1. Приближенные числа. Абсолютная и относительная погрешности. Значащие и верные цифры числа.

К **точным** относятся числа, которые дают истинное значение исследуемой величины.

К **приближенным** относятся числа, близкие к истинному значению, причем степень близости определяется погрешностью вычислений. Это связано с тем, что при измерениях появление приближенных чисел связано с несовершенством измерительных приборов. Кроме того, часто нет необходимости знать точное значение интересующей величины.

Пусть x - точное значение числа, а X - его приближение. Тогда X есть приближенное значение числа x **с недостатком**, если X < x , и приближенное значение **с избытком**, если x < X . Число x будет для самого себя приближением одновременно с недостатком и с избытком.

Если x - точное, неизвестное значение некоторой величины, а X - его приближение, то разность x - X называется ошибкой, или погрешностью приближения. Часто знак ошибки x - X неизвестен, поэтому используется так называемая абсолютная погрешность приближенного числа x, определяемая равенством:

$$\Delta(\widehat{x}) = \left|x - \widehat{x}\right|, \text{ откуда имеем } x = \widehat{x} \pm \Delta(\widehat{x})$$

Изучаемая числовая величина есть величина именованная, т.е. измеряется, например, в сантиметрах, килограммах и т.п. Погрешность имеет ту же размерность. Однако часто возникает необходимость заменить эту погрешность безразмерным числом - **относительной погрешностью**.

$$\delta(\widehat{x}) = \frac{\Delta(\widehat{x})}{\left|\widehat{x}\right|} = \left|\frac{x-\widehat{x}}{\widehat{x}}\right|,$$

относительную погрешность часто выражают в процентах: $\delta(\widehat{x}) = \frac{\Delta(\widehat{x})}{\left|\widehat{x}\right|} \cdot 100\%$

Первая слева отличная от нуля цифра числа и все расположенные справа от нее, называются **значащими**.

Цифра числа, записанного в десятичной системе, называется **верной**, если если абсолютная погрешность числа x не превосходит одной единицы соответствующего разряда десятичного числа. Цифры, не являющиеся верными, называются **сомнительными**. Если верная цифра - значащая, то она называется **верной значащей** цифрой.

Вид записи приближенного числа должен показывать его абсолютную погрешность, которая не должна превосходить единицы последнего разряда, сохраняемого при записи. Другими словами, приближенные числа принято записывать таким образом, чтобы все цифры числа, кроме нулей впереди, если они есть, были значащими и верными цифрами.

2. Погрешности арифметических операций - погрешности суммы, разности, произведения, частного приближенных чисел. Погрешность вычисления функции одной и многих переменных.

Пусть x и y – точные числа, $\widehat{x}$ и $\widehat{y}$ – их приближения, $\Delta(\widehat{x})$ и $\Delta(\widehat{y})$ – их абсолютные, а $\delta(\widehat{x})$ и $\delta(\widehat{y})$ – их относительные погрешности соответственно.

Погрешность суммы и разности.

$\Delta(\widehat{x}+\widehat{y}) \leq \Delta(\widehat{x}) + \Delta(\widehat{y})$, т.е. абсолютная погрешность суммы не превосходит суммы их абсолютных погрешностей

$\Delta(\widehat{x}-\widehat{y}) \leq \Delta(\widehat{x}) + \Delta(\widehat{y})$, аналогично для разности

$\Delta(\widehat{x_1}+\widehat{x_2}+\ \ldots+\widehat{x_n}) \leq \Delta(\widehat{x_1}) + \Delta(\widehat{x_2}) + \ \ldots + \Delta(\widehat{x_n})$

Из полученных результатов, вытекают следующие правила сложения:

1) выделяются числа, имеющие наибольшую абсолютную погрешность
2) более точные числа округляются таким образом, чтобы сохранить в них на один знак больше, чем в выделенном числе
3) производится сложение или вычитание всех чисел с учетом сохраненных знаков
4) полученный результат округляется на один знак

Относительная погрешность суммы и разности определяются соотношениями:

1) $\delta(\widehat{x}+\widehat{y}) = \frac{\Delta(\widehat{x}+\widehat{y})}{|\widehat{x}+\widehat{y}|} \leq \frac{\Delta(\widehat{x})+\Delta(\widehat{y})}{|\widehat{x}+\widehat{y}|}$
2) $\delta(\widehat{x}-\widehat{y}) = \frac{\Delta(\widehat{x}-\widehat{y})}{|\widehat{x}-\widehat{y}|} \leq \frac{\Delta(\widehat{x})+\Delta(\widehat{y})}{|\widehat{x}-\widehat{y}|}$

Погрешность произведения.

$\Delta(\widehat{x}\cdot\widehat{y}) \leq |\widehat{x}|\Delta(\widehat{y}) + |\widehat{y}|\Delta(\widehat{x}) + \Delta(\widehat{x})\Delta(\widehat{y})$, из чего можно получить правила умножения:

1) выделяется число с наименьшим количеством верных значащих цифр
2) оставшиеся сомножители округляются таким образом, чтобы они содержали на одну значащую цифру больше, чем количество верных значащих цифр в выделенном числе
3) в произведении сохраняется столько значащих цифр, сколько верных значащих цифр имеет выделенное число

$\delta(\widehat{x}\cdot\widehat{y}) = \frac{\Delta(\widehat{x}\cdot\widehat{y})}{|\widehat{x}\cdot\widehat{y}|} = \delta(\widehat{x}) + \delta(\widehat{y}) + \delta(\widehat{x})\cdot\delta(\widehat{y})$

Погрешность частного.

$\Delta(\frac{\widehat{x}}{\widehat{y}}) \leq \frac{|\widehat{x}|\cdot\Delta(\widehat{y})+|\widehat{y}|\Delta(\widehat{x})}{\widehat{y}^2|1-\delta(\widehat{y})|}$, из чего можно получить правила деления:

1) выделяется число с наименьшим количеством верных значащих цифр

2) оставшиеся числа округляются таким образом, чтобы они содержали на одну значащую цифру больше, чем выделенное число
3) в полученном результате деления сохраняется столько значащих цифр, сколько верных значащих цифр имеет выделенное число.

$$\delta(\frac{\hat{x}}{\hat{y}}) = \frac{\Delta(\frac{\hat{x}}{\hat{y}})}{\left|\frac{\hat{x}}{\hat{y}}\right|} \leq \frac{\delta(\hat{x})+\delta(\hat{y})}{\left|1-\delta(\hat{y})\right|}$$

На практике при работе с числами достаточно хорошей точности обычно считают, что $\Delta(\hat{x}) \cdot \Delta(\hat{y}) \approx 0$, $\delta(\hat{x}) \cdot \delta(\hat{y}) \approx 0$, $1 - \delta(\hat{y}) \approx 1$. Поэтому вместо вышеперечисленных формул, пользуются следующими:

$$\Delta(\hat{x} \cdot \hat{y}) \leq \left|\hat{x}\right|\Delta(\hat{y}) + \left|\hat{y}\right|\Delta(\hat{x}), \quad \delta(\hat{x} \cdot \hat{y}) \leq \delta(\hat{x}) + \delta(\hat{y})$$

$$\Delta(\frac{\hat{x}}{\hat{y}}) \leq \frac{\left|\hat{x}\right|\cdot\Delta(\hat{y})+\left|\hat{y}\right|\Delta(\hat{x})}{\hat{y}^2}, \quad \delta(\frac{\hat{x}}{\hat{y}}) \leq \delta(\hat{x}) + \delta(\hat{y})$$

3. Представление чисел в ЭВМ. Понятия машинного эпсилон, машинной бесконечности, машинного нуля.

Целые числа, представимые в ЭВМ (целые машинные числа) - это все целые числа, принадлежащие замкнутому интервалу [-N, N]. В языке Паскаль, например - это числа диапазона [-32768, 32767]. Арифметические операции с целыми машинными числами выполняются на ЭВМ точно, если результат принадлежит интервалу. В противном случае фиксируется ошибочная ситуация, а результат операции не определен или не имеет смысла.

Вещественные числа могут быть записаны в двух формах. Обычную форму записи числа в виде: $x = \pm\, a_n a_{n-1} \ldots a_1 a_0, a_{-1} a_{-2} \ldots a_{-m}$, называют записью с фиксированной запятой. В позиционной системе счисления с основанием вэта запись означает, что $x = \pm\, a_n \cdot \beta^n + a_{n-1} \cdot \beta^{n-1} + \ldots + a_1 \cdot \beta + a_0 + a_{-1} \cdot \beta^{-1} + a_{-2} \cdot \beta^{-2} + \ldots + a_{-m} \cdot \beta^{-m}$

О числах, записанных в виде $0,63750 \cdot 10^6; 637,50 \cdot 10^3; 6375,0 \cdot 10$, говорят, что они записаны в форме с плавающей запятой. Запись числа с плавающей запятой, как следует из примера, не является однозначной. Для устранения этой неоднозначности принято первый множитель брать меньше единицы, и он должен состоять только из значащих цифр. Такая форма записи числа называется нормализованной – $x = m \cdot \beta^p$, здесь p - целое число, называемое порядком числа x, m - число с фиксированной запятой, называемое мантиссой числа x. Первая после запятой цифра числа m всегда отлична от нуля.

Для записи вещественных чисел в ЭВМ отводится фиксированное число разрядов (разрядная сетка), в которой выделены разряды для записи мантиссы, порядка, знаков мантиссы и порядка. Таким образом, множество машинных вещественных чисел характеризуется следующими четырьмя целыми константами: основанием счисления β, β >2,

точностью t, нижней границей экспоненты e^- и верхней границей экспоненты e^+. Ненулевые машинные вещественные числа имеют вид

$$x = \pm\left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_t}{\beta^t}\right)\beta^p = \pm d_1 d_2 \ldots d_t \cdot \beta^p$$

где $0 \le d_i < \beta, \quad d_1 \ne 0, \quad e^- \le p \le e^+$.

Минимальное положительное число e которое может быть представлено в ЭВМ, называется машинным нулем: $\varepsilon_0 = \beta^{e^- - 1}$. Максимальное положительное — машинной бесконечностью ($\varepsilon_\infty = \beta^{e^+}(1 - \beta^{-t})$), т.е. машинные вещественные числа $\in [-\varepsilon_\infty, \varepsilon_\infty]$. Число $\varepsilon_1 = \beta^{-t+1}$ называют **машинным эпсилоном**. Оно зависит от разрядности мантиссы и характеризует точность представления чисел.

4. Вычислительные задачи. Корректность и обусловленность вычислительных задач.

Обусловленность и корректность постановки задачи.
Пусть в результате решения задачи по исходному значению величины x находится значение искомой величины y. Если исходная величина имеет абсолютную погрешность $\Delta(x)$, то решение y имеет абсолютную погрешность $\Delta(y)$. Задача называется обусловленной по исходному параметру x, если решение y непрерывно зависит от x. Другими словами, малые погрешности в исходном величине приводят к малым погрешностям в результате расчётов.

Задача называется поставленной **корректно**, если для любых значений исходных данных её решение существует, единственно и обусловлено по исходным данным. Отсутствие обусловленности означает, что даже незначительные погрешности в исходных данных приводят к большим погрешностям в решении или вовсе к неверному результату.

5. Прямая и обратная задачи теории погрешностей.

Для оценки влияния погрешностей округления на результат вычислений по некоторому алгоритму (прямая задача теории погрешностей), предполагают, что этот результат является действительной функцией от входных параметров: $y = f(x_1, x_2, \ldots, x_n)$. Поскольку входные данные $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n$ не точные, то $\hat{y} = f(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ - приближенное значение результата.

Если $y = f(x_1, x_2, \ldots, x_n)$ − непрерывно дифференцируемая функция своих аргументов, то по формуле конечных приращений Лагранжа имеем следующую оценку погрешности:

$$\Delta y = y - \hat{y} = \sum_{k=1}^{n} \frac{df(\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_n)}{dx} \cdot \Delta x_k \le \sum_{k=1}^{n} m_k \cdot \Delta x_k$$

Здесь точка $(\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_n)$ расположена между $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ и $(x_1, x_2, \ldots, x_n)$.

$$m_k = max \frac{df(x_1, x_2, \ldots, x_n)}{dx_k}$$

Оценка показывает, что накопление погрешностей округления зависит от величины входных параметров и от характера алгоритма.

Коэффициенты $K_i = \frac{x_i}{f(x_1,x_2,\ldots,x_n)} \cdot \frac{df(0_1,0_2,\ldots,0_n)}{dx_i}$ являются коэффициентами усиления относительных погрешностей входных данных. Если значения всех коэффициентов усиления ограничены по абсолютной величине единицей, то накопления погрешностей не происходит. Коэффициенты K_i часто называют числами обусловленности задачи.

При анализе погрешностей возникают две задачи:

1) прямая – зная точность исходных данных оценить точность полученного результата
2) обратная - определить точность, с которой необходимо задавать исходную информацию, чтобы обеспечить требуемую точность решения

Для случая дифференцируемой функции одной переменной грубое решение обратной задачи очевидно: если $y = f(x)$, то $|\Delta y| = |f'(x\ *)| \cdot \Delta(x)$ откуда $|\Delta x| \approx \frac{\Delta y}{|f'(x^*)|}$.

Для функций большего числа переменных обратная задача некорректна. Для ее решения необходимы дополнительные условия.

6. Прямые методы решения систем линейных алгебраических уравнений. Метод Гаусса: основная идея и схемы реализации (схема единственного деления и схемы с выбором главных элементов).

Прямые методы используют конечные соотношения и формулы для вычисления неизвестных. К прямым (или точным) методам решения СЛАУ относятся алгоритмы, которые в предположении, что вычисления ведутся без округлений, позволяют получить точное решение системы за конечное число арифметических действий. В прямых методах число необходимых для решения арифметических операций зависит только от вида вычислительной схемы и от порядка матрицы n.

Большинство прямых методов основано на идее последовательного эквивалентного преобразования заданной системы с целью исключения неизвестных из части уравнений. В результате исходная система преобразуется в эквивалентную ей систему с матрицей более простой формы.

В основе методов лежит следующее утверждение: пусть все главные миноры матрицы A отличны от нуля, тогда существуют такие нижняя L и верхняя U треугольные матрицы, что матрица A представима в виде произведения этих матриц: A = LU. Если все элементы диагонали одной из такой треугольных матриц фиксированы (ненулевые), то такое разложение единственно.

Метод Гаусса (схема единственного деления). Основная идея метода состоит в том, что система приводится к эквивалентной ей системе с верхней и треугольной матрицей и единичной диагональю (прямой ход исключения). Из полученной системы последовательно, начиная с последнего уравнения находятся неизвестные (обратный ход исключения).

Метод Гаусса (с выбором главного элемента). Метод заключается в том, что при прямом ходе в алгоритме метода Гаусса на каждом шаге исключения производится выбор наибольшего по модулю элемента в качестве ведущего. Этого достигают перестановкой строк и столбцов матрицы коэффициентов. Наиболее распространенной в вычислительной практике является стратегия выборов главного элемента столбца - нахождение максимального по модулю элемента k-ого столбца матрицы и использование его в качестве ведущего элемента на k-ом шаге исключения. В этом случае для невырожденных систем гарантируется, что ведущие элементы не равны нулю и уменьшается погрешность при делении и последующем вычитании при преобразованиях.

7. Метод прогонки решения систем линейных алгебраических уравнений с трехдиагональной матрицей.

Этот метод является модификацией метода Гаусса для систем специального вида с трехдиагональной матрицей A, т.е. с матрицей, все элементы которой, не лежащие на главной и двух побочных диагоналях, равны нулю.

$$\begin{bmatrix} q_1 & r_1 & 0 & \dots & 0 & 0 & 0 \\ p_2 & q_2 & r_2 & \dots & 0 & 0 & 0 \\ 0 & p_3 & q_3 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & p_{n-1} & q_{n-1} & r_{n-1} \\ 0 & 0 & 0 & \dots & 0 & p_n & q_n \end{bmatrix}.$$

Запишем эту систему в виде равенств:

$$p_i x_{i-1} + q_i x_i + r_i x_{i+1} = b_i,\ i = \overline{1,n},\ p_1 = 0,\ r_n = 0.$$

Находя коэффициенты по рекуррентным формулам:

$$u_1 = -\frac{r_1}{q_1},\ v_1 = \frac{b_1}{q_1},\ u_i = -\frac{r_i}{q_i + p_i u_{i-1}},\ v_i = -\frac{b_i - p_i v_{i-1}}{q_i + p_i u_{i-1}},\ i = \overline{2,n}$$

Обратный ход дает окончательное решение системы по формулам:

$$x_n = v_n,\ x_i = u_i x_{i+1} + v_i,\ i = \overline{n-1,1}$$

8. Обусловленность задачи решения систем линейных алгебраических уравнений. Нормы векторов и матриц. Число обусловленности.

Система уравнений считается **плохо обусловленной**, если малые изменения в коэффициентах матрицы или в правой части вызывают большие изменения в решении. Система уравнений считается **хорошо обусловленной**, если малые изменения в коэффициентах матрицы или в правой части вызывают малые изменения в решении.

Число обусловленности **Cond(A)** оценивает близость матрицы коэффициентов A к вырожденной.

1) Всегда $Cond(A) \geq 1$
2) Если $Cond(A) \geq 1000$ – матрица A плохо обусловлена.
3) Если $1 \leq Cond(A) \leq 100$ – матрица A считается хорошо обусловленной.

Число обусловленности обозначают cond(A) и определяют по формуле: $cond(A) = ||A|| \cdot \left|\left|A^{-1}\right|\right|$. Для оценки величины погрешностей и обусловленности систем используют **нормы векторов** и согласованные с ним **нормы матриц.**

Наиболее употребительными на практике являются следующие векторные нормы:

1) абсолютная векторная норма и 1-норма – $||x||_1 = \sum_{i=1}^{n} \left|x_i\right|$.
2) максимальная векторная норма или ∞-норма – $||x||_\infty = max_{1 \le i \le n} \left|x_i\right|$

Согласованные с векторными соответствующие матричные нормы:

1) столбцовая матричная норма или 1-норма максимальная сумма модулей элементов каждого из столбцов матрицы $||A||_1 = max_{1 \le j \le n} \sum_{i=1}^{n} \left|a_{ij}\right|$
2) строчная матричная норма или ∞-норма – это максимальная сумма модулей элементов каждой из строк матрицы $||A||_\infty = max_{1 \le i \le n} \sum_{j=1}^{n} \left|a_{ij}\right|$

9. Приближенное решение систем линейных алгебраических уравнений итерационными методами. Необходимое и достаточное и достаточное условия сходимости итерационного метода.

К итерационным (или приближенным) методам относятся методы, которые даже в предположении, что вычисления ведутся без округлений, позволяют получить решение системы лишь с заданной точностью. Итерационные методы применяются для решения систем высокого порядка. К ним относятся методы:

1) Метод Якоби (метод простой итерации)
2) Метод Зейделя
3) Метод релаксации
4) Метод скорейшего спуска
5) Метод минимальных поправок

Для решения итерационным методом система линейных алгебраических уравнений из вида $Ax = b$ должна быть приведена к виду $x = Gx + f$, где G – некоторая матрица, f - преобразованный вектор свободных членов.

Затем выбирается начальное приближение - некоторый произвольный вектор $x^{(o)}$ – и по формуле $x^{(k)} = Gx^{(k-1)} + f$ строится рекуррентная последовательность векторов $\{x^{(1)}, x^{(2)}, ..., x^{(k)}, ...\}$.

Для сходимости этой последовательности при любом начальном приближении необходимо и достаточно, чтобы все собственные значения матрицы G были по абсолютное величине меньше единицы.

10. Итерационные методы решения систем линейных алгебраических уравнений: условия завершения итерационного процесса.

Для окончания итерационного процесса на практике используют три способа:

При первом определяют **величину стабилизации** и прекращают вычисления, если она меньше ε, т. .е.:

$$\frac{\left\|x^{(k+1)}-x^{(k)}\right\|}{max\{\left\|x^{(k+1)}\right\|,\left\|x^{(k)}\right\|\}} \leq \varepsilon.$$

Недостатком этого способа является то, что при медленно сходящихся итерациях величина стабилизации может быть малой, хотя приближенное решение сильно отличается от точного.

При втором способе вычисляют **нормы невязки** до начала итерация (для начального приближения) и на каждой итерации. Итерации прекращаются при выполнении неравенства:

$$\frac{\left\|Ax^{(k)}-b\right\|}{\left\|Ax^{(0)}-b\right\|} \leq \varepsilon$$

При третьем способе предварительно оценивается **число итераций**, необходимое для получения заданной точности ε. Если для погрешности итерационного метода выполняются оценки $\left\|x^{(k)}-x\right\| \leq q^{k}\left\|x^{(0)}-x\right\|$, где $q \in (0,1)$, то говорят, что метод сходится со скоростью геометрической прогрессии со знаменателем q. Тогда, зная величину q, можно определить число итераций n, решив неравенство $q^{n} < \varepsilon$.

11. Метод Якоби и метод Зейделя решения систем линейных алгебраических уравнений.

Два классических итерационных метода − метод Якоби и метод Зейделя. Оба используют преобразованную систему

$$x_i = -\frac{1}{a_{ii}}\left[\sum_{j=1, j\neq i}^{n} a_{ij}x_j - b_i\right]$$

Метод Якоби использует следующий алгоритм построения приближений:

$$x_i^{(k)} = -\frac{1}{a_{ii}}\left[\sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k-1)} - b_i\right]$$

Если A − матрица с доминирующей диагональю, т.е. $\sum_{i=1}^{n}\left|a_{ii}\right| > \sum_{j\neq i}^{n}\left|a_{ij}\right|$, то метод Якоби сходится при любом начальном приближении.

Модификация метода Якоби, в которой при k-ой итерации вычисления следующего $x_i^{(k)}$ используется уже вычисленные на этом шаге $x_1^{(k)}, x_2^{(k)}$, …, $x_{i-1}^{(k)}$, называется **методом Зейделя**.

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

и приводит, как правило, к ускорению сходимости.

Приведенные выше методы **Якоби** и **Зейделя** относятся к одношаговым итерационным методам – для нахождения $x^{(k+1)}$ требуется помнить только одну предыдущую итерацию.

12. Функции MATHEMATICA, используемые для работы с векторами, матрицами и системами линейных алгебраических уравнений: Dot[….], LinearSolve[….], Inverse[…], Det[….], Norm[…], MatrixForm[…]…

---

1) Dot[...] – операция произведения матриц, векторов, также можно задать с помощью точки.
2) Inverse[A] – вычисляет обратную матрицу A^{-1} для квадратной матрицы A.
3) MatrixForm[{a, b}, {c, d}] – выводит матрицу в традиционном виде.
4) Det[A] – вычисляет определитель матрицы A.
5) Norm[A] – вычисляет норму (длину) вектора/матрицы A.
6) LinearSolve[A, B] – находит решение X для матричного уравнения $A \cdot X = B$

13. Постановка задачи интерполирования функции. Существование и единственность интерполяционного многочлена.

Задача интерполирования состоит в том, чтобы по значениям функции $f(x)$ в нескольких точках отрезка восстановить ее значения в остальных точках этого отрезка. Такая задача допускает сколь угодно много решений. Задача интерполирования возникает, например, в том случае, когда известны результаты измерения $y_h = f(x_k)$ некоторой физической величины $f(x)$ в точках $x_k, k = 0, 1,$ …, и требуется определить ее значения в других точках. Интерполирование используется также при сгущении таблиц, когда вычисление значений x трудоемко. Иногда возникает необходимость приближенной замены или аппроксимации данной функции другими функциями, которые легче вычислить.

Теорема существования и единственности интерполяционного обобщенного многочлена: для того чтобы для любой функции при любых наборах попарно неравных узлов существовал интерполяционный обобщенный многочлен необходимо и достаточно, чтобы эта система функций была системой Чебышева на отрезке [a; b]. При этом интерполяционный обобщенный многочлен будет единственным.

14. Интерполяционная формула Лагранжа. Погрешность интерполяционной формулы.

$$L_k(x) = \sum_{k=0}^{n} f(x_k) \frac{(x-x_0)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)} =$$

$$= \sum_{k=0}^{n} f(x_k) \frac{\prod_{j=0, j\neq k}^{n} (x-x_j)}{\prod_{j=0, j\neq k}^{n} (x_k-x_j)} = \sum_{k=0}^{n} f(x_k) l_k(x), \quad l_k(x_j) = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}$$

Недостатком формы Лагранжа интерполяционного многочлена является то, что при добавлении новых узлов интерполяционный многочлен надо строить заново. Но он удобен при одновременном интерполировании нескольких функций, заданных в одних и тех же узлах, т.к. коэффициенты одинаковы для всех функций и зависят только от узлов:

$$l_k = \frac{\prod_{j=0, j\neq k}^{n} (x-x_j)}{\prod_{j=0, j\neq k}^{n} (x_k-x_j)}$$

Погрешность интерполяции методом Лагранжа зависит от свойств функции, от расположения узлов интерполяции и точки x. Полином Лагранжа имеет малую погрешность при небольших значениях n (n<20). При больших n погрешность начинает расти, что свидетельствует о том, что метод Лагранжа не сходится (то есть его погрешность не убывает с ростом n).

15. Интерполяционные формулы Ньютона для интерполирования в начале и конце таблицы. Погрешность интерполяционных формул.

Часто интерполирование ведется для функций, заданных равномерными сетками, т.е. шаг таблицы $h = x_{i+1} - x_i$ постоянен. Ограничимся рассмотрением интерполяционных формул Ньютона для этого случая. Введем понятие конечных разностей. Пусть функция задана таблицей с постоянным шагом. Разности между значениями функции в соседних узлах называются конечными разностями первого порядка: $\Delta y_i = y_{i+1} - y_i$, $i = 0 \dots n-1$. Из конечных разностей первого порядка можно получить конечные разности второго порядка: $\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$, $i = 0 \dots n-2$ – это разность между двумя соседними разностями первого порядка. Конечные разности третьего порядка: $\Delta^3 y_i = \Delta^2 y_{i+1} - \Delta^2 y_i$, $i = 0 \dots n-3$

Первая интерполяционная формула Ньютона:

$$P_n(x) = P_n(x_0 + th) = y_0 + \Delta y_0 t + \frac{\Delta^2 y_0}{2!} t(t-1) + \frac{\Delta^3 y_0}{3!} t(t-1)(t-2) + \dots$$

$$\dots + \frac{\Delta^n y_0}{n!} t(t-1)(t-2)\dots(t-n+1),$$

где $t = \frac{x-x_0}{h}$. Она применяется, когда точка x, в которой необходимо найти приближенное значение функции, находится в начале отрезка интерполяции. В качестве точки x_0 берется ближайшая к x точка таблицы слева. Точки, правее той, которую взяли за x_0, перенумеровываются. Значение функции в точке x_0 и точках правее нее используются для нахождения конечных разностей, поэтому точек в таблице, правее той, которую мы взяли за x_0, должно быть достаточно, чтобы построить полином нужной степени. Используются точки «впереди» x, поэтому первую интерполяционную формулу Ньютона называют формулой для интерполирования «вперед».

Когда x ближе к концу отрезка интерполяции, формула интегрирования «вперед» может не позволить построить полином нужной степени n. В этом случае лучше использовать узлы слева от точки x. Для этого используется формула интерполирования «назад» – **вторая интерполяционная формула Ньютона**:

$$P_n(x) = P_n(th + x_n) = y_n - \Delta y_{n-1} t + \frac{\Delta^2 y_{n-2}}{2!} t(t+1) + \frac{\Delta^3 y_{n-3}}{3!} t(t+1)(t+2) + \ldots$$

$$\ldots + \frac{\Delta^n y_0}{n!} t(t+1)(t+2)\ldots(t+n-1),$$

Погрешность замены функции интерполяционным многочленом (погрешность интерполирования) $R_n(x) = f(x) - P_n(x)$ зависит от многих факторов – свойств интерполируемой функции, количества и расположения узлов на отрезке, положения точки x относительно узлов. На практике при интерполировании, как правило, используются многочлены не выше 5-й степени.

16. Разделенные разности первого и высших порядков. Таблица разделенных разностей.

Разделенными разностями первого порядка называются отношения:

$$f(x_i, x_j) = \frac{f(x_j) - f(x_i)}{x_j - x_i},\ i \neq j;\ i, j = \overline{0, n}$$

Будем рассматривать разделенные разности, составленные по соседним узлам. По этим разделенным разностям первого порядка можно построить разделенные разности второго порядка:

$$f(x_{n-2}, x_{n-1}, x_n) = \frac{f(x_{n-1}, x_n) - f(x_{n-2}, x_{n-1})}{x_n - x_{n-2}}$$

Аналогично определяются разделенные разности более высокого (k+1)-го порядка по уже известным разностям порядка k:

$$f(x_j, x_{j+1}, \ldots, x_{j+k}, x_{j+k+1}) = \frac{f(x_{j+1}, x_{j+2}, \ldots, x_{j+k+1}) - f(x_j, x_{j+1} \ldots, x_{j+k})}{x_{j+k+1} - x_j}$$

При вычислении разделенных разностей принято записывать их в виде таблицы.

Заметим, что добавление нового узла не изменит уже вычисленных коэффициентов и таблица будет просто дополнена новым столбцом и новой наклонной строкой разделенных разностей.

x_0	$f(x_0)$				
		$f(x_0,x_1)$			
x_1	$f(x_1)$		$f(x_0,x_1,x_2)$		
		$f(x_1,x_2)$		...	
x_2	$f(x_2)$		...		$f(x_0,x_1,\dots,x_n)$
		...			
...	...		$f(x_{n-2},x_{n-1},x_n)$		
...	...	$f(x_{n-1},x_n)$			
x_n	$f(x_n)$				

17. Интерполяция кусочно-непрерывными многочленами. Сплайны. Интерполяционный кубический сплайн.

Полиномиальным сплайном называют функцию, которая непрерывна на отрезке [a, b], имеет на этом отрезке несколько непрерывных производных и на каждом частичном отрезке является алгебраическим многочленом. Максимальная по всем частичным отрезкам степень многочлена называется **степенью сплайна**, а разность между степенью сплайна и порядком наивысшей непрерывной производной - **дефектом сплайна**. На практике наиболее широкое распространение получили сплайны первой и третьей степени.

Пусть отрезок [a, b], на котором определена непрерывная функция $f(x)$, разбит узлам $a = x_0 < x_1 < \dots < x_n = b$ на n частичных отрезков $[x_k, x_{k+1}]$, $k = \overline{0, n-1}$, где $h_k = x_{k+1} - x_k$ – длина каждого отрезка. Значения функции в узлах известны.

$$S_{1,k}(x) = f_k + \frac{x - x_k}{h_k}(f_{k+1} - f_k)$$

Дефект сплайна 1-ой степени равен единице.

Интерполяционным кубическим сплайном, соответствующим данной функции $f(x)$ и данным узлам x_k, называют функцию $S_3(x)$, которая удовлетворяет следующим условиям:

1) на каждом отрезке $[x_k, x_{k+1}]$, $k = \overline{0, n-1}$ функция $S_3(x)$ является многочленом третьей степени вида $S_3(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3$
2) функция $S_3(x)$, а также ее первая и вторая производные непрерывны на отрезке [a, b]
3) в узлах выполняются условия интерполирования $S_3(x) = f(x_k)$, $k = \overline{0, n}$

Задача построения кубического сплайна сводится к нахождению 4n неизвестных коэффициентов a_k, b_k, c_k, d_k. Для ее решения используют условия интерполирования в n+1 узле и условия непрерывности сплайна и его первой и второй производных во внутренних n узлах, что дает в общей сложности 4n - 2 линейных уравнения. Два недостающих уравнения получают из условий закрепления концов сплайна, задавая граничные условия для $S_3(x)$ т.е. значения сплайна или его производных на концах отрезка.

В частности, при свободном закреплении концов можно приравнять к нулю кривизну линии в этих точках, откуда следует равенство нулю вторых производных в этих точках:

$S_3''(a) = S_3''(b) = 0$. Сплайн, удовлетворяющий им, называется *естественным кубическим сплайном* и является самой гладкой среди всех функций данного класса, интерполирующих заданную функцию $f(x)$. Он единственный из них обладает свойством минимальной кривизны.

Составим линейную алгебраическую систему для нахождения неизвестных коэффициентов a_k, b_k, c_k, d_k. Из условий интерполирования в узлах получаем 2n уравнений:

$$S_{3,k}(x_k) = f_k = a_k, \quad S_{3,k}(x_{k+1}) = a_k + b_k h_k + c_k h_k^2 + d_k h_k^3 = f_{k+1}$$

18. Среднеквадратическое приближение функций алгебраическими многочленами. Метод наименьших квадратов нахождения коэффициентов многочлена наилучшего приближения.

Среднеквадратичное приближение, мерой отклонения является **евклидова норма**:

$$\left\| f(x) - \varphi(x, c_0, c_1, \ldots, c_n) \right\|_E = (f - \varphi, f - \varphi)^{1/2}$$

Если $f(x)$ задана во всех точках некоторого отрезка [a, b] и функции $f(x)$ и $\varphi(x)$ интегрируемых с квадратом на [a, b], то

$$\left\| f(x) - \varphi(x, c_0, c_1, \ldots, c_n) \right\|_E = \left(\int_a^b (f(x) - \varphi(x, c_0, c_1, \ldots, c_n))^2 dx \right)^{1/2}$$

Такую задачу можно графически интерпретировать как минимизацию длин отрезков отклонений заданных точек (x_i, y_i) от кривой $y = g(x)$. Среднеквадратичная аппроксимация функций используются обычно в тех случаях, когда значения приближаемой функции известны в достаточно большом количестве точек m, но для нее не удастся построить интерполяционный многочлен обеспечивающий достаточную точность на отрезке [a, b]:

1) приближаемая функция не обладает достаточной гладкостью
2) значения функции известны в достаточно большом количестве точек, но со случайными ошибками

Способ решения задачи среднеквадратичного приближения называется методом наименьших квадратов (МНК) и заключается в построении такого многочлена $P_n^*(x)$, для которого сумма квадратов отклонений его значений в узлах от табличных значений была бы минимальной:

$$\min S(c_0, c_1, c_2, \ldots, c_n) = \min S\left(\sum_{k=0}^{m} (P_n(x_k, c_0, c_1, c_2, \ldots, c_n) - f(x_k))^2 \right)$$

Необходимое условие локального экстремума функции имеет вид:

$$\frac{\partial S}{\partial c_i} = \sum_{k=0}^{n} 2(c_0 + c_1 x_k + c_2 x_k^2 + \ldots + c_n x_k^n - f(x_k)) \cdot x_k^i = 0, \; i = 0, 1, \ldots, n$$

что приводит к следующей системе линейных уравнений (нормальной системе) для нахождения коэффициентов многочлена наилучшего среднеквадратичного приближения в степенной форме:

$$P_n(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n$$

Выбор степенных функций не является оптимальным с точки зрения нормальной системы, т.к. в этом случае матрица системы часто плохо обусловлена. Поэтому на практике для построения многочленов наилучшего квадратного приближения не рекомендуется использовать степенные функций, где n > 4, т.к. будет ухудшаться качество аппроксимации. При этом желательно, чтобы число точек m было больше степени многочлена n как минимум в полтора-два раза.

19. Равномерное приближение функций. Существование многочлена наилучшего равномерного приближения. Теорема Чебышева об альтернансе.

Мерой близости аппроксимирующего многочлена $P_n(x)$ к исходной функции при равномерном приближении является чебышевская норма:

$$\left\|f(x) - P_n(x)\right\|_{C[a,b]} = max\left|f(x) - P_n(x)\right| \text{ для } x \in [a, b]$$

равная максимальному отклонению этих функций друг от друга на отрезке [a, b].

Теорема Чебышева об альтернансе. Если $f(x)$ – непрерывная на замкнутом отрезке [a, b] функция, то среди всех алгебраических многочленов n-ой степени $P_n(x)$ существует единственный многочлен наилучшего равномерного приближения $P_n{}^*(x)$, для которого отклонение минимально. Чтобы многочлен $P_n(x)$ был многочленом наилучшего равномерного приближения, необходимо и достаточно существование на на [a,b] по крайней мере n+2 точек $a \le x_0 < x_1 < \ ... < x_{n+1} \le b$ таких, что

$$f(x_i) - P_n^*(x) = \sigma(-1)^i\left\|f(x) - P_n(x)\right\|_{C[a,b]},\ i = \overline{0, n+1},\ \text{где}\ \sigma = signum[f(x_0) - P_n(x_0)]$$

Если производная $f^{(n+1)}(x)$ не меняет знак на [a, b], то $a = x_0, b = x_{n+1}$. Точки $x_0, x_1, \ ..., \ x_{n+1}$ называются точками чебышевского альтернанса.

20. Функции MATHEMATICA , используемые для приближения функций и построения графиков: InterpolatingPolinomial[….], FindFit[….], Interpolation[….], SplineFit[….], FindMaximum[….], Table[….], TableForm[….], ColumnForm[….], ListPlot[….], Plot[….], Show[….].

---

1) Interpolation[data, Method→"Spline"s] – встроенная сплайн-интерполяция.

2) SplineFit[data] - строит интерполяционный кубический сплайн для функции f(x), заданной таблично. Предварительно необходимо загрузить соответствующий стандартный пакет с помощью команды: <<NumericalMath`SplineFit`.

3) ColumnForm[lst] эквивалентно ColumnForm[lst, Left, Below].

4) InterpolatingPolinomial[lst, x] – многочлен по переменной x, который в узловых точках {x1, x2, …} принимает заданные значения {y1, y2, …}. В общем случае аргумент lst представлен списком {{x1, y1}, {x2, y2}, …}. Список {{1, y1}, {2, y2}, …} можно задать как {y1, y2, …}.

5) FindMaximum [{f [x],a≤x≤b}, x] находит локальный максимум функции f(x) на отрезке [a,b]

6) ListPlot [lst, optns] предназначена для построения графика по точкам. В общем случае аргумент lst представлен списком {{x1, y1}, {x2, y2}, …}, где (x1, y1), (x2, y2), … – координаты отмечаемых точек. Список {{1, y1}, {2, y2}, …} можно задать как {y1, y2, …}. Список также можно создать какой либо функцией, например Range или Table.

7) Plot [{f1, f2, …}, {x, xmin, xmax}, optns] предназначена для построения графиков функций y = f1(x), y = f2(x), … при изменении независимой переменной x в пределах от xmin до xmax. При этом используется прямоугольная (декартова) система координат. Необязательные аргументы optns (опции), общие и для других графических функций, служат для настройки вида графиков. Если опции не указаны, то их стандартные значения устанавливаются автоматически.

8) Show [plot, opns] выводит на экран уже сформированный (например, функцией plot) график. С помощью второго параметра можно изменить значения опций в той графической функции, при помощи которой был получен рисунок.
   Show [{plot1, plot2, …}, opns] совмещает в одном графическом окне несколько графиков. Функция полезна в тех случаях, когда желательно, не вы- числяя заново исходные графики plot1, plot2, …, просмотреть их при иных настройках опций opns или сопоставить.

9) Table [expr, n] – список из n значений одного и того же выражения expr.
   Table [expr, {i, m, n, d}] – список значений выражения expr, зависящего от параметра i, для i от m до n с шагом d.
   Table [expr, {i, m, n}] эквивалентно Table [expr, {i, m, n, 1}]

10) TableForm [lst, opns] выводит на экран двухуровневый список lst в виде таблицы, высота строк и ширина столбцов которой определяются максимальными размерами элементов списка. Линейный список представляется строкой или колонкой в зависимости от значения (Row или Column) опции TableDirections. Если установить опцию TableHeadings, то можно вывести названия для строк и столбцов.

11) FindFit [data, expr, pars, vars] применяется при интерполировании экспериментальных данных методом наименьших квадратов. Интерполирующая функция строится в виде выражения от переменных vars. Аргумент pars – список параметров, значения которых нужно найти. Исходные данные data могут быть заданы списком {{x1, y1}, {x2, y2}, …}.

21. Постановка задачи поиска корня нелинейного уравнения: отделение корней (аналитическое и графическое), уточнение значения корня, погрешность определения корня.

Нелинейное алгебраическое уравнение с одной переменной в общем случае может быть записано в виде $f(x) = 0$. Поэтому более точная постановка задачи состоит в нахождении корней уравнения $f(x) = 0$, расположенных в заданной области комплексной плоскости.

Отделение корней, т.е. установление возможно тесных промежутков, в каждом из которых содержится один, и только один, корень уравнения; Для **аналитического отделения** корней используют следующие теоремы математического анализа:

1) Теорема Больцано-Коши. Если функция $f(x)$ непрерывна на отрезке $[a, b]$ и на его концах принимает значения разных знаков, т.е. $f(a) \cdot f(b) < 0$,то внутри отрезка $[a, b]$ существует по крайней мере один корень уравнения $f(x) = 0$.
2) Если функция $f(x)$ непрерывна на отрезке [a, b] и $-\ f(a) \cdot f(b) < 0$ и производная $f'(x)$ существует и сохраняет постоянный знак в интервале $[a, b]$, то внутри отрезка [a, b] существует ровно один корень уравнения $f(x) = 0$.

Таким образом, чтобы отделить корень аналитически, нужно:

1) найти критические точки x_1, x_2, ..., x_k, x_{k+1}, ..., функции $f(x)$, т.е. точки из области определения функции, в которых производная $f'(x)$ равна нулю, бесконечности или в которых она не существует;
2) вычислить значения функции $f(x)$ в этих точках и ее значения на концах области определения;
3) по знакам функции $f(x)$ и ее производной $f'(x)$ определить интервалы, на которых уравнение имеет ровно один корень;
4) сузить полученные интервалы до нужной длины, так чтобы на его концах функция принимала значения разных знаков.

Графическое отделение корней состоит в построении графика функции $y = f(x)$ и нахождении тех значений x, при которых график пересекает ось абсцисс – они и являются корнями уравнения $f(x) = 0$. Если построение графика функции $y = f(x)$ вызывает затруднение, то от исходного уравнения $f(x) = 0$ следует перейти к равносильному уравнению вида $\varphi_1(x) = \varphi_2(x)$ таким образом, чтобы графики функций $y = \varphi_1(x)$ и $y = \varphi_2(x)$ были достаточно просты. Абсциссы точек пересечения этих графиков и будут корнями уравнения.

После выполнения этапа отделения корней переходят к следующему этапу приближенного решения уравнений – уточнению корней. Пусть искомый корень уравнения отделен, т.е. найден отрезок [a, b], на котором имеется только один корень уравнения. Для вычисления этого корня с требуемой точностью ε обычно применяют какую-либо итерационную процедуру построения числовой последовательности значений {xn}, сходящейся к искомому корню. Начальное приближение x_0 выбирают из отрезка [a, b].

22. Метод половинного деления и метод хорд нахождения приближенного корня нелинейного уравнения. Достоинства и недостатки методов.

Это простейший и надежный алгоритм уточнения простого корня. Он состоит в построении последовательности вложенных отрезков $[a_n, b_n]$, "стягивающихся" к корню. Каждый последующий отрезок получают делением пополам предыдущего и выбором той половины, на концах которой функция принимает значения разных знаков. Погрешность между точным значением корня ξ и приближенным x_n не превосходит длины отрезка $[a_n, b_n]$.

Число итераций n, необходимых для достижения заданной точности $\varepsilon\varepsilon$, легко определяется из неравенства $\frac{n-a}{2^n} < \varepsilon$.

Преимущества:

1) сходится для любых непрерывных функций, в том числе не дифференцируемых;
2) отрезки $[a_n, b_n]$ всегда содержат решение ξ;
3) устойчив к ошибкам округления.

Недостатки:

1) невелика скорость сходимости;
2) неприменим для отыскания кратных корней четного порядка.

Метод хорд подобно методу бисекции строит последовательность вложенных отрезков $[a_n, b_n]$, но в качестве x_n берется абсцисса точки пересечения с осью OX прямой линии (хорды), соединяющей точки $(a_n, f(a_n))$ и $(b_n, f(b_n))$: $x_n = a_n - \frac{f(a_n)}{f(b_n)-f(a_n)}(b_n - a_n)$.

1) Если $f(a) \cdot f''(x) > 0$ на [a; b], то $x_0 = b,\ x_{n+1} = a - \frac{f(a)}{f(x_n)-f(a)}(x_n - a)$
2) Если $f(b) \cdot f''(x) > 0$ на [a; b], то $x_0 = a,\ x_{n+1} = x_n - \frac{f(x_n)}{f(b)-f(x_n)}(b - x_n)$

Если функция $f(x)$ непрерывна на [a, b] и на этом отрезке существует ровно один корень уравнения, то метод хорд сходится, если вторая производная $f''(x)$ сохраняет знак на [a, b]. Возможны два случая: метод сходится линейно, но близость двух очередных приближений не всегда означает, что корень найден с требуемой точностью. Более надежным практическим критерием окончания итераций в методе хорд является выполнение неравенства:

$$\frac{(x_n - x_{n-1})^2}{|2x_{n-1} - x_n - x_{n-2}|} < \varepsilon$$

23. Метод Ньютона и метод секущих нахождения приближенного корня нелинейного уравнения. Достоинства и недостатки метода Ньютона.

Для начала вычислений требуется одного начального приближения x_0, последующие приближения вычисляются по формуле:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ f'(x_n) \neq 0.$$

Метод имеет квадратичную скорость сходимости для простого корня, но очень чувствителен к выбору начального приближения $x0$. При произвольном начальном приближении итерации сходятся, если всюду $|f(x)f''(x)| < (f'(x))^2$, в противном случае сходимость будет только при x_0, достаточно близком к корню.

Существует несколько достаточных условий сходимости:

1) Если производные $f'(x)$ и $f''(x)$ сохраняет знак в окрестности корня, рекомендуется выбирать x_0 так, чтобы $f(x_0) \cdot f''(x_0) > 0$.
2) Если, кроме этого, для отрезка [a, b], содержащего корень, выполняются условия $\left|\frac{f(a)}{f'(a)}\right| < b - a,\ \left|\frac{f(b)}{f'(b)}\right| < b - a$, то метод сходится для любых x_0 на [a, b].

Метод Ньютона весьма быстро сходится, точность каждого приближения в этом методе пропорциональна квадрату точности предыдущего. Основной недостаток метода - необходимость достаточно точного начального приближения.

Метод секущих. Этот метод также является модификацией метода Ньютона, в котором производная $f'(x_n)$ заменена ее разностным приближением:

$$f'(x_n) \approx \frac{f(x_n) - f(x_n - 1)}{x_n - x_{n-1}}, \qquad x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \cdot f(x_n),$$

Метод является двухшаговым и каждое новое приближение к корню строится с использованием двух предыдущих приближений, поэтому для начала итерационного процесса следует выбрать два приближения $x0$ и $x1$ из отрезка [a, b]. При одинаковом объеме вычислений в методе секущих можно сделать больше итераций и получить более высокую точность.

24. Метод простой итерации нахождения приближенного корня нелинейного уравнения. Достаточное условие сходимости метода простой итерации. Приведение к виду, удобному для применения метода.

Метод простой итерации решения уравнения $f(x) = 0$, где $f(x)$ – непрерывная на отрезке [a, b] функция, состоит в замене исходного уравнения эквивалентным ему уравнением $x = \varphi(x)$ – построении последовательности $x_{n+1} = \varphi(x_n)$. Это может быть выполнено бесконечным числом способов. Расчетная формула имеет вид: $x_{n+1} = \varphi(x_n)$, $n = 0, 1, 2, \ldots$, где x_0 – начальное приближение, $x_0 \in [a, b]$. Эти формулы определяют одношаговый общий итерационный метод, называемым методом простых итераций. Установлено, что предел последовательности $x_0, x_1, \ldots, x_n$ при $n \to \infty$, если он существует, является корнем уравнения $f(x) = 0$.

Нахождение корня уравнения называется задачей о неподвижной точке функции $y = \varphi(x)$, т.е. точке пересечения графиков функций $y = \varphi(x)$ и $y = x$. Метод простой итерации не всегда обеспечивает сходимость к корню уравнения. Существование и единственность этого корня основывается на принципе сжимающих отображений.

Достаточным условием сходимости итерационного процесса (при любом начальном приближении) является выполнение неравенства $|\varphi'(x)| < 1$ для $\forall x \in [a; b]$. Метод имеет линейную скорость сходимости.

25. Функции MATHEMATICA для решения нелинейных уравнений: Solve[….], NSolve[….], FindRoot [….], Roots[….], Factor[….]

---

1) Solve[*lhs* == *rhs*, *x*] находит решение уравнения *lhs* == *rhs* относительно переменной *x*. Solve[{*eqn*1, *eqn*2, … }, {*x*1, *x*2, …}] находит решение системы уравнений *eqn*1, *eqn*2, … относительно переменных *x*1, *x*2, …
2) NSolve[*eqn*, *x*] находит численное решение уравнения *eqn* относительно переменной *x*. NSolve[{*eqn*1, *eqn*2, … }, {*x*1, *x*2, …}] находит численное решение заданной системы уравнений.
   NSolve[{*eqn*1, *eqn*2, … }, {*x*1, *x*2, …}, Reals] находит численное решение заданной системы уравнений на множестве действительных чисел.
3) FindRoot [*lhs* == *rhs*, {*x*, *x*0}] находит численное решение уравнения *lhs* == *rhs*, если для переменной *x* выбрано начальное приближение *x*0.
4) Roots[*eqn*, *x*] находит корни полиномиального уравнения *eqn* относительно переменной *x*. Factor[*P*[*x*]] раскладывает многочлен *P*[*x*] на множители с целыми коэффициентами.

26. Постановка задачи численного интегрирования. Квадратурные формулы прямоугольников, трапеций, парабол. Квадратурные формулы Ньютона-Котеса.

Необходимо вычислить определенный интеграл: $I = \int_a^b f(x)dx$ при условии, что пределы интегрирования a и b конечны и $f(x)$ является интегрируемой функцией на всем интервале $x \in [a, b]$. Аналитический метод решения состоит в использовании формулы Ньютона-Лейбница:

$$\int_a^b f(x)dx = F(x)|_a^b = F(b) - F(a)$$

Задача численного интегрирования состоит в нахождении приближенного значения интеграла по заданным или вычисляемым в процессе значениям функции. Приближенное вычисление определенного интеграла основано на замене интеграла конечной суммой по формуле: $\int_a^b f(x)dx \approx \sum_{k=0}^{n} C_k f(x_k)$, называемой **квадратурной формулой**, где C_k – коэффициенты (или веса) квадратурной формулы, точки отрезка интегрирования x_k – узлы квадратурной формулы.

Квадратурные формулы интерполяционного типа (формулы Ньютона-Котеса) получают заменой подынтегральной функции f(x) на [a, b] интерполяционным ногочленом $P_m(x)$ с узлами интерполяции в точках разбиения отрезка интегрирования:

$$\int_a^b f(x)dx \approx \int_a^b P_m(x)dx, \text{ или, точнее } \int_a^b f(x)dx \approx \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} P_m^{(i)}(x)dx$$

Для получения простых формул используют полиномы нулевой, первой и второй степени и, соответственно, получают следующие методы и формулы численного интегрирования:

1) метод прямоугольников (n = 0)

   $\int_a^b f(x)dx \approx (b-a) \cdot f(a)$ -- левые прямоугольники

   $\int_a^b f(x)dx \approx (b-a) \cdot f(b)$ – правые прямоугольники

   $\int_a^b f(x)dx \approx (b-a) \cdot f(\frac{a+b}{2})$ – средние прямоугольники

2) метод трапеций

   $\int_a^b f(x)dx \approx (b-a)(f(a)+f(b))/2$

3) метод Симпсона (парабол)

   $\int_a^b f(x)dx \approx (b-a)(f(a)+4f(\frac{a+b}{2})+f(b))/6$

Очевидно, что во всех случаях замена функции $f(x)$ интерполирующим полиномом приводит к образованию погрешности вычисления значения интеграла. Увеличение числа отрезков разбиения n (уменьшение шага h) ведет к уменьшению погрешности.

27. Численное интегрирование. Квадратурные формулы наивысшей алгебраической точности (формулы Гаусса)

В квадратурной формуле Гаусса узлы и коэффициенты подобраны так, чтобы формула была точна для всех многочленов степени 2m-1. Квадратурная формула Гаусса имеет вид:

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \sum_{k=1}^{m} A_k f(x_k), \qquad x_k = \frac{a+b}{2} + \frac{b-a}{2} t_k$$

где t_k – корни многочленов Лежандра степени m. Если подынтегральная функция достаточно гладкая, то квадратурная формула Гаусса обеспечивает очень высокую точность при небольшом числе узлов. Простейшая формула Гаусса совпадает с формулой средних треугольников. Узлы формулы не совпадают с конечными точками отрезка.

28. Численное интегрирование. Априорная и апостериорная оценка погрешности. Правило Рунге оценки погрешностей.

Вычисление интеграла с заданной точностью по приведенным квадратурным формулам требует либо предварительного определения числа частичных интервалов, либо возможности оценки достигнутой точности (апостериорная оценка) при произвольном числе разбиений отрезка. Полученные выражения остаточных члоенв квадратных формул содержат производные подынтегральной функции в некоторых неизвестных точках отрезка интегрирования. С их помощью получают априорные оценки погрешности интегрирования вида: $R_n(f) \leq C \cdot h^p \cdot M_k$.

Определения шага на основании такой априорной оценки погрешности интегрирования часто оказывается невозможным из-за трудностей определения максимума производных подынтегральной функции. На практике применяют апостериорные оценки погрешности по **правилу Рунге**. Для этого априорные оценки погрешностей квадратурных формул записывают, выделив явно главную часть погрешности, в виде $R(f) = Ah^p + O(h^{p+1})$, где A – коэффициент, зависящий от метода интегрирования и вида подынтегральной функции, p – порядок метода. Вычисляют интеграл по одной и той же формуле дважды – с шагом h и kh (обычно k = 2).

$$I = I_h + Ah^p + O(h^{p+1}), \quad I = I_{kh} + A(kh)^p + O((kh)^{p+1})$$

приравнивают правые части соотношений и определяют главную часть погрешности по первой формуле Рунге: $Ah^p = \frac{I_h - I_{kh}}{k^p - 1}$. Это – апостериорная оценка погрешности и, согласно ей, ошибка более точного приближения в $k^p - 1$ раз меньше разности между двумя приближениями. Уточненное (уточненное по Ричардсону) значение интеграла определяется по второй формуле Рунге: $I_h^T = I_h + \frac{I_h - I_{kh}}{k^p - 1}$. Возможность апостериорно оценить погрешность позволяет вычислять интеграл с заданной точностью путем автоматического выбора шага интегрирования. Если на каком-то частичном отрезке не выполняется неравенство: $\left| \frac{I_{0,5h,i} - I_{h,i}}{2^p - 1} \right| \leq \frac{\varepsilon h_i}{b - a}$, то шаг на этом отрезке надо измельчить ещё в два раза и снова оценить погрешность. Применение такого подхода приводит к адаптивным квадратурам, в которых уменьшается количество вычислений функции в узлах по сравнению с формулами с постоянным шагом за счёт выбора различного шага интегрирования в разных частях интервала в зависимости от поведения функции.

29. Постановка задачи численного дифференцирования. Простейшие формулы численного дифференцирования.

Задача численного дифференцирования состоит в приближенном вычислении значения производной функции $y = f(x)$ в некоторой точке x *, по заданным в конечном числе точек (узлах сетки) значениям этой функции $y_i = f(x_i)$.

Рассмотрим несколько частных случаев интерполяционных многочленов степени n = 1, 2, 3.

На основе линейной – $f'(x) \approx \frac{\Delta y_0}{h}$ для $x \in (x_0 - \delta, x_1 + \delta)$

на основе квадратичной – $f'(x) \approx \frac{1}{h}(\Delta y_0 + \frac{2q-1}{2}\Delta^2 y_0)$ для $(x_0 - \delta, x_2 + \delta)$

на основе кубической – $f'(x) \approx \frac{1}{h}(\Delta y_0 + \frac{2q-1}{2}\Delta^2 y_0 + \frac{3q^2-6q+2}{6}\Delta^3 y_0)$ для $(x_0 - \delta, x_3 + \delta)$

В дальнейшем особый интерес представляют частные случаи формул, связывающие приближенные значения производной функции $f'(x)$ в узлах x_0, x_1, x_2, x_3 с узловыми значениями этой самой функции.

При n = 1:

1) $f'(x_0) \approx y'_0 = \frac{y_1 - y_0}{h}$ – правая разностная производная функция $f(x)$ в точке x_0
2) $f'(x_1) \approx y'_1 = \frac{y_1 - y_0}{h}$ – левая разностная производная функция $f(x)$ в точке x_0

При n = 2 получаем три формулы:

1) $f'(x_0) \approx y'_0 = \frac{1}{2h}(-3y_0 + 4y_1 - y_2)$
2) $f'(x_1) \approx y'_1 = \frac{1}{2h}(-y_0 + y_2)$ – центральная разностная производная
3) $f'(x_2) \approx y'_3 = \frac{1}{2h}(y_0 + 4y_1 + 3y_2)$

При n = 3 получаем четыре формулы для вычисления первой производной:

1) $f'(x_0) \approx y'_0 = \frac{1}{6h}(-11y_0 + 18y_1 - 9y_2 + 2y_3)$
2) $f'(x_1) \approx y'_1 = \frac{1}{6h}(-2y_0 - 3y_1 + 6y_2 - y_3)$
3) $f'(x_2) \approx y'_2 = \frac{1}{6h}(y_0 - 6y_1 + 3y_2 + 2y_3)$
4) $f'(x_3) \approx y'_3 = \frac{1}{6h}(2y_0 - 9y_1 + 18y_2 - 11y_3)$

Наиболее часто используется в приближениях простейшая аппроксимация второй производной с помощью конечной разности второго порядка $\frac{\Delta^2 y_0}{h^2}$ на промежутке $(x_0 - \delta, x_2 + \delta)$, получающийся при n = 2. В частности, в точке x_1имеем приближенное равенство $f''(x_1) \approx y''_1 = \frac{y_0 - 2y_1 + y_2}{h^2}$. Эти формулы используются при построении конечно разностных методов решения краевых задач для обыкновенных дифференциальных уравнений второго порядка.

30. Численное дифференцирование - формулы численного дифференцирования второго порядка точности.

Простейшая симметричная разностная производная. Запишем представление функции $f(x)$ по формулам Тейлора в окрестности точки x_i:

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2}f''(x_i)(x - x_i)^2 + \frac{1}{6}f'''(\xi)(x - x_i)^3$$

Из этого разложения при $x = x_{i-1}$ и $x = x_{i+1}$ получаем соответственно:

$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{1}{2}f''(x_i)h^2 + \frac{1}{6}f'''(\xi_{i+1})h^3$ и

$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{1}{2}f''(x_i)h^2 - \frac{1}{6}f'''(\xi_{i-1})h^3$

Выполнив почленное вычитание этих равенств, получаем:

$$y_{i+1} - y_i = 2hf'(x_i) + \frac{h^3}{6}[(f^m(\xi_{i+1}) + f^m(\xi_{i-1})]$$

откуда с помощью теоремы о среднем, примененной к сумме третьих производных в квадратных скобках, приходим к формуле симметричной аппроксимации $f'(x_i)$ с остаточным членом:

$$f'(x_i) = \frac{y_{i+1} - y_i}{2h} - \frac{h^2}{6}f^m(\xi_i)$$

где ξ_i – некоторая точка из интервала (x_{i-1}, x_{i+1}). Эта формула означает, что аппроксимация первой производной функции симметричной разностной производной $f'(x_i) \approx \frac{y_{i+1} - y_i}{2h}$ имеет второй порядок точности относительно шага $h - O(h^2)$.

31. Функции MATHEMATICA для интегрирования и символьного дифференцирования функций: Integrate [….], NIntegrate [….], D[….], f’[x].

---

1) Integrate[f, x] – дает неопределенный интеграл $\int f dx$

   Integrate[f, {x, a, b}] – дает определенный интеграл $\int_a^b f dx$

   Integrate[f, {x, a, b}, {y, c, d}] – дает кратный интеграл $\int_a^b dx \int_c^d dy \ldots f$

2) NIntegrate[f, {x, a, b}] – даёт численное приближение к $\int_a^b f dx$

   NIntegrate[f, {x, a, b}, {y, c, d}, …] – дает численное приближение к $\int_a^b dx \int_c^d dy \ldots f$

3) D[f, x] – дает частную производную $\partial f / \partial x$

   D[f, {x, n}] – дает кратную производную $\partial^n f / \partial x^n$

D[f, x, y, ...] – дает частную производную $(\partial/\partial y)(\partial/\partial x)f$

D[f, {x, n}, {y, m}, ...] – даёт кратную частную производную $(\partial^m/\partial y^m)(\partial^n/\partial x^n)f$

4) f' – производная функции одного аргумента

32. Численные методы решения задачи Коши для ОДУ. Основные характеристики: явность/неявность, многошаговость.

Рассмотрим задачу Коши для ОДУ первого порядка: требуется найти решение дифференциального уравнения $\frac{du(x)}{dx} = f(x,u),\ x \in [x_0, b]$, удовлетворяющее начальному условию $u(x_0) = u_0$.

В большинстве случаев интегрирование таких уравнений невозможно не только в элементарных функциях, но и в специальных. Рассмотрим численные методы, позволяющие вместо точного решения получить приближенное.

Введем по переменному X равномерную сетку ω_h с шагом h (h > 0), т.е. рассмотрим множество точек $x_k = x_0 + kh,\ k = 1, 2, \ldots, n.$ Точки x_k называются узлами сетки. Введем сеточные функции $u_k = u(x_k),\ y_k = y(x_k),\ f_k = f(x_k, y_k)$, определенные в узлах сетки ω_h. Функции $y_k, f_k = f(x_k, y_k)$ соответствует численному решению разностной задачи, а $u(x)$ – решению дифференциальной.

Численное решение задачи состоит в построении таблицы значений $y_1, y_2, \ldots, y_n$ решения уравнения $y(x)$ в точках $x_1, x_2, \ldots, x_n$. С этой целью дифференциальное уравнение заменяют некоторым разностным $y_{k+1} = \Phi(x_k, y_{k+1}, y_k, \ldots, y_{k-p+1})$, которое необходимо решить на каждом шаге для нахождения y_{k+1}. Выбор функции Ф определяет метод численного решения дифференциального уравнения. Если она не зависит от y_{k+1}, то получают **явный метод** и в противном случае - **неявный.**

Метод, дающий формулу для вычисления y_{k+1} по М-предыдущим значениям $y_k, y_{k-1}, \ldots, y_{r-M+1}$, называется М-шаговым. Существует две группы численных методов решения задачи Коши: одношаговые (или методы Рунге-Кутта) и многошаговые разностные методы.

33. Метод Эйлера, метод Эйлера-Коши численного решения задачи Коши для ОДУ первого порядка.

Вычисление y_{k+1} явным образом по рекуррентной формуле $y_{k+1} = y_k + h \cdot f(x_k, y_k)$ называется методом Эйлера для решения задачи Коши для ОДУ – одношаговый метод, явная схема, неустойчив, первый порядок точности.

Метод Эйлера - Коши. В этом методе первую половину шага совершают с тангенсом угла наклона касательном в предыдущей точке, а вторую половину шага - с тангенсом угла наклона в последующей точке.

$$y_{i+1} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_{i+1}, y_{i+1})).$$

Метод Эйлера-Коши является неявным.

34. Методы Рунге-Кутты численного решения задачи Коши для ОДУ первого порядка.

Идея методов основана на вычислении приближенного решения y_{i+1} в узле x_{i+1} в виде линейной комбинации с постоянными коэффициентами a_m, β_m, γ_m значений функции в промежуточных точках:

$$y_{i+1} = y_i + h[a_1 f(x_i, y_i) + a_2 f(x_i + \beta_1 h, y_i + \beta_2 h k_1) + a_3 f(x_i + \gamma_1 h, y_i + \gamma_2 h k_2) + \ldots]$$
$$k_1 = f(x_i, y_i),\ k_2 = f(x_i + \beta_1 h, y_i + \beta_2 h k_1),\ k_3 = f(x_i + \gamma_1 h, y_i + \gamma_2 h k_2), \ldots$$

Коэффициенты подбирают таким образом, чтобы достичь максимального совпадения с разложением решения в ряд Тейлора по степеням h. В зависимости от старшей степени h^P, с которой учитываются члены ряда, получают разностные схемы разных порядков точности.

Метод Эйлера соответствует случаю p=1. Наиболее известные методы второго порядка (p=2) – метод Эйлера – Коши и модифицированный метод Эйлера.

35. Многошаговые методы решения задачи Коши для ОДУ первого порядка.

Многошаговый метод Адамса (4-шаговый) - для расчета последующей точки необходимо знать координаты четырех предыдущих точек:

$$y_{i+1} = y_i + \frac{h}{24}(55 f(x_i, y_i) - 59 f(x_{i-1}, y_{i-1}) + 37 f(x_{i-2}, y_{i-2}) - 9 f(x_{i-3}, y_{i-3}))$$

В задаче Коши известна только одна начальная точка. Поэтому три последующие точки вычисляются с помощью одношаговых методов, а затем используется 4-шаговых метод Адамса. Данный метод является явным, имеет 4-ый порядок точности.

36. Численные методы решения задачи Коши для ОДУ высших порядков.

Любое дифференциального уравнение высшего порядка можно привести к системе дифференциальных уравнений 1-ого порядка путем замены переменных. Рассмотрим дифференциальное уравнение 2-ого порядка $F(x, y, y', y'') = 0$. Заданы начальные условия x_0, y_0, y'_0. Разрешим уравнение относительно старшей производной: $y'' = f(x, y, y')$. Заменим первую производную y’ функцией z. Тогда y” = z’, а $y'_0 = z_0$.

Получаем систему и решаем её известными методами.

$$\begin{cases} y' = z \\ z' = f(x, y, z) \end{cases}$$

37. Численные методы решения задачи Коши для систем ОДУ.

Рассмотрим систему из двух дифференциальных уравнений 1-ого порядка:

$$\begin{cases} F_1(x, y, z, y', z') = 0 \\ F_2(x, y, z, y', z') = 0 \end{cases}$$

С известными начальными условиями x_0, y_0, z_0. Оба уравнения необходимо разрешить относительно старшей производной:

$$\begin{cases} y' = f_1(x, y, z) \\ z' = f_2(x, y, z) \end{cases}$$

Теперь систему можно решить любым методом применимым для решения ОДУ первого порядка (метод Эйлера (слева), Эйлера-Коши (справа)).

$$\begin{aligned} y_{i+1} &= y_i + h \cdot f_1(x_i, y_i, z_i) \\ z_{i+1} &= z_i + h \cdot f_2(x_i, y_i, z_i) \\ x_{i+1} &= x_i + h \end{aligned}$$

$$\begin{aligned} \tilde{y}_{i+1} &= y_i + h \cdot f_1(x_i, y_i, z_i) \\ \tilde{z}_{i+1} &= z_i + h \cdot f_2(x_i, y_i, z_i) \\ x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + \frac{h}{2} \cdot \left(f_1(x_i, y_i, z_i) + f_1(x_{i+1}, \tilde{y}_{i+1}, \tilde{z}_{i+1})\right) \\ z_{i+1} &= z_i + \frac{h}{2} \cdot \left(f_2(x_i, y_i, z_i) + f_2(x_{i+1}, \tilde{y}_{i+1}, \tilde{z}_{i+1})\right) \end{aligned}$$

38. Функции MATHEMATICA для решения задачи Коши для ОДУ: DSolve[….], DSolve[….], NDSolve[….]

1) DSolve[eqn, y, x] – решает ДУ с функцией y и независимой переменной x;
2) DSolve[eqn, y, {x, x1, x2}] – то же при x в интервале от x1 до x2.
3) NDSolve[eqn, y, {x, x1, x2}] – находит численное решение ДУ при x в интервале [x1, x2]