НА ПРАВАХ РУКОПИСИ

Першин А.Ю., Соколов А.П., Гудым А.В.

Сборник постановок задач на лабораторные работы по курсу «Вычислительная математика»

Учебно-методическое пособие

Москва, 2025

УДК 519.6
ББК 22.19
П26

Учебно-методическое пособие разработано на инициативной основе

Авторы

ПЕРШИН, Антон Юрьевич – Ph. D., доцент кафедры САПР, МГТУ им. Н.Э. Баумана

СОКОЛОВ, Александр Павлович – д-р тех. наук, проф. кафедры САПР, МГТУ им. Н.Э. Баумана

ГУДЫМ, Антон Васильевич – ассистент кафедры САПР, МГТУ им. Н.Э. Баумана

Рецензент

МАРЧЕВСКИЙ, Илья Константинович – д-р физ.-мат. наук, проф. кафедры прикладной математики, МГТУ им. Н.Э. Баумана

П26 Першин А.Ю., Соколов А.П., Гудым А.В. **Сборник постановок задач на лабораторные работы по курсу «Вычислительная математика»**: Учебно-методическое пособие. [Электронный ресурс] — Москва: 2025. — 46 с. URL: https://archrk6.bmstu.ru (облачный сервис кафедры РК6)

Представлены задания, краткие вспомогательные сведения для выполнения лабораторных работ в рамках курса «Вычислительная математика». Теоретические материалы рассматриваются на лекциях в процессе обучения и доступны в форме конспекта [1]: включают сведения о методах и подходах, применяемых в вычислительной математике. Представлены постановки задач по следующим тематикам: «Интерполяционные многочлены Лагранжа и Эрмита. Интерполяция сплайнами. Автоматическое дифференцирование»; «Численное дифференцирование. Численное интегрирование (квадратуры Гаусса, Гаусса–Лобатто). Тригонометрическая аппроксимация. Алгоритм Кули–Тьюки. Дискретное преобразование Фурье. Метод наименьших квадратов. Линейная и нелинейная регрессии. Анализ временных рядов»; «Прямые и итерационные методы решения СЛАУ. Разложение Холецкого и LU-разложение. Положительно определённые матрицы. Матричные нормы. Модель Лоренца. Вынужденные колебания маятника. Разреженные матрицы»; «Задача Коши. Методы Рунге–Кутты, Адамса, Адамса–Башфорта, Адамса–Моултона, многошаговые методы численного интегрирования СОДУ. Анализ вычислительной устойчивости».

Учебно-методическое пособие разработано для студентов 3-го года обучения бакалавриата кафедры «Системы автоматизированного проектирования» (РК6) МГТУ им. Н.Э. Баумана.

УДК 519.6
ББК 22.19

Сборка: [git] • master @ cca3929 • Alexandr Sokolov(2025-03-07 23:57:19 +0300)

Содержание

Введение

Представлены задания, краткие вспомогательные сведения для выполнения лабораторных работ в рамках курса «Вычислительная математика». Теоретические материалы рассматриваются на лекциях в процессе обучения и доступны в форме конспекта [1]: включают сведения о методах и подходах, применяемых в вычислительной математике. Представлены постановки задач по следующим тематикам: «Интерполяционные многочлены Лагранжа и Эрмита. Интерполяция сплайнами. Автоматическое дифференцирование»; «Численное дифференцирование. Численное интегрирование (квадратуры Гаусса, Гаусса–Лобатто). Тригонометрическая аппроксимация. Алгоритм Кули–Тьюки. Дискретное преобразование Фурье. Метод наименьших квадратов. Линейная и нелинейная регрессии. Анализ временных рядов»; «Прямые и итерационные методы решения СЛАУ. Разложение Холецкого и LU-разложение. Положительно определённые матрицы. Матричные нормы. Модель Лоренца. Вынужденные колебания маятника. Разреженные матрицы»; «Задача Коши. Методы Рунге–Кутты, Адамса, Адамса–Башфорта, Адамса–Моултона, многошаговые методы численного интегрирования СОДУ. Анализ вычислительной устойчивости».

Инструкция по выполнению [2] лабораторных работ размещена в облачном сервисе кафедры по адресу `https://archrk6.bmstu.ru` в разделе «70 - Инструкции. Образование».

Лабораторная работа 1

1.1. Требования к знаниям для выполнения

Для выполнения лабораторной работы обучающийся должен обладать знаниями:

– владеть навыками разработки программного обеспечения на языке Python (рекомендуется) или C++ на базовом уровне;

– владеть навыками использования программных инструментов: `numpy`, `matplotlib`;

– знать понятия: интерполяция, интерполяционный полином Лагранжа, принципы интерполяции кубическими сплайн-функциями.

1.2. Интерполяция Лагранжа

Интерполяция Лагранжа является одним из самых важных численных методов и лежит в основе многих методов численного дифференцирования и интегрирования. Точность интерполяции полиномами Лагранжа зависит не только от максимальной степени выбранного подмножества полиномов, но и от расположения узлов. Очевидный, казалось бы, выбор равномерно расположенных узлов может приводить к неожиданным проблемам. Одним из примеров является так называемый эффект Рунге, который выражается в большой осцилляции аппроксимационного полинома вблизи конечных узлов отрезка интерполирования и который предлагается исследовать в базовой части настоящего задания. В продвинутой части предлагается более систематически исследовать влияние расположения узлов и их количества на интерполяцию Лагранжа, используя случайные функции, сгенерированные с помощью аппроксимации Паде.

Задача 1.1 (интерполирование полиномами Лагранжа)
Даны функции[1]:

$$f(x) = \frac{1}{1+25x^2}, \quad f_{n,m}(x) = \frac{\sum\limits_{j=0}^{m} a_j x^j}{1 + \sum\limits_{k=1}^{n} b_k x^k}, \quad x \in [-1;1]. \tag{1}$$

Требуется (базовая часть)

1. Разработать функцию l(i, x, x_nodes), возвращающую значение базисного полинома Лагранжа $l_i(x)$, заданного на узлах с абсциссами x_nodes, в точке x.

2. Написать функцию L(x, x_nodes, y_nodes), которая возвращает значение интерполяционного полинома Лагранжа, заданного на узлах с абсциссами x_nodes и ординатами y_nodes, в точке x.

3. Для равномерно расположенных узлов вывести на экран одновременно графики $f(x)$ и полученного интерполяционного полинома $L(x)$ для следующих количеств узлов: $5, 8, 11, 14, 17, 20, 23$. В результате это должно дать 7 пар графиков. Опишите, что наблюдается при увеличении количества узлов[2]?

4. Повторить предыдущий пункт для чебышёвских узлов. В чем разница между интерполяцией Лагранжа функции $f(x)$ на основе равномерно расположенных узлов и чебышёвских? Сделать выводы.

Требуется (продвинутая часть)

5. Сгенерировать 100 функции $f_{n,m}(x)$, где целые степени $n, m \in [7;15]$ и вещественные коэффициенты $a_j, b_k \in [0;1]$ генерируются случайным образом для каждой из функции.

6. Для нескольких из сгенерированных функций вывести на экран одновременно графики $f_{n,m}(x)$ и соответствующего интерполяционного полинома $L(x)$, построенного по N равномерно расположенным узлам, где N выбирается по собственному усмотрению, но должно быть не меньше 5. На том же графике выведите $L(x)$, построенного по N чебышёвским узлам.

7. Для каждой из функции, сгенерированных в предыдущем пункте, найдите интерполяционные полиномы $L(x)$, построенные по $N \in \{1, 2, \ldots, 30\}$ равномерно расположенным узлам и чебышёвским узлам. Для каждого N рассчитайте расстояние между $f_{n,m}(x)$ и $L(x)$ в лебеговом пространстве L_∞[3]. Рассмотрите несколько графиков зависимости этого расстояния для равномерных и чебышёвских узлов от N и сделайте по ним вывод[4]. Добавьте в отчет один характерный график, который наглядно демонстрирует верность вашего вывода.

8. Объясните, что такое аппроксимация Паде и до какой степени предложенный метод генерации случайных функций $f_{n,m}(x)$ позволяет обобщить выводы предыдущего пункта на произвольные функции.

---

[1] Вторая дробно-рациональная функция известна, как аппроксимация Паде.

[2] Подсказки: используйте разные массивы x-точек для аппроксимации и построения графика. Первые узлы подаются единым массивом на вход функции L() как x_nodes, в то время как каждый узел второй группы используется как аргумент x в функции L(). Разумно использовать, например, 100 узлов для вывода графика.

[3] Подсказка: в лебеговых пространствах под расстоянием подразумевается норма $||g_1 - g_2||$. В пространстве L_∞ нормой является равномерная норма, которую можно определить как $||g(x)||_\infty = \max_{x\in[a;b]} |g(x)|$, предполагая, что функция $g(x)$ рассматривается на интервале $x \in [a;b]$.

[4] Подсказка: отобразите расстояние в логарифмической шкале. В matplotlib используйте для этого semilogy вместо plot.

1.3. Интерполяция кубическими сплайнами

Зачастую вариации аппроксимируемых данных могут быть достаточно велики и нелинейны, в то время как потребность интерполировать данные все еще присутствует. В таком случае логично предположить, что значение аппроксимируемой функции в конкретной точке зависит только от значений функции точек в ее малой окрестности, в то время как влияние удаленных точек пренебрежимо мало (принцип локальности). Один из самых популярных вариантов аппроксимации такого типа является интерполяция кубическими сплайнами. В базовой части этого задания кубические сплайны будут использоваться для аппроксимации изменения ВВП и предсказания его изменения в будущем. В продвинутой части предлагается воспользоваться кубическими сплайнами для интерполяции реальной температуры, измеряемой на метеостанции, в случае различных интервалов измерений.

Задача 1.2 (интерполяция кубическими сплайнами)
Требуется (базовая часть)

1. Разработать функцию qubic_spline_coeff(x_nodes, y_nodes), которая посредством решения матричного уравнения вычисляет коэффициенты естественного кубического сплайна. Для простоты, решение матричного уравнения можно производить с помощью вычисления обратной матрицы с использованием функции numpy.linalg.inv()[5].

2. Написать функции qubic_spline(x, qs_coeff) и d_qubic_spline(x, qs_coeff), которые вычисляют соответственно значение кубического сплайна и его производной в точке x (qs_coeff обозначает матрицу коэффициентов).

3. Используя материалы интернет-ресурса по адресу

https://data.worldbank.org/indicator/NY.GDP.MKTP.CD,

найти данные о динамике ВВП Российской Федерации. Используя естественные граничные условия, требуется определить, в каком году предположительно был наиболее бурный рост ВВП? Предскажите, какой будет ВВП в 2017 и 2018 годах. Какой вывод можно сделать об экстраполяции сплайнами? Применительно к этой задаче объясните, чем был мотивирован выбор кубических сплайнов среди других кусочно-полиномиальных аппроксимаций?

4. Для тех же данных требуется провести интерполяцию Лагранжа, используя уже написанный код. Объясните, в чем в данном случае причина преимущества кусочной интерполяции перед интерполяцией Лагранжа[6]?

Требуется (продвинутая часть)

5. На интернет-ресурсе https://rp5.ru найдите данные о температуре за один произвольный календарный месяц в произвольном городе, первая буква которого совпадает с первой буквой вашей фамилии[7]. Проведите интерполяцию кубическими сплайнами по полученным данным и выведите результат на экран.

---

[5]Однако, следует помнить, что для реальных приложений это недопустимо.

Подсказки. 1. Функция должна возвращать $(N-1) \times 3$-матрицу, где N — количество узлов интерполяции. 2. Матрицы с несколькими диагоналями удобно создавать с помощью функции numpy.diag(a, v), которая возвращает матрицу с массивом a в качестве v-й диагонали (главная диагональ имеет индекс $v = 0$). Таким образом, матрицу с любыми диагоналями можно построить через суммирование результатов вызовов функции numpy.diag(); конкатенация нескольких матриц (или векторов) производится с помощью функции numpy.c_[a, b, c] (конкатенация вдоль колонок) или функции numpy.r_[a, b, c] (конкатенация вдоль строк).

[6]Подсказка: от состояния какого года ВВП в 2008 году зависит больше — 2007 или 1989?

[7]Для выбора города перейдите в раздел сайта “All countries”, где сначала выбирается страна, а затем соответствующий город. Для скачивания метеорологических данных перейдите со страницы

6. Из текущей выборки удалите половину узлов таким образом, что в выборке остаются измерения только в 03:00, 09:00, 15:00, 21:00. Проведите интерполяцию кубическими сплайнами по новой выборке и выведите результаты на экран.

7. Из последней выборки снова удалите половину узлов таким образом, что в выборке остаются измерения только в 03:00, 15:00, и проведите аналогичную интерполяцию кубическими сплайнами.

8. Сравните три полученные аппроксимации эволюции температуры. Оцените, насколько хорошо аппроксимации во втором и третьем случае предсказывают температуру в известные моменты времени, которые были отброшены при фильтрации массива узлов интерполяции[8]. Также оцените, насколько хорошо эти аппроксимации предсказывают температуру, усредненную в течение дня (усредненная температура получается путем вычисления среднего арифметического известных значений температуры в течение каждого из дней).

1.4. Интерполяция Лагранжа и кусочная интерполяция

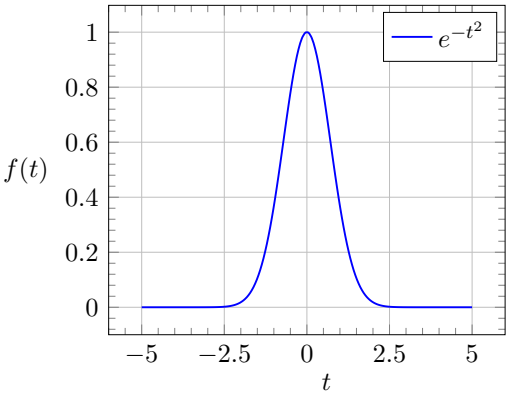
В задании рассматривается влияние количества интерполяционных узлов и их расположения на точность интерполяции полиномами Лагранжа. Первое задание на эту тему было представлено в разделе 1.2. Также предлагается провести исследование зависимости погрешности от того, является ли интерполяция локальной или глобальной.

Задача 1.3 (интерполирование полиномами)
Даны функция

$$f(t) = e^{-t^2}, \quad t \in [-5; 5], \tag{2}$$

и *функция ошибок*[9]

$$\operatorname{erf} x = \frac{2}{\sqrt{\pi}} \int_0^x f(t)dt, \quad x \geq 0. \tag{3}$$



Требуется (базовая часть)

1. Разработать функцию I(i, x, x_nodes), которая возвращает значение базисного полинома Лагранжа $l_i(x)$, заданного на узлах с абсциссами x_nodes, в точке x.

2. Написать функцию L(x, x_nodes, y_nodes), которая возвращает значение интерполяционного полинома Лагранжа, заданного на узлах с абсциссами x_nodes и ординатами y_nodes, в точке x.

3. Для демонстрации работы функции интерполяции L(x, x_nodes, y_nodes) выберите от 5 до 10 произвольных значений некоторой величины и выведите результат работы функции на графике. В качестве интерполируемой величины, на свое усмотрение, можно выбрать любую характеристику доступную в публичных источ-

города в раздел “Weather archive at the weather station”, а затем “Download weather archive”, где вы можете выбрать необходимый временной интервал и формат выгрузки. Обратите внимание, что в выгрузке будут данные, измеренные с интервалом 3 часа.

[8] Для оценки близости удобно пользоваться метриками нормированных пространств – например, равномерной или среднеквадратичной нормой.

[9] Функция ошибок (также называемая *функция ошибок Гаусса*) – неэлементарная функция, возникающая в теории вероятностей, статистике и теории дифференциальных уравнений в частных производных.

никах: статистику температуры воздуха или скорости ветра в городе за несколько дней, курс валют, население страны и пр.

4. Провести следующий анализ.

а) Для равномерно расположенных узлов (случай 1) на отрезке $[-5; 5]$ построить графики $f(x)$ и полученных интерполяционных полиномов $L_{n-1}(x)$ для нескольких различных количеств узлов[10], обозначаемых n. Опишите, что наблюдается при увеличении количества узлов n?

б) Для каждого $n \in [4, 5\,..\,20]$ рассчитайте расстояние между $f(x)$ и $L_{n-1}(x)$ в лебеговом пространстве L_∞[11].

в) Используя формулу для остаточного члена интерполяции[12], аналитически оценить верхнюю границу погрешности интерполяции полиномами Лагранжа $L_{n-1}(x)$ в зависимости от n[13]. Представить в отчёте в графическом виде сравнение полученного результата с зависимостью расстояния между $f(x)$ и $L_{n-1}(x)$ в лебеговом пространстве L_∞ от n[14]. Как соотносятся друг с другом полученные аналитическая и численная оценки погрешности аппроксимации?

5. Повторить пункт 4 для случая оптимально расположенных узлов (случай 2) и для случая кусочно-линейной интерполяции (случай 3, **опциональная часть**: аналитическая оценка верхней границы погрешности в общем виде).

6. Вывести на одном графике зависимости расстояния между $f(x)$ и $L_{n-1}(x)$ в лебеговом пространстве L_∞ от n для всех трёх рассмотренных случаев. Как влияет расположение узлов на погрешность аппроксимации? Какое расположение узлов и для каких n даёт более точную интерполяцию? Как влияет использование локальной или глобальной интерполяции на точность интерполяции?

Требуется (продвинутая часть)

7. Найти приближённое значение функции ошибок $\operatorname{erf} x$ в точке $x = 2$, используя кусочно-линейную интерполяцию для $f(t)$ при $n \in N = \{3, 5, 7, 9\}$ при $t \in [0; 5]$ и сравнить полученные значения между собой.

8. Представить аналитическое выражение для аппроксимации функции $\operatorname{erf} x$, получаемой за счёт использования кусочно-линейной интерполяции $f(t)$ при $n = N_j$,

[10] Рекомендуется использовать разные массивы: для хранения узлов интерполяции с целью построения $L_{n-1}(x)$ и отдельно для построения графиков. Например, массивы интерполяционных узлов (массивы первой группы) размером $n \in [4\,..\,20]$ и, например, массив узлов для вывода графиков с фиксированной длиной 200 (массив второй группы). Т.е. данные об интерполяционных узлах подаются единым массивом на вход функции `L()` как `x_nodes`, в то время как каждый узел из массива второй группы используется как аргумент `x` в функции `L()`.

[11] Напомним, в лебеговых пространствах под расстоянием подразумевается норма $||g_1 - g_2||$. В пространстве L_∞ нормой является равномерная норма.

[12] Теорема об остаточном члене интерполяции для произвольно распределённых узлов, а также частный случай оптимально расположенных узлов описаны в [1]. Случай кусочно-линейной интерполяции разбирается на семинарах.

[13] Допускается численно (приближенно) получить максимальное значение $f^{(n)}(\xi)$, написав соответствующую программную функцию и вычислив значения производной n-го порядка численно в m точках и выбрав максимальное значение (формулу численного дифференцирования выберите самостоятельно, возможно использовать автоматическое дифференцирование). В отчёте о лабораторной работе должен быть представлен и описан алгоритм численного определения максимального значения производной, а также должны быть представлены формулы численного дифференцирования и обоснован их выбор. **Опциональная часть**: Выражение производной $f^{(n)}(\xi) = P_n(\xi)e^{-\xi^2}$, где $P_n(\xi)$ – полином, следует получить аналитически, определив общий вид коэффициентов полинома $P_n(\xi)$.

[14] Расстояние следует отображать в логарифмической шкале. В matplotlib используйте для этого semilogy вместо plot.

где N_j – j-й элемент множества N, $j = M \mod |N|+1$, где M – номер обучающегося по журналу.

9. **Опциональное задание.** Предложить алгоритм оптимизации кода, который бы позволил сократить число арифметических операций при вычислении значения интерполяционного полинома $L_{n-1}(x)$ в точке x. Описать алгоритм в отчёте и обосновать повышение производительности.

1.5. Численное и автоматическое дифференцирование

В лекциях мы рассмотрели два способа нахождения производных: численное и автоматическое дифференцирование. Численное дифференцирование позволяет находить значения производных только с заранее известной погрешностью, и соответствующие формулы характеризуются вычислительной неустойчивостью. Одним из способов улучшения формул численного дифференцирования является экстраполяция Ричардсона, которая позволяет получить аппроксимацию б*о*льшей точности, используя формулу меньшей точности. Экстраполяция Ричардсона активно используются во многих других разделах вычислительной математики, например, в численном интегрировании и численных методах решения задачи Коши, но в рамках базовой части данной лабораторной работы мы продемонстрируем её работоспособность на примере формулы численного дифференцирования первого порядка. Одновременно с подходом численного дифференцирования мощным инструментом является автоматическое дифференцирование, которое позволяет получить точное значение производной (вплоть до машинного эпсилон). Версия автоматического дифференцирования, известная как прямой режим с дуальными числами, реализуется и анализируется в продвинутой части данной лабораторной работы.

Задача 1.4 (численное дифференцирование)
Даны функции

$$f(x) = \frac{x^5 + 2x^4 - 3x^3 + 4x^2 - 5}{x + 2},$$
$$g(x) = x^2 \sin(x),$$

и узел $x_0 = 2$.

Требуется (базовая часть)

1. Вывести улучшенную формулу численного дифференцирования для нахождения значения первой производной функции, используя в качестве основной формулы формулу численного дифференцирования 1-го порядка и применив к ней экстраполяцию Ричардсона. Экстраполяция Ричардсона основывается на том, что аппроксимацию значения производной можно записать в следующем виде:

$$M = N_1(h) + K_1 h + K_2 h^2 + K_3 h^3 + \dots, \tag{4}$$

где M — точное значение производной в заданной точке, $N_1(h)$ — формула численного дифференцирования первого порядка и $K_1, K_2, K_3, \dots$ — константы, появляющиеся в остаточном члене при разложении его в ряд Тейлора. Используя $h/2$ вместо h в выражении выше, выведете новую формулу, которая будет иметь второй порядок точности. Можно ли получить эту формулу другими способами? Если да, то какими?

2. Написать функцию diff1(x_0, h, f), которая возвращает значение первой производной функции f на основе центральной формулы численного дифференцирования 1-го порядка в точке x_0 для шага дифференцирования h[15].

3. Написать функцию diff2(x_0, h, f), которая возвращает значение первой производной функции f на основе новой формулы численного дифференцирования 2-го порядка в точке x_0 для шага дифференцирования h.

4. Рассчитать производную $g'(x)$ в точке x_0 для множества значений $h \in [10^{-16}; 1]$ сначала с помощью функции diff1, а затем с помощью функции diff2[16]. Для обоих случаев постройте log–log графики зависимости абсолютной погрешности численного дифференцирования от шага дифференцирования[17].

Требуется (продвинутая часть)

5. Для случая функций diff1 и diff2 из базовой части ответить на следующие вопросы:

– Каким образом на log–log графике можно увидеть порядок формулы дифференцирования? Докажите это формульно и продемонстрируйте на графике по аналогии с лекциями.

– Каков оптимальный шаг дифференцирования, при котором абсолютная погрешность минимальна? С чем связано существование такого минимума? Обоснуйте свой ответ, ссылаясь на данные log–log графика.

6. Реализовать прямой режим автоматического дифференцирования с использованием дуальных чисел[18].

– Написать класс AutoDiffNumber, использование которого позволяет построить вычислительный граф[19].

– Написать функцию forward_autodiff(fun_args), вычисляющую значение производной функции, вычислительной граф которой передаётся в fun_args.

– Продемонстрировать корректность автоматического дифференцирования на примере функции $f(x)$ и 100 случайных точек $x \in [-1; 1]$ и сравнить полученные значения производных с аналитически полученными значениями.

– Вычислить значения производных в тех же точках, используя функции diff1 и diff2, и сравнить полученные значения с аналитическими и полученными с помощью автоматического дифференцирования.

1.6. Интерполяция в условиях измерений с неопределённостью

Интерполяция, вероятно, является самым простым способом определения недостающих значений некоторой функции при условии, что известны соседние значения. Однако, за кадром зачастую остаётся вопрос о том, насколько точно мы знаем исходные данные для проведения интерполяции или любой другой аппроксимации. К примеру, исходные данные могут быть получены путем снятия показаний с датчиков, которые всегда обладают определенной погрешностью. В этом случае всегда возникает желание оценить влияние подобных погрешностей и неопределенностей

[15]Подсказка: в Python функции можно передавать как обычные параметры метода.

[16]Подсказка: равномерную логарифмическую сетку для h в Python можно построить с помощью функции numpy.logspace().

[17]Подсказка: log–log в matplotlib строятся с помощью функции plt.loglog().

[18]Подсказка: описанная ниже схема является лишь одним из возможных вариантов реализации. Вы безусловно можете использовать свою схему реализации, при учёте того, что соответствующее обоснование будет добавлено в отчёт.

[19]Подсказка: этого легко добиться с помощью перегрузки операторов.

Таблица 1. Значения уровня поверхности вязкой жидкости (Рис. 1)

i	1	2	3	4	5	6	7	8	9	10	11
x_i	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
h_i	3,37	3,95	3,73	3,59	3,15	3,15	3,05	3,86	3,60	3,70	3,02

на аппроксимацию. В этом задании на простейшем примере мы познакомимся с интерполяцией в целом (базовая часть) и проанализируем, как неопределенности влияют на ее предсказания (продвинутая часть).

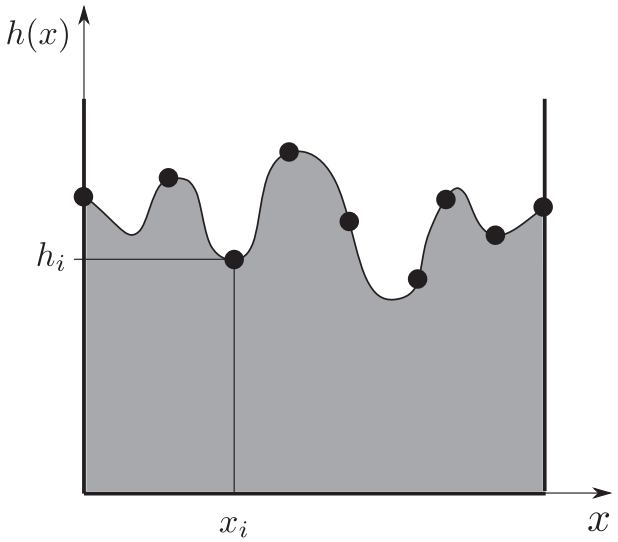


Рис. 1. Поверхность вязкой жидкости, движущейся сквозь некоторую среду (например, пористую). Её значения известны только в нескольких точках

Задача 1.5 (интерполяция кубическими сплайнами)

Требуется (базовая часть)

1. Разработать функцию qubic_spline_coeff(x_nodes, y_nodes), которая посредством решения матричного уравнения вычисляет коэффициенты естественного кубического сплайна. Для простоты, решение разрешающей системы линейных алгебраических уравнений (СЛАУ) можно производить с помощью вычисления обратной матрицы с использованием функции numpy.linalg.inv()[20].

2. Написать функцию qubic_spline(x, qs_coeff), которая вычисляет значение кубического сплайна в точке x (qs_coeff обозначает матрицу коэффициентов).

3. Используя данные в Табл. 1, требуется построить аппроксимацию зависимости уровня поверхности жидкости $h(x)$ от координаты x (Рис. 1) с помощью кубического сплайна и продемонстрировать ее на графике вместе с исходными узлами.

Требуется (продвинутая часть)

[20]Однако следует помнить, что для реальных приложений это недопустимо.

Подсказки. 1. Функция должна возвращать $(N-1) \times 3$-матрицу, где N количество узлов интерполяции. 2. Матрицы с несколькими диагоналями удобно создавать с помощью функции numpy.diag(a, v), которая возвращает матрицу с массивом a в качестве v-й диагонали (главная диагональ имеет индекс $v = 0$). Таким образом, матрицу с любыми диагоналями можно построить через суммирование результатов вызовов функции numpy.diag(). 3. Конкатенация нескольких матриц (или векторов) производится с помощью функции numpy.c_[a, b, c] (конкатенация вдоль колонок) или функции numpy.r_[a, b, c] (конкатенация вдоль строк).

4. Разработать функцию l_i(i, x, x_nodes), которая возвращает значение i-го базисного полинома Лагранжа, заданного на узлах с абсциссами x_nodes, в точке x.

5. Написать функцию L(x, x_nodes, y_nodes), которая возвращает значение интерполяционного полинома Лагранжа, заданного на узлах с абсциссами x_nodes и ординатами y_nodes, в точке x.

6. Известно, что при измерении координаты x_i всегда возникает погрешность, которая моделируется случайной величиной с нормальным распределением с нулевым математическим ожиданием и стандартным отклонением 10^{-2}. Требуется провести следующий анализ[21], позволяющий выявить влияние этой погрешности на интерполяцию.

а) Сгенерировать 1000 векторов значений $[\tilde{x}_1, \ldots, \tilde{x}_{11}]^{\mathrm{T}}$, предполагая, что $\tilde{x}_i = x_i + Z$, где x_i соответствует значению в Табл. 1 и Z является случайной величиной с нормальным распределением с нулевым математическим ожиданием и стандартным отклонением 10^{-2}.

б) Для каждого из полученных векторов построить интерполянт Лагранжа, предполагая, что в качестве абсцисс узлов используются значения $\tilde{x}_i$, а ординат — h_i из Табл. 1. В результате вы должны иметь 1000 различных интерполянтов.

в) Предполагая, что все интерполянты представляют собой равновероятные события, построить такие функции $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$, где $\tilde{h}_l(x) < \tilde{h}_u(x)$ для любого $x \in [0; 1]$, что вероятность того, что значение интерполянта в точке x будет лежать в интервале $[\tilde{h}_l(x); \tilde{h}_u(x)]$ равна 0,9[22].

г) Отобразить на едином графике функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$, усредненный интерполянт и узлы из Табл. 1.

д) Какие участки интерполянта и почему являются наиболее чувствительными к погрешностям?

7. Повторить анализ, описанный в предыдущем пункте, в предположении, что координаты x_i вам известны точно, в то время как измерения уровня поверхности h_i имеют ту же погрешность, что и в предыдущем пункте. Изменились ли выводы вашего анализа?

8. Повторить два предыдущие пункта для случая интерполяции кубическим сплайном. Какие выводы вы можете сделать, сравнив результаты анализа для интерполяции Лагранжа и интерполяции кубическим сплайном?

9. **Опциональное задание.** Изложенный выше анализ позволяет строить доверительные интервалы исключительно для интерполянтов, не оценивая доверительные интервалы с точки зрения предсказаний значений между узлами. Интересным методом интерполяции, позволяющим получить именно такие вероятностные оценки, является регрессия на основе гауссовских процессов, известная также как кригинг. В этом опциональном задании предлагается провести интерполяцию по данным из Табл. 1, используя кригинг[23].

[21] Предложенный подход является типичным примером метода Монте-Карло.

[22] Область между $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$ называется доверительной полосой, по аналогии с доверительным интервалом.

[23] Кригинг реализован, например, в Python пакете scikit-learn.

1.7. Интерполяция параметрическими кубическими сплайнами и автоматическое дифференцирование

Множество Мандельброта (фрактал) — удивительное явление широко известное за пределами соответствующей области математики благодаря бесконечному многообразию форм и ярким визуализациям.

Определение 1
Точки фрактала $\boldsymbol{c} \subset \mathbb{C}$ в пространстве комплексных чисел[24] определяются рекуррентной формулой, задающей последовательность, ограниченную только для точек принадлежащих фракталу:

$$\begin{aligned} &\forall c \in \boldsymbol{c}, \quad \exists Z \in \mathbb{R}\colon |z_i| < Z, \quad i \in \mathbb{N}, \\ &z_i = z_{i-1}^2 + c, \\ &z_0 = 0. \end{aligned} \tag{5}$$

Другими словами, для точек фрактала, независимо от длины последовательности при $i \to \infty$, $|z_i|$ не превысит некоторого заранее заданного действительного числа. Для вычислительного эксперимента следует положить $Z = 1000$, $i \in [0 : 255]$.

В данной работе потребуется интерполировать некоторый фрагмент границы фрактала $\boldsymbol{c}$, описываемой в параметрическом виде неизвестными функциями $x(t)$, $y(t)$, на основе его отдельных точек, используя параметрически задаваемый кубический сплайн.

Задача 1.6 (Интерполяция параметрическими кубическими сплайнами и автоматическое дифференцирование)
Требуется (базовая часть)

1. Используя заранее подготовленный скрипт 1, выбрать произвольную область множества Мандельброта и построить фрагмент его границы (контура), сформировав файл contours.txt. Использование неуникального файла contours.txt считается списыванием, равно как и использование чужого кода.

Файл contours.txt содержит упорядоченную последовательность точек на плоскости $P = \{(x_i, y_i)\}_{i=1}^{N}$, принадлежащих выбранному фрагменту границы фрактала $\boldsymbol{c}$. Сопоставляя каждой паре координат естественную координату t, предполагать, что $x_i = x(t_i)$, $y_i = y(t_i)$. Выбранный контур должен содержать по меньшей мере 100 точек (100 строк в файле contours.txt).

Пример работы python-скрипта[25] приведен на рис. 2.

2. Разработать код для загрузки и визуализации множества точек P из файла contours.txt[26].

3. Задать разреженное множество интерполяционных узлов $\hat{P} = \{(x_j, y_j)\}_{j=1}^{\widehat{N}}$, $\widehat{N} = \lfloor N/M \rfloor$, $j = M \times i$, $\hat{P} \subset P$. Положить $M = 10$.

4. По каждому измерению найти коэффициенты естественного параметрического кубического сплайна a_{jk} и b_{jk}, путём решения соответствующих разрешающих систем линейных алгебраических уравнений (СЛАУ) [1][27], в результате должен

[24] Напомним, что комплексное число имеет вид $c = x + iy$ ($x, y \in \mathbb{R}$, $i^2 = -1$).

[25] Для работы скрипта может потребоваться установить дополнительные пакеты. Рекомендуемая команда для установки: pip install numpy opencv-python matplotlib PyQt5

[26] Рекомендуется использовать функции numpy.loadtxt и matplotlib.pyplot.scatter соответственно. Рабочий код лабораторной работы рекомендуется располагать в файле lab1.py.

[27] Решение СЛАУ следует осуществлять с помощью, например, метода Гаусса. Возможно использовать numpy.linalg.inv (ресурсозатратно).

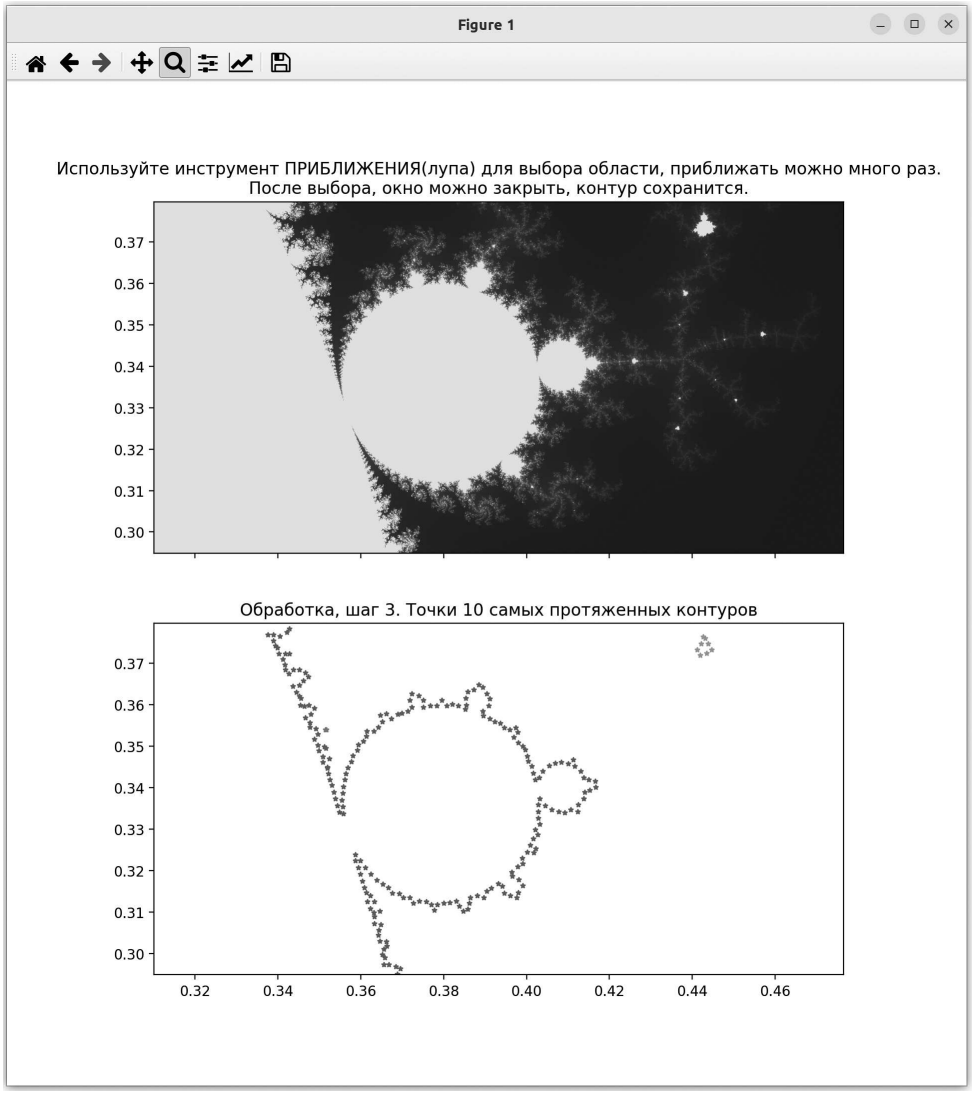


Рис. 2. Сверху — отображение точек фрактала (светлый) на комплексной плоскости, граничные и пр. точки в виде оттенков серого. Снизу — отдельные точки, формирующие приблизительные наиболее длинные непрерывные контуры множества

получится сплайн вида:

$$\begin{aligned}
\tilde{x}(t) &= \sum_{j=1}^{\widehat{N}-1} I_j(t)(a_{j0} + a_{j1}(t-t_j) + a_{j2}(t-t_j)^2 + a_{j3}(t-t_j)^3), \\
\tilde{y}(t) &= \sum_{j=1}^{\widehat{N}-1} I_j(t)\left(b_{j0} + b_{j1}(t-t_j) + b_{j2}(t-t_j)^2 + b_{j3}(t-t_j)^3\right), \\
I_j(t) &= \begin{cases} 1, & t \in [t_j, t_{j+1}) \\ 0, & \text{в противном случае} \end{cases}
\end{aligned} \tag{6}$$

где $I_j(t)$ — индикаторная функция принадлежности интервалу[28].

5. Вычислить расстояния $\rho\left[(\tilde{x}(t_i), \tilde{y}(t_i)), (x(t_i), y(t_i))\right]$ и представить вывод (среднее и стандартное отклонение) в отчёте.

[28]В рамках реализации может не использоваться.

6. Отобразить в отчёте полученный сплайн используя $t \in [0, t_N]$ с частым шагом $h = 0.1$ совместно с исходным множеством точек P, а также узловыми точками $\hat{P}$. С чем связана наблюдаемая ошибка интерполяции? Как её можно уменьшить? Вывод следует привести в отчёте.

7. В результате выполнения базовой части задания, помимо прочих, должна быть разработана функция lab1_base(filename_in:str, factor:int, filename_out:str), где filename_in — входной файл contours.txt, factor — значение параметра M, filename_out — имя файла результата (как правило coeffs.txt), содержащего коэффициенты a_{jk} и b_{jk} в виде матрицы размером $\widehat{N} - 1$ строк на 8 столбцов[29]. Функция lab1_base должна реализовывать базовую часть задания.

Требуется (продвинутая часть)

8. Используя концепцию дуальных чисел $v = a + \varepsilon b$, $\varepsilon^2 = 0$, и перегрузку операторов сложения и умножения в Python, необходимо реализовать класс AutoDiffNum, для автоматического вычисления производной некоторой функции.

9. Реализовать функцию автоматического расчёта первой производной кубического сплайна $G(t) = \frac{d}{dt}\left(\tilde{x}(t), \tilde{y}(t)\right)$.

10. Реализовать функцию построения нормали $R(t_j)$ к заданному вектору $G(t_j)$.

11. Построить векторы $G(t_j)$ и $R(t_j)$ в соответствующих точках сплайна, выбрав наглядную частоту прореживания (не менее 5 точек на контур) и масштаб.

12. **Опциональное задание.** Для каждой пропущенной точки (x_i, y_i) исходного контура найти (численно) ближайшую на соответствующем участке сплайна. Фактически нужно решить оптимизационную задачу:

$$t^* = \underset{t \in [0, t_N]}{\operatorname{argmin}} \sqrt{\left(\tilde{x}(t) - x_i\right)^2 + \left(\tilde{y}(t) - y_i\right)^2}. \tag{7}$$

К примеру, используя простейший метод деления отрезка пополам (дихотомии) до 10 итераций. В отчёте необходимо отобразить на графике соответствующие точки, выбрав наглядную частоту прореживания и привести среднюю оценку погрешности. Сравнить результаты с вычисленными ранее значениями

$$\rho\left[\left(\tilde{x}(t_i), \tilde{y}(t_i)\right), \left(x(t_i), y(t_i)\right)\right].$$

Все полученные в работе изображения, включаемые в отчёт, должны быть сохранены в векторном формате (PDF[30] или EPS и размещены рядом с исходным кодом разработанной программы).

[29] Сначала блок коэффициентов a_{jk}, а далее блок коэффициентов b_{jk}.

[30] matplotlib поддерживает pdf-формат для сохранения

Лабораторная работа 2

2.1. Требования к знаниям для выполнения

Для выполнения лабораторной работы обучающийся должен обладать знаниями:

– владеть навыками разработки программного обеспечения на языке Python (рекомендуется) или C++ на базовом уровне;

– владеть навыками использования Python библиотеки `numpy`;

– знать понятия: численное дифференцирование, численное интегрирование, метод наименьших квадратов, тригонометрические полиномы, быстрое преобразование Фурье.

2.2. Численное дифференцирование и интегрирование

Известно, что в случае достаточной гладкости функции, порядок точности выбранной формулы численного дифференцирования определяется степенью, в которую возводится шаг дифференцирования h в выражении для остаточного члена $R(h)$ этой формулы, для которого $R(h) \sim \mathcal{O}(h^N)$.

Например, если $f(x) \in C^4[a;b]$ и $x_1 - h, x_1, x_1 + h \in [a;b]$, тогда можно получить следующую формулу численного дифференцирования второго порядка для второй производной функции $f(x)$ в точке x_1:

$$f''(x_1) = \frac{f(x_1 - h) - 2f(x_1) + f(x_1 + h)}{h^2} - \underbrace{\frac{h^2}{12} f^{(4)}(\xi)}_{=R(h)}, \quad \xi \in (x_1 - h; x_1 + h).$$

Порядок формулы также легко продемонстрировать, если построить график зависимости абсолютной погрешности численного дифференцирования в некоторой заданной точке от шага дифференцирования h в логарифмическом масштабе по обеим осям (т.н. *log–log график*). В базовой части лабораторной работы предлагается построить подобный график для формулы численного дифференцирования второго порядка точности. Формулы численного дифференцирования более высокого порядка часто выводят с помощью разложения в ряд Тейлора и последующего вычисления значений ряда в нескольких узлах.

При выводе формулы численного дифференцирования для определения m-й производной $f^{(m)}(x)$ увеличение количества узлов приводит к увеличению порядка точности формулы, а также к увеличению количества арифметических операций с плавающей запятой, что может увеличивать вычислительную погрешность. В продвинутой части задании предлагается вывести формулу численного дифференцирования 4-го порядка и проанализировать её вычислительную устойчивость.

Аналогичный анализ можно произвести и для формул численного интегрирования или *квадратурных формул*. В базовой части лабораторной работы предлагается исследовать составную формулу Симпсона, в то время как в продвинутой части необходимо вывести и исследовать квадратурную формулу Гаусса со степенью точности 5.

Задача 2.1 (численное дифференцирование и интегрирование)
Даны функции:

$$g_1(x) = xe^x, \tag{8}$$
$$g_2(x) = x^2 \sin 3x, \quad x \in [0; \pi]; \tag{9}$$
$$g_3(x) = \sin \frac{\pi}{x}, \quad x \in (0; 1]. \tag{10}$$

Требуется (базовая часть)

1. Написать функцию diff2(x_0, h, f), которая возвращает значение первой производной функции f на основе центральной формулы численного дифференцирования 2-го порядка в точке x_0 для шага дифференцирования h[31].

2. Рассчитать производные $g_1'(x)$ в точке $x_0 = 3$ и $g_3'(x)$ в точке $x_0 = 0.01$ для множества значений $h \in [10^{-16}; 1]$ с помощью функции diff2[32]. Постройте log–log графики зависимости абсолютной погрешности численного дифференцирования от шага дифференцирования для двух указанных случаев[33].

3. Ответьте на вопросы согласно п. 10.

4. Написать функцию composite_simpson(a, b, n, f) численного интегрирования функции f на отрезке [a; b] по n узлам с помощью составной формулы Симпсона.

5. Рассчитать интегралы $\int_0^\pi g_2(x)dx$ и $\int_\varepsilon^1 g_3(x)dx$, где $0 < \varepsilon < 0.01$, с помощью составной формулы Симпсона для множества значений $n \in [3; 9999]$. Постройте log–log графики зависимостей абсолютной погрешности численного интегрирования от шага интегрирования.

6. Сравните порядок точности составной формулы Симпсона, полученный "с помощью log–log графика", с аналитическим порядком точности этой формулы. Существует ли оптимальный шаг интегрирования для данной формулы, минимизирующий достижимую погрешность? Обоснуйте свой ответ.

7. Вывести центральную формулу численного дифференцирования 4-го порядка точности вместе с остаточным членом, аппроксимирующую первую производную $f'(x)$ по значениям $f(x)$ в 5-и узлах[34]:

$$f'(x_0) \approx Af(x_0 - 2h) + Bf(x_0 - h) + Cf(x_0) + Df(x_0 + h) + Ef(x_0 + 2h). \tag{11}$$

Продемонстрируйте, что формула действительно имеет 4-й порядок точности.

Требуется (продвинутая часть)

8. Написать функцию diff4(x_0, h, f), которая возвращает значение первой производной функции f на основе центральной формулы численного дифференцирования 4-го порядка в точке x_0 для шага дифференцирования h.

9. Рассчитать производные $g_1'(x)$ в точке $x_0 = 3$ и $g_3'(x)$ в точке $x_0 = 0.01$ для множества значений $h \in [10^{-16}; 1]$ с помощью функции diff4[35]. Добавьте log–log график зависимости абсолютной погрешности численного дифференцирования от шага дифференцирования к соответствующему графику для diff2[36].

[31] Подсказка: в Python функции можно передавать как обычные параметры метода.

[32] Подсказка: равномерную логарифмическую сетку для h в Python можно построить с помощью функции numpy.logspace().

[33] Подсказка: log–log в matplotlib строятся с помощью функции plt.loglog().

[34] Обучающимся с **чётным** номером по журналу вывод следует осуществлять с помощью разложения функции $f(x)$ в ряд Тейлора, а с **нечетным** – используя представление $f(x)$ в виде полинома Лагранжа с остаточный членом.

[35] Подсказка: равномерную логарифмическую сетку для h в Python можно построить с помощью функции numpy.logspace().

[36] При выполнении продвинутой части вывод отдельных log–log графиков согласно п.2 **осуществлять не нужно**.

10. Ответьте на следующие вопросы, для случая применения функции diff2 (**базовая часть**) и для случая применения функции diff4 (**продвинутая часть**).

а) Каким образом на log–log графике можно увидеть порядок точность формулы дифференцирования? Представьте аналитическое доказательство, а также продемонстрируйте порядок точности на графике.

б) Совпадает ли порядок точности выведенной формулы численного дифференцирования на log–log графике с её фактическим порядком точности?

в) Каков оптимальный шаг дифференцирования, при котором абсолютная погрешность минимальна? С чем связано существование такого минимума? Обоснуйте свой ответ, ссылаясь на данные log–log графика.

г) Сравните оптимальный шаг дифференцирования и соответствующую минимально достижимую погрешность для формул 2-го и 4-го порядка точности. Как вы думаете, чем обоснована разница между ними?

11. С помощью теоремы о корнях многочленов Лежандра (см. [1]), вывести квадратурную формулу Гаусса (далее *квадратура Гаусса*), имеющую степень точности 5. Сколько узлов необходимо для использования такой формулы?

12. Написать функцию gauss_quad5(f) численного интегрирования функции f с помощью квадратуры Гаусса пятой степени точности.

13. Доказать, что квадратура Гаусса имеет степень точности 5, с помощью следующего вычислительного эксперимента[37]:

– постройте последовательность полиномов $P_0(x)$, $P_1(x)$, $P_2(x)$, $P_3(x)$, $P_4(x)$, $P_5(x)$, $P_6(x)$, имеющих степени соответственно 0, 1, 2, 3, 4, 5, и 6, используя случайно сгенерированные значения коэффициентов полиномов[38];

– проинтегрируйте их на интервале $[0; 2]$ аналитически и с помощью функции gauss_quad5(f);

– посчитайте абсолютную погрешность и сделайте вывод о степени точности выведенной квадратуры.

2.3. Численное интегрирование с помощью тригонометрических интерполянтов

В классическом курсе вычислительной математики квадратуры численного интегрирования строятся путем аналитического интегрирования многочленов, интерполирующих подынтегральное выражение. Новый класс квадратур можно получить путем интерполяции подынтегральной функции тригонометрическими полиномами, т.е. путем ее замены на тригонометрический интерполянт. Мотивация в построении подобных квадратур лежит в основном преимуществе интерполяции тригонометрическими полиномами — свойстве спектральной сходимости, то есть сходимости быстрее, чем $O(h^N)$ для любого N. Безусловно, такой подход оправдан только к том случае, когда подынтегральная функция является периодической на данном интервале интегрирования, так как в противном случае образует разрыв на границах интегрирования. В этом задании мы разработаем алгоритм подобного численного интегрирования и, используя быстрое преобразование Фурье для нахождения коэффициентов тригонометрического интерполянта, проведем анализ сходимости квадратур в зависимости от гладкости подынтегрального выражения.

[37] Выкладки и рассчитанные значения должны быть представлены в отчёте.

[38] Случайные числа в Python генерируются функцией numpy.random.randn().

Задача 2.2 (БПФ)
Даны интегралы

$$I_1 = \int\limits_{-\frac{\pi}{4}}^{\pi} x^3 dx, \quad I_2 = \int\limits_{-\frac{\pi}{4}}^{\pi} |x| dx, \quad I_3 = \int\limits_{-\frac{\pi}{4}}^{\pi} \frac{dx}{1 + \sin^2\left(\frac{x}{2}\right)}, \quad I_4 = \int\limits_{-\frac{\pi}{4}}^{\pi} \sin(8x) dx. \tag{12}$$

Требуется

1. Вывести общее выражение для формулы численного интегрирования путем аналитического интегрирования тригонометрического ряда, заменяющего подынтегральную функцию[39].

2. Используя алгоритм Кули–Тьюки, написать функцию fft_coeff(y_nodes), которая вычисляет и возвращает комплексные коэффициенты тригонометрического полинома, интерполирующего узлы y_nodes, равномерно распределенные на отрезке $[-\pi; \pi]$.

3. Написать функцию spectral_integral(f, N), которая вычисляет значение интеграла функции f, интерполируемой в N узлах с помощью тригонометрического ряда, на интервале $[-\frac{\pi}{4}; \pi]$. Функция spectral_integral должна использовать внутри себя функцию fft_coeff.

4. Для каждого из интегралов I_1, I_2, I_3 и I_4 провести следующий анализ.

– Найти точное значение интеграла.

– Найти приближенное значение интеграла с помощью функции spectral_integral для $N = 2^{\tilde{n}}$, где $\tilde{n} \in 1, \ldots, 8$.

– Для каждого N найти относительную погрешность вычислений δ и вывести на экран график зависимости δ от N. Необходимо использовать логарифмические шкалы на оси ординат и/или абсцисс для наилучшей демонстрации зависимости[40].

5. Объяснить, как можно использовать полученные логарифмические графики для оценки формы зависимости погрешности интегрирования от числа узлов.

6. Ответить, различаются ли зависимости погрешности интегрирования от числа узлов в случаях вычисления интегралов I_1, I_2, I_3 и I_4 и, если различаются, объяснить, с чем это связано.

---

[39] Следует отметить, что в этом случае квадратура будет являться функцией коэффициентов тригонометрического ряда. Действительно, пусть $\int\limits_{-\pi/4}^{\pi} f(x)dx$ — рассматриваемый интеграл. Тогда при замене $f(x)$ тригонометрическим интерполянтом вида $f(x) \approx \sum\limits_{k=-n}^{n} \hat{a}_k e^{ikx}$ интеграл будет зависит только от коэффициентов $\hat{a}_k$, которые по определению тригонометрического интерполянта зависят от значений функции $f(x)$.

При интегрировании тригонометрического интерполянта в экспоненциальной форме появляются комплексные значения, в то время как значение интеграла должно быть вещественным числом. Из этого затруднения можно выйти двумя способами. Во-первых, вместо экспоненциальной формы ряда можно воспользоваться тригонометрической формой, выраженной через sin и cos, что приводит к интегрированию вещественнозначной функции. Во-вторых, после интегрирования ряда в экспоненциальной форме можно наложить определенные условия на коэффициенты тригонометрического ряда, гарантирующие, что значение интеграла является вещественным числом (аналогичная процедура выполняется в лекциях при выводе тригонометрического интерполянта в экспоненциальной форме). Если подобные условия не нужны и значение интеграла автоматически получается вещественным, это необходимо продемонстрировать в выводе. Конкретный выбор способа остается за исполнителем лабораторной работы.

[40] Подсказка: в matplotlib логарифмическая шкала на оси ординат задается с помощью функции plt.semilog(). Логарифмическая шкала на обеих осях задается с помощью функции plt.loglog().

2.4. Полиномиальная регрессия (вариант 4)

Полиномиальная регрессия является простейшим случаем нелинейной регрессии и может использоваться для вычисления кривых, приближающихся к исходным данным. Часто за рамками обсуждения регрессии в курсе вычислительной математики остается важная статистическая гипотеза о данных, которая делается неявно при задании степени многочлена. Эта гипотеза гласит, что исходные данные действительно описываются многочленом некоторой степени с неизвестными коэффициентами с добавлением шума в виде случайной величины с нормальным распределением. Так как *a priori* степень искомого многочлена неизвестна, ее необходимо каким-то образом выбрать. Из гипотезы ясно, что при выборе слишком маленькой степени многочлена теряется много полезной информации, в то время как при выборе слишком большой степени кривая начинает описывать случайный шум. Эта проблема известна в математической статистике как переобучение. В этом задании с помощью вычислительного эксперимента мы исследуем взаимовлияния между количеством данных, используемых для регрессии, степенью искомого полинома и амплитудой шума.

Задача 2.3 (Регрессия)
Дана функция

$$f(x) = -10x^2 + 1{,}5x + 1 + \sigma X, \tag{13}$$

где $x \in [-1; 1]$ и X — случайная величина, нормально распределенная на интервале $[-1; 1]$.

Требуется

1. Написать функцию poly_regression(x_nodes, y_nodes, degree), которая возвращает коэффициенты многочлена степени degree, наилучшим образом приближающегося к точкам с абсциссами x_nodes и ординатами y_nodes[41].

2. Для каждого N из множества $\{2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$ с помощью функции $f(x)$ сгенерировать начальный набор данных $D_{regr}^{(N)}$ и проверочный набор данных $D_{test}^{(N)}$, где N обозначает число точек в соответствующем наборе данных [42]. Затем для каждого N, каждого σ из множества $\{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$ и каждого p из множества $\{1, 2, 4, 8, 12, 16, 20\}$ провести следующий анализ.

– С помощью набора данных $D_{regr}^{(N)}$ и функции poly_regression построить многочлен степени p, наилучшим образом приближающийся к данным.

– Вычислить среднеквадратичную погрешность аппроксимации $\varepsilon_{regr}^{(N,p)}$ данных $D_{regr}^{(N)}$ полученным многочленом.

– Вычислить среднеквадратичную погрешность аппроксимации $\varepsilon_{test}^{(N,p)}$ данных $D_{test}^{(N)}$ полученным многочленом.

3. Вывести на экран несколько характерных примеров графиков многочленов вместе с начальными и проверочными данными.

4. Ответить на следующие вопросы.

– Как влияет увеличение числа начальных и проверочных данных на $\varepsilon_{regr}^{(N,p)}$ и $\varepsilon_{test}^{(N,p)}$? Выведите на экран графики, иллюстрирующие ответ, и сделайте вывод.

– Как влияет увеличение степени многочлена p на $\varepsilon_{regr}^{(N,p)}$ и $\varepsilon_{test}^{(N,p)}$? Выведите на экран графики, иллюстрирующие ответ, и сделайте вывод.

[41] Под приближением здесь понимается приближение в среднеквадратичном смысле.

[42] Подсказка: в первую очередь следует сгенерировать N случайных x и X, что реализуется с помощью функций numpy.random.rand() и numpy.random.randn() для равномерно распределенной случайной величины x и нормально распределенной случайной величины X соответственно.

– Как влияет увеличение числа начальных и проверочных данных на относительную погрешность коэффициентов многочлена при сравнении с $f(x)$ в случае фиксированного σ? Что происходит при увеличении σ? Выведите на экран графики, иллюстрирующие ответ, и сделайте вывод. Как может повлиять изменение функции распределения случайной величины X на сделанный вывод?

5. Исходя из ответов на предыдущие вопросы, сделать общий вывод о свойствах сходимости полиномиальной регрессии в случае зашумленных данных.

2.5. Нелинейная регрессия

Задача нелинейной регрессии естественным образом встает при попытке оценить значения параметров некоторой нелинейной функции известной формы по имеющимся дискретным данным. Подобная нелинейная функция может, к примеру, быть решением модельной системы уравнений, оценивающей значение некоторой величины во времени. В таком случае нелинейная регрессия может быть использована для построения предсказаний изменения этой величины в будущем. Характерным примером являются модели эпидемий, простейшими примерами которых являются модель экспоненциального роста и SIS-модель (“Susceptible–Infected–Susceptible model”). Обе модели имеют нелинейные аналитические решения и могут быть использованы для предсказания динамики распространения эпидемии. Модель экспоненциального роста обычно используется для анализа распространения эпидемии в первые ее недели, в то время как SIS-модель помогает охарактеризовать долгосрочную динамику эпидемии. В этом задании мы воспользуемся нелинейной регрессией, чтобы охарактеризовать и, по возможности, предсказать распространение коронавирусной инфекции COVID-19 в различных странах по имеющимся в открытых источниках данным.

Задача 2.4 (Регрессия)
Дана модель экспоненциального роста:

$$\frac{d}{dt}I = (\beta - \gamma)I, \tag{14}$$

где I — количество инфицированных людей, β — среднее число контактов, приходящихся на человека в единицу времени, и γ — это среднее число выздоровевших, приходящихся на человека в единицу времени (т.е., средняя скорость иммунизации).

Также дана SIS-модель:

$$\frac{d}{dt}I = (\beta - \gamma)I - \frac{\beta}{N}I^2, \tag{15}$$

где N — число людей в популяции (например, число жителей страны).

Требуется

1. Найти решение модели экспоненциального роста и SIS-модели при условии, что $I(0) = I_0$ и продемонстрировать детальный вывод этого решения[43]. После этого приведите решение модели экспоненциального роста к форме

$$I(t) = I_0 e^{\chi t}, \tag{16}$$

[43]Замена $x = 1/I$ может помочь найти решение SIS-модели.

Таблица 2. Соответствие стран первым буквам фамилий студентов (к задаче 2.4)

А	Австрия	З, И	Италия	П	Португалия	Х	США
Б	Бельгия	К	Канада	Р	Россия	Ц/Ю	Южная Корея
В	Великобритания	Л	Россия	С	США	Ч	Чили
Г	Германия	М	Мексика	Т	Турция	Ш/Щ	Швейцария
Д	Дания	Н	Нидерланды	У	США	Э	Эквадор
Е/Ё/Ж	Великобритания	О	Россия	Ф	Франция	Я	Япония

где $\chi = \beta - \gamma$, и решение SIS-модели к форме

$$I(t) = \frac{I_\infty}{1 + \left(\frac{I_\infty}{I_0} - 1\right) e^{-\chi t}}, \tag{17}$$

где I_∞ обозначает предельное значение $I(t)$ при $t \to \infty$, т.е. $I_\infty = \lim\limits_{t\to\infty} I(t)$.

2. Сформировать выборку данных зависимости числа инфицированных от времени в данной стране. Страна выбирается исходя из первой буквы вашей фамилии (соответствия первых букв фамилий странам указаны в табл. 2). Актуальные данные в формате CSV можно найти на сайте https://ourworldindata.org/coronavirus-source-data[44]

3. Предполагая, что в первые недели[45]. распространения вируса в выбранной стране рост числа зараженных описывается экспоненциальной моделью, требуется оценить значение коэффициента $\chi = \beta - \gamma$ с помощью нелинейной регрессии и нормального уравнения, взяв в качестве аппроксимирующей функции решение экспоненциальной модели[46] и начального числа инфицированных $I_0 = 20 \ldots 30$[47] и вывести на экран в логарифмической шкале полученное решение вместе с исходными дискретными данными о числе инфицированных. Требуется подробно описать формулировку задачи регрессии в вашем случае и явно указать выражения для отдельных векторов и матрицы, входящих в нормальное уравнение.

4. Подставив найденное значение коэффициента $\chi = \beta - \gamma$ в решение SIS-модели, оценить значение коэффициента I_∞ с помощью нелинейной регрессии, взяв в качестве аппроксимирующей функции решение SIS-модели[48] и вывести на экран полученное решение вместе с исходными дискретными данными о числе инфицированных. Требуется подробно описать формулировку задачи регрессии в вашем случае.

5. Ответить на следующие вопросы.

[44] На момент написания этого задачи csv-таблица числа зараженных находилась по адресу: https://covid.ourworldindata.org/data/ecdc/total_cases.csv. Эта таблица регулярно обновляется.

[45] Предположим, что экспоненциальная модель верна для первых 10–20 дней, где «самый первый день» соответствует тому дню, когда наблюдается I_0 зараженных. Выбор конкретного числа дней для анализа экспоненциальной модели зависит от конкретной страны и может быть определен путем вывода на экран данных о числе зараженных в логарифмической шкале.

[46] Обратите внимание, что несмотря на то, что коэффициент χ входит в решение экспоненциальной модели нелинейно, мы всегда можем исправить эту ситуацию, взяв логарифм от $I(t)$.

[47] Выбор конкретного числа инфицированных, от которых стоит производить отсчет, остается за вами.

[48] Обратите внимание, что решение SIS-модели является нелинейным и мы не можем без потери качества аппроксимации выразить коэффициент I_∞ так, что он входит линейно в формулу для решения. Поэтому в данном случае требуется только сформулировать метод наименьших квадратов, а затем найти оптимальное значение I_∞ с помощью минимизации суммы квадратов отклонений, используя, например, scipy.optimize.minimize.

а) Какова погрешность полученной аппроксимации (решения SIS-модели) относительно нормы L_∞? Относительно нормы L_2[49]?

б) Какой критерий вы бы использовали для определения максимального числа дней, в течение которых использование модели экспоненциального роста оправдано[50]? Каково такое максимальное число дней в вашем случае в соответствии с вашим критерием?

в) Предскажите, сколько человек в соответствии с полученной аппроксимацией (решения SIS-модели) будут инфицированы в выбранной стране при $t \to \infty$ и через сколько дней после начала эпидемии наступит ее окончание[51].

2.6. Регрессия и анализ временных рядов

Классическая задача регрессии заключается в предсказании значений данного временного ряда. Для этого в первую очередь необходимо построить кривую, оптимально приближающуюся к исходным данным. Выбор формы кривой является важной статистической гипотезой о данных — соответствующие подходы к анализу таких гипотез изучаются в математической статистике. С точки вычислительной математики нам необходимо решить задачу оптимизации, в контексте нашего курса полученную с помощью метода наименьших квадратов. Контроль сложности модели при этом организуется с помощью регуляризации. В данной лабораторной работе предлагается исследовать конкретный временной ряд, известный как график Килинга и демонстрирующий изменение концентрации атмосферного углекислого газа на основе измерений в обсерватории Мауна-Лоа. Именно работы Килинга привлекли общественное внимание к этой проблеме.

Задача 2.5 (Регрессия)
Требуется (базовая часть)

1. Написать функцию poly_regression(x_nodes, y_nodes, degree, l), которая возвращает коэффициенты многочлена степени degree, наилучшим образом приближающегося[52] к точкам с абсциссами x_nodes и ординатами y_nodes, используя L_2-регуляризацию со значением гиперпараметра l. Коэффициенты должны вычисляться с помощью подходящего нормального уравнения.

2. Сформировать выборку данных зависимости усредненной месячной концентрации атмосферного углекислого газа от времени на основе измерений в обсерватории Мауна-Лоа. Данные можно найти на сайте NOAA: https://www.esrl.noaa.gov/gmd/aftp/data/trace_gases/co2/flask/surface/co2_mlo_surface-flask_1_ccgg_month.txt, или на сайте Scripps CO_2 Program: https://scrippsco2.ucsd.edu/assets/data/atmospheric/stations/in_situ_co2/weekly/weekly_in_situ_co2_mlo.csv. Полученный набор данных мы будем обозначать D. Из данных следует исключить начальный период, где значения концентрации заметно ниже тренда.

3. Разбить набор данных D на два набора одинакового размера: D_{train}, который будет использоваться для решения нормального уравнения, и D_{valid}, который бу-

[49] Так как исходные данные являются дискретными точками, в обоих случаях можно рассматривать нормы конечномерного нормированного пространства, которые были пройдены на первой лекций. Абсолютная и относительная погрешности вычисляются аналогично формулам для скалярных величин, где модуль заменяется на соответствующую норму.

[50] Существует множество вариантов ответа на данный вопрос.

[51] Предположим, что эпидемия заканчивается тогда, когда прирост общего числа заболевших за один день становится меньше 1% от общего числа заболевших за предыдущий день.

[52] Под приближением здесь понимается приближение в среднеквадратичном смысле.

дет использоваться для поиска оптимального значения гиперпараметра. Разбиение следует произвести случайным образом.

4. Для каждого p из множества $\{1, 2, 3, 4, 5, 10, 20\}$ и случая отсутствия L_2-регуляризации провести следующий анализ:

– С помощью набора данных D_{train} и функции poly_regression построить многочлен степени p, наилучшим образом приближающийся к данным.

– Вычислить среднеквадратичные погрешности $\varepsilon^{(p)}_{train}$ и $\varepsilon^{(p)}_{valid}$ аппроксимации полученным многочленом наборов данных D_{train} и D_{valid} соответственно.

5. Вывести на отдельных графиках полученные многочлены вместе с данными, использованными для решения нормального уравнения, и добавить таблицу полученных среднеквадратичных погрешностей. Сделать вывод о наиболее подходящей степени p, используя зависимости $\varepsilon^{(p)}_{train}$ и $\varepsilon^{(p)}_{valid}$ от p.

6. Для $p = 20$ найти оптимальные значения коэффициентов полинома, используя L_2-регуляризацию и график зависимости $\varepsilon^{(20)}_{valid}$ от значений гиперпараметра. Сделать вывод об использовании L_2-регуляризации.

Требуется (продвинутая часть)

1. Привести временной ряд, определенный данными D, к стационарному виду, исключив из него тренд[53]. Продемонстрировать стационарный временной ряд на графике.

2. Используя алгоритм Кули–Тьюки, написать функцию fft_coeff(y_nodes), которая вычисляет и возвращает комплексные коэффициенты тригонометрического полинома, интерполирующего узлы y_nodes.

3. Определить периоды сезонности данного временного ряда, идентифицировав его характерные частоты с помощью функции fft_coeff(y_nodes) и выведя связанные с ними амплитуды коэффициентов тригонометрического ряда на графике.

2.7. Использование аппроксимаций для численной оптимизации

Методы аппроксимации, такие как интерполяция и численное интегрирование, часто используются как составные блоки других, более сложных численных методов. В данной лабораторной работе мы рассмотрим одну из старейших задач вариационного исчисления: задачу о брахистохроне, т.е. задачу о кривой наискорейшего спуска. Она состоит в нахождении такой кривой, по которой материальная точка из точки $(x, y) = (0, 0)$ достигнет точки $(x, y) = (a, y_a)$ под действием силы тяжести за наименьшее время (здесь и далее ось y направлена вниз). Решением этой задачи является такая кривая $y(x)$, которая минимизирует следующий функционал, являющийся полным временем движения материальной точки:

$$\mathcal{F}[y] = \int_0^a \sqrt{\frac{1 + \big(y'(x)\big)^2}{2gy(x)}}\,dx, \tag{18}$$

где g обозначает ускорение свободного падения, и $y'(x) = dy/dx$.

[53] Это можно сделать одним из двух способов. С одной стороны, вы можете буквально вычесть из временного ряда тренд, который вы выделили в предыдущих пунктах задания, получив таким образом временной ряд флуктуаций вокруг тренда. С другой стороны, вы можете перейти к стационарному временному ряду через взятие разностей, т.е. переходу от ряда $x(t)$ к ряду $x'(t) = x(t) - x(t-1)$. Последнее легко сделать в numpy с помощью функции numpy.diff.

Представленная задача имеет аналитическое решение, которым является параметрически заданная циклоида:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = C \begin{bmatrix} t - \frac{1}{2}\sin(2t) \\ \frac{1}{2} - \frac{1}{2}\cos(2t) \end{bmatrix}, \tag{19}$$

где $t \in [0; T]$ и C, T являются константами, значения которых находятся из граничного условия.

В базовой части задания требуется воспользоваться численным интегрированием для нахождения полного времени движения материальной точки по кривой наискорейшего спуска. В продвинутой части требуется разработать метод для нахождения аппроксимации этой кривой. Здесь и далее принимается $a = 2$ и $y_a = 1$. Константы циклоиды для этого граничного условия равны $C = 1{,}03439984$, $T = 1{,}75418438$.

Задача 2.6 (аппроксимация)

Требуется (базовая часть)

1. Разработать функции composite_simpson(a, b, n, f) и composite_trapezoid(a, b, n, f) численного интегрирования некоторой функции f на интервале $[a; b]$ по n узлам с помощью составной формулы Симпсона и составной формулы трапеций соответственно.

2. Рассчитать интеграл (18) для функции $y(x)$, соответствующей кривой наискорейшего спуска, с помощью составной формулы Симпсона и составной формулы трапеций для множества значений $n \in [3; 9999]$.

3. На одной координатной плоскости постройте log–log графики зависимостей абсолютной погрешности численного интегрирования от шага интегрирования для обоих формул.

4. Объясните, каким образом по полученному графику можно определить порядок точности формулы.

5. Для обоих формул сравните известные аналитические порядки точности с порядками точности, получаемыми с помощью соответствующих графиков. Обоснуйте алгоритм сравнения[54].

6. Существует ли оптимальный шаг интегрирования для первой или второй формулы, минимизирующий достижимую погрешность? Обоснуйте свой ответ.

Требуется (продвинутая часть)

1. Используя кусочно-линейную интерполяцию с равноудалёнными узлами, преобразовать задачу о минимизации функционала (18) к полудискретной форме, где аргументами минимизации будут параметры кусочно-линейной интерполяции.

2. Далее, используя составную формулу Симпсона, преобразовать задачу к полностью дискретной форме.

3. Решить полученную задачу минимизации, используя различные конфигурации дискретизации: с шагом интерполяции и шагом интегрирования от 10^{-3} до 1[55].

4. Используя log–log графики и линии уровня, оценить зависимость погрешности решения от шага интерполяции и шага интегрирования[56].

[54] Подсказка: для оценки порядка точности на основе log–log графика с помощью линейной регрессии может быть построена прямая, для которой следует определить угол наклона.

[55] Минимальное значение можно найти, например, с помощью scipy.optimize.minimize.

[56] Погрешность решения в данном случае рассчитывается как расстояние между двумя функциями. В данном случае разумно использовать L_2-норму для вычисления расстояния (раздел «1.3.3. Некоторые понятия функционального анализа» в конспекте лекций).

5. **Опциональное задание ☒ 1.** Использовать метод градиентного спуска и автоматическое дифференцирования для численного решения задачи оптимизации.

6. **Опциональное задание ☒ 2.** Использовать интерполяцию кубическими сплайнами (возможно, с модификациями) вместо кусочно-линейной интерполяции.

2.8. Быстрое преобразование Фурье

В современной прикладной математике для аппроксимации периодических данных и решений дифференциальных уравнений активно используются тригонометрические полиномы и спектральные методы соответственно. Их основным преимуществом является свойство спектральной сходимости, то есть сходимости быстрее, чем $O(h^N)$ для любого N. Свойство спектральной сходимости, как и свойство равномерной сходимости, являющееся следствием теоремы Стоуна–Вейерштрасса, сильно зависит от класса непрерывности аппроксимируемой функции. Одним из самых распространенных случаев использования тригонометрических полиномов является интерполяция периодической функции через дискретное преобразование Фурье, которое алгоритмически реализуется с помощью быстрого преобразования Фурье.

Задача 2.7 (БПФ)
Даны функции

$$\begin{aligned} f_1(x) &= 5 + 4\cos 2x + 2\sin 3x - \cos 4x, \\ f_2(x) &= |x|, \\ f_3(x) &= \begin{cases} -1 & -\pi \le x < 0 \\ 1 & 0 \le x \le \pi \end{cases} \end{aligned} \tag{20}$$

заданные на интервале $x \in [-\pi; \pi]$.

Требуется:

1. Используя алгоритм Кули–Тьюки, написать функцию fft_coeff(y_nodes), которая вычисляет и возвращает комплексные коэффициенты тригонометрического полинома, интерполирующего узлы y_nodes, равномерно распределенные на отрезке $[-\pi; \pi]$.

2. Протестировать корректность результатов работы функции fft_coeff(y_nodes) с помощью БПФ для функции $f_1(x)$. Пользуясь выкладками из лекций, объясните, как связаны возвращаемые комплексные коэффициенты (и их индексы) с исходной функцией.

3. Написать функцию trigonometric_interpolant(x, coeffs), которая вычисляет значение тригонометрического полинома с коэффициентами coeffs в точке x.

4. Используя функции trigonometric_interpolant и fft_coeff, произвести тригонометрическую интерполяцию функции $f_2(x)$ для $N = 2^{\tilde{n}}$, где $\tilde{n} \in 1, \ldots, 8$ и вывести результаты в виде графиков. Проанализируйте непрерывность функции $f_2(x)$ и исходя из графиков сделайте вывод о сходимости подобного приближения:
   – является ли сходимость равномерной?
   – является ли сходимость среднеквадратической?

5. Повторите те же шаги для функции $f_3(x)$ и ответьте на те же вопросы. В чем по вашему мнению причина различий?

Лабораторная работа 3

3.1. Требования к знаниям для выполнения

Для выполнения лабораторной работы обучающийся должен обладать знаниями:

– владеть навыками разработки программного обеспечения на языке Python (рекомендуется) или C++ на базовом уровне;

– владеть навыками использования программных инструментов: numpy;

– знать понятия: метод Гаусса, частичный выбор главного элемента, метод LU-разложение, метод сопряженных градиентов.

3.2. LU-разложение

Многие прямые методы решения СЛАУ построены на той или иной модификации метода Гаусса. Характерным примером является LU-разложение, напрямую выводимое из метода Гаусса. Наследуя плюсы и минусы последнего, LU-разложение оказывается подверженным проблемам, связанным с вычислительной устойчивостью, которые можно сгладить, используя частичный выбор главного элемента. В этом задании предлагается реализовать LU-разложение и исследовать его вычислительную устойчивость.

Задача 3.1 (LU-разложение)
Дана СЛАУ $\boldsymbol{A_1 x} = \boldsymbol{b_1}$:

$$\begin{bmatrix} 1 & 1 & 0 & 3 \\ 2 & 1 & -1 & 1 \\ 3 & -1 & -1 & 2 \\ -1 & 2 & 3 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ -3 \\ 4 \end{bmatrix}. \tag{21}$$

Дана СЛАУ $\boldsymbol{A_2 x} = \boldsymbol{b_2}$:

$$\begin{bmatrix} 3 & 1 & -3 \\ 6 & 2 & 5 \\ 1 & 4 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -16 \\ 12 \\ -39 \end{bmatrix}. \tag{22}$$

Требуется (базовая часть)

1. Написать функцию lu(A), которая производит LU-разложение матрицы A и возвращает матрицы L и U.

2. Написать функцию solve(L, U, b), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, где матрица $\boldsymbol{A}$ представлена в виде LU-разложения.

3. Найти решение СЛАУ (21) с помощью разработанной функции lu(A) и сравнить с точным решением этой СЛАУ: $\boldsymbol{x} = [-1, 2, 0, 1]^{\mathrm{T}}$.

Требуется (продвинутая часть)

1. Доказать, что для решения СЛАУ (22) необходимо на определённой итерации метода Гаусса произвести перестановку строк.

2. Модифицировать функцию lu(A, permute) так, чтобы она принимала аргумент permute и возвращает матрицы L, U и P. Если permute = True, то LU-разложение должно происходить с частичным выбором главного элемента, а возвращаемая матрица P при этом должна быть соответствующей матрицей перестановок. Если permute = False, то частичного выбора происходить не должно, а возвращаемая матрица P должна быть единичной.

3. Модифицировать функцию solve(L, U, P, b) так, чтобы она принимала на вход аргумент P и возвращала решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, где матрица $\boldsymbol{A}$ представлена в виде LU-разложения с матрицей перестановок $\boldsymbol{P}$, т.е. $\boldsymbol{PA} = \boldsymbol{LU}$.

4. Найти решение СЛАУ (22) с помощью разработанной функции solve(L, U, P, b), обозначаемое здесь и далее $\hat{\boldsymbol{x}}$.

5. Доказать, что модифицированная СЛАУ, полученная с помощью добавления к элементам a_{11} и b_1 СЛАУ (22) малого числа 10^{-p}, где p — произвольное целое число, имеет то же решение, что и исходная СЛАУ. Здесь и далее решение модифицированной СЛАУ будет обозначаться как $\tilde{\boldsymbol{x}}$.

6. С помощью разработанных функций lu(A, permute) и solve(L, U, P, b) найти решения модифицированной СЛАУ для $p \in [0; 12]$ для двух случаев:

– без частичного выбора главного элемента,

– с частичным выбором главного элемента.

7. Для обоих случаев построить log–log график зависимости относительной погрешности вычисления $E = \frac{||\hat{\boldsymbol{x}} - \tilde{\boldsymbol{x}}||_\infty}{||\hat{\boldsymbol{x}}||_\infty}$ от p. Проанализировав полученные результаты, сделайте подробный вывод о вычислительной устойчивости/неустойчивости решения СЛАУ с помощью LU-разложения для конкретного рассматриваемого случая и опишите, что нужно в этом контексте иметь в виду потенциальному пользователю ваших функций lu(A, permute) и solve(L, U, P, b).

3.3. Метод сопряженных градиентов

Системы линейных алгебраических уравнений (СЛАУ) зачастую появляются в результате дискретизации пространства в задачах, построенных на основе дифференциальных уравнений в частных производных. Действительно, после дискретизации частных производных конечными разностями линейные дифференциальные уравнения превращаются в линейные разностные уравнения, которые всегда можно представить в виде СЛАУ, задавшись соответствующими граничными условиями. В этом задании мы построем СЛАУ с помощью дискретизации двумерного уравнения Пуассона и решим ее с помощью метода сопряженных градиентов.

Задача 3.2 (Метод сопряженных градиентов)

Дано двумерное уравнение Пуассона, являющееся общей формой стационарного уравнения теплопроводности и описывающее стационарное поле температуры:

$$-\frac{\partial^2 T}{\partial x^2} - \frac{\partial^2 T}{\partial y^2} = f(x, y), \tag{23}$$

где $T = T(x, y)$ — температура в точке (x, y), $f(x, y) = 1$ — функция тепловых источников, описывающая в данном случае равномерный нагрев. Рассматривается пространство $(x, y) \in [0; 1] \times [0; 1]$, а также однородные (т.е. нулевые) граничные условия: $T(x, 0) = T(0, y) = T(x, 1) = T(1, y) = 0$.

Требуется

1. Построить разрешающую СЛАУ, проведя дискретизацию пространства с помощью замены частных производных в данном уравнении формулой численного дифференцирования второго порядка для второй производной (см. лекцию 4). Для этого необходимо построить сетку с помощью равномерно распределенных узлов:

$$x_i = ih, \quad i = 0, \ldots, N,$$
$$y_j = jh, \quad j = 0, \ldots, N,$$

где $h = \frac{1}{N}$ и $N+1$ является числом узлов вдоль каждой из координат. Формула численного дифференцирования относительно определенной координаты применяется так, что вторая координата рассматривается зафиксированной. Вектором решения СЛАУ должен быть модифицированный вектор

$$\boldsymbol{T} = [T_{0,0}, T_{1,0}, \ldots, T_{N,0}, T_{0,1}, T_{1,1}, \ldots, T_{N,2}, \ldots, T_{N-1,N}, T_{N,N}],$$

где $T_{i,j} = T(x_i, y_j)$, из которого следует исключить элементы, связанные с заданными граничными условиями. Из матрицы коэффициентов так же должны быть исключены строки и столбцы, ассоциированные с граничными условиями. Матрицу коэффициентов желательно представить в виде блочной матрицы.

2. Описать свойства полученной матрицы:

– является ли матрица положительно определенной?

– является ли матрица ленточной? Если да, вычислите ширину ленты.

– обладает ли матрица строгим (нестрогим) диагональным преобладанием?

3. Сделать вывод о применимости метода сопряженных градиентов и вычислительной устойчивости/неустойчивости решения исходя из свойств матрицы.

4. Написать функцию conjugate_gradient(A, b, eps), которая возвращает решение СЛАУ, построенной из матрицы коэффициентов A и правого вектора b и со среднеквадратичной нормой вектора невязки строго меньше eps. Решение должно производиться с помощью метода сопряженных градиентов.

5. Найти решение СЛАУ, полученной в первом пункте, с помощью функции conjugate_gradient при $N = 9$ и $N = 17$ и среднеквадратичной норме вектора невязки строго меньше 10^{-4}:

– укажите получившиеся размерности матриц коэффициентов и оцените, сколько требуется памяти для хранения этих матриц в памяти компьютера в предположении, что каждый элемент матрицы представляется в виде числа с двойной точностью;

– для каждого N выведите на экран линии уровня решения[57];

– сравните результаты и сделайте вывод.

6. Для $N = 9$ и $N = 17$ построить график зависимости среднеквадратичной нормы вектора невязки от номера итерации метода сопряженных градиентов и проанализировать полученные зависимости. Для этого необходимо модифицировать функцию conjugate_gradient так, чтобы она дополнительно возвращала среднеквадратичную норму вектора невязки на каждой итерации. Ось ординат на графике должна быть отображена в логарифмической шкале.

3.4. Модель Лоренца

Численные методы решения задачи Коши для систем обыкновенных дифференциальных уравнений (ОДУ) 1-го порядка активно используются при построении классификации режимов динамических систем, поведение которых претерпевает качественные изменения, называемыми бифуркациями, при изменении одного или несколько параметров динамической системы. Использование численных методов почти всегда обусловлено нелинейностью этих систем, которые, в случае размерности системы 3 и больше, могут порождать динамический хаос. В этом задании мы исследуем одну из самых известных моделей динамического хаоса, модель Лоренца, и построим простейшую классификацию её режимов с помощью численных методов решения задачи Коши и случайно сгенерированных начальных условий.

[57] Линии уровня с заполнением в matplotlib.pyplot строятся с помощью функции contourf()

Задача 3.3 (Модель Лоренца)
Дана система ОДУ 1-го порядка $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x})$, где $\boldsymbol{x} = \boldsymbol{x}(t) = \big[x(t), y(t), z(t)\big]^{\mathrm{T}}$:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sigma(y-x) \\ x(r-z)-y \\ xy - bz \end{bmatrix}, \tag{24}$$

где $\sigma = 10$ и $b = 8/3$.

Требуется

1. Найти аналитически все стационарные позиции заданной системы ОДУ[58].

2. Написать следующие функции, каждая из которых возвращает дискретную траекторию системы ОДУ с правой частью, заданную функцией f, начальным условием x_0, шагом по времени h и конечным временем t_n:

– euler(x_0, t_n, f, h), где дискретная траектория строится с помощью метода Эйлера;

– implicit_euler(x_0, t_n, f, h), где дискретная траектория строится с помощью неявного метода Эйлера[59];

– adams_bashforth_moulton(x_0, t_n, f, h), где дискретная траектория строится с помощью метода Адамса–Башфорта–Моултона 4-го порядка[60].

3. Выбрать «численный метод по умолчанию» из функций euler, implicit_euler и adams_bashforth_moulton и «шаг по умолчанию» h, предполагая, что ваш выбор даст наиболее точное решение. Аргументируйте свой выбор.

4. Для каждого из значений параметра $r \in \{0, 10, 20, 30\}$ провести следующий анализ[61].

а) Сгенерировать набор случайных[62] начальных условий $\{(x_i, y_i, z_i)\}_{i=1}^{100}$.

б) Для каждого случайного начального условия следует найти решение[63] данной системы ОДУ с помощью численного метода по умолчанию.

в) Вывести подмножество[64] полученных траекторий на одном графике в виде фазового портрета[65].

г) Охарактеризуйте наблюдаемый динамический режим.

i. Наблюдается ли сходимость решений независимо от начального условия к одной точке? Если да, то как вы охарактеризуете эту точку? Если нет, то как вы охарактеризуете получаемую структуру решения?

ii. Наблюдаются ли какие-то особенности поведения решения при переходе от начального условия к характеризуемому динамическому режиму?

[58] Стационарной позицией динамической системы является постоянная во времени траектория $\boldsymbol{x}^*(t) = \text{const}$. Очевидно, что для стационарной позиции $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{0}$, из чего следует, что стационарные позиции можно найти, решив нелинейное в общем случае уравнение $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$.

[59] Для решения нелинейных уравнений вы можете использовать функцию scipy.optimize.root

[60] Для нахождения начальных условий для этого метода вы можете использовать функцию implicit_euler

[61] При необходимости вы можете рассмотреть дополнительные значения параметра r

[62] Начальные условия разумно генерировать, исходя из равномерного распределения, и, к примеру, следующих ограничений: $x(0) \in [-50; 50]$, $y(0) \in [-50; 50]$ и $z(0) \in [0; 70]$. Случайные числа в Python генерируются функцией numpy.random.rand().

[63] Подразумевается, что начальное время t_{start} равно нулю и конечное время вы определяете самостоятельно, исходя из особенностей поведения системы при данном значении параметра.

[64] Вам следует выбрать траектории таким образом, чтобы фазовый портрет наглядно демонстрировал описанный вами режим динамической системе для достаточно большого количества траекторий.

[65] В зависимости от значения параметра r, вы можете использовать либо двумерные портреты, т.е. плоскости, где по осях отложены две координаты из x, y, z, либо трёхмерные портреты.

5. Используя наиболее сложный динамический режим как пример, ответить на следующие вопросы:

а) Сколько суммарно времени в секундах занимает нахождение всех решений для всех сгенерированных начальных условий[66]?

б) Как влияет увеличение и уменьшение шага по умолчанию h на фазовый портрет и время нахождения решений?

в) Как влияет на фазовый портрет и время нахождения решений замена численного метода по умолчанию на каждый из двух оставшихся? Как влияет на фазовый портрет и время нахождения решений увеличение и уменьшение шага по умолчанию h в каждом из двух случаев?

3.5. Вынужденные колебания маятника

Численные методы решения задачи Коши для систем обыкновенных дифференциальных уравнений (ОДУ) 1-го порядка активно используются для изучения динамических систем, траектории которых не удаётся найти с помощью аналитических методов. Одним из простейших примеров являются вынужденные колебания маятника без предположения о малости угла отклонения маятника от вертикальной оси. Подобная математическая модель является фундаментальной для робототехники, и отчасти отражает процессы, моделируемые, например, в случае рук-манипуляторов. В данной лабораторной работе исследуются траектории, являющиеся решениями соответствующих задач Коши на основе такой модели, и проанализируем, к каким решениям они сходятся в зависимости от начальных условий.

Задача 3.4 (Вынужденные колебания маятника)
Дано ОДУ 2-го порядка:

$$\frac{d^2\theta}{dt^2} + 0{,}1\frac{d\theta}{dt} + \sin(\theta) = \cos(t), \tag{25}$$

где $\theta(t)$ обозначает угол отклонения маятника от вертикальной оси как функцию времени t.

Требуется (базовая часть)

1. Преобразовать данное ОДУ в систему ОДУ 1-го порядка.

2. Разработать следующие функции, каждая из которых возвращает дискретную траекторию системы ОДУ с правой частью, заданной функцией `f`, начальным условием `x_0`, шагом по времени `h` и конечным временем `t_n`:

– `runge_kutta(x_0, t_n, f, h)`, где дискретная траектория строится с помощью явного метода Рунге–Кутты 4-го порядка;

– `adams_moulton(x_0, t_n, f, h)`, где дискретная траектория строится с помощью неявного трёхшагового метода Адамса–Моултона[67] (выполняется в рамках продвинутой части);

– `milne_simpson(x_0, t_n, f, h)`, где дискретная траектория строится с помощью метода Милна–Симпсона (схема предиктор–корректор).

3. Для каждого из реализованных методов:

[66] Замеры времени в секундах в Python производятся, например, с помощью функции `timeit.default_timer()`. Вызвав эту функцию до и после нужного участка кода, а затем найдя разницу между полученными замерами, вы найдете время исполнения кода.

[67] Для решения нелинейных уравнений вы можете использовать функцию `scipy.optimize.root`.

Для нахождения начальных условий для этого метода вы можете использовать функцию `runge_kutta`

– Численно каждым из методов найти траектории заданной динамической системы, используя шаг $h = 0.1$ и 15 различных начальных условий[68], для которых: $\theta(0) = 0$ и $\frac{d\theta}{dt}|_{t=0}$ следует выбрать случайно из интервала $[1{,}85; 2{,}1]$.

– Вывести полученные траектории на едином графике как зависимости $\theta(t)$ (для каждого метода на отдельном графике).

4. В чем принципиальные отличия реализованных методов друг от друга? В чем они схожи?

5. Для каждой из схем каково значение шага, при котором она становится неустойчивой[69]?

Требуется (продвинутая часть)

6. Вывести разными цветами фазовые траектории на едином двумерном графике: по оси абсцисс θ, по оси ординат — $\frac{d\theta}{dt}$, при всех различных начальных условиях (для каждого метода на отдельном графике).

7. Зафиксировать одно начальное условие (произвольно). Вывести фазовые траектории на одном двумерном графике, формируемые разными методами. Сделать вывод.

8. Какая из схем является наиболее затратной с точки зрения времени вычислений при произвольном значении шага, дающем устойчивое решение для каждой из схем[70]? Наименее затратной?

9. Как вы можете охарактеризовать асимптотические состояния, к которым сходится решение в зависимости от начальных условий? Опишите их физический смысл.

Опциональное задания.

10. Построить области притяжения[71] каждого из асимптотических состояний:

а) Каждому из асимптотических состояний назначить свой цвет.

б) Построить множество начальных условий на основе структурированной сетки[72], построенной путём разбиения области $[-4\pi; 4\pi] \times [-5, 5]$.

в) Выбрав наименее затратную по времени схему, построить траектории для каждого из начальных условий, соответствующих узлам сетки, и классифицировать, к кому асимптотическому состоянию сходится каждая из траекторий. Описать алгоритм, с помощью которого вы определяете, что данная траектория сходится к определённому асимптотическому состоянию.

г) Вывести на экран график начальных условий, имеющих цвет соответствующих им асимптотических состояний.

11. Какие характерные детали полученных областей притяжения вы можете выделить?

---

[68] Начальные условия определяются одинаковыми для разных методов должны быть одинаковыми.

[69] Это значение может быть найдено эмпирически.

[70] Замеры времени в секундах в Python производятся, например, с помощью функции timeit.default_timer(). Вызвав эту функцию до и после нужного участка кода, а затем найдя разницу между полученными замерами, вы найдете время исполнения кода.

[71] Областью притяжения определённого асимптотического состояния динамической системы называет множество таких начальных условий $\theta(0)$ и $\frac{d\theta}{dt}|_{t=0}$, что соответствующие им траектории стремятся к данному асимптотическому состоянию при $t \to \infty$.

[72] Количество узлов в сетке определяется возможностями вашего компьютера. Взглянув на график областей притяжения, вам должно быть ясно, стоит ли уменьшить шаг сетки. Максимальный достаточный размер: 600×400.

3.6. Модель биологического нейрона

Численные методы решения задачи Коши для систем обыкновенных дифференциальных уравнений (ОДУ) 1-го порядка активно используются далеко за пределами стандартных инженерных задач. Примером области, где подобные численные методы крайне востребованы, является нейробиология, где открытые в XX веке модели биологических нейронов выражаются через дифференциальные уравнения 1-го порядка. Математическая формализация моделей биологических нейронов также привела к появлению наиболее реалистичных архитектур нейронных сетей, известных как спайковые нейронные сети (Spiking Neural Networks). В данной лабораторной работе мы исследуем одну из простейших моделей подобного типа: модель Ижикевича.

Задача 3.5 (Модель Ижикевича)
Дана система из двух ОДУ 1-го порядка:

$$\begin{cases} \dfrac{dv}{dt} = f_1(u,v) = 0{,}04v^2 + 5v + 140 - u + I; \\ \dfrac{du}{dt} = f_2(u,v) = a(bv - u); \end{cases} \tag{26}$$

и дополнительного условия, определяющего возникновение импульса в нейроне:

$$\text{если } v \geqslant 30, \text{ то } \begin{cases} v \leftarrow c; \\ u \leftarrow u + d; \end{cases} \tag{27}$$

где v — потенциал мембраны (мВ), u — переменная восстановления мембраны (мВ), t — время (мс), I — внешний ток, приходящий через синапс в нейрон от всех нейронов, с которыми он связан.

Описания параметров представленной системы:

a — задаёт временной масштаб для восстановления мембраны (чем больше a, тем быстрее происходит восстановление после импульса);

b — чувствительность переменной восстановления к флуктуациям разности потенциалов;

c — значение потенциала мембраны сразу после импульса;

d — значение переменной восстановления мембраны сразу после импульса.

Требуется (базовая часть)

1. Реализовать следующие функции, каждая из которых возвращает дискретную траекторию системы ОДУ с правой частью, заданной функцией `f`[73], начальным условием `x_0`, шагом по времени `h` и конечным временем `t_n`:

– `euler(x_0, t_n, f, h)`, где дискретная траектория строится с помощью метода Эйлера;

– `implicit_euler(x_0, t_n, f, h)`, где дискретная траектория строится с помощью неявного метода Эйлера[74];

– `runge_kutta(x_0, t_n, f, h)`, где дискретная траектория строится с помощью метода Рунге–Кутты 4-го порядка.

2. Для каждого из реализованных методов численно найти траектории заданной динамической системы, используя шаг $h = 0{,}5$ и характерные режимы, указанные в табл. 3. В качестве начальных условий можно использовать $v(0) = c$ и $u(0) = bv(0)$. Внешний ток принимается равным $I = 5$.

[73] В контексте поставленной задачи: $f = [f_1(u,v), f_2(u,v)]^{\mathrm{T}}$ (система (26))

[74] Для решения нелинейных уравнений вы можете использовать функцию `scipy.optimize.root`

Таблица 3. Характерные режимы заданной динамической системы и соответствующие значения ее параметров

Режим	a	b	c	d
Tonic spiking (TS)	0,02	0,2	-65	6
Phasic spiking (PS)	0,02	0,25	-65	6
Chattering (C)	0,02	0,2	-50	2
Fast spiking (FS)	0,1	0,2	-65	2

3. Вывести полученные траектории на четырёх отдельных графиках как зависимости потенциала мембраны v от времени t, где каждый график должен соответствовать своему характерному режиму работы нейрона.

4. По полученным графикам кратко описать особенности указанных режимов.

Требуется (продвинутая часть)

5. Объяснить, в чем состоят принципиальные отличия реализованных методов? В чем они схожи?

6. Произвести интегрирование во времени до 1000 мс нейронной сети с помощью метода Эйлера, используя следующую информацию.

(а) Динамика каждого нейрона в нейронной сети описывается заданной моделью Ижикевича. В нейронной сети имеется 800 возбуждающих нейронов и 200 тормозных. Возбуждающие нейроны имеют следующие значения параметров: $a = 0{,}02$, $b = 0{,}2$; $c = -65 + 15\alpha^2$; $d = 8 - 6\beta^2$ и внешний ток в отсутствие токов от других нейронов равен $I = I_0 = 5\xi$, где α, β и ξ — случайные числа от 0 до 1 (распределение равномерное). Тормозные нейроны имеют следующие значения параметров: $a = 0{,}02 + 0{,}08\gamma$; $b = 0{,}25 - 0{,}05\delta$; $c = -65$; $d = 2$ и внешний ток в отсутствие токов от других нейронов равен $I = I_0 = 2\zeta$, где γ, δ и ζ — случайные числа от 0 до 1. В качестве начальных условий используются значения $v(0) = -65$ и $u(0) = bv(0)$.

(б) Нейронная сети может быть смоделирована с помощью полного графа. Матрица смежности $\boldsymbol{W}$ этого графа описывает значения токов, передаваемых от нейрона к нейрону в случае возникновения импульса. То есть, при возникновении импульса нейрона j внешний ток связанного с ним нейрона i единовременно увеличивается на величину W_{ij} и затем сразу же падает до нуля, что и моделирует передачу импульса по нейронной сети. Значение W_{ij} равно $0{,}5\theta$, если нейрон j является возбуждающим, и $-\tau$, если тормозным, где θ и τ — случайные числа от 0 до 1.

7. Вывести на экран импульсы всех нейронов как функцию времени[75] и определить частоты характерных синхронных (или частично синхронных) колебаний нейронов в сети.

[75] К примеру, вы можете отобразить график в форме scatter plot, где ось абсцисс будет соответствовать времени, ось ординат — номеру нейрона, а отдельный импульс будет обозначаться чёрной точкой.

Лабораторная работа 4

4.1. Требования к знаниям для выполнения

Для выполнения лабораторной работы обучающийся должен обладать знаниями:

– владеть навыками разработки программного обеспечения на языке Python (рекомендуется) или C++ на базовом уровне;

– владеть навыками использования программных инструментов: `numpy`;

– знать понятия: метод Ньютона, метод градиентного спуска, метод Рунге–Кутты четвертого порядка.

4.2. Модель Лотки–Вольтерры

Задача о нахождении корней нелинейных систем алгебраических уравнений имеет множество приложений в самых разных областях. В частности, подобную задачу необходимо решать при нахождении стационарных позиций динамических систем. Одним из общих видов динамических систем является система ОДУ вида $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x})$, где решение такой системы $\boldsymbol{x}^*(t)$ называется траекторией динамической системы. Стационарной позицией динамической системы является постоянная во времени траектория $\boldsymbol{x}^*(t) = \text{const}$. Очевидно, что для стационарной позиции $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{0}$, из чего следует, что стационарные позиции можно найти, решив нелинейное в общем случае уравнение $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. В данной лабораторной работе мы будем исследовать одну из самых известных динамических систем — модель Лотки–Вольтерры, которая моделирует взаимодействие между «жертвами» и «хищниками». Подобные модели активно используются для описания динамических процессов в биологии, химии, социологии и экономике.

Задача 4.1 (модель Лотки–Вольтерры)

Дана модель Лотки–Вольтерры в виде системы ОДУ $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x})$, где $\boldsymbol{x} = \boldsymbol{x}(t) = \\ = \big[x(t), y(t)\big]^{\mathrm{T}}$:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \alpha x - \beta xy \\ \delta xy - \gamma y \end{bmatrix}, \tag{28}$$

где x — количество «жертв», y — количество «хищников», $\alpha = 3$ — коэффициент рождаемости «жертв», $\beta = 0{,}002$ — коэффициент убыли «жертв», $\delta = 0{,}0006$ — коэффициент рождаемости «хищников», $\gamma = 0{,}5$ — коэффициент убыли «хищников».

Требуется (базовая часть)

1. Написать функцию `rk4(x_0, t_n, f, h)`, возвращающая дискретную траекторию системы ОДУ с правой частью, заданную функцией `f`, начальным условием `x_0`, шагом по времени `h` и конечным временем `t_n`, полученную с помощью метода Рунге–Кутты 4-го порядка.

2. Найти траектории для заданной системы для ряда начальных условий $x_i^{(0)} = \\ = 200i$, $y_j^{(0)} = 200j$, где $i, j = 1, \ldots, 10$.

3. Вывести все полученные траектории на одном графике в виде фазового портрета. Объясните, какой вид имеют все полученные траектории. В качестве подтверждения выведите на экран совместно графики траекторий $x(t)$ и $y(t)$ для одного репрезентативного случая.

Требуется (продвинутая часть)

1. Найти аналитически все стационарные позиции заданной системы ОДУ.

2. Отметить на фазовом портрете, полученном в базовой части, найденные стационарные позиции. Объясните, что происходит с траекториями заданной системы при приближении к каждой из стационарных позиций.

3. Написать функцию newton(x_0, f, J), которая, используя метод Ньютона, возвращает корень векторной функции f с матрицей Якоби J и количество проведённых итераций. Аргументы f и J являются функциями, принимающими на вход вектор x и возвращающими соответственно вектор и матрицу. В качестве критерия остановки следует использовать ограничение на относительное улучшение[76]: $||\boldsymbol{x}^{k+1} - \boldsymbol{x}^k||_\infty < \varepsilon$, где $\varepsilon = 10^{-8}$.

4. Написать функцию gradient_descent(x_0, f, J), которая, используя метод градиентного спуска, возвращает корень векторной функции f с матрицей Якоби J и количество проведённых итераций. Используйте тот же критерий остановки, что и в предыдущем пункте.

5. Используя каждую из функций newton() и gradient_descent(), провести следующий анализ.

а) Найти стационарные позиции как нули заданной векторной функции $\boldsymbol{f}(\boldsymbol{x})$ для ряда начальных условий $x_i^{(0)} = 15i$, $y_j^{(0)} = 15j$, где $i, j = 0, 1, \ldots, 200$.

б) Для каждой полученной стационарной позиции рассчитать её супремум-норму, что в результате даст матрицу супремум-норм размерности 201×201.

в) Вывести на экран линии уровня[77] с заполнением для полученной матрицы относительно значений $x_i^{(0)}, y_j^{(0)}$.

г) Описать наблюдения, исходя из подобной визуализации результатов.

д) Найти математическое ожидание и среднеквадратическое отклонение количества итераций.

е) Выбрать некоторую репрезентативную начальную точку из $x_i^{(0)}, y_j^{(0)}$ и продемонстрировать степень сходимости метода с помощью соответствующего log–log-графика[78].

6. Проанализировав полученные результаты, сравнить свойства сходимости метода Ньютона и метода градиентного спуска.

4.3. Многошаговые методы численного решения задачи Коши

При создании численных схем решения дифференциальных уравнений в частных производных дискретизация времени приводит к системе обыкновенных дифференциальных уравнений (ОДУ) с соответствующей задачей Коши. Численные методы решения таких систем можно разделить на одношаговые, использующие информацию только с предыдущего или только с последующего временного шага при нахождении решения на текущем шаге, и многошаговые, использующие информацию с нескольких итераций по времени. Многошаговые методы дополнительно требуют использования одношагового метода для проведения первых итераций, инициализирующих многошаговый метод. В этом задании мы используем метод

[76] Супремум-норма находится с помощью функции np.linalg.norm() с параметром ord=np.inf.

Вместо явного вычисления обратной матрицы Якоби следует использовать двухшаговую процедуру, описанную в лекциях, где решение СЛАУ следует искать с помощью функции solve(), написанной в рамках предыдущей лабораторной.

[77] Линии уровня с заполнением в matplotlib.pyplot строятся с помощью функции contourf().

[78] Для этого следует воспользоваться определением степени сходимости, «отбросив» предел.

Адамса–Башфорта 4-го порядка и инициализирующий его метод Рунге–Кутты 4-го порядка для нахождения решения нестационарного уравнения теплопроводности.

Задача 4.2 (многошаговые методы численного решения задачи Коши)
Дано нестационарное уравнение теплопроводности:

$$\frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial x^2} - \frac{\partial^2 T}{\partial y^2} = f(x, y),$$

где $T = T(x, y, t)$ — температура в точке (x, y) в момент времени t, $f(x, y) = 1$ — функция тепловых источников, описывающая в данном случае равномерный нагрев. Рассматривается пространство $(x, y) \in [0; 1] \times [0; 1]$, однородные (т.е. нулевые) граничные условия $T(x, 0, t) = T(0, y, t) = T(x, 1, t) = T(1, y, t) = 0$ и нулевые начальные условия $T(x, y, 0) = 0$.

Требуется

1. Используя результаты лабораторной работы № 3 (вариант 2), провести дискретизацию пространства с $N = 18$ узлами вдоль каждого направления и дискретизацию по времени с шагом Δt, используя метод Адамса–Башфорта 4-го порядка и метод Рунге–Кутты 4-го порядка. Например, для метода Адамса–Башфорта 4-го порядка результатом дискретизации должен быть итерационный метод вида

$$\boldsymbol{T}_{n+1} = \boldsymbol{T}_n + \Delta t \sum_{k=1}^{3} (a_k \boldsymbol{A} \boldsymbol{T}_{n-k+1} + \boldsymbol{f}), \tag{29}$$

где $\boldsymbol{A}$ и $\boldsymbol{f}$ были выведены в лабораторной работе № 3 (вариант 2).

2. Написать функцию ab4(), которая проводит одну итерацию метода Адамса–Башфорта 4-го порядка, используя решения системы ОДУ на трёх предыдущих итерациях. Аргументы функции следует определить самостоятельно.

3. Написать функцию rk4(), которая проводит одну итерацию метода Рунге–Кутты, используя решение системы ОДУ на предыдущей итерации. Аргументы функции следует определить самостоятельно.

4. Написать функцию ode_solve(f, t_final, delta_t), которая находит решение ОДУ с правой частью, выраженной функцией f, до момента времени t_final с шагом по времени delta_t, используя метод Рунге–Кутты 4-го порядка для инициализации первых двух шагов и метод Адамса–Башфорта 4-го порядка для дальнейших итераций.

5. Проведя несколько вычислительных экспериментов с помощью функции ode_solve(), определить с точностью до порядка максимальное значение Δt, обозначаемое $\Delta t_{\max}$, при котором решение заданного дифференциального уравнения является неустойчивым. Требуется продемонстрировать неустойчивость решения с помощью графика зависимости температуры, усреднённой по области $[0; 1] \times [0; 1]$, от времени.

6. Используя Δt на порядок меньшее, чем $\Delta t_{\max}$, построить:

– линии уровня функции $T(x, y, t)$ для нескольких моментов времени, демонстрирующих сходимость решения;

– график зависимости температуры, усреднённой по области $[0; 1] \times [0; 1]$, от времени.

7. Сравнить решение, к которому сходится численное решение заданного дифференциального уравнения, с решением, полученным в лабораторной работе № 3 (вариант 2). Сравнив их дополнительно с решением, полученным при шаге $\Delta t_{\max}$, сделать вывод об устойчивости решения и устойчивости метода.

4.4. Устойчивость прямых методов решения СЛАУ

Решение систем линейных алгебраических уравнений (СЛАУ) — ключевой этап огромного множества задач вычислительной математики и анализа данных. В случае плотных матриц сравнительно небольшой размерности, для нахождения решения СЛАУ часто применяются прямые методы, такие как метод Гаусса, метод прогонки или разложение Холецкого. В то же время известно, что многие прямые методы обладают вычислительной неустойчивостью и могут приводить к некорректному решению для некоторых матриц коэффициентов. В этой лабораторной работе рассматриваются матрицы нескольких видов и с помощью генерации большого количества случайных матриц демонстрируется наличие или отсутствие вычислительной неустойчивости у метода Гаусса, метода прогонки и разложения Холецкого.

Задача 4.3 (устойчивость прямых методов решения СЛАУ)
Требуется (базовая часть)

1. Написать функцию gauss(A, b, pivoting), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, полученное с помощью метода Гаусса. Если параметр pivoting = True, то решение должно находиться с частичным выбором главного элемента. Если pivoting = False, то выбора главного элемента происходить не должно.

2. Написать функцию thomas(A, b), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, полученное с помощью метода прогонки[79].

3. Среди реализованных методов, включая два варианта метода Гаусса, выбрать тот, который минимизирует вычислительные погрешности[80] для случая квадратных матриц общего вида. В рамках задания такой метод будем называть «универсальным».

4. Разработать и описать алгоритм генерации случайных невырожденных матриц размерности 6×6 с элементами $a_{ij} \in \mathbb{R}, |a_{ij}| < 1$ общего и 3-х диагонального вида.

5. Сгенерировав 1000[81] случайных матриц $\boldsymbol{A}^{(j)}$ каждого типа с 32-битным float-представлением элементов[82] необходимо провести следующий эксперимент.

а) Выбрать «специальный» вычислительно-эффективный метод по типу матрицы[83].

б) Для каждой СЛАУ $\boldsymbol{A}^{(j)}\boldsymbol{x} = [1, 1, 1, 1, 1, 1]^{\mathrm{T}}$ найти решение с помощью «универсального» и «специального» методов, а затем найти относительную погрешность вычислений с помощью среднеквадратичной и супремум-нормы. Вывести на экран распределения погрешностей в виде гистограмм.

в) Является ли выбранный «специальный» метод вычислительно устойчивым? Почему?

[79] Метод прогонки был предложен И.М. Гельфандом и О.В. Локуциевским (в 1952 г.; опубликовано в 1960 и 1962 гг.), а также независимо другими авторами. За рубежом он известен как «алгоритм Томаса» (*Thomas algorithm*, по имени британского физика Llewellyn Hilleth Thomas), или «трёхдиагональный матричный алгоритм» (*tridiagonal matrix algorithm*), иногда также называют «метод заметания» *sweep method*, последний, однако, имеет более общий смысл.

[80] Описать в отчёте метод определения вычислительной погрешности.

[81] Количество можно варьировать для получения наглядных результатов анализа.

[82] В numpy-массивах нужный тип данных задается аргументом конструктора dtype=numpy.float32

[83] По-сути, из трёх реализованных методов выбирается один с минимальным количество арифметико-логических операций.

Требуется (продвинутая часть)

1. Расширить генератор из задания 4 для получения положительно-определенных матриц[84].

2. Написать функцию cholesky(A, b), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, полученное с помощью разложения Холецкого.

3. Провести требуемый в задании 5 анализ учитывая метод Холецкого (сравнить с «универсальным» методом).

4. Для всех рассмотренных ранее матриц вывести на экран распределение спектральных радиусов[85] и распределение чисел обусловленности[86] в виде гистограмм. Сделать вывод.

5. Влияет ли значение спектрального радиуса матрицы на вычислительную устойчивость рассмотренных алгоритмов? Если да, то как?

6. Влияет ли значение отношения максимального по модулю собственного числа к минимальному по модулю собственному числу на вычислительную устойчивость алгоритма? Если да, то как?

7. Влияет ли число обусловленности на вычислительную устойчивость алгоритма? Если да, то как?

Опциональное задание.

1. Написать функцию iter_refinement(A, b), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, полученное с помощью итерационного уточнения[87], полученного на основе решения методом Гаусса с частичным выбором главного элемента[88].

2. Сравнить данный метод решения с «универсальным» (оценить разницу решений) для всех приведенных ранее матриц.

4.5. Методы решения СЛАУ с разреженными матрицами

Системы линейных алгебраических уравнений неизбежно появляются как промежуточный или конечный этап в нахождении численных решений в ряде методов вычислительной математики и анализа данных. Часто матрицы большой размерности оказываются разреженными, что требует использования более компактных способов хранения матриц в памяти устройства. СЛАУ, основанные на таких матрицах, чаще всего решаются с использованием итерационных или полупрямых методов. Выбор конкретного метода основан на заранее известных свойствах матрицы и эмпирических проверках. В этой лабораторной работе мы рассмотрим несколько методов решения СЛАУ и изучим скорость их сходимости на примере матрицы коэффициентов, полученной в MSC/NASTRAN.

Задача 4.4 (методы решения СЛАУ с разреженными матрицами)

Требуется (базовая часть)

1. Сформировать матрицу коэффициентов $\boldsymbol{A}$ и правый вектор СЛАУ $\boldsymbol{b}$, используя следующие файлы: https://archrk6.bmstu.ru/index.php/f/837681 и https://archrk6.bmstu.ru/index.php/f/837678 соответственно[89]. Данная матрица является положительно определенной.

[84] Можно использовать метод перебора, но иные способы также приветствуются.

[85] Собственные числа матрицы вычисляются с помощью функции numpy.linalg.eigvals().

[86] Число обусловленности матрицы вычисляется с помощью функции numpy.linalg.cond().

[87] Можно использовать 64-битную арифметику с плавающей точкой для шага уточнения.

[88] Выбор количества итераций и соответствующая аргументация остаётся за вами.

[89] Удобнее всего для загрузки файлов сразу в numpy массивы использовать функцию numpy.loadtxt.

2. Получить решение с помощью наиболее подходящего прямого метода (из класса методов последовательного исключения). Данное решение мы далее будем называть точным.

3. Вычислить число обусловленности матрицы $\boldsymbol{A}$ и сделать вывод о возможности использования чисел с одинарной точностью для хранения матрицы и проведения всех дальнейших вычислений. При положительном решение, далее необходимо использовать числа с одинарной точностью[90].

4. Понизить число обусловленности как минимум на два порядка, используя предобуславливание[91].

5. Написать функцию jacobi(A, b), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, полученное с помощью метода Якоби.

6. Написать функцию gauss_seidel(A, b), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, полученное с помощью метода Гаусса–Зейделя.

7. Для случаев с предобуславливанием и без него требуется следующее.

а) Найти решение исходной СЛАУ с помощью метода Якоби.

б) Найти решение исходной СЛАУ с помощью метода Гаусса–Зейделя.

в) Вывести на экран зависимость среднеквадратичной нормы абсолютной погрешности от номера итерации для каждого из методов[92].

г) Вывести на экран зависимость среднеквадратичной нормы невязки от номера итерации для каждого из методов[91].

8. Объяснить различия в скорости сходимости для всех рассмотренных случаев.

Требуется (продвинутая часть)

1. Написать функцию visualize_matrix(A), которая визуализирует матрицу как двумерный рисунок, используя следующий подход: элементы матрицы, близкие к нулю, выводятся на экран как белые квадраты (или не выводятся вовсе), а остальные элементы выводятся как квадраты, имеющие цвет из непрерывной цветовой палитры, которую необходимо выбрать самостоятельно[93].

2. Визуализировать матрицу $\boldsymbol{A}$ и сделать вывод о ее разреженности. Является ли данная матрица ленточной? Если да, то какова ширина ленты?

3. Написать и подробно описать класс SparseMatrix, позволяющий хранить разреженную матрицу в формате CSR (англ. *compressed sparse row* или сжатое хранение в строке). Этот формат особенно эффективен при большом количестве перемножений матриц на вектора. В нем исходная матрица представляется в виде трёх массивов:

– массив, хранящий только ненулевые значения, которые берутся порядок из первой непустой строки;

– массив, хранящий номера столбцов, соответствующих ненулевым значениям из первого массива;

– массив, i-й элемент которого хранит количество ненулевых элементов в строках до $i-1$ включительно; в первом его элементе хранится 0, а последний элемент совпадает с числом ненулевых элементов исходной матрицы.

В классе SparseMatrix должны быть реализованы все матричные операции, необходимые в дальнейших пунктах задания.

[90] Не забудьте о диапазоне чисел с одинарной точностью.

[91] Обратите внимание, что это приведет к изменению СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$.

[92] Желательно вывести зависимости для всех случаев (с/без предобуславливания, метод Якоби, метод Гаусса–Зейделя) на одном графике.

[93] В matplotlib цветовые палитры называются colormaps, и примером подходящей палитры является seismic.

4. Написать функцию conjugate_gradient_method(A, b), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, полученное с помощью метода сопряжённых градиентов.

5. Для случаев с предобуславливанием и без него требуется следующее.

а) Найти решение исходной СЛАУ с помощью метода сопряжённых градиентов, храня матрицу $\boldsymbol{A}$ в формате CSR.

б) Вывести на экран зависимость среднеквадратичной нормы абсолютной погрешности от номера итерации[94].

в) Вывести на экран зависимость среднеквадратичной нормы невязки от номера итерации[93].

6. Сделать вывод о скорости сходимости метода Якоби, метода Гаусса–Зейделя и метода сопряжённых градиентов для данной матрицы коэффициентов и выбрать наилучший метод.

4.6. Спектральное и сингулярное разложения

Спектральное разложение (разложение на собственные числа и вектора) и сингулярное разложение, то есть обобщение первого на прямоугольные матрицы, играют настолько важную роль в прикладной линейной алгебре, что тяжело придумать область, где одновременно используются матрицы и не используются указанные разложения в том или ином контексте. В базовой части лабораторной работы мы рассмотрим метод главных компонент (англ. *Principal Component Analysis*, *PCA*), без преувеличения самый популярный метод для понижения размерности данных, основой которого является сингулярное разложение. В продвинутой части мы рассмотрим куда менее очевидное применение разложений, а именно одну из классических задач спектральной теории графов — задачу разделения графа на сильно связанные компоненты (кластеризация на графе).

Задача 4.5 (спектральное и сингулярное разложения)

Требуется (базовая часть)

1. Написать функцию pca(A), принимающую на вход прямоугольную матрицу данных A и возвращающую список главных компонент и список соответствующих стандартных отклонений[95].

2. Скачать набор данных Breast Cancer Wisconsin Dataset: https://archrk6.bmstu.ru/index.php/f/854843[96].

– Указанный набор данных содержит информацию о 569 пациентах с опухолью, которых обследовали на предмет наличия рака молочной железы. В каждом обследовании опухоль была классифицирована экспертами как доброкачественная (*benign*, 357 пациентов) или злокачественная (*malignant*, 212 пациентов) на основе детального исследования снимков и анализов. Дополнительно на основе снимков был автоматически выявлен и задокументирован ряд характеристик опухолей: радиус, площадь, фрактальная размерность и так далее (всего 30 характеристик). Постановку диагноза можно автоматизировать, если удастся создать алгоритм, классифицирующий опухоли исключительно на основе этих автоматически получаемых характеристик. Указанный файл является таблицей, где отдельная строка

[94] Желательно вывести эти зависимости на том же графике, который использовался для методов Якоби и Гаусса–Зейделя.

[95] Для нахождения собственных чисел и векторов вы можете использовать функцию numpy.linalg.eig.

[96] Данные взяты с сайта https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

соответствует отдельному пациенту. Первый элемент в строке обозначает ID пациента, второй элемент — диагноз (M — malignant, B — benign), и оставшиеся 30 элементов соответствуют характеристикам опухоли (их детальное описание находится в файле https://archrk6.bmstu.ru/index.php/f/854842).

3. Найти главные компоненты указанного набора данных, используя функцию pca(A).

4. Вывести на экран стандартные отклонения, соответствующие номерам главных компонент.

5. Продемонстрировать, что проекций на первые две главные компоненты достаточно для того, чтобы произвести сепарацию типов опухолей (доброкачественная и злокачественная) для подавляющего их большинства. Для этого необходимо вывести на экран проекции каждой из точек на экран, используя scatter plot.

6. **Опциональное задание ☒ 1.** Постройте классификатор в полученном пространстве пониженной размерности, используя любой из классических алгоритмов машинного обучения (например, логистическую регрессию или метод опорных векторов) и, при необходимости, кросс-валидацию.

Требуется (продвинутая часть)

1. Построить лапласианы (матрицы Кирхгофа) L для трех графов:
– полный граф G_1, имеющий 10 узлов;
– граф G_2, изображенный на рис. 3;
– граф G_3, матрица смежности которого хранится в файле https://archrk6.bmstu.ru/index.php/f/854844,
где лапласианом графа называется матрица $L = D - A$, где A — матрица смежности и D — матрица, на главной диагонали которой расположены степени вершин графа, а остальные элементы равны нулю.

2. Доказать, что лапласиан неориентированного невзвешенного графа с n вершинами является положительно полуопределенной матрицей, имеющей n неотрицательных собственных чисел, по крайней мере одно из которых равно нулю.

3. Найти спектр каждого из указанных графов, т.е. найти собственные числа и вектора их лапласианов. Какие особенности спектра каждого из графов вы можете выделить? Какова их связь с количеством кластеров[97]?

4. Найти количество кластеров в графе G_3, используя второй собственный вектор лапласиана. Для демонстрации кластеров выведите на графике исходную матрицу смежности и ее отсортированную версию[98].

5. **Опциональное задание ☒ 2.** Реализуйте алгоритм DBSCAN и произведите с помощью него кластеризацию графа G_2.

Вопросы и ответы

Вопрос 1 Какой должен быть размер шрифта текстовых подписей, включенных в состав иллюстрации?

[97] В контексте графов кластером называется подграф с большой плотностью связей. Например, граф, изображенный на рис. 3 имеет три кластера.

Попробуйте взглянуть на собственный вектор, соответствующий второму по величине собственному числу. Его сортировка позволяет выявить не только количество кластеров, но и вершины, принадлежащие конкретному кластеру.

[98] Это можно сделать, например, с помощью функции matplotlib.pyplot.matshow.

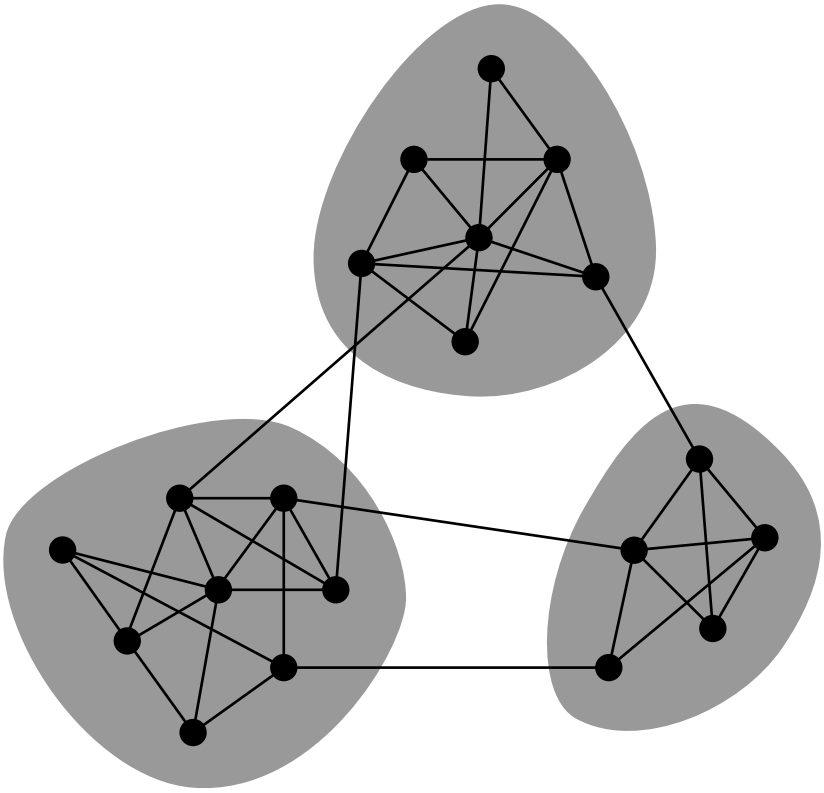


Рис. 3. Граф, содержащий три кластера

Ответ. Шрифт текста на иллюстрациях должен быть сравним со шрифтом подписи к иллюстрации и может быть немногим меньше шрифта основного текста документа.

Комментарий. Разрешение иллюстраций не должно быть ниже 300 dpi, что позволит осуществлять некоторое масштабирование без потери качества текстовых подписей.

Вопрос 2 Могут ли использоваться различные шрифты в одном документе (в части размера, курсива, полужирного, типа)?

Ответ. Нет.

Комментарий. Применение различных шрифтов в одном документе для подготовки основного текста недопустимо и является признаком некомпетентности. Каждый шрифт используется для решения специальный задач: выделение заголовков и подзаголовков (увеличенный, полужирный), написание основного текста (обычный), выделение терминов (курсив), подписи к рисункам, таблицам и листингам (уменьшенный, обычный).

Вопрос 3 Какого размера должна быть одна иллюстрация на странице?

Ответ. Субъективно с точки зрения автора: для определения размера одной иллюстрации по ширине текста на странице следует использовать правило золотого сечения.

Комментарий. В дополнение следует отметить, что размер иллюстрации должен быть минимально возможным, но достаточным для представления необходимой информации. Не следует оставлять на иллюстрациях лишние поля и непропроционально большие пустые пространства.

Вопрос 4 Каким форматам следует отдавать предпочтение при подготовке иллюстраций?

Ответ. Векторным (например, EPS) и лишь затем растровым (JPG, PNG) с расширением не ниже 300 dpi для полутоновых или цветных изображений и не ниже 600 dpi для схем.

Комментарий. Векторные форматы не зависят от размера области представления, позволяют масштабировать изображение с сохранением качества.

Вопрос 5 Насколько допустима вставка чужих иллюстраций в свои документы?

Ответ. Крайне нежелательна.

Комментарий. Если осуществляется вставка чужих иллюстраций, то это следует делать с обязательной ссылкой на первоисточник. В противном случае такое заимствование может расцениваться максимум как плагиат, и как минимум — некомпетентность.

Приложение

Вспомогательный исходный код к варианту 6 лабораторной работы 1

Listing 1. Скрипт для генерации множества точек фрагмента контура множества Мандельброта

```
from typing import Optional, Tuple
import time
import numpy as np
import cv2
import matplotlib
import matplotlib.pyplot as plt

class MandelbrotSetOnGrid():
    """ Генератор множества Мандельброта """
    def __init__(self, resolution_x: int, max_iterations: int = 255, escape_radius: float =
    1000):
        self.resolution_x = resolution_x
        self.max_iterations = max_iterations
        self.escape_radius = escape_radius

    def calculate_mandelbrot_set(self, pt_complex_bounds: Tuple[float, float, float, float]) 
    > np.ndarray:
        c = self._complex_matrix(pt_complex_bounds)
        stability_map = np.empty(c.shape, dtype=np.float64)
        stability_map.fill(self.max_iterations)
        z = 0
        for iteration in range(self.max_iterations):
            z = z ** 2 + c
            z_norm = abs(z)
            subset = z_norm > self.escape_radius
            stability_map[subset] = iteration
            z[subset], c[subset] = 0, 0 # avoid overflow
        stability_map /= self.max_iterations
        self.comp_mode = "np"
        return stability_map

    def _complex_matrix(self, pt_complex_bounds: Tuple[float, float, float, float]) -> np.
    ndarray:
        dx = pt_complex_bounds[1] - pt_complex_bounds[0]
        dy = pt_complex_bounds[3] - pt_complex_bounds[2]
        x = np.linspace(pt_complex_bounds[0], pt_complex_bounds[1], self.resolution_x)
        y = np.linspace(pt_complex_bounds[2], pt_complex_bounds[3], int(np.abs(dy/dx) * self.
    resolution_x))
        return x[np.newaxis, :] + y[:, np.newaxis] * 1j

class MandelbrotDraw(MandelbrotSetOnGrid):
    """ Рисование множества Мандельброта """
    def __init__(self, axs=None, *args, **kwargs):
```

```
40          super().__init__(*args, **kwargs)
41
42          if axs is None:
43              self.fig, self.ax_main = plt.subplots()
44              self.axs = (self.ax_main,)
45          else:
46              self.axs = axs
47              self.ax_main = self.axs[0]
48
49          self.pt_complex_bounds = None
50
51          self.draw_mandelbrot_map((-2., 2., -1., 1.))
52
53      def postproc(self, mandelbrot_map: np.ndarray):
54          pass
55
56      def draw_mandelbrot_map(self, pt_complex_bounds: Tuple[float, float, float, float]):
57          start_time = time.time()
58          for ax in self.axs:
59              ax.clear()
60
61          self.pt_complex_bounds = np.array(pt_complex_bounds)
62          # При отображении данной матрицы, она переворачивается по конвенции отображения картинок(OY
        направлен вниз),
63          # поэтому нужно отражение по вертикали
64          mandelbrot_map = self.calculate_mandelbrot_set(pt_complex_bounds)[::-1,:]
65          im = self.ax_main.imshow(mandelbrot_map, extent=pt_complex_bounds)
66
67          self.ax_main.callbacks.connect('ylim_changed', self._on_ylims_change)
68          self.ax_main.set_title(
69              "Используйте␣инструмент␣ПРИБЛИЖЕНИЯлупа()␣для␣выбора␣области,␣приближать␣можно␣много␣раз.\n"
70              "После␣выбора,␣окно␣можно␣закрыть,␣контур␣сохранится.")
71
72          self.postproc(mandelbrot_map)
73          print(f"Timer␣{time.time()␣- start_time:.3f} sec, mode: {self.comp_mode}", end="\r")
74
75      def _on_ylims_change(self,event_ax):
76          xlim = self.ax_main.get_xlim()
77          ylim = self.ax_main.get_ylim()
78          pt_complex_bounds = np.array((*xlim, *ylim))
79          if self.pt_complex_bounds is None or np.any(pt_complex_bounds != self.
        pt_complex_bounds):
80              self.draw_mandelbrot_map(pt_complex_bounds)
81
82  class MandelbrotContourDraw(MandelbrotDraw):
83      """␣Выделение␣точек␣контура␣множества␣Мандельброта␣"""
84      def __init__(self, debug:bool=False, *args, **kwargs):
85
86          self.debug = debug
87          ax_config = (2,2) if self.debug else (2,1)
88          self.fig, axs = plt.subplots(*ax_config, sharex=True, sharey=True)
89          self.ax_contour = axs.flatten()[-1]
90          self.fig.set_size_inches((15,7))
91
92          # Визуализация N-самых длинных контуров
93          self.contours_draw_num = 10
94          # Шаг (прореживание) точек на контуре для визуализации
95          self.viz_step=10
96
97          super().__init__(axs.flatten(), *args, **kwargs)
98
99      def postproc(self, mandelbrot_map: np.ndarray):
100         image_blur = cv2.GaussianBlur(mandelbrot_map, (7,7), sigmaX=2)
101         _, image_bin = cv2.threshold(image_blur, 0.9, 1, cv2.THRESH_BINARY)
102         self.contours = self._find_contours(image_bin)
103
104         for contour in self.contours[:self.contours_draw_num]:
105             t_grid = np.arange(0, contour.shape[1], self.viz_step, dtype=np.uint32) # сетка
        аргументов
106             if t_grid.size < 2:
107                 continue
108             self.ax_contour.scatter(contour[0,t_grid], contour[1,t_grid], s=10, marker="*")
109         self.ax_contour.set_aspect(1)
110         self.ax_contour.set_title(f"Обработка,␣шаг␣3.␣Точки␣{self.contours_draw_num}␣самых␣
        протяженных␣контуров")
```

```
111
112         # сохраняем самый длинный контур в файл
113         np.savetxt("contour.txt", self.contours[0].T)
114
115         if self.debug:
116             self.axs[1].imshow(image_blur, extent=self.pt_complex_bounds)
117             self.axs[1].set_title("Обработка, шаг 1. Размытое изображение")
118             self.axs[2].imshow(image_bin,  extent=self.pt_complex_bounds)
119             self.axs[2].set_title("Обработка, шаг 2. Бинаризованное изображение")
120
121     def _find_contours(self, image_bin:np.ndarray):
122         contours_pix, _ = cv2.findContours(image=image_bin.astype(np.uint8), mode=cv2.
    RETR_LIST, method=cv2.CHAIN_APPROX_NONE)
123         h, w = image_bin.shape[:2]
124         # OpenCV детектирует точки на границах(баг), исключаем их из рассмотрения
125         pad = 1
126         contours_pix_clean = []
127         for contour_pix in contours_pix:
128             contour_pix = contour_pix[:,0,:].T
129             subset = (contour_pix[0] >= pad) & (contour_pix[1] >= pad) & (contour_pix[0] < w-
    pad) & (contour_pix[1] < h-pad)
130             bounding_contour_ids = np.where(subset[:-1] != subset[1:])[0]
131             if bounding_contour_ids.size > 0:
132                 splits = np.r_[0, bounding_contour_ids, contour_pix.shape[1]-1]
133                 for i in range(splits.size - 1):
134                     if subset[splits[i]+1]:
135                         contours_pix_clean.append(contour_pix[:, splits[i]:splits[i+1]+1])
136             else:
137                 contours_pix_clean.append(contour_pix)
138
139         # Преобразование_ пикселей(исходные координаты найденных контуров) в комплексную плоскость
140         b = self.pt_complex_bounds
141         M_im2complex = np.array((
142             ((b[1] - b[0])/w,                0, b[0]),
143             (               0, -(b[3] - b[2])/h, b[3]),# координаты изображения "растут" в обратном
    направлении, нужен "-"
144         ))
145         contours_complex = []
146         for contour_pix in contours_pix_clean:
147             contour_pix_h = np.r_[contour_pix, np.ones((1, contour_pix.shape[1]))]
148             contours_complex.append( M_im2complex @ contour_pix_h )
149         # Сортируем по длине(количеству точек в цепочке)
150         return sorted(contours_complex, key = lambda contour: contour.shape[1], reverse=True)
151
152 if __name__ == '__main__':
153     print("ВНИМАНИЕ! Если возникает ошибка с QT5, замените в коде ниже 'Qt5Agg' to 'TKAgg'")
154     matplotlib.use("Qt5Agg")
155     md = MandelbrotContourDraw(resolution_x=1024)
156     #md = MandelbrotDraw(resolution_x=1024)
157     plt.show()
```

Аббревиатуры

СЛАУ система линейных алгебраических уравнений. 14

Список литературы

[1] Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140. URL: https://archrk6.bmstu.ru/f/854491. (облачный сервис кафедры РК6).

[2] Соколов А.П., Першин А.Ю. Инструкция по выполнению лабораторных работ (общая). Москва: Соколов, А.П., 2018-2023. С. 19. URL: https://archrk6.bmstu.ru/f/790009. (облачный сервис кафедры РК6).