

**КУРС ЛЕКЦИЙ
ПО ДИСЦИПЛИНЕ
ЧИСЛЕННЫЕ МЕТОДЫ
для специальности
6-05-0612-01 Программная инженерия**

Лекция 1. Основы теории погрешностей (3ч. – 1ч. ЛК+2ч. ЛР)

1. Источники и классификация погрешностей. Абсолютная и относительная погрешности. Значащие и верные цифры.
2. Погрешности (относительные) арифметических операций. Погрешность функции одной и многих переменных. Обусловленность вычислительной задачи.
3. Представление чисел в ЭВМ. Понятия машинного эпсилон, машинной бесконечности, машинного нуля.
4. Вычислительные задачи. Корректность и обусловленность вычислительных задач. Вычислительные алгоритмы. Катастрофическая потеря точности.

Откуда же могут возникнуть ошибки? Таких причин много. Во-первых, исходные данные для вычислений часто получаются из эксперимента, а каждый эксперимент может дать результат с ограниченной точностью. Во-вторых, в процессе вычислений приходится использовать иррациональные величины, такие, например, как π , e , $\sqrt{2}$. Так как при вычислении на ЭВМ мы можем использовать ограниченное количество разрядов, то эти числа также будут представлены лишь приближенно. В-третьих, во многих случаях существующие методы решений задач могут дать точный ответ лишь после бесконечного числа шагов. Так как практически это осуществить нельзя, то мы будем вынуждены остановиться на каком-то конечном шаге и, следовательно, не достигнем точного значения, например, при вычислении суммы ряда мы ограничиваемся суммой конечного числа первых членов. В-четвертых, уже при таких простейших операциях, как умножение и деление, у нас может сильно возрасти количество разрядов и результаты могут не помещаться в памяти машины. В этом случае мы будем вынуждены отбросить некоторое количество разрядов. Наконец, исходные погрешности будут последовательно переходить, преобразовываясь, от операции к операции и порождать новые погрешности.

Итак, под погрешностью понимается некоторая величина, характеризующая точность результата.

Существует три вида погрешностей:

- 1 – **неустраняемая погрешность** (возникающая из-за неточности исходной информации, например, неточности измерений);

2 – **погрешность метода** (заданная нам задача заменяется другой, и в силу этого мы получаем ошибку);

3 – **погрешность вычислений** (возникающая из-за округлений).

Основная задача теории погрешностей – указание области неопределённости результата.

Абсолютная и относительная погрешности

Значащими цифрами числа называются все цифры в его записи, начиная с первой ненулевой слева:

Примеры.

1. $x = 2,396029$ – все цифры (и 0!) значащие;

2. $x = 0,00267$ – значащие только 2, 6, 7; первые три нуля – незначащие, т.к. они служат для вспомогательной цели – определению положения цифр 2, 6, 7, поэтому может быть принята запись $x = 2,67 \cdot 10^{-3}$;

3. $x = 2270000$ и $x = 2,27 \cdot 10^6$ – в первой записи все семь цифр (и последние четыре нуля!) значащие; во второй – значащие только 2, 2, 7.

Если известно, что число c – число точное и $c = 3200$, то для него нельзя использовать запись $c = 3,2 \cdot 10^3$, т.к. тем самым два нуля переводятся в разряд незначащих цифр.

Итак, **значащими цифрами числа** называются все цифры в его записи, начиная с первой ненулевой слева. Значащие цифры подразделяют на **верные** и **сомнительные**, исходя из понятия абсолютной и относительной погрешности числа.

Приближённым числом a называется число, незначительно отличающееся от такого числа A и заменяющее его в вычислениях.

Ошибкой или **погрешностью** приближенного числа a называется разность между точным числом A и его приближенным значением $\Delta a = A - a$. Если $A > a$, то ошибка положительна ($\Delta a > 0$), если $A < a$, то ошибка отрицательна ($\Delta a < 0$).

Величина $\Delta = |A - a|$ называется **абсолютной погрешностью** приближенного числа a .

Отношение абсолютной погрешности к модулю точного значения

$$\delta = \frac{\Delta}{|A|}, \quad (1.1)$$

называется **относительной погрешностью** приближенного числа a .

Таким образом, абсолютная и относительная погрешности связаны формулой

$$\Delta = \delta |A|. \quad (1.2)$$

Абсолютная погрешность является мерой ошибки числа a , а относительная погрешность определяет меру точности приближенного числа a . На практике известны лишь границы абсолютной и относительной погрешности приближенного числа.

Любое число Δ_a такое, что $|A - a| \leq \Delta_a$ называется **предельной абсолютной погрешностью**, а δ_a такое, что $\frac{\Delta}{|A|} \leq \delta_a$ называется **предельной относительной погрешностью**. Отсюда следует, что точное число A заключено в границах $a - \Delta_a \leq A \leq a + \Delta_a$. Для краткости пишут $A = a \pm \Delta_a$.

В качестве Δ_a и δ_a следует выбирать ближайшие к Δ числа, но большие их. В качестве δ_a числа a можно принять

$$\delta_a = \frac{\Delta_a}{a - \Delta_a}. \quad (1.3)$$

Пример. $\Delta = 0,0127, \Delta_a = 0,013; \delta = 0,0374, \delta_a = 0,038$.

Так как точное значение A , как правило, неизвестно $A \approx a$, то при достаточно малой абсолютной погрешности Δ можно считать, что $\Delta_a = |a| \delta_a$.

Относительная погрешность – это безразмерная и нормированная величина, поэтому её удобно использовать для сравнения точности различных приближённых величин. Относительная погрешность обычно выражается в процентах, и её принято записывать не более чем с двумя-тремя значащими цифрами (знаками). Относительная погрешность приближенного числа связана с количеством его верных знаков. Ориентировочно можно считать, что наличие только одного верного знака соответствует относительной погрешности порядка 10% (т.е. 10^{-1}), двух верных знаков – погрешности порядка 1% (т.е. 10^{-2}), трёх верных знаков – погрешности порядка 0,1% (т.е. 10^{-3}) и т.д.

Количество верных значащих цифр числа отсчитывается от первой значащей цифры до первой значащей цифры его абсолютной погрешности; остальные цифры числа называют сомнительными. В окончательных результатах вычислений обычно оставляют верные цифры и одну сомнительную.

Пример. Число $S = 20,7426$ с абсолютной погрешностью $\Delta_S = 0,0926$ имеет три верных знака – 2, 0, 7; остальные знаки – сомнительные.

Связь абсолютной и относительной погрешностей с числом верных значащих цифр

Пусть число a в позиционной системе с основанием 10 имеет вид

$$a = \alpha_m 10^m + \alpha_{m-1} 10^{m-1} + \alpha_{m-2} 10^{m-2} + \dots + \alpha_{m-n+1} 10^{m-n+1} + \dots,$$

где α_i – цифры числа a ($\alpha_i = 0, 1, 2, \dots, 8, 9$) и $\alpha_m \neq 0$.

Определение 1. Все сохраняемые в десятичном представлении числа цифры α_i называются **значащими** и нуль, который находится между значащими цифрами числа или является представителем сохраняемого десятичного разряда за исключением нулей, которые служат лишь для обозначения разрядов.

Пример. $a = 135,76054$ – 8 значащих цифр; $a = 0,00751$ – 3 значащих цифры, т.к. его можно представить $0,751 \cdot 10^{-2}$.

В представлении приближенных чисел используется понятие верных значащих цифр в узком и широком смыслах.

Определение 2. Первые n значащих цифр приближенного числа a являются **верными в узком смысле**, если абсолютная погрешность этого числа не превышает половины единицы разряда, выражаемого n -ой значащей цифрой, считая слева направо.

Определение 3. Первые n значащих цифр приближенного числа a являются **верными в широком смысле**, если абсолютная погрешность этого числа не превышает единицы разряда, выражаемой n -ой значащей цифрой, считая слева направо.

Математически это записывается так:

- в узком смысле

$$\Delta = |A - a| \leq \frac{1}{2} \cdot 10^{-n+m+1}, \quad (1.4)$$

- в широком смысле

$$\Delta = |A - a| \leq 1 \cdot 10^{-n+m+1}, \quad (1.4')$$

где m – показатель степени старшего десятичного разряда, n – количество верных значащих цифр числа a .

Пример. Пусть в числе a все цифры верные.

$$a = 435,7605 = 4 \cdot 10^2 + 3 \cdot 10^1 + 5 \cdot 10^0 + 7 \cdot 10^{-1} + 6 \cdot 10^{-2} + 0 \cdot 10^{-3} + 5 \cdot 10^{-4}, \\ m = 2, \quad n = 7.$$

Определим абсолютную погрешность в узком и широком смысле.

Согласно формул (1.4) и (1.4') следует, что

$$\Delta = 0,5 \cdot 10^{-7+2+1} = 0,5 \cdot 10^{-4}; \quad \Delta = 1 \cdot 10^{-7+2+1} = 1 \cdot 10^{-4}.$$

Пример. Определить число верных значащих цифр числа $a = 4,305642$, если $\Delta = \pm 0,0004 = \pm 0,4 \cdot 10^{-3}$ в узком смысле.

$$\begin{aligned}
0,4 \cdot 10^{-3} &\leq 0,5 \cdot 10^0, & n=1 \\
0,4 \cdot 10^{-3} &\leq 0,5 \cdot 10^{-1}, & n=2 \\
0,4 \cdot 10^{-3} &\leq 0,5 \cdot 10^{-2}, & n=3 \\
0,4 \cdot 10^{-3} &\leq 0,5 \cdot 10^{-3}, & n=4 \\
0,4 \cdot 10^{-3} &> 0,5 \cdot 10^{-4}.
\end{aligned}$$

По определению 2, $\Delta = 0,4 \cdot 10^{-3} \leq 0,5 \cdot 10^{0-4+1} = 0,5 \cdot 10^{-3}$. Следовательно, в числе a четыре верных значащих цифры в узком смысле.

Теорема 1. Если приближенное число a имеет n верных десятичных знаков в узком смысле, то относительная погрешность удовлетворяет неравенству

$$\delta \leq \frac{1}{\alpha_m} 10^{-n+1} \quad \text{и} \quad \delta_a = \frac{1}{\alpha_m} 10^{-n+1} = \frac{1}{\alpha_m} \left(\frac{1}{10} \right)^{n-1}, \quad (1.5)$$

где $\alpha_m \neq 0$ – первая значащая цифра числа a , если $n > 2$, то

$$\delta < \frac{1}{2\alpha_m} 10^{-n+1} \quad \text{и} \quad \delta_a = \frac{1}{2\alpha_m} \left(\frac{1}{10} \right)^{n-1}. \quad (1.5')$$

Формулы (1.5) или (1.5') позволяют определить предельную относительную погрешность по количеству n верных знаков. Если известно, что имеет место оценка

$$\delta < \frac{1}{2} 10^{-n+1},$$

то для абсолютной погрешности справедливо неравенство

$$\Delta \leq \frac{\alpha_m + 1}{2} 10^{-n+m},$$

то есть число a имеет, по крайней мере, n верных знаков.

Прямая задача теории погрешностей

Пусть $u = f(x_1, x_2, \dots, x_n)$ – дифференцируемая функция n аргументов, которые являются приближенными числами, абсолютная погрешность которых $|\Delta x_i|$ мала.

Прямая задача теории погрешностей заключается в вычислении предельной абсолютной и относительной погрешностей функции u при известной погрешности аргументов $|\Delta x_i|$.

Если $|\Delta x_i|$ достаточно малы, то для вычисления предельной погрешности справедливы формулы

$$\Delta u \leq \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| |\Delta x_i|, \quad (1.6)$$

$$\Delta_u \leq \sum_{i=1}^n \left| \frac{\partial u}{\partial x_i} \right| |\Delta x_i|, \quad (1.7)$$

$$\delta u \leq \sum_{i=1}^n \left| \frac{\frac{\partial f}{\partial x_i}}{u} \right| |\Delta x_i| = \sum_{i=1}^n \left| \frac{\partial}{\partial x_i} \ln f(x_1, \dots, x_n) \right| |\Delta x_i|, \quad (1.8)$$

$$\delta_u = \sum_{i=1}^n \left| \frac{\partial}{\partial x_i} \ln u \right| |\Delta x_i|. \quad (1.9)$$

Следствие 1. Если $u = \sum_{i=1}^n x_i$, то из (1.6) и (1.7) следует, что за предельную абсолютную погрешность суммы можно взять сумму абсолютных погрешностей слагаемых $\Delta_u = \sum_{i=1}^n |\Delta x_i|$. Здесь имеется в виду алгебраическая сумма.

Следствие 2. Если все слагаемые одного знака, то из (1.8) и (1.9) следует, что $\delta \leq \delta_{\max}$, где $\delta_{\max}$ – максимальная из относительных погрешностей слагаемых.

Следствие 3. Если $u = \prod_{i=1}^n x_i$, то из (1.9) следует, что за предельную относительную погрешность сомножителей можно взять сумму относительных погрешностей сомножителей, то есть $\delta_u = \sum_{i=1}^n \delta_{x_i}$.

Следствие 4. Если $u = x^m$, то из (1.7) и (1.9) следует, что $\Delta_u = |m| x^{m-1} |\Delta x|$, $\delta_u = m \delta_x$, т.е. предельная относительная погрешность степени в m раз больше относительной погрешности основания.

Следствие 5. Если $u = \sqrt[m]{x}$, то из (1.8) и (1.9) следует, что $\delta_u = \frac{1}{m} \delta_x$, т.е. предельная относительная погрешность корня в m раз меньше предельной относительной погрешности подкоренного выражения.

Замечание. Пусть $u = x_1 - x_2$, где x_1, x_2 – близкие числа, тогда $\Delta_u = |\Delta x_1| + |\Delta x_2|$ (абсолютная погрешность равна сумме абсолютных погрешностей уменьшаемого и вычитаемого) и для относительной погрешности получим формулу

$$\delta_u = \frac{|\Delta x_1| + |\Delta x_2|}{x_1 - x_2}.$$

Так как при достаточно близких значениях x_1, x_2 в знаменателе получится малое число, то относительная погрешность может резко

возрастать, что приведет к потере точности по сравнению с относительными погрешностями $\delta_{x_1}, \delta_{x_2}$.

В случае частного двух величин $x = \frac{x_1}{x_2}$, x_1 и $x_2 > 0$ будем иметь для

относительной погрешности $\delta_x = \sum_{i=1}^n \delta_{x_i}$ (т.е. складываются относительные погрешности чисел), абсолютная погрешность вычисляется по его относительной погрешности $\Delta_x = |x| \cdot \delta_x$.

Обратная задача теории погрешностей

Пусть нужно вычислить $u = f(x_1, x_2, \dots, x_n)$, где f – дифференцируемая функция n аргументов x_i . **Обратная задача** заключается в том, чтобы вычислить допустимые погрешности аргументов x_i с тем, чтобы абсолютная погрешность вычисления функции u не превышала заданную величину. Эта задача математически не определена.

Для того чтобы задача была математически решена необходимо наложить ограничения на определение погрешности элементов с помощью трёх принципов.

1. Принцип равных влияний

Пусть погрешности распределены так, что

$$\left| \frac{\partial u}{\partial x_i} \right| |\Delta x_i| = \left| \frac{\partial u}{\partial x_j} \right| |\Delta x_j|,$$

для любых i, j одинаково влияют на образование общей погрешности Δ_u . Из (1.6) получим

$$\Delta_u = \sum_{i=1}^n \left| \frac{\partial u}{\partial x_i} \right| |\Delta x_i|; \quad \frac{\partial u}{\partial x_1} |\Delta x_1| = \frac{\partial u}{\partial x_2} |\Delta x_2| = \dots = \frac{\partial u}{\partial x_n} |\Delta x_n| = \frac{\Delta_u}{n}.$$

Следовательно,

$$|\Delta x_i| = \frac{\Delta_u}{n \left| \frac{\partial u}{\partial x_i} \right|}, \quad (i = \overline{1, n}). \quad (1.10)$$

2. Принцип равных абсолютных погрешностей

Пусть все аргументы имеют одинаковую абсолютную погрешность $|\Delta x_i| = |\Delta x_j| = \Delta$, для любых i, j , тогда из (1.7) следует, что

$$\Delta_u = \sum_{i=1}^n \left| \frac{\partial u}{\partial x_i} \right| \Delta.$$

Отсюда

$$\Delta = \frac{\Delta_u}{\sum_{i=1}^n \left| \frac{\partial u}{\partial x_i} \right|}, \quad (i = \overline{1, n}). \quad (1.11)$$

3. Принцип равных относительных погрешностей

Пусть относительные погрешности равны между собой, т.е.

$$\frac{|\Delta x_i|}{|x_i|} = \frac{|\Delta x_j|}{|x_j|} = k,$$

откуда $\Delta x_i = k|x_i|$, для любых i, j .

Тогда из (1.10) получим

$$\begin{aligned} \Delta_u &= k \sum_{i=1}^n \left| x_i \frac{\partial u}{\partial x_i} \right|, \\ \Delta_u &= \sum_{i=1}^n \frac{\partial u}{\partial x_i} |\Delta x_i| = k \sum_{i=1}^n \left| \frac{\partial u}{\partial x_i} \right| |x_i|, \\ k &= \frac{\Delta_u}{\sum_{i=1}^n \left| x_i \frac{\partial u}{\partial x_i} \right|}. \end{aligned}$$

Следовательно, окончательно получим,

$$\Delta x_i = \frac{|x_i| \Delta_u}{\sum_{j=1}^n \left| x_j \frac{\partial u}{\partial x_j} \right|}, \quad (i = \overline{1, n}). \quad (1.12)$$

При решении обратной задачи все постоянные, которые не могут быть заданы точно, должны рассматриваться приближенно.

Правила округления приближенных чисел

При решении прямой и обратной задачи в теории погрешностей требуется округление приближенных чисел, так как лишние значащие цифры не увеличивают точность результата, а увеличивают время вычисления.

При округлении приближенного числа поступают следующим образом:

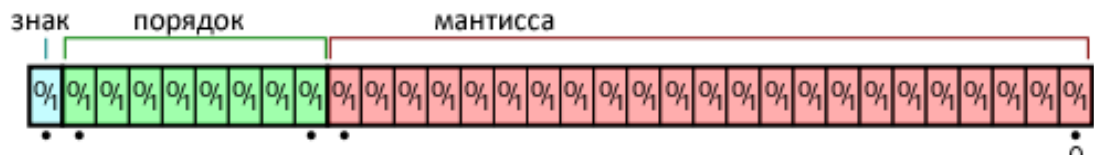
1. Если последняя из отброшенных цифр больше 5, то к последней оставшейся цифре прибавляют 1.
2. Если последняя из отброшенных цифр меньше 5, то последняя оставшаяся цифра не меняется.
3. Если последняя из отброшенных цифр равна 5, а за ней ненулевые, то к последней оставшейся цифре прибавляют 1.
4. Правило четной цифры: если последняя из отброшенных цифр равна 5, а за ней нулевые, то к последней оставшейся цифре прибавляют 1, если она нечетная и последняя цифра не меняется, если она четная.
5. При выполнении арифметических действий над приближенными числами, нужно среди заданных аргументов выбирать наименее точное

число, у которого при сложении и вычитании меньше всего десятичных знаков после запятой, при умножении и делении меньше всего верных значащих цифр. Остальные аргументы округляются, оставляя одну или две дополнительные цифры. Затем производятся арифметические действия, а результат округляется на одну или две значащие цифры.

В далекие времена, для IT-индустрии это 70-е годы прошлого века, ученые-математики (так раньше назывались и программисты) сражались как Дон-Кихоты в неравном бою с компьютерами, которые тогда были размером с маленькие ветряные мельницы. Задачи ставились серьезные: поиск вражеских подлодок в океане по снимкам с орбиты, расчет баллистики ракет дальнего действия, и прочее. Для их решения компьютер должен оперировать действительными числами, которых, как известно, континуум (множество вещественных чисел), тогда как память конечна. Поэтому приходится отображать этот континуум на конечное множество нулей и единиц. В поисках компромисса между скоростью, размером и точностью представления ученые предложили числа с плавающей запятой (или плавающей точкой). Арифметика с плавающей запятой почему-то считается экзотической областью компьютерных наук, учитывая, что соответствующие типы данных присутствуют в каждом языке программирования. Например, решая одну и ту же задачу на CPU (центральный процессор, обрабатывающий данные любого вида) и GPU (графический процессор, способный быстро обрабатывать большие объемы данных) можно получить разный результат. Оказывается, что в потайных углах этой области скрываются очень любопытные и странные явления: некоммутативность (зависимость результата операции над чем-либо от перестановки её элементов) и неассоциативность (это не арифметическая сумма составляющих её элементов: эти элементы нельзя без искажения смысла использовать в другой последовательности; неассоциативность и некоммутативность информации – любая информация – это не арифметическая сумма составляющих её элементов, эти элементы нельзя использовать в другой последовательности. Как говорится, сначала надо думать, а потом делать, но никак не наоборот) арифметических операций, ноль со знаком, разность неравных чисел дает ноль, и прочее. Корни этого айсберга уходят глубоко в математику, остановимся на том, что лежит на поверхности.

Множество целых чисел бесконечно, но мы всегда можем подобрать такое число бит, чтобы представить любое целое число, возникающее при решении конкретной задачи. Множество действительных чисел не только бесконечно, но еще и непрерывно, поэтому, сколько бы мы не взяли бит, мы неизбежно столкнемся с числами, которые не имеют точного представления. Числа с плавающей запятой – один из возможных способов представления действительных чисел, который является компромиссом между точностью и диапазоном принимаемых значений. Число с плавающей запятой состоит из набора отдельных разрядов, условно разделенных на знак, порядок и

мантиссу. Порядок и мантисса – целые числа, которые вместе со знаком дают представление числа с плавающей запятой в следующем виде:



Математически это записывается так:

$$(-1)^s \times M \times B^E,$$

где s – знак, B – основание, E – порядок, M – мантисса.

Основание определяет систему счисления разрядов. Математически доказано, что числа с плавающей запятой с базой $B=2$ (двоичное представление) наиболее устойчивы к ошибкам округления, поэтому на практике встречаются только базы 2 и, реже, 10. Для дальнейшего изложения будем всегда полагать $B=2$, и формула числа с плавающей запятой будет иметь вид:

$$(-1)^s \times M \times 2^E.$$

Что такое мантисса и порядок? **Мантисса** – это целое число фиксированной длины, которое представляет старшие разряды действительного числа. Допустим наша мантисса состоит из трех бит ($|M|=3$). Возьмем, например, число «5», которое в двоичной системе будет равно 101_2 . Старший бит соответствует $2^2=4$, средний (который у нас равен нулю) $2^1=2$, а младший $2^0=1$. **Порядок** – это степень базы (двойки) старшего разряда. В нашем случае $E=2$. Такие числа удобно записывать в так называемом стандартном (научном) виде, например, « $1.01e+2$ ». Сразу видно, что мантисса состоит из трех знаков, а порядок равен двум.

Допустим мы хотим получить дробное число, используя те же 3 бита мантиссы. Мы можем это сделать, если возьмем, скажем, $E=1$. Тогда наше число будет равно

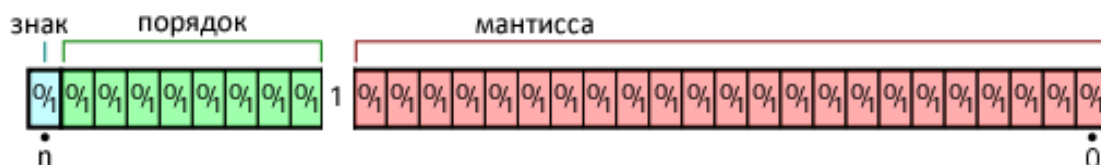
$$1,01e+1 = 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} = 2 + 0,5 = 2,5.$$

Здесь, поскольку $E=1$, степень двойки первого разряда (который идет перед запятой), равна «1». Два других разряда, расположенных правее (после запятой), обеспечивают вклад 2^{E-1} и 2^{E-2} (2^0 и 2^{-1} соответственно). Очевидно, что регулируя E одно и то же число можно представить по-разному. Рассмотрим пример с длиной мантиссы $|M|=4$. Число «2» можно представить в следующем виде:

$$2 = 10 \text{ (в двоичной системе)} = 1.000e+1 = 0.100e+2 = 0.010e+3. \\ (E=1, E=2, E=3 \text{ соответственно}).$$

Обратим внимание, что одно и то же число имеет несколько представлений. Это не удобно для оборудования, т.к. нужно учитывать множественность представления при сравнении чисел и при выполнении над ними арифметических операций. Кроме того, это не экономично, поскольку

число представлений – конечное, а повторения уменьшают множество чисел, которые вообще могут быть представлены. Поэтому уже в самых первых машинах начали использовать трюк, делая первый бит мантиссы всегда положительным. Такое представление называли **нормализованным**.



Это экономит один бит, так как неявную единицу не нужно хранить в памяти, и обеспечивает уникальность представления числа. В нашем примере «2» имеет единственное нормализованное представление («1.000e+1»), а мантисса хранится в памяти как «000», т.к. старшая единица подразумевается неявно. Но в нормализованном представлении чисел возникает новая проблема – в такой форме невозможно представить ноль.

Строго говоря, нормализованное число имеет следующий вид:

$$(-1)^s \times 1.M \times 2^E.$$

Качество решения задач во многом зависит от выбора представления чисел с плавающей запятой. Мы плавно подошли к проблеме стандартизации такого представления.

Исторические аспекты проблемы. В 60-е и 70-е годы прошлого века не было единого стандарта представления чисел с плавающей запятой, способов округления, арифметических операций. В результате программы были крайне не портатбельны (не портатбельная программа при своей инсталляции «размазывает» себя по системе: 1) Основная часть программы записывается в создаваемую для нее директорию в Programm Files; 2) Часть программного кода, расположенную в библиотеках dll, инсталляционная программа помещает в директорию Windows (как правило в System 32 и др.); 3) Настройки программы записываются в реестр Windows – специальную базу данных Windows. Следствием такого «размазывания» является то, что если просто скопировать (например, на флешку) директорию программы (см. пункт 1) и попытаться запустить программу с флешки на другом компьютере, то она работать НЕ будет (как правило), т.к. для ее работы будет не хватать библиотек dll и информации из реестра (см. пункты 2 и 3)).

Но еще большей проблемой было то, что у разных компьютеров были свои «странности» и их нужно было знать и учитывать в программе. Например, разница двух не равных чисел возвращала ноль. В результате выражения «X=Y» и «X-Y=0» вступали в противоречие. Умельцы обходили эту проблему очень хитрыми способами, например, делали присваивание «X=(X-X)+X» перед операциями умножения и деления, чтобы избежать подобных проблем.

Инициатива создать единый стандарт для представления чисел с плавающей запятой подозрительно совпала с попытками в 1976 году

компанией Intel разработать «лучшую» арифметику для новых сопроцессоров к 8086 и i432. За разработку взялись самые известные ученые в этой области, проф. Джон Палмер и Уильям Кэхэн. Последний в своем интервью высказал мнение, что серьезность, с которой Intel разрабатывала свою арифметику, заставила другие компании объединиться и начать процесс стандартизации.

Все были настроены серьезно, ведь очень выгодно продвинуть свою архитектуру и сделать ее стандартной. Свои предложения представили компании DEC, National Superconductor, Zilog, Motorola. Производители мейнфреймов Cray и IBM наблюдали со стороны. Компания Intel, разумеется, тоже представила свою новую арифметику. Авторами предложенной спецификации стали Уильям Кэхэн, Джероми Кунен и Гарольд Стоун и их предложение сразу прозвали «К-С-S».

Практически сразу же были отброшены все предложения, кроме двух: VAX от DEC и «К-С-S» от Intel. Спецификация VAX была значительно проще, уже была реализована в компьютерах PDP-11, и было понятно, как на ней получить максимальную производительность. С другой стороны, в «К-С-S» содержалось много полезной функциональности, такой как «специальные» и «денормализованные» числа (об этом ниже).

В «К-С-S» все арифметические алгоритмы заданы строго и требуется, чтобы в реализации результат с ними совпадал. Это позволяет выводить строгие выкладки в рамках этой спецификации. Если раньше математик решал задачу численными методами и доказывал свойства решения, не было никакой гарантии, что эти свойства сохранятся в программе. Строгость арифметики «К-С-S» сделала возможным доказательство теорем, опираясь на арифметику с плавающей запятой.

Компания DEC сделала все, чтобы ее спецификацию сделали стандартом. Она даже заручилась поддержкой некоторых авторитетных ученых в том, что арифметика «К-С-S» в принципе не может достигнуть такой же производительности, как у DEC. Ирония в том, что Intel знала, как сделать свою спецификацию такой же производительной, но эти хитрости были коммерческой тайной. Если бы Intel не уступила и не открыла часть секретов, она бы не смогла сдержать натиск DEC.

➤ Подробнее о баталиях при стандартизации смотрите в интервью профессора Кэхэна, а мы рассмотрим, как выглядит представление чисел с плавающей запятой сейчас.

Представление чисел с плавающей запятой сегодня. Разработчики «К-С-S» победили и теперь их детище воплотилось в стандарт IEEE754. Числа с плавающей запятой в нем представлены в виде знака (s), мантиссы (M) и порядка (E) следующим образом:

$$(-1)^s \times 1.M \times 2^E.$$

Замечание. В новом стандарте IEEE754-2008 кроме чисел с основанием 2 присутствуют числа с основанием 10, так называемые *десятичные* (decimal) числа с плавающей запятой.

Чтобы не загромождаться чрезмерной информацией, которую можно найти в Википедии, рассмотрим только один тип данных, с одинарной точностью (float). Числа с половинной, двойной и расширенной точностью обладают теми же особенностями, но имеют другой диапазон порядка и мантиссы. В числах одинарной точности (float/single) порядок состоит из 8 бит, а мантисса – из 23. Эффективный порядок определяется как E-127. Например, число 0,15625 будет записано в памяти как:



В этом примере:

- Знак $s=0$ (положительное число).
- Порядок $E=01111100_2-127_{10} = -3$.
- Мантисса $M = 1.01_2$ (первая единица не явная).
- В результате наше число $F = 1.01_2 \cdot 2^{-3} = 2^{-3} + 2^{-5} = 0,125 + 0,03125 = 0,15625$.

Здесь мы имеем дело с двоичным представлением числа «101» со сдвигом запятой на несколько разрядов влево. $1,01$ – это двоичное представление, означающее $1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$. Сдвинув запятую на три позиции влево получим $1,01 \cdot 2^{-3} = 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5} = 1 \times 0,125 + 0 \times 0,0625 + 1 \times 0,03125 = 0,125 + 0,03125 = 0,15625$.

Специальные числа: ноль, бесконечность и неопределенность. В IEEE754 число «0» представляется значением с порядком, равным $E=E_{\min}-1$ (для single это -127) и нулевой мантиссой. Введение нуля как самостоятельного числа (т.к. в нормализованном представлении нельзя представить ноль) позволило избежать многих странностей в арифметике. И хотя операции с нулем нужно обрабатывать отдельно, обычно они выполняются быстрее, чем с обычными числами.

Также в IEEE754 предусмотрено представление для специальных чисел, работа с которыми вызывает исключение. К таким числам относятся бесконечность ($\pm\infty$) и неопределенность (NaN). Эти числа позволяют вернуть адекватное значение при переполнении. Бесконечности представлены как числа с порядком $E=E_{\max}+1$ и нулевой мантиссой. Получить бесконечность можно при переполнении и при делении ненулевого числа на ноль. Бесконечность при делении разработчики определили исходя из существования пределов, когда делимое и делитель стремятся к какому-то числу. Соответственно, $x/0 = \pm\infty$ (например, $3/0 = +\infty$, а $-3/0 = -\infty$), т.к. если делимое стремится к константе, а делитель к нулю, предел равен бесконечности. При $0/0$ предел не существует, поэтому результатом будет неопределенность.

Неопределенность или NaN (от not a number) – это представление, придуманное для того, чтобы арифметическая операция могла всегда вернуть какое-то не бессмысленное значение. В IEEE754 NaN представлен как число, в котором $E=E_{\max}+1$, а мантисса не нулевая. Любая операция с NaN возвращает NaN. При желании в мантиссу можно записывать информацию, которую программа сможет интерпретировать. Стандартом это не оговорено и мантисса чаще всего игнорируется.

Как можно получить NaN? Одним из следующих способов:

- $\infty + (-\infty)$.
- $0 \times \infty$.
- $0/0$, ∞/∞ .
- $\text{sqrt}(x)$, где $x < 0$.

По определению $\text{NaN} \neq \text{NaN}$, поэтому, для проверки значения переменной нужно просто сравнить ее с собой.

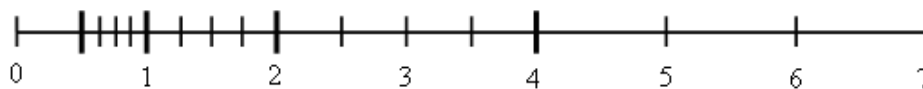
Зачем нулю знак? Любопытный читатель вероятно уже заметил, что в описанном представлении чисел с плавающей запятой существует два нуля, которые отличаются только знаком. Так, $3 \cdot (+0) = +0$, а $3 \cdot (-0) = -0$. Но при сравнении $+0 = -0$. В стандарте знак сохранили умышленно, чтобы выражения, которые в результате переполнения или потери значимости превращаются в бесконечность или в ноль, при умножении и делении все же могли представить максимально корректный результат. Например, если бы у нуля не было знака, выражение $1/(1/x) = x$ не выполнялось бы верно при $x = \pm\infty$, так как $1/\infty$ и $1/-\infty$ равны 0.

Еще один пример:

$$(+\infty/0) + \infty = +\infty, \text{ тогда как } (+\infty/-0) + \infty = \text{NaN}.$$

Чем бесконечность в данном случае лучше, чем NaN? Тем, что, если в арифметическом выражении появился NaN, результатом всего выражения всегда будет NaN. Если же в выражении встретились бесконечность, то результатом может быть ноль, бесконечность или обычное число с плавающей запятой. Например, $1/\infty = 0$.

Денормализованные числа. Понятие денормализованных (subnormal) чисел рассмотрим на простом примере. Пусть имеем нормализованное представление с длиной мантиссы $|M|=2$ бита (+ один бит нормализации) и диапазоном значений порядка $-1 \leq E \leq 2$. В этом случае получим 16 чисел:



Крупными штрихами показаны числа с мантиссой, равной 1,00. Видно, что расстояние от нуля до ближайшего числа ($0 - 0,5$) больше, чем от этого числа к следующему ($0,5 - 0,625$). Это значит, что разница двух любых чисел от 0,5 до 1 даст 0, даже если эти числа не равны. Что еще хуже, в пропасть между 0,5 и 0 попадает разница чисел, больших 1. Например, « $1,5 - 1,25 = 0$ » (см. картинку).

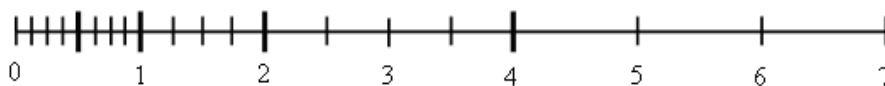
В «околонулевою яму» подпадает не каждая программа. Согласно статистике 70-х годов в среднем каждый компьютер сталкивался с такой проблемой один раз в месяц. Учитывая, что компьютеры приобретали массовость, разработчики «К-С-S» посчитали эту проблему достаточно серьезной, чтобы решать ее на аппаратном уровне. Предложенное ими решение состояло в следующем. Мы знаем, что при $E=E_{\min}-1$ (для float это «-127») и нулевой мантиссе число считается равным нулю. Если же мантисса не нулевая, то число считается не нулевым, его порядок полагается $E=E_{\min}$, причем неявный старший бит мантиссы полагается равным нулю. Такие числа называются **денормализованными**.

Строго говоря, числа с плавающей запятой теперь имеют вид:

$(-1)^s \times 1.M \times 2^E$ – если $E_{\min} \leq E \leq E_{\max}$ (нормализованные числа);

$(-1)^s \times 0.M \times 2^{E_{\min}}$ – если $E=E_{\min}-1$ (денормализованные числа).

Вернемся к примеру. Наш $E_{\min}=-1$. Введем новое значение порядка, $E=-2$, при котором числа являются денормализованными. В результате получаем новое представление чисел:



Интервал от 0 до 0,5 заполняют денормализованные числа, что дает возможность не проваливаться в 0 в рассмотренных выше примерах (0,5-0,25 и 1,5-1,25). Это сделало представление более устойчиво к ошибкам округления для чисел, близких к нулю.

Но использование денормализованного представления чисел в процессоре не дается бесплатно. Из-за того, что такие числа нужно обрабатывать по-другому во всех арифметических операциях, трудно сделать работу в такой арифметике эффективной. Это накладывает дополнительные сложности при реализации АЛУ (арифметико-логическое устройство) в процессоре. И хоть денормализованные числа очень полезны, они не являются панацеей и за округлением до нуля все равно нужно следить. Поэтому эта функциональность стала камнем преткновения при разработке стандарта и встретила самое сильное сопротивление.

Очередность чисел в IEEE754. Одна из удивительных особенностей представления чисел в формате IEEE754 состоит в том, что порядок и мантисса расположены друг за другом таким образом, что вместе образуют последовательность целых чисел $\{n\}$ для которых выполняется:

$$n < n+1 \Rightarrow F(n) < F(n+1),$$

где $F(n)$ – число с плавающей запятой, образованное от целого n , разбиением его битов на порядок и мантиссу.

Поэтому если взять положительное число с плавающей запятой, преобразовать его к целому, прибавить «1», мы получим следующее число, которое представимо в этой арифметике. На Си это можно сделать так:


```
float a=0.5;
int n = *((int*) &a);
float b = *((float*) &(&n));
printf("После %e следующее число: %e, разница (%e)\n", a, b, b-a);
```

Этот код будет работать только на архитектуре с 32-битным int.

Проблемы в арифметике с плавающей запятой. Рассмотрим особенности арифметики с плавающей запятой, к которым нужно проявить особую осторожность при программировании.

Округление. С ошибками из-за погрешностей округления в современной арифметике с плавающей запятой встретиться сложно, особенно если использовать двойную точность. Правило округления в стандарте IEEE754 говорит о том, что результат любой арифметической операции должен быть таким, как если бы он был выполнен над точными значениями и округлен до ближайшего числа, представимого в этом формате. Это требует от АЛУ дополнительных усилий и некоторые опции компилятора (такие как «-ffast-math» в gcc) могут отключить такое поведение. Особенности округления в IEEE754:

- Округление до ближайшего в стандарте сделано не так как мы привыкли. Математически показано, что если 0,5 округлять до 1 (в большую сторону), то существует набор операций, при которых ошибка округления будет возрастать до бесконечности. Поэтому в IEEE754 применяется правило округления до четного. Так, 12,5 будет округлено до 12, а 13,5 – до 14.

- Самая опасная операция с точки зрения округления в арифметике с плавающей запятой – это вычитание. При вычитании близких чисел значимые разряды могут потеряться, что может в разы увеличить относительную погрешность.

- Для многих широко распространенных математических формул математики разработали специальную форму, которая позволяет значительно уменьшить погрешность при округлении. Например, расчет формулы « $x^2 - y^2$ » лучше вычислять используя формулу « $(x-y)(x+y)$ ».

Неассоциативность арифметических операций. В арифметике с плавающей запятой правило $(a*b)*c = a*(b*c)$ не выполняется для любых арифметических операций. Например,

$$(10^{20}+1)-10^{20}=0 \neq (10^{20}-10^{20})+1=1.$$

Допустим у нас есть программа суммирования чисел.

```
double s = 0.0;
for (int i=0; i<n; i++) s = s + t[i];
```

Некоторые компиляторы по умолчанию могут переписать код для использования нескольких АЛУ одновременно (будем считать, что n делится на 2):


```
double sa[2], s;
sa[0]=sa[1]=0.0;
for (int i=0; i<n/2; i++) {
    sa[0]=sa[0]+t[i*2+0];
    sa[1]=sa[1]+t[i*2+1];
}
S=sa[0]+sa[1];
```

Так как операции суммирования не ассоциативны, эти две программы могут выдать различный результат.

Числовые константы. Нужно помнить, что не все десятичные числа имеют двоичное представление с плавающей запятой. Например, число «0,2» будет представлено как «0,200000003» в одинарной точности. Соответственно, «0,2 + 0,2 ≈ 0,4». Абсолютная погрешность в отдельном случае может и не высока, но если использовать такую константу в цикле, можем получить накопленную погрешность.

Выбор минимума из двух значений. Допустим из двух значений нам нужно выбрать минимальное. В Си это можно сделать одним из следующих способов:

1. $x < y$? x: y
2. $x \leq y$? x: y
3. $x > y$? y: x
4. $x \geq y$? y: x

Часто компилятор считает их эквивалентными и всегда использует первый вариант, так как он выполняется за одну инструкцию процессора. Но если мы учтем ±0 и NaN, эти операции никак не эквивалентны:

x	y	$x < y$? x: y	$x \leq y$? x: y	$x > y$? y: x	$x \geq y$? y: x
+0	-0	-0	+0	+0	-0
NaN	1	1	1	NaN	NaN

Сравнение чисел. Очень распространенная ошибка при работе с float-ами возникает при проверке на равенство. Например,

```
float fValue = 0.2;
if (fValue == 0.2) DoStuff();
```

Ошибка здесь, во-первых, в том, что 0,2 не имеет точного двоичного представления, а во-вторых 0,2 – это константа двойной точности, а переменная fValue – одинарной, и никакой гарантии о поведении этого сравнения нет.

Лучший, но все равно ошибочный способ, это сравнивать разницу с допустимой абсолютной погрешностью:

```
if (fabs(fValue - fExpected) < 0.0001) DoStuff(); // fValue=fExpected?
```

Недостаток такого подхода в том, что погрешность представления числа увеличивается с ростом самого этого числа. Так, если программа ожидает «10000», то приведенное равенство не будет выполняться для ближайшего соседнего числа (10000,000977). Это особенно актуально, если в программе имеется преобразование из одинарной точности в двойную.

Выбрать правильную процедуру сравнения сложно, проблематику этого вопроса можно найти в статье [Брюса Доусона](#) (Random ASCII – tech blog of Bruce Dawson). В ней предлагается сравнивать числа с плавающей запятой преобразованием к целочисленной переменной. Это лучший, хотя и не портатбельный способ:

```
bool AlmostEqual2sComplement(float A, float B, int maxUlp) {
    // maxUlp не должен быть отрицательным и не слишком большим, чтобы
    // NaN не был равен ни одному числу
    assert(maxUlp > 0 && maxUlp < 4 * 1024 * 1024);
    int aInt = *(int*)&A;
    // Уберем знак в aInt, если есть, чтобы получить правильно упорядоченную последовательность
    if (aInt < 0) aInt = 0x80000000 - aInt;
    //aInt &= 0x7fffffff; //(см. комментарий пользователя Vayun)
    // Аналогично для bInt
    int bInt = *(int*)&B;
    if (bInt < 0) bInt = 0x80000000 - bInt;
    /*aInt &= 0x7fffffff;*/
    unsigned int intDiff = abs(aInt - bInt); /*(см. комментарий пользователя Vayun)*/
    if (intDiff <= maxUlp)
        return true;
    return false;
}
```

В этой программе maxUlp (от Units-In-Last-Place) – это максимальное количество чисел с плавающей запятой, которое может лежать между проверяемым и ожидаемым значением. Другой смысл этой переменной – это количество двоичных разрядов (начиная с младшего) в сравниваемых числах разрешается упустить. Например, maxUlp=16, означает, что младшие 4 бита ($\log_2 16$) могут не совпадать, а числа все равно будут считаться равными. При этом, при сравнении с числом 10000 абсолютная погрешность будет равна 0,0146, а при сравнении с 0.001, погрешность будет менее 0.00000001 (10^{-8}).

Проверка полноты поддержки IEEE754. Можно думать, что если процессоры полностью соответствуют стандарту IEEE754, то любая программа, использующая стандартные типы данных (такие как float/double в Си), будет выдавать один и тот же результат на разных компьютерах? Это мнение ошибочно. На портатбельность и соответствие стандарту влияет компилятор и опции оптимизации. Уильям Кэхэн написал программу на Си (есть версия и для Фортрана), которая позволяет проверить удовлетворяет ли связка «архитектура+компилятор+опции» IEEE754. Называется она «Floating

point paranoia» и ее исходные тексты доступны для скачивания. Аналогичная программа доступна для GPU. Так, например, компилятор Intel (icc) по умолчанию использует «расслабленную» модель IEEE754, и в результате не все тесты выполняются. Опция «-fp-model precise» позволяет компилировать программу с точным соответствием стандарту. В компиляторе GCC есть опция «-ffast-math», использование которой приводит к несоответствию IEEE754.

Сейчас арифметика с плавающей запятой почти совершенна. Практически всегда наивный подход сработает, и программа, не учитывающая все ее особенности, выдаст правильный результат, а описанные подводные камни касаются только экзотических случаев. Но нужно всегда оставаться бдительным: в таком вопросе как компьютерная математика легко совершить ошибку.

Вычислительная задача – это одна из трех типов математических задач решение которой необходимо получить численно. Принципиальные типы математических задач:

- прямая (непосредственное вычисление решения);
- обратная (вычисление параметров математической модели);
- задача идентификации (идентификация математической модели).

Модель – идеализированное представление, обладающее определенной (достаточной для решения задачи моделирования) степенью адекватности (подобия), исследуемой системы (объекта, процесса).

Символьные (аналитические) методы. Аналитические методы решения математических задач – это методы получения решения в буквенно-символьном виде включая различные математические преобразования исходной задачи. Аналитические подходы решения задач, как правило:

- не эффективны по быстродействию получения решения;
- имеют более низкую точность полученного решения;
- не могут быть применены в автономных технических системах;
- не применимы для решения современных сложных технических задач.

Численные методы (изучаемые в курсе дисциплины Методы вычислений) – это алгоритмы и их реализации для решения математических задач, когда получаемый результат получается в виде, как правило, набора чисел. С развитием вычислительной техники, как правило, под получаемым набором чисел понимают набор цифровых данных, хранящихся в том или ином виде в памяти вычислителя.

Численные методы – это не только альтернатива символьным (аналитическим) подходам при решении математических задач, но и в подавляющем большинстве случаев – единственный реализуемый подход решения этих задач.

Аналитические подходы решения задач, как правило:

- не эффективны по быстродействию получения решения;
- имеют более низкую точность полученного решения;

- не могут быть применены в автономных технических системах.

Требования, предъявляемые к численным методам:

по требованиям к разработке алгоритмов:

- достаточный уровень быстродействия (численное решение задачи должно завершаться за ограниченное время, отведенное на это решение обобщенной задачей);

- достижимость (устойчивость/сходимость) решения (численное решение задачи должно быть близким к ожидаемому с теоретической точкой зрения и желательно быть единственным);

- минимальность ошибки (погрешности) вычислений (вычислительная ошибка, с которой получается достижимое итоговое решение должна быть сведена к минимуму);

по оптимальности реализации алгоритма:

- минимальность временной вычислительной сложности (время, затрачиваемое на вычисления должно сводиться к минимуму);

- минимальность пространственной вычислительной сложности (использование вычислительных ресурсов – объема используемой ОЗУ или процессорного времени должно сводиться к минимуму).

Классификация численных методов:

- методы эквивалентных преобразований;
- методы аппроксимации;
- прямые методы;
- итерационные методы;
- стохастические методы.

Характеристики вычислительных задач и методов:

- погрешности получаемого решения;
- корректность и обусловленность задачи;
- устойчивость и/или сходимость решения.

Виды численных методов по решаемым задачам:

- методы векторно-матричных преобразований и разложений;
- методы решения линейных и нелинейных систем уравнений;
- методы интерполяции функций;
- методы интегрирования функций;
- методы дифференцирования функций;
- методы решения задач оптимизации (непосредственный численный поиск экстремумов);
- методы решения задачи Коши (методы интегрирования систем дифференциальных уравнений).

Методы эквивалентных преобразований. В этих методах исходная задача (ее постановка, начальные данные, математическое описание) заменяется на другую, имеющую тоже решение. Как правило, метод используется тогда, когда в исходной постановке задача не может быть решена, либо в новой постановке процесс получения решения более

эффективен. В данном случае речь не идет об уменьшении погрешности вычислений, хотя численно, де-факто, это возможно. В большей степени метод свойственен для аналитических подходов, но для численных явным примером может служить замена исходной задачи вычисления обратной матрицы на решение этой же задачи, но с использованием матричных разложений.

Методы аппроксимации. В этих методах исходная задача (ее постановка, начальные данные, математическое описание) заменяется (аппроксимируется) на другую, имеющую решение близкое к исходной. Неточность аппроксимации проявляется в появлении в результатах вычислений ошибки аппроксимации. Если ошибка аппроксимации стремится к нулю при стремлении параметров метода аппроксимации к некоторому значению, то говорят, что аппроксимация сходится. Примером метода аппроксимации является вычисление определенного интеграла, когда подынтегральная функция заменяется на последовательность более простых функций. Аналогично происходит и при решении задач интерполяции и решении задачи Коши. Отдельным классом методов аппроксимации можно считать задачи линеаризации нелинейных функционалов.

Прямые методы. В этих методах решение задачи получается за конечное число вычислительных операций, которое в общем случае зависит от размера входных и данных. Большинство прямых методов было разработано до эпохи компьютеризации, и несмотря на то, что их алгоритм достаточно легко программируется, как правило, эти методы при решении практических задач практически не используются, т.к. очень чувствительны к ошибкам округления. Например, решение СЛАУ методом Крамера для любой матрицы 3×3 потребует одинакового числа операций.

Итерационные методы. В этих методах решение задачи получается с помощью построения итерационного процесса, т.е. некоторого итерационного процесса приближения к истинному решению, где количество итераций, в теоретическом смысле, не фиксировано. Итерационный процесс, в зависимости от решаемой задачи может завершаться при выполнении одного или нескольких условий, например:

- количество итераций превысило заданное максимальное значение;
- разность решений, полученных на текущей и предыдущей итерациях меньше некоторого заданного значения;
- разность некоторых функционалов, полученных на текущей и предыдущей итерациях меньше некоторого заданного значения.

Для итерационных методов важно свойство сходимости метода к истинному решению. Сходимость зависит от многих факторов, но, как правило, связывается со значениями начальных условий. Множество начальных условий называется областью сходимости, если для них метод сходится.

Стохастические методы. В этих методах решение задачи получается с помощью моделирования случайных экспериментов, как правило,

многократного, и построении (вычислении) статистических оценок. Теоретически стохастические методы могут выдавать решение с большой точностью, но при этом затраченное время становится слишком большим. Однако для ряда задач, особенно при наличии случайных составляющих в постановке задачи, стохастические методы являются единственно применимыми. Примером стохастических методов могут служить методы имитационного моделирования Монте-Карло для вычисления интегралов сложных фигур. Эти методы, по сути, представляют собой концепцию многократного повторения одного и того же процесса, но с различной реализацией стохастических процессов.

Корректность вычислительной задачи. Анализ важнейших требований, предъявляемых к различным прикладным задачам, приводит к понятию корректности математической задачи.

Вычислительная задача называется корректной, если выполняются следующие три требования:

а) решение этой задачи $y \in Y$ существует при любых $x \in X$;

б) это решение единственно;

в) решение устойчиво по отношению к малым возмущениям входных данных.

Если не выполнено хотя бы одно из условий, задача называется некорректной.

Существование решения – естественное требование. Отсутствие решения свидетельствует либо о непригодности принятой математической модели, либо о неправильной постановке самой математической модели.

Неединственность – неприятное свойство вычислительной задачи. Ее причиной, помимо уже перечисленных условий, может быть естественное свойство решаемой задачи. Неединственность может быть ликвидирована введением некоторых дополнительных ограничений на решение. Иногда за решение задачи принимается множество решений.

Решение y называется устойчивым по входным данным x , если y зависит от x непрерывным образом. Строго формальное определение устойчивости решения похоже на определения предела функции: если для любого $\varepsilon > 0$ найдется такое $\delta = \delta(\varepsilon) > 0$, что для любого $x_i \in X$, удовлетворяющего условию $\Delta(x_i) < \delta$, найдется соответствующее $y_i \in Y$, такое, что $\Delta(y_i) < \varepsilon$, решение будет устойчиво. Иными словами, для устойчивости вычислительной задачи ее решение теоретически можно найти со сколь угодно высокой точностью ε , если обеспечена высокая точность δ исходных данных. Неустойчивость решения y означает, что $\exists \varepsilon_0 > 0$, что какое бы малое $\delta > 0$ ни было задано, найдутся такие данные x_i , что $\Delta(x_i) < \delta$, но $\Delta(y_i) > \varepsilon_0$.

Обусловленность вычислительной задачи. Под обусловленностью вычислительной задачи понимают ее чувствительность к малым погрешностям входных данных. Задачу называют хорошо обусловленной,

если малым погрешностям входных данных отвечают малые погрешности решения, и плохо обусловленной, если возможны сильные изменения решения. Для измерения количественной стороны обусловленности используют число обусловленности. Грубо говоря, это коэффициент возможного возрастания погрешностей в решении по отношению к вызвавшим их погрешностям входных данных, т.е. $\Delta(y_i) = v_{\Delta} \cdot \Delta(x_i)$, где v_{Δ} – абсолютное число обусловленности. Если $\delta(y_i) = v_{\delta} \cdot \delta(x_i)$, то v_{δ} – относительное число обусловленности. Для плохо обусловленной задачи $v \gg 1$.

Вычислительный алгоритм – точное предписание действий над входными данными, задающие вычислительный процесс, направленный на преобразование входных данных в полностью определяемый этими данными результат.

Вычислительный алгоритм складывается из 2-х частей:

1. Абстрактный – записанный математическими знаками.
2. Программный – программа, записанная на языке программирования.

К вычислительным алгоритмам в целом предъявляются требования корректности и хорошей обусловленности.

Требования к абстрактным вычислительным алгоритмам:

1. Экономичность (число операций).
2. Точность (обеспечение приемлемой погрешности).
3. Экономия памяти (большие массивы данных).
4. Простота алгоритма.

Требования к программным реализациям:

1. Надежность (отсутствие ошибок, получение нужного результата).
2. Работоспособность (способность программы выявлять недопустимые данные и обнаруживать критические для алгоритма ситуации).
3. Переходимость (возможность работы на различных платформах и операционных системах без изменений).
4. Поддерживаемость (легкость модификации программы).
5. Простота в использовании.

Схема:

- 1) Постановка задачи:
 - 1.1. Формулировка задачи, выбор параметров.
 - 1.2. Определение цели и критериев, качеств, описание задачи.
- 2) Составление математического описания:
 - 2.1. Аналитические методы.
 - 2.2. Экспериментальные методы.
 - 2.3. Экспериментально-аналитические методы.
- 3) Составление алгоритма и «написание» программы:
 - 3.1. Выбор численных методов.
 - 3.2. Составление алгоритма.
 - 3.3. Программирование.
 - 3.4. Отладка программы.

4) Установление адекватности модели объекту.

5) Использование модели.

Катастрофическая потеря точности. Иногда короткая последовательность вычислений приводит от исходных данных, известных с высокой точностью, к результату, содержащему недопустимо мало верных цифр или вообще не имеющему ни одной верной цифры. В этом случае принято говорить о катастрофической потере точности.

Замечание. Не всегда катастрофическая потеря точности в промежуточных вычислениях действительно является катастрофой. Все зависит от того, как в дальнейшем используется результат.

Лекция 2. Обзор инструментальных программных средств (1ч. ЛК)

1. Обзор инструментальных программных средств пакетов прикладных программ Excel, Mathcad, Maple, Mathematica.
2. Алгоритмический язык Python при применении численных методов.

В последние двадцать лет возникло и получило бурное развитие новое фундаментальное научное направление – компьютерная математика, которая зародилась на стыке математики и информатики. Первыми серьезными средствами для автоматизированного выполнения массовых научно-технических расчетов были программируемые микрокалькуляторы. Предвестниками систем компьютерной математики стали специализированные программы для математических численных расчетов, работающие в среде Microsoft MS-DOS. Это – Eureka, Mercury, первые версии системы MathCAD и MatLAB под операционную систему MS-DOS. Вслед за этим на основе достижений компьютерной математики появились новейшие программы символьной математики или компьютерной алгебры. Среди них особенно большую известность получили системы MathCAD под Windows, Derive, Mathematica и Maple. Созданные для проведения символьных (аналитических) преобразований математических выражений, эти системы были в поразительно короткое время доведены до уровня, позволяющего резко облегчить, а подчас и заменить труд самой почитаемой научной элиты мира – математиков-теоретиков и аналитиков.

В последнее время появившиеся тенденции к объединению передовых разработчиков математических систем привели к тому, что лидеры в данной области, такие, как Maple, MathCAD, Mathematica, MatLAB, интегрировались и теперь стоят приблизительно на одной ступени развития. Различия между возможностями этих систем практически стерлись и сохранились лишь в каких-то базовых элементах и визуализации программных документов.

Системы компьютерной математики класса Maple были созданы корпорацией Waterloo Maple, как система компьютерной алгебры с расширенными возможностями в области символьных вычислений. Система содержит средства для выполнения быстрых численных расчетов, лежащих в основе математического моделирования различных явлений окружающего мира, систем и устройств различного назначения. Все это сочетается с новейшими и весьма эффективными средствами визуализации вычислений.

Maple – типичная интегрированная система. Она объединяет в себе мощный язык программирования (он же язык для интерактивного общения с системой), редактор для подготовки и редактирования документов и программ, многооконный пользовательский интерфейс с возможностью работы в диалоговом режиме, справочную систему с тысячей примеров, ядро алгоритмов и правил преобразования математических выражений, численный и символьный процессоры, систему диагностики, библиотеки встроенных и дополнительных функций, пакеты функций сторонних

производителей и поддержку некоторых других языков программирования и программ. Ко всем этим средствам имеется полный доступ прямо из окна системы. Она реализована на больших ЭВМ, ПК класса IBM PC, Macintosh и др.

Ядро системы Maple используется целым рядом систем компьютерной математики, например, таких, как MatLAB и MathCAD.

Система Mathematica разработана фирмой Wolfram Research. Основная идея разработчиков – объединить все известные понятия и методы математики в единую универсальную систему, способную функционировать на любой вычислительной платформе. Эта система дает возможность решать большое количество достаточно сложных задач, не вдаваясь в тонкости программирования, что привело к её широкому использованию в таких науках, как физика, биология, экономика и т.д.

Система Mathematica состоит из двух частей – ядра, которое, собственно, и проводит вычисления, выполняя заданные команды, и интерфейсного процессора, фактически задающего внешнее оформление и характер взаимодействия с пользователем и программой. Пользователь записывает все выкладки в основном рабочем документе программы – notebook («записная книжка»). Внешний вид рабочего документа на экране монитора в большей или меньшей степени зависит от типа платформы (Windows, Macintosh, Unix) и определяется интерфейсным процессором, своим для каждой платформы.

Notebook— полностью интерактивный документ, содержащий текст, таблицы, графики, вычисления и другие элементы. Документы могут быть представлены на экране с различным полиграфическим исполнением, могут включать в себя всевозможные математические или специальные символы. Однако внутреннее представление документов, с которыми работает ядро программы, всегда – не форматируемый текст в виде печатаемых (printable) 7-битовых ASCII-символов. Это позволяет легко переписывать документы с одной платформы на другую.

Язык программирования Mathematica рассматривается как единый универсальный язык программирования и математики. Он является беспрецедентно гибким и интуитивным языком, содержит уникальную комбинацию математических и вычислительных обозначений. Основная унифицирующая идея языка – рассматривать любой объект как символьное выражение (это касается не только привычных структур данных, но и графиков, звуков и даже ячеек, и даже самого документа). Это делает легким добавление к системе Mathematica новых конструкций языка и изменения интерфейса программы.

Язык включает в себя представление широко известных развитых методов программирования компьютерной науки и добавляет множество новых. На языке программирования можно всегда написать программу в наиболее естественном виде, так как Mathematica включает в себя множество парадигм программирования: процедурное программирование; основанное

на операциях со списками (list-based); основанное на операциях со строками (string-based); функциональное; объектно-ориентированное; программирование, задающее правила преобразования выражений («правила переписывания термов»).

В виде встроенных функций реализовано большое множество высокоинтеллектуальных математических операций. В состав Mathematica входят стандартные дополнения, включающие в себя подпрограммы (пакеты), которые значительно расширяют функциональные возможности системы в таких областях, как алгебра, аналитические и численные расчеты, графика, дискретная математика, геометрия, теория чисел, статистика и др.

Система MathCAD разработана фирмой MathSoft. Она существенно отличается от аналогичных прикладных систем, таких, как MatLAB, Mathematica, Maple, тем, что является единственной прикладной системой, в которой описания математических задач и их решений задаются с помощью обычных в математике символов, формул и операторов, а документ MathCAD выглядит как страницы учебника или научной статьи.

MathCAD – это популярная система компьютерной математики, предназначенная для решения математических задач в самых разных областях науки, техники и образования. Ранние версии MathCAD вообще не имели средств обычного программирования, а имели лишь средства визуально-ориентированного программирования в виде шаблонов математических операций, из которых составлялись математические выражения. Возможность задания программных модулей появилась, начиная с версии MathCAD PLUS 6.0. Программные модули, в сущности, являются функциями, но описанными с применением программных средств.

Последние версии MathCAD допускают применение внешних расширений: электронных книг, пакетов расширений и библиотек. Внешние библиотеки чаще всего являются справочниками. Пакеты расширений и сопровождающие их электронные книги имеют различное назначение и ориентированы, как правило, на выполнение узкоспециальных вычислений. Пакеты расширений предоставляют следующие возможности: осуществлять вейвлет преобразования, анализ временных рядов, финансовые расчеты, обработку сигналов и изображений, расчеты по прикладной статистике, механике, дифференциальным уравнениям и т.д. Существенными дополнениями к системе являются пакеты расширения графических возможностей, численных методов.

В MathCAD возможна интеграция с текстовым процессором Word, электронными таблицами Excel, графической системой Axum, пакетом научной и инженерной графики Visio, с пакетом SmartSketch LE, который позволяет включать в состав документов MathCAD довольно сложные конструкторские чертежи, с пакетом VisSim – моделирующей программой, в которой объекты задаются блоками, между которыми указываются дополнительные связи.

Привычный вид математических формул, встроенный язык программирования, широкие возможности в отображении графической информации, представление текстовых комментариев с использованием любых символов, доступных в Windows, возможность проводить расчеты любой сложности и готовить документы высокого качества - все это характеризует систему MathCAD.

Табличный процессор EXCEL входит в самый популярный пакет автоматизации офисной деятельности Microsoft Office. EXCEL – одна из самых мощных и гибких систем обработки электронных таблиц. Эта система может работать не только с двумерными, но и с трехмерными таблицами, представленными листами с двумерными таблицами.

EXCEL широко используется для подготовки прекрасно иллюстрированных финансово-экономических и других документов. Этот процессор содержит сотни математических и экономических функций, что позволяет решать множество задач в области естественных и технических наук.

Возможности системы EXCEL можно использовать для анализа внешних данных, представленных в Microsoft FoxPro, Access, Paradox, dBASE, SQL Server, а также базы данных сторонних производителей, поддерживающих технологию OLE (Object Linking and Embedding – связывания и внедрения объектов).

В системе EXCEL можно создавать макросы и приложения с помощью Visual Basic for Applications (VBA). Visual Basic for Applications – это среда разработки приложений на базе программ, входящих в пакет Microsoft Office. Одно из главных достоинств языка Visual Basic for Applications заключается в том, что созданные средствами EXCEL VBA-макросы можно без труда использовать в других программах фирмы Microsoft.

Рекомендуемый литературный источник по практическому применению численных методов при использовании алгоритмического языка Python доступен по ссылке <https://djvu.online/file/scNn74KtRwjxI> (Вабищевич, П.Н. Численные методы: вычислительный практикум / П.Н. Вабищевич. – М.: Книжный дом «ЛИБРОКОМ», 2010. – 320 с.). Здесь рассматриваются прямые и итерационные методы линейной алгебры, системы нелинейных уравнений, задачи минимизации функций, задачи интерполирования функций, численного интегрирования, задачи с начальными данными для обыкновенных уравнений. В книге на уровне пользователя применяются специализированные математические пакеты, на уровне разработчика проводится программирование базовых алгоритмов численного анализа. Часть материала посвящена информации о программном обеспечении, кратко описаны основные элементы языка Python и используемые пакеты.

Лекция 3. Обусловленность задачи решения систем линейных алгебраических уравнений (2ч. УСР)

1. Нормы векторов и матриц.
2. Обусловленность задачи решения СЛАУ. Число обусловленности.

Материал для изучения находится в файле «Лекции УСР.docx».

ВМИП ЧМ Лекции ПРОИ

Лекция 4. Прямые методы решения систем линейных алгебраических уравнений (4ч. – 2ч. ЛК+2ч. ЛР)

1. Классификация уравнений и систем уравнений. Система линейных алгебраических уравнений (СЛАУ) и задачи, возникающие при решении СЛАУ.

2. Прямые методы решения СЛАУ. Метод Гаусса: основная идея и схемы реализации (схема единственного деления и с выбором главных элементов). Алгоритмизация метода Гаусса.

3. LU-разложение матрицы и его использование для решения СЛАУ, вычисление определителя и нахождения обратной матрицы.

4. Метод прогонки. Алгоритм и трудоемкость метода.

К системам линейных алгебраических уравнений сводятся многие задачи вычислительной математики. Из курса высшей алгебры известно, что существуют различные способы решения линейных систем, однако ещё и сейчас трудно указать способы, наиболее эффективные с точки зрения скорости получения решения с достаточной степенью точности, а при использовании ЭВМ – при требовании минимального объема памяти. Известное правило Крамера в этом смысле не выгодно, поскольку этот способ решения линейной системы с n неизвестными приводит к вычислению $n + 1$ определителей порядка n , что представляет собой очень трудоёмкую операцию при сколько-нибудь большом числе n (порядка $n^2 n!$ умножений и делений). Следовательно, для решения больших систем необходимо выбрать другой способ вычисления неизвестных и сделать его менее трудоёмким.

Не останавливаясь на вопросах исследования систем линейных уравнений, в дальнейшем будем предполагать, что система имеет единственное решение. Поэтому основной задачей будет изучение универсальных вычислительных алгоритмов, используемых для нахождения единственного решения системы линейных уравнений, когда число переменных совпадает с числом уравнений.

Используемые практические методы решения систем линейных алгебраических уравнений можно разделить на две группы:

1) **точные методы;**

2) **итерационные (приближенные) методы.**

Точными являются такие **методы**, которые позволяют получить решение системы после выполнения конечного числа арифметических операций над коэффициентами системы и их свободными членами. Причем решение получится точным только тогда, когда коэффициенты и правые части системы известны точно и все арифметические действия над ними выполняются без округлений. Чаще всего эти методы реализуются в два этапа. На первом этапе преобразуют систему к одному из простых видов. На втором решают упрощенную систему и получают значения неизвестных. Из точных методов рассмотрим метод Гаусса, квадратного корня, Холецкого.

Однако на практике даже этими методами не всегда удается получить точное решение, ибо в ЭВМ точные коэффициенты представляются приближенно с некоторой погрешностью, а в процессе вычислений необходимо проводить округление чисел.

Итерационными являются **методы**, позволяющие получать решение системы с заданной точностью путем сходящихся бесконечных процессов. Из приближенных методов рассмотрим ниже метод итераций, Зейделя.

Особое место среди них занимают вероятностные методы. Отметим, что классы задач, для решения которых обычно применяются методы вышеуказанных групп, можно условно назвать классами задач с малым, средним, большим числом неизвестных.

В настоящее время точные методы целесообразно применять, когда порядок системы $n \leq 10^3$, и итерационные, если $n \leq 10^6$.

Методы последовательного исключения неизвестных. Пусть дана система n линейных алгебраических уравнений с n неизвестными

$$\left. \begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ \dots &\dots \dots \dots \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n \end{aligned} \right\} \quad (4.1)$$

или в матричной форме

$$Ax = b, \quad (4.2)$$

где $A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$ – матрица коэффициентов,

$b = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$, $x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$ – столбец свободных членов и столбец неизвестных

соответственно.

Если матрица A неособенная, т.е.

$$\det A = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} \neq 0,$$

то система (4.1) имеет единственное решение.

Метод решения задачи относится к классу точных, если в предположении отсутствия округления он дает точное решение задачи после

конечного числа арифметических и логических операций. Для полностью заполненной матрицы системы при использовании точных методов требуемое число операций имеет порядок n^3 . Поэтому необходимо, чтобы такой порядок числа операций был приемлем для данной ЭВМ. Другие ограничения определяются структурой и объемом памяти ЭВМ. Наиболее известными из точных методов решения СЛАУ являются **методы Гаусса**.

Метод Гаусса (схема единственного деления, с ведущим элементом). Метод Гаусса и его модификации основаны на приведении с помощью элементарных преобразований исходной системы к системе верхней треугольной или диагональной матрице.

В схеме единственного деления на каждом шаге строка делится на элемент, стоящий на главной диагонали (ведущий элемент), и исключаются элементы под главной диагональю. Предположим, что $A^{(0)} = A$, $b^{(0)} = b$ и $k-1$ шагов метода уже сделаны. Тогда на k -ом шаге расчетные формулы имеют вид

$$a_{kj}^{(k)} = \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}, j = \overline{k, n}, \quad (4.3)$$

$$b_k^{(k)} = \frac{b_k^{(k-1)}}{a_{kk}^{(k-1)}}, \quad (4.4)$$

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{kj}^{(k)} \cdot a_{ik}^{(k-1)}, \quad (4.5)$$

$$b_i^{(k)} = b_i^{(k-1)} - b_k^{(k)} \cdot a_{ik}^{(k-1)}, j = \overline{k, n}, i = \overline{k+1, n}. \quad (4.6)$$

После n -го шага матрица системы принимает вид

$$\begin{pmatrix} 1 & a_{12}^{(n)} & a_{13}^{(n)} & \dots & a_{1n}^{(n)} \\ 0 & 1 & a_{23}^{(n)} & \dots & a_{2n}^{(n)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}. \quad (4.7)$$

Процесс приведения матрицы исходной системы к системе с верхней диагональной матрицей называется прямым ходом метода Гаусса (формулы (4.3)–(4.6)), а процесс получения значений неизвестных – обратным ходом (формулы (4.8)). Неизвестные из преобразованной системы находятся по формулам

$$x_n = b_n^{(n)}, x_i = b_i^{(n)} - \sum_{j=i+1}^n a_{ij}^{(n)} x_j, i = n-1, n-2, \dots, 1. \quad (4.8)$$

Число арифметических операций (если подсчитывать только число умножений и делений, т.к. выполнение этих операций на ЭВМ требует гораздо больше времени, чем выполнение сложений и вычитаний), необходимых для решения системы (4.1) по схеме единственного деления:

– для осуществления прямого хода метода Гаусса необходимо выполнить

$$\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} = \frac{n(n+1)(n+2)}{3}$$

действий.

– для осуществления обратного хода метода Гаусса по формулам (4.8) необходимо выполнить $\sum_{k=1}^{n-1} (n-k) = \frac{n(n-1)}{2}$ умножений.

В результате для реализации схемы единственного деления метода Гаусса необходимо выполнить

$$\frac{n(n+1)(n+2)}{3} + \frac{n(n-1)}{2} = \frac{n(2n^2 + 9n + 1)}{6}$$

операций умножения и деления.

Итак, для реализации метода Гаусса требуется примерно $\frac{1}{3}n^3$ арифметических операций, причем большинство из них приходится на прямой ход.

Ограничение метода единственного деления заключается в том, что ведущие элементы на k -ом шаге исключения не равны нулю, т.е. $a_{kk}^{k-1} \neq 0$.

Но если ведущий элемент близок к нулю, то в процессе вычисления может накапливаться погрешность. В этом случае на каждом шаге исключают не x_k , а x_j (при $j \neq k$). Такой подход называется методом выбора главного элемента. Для этого выбирают неизвестные x_j с наибольшим по абсолютной величине коэффициентом либо в строке, либо в столбце, либо во всей матрице.

Связь метода Гаусса с разложением матрицы на множители. Теорема об LU-разложении. Пусть дана система $Ax = b$ (4.2), которая при прямом ходе преобразуется в эквивалентную систему с матрицей (4.7) – верхней треугольной матрицей с единицами на главной диагонали. Т.е. после преобразования исходной системы по схеме единственного деления получаем систему вида $Cx = b^{(n)}$, где матрица C это матрица (4.7). Подставляя в (4.1) выражение для $b^{(n)}$ в виде $b^{(n)} = B^{-1}b$ приходим к уравнению $Cx = B^{-1}b$ или, что то же самое, к уравнению $BCx = b$. Сопоставляя последнюю систему с системой (4.1) приходим к выводу, что при применении метода Гаусса матрица A системы есть произведение нижней треугольной матрицы B , у которой элементы на главной диагонали не равны нулю на верхнюю треугольную матрицу C с единичной главной диагональю, т.е. $A = B \cdot C$.

Таким образом, если задана матрица A и вектор b , то в методе Гаусса сначала производится разложение этой матрицы A на произведение B и C , а затем последовательно решаются две системы:

$$By = b, \tag{4.9}$$

$$Cx = y.$$

Из последней системы находят искомый вектор x . При этом разложение матрицы A на произведение $B \cdot C$ – есть прямой ход метода Гаусса, а решение систем (4.9) – обратный ход. Обозначим нижнюю треугольную матрицу через L , верхнюю треугольную матрицу – U .

Введем обозначения: Δ_j – угловой минор порядка j матрицы A , т.е.

$$\Delta_1 = a_{11},$$

$$\Delta_2 = \det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix},$$

...

$$\Delta_n = \det(A).$$

Теорема (об LU-разложении). Пусть все угловые миноры матрицы A не равны нулю ($\Delta_j \neq 0$ для $j = \overline{1, n}$). Тогда матрицу A можно представить единственным образом в виде произведения $A = L \cdot U$.

Идея доказательства. Рассмотрим матрицу A второго порядка и будем искать разложение этой матрицы в виде L и U :

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{pmatrix} \cdot \begin{pmatrix} 1 & u_{12} \\ 0 & 1 \end{pmatrix}.$$

Сопоставляя эти два равенства, определяем элементы матриц L и U (перемножим и приравняем неизвестные). Система имеет единственное решение. Методом математической индукции сказанное можно обобщить для матрицы размерности $n \times n$.

Следствие. Метод Гаусса (схема единственного деления) можно применять только в том случае, когда угловые миноры матрицы A не равны нулю.

Метод Гаусса с выбором главного элемента. Иногда может оказаться, что система (4.1) имеет единственное решение, хотя какой-либо из главных миноров матрицы A равен нулю. Кроме того, заранее неизвестно, все ли главные миноры матрицы отличны от нуля. В этих случаях обычный метод Гаусса может оказаться непригодным так, как в процессе вычисления какой-то ведущий элемент $a_{kk}^{(k-1)}$ станет равным нулю. Кроме того, если на главной диагонали элемент $a_{kk}^{(k-1)}$ мал, то деление на этот элемент приводит к значительным ошибкам округления.

Избежать указанных трудностей позволяет **метод Гаусса с выбором главного элемента**. В этом методе исключается не следующее по порядку неизвестное, а то неизвестное, коэффициент при котором является наибольшим по модулю. При применении такого варианта метода Гаусса не будет происходить деление на нуль, если матрица коэффициентов содержит

нулевые элементы. Различают три варианта метода Гаусса с выбором главного элемента:

а) метод Гаусса с выбором главного элемента по строке: в системе на каждом шаге исключения проводится соответствующая перенумерация переменных;

б) метод Гаусса с выбором главного элемента по столбцу: на каждом шаге исключения проводится перенумерация уравнений;

в) метод Гаусса с выбором главного элемента во всей матрице системы: в этом случае проводится перенумерация и переменных и уравнений.

Метод Гаусса с выбором главного элемента позволяет уменьшить погрешность округления, но при решении плохо обусловленных систем устойчивость этого метода может оказаться недостаточной.

Вычислительная устойчивость методов решения СЛАУ. Решение системы (4.2) задается формулой $x = A^{-1}b$. Влияние ошибок округления может привести к тому, что в процессе счета будет получена система уравнений, не равносильная исходной. Возникает вопрос об устойчивости метода решения.

Пусть A и b – заданные величины, а $A + dA$ и $b + db$ – близкие к ним. Будем рассматривать dA , db и dx как дифференциалы. Тогда из формулы (4.2) имеем $dA \cdot x + A \cdot dx = db$. Откуда $dx = A^{-1}(db - dA \cdot x)$. Отсюда следует, что если элементы обратной матрицы велики, то незначительная ошибка в элементах исходной матрицы или правой части может повлечь за собой значительное изменение в решении. Поэтому при выборе метода решения системы нужно обращать внимание на условия его устойчивости.

Вычисления определителя по схеме единственного деления Гаусса. Так как определитель треугольной матрицы равен произведению диагональных элементов, то

$$\det A = \prod_{i=1}^n a_{ii}^{(i-1)} = a_{11}^{(0)} \cdot a_{22}^{(1)} \cdot \dots \cdot a_{nn}^{(n-1)},$$

где $a_{ii}^{(i-1)}$ – ведущий элемент на шаге номера $i - 1$.

Метод Гаусса с выбором главного элемента. По свойству определителей вычитание строки из строки не меняет значение определителя. При перестановке строк или столбцов матрицы определитель меняет знак. Так как определитель треугольной матрицы равен произведению диагональных элементов, то после приведения матрицы к верхней треугольной получаем, что

$$\det A = \pm \prod_{i=1}^n a_{ii}^{(i-1)},$$

где знак определителя зависит от того, четным или нечетным было суммарное количество перестановок строк или столбцов матрицы.

Обратная матрица к матрице системы – метод Гаусса. Задача нахождения обратной матрицы вплотную примыкает к задаче решения систем линейных алгебраических уравнений вида $Ax = b$. Пусть A^{-1} –

обратная матрица к матрице A . Умножим обе части последнего матричного уравнения слева на матрицу A^{-1} . В результате получим $A^{-1} \cdot Ax = A^{-1}b$. Откуда $Ex = A^{-1}b$ или $x = A^{-1}b$. Таким образом, зная обратную матрицу A^{-1} , можно определить решение системы уравнений.

Используем метод Гаусса для нахождения обратной матрицы. Пусть обратная матрица A^{-1} к матрице

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \text{ имеет вид } A^{-1} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix}.$$

Из определения обратной матрицы следует, что $A \cdot A^{-1} = E$, т. е.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

Отсюда следует, что

$$Ax_j = e_j, \quad j = 1, 2, \dots, n, \quad (4.10)$$

где x_j и e_j – j -е столбцы матриц A^{-1} и E соответственно. В столбце e_j равны нулю все компоненты, кроме j -й, которая равна 1. Таким образом, для нахождения j -го столбца обратной матрицы нужно решить систему уравнений (4.10). Решив n таких систем для $j = 1, 2, \dots, n$, найдем все столбцы x_j и, следовательно, обратную матрицу.

Поскольку при разных j матрица A системы (4.10) не меняется, исключение неизвестных при использовании метода Гаусса (прямой ход) проводится только один раз, причем сразу для всех правых частей – столбцов e_j . Затем для каждой из систем (4.10) делается обратный ход с соответствующей преобразованной правой частью.

Оценки показывают, что этот способ является экономичным и требует примерно лишь в три раза больше действий, чем при решении одной системы уравнений.

Метод прогонки – метод решения СЛАУ с трехдиагональной матрицей. Метод прогонки разработан специально для решения «трёхчленной системы» линейных алгебраических уравнений, т.е. каждое из которых содержит три соседних неизвестных.

Рассмотрим систему трехточечных уравнений

$$\begin{aligned} b_0 y_0 - c_0 y_1 &= f_0, & i = 0 \\ -a_i y_{i-1} + b_i y_i - c_i y_{i+1} &= f_i, & i = \overline{1, n-1} \\ -a_n y_{n-1} + b_n y_n &= f_n, & i = n. \end{aligned} \quad (4.11)$$

предположим, что коэффициенты отличны от нуля. Матрица этой системы является трехдиагональной и имеет вид

$$\begin{pmatrix} b_0 & -c_0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ -a_1 & b_1 & -c_1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & -a_2 & b_2 & -c_2 & \dots & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -a_{N-2} & b_{N-2} & -c_{N-2} & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -a_{N-1} & b_{N-1} & -c_{N-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & -a_N & b_N \end{pmatrix} \quad (4.12)$$

Пусть мы хотим применить к системе (4.12) метод Гаусса: из первого уравнения

$$y_0 = \frac{c_0}{b_0} y_1 + \frac{f_0}{b_0}.$$

Обозначим

$$\alpha_1 = \frac{c_0}{b_0}, \beta_1 = \frac{f_0}{b_0}. \quad (4.13)$$

Тогда

$$y_0 = \alpha_1 y_1 + \beta_1. \quad (4.14)$$

Запишем уравнение при $i = 1$ (в (4.11) подставляем $i = 1$) :

$$-a_1 y_0 + b_1 y_1 - c_1 y_2 = f_1,$$

Подставим вместо y_0 его значение из (4.14):

$$-a_1 (\alpha_1 y_1 + \beta_1) + b_1 y_1 - c_1 y_2 = f_1,$$

Выражая y_1 через y_2

$$(b_1 - a_1 \alpha_1) y_1 = c_1 y_2 + f_1 + a_1 \beta_1, \Rightarrow y_1 = \alpha_2 y_2 + \beta_2,$$

где $\alpha_2 = \frac{c_1}{b_1 - a_1 \alpha_1}, \beta_2 = \frac{f_1 + a_1 \beta_1}{b_1 - a_1 \alpha_1}$ и т.д. продолжая процесс получим

$$y_i = \alpha_{i+1} y_{i+1} + \beta_{i+1}, \quad (4.15)$$

где

$$\alpha_{i+1} = \frac{c_i}{b_i - a_i \alpha_i}, \beta_{i+1} = \frac{f_i + a_i \beta_i}{b_i - a_i \alpha_i}, \quad b_i - a_i \alpha_i \neq 0, \quad i = \overline{1, n-1}. \quad (4.16)$$

Если взять выражение (4.15) для значения $i = n-1$ и добавить последнее уравнение системы (4.11), то получим систему из 2-х уравнений относительно y_{n-1} и y_n

$$\begin{cases} y_{n-1} = \alpha_n y_n + \beta_n \\ -a_n y_{n-1} + b_n y_n = f_n \end{cases}.$$

Разрешая их относительно y_n , получим

$$y_n = \frac{f_n + a_n \beta_n}{b_n - a_n a_n} = \beta_{n+1}. \quad (4.17)$$

Описанный метод решения систем линейных алгебраических уравнений с трехдиагональной матрицей называется **методом правой прогонки**. Все вычисления как бы «прогоняются» два раза. Сначала вычисляются коэффициенты α_k и β_k – вспомогательные числа, которые называются **прогоночными коэффициентами**. Для этого используются формулы (4.13), (4.16) в порядке возрастания индекса i . При этом для вычисления значений α_1 и β_1 используется краевое условие на левом конце. Эти формулы описывают **прямой ход метода прогонки**. Формулы (4.15) и (4.17) – реализуют **обратный ход**. На первом шаге обратного хода происходит согласование полученных чисел α_n, β_n с краевым условием на правом конце отрезка, после чего последовательно получаются значения искомой функции y_i в порядке убывания индекса i .

Для решения системы (4.11) необходимо выполнить $5n + 1$ операций умножения и деления. Алгоритм метода правой прогонки будет корректен, если в формулах (4.16) отличны от нуля значения $b_i - a_i \alpha_i$. Кроме того, если все α_k по модулю больше единицы, то может произойти сильное увеличение погрешности, и, если n достаточно велико, то полученное реальное решение будет значительно отличаться от искомого решения. Если погрешность вычислений ($\Delta y_k = y_k - \tilde{y}_k$) не возрастает в процессе решения системы (4.1), то метод прогонки устойчив.

Лекция 5. Итерационные методы решения СЛАУ (4ч. – 2ч. ЛК+2ч. ЛР)

1. Особенности и преимущества применения итерационных методов.
2. Метод простой итерации: алгоритм и теоремы сходимости.
3. Метод Зейделя: алгоритм и теоремы сходимости.

Особенности и преимущества применения итерационных методов.

Численные методы решения СЛАУ $Ax = b$, как уже отмечалось в предыдущей лекции, делятся на два больших класса: **прямые** (эти методы в предположении отсутствия округлений дают точное решение СЛАУ после конечного, определяемого заранее числа арифметических операций) и **итерационные** (точное решение в общем случае получается за бесконечное число шагов).

В приложениях часто встречаются системы, элементы матриц которых вычисляются по простым формулам или алгоритмам, а потому для решения систем такого рода желательно было бы иметь такие методы, в которых элементы матриц вообще можно было бы не хранить, а генерировать их по мере необходимости. Очевидно, что рассмотренные ранее прямые методы, в частности, различные варианты метода Гаусса, не обладают таким свойством, т.к. в процессе решения элементы матрицы изменяются. Кроме того, в приложениях часто встречаются задачи, в которых матрица СЛАУ имеет слишком большой размер, что для ее хранения не хватает не только оперативной памяти, но и памяти на внешних носителях. Объем вычислений при решении таких систем прямыми методами занимает очень много времени. В этом случае желательно пользоваться такими методами, которые не меняют элементов матрицы. При этом в оперативной памяти хотелось бы хранить лишь несколько строк (или столбцов) матрицы СЛАУ. Всем перечисленным выше пожеланиям удовлетворяют итерационные методы решения СЛАУ.

Итерационные методы дают возможность найти решение системы как предел бесконечного вычислительного процесса, позволяющего по уже найденным приближениям к решению построить следующее приближение. Так как приближенное решение является вектором или матрицей, то итерационный процесс следует рассматривать как предел последовательности векторов.

Пусть в n -мерном векторном пространстве дана последовательность векторов: $x^k = (x_1^k, x_2^k, \dots, x_n^k)$, $k = 1, 2, \dots$.

Вектор $x = (x_1, x_2, \dots, x_n)$ называется пределом этой последовательности, если существует каждый из n указанных пределов:

$$\lim_{k \rightarrow \infty} x_i^k = x_i, \quad i = \overline{1, n}, \quad k = 1, 2, \dots$$

Аналогично, если дана последовательность квадратичных матриц $A_k = [a_{ij}^k]$, то матрицу $A = [a_{ij}]$ называют пределом этой

последовательности, если существует n^2 пределов $\lim_{k \rightarrow \infty} a_{ij}^k = a_{ij}$, $i, j = \overline{1, n}$, $k = 1, 2, \dots$.

Говорят, что последовательность векторов x^k сходится к вектору x по норме, если норма $\|x - x_k\| \xrightarrow{k \rightarrow \infty} 0$.

Основные особенности итерационных методов:

1. Чтобы начать итерационный процесс, нужно знать начальные приближения к решению.

2. На каждой итерации элементы матрицы СЛАУ A не меняются.

3. Самоисправляемость итерационных методов: если в точном методе ошибка в вычислении ведет к ошибке в результате, то в случае сходящегося итерационного процесса ошибка в каком-то приближении исправляется в последующих вычислениях. Для такого исправления требуется, как правило, только выполнение нескольких единообразных шагов в вычислении.

4. Объем вычислений для получения каждого последующего приближения сравним с объемом при умножении матрицы A на некоторый вектор. Чаще всего вычисление на каждой итерации и сводится к умножению матрицы на вектор.

5. Если известна структура матрицы или известны формулы для вычисления элементов матрицы, то при реализации вычислений на ЭВМ всю матрицу не обязательно хранить в памяти.

6. Условие сходимости и скорость сходимости каждого итерационного метода (процесса) существенно зависят от свойств системы уравнений, т.е. от свойств матрицы системы и от выбора начальных приближений.

7. Для систем среднего и малого размера, как правило, более предпочтительными являются прямые методы. Итерационные методы применяются, главным образом, для решения задач большого размера.

Метод простой итерации. Пусть дана система линейных алгебраических уравнений:

$$Ax = f \quad (5.1)$$

с неособенной матрицей A .

В методе простой итерации необходимо исходную систему (5.1) предварительно привести к каноническому виду

$$x = Bx + b. \quad (5.2)$$

Представим матрицу A в виде: $A = C + D$. Тогда $(C + D)x = f$ или $x = -C^{-1}Dx + C^{-1}f$, тогда $B = -C^{-1}D$, $b = C^{-1}f$, $\det C \neq 0$. Для матрицы A с диагональным преобладанием в качестве матрицы C следует взять матрицу из диагональных элементов $[a_{ii}]$, то есть $C = [a_{ii}]$. Тогда $\|B\| < 1$.

Пусть известно начальное приближение: $x^0 = (x_1^0, x_2^0, \dots, x_n^0)$ к точному решению $x = (x_1, x_2, \dots, x_n)$. Все следующие приближения определим следующим образом

$$x^{k+1} = Bx^k + b, \quad k = 1, 2, \dots, \quad (5.3)$$

где $b = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_n \end{pmatrix}$

или (5.3) в развернутом виде

$$\begin{cases} x_1^{k+1} = b_{11}x_1^k + b_{12}x_2^k + b_{13}x_3^k + \dots + b_{1n}x_n^k + \beta_1 \\ x_2^{k+1} = b_{21}x_1^k + b_{22}x_2^k + b_{23}x_3^k + \dots + b_{2n}x_n^k + \beta_2 \\ \dots \\ x_n^{k+1} = b_{n1}x_1^k + b_{n2}x_2^k + b_{n3}x_3^k + \dots + b_{nn}x_n^k + \beta_n \end{cases} \quad (5.3')$$

Если последовательность приближений сходится к некоторому предельному вектору x^* , то он будет решением системы. Действительно, если в равенстве (5.3) перейти к пределу при $k \rightarrow \infty$, считая, что $x^k \rightarrow x^*$, то в пределе получим $x^* = Bx^* + b$.

Замечание 5.1. В качестве начального вектора для итерационного процесса (5.3) ((5.3')) рекомендуется брать вектор b .

Итерационный процесс (5.3) ((5.3')) прерывают при выполнении условия

$$\|x^{k+1} - x^k\| < \varepsilon,$$

где $\varepsilon > 0$ – заданная точность, которую необходимо достигнуть при решении задачи.

Условие сходимости метода простой итерации. Для выяснения условия сходимости последовательности (x^k) докажем ряд теорем.

Теорема 5.1. Для того, чтобы последовательность приближений x^k сходилась, достаточно, чтобы все собственные значения матрицы B были по модулю меньше единицы,

$$|\lambda_i| < 1, \quad i = \overline{1, n}. \quad (5.4)$$

Доказательство. Найдем выражение приближения x^k через значение x^0

$$\begin{aligned} x^k &= Bx^{k-1} + b = B[Bx^{k-2} + b] + b = B^2x^{k-2} + (E + B)b = \dots = \\ &= B^k x^0 + (E + B + B^2 + \dots + B^{k-1})b. \end{aligned}$$

Тогда, учитывая (5.4) и определение при $|\lambda_i| < 1$ справедливо равенство

$$E + A + A^2 + \dots + A^m = (E - A)^{-1},$$

сразу следует, что при $k \rightarrow \infty$, $B^k \rightarrow 0$ и

$$(E + B + B^2 + \dots + B^{k-1}) \rightarrow (E - B^{-1}),$$

откуда $x^k \rightarrow (E - B^{-1})b = x$. Теорема доказана.

Теорема 5.2. Если требовать, чтобы последовательность x^k сходилась к x^* , при любом начальном векторе x^0 , то условие (5.4) является и необходимым.

Доказательство. Пусть для всякого начального приближения x^0 будет $x^k \rightarrow x^*$. Тогда имеем

$$x^* - x^k = (Bx^* + b) - (Bx^{k-1} + b) = B(x^* - x^{k-1}) = \dots = B^k(x^* - x^0).$$

При $k \rightarrow \infty$ разность $(x^* - x^k) \rightarrow 0$, поэтому последний член в цепи этих равенств должен стремиться к нулю, каким бы ни был вектор $(x^* - x^0)$. Откуда следует, что $B^k \rightarrow 0$. Последнее будет верно тогда, когда верно условие (5.4). Теорема доказана.

Применение теорем 5.1 и 5.2 возможно при известных собственных значениях матрицы или, по крайней мере, их границ. Определение собственных значений матрицы B часто является непростой задачей.

Рассмотрим *достаточные* признаки сходимости итерационного процесса.

Теорема 5.3. Для того, чтобы последовательность приближений x^k в методе простой итерации сходилась, достаточно, чтобы какая-либо норма матрицы B была меньше единицы.

Доказательство. Если $\|B\| < 1$, то согласно лемме о том, что модули собственных значений матрицы не превосходят любой из ее норм, то есть $|\lambda_i| < \|B\|$ и $\|B\| < 1$, то следует, что все собственные значения матрицы B по модулю $|\lambda_i| < 1$ и, согласно теореме 5.1, последовательность (x^k) сходится. Теорема доказана.

Следствием теоремы 5.3 и равенств, определяющих норму матриц, является следующая теорема.

Теорема 5.4. Последовательность x^k в методе простой итерации сходится, если для матрицы B выполняется одно из неравенств

$$1) \quad \|B\|_1 = \max_{1 \leq i \leq n} \sum_{j=1}^n |b_{ij}| < 1.$$

$$2) \quad \|B\|_2 = \max_{1 \leq j \leq n} \sum_{i=1}^n |b_{ij}| < 1.$$

$$3) \quad \|B\|_3 = \sqrt{\sum_{i,j=1}^n |b_{ij}|^2} < 1.$$

В большинстве случаев важно знать с какой скоростью x^k сходилась к x^* , и оценить погрешность $x^* - x^k$ замены точного решения системы x^* приближением x^k .

Теорема 5.5. Если какая-либо норма матрицы B , согласованная с рассмотренной нормой вектора x , меньше единицы, то верна следующая оценка погрешности приближения в методе простой итерации

$$\|x^* - x^k\| \leq \|B\|^k \|x^0\| + \frac{1}{1 - \|B\|} \|B\|^k \|b\|. \quad (5.5)$$

Доказательство. Согласно теореме 5.1, для x^k можно записать $x^k = B^k x^0 + (E + B + B^2 + \dots + B^{k-1})b$.

Так как $\|B\| < 1$, то $x^* = (E + B + B^2 + \dots)b$.

Поэтому

$$x^* - x^k = (B^k + B^{k+1} + \dots)b - B^k x^0 \quad (5.6)$$

и норма

$$\|x^* - x^k\| \leq (\|B\|^k + \|B\|^{k+1} + \dots)\|b\| + \|B\|^k \|x^0\| = \|B\|^k \|x^0\| + \frac{\|B\|^k \|b\|}{1 - \|B\|}.$$

Часто за x^0 принимают вектор b . В этом случае оценка (5.5) упрощается. Подставим в (5.6) $x^0 = b$, следовательно, получим

$$\|x^* - x^k\| \leq \frac{1}{1 - \|B\|} \|B\|^{k+1} \|b\|. \quad (5.7)$$

Теорема доказана.

Замечание 5.2. Если задана точность $\varepsilon > 0$, то используя оценку (5.7)

$\frac{\|B\|^{k+1}}{1 - \|B\|} \|b\| \leq \varepsilon$, можно определить необходимое число итерации k для получения требуемой точности.

Метод Зейделя. Пусть дана система алгебраических уравнений $Ax = f$ с неособенной матрицей A . Вспомним, случай канонической формы системы для метода простой итерации

$$x = Bx + b. \quad (5.8)$$

В методе простой итерации следующее приближение $x^{k+1} = (x_1^{k+1} \dots x_n^{k+1})$ находится по предыдущему $x^k = (x_1^k \dots x_n^k)$ подстановкой x^k в правую часть всех уравнений системы (5.8).

Результат подстановки записывается в развернутом виде (5.3') или

$$\begin{cases} x_1^{k+1} = b_{11}x_1^k + b_{12}x_2^k + b_{13}x_3^k + \dots + b_{1n}x_n^k + \beta_1 \\ x_2^{k+1} = b_{21}x_1^k + b_{22}x_2^k + b_{23}x_3^k + \dots + b_{2n}x_n^k + \beta_2 \\ \dots \\ x_n^{k+1} = b_{n1}x_1^k + b_{n2}x_2^k + b_{n3}x_3^k + \dots + b_{nn}x_n^k + \beta_n \end{cases} \quad (5.9)$$

В этом случае при расчетах *опускаются две возможности улучшения итераций*:

1. Разумный выбор порядка уравнений для подстановок.
2. Немедленный ввод в вычисления каждого из полученных исправленных значений неизвестных.

Изменяя, если необходимо, нумерацию уравнений и неизвестных, будем считать, что уравнения для подстановки берутся в порядке роста их номеров. Для каждого шага приближения порядок для уравнений подстановки может быть своим, т.е. перестановки уравнений и неизвестные свои. Следовательно, матрица B и свободные члены β_i будут иметь соответствующие изменения. Учитывая этот факт выбора порядка привлечения уравнений для подстановок, можем записать итерацию в методе Зейделя в следующем виде

$$\begin{cases} x_1^{k+1} = b_{11}x_1^k + b_{12}x_2^k + b_{13}x_3^k + \dots + b_{1n}x_n^k + \beta_1 \\ x_2^{k+1} = b_{21}x_1^{k+1} + b_{22}x_2^k + b_{23}x_3^k + \dots + b_{2n}x_n^k + \beta_2 \\ x_3^{k+1} = b_{31}x_1^{k+1} + b_{32}x_2^{k+1} + b_{33}x_3^k + \dots + b_{3n}x_n^k + \beta_3 \\ \dots \\ x_n^{k+1} = b_{n1}x_1^{k+1} + b_{n2}x_2^{k+1} + \dots + b_{n,n-1}x_{n-1}^{k+1} + b_{nn}x_n^k + \beta_n \end{cases} \quad (5.10)$$

После определения вектора x^{k+1} устанавливаем порядок подстановок в уравнения значений $x_i^{k+1}, i = \overline{1, n}$ и вычисляем вектор x^{k+2} и т.д.

Замечание 5.3. Нетрудно видеть, что хотя в правую часть (5.10) входит вектор x^{k+1} , формально который ещё не построен, фактически для вычисления i -ой компоненты вектора x^{k+1} в выписанной формуле используются лишь компоненты вектора x^{k+1} с номерами, меньшими i и к этому моменту уже вычисленные.

Порядок выбора уравнения для подстановок можно построить исходя из принципа максимальной погрешности ε^k . В первую очередь получаем ту составляющую решения, которая найдена менее точно, чтобы при нахождении решения других составляющих подставлять ее в улучшенное значение. Порядок выбора уравнений можно построить, используя невязки, полученные при подстановке неизвестных в уравнения и вычитания этих уравнений из правой части, то есть $(f - Ax) = \delta$ – невязка. Нумерацию уравнений системы проводят в порядке убывания невязок.

Обычно используется стационарный метод Зейделя, когда при итерациях порядок уравнений и неизвестных не меняется. В этом случае матрица B остается постоянной на всех шагах, а составляющая всех приближений определяется по значениям x_i^k , в соответствии с записанной системой (5.10).

Разложим матрицу B на сумму матриц D и C :

$$D = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ b_{21} & 0 & 0 & \dots & 0 \\ b_{31} & b_{32} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & b_{n3} & \dots & 0 \end{pmatrix}, \quad C = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ 0 & b_{22} & b_{23} & \dots & b_{2n} \\ 0 & 0 & b_{33} & \dots & b_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & b_{nn} \end{pmatrix}.$$

Тогда равенство (5.10) можно записать в форме матричного равенства

$$x^{k+1} = Dx^{k+1} + Cx^k + b.$$

Откуда следует, что $(E - D)x^{k+1} = Cx^k + b$.

Так как

$$\det(E - D) = \det \begin{pmatrix} 1 & 0 & \dots & 0 \\ -b_{21} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ -b_{n1} & -b_{n2} & \dots & 1 \end{pmatrix} = 1,$$

то матрица $E - D$ имеет обратную матрицу. Тогда равенство (5.10) равносильно

$$x^{k+1} = (E - D)^{-1} Cx^k + (E - D)^{-1} b. \quad (5.11)$$

Следовательно, стационарный метод Зейделя равносильен методу простой итерации, примененному к системе

$$x = (E - D)^{-1} Cx + (E - D)^{-1} b.$$

Отсюда для сходимости стационарного процесса Зейделя (5.10) при любом векторе x^0 начального приближения необходимо и достаточно, чтобы все собственные значения матрицы $(E - D)^{-1} C$ были по модулю меньше единицы. Т.е. корни уравнения $|(E - D)^{-1} C - \lambda E| = 0$ должны быть меньше единицы. Если учесть, что $|(E - D)^{-1}| = |E - D|^{-1} = 1$, то можно написать следующие соотношения

$$\begin{aligned} |(E - D)^{-1} C - \lambda E| &= |(E - D)^{-1} [C - \lambda(E - D)]| = \\ &= |(E - D)^{-1}| |C + \lambda D - \lambda E| = |C + \lambda D - \lambda E|. \end{aligned}$$

Тогда верна теорема.

Теорема 5.6. Для того, чтобы стационарный метод Зейделя сходился при любом начальном векторе приближения x^0 , необходимо и достаточно, чтобы все корни уравнения

$$|C + \lambda D - \lambda E| = \begin{vmatrix} b_{11} - \lambda & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} - \lambda & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nn} - \lambda \end{vmatrix} = 0 \quad (5.12)$$

были по модулю меньше единицы.

Лемма 5.1. Если в матрице $A = [a_{ij}]$, $i, j = \overline{1, n}$, диагональные элементы a_{ii} доминируют по строкам или столбцам, т.е. если $\sum_{j=1, j \neq i}^n |a_{ij}| < |a_{ii}|$, $i = \overline{1, n}$ или $\sum_{i=1, i \neq j}^n |a_{ij}| < |a_{jj}|$, $j = \overline{1, n}$, то определитель матрицы $\det A \neq 0$.

Теорема 5.7. Для сходимости стационарного метода Зейделя (5.11) достаточно, чтобы выполнялось хотя бы одно из условий

$$\|B\|_1 = \max_{1 \leq i \leq n} \sum_{j=1}^n |b_{ij}| < 1, \quad (5.13)$$

$$\|B\|_2 = \max_{1 \leq j \leq n} \sum_{i=1}^n |b_{ij}| < 1. \quad (5.14)$$

Для доказательства достаточно показать, что при выполнении условий (5.13), (5.14) все корни уравнения (5.12) по модулю меньше единицы.

Замечание 5.4. Сходимость метода Зейделя имеет место, если выполнено хотя бы одно из условий:

1. Матрица A исходной СЛАУ обладает свойством диагонального преобладания.
2. Матрица A является положительно определенной (определители всех угловых миноров матриц положительны или все собственные значения матрицы положительны).

Замечание 5.5. Обычно метод Зейделя дает лучшую сходимость, чем метод простой итерации, но, вообще говоря, он приводит к более громоздким вычислениям. Процесс Зейделя может сходиться даже в том случае, если расходится процесс итерации. Однако это бывает не всегда. Возможны случаи, когда процесс Зейделя сходится медленнее процесса итерации. Более того, могут быть случаи, когда процесс итерации сходится, а процесс Зейделя расходится.

Каноническая форма итерационного процесса. Пусть дана система линейных алгебраических уравнений:

$$Ax = f. \quad (5.15)$$

Для ее решения выбирают начальное приближение x_0 и последовательно находят приближенное решение уравнения (5.15). Значение итерации x_{k+1} выражается через известные предыдущие итерации $x_k, x_{k-1}, \dots$, если при вычислении x_{k+1} используется только **одна** предыдущая итерация x_k , то итерационный метод называется **одношаговым или двухслойным**. Если же x_{k+1} выражается через **две** итерации x_k и x_{k-1} , то метод называется **двухшаговым или трехслойным**.

Будем считать, что A – линейный оператор в конечном пространстве H со скалярным произведением. Любой **двухслойный** итерационный метод можно записать в следующей **канонической** форме:

$$B \frac{x_{k+1} - x_k}{\tau_{k+1}} + Ax_k = f, \quad k = 0, 1, 2, \dots, \quad (5.16)$$

для всех $x_0 \in H$ с итерационными параметрами $\tau_1, \tau_2, \dots, \tau_{k+1}, \tau_{k+1} > 0$. Оператор B может зависеть от номера итерации k . Будем предполагать, что B не зависит от номера k и имеет обратный оператор B^{-1} . Если $B = E$ – единичный оператор, то итерационный метод

$$\frac{x_{k+1} - x_k}{\tau_{k+1}} + Ax_k = f, \quad k = 0, 1, 2, \dots, \quad (5.17)$$

для всех $x_0 \in H$ называется **явным**: x_{k+1} находится по явной формуле $x_{k+1} = x_k - \tau_{k+1}(Ax_k - f)$. В общем случае, при $B \neq E$ метод (7.16) называется **неявным** итерационным методом. Для определения x_{k+1} надо решить уравнение

$$Bx_{k+1} = Bx_k - \tau_{k+1}(Ax_k - f) = F_k, \quad k = 0, 1, 2, \dots \quad (5.18)$$

Очевидно, необходимо требовать, чтобы объем вычислений для нахождения решения $Bx_{k+1} = F_k$ был меньше, чем объем вычислений для прямого решения системы (5.15).

Точность итерационного метода (5.16) характеризуется величиной погрешности $z_k = x_k - x$, то есть разностью между решением уравнения (5.16) и точным решением x исходной системы линейных алгебраических уравнений (5.15). Подставив $x_k = z_k + x$ в (5.16), придем к однородному уравнению для погрешности

$$B \frac{z_{k+1} - z_k}{\tau_{k+1}} + Az_k = 0, \quad k = 0, 1, 2, \dots, \quad z_0 = x_0 - x \in H. \quad (5.19)$$

Говорят, что итерационный процесс (метод) сходится в H_D , если $\lim_{k \rightarrow \infty} \|z_k\|_D = 0$, где $\|z\|_D = \sqrt{(Dz, z)}$ и $D = D^* > 0$.

Обычно задается некоторая относительная погрешность $\varepsilon > 0$, с которой надо найти приближенное решение x_k , и прекращают вычисления как только выполнится условие

$$\|x_k - x_{k-1}\|_D \leq \varepsilon \|x_0 - x\|_D . \quad (5.20)$$

Если $n = n(\varepsilon)$ – наименьшее из чисел для которых (5.20) выполняется, то общее число арифметических действий, которое затрачивается для нахождения приближенного решения уравнения (5.15) будет $Q_n(\varepsilon) = n(\varepsilon)q$, где q – число действий, затраченных для нахождения одной итерации. Задача состоит в минимизации $Q_n(\varepsilon)$ путем выбора оператора B и параметров $\{\tau_{k+1}\}$.

ВМИП ЧМ Лекции ПРОИ

Лекция 6. Интерполирование функций (6ч. – 2ч. ЛК+4ч. ЛР)

1. Постановка задачи глобальной полиномиальной интерполяции.
2. Узлы интерполяции. Существование и единственность интерполяционного многочлена.
3. Многочлен Лагранжа. Погрешность интерполяции.
4. Интерполяционный многочлен Ньютона с конечными и с разделенными разностями.
5. Интерполяция сплайнами.
6. Численное дифференцирование. Погрешность формул численного дифференцирования.

Постановка задачи интерполирования. Рассмотрим на отрезке $[a, b]$ некоторую m – кратно дифференцируемую функцию $f(x)$. Пусть в k_0 точках $x_{01}, x_{02}, x_{03}, \dots, x_{0k_0}$ известны ее значения $f(x_{01}), f(x_{02}), f(x_{03}), \dots, f(x_{0k_0})$, в k_1 точках $x_{11}, x_{12}, x_{13}, \dots, x_{1k_1}$ известны значения первой производной $f^{(1)}(x_{11}), f^{(1)}(x_{12}), \dots, f^{(1)}(x_{1k_1})$ и в k_m точках известны значения m -ой производной $f^{(m)}(x_{m1}), f^{(m)}(x_{m2}), \dots, f^{(m)}(x_{mk_m})$. Значения функции и ее производных называются **данными интерполирования**, а точки x_{ij} – **узлами интерполирования**.

Задача интерполирования заключается в отыскании функции $\varphi(x)$ из некоторого класса Φ такой, что выполняется условие

$$\varphi^{(i)}(x_{ij}) = f^{(i)}(x_{ij}), \quad i=1,2,\dots,m, \quad j=1,2,\dots,k_i. \quad (6.1)$$

Пусть $n = k_0 + k_1 + \dots + k_m$. Рассмотрим на отрезке $[a, b]$ последовательность линейно независимых m – кратно дифференцируемых функций: $\omega_1(x), \omega_2(x), \dots, \omega_n(x)$. В качестве семейства Φ возьмем всевозможные линейные комбинации первых n функций с произвольными коэффициентами

$$\varphi(x) = a_1\omega_1(x) + a_2\omega_2(x) + \dots + a_n\omega_n(x).$$

Из условия (6.1) получим систему линейных алгебраических уравнений для определения коэффициентов a_i , $i=1,2,\dots,n$

$$a_1\omega_1^{(i)}(x_{ij}) + a_2\omega_2^{(i)}(x_{ij}) + \dots + a_n\omega_n^{(i)}(x_{ij}) = f^{(i)}(x_{ij}). \quad (6.2)$$

Система (6.2) будет иметь единственное решение в том случае, если ее определитель отличен от нуля.

Примеры интерполяционных функций. Приведем примеры интерполяционных функций.

1. Рассмотрим следующую систему линейно независимых функций: $1, x, x^2, x^3, \dots, x^n, \dots$. Тогда семейством Φ является совокупность алгебраических многочленов вида

$$P_n(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_{n-1}x + a_n. \quad (6.3)$$

Геометрически это означает, что нужно найти алгебраическую кривую $y = P_n(x)$, проходящую через систему точек $M_i(x_i, y_i)$ ($i=1,2,\dots,n$) (рисунок 6.1).

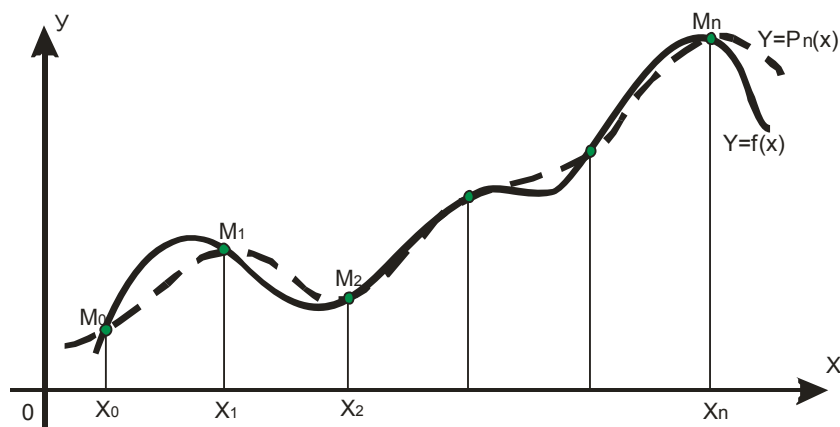


Рисунок 6.1 – Интерполяционный многочлен

Система уравнений, из которой определяются коэффициенты a_i из (6.3), будет

$$P_n(x_i) = a_0 x_i^n + a_1 x_i^{n-1} + a_2 x_i^{n-2} + \dots + a_{n-1} x_i + a_n = f(x_i), \quad (6.4)$$

или

$$a_0 x_0^n + a_1 x_0^{n-1} + a_2 x_0^{n-2} + \dots + a_{n-1} x_0 + a_n = f(x_0),$$

$$a_0 x_1^n + a_1 x_1^{n-1} + a_2 x_1^{n-2} + \dots + a_{n-1} x_1 + a_n = f(x_1),$$

$$\dots \dots \dots$$

$$a_0 x_n^n + a_1 x_n^{n-1} + a_2 x_n^{n-2} + \dots + a_{n-1} x_n + a_n = f(x_n).$$

Ее определитель является определителем **Вандермонда**,

$$\begin{vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{vmatrix}$$

и он отличен от нуля для различных между собой значениях x_i .

Интерполирование полиномами вида (6.3) называется *алгебраическим*. Многочлен $P_n(x)$ называется **интерполяционным многочленом**. Точки x_i ($i = 1, 2, \dots, n$) называются **узлами интерполяции**.

2. Для интерполирования периодических функций с периодом 2π применяется система тригонометрических функций: $1, \cos x, \sin x, \cos 2x, \sin 2x, \cos 3x, \sin 3x, \dots$. Линейная комбинация первых $2n+1$ функций является тригонометрическим многочленом степени n

$$\varphi_n(x) = a_0 + \sum_{k=1}^n (a_k \cos kx + b_k \sin kx). \quad (6.5)$$

Интерполирование с помощью полиномов (6.5) называется *тригонометрическим*.

Пусть для функции $f(x)$ построена интерполирующая функция $\varphi(x)$. Тогда, если определяется значение $f(x)$ в точке $\bar{x}$, лежащей внутри отрезка $[a, b]$ интерполирования, то такое восстановление функции называется

интерполяцией. Если же точка $\bar{x}$ лежит вне отрезка $[a, b]$, то такое восстановление функции называется **экстраполяцией.**

Конечные разности и их свойства. Конечные разности в вычислительной математике имеют значение, аналогичное дифференциалам в анализе бесконечно малых величин.

Пусть даны **равноотстоящие** друг от друга узлы $x_k = x_0 + kh$ ($k = 0, 1, 2, \dots$) и известны соответствующие значения функции $y_k = f(x_k) = f(x_0 + kh)$. Здесь $h = \Delta x = x_k - x_{k-1}$ – некоторое фиксированное значение аргумента. (или рассмотрим таблицу значений функции с **постоянным шагом h** , для которой значения аргумента определяются формулой $x_k = x_0 + kh$ ($k = 0, 1, 2, \dots$), значения функции $y_k = f(x_k) \Rightarrow$ узлы x_k – равноотстоящие).

Конечными разностями нулевого порядка называются величины y_k равные значениям функции $f(x_k)$ в узлах x_k , т.е. $y_k = f(x_k)$. Конечной разностью *первого* порядка называется разность между значениями функции в соседних узлах интерполяции

$$\Delta y_0 = y_1 - y_0 \equiv f(x_0 + h) - f(x_0), \quad \Delta y_1 = y_2 - y_1, \\ \Delta y_2 = y_3 - y_2, \dots, \quad \Delta y_k = y_{k+1} - y_k. \quad (6.6)$$

Конечные разности *второго* порядка определяются равенствами по отношению к разностям первого порядка (дельта два игрек нулевое)

$$\Delta^2 y_0 = \Delta y_1 - \Delta y_0, \quad \Delta^2 y_1 = \Delta y_2 - \Delta y_1, \dots, \\ \Delta^2 y_k = \Delta y_{k+1} - \Delta y_k = y_{k+2} - y_{k+1} - (y_{k+1} - y_k) \equiv y_{k+2} - 2y_{k+1} + y_k.$$

Разности n -го порядка определяются по формуле

$$\Delta^n y_k = \Delta^{n-1} y_{k+1} - \Delta^{n-1} y_k, \quad k = 0, 1, 2, \dots. \quad (6.7)$$

Конечные разности любого порядка легко выражаются через значения функции

$$\Delta^n y_0 = y_n - \frac{n}{1!} y_{n-1} + \frac{n(n-1)}{2!} y_{n-2} + \dots + (-1)^n y_0 = \sum_{i=0}^n (-1)^i C_n^i y_{n-i}, \quad (6.8)$$

где $C_n^k = \frac{n!}{k!(n-k)!}$ (число сочетаний из n по k).

Доказательство проведем по индукции. Пусть эта формула верна для $n = k$. Покажем, что она будет верна и при $n = k + 1$.

$$\Delta^{k+1} y_0 = \Delta^k y_1 - \Delta^k y_0 = \sum_{i=0}^k (-1)^i C_k^i y_{k-i+1} - \sum_{i=0}^k (-1)^i C_k^i y_{k-i} = \\ = \sum_{i=0}^k (-1)^i C_k^i y_{k-i+1} - \sum_{i=1}^{k+1} (-1)^i C_k^{i-1} y_{k-i+1} = \sum_{i=0}^{k+1} (-1)^i (C_k^i + C_k^{i-1}) y_{k-i+1} = \\ = \sum_{i=0}^{k+1} (-1)^i \left(\frac{k!}{i!(k-i)!} + \frac{k!}{(i-1)!(k-i+1)!} \right) y_{k-i+1} = \\ = \sum_{i=0}^{k+1} (-1)^i \left(\frac{k!}{i!(k+1-i)!} (k-i+1+i) \right) y_{k+1-i} = \\ = \sum_{i=0}^{k+1} (-1)^i \frac{(k+1)!}{i!(k+1-i)!} y_{k+1-i} = \sum_{i=0}^{k+1} (-1)^i C_{k+1}^i y_{k+1-i}.$$

Аналогично доказывается формула

$$y_n = y_0 + \frac{n}{1!} \Delta y_0 + \frac{n(n-1)}{2!} \Delta^2 y_0 + \dots + \Delta^n y_0 = \sum_{i=0}^n C_n^i \Delta^i y_0 = (1 + \Delta)^n y_0. \quad (6.9)$$

Из определения конечных разностей вытекают следующие свойства

1. Если $f(x) = u(x) + v(x)$, то $\Delta f(x) = \Delta u(x) + \Delta v(x)$; если функция представлена в виде суммы двух функций, то конечная разность исходной функции представляется как сумма конечных разностей слагаемых.

2. Если $f(x) = cu(x)$, $c = \text{const}$, то $\Delta f(x) = c \Delta u(x)$; при умножении функции на const, конечные разности умножаются на тот же множитель.

3. Конечные разности n -го порядка от многочлена степени n постоянны $\Delta^n P_n(x) = n! a_0 h^n = \text{const}$, а $\Delta^{n+1} P_n(x) = 0$;

4. $\Delta^m (\Delta^n f(x)) = \Delta^{m+n} f(x)$.

Таблицу конечных разностей обычно располагают следующим образом:

Таблица 6.1 – Конечные разности

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$	$\Delta^5 f$	$\Delta^6 f$
x_0	f_0						
		Δf_0					
x_1	f_1		$\Delta^2 f_0$				
		Δf_1		$\Delta^3 f_0$			
x_2	f_2		$\Delta^2 f_1$		$\Delta^4 f_0$		
		Δf_2		$\Delta^3 f_1$		$\Delta^5 f_0$	
x_3	f_3		$\Delta^2 f_2$		$\Delta^4 f_1$		$\Delta^6 f_0$
		Δf_3		$\Delta^3 f_2$		$\Delta^5 f_1$	
x_4	f_4		$\Delta^2 f_3$		$\Delta^4 f_2$		
		Δf_4		$\Delta^3 f_3$			
x_5	f_5		$\Delta^2 f_4$				
		Δf_5					
x_6	f_6						

Теорема о существовании интерполяционного многочлена. Пусть на отрезке $[a, b]$ в $n+1$ узле $x_k = x_0 + kh$ ($k=0,1,2,\dots,n$) заданы значения ограниченной функции $f(x)$: $y_0 = f(x_0)$, $y_1 = f(x_1)$, $y_2 = f(x_2)$, ..., $y_n = f(x_n)$. Поставим задачу нахождения полинома $P_n(x)$ степени не выше n такого, чтобы выполнялось условие

$$f(x_i) = P_n(x_i), \quad i = 0, 1, \dots, n. \quad (6.10)$$

Теорема 6.1. Существует и притом единственный многочлен степени не выше n , для которого выполняется условие (6.10).

Доказательство. Пусть многочлен $P_n(x)$ имеет вид (6.3). Используя (6.10), для определения коэффициентов a_i , получим систему (6.4) линейных алгебраических уравнений, которую запишем в виде

$$a_0 x_0^n + a_1 x_0^{n-1} + a_2 x_0^{n-2} + \dots + a_{n-1} x_0 + a_n = f(x_0),$$

Формулу (6.15) выгодно использовать для интерполирования в окрестности начального значения x_0 . Поэтому ее часто называют формулой для **интерполирования вперед**. В этой формуле из таблицы конечных разностей используются $\Delta^k f_0$ верхней диагонали.

Остаточный член в формуле (11.15) имеет вид

$$R_n(x) = h^{n+1} \frac{q(q-1)\dots(q-n)}{(n+1)!} f^{(n+1)}(\xi),$$

где ξ – некоторая внутренняя точка наименьшего промежутка, содержащего все узлы x_i , $i=1,2,\dots,n$ и точку x . При наличии дополнительного узла x_{n+1} на практике пользуются более удобной приближенной формулой

$$R_n(x) \approx \frac{\Delta^{n+1} y_0}{(n+1)!} q(q-1)\dots(q-n).$$

При $n=1$ из (6.15) получается формула *линейного* интерполирования

$$P_1(x) = y_0 + q\Delta y_0.$$

При $n=2$ из (6.15) имеет место формула *параболического* или *квадратического* интерполирования

$$P_2(x) = y_0 + q\Delta y_0 + \frac{q(q-1)}{2!} \Delta^2 y_0.$$

Вторая интерполяционная формула Ньютона для равноотстоящих узлов. Первая интерполяционная формула Ньютона практически неудобна для интерполирования функций в конце таблицы. Поэтому когда точка интерполирования лежит вблизи точки x_n удобно пользоваться **второй интерполяционной формулой Ньютона**, которая имеет вид

$$P_n(x) = y_n + \frac{\Delta y_n}{1!h} (x-x_n) + \dots + \frac{\Delta^n y_n}{n!h^n} (x-x_n)(x-x_{n-1})\dots(x-x_1). \quad (6.16)$$

Вводя новую переменную $q = \frac{x-x_n}{h}$, эту формулу перепишем в виде

$$P_n(x(q)) = y_n + q\Delta y_{n-1} + \frac{q(q+1)}{2!} \Delta^2 y_{n-2} + \dots + \frac{q(q+1)(q+2)\dots(q+n-1)}{n!} \Delta^n y_0. \quad (6.17)$$

В формуле (6.17) из таблицы конечных разностей используются $\Delta^k f_i$ нижней диагонали.

Остаточный член формулы (6.17) имеет вид

$$R_n(x) = h^{n+1} \frac{q(q+1)\dots(q+n)}{(n+1)!} f^{(n+1)}(\xi), \quad (6.17')$$

где точка ξ имеет тот же смысл, что и ранее.

Отметим, что формулы Ньютона используются и для экстраполирования функций. Если $\bar{x} < x_0$, то для *экстраполирования назад* используют *первую интерполяционную формулу Ньютона*. Если $\bar{x} > x_0$, то для *экстраполирования вперед* используют *вторую интерполяционную формулу Ньютона*. Следует заметить, что операция экстраполирования менее точна, чем операция интерполирования в узком смысле.

Разностные отношения и их свойства. В том случае, когда значения аргумента являются неравноотстоящими для исследования и вычисления функции используются разностные отношения (разделенные разности). Пусть на отрезке $[a,b]$ заданы произвольные попарно различные узлы

интерполирования $x_0, x_1, x_2, \dots, x_{n-1}, x_n$, в которых известны значения некоторой функции $f(x)$: $y_0 = f(x_0), y_1 = f(x_1), y_2 = f(x_2), \dots, y_n = f(x_n)$.

Разностными отношениями первого порядка называются величины

$$f(x_0, x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}, \quad f(x_1, x_2) = \frac{f(x_2) - f(x_1)}{x_2 - x_1}, \quad \dots, \quad f(x_k, x_{k+1}) = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}. \quad (6.18)$$

Иногда вместо $f(x_i, x_{i+1})$ для обозначения разделенных разностей используют выражение $[x_i, x_{i+1}]$. И так для разных порядков разделенных разностей.

По разностным отношениям первого порядка составляются разностные отношения **второго порядка**

$$f(x_0, x_1, x_2) = \frac{f(x_1, x_2) - f(x_0, x_1)}{x_2 - x_0}, \quad f(x_1, x_2, x_3) = \frac{f(x_2, x_3) - f(x_1, x_2)}{x_3 - x_1}, \dots \quad (6.19)$$

Разностные отношения любого порядка n ($n=1, 2, 3, \dots$) определяются при помощи разностных отношений предыдущего порядка $n-1$ по формуле

$$f(x_0, x_1, \dots, x_{n-1}, x_n) = \frac{f(x_1, x_2, \dots, x_{n-1}, x_n) - f(x_0, x_1, \dots, x_{n-1})}{x_n - x_0}. \quad (6.20)$$

Удобно располагать таблицу разделенных разностей следующим образом

Таблица 6.2 – Таблица разделенных разностей

x_0	$f(x_0)$			
		$f(x_0, x_1)$		
x_1	$f(x_1)$		$f(x_0, x_1, x_2)$	
		$f(x_1, x_2)$		$f(x_0, x_1, x_2, x_3)$
x_2	$f(x_2)$		$f(x_1, x_2, x_3)$	
		$f(x_2, x_3)$		$f(x_1, x_2, x_3, x_4)$
x_3	$f(x_3)$		$f(x_2, x_3, x_4)$	
		$f(x_3, x_4)$		
x_4	$f(x_4)$			

Используя определение, можно показать, что разностное отношение является симметрической функцией своих аргументов так, что выполняется равенство (покажем ниже)

$$f(x_0, x_1, \dots, x_i, x_{i+1}, \dots, x_{n-1}, x_n) = f(x_0, x_1, \dots, x_{i+1}, x_i, \dots, x_{n-1}, x_n). \quad (6.21)$$

Лемма 6.1. Если $P_n(x)$ полином степени n , то его разделенная разность $n+1$ порядка равна нулю для любой системы попарно различных между собой чисел $x, x_0, x_1, x_2, \dots, x_{n-1}, x_n$

$$P_n(x, x_0, x_1, x_2, \dots, x_{n-1}, x_n) \equiv 0. \quad (6.22)$$

Доказательство. Так как полином $P_n(x) - P_n(x_0)$ имеет корень в точке x_0 , по определению разделенной разности, получим

$$P_n(x, x_0) = \frac{P_n(x_0) - P_n(x)}{x_0 - x} = P_{n-1}(x).$$

С учетом (6.19) полином $P_n(x_0, x_1) - P_n(x, x_0)$ обращается в нуль в точке x_1 и вторая разделенная разность будет полиномом степени $n-2$

$$P_n(x, x_0, x_1) = \frac{P_n(x_0, x_1) - P_n(x, x_0)}{x_1 - x} = P_{n-2}(x).$$

Продолжая аналогичные рассуждения, приходим к выводу, что

$$P_n(x, x_0, x_1, x_2, \dots, x_{n-1}, x_n, x_{n+1}) = P(x_0) = \text{const},$$

и, следовательно, для разностного отношения $n+1$ порядка справедливо равенство (6.22). Лемма доказана.

Отметим некоторые **свойства** разностных отношений.

1. Свойство аддитивности. Разделенная разность суммы или разности функций равна сумме или разности разделенных разностей слагаемых, соответственно уменьшаемого и вычитаемого, т.е. если $f(x) = u(x) + v(x)$, то $f(x_0, x_1) = u(x_0, x_1) + v(x_0, x_1)$.

2. Свойство подобия. Постоянный множитель можно выносить за знак разделенной разности, т.е. если $f(x) = cu(x)$, где $c = \text{const}$, то $f(x_0, x_1) = cu(x_0, x_1)$.

Свойства 1, 2 сформулированы для разностных отношений первого порядка, но они верны для разностных отношений любых порядков.

Следующее свойство получается как следствие из представления разностного отношения через значения функции. Для разностного отношения первого порядка, по определению, имеем,

$$f(x_0, x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f(x_0)}{x_0 - x_1} + \frac{f(x_1)}{x_1 - x_0}. \quad (6.23)$$

Для разностного отношения второго порядка получим

$$\begin{aligned} f(x_0, x_1, x_2) &= \frac{1}{x_2 - x_0} \{f(x_1, x_2) - f(x_0, x_1)\} = \\ &= \frac{1}{x_2 - x_0} \left\{ \frac{f(x_1)}{x_1 - x_2} + \frac{f(x_2)}{x_2 - x_1} - \frac{f(x_0)}{x_0 - x_1} - \frac{f(x_1)}{x_1 - x_0} \right\} = \\ &= \frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2)} + \frac{f(x_1)}{(x_1 - x_0)(x_1 - x_2)} + \frac{f(x_2)}{(x_2 - x_0)(x_2 - x_1)}. \end{aligned} \quad (6.24)$$

При помощи индукции можно показать, что для любого n верно следующее представление

$$f(x_0, x_1, x_2, \dots, x_{n-1}, x_n) = \sum_{i=0}^n \frac{f(x_i)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)} = \sum_{i=0}^n \frac{f(x_i)}{\omega'(x_i)}, \quad (6.25)$$

где $\omega(x) = (x - x_0)(x - x_1) \dots (x - x_n) = \prod_{j=1}^n (x - x_j)$, а $\omega'(x_i)$ значение производной от $\omega(x)$ в точке x_i .

Если выполнить любую перестановку аргументов $x_0, x_1, \dots, x_n$, то в последней части (6.25) переменятся местами слагаемые, что не изменит сумму. Это дает сформулировать следующее свойство:

3. Свойство симметрии. Разностное отношение $f(x_0, x_1, \dots, x_{n-1}, x_n)$ есть симметричная функция своих аргументов (см. (6.21))

$$f(x_0, x_1, \dots, x_i, x_{i+1}, \dots, x_{n-1}, x_n) = f(x_0, x_1, \dots, x_{i+1}, x_i, \dots, x_{n-1}, x_n).$$

4. Если $f(x)$ есть многочлен степени n , то разностное отношение n -го порядка $f(x_0, x_1, \dots, x_n)$ не зависит от $x_0, x_1, x_2, \dots, x_n$ и равняется коэффициенту при старшей степени x в многочлене $f(x)$. Все разностные отношения порядка большего n равны нулю (лемма 6.1).

Приведем выражение произвольного значения функции $f(x_n)$ через начальное её значение $f(x_0)$ и начальные значения разностных отношений $f(x_0, x_1)$, $f(x_0, x_1, x_2)$, $f(x_0, x_1, x_2, x_3), \dots$. По определению $f(x_0, x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$ и, следовательно, $f(x_1) = f(x_0) + (x_1 - x_0)f(x_0, x_1)$.

Ввиду полученного результата и определения (6.24) будем иметь

$$\begin{aligned} f(x_2) &= f(x_1) + (x_2 - x_1)f(x_1, x_2) = f(x_0) + (x_1 - x_0)f(x_0, x_1) + \\ &+ (x_2 - x_1)[f(x_0, x_1) + (x_2 - x_0)f(x_0, x_1, x_2)] = \\ &= f(x_0) + (x_2 - x_0)f(x_0, x_1) + (x_2 - x_0)(x_2 - x_1)f(x_0, x_1, x_2). \end{aligned}$$

Используя индукцию, получим выражение для любого n

$$\begin{aligned} f(x_n) &= f(x_0) + (x_n - x_0)f(x_0, x_1) + (x_n - x_0)(x_n - x_1)f(x_0, x_1, x_2) + \dots + \\ &+ (x_n - x_0)(x_n - x_1) \dots (x_n - x_{n-1})f(x_0, x_1, x_2, \dots, x_n). \end{aligned} \quad (6.26)$$

Установим связь между разностными отношениями и конечными разностями. Предположим, что значения аргумента $x_0, x_1, x_2, \dots, x_n$ являются равноотстоящими $x_0, x_1 = x_0 + h, \dots, x_n = x_0 + nh$. Тогда получим

$$f(x_0, x_1) = f(x_0, x_0 + h) = \frac{f(x_0 + h) - f(x_0)}{(x_0 + h) - x_0} = \frac{y_1 - y_0}{h} = \frac{\Delta y_0}{1!h}. \quad (6.27)$$

Для разностного отношения второго порядка имеем

$$\begin{aligned} f(x_0, x_1, x_2) &= f(x_0, x_0 + h, x_0 + 2h) = \\ &= \frac{f(x_0 + h, x_0 + 2h) - f(x_0, x_0 + h)}{(x_0 + 2h) - x_0} = \frac{1}{2h} \frac{\Delta y_1 - \Delta y_0}{1!h} = \frac{\Delta^2 y_0}{2!h^2}. \end{aligned}$$

И для n -го разностного отношения имеет место равенство

$$f(x_0, x_0 + h, \dots, x_0 + nh) = \frac{\Delta^n y_0}{n!h^n}, \quad y_i = f(x_0 + ih). \quad (6.28)$$

Интерполяционный многочлен Лагранжа. Построим формулу для произвольного расположения узлов интерполирования.

Пусть на отрезке $[a, b]$ заданы произвольные попарно различные узлы интерполирования $x_0, x_1, x_2, \dots, x_{n-1}, x_n$, в которых известны значения некоторой функции $f(x)$: $y_0 = f(x_0)$, $y_1 = f(x_1)$, $y_2 = f(x_2), \dots, y_n = f(x_n)$. Построим полином $L_n(x)$, для которого выполняется равенство

$$L_n(x_i) = f(x_i), \quad i = 0, 1, \dots, n. \quad (6.29)$$

Решим сначала вспомогательную задачу и построим полином $P_{n,i}(x)$ степени n , удовлетворяющий следующему условию

$$P_{n,i}(x_j) = \begin{cases} 0, & \text{при } i \neq j \\ 1, & \text{при } i = j. \end{cases} \quad (6.30)$$

Из условия (6.30) следует, что полином $P_{n,i}(x)$ имеет n корней в узлах $x_0, x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_{n-1}, x_n$. Следовательно, он может быть записан в виде

$$P_{n,i}(x) = k_i(x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n).$$

Так как $P_{n,i}(x_i) = 1$, то для коэффициентов k_i получаем выражение

$$k_i = ((x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n))^{-1}.$$

Очевидно, что тогда полином $L_n(x)$, удовлетворяющий (6.29), можно записать в виде

$$L_n(x) = \sum_{i=0}^n P_{n,i}(x) y_i. \quad (6.31)$$

Формула (6.31) называется **интерполяционной формулой Лагранжа**. Обычно формулу Лагранжа записывают в другом виде. Рассмотрим полином $\omega(x)$ степени $n+1$

$$\omega(x) = (x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_i)(x - x_{i+1}) \dots (x - x_n).$$

Для первой производной справедливо выражение

$$\omega'(x) = \sum_{j=0}^n \prod_{i \neq j} (x - x_i).$$

С учетом введенных обозначений **полином Лагранжа** записывается в виде

$$L_n(x) = \sum_{i=0}^n \frac{\omega(x)}{(x - x_i)\omega'(x_i)} f(x_i), \quad (6.32)$$

или

$$L_n(x) = \sum_{i=0}^n y_i \cdot \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}, \quad i = \overline{0, n}, \quad (6.33)$$

или

где

$$L_n(x) = \sum_{i=0}^n y_i \cdot L_i^n(x), \quad i = \overline{0, n},$$

$$L_i^n(x) = \frac{(x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}, \quad i = \overline{0, n} -$$

коэффициенты формулы Лагранжа.

Укажем на некоторые особенности лагранжевой формулы (6.32), (6.33). Свойства интерполяционного многочлена зависят от двух факторов: от выбора узлов x_i и от интерполируемой функции f . В формуле (6.33) оба фактора разделены, т.к. многочлены $L_i^n(x)$ зависят только от узлов, а свойства функции f учитываются множителями $f(x_i)$ (или y_i). В отношении вычислений формула Лагранжа удобна при интерполировании **многих функций в одной точке** x , так как значения множителей $L_i^n(x)$ можно вычислить однажды и для всех функций. Но вычислительное применение формулы Лагранжа имеет существенный недостаток, т.к. нужно заранее определять число $n+1$ узлов, необходимое для достижения заданной точности. Желание избежать ненужной затраты труда побуждает нас стараться обойтись наименьшим числом узлов и нередко оказывается, что заданное нами число узлов является недостаточным или бывает необходимо проверить точность полученного результата. В обоих случаях к взятым узлам добавляют еще один или несколько узлов и выполняют вычисления заново. Тогда в формуле (6.32) не только добавляются новые члены, но необходимо пересчитывать и все ранее найденные члены суммы, так как в них появятся новые множители.

Погрешность формулы Лагранжа. Проводя интерполирование по формуле (6.32), (6.33), мы допускаем некоторую погрешность

$R_n(x) = f(x) - L_n(x)$, которая является нулевой в общем случае только в узлах интерполирования.

Будем считать, что на отрезке интерполирования $[a, b]$ функция $f(x)$ имеет все производные вплоть до $n+1$ порядка включительно. Рассмотрим вспомогательную функцию $\Psi(x) = f(x) - L_n(x) - k\omega(x)$. Очевидно, что $\Psi(x)$ имеет $n+1$ корней в узлах $x_0, x_1, x_2, \dots, x_{n-1}, x_n$. Выберем произвольную точку $\bar{x} \in [a, b]$ и подберем постоянную k так, чтобы выполнялось равенство

$$\Psi(\bar{x}) = f(\bar{x}) - L_n(\bar{x}) - k\omega(\bar{x}) = 0, \quad (6.34)$$

Тогда отсюда определяется значение k

$$k = \frac{f(\bar{x}) - L_n(\bar{x})}{\omega(\bar{x})},$$

знаменатель этой дроби отличен от нуля, т.к. $x \neq x_i, i = \overline{0, n}$

Функция $\Psi(\bar{x}) = f(\bar{x}) - L_n(\bar{x}) - k\omega(\bar{x})$ обращается в нуль на $[a, b]$ в $n+2$ точках $x, x_0, x_1, \dots, x_n$.

Следовательно, по теореме Ролля производная $\Psi'(\bar{x})$ обращается в нуль по крайней мере $n+1$ раз на интервале (a, b) . Пусть эти значения $\bar{x}$ будут: $\xi_0^{(1)}, \xi_1^{(1)}, \xi_2^{(1)}, \dots, \xi_n^{(1)}$.

Применим снова теорему Ролля к функции $\Psi'(\bar{x})$. Получим по крайней мере n точек $\xi_0^{(2)}, \xi_1^{(2)}, \xi_2^{(2)}, \dots, \xi_{n-1}^{(2)}$ таких, что

$$\Psi''(\xi_0^{(2)}) = \Psi''(\xi_1^{(2)}) = \Psi''(\xi_2^{(2)}) = \dots = \Psi''(\xi_{n-1}^{(2)}) = 0.$$

Продолжая этот процесс дальше, найдем, что существует по крайней мере одна точка ξ на интервале (a, b) , в которой $\Psi^{(n+1)}(\xi) = 0$, но $\Psi^{(n+1)}(\bar{x}) = f^{(n+1)}(\bar{x}) - k(n+1)!$, т.к. производная порядка $n+1$ от многочлена $L_n(x)$ степени n равна нулю, а производная от многочлена $\omega(x)$ степени $n+1$ со старшим коэффициентом 1 равна $(n+1)!$

Положив в последнем равенстве $\bar{x} = \xi$, получим $k = \frac{f^{(n+1)}(\xi)}{(n+1)!}$.

Отсюда $f(x) - L_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega(x) = (x - x_0)(x - x_1) \dots (x - x_n)$

или для оценки погрешности интерполирования на отрезке $[a, b]$ имеем

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega(x). \quad (6.35)$$

Полагая $M_{n+1} = \sup_{x \in [a, b]} |f^{(n+1)}(x)|$, для погрешности интерполирования (11.35)

можно воспользоваться оценкой

$$|f(x) - L_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |(x - x_0)(x - x_1) \dots (x - x_n)|. \quad (6.36)$$

Выражения (6.35), (6.36) могут служить оценкой отклонения $f(x)$ от $L_n(x)$, если производная $f^{(n+1)}(\xi)$ может быть оценена.

Первая и вторая интерполяционные формулы Ньютона для неравноотстоящих узлов. Используя понятие разностных отношений, формулу Лагранжа можно записать в виде аналогичном первой и второй интерполяционным формулам Ньютона для равноотстоящих узлов. Пусть $L_n(x)$ полином Лагранжа, для которого выполняется условие (6.31) или

$$L_n(x_i) = f(x_i), \quad i = 0, 1, \dots, n. \quad (6.37)$$

Тогда по определению разностного отношения 1 порядка $L_n(x, x_0) = \frac{L_n(x_0) - L_n(x)}{x_0 - x}$.

Откуда получаем $L_n(x) = L_n(x_0) + L_n(x, x_0)(x - x_0)$. По определению, разностного отношения второго порядка имеем

$$L_n(x, x_0, x_1) = \frac{L_n(x_0, x_1) - L_n(x, x_0)}{x_1 - x_0}.$$

Откуда $L_n(x, x_0) = L_n(x_0, x_1) + L_n(x, x_0, x_1)(x - x_1)$. Последовательно выражая разностные отношения m -го порядка через отношения $m+1$ -го порядка по формуле

$$L_n(x, x_0, \dots, x_{m-1}) = L_n(x_0, x_1, \dots, x_m) + L_n(x, x_0, \dots, x_m)(x - x_m),$$

окончательно для полинома $L_n(x)$ получим представление

$$\begin{aligned} L_n(x) = & L_n(x_0) + L_n(x_0, x_1)(x - x_0) + L_n(x_0, x_1, x_2)(x - x_0)(x - x_1) + \\ & + \dots + L_n(x_0, x_1, \dots, x_i)(x - x_0)(x - x_1) \dots (x - x_{i-1}) + \dots + \\ & + L_n(x_0, x_1, \dots, x_n)(x - x_0)(x - x_1) \dots (x - x_{n-1}) + \\ & + L_n(x, x_0, x_1, \dots, x_n)(x - x_0)(x - x_1) \dots (x - x_n). \end{aligned}$$

Учитывая (6.37) и то, что $L_n(x, x_0, \dots, x_n) = 0$ (это следует из леммы 6.1: разнесенная разность $n+1$ порядка от многочлена степени n равна 0), получим **первую интерполяционную формулу Ньютона для неравноотстоящих узлов**

$$\begin{aligned} P_n(x) = & f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) + \dots + \\ & + f(x_0, x_1, x_2, \dots, x_n)(x - x_0)(x - x_1) \dots (x - x_{n-1}). \end{aligned} \quad (6.38)$$

Аналогичным образом можно построить и **вторую интерполяционную формулу Ньютона для неравноотстоящих узлов**, которая имеет вид

$$\begin{aligned} P_n(x) = & f(x_n) + f(x_n, x_{n-1})(x - x_n) + f(x_n, x_{n-1}, x_{n-2})(x - x_n)(x - x_{n-1}) + \\ & + \dots + f(x_n, x_{n-1}, x_{n-2}, \dots, x_0)(x - x_n)(x - x_{n-1}) \dots (x - x_1). \end{aligned} \quad (6.39)$$

Формула Ньютона имеет строение более сложное, чем интерполяционная формула Лагранжа (6.32), (6.33) и требует составление разностных отношений $f(x_0, x_1, x_2, \dots, x_{n-1}, x_k)$, $k = \overline{1, n}$. При вычислительном процессе формулы (6.38), (6.39) более удобны, чем формулам Лагранжа. При добавлении к $x_0, x_1, x_2, \dots, x_{n-1}, x_n$ нового узла x_{n+1} все ранее найденные члены сохраняются и в формуле добавляется ещё один член $(x - x_0)(x - x_1) \dots (x - x_n)f(x_0, x_1, \dots, x_n, x_{n+1})$. Это позволяет не задавать заранее число узлов и постепенно увеличивать точность результата, добавляя последовательно по одному новому узлу. В отличие от вычислений по формуле Лагранжа, добавление узла(ов) не приводит к повторению всей проделанной работы заново.

Кроме формулы (6.36) для оценки погрешности получим еще одну исходя из формулы Ньютона. Добавим к узлам $x_0, x_1, x_2, \dots, x_{n-1}, x_n$ еще один узел x т.е. точку, в которой вычисляется значение функции $f(x)$. Тогда по формуле выражающей значения функции в узле через начальные разностные отношения, получим

$$\begin{aligned} f(x) = & f(x_0) + f(x_0, x_1)(x-x_0) + f(x_0, x_1, x_2)(x-x_0)(x-x_1) + \dots + \\ & + f(x_0, x_1, x_2, \dots, x_n)(x-x_0)(x-x_1)\dots(x-x_{n-1}) + \\ & + f(x_0, x_1, \dots, x_n, x)(x-x_0)\dots(x-x_{n-1})(x-x_n). \end{aligned}$$

Заметим, что в правой части сумма всех слагаемых без последнего есть интерполяционная формула Ньютона (6.38). Тогда

$$f(x) = P_n(x) + f(x_0, x_1, \dots, x_n, x)(x-x_0)\dots(x-x_{n-1})(x-x_n),$$

или

$$R(x) = f(x) - P_n(x) = \omega(x)f(x_0, x_1, \dots, x_n, x). \quad (6.40)$$

Сравнивая формулы (6.35) ($R_n(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega(\bar{x})$) (для $\forall \bar{x} \in [a, b]$ – погрешность формулы Лагранжа) и (6.40) получим, что

$$f(x_0, x_1, \dots, x_n, x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

Минимизация остатка интерполирования. Пусть функция $f(x)$ приближается на отрезке $[a, b]$ с помощью интерполяционного многочлена степени n с узлами интерполяции $x_0, x_1, x_2, \dots, x_{n-1} \in [a, b]$. Рассмотрим выражение для погрешности в виде

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega(x). \quad (6.41)$$

Здесь $\omega(x) = (x-x_0)(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)$.

Поставим перед собой задачу: путем выбора узлов интерполирования минимизировать погрешность (6.41) на отрезке $[a, b]$ для функции $f(x)$. Если считать, что функция $f^{(n+1)}(x)$ мало меняется на отрезке $[a, b]$, то нам нужно на этом отрезке минимизировать величину $\omega(x) = (x-x_0)(x-x_1)\dots(x-x_n)$. Эта задача вплотную примыкает к задаче, решенной русским математиком П.Л. Чебышевым.

Среди всех многочленов степени n с коэффициентом равным единице при старшей степени найти многочлен, наименее уклоняющийся от нуля на отрезке $[-1, 1]$.

Многочленом Чебышева, определенным на отрезке $[-1, 1]$, называется многочлен

$$T_n(x) = \cos(n \cdot \arccos x), \quad n \geq 0.$$

Учитывая, что

$$\begin{aligned} \cos(n+1)\varphi &= \cos n\varphi \cos \varphi - \sin n\varphi \sin \varphi = 2 \cos n\varphi \cos \varphi - \\ &- (\cos n\varphi \cos \varphi + \sin n\varphi \sin \varphi) = 2 \cos n\varphi \cos \varphi - \cos(n-1)\varphi, \end{aligned}$$

и, считая $\varphi = \arccos x$, получим следующую рекуррентную формулу

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n = 1, 2, \dots \quad (6.42)$$

Приведем несколько частных выражений многочлена Чебышева $T_n(x)$, $n \geq 0$

$$T_0(x) = 1, T_1(x) = x.$$

Из формулы (6.42), получаем, например,

$$T_2(x) = 2x^2 - 1, T_3(x) = 4x^3 - 3x, T_4(x) = 8x^4 - 8x^2 + 1, \dots$$

Из соотношения (6.42) вытекают следующие **свойства** многочленов Чебышева

1. При четном n многочлен содержит только четные степени x , а при нечетном n – только нечетные степени x .
2. Старший коэффициент многочлена Чебышева $T_n(x)$ при $n \geq 1$ равен 2^{n-1} .
3. Многочлен $T_n(x)$ имеет n действительных корней на интервале $(-1, 1)$ в точках $x_i = \cos \frac{\pi(2i+1)}{2n}$, $i = 0, 1, 2, \dots, n-1$.
4. Многочлен $T_n(x)$ принимает максимальное по модулю значение в точках $x_m = \cos \frac{\pi m}{n}$, $m = 0, 1, 2, \dots, n$, которые называются точками **чебышевского альтернанса**. При этом справедливо равенство $T_n(x_m) = (-1)^m$. $T_n(-1) = -1$, $T_n(+1) = 1$.

Теорема 6.2. Среди всех многочленов степени n со старшим коэффициентом равным единице наименьшее значение максимума модуля на отрезке $[-1, 1]$ имеет многочлен

$$\bar{T}_n(x) = 2^{1-n} T_n(x), \quad n = 1, 2, 3, \dots,$$

где $T_n(x)$ – многочлен Чебышева.

Доказательство проведем методом от противного. Пусть существует многочлен $P_n(x) = x^n + \alpha_1 x^{n-1} + \dots + \alpha_{n-1} x + \alpha_n$ такой, что справедливо неравенство

$$\max_{[-1, 1]} |P_n(x)| \leq \max_{[-1, 1]} |\bar{T}_n(x)| = 2^{1-n}. \quad (6.43)$$

Тогда $R_{n-1}(x) = \bar{T}_n(x) - P_n(x)$ является многочленом степени $n-1$. Так как $T_n(x_m) = (-1)^m$, то $R_{n-1}(x_m) > 0$ в точках $x_0, x_2, x_4, \dots$ и $R_{n-1}(x_m) < 0$ в точках $x_1, x_3, x_5, \dots$. Следовательно, $R_{n-1}(x)$ меняет знак в n точках интервалов $[x_0, x_1], [x_1, x_2], \dots, [x_{n-1}, x_n]$, следовательно, имеет n корней. Но многочлен степени $n-1$ не может иметь более $n-1$ корней. Поэтому соотношение (6.42) невозможно и для всякого многочлена $P_n(x)$ должно выполняться неравенство

$$\max_{[-1, 1]} |\bar{T}_n(x)| = 2^{1-n} \leq \max_{[-1, 1]} |P_n(x)|.$$

Теорема доказана.

Используем доказанную теорему для минимизации отклонения многочлена $\omega(x)$ от нуля на отрезке $[a, b]$. Отрезок $[a, b]$ заменой переменных $t = \frac{2x}{b-a} - \frac{b+a}{b-a}$ переводится в отрезок $[-1, 1]$. Поэтому, чтобы многочлен $\omega(x)$ наименее уклонялся от нуля, на отрезке $[a, b]$ в качестве узлов интерполирования следует взять точки $x_k = \frac{t_k(b-a) + b+a}{2}$, где $t_k = \cos \frac{(2k+1)\pi}{2(n+1)}$ –

корни многочлена Чебышева степени $n+1$. Тогда $x_k - x = \frac{(b-a)(t_k - t)}{2}$ и для величины отклонения $\omega(x)$ от нуля справедлива формула

$$\omega(x) = \frac{(b-a)^{n+1}}{2^{n+1}} \omega(t) = \frac{(b-a)^{n+1}}{2^{n+1}} \bar{T}_{n+1}(x) \leq \frac{(b-a)^{n+1}}{2^{2n+1}}.$$

При указанном выборе узлов интерполирования из (6.41) для погрешности формулы Лагранжа получим оценку

$$|R_n(x)| \leq \frac{|f^{(n+1)}(\xi)|}{(n+1)!} \frac{(b-a)^{n+1}}{2^{2n+1}}. \quad (6.44)$$

Определение интерполяционного сплайна. Аналитические приближения требуют довольно много хороших свойств функции, в частности существование производных высокого порядка, что не всегда имеет место на практике. К тому же далекие от точки интерполирования узлы мало влияют на значение функции в этой точке. Поэтому естественно поступить следующим образом: разбить весь отрезок интерполирования $[a, b]$ на участки $[x_i, x_{i+1}]$ и на каждом из этих участков вычислять свою интерполяционную функцию, определенным образом «сшивая» их в узлах. В том случае, когда эти функции являются многочленами, **интерполирование называется кусочно-полиномиальным.**

Разобьем отрезок $[a, b]$ на n частей $[x_0, x_1], [x_1, x_2], \dots, [x_{n-1}, x_n]$; $x_0 = a, x_n = b$. Обозначим это разбиение через Δ . Назовем **сплайном** $S_m^\Delta(f, x)$ **порядка m** функцию, являющуюся многочленом степени m на каждом из отрезков $[x_{i-1}, x_i]$

$$S_m^\Delta(f, x) = P_{im}(x) = a_{i0} + a_{i1}x + \dots + a_{im}x^m \text{ при } x \in [x_{i-1}, x_i] \quad (6.45)$$

и удовлетворяющую **условиям непрерывности производных до порядка $m-1$ во внутренних точках** $x_1, x_2, \dots, x_{n-1}$

$$P_{im}^k(x_i) = P_{i+1,m}^k(x_i), \quad k = 0, 1, \dots, m-1; \quad i = 1, 2, \dots, n-1, \quad (6.46)$$

и **условиям совпадения значений сплайна и функции во всех узлах** $x_0, x_1, x_2, \dots, x_{n-1}, x_n$

$$P_{im}(x_i) = y_i, \quad i = 0, 1, \dots, n. \quad (6.47)$$

В этой задаче необходимо определить $m+1$ коэффициент для каждого из n уравнений, то есть, всего $n(m+1)$ коэффициентов a_{ij} , $i = 1, 2, \dots, n$; $j = 0, 1, \dots, m$. Условия непрерывности дают $(n-1)m$ уравнений; условия совпадения — $n-1$ уравнение: всего имеем $n(m+1) - m + 1$ уравнений. Остальные $m-1$ уравнений выбираются исходя из некоторых дополнительных условий.

И так, **сплайном** называется функция, которая вместе с несколькими производными непрерывна на заданном отрезке $[a, b]$, а на каждом частичном отрезке $[x_{i-1}, x_i]$ является некоторым алгебраическим многочленом. Максимальная по всем отрезкам степень многочленов называется **степенью сплайна**, а разность между степенью сплайна и порядком наивысшей непрерывной производной на $[a, b]$ называется **дефектом сплайна**. Сплайн, который принимает в узлах те же значения, что и функция $f(x)$ называется

интерполяционным. На практике широко используются сплайны третьей степени – $S_3(x)$. Такие сплайны называются **кубическими**.

Пусть $k_i = S_3^{(1)}(x_i)$ – наклоны сплайна в точке x_i . Если заданы значения f_i, f_{i+1} и k_i, k_{i+1} на $[x_i, x_{i+1}]$ и $h_i = x_{i+1} - x_i$, то интерполяционный сплайн имеет вид

$$S_3(x) = \frac{(x_{i+1} - x)^2(2(x - x_i) + h_{i+1})}{h_{i+1}^3} f_i + \frac{(x - x_i)^2(2(x_{i+1} - x) + h_{i+1})}{h_{i+1}^3} f_{i+1} + \frac{(x_{i+1} - x)^2(x - x_i)}{h_{i+1}^2} k_i + \frac{(x - x_i)^2(x - x_{i+1})}{h_{i+1}^2} k_{i+1}. \quad (6.48)$$

О способах задания сплайнов. Существует несколько способов задания интерполяционных сплайнов. Пусть узлы интерполирования – равноотстоящие.

Способ 1. Определяя наклоны k_i по формулам

$$k_i = \frac{f_{i+1} - f_{i-1}}{2h}, \quad i = 1, 2, \dots, n-1, \quad (6.49)$$

$$k_0 = \frac{4f_1 - f_2 - 3f_0}{2h}, \quad k_n = \frac{3f_n + f_{n-2} - 4f_{n-1}}{2h}, \quad (6.50)$$

где $h = \frac{b-a}{n}$, кубический сплайн можно записать в виде (11.48), причём $h_{i+1} = h$.

Способ 2. Если известны значения производной $f_i^{(1)}$, то полагаем $k_i = f_i^{(1)}$. Затем сплайн записывается в форме (6.48).

Первый и второй способы являются локальными и гарантируют непрерывность первых производных сплайна $S_3^{(1)}(x)$ в узлах x_i и дефект равен 2.

Способ 3. Будем строить сплайн $S_3(x)$ так, чтобы выполнялись условия

1) $S_3(x) \in C_{[a,b]}^2$;

2) на каждом из отрезков сплайн является многочленом третьей степени;

3) в узлах выполняются условия $S_3(x_i) = f(x_i)$, $i = 0, 1, \dots, n$ – условия совпадения (6.47);

4) удовлетворяет некоторым граничным условиям (определим их позже).

Будем искать сплайн на отрезке $[x_{i-1}, x_i]$ в виде

$$S_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3. \quad (6.51)$$

Из условия 3) следует, что $S_i(x_i) = a_i = f_i$, и $a_0 = f_0$. Дифференцируя получим

$$b_i = S_i^{(1)}(x_i); \quad c_i = S_i^{(2)}(x_i); \quad d_i = S_i^{(3)}(x_i);$$

Из условия непрерывности функции: $S_i(x_i) = S_{i+1}(x_i)$, следует, что

$$a_i = a_{i+1} + b_{i+1}(x_i - x_{i+1}) + \frac{c_{i+1}}{2}(x_i - x_{i+1})^2 + \frac{d_{i+1}}{6}(x_i - x_{i+1})^3,$$

и так как $h_i = x_i - x_{i-1}$, тогда можно записать

$$a_i = a_{i+1} - b_{i+1}h_{i+1} + \frac{c_{i+1}}{2}h_{i+1}^2 - \frac{d_{i+1}}{6}h_{i+1}^3,$$

учитывая интерполяционность, приходим к уравнению

$$f_i - f_{i-1} = h_i b_i - \frac{c_i}{2}h_i^2 + \frac{d_i}{6}h_i^3, \quad i = 1, \dots, n. \quad (\text{во внутренних точках}) \quad (6.52)$$

Из условия непрерывности первой производной (6.46): $S_i^{(1)}(x_i) = S_{i+1}^{(1)}(x_i)$ следует, что

$$b_i = b_{i+1} - c_{i+1}h_{i+1} + \frac{d_{i+1}}{2}h_{i+1}^2, \quad \text{или} \\ c_i h_i - \frac{d_i}{2}h_i^2 = b_i - b_{i-1}, \quad i = 2, 3, \dots, n. \quad (6.53)$$

Из условия непрерывности второй производной (6.46): $S_i^{(2)}(x_i) = S_{i+1}^{(2)}(x_i)$ следует, что

$$c_i = c_{i+1} - d_{i+1}h_{i+1}, \quad \text{или} \\ c_i - c_{i-1} = d_i h_i, \quad i = 2, \dots, n. \quad (6.54)$$

Уравнения (6.52) – (6.54) – образуют систему из $3n-2$ уравнений относительно $b_1, b_2, \dots, b_n, c_1, \dots, c_n, d_1, \dots, d_n$ – $3n$ неизвестных. Дополняя эту систему условиями на границе для $S_3(x)$, получим полную систему из $3n$ уравнений.

Типы граничных условий. Рассмотрим различные типы граничных условий.

1. Пусть заданы значения $f_0^{(2)} = c_0$ и $f_n^{(2)} = c_n$. Из уравнений (6.52) – (6.54) получается система

$$\begin{cases} c_0 = f_0^{(2)} \\ h_i c_{i-1} + 2(h_i + h_{i+1})c_i + h_{i+1}c_{i+1} = 6\left(\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}\right), \quad i = 1, 2, \dots, n-1. \\ c_n = f_n^{(2)} \end{cases} \quad (6.55)$$

Эта система имеет диагональное преобладание и может быть эффективно решена методом трехточечной прогонки. Затем определяются остальные неизвестные

$$d_i = \frac{c_i - c_{i-1}}{h_i}, \quad b_i = \frac{h_i}{2}c_i - \frac{h_i^2}{6}d_i + \frac{f_i - f_{i-1}}{h_i}, \quad i = 1, \dots, n.$$

Если $f^{(2)}(a) = f^{(2)}(b) = 0$, то в системе (6.55) граничные условия будут $c_0 = 0, c_n = 0$.

2. Если на границе заданы значения первых производных $f_a^{(1)} = f_0^{(1)} = b_0$ и $f_b^{(1)} = f_n^{(1)} = b_n$, то систему (6.52) – (6.54) можно привести к виду

$$\begin{cases} b_0 = f_0^{(1)} \\ 2h_{i+1}b_{i-1} + 4(h_i + h_{i+1})b_i + 2h_i b_{i+1} = 6\left(\frac{h_i}{h_{i+1}}(f_{i+1} - f_i) + \frac{h_{i+1}}{h_i}(f_i - f_{i-1})\right), \quad i = 1, 2, \dots, n-1. \\ b_n = f_n^{(1)} \end{cases} \quad (6.56)$$

Эта система также эффективно может быть решена методом прогонки. Для равномерной сетки $h_i = h_{i+1}$, для внутренних точек получим

$$b_{i-1} + 4b_i + b_{i+1} = 3 \left(\frac{f_{i+1} - f_{i-1}}{h} \right). \quad (6.57)$$

Если эти граничные условия использовать при решении системы (6.55), то на границе дополняем уравнением

$$\begin{aligned} \frac{2}{3}c_0 + \frac{1}{3}c_1 &= \frac{2}{h_1} \left(\frac{f_1 - f_0}{h_1} - f_0^{(1)} \right), \\ \frac{1}{3}c_{n-1} + \frac{2}{3}c_n &= \frac{2}{h_n} \left(f_n^{(1)} - \frac{f_n - f_{n-1}}{h_n} \right). \end{aligned} \quad (6.58)$$

3. Если на границе заданы значения только функции f_0 и f_n , то можно решить систему (6.56) с краевыми условиями

$$\begin{aligned} b_0 &= \frac{1}{6h} (-11f_0 + 18f_1 - 9f_2 + 2f_3), \\ b_n &= \frac{1}{6h} (11f_n - 18f_{n-1} + 9f_{n-2} - 2f_{n-3}). \end{aligned} \quad (6.59)$$

4. Пусть $f(x)$ периодическая функция с периодом $b-a$, тогда следует считать, что $f_0 = f_n$, $S_3^{(1)}(a) = S_3^{(1)}(b)$ и $S_3^{(2)}(a) = S_3^{(2)}(b)$. Если использовать систему (6.55), то ее можно дополнить краевыми условиями

$$\begin{aligned} c_0 &= c_n, \\ c_1 h_1 + 2(h_n + h_1)c_n + c_{n-1} h_n &= 6 \left(\frac{f_1 - f_n}{h_1} - \frac{f_n - f_{n-1}}{h_n} \right). \end{aligned} \quad (6.60)$$

Для равномерной сетки из (6.60) получим уравнение

$$c_1 + 4c_n + c_{n-1} = \frac{6}{h^2} (f_1 - 2f_n + f_{n-1}).$$

Если использовать систему (6.56), то ее нужно дополнить условиями

$$\begin{aligned} b_0 &= b_n, \\ \frac{f_1 - f_0}{h_1^2} - \frac{f_{n-1} - f_0}{h_n^2} &= \frac{1}{3h_1} (2b_0 + b_1) + \frac{1}{3h_n} (2b_0 + b_{n-1}). \end{aligned} \quad (6.61)$$

Для равномерной сетки из (6.61) получим уравнение

$$\frac{3(f_1 - f_{n-1})}{h} = b_1 + 4b_0 + b_{n-1}.$$

Если решается система (6.55) и получены значения c_i , то кубический сплайн можно сразу записать в виде на отрезке $[x_{i-1}, x_i]$

$$\begin{aligned} S_3(x) &= c_{i-1} \frac{(x_i - x)^3}{6h_i} + c_i \frac{(x - x_{i-1})^3}{6h_i} + (f_{i-1} - \frac{c_{i-1}h_i^2}{6}) \frac{x_i - x}{h_i} + \\ &+ (f_i - \frac{c_i h_i^2}{6}) \frac{x - x_{i-1}}{h_i}. \end{aligned} \quad (6.62)$$

О погрешности приближения сплайнами. Следующая теорема дает ответ о погрешности приближения сплайнами.

Теорема 6.3. Если $f(x) \in C_{k+1}[a, b]$, $0 \leq k \leq 3$, то интерполяционный сплайн $S_3(x)$, построенный способами 2 или 3, удовлетворяет неравенству

$$\max_{[x_i, x_{i+1}]} |f^{(m)}(x) - S_3(x)| \leq ch^{k+1-m} \max_{[a, b]} |f^{(k+1)}(x)|,$$

где $i = 0, 1, \dots, n-1$, $m = 0, 1, \dots, k$, c — не зависящая от h, i, f постоянная. Если $f(x) \in C_4[a, b]$, то можно получить $c = 1$. Если сплайн $s_3(x)$ построен по способу 1, то данная теорема справедлива при $0 \leq k \leq 2$.

К численному дифференцированию приходится прибегать в том случае, когда функция $f(x)$, для которой нужно найти производную, задана **таблично** или же **функциональная зависимость** x и $f(x)$ имеет **очень сложное аналитическое выражение**. В первом случае методы дифференциального исчисления неприменимы (поскольку производные берутся от непрерывных функций), а во втором случае их использование вызывает значительные трудности.

Задача численного дифференцирования ставится следующим образом: пусть на отрезке $[a, b]$ рассматривается функция $f(x)$, имеющая непрерывную производную порядка $n+1$. Возьмем на $[a, b]$ $n+1$ различных узлов (для удобства) $x_0 < x_1 < \dots < x_n$ и пусть в них известны значения функции $y_0 = f(x_0)$, $y_1 = f(x_1)$, ..., $y_n = f(x_n)$. Требуется найти значение k -ой производной от функции $f(x)$ в любой точке $x \in [a, b]$. Построим интерполяционный многочлен $P_n(x)$ для функции $f(x)$ степени n с погрешностью $R(x)$. Тогда запишем $f(x) = P_n(x) + R(x)$. Для производной порядка k имеем $f^{(k)}(x) = P_n^{(k)}(x) + R^{(k)}(x)$. Если погрешность мала ($R^{(k)}(x) \rightarrow 0$), то пренебрегая ею, получим формулу для приближенного вычисления производной $f^{(k)}(x) \approx P_n^{(k)}(x)$. Пользоваться этой формулой целесообразно при небольших порядках k , когда $k \leq n$, так как все производные от $P_n(x)$ порядка выше n тождественно равны нулю.

Получим некоторые явные формулы для численного вычисления производных при некоторых видах многочлена $P_n(x)$.

Дифференцирование для равноотстоящих узлов. Если узлы равноотстоящие, т.е. $x_{i+1} - x_i = h = \text{const}$ ($i = \overline{0, n-1}$), то для вычисления производных удобно пользоваться, например, первой интерполяционной формулой Ньютона. Пусть $f(x) \approx P_n(x)$ (см. формулу (6.15))

$$P_n(x(q)) = y_0 + q\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_0 + \frac{q(q-1)(q-2)}{3!}\Delta^3 y_0 + \frac{q(q-1)(q-2)(q-3)}{4!}\Delta^4 y_0 + \dots$$

где $q = \frac{x-x_0}{h}$.

Чтобы легче было брать производные, перепишем эту формулу, раскрыв скобки

$$P_n(x(q)) = y_0 + q\Delta y_0 + \frac{q^2 - q}{2!}\Delta^2 y_0 + \frac{q^3 - 3q^2 + 2q}{6}\Delta^3 y_0 + \frac{q^4 - 6q^3 + 11q^2 - 6q}{24}\Delta^4 y_0 + \dots$$

Продифференцируем последнее равенство, получим выражение первой производной. Учитывая, что $\frac{dP}{dx} = \frac{dP}{dq} \frac{dq}{dx} = \frac{1}{h} \frac{dP}{dq}$ ($\frac{dq}{dx} = \frac{d}{dx} \left(\frac{x-x_0}{h} \right) = \frac{1}{h} \frac{d}{dx} (x-x_0) = \frac{1}{h}$), будем иметь

$$P_n^{(1)}(x) = \frac{1}{h} (\Delta y_0 + \frac{2q-1}{2}\Delta^2 y_0 + \frac{3q^2-6q+2}{6}\Delta^3 y_0 + \frac{2q^3-9q^2+11q-3}{12}\Delta^4 y_0 + \dots). \quad (6.63)$$

Далее для второй производной будем иметь

$$P_n^{(2)}(x) = \frac{1}{h^2}(\Delta^2 y_0 + (q-1)\Delta^3 y_0 + \frac{6q^2 - 18q + 11}{12}\Delta^4 y_0 + \dots). \quad (6.64)$$

Аналогично вычисляются производные высших порядков. Если производная вычисляется в узловой точке, то формулы упрощаются. Например, если $x = x_0$, то $q = 0$ и для первой и второй производной получим

$$\begin{aligned} P_n^{(1)}(x_0) &= \frac{1}{h}(\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3} - \frac{\Delta^4 y_0}{4} + \dots), \\ P_n^{(2)}(x_0) &= \frac{1}{h^2}(\Delta^2 y_0 - \Delta^3 y_0 + \frac{11}{12}\Delta^4 y_0 - \dots). \end{aligned} \quad (6.65)$$

Так как $R_n(x) = f(x) - P_n(x) = h^{n+1} \frac{q(q-1)\dots(q-n)f^{(n+1)}(\xi)}{(n+1)!}$ (остаточный член формулы Ньютона), то, считая, что $f(x) \in C_{[a,b]}^{n+2}$ (функция непрерывна на $[a,b]$ с производной $n+2$ порядка), получим $(\frac{dR_n}{dx} = \frac{dR_n}{dq} \frac{dq}{dx} = \frac{1}{h} \frac{dR_n}{dq})$

$$R_n^{(1)}(x) = \frac{h^n}{(n+1)!} \cdot \left(f^{(n+1)}(\xi) \frac{d}{dq} [q(q-1)\dots(q-n)] + q(q-1)\dots(q-n) \frac{d}{dq} f^{(n+1)}(\xi) \right).$$

В узлах интерполирования последняя формула примет вид (используем для вывода этой формулы формулу Ньютона в виде (6.14))

$$R_n^{(1)}(x_i) = (-1)^{n-i} \frac{h^n i!(n-i)!}{(n+1)!} f^{(n+1)}(\xi). \quad (6.66)$$

Дифференцирование для неравноотстоящих узлов. Для случая неравноотстоящих узлов удобно пользоваться формулами Лагранжа или Ньютона для неравноотстоящих узлов.

Формула Лагранжа имеет вид (6.37)

$$L_n(x) = \omega_n(x) \cdot \sum_{i=0}^n \frac{f(x_i)}{(x-x_i)\omega_n^{(1)}(x_i)},$$

где $\omega_n(x) = (x-x_0)(x-x_1)\dots(x-x_n)$.

Тогда первая производная запишется в форме

$$L_n^{(1)}(x) = \omega_n^{(1)}(x) \cdot \sum_{i=0}^n \frac{f(x_i)}{(x-x_i)\omega_n^{(1)}(x_i)} - \omega_n(x) \cdot \sum_{i=0}^n \frac{f(x_i)}{(x-x_i)^2 \omega_n^{(1)}(x_i)}. \quad (6.67)$$

Учитывая, что

$$\omega_n^{(1)}(x) = \sum_{j=0}^n (x-x_0)\dots(x-x_{j-1})(x-x_{j+1})\dots(x-x_n) = \omega_n(x) \cdot \sum_{i=0}^n \frac{1}{x-x_i},$$

(можно рассмотреть для $n=2$: $\omega_2(x) = (x-x_0)(x-x_1)(x-x_2)$,

$$\omega_2^{(1)}(x) = (x-x_0)'[(x-x_1)(x-x_2)] + (x-x_0)[(x-x_1)(x-x_2)]' =$$

$$[(x-x_1)(x-x_2)] + (x-x_0)[(x-x_1)'(x-x_2) + (x-x_1)(x-x_2)'] =$$

$$[(x-x_1)(x-x_2)] + (x-x_0)(x-x_2) + (x-x_0)(x-x_1).$$

Формулу (6.67) можно переписать в ином виде. Итак, первая производная с использованием формулы Лагранжа будет иметь вид

$$L_n^{(1)}(x) = \omega_n(x) \cdot \left(\sum_{k=0}^n \frac{1}{x - x_k} \cdot \sum_{i=0}^n \frac{f(x_i)}{(x - x_i) \omega_n^{(1)}(x_i)} - \sum_{i=0}^n \frac{f(x_i)}{(x - x_i)^2 \omega_n^{(1)}(x_i)} \right).$$

Поскольку вычисления производных с использованием этой формулы несколько громоздко, то на практике более удобно пользоваться формулой Ньютона (6.43)

$$P_n(x) = f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) + \\ + f(x_0, x_1, x_2, x_3)(x - x_0)(x - x_1)(x - x_2) + \\ + \dots + f(x_0, x_1, x_2, \dots, x_n)(x - x_0)(x - x_1) \dots (x - x_{n-1}).$$

Для построения формул численного дифференцирования введем следующие обозначения $\alpha_0 = x - x_0$, $\alpha_1 = x - x_1$, ..., $\alpha_n = x - x_n$, т.е.

$$P_n(x) = f(x_0) + f(x_0, x_1) \cdot \alpha_0 + f(x_0, x_1, x_2) \cdot \alpha_0 \alpha_1 + f(x_0, x_1, x_2, x_3) \cdot \alpha_0 \alpha_1 \alpha_2 + \dots \\ + f(x_0, x_1, x_2, x_3, \dots, x_n) \cdot \alpha_0 \alpha_1 \alpha_2 \dots \alpha_{n-1}.$$

Для вычисления первой производной проведем вспомогательные вычисления

$$(f(x_0, x_1) \cdot \alpha_0)'_x = f(x_0, x_1) \cdot (x - x_0)'_x = f(x_0, x_1), \\ (\alpha_0 \cdot \alpha_1)'_x = \alpha'_0 \cdot \alpha_1 + \alpha_0 \cdot \alpha'_1 = \alpha_0 + \alpha_1, \text{ и т.д.}$$

Тогда первая производная запишется

$$P_n^{(1)}(x) = f(x_0, x_1) + f(x_0, x_1, x_2)(\alpha_0 + \alpha_1) + \\ + f(x_0, x_1, x_2, x_3)(\alpha_0 \alpha_1 + \alpha_1 \alpha_2 + \alpha_0 \alpha_2) + \dots + \\ + f(x_0, x_1, x_2, \dots, x_n) \sum_{k=0}^{n-1} \alpha_0 \alpha_1 \dots \alpha_{k-1} \alpha_{k+1} \dots \alpha_{n-1}. \quad (6.68)$$

Для вычисления второй производной проведем вспомогательные вычисления

$$f(x_0, x_1)'_x = 0, \\ (f(x_0, x_1, x_2)(\alpha_0 + \alpha_1))'_x = f(x_0, x_1, x_2) \cdot (\alpha_0 + \alpha_1)'_x = f(x_0, x_1, x_2) \cdot [\alpha'_0 + \alpha'_1] = \\ = 2 \cdot f(x_0, x_1, x_2), \quad \text{и т.д.}$$

Вторая производная примет вид

$$f^{(2)}(x) \approx P_n^{(2)}(x) = 2!(f(x_0, x_1, x_2) + f(x_0, x_1, x_2, x_3)(\alpha_0 + \alpha_1 + \alpha_2) + \dots). \quad (6.69)$$

Аналогичным образом можно вычислить производные более высокого порядка.

Пример решения задачи. Функция $y = f(x)$ задана таблично.

x	1	2	3	4
y	3	7	19	42

Требуется найти значение 1-ой и 2-ой производных в точке $x=1$.

Решение.

Т.к. $x=1$ – узел интерполяции, то для вычисления производных можно воспользоваться упрощенной формулой (6.65). Для применения этой формулы составим таблицу конечных разностей:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
1	3	4	8	3
2	7	12	11	
3	19	23		
4	42			

Поскольку шаг $h=1$, находим значение первой и второй производных функции в точке $x=1$:

$$f^{(1)}(x_0) = \frac{1}{h} (\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3}),$$

$$f^{(1)}(1) = \frac{1}{1} \left[4 - \frac{1}{2} \cdot 8 + \frac{1}{3} \cdot 3 \right] = 4 - 4 + 1 = 1,$$

$$f^{(2)}(x_0) = \frac{1}{h^2} (\Delta^2 y_0 - \Delta^3 y_0).$$

$$f^{(2)}(1) = \frac{1}{1^2} [8 - 3] = 5.$$

Погрешность формул численного дифференцирования. Рассмотрим два случая: когда $\bar{x} \notin [a, b]$ и $\bar{x} \in [a, b]$, $\bar{x}$ – точка, в которой производится оценка.

1. Пусть точка $\bar{x} \notin [a, b]$, т.е. точка $\bar{x}$ не принадлежит минимальному отрезку $[a, b]$, содержащему узлы интерполирования, то остаточный член может быть представлен простым выражением. Для этого рассмотрим многочлен

$$Q(x) = L_n(x) + k\omega_n(x),$$

где $k = \text{const}$.

Он совпадает с функцией $f(x)$ в точках $x_0, x_1, \dots, x_n$ (т.е. $f(x_i) = Q(x_i)$, $i = \overline{0, n}$). Подберём постоянную k так, чтобы в точке $\bar{x}$, для которой производится оценка, имело место равенство

$$Q^{(k)}(\bar{x}) = L_n^{(k)}(\bar{x}) + k\omega_n^{(k)}(\bar{x}) = f^{(k)}(\bar{x}).$$

Это возможно, так как все корни уравнения $\omega_n^{(k)}(x) = 0$ лежат в наименьшем отрезке, содержащем $x_0, x_1, \dots, x_n$.

Рассмотрим вспомогательную функцию $\varphi(x) = f(x) - L_n(x) - k\omega_n(x)$. Функция $\varphi(x)$ имеет $n+1$ нулей в точках $x_0, x_1, x_2, \dots, x_{n-1}, x_n$ по построению. На основании теоремы Ролля $\varphi^{(1)}(x)$ будет иметь n , $\varphi^{(2)}(x)$ – $n-1$ и $\varphi^{(k)}(x)$ будет иметь $n-k+1$ нулей внутри отрезка $[a, b]$. Но в силу выбора постоянной k она обратится в нуль и в точке $\bar{x}$, лежащей вне этого отрезка, т.е. $\varphi^{(k)}(\bar{x}) = 0$. Тогда $\varphi^{(k)}(x)$ будет иметь $n-k+2$ нулей и, следовательно, $\varphi^{(n+1)}(x)$ обращается в нуль, по крайней мере, в одной точке ξ ($\xi \notin [a, b]$).

Но

$$\varphi^{(n+1)}(\xi) = f^{(n+1)}(\xi) - L_n^{(n+1)}(\xi) - k\omega_n^{(n+1)}(\xi) = f^{(n+1)}(\xi) - k\omega_n^{(n+1)}(\xi) = f^{(n+1)}(\xi) - k(n+1)! = 0$$

Отсюда

$$k = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

и получим выражение для погрешности дифференцирования (выражение остаточного члена)

$$R_n^{(k)}(f) = f^{(k)}(\bar{x}) - L_n^{(k)}(\bar{x}) = k \cdot \omega_n^{(k)}(\bar{x}) = \frac{\omega_n^{(k)}(\bar{x})}{(n+1)!} f^{(n+1)}(\xi). \quad (6.70)$$

2. Пусть точка $\bar{x} \in [a, b]$, т.е. точка $\bar{x}$ принадлежит минимальному отрезку $[a, b]$, содержащему узлы интерполирования, то остаточный член может быть представлен в виде подобном (6.70). Итак, пусть $x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n$ и $\bar{x} \in [x_0, x_n]$. Разность $f^{(k)}(x) - L_n^{(k)}(x)$ обращается на отрезке $[x_0, x_n]$ в нуль, по крайней мере, $n - k + 1$ раз. Пусть это будут точки $\varepsilon_0, \varepsilon_1, \varepsilon_2, \dots, \varepsilon_{n-k}$ (так как k -я производная отнимает k нулей). Легко видеть, что $x_i < \varepsilon_i < x_{i+k}$. Функция

$$\begin{aligned} \Psi(x) &= f^{(k)}(x) - L_n^{(k)}(x) - c(x - \varepsilon_0)(x - \varepsilon_1) \dots (x - \varepsilon_{n-k}) = \\ &= f^{(k)}(x) - L_n^{(k)}(x) - c\tilde{\omega}(x), \end{aligned} \quad (6.70')$$

будет обладать этим свойством (т.е. обращаться в нуль на $[x_0, x_n]$), как бы мы ни выбирали постоянную c .

Пусть $\bar{x} \neq \varepsilon_i$, $i = 0, 1, \dots, n - k$. Тогда константу c можно подобрать так (т.е. чтобы $\tilde{\omega}(\bar{x}) \neq 0$), чтобы $\Psi(\bar{x}) = 0$. При этом функция $\psi(x)$ на отрезке $[x_0, x_n]$ будет обращаться в нуль по крайней мере $n - k + 1$ раз. Проводя такие же рассуждения, что и для функции $\varphi^{(k)}(x)$, мы придём к выводу, что найдётся такая точка $\xi \in [x_0, x_n]$, что $\psi^{(n-k+1)}(\xi) = 0$, т.е. $\Psi^{(n-k+1)}(\xi) = f^{(n+1)}(\xi) - c(n - k + 1)! = 0$,

$$\text{и } c = \frac{f^{(n+1)}(\xi)}{(n - k + 1)!}.$$

Таким образом, для оценки погрешности, используя формулу (6.70'), получим

$$R_n^{(k)}(f) = f^{(k)}(\bar{x}) - L_n^{(k)}(\bar{x}) = \frac{\tilde{\omega}_1(\bar{x}) f^{(n+1)}(\xi)}{(n - k + 1)!}. \quad (6.71)$$

При использовании формул численного дифференцирования может произойти существенная потеря точности. Так, с ростом порядка производной обычно резко падает точность численного дифференцирования. Поэтому на практике редко применяют формулы численного дифференцирования для производных выше второго порядка.

Лекция 7. Равномерное приближение функций (2ч. УСР)

1. Постановка задачи приближения функций.
2. Различные способы задания нормы в нормированном пространстве.
3. Многочлен наилучшего равномерного приближения.
4. Многочлены Чебышева.

Материал для изучения находится в файле «Лекции УСР.docx».

ВМИП ЧМ Лекции ПРОИ

Лекция 8. Среднеквадратическое приближение функций (4ч. – 2ч. ЛК+2ч. ЛР)

1. Метод наименьших квадратов: общая характеристика метода.
2. Построение эмпирических формул методом наименьших квадратов: линейная зависимость, квадратичная зависимость.

Постановка задачи. Пусть R пространство функций $f(x)$ интегрируемых с квадратом на отрезке $[a, b]$. **Скалярным произведением** функций $f(x)$ и $\varphi(x)$ называется выражение $(f, \varphi) = \int_a^b f(x)\varphi(x)dx$. **Нормой** элемента f называется число $\|f\|^2 = \int_a^b f^2(x)dx$. Если $(f, \varphi) = 0$, то говорят, что функции $f(x)$ и $\varphi(x)$ **ортогональны**. Пространство с такими свойствами обозначается $L_2[a, b]$.

Рассмотрим подпространство H состоящее из линейных комбинаций $n+1$ линейно независимых функций $\varphi_0, \varphi_1, \dots, \varphi_n$

$$\varphi = c_0\varphi_0 + c_1\varphi_1 + \dots + c_n\varphi_n.$$

Это подпространство называется **подпространством обобщенных многочленов** по системе функций $\{\varphi(x_i)\}$.

Среднеквадратичным отклонением элемента $\varphi \in H$ от элемента $f \in L_2$ называется величина

$$S^2 = \int_a^b (f(x) - \varphi(x))^2 dx. \text{ (если функция задана аналитически)}$$

В методе наименьших квадратов приближение элемента $f \in R$ элементом $\bar{\varphi} \in H$ осуществляется таким образом, чтобы величина среднеквадратичного отклонения была минимальна в L_2

$$S^2(c_0, \dots, c_n) = \|f - \bar{\varphi}\|^2 = \inf_{\varphi \in H[a, b]} \|f - \varphi\|^2.$$

Если такой элемент $\bar{\varphi}$ существует, он называется элементом наилучшего среднеквадратичного приближения функции $f \in R$ в подпространстве H .

Практика показывает, что приближающие функции, построенные по методу среднеквадратичного приближения, гораздо лучше описывают реальную функцию $f(x)$, чем интерполяционные многочлены.

Теорема об элементе наилучшего среднеквадратичного приближения.

Теорема 8.1. Для того чтобы элемент $\bar{\varphi}(x) \in H$ был элементом наилучшего среднеквадратичного приближения, необходимо и достаточно, чтобы скалярное произведение

$$(f - \bar{\varphi}, \varphi) = 0 \tag{8.1}$$

для любого элемента $\varphi \in H$.

Доказательство. Необходимость. Пусть существует такой элемент φ_1 , что $(f - \bar{\varphi}, \varphi_1) = a \neq 0$. Без нарушения общности можно считать, что $\|\varphi_1\| = 1$. Возьмем элемент $\varphi_2 = \bar{\varphi} + a\varphi_1$. Тогда имеем

$$\begin{aligned}\|f - \varphi_2\|^2 &= (f - \varphi_2, f - \varphi_2) = (f - \bar{\varphi} - a\varphi_1, f - \bar{\varphi} - a\varphi_1) = \\ &= (f - \bar{\varphi}, f - \bar{\varphi}) - a(\varphi_1, f - \bar{\varphi}) - a(f - \bar{\varphi}, \varphi_1) + a^2(\varphi_1, \varphi_1) = \\ &= \|f - \bar{\varphi}\|^2 - 2a^2 + a^2 = \|f - \bar{\varphi}\|^2 - a^2.\end{aligned}$$

Отсюда следует $\|f - \varphi_2\|^2 < \|f - \bar{\varphi}\|^2$, что невозможно, так как $\bar{\varphi}$ является элементом наилучшего приближения.

Достаточность. Пусть $\varphi(x) \in H$. Тогда

$$\begin{aligned}\|f - \varphi\|^2 &= (f - \bar{\varphi} + \bar{\varphi} - \varphi, f - \bar{\varphi} + \bar{\varphi} - \varphi) = \\ &= \|f - \bar{\varphi}\|^2 + (\bar{\varphi} - \varphi, f - \bar{\varphi}) + (f - \bar{\varphi}, \bar{\varphi} - \varphi) + \|\bar{\varphi} - \varphi\|^2 = \|f - \bar{\varphi}\|^2 + \|\bar{\varphi} - \varphi\|^2.\end{aligned}$$

Если $\varphi \neq \bar{\varphi}$, то получим $\|f - \varphi\|^2 > \|f - \bar{\varphi}\|^2$ для произвольного $\varphi \in H$, то есть $\bar{\varphi}$ — элемент наилучшего среднеквадратичного приближения в подпространстве H . Теорема доказана.

Построение многочлена наилучшего приближения для функции на отрезке. Из теоремы 8.1 в частности следует, что если взять в качестве элементов из H последовательно $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$, то из условия ортогональности (8.1) получим

$$(f - \bar{\varphi}, \varphi_i(x)) = 0, \quad (8.2)$$

для любого $i = \overline{0, n}$.

Систему (8.2) можно переписать в виде

$$c_0(\varphi_0, \varphi_i) + c_1(\varphi_1, \varphi_i) + \dots + c_n(\varphi_n, \varphi_i) = (f, \varphi_i), \quad i = \overline{0, n}. \quad (8.3)$$

Матрица системы (8.3) $A = \{(\varphi_i, \varphi_j)\}$ симметрична, положительно определена и ее определитель, называемый определителем Грамма, для системы линейно независимых функций $\{\varphi_i(x)\}$ отличен от нуля. Следовательно, система (8.2) имеет единственное решение.

Возьмем в качестве H класс алгебраических многочленов

$$P_n(x) = c_0 + c_1x + \dots + c_nx^n.$$

Тогда $(\varphi_i, \varphi_j) = \int_a^b x^i x^j dx$, $(f, \varphi_i) = \int_a^b f(x)x^i dx$. Обозначим $s_i = \int_a^b x^i dx$, $m_i = \int_a^b f(x)x^i dx$.

Перепишем систему (8.3) в виде

$$\begin{cases} c_0s_0 + c_1s_1 + \dots + c_ns_n = m_0 \\ c_0s_1 + c_1s_2 + \dots + c_ns_{n+1} = m_1 \\ \dots \\ c_0s_n + c_1s_{n+1} + \dots + c_ns_{2n} = m_n \end{cases} \quad (8.4)$$

Определитель этой системы как определитель Грамма линейно независимых на $[a, b]$ функций $1, x, x^2, \dots, x^n$ всегда положителен и система имеет единственное решение.

Если $\varphi = \bar{\varphi}$, то из (8.1) следует

$$\|f - \bar{\varphi}\|^2 = \|f\|^2 - \|\bar{\varphi}\|^2. \quad (8.5)$$

В этом случае, если система функций $\{\varphi_i(x)\}$ ортонормированна, т.е.

$$(\varphi_k, \varphi_l) = \begin{cases} 0, & \text{если } k \neq l \\ 1, & \text{если } k = l \end{cases},$$

то система (8.3) решается в явном виде

$$c_k = (f, \varphi_k), \quad k = \overline{0, n}. \quad (8.6)$$

При этом погрешность можно вычислить по формуле

$$\|f - \bar{\varphi}\|^2 = \|f\|^2 - \sum_{k=0}^n c_k^2. \quad (8.7)$$

Примеры ортогональных систем многочленов. Примером ортогональной системы функций являются многочлены Лежандра

$$L_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n, \quad (8.8)$$

которые ортогональны на отрезке $[-1, +1]$ и $\|L_n(x)\| = \sqrt{\frac{2}{2n+1}}$.

Эти многочлены можно вычислить по рекуррентной формуле

$$(n+1)L_{n+1}(x) - (2n+1)xL_n(x) + nL_{n-1}(x) = 0. \quad (8.9)$$

Многочленом наилучшего среднеквадратичного приближения по отрезку $[-1, +1]$ будет многочлен

$$P_n(x) = \sum_{k=0}^n c_k L_k(x), \quad (8.10)$$

где $c_k = \frac{2k+1}{2} \int_{-1}^{+1} f(x) L_k(x) dx$.

Отметим, что ряд Фурье по многочленам Лежандра для любой непрерывной функции $f(x)$ на отрезке $[-1, +1]$ сходится равномерно.

Многочлены Лежандра применяются в образовании сферических функций, в которых решается ряд задач математической физики.

Точечная квадратичная аппроксимация. На практике часто бывает, что заданный порядок m приближающего полинома $P_m(x)$ значительно меньше числа узлов n . В этом случае интерполирование становится невозможным и приходится прибегать к иным приемам построения приближающего полинома для данной функции. Обычно здесь используют *точечный способ наименьших квадратов*.

Пусть на отрезке $[a, b]$ задана система точек $x_0, x_1, x_2, \dots, x_{n-1}, x_n$. При точечном квадратичном аппроксимировании за меру отклонения полинома

$$P_m(x) = a_0 + a_1 x + \dots + a_m x^m \quad (8.11)$$

от данной функции $y = f(x)$ на множестве точек берут величину

$$S_m = \sum_{i=0}^n (P_m(x_i) - f(x_i))^2, \quad (8.12)$$

равную сумме квадратов отклонений полинома $P_m(x)$ от функции $f(x)$ на заданной системе точек. (8.12) – *величина среднеквадратичного отклонения*.

Очевидно, что S_m есть функция коэффициентов $a_0, a_1, \dots, a_m$. Эти коэффициенты надо подобрать так, чтобы величина S_m была наименьшей.

Полученный полином называется *аппроксимирующим* для данной функции, а процесс построения этого полинома – *точечной квадратичной аппроксимацией* или *точечным аппроксимированием* функции.

Если $m < n$, то для минимизации S_m , для решения задачи точечного квадратичного аппроксимирования, воспользуемся общим приемом дифференциального исчисления. Найдем частные производные от величины

$$S_m = \sum_{i=0}^n (a_0 + a_1 x_i + a_2 x_i^2 + \dots + a_m x_i^m - y_i)^2,$$

где $y_i = f(x_i)$, по всем переменным $a_0, a_1, \dots, a_m$. Приравнявая эти частные производные нулю, получим для определения неизвестных $a_0, a_1, \dots, a_m$ систему $m+1$ уравнений с $m+1$ неизвестными

$$\begin{cases} \frac{1}{2} \frac{\partial S_m}{\partial a_0} = \sum_{i=0}^n (a_0 + a_1 x_i + a_2 x_i^2 + \dots + a_m x_i^m - y_i) \cdot 1 = 0 \\ \frac{1}{2} \frac{\partial S_m}{\partial a_1} = \sum_{i=0}^n (a_0 + a_1 x_i + a_2 x_i^2 + \dots + a_m x_i^m - y_i) \cdot x_i = 0 \\ \dots \\ \frac{1}{2} \frac{\partial S_m}{\partial a_m} = \sum_{i=0}^n (a_0 + a_1 x_i + a_2 x_i^2 + \dots + a_m x_i^m - y_i) \cdot x_i^m = 0. \end{cases} \quad (8.13)$$

Введем обозначения

$$s_k = x_0^k + x_1^k + \dots + x_n^k = \sum_{i=0}^n x_i^k, \quad k = 0, 1, 2, \dots; \quad t_k = x_0^k y_0 + x_1^k y_1 + \dots + x_n^k y_n = \sum_{i=0}^n x_i^k y_i, \quad k = 0, 1, \dots$$

Тогда систему (8.13) можно записать в виде

$$\begin{cases} a_0 s_0 + a_1 s_1 + a_2 s_2 + \dots + a_m s_m = t_0 \\ a_0 s_1 + a_1 s_2 + a_2 s_3 + \dots + a_m s_{m+1} = t_1 \\ \dots \\ a_0 s_m + a_1 s_{m+1} + a_2 s_{m+2} + \dots + a_m s_{2m} = t_m. \end{cases} \quad (8.14)$$

где $s_0 = n + 1$.

Если среди точек x_i , $i = \overline{1, n}$ нет совпадающих, то определитель системы (8.14) отличен от нуля и, следовательно, эта система имеет единственное решение. Полином (8.11) с коэффициентами a_i , определенными из (8.14), будет обладать *минимальным квадратичным отклонением* S_{min} .

Если $m = n$, то в качестве полинома $P_m(x)$ можно взять интерполяционный полином Лагранжа $L_n(x)$, так как для него величина среднеквадратичного отклонения равна нулю

$$S_{min} = \sum_{i=0}^n (L_n(x_i) - f(x_i))^2 = 0.$$

Вычислим, например, определитель системы (8.14), когда $m = 1$. Тогда

$$\begin{cases} a_0 s_0 + a_1 s_1 = t_0 \\ a_0 s_1 + a_1 s_2 = t_1 \end{cases}$$

и $\det = s_0 s_2 - s_1^2$. Учитывая неравенство Коши-Буняковского $\left(\sum_{i=0}^n x_i\right)^2 \leq \left(\sum_{i=0}^n x_i^2\right)(n+1)$,

для попарно различных узлов x_i , получим $\det = (n+1)\sum_{i=0}^n x_i^2 - \left(\sum_{i=0}^n x_i\right)^2 > 0$. Отметим,

что матрица системы (8.14) положительно определена, и для ее решения можно использовать, например, метод Зейделя или метод простой итерации.

Для составления системы (8.14) рекомендуется схема способа наименьших квадратов, приведенная в таблице, где принято $m=2$.

x_i^0	x_i	x_i^2	x_i^3	x_i^4	y_i	$x_i y_i$	$x_i^2 y_i$
1	x_0	x_0^2	x_0^3	x_0^4	y_0	$x_0 y_0$	$x_0^2 y_0$
1	x_1	x_1^2	x_1^3	x_1^4	y_1	$x_1 y_1$	$x_1^2 y_1$
1	x_2	x_2^2	x_2^3	x_2^4	y_2	$x_2 y_2$	$x_2^2 y_2$
1	x_3	x_3^2	x_3^3	x_3^4	y_3	$x_3 y_3$	$x_3^2 y_3$
1	x_4	x_4^2	x_4^3	x_4^4	y_4	$x_4 y_4$	$x_4^2 y_4$
S_0	S_1	S_2	S_3	S_4	t_0	t_1	t_2

В общем случае, когда аппроксимирующий полином для данной функции $f(x)$ является обобщенным

$$P_m(x) = a_0 \varphi_0(x) + a_1 \varphi_1(x) + \dots + a_m \varphi_m(x),$$

для нахождения его коэффициентов $a_0, a_1, \dots, a_m$ по способу наименьших квадратов приходится минимизировать сумму квадратов

$$S_m = \sum_{i=0}^n (a_0 \varphi_0(x_i) + a_1 \varphi_1(x_i) + \dots + a_m \varphi_m(x_i) - f(x_i))^2,$$

где $x_0, x_1, \dots, x_n$ – заданная система точек. Используя необходимые условия экстремума, получаем систему уравнений

$$\begin{cases} c_0(\varphi_0, \varphi_0) + c_1(\varphi_1, \varphi_0) + \dots + c_m(\varphi_m, \varphi_0) = (f, \varphi_0) \\ c_0(\varphi_0, \varphi_1) + c_1(\varphi_1, \varphi_1) + \dots + c_m(\varphi_m, \varphi_1) = (f, \varphi_1) \\ \dots\dots\dots \dots\dots\dots \dots\dots\dots \dots\dots\dots \dots\dots\dots \\ c_0(\varphi_0, \varphi_m) + c_1(\varphi_1, \varphi_m) + \dots + c_m(\varphi_m, \varphi_m) = (f, \varphi_m) \end{cases}.$$

Из этой системы определяются коэффициенты $a_0, a_1, \dots, a_m$.

Понятие о равномерном приближении функций. Квадратичная аппроксимация функций основывалась на способе наименьших квадратов. Уточняя понятие квадратичного отклонения, введём соответствующее

расстояние Δ между данной непрерывной функцией $f(x)$ и непрерывным аппроксимирующим обобщенным полиномом $Q_m(x)$ так называемое среднее квадратичное отклонение.

Под **средним квадратичным отклонением функции** $f(x)$ и полинома $Q_m(x)$ на множестве точек $X = \{x_0, x_1, \dots, x_n\}$ понимается число

$$\Delta = \sqrt{\frac{1}{n} \sum_{i=0}^n (f(x_i) - Q_m(x_i))^2} \quad (8.15)$$

Если функция $f(x)$ задана аналитически, в этом случае аппроксимация называется **интегральной**, а среднее квадратичное отклонение на отрезке $[a, b]$ определяется формулой

$$\Delta = \sqrt{\frac{1}{b-a} \int_a^b (f(x) - Q_m(x))^2 dx}. \quad (8.16)$$

Формулу (8.16) можно рассмотреть как предельный случай формулы (8.15) при $n \rightarrow \infty$. Действительно, выбирая на отрезке $[a, b]$ систему равноотстоящих точек

$$a = x_1, x_2, \dots, x_n = b, \text{ где } \Delta x_i = x_{i+1} - x_i = \frac{b-a}{n}, (i = 1, 2, \dots, n-1),$$

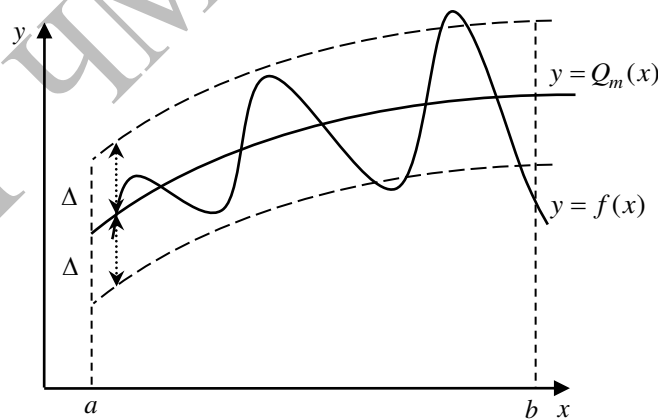
будем иметь

$$\Delta_n^2 = \frac{1}{n} \sum_{i=1}^n (f(x_i) - Q(x_i))^2 = \frac{1}{b-a} \sum_{i=1}^n (f(x_i) - Q(x_i))^2 \Delta x_i.$$

Отсюда при $n \rightarrow \infty$ получим,

$$\Delta^2 = \lim_{n \rightarrow \infty} \Delta_n^2 = \frac{1}{b-a} \int_a^b (f(x) - Q(x))^2 dx.$$

Если среднее квадратичное отклонение Δ мало, то для большинства значений x из $[a, b]$ абсолютная величина $|f(x) - Q(x)|$ также мала при $x \in [a, b]$. Следовательно $\Delta < \varepsilon$, где $\varepsilon > 0$.

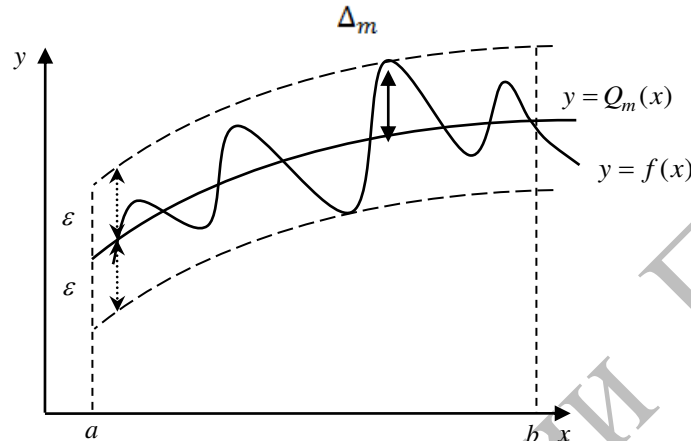


Такая ситуация наблюдается в ряде случаев при обработке наблюдений. Однако с помощью приближения функции мы можем получить закон изменения результатов наблюдений. Во многих случаях, например, при обработке результатов эксперимента оно сглаживает отдельные локальные неровности. Однако иногда для приближения функций ставят более жесткие условия: требуются гарантировать, чтобы отклонение на отрезке $[a, b]$

функции $f(x)$ и полинома $Q(x)$ было меньше заданной величины, то есть равно величине абсолютного отклонения.

Абсолютным отклонением на отрезке $[a, b]$ обобщенного полинома $Q(x)$ от непрерывной функции $f(x)$ называется число

$$\Delta(f, Q_m) = \Delta_m = \max_{x \in [a, b]} |f(x) - Q_m(x)|. \quad (8.17)$$



Если $\Delta_m \leq \varepsilon$, то из формулы (8.17) следует $|f(x) - Q_m(x)| \leq \varepsilon$ для всех точек x на отрезке $[a, b]$. Тогда говорят, что обобщенный полином $Q_m(x)$ на отрезке $[a, b]$ *равномерно приближает функцию $f(x)$ с точностью до ε* .

Для случая алгебраических многочленов

$$Q_m(x) = \sum_{k=0}^m a_k x^k$$

справедлива теорема Вейерштрасса.

Теорема Вейерштрасса. Если функция $f(x)$ непрерывна на $[a, b]$ то, как бы мало ни было положительное число ε , найдется полином $Q_m(x)$ достаточно высокой степени $m < n$ абсолютное отклонение которого от данной функции $f(x)$ на отрезке $[a, b]$ меньше ε , то есть для всех точек $x \in [a, b]$ имеет место неравенство $|f(x) - Q_m(x)| < \varepsilon$.

Замечание 8.1. Контроль правильности вычисления

1. Прежде всего, после решения системы уравнений, например, (8.14), необходимо подставить коэффициенты в уравнения системы (8.14) и проверить, совпадает ли значение левой части с правой;

2. Поскольку аппроксимирующий многочлен не совпадает со значением функции на заданном множестве точек, то следует вычислять по формуле (8.15) величину Δ для точечного метода наименьших квадратов или по формуле (8.16) для интегрального метода наименьших квадратов. Если после подстановки в многочлены значения x , где известно значение функции, отклонение между ними не будет превосходить величину Δ , то аппроксимирующий многочлен построен правильно.

Метод ортогональных полиномов. Если степень аппроксимирующего полинома сравнительно велика, то вычисления по способу наименьших квадратов становятся громоздкими. В этом случае иногда выгодно использовать новый метод построения аппроксимирующего полинома, основанный на понятии ортогональных функций.

Две функции $\varphi(x)$ и $\psi(x)$ называются **ортогональными на множестве точек** $X = \{x_0, x_1, \dots, x_n\}$, если $\sum_X \varphi(x_i)\psi(x_i) = 0$.

Система функций $\{\varphi(x_i)\}$ называется **ортогональной на данном множестве** X , если функции этой системы попарно ортогональны между собой на этом множестве X .

Функция $\varphi(x)$ обращающаяся в нуль в точках, $x_0, x_1, x_2, \dots, x_{n-1}, x_n$ ортогональна на этом множестве точек к любой другой функции. Поэтому в дальнейшем будем предполагать, что

$$\sum_{i=0}^n \varphi^2(x_i) > 0,$$

т.е. не все точки x_i ($i = 0, 1, \dots, n$) являются нулями рассматриваемых функций $\varphi(x)$.

Назовем величину

$$\sqrt{\sum_{i=0}^n \varphi^2(x_i)} = \|\varphi\|_X,$$

нормой функции $\varphi(x)$ на множестве X .

Если система $\{\varphi_k(x)\}$ ортогональна на множестве X и для всех k выполнено равенство

$$\|\varphi_k(x)\| = 1,$$

то такая система функций называется **ортонормированной**.

Функции $f_k(x), k = 0, 1, \dots, m$ называются **линейно независимыми на множестве** $X = \{x_0, x_1, \dots, x_n\}$, если они определены на этом множестве и из равенств $\alpha_0 f_0(x_i) + \alpha_1 f_1(x_i) + \dots + \alpha_m f_m(x_i) = 0, (i = 0, 1, \dots, n)$ следует, что все постоянные $\alpha_k = 0$ ($k = 0, 1, \dots, m$). В противном случае функции $f_k(x)$ называются **линейно зависимыми на X .**

Лемма 8.1. Функции $\varphi_k(x), k = 0, 1, \dots, m$, ортогональные на множестве $X = \{x_0, x_1, \dots, x_n\}$ и имеющие ненулевые нормы, линейно независимы на X .

Рассмотрим систему ортогональных полиномов

$$P_0(x), P_1(x), \dots, P_m(x) \quad (8.18)$$

на множестве X , т.е.

$$\sum_{i=0}^n P_j(x_i) \cdot P_k(x_i) = 0 \quad (8.19)$$

для всех $j \neq k$, и таких, что $s_j = \|P_j\|_X^2 = \sum_{i=0}^n P_j^2(x_i) > 0$. Так как полиномы $P_j(x)$ линейно независимы на X в силу леммы следует линейная независимость на $(-\infty, +\infty)$, то произвольный многочлен $Q_m(x)$ можно представить в виде линейных комбинаций полиномов

$$Q_m(x) = b_0 P_0(x) + b_1 P_1(x) + \dots + b_m P_m(x), \quad (8.20)$$

где $b_0, b_1, \dots, b_m$ – некоторые постоянные числа.

Это $Q_m(x)$ представление называется **разложением полинома по системе (8.18)**. Коэффициенты b_i в разложении (8.20) могут быть найдены путем последовательного деления.

В случае если полиномы $P_j(x)$ ($j = 0, 1, \dots, m$) ортогональны на множестве точек X , можно получить явные формулы для коэффициентов $b_0, b_1, \dots, b_m$ разложения (8.20). Для этого умножим (8.20) на $P_k(x)$, $k \leq m$ и просуммируем полученное выражение по системе точек $x_0, x_1, x_2, \dots, x_{n-1}, x_n$.

Тогда получим

$$\begin{aligned} \sum_{i=0}^n Q_m(x_i) P_k(x_i) &= \\ &= b_0 \sum_{i=0}^n P_0(x_i) P_k(x_i) + b_1 \sum_{i=0}^n P_1(x_i) P_k(x_i) + \dots \\ &+ b_k \sum_{i=0}^n P_k^2(x_i) + \dots + b_m \sum_{i=0}^n P_m(x_i) P_k(x_i). \end{aligned}$$

Учитывая условия ортогональности (13.19), находим

$$\sum_{i=0}^n Q_m(x_i) \cdot P_k(x_i) = b_k \sum_{i=0}^n P_k^2(x_i)$$

и, следовательно,

$$b_k = \frac{\sum_{i=0}^n Q_m(x_i) \cdot P_k(x_i)}{\sum_{i=0}^n P_k^2(x_i)}, \quad k = 0, 1, \dots, m. \quad (8.21)$$

Коэффициенты b_k ($k = 0, 1, \dots, m$) вычисленные по формулам (8.21) называются коэффициентами Фурье полинома $Q_m(x)$ относительно данной системы ортогональных на X функций $P_k(x)$ ($k = 0, 1, \dots, m$). Если эта система ортонормирована на X , т.е.

$$\sum_{i=0}^n P_k^2(x_i) = 1, \quad (k = 0, 1, \dots, m),$$

то формулы (8.21) принимают более простой вид

$$b_k = \sum_{i=0}^n Q_m(x_i) \cdot P_k(x_i), \quad k = 0, 1, \dots, m. \quad (8.22)$$

Вернемся к задаче аппроксимирования заданной функции $y = f(x)$ на множестве точек X полиномом данной степени m ($m \leq n$). Искомый полином $Q_m(x)$ для которого квадратичное отклонение

$$S_m = \sum_{i=0}^n [Q_m(x_i) - f(x_i)]^2 = \min ,$$

будем искать в виде (8.20).

Вычислим величину S_m для случая, когда b_k являются коэффициентами Фурье. Тогда будем иметь

$$\begin{aligned} S_m &= \sum_{i=0}^n (b_0 P_0(x_i) + b_1 P_1(x_i) + \dots + b_m P_m(x_i) - y_i)^2 = \\ &= \sum_{i=0}^n \left[\sum_{j=0}^m b_j^2 P_j^2(x_i) + 2 \sum_{k=0}^{m-1} \sum_{j=k+1}^m b_j b_k P_j(x_i) P_k(x_i) - 2 \sum_{j=0}^m b_j P_j(x_i) y_i + y_i^2 \right] = \\ &= \sum_{j=0}^m (b_j^2 s_j - 2 b_j c_j) + \sum_{i=0}^n y_i^2, \end{aligned}$$

где $s_j = \sum_{i=0}^n P_j^2(x_i)$, $c_j = \sum_{i=0}^n P_j(x_i) y_i$.

Выделим в последнем выражении полный квадрат

$$S = \sum_{j=0}^m s_j \left(b_j - \frac{c_j}{s_j} \right)^2 - \sum_{j=0}^m \frac{c_j^2}{s_j} + \sum_{i=0}^n y_i^2 .$$

Ясно, что минимальное значение S принимает тогда, когда

$$\sum_{j=0}^m s_j \left(b_j - \frac{c_j}{s_j} \right)^2 = 0 \quad \text{или} \quad b_j = \frac{c_j}{s_j} = \frac{\sum_{i=0}^n P_j(x_i) y_i}{\sum_{i=0}^n P_j^2(x_i)} .$$

Таким образом, искомый полином $Q_m(x)$ имеет вид

$$Q_m(x) = \sum_{j=0}^m \frac{c_j}{s_j} P_j(x) . \quad (8.23)$$

Величину квадратичного отклонения полинома $Q_m(x)$ можно вычислить по формуле

$$S = \sum_{i=0}^n y_i^2 - \sum_{j=0}^m \frac{c_j^2}{s_j} . \quad (8.24)$$

Интегральная квадратичная аппроксимация функций. Функцию $f(x) \in L_2$ из пространства L_2 можно приближать с помощью обобщенного полинома $Q(x)$ вида

$$Q(x) = c_0 \varphi_0(x) + c_1 \varphi_1(x) + \dots + c_m \varphi_m(x) , \quad (8.25)$$

так, чтобы величина $J_m(c_0, c_1, \dots, c_m) = \int_a^b [Q(x) - f(x)]^2 dx$ принимала минимальное значение.

Система интегрируемых функций $\{\varphi_i(x)\}$ называется **ортogonalной** на отрезке $[a, b]$, если $(\varphi_i, \varphi_k) = \int_a^b \varphi_i(x) \varphi_k(x) dx = 0$ для всех $i \neq k$. Величина $\|\varphi_m\| = (\varphi_m, \varphi_m)^{\frac{1}{2}}$ называется **евклидовой нормой** функции $\varphi_m(x)$. Если $\|\varphi_m\| = 1$ для любого m , то система функций $\{\varphi_i(x)\}$ называется **ортонормированной**.

Пусть в (8.25) система $\{\varphi_i(x)\}$ ортogonalна. Определим коэффициенты разложения c_i из условия $\frac{\partial J_m}{\partial c_j} = 0, j = 0, 1, \dots, m$. Дифференцируя J_m , получим систему

$$\frac{1}{2} \frac{\partial J_m}{\partial c_j} = \sum_{i=0}^m c_i (\varphi_i, \varphi_j) - (f, \varphi_j) = 0, \quad j = 0, 1, \dots, m. \quad (8.26)$$

Учитывая ортogonalность системы $\{\varphi_i(x)\}$, вычислим коэффициенты c_i

$$c_i = \frac{(f, \varphi_i)}{(\varphi_i, \varphi_i)} = \frac{(f, \varphi_i)}{\|\varphi_i\|^2}.$$

Если система $\{\varphi_i(x)\}$ ортонормированна, то $c_i = \int_a^b f(x) \varphi_i(x) dx$. Коэффициенты c_i называются **коэффициентами Фурье** функции $f(x)$ относительно заданной ортogonalной системы $\{\varphi_i(x)\}$ $i = 0, 1, \dots, m$. Покажем, что коэффициенты Фурье доставляют минимум функции J_m . Составим второй дифференциал

$$d^2 J_m = \sum_{i=0}^m \frac{\partial^2 J_m}{\partial c_i^2} dc_i^2 + 2 \sum_{j < i} \frac{\partial^2 J_m}{\partial c_j \partial c_i} dc_j dc_i.$$

Так как $\frac{\partial^2 J_m}{\partial c_j \partial c_i} = 2(\varphi_i, \varphi_j) = 0$, получим

$$d^2 J_m = \sum_{i=0}^m \frac{\partial^2 J_m}{\partial c_i^2} dc_i^2 = 2 \sum_{i=0}^m \|\varphi_i\|^2 dc_i^2.$$

Если $\sum_{i=0}^m dc_i^2 \neq 0$, то $d^2 J_m > 0$ и, следовательно, коэффициенты c_i доставляют минимум J_m .

Таким образом, обобщенный полином $Q(x)$ с коэффициентами Фурье имеет наименьшее квадратичное отклонение от функции $f(x)$ по сравнению с другими обобщенными полиномами того же порядка. Вычислим величину этого отклонения

$$J_m = \int_a^b [f(x) - \sum_{i=0}^m c_i \varphi_i(x)]^2 dx = \int_a^b [f^2(x) - 2 \sum_{i=0}^m c_i \varphi_i(x) f(x) + \sum_{i=0}^m c_i^2 \varphi_i^2(x)] dx =$$

$$\int_a^b f^2(x) dx - \sum_{i=0}^m c_i^2 \int_a^b \varphi_i^2(x) dx.$$

Отсюда получаем $J_m = \|f\|^2 - \sum_{i=0}^m c_i^2 \|\varphi_i(x)\|^2 \geq 0$.

Лекция 9. Итерационные методы решения нелинейных уравнений и систем уравнений (6ч. – 2ч. ЛК+4ч. ЛР)

1. Постановка задачи поиска корня нелинейного уравнения.
2. Локализация корней, методы уточнения корня – метод бисекции, метод простой итерации. Достаточное условие сходимости метода простой итерации. Приведение к виду, удобному для применения метода.
3. Метод Ньютона. Достоинства и недостатки метода Ньютона.
4. Скорость сходимости итерационных методов решения нелинейных уравнений.
5. Решение систем нелинейных уравнений: методы простых итераций, Ньютона.

Теоретические аспекты и общая постановка задачи. Одной из важных задач прикладной математики является задача решения нелинейных уравнений, встречающихся в разных областях научных исследований.

Под нелинейными уравнениями понимаются алгебраические и трансцендентные уравнения с одним неизвестным в следующем виде

$$f(x) = 0, \quad (9.1)$$

где x – действительное число, $f(x)$ – заданная нелинейная функция.

Алгебраическое уравнение – это уравнение содержащие только алгебраические функции, которое можно представить многочленом n -ой степени с действительными коэффициентами (целые, рациональные, иррациональные) в следующем виде

$$y = a_0 \cdot x^n + a_1 \cdot x^{n-1} + \dots + a_n.$$

Трансцендентное уравнение – это уравнение содержащие в своем составе функции, которые являются не алгебраическими. Простейшими примерами таких функций служат показательная функция, тригонометрическая функция, логарифмическая функция и т.д.

Решением нелинейного уравнения называют совокупность (группу) чисел $\{\lambda_1, \lambda_2, \dots, \lambda_n\} \in R$, которые, будучи подставлены на место неизвестной $\{x_1, x_2, \dots, x_n\}$, обращают уравнение в тождество

$$f(\lambda) = 0.$$

Для решения нелинейных уравнений существует несколько методов: графические, аналитические и численные методы.

Графические методы наименее точны, но позволяют в сложных уравнениях определить наиболее приближенные значения, с которых в дальнейшем можно начинать находить более точные решения уравнений.

Аналитические методы (или прямые методы) позволяют определить точные значения решения уравнений. Данный метод позволяет записать корни в виде некоторого соотношения (формул). Подобные методы развиты для решения простейших тригонометрических, логарифмических, показательных, а также алгебраических уравнений. Однако подавляющее

большинство нелинейных уравнений, встречающихся на практике, не удастся решить прямыми методами. В таких случаях обращаются к **численным методам**, позволяющим получить приближенное значение корня с любой заданной точностью ε .

Численные методы решения нелинейных уравнений – это итерационный процесс расчета, который состоит в последовательном уточнении начального приближения значений корней уравнения. При численном подходе задача о решении нелинейных уравнений разбивается на два этапа:

Требуется найти корни уравнения (9.1). Эта задача решается в два этапа:

- 1) отделение корней (локализация), т.е. определение отрезков $[a, b]$, в которых содержится только один корень уравнения;
- 2) уточнение корней – доведение их до заданной точности.

Под **локализацией корней** понимается процесс отыскания приближенного значения корня или нахождение таких отрезков, в пределах которых содержится единственное решение.

Под **уточнением корней** понимается процесс вычисления приближенных значений корней с заданной точностью по любому численному методу решения нелинейных уравнений.

Недостатком почти всех итерационных методов нахождения корней является то, что они при однократном применении позволяют найти лишь один корень функции, к тому же, мы не знаем какой именно. В случае повторения итерационного процесса при изменении стартовых точек отсутствуют гарантии, что найдется новый корень уравнения, так как итерационный процесс может сойтись к найденному корню.

Отделение корней. Для отделения корней полезна следующая теорема из математического анализа.

Теорема 9.1. (Теорема о существовании корней) Если непрерывная функция $f(x)$ принимает значения разных знаков на концах отрезка $[\alpha, \beta]$, т.е. $f(\alpha)f(\beta) < 0$, то внутри этого отрезка содержится по меньшей мере один корень уравнения $f(x) = 0$, т.е. найдется хотя бы одно число $\xi \in (\alpha, \beta)$ такое, что $f(\xi) = 0$ (рисунок 9.1).

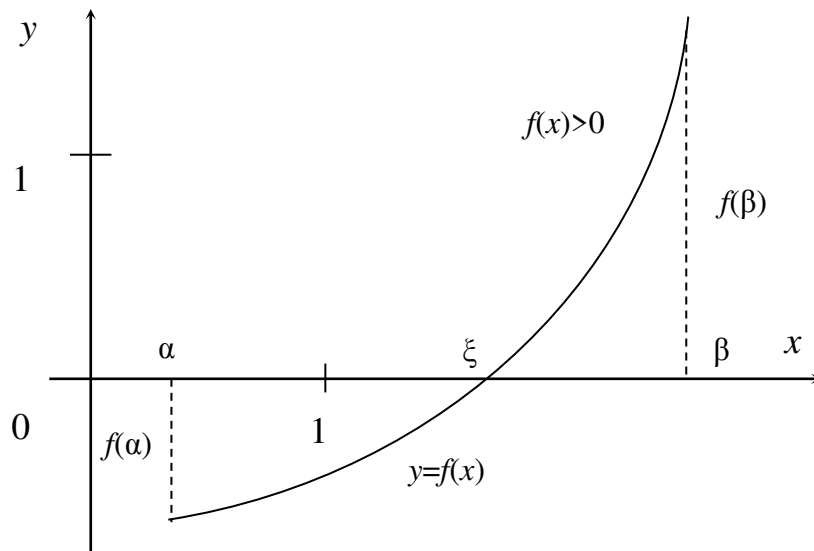


Рисунок 9.1 – Теорема о существовании корней

Корень ξ заведомо будет единственным, если производная $f'(x)$ существует и сохраняет постоянный знак внутри интервала (α, β) , т.е. если $f'(x) > 0$ (или $f'(x) < 0$) при $\alpha < x < \beta$ (рисунок 9.1).

Процесс отделения корней начинается с установления знаков функции $f(x)$ в граничных точках $x = a$ и $x = b$ области её существования. Затем определяются знаки функции $f(x)$ в ряде промежуточных точек $x = \alpha_1, \alpha_2, \dots$, выбор которых учитывает особенности функции $f(x)$. Если окажется, что $f(\alpha_k)f(\alpha_{k+1}) < 0$, то в силу теоремы 9.1 в интервале (α_k, α_{k+1}) имеется корень уравнения $f(x) = 0$. Нужно тем или иным способом убедиться, является ли этот корень единственным. Для отделения корней практически часто бывает достаточно провести процесс половинного деления, приближенно деля данный интервал (α, β) на две, четыре, восемь и т.д. равных частей (до некоторого шага) и определяя знаки функции $f(x)$ в точках делений.

Полезно помнить, что алгебраическое уравнение n -й степени

$$a_0 x^n + a_1 x^{n-1} + \dots + a_{n-1} x + a_n = 0$$

имеет не более n действительных корней. Поэтому, если для такого уравнения мы получили n переменных знаков, то все корни его отделены.

Аналитический метод отделения корней. Аналитически корень уравнения можно определить, используя некоторые свойства функции, изучаемы в курсе математического анализа. Если в уравнении (9.1) функция $y = f(x)$ непрерывная, то следует воспользоваться следующими известными фактами:

1 – Если на концах некоторого отрезка непрерывная функция принимает значения разных знаков, то на этом отрезке уравнение (9.1) имеет, по крайней мере, один корень.

2 – Если при этом функция имеет первую производную, не меняющую знака, то корень будет единственным.

3 – Пусть аналитическая функция $y = f(x)$ на концах $[a, b]$ принимает значения разных знаков, т.е. $f(a)f(b) < 0$, то между a и b имеется нечетное число корней уравнения (9.1); если же на концах $[a, b]$ функция принимает значения одинаковых знаков, т.е. $f(a)f(b) > 0$, то между a и b или нет корней, или их имеется четное число (с учетом кратности);

Для непрерывной на отрезке $[a, b]$ функции $f(x)$ можно предложить следующий порядок действий для отделения корней аналитическим методом:

1. Найти $f'(x)$ и определить критические точки.
2. Составить таблицу знаков функции $f(x)$, полагая x равным:
 - а) критическим значениям производных или ближайшим к ним;
 - б) граничным значениям области допустимых значений неизвестных.
3. Отделить интервалы, на концах которых функция принимает значения разных знаков. Внутри этих интервалов содержится по одному корню.

Пример 9.1. Отделить корни уравнения $f(x) = x^3 - 6x + 2 = 0$.

Решение. Составляем приближительную схему

x	$-\infty$	-3	-1	0	1	3	$+\infty$
$sign f(x)$	-	-	+	+	-	+	+

Из этой схемы видно, что рассматриваемое уравнение имеет три действительных корня, лежащих в интервалах $(-3, -1)$, $(0, 1)$ и $(1, 3)$, из таблицы видно, что имеет место 3 смены знаков функции.

Графический метод отделения корней. Для графического метода отделения корней существует два способа:

1. Строим график функции $y = f(x)$ (рисунок 9.2).

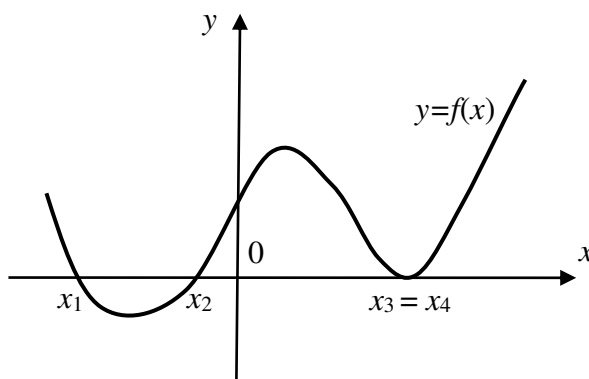


Рисунок 9.2 – Первый способ отделения корней

Абсциссы точек пересечения графика функции с осью Ox будут грубыми значениями корней уравнения (9.1). За окрестности корней принимаются интервалы, внутри которых находится единственный корень.

2. Представим функцию $f(x)$ в виде $f(x) = \varphi(x) - \psi(x)$, где функции $\varphi(x)$ и $\psi(x)$ имеют более простой вид, чем функция $f(x)$. Построим графики функций $\varphi(x)$ и $\psi(x)$. Абсциссы точек пересечения этих графиков будут приближенными значениями корней. Затем выделяют интервалы внутри, которых находится только один корень.

Пример 9.2. Отделить корни уравнения $x - e^{-x} = 0$.

Решение. Пусть $\varphi(x) = x$, $\psi(x) = e^{-x}$. Строим графики этих функций, рисунок 9.3.

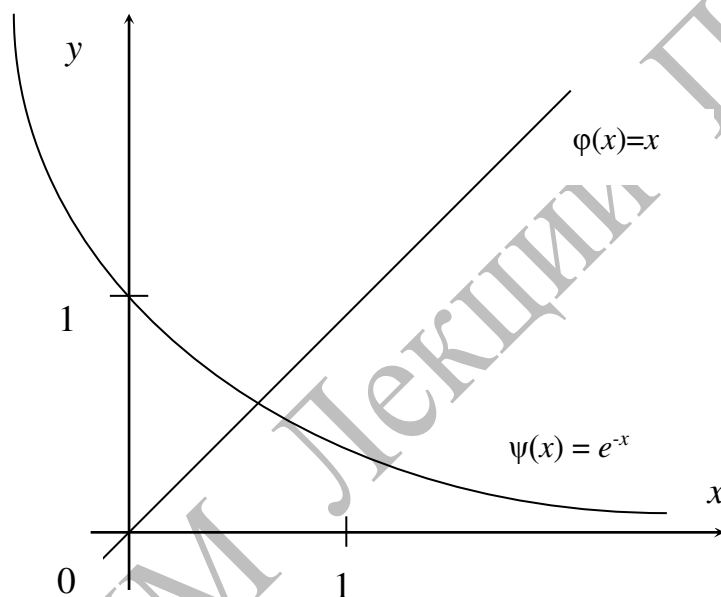


Рисунок 9.3 – Второй способ отделения корней. Графики функций $\varphi(x)$ и $\psi(x)$

Из графиков построенных функций следует, что корень уравнения находится на отрезке $[0, 1]$.

Определение числа действительных корней. Пусть дано алгебраическое уравнение n степени

$$f(x) = a_0 x^n + a_1 x^{n-1} + \dots + a_{n-1} x + a_n = 0, \quad (9.2)$$

где $a_0, a_1, \dots, a_n$ – действительные коэффициенты.

Определить приближенно **число** действительных корней уравнения (9.2) можно с помощью **правила Декарта**: число положительных корней уравнения (9.2) с учетом их кратностей равно числу перемен знаков в последовательности коэффициентов $a_0, a_1, \dots, a_n$ (причем равные нулю коэффициенты не учитываются) или меньше этого числа на четное число. Для определения числа отрицательных корней достаточно применить правило Декарта к многочлену $f(-x)$.

Более точно число корней уравнения (9.2) можно определить, **используя теорему Штурма**: Пусть уравнение (9.2) не имеет кратных корней на некотором отрезке $[a, b]$. Обозначим через $f_1(x)$ производную $f'(x)$; через $f_2(x)$ остаток от деления $f(x)$ на $f_1(x)$, взятый с обратным знаком; через $f_3(x)$ остаток от деления $f_1(x)$ на $f_2(x)$, взятый с обратным знаком; и т.д. до тех пор, пока не получим $f_n(x) = \text{const}$. Получился так называемый ряд Штурма

$$f(x), f_1(x), f_2(x), \dots, f_n(x). \quad (9.3)$$

Число действительных корней уравнения (9.2), расположенных на отрезке $[a, b]$, равно разности между числом перемен знаков в последовательности (9.3) при $x = a$ и числом перемен знаков в последовательности (9.3) при $x = b$.

Практическое применение теоремы Штурма сводится к следующему: определяются границы отрезка, на котором расположены действительные корни уравнения (9.2) и их число. Полученный отрезок $[a, b]$ делится на некоторое число частей точками α_i : $a = \alpha_0 < \alpha_1 < \alpha_2 < \dots < \alpha_{n-1} < \alpha_n = b$. Рассматривая отрезок $[\alpha_i, \alpha_{i+1}]$ по теореме Штурма, определяется число корней на этом отрезке. Если окажется, что их больше одного, то этот отрезок делится пополам и теорема Штурма применяется к каждому полученному отрезку. Процесс продолжается до тех пор, пока на каждой части отрезка $[a, b]$ уравнение (9.2) будет иметь не больше одного действительного корня.

Итак, в вычислительной практике обычно используются следующие способы отделения корней:

- 1) средствами машинной графики: функция представляется на экране ВМ и приближенно определяются отрезки, которым принадлежат точки x_i (рисунок 9.2);
- 2) средствами математического анализа с помощью исследования функций и построения графиков;
- 3) формированием простых функций и таких, что получается равносильное уравнение в виде $f(x) = \varphi(x) - \psi(x)$, и дальнейшим построением графиков этих функций (рисунок 9.3).

Методы уточнения корней. На данном этапе задача состоит в получении приближенного значения корня, принадлежащего отрезку $[a, b]$, с заданной точностью (погрешностью) ε . Это означает, что вычисленное значение корня $\tilde{x}$ должно отличаться от точного x^* не более чем на величину ε , т.е. $|x^* - \tilde{x}| \leq \varepsilon$.

Существует большое количество численных методов решения нелинейных уравнений для уточнения корней:

- метод половинного деления;

- метод хорд;
- метод простой итерации;
- метод Ньютона;
- метод парабол (Метод Мюллера);
- метод Ридерса;
- метод Дэккера и Брэнта;
- метод Лобачевского.

Все методы уточнения основаны на последующем приближении. Часто используемые методы: итераций, хорд, Ньютона, комбинированный.

Метод итераций. Уравнение (9.1) заменяется равносильным ему уравнением

$$x = \varphi(x). \quad (9.4)$$

Допустим, нам известно некоторое начальное приближение x_0 . Тогда, в простейшем методе итераций, все дальнейшие приближения строятся по формуле

$$x_{k+1} = \varphi(x_k), \quad k = 0, 1, 2, \dots \quad (9.5)$$

Если последовательность (9.5) сходится при некотором выборе начального приближения x_0 , то существует предел $\xi = \lim_{k \rightarrow \infty} x_k$ и предполагая функцию $\varphi(x)$ непрерывной, найдем

$$\lim_{k \rightarrow \infty} x_k = \varphi(\lim_{k \rightarrow \infty} x_{k-1}) \text{ или } \xi = \varphi(\xi).$$

Таким образом, ξ является корнем уравнения (9.4) и может быть вычислен по формуле (9.5) с любой степенью точности. Процесс (9.5) называется простой одношаговой итерацией.

Геометрически способ итерации может быть пояснен следующим образом: построим на плоскости xOy графики функций $y = x$ (биссектриса первого координатного угла) и $y = \varphi(x)$. Каждый действительный корень ξ уравнения (9.4) является абсциссой точки пересечения кривой $y = \varphi(x)$ с прямой $y = x$. Следующие два рисунка поясняют сходящийся итерационный процесс (рисунок 9.4–9.5).

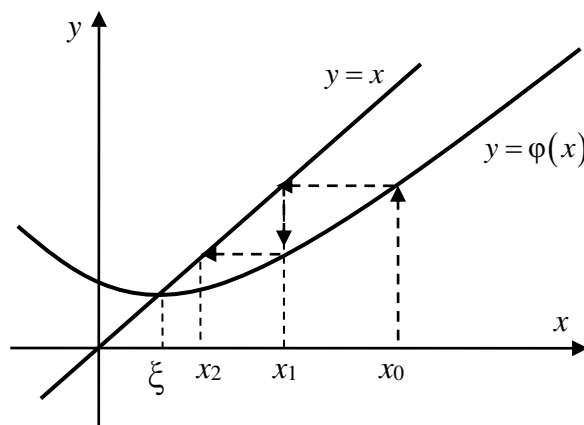


Рисунок 9.4 – Сходящийся процесс при $0 < \varphi'(x) < 1$

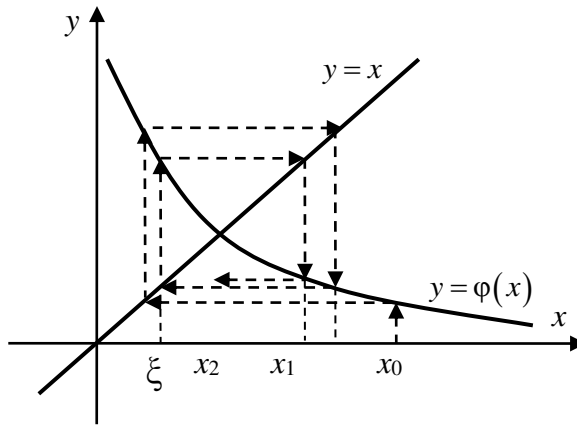


Рисунок 9.5 – Сходящийся процесс при $\phi'(x) < 0$, но $|\phi'(x)| < 1$

На рисунках 9.6 и 9.7 схематично изображен расходящийся итерационный процесс. О динамике процесса можно проследить по стрелкам, имеющимся на рисунках.

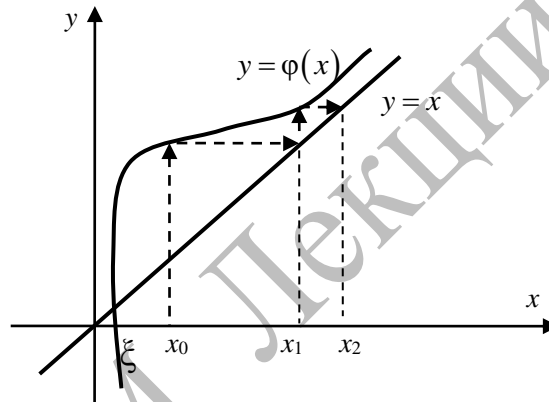


Рисунок 9.6 – Расходящийся процесс при $\phi'(x) < 1$

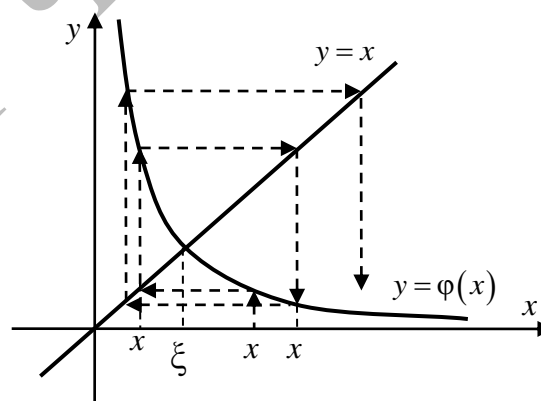


Рисунок 9.7 – Расходящийся процесс при $|\phi'(x)| > 1$

Поэтому для практического применения метода итерации нужно выяснить достаточные условия сходимости итерационного процесса.

Теорема 9.2. Пусть $\phi(x)$ определена и дифференцируема на $[a, b]$ и все её значения принадлежат отрезку $[a, b]$. Если существует такое q , что

$$|\varphi'(x)| \leq q < 1, \quad (9.6)$$

то итерационный процесс (9.5) сходится к единственному корню на $[a, b]$.

Для скорости сходимости итерационного процесса, справедливы соотношения

$$|\xi - x_n| \leq q^n |\xi - x_0|, \quad (9.7)$$

$$|\xi - x_n| \leq \frac{q^n}{1-q} |x_1 - x_0|, \quad (9.8)$$

$$|\xi - x_0| \leq \frac{q}{1-q} |x_n - x_{n-1}|. \quad (9.9)$$

Из (9.7) следует, что метод сходится со скоростью геометрической прогрессии со знаменателем q . Формула (9.8) позволяет определить достаточное число итераций n при выбранном $|x_1 - x_0|$. На практике для оценки сходимости удобно пользоваться (9.9). Так, если $\varepsilon > 0$, $|x_n - x_{n-1}| \leq \frac{1-q}{q} \varepsilon$, то корни уравнения определяются с точностью до ε .

Обычно для получения $\varphi(x)$ пользуются следующим способом приведения (9.1) к виду (9.4). Строят уравнение

$$x = x - \lambda f(x), \quad (9.10)$$

где λ – параметр. Пусть $f'(x)$ постоянного знака на отрезке $[a, b]$. Для нахождения λ положим

$$\lambda = \begin{cases} \frac{k}{M_1}, & f'(x) > 0 \\ -\frac{k}{M_1}, & f'(x) < 0 \end{cases}, \quad k = 1, 2, \dots, \quad M_1 = \max_{x \in [a, b]} |f'(x)|,$$

В этом случае будет верно $|\varphi'(x)| < 1$, значит итерационный процесс по формуле (9.10) будет сходящимся.

Метод хорд. Пусть дано уравнение (9.1) с непрерывными на отрезке $[a, b]$ функцией $f(x)$ и ее производными $f'(x), f''(x)$. Корень считается отделённым на $[a, b]$ и притом единственным.

Рассмотрим случай, когда $f'(x), f''(x)$ – одного знака. Пусть для определенности $f(a) < 0 < f(b)$, $f'(x) > 0, f''(x) > 0$, т.е. $f'(x)f''(x) > 0$. График функции $y = f(x)$ проходит через точки $A_0(a, f(a)), B(b, f(b))$ (рисунок 9.8).

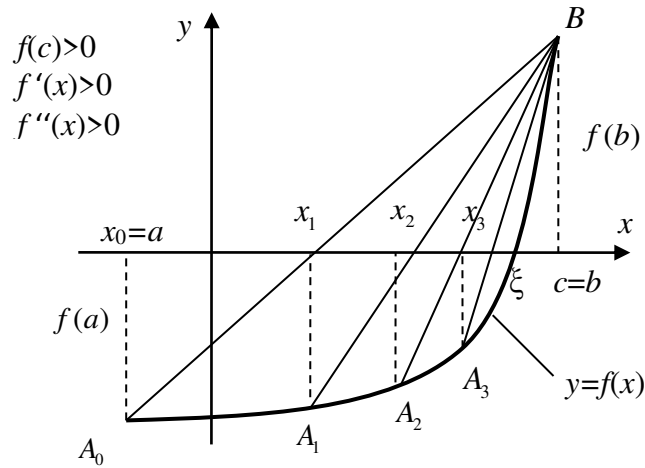


Рисунок 9.8 – Иллюстрация метода хорд $f(c)f''(c) > 0$

Искомый корень ξ уравнения $f(x) = 0$ – есть абсцисса пересечения графика $y = f(x)$ с осью Ox . Эта точка нам известна, но вместо нее можно взять точку x – точку пересечения хорды A_0B с осью Ox . Это и будет первое приближение к корню ξ . Уравнение хорды, проходящей через две точки A_0 и B , запишется

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - a}{b - a}.$$

Положив $y = 0$, найдем $x = x_1$

$$x = a - \frac{f(a)(b - a)}{f(b) - f(a)}.$$

Следовательно, последовательные приближения при этом можно вычислить по следующей итерационной формуле

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - c)}{f(x_n) - f(c)}. \quad (9.11)$$

В формуле (9.11) точка c является неподвижной и в качестве ее берется тот конец $[a, b]$, для которого выполняется условие $f(c)f''(c) > 0$, а за начальное приближение x_0 выбирается противоположный конец $[a, b]$, так что $f(x_0)f(c) < 0$ (в нашем случае $c = b$, $x_0 = a$).

Описанный метод называется также **методом секущих** или **методом линейной интерполяции**. Последовательные приближения в методе хорд образуют – монотонную, ограниченную сверху или снизу корнем ξ последовательность. При этом справедлива оценка

$$|\xi - x_n| \leq \frac{M_1 - m_1}{m_1} |x_n - x_{n-1}|, \quad (9.12)$$

При решении задачи на ЭВМ целесообразно отрезок $[a, b]$ выбирать столь малым, чтобы выполнялось условие $M_1 \leq 2m_1$. В этом случае итерационный процесс сходится быстро.

$$x = x - \frac{f(x)(x-c)}{f(x)-f(c)} = \varphi(x), \text{ где } f(c)f''(c) > 0.$$

94

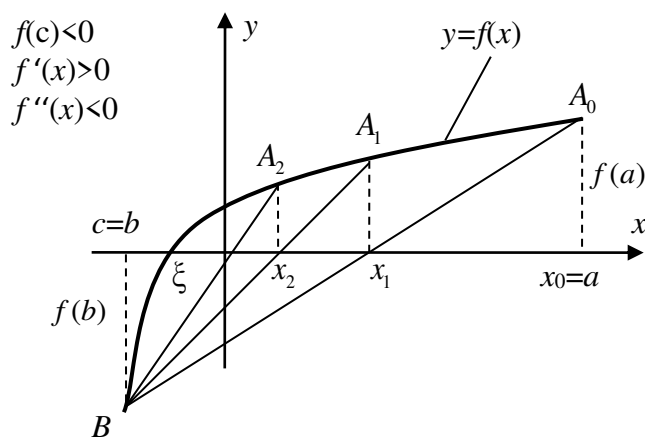


Рисунок 9.11 – Иллюстрация метода хорд $f(c) < 0, f'(c) > 0$

Из приведенного выше следует, что выбор неподвижной точки осуществляется следующим образом:

1 – неподвижен тот конец отрезка, для которого знак функции $f(x)$ совпадает со знаком ее второй производной $f''(x)$.

2 – последовательные приближения x_n лежат по ту сторону корня ξ , где функция $f(x)$ имеет знак, противоположный знаку ее второй производной $f''(x)$.

В обоих случаях каждое следующее приближение x_{n+1} ближе к корню ξ , чем предшествующее x_n .

Метод Ньютона. Метод Ньютона (касательных) позволяет привести решение нелинейных уравнений к решению последовательности линейных задач. Достигается это при помощи выделения из нелинейного уравнения его главной линейной части.

Пусть корень уравнения (9.1) отделен на $[a, b]$, причем $f'(x), f''(x)$ непрерывны и сохраняют определенные знаки и $f'(x)$ не обращается в нуль на $[a, b]$. Метод Ньютона заключается в том, что кривая уравнения $y = f(x)$ заменяется касательной к ней. Имея n -ое приближение корня $\xi \approx x_n \in [a, b]$, уточним его по методу Ньютона следующим образом: положим

$$\xi = x_n + h_n, \quad (9.13)$$

где h_n считаем малой величиной. Теперь, применяя формулу Тейлора, получим

$$f(\xi) = 0 = f(x_n + h_n) = f(x_n) + h_n f'(x_n).$$

Откуда следует, что

$$h_n = -\frac{f(x_n)}{f'(x_n)}.$$

Внося эту поправку в формулу (9.13), найдем следующее приближение корня

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \dots \quad (9.14)$$

Геометрически метод Ньютона (9.14) эквивалентен замене небольшой дуги кривой $y = f(x)$ касательной, проведенной в некоторой точке касания (рисунок 9.12).

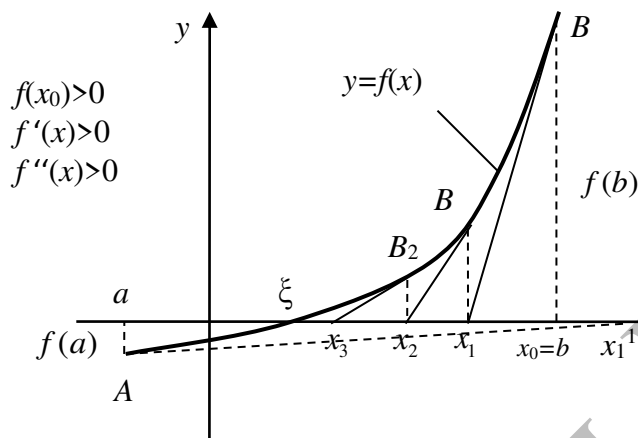


Рисунок 9.12 – Иллюстрация метода Ньютона

Если положить (рисунок 9.12) $x_0 = a$ и, следовательно, $f(x_0)f''(x_0) < 0$ то, проведя касательную к кривой $y = f(x)$ в точке $A(a, f(a))$, получили бы точку x_1^1 , лежащую **вне** отрезка $[a, b]$, и можем не прийти к корню $\xi \approx x_n \in [a, b]$ уравнения (9.1). Поэтому за начальное приближение x_0 в формуле (9.14) выбирается тот конец отрезка $[a, b]$, для которого знак функции совпадает со знаками второй производной, т. е. $f(x_0)f''(x_0) > 0$.

Теорема 9.3. Если $f(a)f(b) < 0$, причем $f'(x), f''(x)$ отличны от нуля и сохраняют определенные знаки при $x \in [a, b]$, удовлетворяющего неравенству $f(x_0)f''(x_0) > 0$, то можно вычислить методом Ньютона по формуле (9.14) единственный корень ξ уравнения (9.1) с любой степенью точности.

Из формулы (9.14) видно, что чем больше численное значение производной $f'(x)$ в окрестности данного корня, тем меньше поправка, которую нужно прибавить к n -му приближению, чтобы получить $(n+1)$ -ое приближение. Поэтому метод Ньютона особенно удобно применять тогда, когда в окрестности данного корня график функции имеет большую крутизну. Но, если численное значение производной $f'(x)$ вблизи корня мало, то поправки будут велики, и вычисление корня по этому методу может оказаться очень долгим, а иногда и вовсе невозможным. Следовательно, если кривая $y = f(x)$ вблизи точки пересечения с осью Ox почти горизонтальна, то применять метод Ньютона для решения уравнения (9.1) не рекомендуется.

Для оценки погрешности n -го приближения x_n можно воспользоваться формулой

$$|\xi - x| \leq \frac{|f(x_n)|}{m_1} \quad (9.15)$$

или формулой

$$|\xi - x| \leq \frac{M_2}{2m_1} (x_n - x_{n-1})^2, \quad (9.16)$$

где $M_2 = \max_{x \in [a,b]} |f''(x)|$, $m_1 = \min_{x \in [a,b]} |f'(x)|$.

В общем случае совпадение с точностью до ε двух последовательных приближений x_n и x_{n+1} вовсе не гарантирует, что с той же точностью совпадает значение x_{n+1} и корень ξ (рисунок 9.13).

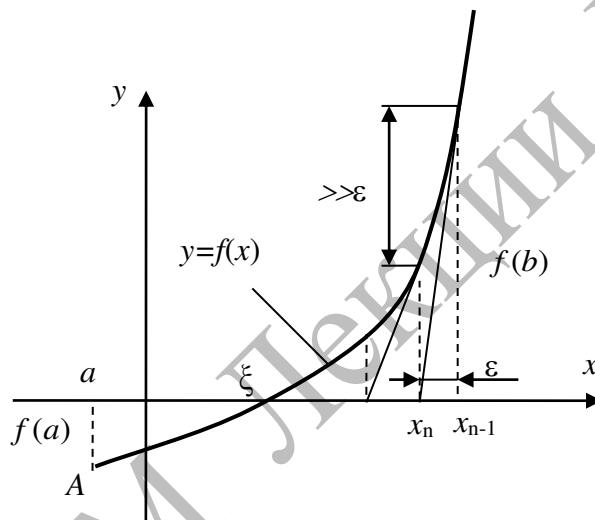


Рисунок 9.13 – О двух последовательных приближениях

Оценки (9.15), (9.16) указывают на квадратичную сходимость метода Ньютона. Поэтому если $f'(x)$ мало меняется на $[a, b]$, то можно пользоваться видоизмененной формулой Ньютона

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}, \quad n = 0, 1, 2, \dots \quad (9.17)$$

В этом случае для получения требуемой точности в отличие от формулы (9.14) необходимо сделать лишь несколько дополнительных шагов.

Комбинированный метод. Комбинируя метод секущих и метод Ньютона, получается нестационарный метод отыскания действительных корней уравнения (9.1). Преимущество получаемого метода заключается в том, что при прежних предположениях относительно $f'(x)$, $f''(x)$ последовательные приближения x_n и x_{n+1} лежат по разные стороны от корня, и поэтому можно следить в процессе вычислений за достигнутой точностью. В то же время он сходится значительно быстрее метода секущих.

Пусть на отрезке $[a, b]$ содержится единственный корень уравнения (9.1), а $f'(x)$, $f''(x)$ на этом отрезке не меняют знаков.

Если $f(a)f''(a) > 0$, то находим x_0 и x_1 по формулам

$$x_0 = a - \frac{f(a)}{f'(a)}, \quad x_1 = \frac{af(b) - bf(a)}{f(b) - f(a)}, \quad (9.18)$$

а следующие приближения находятся по формулам

$$x_{2n} = x_{2n-2} - \frac{f(x_{2n-2})}{f'(x_{2n-2})}, \quad x_{2n+1} = \frac{x_{2n-2}f(x_{2n-1}) - x_{2n-1}f(x_{2n-2})}{f(x_{2n-1}) - f(x_{2n-2})}. \quad (9.19)$$

Если $f(b)f''(b) > 0$, то находим x_0 и x_1 по формулам

$$x_0 = b - \frac{f(b)}{f'(b)}, \quad x_1 = \frac{af(b) - bf(a)}{f(b) - f(a)}, \quad (9.20)$$

а следующие приближения находятся по формулам (9.19).

Геометрическая интерпретации комбинированного метода выглядит следующим образом:

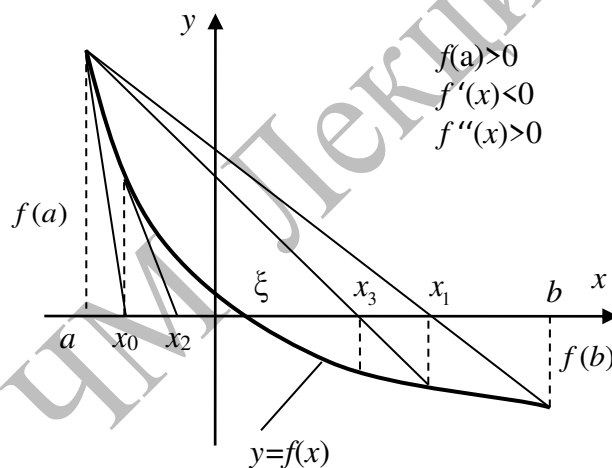


Рисунок 9.14 – Комбинированный метод $f(a) > 0$, $f'(x) < 0$

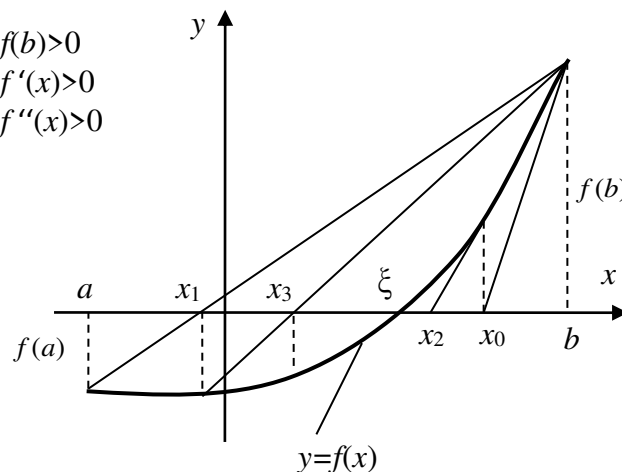


Рисунок 9.15 – Комбинированный метод $f(b) > 0$, $f'(x) > 0$

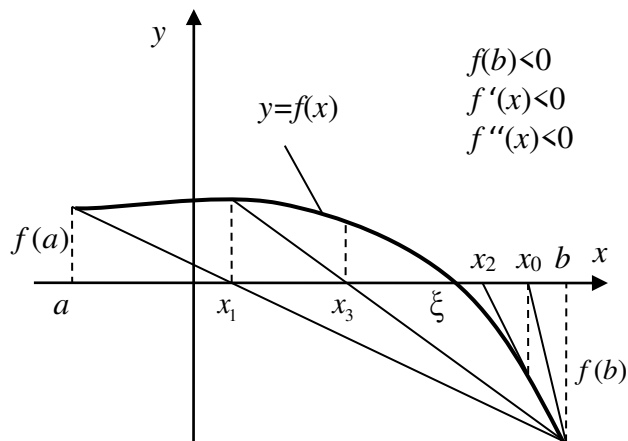


Рисунок 9.16 – Комбинированный метод $f(b) < 0$, $f'(x) < 0$

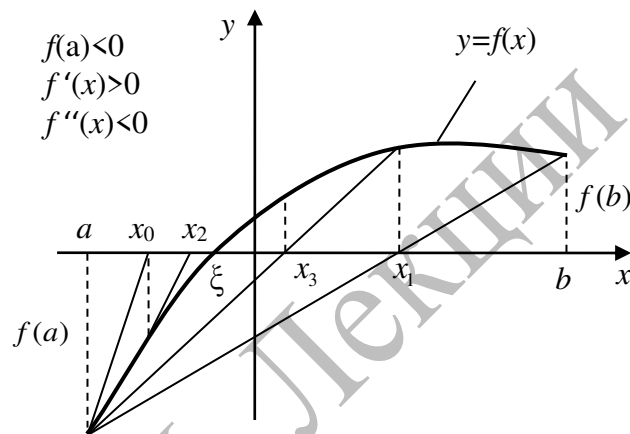


Рисунок 9.17 – Комбинированный метод $f(a) < 0$, $f'(x) > 0$

Решение систем нелинейных уравнений. Общая постановка задачи. Пусть дана система нелинейных уравнений с n неизвестными

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0, \\ f_2(x_1, x_2, \dots, x_n) = 0, \\ \dots \dots \dots \\ f_n(x_1, x_2, \dots, x_n) = 0, \end{cases} \quad (9.21)$$

или $f_i(x_1, x_2, \dots, x_n) = 0, i = 1, \dots, n$,

или, более коротко, в векторной форме

$$f(x) = 0, \quad (9.22)$$

где x – вектор неизвестных величин, f – вектор-функция

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \quad f = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \dots \\ f_n(x) \end{pmatrix} \quad 0 = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \end{pmatrix}$$

Рассматривается задача: с заданной точностью решить систему уравнений (9.21) ((9.22)). Требуется найти такой вектор x , который при подстановке в систему вида (9.21) превращает каждое уравнение в верное числовое равенство.

Особенностью этой задачи является ее сложность. Дело в том, что в общем случае неизвестно количество решений системы (9.21), в том числе и имеет ли система хотя бы одно решение. Если же решение существует, то корни могут быть найдены только приближенно, поскольку для системы (9.21) в принципе не существует прямых методов нахождения решений. В редких случаях для решения такой системы удастся применить метод последовательного исключения неизвестных и свести решение исходной задачи к решению одного нелинейного уравнения с одним неизвестным. Значения других неизвестных величин находятся соответствующей подстановкой в конкретные выражения.

Однако в подавляющем большинстве случаев для решения систем нелинейных уравнений используются итерационные методы.

В дальнейшем предполагается, что ищется изолированное решение нелинейной системы. Как и в случае одного нелинейного уравнения, локализация решения может осуществляться на основе специфической информации по конкретной решаемой задаче (например, по физическим соображениям), и с помощью методов математического анализа. При решении системы двух уравнений, достаточно часто удобным является графический способ, когда месторасположение корней определяется как точки пересечения кривых $f_1(x_1, x_2) = 0$, $f_2(x_1, x_2) = 0$ на плоскости (x_1, x_2) .

Метод простой итерации для систем нелинейных уравнений n -го порядка. Аналогично методу простой итерации для нелинейных уравнений, построим рассуждения относительно решения систем нелинейных уравнений этим методом.

Пусть необходимо решить систему нелинейных уравнений вида (9.21). Приведем эту систему предварительно к виду

$$\begin{cases} x_1 = \Phi_1(x_1, x_2, \dots, x_n), \\ x_2 = \Phi_2(x_1, x_2, \dots, x_n), \\ \dots \quad \dots \quad \dots \\ x_n = \Phi_n(x_1, x_2, \dots, x_n), \end{cases} \quad (9.3)$$

или $x_i = \Phi_i(x_1, x_2, \dots, x_n), i = 1, \dots, n$

или, в векторной форме, $x = \Phi(x)$.

Пусть $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})$ – начальное приближение. Последующие приближения в методе простой итерации находятся по формулам

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}), \\ x_2^{(k+1)} = \varphi_2(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}), \\ \dots \quad \dots \quad \dots \\ x_n^{(k+1)} = \varphi_n(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}), \end{cases} \quad (9.4)$$

или $x_i^{(k+1)} = \varphi_i(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}), i = 1, \dots, n$,

или, в векторной форме, $x^{(k+1)} = \Phi(x^{(k)}), k = 0, 1, \dots$

Если последовательность векторов $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})$ сходится к вектору $x^* = (x_1^*, x_2^*, \dots, x_n^*)$, а функции $\varphi_i(x)$ непрерывны, то вектор x^* является решением системы (9.22).

Допустим, что в некоторой выпуклой области G функции $\varphi_i(x)$ имеют непрерывные первые производные $\frac{\partial \varphi_i(x)}{\partial x_j}$, $M_{i,j} = \max_G \left| \frac{\partial \varphi_i(x)}{\partial x_j} \right|$ и в G система (9.21) имеет единственное решение $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$. Будем считать, что при некотором начальном приближении $x^{(0)}$ все последующие приближения $x^{(k)}$ не выходят из области G .

Теорема 9.4. Для сходимости метода итераций достаточно, чтобы все собственные значения матрицы $M = [M_{i,j}]$ были по модулю меньше 1, а начальное приближение $x^{(0)}$ было достаточно близко к решению α .

Условие теоремы будет выполнено, если какая-нибудь из норм матрицы M меньше единицы. Следовательно, для сходимости метода итерации достаточно выполнения одного из условий

$$\sum_{j=1}^n M_{i,j} < 1, i = \overline{1, n}, \quad \sum_{i=1}^n M_{i,j} < 1, j = \overline{1, n}, \quad \sum_{i,j=1}^n M_{i,j} < 1.$$

Метод простой итерации для систем нелинейных уравнений второго порядка. Пусть дана система двух уравнений с двумя неизвестными

$$\begin{cases} F_1(x, y) = 0 \\ F_2(x, y) = 0 \end{cases} \quad (9.25)$$

действительные корни которых требуется найти с заданной степенью точности.

Предположим, что система (9.25) допускает лишь изолированные корни. Число этих корней и их приближенные значения можно уста-

новить, построив кривые $F_1(x, y) = 0$ и $F_2(x, y) = 0$, и определив координаты их точек пересечения.

Для применения метода итераций система (9.25) приводится к виду

$$\left. \begin{aligned} x &= \varphi_1(x, y) \\ y &= \varphi_2(x, y) \end{aligned} \right\}. \quad (9.26)$$

Функции $\varphi_1(x, y)$, $\varphi_2(x, y)$ называются *итерирующими*.

Алгоритм решения задается формулами

$$\left. \begin{aligned} x_{n+1} &= \varphi_1(x_n, y_n) \\ y_{n+1} &= \varphi_2(x_n, y_n) \end{aligned} \right\}, \quad n = 0, 1, 2, \dots \quad (9.27)$$

где x_0, y_0 – некоторое начальное приближение.

Имеет место следующая

Теорема 9.5. Пусть в некоторой замкнутой окрестности $R (a \leq x \leq A, b \leq y \leq B)$ имеется одно и только одно решение $x = \xi, y = \eta$ системы (9.26). Если

- 1) функции $\varphi_1(x, y)$, $\varphi_2(x, y)$ определены и непрерывно дифференцируемы в R ,
- 2) начальные приближения x_0, y_0 и все последующие приближения $x_n, y_n, n = 1, 2, \dots$ принадлежат R ,
- 3) в R выполнены неравенства

$$\left. \begin{aligned} \left| \frac{\partial \varphi_1}{\partial x} \right| + \left| \frac{\partial \varphi_2}{\partial x} \right| &\leq q_1 < 1 \\ \left| \frac{\partial \varphi_1}{\partial y} \right| + \left| \frac{\partial \varphi_2}{\partial y} \right| &\leq q_2 < 1 \end{aligned} \right\}, \quad (9.28)$$

то процесс последовательных приближений (9.27) сходится к решению $x = \xi, y = \eta$ системы, т.е.

$$\lim_{n \rightarrow \infty} x_n = \xi, \quad \lim_{n \rightarrow \infty} y_n = \eta.$$

Эта теорема остается верной, если условие (9.28) заменить условием

$$\left. \begin{aligned} \left| \frac{\partial \varphi_1}{\partial x} \right| + \left| \frac{\partial \varphi_1}{\partial y} \right| &\leq q_1 < 1 \\ \left| \frac{\partial \varphi_2}{\partial x} \right| + \left| \frac{\partial \varphi_2}{\partial y} \right| &\leq q_2 < 1 \end{aligned} \right\} \quad (9.29)$$

– условие сходимости итерационного процесса.

Оценка погрешности n -го приближения дается неравенством

$$|\xi - x_n| + |\eta - y_n| \leq \frac{M}{1 - M} (|x_n - x_{n-1}| + |y_n - y_{n-1}|),$$

где M – наибольшее из чисел q_1, q_2 , входящих в неравенства (9.28) или в неравенства (9.29). Сходимость метода итераций считается хорошей, если $M < \frac{1}{2}$, при этом $\frac{M}{1-M} < 1$, так что если в двух последовательных приближениях совпадают, скажем, первые три десятичных знака после запятой, то ошибка последнего приближения не превосходит 0,001.

Метод Ньютона для систем нелинейных уравнений n -го порядка. Метод Ньютона применяется к решению систем уравнений вида (9.21)

$$f_i(x_1, x_2, \dots, x_n) = 0, i = 1, \dots, n$$

или, в векторной форме, (9.22) $f(x) = 0$.

Введем матрицу Якоби $J(x)$ для функций $f_i(x) = 0, i = 1, \dots, n$, которые будем предполагать непрерывно дифференцируемыми

$$J(x) = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \dots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \dots & \frac{\partial f_2(x)}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \dots & \frac{\partial f_n(x)}{\partial x_n} \end{bmatrix}. \quad (9.30)$$

Пусть задано начальное приближение $x^{(0)}$. Вместо нелинейного уравнения (9.22) решаем линейное уравнение

$$f(x^{(0)}) + J(x^{(0)})(x - x^{(0)}) = 0. \quad (9.31)$$

Если $\det J(x^{(0)}) \neq 0$, то уравнение (10.11) имеет единственное решение, которое обозначим $x^{(1)}$. Вначале удобно решить уравнение (9.31) относительно $\Delta x^{(0)} = x - x^{(0)}$, а затем вычислить $x^{(1)} = x^{(0)} + \Delta x^{(0)}$. Если найдено $x^{(k)}$, то $x^{(k+1)}$ вычисляем по формуле $x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$, а поправку $\Delta x^{(k)} = (\Delta x_1^{(k)}, \dots, \Delta x_n^{(k)})$ находим из системы

$$f(x^{(k)}) + J(x^{(k)})\Delta x^{(k)} = 0, \quad (9.31^*)$$

где $J(x^{(k)})$ – матрица неособенная, т.е. $\det J(x^{(k)}) \neq 0$, то поправка выражается следующим образом $\Delta x^{(k)} = -f(x^{(k)}) \cdot J^{-1}(x^{(k)})$, где $J^{-1}(x^{(k)})$ – матрица, обратная матрице Якоби.

Т.о., последовательные приближения находятся по формуле

$$x^{(k+1)} = x^{(k)} - f(x^{(k)}) \cdot J^{-1}(x^{(k)}), k = 0, 1, \dots,$$

т.е. для вычисления последовательных приближений на каждом шаге необходимо вычислять обратную матрицу, в которой вместо $x^{(k)}$ берется предыдущее приближение.

Если искать решение в координатной форме (система (9.31*) примет вид)

$$\left. \begin{aligned} \frac{\partial f_1(x^{(k)})}{\partial x_1} \Delta x_1^{(k)} + \frac{\partial f_1(x^{(k)})}{\partial x_n} \Delta x_n^{(k)} &= -f_1(x^{(k)}) \\ &\dots \dots \dots \\ \frac{\partial f_n(x^{(k)})}{\partial x_1} \Delta x_1^{(k)} + \frac{\partial f_n(x^{(k)})}{\partial x_n} \Delta x_n^{(k)} &= -f_n(x^{(k)}) \end{aligned} \right\}, \quad (9.32)$$

то на каждом шаге для отыскания поправок $\Delta x^{(k)} = (\Delta x_1^{(k)}, \dots, \Delta x_n^{(k)})$ к найденному приближению $x^{(k)}$ приходится решать систему линейных алгебраических уравнений. Решив (9.32) любым известным методом (например, методом Гаусса) находим новое приближение $x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$.

Если нулевое приближение выбрано удачно, то метод Ньютона сходится, причем очень быстро (обычно за 3-5 итераций). Поэтому на практике этот метод используют чаще всего.

Критерием окончания итераций является условие $\|x^{(k+1)} - x^{(k)}\| \leq \varepsilon$.

Метод Ньютона для систем нелинейных уравнений второго порядка. Пусть дана система

$$\left. \begin{aligned} F(x, y) &= 0 \\ G(x, y) &= 0 \end{aligned} \right\}. \quad (9.33)$$

Согласно методу Ньютона, последовательные приближения вычисляются по формулам

$$x_{n+1} = x_n - \frac{A_n}{J_n}, \quad y_{n+1} = y_n - \frac{B_n}{J_n},$$

где

$$\begin{aligned} A_n &= \begin{vmatrix} F(x_n, y_n) & F'_y(x_n, y_n) \\ G(x_n, y_n) & G'_y(x_n, y_n) \end{vmatrix}, \\ B_n &= \begin{vmatrix} F'_x(x_n, y_n) & F(x_n, y_n) \\ G'_x(x_n, y_n) & G(x_n, y_n) \end{vmatrix}, \\ J_n &= \begin{vmatrix} F'_x(x_n, y_n) & F'_y(x_n, y_n) \\ G'_x(x_n, y_n) & G'_y(x_n, y_n) \end{vmatrix} \neq 0. \end{aligned}$$

Метод Ньютона сходится, если начальное приближение выбрано удачно (при достаточной близости к решению системы) и матрица $J(x^*)$ невырожденная. На практике итерации обычно оканчивают, если

$\|x^{(n+1)} - x^{(n)}\| \leq \varepsilon$. Для выбора начального приближения применяют, чаще всего, графический метод.

ВМИП ЧМ Лекции ПРОИ

Лекция 10. Квадратурные формулы численного интегрирования (4ч. – 2ч. ЛК+2ч. ЛР)

1. Постановка задачи численного интегрирования.
2. Квадратурные формулы Ньютона-Котеса.
3. Формулы прямоугольников, трапеций, Симпсона и их оценки погрешности.
4. Правило Рунге оценки погрешностей численного интегрирования.
5. Квадратурные формулы наивысшей алгебраической степени точности.

Постановка задачи численного интегрирования. Пусть $[a, b]$ – любой отрезок числовой оси, и рассматривается интеграл $\int_a^b F(x)dx$. Ставится задача: найти его приближённое значение по n значениям $F(x_i)$ функции F в точках x_i ($i = 1, 2, \dots, n$). Формулы для вычисления интегралов называют **квадратурными**.

Многие правила приближённых квадратур основаны на замене интегрируемой функции F на всём отрезке $[a, b]$ или на его частях на более простую функцию φ , близкую к F , легко интегрируемую точно и принимающую в узлах x_i те же значения $F(x_i)$, что и F . В качестве такой функции берут алгебраический или тригонометрический многочлен, либо дробно-рациональную функцию.

Когда отрезок интегрирования конечный и интегрируемая функция F имеет высокую гладкость, то можно рассчитывать хорошо приблизить её многочленом невысокой степени. Если же сама функция F или её производные невысоких порядков имеют особенности или даже обращаются в ∞ , то это затруднит приближение F или сделает его вообще невозможным. В этом случае мы должны будем заранее освободиться от таких особенностей путём их выделения. Делается это при помощи разложения F на два сомножителя $F(x) = p(x)f(x)$, где $p(x)$ имеет такие же особенности, как и $F(x)$, а $f(x)$ – есть достаточно гладкая функция, и интеграл рассматривается в форме $\int_a^b p(x)f(x)dx$.

Такое же представление применяется и при вычислении несобственных интегралов вида $\int_a^\infty F(x)dx$. Причём, F целесообразно разложить на множители $F(x) = p(x)f(x)$, из которых первый $p(x)$ характеризует закон убывания F при $x \rightarrow \infty$, а $f(x)$ – является гладкой функцией, допускающей хорошее приближение алгебраическими многочленами или рациональными функциями.

Функция $p(x)$ называется **весовой функцией** или **весом**. При построении определённого квадратурного правила она считается фиксированной и $p(x) > 0$.

Квадратурная сумма и связанные с ней задачи. Будем строить приближённые формулы вычислений вида

$$\int_a^b p(x)f(x)dx \approx \sum_{k=1}^n A_k f(x_k), \quad x_k \in [a, b]. \quad (10.1)$$

Числа A_k называются **квадратурными коэффициентами**, x_k – **квадратурными узлами**, а правая часть формулы – **квадратурной суммой**.

Формула (10.1) содержит $2n+1$ параметров: n, A_k, x_k ($k=1,2,\dots,n$). Их следует выбирать так, чтобы (10.1) давала, возможно, лучший результат при интегрировании избранного класса функций f . Роль n – очевидна: чем больше n , тем больше слагаемых в квадратурной сумме, и тем большей точности можно достичь путём выбора A_k и x_k . Поэтому при построении формулы число n считают фиксированным и рассматривают задачу о выборе A_k и x_k . В различных квадратурных методах одно из множеств: либо множество коэффициентов A_k , либо множество узлов x_k также может быть зафиксированным. Правом выбора их обычно пользуются для следующих целей.

а) Увеличение степени точности

Говорят, что квадратурная формула (10.1) имеет степень точности m , если она является точной для функций $g_i(x)$ ($i=1,2,\dots,m$), т.е.

$$\int_a^b p(x)g_i(x)dx = \sum_{k=1}^n A_k g_i(x_k)$$

и не является точной для $g_{m+1}(x)$.

Можно стремиться к тому, чтобы при помощи выбора параметров A_k и x_k сделать степень точности формулы (10.1) наивысшей возможной. Такие формулы впервые были рассмотрены Гауссом и их часто называют **формулами наивысшей степени точности**.

б) Минимизация погрешности

Остаточный член формулы (15.1) имеет вид

$$R_n(f) = \int_a^b p(x)f(x)dx - \sum_{k=1}^n A_k f(x_k). \quad (10.2)$$

За величину, характеризующую точность формулы на множестве F функций f может быть принята

$$\sup_f |R_n(f)| = M(A_1, \dots, A_n; x_1, \dots, x_n).$$

Путём выбора узлов x_k и коэффициентов A_k можно добиться, чтобы величина M имела бы наименьшее значение.

с) Упрощение вычислений

Можно при помощи выбора параметров A_k и x_k стремиться сделать, возможно, более простыми вычисления по формуле (10.1), например, взять равноотстоящими узлы, или взять равные коэффициенты.

Общая квадратурная формула. Предположим, что узлы x_k ($k=1,2,\dots,n$) выбраны и являются равноотстоящими. При построении квадратурной формулы (10.1) мы можем выбирать лишь коэффициенты A_k . Выполним интерполирование $f(x)$ по её значениям в узлах x_k при помощи интерполяционного многочлена Лагранжа степени $n-1$, т. е.

$$f(x) = L_n(x) + r(x),$$

где $L_n(x) = \sum_{k=1}^n \frac{\omega(x)}{(x-x_k)\omega'(x_k)} f(x_k)$, $\omega(x) = (x-x_1)\dots(x-x_n)$, $r(x)$ – погрешность интерполирования.

Тогда

$$\int_a^b p(x)f(x)dx = \int_a^b p(x)L_n(x)dx + \int_a^b p(x)r(x)dx. \quad (10.3)$$

Вычислим первый интеграл в правой части

$$\begin{aligned} \int_a^b p(x)f(x)dx &\approx \int_a^b p(x)L_n(x)dx = \int_a^b p(x) \sum_{k=1}^n \frac{\omega(x)}{(x-x_k)\omega'(x_k)} f(x_k)dx = \\ &= \sum_{k=1}^n f(x_k) \int_a^b \frac{p(x)\omega(x)}{(x-x_k)\omega'(x_k)} dx = \sum_{k=1}^n A_k f(x_k). \end{aligned}$$

Таким образом, интерполяционная квадратурная формула принимает вид

$$\int_a^b p(x)f(x)dx \approx \sum_{k=1}^n A_k f(x_k), \quad A_k = \int_a^b p(x) \frac{\omega(x)}{(x-x_k)\omega'(x_k)} dx. \quad (10.4)$$

Ее погрешность (10.2) определяется следующим образом

$$R_n(f) = \int_a^b p(x)f(x)dx - \sum_{k=1}^n A_k f(x_k) = \int_a^b p(x)r(x)dx. \quad (10.5)$$

Теорема о точности квадратурной формулы. Интерполяционная формула (10.4) характеризуется следующей теоремой о степени точности.

Теорема 10.1. Для того чтобы квадратурная формула (10.4) была точной для алгебраических многочленов степени не больше n , необходимо и достаточно, чтобы она была интерполяционной.

Доказательство. Необходимость. Пусть (10.4) является точным для любых многочленов степени не выше n . Возьмём функцию $f(x) = \frac{\omega(x)}{(x-x_k)\omega'(x_k)} = \omega_k(x)$. Это есть многочлен степени $n-1$ (т.е. не выше n), и если формула (10.4) верна для любых многочленов степени $n-1$, то она должна быть точной и для $\omega_k(x)$. Поэтому верны равенства

$$\int_a^b p(x)\omega_k(x)dx = \int_a^b p(x) \frac{\omega(x)}{(x-x_k)\omega'(x_k)} dx = \sum_{i=1}^n A_i \omega_k(x_i) = A_k,$$

так как, $\omega_k(x_i) = 0$, $i \neq k$, $\omega_k(x_k) = 1$, то формула (15.4) действительно является интерполяционной, так как её коэффициенты имеют значения, указанные в (10.4).

Достаточность. Пусть f – любой многочлен степени не выше n и квадратурное правило интерполяционное. Убедимся в том, что для него

равенство (10.4) будет выполняться точно. Интерполируем f по значениям в узлах x_k ($k=1,2,\dots,n$). Интерполирование будет точным

$$f(x) = \sum_{k=1}^n \frac{\omega(x)}{(x-x_k)\omega'(x_k)} f(x_k).$$

Кроме того, если A_k имеют значения, указанные в (10.4), то верны равенства

$$\int_a^b p(x)f(x)dx = \sum_{k=1}^n f(x_k) \int_a^b \frac{p(x)\omega(x)}{(x-x_k)\omega'(x_k)} dx = \sum_{k=1}^n A_k f(x_k),$$

и, следовательно, равенство (10.4) для $f(x)$ выполняется точно. Теорема доказана.

Если квадратурное правило является точным для всех многочленов степени n , то говорят, что оно имеет точность n . Из теоремы следует, что всякое квадратурное правило, степень которого не меньше n , является интерполяционным.

Остановимся на оценке погрешности. Если функция $f(x)$ имеет непрерывную производную порядка n на $[a,b]$, то на отрезке, содержащем точки $x_1, x_2, \dots, x_n, x$, существует точка ξ такая, что

$$r(x) = \frac{\omega(x)}{n!} f^{(n)}(\xi).$$

Тогда для (10.4) $R_n(f)$ имеет место следующее выражение

$$R_n(f) = \frac{1}{n!} \int_a^b p(x)\omega(x)f^{(n)}(\xi)dx.$$

Если же $|f^{(n)}(x)| < M_n$, $x \in [a,b]$, то имеем оценку погрешности квадратуры

$$|R_n(f)| \leq \frac{1}{n!} M_n \int_a^b |p(x)\omega(x)| dx, \quad (10.6)$$

где $M_n = \max_{\xi \in [a,b]} |f^{(n)}(\xi)|$.

Формулы Ньютона-Котеса. Формулы для приближенного вычисления интегралов, которые получаются путём замены подынтегрального выражения интерполяционным многочленом Лагранжа с узлами, разбивающими промежуток интегрирования на равные части, называются формулами Ньютона-Котеса. Формулы Ньютона-Котеса различаются степенями использованных интерполяционных многочленов. Чтобы не иметь дело с многочленами высоких степеней, обычно разбивают промежуток интегрирования на отдельные участки, применяют формулы Ньютона-Котеса с невысокими степенями на каждом участке и потом складывают полученные результаты (что дает так называемые составные формулы).

Пусть узлы x_k — равноотстоящие. Построим интерполяционные квадратурные формулы, считая, что на отрезке $[a,b]$ задано $n+1$ равноотстоящих узлов x_k , таких, что $x_0 = a$, $x_k = x_{k-1} + h$, $x_n = b$ и $h = \frac{b-a}{n}$. Введем новую переменную $q = \frac{x-x_0}{h}$. Тогда многочлен Лагранжа примет вид

$$L_n(x) = \sum_{k=0}^n \frac{(-1)^{n-k} q(q-1)\dots(q-n)}{k!(n-k)!(q-k)} f(x_k).$$

Подставим это выражение в (10.3) $(\int_a^b p(x)f(x)dx = \int_a^b p(x)L_n(x)dx + \int_a^b p(x)r(x)dx)$, получим

$$\int_a^b p(x)f(x)dx = \sum_{k=0}^n \frac{(-1)^{n-k} h^n}{k!(n-k)!} \int_0^n p(a+qh) \frac{q(q-1)\dots(q-n)}{q-k} dq f(x_k) + R_n(f).$$

Пусть $A_k = (b-a)B_k^n$, тогда последняя формула переписывается в виде

$$\int_a^b p(x)f(x)dx \approx (b-a) \sum_{k=0}^n B_k^n f(x_k), \quad (10.7)$$

где

$$B_k^n = \frac{(-1)^{n-k}}{nk!(n-k)!} \int_0^n p(a+qh) \frac{q(q-1)\dots(q-n)}{q-k} dq. \quad (10.8)$$

Квадратурные правила (10.7) называют **формулами Ньютона-Котеса**, а коэффициенты (10.8) – **коэффициентами Ньютона-Котеса**.

Для постоянной весовой функции $p(x) \equiv 1$ формула Ньютона-Котеса имеет вид

$$\int_a^b f(x)dx \approx (b-a) \sum_{k=0}^n B_k^n f(x_k), \quad (10.9)$$

а коэффициенты B_k^n определяются по формулам

$$B_k^n = \frac{(-1)^{n-k}}{nk!(n-k)!} \int_0^n \frac{q(q-1)\dots(q-n)}{q-k} dq. \quad (10.10)$$

Эти коэффициенты не зависят от промежутка интегрирования и могут быть вычислены раз и навсегда.

Если положить в (10.8) $f(x) = 1$, то получим

$$(b-a) = (b-a) \cdot \sum_{k=0}^n B_k^n \cdot 1,$$

и, следовательно, $\sum_{k=0}^n B_k^n = 1$.

Сравним B_k^n и B_{n-k}^n :

$$\frac{(-1)^{n-k}}{nk!(n-k)!} \int_0^n \frac{q(q-1)\dots(q-n)}{q-k} dq = \frac{(-1)^{n-(n-k)}}{n(n-k)!(n-(n-k))!} \int_0^n \frac{q(q-1)\dots(q-n)}{q-n+k} dq.$$

Введем замену $q-n = -t \Rightarrow q = -(t-n)$, $q-1 = -(t-n+1) \dots \Rightarrow q-n+k = -(t-k)$, $dq = -dt$,

$$B_{n-k}^n = \frac{(-1)^{n-k}}{nk!(n-k)!} \int_0^n \frac{t(t-1)\dots(t-n)}{t-k} dt = B_k^n.$$

Коэффициенты B_k^n вычислены до $n=20$. С ростом n коэффициенты B_k^n по модулю неограниченно возрастают. При больших n формулы Ньютона-Котеса становятся мало пригодными для вычислений, так как все значительнее сказывается неустраняемая (возникающая из-за неточности исходной информации, например, неточности измерений) и вычислительная погрешность (возникающая из-за округлений). Выпишем значения коэффициентов B_k^n для некоторых значений n :

$$\begin{aligned}
n = 1, \quad B_0^1 = B_1^1 &= \frac{1}{2}; \\
n = 2, \quad B_0^2 = B_2^2 &= \frac{1}{6}, B_1^2 = \frac{4}{6}; \\
n = 3, \quad B_0^3 = B_3^3 &= \frac{1}{8}, B_1^3 = B_2^3 = \frac{3}{8}.
\end{aligned}$$

Замечание 10.1. Формула (10.10) имеет степень точности n , если число узлов её $n+1$ — является чётным, и степень $n+1$, если число узлов её $n+1$ — является нечётным.

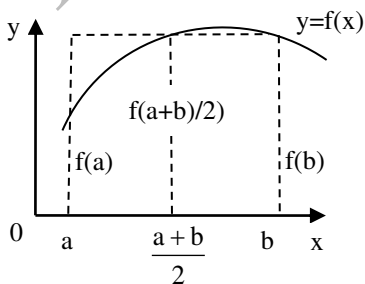
Простейшие формулы Ньютона-Котеса. Для повышения точности интегрирования отрезок $[a, b]$ часто делят на несколько частей, затем применяют избранную квадратурную формулу к каждой отдельной части и результаты складывают (в результате получают составные (обобщенные) формулы численного интегрирования). Этот метод является общим, и им можно пользоваться при применении всякой квадратурной формулы. Для многих формул интерполяционных квадратур погрешность $R_n(f)$ зависит от величин отрезка интегрирования следующим образом

$$R_n(f) = (b-a)^k C(a, b), \quad (10.11)$$

где $C(a, b)$ — медленно изменяющаяся функция от a до b и $k \in \mathbb{N}$. Такая зависимость показывает, что если мы уменьшим отрезок интегрирования в m раз, то $R_n(f)$ при этом уменьшится приблизительно в m^k раз. Для вычисления интеграла по всему отрезку $[a, b]$ разделим его на m равных частей и вычислим при помощи выбранной формулы интегралы по всем частичным отрезкам. В каждом случае погрешность будет приблизительно в m^k раз меньше, чем (10.11). При сложении всех таких интегралов получится результат, погрешность которого будет приблизительно в m^{k-1} раз меньше, чем погрешность (10.11), когда формула применяется для вычисления интеграла по всему отрезку $[a, b]$. Если $k > 1$, то произойдёт уменьшение погрешности тем больше, чем больше k . Описанный способ увеличения точности применим сейчас к простейшим формулам Ньютона-Котеса.

Формула прямоугольников (средних). В случае, когда $n=0$ функция $f(x)$ на отрезке $[a, b]$ заменяется интерполяционным многочленом нулевого порядка, построенным по значениям $f(x)$ в средней точке $x = \frac{a+b}{2}$, т. е. функцией $P_0(x) = f\left(\frac{a+b}{2}\right)$ на отрезке $[a, b]$. Таким образом,

$$\int_a^b f(x) dx = (b-a) f\left(\frac{a+b}{2}\right) + R(f). \quad (10.12)$$



(см. рисунок) Геометрически это означает, что площадь криволинейной трапеции, выражаемой $\int_a^b f(x) dx$, заменяется площадью **прямоугольника** с

основанием $b - a$ и высотой $f\left(\frac{a+b}{2}\right)$. Поэтому формулу (10.12) и называют формулой прямоугольника (средних прямоугольников).

Если $f(x)$ на отрезке $[a, b]$ имеет производную второго порядка, то для остаточного члена $R(f)$ можно записать простое выражение

$$R(f) = \frac{(b-a)^3}{24} f''(\eta), \text{ где } a \leq \eta \leq b.$$

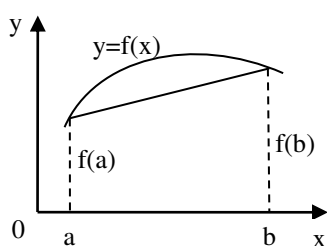
В целях повышения точности интегрирования разобьём отрезок $[a, b]$ на n равных частей длины $h = \frac{b-a}{n}$ и, применяя формулу (10.12) к каждому частичному отрезку. Просуммируем результаты по всем отрезкам, получим окончательную формулу прямоугольников

$$\int_a^b f(x) dx = \frac{b-a}{n} \left[f\left(a + \frac{h}{2}\right) + f\left(a + \frac{3h}{2}\right) + \dots + f\left(a + \frac{(2n-1)h}{2}\right) \right] + R, \quad (10.13)$$

где погрешность определяется формулой

$$R_1 = \frac{(b-a)^3}{24n^2} f''(\xi) = \frac{(b-a)h^2}{24} f''(\xi), \quad a \leq \xi \leq b. \quad (10.14)$$

Формулу (10.13) называют **обобщённой формулой прямоугольника**.



Формула трапеций. Пусть $n=1$, тогда линейное интерполирование выполняется по двум значениям $f(a)$ и $f(b)$, принимаемым функцией $f(x)$ на концах a и b , т. е. кривая $y = f(x)$ заменяется хордой, соединяющей конечные точки кривой (см. рисунок).

Интеграл от интерполяционного многочлена даст площадь трапеции ABCD. Поэтому и соответствующая формула численного интегрирования получила название **формулы трапеций**. Площадь трапеции ABCD, очевидно, равна

$$\frac{b-a}{2} [f(a) + f(b)].$$

Таким образом,

$$\int_a^b f(x) dx = \frac{b-a}{2} [f(a) + f(b)] + R(f). \quad (10.15)$$

Если $f''(x)$ — непрерывная функция на $[a, b]$, то для погрешности имеем

$$R(f) = -\frac{(b-a)^3}{12} f''(\eta), \quad \eta \in [a, b]. \quad (10.16)$$

Для увеличения точности формулы трапеций (10.15), разделим отрезок $[a, b]$ на n равных частей длины $h = \frac{b-a}{n}$. Рассмотрим частичный отрезок $[a+kh, a+(k+1)h]$. Для него получим

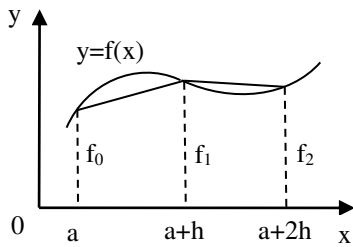
$$\int_{a+kh}^{a+(k+1)h} f(x) dx = \frac{h}{2} (f_k + f_{k+1}) + R_k, \quad f_k = f(a+kh),$$

и согласно (10.16)

$$R_k = -\frac{h^3}{12} f''(\eta_k), \quad a+kh \leq \eta_k \leq a+(k+1)h.$$

Сумма интегралов по всем частичным отрезкам даёт **обобщённую формулу трапеций**

$$\int_a^b f(x)dx \approx \frac{b-a}{n} \left[\frac{1}{2} f_0 + f_1 + f_2 + \dots + f_{n-1} + \frac{1}{2} f_n \right] + R, \quad (10.17)$$



где

$$R = R_0 + R_1 + \dots + R_{n-1} = -\frac{h^3}{12} [f''(\eta_0) + f''(\eta_1) + \dots + f''(\eta_{n-1})] =$$

$$= -\frac{(b-a)^3}{12n^2} \frac{f''(\eta_0) + f''(\eta_1) + \dots + f''(\eta_{n-1})}{n}, \quad a + kh \leq \eta_k \leq a + (k+1)h.$$

Величина $\frac{f''(\eta_0) + f''(\eta_1) + \dots + f''(\eta_{n-1})}{n}$ есть среднее

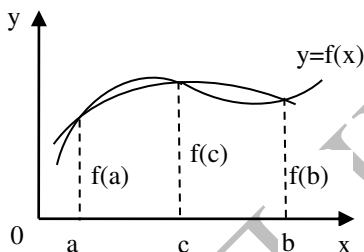
арифметическое значение, составленное из n значений второй производной f'' в n точках отрезка $[a, b]$. Считая ее непрерывной функцией на $[a, b]$, мы можем выбрать такую точку $\xi \in [a, b]$, что

$$\frac{f''(\eta_0) + f''(\eta_1) + \dots + f''(\eta_{n-1})}{n} = f''(\xi),$$

а значит,

$$R_1 = \frac{nh^3}{12} f''(\xi) = -\frac{(b-a)h^2}{12} \cdot f''(\xi), \quad a \leq \xi \leq b. \quad (10.18)$$

Формула Симпсона (парабол). Пусть $n=2$ и интерполируется функция $f(x)$ по трём точкам $a, \frac{a+b}{2}=c, b$ в которых известны её значения. Интерполяционный многочлен будет иметь вторую степень. Геометрически это означает, что мы проводим параболу через конечные и среднюю точки кривой (см. рисунок).



Квадратурное правило будет имеет вид

$$\int_a^b f(x)dx \approx \frac{b-a}{6} [f(a) + 4f(c) + f(b)]. \quad (10.19)$$

Эта формула называется также формулой **Симпсона**.

Погрешность формулы Симпсона имеет вид

$$R(f) = -\frac{1}{90} \left(\frac{b-a}{2} \right)^5 f^{(4)}(\eta), \quad \eta \in [a, b],$$

$f^{(4)}(x)$ – непрерывная функция на отрезке $[a, b]$.

Формула точна для многочленов до третьей степени включительно, так как в этом случае $f^{(4)}(x) \equiv 0$.

Формула Симпсона также может быть применена не сразу ко всему отрезку, а к отдельным частям его. Разделим $[a, b]$ на **чётное число** n ($n=2m$) равных частей длины $h = \frac{b-a}{n} = \frac{b-a}{2m}$ и возьмём сдвоенный частичный отрезок

$[a + (k-1)h, a + (k+1)h]$. Тогда, учитывая, что $f_k = f(a + kh)$, имеем

$$\int_{a+(k-1)h}^{a+(k+1)h} f(x)dx = \frac{h}{3} (f_{k-1} + 4f_k + f_{k+1}) + R_k.$$

Применив последнее равенство ко всем сдвоенным отрезкам $[a, a+2h]$, $[a+2h, a+4h]$, ..., и сложив почленно результаты, построим общее правило парабол или правило Симпсона

$$\int_a^b f(x)dx = \frac{b-a}{3n} [f_0 + f_n + 2(f_2 + f_4 + \dots + f_{n-2}) + 4(f_1 + f_3 + \dots + f_{n-1})] + R(f), \quad (10.20)$$

и

$$R(f) = -\frac{1}{90} h^5 [f^{(4)}(\eta_1) + f^{(4)}(\eta_3) + \dots + f^{(4)}(\eta_{n-1})].$$

Если функция $f^{(4)}(x)$ непрерывна на отрезке $[a, b]$, то существует такая точка ξ , что

$$\frac{2}{n} [f^{(4)}(\eta_1) + f^{(4)}(\eta_3) + \dots + f^{(4)}(\eta_{n-1})] = f^{(4)}(\xi)$$

и для погрешности $R(f)$ получим выражение

$$R_2 = -\frac{mh^5}{90} \cdot f^{(4)}(\xi) = -\frac{(b-a)^5}{180n^4} \cdot f^{(4)}(\xi) = -\frac{h^4(b-a)}{180} \cdot f^{(4)}(\xi), \quad a \leq \xi \leq b. \quad (10.21)$$

Замечание 10.2. Остаточный член формулы средних (10.14) примерно вдвое меньше, чем у формулы трапеций (10.18). Поэтому если значения функции одинаково легко определяются в любых точках, то лучше вести расчет по более точной формуле средних. Формулу трапеций применяют в тех случаях, когда функция задана только в узлах сетки, а в серединах интервалов неизвестна.

Замечание 10.3. Знаки главного члена погрешности у формул трапеций и средних разные (см. (10.14) и (10.18)). Поэтому, если есть расчеты по обеим формулам, то точное значение интеграла лежит, как правило, в вилке между ними. Деление этой вилки в отношении 2:1 дает уточненный результат, соответствующий формуле Симпсона.

Квадратурные формулы наивысшей алгебраической степени точности (НАСТ). Постановка задачи. На предыдущей лекции мы получили формулы численного интегрирования с равноотстоящими узлами путем замены подынтегральной функции алгебраическим интерполяционным многочленом с равноотстоящими узлами интерполирования.

Если мы не будем требовать равномерного распределения узлов на отрезке интегрирования, то можно получить формулы, точные для многочленов более высокой степени. Будем строить квадратурные формулы интерполяционного типа, имеющие при заданном числе узлов n наивысшую алгебраическую степень точности, т.е. будем находить узлы и коэффициенты квадратурных формул из условия, что их остаточные члены обращаются в нуль для всех многочленов максимально высокой степени.

Итак, имеем задачу: на отрезке $[a, b]$ необходимо выбрать узлы x_i , $i=1, 2, \dots, n$ и определить коэффициенты A_i таким образом, чтобы квадратурное правило

$$\int_a^b p(x)f(x)dx \approx \sum_{i=1}^n A_i f(x_i) \quad (10.22)$$

было точным для всех многочленов наивысшей возможной степени m . При построении квадратурной формулы будем считать, что весовая функция $p(x)$ удовлетворяет условиям

$$\int_a^b |p(x)x^i| dx < q, \quad i > 0, \quad \text{и} \quad \int_a^b |p(x)| dx > 0.$$

В правиле (10.22) содержится $2n$ неизвестных: $A_i, x_i, i=1,2,\dots,n$, поэтому следует ожидать, что при надлежащем выборе узлов и квадратурных коэффициентов оно будет точным для всех многочленов степени $m=2n-1$.

Теорема 10.2. Для того чтобы квадратурное правило (16.1) было точным для всех многочленов степени не выше $2n-1$, необходимо и достаточно выполнение условий:

1. Правило (10.22) должно быть интерполяционным, то есть, коэффициенты должны вычисляться по формулам:

$$A_k = \int_a^b p(x) \frac{\omega(x)}{(x-x_k)\omega'(x_k)} dx. \quad (10.23)$$

2. Многочлен $\omega(x) = (x-x_1)(x-x_2)\dots(x-x_n)$ должен быть ортогонален на отрезке $[a,b]$ по весу $p(x)$ ко всякому многочлену $Q(x)$ степени меньше n :

$$\int_a^b p(x)\omega(x)Q(x)dx = 0. \quad (10.24)$$

Итак, задача о построении квадратурного правила (16.1) сводится к задаче об отыскании многочлена $\omega(x)$, который обладает свойством 2 (равенство (10.24)). Покажем, что такой многочлен существует, все его корни действительны, различны и принадлежат отрезку $[a,b]$, о чем говорят следующие леммы.

Лемма 10.1. Если весовая функция $p(x)$ не меняет знак на отрезке $[a,b]$, то существует и притом единственный многочлен $\omega(x)$ ортогональный на $[a,b]$ по весу $p(x)$ ко всякому многочлену степени меньше n .

Доказательство. Будем искать многочлен в виде $\omega(x) = x^n + a_1x^{n-1} + \dots + a_{n-1}x + a_n$. Из условия ортогональности для определения коэффициентов a_i получим систему n уравнений

$$\int_a^b p(x)[x^n + a_1x^{n-1} + \dots + a_{n-1}x + a_n]x^i dx = 0, \quad i = 0, 1, \dots, n-1.$$

Достаточно показать, что соответствующая однородная система имеет лишь нулевое решение

$$\int_a^b p(x)[a_1x^{n-1} + \dots + a_{n-1}x + a_n]x^i dx = 0, \quad i = 0, 1, \dots, n-1.$$

Умножим последовательно каждое уравнение на $a_n, a_{n-1}, \dots, a_1$ и просуммируем. В результате получим уравнение

$$\int_a^b p(x)[a_1x^{n-1} + a_2x^{n-2} + \dots + a_{n-1}x + a_n]^2 dx = 0.$$

В силу того, что $p(x)$ не меняет знак на $[a,b]$, то последнее равенство возможно лишь тогда, когда все $a_i = 0$. Теорема доказана.

Лемма 10.2. Если $p(x)$ не меняет знак на отрезке $[a, b]$ и многочлен $\omega(x)$ ортогонален на $[a, b]$ ко всякому многочлену $Q(x)$ степени меньше n , то все корни $\omega(x)$ действительны, различны и лежат внутри $[a, b]$.

Доказательство. Пусть $\xi_1, \xi_2, \dots, \xi_m$ – корни, которые лежат внутри $[a, b]$ и имеет нечетную кратность. Допустим, что $m < n$. Построим многочлен $\rho(x) = (x - \xi_1)(x - \xi_2) \dots (x - \xi_m)$. Для него должно выполняться равенство

$$\int_a^b p(x)\omega(x)\rho(x)dx = 0.$$

С другой стороны, многочлен $\omega(x)\rho(x)$ содержит только четные степени и поэтому

$$\int_a^b p(x)\omega(x)\rho(x)dx \neq 0.$$

Значит, $m = n$. Теорема доказана.

Итак, таким образом, квадратурное правило наивысшей алгебраической степени точности (10.22) существует при любом n и является единственным. Можно показать, что ни при каком выборе узлов x_i и коэффициентов A_i , $i = 1, 2, \dots, n$ квадратурное правило не будет точным для всех многочленов степени $2n$. Если $p(x) \geq 0$ и квадратурное правило является точным для всех многочленов степени $2n-2$, то все квадратурные коэффициенты $A_i > 0$. Действительно, если $f(x) = \left[\frac{\omega(x)}{x - x_i} \right]^2$, то

$\int_a^b p(x)f(x)dx = A_i[\omega'(x_i)]^2$. Откуда получаем

$$A_i = \int_a^b p(x) \left[\frac{\omega(x)}{(x - x_i)\omega'(x_i)} \right]^2 dx > 0.$$

Квадратурная формула Гаусса. Квадратурные формулы наивысшей алгебраической степени точности для случая $p(x) \equiv 1$ впервые построил Гаусс. Поэтому и для любого $p(x)$ подобные формулы называют формулами Гаусса.

Построим квадратурное правило, т.е. найдем общий вид $\omega(x)$, когда $p(x) \equiv 1$. Введем обозначения

$$\int_a^x \omega(x)dx = \varphi_1(x),$$

$$\int_a^x \varphi_1(x)dx = \varphi_2(x), \dots, \int_a^x \varphi_{n-1}(x)dx = \varphi_n(x).$$

Будем вычислять интеграл $\int_a^b \omega(x)q(x)dx$, где $q(x)$ – многочлен степени не выше $n-1$. Применяя последовательно формулу интегрирования по частям, учитывая (10.24), получим

$$\begin{aligned} 0 &= \int_a^b \omega(x)q(x)dx = \int_a^b \omega(x)dx q(x) \Big|_a^b - \int_a^b \varphi_1(x)q(x)dx = \dots \\ &= \left[\varphi_1(x)q(x) - \varphi_2(x)q^{(1)}(x) + \varphi_3(x)q^{(2)}(x) - \dots + (-1)^{n-1} \varphi_n(x)q^{(n-1)}(x) \right] \Big|_a^b. \end{aligned}$$

При $x=a$ правая часть уравнения обращается в нуль, так как $\varphi_i(a)=0$, $i = 1, 2, \dots, n$. Поэтому в силу произвольности $q(x)$ справедливы равенства

$$\varphi_1(b) = \varphi_2(b) = \dots = \varphi_n(b) = 0.$$

Следовательно, $\varphi_n(x)$ имеет корни кратности n при $x=a$ и $x=b$ и может быть записана в виде

$$\varphi_n(x) = C(x-a)^n(x-b)^n,$$

где C – постоянная.

Отсюда для $\omega(x)$ получим

$$\omega(x) = C \cdot \frac{d^n}{dx^n} [(x-a)^n(x-b)^n].$$

Постоянная C подбирается из условия, что коэффициент при x^n в $\omega(x)$ равен 1. Чтобы определить величину этого коэффициента, вычислим

$$\frac{d^n}{dx^n} x^{2n} = 2n(2n-1)(2n-2)\dots(n+1)x^n = \frac{(2n)!}{n!} x^n.$$

$$\text{Следовательно, } C = \frac{n!}{(2n)!}.$$

Итак, окончательно получаем

$$\omega(x) = \frac{n!}{(2n)!} \cdot \frac{d^n}{dx^n} [(x-a)^n(x-b)^n]. \quad (10.25)$$

Последовательное применение теоремы Ролля показывает, что все корни уравнения $\omega(x)=0$, действительны, различны и принадлежат интервалу $[a, b]$. Таким образом, их действительно можно использовать в качестве узлов интерполяции и полученная при этом формула численного интегрирования будет удовлетворять поставленным условиям.

Вычисляя корни многочлена (10.25) и коэффициенты A_k по формуле (10.23), мы получим квадратурное правило наивысшей алгебраической степени точности, которое называется **квадратурной формулой Гаусса**.

Остаточный член формулы Гаусса. Исследуем остаточный член полученных формул численного интегрирования. Пусть $f(x)$ – произвольная, достаточное количество раз дифференцируемая функция. Построим интерполяционный многочлен Эрмита $H(x)$ степени $2n-1$, такой что $H(x_i) = f(x_i)$ и $H'(x_i) = f'(x_i)$, причём $x_1, x_2, \dots, x_n$ – корни уравнения $\omega(x) = 0$.

Тогда

$$f(x) = H(x) + \frac{\omega^2(x)}{(2n)!} f^{(2n)}(\eta).$$

Проинтегрируем с весом $p(x)$ левую и правую части последнего выражения

$$\begin{aligned} \int_a^b f(x)p(x)dx &= \int_a^b p(x)H(x)dx + \int_a^b p(x) \frac{\omega^2(x)}{(2n)!} f^{(2n)}(\eta)dx = \\ &= \sum_{i=1}^n A_i H(x_i) + \int_a^b p(x) \frac{\omega^2(x)}{(2n)!} f^{(2n)}(\eta)dx = \sum_{i=1}^n A_i f(x_i) + \int_a^b p(x) \frac{\omega^2(x)}{(2n)!} f^{(2n)}(\eta)dx = \end{aligned}$$

$$= \sum_{i=1}^n A_i f(x_i) + \frac{f^{(2n)}(\xi)}{(2n)!} \int_a^b p(x) \omega^2(x) dx.$$

Таким образом, остаточный член будет иметь вид

$$R(f) = \frac{f^{(2n)}(\xi)}{(2n)!} \int_a^b p(x) \omega^2(x) dx.$$

При $p(x) \equiv 1$ выражение для $R(f)$ можно упростить, интегрируя по частям. Воспользуемся ранее введенными функциями $\varphi_i(x)$

$$\begin{aligned} \int_a^b \omega(x) \omega(x) dx &= \varphi_1(x) \omega(x) \Big|_a^b - \int_a^b \varphi_1(x) \omega'(x) dx = \dots = (-1)^n \int_a^b \varphi_n(x) \omega^{(n)}(x) dx \\ &= (-1)^n n! \int_a^b \varphi_n(x) dx. \end{aligned}$$

Отсюда следует, что

$$\int_a^b \omega^2(x) dx = (-1)^n \frac{n!}{(2n)!} n! \int_a^b (x-a)^n (x-b)^n dx.$$

Вычислим интеграл по частям, получим

$$\begin{aligned} \int_a^b (x-a)^n (x-b)^n dx &= - \int_a^b \frac{(x-a)^{n+1} (x-b)^{n-1}}{(n+1)} n dx = \dots \\ &= (-1)^n \frac{(n!)^2 (b-a)^{2n+1}}{(2n)! (2n+1)} dx. \end{aligned}$$

Возвращаемся

$$\int_a^b \omega^2(x) dx = (-1)^{2n} \frac{(n!)^4 (b-a)^{2n+1}}{[(2n)!]^2 (2n+1)}.$$

Итак, при $p(x) \equiv 1$ погрешность $R(f)$ принимает вид

$$R(f) = \frac{(b-a)^{2n+1} (n!)^4}{[(2n)!]^3 (2n+1)} f^{(2n)}(\xi). \quad (10.26)$$

Коэффициенты формул Гаусса. Коэффициенты A_k при известных значениях x_k могут быть вычислены по формуле

$$A_k = \frac{(n!)^4 (b-a)^{2n+1}}{[(2n)!]^2 (x_k - a)(b - x_k) \omega'^2(x_k)}. \quad (10.27)$$

Все корни многочлена $\omega(x)$ расположены симметрично относительно средней точки $\frac{a+b}{2}$, так, что для всякой точки x_k найдется симметричная ей точка x_{n-k+1} и, следовательно, $A_k = A_{n-k+1}$, то есть коэффициенты при $f(x_k)$ и $f(x_{n-k+1})$ совпадают.

Корни x_k и коэффициенты A_k можно вычислить для фиксированного отрезка $[-1, 1]$. Любой отрезок $[a, b]$ путем замены $x = \frac{b+a}{2} + \frac{b-a}{2} t$ приводится к отрезку $[-1, 1]$ и коэффициенты можно вычислить раз и навсегда. Запишем квадратурное правило (10.22) для отрезка $[-1, 1]$

$$\int_{-1}^1 f(x)dx = 2 \sum_{k=1}^n A_k f(x_k) + \frac{2^{2n+1}(n!)^4}{[(2n)!]^3(2n+1)} f^{(2n)}(\xi). \quad (10.28)$$

Тогда для общего отрезка $[a, b]$ получим

$$\begin{aligned} \int_a^b f(x)dx &= \int_{-1}^1 f\left(\frac{a+b}{2} + \frac{b-a}{2}t\right) \frac{b-a}{2} dt = \\ &= (b-a) \sum_{k=1}^n A_k f\left(\frac{a+b}{2} + \frac{b-a}{2}t_k\right) + \left(\frac{b-a}{2}\right) R(f). \end{aligned} \quad (10.29)$$

Выпишем значение узлов x_k , коэффициентов A_k и погрешностей $R(f)$ для нескольких первых значений n

$$n=1, \quad x_1=0, \quad A_1=2, \quad R_1(f)=\frac{1}{3}f''(\xi);$$

$$n=2, \quad x_2=-x_1=0.5773502692, \quad A_1=A_2=1, \quad R_2(f)=\frac{1}{135}f^{(4)}(\xi);$$

$$n=3, \quad x_2=0, \quad x_3=-x_1=0.7745966692, \quad A_2=\frac{8}{9}, \quad A_1=A_3=\frac{5}{9},$$

$$R_2(f)=\frac{1}{15750}f^{(6)}(\xi);$$

$$\begin{aligned} n=4, \quad x_3=-x_2=0.3399810436, \quad x_4=-x_1=0.8611363116, \\ A_2=A_3=0.6521451549, \quad A_1=A_4=0.3478548451, \quad R_2(f)=\frac{1}{3472875}f^{(8)}(\xi). \end{aligned}$$

Из приведенных значений видно, что с ростом n знаменатель в выражении для погрешности растет очень быстро. Поэтому для $n \geq 4$ квадратурное правило Гаусса имеет высокую точность. Отметим, что $\sum_{i=1}^n A_i = 1$, $A_i > 0$ для любого i и, следовательно, с ростом n коэффициенты A_i остаются ограниченными величинами. На практике формулы можно применять для небольших n в виде обобщенных формул: $h = \frac{b-a}{m}$. Полагаем в (10.29)

$$\begin{aligned} \int_{a+ih}^{a+(i+1)h} f(x)dx &= \int_{-1}^1 f\left(a + (i+0.5)h + \frac{h}{2}t\right) \cdot \frac{h}{2} dt \\ &= h \sum_{k=1}^n A_k f\left(a + (i+0.5)h + \frac{h}{2}t_k\right) + \frac{h^{2n+1}(n!)^4}{[(2n)!]^3(2n+1)} f^{(2n)}(\xi). \end{aligned}$$

Для всего отрезка получим формулу

$$\begin{aligned} \int_a^b f(x)dx &= h \sum_{k=1}^n A_k \sum_{i=0}^{m-1} f\left(a + (i+0.5)h + \frac{h}{2}t_k\right) \\ &\quad + \frac{h^{2n}(b-a)(n!)^4}{[(2n)!]^3(2n+1)} f^{(2n)}(\bar{\eta}). \end{aligned}$$

При $n=2$ составная формула имеет простой вид

$$\int_a^b f(x)dx = \frac{h}{2} \sum_{i=0}^{m-1} [f(a + h(i + 0.5(1 + t_1))) + f(a + h(i + 0.5(1 - t_1)))] \\ + \frac{h^4(b-a)}{135} f^{(4)}(\bar{\eta}),$$

где $t_1 = 0.577350$.

ВМИП ЧМ ЛЕКЦИИ ПРОИ

Лекция 11. Приближенное вычисление кратных интегралов (4ч. – 2ч. ЛК+2ч. ЛР)

1. Понятие о кубатурных формулах.
2. Метод повторного применения квадратурных формул.
3. Метод замены подынтегральной функции интерполяционным многочленом.
4. Кубатурная формула трапеций на прямоугольной сетке. Кубатурная формула средних на прямоугольной сетке. Кубатурная формула Симпсона. Практическое применение.

Кубатурные формулы предназначены для вычисления двойных интегралов. Пусть необходимо вычислить двойной интеграл по области D с гладкой границей $J = \iint_D f(x, y) dx dy$. Такие интегралы не всегда могут быть вычислены с помощью элементарных функций. Существующие приближенные методы вычисления – кубатурные формулы, основанные на том, что область D покрывается сеткой, в узлах сетки вычисляются значения функции $f(x_i, y_j)$ ($i = \overline{0, n}, j = \overline{0, m}$) и производится их суммирование по определенным правилам. При этом возникают следующие трудности: 1) выбор сетки; 2) аппроксимация границы области, если область не прямоугольная; 3) оценка точности вычислений.

Метод повторного применения квадратурных формул. Пусть необходимо приближенно вычислить кратный интеграл

$$I = \iiint_G \dots \int f(x_1, x_2, \dots, x_n) dx_1 dx_2 \dots dx_n = \int_G f(x_1, x_2, \dots, x_n) dx_1 dx_2 \dots dx_n ,$$

где G – область n -мерного пространства.

Рассмотрим несколько способов построения формул численного интегрирования вида

$$\int_G f(x_1, x_2, \dots, x_n) dx_1 dx_2 \dots dx_n = \sum_{i=1}^N c_i(p_i) + R(f) . \quad (11.1)$$

Формулы (11.1) называются **кубатурными формулами**, а точки $p_i \in G$ называются **узлами кубатурной формулы**, c_i ($i = \overline{0, n}$) – коэффициенты кубатурной формулы, $R(f)$ – остаточный член.

Для простоты изложения допустим, что область G – прямоугольник $G = \{a \leq x \leq b, c \leq y \leq d\}$ и нужно вычислить интеграл

$$J = \int_a^b \int_c^d f(x, y) dx dy .$$

Представим этот интеграл в виде $I = \int_a^b dx \int_c^d f(x, y) dy = \int_a^b F(x) dx,$

где $F(x) = \int_c^d f(x, y) dy$. Для вычисления $J = \int_a^b F(x) dx$ воспользуемся квадратурной формулой Симпсона

$$J = \int_a^b F(x) dx = \frac{b-a}{6} \left[F(a) + 4F\left(\frac{a+b}{2}\right) + F(b) \right] + R_1(F(x)), \quad (11.2)$$

где

$$\begin{aligned} R_1(F(x)) &= -\frac{(b-a)^5}{16 \cdot 180} F^{(4)}(\xi) = -\frac{(b-a)^5}{16 \cdot 180} \int_c^d \frac{\partial^4 f(\xi, y)}{\partial x^4} dy = \\ &= -\frac{(b-a)^5 (d-c)}{16 \cdot 180} \frac{\partial^4 f(\xi, \eta)}{\partial x^4}, \quad (a < \xi < b, \quad c < \eta < d). \end{aligned}$$

В свою очередь $F(x)$ можно представить в виде

$$F(x) = \int_c^d f(x, y) dy = \frac{d-c}{6} \left[f(x, c) + 4f\left(x, \frac{c+d}{2}\right) + f(x, d) \right] + R_y(f(x, y)),$$

где $R_y(f(x, y)) = -\frac{(d-c)^5}{16 \cdot 180} \frac{\partial^4 f(x, \xi)}{\partial y^4}$.

Подставляя выражение для $F(x)$ в (11.2), получим

$$\begin{aligned} \int_a^b F(x) dx &= \int_a^b \int_c^d f(x, y) dx dy = \frac{(b-a)(d-c)}{36} \left[f(a, c) + 4f\left(a, \frac{c+d}{2}\right) + f(a, d) + 4f\left(\frac{a+b}{2}, c\right) + \right. \\ &\quad \left. + 16f\left(\frac{a+b}{2}, \frac{c+d}{2}\right) + 4f\left(\frac{a+b}{2}, d\right) + f(b, c) + 4f\left(b, \frac{c+d}{2}\right) + f(b, d) \right] - \\ &\quad - \frac{(b-a)(d-c)}{2^5 \cdot 90} \cdot \left[(b-a)^4 \frac{\partial^4 f(\xi, \eta)}{\partial x^4} + (d-c)^4 \frac{\partial^4 f(\xi_1, \eta_1)}{\partial y^4} + \right. \\ &\quad \left. + \frac{(b-a)^4 (d-c)^4}{2^5 \cdot 90} \frac{\partial^8 f(\xi_2, \eta_2)}{\partial x^4 \partial y^4} \right], \quad (a < \xi_i < b, \quad c < \eta_i < d). \end{aligned} \quad (11.3)$$

Формула (11.3) называется **кубатурной формулой Симпсона**. Эта формула содержит значения функции $f(x, y)$ в 9 точках. Из вида остаточного члена следует, что она дает точное значение интеграла, если $f(x, y)$ является произвольным многочленом степени не выше трех.

Таким способом можно получить и другие кубатурные формулы, если для вычисления интегралов $\int_a^b F(x) dx$ и $F(x) = \int_c^d f(x, y) dy$ применить те или иные квадратурные формулы.

Аналогично можно вычислить и интеграл более высокой кратности, причем по разным переменным можно использовать различные квадратурные формулы.

Метод замены подынтегральной функции интерполяционным многочленом. Другой путь получения кубатурных формул состоит в замене подынтегральной функции некоторым интерполяционным многочленом.

Заменим в интеграле $J = \iint_G f(x, y) dx dy$ функцию $f(x, y)$ интерполяционным многочленом вида

$$L(x, y) = \sum_{i=1}^N f(x_i, y_i) L_i(x, y),$$

где многочлены $L_i(x, y)$ не зависят от $f(x, y)$ функции и обладают свойством

$$L_i(x_j, y_j) = \begin{cases} 1 & \text{при } i = j \\ 0 & \text{при } i \neq j. \end{cases}$$

После замены подынтегральной функции получим

$$\begin{aligned} J &= \iint_G f(x, y) dx dy = \iint_G L(x, y) dx dy + \iint_G R(f(x, y)) dx dy = \\ &= \sum_{i=1}^N c_i f(x_i, y_i) + \bar{R}(f(x, y)), \text{ где } c_i = \iint_G L_i(x, y) dx dy. \end{aligned}$$

Вычисление коэффициентов c_i для простых областей G не вызывает затруднений.

Если область G – прямоугольник, т.е. $G = \{a \leq x \leq b, c \leq y \leq d\}$. Рассмотрим сетку узлов, образуемых в пересечении прямых $x_i = a + ih, y_j = c + jl, h = \frac{b-a}{n}, l = \frac{d-c}{m} (i = \overline{0, n}, j = \overline{0, m})$.

Тогда имеет место следующая интерполяционная формула

$$f(x, y) = L(x, y) = \sum_{i=0}^n \sum_{j=0}^m f(x_i, y_j) \frac{\omega_n(x) \omega_m(y)}{(x-x_i) \omega'_n(x_i) \omega'_m(y_j)} + R(x, y), \quad (11.4)$$

где $\omega_n(x) = (x-x_0)(x-x_1) \dots (x-x_n), \omega_m(y) = (y-y_0)(y-y_1) \dots (y-y_m)$.

Проводя интегрирование по формуле прямоугольника, получим

$$\iint_{a, c}^{b, d} f(x, y) dx dy = \sum_{i=0}^n \sum_{j=0}^m c_{ij} f(x_i, y_j) + R(f), \quad (11.5)$$

где

$$c_{ij} = \int_a^b \frac{\omega_n(x)}{(x-x_i) \omega'_n(x_i)} dx \cdot \int_c^d \frac{\omega_m(y)}{(y-y_j) \omega'_m(y_j)} dy,$$

$$R(f) = \iint_{a, c}^{b, d} R(x, y) dx dy.$$

В нашем случае

$$c_{ij} = (b-a)(d-c) B_i^n B_j^m,$$

где B_i^n, B_j^m – коэффициенты Ньютона-Котеса (10.8).

В интерполяционной формуле (11.4) узлы интерполирования можно брать и неравноотстоящие по x и y , но так, чтобы они были точками пересечения прямых $x = x_i, y = y_j$. Выбирая их разными способами, будем получать различные кубатурные формулы.

Практическое применение. На практике для вычисления кратных интегралов удобнее использовать формулы, дающие высокую точность при минимальном числе узлов (так как такие формулы требуют меньшего объема вычислительной работы), например, **формулы Гаусса**.

Возьмем в качестве x_0, x_1 и y_0, y_1 в области G четыре узла квадратурной формулы Гаусса с двумя ординатами соответственно на отрезках $[a, b]$ и $[c, d]$, т.е.

$$x_0 = \frac{a+b}{2} - \frac{b-a}{2\sqrt{3}}, x_1 = \frac{a+b}{2} + \frac{b-a}{2\sqrt{3}}, y_0 = \frac{c+d}{2} - \frac{d-c}{2\sqrt{3}}, y_1 = \frac{c+d}{2} + \frac{d-c}{2\sqrt{3}}. \quad (11.6)$$

Тогда интегрированием (11.4) при $n = m = 1$ и x_0, x_1, y_0, y_1 вида (11.6) получим кубатурную формулу

$$\int_a^b \int_c^d f(x, y) dx dy = \frac{(b-a)(d-c)}{4} [f(x_0, y_0) + f(x_0, y_1) + f(x_1, y_0) + f(x_1, y_1)] + R(f). \quad (11.7)$$

Погрешность этой формулы определится соотношением

$$R(f(x, y)) = \frac{(b-a)(d-c)}{24^3 \cdot 5} \left[(b-a)^4 \frac{\partial^4 f(\xi, \eta)}{\partial x^4} + (d-c)^4 \frac{\partial^4 f(\xi_1, \eta_1)}{\partial y^4} - \frac{(b-a)^4 (d-c)^4}{24^3 \cdot 5} \frac{\partial^8 f(\xi_2, \eta_2)}{\partial x^4 \partial y^4} \right], ((\xi_i, \eta_i) \in G). \quad (11.8)$$

Таким образом, остаток кубатурной формулы Гаусса, построенной по четырем точкам, может оказаться меньше остатка кубатурной формулы Симпсона, построенной для девяти точек.

Примеры решения задач. Вычислить приближенно двойной интеграл

$$\int_0^1 \int_0^1 \frac{x^2}{1+x^2} \cdot \frac{y^2}{1+y^2} dx dy.$$

1. Использовать кубатурную формулу Симпсона с шагом $h_x = h_y = \frac{b-a}{2}$.

Решение

Кубатурная формула Симпсона для $n = 2$ имеет вид (11.3)

$$\int_a^b \int_c^d f(x, y) dx dy = \frac{(b-a)(d-c)}{36} [f(a, c) + f(b, c) + 4f\left(a, \frac{c+d}{2}\right) + f\left(\frac{a+b}{2}, c\right) + f\left(\frac{a+b}{2}, d\right) + f\left(b, \frac{c+d}{2}\right)] + 16f\left(\frac{a+b}{2}, \frac{c+d}{2}\right) + f(a, d) + f(b, d)] - R_2(f).$$

Найдем $h_x = h_y = \frac{1-0}{2} = 0.5$.

Найдем значения функции $f(x, y) = \frac{x^2}{1+x^2} \cdot \frac{y^2}{1+y^2}$ в точках, необходимых для построения последней формулы

		<i>c</i>		<i>d</i>	
		<i>f</i>	0	0.5	1
<i>a</i>	0	0	0	0	0
	0.5	0	0.04	0.1	
<i>b</i>	1	0	0.1	0.25	

Итак,

$$\int_0^1 \int_0^1 \frac{x^2}{1+x^2} \cdot \frac{y^2}{1+y^2} dx dy = \frac{1}{36} (4 \cdot (0,1 + 0,1) + 16 \cdot 0,04 + 0,25) =$$

$$= \frac{1}{36} (0,8 + 0,64 + 0,25) = \frac{1,69}{36} = 0,0469.$$

2. Использовать кубатурную формулу Гаусса для $n = 2$.

Решение

Кубатурная формула Гаусса для $n = 2$ имеет вид (11.7) с узлами (11.6).

Вычислим узлы сетки (11.6)

$$x_0 = \frac{1}{2} - \frac{1}{2\sqrt{3}} \approx 0.5 - 0.2887 \approx 0.2113;$$

$$x_1 = \frac{1}{2} + \frac{1}{2\sqrt{3}} \approx 0.5 + 0.2887 \approx 0.7887;$$

$$y_0 = \frac{1}{2} - \frac{1}{2\sqrt{3}} \approx 0.2113;$$

$$y_1 = \frac{1}{2} + \frac{1}{2\sqrt{3}} \approx 0.7887.$$

Найдем значения функции $f(x, y) = \frac{x^2}{1+x^2} \cdot \frac{y^2}{1+y^2}$ в узлах, необходимых для построения последней формулы (11.7)

		x_0	x_1
f		0.2113	0.7887
y_0	0.2113	0.00174	0.0164
y_1	0.7887	0.0164	0.1471

Таким образом,

$$\int_0^1 \int_0^1 \frac{x^2}{1+x^2} \cdot \frac{y^2}{1+y^2} dx dy = \frac{1}{4} (0.00174 + 2 \cdot 0.0164 + 0.1471) = \frac{0.1816}{4} = 0.04539.$$

Лекция 12. Решение задачи Коши для уравнения первого порядка (6ч. – 4ч. ЛК+2ч. ЛР)

1. Постановка задачи Коши и ее геометрический смысл. Дискретизация задачи.
2. Основные характеристики численных методов решения задачи Коши: явность/неявность, многшаговость. Аппроксимация, устойчивость и сходимости численных методов.
3. Понятие о локальной и глобальной погрешностях.
4. Методы Эйлера. Модификации метода Эйлера 2-го порядка точности.
5. Идея построения методов Рунге-Кутты. Порядок точности методов. Правило Рунге оценки погрешностей решения задачи Коши. Организация вычислений с автоматическим выбором шага.

Постановка задачи Коши и ее геометрический смысл. Дискретизация задачи. Простейшим дифференциальным уравнением является уравнение первого порядка

$$y' = f(x, y). \quad (12.1)$$

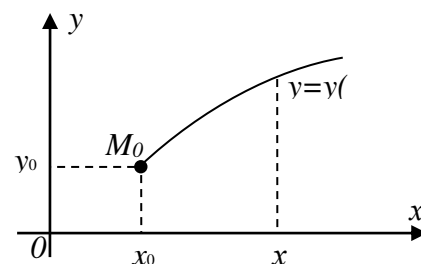
Основная задача, относящаяся к этому уравнению, есть **задача Коши**: найти решение уравнения (12.1)

$$y = y(x), \quad (12.2)$$

удовлетворяющее *начальному условию*

$$y(x_0) = y_0, \quad (12.3)$$

т.е. найти интегральную кривую $y = y(x)$, проходящую через заданную точку $M_0(x_0, y_0)$.



Для дифференциального уравнения n -го порядка

$$y^{(n)} = f(x, y, y', \dots, y^{(n-1)})$$

задача Коши состоит в нахождении решения $y = y(x)$, удовлетворяющего *начальным условиям*

$$y(x_0) = y_0, \quad y'(x_0) = y'_0, \quad \dots, \quad y^{(n-1)}(x_0) = y_0^{(n-1)},$$

где $x_0, y_0, y'_0, \dots, y_0^{(n-1)}$ – заданные числа.

Рассмотрим *нормальную систему* n -го порядка обыкновенных дифференциальных уравнений:

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, \dots, y_n) \\ \dots \dots \dots \\ \frac{dy_n}{dx} = f_n(x, y_1, \dots, y_n) \end{cases}, \quad (12.4)$$

где x – независимая переменная, $y_1, \dots, y_n$ – искомые функции.

Воспользовавшись векторными обозначениями

$$y = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}, \quad \frac{dy}{dx} = \begin{pmatrix} \frac{dy_1}{dx} \\ \dots \\ \frac{dy_n}{dx} \end{pmatrix}, \quad f = \begin{pmatrix} f_1 \\ \dots \\ f_n \end{pmatrix},$$

Систему (12.4) запишем в виде

$$\frac{dy}{dx} = f(x, y). \quad (12.5)$$

Под *решением системы* (12.5) понимается любая совокупность функций $y = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}$, где $y_1 = \varphi_1(x), \dots, y_n = \varphi_n(x)$, которая, будучи подставлена в

систему (12.5), обращает все уравнения этой системы в тождество. Так как система дифференциальных уравнений имеет бесконечное множество решений, то для выделения одного конкретного решения $y = y(x)$, кроме уравнения, нужны дополнительные условия. В простейшем случае задаются начальные условия

$$y(x_0) = y^{(0)}, \quad (12.6)$$

т.е. получаем *задачу Коши*: найти решение $y = y(x)$ системы дифференциальных уравнений (12.4) или (12.5), удовлетворяющее заданным начальным условиям (12.6), где x_0 – фиксированное значение независимой

переменной, $y^{(0)} = \begin{pmatrix} y_1^{(0)} \\ \dots \\ y_n^{(0)} \end{pmatrix}$ – данная система чисел. Геометрически это значит,

что требуется отыскать интегральную кривую L , проходящую через заданную точку $M_0(x_0, y_1^{(0)}, \dots, y_n^{(0)})$ пространства $R^{n+1} = \{x, y_1, \dots, y_n\}$.

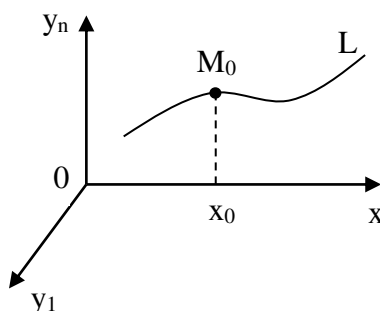


Рисунок 12.1 – Геометрическая интерпретация задачи Коши для системы дифференциальных уравнений

О методах решения дифференциальных уравнений. В курсах по дифференциальным уравнениям изучаются методы, позволяющие выразить решение дифференциального уравнения через элементарные функции, либо представить его при помощи квадратур от элементарных функций. Эти

методы называют **точными**. Классы уравнений, для которых применимы точные методы, сравнительно узки и охватывают только малую часть возникающих на практике задач. Указанное обстоятельство привело к созданию большого числа методов приближенного решения дифференциальных уравнений. **Приближенные методы** в зависимости от формы, в которой они представляют решение, можно разделить на три группы:

- 1) **аналитические** методы, дающие приближенное решение дифференциального уравнения в виде аналитического выражения;
- 2) **графические** методы, дающие приближенное решение в виде графика;
- 3) **численные** методы, дающие приближенное решение в виде таблицы.

Численные методы не позволяют найти общее решение уравнения (12.1) или системы уравнений (12.5). С их помощью можно найти лишь решение начальных или краевых задач. Численные методы применимы к задачам, имеющим единственное решение (корректно поставленным задачам). В некоторых случаях условий корректности может оказаться недостаточно. Надо чтобы задача была хорошо обусловлена (устойчива), т.е. чтобы малые изменения в задании исходных данных приводили к достаточно малым изменениям искомого решения.

Метод Эйлера. Рассмотрим задачу Коши

$$y' = f(x, y), \quad a \leq x \leq b \quad (12.7)$$

$$y(a) = y_0. \quad (12.8)$$

На отрезке $[a, b]$ выберем конечное множество точек x_i ($i = 0, 1, \dots, N$), причем будем считать, что $a = x_0 < x_1 < \dots < x_N = b$. Искомую интегральную кривую $y = y(x)$, проходящую через точку $M_0(x_0, y_0)$ приближенно заменим

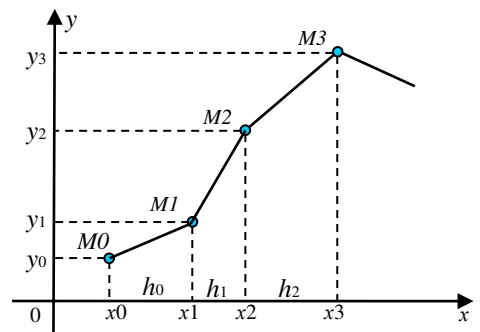
ломаной $M_0M_1M_2\dots$ с вершинами $M_i(x_i, y_i)$, звенья которой M_iM_{i+1} прямолинейны между прямыми $x = x_i$ и $x = x_{i+1}$ и имеют подъем $\frac{y_{i+1} - y_i}{h_i} = f(x_i, y_i)$. Таким образом, звенья M_iM_{i+1} ломаной Эйлера в каждой вершине M_i имеют направление $y'_i = f(x_i, y_i)$, совпадающее с направлением интегральной кривой уравнения (12.7), проходящей через точки M_i .

В методе Эйлера (метод ломаных) приближенное значение $y_i = y(x_i)$ вычисляется по формуле

$$y_{i+1} = y_i + h_i f(x_i, y_i), \quad i = 0, 1, 2, \dots, N-1, \quad (12.9)$$

где $h_i = x_{i+1} - x_i$. Если x_i – равноотстоящие точки, то $h_i = h$.

Для оценки точности полученного приближенного значения на практике пользуются двойным пересчетом: расчет на отрезке $[x_i, x_{i+1}]$



повторяют с шагом $h_i/2$ и погрешность более точного решения y_i^* (при шаге $h_i/2$) оценивают по формуле

$$|y_i^* - y(x_i)| \approx |y_i^* - y_i|.$$

Метод Эйлера легко распространяется и на системы дифференциальных уравнений. Приближенное решение задачи Коши для системы

$$\begin{cases} y' = f_1(x, y, z) \\ z' = f_2(x, y, z) \end{cases}, \quad \begin{matrix} y(x_0) = y_0 \\ z(x_0) = z_0 \end{matrix}, \quad a \leq x \leq b$$

Вычисляется последовательно по следующим формулам

$$\begin{aligned} y_{i+1} &= y_i + h_i f_1(x_i, y_i, z_i) \\ z_{i+1} &= z_i + h_i f_2(x_i, y_i, z_i), \quad i = 0, 1, 2, \dots, N-1. \end{aligned}$$

Метод Эйлера является простейшим численным методом интегрирования дифференциальных уравнений. К его недостаткам относится малая точность и систематическое накопление ошибок.

Модификации метода Эйлера. Рассмотрим дифференциальное уравнение (12.7) с начальным условием (12.8). На отрезке $[a, b]$ зададим конечное множество точек $\{x_i\} (i = 0, 1, \dots, N)$, $a = x_0 < x_1 < \dots < x_N = b$, $h_i = x_{i+1} - x_i (i = 0, 1, \dots, N-1)$.

Согласно методу Эйлера, будем иметь

$$y_{i+1} = y_i + h_i f_i, \quad f_i = f(x_i, y_i) \quad (12.10)$$

Более точным, в отличие от метода Эйлера, является **усовершенствованный метод ломаных**, при котором сначала вычисляют промежуточные значения

$$x_{i+\frac{1}{2}} = x_i + \frac{h_i}{2}, \quad y_{i+\frac{1}{2}} = y_i + \frac{h_i}{2} f_i.$$

и находят значение поля интегральных кривых в средней точке $(x_{i+\frac{1}{2}}, y_{i+\frac{1}{2}})$, то есть $f_{i+\frac{1}{2}} = f(x_{i+\frac{1}{2}}, y_{i+\frac{1}{2}})$, а затем полагают

$$y_{i+1} = y_i + h_i f_{i+\frac{1}{2}}. \quad (12.11)$$

Другой модификацией метода Эйлера является **усовершенствованный метод Эйлера-Коши**, при котором сначала определяется «грубое приближение» решения $\tilde{y}_{i+1} = y_i + h_i f_i$, исходя из которого, находится направление поля интегральных кривых $\tilde{f}_{i+1} = f(x_{i+1}, \tilde{y}_{i+1})$. Затем приближенно полагают

$$y_{i+1} = y_i + h_i \frac{f_i + \tilde{f}_{i+1}}{2}. \quad (12.12)$$

Остаточные члены первого (12.11) и второго (12.12) улучшенных методов Эйлера на каждом шаге имеют порядок $O(h^3)$. Оценка погрешности в точке x_i может быть получена с помощью двойного пересчета: расчет повторяют с шагом $h_i/2$ и погрешность более точного решения y_i^* (при шаге $h_i/2$) оценивают приближенно по формуле

$$|y_i^* - y(x_i)| \approx \frac{1}{3} |y_i^* - y_i|,$$

где $y(x)$ – точное решение дифференциального уравнения.

Усовершенствованный метод Эйлера-Коши можно еще более уточнить, применяя **итерационную обработку** каждого значения y_i . А именно, исходя из грубого приближения

$$y_{i+1}^{(0)} = y_i + h_i f(x_i, y_i)$$

строим итерационный процесс

$$y_{i+1}^{(k)} = y_i + \frac{h_i}{2} (f(x_i, y_i) + f(x_{i+1}, y_i^{(k-1)})) \quad (k = 1, 2, \dots).$$

Итерации продолжают до тех пор, пока в пределах требуемой точности два последовательных приближения $y_{i+1}^{(k-1)}$ и $y_{i+1}^{(k)}$ не совпадут. После чего $y_{i+1}^{(k)}$ принимают за приближенное значение $y(x_{i+1})$. Если же алгоритм уточнения после трех-четырех итераций не приводит к совпадению требуемого числа десятичных знаков, то следует уменьшить шаг вычислений h_i .

Метод Эйлера и его модификации являются простейшими представителями конечно-разностных методов (шаговых методов) и являются одношаговыми.

Метод Рунге-Кутты. Рассмотрим задачу Коши для обыкновенного дифференциального уравнения (12.7) с начальным условием (12.8). Выбирая шаг $h = \frac{b-a}{N}$, на отрезке $[a, b]$ введем сетку $x_i = x_0 + ih$, $y_i = y(x_i)$, $i = 0, 1, 2, \dots$. Вычислим числа

$$K_1^{(i)} = hf(x_i, y_i), \quad K_2^{(i)} = hf\left(x_i + \frac{h}{2}, y_i + \frac{K_1^{(i)}}{2}\right),$$

$$K_3^{(i)} = hf\left(x_i + \frac{h}{2}, y_i + \frac{K_2^{(i)}}{2}\right), \quad K_4^{(i)} = hf(x_i + h, y_i + K_3^{(i)}).$$

Согласно обычному методу Рунге-Кутты последовательные приближенные значения $y_i = y(x_i)$ искомой функции y определяются по формуле

$$y_{i+1} = y_i + \Delta y_i, \quad (12.13)$$

где $\Delta y_i = \frac{1}{6} (K_1^{(i)} + 2K_2^{(i)} + 2K_3^{(i)} + K_4^{(i)})$, $i = 0, 1, 2, \dots$

Шаг расчета можно менять при переходе от одной точки к другой. Для контроля правильности выбора шага h рекомендуется вычислять дробь

$$\theta = \frac{|K_2^{(i)} - K_3^{(i)}|}{|K_1^{(i)} - K_2^{(i)}|}.$$

Величина θ не должна превышать нескольких сотых. В противном случае шаг h следует уменьшить.

Погрешность этого метода на каждом шаге есть величина порядка h^5 , если $f(x, y) \in C^{(5)}$, а на всем отрезке $[x_0, X]$ порядок точности равен h^4 .

Оценка погрешности метода очень затруднительна. Грубую оценку можно получить с помощью двойного просчета по формуле:

$$|y_i^* - y(x_i)| \approx \frac{1}{15} |y_i^* - y_i|,$$

где $y(x_i)$ – значение точного решения (12.7) в точке x_i , а y_i^* и y_i – приближенные значения, полученные с шагом $\frac{h}{2}$ и h .

Метод Рунге-Кутты обладает значительной точностью и, несмотря на свою трудоемкость, широко используется при численном решении дифференциальных уравнений. Кроме того, важным преимуществом этого метода является возможность применения «переменного шага».

Заметим, что для начала вычислений по методу Рунге-Кутты не нужно строить начальный отрезок.

При вычислении приближенного решения задачи (12.7) – (12.8) по формуле (12.13) удобно пользоваться схемой, приведенной в следующей таблице:

i	x	y	$K = hf(x, y)$	Δy
0	x_0	y_0	$K_1^{(0)}$	$K_1^{(0)}$
	$x_0 + \frac{h}{2}$	$y_0 + \frac{K_1^{(0)}}{2}$	$K_2^{(0)}$	$2K_2^{(0)}$
	$x_0 + \frac{h}{2}$	$y_0 + \frac{K_2^{(0)}}{2}$	$K_3^{(0)}$	$2K_3^{(0)}$
	$x_0 + h$	$y_0 + K_3^{(0)}$	$K_4^{(0)}$	$K_4^{(0)}$
				$\frac{1}{6} \sum = \Delta y_0$
1	x_1	y_1	...	...
	...	...	...	...

Метод Рунге-Кутты применим также для приближенного решения систем обыкновенных дифференциальных уравнений. В этом случае в приведенных выше формулах величины $y, f, y_0, y_i, \Delta y_i, K_1^{(0)}, K_2^{(0)}, K_3^{(0)}, K_4^{(0)}, y_1$ являются векторами, а x, x_0, x_i, h – скалярными величинами.

Лекция 13. Решение задачи Коши для систем дифференциальных уравнений (2ч. УСР)

1. Решение задачи Коши для систем дифференциальных уравнений первого порядка.

2. Решение задачи Коши для систем дифференциальных уравнений m -го порядка.

Материал для изучения находится в файле «Лекции УСР.docx».

ВМИП ЧМ Лекции ПРОИ