) \ldots)),$$

для обобщения такого алгоритма при $x = x_k$:

$$b_n = a_n,\ b_{n-1} = a_{n-1} + x_k \cdot b_n,\ b_{n-2} = a_{n-1} + x_k \cdot b_{n-1},$$

такая рекуррентная формула, $b_0 = a_0 + x_1 \cdot b_1 = P_n(x_k)$.

Следующая проблема: как вычислить производную в точке x_k? Тут применяется хитрейшая хитрость:

$$P_n(x) = (x - x_k) \cdot G_{n-1} + P_n(x_k) \tag{1.4.16}$$

— теорема Безу, теперь $P'_n(x) = G_{n-1} + (x - x_k) \cdot G'_{n-1}$, теперь для $x = x_k$: $P'_n(x_k) = G_{n-1}(x_k)$, но $G_{n-1} = \bar{b}_1 + \bar{b}_2 x + \ldots + \bar{b}_n x^{n-1}$, посмотрим внимательно формулу (1.4.16):

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = (x - x_k) \cdot (\bar{b}_1 + \bar{b}_2 x + \ldots + \bar{b}_{n-1} x^{n-1}) + \underbrace{(a_0 + a_1 x_k + \ldots + a_n x_k^n)}_{=b_0},$$

левая часть совпадает с правой, поэтому коэффициенты полиномов совпадают, тогда $a_0 = b_0 - x_k \cdot \bar{b}_1$, тогда $b_0 = a_0 + x_k \cdot \cdot \bar{b}_1$, $a_1 = \bar{b}_1 - b_2 \cdot x_k \Rightarrow b_1 = a_1 + x_k \cdot \bar{b}_2$, $b_2 = a_2 + x_k \cdot \bar{b}_3$ и мы получим, что коэффициенты $\bar{b}$ вычисляются как и без черты, то есть $\bar{b}_i \equiv b_i$, для $i = 1, 2, \ldots, n$. А дальше — дело техники, вычисляем производные: $c_n = b_n$, $c_{n-1} = b_{n-1} + x_k \cdot \cdot c_n$, $\ldots$, $c_1 = b_1 + x_k \cdot c_2$, $c_1 = G_{n-1}(x_k)$, поэтому

$$x_{k+1} = x_k - \frac{P_n(x_k)}{P'_n(x_k)} = x_k - \frac{b_0}{c_1}$$

— очень просто и экономично. Это и есть схема Горнера[I)].

При советской власти анекдот был: американцы полетели на Луну, наше политбюро ЦК вызывает наших космонавтов и говорят: „Полетите на Солнце!“ — на что космонавты отвечают: „Ну как же мы полетим, сгорим же?“ — „Вы что же думаете, в политбюро одни дураки сидят? Полетите ночью!“ Вот так и здесь.

1.4.3 Способы повышения скорости сходимости итерационных методов

Мы рассматривали с вами одношаговые методы релаксации: $x_{k+1} = x_k + \theta_k \cdot f(x_k)$ и мы улучшали скорость сходимости так: брали стационарный процесс, простейшая формула, но давало линейный рост, то есть погрешность $|\varepsilon_{k+1}| \leqslant q \cdot |\varepsilon_k|$. А когда мы привлекали для θ_k производную в методе Ньютона: $\theta_k = 1/f'(x_k)$, то нам позволила увеличить скорость новая производная второго порядка, то есть чем больше информации о функции имеем, тем лучше. А если мы узнаем вторую, третью и так далее производные:

1. Великий русский вычислитель Чёбышев (иногда Чебышёв), много сделал для вычислительной математики и метод Ньютона был долго непобедимым, а он придумал целый класс, превосходящих по скорости вычисления, но надо привлечь большее число производных.

   В методе Ньютона

   $$x = \varphi(x),\ \varphi(x) = x - \frac{1}{f'(x)} \cdot f(x)$$

   и всё дело в том, что

   $$\varphi'(x_*) = 1 - \frac{f'(x_*)}{f'(x_*)} + \frac{f(x_*)}{(f'(x_*))^2} \cdot f''(x_*) = 0,$$

   а это и определяет скорость сходимости. Тогда погрешность $\varepsilon_{k+1} = x_{k+1} - x_* = \varphi(x_k) - \varphi(x_*)$, разложим в ряд:

   $$\varphi(x_k) = \varphi(x_*) + \varphi'(x_*) \cdot (x_k - x_*) + \frac{\varphi''(x_*)}{2} \cdot (x_k - x_*)^2 + \ldots,$$

   тогда у нас

   $$\varepsilon_{k+1} = \varphi'(x_*) \cdot \varepsilon_k + \varphi''(x_*) \cdot \frac{\varepsilon_k^2}{2}$$

   и мы получаем, что $|\varepsilon_{k+1}| \leqslant c \cdot \varepsilon_k^2$. А если мы обнулим и вторую производную, то мы можем строить таким образом ещё способы: $x = \varphi(x) = x + a_1(x) \cdot f(x) + a_2(x) \cdot f^2(x) + \ldots$.

[I)] Для простоты можно вспомнить, что $P'_n(x) = a_1 + x \cdot (2a_2 + x \cdot (3a_3 + x \cdot (\ldots((n-1)a_{n-1} + nx \cdot a_n) \ldots)))$ и выбрать отсюда последовательность $c_k = a_{k-1} + x \cdot c_{k+1}$.

Чебышев это предложил и сколько мы захотим, столько используем. Подберём таким образом, чтобы $\varphi'(x_*) = 0$, если полином первого порядка, то $\varphi'(x) = 1 + a_1 \cdot f'(x) + a_1' \cdot f(x)$, тогда при $x = x_*$: $\varphi'(x_*) = 0$, если $a_1 = -1/f'(x)$, а это получается метод Ньютона. А если мы хотим кубическую сходимость, потребуем, чтобы $\varphi''(x_*) = 0$, тогда при $x = x_*$ выбираем коэффициенты для

$$\varphi(x) = x + a_1 \cdot f(x) + a_2(x) \cdot f^2(x),$$
$$\varphi''(x) = 2 \cdot a_1(x) \cdot f'(x) + a_1 \cdot f''(x) + 2 \cdot a_2(x)(f')^2.$$

Этот метод ещё называется методом Стефенсона у иностранцев.

Здесь мы вынуждены вычислять f'', но ничего бесплатно не даётся. То же: то ли вы пошли пешком и не понятно, когда дойдёт, а другое, если вы пошли пешком с определённой скоростью

2. Многошаговые методы. В методе хорд мы через две точки получили третью. Пусть теперь $x_1 = a$, $x_2 = b$, x_3 — пересечение оси и прямой через точки $(x_1,\, f(x_1))$ и $(x_2,\, f(x_2))$. Проведём через эти 3 точки кривой кривую второго порядка — параболу, интуитивно понятно, что это даст лучшее приближение, то есть $f(x) = y(x) = a_0 + a_1 x + a_2 x^2$ и требуем, чтобы она проходила через x_{k-1}, x_k и x_{k+1} и когда обращается в нуль, будет следующее приближение x_{k+2}.

   У этого полинома три коэффициента, но у нас есть три уравнения:

$$\begin{cases} y(x_{k-1}) &= a_0 + a_1 x_{k-1} + a_2 x_{k-1}^2, \\ y(x_k) &= a_0 + a_1 x_k + a_2 x_k^2, \\ y(x_{k+1}) &= a_0 + a_1 x_{k+1} + a_2 x_{k+1}^2. \end{cases}$$

   Эта задача всегда имеет решение, в интерполировании, когда будем проходить, это выясним, здесь определитель Ван дер Монда

$$W = \begin{vmatrix} 1 & x_{k-1} & x_{k-1}^2 \\ 1 & x_k & x_k^2 \\ 1 & x_{k+1} & x_{k+1}^2 \end{vmatrix}$$

   — решаем и выбираем решение.

   Для кривых большего порядка — линейное интерполирование функций, целая глава для них, надо больше точек. Можно использовать все точки, по скорости сходимости близок к кубическому, но не надо вычислять производные.

Глава 2

Вычислительные методы для систем линейных уравнений

Под системами линейных уравнений понимаются уравнения вида

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n & = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n & = b_2, \\ \vdots & \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n & = b_m. \end{cases}$$

Неизвестными являются величины x_1, x_2, …, x_n; коэффициенты a_{ij}, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ и b_1, b_2, …, b_m считаются заданными. Если $m = n$, то есть система замкнутая, то сколько уравнений, столько и неизвестных и из линейной алгебры известно, что такая система имеет единственное решение. Если неравны, то может быть много чего, используется в линейном программировании, экономике, этим пользуются жулики и начинают воровать. Мы будем изучать $m = n$ — замкнутые системы, мы люди честные.

Обозначим $\vec{x} = \{x_i\}_{i=1}^n$ с помощью линейного оператора $A = \{a_{ij}\}_{i,j=1}^n$, приводящим к вектору $\vec{b} = \{b_i\}_{i=1}^n$ и задача найти $\vec{x}$. Обычно вектора рассматривают как столбцы

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \ \vec{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

а `A` является матрицей:

$$\mathtt{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}.$$

Наши основные рассуждения будут опираться на линейную алгебру, которую, надеюсь, вы изучили, но я вам всё же прочитаю один маленький параграф.

2.1 Основные положения линейной алгебры

Я бы назвал ещё так: «основные сведения, которые должен знать студент». $\vec{x} = \{x_i\}_{i=1}^n$ — вектор размерности n, но наряду с векторами столбцами, мы будем говорить и векторах строках.

2.1.1 Векторы

Выполняются операции:

1. $\vec{z} = \alpha \cdot \vec{x} \Rightarrow z_i = \alpha \cdot x_i$, $i = 1, 2, \ldots, n$ — умножение на скаляр;
2. $\vec{z} = \vec{x} \pm \vec{y} \Rightarrow z_i = x_i \pm y_i$, $i = 1, 2, \ldots, n$ — сложение (линейная комбинация);
3. $(\vec{x}, \vec{y}) = (\vec{y}, \vec{x}) = \sum_{i=1}^n x_i \cdot y_i$ — скалярное произведение;

2.1.2 Норма вектора

Подчиняется некоторым аксиомам, что попало взять нельзя:

1. $\|\vec{x}\| = 0 \Rightarrow x_i = 0$, $i = 0, 1, \ldots, n$ — норма не отрицательна;
2. $\|\alpha \cdot \vec{x}\| = |\alpha| \cdot \|\vec{x}\|$;
3. $\|\vec{x} + \vec{y}\| \leqslant \|\vec{x}\| + \|\vec{y}\|$.

Самые известные нормы:

1. $\|\vec{x}\|_1 = \max_{1 \leqslant i \leqslant n} |x_i|$;
2. $\|\vec{x}\|_2 = \sum_{i=1}^{n} |x_i|$;
3. $\|\vec{x}\|_3 = \|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$ — Евклидова норма, называют её *расстоянием* (длиной).

18.03.05

2.1.3 Матрицы $(n \times n)$, основные понятия

1. *Блоком матрицы* (подматрицей) $\mathtt{A} = \{a_{ij}\}_{i,j=1}^{n}$ называется совокупность элементов $\{a_{ij}\}_{i=i_1,\,j=j_1}^{n_1,\,n_2}$, где n_1, $n_2 < n$, а i, $j = 1, 2, \ldots, n$, то есть можно представить матрицу:

$$\mathtt{A} = \begin{pmatrix} \boxed{\begin{matrix} \boxed{a_{11}} & a_{12} \\ a_{21} & a_{22} \end{matrix}} & \begin{matrix} \ldots & a_{1n} \\ \ldots & a_{2n} \end{matrix} \\ \begin{matrix} \vdots & \vdots \end{matrix} & \begin{matrix} \ddots & \vdots \end{matrix} \\ \begin{matrix} a_{n1} & a_{n2} \end{matrix} & \begin{matrix} \ldots & a_{nn} \end{matrix} \end{pmatrix}.$$

2. Матрицу $\mathtt{A}$ называют *транспонированной* и обозначают $\mathtt{A}^T = \mathtt{C}$, если $\mathtt{C}$ такая матрица, что $\mathtt{C} = \{c_{ij}\}_{i,j=1}^{n}$ такие, что $c_{ij} = a_{ji}$, то есть матрица $\mathtt{C}$ — это матрица $\mathtt{A}$, у которой поменяли местами строки и столбцы, примерно так:

$$\mathtt{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \Rightarrow \mathtt{C} = \begin{pmatrix} a_{11} & a_{21} & \ldots & a_{n1} \\ a_{12} & a_{22} & \ldots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \ldots & a_{nn} \end{pmatrix}.$$

3. Матрицей $\mathtt{C} = \alpha \cdot \mathtt{A}$ называют такую матрицу, у которой $c_{ij} = \alpha \cdot a_{ij}$ — операция умножения матрицы на скаляр.
4. Кроме того имеет смысл операция сложения: $\mathtt{C} = \mathtt{A} + \mathtt{B}$, если $c_{ij} = a_{ij} + b_{ij}$ (здесь везде имеется ввиду, что i, $j = 1, 2, \ldots, n$).
5. Вектор $\vec{y} = \mathtt{A} \cdot \vec{x}$, если его компоненты $y_i = \sum_{j=1}^{n} a_{ij} \cdot x_j$.
6. Произведением двух матриц $\mathtt{C} = \mathtt{A} \cdot \mathtt{B}$ (иногда пишется $\mathtt{A} \times \mathtt{B}$) называется такая матрица, где $c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$.
7. Матрица $\mathtt{C} = \mathtt{A}^{-1}$, называемая *обратной*, если $\mathtt{A}^{-1} \cdot \mathtt{A} = \mathtt{A} \cdot \mathtt{A}^{-1} = \mathtt{E}$, где $\mathtt{E}$ — единичная матрица, которая имеет элементы $\mathtt{E} = \{e_{ij}\}_{i,j=1}^{n}$ такие, что

$$e_{ij} = \begin{cases} 1, & i = j; \\ 0, & i \neq j. \end{cases},$$

иногда обозначается

$$\mathtt{I} = \mathtt{E} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{pmatrix}.$$

8. Число D называется *детерминантом* матрицы $\mathtt{A}$: $\det \mathtt{A} = D(\mathtt{A})$, который вычисляется наиболее конструктивно для вычислительных методов таким образом:

   ▷ Для первого порядка:

$$\mathtt{A}^{(1)} = a_{11},\ \det \mathtt{A}^{(1)} = a_{11};$$

▷ Для второго порядка:

$$\mathtt{A}^{(2)} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix},\ \det \mathtt{A}^{(2)} = a_{11} \cdot a_{22} - a_{21} \cdot a_{12};$$

▷ Для третьего порядка:

$$\mathtt{A}^{(3)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix},\ \det \mathtt{A}^{(3)} = a_{11} \cdot \det \mathtt{A}_1^{(2)} - a_{12} \cdot \det \mathtt{A}_2^{(2)} + a_{13} \cdot \det \mathtt{A}_3^{(2)},$$

где $\mathtt{A}_i^{(2)}$ получается из матрицы $\mathtt{A}$ вычёркиванием i-го столбца и i-ой строки.

Вот такое рекуррентное определение, но это скорее теоретическое определение, так как требует $n \cdot n!$ операций для вычисления. Если $\det \mathtt{A} \neq 0$, то такая матрица называется *обратимой*.

9. $\lambda = \lambda(\mathtt{A})$ называется *собственным значением* матрицы, если у уравнения $\mathtt{A} \cdot \vec{x} = \lambda \cdot \vec{x}$ имеется ненулевое решение $\vec{x}$, то есть если привести подобные, мы получаем систему уравнений $(\mathtt{A} - \lambda \cdot \mathtt{E}) \cdot \vec{x} = 0$ и если существует λ, при котором хотя бы одна компонента $\vec{x}$ не равна нулю, то λ называется собственным значением $\mathtt{A}$, а эти ненулевые решения $\vec{x}$ называются *собственными векторами*.

10. *Число обусловленности*, иногда обозначают $\mathrm{Cond}(\mathtt{A}) \equiv C(\mathtt{A}) \equiv \nu(\mathtt{A})$ — очень важное число в вычислительной математике, показывающее, на сколько плохая матрица. Оно показывает следующий фактор: если мы при решении системы уравнений $\mathtt{A} \cdot \vec{x} = \vec{b}$ умножим на $\mathtt{A}^{-1}$, то есть $\mathtt{E} \cdot \vec{x} = \mathtt{A}^{-1} \cdot \vec{b}$, то есть $\mathtt{E} \cdot \vec{x} = \vec{y}$, а $y_i = x_i$, то есть $\mathtt{A}^{-1} \cdot \vec{b}$ — новый вектор с компонентами нашего вектора $\vec{x}$.

    Но найти обратную матрицу трудно, мы так не делаем, это теоретическая часть. Так, когда мы ищем компоненты вектора $\vec{x}$, возникает потребность в арифметических операциях, приходится выполнять десятки миллионов и погрешности вычислений могут накапливаться. Так в вычислительных методах есть анализ вычислительной погрешности, допустим мы вычислили систему уравнений и возникли ошибки округления, мы рассуждаем так: все ошибки округления — те, которые возникли при задании правой части — вектора $\vec{b}$, то есть можно представить $\vec{b}' = \vec{b} + \delta\vec{b}$, где $\delta\vec{b}$ возникает из-за ошибок округления, мы туда сбросим все погрешности, а компьютер считаем идеальным. Тогда в системе

$$\mathtt{A} \cdot \vec{x}' = \vec{b}' = \vec{b} + \delta\vec{b} \tag{2.1.1}$$

    мы получим не $\vec{x}$, а какой-то $\vec{x}'$, а нам-то надо решить другую систему

$$\mathtt{A} \cdot \vec{x} = \vec{b}, \tag{2.1.2}$$

    но от чего зависит разность $\vec{x}' - \vec{x} = \delta\vec{x}$, как они связаны, вот возникает вопрос. Мы рассмотрим разность между (2.1.1) и (2.1.2), приведём подобные и получим $\mathtt{A} \cdot \delta\vec{x} = \delta\vec{b}$ и тогда $\delta\vec{x} = \mathtt{A}^{-1} \cdot \delta\vec{b}$ и норма, то есть величина этой погрешности $\|\delta\vec{x}\|$, зависит от этого вектора и зависит от $\mathtt{A}^{-1}$, надо как-то оценивать матрицу, поэтому оценим связь между погрешностью в решении и погрешностью: $\|\delta\vec{x}\| \leqslant \|\delta\vec{b}\|$ и этот коэффициент надо заранее оценить, для этого возникает потребность вычисления нормы матрицы.

11. Введём аксиомы для классического понятия нормы матрицы:

    (а) $\|\mathtt{A}\| > 0$, если $a_{ij} \neq 0$;

    (б) $\|\alpha \cdot \mathtt{A}\| = |\alpha| \cdot \|\mathtt{A}\|$, где, естественно, α — скаляр;

    (в) $\|\mathtt{A} \cdot \mathtt{B}\| \leqslant \|\mathtt{A}\| \cdot \|\mathtt{B}\|$;

    (г) $\|\mathtt{A} + \mathtt{B}\| \leqslant \|\mathtt{A}\| + \|\mathtt{B}\|$.

    Но как теперь вычислить это число? Мы умеем вычислять нормы векторов, через них и вычислим *подчинённую норму*

$$\|\mathtt{A}\| = \max_{\|x\| \neq 0} \frac{\|\mathtt{A} \cdot \vec{x}\|}{\|\vec{x}\|}$$

    и, оказывается, что если мы возьмём первую норму вектора $\vec{x}$ (максимум по модулю), то

$$\|\mathtt{A}\|_1 = \max_{1 \leqslant i \leqslant n} S_i,\ \text{где}\ S_i = \sum_{j=1}^{n} |a_{ij}|.$$

Так же получается

$$\|\mathtt{A}\|_2 = \max_{1\leqslant j\leqslant n} R_j, \text{ где } R_j = \sum_{i=1}^{n} |a_{ij}|;$$
$$\|\mathtt{A}\|_3 = \max_{1\leqslant i\leqslant n} \lambda_i(\mathtt{A}^T\cdot\mathtt{A}), \text{ где } \lambda_i(\mathtt{A}^T\cdot\mathtt{A})$$

— собственное значение матрицы $\mathtt{A}^T\cdot\mathtt{A}$.

12. Нормы $\|\vec{x}\|$ и $\|\mathtt{A}\|$ называются *согласованными*, если $\|\mathtt{A}\cdot\vec{x}\|\leqslant\|\mathtt{A}\|\cdot\|\vec{x}\|$, то есть это число и эти два числа.

Теперь, раз $\delta\vec{x} = \mathtt{A}^{-1}\cdot\delta\vec{b}$, то если норма согласованная, то $\|\delta\vec{x}\|\leqslant\|\mathtt{A}^{-1}\|\cdot\|\delta\vec{b}\|$, а подчинённая норма всегда согласованная, можно доказать. Обычно рассматривают не саму погрешность, а относительную $\|\delta\vec{x}\|/\|\vec{x}\|$, поэтому перепишем

$$\|\delta\vec{x}\|\leqslant\|\mathtt{A}^{-1}\|\cdot\frac{\|\delta\vec{b}\|}{\|\vec{b}\|}\cdot\|\vec{b}\|\leqslant_{(\|\vec{b}\|\leqslant\|\mathtt{A}\|\cdot\|\vec{x}\|)}\|\mathtt{A}^{-1}\|\cdot\frac{\|\delta\vec{b}\|}{\|\vec{b}\|}\cdot\|\mathtt{A}\|\cdot\|\vec{x}\|$$

и тогда получаю неравенство

$$\frac{\|\delta\vec{x}\|}{\|\vec{x}\|}\leqslant(\|\mathtt{A}^{-1}\|\cdot\|\mathtt{A}\|)\cdot\frac{\|\delta\vec{b}\|}{\|\vec{b}\|}$$

и вот число $C(\mathtt{A}) = \|\mathtt{A}^{-1}\|\cdot\|\mathtt{A}\|$ и называется обусловленностью (condition), вот это — показатель, хорошая наша задача или нет. Если плохая, говорят задача плохо обусловлена.

Существуют свойства $C(\mathtt{A})$:

(а) $C(\mathtt{A})\geqslant 1$ всегда, действительно $C(\mathtt{E}) = 1 = C(\mathtt{A}\cdot\mathtt{A}^{-1})\leqslant\|\mathtt{A}\|^2\cdot\|\mathtt{A}^{-1}\|^2$, это свойство очень тревожное.

(б) $C(\mathtt{A})\leqslant|\lambda_{max}(\mathtt{A})|/|\lambda_{min}(\mathtt{A})|$ — легко доказывается.

(в) Есть свойство $C(\mathtt{A}\cdot\mathtt{B})\leqslant C(\mathtt{A})\cdot C(\mathtt{B})$.

Возникают некоторые вопросы, $C(\mathtt{A}) = \|\mathtt{A}^{-1}\|\cdot\|\mathtt{A}\|$, умножим на α матрицу $\mathtt{A}$, а вдруг показатель улучшится: $C(\alpha\cdot\mathtt{A}) = C(\mathtt{A})$, если бы была плохая матрица, вы её не улучшите. Например для матрицы $n\times n$

$$\begin{pmatrix} 1/2 & 0 & \dots & 0\\ 0 & 1/2 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & \dots & 0 & 1/2\end{pmatrix}$$

$\det\mathtt{A} = 1/2^n$ — очень маленькое число, здесь компонента $x_i = 2\cdot(b_i+\delta b_i)$, то есть $\delta\vec{x} = 2\cdot\delta\vec{b}$, то есть малый определитель ничего не значит. Рассмотрим

$$\mathtt{A} = \begin{pmatrix} 1 & -1 & \dots & -1\\ 0 & 1 & \dots & -1\\ \vdots & \vdots & \ddots & \vdots\\ 0 & \dots & 0 & 1\end{pmatrix},$$

для неё $\det\mathtt{A} = 1$, а число обусловленности ужасное и ошибка возрастает на 2^n.

2.2 Прямые методы и особые применения к решению систем линейных уравнений

Под *прямыми методами* (прочными) называются такие методы решения $\mathtt{A}\cdot\vec{x} = \vec{b}$, которые позволяют получить решение компонент вектора $\vec{x}$ за конечное число арифметических операций. Но с практической точки зрения существуют такие, что нужно конечное число операций, но на компьютере не вычисляется. Например правило Крамера: $x_i = D_i/D$, где $D = \det\mathtt{A}$, а $D_i = \det\mathtt{A}_i$, где $\mathtt{A}_i$ — полученные из матрицы $\mathtt{A}$ заменой i-го столбца вектором $\vec{b}$. Но для матрицы 20×20 понадобится порядка 10^6 лет, конечное число операций, но долговато.

У нас есть система $\mathtt{A}\cdot\vec{x} = \vec{b}$, представив $\mathtt{A} = \mathtt{A}_1\cdot\mathtt{A}_2$, где множители $\mathtt{A}_1$ и $\mathtt{A}_2$ выбраны таким образом, чтобы легко было обратить. Получается $\mathtt{A}_1\cdot\mathtt{A}_2\cdot\vec{x} = \vec{b}$ и математически вычисляется $\vec{y} = \mathtt{A}_2\cdot\vec{x}$ и получаем систему уравнений $\mathtt{A}_1\cdot\vec{y} = \vec{b}$, то есть $\vec{y} = \mathtt{A}_1^{-1}\cdot\vec{b}$ и теперь систему уравнений $\mathtt{A}_2\cdot\vec{x} = \vec{y}$ я решаю как $\vec{x} = \mathtt{A}_2^{-1}\cdot\vec{y}$. Таким образом нужны специального вида матрицы.

2.2.1 Матрицы специального вида

К ним будем стараться привести:

1. Диагональные (D-матрицы), обозначается $\mathtt{D} = \operatorname{Diag}(d_i)_{i=1}^n$ и они имеют вид

$$\begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & d_n \end{pmatrix},$$

   то есть кроме диагональных, все остальные элементы равны нулю. Интересные свойства:

   (а) $\alpha \cdot \mathtt{D} = \mathtt{D}_1$ — тоже диагональная, где $d_i^{(1)} = \alpha \cdot d_i$;

   (б) $\mathtt{D}^{(1)} + \mathtt{D}^{(2)} = \mathtt{D}^{(3)}$ — по правилу сложения;

   (в) $\mathtt{D}^{(1)} \cdot \mathtt{D}^{(2)} = \mathtt{D}^{(3)}$ — диагональная;

   (г) Определитель $\det \mathtt{D} = d_1 \cdot d_2 \cdot \ldots \cdot d_n$.

   Как решаются системы уравнений таких матриц (мечта для вычислителей): $\mathtt{D} \cdot \vec{x} = \vec{b} \Rightarrow x_i = b_i/d_i$, $i = 1, 2, \dots, n$, то есть уходит число операций порядка $n = M_0$, решаются мгновенно.

2. Треугольные (T-матрицы) $\mathtt{T} = \{t_{ij}\}_{i,j=1}^n$, различают двух типов: верхнюю треугольную и нижнюю. Нижняя треугольная имеет вид

$$\mathtt{T} = \begin{pmatrix} t_{11} & 0 & \dots & 0 \\ t_{21} & t_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \dots & t_{nn} \end{pmatrix}, \text{ где } t_{ij} = \begin{cases} 0, & j > i; \\ t_{ij}, & j \leqslant i. \end{cases}.$$

   Аналогично введём понятие верхней треугольной (иногда их называют правой и левой). Свойства:

   (а) $\det \mathtt{T} = t_{11} \cdot t_{22} \cdot \ldots \cdot t_{nn}$;

   (б) $\mathtt{T}^{(1)} + \mathtt{T}^{(2)} = \mathtt{T}^{(3)}$;

   (в) $\alpha \cdot \mathtt{T} = \mathtt{T}^{(2)}$;

   (г) $\mathtt{T}^{(1)} \cdot \mathtt{T}^{(2)} = \mathtt{T}^{(3)}$ — того же типа;

   (д) $\mathtt{T}^{-1} = \mathtt{T}^{(2)}$ — того же типа.

   Системы уравнений выглядят как

$$\begin{aligned} t_{11}x_1 + t_{12}x_2 + t_{13}x_3 + \ldots + t_{1n}x_n &= b_1, \\ t_{22}x_2 + t_{23}x_3 + \ldots + t_{2n}x_n &= b_2, \\ &\vdots \\ t_{nn}x_n &= b_n. \end{aligned}$$

   и решение ищется в помощью алгоритма: из последнего уравнения легко вычисляется, что $x_n = b_n/t_{nn}$, из предпоследнего тогда можно вычислить

$$x_{n-1} = \frac{1}{t_{n-1\,n-1}} \cdot (b_{n-1} - t_{n-1\,n} \cdot x_n)$$

   и так далее, это называется *алгоритм бегущего счёта*:

$$x_i = \frac{1}{t_{i\,i}} \cdot \left(b_i - \sum_{k=i+1}^{n} t_{i\,k} \cdot x_k \right).$$

   Потребуется операций порядка n^2; если матрица нижняя треугольная, то алгоритм называется *прямым*.

3. Ортогональные — матрица $\mathtt{Q}$, если $\mathtt{Q}^{-1} = \mathtt{Q}^T$, тогда уравнение $\mathtt{Q} \cdot \vec{x} = \vec{b}$ умножаем на $\mathtt{Q}^T$ и получаем, что $\mathtt{E} \cdot \vec{x} = \mathtt{Q}^T \cdot \vec{b} = \vec{b}'$, то есть $x_i = b'_i$. Обладает некоторыми свойствами:

   (а) $\det \mathtt{Q} = \pm 1$;

   (б) $\|\mathtt{Q} \cdot \vec{x}\|_3 = \|\vec{x}\|_3$ (сумма квадратов);

(в) $\mathtt{Q}^{(1)} \cdot \mathtt{Q}^{(2)} = \mathtt{Q}^{(3)}$;

(г) А вот сумма $\mathtt{Q}^{(1)} + \mathtt{Q}^{(2)} \neq \mathtt{Q}^{(3)}$.

4. Симметричные матрицы: $\mathtt{A}$ такая, если она совпадает с $\mathtt{A}^T$. Введём понятие *положительной определённости*, то есть матрица называется положительно определённой, если $(\mathtt{A} \cdot \vec{x}, \vec{x}) = (\vec{x}, \mathtt{A} \cdot \vec{x}) > 0$, при $\vec{x} \neq 0$. У них собственные числа $\lambda(\mathtt{A})$ всегда вещественные, кроме того у положительно определённых матриц $\lambda(\mathtt{A}) > 0$. Может быть представлена в виде $\mathtt{S}^T \cdot \mathtt{S} = \mathtt{P} \cdot \mathtt{P}^T$, где $\mathtt{S}$ или $\mathtt{P}$ — треугольные матрицы. Кроме того $\mathtt{A} = \mathtt{Q}^T \cdot \Lambda \cdot \mathtt{Q}$, где $\mathtt{Q}$ — ортогональная матрица, а Λ — диагональная, то есть $\Lambda = \operatorname{Diag}(\lambda_i^{\mathtt{A}})_{i=1}^n$, где $\lambda_i^{\mathtt{A}}$ — собственные значения матрицы $\mathtt{A}$.

   Для решения системы $\mathtt{A} \cdot \vec{x} = \vec{b}$ приведём к произведению треугольных или диагональных матриц и считаем $\mathtt{A} \cdot \vec{x} = \vec{b}$: умножим на $\mathtt{A}^T$, получу новую систему $\mathtt{A}' \cdot \vec{x} = \vec{b'}$, где $\mathtt{A}' = \mathtt{A}^T \cdot \mathtt{A} = \mathtt{A} \cdot \mathtt{A}^T$ и мы получаем новую задачу с симметричными матрицами, но не всегда продуктивно, умножать много надо.

23.03.05

Нет и вряд ли будут такие ЭВМ, которые позволили бы вычислять теоретическими методами и так решать уравнения, пусть система $\mathtt{A} \cdot \vec{x} = \vec{b}$, тогда $\mathtt{E} \cdot \vec{x} = \mathtt{A}^{-1} \cdot \vec{b}$ и

$$x_i = \sum_{j=1}^{n} a_{ij}^{-1} \cdot b_j,$$

где a_{ij}^{-1} — это элементы матрицы $\mathtt{A}^{-1}$, это не есть $1/a_{ij}$. Как я всегда говорил, нормальные герои идут в обход, выберем $\mathtt{A} = \mathtt{A}^{(1)} \cdot \mathtt{A}^{(2)}$, чтобы они были удобного вида, который мы изучали, то есть стараемся свести задачу, заводим переменную $\vec{y} = \mathtt{A}^{(2)} \cdot \vec{x}$, тогда $\mathtt{A}^{(1)} \cdot \vec{y} = \vec{b} \Rightarrow \vec{y} = (\mathtt{A}^{(1)})^{-1} \cdot \vec{b}$, откуда $\vec{x} = (\mathtt{A}^{(2)})^{-1} \cdot \vec{y}$, то есть задача разбивается на две через вспомогательную переменную $\vec{y}$, если так разложить $\mathtt{A}$ получится. Иногда делают $\mathtt{A}^{(2)} \cdot \vec{x} = (\mathtt{A}^{(1)})^{-1} \cdot \vec{b}$, мы разные будем рассматривать способы. Все эти матрицы будут невырожденными, так как $\det(\mathtt{A} \cdot \mathtt{B}) = \det \mathtt{A} \cdot \det \mathtt{B}$ и если хоть одна вырожденная, то и произведение будет вырожденное.

2.2.2 Критерий невырожденности матриц

Есть критерий необходимый и достаточный, рассмотрим:

Необходимые условия

1. Матрица $\mathtt{A}$ невырожденная, если её строки образуют линейно независимую систему векторов;
2. Матрица $\mathtt{A}$ невырожденная, если её столбцы образуют линейно независимую систему векторов;
3. $\det \mathtt{A} \neq 0$;
4. $\lambda_i(\mathtt{A}) \neq 0$, $i = 1, 2, \ldots, n$, так как определитель равен произведению собственных значений.

Достаточные условия

Теорема 2.2.2.1 (Адамара)**.**

▷ Звучит примерно так: если диагональные элементы

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}} |a_{ij}| = S_i,$$

то матрица невырожденная.

Следствие 2.2.2.2 (По столбцам).

▷ Если

$$|a_{jj}| > \sum_{\substack{i=1 \\ j \neq i}} |a_{ij}| = S_j,$$

то матрица невырожденная.

▷ **Доказательство**.

∘ Доказывается от противного и используется такой критерий: матрица $\mathtt{A}$ невырожденная, если в решении $\mathtt{A} \cdot \vec{x} = 0$ решение только $\vec{x} = \vec{0}$.

□

Вот Ольга Таусски обобщила эту теорему, добавила знак $\geqslant$, но пришлось добавить, что должно выполняться строго хотя бы для одного i и матрица должна быть неразложимой (матрица $\mathtt{A}$ — неразложима, если она не может быть представлена с помощью элементарных преобразований в виде

$$\mathtt{A} = \begin{pmatrix} \mathtt{A}_{11} & \mathtt{A}_{12} \\ 0 & \mathtt{A}_{22} \end{pmatrix},$$

где $\mathtt{A}_{ij}$ — это блоки матрицы).

Вооружившись такими знаниями мы попытаемся разобрать самый популярный метод:

2.3 Метод Гаусса

Мы несколько их рассмотрели, но основной — схема единственного деления.

Теорема 2.3.1 (LU-разложение матрицы)**.**

▷ Эта теорема звучит так и должна была доказываться на линейной алгебре: если главные миноры матрицы $\mathtt{A}$ отличаются от нуля, то матрица $\mathtt{A}$ может быть представлена в виде произведения $\mathtt{L} \cdot \mathtt{U}$, где $\mathtt{L}$ — нижняя треугольная матрица (иногда обозначается $\mathtt{L} = (*\backslash 0|)$), а $\mathtt{U}$ — верхняя. Если элементы матрицы $\mathtt{U}$ — $u_{ii} = 1$, то это разложение единственно.

Здесь мы встретились с главными минорами M_i — называются определители подматриц матрицы $\mathtt{A}$, то есть $M_m = \det \mathtt{A}^{(m)}$, где $\mathtt{A}^{(m)} = (a_{ij})_{i,j=1}^{m}$, то есть можно нарисовать как

$$\mathtt{A} = \begin{pmatrix} \boxed{\begin{matrix} \boxed{a_{11}} & a_{12} \\ a_{21} & a_{22} \end{matrix}} & \dots & a_{1n} \\ \vdots \ \vdots & \ddots & \vdots \\ a_{n1} \quad a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Матрица с *диагональным преобладанием* (то есть которая удовлетворяет условию Адамара), легко видеть (как говорят математики — очевидно), имеют ненулевые главные миноры.

Рассмотрим алгоритм:

▷ Прямой ход (здесь надо свести систему уравнений к $\mathtt{U} \cdot \vec{x} = \vec{b}'$, где $\mathtt{U}$ — верхняя треугольная).

1. Рассматривается первое решение системы $\mathtt{A} \cdot \vec{x} = \vec{b}$ или

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

где $a_{11} \neq 0$ — первый минор (мы знаем из теоремы), тогда я могу всё это разделить на a_{11}:

$$x_1 + a_{12}^{(1)} \cdot x_2 + \ldots + a_{1n}^{(1)} \cdot x_n = b_1^{(1)},$$

где $a_{1j}^{(1)} = a_{1j}/a_{11}$, $j = 2, 3, \ldots, n$ и $b_1^{(1)} = b_1/a_{11}$ (j от двух, так как можно при делении не получить при x_1 единичку, которая нам нужна, с данными надо обращаться очень аккуратно).

Для $i \geqslant 2$:

$$a_{i2}^{(1)} \cdot x_2 + a_{i3}^{(1)} \cdot x_3 + \ldots + a_{in}^{(1)} \cdot x_n = b_i^{(1)},$$

то есть исключаем из всех уравнений переменную x_1, каким образом: от i-ой строки вычтем первую, умноженную на a_{i1}, как при этом вычисляются коэффициенты:

$$\mathtt{A}^{(1)} = \begin{pmatrix} 1 & * & * & \dots & * \\ 0 & * & * & \dots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & * & * & \dots & * \end{pmatrix},$$

то есть по формуле

$$a_{ij}^{(1)} = a_{ij} - a_{i1} \cdot \frac{a_{1j}}{a_{11}},$$

где $2 \leqslant i \leqslant n$, а $j = 2,\ 3,\ \ldots,\ n$, правая часть тоже

$$b_i^{(1)} = b_i - a_{i1} \cdot \frac{b_1}{a_{11}}.$$

2. Рассматривается второе уравнение системы с матрицей уже $\mathtt{A}^{(1)}$. В этом уравнении каждый элемент матрицы (слагаемое) делится на a_{22}, то есть

$$x_2 + a_{23}^{(2)} \cdot x_3 + \ldots + a_{2n}^{(2)} \cdot x_n = b_2^{(2)},$$

где $a_{22}^{(1)}$ по формуле равняется

$$a_{22}^{(1)} = \frac{a_{22} \cdot a_{11} - a_{12} \cdot a_{21}}{a_{11}}$$

— минор второго порядка, делёный на минор первого порядка, но $M_1 \neq 0$, следовательно имею формально право разделить:

$$a_{2j}^{(2)} = \frac{a_{2j}^{(1)}}{a_{22}^{(1)}},\ j = 3,\ 4,\ \ldots,\ n, \quad b_2^{(2)} = \frac{b_2^{(1)}}{a_{22}^{(1)}},$$

то есть мы повторяем алгоритм для следующей строки и так далее до n тем же способом, что и в предыдущих случаях. Делим на a_{i2}, в итоге

$$\begin{cases} x_1 + a_{12}^{(1)} \cdot x_2 + a_{13}^{(1)} \cdot x_3 + \ldots + a_{1n}^{(1)} \cdot x_n = b_1^{(1)}, \\ \qquad\quad x_2 + a_{23}^{(2)} \cdot x_3 + \ldots + a_{2n}^{(2)} \cdot x_n = b_2^{(2)}, \\ \qquad\qquad\qquad \vdots \\ \qquad\qquad\qquad\qquad\quad x_n = b_n^{(n)}. \end{cases}$$

Пусть мы получили систему уравнений с матрицей

$$\mathtt{A}^{(k)} = \begin{pmatrix} \boxed{\begin{matrix} 1 & & * \\ & \ddots & \\ 0 & & 1 \end{matrix}} & * \\ 0 & \boxed{a_{k+1\,k+1}^{(k)}} \end{pmatrix},$$

то есть можно представить в виде

$$\begin{pmatrix} \mathtt{A}_{11}^{(k)} & \mathtt{A}_{12}^{(k)} \\ \mathtt{A}_{21}^{(k)} & \mathtt{A}_{22}^{(k)} \end{pmatrix}$$

— через блоки, где блок $\mathtt{A}_{22}$ называется *активным блоком* матрицы $\mathtt{A}$, а все остальные называются *пассивными*, почему так? Потому что при дальнейших преобразованиях элементы пассивных матриц не меняются, меняются у активной матрицы. Матрица $\mathtt{A}_{11}$ — верхняя диагональная с единицами на диагонали, а $\mathtt{A}_{21}$ вообще нулевая.

k+1. Исключение из всех строк с номерами большими $(k+1)$ — x_{k+1}. Мы рассматриваем $(k+1)$-ое уравнение, оно имеет вид

$$a_{k+1\,k+1}^{(k)} \cdot x_{k+1} + a_{k+1\,k+2}^{(k)} \cdot x_{k+2} + \ldots + a_{k+1\,n}^{(k)} \cdot x_n = b_{k+1}^{(k)},$$

поступаем по аналогии с предыдущими шагами (в компьютере это цикл), $a_{k+1\,k+1}^{(k+1)}$ делаем единицей, а остальные

$$a_{k+1\,j}^{(k+1)} := \frac{a_{k+1\,j}^{(k)}}{a_{k+1\,k+1}^{(k)}},\ j = k+1,\ k+2,\ \ldots,\ n;\ b_{k+1}^{(k+1)} = \frac{b_{k+1}^{(k)}}{a_{k+1\,k+1}^{(k)}}.$$

Получаются уравнения

$$\begin{cases} x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + a_{14}^{(1)}x_4 + \ldots + a_{1n}^{(1)}x_n = b_1^{(1)} \\ \vdots \\ x_{k+1} + a_{k+1\,k+2}^{(k+1)}x_{k+2} + \ldots + a_{k+1\,n}^{(k+1)}x_n = b_{k+1}^{(k+1)} \\ a_{k+2\,k+1}^{(k)}x_{k+1} + a_{k+2\,k+2}^{(k)}x_{k+2} + \ldots + a_{k+2\,n}^{(k)}x_n = b_{k+2}^{(k)} \\ \vdots \\ a_{n\,k+1}^{(k)}x_{k+1} + a_{n\,k+2}^{(k)}x_{k+2} + \ldots + a_{n\,n}^{(k)}x_n = b_n^{(k)} \end{cases}$$

И приступаем исключать x_{k+1} из всех уравнений с номерами $i > k+1$ таким же способом: умножаем на $a_{i\,k+1}^{(k)}$ и получаем

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - a_{i\,k+1}^{(k)} \cdot \frac{a_{k+1\,j}^{(k)}}{a_{k+1\,k+1}^{(k)}}$$

и правая часть

$$b_i^{(k+1)} = b_i^{(k)} - a_{i\,k+1}^{(k)} \cdot \frac{b_{k+1}^{(k)}}{a_{k+1\,k+1}^{(k)}},$$

здесь $k+2 \leqslant i \leqslant n$, а $k+2 \leqslant j \leqslant n$.

n-1. На предпоследнем шаге мы получаем матрицу

$$\mathtt{A}^{(n-1)} = \begin{pmatrix} 1 & & * & \\ & \ddots & & * \\ 0 & & 1 & \\ & & & a_{nn}^{(n-1)} \end{pmatrix}.$$

n. На последнем шаге $\mathtt{A} = \mathtt{U}$, причём диагональные элементы равны единице и надо решить систему уравнений $\mathtt{U} \cdot \vec{x} = \vec{b}^{(n)}$. Мы умеем это решать методом обратной подстановки; возникает вопрос: а где же матрица $\mathtt{L}$? Она у нас в процессе вычислялась. Когда мы делили и отнимали — это просто матричные преобразования, чтобы умножить строку матрицы на число λ, надо умножить эту матрицу на диагональную с элементами

$$\begin{pmatrix} 1 & 0 & \ldots & \ldots & 0 \\ 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \lambda & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & \ldots & 0 & 1 \end{pmatrix}$$

А теперь что значит сложить в матрице строку со следующей? Нужно в позицию a_{ij} поставить единицу, то есть $\mathtt{A}^{(1)} = \mathtt{L}^{(1)} \cdot \mathtt{A}$, где

$$\mathtt{L}^{(1)} = \begin{pmatrix} 1/a_{11} & & 0 & \\ -a_{21}/a_{11} & 1 & & \\ & & 1 & \\ 0 & & & \ddots \end{pmatrix}$$

И матрица $\mathtt{L}^{(n)} \cdot \mathtt{L}^{(n-1)} \cdot \ldots \cdot \mathtt{L}^{(1)}$ — будет обратной к $\mathtt{L}$ из нашей теоремы. И она тоже нижняя треугольная. Если матрицу запоминать, это называется *элиминативной формой* сохранения (есть такая книжка Куарсен — „Разреженные матрицы“, вот там это всё подробно).

▷ Обратный ход (решение системы с обратной матрицей $\mathtt{U}^{-1} \cdot \vec{b}$).

2.3.2 Заключение

При реализации этого метода возникает такая проблема: деление на диагональные элементы $a_{k+1\,k+1}^{(k)}$, но если матрица имеет главные миноры отличные от нуля, то и они отличаются, можно показать, но формально в компьютере 10^{-18} — отличается от нуля, но начинаешь делить и получаешь проблемы. Так вот есть модификации метода Гаусса и работают даже для главного минора равного нулю. Как приспособить:

2.4 Модификации метода Гаусса

Нельзя делить на малые числа, например

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \vec{b}$$

— имеет решение $x_1 = 1$ и $x_2 = 1$ — легко видеть, а методом Гаусса не можем решать, так как матрица вырожденная, $a_{11} = 0$, $\det \mathtt{A} = -1$. Та же проблема, когда $a_{11} = \varepsilon$, разделим и получаем огромную ошибку.

2.4.1 Метод Гаусса с перестановками

Здесь элементы $a^{(k)}_{k+1\,k+1}$ называются *ведущими*, обозначим исходную матрицу $\mathtt{A} = \mathtt{A}^{(0)}$, есть несколько модификаций:

1. Выбор ведущего элемента по столбцам (с выбором), алгоритм такой: рассматриваем первое уравнение системы, если $a_{11} = 0$ или по модулю $< \varepsilon$, то мы переставляем уравнения, то есть перегруппировываем чтобы первое уравнение было то, у которого элемент $a_{i1} \neq 0$ и $|a_{i1}| > \varepsilon$. Давайте тогда возьмём не просто так, а максимальное по модулю значение (цикл по первому столбцу), пусть i_*, назовём его ведущим (программисты делают шкалу перемещений, матрицу не трогают, у программистов мозги набекрень). Надо поменять местами в столбце $\vec{b}$, надо послать куда надо (если вы не поняли, что это значит, то переслать).

   И по аналогии с остальными столбцами.

2. Модификация по строкам (ведь строки и столбцы у матрицы равноправные). Мы начинаем с

   $$a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n = b_1$$

   и нам не нравится a_{11}, тогда можно найти в этой строке максимальный по модулю элемент, мы переставляем этот столбец j_*, это эквивалентно перенумерованию неизвестных, то есть x_{j_*} становится x_1.

3. Ещё один, называется *методом оптимальноо исключения* в смысле выделения памяти, так как расходуется только на одну строку. Сейчас проблемы с памятью нет и это используется, когда не известны все элементы сразу, вычисляет по ходу получения. 30.03.05

   Шаги алгоритма:

   (а) Разделим на диагональный элемент первое уравнение:

   $$\begin{aligned} x_1 + a^{(1)}_{12} x_2 + \ldots + a^{(1)}_{1n} x_n &= b^{(1)}_1, \\ a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n &= b_2. \end{aligned}$$

   (б) С помощью комбинации первого и второго уравнений, устраняем неизвестную компоненту x_1 из второго, понятно, изменяем матрицу и правый коэффициент, потом делим на $a^{(1)}_{22}$ и получаем уравнение в виде

   $$x_2 + a^{(2)}_{23} x_3 + \ldots + a^{(2)}_{2n} x_n = b^{(2)}_2.$$

   (в) Теперь вызывается строка

   $$a_{31} x_1 + a_{32} x_2 + \ldots + a_{3n} x_n = b_3$$

   и с помощью первых двух уравнений из неё устраняются x_1 и x_2 и получаем

   $$x_3 + a^{(3)}_{34} x_4 + \ldots + a^{(3)}_{3n} x_n = b^{(3)}_3$$

   и так далее (как говорят математики, а программисты делают цикл).

   (k) Матрица

   $$\mathtt{A}^k = \begin{pmatrix} 1 & & * & & \\ & \ddots & & * & \\ 0 & & 1 & & \\ & \ldots & & (a_{ij}) & \end{pmatrix},$$

   где до $k+1$ строки — пассивная часть. Начиная с $(k+1)$-ой строки мы имеем активную часть и уравнение

   $$a_{k+1\,1} x_1 + a_{k+1\,2} x_2 + \ldots + a_{k+1\,n} x_n = b_{k+1},$$

исключаем x_1, x_2, …, x_k и получаем

$$\bar{a}^{(k+1)}_{k+1\,k+1}x_{k+1} + \bar{a}^{(k+1)}_{k+1\,k+2}x_{k+2} + \ldots + \bar{a}^{(k+1)}_{k+1\,n}x_n = \bar{b}^{(k+1)}_{k+1},$$

где $\bar{a}$ — предварительные результаты, делим на $\bar{a}^{(k+1)}_{k+1\,k+1}$ и получаем

$$x_{k+1} + a^{(k+1)}_{k+1\,k+2}x_{k+2} + \ldots + a^{(k+1)}_{k+1\,n}x_n = b^{(k+1)}_{k+1}.$$

2.4.2 Метод Жордана

Обычно реализуется в математических пакетах с некоторыми модификациями, я расскажу классический. Он соединяет прямой и обратный ход, шаги алгоритма:

1. Приводим первое уравнение к виду

$$x_1 + a^{(1)}_{12}x_2 + a^{(1)}_{13}x_3 + \ldots + a^{(1)}_{1n}x_n = b^{(1)}_1.$$

2. Рассмотрим второе уравнение

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2,$$

умножим первое уравнение на a_{21} и вычтем отсюда:

$$a^{(2)}_{22}x_2 + a^{(2)}_{23}x_3 + \ldots + a^{(2)}_{2n}x_n = b^{(2)}_2,$$

приведём его к виду, у которого единицы на диагонали, а остальные элементы в общем случае не нули. А теперь исключим x_2 из первого уравнения:

$$\begin{cases} a^{(2)}_{11}x_1 + a^{(2)}_{13}x_3 + \ldots + a^{(2)}_{2n}x_n = b^{(2)}_1, \\ a^{(2)}_{22}x_2 + a^{(2)}_{13}x_3 + \ldots + a^{(2)}_{2n}x_n = b^{(2)}_2. \end{cases}$$

и так далее.

3. Рассмотрим третье уравнение

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots + a_{3n}x_n = b_3,$$

преобразуем и исключим что получится из всех с номерами меньшими трёх (то есть из первого и второго) и, как говорят математики, и так далее.

n. В результате получаем

$$\begin{pmatrix} * & & \\ & \ddots & \\ & & * \end{pmatrix} = \begin{pmatrix} b^{(n)}_1 \\ \vdots \\ b^{(n)}_n \end{pmatrix},$$

то есть мы приводим не к верхне-диагональному виду, а к диагональному.

2.4.3 Метод квадратного корня

Для симметричных матриц часто применяется, число операций в 2 раза меньше, чем у Гаусса, так как использует симметрию, но зато использует операцию вычисления корня, а это намного дороже умножения.

Идея, что если $\mathtt{A} = \mathtt{A}^T$ — симметричная, то она может быть представлена в виде двух матриц: $\mathtt{A} = \mathtt{S}^T \cdot \mathtt{S}$, где $\mathtt{S}$ — верхне-треугольная (можно наоборот), то есть $\mathtt{S}$ имеет вид $(0 \backslash * |)$. Тогда идея решения уравнения $\mathtt{A} \cdot \vec{x} = \vec{b}$ сводится к $\mathtt{S}^T \cdot \mathtt{S} \cdot \vec{x} = \vec{b}$, введём вспомогательную переменную $\mathtt{S} \cdot \vec{x} = \vec{y}$, решаем с помощью прямой подстановки

$$\mathtt{S}^T \cdot \vec{y} = \vec{b} \Rightarrow \vec{y} = (\mathtt{S}^T)^{-1} \cdot \vec{b}$$

(просто так обозначили, мы искать обратную матрицу не будем) и решаем с помощью метода обратной подстановки $\vec{x} = \mathtt{S}^{-1} \cdot \vec{y}$, требует порядка n^2 операций и мы это рассмотрим.

Проблема в том, как найти коэффициенты

$$s_{ij} = \begin{cases} 0, & i > j; \\ *, & i \leqslant j. \end{cases},$$

наша задача их найти. Считаем, что матрица положительно определённая, чтобы $\mathtt{A} = \mathtt{S}^T \cdot \mathtt{S}$ вычислялась, иначе надо вычислять знаки на диагонали и $\mathtt{A} = \mathtt{S}^T \cdot \mathtt{D} \cdot \mathtt{S}$, где $\mathtt{D}$ — имеет элементы ± 1. Что такое $\mathtt{A} > 0$, забыл человек, это когда $(\vec{x},\, \mathtt{A} \cdot \vec{x}) > 0$.

Алгоритм называется прямым треугольным, $\mathtt{S}^T$ имеет вид

$$\begin{pmatrix} s_{11} & & 0 \\ \vdots & \ddots & \\ s_{1n} & \dots & s_{nn} \end{pmatrix},$$

умножаем на $\mathtt{S}^T$:

$$\begin{pmatrix} s_{11} & & 0 \\ \vdots & \ddots & \\ s_{1n} & \dots & s_{nn} \end{pmatrix} \cdot \begin{pmatrix} s_{11} & \dots & s_{1n} \\ & \ddots & \vdots \\ 0 & & s_{nn} \end{pmatrix} = \mathtt{A} = \begin{pmatrix} a_{11} & & * \\ & \ddots & \\ * & & a_{nn} \end{pmatrix}$$

— симметричная, есть даже алгоритм Дулита-Крауна, но он не хороший, не русский, я его вам рассказывать не буду. Вот так с детской непосредственностью умножаем матрицу на матрицу и приравниваем:

$$s_{11}^2 = a_{11} \Rightarrow s_{11} = \sqrt{a_{11}},$$

теперь $s_{11} \cdot s_{12} = a_{12}$, но s_{11} мы уже нашли, следовательно

$$s_{12} = \frac{a_{12}}{s_{11}},\ s_{1j} = \frac{a_{1j}}{s_{11}} - \forall j \geqslant 2,\ s_{ii} = \left(a_{ii} - \sum_{k=1}^{n} s_{ki}^2 \right)^{1/2},$$
$$s_{ij} = \frac{1}{s_{11}} \cdot \left(a_{ij} - \sum_{k=1}^{i-1} s_{ki} \cdot s_{kj} \right),\ j = i+1,\ i+2,\ \dots,\ n.$$

Это основные модификации, которые вы можете встретить в литературе, основные способы. Для матриц порядка $> 100 \times 100$ их использовать не имеет смысла, более эффективны итерационные методы, особенно для разреженных матриц, когда много нулей. Иногда используют процедурный метод, когда преобразуют преобразованием Хаус-Холдера уравнения. Правда это преобразование тоже дорогое, за всё надо платить и решать, что дороже, наше время или машины.

2.5 Метод исключающий неизвестные в случае трёхдиагональной матрицы (метод прогонки)

Это так в России он называется, *метод прогонки*, в английской литературе W-sweet, где sweet — приятно, иногда по научному „методом факторизации“, W-Voliaj — на французском (один француз к нам приезжал как-то, несколько слов знал по-русски — спасибо и прогонка). Не понятно, украли ли американцы, но открыли два советских учёных Гельфан и кто-то ещё. Годунов-Ряденький „Введение в теорию разностных схем“ в 62 году в приложении впервые этот метод опубликовал, а был он для оборонных целей. Есть несколько модификаций, для пяти-диагональных матриц и другие.

В трёх-диагональной матрице уравнения имеют вид

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + 0 + \ldots + 0 & = b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + 0 + \ldots + 0 & = b_2, \\ \vdots & \\ 0 + \ldots + 0 + a_{i\,i-1}x_{i-1} + a_{ii}x_i + a_{i\,i+1}x_{i+1} + 0 + \ldots + 0 & = b_i, \\ \vdots & \\ 0 + 0 + \ldots + 0 + \ldots \quad a_{n\,n-1}x_{n-1} + a_{nn}x_n & = b_n. \end{cases},$$

то есть матрица

$$\mathtt{A} = \begin{pmatrix} a_{11} & a_{12} & & 0 \\ a_{21} & a_{22} & \ddots & \\ & \ddots & \ddots & a_{n-1\,n} \\ 0 & & a_{n\,n-1} & a_{nn} \end{pmatrix}$$

и a_{11}, a_{22}, …, a_{nn} называется *главной диагональю*. Писать здесь два индекса обычно не очень удобно, то есть элементов всего $3n$, поэтому переобозначают и записывают в виде

$$\begin{cases} B_1x_1 - C_1x_2 & = F_1, \\ -A_2x_1 + B_2x_1 - C_2x_3 & = F_2, \\ \vdots & \\ -A_ix_{i-1} + B_ix_i - C_ix_{i+1} & = F_i, \\ \vdots & \\ -A_nx_{n-1} + b_nx_n & = F_n. \end{cases}$$

и матрица `A` получается вида

$$\mathtt{A} = \begin{pmatrix} B_1 & -C_1 & & 0 \\ -A_2 & B_2 & -C_2 & \\ & \ddots & \ddots & \\ 0 & & -A_n & B_n \end{pmatrix}.$$

Можно указать, что это частный случай Гаусса, но появляется не так, через дифференциальные уравнения, есть даже преобразование Рекатти. Просто когда много нулей, то не надо на них умножать и, не дай Бог, делить. Шаги алгоритма:

▷ Прямая прогонка, здесь преобразуем матрицу к двух-диагональному виду.

1. Приводим первое уравнение переносом второго слагаемого, делённого на B_1, к виду

$$x_1 = \alpha_1x_2 + \beta_1,$$

где $\alpha_1 = C_1/B_1$, а $\beta_1 = F_1/B_1$. Рассмотрим второе уравнение и исключим переменную x_1 с помощью первого уравнения:

$$-A_2 \cdot (\alpha_1x_2 + \beta_1) + B_2x_2 - C_2x_3 = F_2,$$

приведём подобные и преобразуем: $x_2 = \alpha_2x_3 + \beta_2$, где

$$\alpha_2 = \frac{C_2}{B_2 - A_2\alpha_1},\ \beta_2 = \frac{F_2 + A_2\beta_1}{B_2 - A_2\alpha_1}.$$

Теперь рассмотрим третье уравнение и с ним проведём те же исключения, а общую формулу получим с помощью индукции, мы говорим, что $x_{i-1} = \alpha_{i-1}x_i + \beta_{i-1}$ (я могу так говорить, так как первый шаг индукции уже выполнен). Тогда в уравнении $-A_ix_{i-1} + B_ix_i - C_ix_{i+1} = F_i$ делаем те же преобразования и получаем

$$-A_i \cdot (\alpha_{i-1}x_i + \beta_{i-1}) + B_ix_i - C_ix_{i+1} = F_i.$$

Получается уравнение вида $x_i = \alpha_ix_{i+1} + \beta_i$, где α_i и β_i — прогоночные коэффициенты, вычисляются через предыдущие:

$$\alpha_i = \frac{C_i}{B_i - A_i\alpha_{i-1}},\ \beta_i = \frac{F_i + A_i\beta_{i-1}}{B_i - A_i\alpha_{i-1}},\ 2 \leqslant i \leqslant n-1.$$

n-1. Здесь

$$x_{n-1} = \alpha_{n-1}x_n + \beta_{n-1}, \tag{2.5.3}$$

ведь формула справедлива, она ещё называется рекуррентной.

n. Рассмотрим сумму уравнения (2.5.3) умноженного на A_n и $-A_nx_{n-1} + B_nx_n = F_n$, получится $x_n = \beta_n$ и α здесь нет, так как нет следующего слагаемого x_{n+1}, он равен нулю.

▷ Обратная прогонка, здесь идёт поиск неизвестной $\vec{x}$. Идя по $n-1$, $n-2$, …, 1 — обратная рекурсия и вычисляя x_i, то есть по формуле (2.5.3). Здесь операций (8 или 9) $\cdot\, n$, зависит от того, как считаем знаменатель, можно сразу не делить.

Всё было бы прекрасно, если бы мы были уверены, что коэффициенты α_i и особенно β_i ограничены, вдруг они начнут влиять на ошибку округления (это *оценкой прогоночных коэффициентов* ещё называется).

Мы его проверим для матрицы с диагональным преобладанием. Цивилизованному человеку нельзя жить без этого метода (они ущербны, скажите тем, кто не ходит на лекции, что у них в образовании дырка, они не могут отличить самогонку от прогонки). У нас $B_i > 0$, $B_1 - C_1 = \alpha_1 > 0$ — диагональное преобладание в первой строке, а дальше

$B_i - (|A_i| + |C_i|) = D_i > 0$ — условное диагональное преобладание элементов D_i над не диагональными элементами, ну и в конце последовательности $B_n - |A_n| = D_n > 0$. Берём $\alpha_1 = C_1/B_1$ или $|\alpha_1| \leqslant |C_1|/B_1$, тогда подставим

$$|\alpha_1| \leqslant \frac{|C_1|}{B_1} = \frac{|C_1|}{D_1 + |C_1|} < 1$$

— согласно условию диагонального преобладания. Рассмотрим следующие

$$\alpha_i = \frac{C_i}{B_i - A_i\alpha_{i-1}},$$

попробуем оценить α_i, при этом будем использовать индукцию, считаем, что $|\alpha_{i-1}| < 1$, тогда

$$|\alpha_i| \leqslant \frac{|C_i|}{B_i - |A_i| \cdot |\alpha_{i-1}|} \leqslant \frac{|C_i|}{B_i - |A_i| \cdot 1} = \frac{|C_i|}{D_i + |C_i|} < 1,$$

то есть наши коэффициенты α_i не растут. Когда делаем обратную прогонку, $x_i = \alpha_i \cdot x_{i+1}$, но если мы ошиблись, то $\tilde{x}_i - x_i = \alpha_i\delta_i$, вот почему важно $\alpha_i < 1$, что ошибки округления не возрастают.

Но может ещё расти коэффициент β, ведь ещё есть правая часть F. Пусть

$$F = \max_{1<i\leqslant n} |F_i|,$$

а $D_i \geqslant D_{min} > 0$ — строгое диагональное преобладание, тогда рассмотрим

$$\beta_i = \frac{F_i + A_i\beta_{i-1}}{B_i - A_i\alpha_{i-1}}$$

и задача оценить $|\beta_i|$, получить априорную оценку. Надо прощупать, что такое

$$|\beta_1| = \frac{F_1}{B_1} \leqslant \frac{|F_1|}{B_1},$$

теперь я делаю хитрейшую хитрость:

$$|\beta_1| \leqslant \frac{|F_1|}{B_1} \cdot \frac{|D_1|}{D_1} = \frac{D_1}{B_1} \cdot \frac{F_1}{D_1} \leqslant (1 - |\alpha_1|) \cdot \frac{F}{D} < \frac{F}{D}$$

и постараемся получить такую оценку для других β_i по индукции:

$$|\beta_i| \leqslant \frac{D_i \cdot \frac{|F_i|}{D_i} + |A_i| \cdot (1 - |\alpha_{i-1}|) \cdot \frac{F}{D}}{|B_i - A_i\alpha_{i-1}|} \leqslant \frac{F}{D} \cdot \frac{B_i - |A_i| - |C_i| + |A_i| - |A_i| \cdot |\alpha_{i-1}|}{|B_i - A_i\alpha_{i-1}|}$$

и если поднапрячься немножко, а студент не хочет работать на лекции, то это равно

$$= \frac{F}{D} \cdot \left[\frac{B_i - |A_i| \cdot |\alpha_{i-1}|}{|B_i - A_i\alpha_{i-1}|} - |\alpha_i|\right] \leqslant \frac{F}{D} \cdot (1 - |\alpha_i|),$$

так как

$$\alpha_i = \frac{C_i}{B_i - A_i\alpha_{i-1}}$$

— то, что мы постулировали для $(i-1)$, то есть $\forall i$ оценка β_i справедлива. А что она даёт: априорно, до программирования, до вычислений на ЭВМ мы можем знать оценку наших результатов, мы можем и без компьютера знать ответ, но не с точность до после запятой, а в каких пределах даст компьютер. 06.04.05

Теперь вернёмся к рекуррентной формуле $x_i = \alpha_i x_{i+1} + \beta_i$, $\alpha_i < 1$ и β_i мы оценили: $|x_n| = |\beta_n| \leqslant F/D$ и тогда рассматривая обратную прогонку начиная с $(n-1)$ я получаю

$$|x_{n-1}| \leqslant |\alpha_{n-1}| \cdot \frac{F}{D} + (1 - |\alpha_{n-1}|) \cdot \frac{F}{D} = \frac{F}{D},$$

дальше $(n-2)$ и так далее, то есть оценки дают оценить и решение, надо только найти максимальное F и минимальное D, это используется для проверки работы программы.

На этом мы заканчиваем точные методы, есть ещё модификации, но мы на них не будем останавливаться, они все уже заложены в линейной алгебры пакеты.

Глава 3

Итерационные методы решения систем линейных и нелинейных уравнений

3.1 Введение, условие сходимости итерационных методов

Рассмотрим линейную систему уравнений $\mathtt{A} \cdot \vec{x} = \vec{b}$, считаем, что матрица $\mathtt{A}$ невырожденная, то есть имеется решение $\vec{x} = \mathtt{A}^{-1} \cdot \vec{b}$ — может быть найдено теоретически. Но если матрица большого порядка и имеется большое количество нулей, теоретически можно, но практически неприемлимо для вычислений на ЭВМ, поэтому используют итерационные методы, которые дают решения с какой-то точностью заранее заданной величины ε: $\|\tilde{x} - \vec{x}\| < \vec{\varepsilon}$ (получено решение с любой наперёд заданной точностью, но это теоретически, так как компьютер не может без ошибки вычислить ни одной арифметической операции, так что мы знаем, что можно получить лишь с точностью до ошибки округления).

Все наши методы имеют ошибки задания в исходных данных — в $\mathtt{A}$ и в $\vec{b}$, какая разница, что будем точно вычислять, надо получить с точностью большей на порядок, чем исходные данные — вот какая современная концепция.

Зададим начальное приближение $x^{(0)}$, обычно задают правую часть, $\vec{b}$ и затем с помощью какой-то линейной процедуры приближения строится последовательность $\vec{x}^{(1)}$, $\vec{x}^{(2)}$, …, $\vec{x}^{(k)}$ такая, что $\vec{x}^{(k)} \to \mathtt{A}^{-1} \cdot \vec{b}$. Когда было одно уравнение, мы строили производную функцию и приближали $x^{(k+1)} = \varphi(x^{(k)})$ — итерационный метод. Но там было уравнение $f(x) = 0$, а здесь $\mathtt{A} \cdot \vec{x} = \vec{b}$, то есть целая система со специальным видом и процесс уже другой, то есть есть некоторая матица $\mathtt{S}$ такая, что $\vec{x}^{(k+1)} = \mathtt{S} \cdot \vec{x}^{(k)} + \vec{F}$, где $k = 0, 1, 2, \ldots$ — то k, которое обеспечивает определённую точность. Но $\mathtt{S}$ должна иметь связь с уравнением, пусть $\vec{x}^{(0)} = \mathtt{A}^{-1} \cdot \vec{b}$ и решаем $\vec{x}^{(k+1)} = \mathtt{S} \cdot \vec{x}^{(k)} + \vec{F}$, должно быть точно, мы подставляем

$$\mathtt{S} \cdot \mathtt{A}^{-1} \cdot \vec{b} + \vec{F} = \mathtt{A}^{-1} \cdot \vec{b}$$

и отсюда получается $\vec{F} = (\mathtt{E} - \mathtt{S}) \cdot \mathtt{A}^{-1} \cdot \vec{b}$, то есть свободное слагаемое должно быть пропорционально правой части $\vec{b}$, пусть по матрице $\mathtt{B}^{-1}$, то есть $\vec{F} = \mathtt{B}^{-1} \cdot \vec{b}$ — линейно-образная зависимость от $\vec{b}$ через матрицу, то есть линейные преобразования.

Тогда из формулы $\vec{x}^{(k+1)} = \mathtt{S} \cdot \vec{x}^{(k)} + \mathtt{B}^{-1} \cdot \vec{b}$, то есть $\mathtt{B}^{-1} = (\mathtt{E} - \mathtt{S}) \cdot \mathtt{A}^{-1}$, отсюда $\mathtt{B}^{-1} \cdot \mathtt{A} = (\mathtt{E} - \mathtt{S})$, откуда $\mathtt{S} = (\mathtt{E} - \mathtt{B}^{-1} \cdot \mathtt{A})$ и окончательно

$$x^{(k+1)} = (\mathtt{E} - \mathtt{B}^{-1} \cdot \mathtt{A}) \cdot \vec{x}^{(k)} + \mathtt{B}^{-1} \cdot \vec{b}, \tag{3.1.1}$$

а решение точное имеет вид $\vec{x} = \mathtt{A}^{-1} \cdot \vec{b}$, тогда мы что должны сообразить, мы приближаемся к способу построения матрицы $\mathtt{S}$, а мы свели к $\mathtt{B}$, если $\mathtt{B}$ близка к матрице $\mathtt{A}$, тогда $\mathtt{S}$ будет близкой к нулю, но $\mathtt{B} \approx \mathtt{A}$ должна легко обращаться, но обращать мы её не хотим, это почти невозможно, а берём примерно.

Обычно строят по методу релаксации

$$\mathtt{B} \cdot \frac{\vec{x}^{(k+1)} - \vec{x}^{(k)}}{\tau} + \mathtt{A} \cdot \vec{x}^{(k)} = \vec{b},$$

а он хорошо согласуется с формулой (3.1.1), тогда

$$\vec{x}^{(k+1)} = (\mathtt{E} - \tau \cdot \mathtt{B}^{-1} \cdot \mathtt{A}) \cdot \vec{x}^{(k)} + \tau \cdot \mathtt{B}^{-1} \cdot \vec{b}, \tag{3.1.2}$$

по сравнению с (3.1.1) появляется параметр τ релаксации. Но нужно построить матрицу, которая легко обращается, некоторые способы берут вообще единицу, но какие требования на $\mathtt{S}$:

$$\vec{x}^{(k+1)} = \mathtt{S} \cdot (\mathtt{S} \cdot \vec{x}^{(k-1)} + \vec{F}) + \vec{F} = \mathtt{S} \cdot (\mathtt{S} \cdot (\mathtt{S} \cdot \vec{x}^{(k-2)} + \vec{F}) + \mathtt{S} \cdot \vec{F}) + \vec{F} \tag{3.1.3}$$

и так далее и в конце концов я получу ряд

$$(\mathtt{E} + \mathtt{S} + \ldots + \mathtt{S}^k) \cdot \vec{F} + \mathtt{S}^{k+1} \cdot \vec{x}^{(0)}$$

и формула (3.1.3) показывает, как решать зависимость $\mathtt{S}$ и $\vec{F}$, которые мы строим, только как ими распорядиться, чтобы $\vec{x}^{(k+1)}$ сходилось к $\mathtt{A}^{-1} \cdot \vec{b}$? Есть специально теорема:

Теорема 3.1.1 (фон Неймана)**.**

▷ Великий математик, отец кибернетики, добавил при фашизме себе „фон“, чтобы не попасть в концлагерь.

Если собственные числа $|\lambda(\mathtt{S})| < 1$, то $(\mathtt{E} + \mathtt{S} + \ldots + \mathtt{S}^k) \to (\mathtt{E} - \mathtt{S})^{-1}$.

Замечание 3.1.1.1 (Условия теоремы)**.**

▷ Условие $|\lambda(\mathtt{S})| < 1$ является необходимым и достаточным.

▷ Если $\|\mathtt{S}\| < 1$, то это условие является достаточным для того, чтобы матричный ряд сходился, так как $|\lambda(\mathtt{S})| < \|\mathtt{S}\|$ — для любой нормы.

Но оценить собственные значения сложно, а норма матрицы вычисляется легко, так что это важно. Доказательство этой теоремы есть в курсе линейной алгебры, у нас немного другой курс.

Зачем нам эта теорема нужна:

$$\vec{x}^{(\infty)} = (\mathtt{E} - \mathtt{S})^{-1} \cdot \vec{F} + \vec{0}$$

— вот, что получается в пределе, но если вспомнить $\mathtt{S}$ и $\vec{F}$, то

$$\vec{x}^{(\infty)} = (\mathtt{E} - \mathtt{E} + \tau \cdot \mathtt{B}^{-1} \cdot \mathtt{A})^{-1} \cdot \tau \cdot \mathtt{B}^{-1} \cdot \vec{b} = \frac{1}{\tau} \cdot \mathtt{A}^{-1} \cdot \mathtt{B} \cdot \tau \cdot \mathtt{B}^{-1} \cdot \vec{b} = \mathtt{A}^{-1} \cdot \vec{b},$$

то есть $\vec{x}^{(\infty)}$ и есть наше решение исходной задачи. То есть мы должны выбрать матрицу $\mathtt{S}$ такой, чтобы $\|S\| < 1$ — конструктивное достаточное условие сходимости нашего процесса, то есть мы получаем условие $\|\mathtt{E} - \tau \cdot \mathtt{B}^{-1} \cdot \mathtt{A}\| < 1$. Но здесь стоит $\mathtt{B}^{-1}$ и $\mathtt{A}$ — трудно проверить, поэтому русский учёный Самарский из Москвы, сейчас в преклонном возрасте, сформулировал теорему:

Теорема 3.1.2 (Самарского)**.**

▷ Если матрица $\mathtt{A}$ положительно определённая, то итерационный процесс по формулам (3.1.1) и (3.1.2) сходится, если матрица $\mathtt{B} > \frac{\tau}{2} \cdot \mathtt{A}$, что более конструктивно, но для определённого класса матриц.

Замечание 3.1.2.1 (К условиям)**.**

▷ Это означает, что $\mathtt{C} = \mathtt{B} - \frac{\tau}{2} \cdot \mathtt{A} > 0$ — положительно определённая. Если матрица $\mathtt{A}$ не положительно определённая, делаем трансформацию Гаусса. то есть умножаем систему на $\mathtt{A}^T$: $\mathtt{A}^T \cdot \mathtt{A} \cdot \vec{x} = \mathtt{A}^T \cdot \vec{b}'$, то есть $\bar{\mathtt{A}} \cdot \vec{x} = \vec{b}$, но $\bar{\mathtt{A}} = \mathtt{A} \cdot \mathtt{A}^T$ — симметричная и всегда положительно определена — согласно линейной алгебре, то есть мы сравнительно легко, но для большой матрицы перемножение тяжёлое.

▷ При конструировании, $\mathtt{B}$ может быть не симметрична, тогда в теореме должно быть другое условие:

$$\frac{\mathtt{B} + \mathtt{B}^T}{2} > \frac{\tau}{2} \cdot \mathtt{A},$$

где $\mathtt{A} = \mathtt{A}^T > 0$ (иначе плохо проверять условие для $\mathtt{B}$ несимметричной и $\mathtt{A}$ симметричной).

Доказательство у неё не длинное, но мы его рассматривать не будем. Конструируем критерий, который поможет нам построить матрицу $\mathtt{S}$, а на самом деле $\mathtt{B}$, повторим: во-первых должна быть легко обратима, например диагональная или треугольная.

Допустим мы построили такую матрицу $\mathtt{S}$, что всё сходится, теперь осталось сформировать критерий остановки для какой-то погрешности ε. Но $\vec{x}$ — это вектор, имеющий n компонент, это значит, что надо какую-то норму $\|\vec{x}\|$ оценить. В линейной алгебре есть теорема, что если последовательность векторов $\vec{x}_1$, $\vec{x}_2$, …, $\vec{x}_n$ сходится к некоторому пределу $\vec{x}$, то и каждая компонента вектора $\vec{x}_k$ сходится к компоненте $\vec{x}$. Поэтому если докажем для сходимости по норме, то и по компонентам докажем. Пусть погрешность $\vec{\varepsilon}_k = \vec{x}^{(k)} - \vec{x}$ и нас волнует $\|\vec{\varepsilon}_k\|$. Тогда надо написать уравнения для ошибки, возьмём

$$\mathtt{B} \cdot \frac{\vec{x}^{(k+1)} - \vec{x}^{(k)}}{\tau} + \mathtt{A} \cdot \vec{x}^{(k)} = \vec{b},$$

теперь я возьму и вычту из левой и правой части $\vec{b}$, а к левой части добавлю и отниму $\vec{x}/\tau$, тогда

$$\mathtt{B}\cdot\frac{\vec{\varepsilon}^{(k+1)}-\vec{\varepsilon}^{(k)}}{\tau}+\mathtt{A}\cdot\vec{\varepsilon}^{(k)}=0$$

и я получаю отсюда легко $\vec{\varepsilon}^{(k+1)}=\mathtt{S}\cdot\vec{\varepsilon}^{(k)}$ или это равно $\mathtt{S}^{k+1}\cdot\vec{x}^{(0)}$. Отсюда вытекает неравенство $\|\vec{\varepsilon}^{(k+1)}\|\leqslant\|\mathtt{S}^{k+1}\|\cdot\cdot\|\vec{\varepsilon}^{(0)}\|$. Это и есть основной критерий остановки процесса, мы говорим, что

$$\frac{\|\varepsilon^{(k+1)}\|}{\|\varepsilon^{(0)}\|}<\|\mathtt{S}\|^{k+1}<\varepsilon,$$

эта относительная ошибка не очень удобна иногда для проверки, так как $\vec{\varepsilon}^{(0)}$ зависит от $\vec{x}^0$, но $\vec{x}$ мы не знаем, оценка чисто теоретическая. Поэтому используют другой способ от $\vec{x}^{(k)}$ и $\vec{x}^{(k+1)}$, пусть $\rho=\|\mathtt{S}\|<1$ — построили, тогда как остановиться:

$$\|\vec{\varepsilon}^{(k+1)}\|=\|\vec{x}^{(k+1)}-\vec{x}^{(k)}\|\leqslant\rho\cdot\|\vec{\varepsilon}^{k}\|=\rho\cdot\|\vec{x}^{(k)}-\vec{x}\|,$$

где $\vec{x}$ — точное решение, теперь сделаем хитрейшую хитрость, добавим и вычтем $\vec{x}^{(k+1)}$ в правой норме:

$$\|\vec{\varepsilon}^{(k+1)}\|\leqslant\rho\cdot\|\vec{x}^{(k)}-\vec{x}^{(k+1)}\|+\rho\cdot\|\varepsilon^{(k+1)}\|,$$

в итоге я получаю критерий

$$\|\vec{\varepsilon}^{(k+1)}\|\leqslant\frac{\rho}{1-\rho}\cdot\|\vec{x}^{(k)}-\vec{x}^{(k+1)}\|<\varepsilon$$

— это я могу легко в компьютере проверить, нашёл разность ρ и $(1-\rho)$, мы должны заранее вычислить на всю задачу один раз и проверять. Это конструктивный критерий, который мы можем проверить в компьютере, важно, чтобы $\rho<1$, иначе не сойдётся.

3.2 Способы конструирования итерационных процессов

1. *Метод простой итерации*, называется простой, потому что может получиться из метода релаксации для $\tau=$ $=1$, $\mathtt{B}=\mathtt{E}$ и из (3.1.1) и (3.1.2) мы получаем, что

   $$\vec{x}^{(k+1)}=(\mathtt{E}-1\cdot\mathtt{A})\cdot\vec{x}^{(k)}+1\cdot\mathtt{E}\cdot\vec{b}=(\mathtt{E}-\mathtt{A})\cdot\vec{x}^{(k)}+\vec{b}$$

   — самый простой матричный вид. Мы находим $\vec{x}^{(0)}$ и

   $$x_i^{(k+1)}=(1-a_{ii})\cdot x_i^{(k)}+b_i-\sum_{\substack{j=1\\j\neq i}}^{n}a_{ij}\cdot x_j^{(k)}$$

   — не зависит от лишних переменных, поэтому хорошо распараллеливается, даёт очень хороший результат на параллельных компьютерах, но не всегда сходится. Главное условие что $|\lambda(\mathtt{S})|<1$, а $\mathtt{S}=\mathtt{E}-\mathtt{A}$, тогда $|\lambda(\mathtt{S})|=$ $=|1-\lambda(\mathtt{A})|>1\Rightarrow-1<1-\lambda(\mathtt{A})<1\Rightarrow0<\lambda(\mathtt{A})<2$, видите, какой класс узкий, больше нуля всегда сделаем трансформацией Гаусса, но вот меньше двух, извините.

   Поэтому этот метод надо совершенствовать, создан в XIX веке и его всё время модифицировали.

2. Метод Якоби — великий учёный. Особенно хороший для матриц с диагональным преобладанием, тогда он всегда сходится. В матрице $\mathtt{A}$ выделим диагональные элементы: $\mathtt{A}=\mathtt{D}+\mathtt{C}$, где $\mathtt{D}=\operatorname{Diag}\{a_{ii}\}_{i=1}^{n}$ и в начале берут и умножают систему на $\mathtt{D}^{-1}$, то есть выбирают $\tau=1$, $\mathtt{B}=\mathtt{D}$, получаем релаксации метод

   $$\mathtt{D}\cdot(\vec{x}^{(k+1)}-\vec{x}^{(k)})+\mathtt{A}\cdot\vec{x}^{(k)}=\vec{b}\Rightarrow\vec{x}^{(k+1)}=(\mathtt{E}-\mathtt{D}^{-1}\cdot\mathtt{A})\cdot\vec{x}^{(k)}+\mathtt{D}^{-1}\cdot\vec{b}.$$

   Как изменилась расчётная формула, $\mathtt{D}^{-1}\cdot\mathtt{A}$ — все диагональные элементы единицы, а когда отнимем $\mathtt{E}$, получим

   $$\mathtt{S}=\begin{pmatrix}0&&\frac{a_{ij}}{a_{ii}}\\&\ddots&\\\frac{a_{ij}}{a_{ii}}&&0\end{pmatrix}$$

   и мы получаем

   $$x_i^{(k+1)}=\frac{b_i}{a_{ii}}-\frac{1}{a_{ii}}\cdot\sum_{\substack{j=1\\i\neq j}}^{n}a_{ij}\cdot x_j^{(k)},$$

причём исчезли диагональные элементы. Тогда

$$\|\mathtt{S}\| = \max_{1 \leqslant i \leqslant n} \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}/a_{ii}|,$$

но так как с диагональным преобладанием, то $\|\mathtt{S}\| < 1$ всегда.

Но всё-таки класс довольно-таки узкий, поэтому был изобретён ещё один метод.

3. Метод Гаусса-Зейделя (чаще просто Зейделя). Хотя Гаусс этот метод не знал, Зейдель его не предлагал, но так иногда бывает в математике. Идея в следующем: в методе Якоби мы находили первую компоненту

$$x_1^{(k+1)} = \frac{b_1}{a_{11}} - \frac{1}{a_{11}} \cdot \sum_{j=2}^{n} a_{1j} \cdot x_j^{(k)},\ x_2^{(k+1)} = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}} \cdot x_1^{(k+1)}$$

и так далее, то есть использовали что было раньше. 27.04.05

Рассмотрим систему развёрнуто:

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

Нам так же как и в методе Якоби, первое уравнение делим на a_{11} и выражаем

$$x_1^{(k+1)} = \frac{1}{a_{11}} \cdot \left(b_1 - \sum_{j=2}^{n} a_{1j}x_j^{(k)} \right),$$

а из уравнения два вычисляется x_2, но вместо x_1 подставляется только что вычисленный, то есть

$$x_2^{(k+1)} = \frac{1}{a_{22}} \cdot \left(b_2 - a_{21}x_1^{(k+1)} - \sum_{j=3}^{n} a_{2j}x_j^{(k)} \right)$$

и тогда общая расчётная формула для i-ой компоненты

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \cdot \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_i^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} \right).$$

Обоснование какое у этого метода? Достаточно показать, что матрица $\mathtt{A}$ системы $\mathtt{A} \cdot \vec{x} = \vec{b}$ — симметричная и положительно определённая: $\mathtt{A} = \mathtt{A}^T > 0$. Есть две теоремы: фон Неймана 3.1.1 на стр. 29, необходимая и достаточная, собственные числа матрицы перехода должны быть меньше единицы и теорема Самарского 3.1.2 на стр. 29, где формулы

$$\mathtt{B} \cdot \frac{\vec{x}^{k+1} - \vec{x}^k}{\tau} - \mathtt{A} \cdot \vec{x}^k = \vec{b},$$

если построить такой процесс с помощью релаксационного метода, то должно быть

$$\frac{1}{2} \cdot (\mathtt{B} + \mathtt{B}^T) > \frac{\tau}{2} \cdot \mathtt{A}.$$

Какая у нас тут получается матрица $\mathtt{B}$? На самом деле чтобы записать процесс Гаусса-Зейделя матрицу $\mathtt{A}$ надо представить в виде $\mathtt{A} = \mathtt{D} + \mathtt{L} + \mathtt{U}$. Причём если матрица $\mathtt{A}$ симметричная, то $\mathtt{L} = \mathtt{U}^T$, если она положительно определена, то известно из матричного исчисления, что $\mathtt{D} > 0$, τ у нашего метода равна единице оказывается, итого

$$(\mathtt{D} + \mathtt{L}) \cdot (\vec{x}^{k+1} - \vec{x}^k) + \mathtt{A} \cdot \vec{x}^k = \vec{b}$$

и $\mathtt{B} = \mathtt{D} + \mathtt{L}$ — не симметричная, а нижне-треугольная, правильно? С такой матрицей легко считать и если я выражу

$$\vec{x}^{k+1} = (\mathtt{D} + \mathtt{L})^{-1} \cdot (\mathtt{U} \cdot \vec{x}^k + \vec{b}),$$

здесь главное построить матрицу $\mathtt{B}$ и проверить теорему Самарского 3.1.2, здесь это будет

$$2\mathtt{D}+\mathtt{L}+\mathtt{U}=\mathtt{B}+\mathtt{B}^T>\mathtt{A}=\mathtt{D}+\mathtt{L}+\mathtt{U}\Rightarrow\mathtt{D}>0$$

согласно требованиям к структуре из задания, мы договорились решать с положительно определённой матрицей. Здесь уже очень широкий спектр случаев, так как всегда трансформацией Гаусса (такой приём) можем получить матрицу, обладающую таким свойством.

На самом деле есть ещё способ улучшить, изменять итерационный параметр τ, так родился ещё один метод.

4. Верхней релаксации, история умалчивает, кто придумал, идея на самом деле от способа Гаусса:

$$(\mathtt{D}+\mathtt{C})\cdot\vec{x}^{k+1}+\mathtt{U}\cdot\vec{x}^k=\vec{b}$$

— вот у нас как выглядит метод, идея метода релаксации: вот здесь диагональные элементы расщепляются на две и часть переносится в $\vec{b}$, тогда выбирается параметр α для

$$(\alpha\mathtt{D}+\mathtt{L})\cdot\vec{x}^{k+1}+((1-\alpha)\cdot\mathtt{D}+\mathtt{U})\cdot\vec{x}^k=\vec{b}.$$

В таком виде этот метод обычно не пишется, вводится параметр релаксации $1/\omega=\alpha$ и в матричной форме получается система

$$(\mathtt{D}+\omega\mathtt{L})\cdot\frac{\vec{x}^{k+1}-\vec{x}^k}{\omega}+\mathtt{A}\cdot\vec{x}^k=\vec{b},$$

и когда $\omega>1$, метод называют верхней релаксацией, если меньше единицы, нижней, это один метод (раньше вместо τ было принято писать ω).

Если проверить теоремой Самарского, то чтобы было $\mathtt{A}>0$, требуется выбрать $0<\omega<2$, но надо ω выбирать оптимальным образом — задача для эллиптических сеточных разностных методов, там сложная наука как выбрать, но можно.

В настоящее время широкие классические методы созданы учениками школы академика Самарского, он сейчас уже старенький.

5. Попеременно треугольным метод, этот метод сейчас наиболее развивается. Считаем, что $\mathtt{A}>0$ и разбиваем

$$\mathtt{A}=\left(\frac{1}{2}\mathtt{D}+\mathtt{L}\right)+\left(\frac{1}{2}\mathtt{D}+\mathtt{U}\right)$$

и обозначим нижне-треугольные матрицы:

$$\mathtt{R}_1=\left(\frac{1}{2}\mathtt{D}+\mathtt{L}\right),\ \mathtt{R}_2=\left(\frac{1}{2}\mathtt{D}+\mathtt{U}\right)$$

и, очевидно, $\mathtt{R}_1^T=\mathtt{R}_2$. И тогда такой итерационный метод, он с начала кажется громоздким:

$$\mathtt{B}=(\mathtt{E}+\omega\mathtt{R}_1)\cdot(\mathtt{E}+\omega\mathtt{R}_2)$$

(иногда параметры ω берутся разными, но мы возьмём попроще) и если

$$\mathtt{B}_1=(\mathtt{E}+\omega\mathtt{R}_1),\ \mathtt{B}_2=(\mathtt{E}+\omega\mathtt{R}_2),$$

то в каноническом виде метод получится

$$\mathtt{B}_1\cdot\mathtt{B}_2\cdot\frac{\vec{x}^{k+1}-\vec{x}^k}{\tau}+\mathtt{A}\cdot\vec{x}^k=\vec{b}.$$

Но как реализовывается на ЭВМ, тут видите матрицы, произведения? Но много преобразований, разные есть способы, переносим, например, и получаем

$$\mathtt{B}_1\cdot\mathtt{B}_2\cdot\vec{x}^{k+1}=\vec{F}^k,$$

где в $\vec{F}^k$ входят все остальные параметры:

$$\vec{F}^k=\tau\vec{b}-\tau\mathtt{A}\cdot\vec{x}^k+\mathtt{B}_1\mathtt{B}_2\cdot\vec{x}^k$$

и надо решить такую систему для неизвестного $\vec{x}^{k+1}$. Какое тут слабое звено, за что зацепиться? Матрицы $\mathtt{B}_1$ и $\mathtt{B}_2$ треугольные, вводим новую переменную $\vec{y}^{k+1}=\mathtt{B}_2\cdot\vec{x}^{k+1}$ и получаем систему $\mathtt{B}_1\cdot\vec{y}^{k+1}=\vec{F}^k$, но $\mathtt{B}_1$ —

треугольная и легко получить $\vec{y}^{k+1} = \mathtt{B}_1^{-1} \cdot \vec{F}^k$, то есть легко реализовать с помощь простого алгоритма прямой или обратной подстановки. Так я получу $\vec{x}^{k+1} = \mathtt{B}_2^{-1} \cdot \vec{y}^{k+1}$ — вот окончательное решение, то есть как я всегда говорил, нормальные герои всегда идут в обход.

Но когда этот метод сходится? У нас видите тут два параметра получается — ω и τ, можно показать, что будет сходиться, если выбрать $\omega > \tau/2$ и в случае симметричной положительно определённом матрицы $\mathtt{A}$. Здесь для конкретной матрицы можно получить оптимальные параметры ω и τ, академик Анатолий Анатольевич Коновалов у нас много занимался этим в институте, вообще-то это целая наука.

Как можно компьютер заставить выбирать эти параметры, пусть лучше он думает, а не мы. Существует большая группа методов (*градиент* ещё их зовут), которые заставляют считать компьютер.

6. Скорейший градиентный спуск, отличается от классической стандартной релаксации, параметр τ переменный и на каждом итерационном шаге имеет своё значение, иногда из соображений, чтобы была минимальная невязка: $\mathtt{A} \cdot \vec{x}^k - \vec{b} = \vec{r}^k$. А простейший способ, чтобы погрешность $\vec{\varepsilon}^k = \vec{x}^k - \vec{x}$ была на k-от шаге минимальна, можно сделать с помощью выбора параметра τ, даже можно $\mathtt{B} = \mathtt{E}$, то есть система

$$\frac{\vec{x}^{k+1} - \vec{x}^k}{\tau_k} + \mathtt{A} \cdot \vec{x}^k = \vec{b},$$

как выбрать τ_k? Для $\vec{r}_k$ надо построить уравнение, сделаем так:

$$\frac{\vec{x}^{k+1} + \vec{x} - \vec{x} - \vec{x}^k}{\tau_k} + \mathtt{A} \cdot \vec{x}^k = \vec{b},$$

а $\vec{b} = \mathtt{A} \cdot \vec{x}$ и я получаю уравнение

$$\frac{\vec{r}^{k+1} - \vec{r}^k}{\tau_k} + \mathtt{A} \cdot \vec{r}^k = \vec{0}$$

— уравнение только для погрешности и надо выбрать τ_k, здесь $\vec{r}^{k+1} = (\mathtt{E} - \tau_k \cdot \mathtt{A}) \cdot \vec{r}^k$ и рассмотрим норму r^k как функцию от τ_k, норму возьмём $(\vec{r}^{k+1}, \vec{r}^{k+1})$ — скалярное произведение и она должна быть минимальной, а как можем добиться, у нас вот только один параметр есть, давайте распишем

$$(\vec{r}^{k+1}, \vec{r}^{k+1}) = \Big((\mathtt{E} - \tau_k \cdot \mathtt{A}) \cdot \vec{r}^k, (\mathtt{E} - \tau_k \cdot \mathtt{A}) \cdot \vec{r}^k\Big) = (\vec{r}_k, \vec{r}_k) - 2\tau_k \cdot (\mathtt{A} \cdot \vec{r}^k, \vec{r}^k) + \tau_k^2 \cdot (\mathtt{A} \cdot \vec{r}^k, \mathtt{A} \cdot \vec{r}^k)$$

и требуется чтобы была минимальна, мы потребуем, чтобы производная равнялась нулю, правильно? Дифференцируем по τ_k и получаем

$$-2 \cdot (\mathtt{A} \cdot \vec{r}^k, \mathtt{A} \cdot \vec{r}^k) + 2\tau_k \cdot (\mathtt{A} \cdot \vec{r}^k, \mathtt{A} \cdot \vec{r}^k) = 0$$

и отсюда находится

$$\tau_k = \frac{(\mathtt{A} \cdot \vec{r}^k, \vec{r}^k)}{(\mathtt{A} \cdot \vec{r}^k, \mathtt{A} \cdot \vec{r}^k)},$$

прекрасно, смотрите, эти все величины мы можем вычислить, матрица $\mathtt{A}$ дана. Да, а как мы вычислим $\vec{r}^k$, вот Канавалов, как он? Теперь надо эти $\vec{r}^k$ вычислить, они вычисляются, но сложно. Это один из способов оптимизации.

Другой способ есть, более простой.

7. Метод простой итерации с выбором стационарного оптимального параметра. Он совсем простой этот метод, потребует больших знаний о матрице, чем больше имеем знаний, тем более оптимальный метод может построить, нужно найти максимальное и минимальное собственное значение, то, как это сделать, я вам расскажу в следующем параграфе.

$$\frac{\vec{x}^{k+1} - \vec{x}^k}{\tau} + \mathtt{A} \cdot \vec{x}^k = \vec{b},$$

матрица $\mathtt{S} = \mathtt{E} - \tau\mathtt{A}$, если $\mathtt{A}$ симметричная и положительно определена, то матрица $\mathtt{S}$ тоже будет и собственные числа тогда у неё вещественные, помните из алгебры? В данном случае собственные числа матрицы $\mathtt{S}$ выглядят очень просто: $1 + \tau\lambda(\mathtt{A})$ и поскольку она симметричная, мы можем взять в качестве нормы $\|\mathtt{S}\|$ собственное число и вычислить параметр τ таким образом, чтобы $\|\mathtt{S}\| = \rho(\tau)$ была минимальной, максимальное собственное число может быть взято в качестве нормы. Если у матрицы $\mathtt{A}$ λ_{max}, $\lambda_{min} > 0$, я могу рассмотреть норму ρ как функцию от τ, которая определена на отрезке $[\lambda_{min}, \lambda_{max}]$, то есть что такая ρ у нас: $\rho = 1 - \tau\lambda$ и надо найти τ. Принимает максимальные значения на концах отрезка и τ определяет наклон, надо выбрать, чтобы максимальное значения были минимальными, а это будет когда они равны, то есть наша прямая проходит через середину отрезка $[\lambda_{min}, \lambda_{max}]$, то есть при $\lambda = (\lambda_{max} + \lambda_{min})/2$, подставляем

$$1 - \tau \cdot \frac{\lambda_{max} - \lambda_{min}}{2} = 0 \Rightarrow \tau = \frac{2}{\lambda_{max} + \lambda_{min}}$$

— это вот называется оптимальной величиной нашего параметра. Но на практике эти величины точно знать и не нужно, с какой-то погрешностью разумной.

Вся проблема в нахождении λ_{min} и λ_{max}, как заставить искать компьютер? Мы помним, что для этого надо решить уравнение n-ой степени, мы это проходили, это же целая проблема.

3.3 Решение частичной проблемы нахождения собственных значений

Есть ещё полная проблема нахождения всех параметров, но это уже сложно, мы ограничимся частичной проблемой. Пусть матрица $\mathtt{A}$ симметричная и положительно определена, тогда она имеет полную систему собственных векторов, правильно? Собственным вектором матрицы $\mathtt{A}$ называется вектора $\vec{\varphi}(m)$ такой, что линейное уравнение $\mathtt{A} \cdot \vec{\varphi}(m) = \lambda(m)\vec{\varphi}(m)$ имеет ненулевое решение. Оказывается их много со своими собственными значениями $\lambda(m)$ и их как раз n штук, если матрица $n \times n$, вот такие нам надо знать факты из алгебры.

Кроме того образуют линейную независимую ортонормированную систему, то есть

$$(\vec{\varphi}(m_1),\, \vec{\varphi}(m_2)) = \begin{cases} 1, & \text{если } m_1 = m_2, \\ 0, & \text{если } m_1 \neq m_2. \end{cases}.$$

Доказывать ничего не нужно нам, это всё есть в алгебре, тогда если вектор $\vec{x}$ можно представить в новом базисе

$$\vec{e}(m) = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

— только на m-ом месте единица, тогда

$$\vec{x} = \sum_m a(m) \cdot \vec{e}(m),$$

но всегда можем взять другой базис — систему собственных векторов матрицы $\mathtt{A}$ теоретически. То есть представляем

$$\vec{x} = \sum c(m) \cdot \vec{\varphi}(m)$$

и если я рассмотрю вектор

$$\vec{x}^{(1)} = \mathtt{A} \cdot \vec{x},$$

то это $\sum_m c(m) \cdot \mathtt{A} \cdot \vec{\varphi}(m)$, но $\mathtt{A} \cdot \vec{\varphi}(m) = \lambda(m)\vec{\varphi}(m)$ и $x^{(1)} = \sum_m c_m^{(n)} \lambda^{(m)} \vec{\varphi}^{(m)}$ — снова сумма, только коэффициенты подправили. И идея вычисления λ_{max}: пусть $\lambda^{(1)} > \lambda^{(2)} > \ldots > \lambda^{(n)}$ и $\lambda^{(1)}$ — максимальное, тогда алгоритм вычисления он итерационный, назначу какой-то $\vec{x}^{(0)}$, потом $\vec{x}^{(1)} = \mathtt{A} \cdot \vec{x}^{(0)}$ и найду

$$(\vec{x}^{(1)},\, \vec{x}^{(1)}) = \sum (c^{(m)}\lambda^{(m)})^2 = \sum (c^{(m)})^2 \cdot (\lambda^{(m)})^2,$$

то есть наши вектора обладают этим свойством, ещё считается

$$(\vec{x}^{(0)},\, \vec{x}^{(1)}) = \sum (c^{(m)})^2 \lambda^{(m)}.$$

А теперь давайте рассмотрим их отношение:

$$\tilde{\lambda} = \frac{(\vec{x}^{(1)},\, \vec{x}^{(1)})}{(\vec{x}^{(0)},\, \vec{x}^{(1)})} = \frac{\sum (c^{(m)}\lambda^{(m)})^2}{\sum (c^{(m)})^2 \lambda^{(m)}} = \frac{(c^{(1)})^2 \cdot (\lambda^{(1)})^2 \cdot \left(1 + \sum \left(\frac{c^{(m)}}{c^{(1)}} \cdot \frac{\lambda^{(m)}}{\lambda^{(1)}}\right)^2\right)}{(c^{(1)})^2 \cdot (\lambda^{(1)}) \cdot \left(1 + \sum \left(\frac{c^{(m)}}{c^{(1)}}\right)^2 \cdot \frac{\lambda^{(m)}}{\lambda^{(1)}}\right)}$$

всё сокращаем и получится

$$\tilde{\lambda} = \lambda^{(1)} \cdot (\ldots)$$

вот смотрите, это $\tilde{\lambda}$ почти $\lambda^{(1)}$, так как $\lambda^{(m)}/\lambda^{(1)} < 1$ всегда и если я буду рассматривать такие скалярные произведения, то степень будет повышаться, потом вычислю

$$\vec{x}^{(k+1)} = \mathtt{A} \cdot \vec{x}^{(k)}$$

и получу

$$\frac{(\vec{x}^{(k+1)},\, \vec{x}^{(k+1)})}{(\vec{x}^{(k)},\, \vec{x}^{(k+1)})} \xrightarrow{k\to\infty} \lambda^{(1)}.$$

Теперь найдём λ_{min}, тогда мы берём и строим матрицу $\mathtt{C} = \lambda^{(1)}\mathtt{E} - \mathtt{A}$, тогда у матрицы $\mathtt{C}$ максимальная $\lambda_{max}(\mathtt{C}) = \lambda^{(1)} - \lambda_{min}(\mathtt{A})$, поэтому найду у $\mathtt{C}$ максимальное собственное число и $\lambda_{min}(\mathtt{A}) = \lambda^{(1)} - \lambda_{max}(\mathtt{C})$ — очень простой метод, легко реализуется.

Вот зачем вы учите всю эту алгебру и математический анализ, для решения уже практических задач, но есть много модификаций и модернизаций этого метода во всяких пакетах.

Мы, конечно, не смогли изучить всех методов, на МЕХ на это выделяется целый семестр под раздел, а у вас надо ещё пять разделов изучить.

Глава 4

Интерполирование функций и приближения функций

Это очень важный раздел, который имеет широкое распространение сейчас. Все графические пакеты, которые у вас имеются в разных системах операционных, так же прикладные программы, они пользуются приёмами всевозможных интерполяций. Чтобы построить график из таблицы значений, вы ищите значения функции между точек. На самом деле там всё дискретно и нет карандаша с линейкой, Билл Гейтс не будет проводить вам кривую. Ещё возникает проблема исследования функций, тех же тригонометрических, они вычисляются достаточно сложно, потому их сводят к наиболее простым, самое простое — это полином, его можно интегрировать, дифференцировать и тому подобное. Мы, конечно, все современные способы не сможем захватить, но такие принципы основные мы рассмотрим.

4.1 Постановка задачи, основные теоремы

4.1.1 Интерполирование функций заданных таблично

То есть заданы значения аргументов каких-то

x_i	f_i
x_0	f_0
x_1	f_1
$\vdots$	$\vdots$

то есть задаётся пары чисел $(x_i,\ f_i)$, $i = 0,\ 1,\ \ldots,\ n$ и задача вычислить значения функции f в тех точках x_*, которые не совпадают с аргументами таблицы: $x_* \neq x_i$. Вы в школе проводили линейную интерполяцию, соединяли точки прямой и получали такие точки. Но какие недостатки: вы никогда не сможете вычислить производную в точках x_i (их ещё узлами называют), к тому же точность очень грубая, так как мы исследуем функцию только по двум значениям, локальная интерполяция и не смотрит на остальные точки.

Поэтому задача построить какую-то функцию, которую было бы легко вычислять, имеет достаточное количество производных (достаточно гладкую) и совпадающую в узлах с таблицей значений. Обычно используют полиномы, так как легко вычисляются на компьютере и мы рассмотрим разные способы. Полиномы легко интегрировать и дифференцировать и на практике чаще всего используют именно полиномы $P_m(x)$, где m — его степень,

$$P_m(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m$$

и первое требование и правило заключается в том, что $P_m(x_i) = f_i - \forall i = 0,\ 1,\ \ldots,\ n$. То есть у нас получается $(n+1)$ уравнение на коэффициенты полинома, то есть мы имеем уравнения

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_m x_0^m = f_0, \\ a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_m x_1^m = f_1, \\ \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_m x_n^m = f_n. \end{cases}$$

здесь неизвестен вектор $\vec{a}$ — самое основное требование интерполяции, задан вектор $\vec{f}$, а нам нужно определить коэффициенты, тогда составим систему уравнений из матрицы

$$\mathtt{A} = \begin{pmatrix} 1 & x_0 & \dots & x_0^m \\ 1 & x_1 & \dots & x_1^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^m \end{pmatrix}$$

и система уравнений тогда $\mathtt{A} \cdot \vec{a} = \vec{f}$. Но мы с вами уже знаем, что имеет единственное решение, когда $m = n$, тогда матрица размера $(n+1) \times (n+1)$, то есть первое условие на полином, что его степень равна n и определяет матрицу $\mathtt{A}$ отличную от нулевой, тогда система имеет решение и мы можем его как-то найти. Этот определитель называется определителем Ван дер Монда, он доказал, что этот определитель всегда отличен от нуля, если среди значений x_i нет совпадающих (это естественно, так как для одной точки нельзя задавать два значения).

Если матрица 1000×1000, то считать, вы знаете, утомительно, а матрица здесь неразрешима и в лоб никто не решает, как я всегда говорил, нормальные герои всегда идут в обход. Всегда надо посидеть и подумать, а вы сразу садитесь за компьютер и начинаете считать, а что получается вы не знаете и никто не знает и компьютер тоже.

Возьмём

$$P_n^{(1)}(x_i) = f_i - \forall i = 0,\, 1,\, \dots,\, n$$

и второй построенный полином

$$P_n^{(2)}(x_i) = f_i,$$

как показать, что они совпадают? Построим третий полином

$$P_n^{(3)}(x) = P_n^{(1)}(x) - P_n^{(2)}(x),$$

имеет ту же степень, не выше, как выглядит:

$$P_n^{(3)}(x) = (a_0^{(1)} - a_0^{(2)}) + (a_1^{(1)} - a_1^{(2)})x + \ldots + (a_n^{(1)} - a_n^{(2)})x^n,$$

тогда смотрим на главное требование, они должны совпадать со значениями функций, их $(n+1)$, это значит, что полином $P_n^{(3)}(x_i) = 0$, правильно? Мы же разность берём, а таких x_i — $(n+1)$ штука, а что значит $P_n^{(3)}(x_i) = 0$, то, что x_i — корни и в алгебре известно, что полином n-ой степени имеет n корней, считая кратные, а у нас $n+1$ корень получился, противоречие, но какой полином может иметь больше корней? Когда все его коэффициенты нуль, тогда хоть что подставлять и будет нуль, то есть эти два полинома совпадают.

Таким образом задачу интерполирования мы свели к отысканию полинома n-ой степени, осталась проблема вычисления его коэффициентов, но как, мы рассмотрим в следующем параграфе.

4.1.2 Функция $f(x)$ задана

Как может быть, допустим мы имеем формулу или какой-то прибор, который может выдать значения на любое входящее число, погрешностью мы пока пренебрегаем, $x \in [a,\, b]$ и имеется способ вычисления. Или функция имеет сложный вид или просто вычисляем каким-то прибором, тогда воспользуемся умением интерполировать, мы для какого-то числа экспериментов строим таблицу аргументов $x_0,\, x_1,\, \dots,\, x_n$ и вычисляем в них $f_0,\, f_1,\, \dots,\, f_n$. А дальше, пользуясь этими значениями, строим полином, то есть свели задачу к предыдущей, как всегда делают математики.

Помните, у вас была теорема Вейерштрасса: для любой непрерывной функции на отрезке $[a,\, b]$ можно построить полином, который будет отличаться от функции на какое-то ε, вот зацепились и знаем, что можем образовать функции построенные достаточно хорошо. Но теорема говорит, что существует и не говорит, как, первый вопрос, а сколько надо взять точек, как оценить ошибку

$$R(x) = P_n(x) - f(x),\, x \in [a,\, b]?$$

Чтобы исследовать этот вопрос, мы выбираем число n и посмотрим как располагать оптимальным оптимальным образом узлы интерполирования, то есть нужно оценить погрешность $R(x = x_*)$ и x_* не совпадает с узлом интерполирования. Чтобы этот вопрос изучить, построим функцию

$$F(x) = P_n(x) - f(x) - K \cdot \omega(x),$$

где K — константа, которую мы можем выбрать, а $\omega(x)$ это полином n-ой степени вида

$$(x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_n),$$

иногда обозначают

$$\prod_{m=0}^{n}(x-x_m).$$

И если мы K выбираем таким образом, что

$$K = R(x_*)/\omega(x_*),$$

а $\omega(x) \neq 0$ никогда, кроме x_i, тогда наша функция $F(x_i) = 0$, $i = 0, 1, \ldots, n$, правильно? Но кроме того имеется ещё точка $x = x_*$, где $F(x_*) = 0$, согласно выбору нашей константы, получаем, что у нас $(n+2)$ точки, где функция равна нулю. Пусть $f(x)$ имеет непрерывные $(n+1)$ производную, то есть

$$|f^{(n+1)}(x)| \leqslant M_{n+1}$$

на отрезке $[a, b]$, тогда согласно математическому анализу эта функция ограничена.

Вспомним теорему Ролля: если непрерывная функция имеет непрерывную производную и принимает одинаковые значения на концах отрезка, то она имеет производную, равную нулю, хотя бы в одной точке интервала (не отрезка, концы не включаются). Примерно так звучит, забыли? Я тоже забыл, но суть я помню. А у нас что: $F(x_i) = 0$ и $F(x_*) = 0$, значит что мы имеем, её первая производная $F' = 0$ в $(n+1)$ точке, $F'' = 0$ образует в n точках соответственно теореме и так далее. И $F^{(n+1)} = 0$ в какой-то одной точке $x = \eta$. Но $\eta \in (a, b)$, тогда согласно обозначениям,

$$F^{(n+1)} = \underbrace{P_n^{(n+1)}}_{=0} - f^{(n+1)}(\eta) - \frac{R(x_*)}{\omega(x_*)} \cdot (n+1)! = 0.$$

После несложных манипуляций я получу, что

$$R(x_*) = -\frac{f^{(n+1)}(\eta)}{(n+1)!} \cdot \omega(x_*),$$

вот как будет выглядеть ошибка в любой точке x_* не совпадающей с узлами сетки и

$$|R(x_*)| \leqslant \frac{M_{n+1}}{(n+1)!} \cdot |\omega(x_*)|.$$

Простейший вывод: чем больше n, тем меньше ошибка, тем глаже функция,

$$|\omega(x_*)| = \left|\prod_{m=0}^{n}(x_* - x_m)\right|,$$

то есть оказывается, что точность зависит от расположения узлов и эта проблема как выбрать наилучшее расположение узлов решил великий математик Чебышёв, от построения полинома с наименее удаляющимися значениями от нуля, то есть лучший из всех полиномов n-ой степени. Есть специальная теория полиномов Чебышева, но нас интересуют только эти узлы, очень просто вычисляются, когда $[a, b] = [-1, 1]$, но всегда можно сделать преобразование, чтобы свести к такому отрезку и там теорема со специальными узлами, но я приведу ответ:

$$x_i = \frac{1}{2} \cdot \left[(b-a) \cdot \cos\left(\frac{(2i+1)\pi}{2n+2}\right) + b + a\right],\ i = 0, 1, \ldots, n$$

узлы, которые рекомендуется использовать для получения наилучших результатов. Если у вас нет возможности использовать эти узлы, вы получите хуже ω, эти расположения называются наилучшими, если так расставлять, то

$$|P(x_*)| = \frac{M_{n+1}}{(n+1)!} \cdot \frac{(b-a)^{n+1}}{2^{n+1}}.$$

4.2 Способы построения интерполяционных полиномов

Пусть имеется система точек (x_i, f_i) и $i = 0, 1, \ldots, n$, но сначала пусть имеется только одно значение (x_0, f_0), как можно построить приближение, проэкстраполировать? Можно, если мы знаем производные функции в этой точке (помните формулу Тейлора?)

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x-x_0)^2}{2!} \cdot f''(x_0) + \ldots + \frac{(x-x_0)^{n+1}}{(n+1)!} \cdot f^{(n+1)}(\eta)$$

— с остаточным членом в форме Лагранжа, мы не можем писать бесконечный ряд, а $\eta \in [a, b]$. Но это не всегда исполняется, так как иногда нельзя вычислять такие производные, но это очень узкий класс функций, хотя все стандартные программы, которые имеются в операционных системах, пользуются такими рядами, все синусы, косинусы, катеты, гипотенузы. Но там специальные приёмы суммирования, точности.

Теперь, когда две точки: (x_1, f_1) и (x_2, f_2), можем использовать линейную интерполяцию, но она плохая, но вы ничего больше и сделать не сможете.

Остаётся общий случай, когда $(n+1)$ узел и способы построения полинома n-ой степени (он единственный, может только иметь разный вид).

1. Классический полином Лагранжа:
$$L_n(x) = P_n(x) = \sum_{i=0}^{n} f_i \cdot \ell_n^{(i)}(x), \tag{4.2.1}$$
где $\ell_n^{(i)}$ — полином степени n и $\ell_n^{(i)}$ элементарный полином или полином Лагранжа, они, исходя из формулы (4.2.1), имеют вид $\ell_n^{(i)}(x_i) = 1$, а все остальные $\ell_n^{(m)}(x_i) = 0$, $m \neq i$, тогда выполняется главное требование интерполирования: $P_n(x_i) = f_i$, то есть получаем требование на полином:
$$\ell_n^{(m)}(x_i) = \begin{cases} 1, & m = i, \\ 0 & m \neq i. \end{cases}$$
— более компактно.

   Тогда какую структуру они должны иметь:
$$\ell_n^{(i)}(x) = A^{(i)} \cdot (x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_{i-1}) \cdot (x - x_{i+1}) \cdot \ldots \cdot (x - x_n),$$
вот какой вид. А как коэффициент $A^{(i)}$ найти? Полином обращается в точках x_i в единицу:
$$\ell_n^{(i)}(x_i) = 1 = A^{(i)} \cdot (x_i - x_0) \cdot (x_i - x_1) \cdot \ldots \cdot (x_i - x_{i-1}) \cdot (x_i - x_{i+1}) \cdot \ldots \cdot (x_i - x_n),$$
тогда получается
$$A^{(i)} = \frac{1}{(x_i - x_0) \cdot (x_i - x_1) \cdot \ldots \cdot (x_i - x_{i-1}) \cdot (x_i - x_{i+1}) \cdot \ldots \cdot (x_i - x_n)},$$
а окончательный вид этих полиномов:
$$\ell_n^{(i)}(x) = \frac{(x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_{i-1}) \cdot (x - x_{i+1}) \cdot \ldots \cdot (x - x_n)}{(x_i - x_0) \cdot (x_i - x_1) \cdot \ldots \cdot (x_i - x_{i-1}) \cdot (x_i - x_{i+1}) \cdot \ldots \cdot (x_i - x_n)}.$$

   Мы не решали систему для $A^{(0)}$, $A^{(1)}$, …, но они будут иметь вид слишком сложный, это только для алгебры, в вычислительных методах его не записывают. Им пользуются для теоретических построений, для вычисления производных, интегралов, а так, на практике, используют полином для вычисления функций; в настоящее время не используется практически. Представьте появление нового значения, чтобы увеличить точность нам придётся всё пересчитывать, в пакетах он не используется. Для практических исследований используется полином Ньютона.

2. Полином Ньютона, он тоже даёт те же значения полинома в точках и больше приспособлен, хотя и в теории его используют. Его обозначают
$$N_n(x) = P_n(x) = C_0 + C_1(x - x_0) + C_2(x - x_0) \cdot (x - x_1) + \ldots + C_n(x - x_0) \cdot \ldots \cdot (x - x_{n-1})$$
и задача заключается в том, чтобы вычислить коэффициенты C_0, C_1 и так далее. Очень удобно для вычисления, так как попадает под схему Горнера, он почти уже записан в этом виде.

   Основное требование: $P_n(x_i) = f_i$, $i = 0, 1, \ldots, n$, из него вытекает соотношения для коэффициентов: $N_n(x_0) = f_0 = C_0$, все скобки сокращаются. Теперь $N_n(x_1) = f_0 + C_1(x_1 - x_0) = f_1$, дальше будут нули, отсюда
$$C_1 = \frac{f_1 - f_0}{x_1 - x_0},$$
это выражение называется *первой разделенной разностью* (в теореме интерполяции, так зовут) и обозначается $f(x_0, x_1)$. Теперь если мы потребуем совпадение нашего полинома в точке x_2, мы получим способ вычисления C_2, это уже будет вторая разделенная разность:
$$C_2 = f(x_0, x_1, x_2) = \frac{f(x_0, x_2) - f(x_0, x_1)}{x_2 - x_1} = \left(\frac{f(x_2) - f(x_0)}{x_2 - x_0} - \frac{f(x_1) - f(x_0)}{x_1 - x_0} \right) / (x_2 - x_1).$$
Третий коэффициент — третья разделенная разность и так далее. На практике поступают следующим образом, пишем такую таблицу:

$$\begin{array}{llll} x_0 & f_0 & f(x_0,\,x_1) & \\ & & & f(x_0,\,x_1,\,x_2) \\ x_1 & f_1 & f(x_1,\,x_2) & \\ & & & f(x_1,\,x_2,\,x_3) \\ x_2 & f_2 & f(x_2,\,x_3) & \\ & & & f(x_2,\,x_3,\,x_4) \\ x_3 & f_3 & f(x_3,\,x_4) & \\ & & \vdots & \end{array} \quad \begin{array}{l} \\ \\ \dots \\ \\ \dots \\ \\ \\ \end{array}$$

Чем удобен, если появляется новая точка, может быть и посередине отрезка где-то, то не обязательно перестраивать всю таблицу, неважно, как они идут, лишь бы не совпадали. Есть всякие модификации, когда шаг сетки постоянный, используются приёмы экономии, модернизации, тогда на вычислительной математике это всё было целесообразно, а сейчас, когда у вас такие компьютеры, все эти ухищрения не очень полезны. Хотя я говорю это крамола, но гнаться за этим не следует.

Часто используют полином Ньютона небольших степеней и много способов в графических пакетах, используют с плавающими отрезками, на самом деле это полином Ньютона, хотя называют сплайнами.

3. Сплайн интерполирование, идея родилась из практической задачи, инженеры придумали такой способ, сплайн в переводе на русский означает гибкая линейка, инженерам не хотелось пользоваться ни полиномом Лагранжа, ни Ньютона, не нужны производные, нам нужны только скорость и ускорение, а остальное только у математиков. И инженер что придумал, он в данные точки забивал гвоздики, а потом прокладывал гибкую металлическую линейку, потом брал и считал — просто и убедительно. Но оказалось, когда математика в 50 годах занялись серьёзно, что это то же самое (покойный профессор Завзялов занимался, сейчас это Классов изучает монографии, бурно развивается). 11.05.05

   Представим гвоздики в виде точек $(x_0,\,f_0),\ (x_1,\,f_1),\ \dots,\ (x_n,\,f_n)$, пусть ещё $h_i = x_i - x_{i-1}$ для $i = 1,\,2,\,\dots,\,n$ и выберем точку x не совпадающую с узлами $x_{i-1} \leqslant x \leqslant x_i$. Построим на отрезках $[x_i,\,x_{i-1}]$ полиномы третьей степени, которыми будем приближать нашу функцию

$$S_i(x) = a_i + b_i \cdot (x - x_i) + \frac{c_i}{2} \cdot (x - x_i)^2 + \frac{d_i}{6} \cdot (x - x_i)^3,$$

   где a_i, b_i, c_i и d_i некоторые неизвестные константы, теперь возьмём от него производную

$$S_i'(x) = b_i + c_i \cdot (x - x_i) + \frac{d_i}{2} \cdot (x - x_i)^2,$$

   итого получается на n отрезках n уравнений и у каждого по четыре неизвестных, нам нужно получить $4n$ уравнений. Для этого рассмотрим известные условия

   (а) В узловых точках значения функции нам известны, тогда для полинома должно выполняться

$$S_i(x_i) = f_i \equiv a_i \text{ — } \forall i = 1,\,2,\,\dots,\,n,$$

   а в точке x_0 так уже не получится

$$S_1(x_0) = f_0 \equiv a_1 + b_1 \cdot (x_0 - x_1) + \frac{c_1}{2} \cdot (x_0 - x_1)^2 + \frac{d_1}{6} \cdot (x_0 - x_1)^3 = a_1 - b_1 h_1 + \frac{c_1}{2} h_1^2 - \frac{d_1}{6} h_1^3,$$

   итого получилось $(n + 1)$ уравнение.

   (б) На границах — узловых точках, полиномы должны принимать одно значение, так как наша функция неразрывна, то есть

$$S_i(x_i) = S_{i+1}(x_i) \Rightarrow a_i = f_i = a_{i+1} - b_{i+1} h_{i+1} + \frac{c_{i+1}}{2} h_{i+1}^2 - \frac{d_{i+1}}{6} h_{i+1}^3 \tag{4.2.2}$$

   для всех $i = 1,\,2,\,\dots,\,n - 1$, то есть это ещё $(n - 1)$ уравнение.

   (в) Чтобы функция была непрерывной, её производные тоже должны совпадать на узлах, отсюда получается ещё одно условие на полиномы

$$S_i'(x_i) = S_{i+1}'(x_i) \Rightarrow b_i = b_{i+1} - c_{i+1} h_{i+1} + \frac{d_{i+1}}{2} h_{i+1}^2 \tag{4.2.3}$$

   опять же для $i = 1,\,2,\,\dots,\,n - 1$, получаем ещё $(n - 1)$ уравнение, но и этого недостаточно.

(г) Можно продолжить в том же духе, приравняв и вторые производные

$$S_i''(x_i) = S_{i+1}''(x_i) \Rightarrow c_i = c_{i+1} - d_{i+1}h_{i+1}, \tag{4.2.4}$$

ещё $(n-1)$ уравнение, не хватает ещё только два уравнения, их мы возьмём из граничных условий линейки, считаем, что она уходит за область гвоздей по прямой — естественные граничные условия.

(д) В начальной и конечной точках, чтобы была прямая, коэффициенты искривления должны быть

$$S_1''(x_0) = 0 = c_1 - d_1h_1, \quad S_n''(x_n) = 0 = c_n.$$

Осталось найти общее решение всех этих уравнений, из уравнения (4.2.2) получаем

$$b_{i+1} = \frac{f_{i+1} - f_i}{h_{i+1}} + \frac{c_{i+1}}{2}h_{i+1} - \frac{d_{i+1}}{6}h_{i+1}^2,$$

из (4.2.4)

$$d_{i+1} = \frac{c_{i+1} - c_i}{h_{i+1}},$$

если подставить это в (4.2.3)

$$\frac{f_i - f_{i-1}}{h_i} + \frac{c_i}{2}\cdot h_i - \frac{d_i}{6}\cdot h_i^2 = \frac{f_{i+1} - f_i}{h_{i+1}} + \frac{c_{i+1}}{2}\cdot h_{i+1} - \frac{d_{i+1}}{6}\cdot h_{i+1}^2 - c_{i+1}h_{i+1} + \frac{d_{i+1}}{2}\cdot h_{i+1}^2$$

$$\frac{c_i}{2}\cdot h_i - \frac{c_i - c_{i-1}}{6}\cdot h_i - \frac{c_{i+1}}{2}\cdot h_{i+1} + c_{i+1}h_{i+1} - \frac{c_{i+1} - c_i}{3}\cdot h_{i+1} = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}$$

$$\frac{c_i}{3}\cdot h_i + \frac{c_{i-1}}{6}\cdot h_i + \frac{c_{i+1}}{6}\cdot h_{i+1} + \frac{c_i}{3}\cdot h_{i+1} = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}$$

$$c_{i+1}h_{i+1} + 2c_i\cdot(h_{i+1} + h_i) + c_{i-1}h_i = 6f(x_{i-1}, x_i, x_{i-1}) = 6\cdot\left[\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}\right].$$

Так как $c_0 = 0$ (его вообще нет) и $c_n = 0$, то решение имеет вид трёхдиагональной матрицы с преобладающими диагональными элементами, то есть погрешность этого метода мы знает — $O(h)$, его мы решаем методом прогонки.

Это был классический кубический сплайн, по теореме Самарского можно оценить его точность

$$|f(x) - S(x)| \leqslant C_1\cdot h^4,$$

где C_1 — независимая константа, а h^4 — средний шаг, отсюда

$$f' \simeq O(h^3), \quad f'' \simeq O(h^2).$$

4.3 Другие способы интерполяции

Допустим нам ещё известны значения производной функции в узловых точках, $(x_i,\, f_i,\, f_i')$ — $i = 0,\, 1,\, \ldots,\, n$, что мы тогда можем сделать

1. Полином Эрмита, выглядит в общем виде очень громоздко, выпишем

$$H_{2n}(x) = \sum_{i=0}^{n}(x - x_i)\cdot\left[\frac{\omega''(x_i)}{\omega'(x_i)}\cdot f_i - f_i'\right]\cdot(\ell_n^i(x))^2,$$

то есть он всегда от чётного числа $2n$, функция $\omega(x) = (x - x_0)\cdot(x - x_1)\cdot\ldots\cdot(x - x_n)$, и последнее — полином Лагранжа $\ell_n^i(x)$. Пользоваться этим полиномом имеет смысл только для достаточно малых n. Поэтому для него математики придумали некоторое обобщение.

2. Обобщённый полином Лагранжа, в нём используется функции

$$\varphi_k(x) = \sin\left(\frac{k\pi x}{b - a}\right),\ k = 0,\, 1,\, \ldots,\, n,$$

тогда полином имеет вид

$$L_n^0(x) = \sum_{k=0}^{n} C_k\cdot\varphi_k(x)$$

и условие на него, что $L_n(x_i) = f_i$ даёт все коэффициенты. Так же здесь можно использовать $\varphi_k(x) = e^{kx}$.

3. Среднее квадратичное приближение, выберем погрешность ε_i, тогда наше вычисляемое $\tilde{f}_i = f_i + \varepsilon_i$, здесь мы не требуем строгого $P_n(x_i) = f_i$, нужно добиться, чтобы $\tau_i = P_n(x_i) - \tilde{f}_i$ было минимальным, тогда выбираем полином

$$R = \sum_{i=0}^{n} \tau_i^2.$$

Глава 5

Численное интегрирование и дифференцирование

Здесь мы будем рассматривать интеграл

$$I = \int_a^b f(x)\,dx$$

для заданной $f(x)$, компьютер, он же у вас дискретный, он не может взять и посчитать определённый интеграл, только с некоторой степенью приближения. Конечно были и есть проекты, которые вычисляют интегралы по формулам, но пока это слабо развито.

5.1 Теоретическая основа формул численного интегрирования

Для вычисления приближённого значения интеграла используют *квадратурные формулы*, которые вычисляют площадь. Итак, есть два подхода, либо это вычисление среднего и вычисление кусочных площадей, либо через приближение полиномом, мы рассмотрим оба способа.

Здесь нам пригодится одна теорема из математического анализа

Теорема 5.1.1 (О среднем в интегральном исчислении)**.**

▷ Для любого интеграла непрерывной функции существует такая точка η из интервала интегрирования, что

$$\int_a^b f(x)\,dx = f(\eta)\cdot(b-a).$$

Но в таком способе возникает ошибка величиной $f(\eta) - f(a)$, но тогда мы возьмём отрезок маленький, тогда и ошибка будет меньше. Разобьём наш обрезок на

$$a = x_0 < x_1 < \ldots < x_n = b,$$

где $h_i = x_{i+1} - x_i$ — произвольный шаг. Тогда по свойствам интеграла, которые вы уже знаете, получаем из интеграла сумму

$$I = \int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx,$$

свели задачу к вычислению интеграла на элементарных отрезках

$$I_i = \int_{x_{i-1}}^{x_i} f(x)\,dx = f(\eta_i)\cdot h_i$$

— по теореме о среднем, которую мы рассмотрели. Теорема эта хороша, но она не говорим нам где эта точка, говорит только, что она существует, поэтому рассмотрим как её найти.

5.2 Способы построения квадратурных формул и их точность

Для такого же разбиения нашего отрезка на x_i, возьмём середины отрезков разбиения $y_i = (x_i + x_{i-1})/2$ и разложим нашу функцию в каждой из них в ряд Тейлора

$$f(x) = f(y_i) + f'(y_i)\cdot(x-y_i) + \frac{f''(y_i)}{2}\cdot(x-y_i)^2 + \frac{f^{(3)}(y_i)}{6}\cdot(x-y_i)^3 + \frac{f^{(4)}(\theta_i)}{24}\cdot(x-y_i)^4$$

с остаточным членом в виде Лагранжа, если взять

$$A_i = f(y_i),\ B_i = f'(y_i),\ C_i = \frac{f''(y_i)}{2},\ D_i = \frac{f^{(3)}(y_i)}{6},\ E_i = \frac{f^{(4)}(\theta_i)}{24},$$

то получим что хотим, здесь A_i даже известны сразу, они проста даны. Рассмотрим определённый интеграл на элементарном отрезке от этого разложения

$$I_i = f(y_i)\cdot x\Big|_{x_{i-1}}^{x_i} + f'(y_i)\cdot\frac{(x-y_i)^2}{2}\Big|_{x_{i-1}}^{x_i} + f''(y_i)\cdot\frac{(x-y_i)^3}{6}\Big|_{x_{i-1}}^{x_i} + f^{(3)}(y_i)\cdot\frac{(x-y_i)^4}{24}\Big|_{x_{i-1}}^{x_i} + f^{(4)}(y_i)\cdot\frac{(x-y_i)^5}{120}\Big|_{x_{i-1}}^{x_i} =$$
$$= f(y_i)\cdot h_i + 0 + \frac{f^{(2)}(y_i)}{6}\cdot\left[\left(\frac{h_i}{2}\right)^3\cdot 2\right] + 0 + \frac{f^{(4)}(\theta_i)}{120}\cdot\left[\left(\frac{h_i}{2}\right)^5\cdot 2\right],$$

но последние ненулевые слагаемые мы считаем малыми, если f' ограничена при соответствующим h_i. Таким образом $R_i = f(y_i)h_i$ — *формула средних* (иногда называют формула прямоугольника), её точность известна

$$O(R_i) = |I_i - R_i| \leqslant \frac{M_2\cdot h_i^3}{24}.$$

Помимо этого способа применяют ещё формулу трапеции

$$T_i = \left(\frac{f_i + f_{i-1}}{2}\cdot h_i\right),$$

её погрешность

$$|T_i - I_i| = O(T_i) \leqslant \frac{M_2\cdot h_i^3}{12},$$

то есть в два раза хуже, но зато у R_i порядок хуже. Итого конечные формулы вычисления для этих способов

$$R = \sum_{i=1}^{n} R_i,\quad T = \sum_{i=1}^{n} T_i.$$

Точность этих формул не достаточно велика, поэтому была придумана ещё особая формула Симпсона, которая комбинирует предыдущие два способа

$$S_i = \frac{2R_i + T_i}{3},$$

оценим её погрешность, для этого выделим числитель

$$2\cdot(I_i - R_i) + (I_i - T_i) = 3I_i - (2R_i + T_i),$$

но мы знаем, что

$$I_i - R_i = \frac{f''(y_i)}{24}\cdot h_i^3 + O(h^5),\quad I_i - T_i = \frac{f''(y_i)}{12}\cdot f_i^3 + O(h^5),$$

при подстановке получится, что $I_i - S_i = O(h^5)$, то есть уже пятый порядок, а в полной сумме получится четвёртый порядок. Для этого способа можно записать конечную формулу для вычисления

$$S_i = \frac{f_{i-1} + 4f_i + f_{i+1}}{6}.$$

18.05.05

5.3 Способы повышения точности квадратурных формул

5.3.1 Адаптивные формулы

Идея очень простая, заключается в том, чтобы заставить компьютер повысить точность, допустим мы по простой формуле посчитали интеграл $I_i = \int_{x_{i-1}}^{x_i} f(x)\,dx$ и возьмём $\tilde{I}_i = I_i + O(h^p)$, где $|O(h^p)| \leqslant C \cdot h^p$, а $\int_a^b f(x)\,dx = \sum_{i=1}^n I_i$. Тогда для того, чтобы повысить точность, мы проводим серию расчётов:

1. $h_1 = h$ — с постоянным шагом и получаем $\tilde{I}_i^{(1)} = I_i + O(h_1^P)$;
2. С шагом $h_2 = q \cdot h_1$, где $0 < q < 1$ — некоторый коэффициент, обычно выбирают $1/2$, но не обязательно, то есть шаг h_2 в два раза меньше. В тех же точках мы получаем

   $$I_i^{(2)} = I_i + O(h_2^p) = I_i + O(q^p \cdot h_1^p).$$

   Мы получили расчёт 1 и расчёт 2, знаем как получать ошибку и мы можем скомбинировать. Дальше идут

   $$\tilde{I}_i^{(1)} = I_i + O(h_1^p) + O(h_1^{p+1}) + \dots$$

   — более высоких порядки и мы постараемся компенсировать главную ошибку.

   Умножим расчёт 1 на q^p и вычтем из расчёта 2:

   $$\tilde{I}_i^{(2)} - q^p \cdot \tilde{I}_i^{(1)} = I_i \cdot (1 - q^p) + 0 + O(h_1^{p+1}).$$

3. Если возьмём

   $$\tilde{I}_i^{(3)} = \frac{\tilde{I}_i^{(2)} - q^p \cdot \tilde{I}_i^{(1)}}{1 - q^p},$$

   то тогда $q^p < q$ и мы получаем $\tilde{I}_i^{(3)} - I_i = O(h_1^{p+1})$ и мы можем добиться для любой наперёд заданной точности, $h_3 = qh_2$ и так далее, но только в точках общих и получим погрешность сколь угодно малую.

Это такой вот рутинный, машинный способ, только надо знать точность расчёта.

5.3.2 Использование интерполяционных формул

Ещё называется *квадратурные интерполяционные формулы*, идея в том, чтобы привлечь вместо $f(x)$ её интерполяцию и здесь привлечён показатель интерполяций (квадратный или кубический сплайны), то есть функцию $f(x)$ на отрезке $[x_{i-1},\, x_i]$ заменяем полиномом второй степени, то есть f_{i-1}, $f_{i-1/2}$ и f_i, парабола у нас как выглядит: $ax^2 + bx + c$, в ней три коэффициента, они определяются из условия, что

$$\begin{cases} f(x_{i-1}) & = ax_{i-1}^2 + bx_{i-1} + c, \\ f(x_{i-1/2}) & = ax_{i-1/2}^2 + bx_{i-1/2} + c, \text{ ,} \\ f(x_i) & = ax_i^2 + bx_i + c. \end{cases}$$

если вы их определили, проинтегрируем, мы получим интеграл, он как раз совпадает с Симпсонами, то есть локальная точность $\tilde{I}_i = I_i + O(h^5)$, а глобальная точность $O(h^4)$, когда мы все отрезки используем.

Но можно заменить функцию, чаще всего делают кубическим сплайном.

Кубические сплайны

То есть функцию $f(x)$ мы заменяем $S_3^i(x)$ и вычисляем интеграл

$$\tilde{I}_i = \int_{x_{i-1}}^{x_i} S_3^{(i)}(x)\,dx,$$

локальная точность $O(h^5)$, а глобальная $O(h^4)$, так как привлекаются две точки, не смотря на кубический сплайн.

Приближение полиномом

Идея в том, что рассматривается $I = \int_a^b f(x)\,dx$ и он аппроксимируется как $\sum_{i=0}^{n} C_i \cdot f_i$, где f_i — вычисленные или заданные значения функции $f(x)$, а C_i — коэффициент, который надо определить и точность вычисляется в зависимости от как мы заменим $f(x)$ на полином

$$P_n(x) = \sum_{i=0}^{n} \ell_n^{(i)}(x) \cdot f_i,$$

где $\ell_n^{(i)}$ — полином Лагранжа, можно и Ньютона использовать и

$$C_i = \int_{x_{i-1}}^{x_i} \ell_n^{(i)}(x)\,dx$$

— для полинома Лагранжа, который мы строили в предыдущей главе. Такие формулы имеют достаточно высокую точность, но при большом n они становятся неустойчивыми и используется больше теоретически, увеличиваются ошибки округления, число интервалов нельзя много брать. Поэтому Гаусс придумал свою гауссовскую квадратуру, немного особняком стоит.

5.3.3 Метод Гаусса

Итак, $\Gamma = P_m = \sum_{i=1}^{m} c_i \cdot f_i$, где m обычно нечётная величина, $m = 2n + 1$ и строится из требования наивысшей алгебраической точности, это означает, что если $f(x) = P_m(x)$, то формула Гаусса должна давать точное значение интеграла (то есть такой способ построения квадратичной формулы, что интеграл даёт точных результат, если полином степени m). Принцип отличается тем, что узлы x_i выбираются специальным способом, то есть

$$\int_a^b \omega(x) \cdot P_k(x)\,dx = 0,\ k = 1,\, 2,\, \ldots,\, m,$$

где $\omega(x) = (x - x_1) \cdot (x - x_2) \cdot \ldots \cdot (x - x_m)$. Из этих уравнений и получается. Есть специальная таблица как вычислять, но редко используется, в основном для теоретических исследований, если $m = 1$, то получается по методу среднего: $I_i = f(y_i)h_i$, формулы там сложные.

Сейчас все эти формулы редко используют, они слишком сложные для этого, обычно берут адаптивный и начинают работу.

5.3.4 Формула Эйлера

Грех не упомянуть формулу Эйлера (это же великий математик), отличается от предыдущей тем, что построение заключается в том, что построив квадратичную формулу, высший порядок точности — пятый, если заданы и производные f_i' и формула имеет вид

$$E_i = T_i - \frac{h_i^3}{12} \cdot \frac{f_i' - f_{i-1}'}{h_i},$$

где T_i — из формулы трапеций,

$$T_i = \frac{h_i^3}{12} \cdot f''(y_i) + O(h^5) = \frac{f_i + f_{i-1}}{2} \cdot h_i,$$

то есть просто взял и учёл погрешность, а вторую производную вычисляю с помощью разностной формулы

$$f''(y_i) \approx \frac{f_i' - f_{i-1}'}{h_i} + O(h_i^2)$$

— центральная разность и согласно теореме численного дифференцирования получается такая точность, тогда он подставил туда и тогда $h^3 \cdot h^2 = h^5$ и даёт нужную точность.

Если построить составную формулу Эйлера, то есть просуммировать, мы получим значение интеграла от a до b, для $E = \sum F_i$ все слагаемые сокращаются и надо будет взять все T_i и отнять $h \cdot (f_0' - f_n')/12$, если шаг n — последний. Она используется, если известны кроме значений самой функции в узлах ещё и производные.

Глава 6

Численные методы решения задач Коши и краевых задач для обыкновенных дифференциальных уравнений

Эта глава будет последней.

6.1 Постановка задачи и свойства решения

Задача Коши обыкновенного дифференциального уравнения имеет вид $y' = f(x, y)$, где y' — производная, а $f(x, y)$ — заданная функция, под решением задачи Коши понимается следующая задача: найти функцию $y(x)$, удовлетворяющую начальному условию $y(x_0) = y_0$. Эта задача ещё называется *с начальным условием*. Вместо y функции может рассматриваться вектор функция

$$\vec{y} = \begin{pmatrix} y_1(x) \\ \vdots \\ y_n(x) \end{pmatrix}$$

— несколько функций, тогда естественно несколько правых частей, система уравнений имеющих такой же вид и условия Коши. Когда будет это необходимо, я буду рассматривать вектор функцию, а обычно — скаляр, так как для вектора функций всё аналогично, единственное, что надо, рассмотреть их несколько. В курсе дифференциальных уравнениях есть соответствующая теорема, показывающая при каких условиях существует единственное решение.

Задача Коши в дифференциальных уравнениях называется *корректно поставленной*, если

1. Решение существует;
2. Решение единственно;
3. Решение непрерывным образом зависит от входных данных, то есть малому изменению входных данных соответствует малое изменение решения, это называется ещё требованием устойчивости, то есть если мы изменим $y_0 \to y_0 + \delta$, то для решения $|y^{\text{нов.}}(x) - y^{\text{стар.}}(x)| = c\delta$.

Такая формулировка корректности математической задачи, это для любого уравнения.

Так вот для уравнения $y' = f(x, y)$ с условием $y(x_0) = y_0$ существуют теоремы существования и единственности, это теоремы Пеано и Коши, мы сформулируем обобщённую теорему.

Теорема 6.1.1 (Существование и единственность решения)**.**

▷ Задача Коши будет корректно поставленной, если функция $f(x)$ с входными данными y_0 непрерывна по обоим аргументам (по x и по y) и по аргументу y удовлетворяет условию Липшица, то есть

$$|f(x, y_1) - f(x, y_2)| \leqslant L \cdot |y_1 - y_2|,$$

мы с вами давали определение 1.2.1.1 на стр. 5, где L — константа, может зависеть от x, но с точки зрения y она константа.

Если функция $f(x, y)$ имеет m непрерывных производных по обоим аргументам, то решение будет иметь $(m + 1)$ непрерывную производную. Общая теорема получилась из курса дифференциальных уравнений. Мы будем всегда считать, что $f(x, y)$ имеет столько производных. сколько надо для построения вычислительного метода.

Под краевой задачей для обычных дифференциальных уравнений подразумевается найти решение $y(x)$, удовлетворяющее в окрестности $x_0 \leqslant x \leqslant x_n$ уравнению $ay'' + by' + cy = f(x)$, с так называемыми краевыми условиями

$$\begin{cases} y(x_0) = y_0, \\ y(x_n) = y_n. \end{cases}$$

— первая краевая задача или с условиями

$$\begin{cases} y'(x_0) = v_0, \\ y'(x_n) = v_n. \end{cases}$$

— вторая краевая задачи (Дирихле и условия Неймона) или с условиями

$$\begin{cases} \alpha_0 y(x_0) + \beta_0 y'(x_0) & = \gamma_0, \\ \alpha_1 y(x_n) + \beta_1 y'(x_n) & = \gamma_1. \end{cases}$$

— третья краевая задача. Три ключевые краевые задачи для этого уравнения, есть ещё и неклассические с нелинейными краевыми условиями, но мы рассматривать не будем, для решения краевых задач второго и третьего рода численные методы позволяют построить такой алгоритм, что решение можно свести к решению задачи первого рода, а потом построить для них, главное научиться решать её, поэтому уравнения рассматривать и не будем за неимением времени.

Кроме того можно с помощью замены искомой функции привести эти условия к однородным, то есть построить задачу, когда $y(x_0) = y(x_n) = 0$, это вы делали, наверное, в курсе дифференциальных уравнений.

Обычно уравнения в таком виде не рассматриваются, так как оператор задаётся как

$$L = a\frac{d^2}{dx^2} + b\frac{d}{dx} + cE,$$

где E — тождественных оператор, и задачу можно записать как $Ly(x) = f(x)$, называется линейной краевой задачей, если коэффициенты a, b и c являются константами или зависят только от x и $a \neq 0$, иначе получаем уравнение первого порядка и краевая задача для неё некорректна, либо не устойчива, либо не существует решение.

Чтобы оператор L был более простой, делают замену $y(x) = e^{\alpha(x)} \cdot z(x)$, где $z(x)$ неизвестная функция, показатель экспоненты $\alpha(x)$ выбран так, чтобы после замены мы получили задачу для $z(x)$ с оператором $z'' + d \cdot z = \varphi(x)$, а потом $y(x)$ восстанавливаем, $\varphi(x)$ — новая правая часть. Если a и b — константы, то $\alpha = (-b/2a) \cdot x$ — из дифференциальных уравнений, если a и b — функции от x, то

$$\alpha = -\int_0^x \frac{b}{2a}\,dx, \quad \varphi(x) = A(x) \cdot f(x)$$

и $A(x)$ как-то определяется. Вместо решения такой сложной задачи, можно решать простую с заданными $\varphi(x)$, α и краевым условием $z(x_0) = z(x_n) = 0$.

Из теории дифференциальных уравнений известно, что если $\alpha \leqslant 0$, то решение такой задачи существует и единственно, также устойчиво, то есть задача корректна, если не выполнено, могут возникнуть проблемы, не единственность или несуществование решения.

Уравнения в частных производных сводятся к этим и вычислительные методы вокруг них вертятся, идея этой концепции — расщипить сложную задачу на более простые, например многомерную в одномерные (Понаморчук этим занимался, сейчас уже на пенсии, Яненко покойник и Самарский, поныне здравствующий).

6.2 Численные методы решения задачи Коши

Основная идея, как я всегда говорю, с чего начинается родина (численные методы), скажем, мы решили задачу Коши $y' = f(x,\, y)$ и любое условие задачи $y(x_0) = y_0$ и мы хотим заставить компьютер решать. Рассмотрим решение в области $x \geqslant x_0$, до бесконечности считать не можем, поэтому обычно ограничиваются $x \leqslant L$. С помощью масштабирования можем считать, что $0 \leqslant \bar{x} \leqslant 1$ (сдвину и сожму), будем считать, что сразу всё проделали. Отрезок $[0,\, 1]$ разбивается на N интервалов (такое принято обозначение), то есть представляем как систему $x_i = hi,\ i = 0,\, 1,\, \ldots,\, N$, а $h = 1/N$. Эта сеточная область называется ω_h, то есть замена области непрерывного изменения аргумента на дискретную, так как компьютеру удобнее работать с дискретными величинами, мы облегчаем ему жизнь. И тогда задаются свойства к решению в точках $y(x_i)$, то есть к полученной дискретной функции дискретного аргумента $y(x_i) = y_i,\ i = 0,\, 1,\, \ldots,\, N$, а компьютеру удобнее получить таблицу.

Некоторые пакеты дают и формулу, но не для всех задач, но это экзотика. То есть мы задаём свободную к решению задачу на элементарном отрезке. И дальше существует два типа методов.

6.2.1 Одношаговые методы

Ещё называются Рунге-Кутты — женщина такая была, иногда, почему-то, пишут методы Рунге-Кутта. (Мужчины сильные натуры, а миром правят бабы — дуры.)

Идея заключается в следующем: на элементарных отрезках функция выглядит как $y'(x) = f(x, y(x)) = F(x)$ какая-то, тогда если я проинтегрирую левую и правую часть, я получу интеграл от x_{i-1} до x_i, таким образом

$$y(x_i) - y(x_{i-1}) = \int_{x_{i-1}}^{x_i} F(x)\,dx =_{\text{(формула средних)}} F(\theta_i) \cdot h_i$$

и я получаю формулу

$$y(x_i) = y(x_{i-1}) + h_i \cdot F(\theta_i),$$

зная x_0 я получу и x_1 и так далее, нет проблем. Но оказывается проблема есть, теорема о среднем говорит, что θ_i существует, но как искать не говорит. Метод Гунге-Кутты говорит как получить, чтобы получить с заданной точностью.

6.2.2 Метод Эйлера

Нужно построить алгоритм нахождения $F(\theta_i)$, Эйлер предлагал взять $F(\theta_i) = f(x_{i-1}, y_{i-1})$, тогда получаем

$$\tilde{y}_i = \tilde{y}_{i-1} + h_i \cdot f(x_{i-1}, y_{i-1})$$

и ясно, мы получим достаточно грубый результат, $\tilde{y}_i - y_i = z_i$ и это не одно и то же, а оценка погрешности $|z_i| \leqslant ?$ Ещё называется *метод ломаных*.

Возникает вопрос, а какая точность? Для этого надо оценить погрешность z_i, построим некоторую формулу для $z_i = \tilde{y}_i - y(x_i)$, выразим $\tilde{y}_i = y_i + z_i$ и подставляем в расчётную формулу, тогда получим соотношения

$$z_i + y_i = z_{i-1} + y_{i-1} + h_i \cdot f(x_{i-1}, y_{i-1} + z_{i-1})$$
$$z_i = z_{i-1} + h_i \cdot \left[f(x_{i-1}, y_{i-1} + z_{i-1}) - \frac{y_i - y_{i-1}}{h_i} \right],$$

но последняя дробь — аппроксимация первой производной, согласно ряду Тейлора

$$y(x_i) = y(x_{i-1}) + h_i \cdot y'(x_{i-1}) + \frac{h_i^2}{2} \cdot y''(\eta_i)$$

— с остаточным членом в форме Лагранжа, тогда получаю

$$y_i = y_{i-1} + h_i \cdot f(x_{i-1}, y_{i-1}) + O_i(h^2),$$

где $|O_i(h^2)| \leqslant h^2 M_2/2$, где

$$h = \max_i h_i, \text{ а } M_2 = \max_{x_1 \leqslant x \leqslant x_n} |y''(x)|,$$

тогда легко получаем, что

$$\frac{y_i - y_{i-1}}{h_i} = f(x_{i-1}, y_{i-1}) + O(h),$$

а функция у нас непрерывна по обоим переменным, то есть

$$f(x_{i-1}, y_{i-1} + z_{i-1}) - f(x_{i-1}, y_{i-1}) = f'_y(x_{i-1}, \delta_i) \cdot z_{i-1},$$

а как оценивается разность по модулю:

$$|f(x_{i-1}, y_{i-1} + z_{i-1}) - f(x_{i-1}, y_{i-1})| \leqslant |z_{i-1}| \cdot M_{21},$$

где

$$M_{21} = \max_{\substack{x_1 \leqslant x \leqslant x_n \\ A \leqslant y \leqslant B}} |f'_y(x, y)|,$$

но мы получили, что

$$|z_i| = \left| z_{i-1} + h_i \cdot \left[f(x_{i-1}, y_{i-1} + z_{i-1}) - f(x_{i-1}, y_{i-1}) + f(x_{i-1}, y_{i-1}) - \frac{y_i - y_{i-1}}{h_i} \right] \right| \Rightarrow$$
$$\Rightarrow |z_i| \leqslant |z_{i-1}| + h_i \cdot [|z_{i-1}| \cdot M_{21}] + h_i \cdot O(h),$$

обозначим $q = 1 + hM_{21}$, тогда приведя подобные получим

$$|z_i| = q \cdot |z_{i-1}| + h \cdot O(h), \quad |z_1| \leqslant h \cdot O(h), \quad |z_2| \leqslant (1+q)h \cdot O(h),$$

тогда

$$|z_i| \leqslant (1 + q + \ldots + q^{i-1})h \cdot O(h) = \left(\frac{1-q^i}{1-q}\right) h \cdot O(h),$$

правильно? Теперь вспомним наше обозначение q, смотрим

$$q^i = \left(1 + \frac{M_{12}}{N}\right)^i \leqslant \left(1 + \frac{M_{21}}{N}\right)^N \leqslant e^{M_{21}}$$

и совсем уже

$$|z_i| \leqslant \frac{e^{M_{21}}}{M_{21}} \cdot \frac{h}{h} \cdot M_{21} = O(h),$$

то есть чем больше у нас число N, тем у нас лучше будет результат, то есть взяли задачу и оценили погрешность, h мы можем назначать, зависит от задачи, а она от заказчика — преподавателя, дающего семестровые задания, а он почти Бог, так что это всё от Бога. Этот метод сходится для очень широкого класса задач, но погрешность не достаточно хороша, первый порядок. Поэтому естественные усилия математиков после этого были, например, на увеличение точности, формулы то здесь простые. 25.05.05

6.2.3 Повышение точности метода Эйлера

База — метод Эйлера, остальное всё крутится вокруг.

Предиктор-Корректор

Точное решение выглядит

$$y(x_i) = y(x_{i-1}) + h_i \cdot F(\theta_i),$$

где

$$y_i - y_{i-1} = \int_{x_{i-1}}^{x_i} f(x,\, y(x))\, dx = F(\theta_i) \cdot (x_i - x_{i-1}),$$

мы вычисляем F по значению f в точках x_i и x_{i-1} по формуле среднего, чтобы повысить точность, можно повысить точность вычисления интеграла

$$F(\theta_i) \cdot (x_i - x_{i-1}) \simeq F(x_{i-1/2}) \cdot h_i, \Rightarrow F(\theta_i) = f(x_{i-1/2},\, \tilde{y}_{i-1/2}),$$

но величина $\tilde{y}_{i-1/2}$ неизвестна и её вычисляют по методу Эйлера

$$\tilde{y}_{i-1/2} = y_{i-1} + \frac{h_i}{2} \cdot f(x_{i-1},\, \tilde{y}_{i-1}),$$

а потом $\tilde{y}_i = \tilde{y}_{i-1} + h_i \cdot f(x_{i-1/2},\, \tilde{y}_{i-1/2})$, можно посчитать, что погрешность второго порядка, так как это даёт формула среднего значения, предиктор здесь предсказывает (на пол шага), а коллектор исправляет, $z_i \leqslant O(h^2)$.

По методу трапеции

Ещё мы использовали метод трапеций для интегралов, то есть

$$\int_{x_i}^{x_{i-1}} f(x,\, y(x))\, dx \simeq (F(x_{i+1}) + F(x_i)) \cdot \frac{h_i}{2},$$

но $F(x_i)$ неизвестна, потому строится предварительно значение $\widetilde{\tilde{y}}_i$ по методу Эйлера $\widetilde{\tilde{y}}_i = y_{i-1} + h_i \cdot f(x_{i-1},\, \tilde{y}_{i-1})$, потом

$$\tilde{y}_i = y_{i-1} + h_i \cdot \left(f(x_{i-1},\, \tilde{y}_{i-1}) + f(x_i,\, \widetilde{\tilde{y}}_i)\right)/2,$$

будет обладать по крайней мере вторым порядком (глобальная точность).

6.2.4 Метод Рунге-Кутты

Построим по-крутому

$$F(\theta_i) = \sum_{n=1}^{m} \sigma_n \cdot F_n,$$

но чтобы была аппроксимация, выбираем $\sum_{n=1}^{m} \sigma_n = 1$, а F_n выбираем таким образом, что $F_1 = f(x_{i-1},\, y_{i-1})$, а все остальные рекуррентно

$$F_n = f(x_{i-1} + a_n h_i,\, y_{i-1} + b_{n1}F_1 + b_{n2}F_2 + \ldots + b_{nm}F_m),$$

здесь минимум третий порядок, чтобы четвёртый, надо формулу Симпсонов. Существует известные способы нахождения σ и самый популярный способ, когда $m = 4$; но я не пишу формулы, так как они уже хорошо реализованы в пакетах, вам главное понимать, почему она едет, почему считает и когда все условия выполнены, не имеет разрывов и прочее.

6.2.5 Многошаговые (многоточечные) методы Адамса

Как в построении квадратурной формулы заменой функции $f(x,\, y)$ на полином $P_n(x)$, подставляем её и получили. Но это не так просто построить полином, нужны значения.

Пусть вычислено несколько значений функции $y_0,\, y_1,\, \ldots,\, y_m$, допустим при помощи метода Рунге-Кутты, тогда для этих значений можем вычислить значения $f_0,\, f_1,\, \ldots,\, f_m$, правильно? Если мы знаем таблицу f и x, то мы вместо $f(x)$ на интервале $x_0 \leqslant x \leqslant x_m$ можем поставить полином

$$f(x) = P_m(x) = \sum_{i=0}^{m} f_i \cdot \ell_m^i(x),$$

где $\ell_m^i(x)$ — элементарный полином Лагранжа, и тогда

$$\tilde{y}_i = \tilde{y}_{i-1} + \int_{x_{i-1}}^{x_i} P_n(x)\,dx = \tilde{y}_{i-1} + \sum_{i=0}^{m} b_{im} \cdot f_i,$$

то есть что мы посчитали, мы используем в дальнейших расчётах, это очень эффективно, это *метод Адамса*. Смотрим, как вычисляются коэффициенты

$$b_{in} = \int_{x_{i-1}}^{x_i} \ell_m^i(x)\,dx.$$

Для метода четвёртого порядка (Адамса) эти величины выглядят таким образом: если $h_i = h$, то значения для $m = 3$ это

$$b_1 = \frac{23}{12}, \quad b_2 = -\frac{4}{3}, \quad b_3 = \frac{5}{12}$$

и так далее, всё подсчитано хорошими людьми. Но прежде надо по какому-то методу вычислить предварительные значения, приходится писать сложные задачи, переключающиеся между двумя способами, но помогает повышать точность.

И в заключении ещё метод вытравливания шага, можно использовать машиной адаптивный метод, просто вывести из метода вычисления интеграла, то есть из квадратурных методов.

6.3 Численные методы для решения краевых задач для обыкновенного дифференциального уравнения второго порядка

Я уже формулировал условие задачи и я говорил, что всё можно свести к задаче с двумя слагаемыми, не умоляя общности, можем решать задачу $-y'' + cy = f(x)$ с однородными краевыми условиями $y(0) = y(1) = 0$, искомой является функция $y(x)$ на отрезке $0 \leqslant x \leqslant 1$. Уравнения общего вида с коэффициентами всегда можно свести, краевые условия к таким, то есть если научимся решать это, научимся решать и общего вида.

С чего начинается Родина: строится сетка

$$\omega_n\colon \left\{ x_i = ih,\ i = 0,\, 1,\, \ldots,\, N,\, h = \frac{1}{N} \right\},$$

то есть от области непрерывного аргумента переходим к функции дискретного аргумента $y(x_i) = y_i$. Ясно, нам вместо дифференциальной оператора задачи

$$Ly(x) = f(x),\ \text{где } L = \frac{d^2}{dx^2} + dE$$

надо перейти к дискретному оператору (E — тождественный оператор), то есть заменить L на L_n. Вам уже рассказывали (смотрите следующую лекцию — прим. ред.), что

$$y''(x)_{x=x_i} \simeq \frac{y(x_i+h)-2y(x_i)-y(x_i-h)}{h^2}+O(h^2) \quad \text{и это} \quad |O(h^2)| \leqslant \frac{h^2}{12}\cdot \max_{0\leqslant x\leqslant 1}|y^{(n)}(x)|.$$

Этот оператор обозначают обычно $\Lambda(y_i)=\left(y_{i+1}+2y_i+y_{i-1}\right)/h^2$, он имеет важную роль, граничные условия $y_0 = = y_N = 0$, обладает важным свойством, оператор $-\Lambda$ даёт

$$-\Lambda(y_i)=\begin{pmatrix} 2 & -1 & & 0 & \\ -1 & 2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ 0 & & & -1 & 2 \end{pmatrix}\cdot\frac{1}{h^2}\vec{y_i},$$

кроме того он имеет собственные функции и значения, здесь они определяются таким образом

$$\lambda(-\Lambda)=\frac{4}{h^2}\cdot\sin^2\left(\frac{m\pi}{2h}\right), \quad m=1,\,2,\,\ldots,\,N-1,$$

матрица порядка $(N-1\times N-1)$. Кроме того собственные функции $y_i^{(m)}=\sqrt{2}\cdot\sin\left(m\pi hi\right)$, то есть идея заключается в том, что мы заменяем оператор на дискретный и получим задачу $-\Lambda y(x_i)+c(x_i)\cdot y(x_i)=f(x_i)$, но здесь $-\Lambda$ заменяем с точностью $O(h^2)$, тогда точное уравнение

$$f(x_i)=-y''+cy'=f(x)\Big|_{x=x_i}, \quad -\Lambda\tilde{y}_i+c_i\tilde{y}_i=f_i$$

— уравнение, которое является аппроксимацией исходного, с точностью $O(h^2)$ и $\tilde{y}_i=y_i+z_i$.

Приводя подобные, могу написать в таком виде

$$-\tilde{y}_{i-1}+(2+h_i^2c_i)\cdot\tilde{y}_i-\tilde{y}_{i+1}=h^2f_i=\varphi_i, \quad i=1,\,2,\,\ldots,\,N-1,$$

уравнений $(N-1)$, а неизвестных $(N+1)$, но есть граничные условия $y_0=y_N=0$, получаем дискретную задачу — линейное уравнение относительно $\tilde{y}_i$, это разностные уравнения второго порядка, мы изучали метод Гаусса, а потом модифицированный для трёх-диагональных матриц, эта система решается методом прогонки, можем всё успешно решить, если $c(x)>0$, тогда у нас диагональное преобладание, все коэффициенты ограничены для любого номера N. Если $c(x)=0$, то есть задача простая, здесь целесообразно использовать не метод прогонки, так как будет на грани устойчивости, слабо устойчивой, используйте *метод дискретного преобразования Фурье* (ещё называют *метод быстрого преобразования Фурье*).

Идея в чём, мы имеем задачу $-\Lambda(\tilde{y}_i)=f_i,\ i=1,\,2,\,\ldots,\,N-1$ и $y_0=y_N=0$, нет диагонального преобладания,

$$y_i=\sum_{m=1}^{n-1}b_m\cdot y_i^{(m)},\ \text{а}\ y_i^{(m)}=\sqrt{2}\cdot\sin\left(m\pi hi\right),$$

я уже писал, и учитываем собственные функции $y_i^{(m)}$, это означает, что $-\Lambda(y_i^{(m)})=\lambda^{(m)}\cdot y_i^{(m)}$, если подставить в исходное уравнение, функция f_i представляется в виде ряда

$$\sum_{m=1}^{N-1}a_m\cdot y_i^{(m)},$$

коэффициенты вычисляются как разложение в ряд Фурье, вы изучали в курсе математического анализа

$$a_m=\sum_{j=1}^{N-1}f_j\cdot y_j^{(m)}\cdot h$$

и тогда подставляем

$$f_i=\sum_{m=1}^{N-1}a_m\cdot y_i^{(m)}=-\sum_{m=1}^{N-1}b_m\cdot(-\Lambda y_i^{(m)})=\sum_{m=1}^{N-1}b_m\lambda^{(m)}\cdot y_i^{(m)},$$

но коэффициенты b_m неизвестные, поэтому получаем уравнение

$$\sum_{m=1}^{N-1}\left(b_m\lambda^{(m)}-a_m\right)\cdot y_i^{(m)}=0,$$

тогда скобка $(b_m\lambda^{(m)} - a_m) = 0$, следовательно $b_m = a_m/\lambda^{(m)}$.

Это очень надёжный метод, он устойчивый, но применяется не только для этой задачи. Конечно, использует большее число шагов, чем метод прогонки. А программисты научились искать быстро эти суммы отдельным методом. Там $N = 2^M = 16,\ 32,\ 64,\ \ldots$. А тогда здесь получается синусы кратных углов, их можно вычислять через угол одинарных углов, мне достаточно один раз обратиться к вычислению функции синуса и косинуса, на этом очень много экономится и метод начинает быть конкурентно-способным с другими методами, правда требует знания математики, какие-то умственные усилия придётся прилагать. Зато очень надёжный. 25.03.05

Глава 7

Численное дифференцирование

Предположим у нас заданы значения функции на узлах 0, h, $2h$, $\ldots$, jh, введём для них обозначения $x_0 = 0$, $x_1 = h$, $\ldots$, $x_j = jh$ и для них заданы значения функции $y(x)$: $y_j(x) = y(x_j) = y(jh)$.

7.1 Аппроксимация первой производной

Начнём с простых примеров, как мы можем аппроксимировать (приблизить) производную:

$$y' = \lim_{h\to 0} \frac{y(x+h) - y(x)}{h},$$

а мы приближённо заменяем

$$y' \approx \frac{y(x+h) - y(x)}{h}$$

и мы получаем формулу $(y_{j+1} - y_j)/h$. Всё зависит от того, в каком узле мы будем вычислять, если мы возьмём произвольную точку x и хотим посчитать с какой точностью аппроксимируем производную в этой точке, то нам надо дополнительный x в узле x_j, тогда введём в обозначение *разностный оператор*

$$\Lambda_h[y] = \frac{y(x+h) - y(x)}{h},$$

дифференциальный оператор $F[y] = y'$ и дадим определение:

ОПР 7.2 (Порядок аппроксимации)**.**
Разностный оператор $\Lambda_h[y]$ с k-ым порядком аппроксимирует дифференциальный оператор $F[y]$, если для всех достаточно гладких функций $y(x)$ (означает, что когда мы хотим, чтобы функция имела столько производных, сколько нужно, но не оговариваем пока сколько), следовательно $\Lambda_h[y] = F[y] + O(h^k)$, это определение общее.

Преобразуем разностный оператор рядом Тейлора по h при $h = 0$:

$$y(x+h) = y(x) + h\cdot y'(x) + \frac{h^2}{2}\cdot y''(x) + \ldots + \frac{h^m}{m!}\cdot y^{(m)}(x) + O(h^m)$$

(далее я буду аргумент x опускать), далее мы можем преобразовать наш оператор, здесь нам понадобится всего три члена:

$$\Lambda_h[y] = (y + hy' + \frac{h^2}{2}\cdot y'' + O(h^3) - y)/h = y' + \frac{h}{2}\cdot y'' + O(h^2) = F[y] + O(h).$$

Если мы изменим точку аппроксимации, то порядок может измениться. Здесь мы рассматриваем простейший оператор, зададим на отрезке $[j,\, j+1]$ — двух точках (называется *двухточечный* или говорят по двухточечному шаблону) и мы в качестве точки приближения взяли точку на границе, пусть теперь она будет в центре между x_j и x_{j+1}, тогда

$$x_j = x - \frac{h}{2},\ \text{а}\ x_{j+1} = x + \frac{h}{2}$$

и если мы хотим посмотреть с какой точностью он приближает в ней, то это будет уже другой оператор, запишем его:

$$\Lambda_h[y] = \left[y\left(x + \frac{h}{2}\right) - y\left(x - \frac{h}{2}\right)\right]/h$$

и, предположим, функция достаточна гладкая, воспользуемся разложением в ряд Тейлора по h при $h = 0$:

$$y\left(x \pm \frac{h}{2}\right) = y \pm \frac{h}{2} \cdot y' + \frac{h^2}{4 \cdot 2!} \cdot y'' \pm \frac{h^3}{2^3 \cdot 3!} \cdot y''' + O(h^4),$$

где y''' нужно для выяснения невязки (ошибки), тогда

$$\Lambda_h[y] = \left(y + \frac{h}{2} \cdot y' + \frac{h^2}{8} \cdot y'' + \frac{h^3}{48} \cdot y''' - y + \frac{h}{2} \cdot y' - \frac{h^2}{48} \cdot y''' + O(h^5)\right)/h =$$

$$=_{\text{(сокращаем то, что сокращается)}} \left(y'h + \frac{h^3}{24} \cdot y''' + O(h^5)\right)/h = y' + \frac{h^2}{24} \cdot y''' + O(h^4)$$

и главный член невязки $h^2 y'''/24$ (здесь 24 бесплатно повышает точность).

Если мы посмотрим в любой другой точке x, расположенной не в центре, мы получим первый порядок аппроксимации.

7.3 Аппроксимация второй производной

Рассмотрим аппроксимацию второй производной: вспомним, что

$$y'' = (y')' = \lim_{h \to 0} \frac{y'(x) - y'(x-h)}{h},$$

так вот, чтобы построить простейший оператор, уберём предел, а

$$y' \approx \frac{y(x+h) - y(x)}{h},$$

подставляя это туда, получаем

$$y'' \approx \frac{1}{h} \cdot \left(\frac{y(x+h) - y(x)}{h} - \frac{y(x) - y(x-h)}{h}\right) = \frac{y(x+h) - 2y(x) + y(x-h)}{h^2}.$$

Если взять $x = x_j$, то мы получаем формулу

$$\Lambda_h[y] = \frac{y_{j+1} - 2y_j + y_{j-1}}{h}$$

— трёх точечный, зависит от трёх точек $j-1$, j и $j+1$, то есть трёхточечный метод (это уже почти интерполяция) на точках $[y-h,\, y,\, y+h]$:

$$\Lambda_h[y] = \frac{1}{h^2} \cdot \left[y + hy' + \frac{h^2}{2} \cdot y'' + \frac{h^3}{6} \cdot y''' + \frac{h^4}{24} \cdot y^{(4)} - 2y + y - hy' + \frac{h^2}{2} \cdot y' - \frac{h^3}{6} \cdot y^{(3)} + \frac{h^4}{24} \cdot y^{(4)} + O(h^6)\right] =$$

$$= y'' + \left(\frac{h^2}{12} \cdot y^{(4)} + O(h^4)\right)$$

здесь в скобках главный член ошибки или невязки аппроксимации и четыре чёрточки уже писать противно, поэтому $y^{(4)}$. Далее можно заметить, что коэффициенты в аппроксимации оператора есть

$$\begin{array}{ccccc} & 1 & & -1 & \\ 1 & & -2 & & 1 \end{array},$$

если сверху напишу нуль, то получится треугольник Паскаля, только со знаками, методом индукции это всё доказывается, так же как вывод бинома Ньютона и поэтому для третьей производной и так далее:

$$\begin{array}{ccccccccccc} & & & & 0 & & & & & & \\ & & & 1 & & -1 & & & & & \\ & & 1 & & -2 & & 1 & & & & \\ & 1 & & -3 & & 3 & & -1 & & & \\ 1 & & -4 & & 6 & & -4 & & 1 & & \\ & & & & \vdots & & & & & & \end{array}$$

и важно понять, что при аппроксимировании первой производной используется две точки и так для n-ой производной нужно использовать шаблон из $n+1$ узла.

$$\Lambda_3 = \frac{y_{j+1} - 3 \cdot y_{j+1} + 3 \cdot y_j - y_{j-1}}{h^4}.$$

Можно я дам домашнее задание для всем? Можно! Нужно доказать, что в тех точках даёт второй порядок аппроксимации и вычислить главные их невязки для этих операторов (третьей и четвёртой).

7.4 Разностные операторы повышенного порядка

7.4.1 Метод последовательного повышения порядка аппроксимации

Рассмотрим для y':

$$\Lambda_h[y] = \frac{y_{j+1} - y_j}{h}$$

на $[x_j,\, x,\, x_{j+1}]$, воспользуемся теми же предложениями, что использовали раньше. Мы получили, что

$$\left[y \cdot \left(x + \frac{h}{2}\right) - y \cdot \left(x - \frac{h}{2}\right)\right] /h = y' + \frac{h^2}{24} \cdot y''' + O(h^4),$$

так вот, наиболее элементарный способ получить четвёртый порядок заключается в том, чтобы из него вычесть оператор Λ^*, аппроксимирующий третью производную на соответствующем шаблоне $[x_{j-1},\ x_j,\, x,\, x_{j+1},\, x_{j+2}]$ аппроксимации и записать оператор Λ^* на нём; соответствующий оператор имеет вид

$$\Lambda^* = \frac{h^2}{24} \cdot \frac{y_{i+2} - 3 \cdot y_{j-1} + 3 \cdot y_j - y_{j-1}}{h^3};$$

тогда

$$\tilde{\Lambda}[y] = \Lambda - \Lambda^* =_{\text{(в результате разложения в ряд Тейлора)}} y' + O(h^4)$$

и сумма невязки Λ и Λ' даст невязку оператора $\tilde{\Lambda}$, для этого надо разложить в ряд Тейлора их. Итого

$$\tilde{\Lambda} = \frac{y_{j+1} - y_j}{h} - \frac{1}{24 \cdot h} \cdot (y_{j+2} - 3 \cdot y_{j+1} - 3 \cdot y_j - y_{j-1}).$$

Аналогичным образом сделаем оператор, который с четвёртого порядка будет аппроксимировать вторую производную. Для этого возьмём

$$\Lambda = \frac{y_{j+1} - 2 \cdot y_j + y_{j-1}}{h^2},$$

вспомним главный член невязки в разложении этого оператора:

$$\Lambda[y] = y^4 + \frac{h^2}{12} \cdot y^{(4)} + O(h^4),$$

поэтому аналогично как для первой производной, из этого оператора можно вычесть $h^2/12$, умноженное на разностный оператор для четвёртой производной, при этом шаблон будет пятиточечный: $[x_{j-2},\, x_{j-1},\, x_j,\, x_{j+1},\, x_{j+2}]$,

$$\Lambda = y^4 + \frac{h^2}{12} \cdot y^{(4)} + O(h^4) - \frac{h^2}{12} \cdot (y_{j+2} - 4 \cdot y_{j-1} + 6 \cdot y_j - 4 \cdot y_{j-1} + y_{j-2})/h^4 = \dots$$

Так можно построить и дальше, этот оператор имеет при разложении вид

$$\Lambda = y'' + \frac{h^4}{m} \cdot y^{(6)} + O(h^6).$$

Чтобы получить порядок шесть аппроксимации, то понятно как действовать, надо вычесть

$$\tilde{\Lambda} = \frac{h^4}{m} \cdot [y^{(6)}]$$

— аппроксимация шестой производной на соответствующих семи точках шаблона.

7.4.2 Общий метод построения оператора k-го порядка для n-ой производной

Предположим у нас есть шаблон $[0,\, h,\, \dots,\, jh]$, мы хотим получить n-ую производную с k-ым порядком, запишем

$$\Lambda[yj] = \frac{1}{h^n} \cdot \sum_j a_j \cdot y(x + jh),$$

где x — точка разложения, а j — зависит от точки разложения, рассмотрим самый общий случай их нахождения где угодно, пусть расстояние от jh до x относится к расстоянию от x до $(j + a)h$ как α к $(1 - \alpha)$, тогда шаблон $[jh,\, x,\, (j + 1)h]$, тогда предположим я аппроксимирую первую производную $(y_{j+1} - y_j)/h$, а хочу в точке x, тогда

$$\frac{y \cdot (x + (1 - \alpha)h) - y \cdot (x - \alpha h)}{h}$$

и j принимает значения $1-\alpha$ и $-\alpha$. Таким образом j может принимать и не целые значения и только если мы будем рассматривать в узлах сетки, будет целой. Здесь мы в точке x можем получить большой порядок, а в других точках только первого.

Разложим в ряд Тейлора:

$$\Lambda[y_i] = \frac{1}{h^n} \cdot \sum_j \left(a_j \cdot \sum_{m=0}^{?} \frac{(jh)^m}{m!} \cdot y^{(m)}(x) \right) = \frac{1}{h^n} \cdot \sum_{m=0}^{?} \left(\frac{h^m}{m!} \cdot y^{(m)} \cdot \sum_j a_j \cdot j^m \right).$$

Чтобы это аппроксимировать, все эти слагаемые при $n < m$ должны равняться нулю, таким образом условия аппроксимации:

1. $\sum_j a_j \cdot j^m = 0 - \forall m = 0,\ 1,\ \ldots,\ n-1$, иначе получим нечто бесконечное при $n \to 0$.
2. При $m = n$ получим
$$\frac{y^{(n)}}{n!} \cdot \sum_j a_j \cdot j^n = y^{(n)}$$
из которого условие
$$\sum_j a_j \cdot j^n = n!,$$
при этом аппроксимацию мы имеем, но порядок любой.
3. Чтобы повысить степень, должны добавить
$$\frac{y^{(n)}}{n!} \cdot \sum_j a_j \cdot j^n = 0$$
при $m = n+1,\ n+2,\ \ldots,\ n+k-1$, где k — порядок аппроксимации, но это не ниже k-го, чтобы точно:
$$\frac{y^{(n)}}{n!} \cdot \sum_j a_j \cdot j^n = y^{(n)} + \frac{h^k}{(n+k)!} \cdot y^{(n+k)} \cdot \sum_j a_j \cdot j^{n+k} \neq 0,$$
то есть
$$\sum_j a_j \cdot j^{n+k} \neq 0.$$

Здесь мы получили условия на коэффициенты и чтобы было разрешимо, необходимо, чтобы ещё один коэффициент a_j появился, то есть на одну увеличим шаблон.

Пример 7.4.2.1 (Применения)**.**

▷ Найдём y' и на первый порядок увеличим на границе шаблона, иногда очень нужно. Шаблон $[x_j,\ x_{j+1},\ x_{j+2}]$, можно методом последовательного приближения, но я хочу получить сразу через формулы:

$$\Lambda = \frac{1}{h} \cdot (a_0 \cdot y_j + a_1 \cdot y_{j+1} + a_2 \cdot y_{j+2}),$$

можно переписать как

$$\Lambda[y(x)] = \frac{1}{h} \cdot (a_0 \cdot y(x) + a_1 \cdot y(x+h) + a_2 \cdot y(x+2h))$$

и чётко видно, что индукция $j = 0,\ 1,\ 2$. Учитывая это начинаем выполнять формулы:

1. $a_0 + a_1 + a_2 = 0$, обратим внимание, что так как $j^0 = 1$ и не ясно, то берём $0^0 \equiv 1$ и $0^m \neq 1$, для $m > 0$, тогда $a_0 = -a_1 - a_2$.
2. $a_1 + 2 \cdot a_2 = 1!$.
3. $a_1 \cdot 1^2 + a_2 \cdot 2^2 = 0$.

Вычтем 3 от 2 и получим $2a_2 = -1 \Rightarrow a_2 = -1/2$, $a_1 = 2$, $a_0 = -1.5$.

Литература

[1] Ахмеров Р.Р., Коробицина Ж.А., Слепцов А.Г. „Основы численного анализа в задачах“, издание НГУ 94 г.

[2] Барахнини В.Б., Шапеев В.П. „Введение в численный анализ“, издание НГУ 97г.

[3] Годунов С.К., Рябенький В.С. „Разностные схемы“, издательство „Наука“, 77г.

[4] Самарский А.А., Гулин А.В. „Численные методы“, издательство „Наука“, 89г.

[5] Учебник Самарского А.А. „Введение в численные методы“, издательство „Наука“, 97г.

[6] Турчак Л.Н. „Основы численных методов“, 82г.

[7] Кто-то там читавший нам лекции.