Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Высшая школа прикладной математики и вычислительной физики

Вычислительные комплексы

Лабораторная работа №5

Работу выполнила:
Т. В. Алпатова
Группа:
3630102/70201
Преподаватель:
А. Н. Баженов

Санкт-Петербург
2021

Содержание

Список иллюстраций

1. Постановка задачи

Решить с помощью субдифференциального метода Ньютона переопределённую систему размера 256×36. Исследовать систему.

Исходные данные: имеем вектор длины 256 распределения светимостей (то есть решение), показания детектора (правая часть, вектор b длины 36) и матрицу длин хорд (матрица 256×36).

2. Теория

Пусть имеется ИСЛАУ $Cy = d, y \in \mathbb{KR}^n$.

Процедура субградиентного метода Ньютона состоит в следующем:

1. Задаем начальное приближение $x^{(0)} \in \mathbb{R}^{2n}$, релаксационный параметр $\tau \in (0; 1]$ и точность $\varepsilon > 0$
2. Строим отображение $\mathcal{G}$:

$$\mathcal{G}(x) = \text{sti } (\mathbf{C}\text{sti }^{-1}(x)) - \text{sti } (\mathbf{d})$$

3. Вычисляем субградиент D^{k-1} отображения $\mathcal{G}$ в точке $x^{(k-1)}$
4. $x^{(k)} = x^{(k-1)} - \tau(D^{k-1})^{-1}\mathcal{G}(x^{(k-1)})$
5. Итерационная процедура повторяется, пока $||x^{(k)} - x^{(k-1)}|| \geq \varepsilon$. В качестве ответа возвращается $sti^{-1}(x^{(k)})$

Начальное приближение можно найти, решив 'среднюю систему':

$$\text{mid}\mathbf{C}\ \dot{x}^{(0)} = \text{sti}\mathbf{d}$$

3. Анализ матрицы

3.1. Структура матрицы

Матрица имеет блочную структуру. Ее следует разделить на две матрицы размера 18*128, что значительно облегчит задачу

Рисунок 1. Исходная матрица

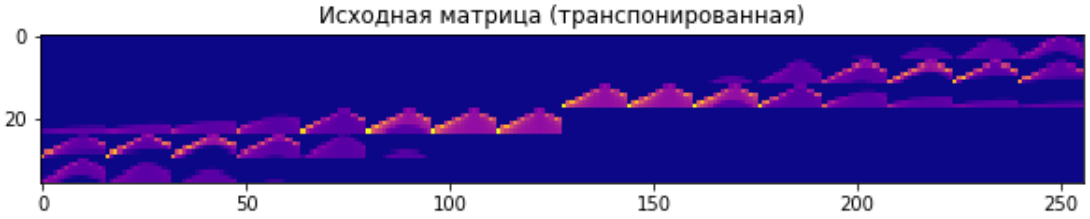
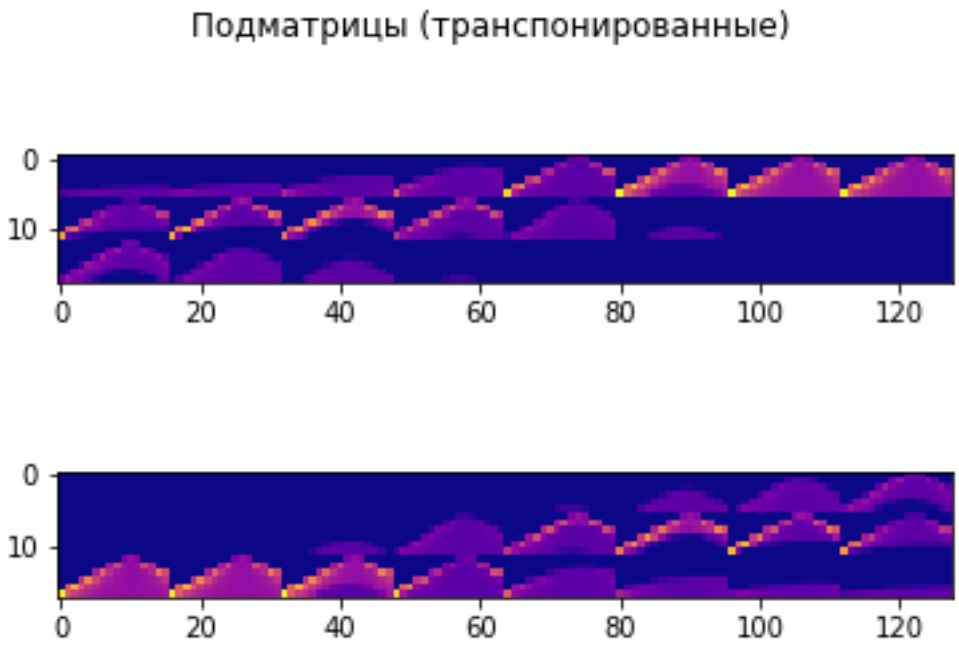


Рисунок 2. Разбиение на две подматрица

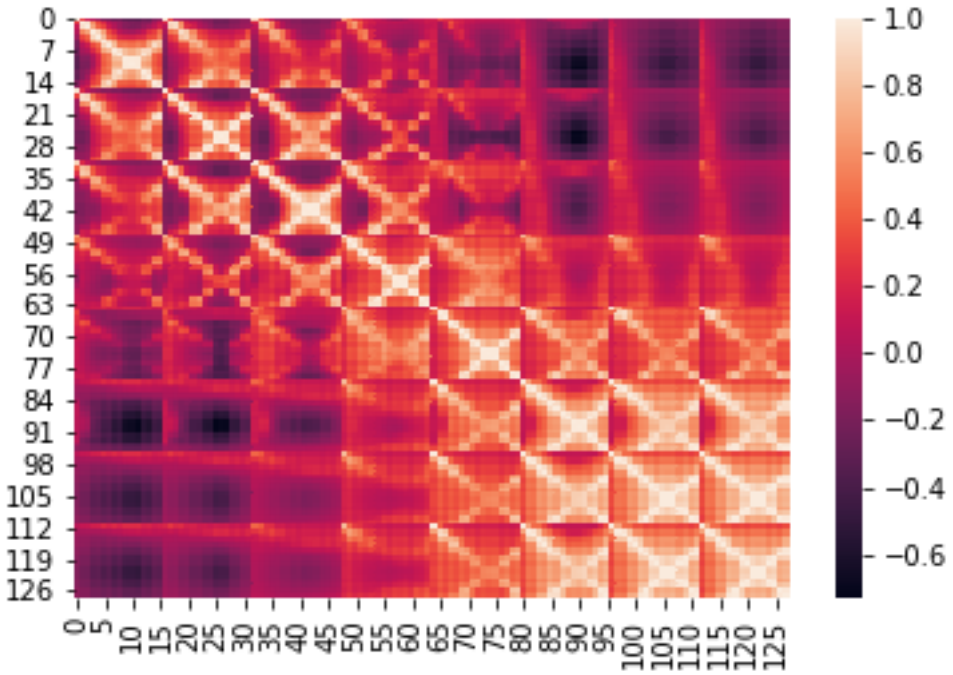


3.2. Корреляционный анализ строк

Структура матрицы такова, что ее строки сильно коррелируют друг с другом, что плохо виляет на решение, так как определитель матрицы получается практически равным нулю. Проведем анализ и попробуем выделить строки, которые бы мало коррелировали между собой.

Посмотрим на матрицу корреляции строк.

Рисунок 3. Матрица корреляции строк



Сразу бросается в глаза структура матрицы корреляции. Вся она разбилась на прямоугольники схожие друг с другом. Прямоугольники имеют размер 15*15. Внутри прямоугольников наблюдается некоторая закономерность. Некоторые строки достаточно сильно коррелируют между собой (имеют коэффициент корреляции по модулю близкий к единице).

Можно отметить, что на границах каждого такого прямоугольника (каждое 15 значение) корреляция строк визуально близка к нулю. Такая гипотеза проверялась на практике. Была взята каждая 15 строка, но получившаяся матрица содержала много нулевых строк, что конечно не является хорошим результатом.

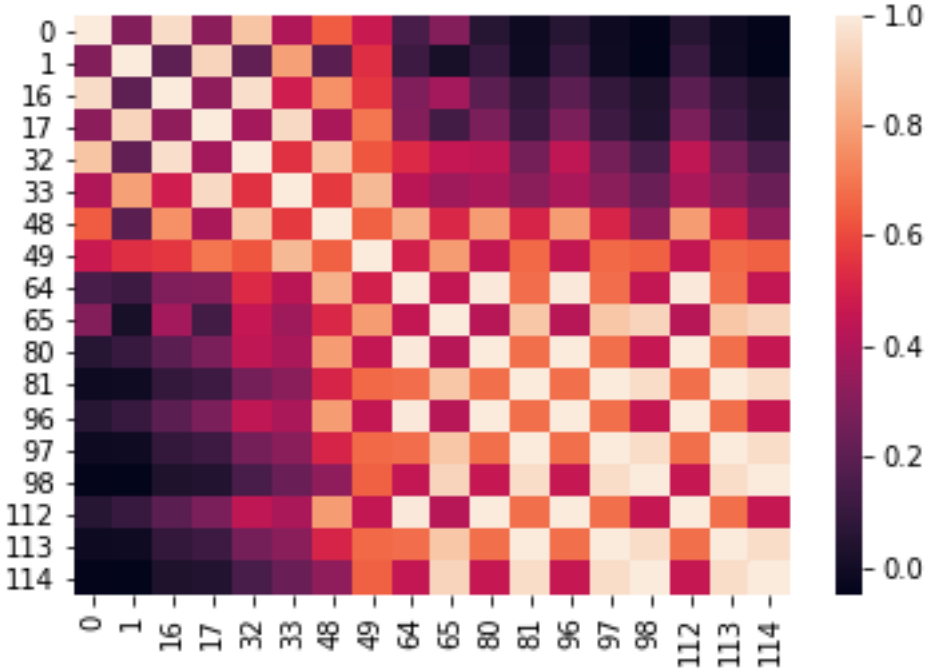
Попробуем выбрать набор из 18 строк, которые наименьшим образом коррелируют между собой не на глаз, а автоматически.

Делать это можно несколькими способами. Можно перебрать все возможные комбинации 18 из 128 строк и посмотреть, у какого набора сумма квадратов корреляций наименьшая. Такой алгоритм совершает полный перебор вариантов. Он имеет огромную вычислительную сложность и не может быть применен на персональном компьютере.

Попробуем поступить по-другому. Будем строить матрицу квадратов корреляций, затем суммировать ее по строкам. После чего отбросим строку с наибольшей такой суммой. И повторим все сначала. Будем повторять так, пока не останется нужное количество строк. Такие строки будут наименее коррелированы между собой.

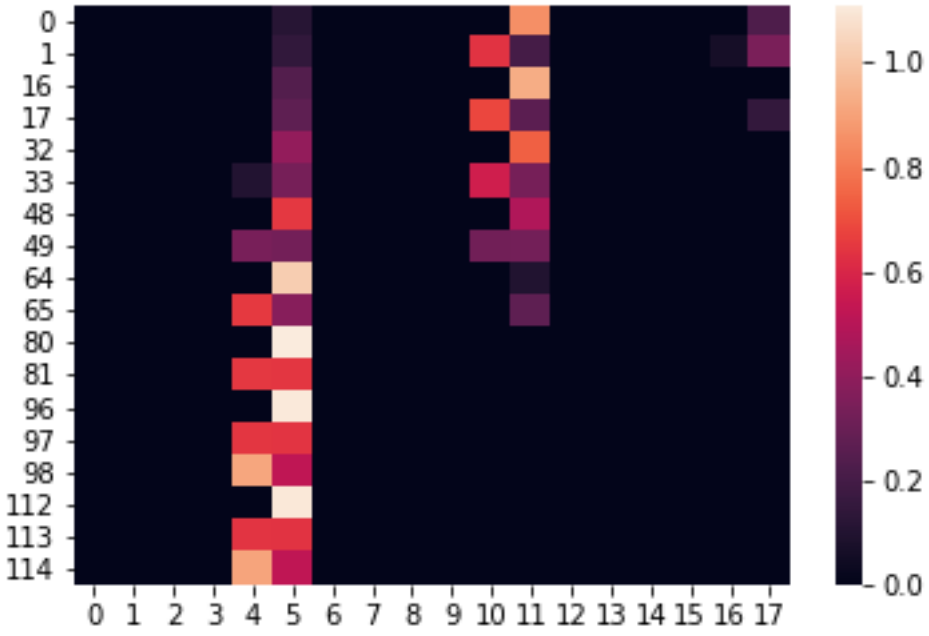
Получим следующую матрицу корреляции и подматрицу исходной матрицы

Рисунок 4. Наименее коррелированные строки матрицы



Кажется, что по сравнению с исходной матрицей корреляции эта матрица содержит не большее количество сильно коррелированных строк (в процентном соотношении)

Рисунок 5. Наименее коррелированная подматрица



Можно отметить, что алгоритм выбрал практически те же строки, что и мы, когда анализировали внешний вид матрицы корреляции. Тем не менее, полученная матрица

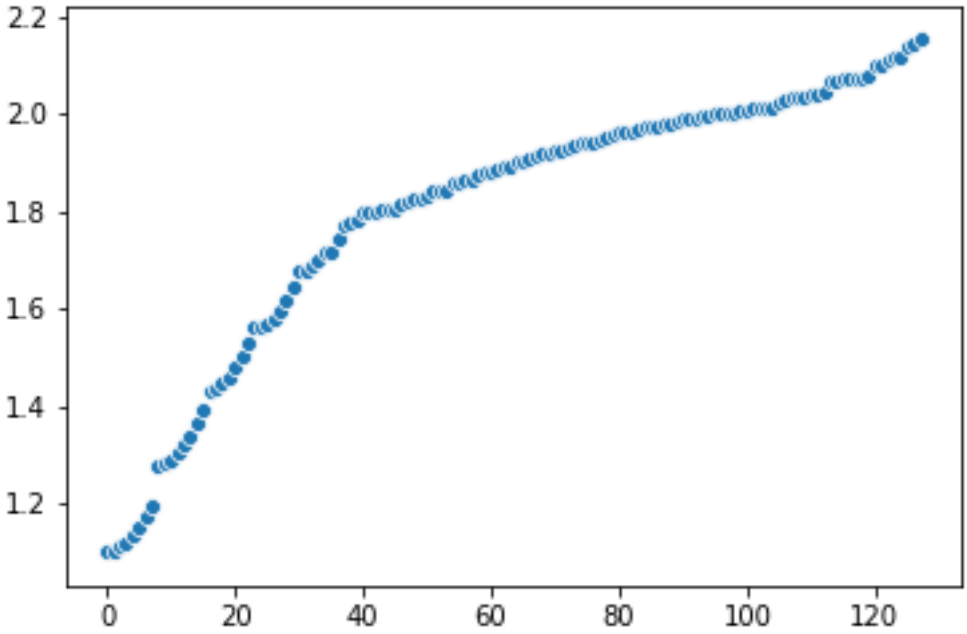
содержит много нулевых столбцов. Такое решение нам не подходит.

Таким образом, корреляционный анализ не дал никаких улучшений.

3.3. Сумма элементов в строках

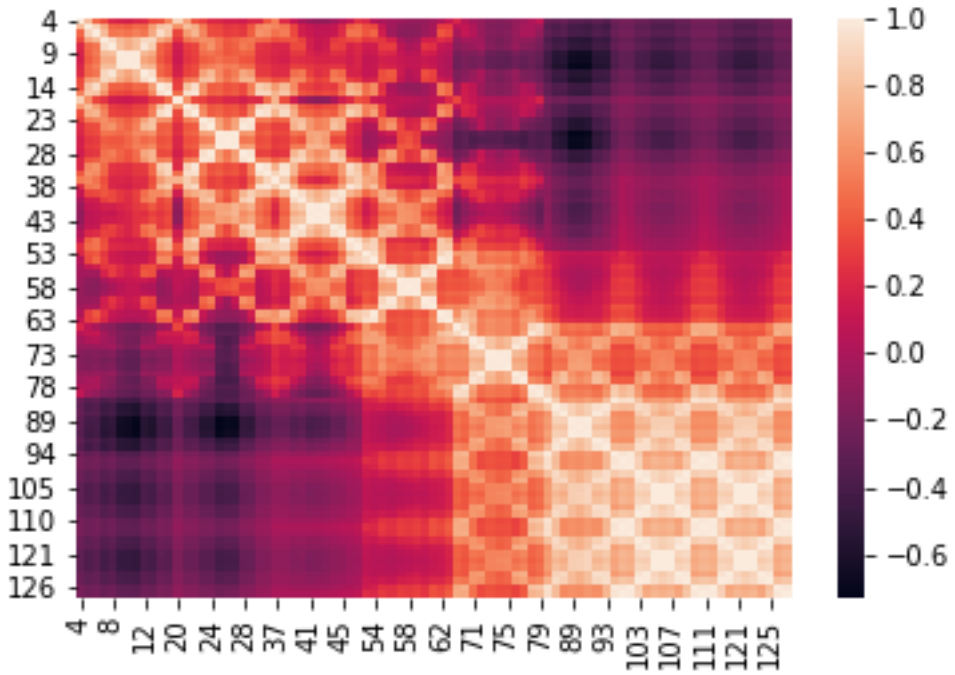
Попробуем другой подход. Отранжируем строки по сумме элементов в строке и изобразим график этих сумм.

Рисунок 6. График сумм строк матрицы



По графику можно отметить, что наблюдается перегиб в районе 40 значения (при сумме элементов в строке $\approx$ 1.8). Отбросим строки, у которых сумма меньше этого значения и снова построим матрицу корреляции

Рисунок 7. Матрица корреляции при отбрасывании строк с маленькой суммой элементов



Как видим, матрица корреляции, не сильно поменяла вид. Сгладились границы между квадратными областями.

Попробуем взять 18 строк, у которых сумма наибольшая(> 2.04). Получается матрица с одним нулевым столбцом. Отбросим этот столбец и любую строку так. Определитель такой матрицы практически равен нулю ($1.3 * 10^{-32}$). Решим такую систему субдифференциальным методом Ньютона. Получаем следующее

Рисунок 8. Подматрица с наибольшей суммой строк

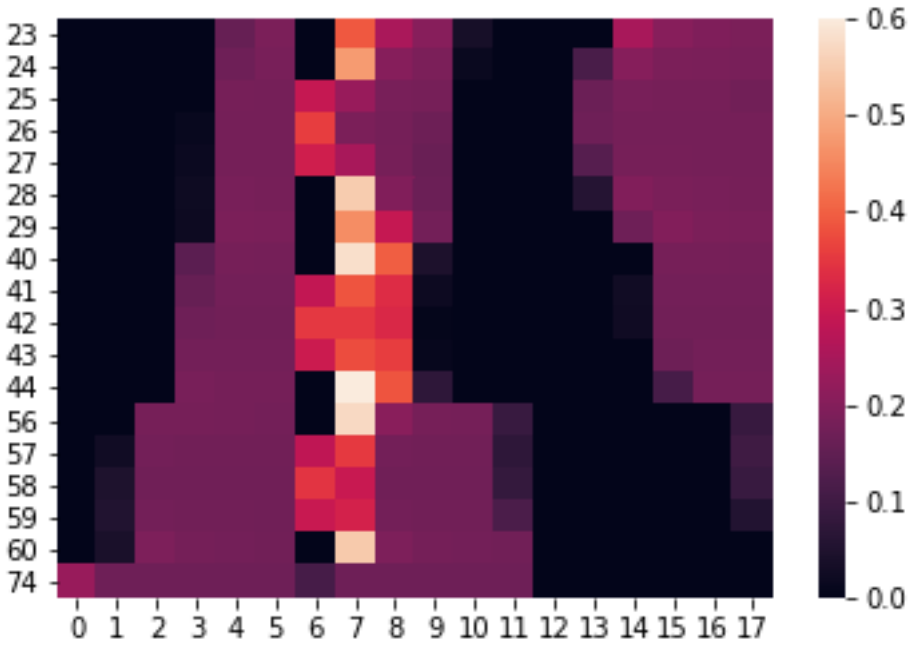
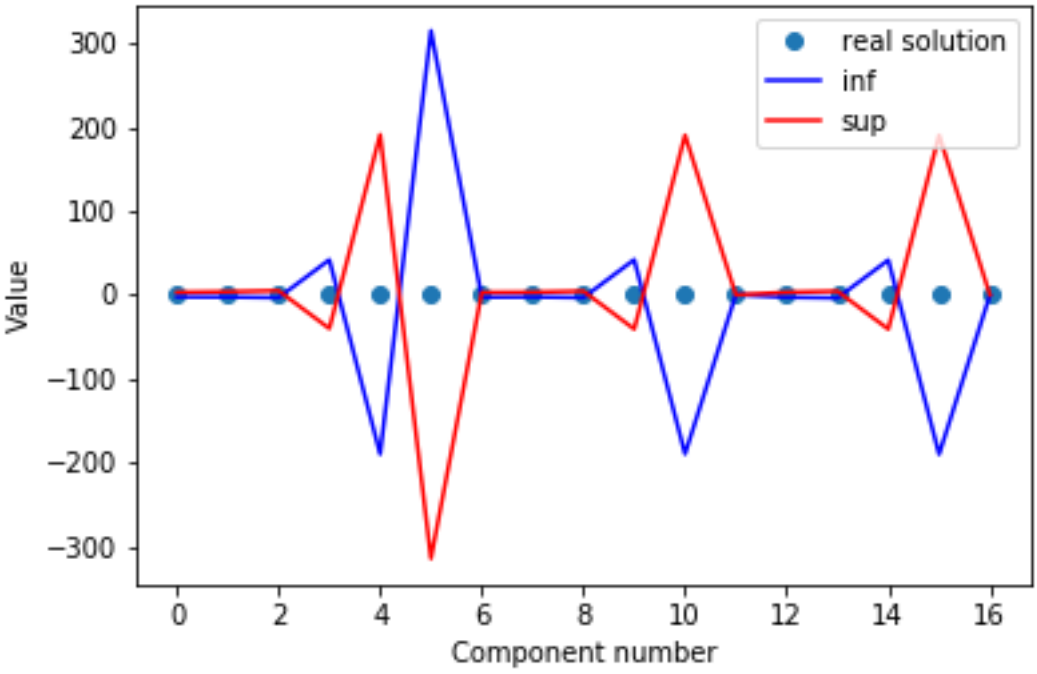


Рисунок 9. Решение для подматрица с наибольшей суммой строк



По графику видно, что для некоторых точек разница между inf и sup достаточно велика.

3.4. Число переменных

Самым простым способом было бы перебрать всевозможные варианты выбора строк и найти среди них те, которые дают матрицу с наибольшим по модулю определителем. Такой вариант конечно не подходит, поскольку переборный алгоритм работал бы долго. Так же возможна ситуация, что даже в случае полного перебора самый большой определить будет близок к нулю. Это значит, что строки сильно коррелируют между собой (что подтверждает и предыдущий раздел).

Все это наводит на мысль, что использование всех 18 переменных не рационально. Проведем анализ и попробуем уменьшить количество переменных. Для этого посмотрим на следующий переборный алгоритм

Пусть мы решили выбрать $2 \leq k \leq 18$ переменных. Тогда вариантов выбрать переменные C_{18}^k, а вариантов выбрать строки из исходной матрицы, чтобы в итоге получить квадратную C_{128}^k. Так же мы хотим для каждой такой матрицы посчитать определить, чтобы найти подматрицу с наибольшим по модулю определителем. Вычисление определителя занимает порядка $O(k^3)$ времени. Итого общая сложность алгоритма $O(k^3 C_{18}^k C_{128}^k)$. Такая величина быстро растет с увеличением k. Запустить этот алгоритм для значений k близких к правой границе значений (скажем 16, 17, 18) лишено всякого смысла. Тем не менее для анализа задачи мы можем посмотреть на работу алгоритма для небольших k. Далее в таблице приведен результат работы алгоритма для таких k.

k	max_det	*номера строк*	*номера переменных*	*время вычисления (сек)*
3	0.0389	0,1,2	10,11,17	0.22
4	0.0476	0,1,2,3	9,10,11,16	3.93
5	0.0046	0,1,2,3,4	9,11,15,16,17	92.65
6	0.0023	0,1,2,3,4,5	8,9,10,11,15,16	2246.32 ($\approx 37min$)

Рассмотрим эту таблицу подробнее. В выборе переменных и в выборе строк наблюдается некоторая закономерность.

- При фиксированном k наилучший результат дает взятие первых k строк
- В выборе переменных так же наблюдается некоторая закономерность, хоть и не такая явная.Так, алгоритм всегда выбирает 11 переменную, три раза выбирает 9, 10 и 16.

Теперь, когда мы нашли некоторую закономерность, попробуем модифицировать алгоритма. Зафиксируем взятие первых 5 строк. Далее снова запустим алгоритм и увеличим k.

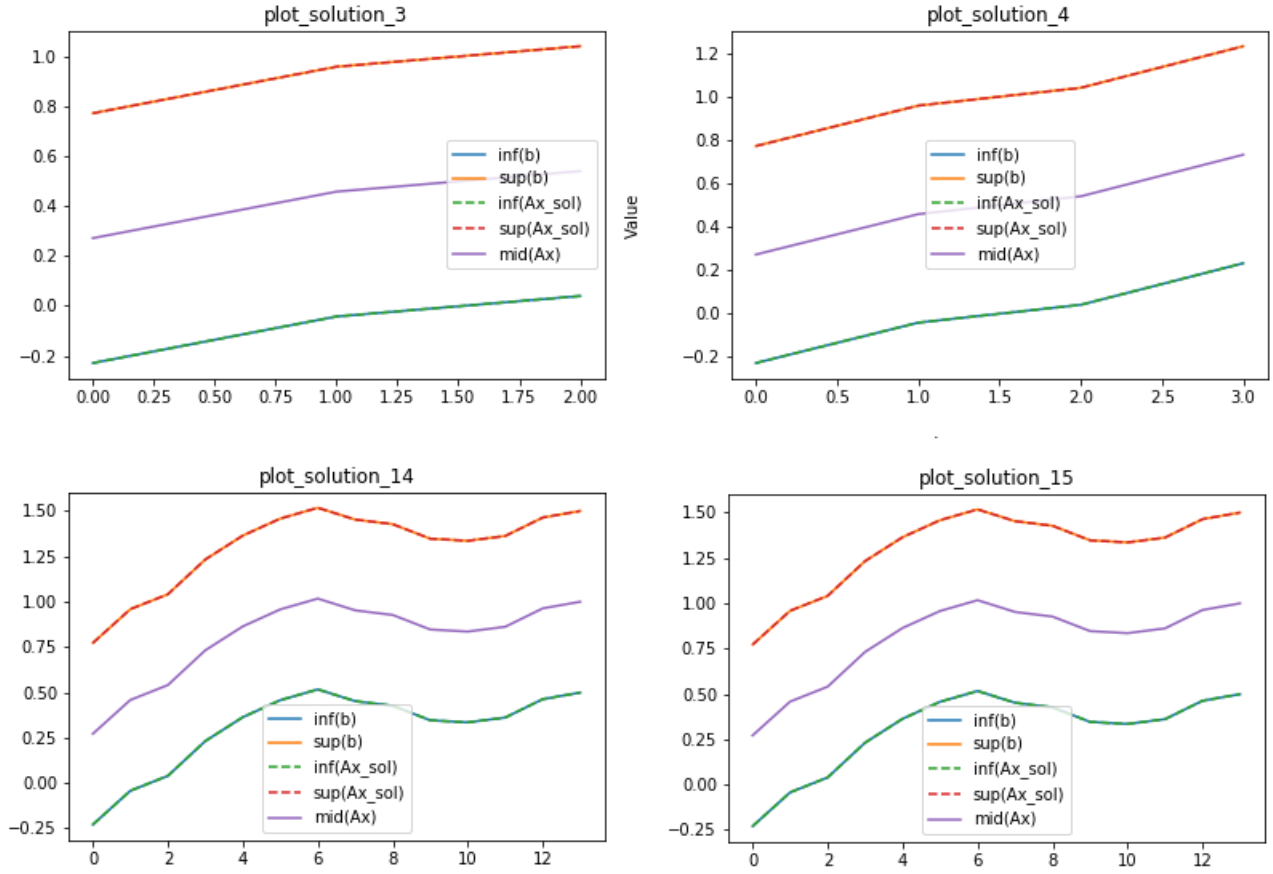
k	max_det	*номера строк*	*номера переменных*	*время вычисления (сек)*
7	$2.4 * 10^{-4}$	6	8, 9, 10, 11, 14, 15, 16	0.88
8	$2.8 * 10^{-5}$	6,7	8, 9, 11, 13, 14, 15, 16, 17	2.89
9	$2.3 * 10^{-7}$	6,7,8	7, 8, 9, 11, 13, 14, 15, 16, 17	1.46
10	$5.7 * 10^{-8}$	6,7,8,9	7, 8, 10, 11, 12, 13, 14, 15, 16, 17	4.25
11	$1.7 * 10^{-9}$.	6,7,8,9,10	6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17	67.05
12	$1.6 * 10^{-11}$.	6,7,8,9,10,11	6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17	1647.55 ($\approx 27min$)

Закономерности, отмеченные ранее верны и для этой таблицы. Отметим, что с увеличением числа переменных значение максимального определителя уменьшается на порядок, а иногда и на два.

Мы знаем какие строки и столбцы надо взять, чтобы для фиксированного k получить определитель наибольшей размерности. Будем пробовать решать систему в для разного количества переменных и сравнивать с решением.

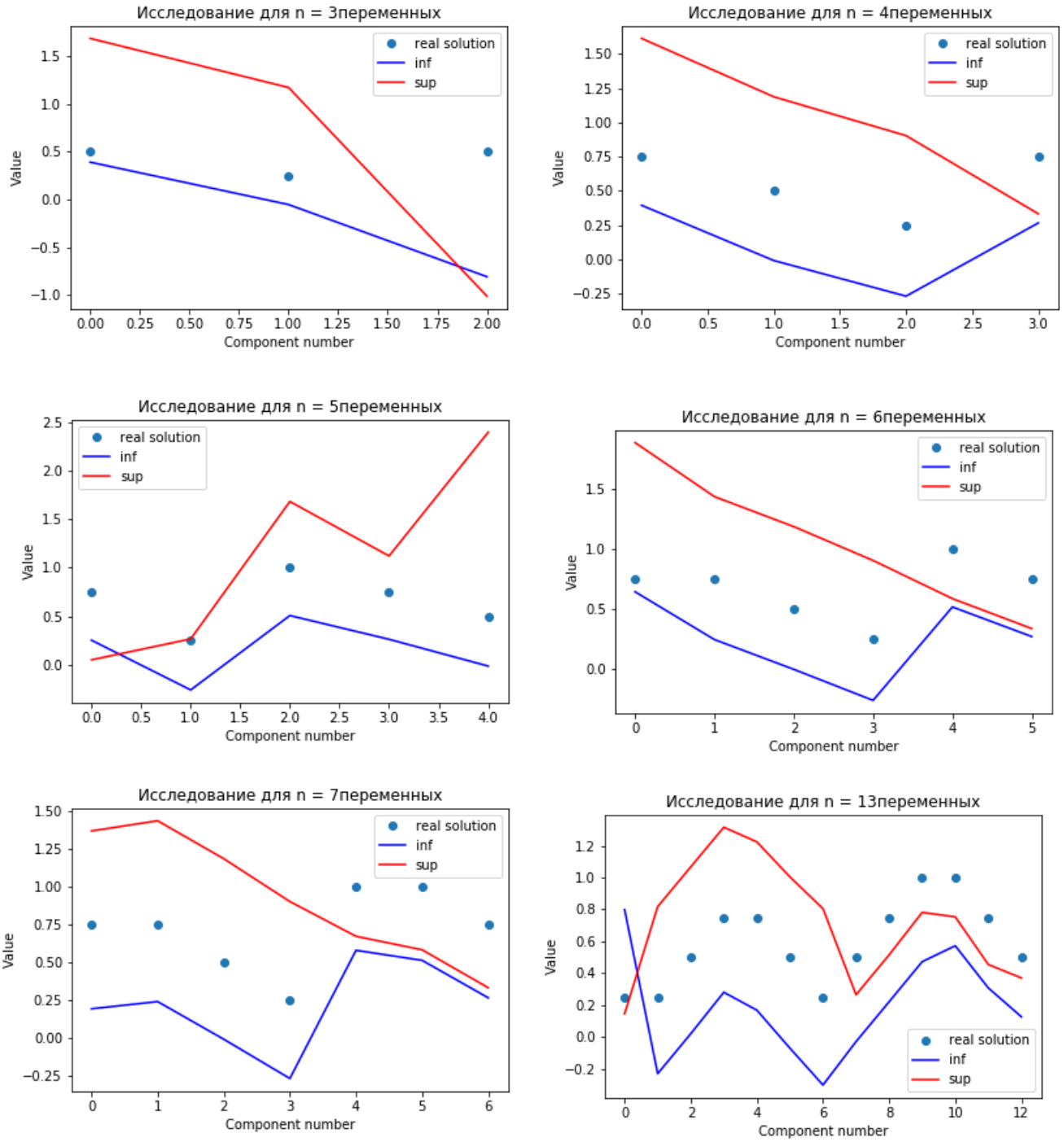
Представим здесь часть получившихся графиков

Рисунок 10. Значение правой части



Для всех k значения $inf(Ax_{sol}), sup(Ax_{sol})$ и $inf(b), sup(b)$ совпадают.

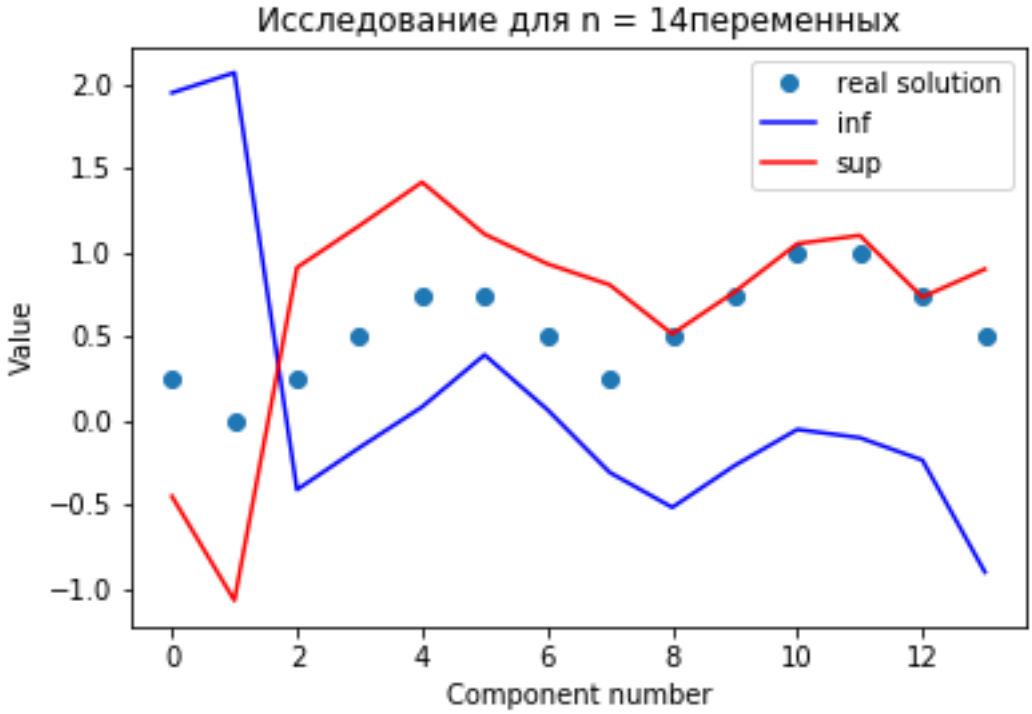
Рисунок 11. Решение для разных n



Как мы можем заметить, не все решения лежат в нужном интервале. Так же, в тех случаях, когда решение не лежит в полученном интервале, расстояние между $inf(x)$ и $sup(x)$ меньше по сравнению с интервалами, в которых лежит решение.

Такое явление наблюдалось для $k \leq 13$. Отдельного внимания заслуживает график для $k = 14$.

Рисунок 12. Решение для n = 14



Как видим, в этом случае все истинные решения лежат внутри полученного интервала. Это наводит на мысль, что использование менее, чем 14 переменных может быть недостаточно.

4. Выводы

По проведенным экспериментам можно заключить следующее:

- корреляционный анализ строк в данной работе не дал никаких значимых результатов
- отбрасывание элементов с наименьшей суммой в строках позволяет решить систему, но размах некоторых интервалов слишком велик. Возможно, имеет смысл изучить этот вопрос. Например, рассмотреть меньшее количество строк с наибольшей суммой. Или же взять порог на сумму в строке, а далее из полученного набора строк составлять матрицы рандомно. Таким образом, при пересечении таких решений возможно уменьшение ширины интервала.
- при анализе числа переменных также можно провести еще эксперименты. Например, с помощью переборного алгоритма выбирать матрицы у которых определитель не равен нулю и далее анализировать решение. Такой анализ опять же возможен только для небольших k.

5. Литература

А.Н. Баженов. Интервальный анализ. Основы теории и учебные примеры: учебное пособие. — СПб. 2020

6. Приложение

Репозиторий с кодом: https://github.com/atani20/comp_complex/tree/master/coursework