Содержание

Задания по вычислительной математике

10 декабря 2018 г.

1 Вычислительная линейная алгебра

1.1 Метод простой итерации

Написать программу для решения линейной системы

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

методом простой итерации (методом релаксации).

Требования к программе:

- Программа должна принимать на вход размерность системы n, создавать случайную положительно определённую матрицу и правую часть такого размера.
- После этого нужно найти точное решение с помощью функции из стандартной библиотеки `numpy.linalg.solve`.
- Перед использованием итерационного метода нужно найти оценку собственных чисел матрицы с помощью кругов Гершгорина, сравнить с точными собственными числами (функция из стандартной библиотеки `numpy.linalg.eigvals`).
- Программа должна вычислять приближенное решение методом простой итерации с произвольным параметром, в том числе с оптимальным значением вычисленным по оценкам собственных чисел и по точным собственным числам. На входе задаётся требуемая точность, которая используется в критерии остановки итераций.
- Программа должна выводить число итераций, точную ошибку (вычисленную по точному решению), а также график зависимости логарифма ошибки от номера итерации.
- Автор программы должен уметь объяснить полученные результаты на основе изученной теории.

1.2 Метод Зейделя

Написать программу для решения линейной системы

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

методом Зейделя.

Требования к программе:

- Программа должна принимать на вход размерность системы n, создавать случайную положительно определённую матрицу и правую часть такого размера.
- После этого нужно найти точное решение с помощью функции из стандартной библиотеки `numpy.linalg.solve`.
- Программа должна вычислять приближенное решение методом Зейделя, **причем в итерационном методе нельзя использовать обращение матриц и матричное умножение: нужно реализовать метод поэлементно с помощью циклов**. На входе задаётся требуемая точность.
- Программа должна выводить число итераций, точную ошибку (вычисленную по точному решению), а также график зависимости логарифма ошибки от номера итерации.
- Автор программы должен уметь объяснить полученные результаты на основе изученной теории.

1.3 Метод Гаусса с выбором главного элемента

Написать программу для решения линейной системы

$$Ax = b$$

методом Гаусса с выбором главного элемента (по строке или по столбцу). Требования к программе

- Программа должна принимать на вход матрицу и правую часть
- Сначала нужно вычислить матрицы L, U и матрицу перестановки P
- После этого нужно решить системы с треугольными матрицами
- **Нельзя использовать матричное умножение и обращение матриц. Метод нужно реализовать с помощью циклов, поэлементно**.
- Программа должна выводить норму разницы между полученным решением и решением из стандартной функции `numpy.linalg.solve`

1.4 Метод Холецкого

Написать программу для решения линейной системы

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

методом Холецкого. Требования к программе

- Программа должна создавать матрицу $A = A^T > 0$ и правую часть
- Сначала нужно вычислить матрицу : $A = CC^T$
- После этого нужно решить 2 системы с треугольными матрицами
- **Нельзя использовать матричное умножение и обращение матриц. Метод нужно реализовать с помощью циклов, поэлементно**.
- Программа должна выводить норму разницы между полученным и решением из стандартной функции `numpy.linalg.solve`. Для проверки правильности разложения, можно использовать функцию `numpy.linalg.cholesky`

2 Теория приближения функций

2.1 Алгоритм Ремеза

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

1. Программа должна принимать на вход функцию f, отрезок $[a, b]$ и степень многочлена n.
2. Программа должна вычислять коэффициенты многочлена наилучшего приближения итерационно, с помощью алгоритма Ремеза.
3. Программа должна выводить $C[a, b]$-норму ошибки, оцененную на подробной сетке (10000 узлов) на отрезке $[a, b]$.
4. Программа должна на одном рисунке выводить график ошибки $e = f(x) - p_n(x)$ и график ошибки $f(x) - L_n(x)$ при интерполяции многочленом степени n по значениям в узлах Чебышёва (для интерполяции можно использовать готовую функцию); на 2-м рисунке должны быть график функции и график многочлена наилучшего приближения.

2.2 L_2 - приближение функций

Даны коэффициенты a_k обобщенного многочлена $f = \sum_k a_k \phi_k$, по набору функций $1, \ln(x), x^{-2}, x^{-1}, x, x^2, x^3$ на отрезке $[0.1, 1]$. написать программу для вычисления коэффициентов b_k наилучшего L_2 приближения $\sum_k b_k \psi_k$ по набору функций $1, x, x^2, x^3, x^4, x^6, x^7$.
Требования к программе:

1. Программа должна вычислять коэффициенты путем решения линейной системы с матрицей Грамма для данной системы функций. Скалярные произведения можно вычислять либо с использованием готовых функций для символьного или численного интегрирования, либо вывести на бумаге готовые формулы и подставить их в код.

2. Программа должна выводить коэффициенты b_k и строить на одном рисунке графики исходной функции и полученного приближения.

3. Для проверки нужно выполнить расчет для

$$a = [198.22, 14.05, 0.039, -1.33, 10.33, -0.125, -0.337]$$

2.3 Интерполяционный многочлен в форме Ньютона

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

1. Программа принимает на вход массив с координатами узлов $[x_0, \ldots, x_n]$, и массив значений функции в этих узлах. Нужно реализовать возможность использовать узлы Чебышёва.

2. Программа должна вычислять таблицу разделенных разностей.

3. Программа должна вычислять значение интерполяционного многочлена в любой точке x за $\mathcal{O}(n)$

4. Программа должна вычислять значение производной интерполяционного многочлена в любой точке x.

5. Программа должна на одном рисунке строить график функции, многочлена (разными цветами), и значений в узлах интерполяции (маркерами), на другом рисунке - графики производной функции $f'(x)$ и производной интерполяционного многочлена $L_n'(x)$.

2.4 Кубический свободный сплайн

Написать программу для построения свободного (2-е производные на концах равны 0) кубического сплайна по табличным данным. Можно использовать готовые функции для решения линейных систем уравнений.
Требования к программе:

1. Программа принимает на вход: массив с координатами узлов $[x_0, \ldots, x_n]$ и массив значений функции f в этих узлах.
2. Программа должна вычислять коэффициенты свободного кубического сплайна.
3. Программа должна строить на 1-м рисунке график исходной функции, интерполяционного сплайна (разными цветами), и значения в точках интерполяции (маркерами).

3 Решение дифференциальных уравнений, решение нелинейных уравнений

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 $||\cdot||_\infty$
- 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:

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

4 Дополнительные задачи на выбор

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

4.1 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

4.2 Спектроскопия динамического рассеяния света

Любое задание отсюда: https://vk.com/numan_research. Эти описания будут в ближайшее время обновляться.