Содержание

Задания на программирование по вычислительной математике

30 января 2023 г.

1 Введение. Машинная арифметика

1.1 Вычисление многочлена по схеме Горнера

Напишите программу для вычисления значения многочлена с коэффициентами $a_0, \ldots, a_d$ по схеме Горнера

Вычислите значения многочлена

$$(x-2)^9 = x^9-18x^8+144x^7-672x^6+2016x^5-4032x^4+5376x^3-4608x^2+2304x-512$$

на равномерной сетке из 1000 узлов на интервале $[1.92, 2.08]$, постройте график 1.

Вычислите значения на той же сетке по формуле $(x-2)^9$, постройте график 2.

Оцените погрешность в значениях многочлена, вызванную ошибками округления (по аналогии со скалярным произведением, разобранным на лекции). Постройте на графике 1 графики верхней и нижней оценок ошибки.

2 Прямые методы решения линейных систем

2.1 Метод Гаусса с выбором главного элемента по столбцу

При написании программы нельзя использовать готовые функции для матричного умножения и обращения матриц кроме случаев, когда это явно описано.

Напишите программу для решения линейной системы

$$Ax = b$$

методом Гаусса с выбором главного элемента по столбцу. Требования к программе

1. Программа должна содержать функцию, принимающую на вход матрицу A и правую часть b
2. Внутри функции нужно сначала вычислить матрицы L, U и матрицу перестановки P, соответствующую выбору главного элемента по столбцу (т.е. перестановке строк): $PA = LU$.
3. После этого в функции нужно решить системы с треугольными матрицами с помощью прямой и обратной подстановок
4. Функция должна возвращать матрицы и вектор решения: L, U, P, x.
5. Программа должна вызывать реализованную функцию для нескольких матриц и правых частей и выводить норму разницы между полученным решением и решением, которое возвращает готовая библиотечная функция, например, `numpy.linalg.solve`.

2.2 Метод Гаусса с выбором главного элемента по строке

При написании программы нельзя использовать готовые функции для матричного умножения и обращения матриц кроме случаев, когда это явно описано.

Напишите программу для решения линейной системы

$$Ax = b$$

методом Гаусса с выбором главного элемента по строке. Требования к программе

1. Программа должна содержать функцию, принимающую на вход матрицу A и правую часть b

2. Внутри функции нужно сначала вычислить матрицы L, U и матрицу перестановки Q, соответствующую выбору главного элемента по строке (т.е. перестановке столбцов): $AQ = LU$.

3. После этого в функции нужно решить системы с треугольными матрицами с помощью прямой и обратной подстановок.

4. Функция должна возвращать матрицы и вектор решения: L, U, Q, x.

5. Программа должна вызывать реализованную фукнцию для нескольких матриц и правых частей и выводить норму разницы между полученным решением, и решением, которое возвращает готовая библиотечная функция, например, `numpy.linalg.solve`.

2.3 Метод Холецкого

При написании программы нельзя использовать готовые функции для матричного умножения и обращения матриц кроме случаев, когда это явно описано.

Напишите программу для решения линейной системы

$$Ax = b, A = A^T > 0$$

методом Холецкого. Требования к программе:

1. Программа должна содержать функцию, принимающую на вход матрицу $A = A^T$ и правую часть b.

2. Внутри функции нужно вычислить матрицу : $A = CC^T$

3. После этого нужно решить 2 системы с треугольными матрицами с помощью прямой и обратной подстановок.

4. Функция должна возвращать матрицу C и вектор решения x.

5. Программа должна вызывать функцию на нескольких матрицах $A = A^T > 0$ и правых частях, и выводить норму разницы между полученным и решением из стандартной функции, например, `numpy.linalg.solve`.

   Для проверки правильности разложения, можно использовать функцию `numpy.linalg.cholesky`

2.4 QR-разложение

При написании программы нельзя использовать готовые функции для матричного умножения и обращения матриц кроме случаев, когда это явно описано.

Напишите программу для решения системы $Ax = b$ методом наименьших квадратов с помощью QR-разложения Требования к программе:

1. Программа должна содержать функцию, принимающую на вход матрицу A и вектор правой части b.

2. Функция должна сначала вычислять QR-разложение матрицы $A \in \mathbb{R}^{m \times n}$, $m \geq n$ с помощью модифицированного алгоритма Грамма-Шмидта.

3. После вычисления QR разложения нужно решить систему методом наименьших квадратов. **Для умножения на матрицу Q^T можно использовать готовую функцию**. Для решения системы с треугольной матрицей R нужно реализовать метод обратной подстановки.

4. Функция должна возвращать матрицы Q, R и вектор решения x.

5. Программа должна вызывать реализованную функцию для

   (a) квадратной невырожденной матрицы

   (b) прямоугольной матрицы с $m > n$ с линейно независимыми столбцами

   и выводить норму разницы между полученными решениями, и решениями из стандартных функций (например, `numpy.linalg.solve`, `numpy.linalg.lstsq`)

3 Итерационные методы решения линейных систем

3.1 Метод Ричардсона

Напишите программу для решения линейной системы с действительной матрицей

$$Ax = b,\ A = A^T > 0$$

итерационным методом Ричардсона.

Требования к программе:

1. Программа должна содержать функцию, которая принимает на вход матрицу A, правую часть b, итерационный параметр τ максимальное число итераций и параметр tol, задающий критерий остановки для нормы невязки. Функция должна возвращать приближенное решение y, и одномерный массив со значениям нормы невязки на каждой итерации. Остановка должна происходить при норме невязки меньше заданного порога tol.

2. Программа должна создавать матрицу $A = A^T$ произвольного размера $n > 100$, правую часть b.

3. Программа должна вычислять оценку собственных чисел с помощью кругов Гершгорина, и вычислять точные собственные числа с помощью готовой функции (например, `numpy.linalg.eigvals`)

4. Программа должна вычислять приближенное решение с помощью реализованной функции при 3-х различных значениях итерационного параметра:

   (a) Произвольное значения из допустимой области

   (b) Оптимальное значение, вычисленное по оценкам с.ч.

   (c) Оптимальное значение, вычисленное по точным с.ч.

5. Программа должна выводить норму разницы между 3-мя приближенными решениями и точным решением, вычисленным с помощью готовой функции, например, `numpy.linalg.solve`

6. Программа должна выводить время работы готовой функции (с помощью которой вычислялось точное решение), и время работы написанной функции.

   Для ускорения можно использовать декоратор `@jit` из пакета `numba`.

7. Программа должна строить графики зависимости логарифма нормы невязки от номера итерации для 3-х значений итерационного параметра (на одном рисунке).

8. Автор программы должен уметь объяснить полученные результаты на основе изученной теории.

3.2 Метод Зейделя

В итерационном методе нельзя использовать обращение матриц и матричное умножение: нужно реализовать метод с помощью циклов.

Напишите программу для решения линейной системы с действительной матрицей

$$Ax = b,\ A = A^T > 0$$

методом Зейделя.

Требования к программе:

1. Программа должна содержать функцию, которая принимает на вход матрицу A, правую часть b, максимальное число итераций и параметр tol для ограничения на норму невязки для критерия остановки итераций. Программа должна возвращать приближенное решение, вычисленное методом Зейделя, и массив со значениями невязки на каждой итерации.

2. Программа должна создавать симметричную положительно определенную матрицу размера $n > 100$, правую часть и вызывать реализованную функцию.

3. Программа должна выводить число итераций, точную ошибку (вычисленную по точному решению), а также график зависимости логарифма невязки от номера итерации.

4. Программа должна выводить время работы готовой функции (с помощью которой вычислялось точное решение), и время работы написанной функции.

   Для ускорения можно использовать декоратор `@jit` из пакета `numba`.

5. Автор программы должен уметь объяснить полученные результаты на основе изученной теории.

3.3 Симметричный метод Зейделя

В итерационном методе нельзя использовать обращение матриц и матричное умножение: нужно реализовать метод с помощью циклов.

Напишите программу для решения линейной системы с действительной матрицей

$$Ax = b,\ A = A^T > 0$$

симметричным методом Зейделя (чередуются итерации с матрицами $L + D$ и $U + D$)

Требования к программе:

1. Программа должна содержать функцию, которая принимает на вход матрицу A, правую часть b, максимальное число итераций и параметр tol для ограничения на норму невязки для критерия остановки итераций. Программа должна возвращать приближенное решение, вычисленное методом Зейделя, и массив со значениями невязки на каждой итерации.

2. Программа должна создавать симметричную положительно определенную матрицу размера $n > 100$, правую часть и вызывать реализованную функцию.

3. Программа должна выводить число итераций, точную ошибку (вычисленную по точному решению), а также график зависимости логарифма невязки от номера итерации.

4. Программа должна выводить время работы готовой функции (с помощью которой вычислялось точное решение), и время работы написанной функции.

   Для ускорения можно использовать декоратор `@jit` из пакета `numba`.

5. Автор программы должен уметь объяснить полученные результаты на основе изученной теории.

3.4 Метод наискорейшего спуска и метод минимальных невязок

Напишите программу для решения линейной системы с действительной матрицей

$$Ax = b,\ A = A^T > 0$$

методом минимальных невязок и методом наискорейшего спуска.

Требования к программе:

1. Программа должна содержать функцию, которая принимает на вход матрицу A, правую часть b, максимальное число итераций, параметр tol для ограничения на норму невязки для критерия остановки итераций, флаг, который определяет с помощью какого из 2-х методов (метода наискорейшего спуска или метода минимальных невязок) нужно вычислять шаг τ. Программа должна возвращать приближенное решение, вычисленное нужным методом, и массив со значениями невязки на каждой итерации.

2. Программа должна создавать симметричную положительно определенную матрицу размера $n > 100$, правую часть и вызывать реализованную функцию.

3. Программа должна выводить число итераций, точную ошибку (вычисленную по точному решению), а также график зависимости логарифма невязки от номера итерации.

4. Программа должна выводить время работы готовой функции (с помощью которой вычислялось точное решение), и время работы написанной функции.

   Для ускорения можно использовать декоратор `@jit` из пакета `numba`.

5. Автор программы должен уметь объяснить полученные результаты на основе изученной теории.

4 Теория приближения функций

4.1 Интерполяционный многочлен в форме Ньютона

Написать программу для вычисления интерполяционного многочлена в форме Ньютона и его производной.
Требования к программе:

1. Программа должна содержать функцию, которая принимает на вход массив с координатами узлов $[x_0, \ldots, x_n]$, массив значений функции в этих узлах, а также массив точек, в которых нужно вычислить значение интерполяционного многочлена и его производной.

   Функция должна вычислять таблицу разделенных разностей, массив значений интерполяционного многочлена в заданных точках, и массив значений 1-й производной интерполяционного многочлена в заданных точках. Сложность вычисления одного значения интерполяционного многочлена должна быть $\mathcal{O}(n)$, где n – число узлов интерполяции.

2. Программа должна вызывать реализованную функцию для равномерной и чебышёвских сеток и значений какой-то гладкой функции f на этих сетках и выводить 2 рисунка:

   (a) графики функции и многочлена (разными цветами) (по значениям на очень подробной сетке), и значения в узлах интерполяции (маркерами)

   (b) графики производной исходной функции f' и производной интерполяционного многочлена L_n' (на очень подробной сетке).

4.2 Кубический свободный сплайн

Написать программу для построения свободного кубического сплайна (2-е производные на концах равны 0) по табличным данным.
Требования к программе:

1. Программа должна содержать функцию, которая принимает на вход: массив с координатами узлов $[x_0, \ldots, x_n]$, массив значений функции f в этих узлах.

2. Функция должна вычислять коэффициенты свободного кубического сплайна на каждом из отрезков. Для решения 3-х диагональной линейной системы относительно коэффициентов нужно написать функцию, которая реализует метод прогонки.

3. Программа должна содержать функцию, которая вычисляет значение кубического сплайна в заданной точке.

4. Программа должна вызывать реализованные функции для равномерной и неравномерной сеток и какой-то тестовой функции и строить на 1-м рисунке график исходной функции, интерполяционного сплайна (разными цветами), и значения в точках интерполяции (маркерами).

4.3 L_2 - приближение функций

Даны коэффициенты a_k обобщенного многочлена $f = \sum_k a_k \phi_k$,

$$\{\phi_k\} = \{1, \ln(x), x^{-2}, x^{-1}, x, x^2, x^3\}, x \in [x_l, x_r]$$

$x_l > 0, x_r > 0$ – параметры, которые можно менять.

Напишите программу для вычисления коэффициентов b_k наилучшего в $L_2[x_l, x_r]$-норме приближения вида $\sum_k b_k \psi_k$ для

$$\{\psi_k\} = \{1, x, \sqrt{x}, x^2, x^3, x^4, x^6\}$$

Требования к программе:

1. Программа должна содержать функцию, которая принимает на вход массив коэффициентов a и возвращает массив коэффициентов b.

   Можно либо вычислять скалярные произведения с использованием готовых функций для символьного или численного интегрирования, либо найти интегралы аналитически и подставить их в код.

2. Программа должна содержать функции, которые по массивам коэффициентов a и b соответственно и по массиву точек x вычисляют значения обобщенных многочленов в этих точках.

3. Программа должна вызывать описанные функции для двух разных отрезков $[x_l, x_r]$, и строить графики исходного многочлена, его приближения, и график разницы между ними (разница - на отдельном рисунке).

4.4 $L_2(w)$ - приближение функций

Дана функция $f(x),\ x \in [-1, 1]$, и функция $w(x) > 0$.

Напишите программу для вычисления коэффициентов c_k наилучшего приближения в пространстве L_2 с весом $w(x)$ для заданного набора функций: $\phi_1, \ldots, \phi_n$.

Требования к программе:

1. Программа должна содержать функцию, которая принимает на вход f, w, массив функций $\phi_1, \ldots, \phi_n$, и возвращает массив коэффициентов c_k.

   Для вычисления скалярных произведений нужно использовать готовые функции, например, `scipy.integrate.quad`.

2. Программа должна вызывать функцию для тестовых функций f, w, и для двух небольших ($n < 10$) наборов базовых функций ϕ_k, и строить графики f, приближения, и график разницы между ними (разница - на отдельном рисунке).

Алгоритм Ремеза (дополнительное задание)

Написать программу для вычисления многочлена наилучшего приближения для функции f в норме $C[a, b]$.
Требования к программе:

1. Программа должна содержать функцию, которая принимает на вход функцию f, отрезок $[a, b]$ и степень многочлена n, и возвращает многочлен (в любом удобном виде, например, набор коэффициентов), а также $C[a, b]$-норму ошибки, оцененную на подробной сетке (10000 узлов) на отрезке $[a, b]$.

2. Функция должна вычислять многочлен наилучшего приближения итерационно, с помощью алгоритма Ремеза.

3. Программа должна вычислять многочлен для какой-то тестовой функции, и выводить 2 рисунка:

   (a) на одном рисунке – график ошибки $e = f(x) - p_n(x)$ и график ошибки $f(x) - L_n(x)$ при интерполяции многочленом степени n по значениям f в узлах Чебышёва (для интерполяции можно использовать готовую функцию, например BarycentricInterpolator)

   (b) на одном рисунке график функции f и график многочлена наилучшего приближения p_n.

4. Убедитесь, что алгоритм работает правильно: ошибка должна иметь $n + 2$ точки альтернанса.

5 Численное дифференцирование и интегрирование

5.1 Конечно-разностная формула на произвольном шаблоне

Написать программу для вычисления приближенного значения производной заданного порядка по значениям функции в заданных узлах.

1. Программа должна содержать функцию, принимающую на вход: массив координат узлов $x_1, \ldots, x_n$, массив значений функции в узлах $f_1, \ldots, f_n$, порядок производной $k \geq 1$, и точку x_0 в которой нужно вычислить значение производной.

2. Функция должна вычислять и возвращать коэффициенты конечно-разностной формулы и приближенное значение производной $f^{(k)}(x_0)$.

3. Программа должна содержать тестовый расчет для гладкой функции: функция должна вызываться для последовательности наборов одного фиксированного количества узлов, чтобы величина $x_i - x_0$ уменьшалась каждый раз вдвое. Программа должна строить график зависимость ошибки от шага $h = \max_i |x_i - x_0|$ в логарифмической шкале. На этом же графике нужно нарисовать прямую с углом наклона, равным порядку аппроксимации.

5.2 Матрицы дифференцирования для чебышёвских узлов

Напишите программу для приближенного вычисления производной функции в узлах, в которых заданы значения самой функции.

1. Программа должна содержать фукнцию, которая принимает на вход границы отрезка $[a, b]$, число узлов, функцию f. Функция должна вычислять чебышёвские узлы, создавать *плотную* матрицу дифференцирования по этим узлам. Матрица получается путем дифференцирования интерполяционного многочлена, построенного *по всем узлам сетки.*

   Функция должна вычислять массив приближенных значений производных в узлах через умножение матрицы дифференцирования

на вектор значений f в узлах. Функция должна возвращать массив значений производной.

2. Программа должна вызывать реализованную функцию для тестовой функции f в цикле для разного количества узлов, и вычислять ∞-норму ошибки для каждого количества узлов.

3. Программа должна строить график зависимости ошибки от числа узлов в логарифмической шкале.

4. Ошибка в логарифмической шкале должна убывать быстрее, чем линейно (сверхлинейно)! Если это не так – ищите ошибку.

5.3 Квадратуры Гаусса

Написать программу для вычисления интеграла с помощью квадратурной формулы Гаусса.

1. Программа должна содержать функцию, которая принимает на вход ссылку на функцию f, отрезок $[a, b]$, и число узлов n.

2. Функция должна вычислять узлы квадратуры Гаусса: нужно создать 3-х диагональную матрицу с коэффициентами 3-х членного реккурентного соотношения (коэффициенты можно взять из литературы) и вычислить собственные числа с помощью готовой функции.

3. Функция должна вычислять веса квадратурной формулы через интегралы от базовых многочленов Лагранжа по узлам квадратуры. Для вычисления интегралов можно вывести аналитические формулы или использовать встроенную функцию для интегрирования.

   Для проверки правильности вычисления весов и узлов можно использовать функцию numpy.polynomial.legendre.leggauss

4. Функция должна возвращать приближенное значение интеграла, вычисленное по квадратурной формуле Гаусса.

5. Программа должна вызывать функцию для многочленов и сравнивать значение с точным значением интеграла. Нужно подтвердить, что формула точна для многочленов нужной степени.

6. Программа должна вычислять интеграл для тестовой функции (не многочлена) и сравнивать полученное значение со значением, которая вернула стандартная функция

5.4 Составные квадратурные формулы

Написать программу для вычисления интеграла по составной квадратурной формуле.

1. Сначала нужно вычислить веса квадратурной формулы с узлами

   $$-1, -1/3, 1/3, +1$$

   для стандартного отрезка $[-1, +1]$.

2. Программа должна содержать функцию, которая принимает на вход ссылку на функцию f, отрезок $[a, b]$, и точность tol.

3. Функция должна вычислять интеграл $\int_a^b f(x)\,dx$ по составной квадратурной формуле на основе выведенной формулы на сгущающихся равномерных сетках. Функция должна вычислять оценку ошибки по правилу Рунге, и прекращать сгущение, когда оценка ошибки станет меньше tol.

4. Функция должна возвращать значение интеграла и число отрезков в последней сетке, для которой выполнился критерий остановки сгущения.

5. Программа должна вычислять интеграл для гладкой функции для разных значений $tol = 2^{-5}, 2^{-10}, \ldots$ и строить график зависимости tol от числа отрезков разбиения в логарифмической шкале.

6. Порядок убывания ошибки должен совпадать с теоретической оценкой

6 Численное решение ОДУ

6.1 Решение нелинейной краевой задачи

Написать программу для численного решения краевой задачи:

$$u''(x) = u(2x^2 - 2\ln u - 2),\ u(0) = 1,\ u(1) = 1/e$$

с помощью конечно-разностного метода 2-го порядка

1. Программа должна содержать функцию, которая принимает на вход количество узлов и порог точности для остановки итераций в методе Ньютона, и возвращает массив значений решения в узлах сетки и координаты узлов сетки.

2. Фукнция должна решать нелинейную систему уравнений, полученную после конечно-разностной аппроксимации исходной задачи, методом Ньютона.

3. Для решения линейной системы на каждой итерации нужно использовать либо готовую либо собственную функцию для решения трехдиагональной системы.

4. Программа должна запускать расчет для сгущающихся равномерных сеток (шаг уменьшается вдвое), и строить в логарифмической шкале график зависимости ошибки от величины шага. За условно точное решение нужно взять численное решение на очень подробной сетке. График должен показывать, что порядок аппроксимации равен 2.

5. Программа должна выводить графики численного и точного решения для тестовой задачи.

6.2 Численное решение задачи Коши

Написать программу для решения задачи Коши для произвольной системы ОДУ:

$$\vec{u}_t = \vec{f}(t, \vec{u})$$
$$\vec{u}(0) = \vec{u}_0$$

методом Рунге-Кутты с таблицей Бутчера:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

Перед решением задачи нужно подставить коэффициенты метода в условия порядка и найти порядок метода.

Требования к программе:

1. Программа должна содержать функцию, принимающую на вход: отрезок $[0, T]$, вектор начального условия $\vec{u}_0$, ссылку на функцию $\vec{f}(t, \vec{u})$, ссылку на функцию для вычисления Якобиана $\frac{\partial \vec{f}}{\partial \vec{u}}$, количество шагов интегрирования, точность для критерия остановки итераций по нелинейности

2. Функция должна решать систему методом Рунге-Кутты, для решения нелинейного уравнения нужно использовать метод Ньютона.

3. Программа должна выводить графики численного и точного решения для тестовой задачи.

4. Программа должна делать расчёты на последовательности вложенных сеток с уменьшением шага вдвое, вычислять «фактический» порядок аппроксимации и строить график зависимости ошибки от шага сетки (или от числа узлов) в логарифмическом масштабе. *Фактический порядок должен быть близок к теоретическому на гладких решениях.*

6.3 Спектральный метод для линейной краевой задачи

Написать программу для численного решения линейной краевой задачи

$$u''(x) = f(x),\ u(-1) = a,\ u(1) = b$$

спектральным методом.

1. Программа должна содержать функцию, принимающую на вход число узлов и ссылку на функцию f.

2. Функция должна строить чебышёвскую сетку, и заполнять матрицу системы уравнений относительно неизвестных значений в узлах. В каждой внутреннем узле нужно использовать конечно-разностную формулу для 2-й производной, построенную по всем узлам сетки. Для решения линейной системы можно использовать готовую фукнцию.

3. Функция должна возвращать массив узлов и массив значений в узлах.

4. Программа должна строить для тестовой задачи с точным решением график зависимости ошибки от числа узлов в логарифмической шкале (нужно в цикле выполнять вычисления на сгущающихся сетках).

5. *Убедитесь, что в логарифмической шкале ошибка убывает быстрее, чем линейно. В противном случае, в программе есть ошибка.*

6. Программа должна выводить графики численного и точного решения для тестовой задачи.

6.4 Численное решение задачи Коши

Написать программу для решения задачи Коши для произвольной системы ОДУ:

$$\vec{u}_t = \vec{f}(t, \vec{u})$$
$$\vec{u}(0) = \vec{u}_0$$

методом Рунге-Кутты с таблицей Бутчера:

$1/4$	$1/4$	0
$3/4$	$1/2$	$1/4$
	$1/2$	$1/2$

Перед решением задачи нужно подставить коэффициенты метода в условия порядка и найти порядок метода.

Требования к программе:

1. Программа должна содержать функцию, принимающую на вход: отрезок $[0, T]$, вектор начального условия $\vec{u}_0$, ссылку на функцию $\vec{f}(t, \vec{u})$, ссылку на функцию для вычисления Якобиана $\frac{\partial \vec{f}}{\partial \vec{u}}$, количество шагов интегрирования, точность для критерия остановки итераций по нелинейности

2. Функция должна решать систему методом Рунге-Кутты, для решения нелинейного уравнения нужно использовать метод Ньютона.

3. Программа должна выводить графики численного и точного решения для тестовой задачи.

4. Программа должна делать расчёты на последовательности вложенных сеток с уменьшением шага вдвое, вычислять «фактический» порядок аппроксимации и строить график зависимости ошибки от шага сетки (или от числа узлов) в логарифмическом масштабе. *Фактический порядок должен быть близок к теоретическому на гладких решениях.*

Задача

Написать программу для решения задачи Коши для произвольного скалярного ОДУ:

$$u_x = f(x, u)$$
$$u(0) = u_0$$

Функции $f(x, u)$ и $f_u(x, u)$ задаются в программе явно.

Перед решением всех задач нужно подставить коэффициенты метода в условия порядка и убедиться, что в них нет ошибки.

Требования к программе:

- Программа должна вычислять приближенное решение на заданном интервале с помощью численного метода (см. ниже) на равномерной сетке с заданным числом узлов
- Для решения нелинейного уравнения в неявном методе нужно запрограммировать метод Ньютона
- Дополнительные начальные значения (в многошаговых методах) должны вычисляться с тем же порядком.
- Программа должна строить график численного решения и график точного решения (если оно известно)
- Программа должна вычислять ошибку численного решения в норме $||.||_\infty$
- Программа должна делать расчёты на последовательности вложенных сеток с уменьшением шага вдвое, вычислять «фактический» порядок аппроксимации и строить график зависимости ошибки от шага сетки (или от числа узлов) в логарифмическом масштабе. **Фактический порядок должен быть близок к теоретическому на гладких решениях. Если нет - ищите ошибку.**

Варианты методов:

1. Метод Р-К 1-го порядка с таблицей Бутчера:

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

2. Метод Р-К 2-го порядка с таблицей Бутчера:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

3. Многошаговый метод 3-го порядка:

$$y_{n+2} = y_{n+1} + h\left(\frac{5}{12}f(x_{n+2}, y_{n+2}) + \frac{2}{3}f(x_{n+1}, y_{n+1}) - \frac{1}{12}f(x_n, y_n)\right)$$

4. Многошаговый метод 4-го порядка:

$$y_{n+3} = y_{n+2} + h\left(\frac{3}{8}f(x_{n+3}, y_{n+3}) + \frac{19}{24}f(x_{n+2}, y_{n+2}) - \frac{5}{24}f(x_{n+1}, y_{n+1}) + \frac{1}{24}f(x_n, y_n)\right)$$

7 Дополнительные задачи на выбор

Задачи могут быть описаны кратко, постановку нужно обсуждать с преподавателем.

7.1 Решение уравнения Пуассона с помощью нейронной сети

Для решения краевой задачи для уравнения Пуассона в круге:

$$u_{xx} + u_{yy} = u^2 + \frac{3}{2}u^3 \tag{1}$$

$$u|_{x^2+y^2=1} = -2 \tag{2}$$

Для выполнения граничного условия можно использовать подстановку:

$$u(x, y) = (1 - x^2 - y^2)n(x, y) - 2 \tag{3}$$

Такая функция удовлетворяет граничному условию для любой ограниченной $n(x, y)$.

Для аппроксимации функции n нужно использовать нейросеть, у которой 2 входа (x и y) и один выход (значение неизвестной функции $n(x, y)$). Функция ошибки, которую нужно минимизировать, это невязка в уравнении:

$$\varepsilon_0 = (u_{xx} + u_{yy} - f(x, y, u))^2 \tag{4}$$

Её можно вычислить дифференцируя нейросеть по входным параметрам (x и y), для этого в каждой реализации есть соответствующая функция, например, в TensorFlow – tf.gradients. Здесь усложнение только в том, что нужно считать 2-е производные.

Обучать нейросеть нужно на точках, внутри области $x^2 + y^2 = 1$, для начала можно попробовать случайно накиданные точки.

Можно использовать статью с такой конфигурацией слоев:

$$2, 64, 64, 64, 64, 64, 64, 1$$

Сигмоиды должны быть гладкими (бесконечно дифференцируемые).

После обучения, нужно сравнить полученную аппроксимацию с аналитическим решением:

$$u = \frac{4}{x^2 + y^2 - 3} \tag{5}$$

Подробная статья с описанием: ссылка