1. Представление чисел в оперативной памяти и связанные с этим погрешности

x=132.879 - Форма с фиксированной точкой

$=13.2879\cdot10^{1}=13287.9\cdot10^{-2}$ Форма с плавающей точкой

$=0.132879\cdot10^{3}$ Нормализованная форма с плавающей точкой (слева мантисса, справа основание системы счистления β, степень=порядок n)

Для фиксир:

ти	**по**		**ба**	**йт**					
±	**0**	**.**	**0**	**0**	**0**	**0**	**1**	**$=10^{-5}$**	**DD=12**
±	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**$=10^{7}$**	

Для плав:

						поря	док		
±	**0**	**0**	**0**	**0**	**1**	**-**	**9**	**$=0.00001\cdot10^{-9}=10^{-14}$**	**DD=23**
±	**9**	**9**	**9**	**9**	**9**	**+**	**9**	**$=0.99999\cdot10^{9}=10^{9}$**	

Пример:

$$\beta=2 \quad k=3 \quad L=-1 \quad U=2$$

±	**1**	**1**	**1**	**+**	**1**	**0**
±				**±**		
±	**0**	**0**	**1**	**-**	**0**	**1**

$$x_{\min}=(\frac{0}{2}+\frac{0}{2^2}+\frac{1}{2^3})\cdot2^{-1}=\frac{1}{16}$$

$$x_{\max}=(\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3})\cdot2^{2}=3.5$$

Кол-во чисел, представленных в разрядах сетки

$$N=2\cdot(\beta-1)\cdot\beta^{k-1}\cdot(U-L+1)+1=$$
$$=(2\cdot1\cdot2^{2}\cdot4+1)=33$$

Абсолютная точность = половине минимального числа

$$\varepsilon = \frac{1}{16} \cdot \frac{1}{2} = \frac{1}{32}$$

Основным и неустранимым источником компьютерных погрешностей является ограничение разрядной сетки машины.

x_0 - точное значение

$\Delta x = x - x_0$ – абсолютная погрешность

$x = x_0 \pm \Delta x$

$\delta_x = \Delta x / x_0$ – относительная погрешность

Алгебра погрешностей

1. $\Delta(x_1 \pm x_2) = \Delta x_1 + \Delta x_2$
2. $\delta(x_1 \cdot x_2) = \delta x_1 + \delta x_2$
   $\delta(x_1 / x_2) = \delta x_1 + \delta x_2$
3. $\delta(x^n) = n \cdot \delta x$
4. $x = x_0 \pm \Delta x \quad y = f(x) \quad \Delta y \approx | f'(x_0) | \cdot \Delta x$

2. Обусловленность задач. Устойчивость и сходимость алгоритмов.

Обусловленность задачи - чувствительность ее решения к малым изменениям входных данных.

В численных методах, число обусловленности характеризует точность решения задачи и является мерой того, насколько задача хорошо или плохо обусловлена.

Если число обусловленности некоего уравнения мало, то уравнение называется хорошо обусловленным. Если же число обусловленности велико, то уравнение называется плохо обусловленным.

Как правило, анализ устойчивости алгоритма решения ОДУ сводится к оценке эволюции погрешности при реализации шага. Предположим, что на i-том шаге решения дифференциального уравнения погрешность составляла e^i. Эта погрешность была внесена различными ошибками округления и численных методов. Если при переходе к (i+1)-му шагу эта погрешность существенно возрастает, то можно говорить о неустойчивости метода. И напротив, если погрешность растёт слабо, то алгоритм устойчив.

Задача называется в*ычислительно неустойчивой*, если малые изменения входных данных приводят к заметным изменениям решения.

Вычислительную устойчивость, например, решения системы уравнения, можно определить следующим образом: допустим мы решили систему уравнения относительно $x_1,...,x_n$, то есть нашли решение $P(x_1,...,x_n)$. Если мы чуть-чуть поменяем значения на $x_1',...,x_n'$, то новое решение $P'(x_1',...,x_n')$ будет в каком-то смысле близким к решению $P(x_1,...,x_n)$.

Сходимость алгоритма - способность итерационного алгоритма достигать оптимума целевой функции, или подходить достаточно близко к нему, за конечное число шагов. Скорость сходимости алгоритмов — один из важнейших показателей качества аналитических моделей. Часто, когда говорят о том, что один алгоритм является более быстрым, чем другой, имеют ввиду именно скорость сходимости.

3. Прямые методы для решения систем линейных уравнений. Метод Гаусса, Краута-Холецки, метод прогонки.

Прямые (или точные) методы, позволяют найти решение за определенное количество шагов. Итерационные методы, основаны на использовании повторяющегося процесса и позволяют получить решение в результате последовательных приближений

Метод Гаусса — классический метод решения системы линейных алгебраических уравнений (СЛАУ). Состоит в постепенном понижении порядка системы и исключении неизвестных.

Метод Гаусса с выбором ведущего элемента

Этот метод наиболее быстрый и минимальный по количеству операций из всех прямых методов. Он разделяется на два этапа: прямой ход и обратный ход Гаусса.

Прямой ход:

Шаг 1*.* Среди элементов матрицы *A* первого столбца выбирается элемент $a_{i,1}$ ***(i= 1,2,…,n)****,* наибольший по модулю (ведущий элемент). Строка

(уравнение) с этим элементом переписывается на первое место.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1, \\ a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{nn}x_n = b_n, \end{cases} \qquad A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ 0 & a_{22} & \ldots & a_{2n} \\ 0 & 0 & \ldots & \ldots \\ 0 & 0 & 0 & a_{nn} \end{bmatrix}.$$

Шаг 2*.* Домножением первого уравнения на коэффициент $c = -a_{j,1} / a_{i,1}$ и сложением его после этого с ***i***-м уравнением, из всех нижних уравнений системы последовательно исключаются (становятся равными ***0****)* коэффициенты при x_1 , далее рассматривается столбец коэффициентов при x_2 и *шаги 1* и *2* повторяются и т.д., до тех пор, пока матрица системы не примет вид верхней треугольной матрицы.

Обратный ход*:* Из самого нижнего уравнения преобразованной системы, где остался ненулевым только коэффициент при x_n , находится $x_n = b_n / a_{nn}$. Найденное решение подставляется в предпоследнее уравнение, откуда находится x_{n-1} и т.д., снизу вверх определяются все неизвестные $x_{n-2}, x_{n-3}, \ldots, x_2, x_1$.

Алгоритм Краута - это фактически метод решения систем общего вида, конкурирующий по быстродействию с общеизвестным методом Гаусса-Жордана, но позволяющий более эффективно использовать решение.

Метод Краута-Холецкий

Ax=B; $A=\begin{pmatrix} a_{11} & \dots & a_{1n} \\ .. & .. & .. \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$ Метод основан на LU разложении

(1) Любую невырожденную квадратную матрицу можно представить в виде двух квадратных A=C*D;

$C=\begin{pmatrix} C_{11} & 0 & 0 \\ \dots & \dots & 0 \\ C_{n1} & \dots & C_{nn} \end{pmatrix}$ $D=\begin{pmatrix} 1 & .. & d_{1n} \\ 0 & 1 & d_{n-1,n} \\ 0 & 0 & 1 \end{pmatrix}$

Алгоритм LU разложения:

1) $c_{i1}=a_{i1}$

2) Все элементы первой строки D – это первая строка A деленная на C_{11}(или a_{11}): $d_{1j}=a_{1j}/c_{11}$

3) $C_{ij}=a_{ij}-\sum_{k=1}^{j-1} C_{ik} * d_{kj}$

4) $d_{ij} = (a_{ij} - \sum_{k=1}^{i-1} C_{ik} * d_{kj}) / C_{ii}$

3),4) – чередуются

(2) Ax=B -> CDx=B

$\begin{cases} Dx = Y \\ CY = B \end{cases}$ => (из 2 уравнения) $\begin{cases} c_{11}y_1 = b_1 \\ c_{21}y_1 + c_{22}y_2 = b_2 \\ \dots \end{cases}$,

для решения используем обратный ход Гаусса (сверху вниз).

(3) Решение Dx=Y

$\begin{cases} x_1 + d_{12}x_2 + \dots + d_{1n}x_n = y_1 \\ \dots \\ x_{n-1} + d_{n-1,n}x_n = y_{n-1} \\ x_n = y_n \end{cases}$ для решения используем обратный ход Гаусса.

Метод прогонки

Для решения систем *A x = b* с трехдиагональной матрицей наиболее часто применяется *метод прогонки*, являющийся адаптацией метода Гаусса к этому случаю.

Запишем систему уравнений

$$d_1x_1 + e_1x_2 \qquad\qquad = b_1$$

$$
\begin{aligned}
c_2x_1 + d_2x_2 + e_2x_3 &= b_2 \\
c_3x_2 + d_3x_3 + e_3x_4 &= b_3 \\
\dots \quad \dots \quad \dots & \\
c_{n-1}x_{n-2} + d_{n-1}x_{n-1} + e_{n-1}x_n &= b_{n-1} \\
c_nx_{n-1} + d_nx_n &= b_n
\end{aligned}
$$

в матричном виде: $A\,x = b$, где

$$
A = \begin{bmatrix}
d_1 & e_1 & 0 & 0 & \cdots & 0 \\
c_2 & d & e_2 & 0 & \cdots & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & 0 & c_{n-1} & d_{n-1} & e_{n-1} \\
0 & \cdots & 0 & 0 & c_n & d_n
\end{bmatrix}
$$

Выпишем формулы метода прогонки в порядке их применения.

1. Прямой ход метода прогонки (вычисление вспомогательных величин):

$$
\begin{aligned}
&\alpha_2 = -e_1 / d_1 \\
&\beta_2 = b_1 / d_1 \\
&\alpha_{i+1} = -e_i / [d_i + c_i\alpha_i],\ i=2, \dots, n-1 \\
&\beta_{i+1} = [-c_i\beta_i + b_i] / [d_i + c_i\alpha_i],\ i=2, \dots, n-1
\end{aligned}
\qquad \textbf{(1.9)}
$$

2. Обратный ход метода прогонки (нахождение решения):

$$
\begin{aligned}
&x_n = [-c_n\,\beta_n + b_n] / [d_n + c_n\alpha_n] \\
&x_i = \alpha_{i+1}\,x_{i+1} + \beta_{i+1},\ i = n-1, \dots, 1
\end{aligned}
\qquad \textbf{(1.10)}
$$

Метод прогонки можно применять, если нигде в формулах знаменатели не равны нулю. Доказано следующее

УТВЕРЖДЕНИЕ 1.1 Для применимости формул метода прогонки достаточно выполнения условий диагонального преобладания у матрицы A, то есть

$$| d_i | \geq | c_i | + | e_i |$$

причем хотя бы одно неравенство должно быть строгим.

4. Обусловленность матрицы и точность решения системы. Нормы вектора и матрицы. Способы оценки числа обусловленности матрицы.

Величина $\mu(A) = ||A|| \cdot ||A^{-1}||$ называется числом обусловленности матрицы A. Число обусловленности определяет, насколько погрешность входных данных может повлиять на решение системы; всегда $\mu \geq 1$. Хотя число обусловленности матрицы зависит от выбора нормы, если матрица хорошо обусловлена, то её число обусловленности будет мало при любом выборе нормы, а если она плохо обусловлена, то её число обусловленности будет велико при любом выборе нормы. Таким образом, обычно норму выбирают исходя из соображений удобства.

Обусловленность обозначается как cond(A), где A - матрица, и находится через норму. Норм бывает две штуки:

- Евклидова норма: $||x|| = \sqrt{x_1{}^2 + \ ... + x_n{}^2}$
- Манхеттенская норма: ||x|| = |x1| + ... + |xn|

Норма матрицы = ||A|| = MAX{a_j} - максимальная норма столбца

Теоретически число обусловленности можно найти как Cond(A) =||A|| * $||A^{-1}||$. В силу сложности нахождения обратной матрицы используют другой способ нахождения числа обусловленности: пусть имеется СЛАУ A*X=B. найдём решение для этого СЛАУ. Теперь изменим MAX{B_i} на 5%, и решим новую СЛАУ заново. Когда изменили B, изменился и X (решение). формула нахождения числа обусловленностей выглядит так:

$$\frac{||deltaX||}{||X||} \leq Cond(A) * \frac{||deltaB||}{||B||}$$

Если Det(A)=0, у СЛАУ бесконечно много решений. Cond(A) может принимать значения от 1 до бесконечности, но:

- от 1 до 10 - матрица хорошо обусловлена
- от 10 до 1000 - "серая зона" (}{3 что там получится)
- от 1000 - потрясающе плохо обусловленная матрица.

5. Итерационные методы решения систем линейных уравнений.

Метод Якоби(метод простых итераций)

Для того, чтобы построить итеративную процедуру метода Якоби, необходимо провести предварительное преобразование системы уравнений $A\vec{x} = \vec{b}$ к итерационному виду $\vec{x} = B\vec{x} + \vec{g}$. Оно может быть осуществлено по одному из следующих правил:

- $B = E - D^{-1}A = D^{-1}(D - A), \quad \vec{g} = D^{-1}\vec{b};$
- $B = -D^{-1}(L + U) = -D^{-1}(A - D), \quad \vec{g} = D^{-1}\vec{b}$
- $D_{ii}^{-1} = 1/D_{ii}, D_{ii}! = 0, i = 1, 2, ..., n$

где в принятых обозначениях D означает матрицу, у которой на главной диагонали стоят соответствующие элементы матрицы A, а все остальные нули; тогда как матрицы U и L содержат верхнюю и нижнюю треугольные части A, на главной диагонали которых единицы(см. LU-разложение), E — единичная матрица.

Тогда процедура нахождения решения имеет вид:

$$\vec{x}^{(k+1)} = B\vec{x}^{(k)} + \vec{g},$$

где k счётчик итерации.

В отличие от метода Гаусса-Зейделя мы не можем заменять $x_i^{(k)}$ на $x_i^{(k+1)}$ в процессе итерационной процедуры, т.к. эти значения понадобятся для остальных вычислений. Это наиболее значимое различие между методом Якоби и методом Гаусса-Зейделя решения СЛАУ. Таким образом на каждой итерации придётся хранить оба вектора приближений: старый и новый.

Условие окончания итерационного процесса при достижении точности ε в упрощённой форме имеет вид:

$$\| x^{(k+1)} - x^{(k)} \| \leq \varepsilon$$

Существует более точное условие окончания итерационного процесса, которое более сложно и требует дополнительных вычислений.

Метод Гаусса-Зейделя - ускоренный метод простых итераций.
X хранится не в 2-х массивах(старом и новом, как в Якоби), а в одном. Следовательно, когда ищем следующий x_i , то подставляем x_{i-1} который был расчитан на предыдущем шаге.

6. Интерполяция и аппроксимация. Интерполяция по Лагранжу и Ньютону.

Интерполя́ция — в вычислительной математике способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений. Как правило, на основании наборов, полученных экспериментальным путём или методом случайной выборки, требуется построить функцию, на которую могли бы с высокой точностью попадать другие получаемые значения. Такая задача называется **аппроксимацией** кривой. **Интерполяцией** называют такую разновидность аппроксимации, при которой кривая построенной функции проходит точно через имеющиеся точки данных.

Пусть на отрезке $a \leqslant x \leqslant b$ задана сетка $\bar{\omega} = \{x_0 = a < x_1 < \ldots < x_n = b\}$ и в ее узлах заданы значения функции $y(x)$, равные $y(x_0) = y_0, \ldots, y(x_i) = y_i, \ldots, y(x_n) = y_n$. Требуется построить *интерполянту* — функцию $f(x)$, совпадающую с функцией $y(x)$ в узлах сетки:

$$f(x_i) = y_i, \quad i = 0, 1, \ldots, n. \qquad (1)$$

Основная цель интерполяции — получить быстрый (экономичный) алгоритм вычисления значений $f(x)$ для значений x, не содержащихся в таблице данных.

Существует также близкая к интерполяции задача, которая заключается в аппроксимации какой-либо сложной функции другой, более простой функцией. Если некоторая функция слишком сложна для производительных вычислений, можно попытаться вычислить её значение в нескольких точках, а по ним построить, то есть интерполировать, более простую функцию. Разумеется, использование упрощенной функции не позволяет получить такие же точные результаты, какие давала бы первоначальная функция. Но в некоторых классах задач достигнутый выигрыш в простоте и скорости вычислений может перевесить получаемую погрешность в результатах.

по Лагранжу

$L_n(x)=y_0 \cdot l_0(x)+y_1 \cdot l_1(x)+\ldots+y_n \cdot l_n(x)$

$$l_i(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

$$P_n(x) = \sum_{k=0}^{n} l_k(x)\, y_k = \sum_{k=0}^{n} y_k \prod_{i \neq k} \frac{x - x_i}{x_k - x_i}$$

по Ньютону

x	x_0	...	x_n
y	y_0	...	y_n

$N_n(x)=a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots$

Коэффициент **a** расчитывается как разделённые разности:

$$a_0 = \Delta^{(0)} y_0 = y_0$$

$$a_1 = \Delta^{(1)} y_0 = \frac{y_1 - y_0}{x_1 - x_0}$$

$$a_2 = \Delta^{(2)} y_0 = \frac{\Delta^{(1)} y_1 - \Delta^{(1)} y_0}{x_2 - x_0}$$

$$\vdots$$

$$a_n = \Delta^{n2)} y_0 = \frac{\Delta^{(n-1)} y_1 - \Delta^{(n-1)} y_0}{x_n - x_0}$$

разделенная разность первого порядка: $y(x_i, x_j) = [y(x_i) - y(x_j)]/(x_i - x_j)$;

разделенная разность второго порядка: $y(x_i, x_j, x_k) = [y(x_i, x_j) - y(x_j, x_k)]/(x_i - x_k)$ и т. д. Если $y(x) = P_n(x)$ — полином степени n, то для него первая разделенная разность $P(x, x_0) = [P(x) - P(x_0)]/(x - x_0)$ есть полином степени $n-1$, вторая разность $P(x, x_0, x_1)$ — полином степени $n-2$ и т. д., так что $(n+1)$-я разделенная разность равна нулю.

- реализуема для кол-ва точек в пределе 10
- не устойчив к выбросам

7. Сплайн-интерполяция.

Для проведения гладких кривых через узловые значения функции чертежники испльзуют упругую муталлическую линейку, совмещая её с узловыми точками. Математическая теория подобной аппроксимации называется теорией сплайн-функций. Один из наиболее распротранённых методов является интерполяция кубическими сплайнами. Используя законы упругости, можно установить, что недеформируемая линейка между соседними узлами проходит по линии, удовлетворяющей уравнению

$Фи^4(x) = 0$

Если в качестве функции Фи выбрать полином, то в соотвествии с выше преведенным уравнением степень полинома должна быть не выше третьей. Такой полином называют кубическим сплайном, который записывают в виде:

$$Фи_i(x) = a_i + b_i(x - x_i - 1) + c_i(x - x_i - 1)^2 + d_i(x - x_i - 1)^3$$

где a_i, b_i, c_i, d_i - коэффициенты сплайна, определяемые из дополнительных условий.

В отличии от полиномиальной интерполяции, когда вся аппроксимируемая зависимость описывается одним полиномом, при сплайновой интерполяции на каждом интервале строится отдельный полином третьей степени со своими коэффициентами. Коэффициенты сплайнов определяются из условий сшивания соседных сплайнов в узловых точках:

1) Равенство значений сплайнов $Фи_i(x)$ и аппроксимируемой функции f(x) в узлах - условия Лагранжа

$$Фи_i(x_{i-1}) = f_{i-1}, \quad Фи_i(x_i) = f_i;$$

2) Непрерывность первой и второй производных от сплайнов в узлах

$$Фи'_i(x_i) = Фи'_{i+1}(x_i),$$
$$Фи''_i(x_i) = Фи''_{i+1}(x_i).$$

Кроме перечисленных условий необходимо задать условия на концах, т.е. в точках x0 и xn. В общем случае эти условия зависят от конкретной задачи. Довольно часто используются условия свободных концов сплайнов. Если линейка не закреплена в точках вне интервала [x0, xn], то там она описывается уравнением прямой, т.е. полиномом первой степени. Следовательно, исходя из условий непрерывности вторых производных сплайнов на концах интервала, запишем

соотношения:

Фи"1(x_0) = 0
Фи"n(x_n) = 0

8. Аппроксимация по методу наименьших квадратов и ее виды.

Аппроксимация по методу наименьших квадратов
Алгебраическая аппроксимация (приближение) функции многочленом k-го порядка (k < (n + 1), где n + 1 – количество точек таблицы) имеет общий вид:

$$\varphi_k(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_k x^k,$$

где коэффициенты многочлена a_i, (i=0, 1,..., k) находятся из условия минимизации

функционала квадрата ошибки:

$$S(x, a_0, a_1, \ldots, a_k) = \sum_{i=0}^{n} (\varphi_k(x_i) - f(x_i))^2 = \sum_{i=0}^{n} (a_0 + a_1 x + a_2 x^2 + \ldots + a_k x^K - f(x_i))^2.$$

Требование равенства 0 частных производных функционала S(x,a_0 ,a_1 ,...,a_k)

$$\begin{cases} \dfrac{dS}{da_0} = 0, \\ \ldots\ldots\ldots\ldots \\ \dfrac{dS}{da_k} = 0, \end{cases}$$

дает систему линейных уравнений порядка (k+1):

$$\begin{cases} a_0(n+1) + a_1 \sum_{i=0}^{n} x_i + a_2 \sum_{i=0}^{n} x_i^2 + \ldots + a_k \sum_{i=0}^{n} x_i^k = \sum_{i=0}^{n} f(x_i), \\ a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 + a_2 \sum_{i=0}^{n} x_i^3 + \ldots + a_k \sum_{i=0}^{n} x_i^{k+1} = \sum_{i=0}^{n} x_i \cdot f(x_i), \\ a_0 \sum_{i=0}^{n} x_i^2 + a_1 \sum_{i=0}^{n} x_i^3 + a_2 \sum_{i=0}^{n} x_i^4 + \ldots + a_k \sum_{i=0}^{n} x_i^{k+2} = \sum_{i=0}^{n} x_i^2 \cdot f(x_i), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_0 \sum_{i=0}^{n} x_i^k + a_1 \sum_{i=0}^{n} x_i^{k+1} + a_2 \sum_{i=0}^{n} x_i^{k+2} + \ldots + a_k \sum_{i=0}^{n} x_i^{2k} = \sum_{i=0}^{n} x_i^k \cdot f(x_i), \end{cases}$$

решение которой, например методом исключения Гаусса, даст искомые коэффициенты аппроксимации по методу наименьших квадратов. Иными словами, найденный аппроксимирующий многочлен $\phi_k(x)$ будет из всех многочленов k-го порядка проходить наиболее близко к заданным значениям функции (табличным точкам). Отметим, что если k = n, то аппроксимирующий многочлен автоматически превращается в его частный случай – интерполирующий многочлен.

9. Аппроксимация в базисе взаимно ортогональных функций. Аппроксимация Фурье, Чебышева, Лежандра, Лаггера.

$$\int_a^b f_i(x) * f_j(x) dx = \begin{cases} S, i = j \\ 0, i \neq j \text{ (ортоганальные)} \end{cases}$$

Функции с взаимной ортогональностью: sin(x); cox(x); sin(n-1)x; cos(2x).

Если точек в таблице много, можно заменить знак интеграла на сумму.

Аппроксимация Фурье: функции обладают взаимной ортогональностью на интервале [-π; π]

Полином Чебышева: $T_0(x) = 1;\ T_1(x) = x; T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$

Полином Весселя: $U_0 = 1; U_1 = 2x; U_{k+1} = 2xU_k - U_{k-1}$

Полином Лежандра: $P_0 = 1; P_1 = x; P_{k+1} = \frac{(2k+1)xP_{x-k}P_{k-1}}{k+1}$

Полином Лаггера: $L_0=1$, $L_1=1-x$, $L_{k+1}=((2k+1-x)L_k - k\cdot L_{k-1})/(k+1)$

10. Аппроксимация Безье. Полиномы Бернштейна. Итерационный алгоритм.

Полином Бернштейна.

$$C(i,n) := \frac{n!}{i!\,(n-i)!}$$

$$Bn(i,n,t) := C(i,n)\,t^{i}(1-t)^{n-i}$$

Аппроксимация Безье.

1. Задаем количество точек, по которым будем аппроксимировать

$$x := \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad y := \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad n := 3$$

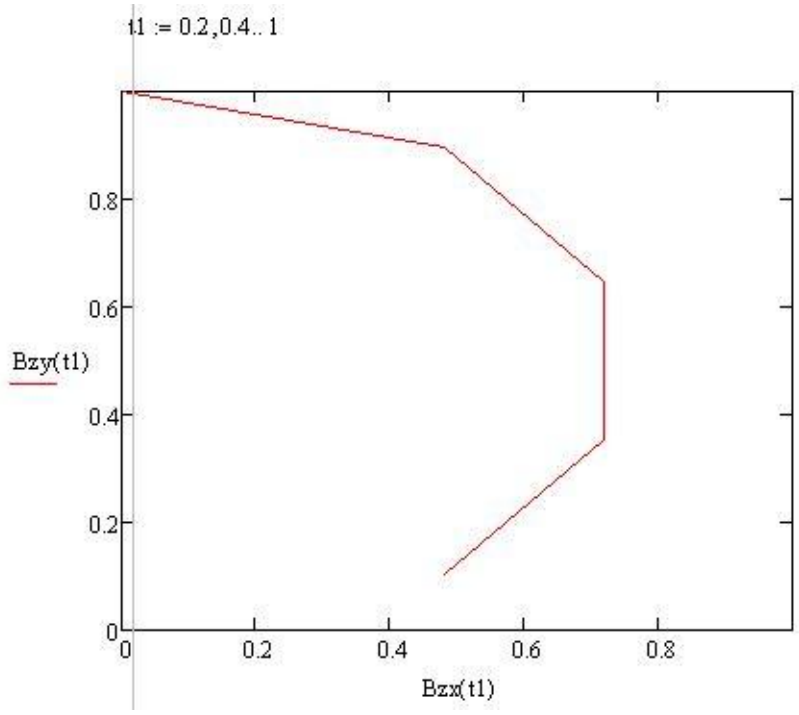
n = (колво. точек) - 1

2. Считаем значения точек по **X** и **Y** с помощью аппроксимации Безье

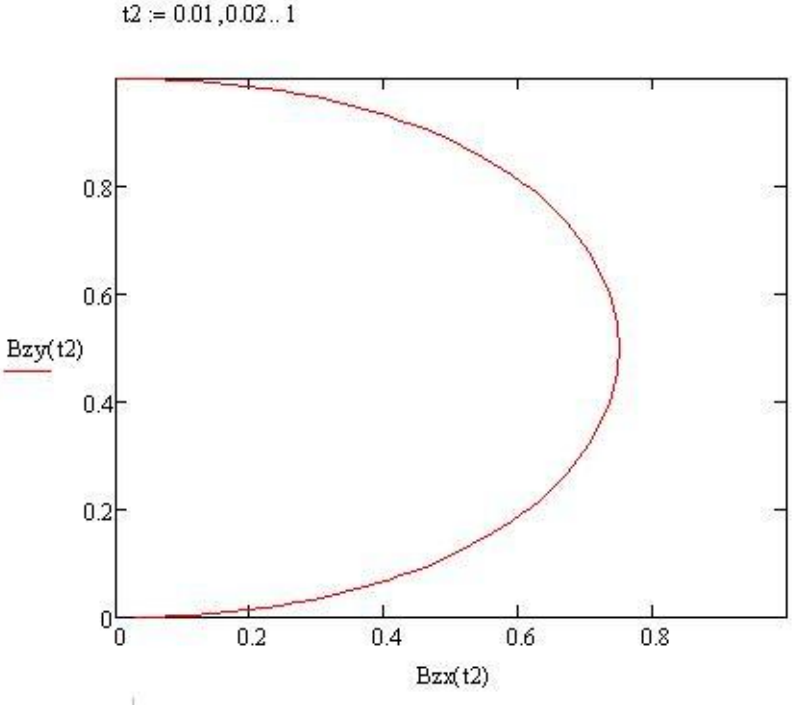
$$Bzx(tx) := \sum_{i=0}^{n} \left(x_{i,0}\,Bn(i,n,tx)\right)$$

$$Bzy(ty) := \sum_{i=0}^{n} \left(y_{i,0}\,Bn(i,n,ty)\right)$$

Результат с шагом 0.2



Результат с шагом 0.01



x,y - точки
t1,t2 - шаги

Метод является итерационным, так как количество итераций зависит от выбранного шага.

11. Численное дифференцирование. Конечные разности. Порядок и оценка точности разностной схемы.

1. $y' = \frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$ *алгебраическое*
2. $y'(x_0) = tg\alpha$ *геометрическое*
3. $y'(x) \to$ *мгновенная _ скорость* *физическое*

Конечные разности

$$y \approx \frac{\Delta y}{\Delta x}$$

1. $y'_k = \frac{y_{k+1} - y_k}{h} = y'(x_k) + O(h)$ *Правая _ разность*
2. $y'_k = \frac{y_k - y_{k-1}}{h} = y'(x_k) - O(h)$ *Левая _ разность*
3. $y'_k = \frac{y_{k+1} - y_{k-1}}{2h} = y'(x_k) + O(h^2)$ *Центральная _ разность*

O(h) - ошибка; у метода центральных разностей получается минимальной, по сравнению с двумя предыдущими методами.

При численном дифференцировании функции, заданной в виде таблицы с шагом h, эта погрешность зависит от h, и ее записывают в виде $R^{(k)} = O(h^r)$ (O большое от h^r) [1]. Показатель степени r называется *порядком* погрешности аппроксимации производной (или порядком точности данной аппроксимации). При этом предполагается, что значение шага по модулю меньше единицы.

Оценку погрешности легко проиллюстрировать с помощью ряда Тейлора

$$f(x+\Delta x) = f(x) + f'(x)\Delta x + \frac{f''(x)}{2!}\Delta x^2 + \frac{f'''(x)}{3!}\Delta x^3 + \dots$$

Пусть функция $f(x)$ задана в виде таблицы $f(x_i) = y_i$, $(i = 0, 1, \dots, n)$. Запишем ряд Тейлора при $x = x_1$, $\Delta x = -h$ с точностью до членов порядка h^2:

$$y_0 = y_1 - y_1' h + O(h^2).$$

Отсюда найдем значение производной в точке $x = x_1$:

$$y_1' = \frac{y_1 - y_0}{h} + O(h).$$

Это выражение совпадает с формулой (3.3), которая, как видно, является аппроксимацией первого порядка ($r = 1$). Аналогично, записывая ряд Тейлора при $\Delta x = h$, можно получить аппроксимацию (3.4). Она также имеет первый порядок.

Используем теперь ряд Тейлора для оценки погрешностей аппроксимаций (3.5) и (3.6). Полагая $\Delta x = h$ и $\Delta x = -h$ соответственно, получаем

$$\begin{aligned} y_2 &= y_1 + y_1' h + \frac{y_1''}{2!} h^2 + \frac{y_1'''}{3!} h^3 + O(h^4), \\ y_0 &= y_1 - y_1' h + \frac{y_1''}{2!} h^2 - \frac{y_1'''}{3!} h^3 + O(h^4). \end{aligned} \qquad (3.8)$$

Вычитая эти равенства одно из другого, после очевидных преобразований получаем

$$y_1' = \frac{y_2 - y_0}{2h} + O(h^2).$$

Это аппроксимация производной (3.5) с помощью центральных разностей. Она имеет второй порядок.

Складывая равенства (3.8), находим оценку погрешности аппроксимации производной второго порядка вида (3.6):

$$y_1'' = \frac{y_2 - 2y_1 + y_0}{h^2} + O(h^2).$$

12. Методы Ньютона-Котеса для численного интегрирования.

Численное интегрирование

Методы численного интегрирования (квадратурные формулы) используют геометрическую трактовку интеграла как площади, ограниченной графиком функции, осью абсцисс и пределами интегрирования. Площадь определяется как сумма площадей элементарных криволинейных трапеций с высотой h. Все методы отличаются способом вычисления площади элементарного фрагмента S. Наиболее известны квадратурные формулы Ньютона-Котеса m-го порядка, в том числе:

*Si = h*f(xi)* - *метод прямоугольников*, m=0
*Si = h/2 * [f(xi-1) + f(xi)]* - *метод трапеций*, m=1
*Si = h/6 * [f(xi-1) + f*f(xi) + f(xi+1)]* - *метод Симпсона*, m=2

Тогда определённые интеграл есть сумма площадей S_i: $I = S_1 + S_2 + S_3 + \dots + S_n$.
А неопределенный, или интеграл с переменным верхним пределом, как функция от аргумента x_i есть:

$F(x_i) = F(x_{i-1}) + S_i$, $F(x_0) = 0$

Квадратурные формулы Гаусса

В рассмотренных выше формулах численного интегрирования используются равноотстоящие узлы и произвольное разбиение отрезка интегрирования. Основная идея квадратуры Гаусса состоит в следующем: при заданном числе интервалов разбиения (порядке квадратурной формулы Гаусса) следует разложить их концы так, чтобы получить наивысшую точность интегрирования. В математическом плане это означает выбор коэффициентов A_i и узлов t_i $(i = 1,2,..,k)$ квадратурных формул Гаусса

$$\int_{-1}^{1} f(t)dt \approx \sum_{i=1}^{k} A_i \cdot f(t_i) \quad + \quad R_k$$

такими, чтобы формулы были точны для многочленов наивысшей возможной степени N. Можно показать, что при k узлах точно интегрируются все многочлены степени $N \le 2k - 1$.

Правило определения узлов и коэффициентов квадратурной формулы Гаусса следующее. Узлы t_i являются корнями многочлена Лежандра соответствующего порядка $P_k(t) = 0$, который определяется с помощью производных:

$$P_k(t) = \frac{1}{2^k \cdot k!} \frac{d^k\{(t^2-1)^k\}}{dt^k}$$

или через рекуррентные соотношения:

$$P_0(t) = 1, \quad P_1(t) = t, \quad P_{m+1}(t) = \frac{(2m+1)\cdot t \cdot P_m(t) - m \cdot P_{m-1}(t)}{m+1}$$

Коэффициенты A_i вычисляются через найденные узлы t_i по формуле:

$$A_i = \frac{2}{(1-t_i^{\,2})\cdot[P_i'(t_i)]^2}, \quad i = 1,2,...,k$$

Погрешность усечения, определяемую остаточным членом R_k , можно оценить по выражению

$$R_k = \frac{2^{2k+1}}{(2k+1)\cdot(2k)!}\left[\frac{(k!)^2}{(2k)!}\right]^2 \cdot f^{(2k)}(t), \quad t \in [-1,1]$$

При вычислении определенного интеграла $\int_a^b f(x)dx$ отрезок $[-1,1]$ преобразуется в отрезок $[a,b]$ путем замены переменной

$$x_i = \frac{b+a}{2} + \frac{b-a}{2} \cdot t_i .$$

В результате квадратурная формула Гаусса k-го порядка приобретает вид

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \cdot \sum_{i=1}^{k} A_i \cdot f(x_i) \quad + \quad R_k^* , \text{ где ошибка } \quad R_k^* = \left[\frac{b-a}{2}\right]^{2k+1} \cdot R_k .$$

Квадратурная формула Гаусса обеспечивает высокую точность интегрирования при небольшом числе узлов и используется для вычисления интегралов от аналитически неинтегрируемых функций.

14. Методы решения нелинейных уравнений. Методы бисекции и "золотого сечения", секущих (хорд) и парабол.

Классификация:
1. *Алгебраические*:

$$a_0 + a_1 x + \cdots + a_n x^n = 0$$

2. *Трансцендентные*: все остальные (тригонометрические, логарифмические)

Все численные методы предназначены для поиска одного вещественного корня.
За одну реализацию может быть найден один корень.
Для запуска любого численного метода в качестве входного параметра необходимо задать интервал, где существует хотя бы один корень или бесконечное количество корней.

$$X^* \in [a,b], \text{если } f(a) * f(b) \leq 0$$

Если точность – γ, решение найдено, если $|X_{k+1} - X_k| < \gamma$

В научной и инженерной практике часто возникает необходимость решения уравнений вида f (x) = 0 , где функция f(x) определена и непрерывна на некотором
интервале a ≤ x ≤ b . Аналитическое решение имеют только алгебраические нелинейные уравнения (в которых функция f(x) есть многочлен степени n:

$f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_n \cdot x^n$) порядка не выше $n = 4$.

Решение уравнения с любым другим видом функции f(x), называемого тогда *трансцендентным*, может быть получено только численными методами.
Все численные методы решения нелинейных уравнений есть *итерационные методы*, в которых решение достигается с заданной точностью путем последовательных приближений (итераций). Количество итераций, необходимых каждому методу для достижения решения с заданной точностью, определяет скорость сходимости каждого метода.
Все численные методы за одну реализацию определяют только один из возможных нескольких корней нелинейного уравнения. Процесс отыскания корня уравнения f(x) = 0 состоит из двух этапов:
1) отделения корней, т.е. отыскания интервалов по x, на которых располагается только по одному действительному корню;
2) уточнения значений корней на каждом интервале до некоторой заданной степени точности ε .

Первый этап выполняется различными способами, в том числе графическим, исходя из физического смысла задачи и т.д. В любом методе проверка (верификация) выбранного интервала, где есть только один корень, осуществляется выполнением условия f (a) · f (b) ≤ 0 , т.е. на границах интервала функция f(x) имеет разные знаки.
Второй этап, определяющий способ приближения к решению, и задает вид численного метода. На каждой k-й итерации метод определяет k-е приближение к истинному решению x, поэтому условие – есть условие достижения заданной точности, т.е. прекращения работы алгоритма.

Метод бисекции (деления отрезка пополам)

Для нахождения решения уравнения $f(x)=0$ интервал, где находится только один корень, делится пополам: $x_0=\frac{a+b}{2}$. Для каждой из половинок проверяется условие существования корня $f(a)\cdot f(x_0)\le 0$ или $f(x_0)\cdot f(b)\le 0$. Интервал, для которого условие не выполняется, отбрасывается. Таким образом, после первой итерации имеем интервал вдвое меньшей длины, чем ***[a, b]***, где опять есть только один корень. Этот процесс повторяется до тех пор, пока длина интервала не станет меньше заданной точности $|x_{k+1}-x_k|<\varepsilon$.

Метод "золотого сечения" (Фибоначчи)

Отличается от метода бисекции только тем, что интервал делится не пополам, а в пропорции "золотого сечения":

$\frac{I_{min}}{I_{max}}=\frac{I_{max}}{I_{max}+I_{min}}$, $I_{min}=|x_0-a|$, $I_{max}=|b-x_0|$, или приближенно

$x_0\approx a+0.618\cdot(b-a)$, если $|f(a)|\ge|f(b)|$ и, наоборот, $x_0\approx a+0.382\cdot(b-a)$, если $|f(a)|<|f(b)|$. В большинстве случаев такой метод сходится быстрее метода бисекции.

Метод хорд (секущих)

Отличается от метода бисекции тем, что новая точка, разделяющая исходный интервал ***[a, b]*** на два участка, находится как точка пересечения с осью *OX* хорды – отрезка, соединяющего точки ***f(a)*** и ***f(b)***. Уравнение хорды определяет алгоритм метода:

$x_{k+1}=x_k-\frac{x_k-x_{k-1}}{f(x_k)-f(x_{k-1})}\cdot f(x_k)$, где $k=0,1,2,\ldots$ и $x_0=a$, $x_1=b$.

Метод парабол

Во многом сходен с методом хорд (секущих). На интервале ***[a, b]*** выбирается точка x_0. По трем точкам $f(a), f(x_0), f(b)$ интерполируется парабола $y(x)=c_0+c_1\cdot x+c_2\cdot x^2$. Новое значение x_{k+1}, разделяющее предыдущий интервал на две части, выбирается как корень квадратного уравнения $y(x)=0$, принадлежащий исходному интервалу. Корни квадратного уравнения определяются аналитически. Затем выбирается интервал, на котором есть искомый корень $f(x)=0$, и процесс повторяется до достижения заданной точности. Решение находится за 2–4 итерации.

15. Метод Ньютона, метод простых итераций для решения нелинейного уравнения. Оценка сходимости метода

Метод Ньютона

Одномерный случай

Для того, чтобы решить уравнение $f(x) = 0$, пользуясь методом простой итерации, необходимо привести его к виду $x = \phi(x)$, где ϕ — сжимающее отображение. Чтобы отображение было наиболее эффективно, необходимо, чтобы в точке очередной итерации x^* выполнялось $\phi'(x^*) = 0$. Будем искать решение данного уравнения в виде $\phi(x) = x + \alpha(x)f(x)$, тогда:

$$\phi'(x^*) = 1 + \alpha'(x^*)f(x^*) + \alpha(x^*)f'(x^*) = 0$$

Воспользуемся тем, что $f(x) = 0$, и получим окончательную формулу для $\alpha(x)$:

$$\alpha(x) = -\frac{1}{f'(x)}$$

С учётом этого сжимающая функция примет вид:

$$\phi(x) = x - \frac{f(x)}{f'(x)}$$

Тогда алгоритм нахождения численного решения уравнения $f(x) = 0$ сводится к итерационной процедуре вычисления:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Многомерный случай

Обобщим полученный результат на многомерный случай.

Выбирая некоторое начальное приближение $\vec{x}^{[0]}$, находят последовательные приближения $\vec{x}^{[j+1]}$ путем решения систем уравнений:

$$f_i + \sum_{k=1}^{n} \frac{\partial f_i}{\partial x_k}(x_k^{[j+1]} - x_k^{[j]}) = 0, \quad i = 1, 2, \ldots, n,$$

где $x^{[j]} = \left(x_1^{[j]} \ldots x_k^{[j]} \ldots x_n^{[j]}\right), \quad j = 0, 1, 2, \ldots$

Для понимания реализация в маткаде:

```
x(a,b) := | x1 ← (a + b)/2
          | delta ← 9999
          | while |delta| > ε
          |     | x2 ← x1 − f(x1) / (d/dx1 f(x1))
          |     | delta ← |x2 − x1|
          |     | x1 ← x2
          | return x2
```

Метод простых итераций

Исходное нелинейное уравнение $f(x)=0$ трансформируется к виду, удобному для итерации $x=\varphi(x)$, например, прибавлением x к обеим частям исходного уравнения $x=f(x)+x$, где $\varphi(x)=f(x)+x$. Тогда итерационный алгоритм задается выражением $x_{k+1}=\varphi(x_k)$. Метод сходится, если выполняется условие $|\varphi'(x)|<1$ на интервале определения *[a, b]*. Если условие не выполняется, необходимо изменить исходное уравнение, умножив его на корректирующую функцию $\xi(x)$, которая не имеет корней на заданном интервале или равна константе. Тогда новое уравнение будет иметь те же решения, что и исходное, а подбором функции $\xi(x)$ можно добиться выполнения условия сходимости метода.

Для понимания реализация на маткаде:

```
x(a,b) := | x1 ← (a + b)/2
          | delta ← 9999
          | while |delta| > ε
          |     | x2 ← x1 + f(x1)
          |     | delta ← |x2 − x1|
          |     | x1 ← x2
          | return x2
```

16. Методы решения системы нелинейных уравнений. Метод Ньютона и метод простых итераций.

$$\begin{cases} f_1(x_1,\ldots,x_n)=0 \\ \quad\vdots \\ f_n(x_1,\ldots,x_n)=0 \end{cases} \Rightarrow F(X)=0$$

Метод Ньютона(касательных)

$$X(k+1)=X(k)-W^{-1}(X(k))\cdot F(X(k))$$

, где

$$W(X)=\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$ - матрица Якоби,

$$X(0)=\begin{bmatrix}0\\ \vdots\\ 0\end{bmatrix}$$

Метод простых итераций

$$\begin{matrix}x_1+\\ \\ x_n+\end{matrix}\begin{cases}f_1(x_1,\dots,x_n)=0+x_1\\ \vdots \\ f_n(x_1,\dots,x_n)=0+x_n\end{cases}\Rightarrow F(X)=0\rightarrow X=\varphi(X)$$

$$X(k+1)=\varphi(X(k))$$

$\|W(X(0))\|<1$ - норма матрицы

17. Задачи оптимизации. Методы спуска.

Задачей оптимизации в называется задача о нахождении экстремума (минимума или максимума) вещественной функции в некоторой области.
Найти множество X_{min}, которая даёт $F(x_1,\dots,x_n) = min$

Методы спуска
1. *Метод Монте-Карло* (случайного поиска, тыка)
2. *Метод спуска* (научного тыка, т.е. выбор нового опирается на предыдущего)
   - покоординатный
   - градиентный
   - метод наискорейшего спуска (сочетание первых двух) - вначале считается градиент, он задаёт направление, после используем покоординатный . Спусаемся, пока уменьшаемся, потом пересчитывается градиент, и повторяем.

18. Оптимизация на основе генетических алгоритмов.

Генети́ческий алгори́тм — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путем последовательного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию. Является разновидностью эволюционных вычислений (англ. *evolutionary computation*). Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

1. генерация стартовой популяции (мин 500)
2. кроссовер(побитовое скрещивание)
   для каждой пары задаётся случайная точка кроссовера, например:

папа	**1**	**0**	**1**	1	0	0	1	1
мама	**1**	**1**	**1**	0	0	1	0	0
ребёнок1	**1**	**0**	**1**	0	0	1	0	0
ребёнок2	**1**	**1**	**1**	1	0	0	1	1

теперь популяция в 2 раза больше
3. селекция(отбор)

-элитный: строим точки, сортируем по возрастанию, выбираем наименьшие
-отбор по принципу "рулетка"(Капитал Шоу ПолеЧудес)
-турнирный отбор
после чего популяция уменьшается в 2 раза
4. проверка устойчивости популяции(мутация)
задана вероятность мутации, равная 0.001, при которой инвертируется произвольный бит

шаги 2-4 - это поколение
После 10к поколений более 85% будет иметь одинаковый вид

19. Полуаналитические методы решения дифференциальных уравнений. Метод Пикара и метод последовательного дифференцирования для решения задачи Коши.

В результате решения ДУ полуаналитическими методами на выходе – сумма ряда, и точность решения регулируется длиной ряда.

Метод Пикара

Идея: пусть

$$y' = f(t,y) \rightarrow \int_0^t y' = \int_0^t f(t,y)dt$$

Пример алгоритма метода:

Пусть у нас есть функция y’ = f(t,y) и начальное условие y(0).

Тогда первый шаг:

$$\int_0^t f(t,y)dt \rightarrow \int_0^t f\big(t,y(0)\big)dt$$

Второй шаг:

$$y(t) = y(0) + \int_0^t f\big(t,y(0)\big)dt \rightarrow y(t) = y(0) + \int_o^t f\big(t,y(t)\big)dt$$

где под вторым интегралом y(t) –результат поиска y(t) на предыдущем шаге. Так можно подставлять в текущую функцию значения предыдущих результатов до посинения, получая всё более совершенную точность. Однако ряд при каждой подстановке на выходе получается всё длиннее, что приводит к увеличению длительности расчётов.

Метод последовательного дифференцирования

В окрестности решения точку можно разложить на ряд Тейлера (упрощённо – Макларена).

$$y(t) = y(0) + y'(0)t + \frac{y''(0)}{2!}t^2 + \frac{y'''(0)}{3!}t^3 + \cdots$$

Пример:

Y’=-3y+5t, y(0)=2

*1. y’(0)=-3*2+5*0=-6*

2. y’’=-3y’+5
y’’(0)=-3-6+5=23*

3. y’’’=-3’’
y’’’(0)=69

20. Численные методы решения дифференциальных уравнения в форме задачи Коши. Явные и неявные схемы. Методы Рунге-Кутты для одного дифференциального уравнения и для системы ОДУ.

Зада́ча Коши́ — одна из основных задач теории дифференциальных уравнений (обыкновенных и с частными производными); состоит в нахождении решения (интеграла) дифференциального уравнения, удовлетворяющего так называемым начальным условиям (начальным данным).

Задача Коши обычно возникает при анализе процессов, определяемых дифференциальным законом и начальным состоянием, математическим выражением которых и являются уравнение и начальное условие (откуда терминология и выбор обозначений: начальные данные задаются при $t = 0$, а решение отыскивается при $t > 0$).

От краевых задач задача Коши отличается тем, что область, в которой должно быть определено искомое решение, здесь заранее не указывается. Тем не менее, задачу Коши можно рассматривать как одну из краевых задач.

Основные вопросы, которые связаны с задачей Коши, таковы:

1. Существует ли (хотя бы локально) решение задачи Коши?
2. Если решение существует, то какова область его существования?
3. Является ли решение единственным?
4. Если решение единственно, то будет ли оно корректным, то есть непрерывным (в каком-либо смысле) относительно начальных данных?

Говорят, что задача Коши имеет единственное решение, если она имеет решение $y = f(x)$ и никакое другое решение не отвечает интегральной кривой, которая в сколь угодно малой выколотой окрестности точки (x_0,y_0) имеет поле направлений, совпадающее с полем направлений $y = f(x)$. Точка (x_0,y_0) задаёт начальные условия.

Явные методы

Явными называют такие методы, в которых последующее значение функции явно выражается через предыдущее. Простейший пример – метод Эйлера:

$$y_{k+1} = y_k + h \cdot f(x_k, y_k).$$

Явные методы более устойчивы, но обладают свойством накапливать ошибку с объемом вычислений.

Неявные методы

В таких методах неизвестное значение функции в новом узле входит как в левую, так и в правую часть разностной схемы. Например, метод Эйлера–Коши:

$$y_{k+1} = y_k + \frac{h}{2}(f(x_k, y_k) + f(x_{k+1}, y_{k+1})).$$

Неявный метод требует предварительного определения y_k, которое можно вычислить с использованием явного метода, например метода Эйлера (см. выше). Такие методы неустойчивы, но обладают большей точностью.

Методы Рунге–Кутта

Различные методы данной группы отличаются друг от друга объемом производимых вычислений и получаемой при этом точностью. Они включают в себя комбинации явных и неявных методов, в том числе и метод Эйлера (метод Рунге–Кутта 0-го порядка), метод Эйлера–Коши (метод Рунге–Кутта 1-го порядка). В качестве примера приведем метод Рунге–Кутта 4-го порядка:

$$y_{k+1} = y_k + \frac{h}{6}\cdot\left[f_1 + 2f_2 + 2f_3 + f_4\right],$$

$$f_1 = f(x_k, y_k), \qquad f_2 = f\left(x_{k+\frac{1}{2}}, y_k + \frac{h}{2}\cdot f_1\right),$$

где

$$f_3 = f\left(x_{k+\frac{1}{2}}, y_k + \frac{h}{2}\cdot f_2\right), \qquad f_4 = f(x_{k+1}, y_k + h\cdot f_3).$$

Здесь точность повышается за счет использования фиктивных промежуточных точек на половине шага $x_{k+\frac{1}{2}}, y_{k+\frac{1}{2}}$.

21. Многошаговые схемы. Прогноз и коррекция. Метод Адамса-Башфорта.

Многошаговые схемы - это такие схемы, в которых, для определения текущей точки, нужно, используя явные и неявные схемы, найти предыдущие, и скорректировать их.

Прогнозом называется нахождение точки явным методом, а ***коррекцией*** - подстановка найденной в прогнозе точки в неявный метод.

Метод Адамса-Башфорта - метод, в котором для прогноза используется результат, полученный явным методом *Адамса*, а для коррекции - неявный метод *Башфорта*.

Адамса: $y_{k+1} = y_k + (h / 24) * [55f(t_k,y_k) - 59f(t_{k-1},y_{k-1}) + 37f(t_{k-2},y_{k-2}) - 9f(t_{k-3},y_{k-3})]$

Башфорта: $y_{k+1} = (h / 24) * [9f(t_{k+1},y_{k+1}) + 19f(t_k,y_k) - 5f(t_{k-1},y_{k-1}) + f(t_{k-2},y_{k-2})]$

22. Методы решения "жестких" ОДУ. Методы Гира.

Чем жестче уравнение, тем больше шагов в обычных численных методах требуется для его устойчивого решения. Для решения более жестких уравнений требуются миллионы, миллиарды и даже большее число шагов. Строгого общепринятого математического определения жестких ОДУ нет. Принято считать, что жесткие системы - это те уравнения, решение которых получить намного с помощью определённых неявных методов, чем с помощью явных методов.

Неявные алгоритмы Гира наиболее эффективны для решения так называемых жестких уравнений, особенностью которых является медленное изминение их решений при наличии быстро затухающих возмущений.

Изначально требуется найти значения функции, используя любой явный метод, после чего используя данную формулу можно найти следующие решения (причём при вычислении каждого следующего решения нужно совершить несколько итерационных приблежений), для запуска алгоритма также требуется знать некоторое количество начальных решений (формула для вычисления метода гира 3го порядка):

$Y_i = 18/11 * Y_{i-1} - 9/11 * Y_{i-2} + 2/11 * Y_{i-3} + 6/11 * h * f(x)$

Метод Гира относится к многошаговым, т.н. методам дифференцирования назад. Общий вид ФДН метода таков:

$$u_{n+1} + \sum_{j=1}^{k} \alpha_j u_{n+1-j} = \tau\beta f_{n+1},$$

где коэффициенты выбираются из условий аппроксимации метода.

Таблица коэффициентов методов Гира

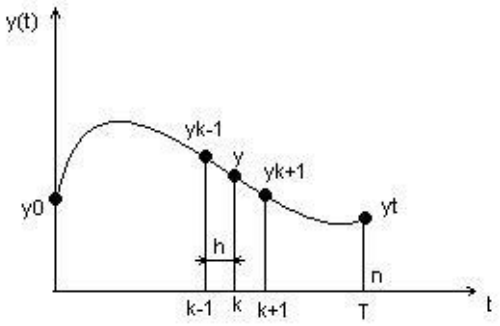
k	B	a_0	a_1	a_2	a_3	a_4	a_5
1	1	- 1					
2	2/3	1/3	- 4/3				
3(86)	6/11	- 2/11	9/11	- 18/11			
4(73, 35)	12/25	- 3/25	16/25	36/25	- 48/25		
5(51, 84)	$\frac{60}{137}$	$\frac{-12}{137}$	$\frac{75}{137}$	$\frac{-200}{137}$	$\frac{300}{137}$	$\frac{-300}{137}$	
6(17, 84)	$\frac{600}{147}$	$\frac{10}{147}$	$\frac{-72}{147}$	$\frac{225}{147}$	$-\frac{400}{147}$	$\frac{450}{147}$	$\frac{-360}{147}$

23. Методы решения краевых задач. Метод стрельбы и метод конечных разностей.

Краевая задача — дифференциальное уравнение (система дифференциальных уравнений) с заданными линейными соотношениями между значениями искомых функций на начале и конце интервала интегрирования.

Решение краевой задачи ищется в виде суммы линейной комбинации решений однородных задач Коши, соответствующих заданному уравнению при линейно независимых векторах начальных условий, и решения неоднородной задачи Коши с произвольными начальными условиями.

Метод конечных разностей



$$y \to y_k; y' = \frac{y_{k+1} - y_{k-1}}{2h}; y'' = \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2}$$

$$k \to \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} + p(t_k)\frac{y_{k+1} - y_{k-1}}{2h} + q(t_k)y = f(t_k)$$

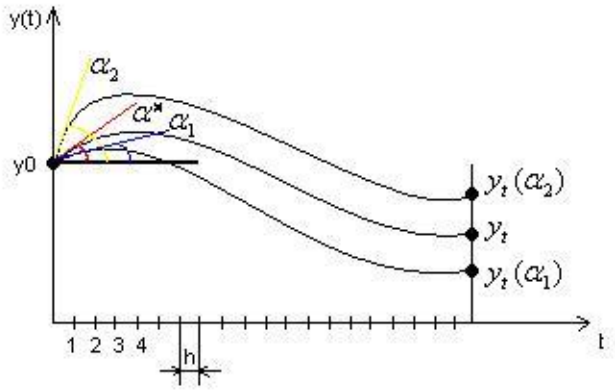
k+1,k, k-1 - ?

получаем конечно-разностный аналог дифференциального уравнения. Заменяем производные на конечные разности.

$$\to \left(\frac{1}{h^2} - \frac{p(t_k)}{2h}\right)y_{k-1} + \left(q(t_k) - \frac{2}{h^2}\right)y_k + \left(\frac{1}{h^2} + \frac{p(t_k)}{2h}\right)y_{k+1} = f(t_k)$$ $A_{kk-1}y_{k-1} + A_{kk}y_k + a_{kk+1}y_{k+1} = b_k$

В результате получаем трёхдиагональную матрицу, решая которую находим А.

Метод стрельбы



Трансформация краевой задачи в задачу Коший
tg(*a**) = y'(0) //a=alpha
(1) Выбираем произвольный угол прицеливания
-90° < *a*1 < 90°
tg(*a*1) = y'(0)
(2) Выбираем *a*2 = *a*1 + delta *a* - определяем чувствительность отклонения

$$\frac{\partial y_t}{\partial \alpha} = \frac{y_t(\alpha_2) - y_t(\alpha_1)}{\Delta \alpha}$$

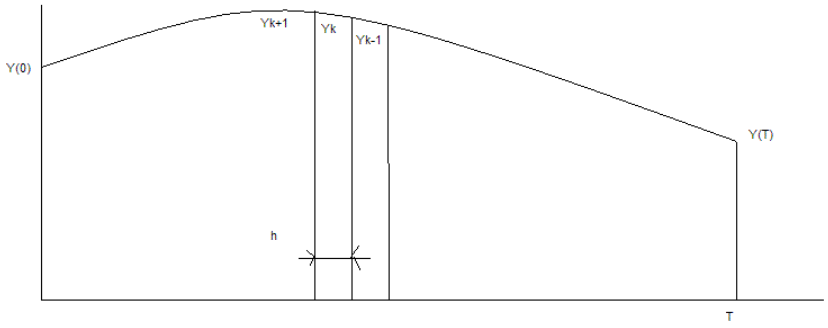
- чувствительность.

(3)

$$\Delta^* \alpha = \frac{y_t(\alpha_1) - y_t}{\frac{\partial y_t}{\partial \alpha}}; \alpha^* = \alpha_1 + \Delta^* \alpha$$

24. Численно-аналитические методы решения краевых задач. Метод коллокаций.

От классической задачи Коши краевая задача отличается постановкой условий:
y''=f(t,y,y'), y(0), y(t), т.е. задача - найти y(t), удовлетворяющий требованиям и проходящий через две указанные в ограничениях точки. Методы решения краевые задачи годятся только для решения линейных дифференциальных уравнения второго порядка.
Идея решения:



T_k=k*h

Метод конечных разностей

$$y \to y_k; y' = \frac{y_{k+1} - y_{k-1}}{2h}; y'' = \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2}$$

$$k \to \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} + p(t_k)\frac{y_{k+1} - y_{k-1}}{2h} + q(t_k)y = f(t_k)$$

k+1,k, k-1 - ?

получаем конечно-разностный аналог дифференциального уравнения. Заменяем производные на конечные разности.

$$\rightarrow \left(\frac{1}{h^2} - \frac{p(t_k)}{2h}\right) y_{k-1} + \left(q(t_k) - \frac{2}{h^2}\right) y_k + \left(\frac{1}{h^2} + \frac{p(t_k)}{2h}\right) y_{k+1} = f(t_k)$$

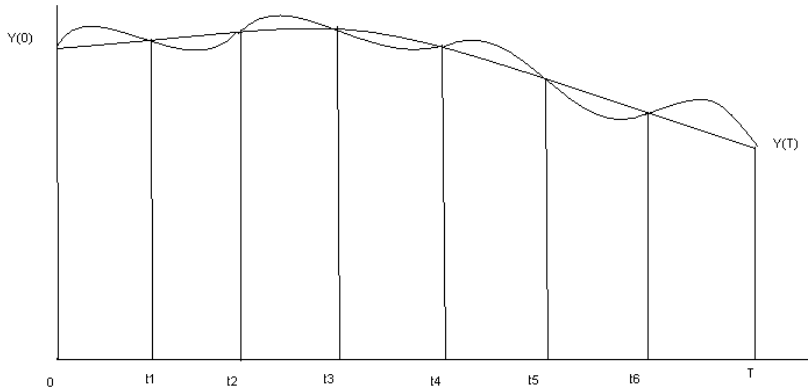
$$A_{kk-1}y_{k-1} + A_{kk}y_k + a_{kk+1}y_{k+1} = b_k$$

В результате получаем трёхдиагональную матрицу, решая которую находим A.

Метод коллокаций

Возьмём некоторые точки во времени, где поведение функции нас интересует: *точки коллокации*. Потребуем, чтоб в этих точках невязка была = 0. Точек надо столько, сколько есть неизвестных коэфициентов. Идея метода близка к интерполяции.

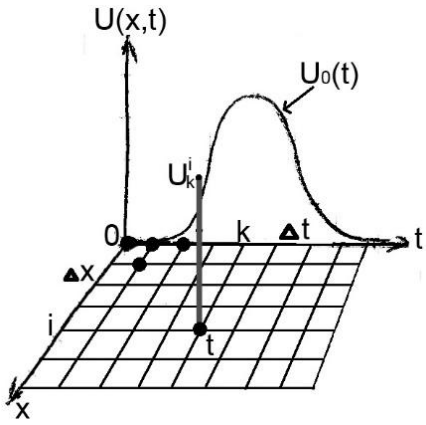
$$\begin{cases} R(t_1, a_1 \dots a_n) = 0 \\ \dots \\ R(t_n, a_1 \dots a_n) = 0 \end{cases}$$



25. Метод конечных разностей для решения дифференциальных уравнений в частных производных. Синтез сетки решения.

23, 24 вопрос, там есть метод конечных разностей.

Синтез сетки

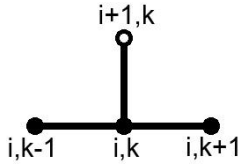


$U(0,t)=U_0(t)$

$U(x,0)=0$

$$t_k = k \cdot \Delta t$$
$$x_i = i \cdot \Delta x$$
$$U(x_i, t_k)$$

$$\Delta x \le 0.01 \cdot v_0 \cdot \Delta t$$
$$Ux = \frac{1}{v_0} Ut$$
$$\frac{U_k^{i+1} - U_k^i}{\Delta x} = -\frac{1}{v_0}\left(\frac{U_{k+1}^i - U_{k-1}^i}{2\Delta t}\right)$$
$$U_{k+1}^i = U_k^i - \frac{\Delta x}{2v_0 \cdot \Delta t}\left(U_{k+1}^i - U_{k-1}^i\right)$$



26. Методы анализа сигналов и графических изображений. Морфологические алгоритмы. Клеточные автоматы.

Есть:
- метод конечных элементов (идея Стэлс, всё треугольниками)
- метод граничных элементов - дана хрень, с каким-нибудь шагом идём по поверхности, и ставим линии, перпендикулярные поверхности, с какой-нибудь определённой высотой, когда прошли по всей поверхности, то проводим новую поверхность, но уже в вершинах перпендикуляров. В результате - фигура стала меньше/больше, в зависимости от того в какую сторону направляли перпендикуляры.
- морфологические методы

Морфологические методы
$x_{k+1}=f(x_k)$
$x_{k+1}=r \cdot x_k(1-x_k)$
$0<r<4 \quad 0<x_0<1$
(при $3.57<r<3.999$, ф-ия выглядит как случайная)

1. **Дилатация** (буквы становятся **жирней**)
2. *Эрозия* (**буквы становятся** *тонкими*)
3. **Скле***йка* (**Дилатация** + *Эрозия*)
4.*Размы***вание** (наоборот)