Алгоритмы и программные средства решения тропических линейных векторных уравнений

Дядичкин Михаил, гр. 422

Санкт-Петербургский государственный университет
Математико-механический факультет
Кафедра статистического моделирования

Научный руководитель: профессор, д.ф.-м.н. Н.К. Кривулин
Рецензент: старший преподаватель, кафедра информационно-аналитических систем, К.К. Смирнов

6 июня 2023 г.

Введение: задача планирования расписания поездов

Задача планирования расписания поездов.

- Две группы I и J конечных станций и группа K промежуточных станций пересадки.
- Можно доехать от станции $i \in I$ до станции $k \in K$ и от станции $j \in J$ до станции $k \in K$.
- Проехать от i до j без пересадки нельзя.
- Каждый поезд отправляется из k по обратному маршруту не раньше, чем на станцию k прибудут пассажиры всех других поездов, направляющихся в k.

Для решения необходимо учитывать следующие условия:

- Временные ограничения: каждый поезд должен прибывать и отправляться из каждой станции в определенное время, учитывая расписание других поездов.
- Минимизация задержек: необходимо минимизировать задержки прибытия и отправления каждого поезда на каждой станции.

Введение: формальная постановка задачи

- a_{ki} — (известное) время движения поезда от станции i до станции k,
- b_{kj} — (известное) время движения поезда от станции j до станции k,
- x_i — (неизвестное) время отправления поезда со станции i,
- y_j - (неизвестное) время отправления поезда со станции j.

Поезда со станций группы I прибудут на станцию k из K к моменту времени $s_k = \max_i(a_{ki} + x_i)$.
Поезда со станций группы J прибудут на станцию k из K к моменту времени $t_k = \max_j(b_{kj} + y_j)$.
Для обеспечения возможности пересадки и сокращения времени ожидания прибывших на станцию k пассажиров со станций I и J необходимо минимизировать $\max_k |s_k - t_k|$.
Решение задачи сводится к поиску x_i, y_j, которые обеспечивают точное или приближенное решение системы уравнений

$$\begin{cases} \max_i(a_{1i} + x_i) & = \max_j(b_{1j} + y_j), \\ & \vdots \\ \max_i(a_{mi} + x_i) & = \max_j(b_{mj} + y_j), \end{cases}$$

где m – число станций пересадки.

Тропическая математика

Пусть X — множество с операциями сложения $\oplus$ и умножения $\otimes$. $\langle X, \oplus, \otimes \rangle$ является коммутативным полукольцом с нулем и единицей, в котором:

- сложение идемпотентно $(x \oplus x = x)$,
- каждый ненулевой элемент имеет обратный по умножению.

Обозначим нулевой и единичный элементы полукольца символами $\mathbb{0}$ и $\mathbb{1}$.
На множестве определено отношение $\leq$ частичного порядка.
Выполняется свойство монотонности сложения и умножения.
Положим $X_+ = X \setminus \{\mathbb{0}\}$. Тогда для любого $x \in X_+$ и целого $p > 0$:

$$\mathbb{0}^p = \mathbb{0}, \quad x^0 = \mathbb{1}, \quad x^p = x^{p-1} \otimes x = x \otimes x^{p-1}, \quad x^{-p} = (x^{-1})^p.$$

Операция возведения в целую степень может быть естественным образом расширена на полукольцо для случая степени с рациональным показателем.
В данной работе используется полукольцо

$$\mathbb{R}_{\max,+} = \{\mathbb{R} \cup \{-\infty\}, \max, +\}.$$

Определения

Определение (Сопряженно транспонированный вектор)

Для вектора $x = (x_1, \ldots, x_n)^{\mathrm{T}}$ *определим вектор* $x^- = (x_1^-, \ldots, x_n^-)$, *где* $x_i^- = x_i^{-1}$, *если* $x_i \neq \mathbb{0}$, *и* $x_i^- = \mathbb{0}$ *в противном случае. Такой вектор* x^- *называется сопряженно транспонированным вектором.*

Матрица A — регулярная, если она не имеет нулевых строк и столбцов. Если матрица A является регулярной, определим величину

$$\Delta(A, b) = (A(b^- A)^-)^- b.$$

Пусть матрица A состоит из столбцов $(a_1, \ldots, a_n)$, тогда введем обозначение для линейной оболочки столбцов матрицы $\mathrm{A} = \operatorname{span}\{a_1, \ldots, a_n\}$.

Лемма (Кривулин, 2009)

Для любой матрицы A *и вектора* $b > 0$ *выполняется:*

$$\rho(\mathrm{A}, b) = \sqrt{\Delta(A, b)}.$$

Если $\Delta(A, b) < \infty$, то минимум величины $\rho(Ax, b)$ достигается при $x = \sqrt{\Delta(A, b)}(b^- A)^-$.

Постановка задачи

Система уравнений

$$\begin{cases} \max_i(a_{1i} + x_i) & = \max_j(b_{1j} + y_j), \\ & \vdots \\ \max_i(a_{mi} + x_i) & = \max_j(b_{mj} + y_j), \end{cases}$$

где $1 \le i \le n_1$, $1 \le j \le n_2$.
Перепишем систему в терминах полукольца $\mathbb{R}_{\max,+}$

$$\begin{cases} a_{11}x_1 \oplus \ldots \oplus a_{1n_1}x_{n_1} & = b_{11}y_1 \oplus \ldots \oplus b_{1n_2}y_{n_2}, \\ & \vdots \\ a_{m1}x_1 \oplus \ldots \oplus a_{mn_1}x_{n_1} & = b_{m1}y_1 \oplus \ldots \oplus b_{mn_2}y_{n_2}. \end{cases}$$

Система равносильна матричному уравнению

$$Ax = By,$$

где матрицы A и B имеют размерности $m \times n_1$, $m \times n_2$ соответственно, вектора x и y имеют размерности n_1 и n_2.

Основная часть

В рамках данной работы

1. было найдено аналитическое решение уравнения $Ax = Bx$ для матриц размерности $1 \times n$ и $m \times 1$,
2. была написана программа, основанная на алгоритме, описанном в (Кривулин, 2023),
3. была продемонстрирована работа программы в двух случаях:
   - решение существует, его поиск,
   - решения не существует, необходимо найти оптимальные значения,
4. были исследованы возникающие зависимости для разных размерностей матриц,
5. было реализовано графическое представление:
   - линейные оболочки матриц 3×3,
   - работа алгоритма для матриц 2×2,
   - работа алгоритма для матриц 3×3.

Алгоритм (Кривулин, 2023)

Входные данные: A, B. Выходные данные: Δ_*, x^*, y^*.

1. $i = 0$, выбрать вектор x_0
2. Вычислить
$$\Delta_i = (B((Ax_i)^- B)^-)^- Ax_i, \quad y_{i+1} = \sqrt{\Delta_i}((Ax_i)^- B)^-$$
3. Если $\Delta_i = 0$ или $y_{i+1} = y_j$ для какого-то $j < i$, то
$$\Delta_* = \Delta_i, \quad x^* = x_i, \quad y^* = y_{i+1}$$
и закончить, иначе $i = i + 1$
4. Вычислить
$$\Delta_i = (A((By_i)^- A)^-)^- By_i, \quad x_{i+1} = \sqrt{\Delta_i}((By_i)^- A)^-$$
5. Если $\Delta_i = 0$ или $x_{i+1} = x_j$ для какого-то $j < i$, то
$$\Delta_* = \Delta_i, \quad x^* = x_{i+1}, \quad y^* = y_i$$
и закончить, иначе $i = i + 1$
6. Перейти к шагу 2

Пример 1: решение существует

Входные данные:

$$A = \begin{pmatrix} 103 & 150 & 20 \\ 40 & 13 & 50 \\ 78 & 150 & 100 \end{pmatrix}, B = \begin{pmatrix} 270 & 50 & 140 \\ 40 & 2 & 80 \\ 80 & 100 & 190 \end{pmatrix}.$$

Начальный вектор: $x_0 = (150, 50, 100)^{\mathrm{T}}$.

Ход алгоритма:

1. $i = 0$: $\Delta_0 = 60$, $y_1 = (13, 158, 68)^{\mathrm{T}}$
2. $i = 1$: $\Delta_1 = 25$, $x_2 = (132.5, 120.5, 122.5)^{\mathrm{T}}$
3. $i = 2$: $\Delta_2 = 0$, $y_3 = (0.5, 170.5, 80.5)^{\mathrm{T}}$

Решение: $x^* = (132.5, 120.5, 122.5)^{\mathrm{T}}$, $y^* = (0.5, 170.5, 80.5)^{\mathrm{T}}$.

Пример 2: решения не существует

Входные данные:

$$A = \begin{pmatrix} 103 & 150 & 20 \\ 40 & 13 & 50 \\ 78 & 150 & 100 \\ 24 & 221 & 42 \\ 21 & 42 & 57 \end{pmatrix}, B = \begin{pmatrix} 270 & 50 & 140 \\ 40 & 2 & 80 \\ 80 & 100 & 190 \\ 17 & 88 & 534 \\ 342 & 35 & 1 \end{pmatrix}.$$

Начальный вектор: $x_0 = (150, 50, 100)^{\mathrm{T}}$.

Ход алгоритма:

1. $i = 0$: $\Delta_0 = 75$, $y_1 = (-133.5, 165.5, -225.5)^{\mathrm{T}}$
2. $i = 1$: $\Delta_1 = 48$, $x_2 = (136.5, 89.5, 141.5)^{\mathrm{T}}$
3. $i = 2$: $\Delta_2 = 48$, $y_3 = (-119.5, 165.5, -199.5)^{\mathrm{T}}$
4. $i = 3$: $\Delta_3 = 48$, $x_4 = (136.5, 89.5, 141.5)^{\mathrm{T}}$

Оптимальные значения:
$x^* = (136.5, 89.5, 141.5)^{\mathrm{T}}, y^* = (-119.5, 165.5, -199.5)^{\mathrm{T}}$.

Зависимость числа итераций от размерности матриц

- Зависимость количества итераций от размерности матриц является мерой эффективности алгоритма
- Алгоритм может заканчивать работу за разное количество итераций

Изучим зависимости между количеством строк и столбцов матриц A и B при решении уравнения $Ax = By$, и количеством итераций алгоритма для поиска решения.
Реализация:

- Выбирались размерности матриц
- Значения матриц генерировались как случайные равновероятные целые числа от 1 до 100
- Моделировалась работа алгоритма тысячу раз
- Производился анализ полученных результатов

Результаты. Матрицы 3×3

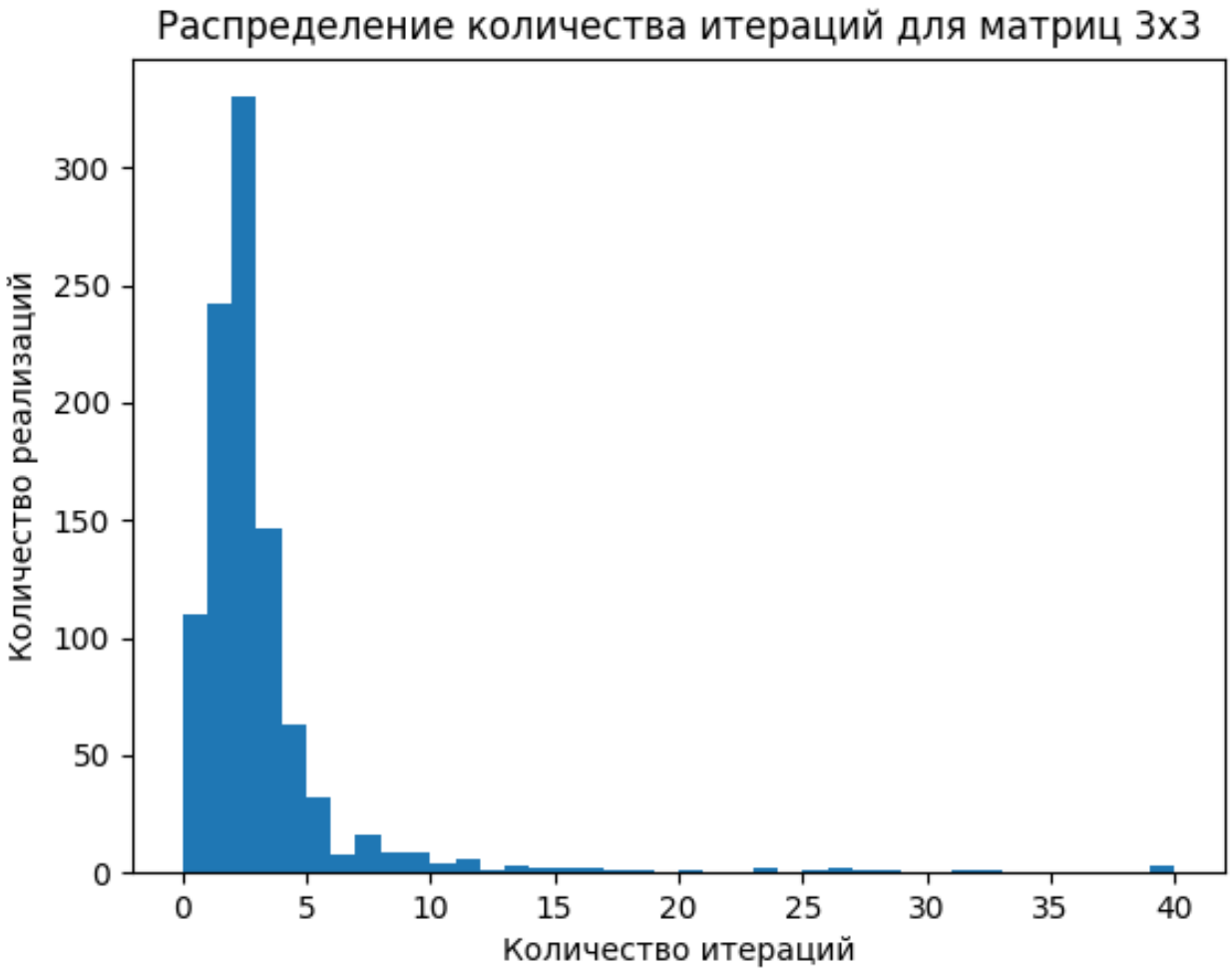

Рис. 1: Гистограмма количества итераций

Результаты, выводы

Результаты:

Размер матриц	3×3	4×4	5×5
Решения не были найдены	359	370	397
Решения были найдены	641	630	603
Завершение из-за ограничения итераций	3	43	80
Итераций больше, чем число столбцов	80	86	114

Выводы:

- С ростом размерности матриц растет количество реализаций, в которых алгоритм завершает работу из-за ограничений на число итераций
- С ростом размерности матриц растет среднее количество итераций
- С ростом размерности матриц количество решенных уравнений падает
- С ростом размерности матриц большее количество уравнений не решается за максимальное заданное количество итераций

Визуализация. Матрицы 3×3

Введем декартову систему координат Ouv на плоскости $x + y + z = 0$.
Начало отсчета совпадает с началом отсчета системы $Oxyz$.
Ось Ov — проекция оси Oz на плоскость.
x', y', z' - проекции осей x, y, z на плоскость Ouv.
Выражаем u, v через x, y, z:

$$x' = \frac{x}{\sqrt{2}}, \quad y' = \frac{y}{\sqrt{2}}, \quad z' = \frac{z}{\sqrt{2}}$$

$$v = z' - x' \cos\left(\frac{\pi}{3}\right) - y' \cos\left(\frac{\pi}{3}\right) = \frac{1}{2\sqrt{2}}(2z - x - y)$$

$$u = x' \sin\left(\frac{\pi}{3}\right) - y' \sin\left(\frac{\pi}{3}\right) = \frac{\sqrt{3}}{2\sqrt{2}}(x - y)$$

Получаем выражения для u и v :

$$u = \frac{1}{3}\frac{\sqrt{3}}{2\sqrt{2}}(3\hat{x} - 3\hat{y}) = \frac{\sqrt{3}}{2\sqrt{2}}(\hat{x} - \hat{y})$$

$$v = \frac{1}{3}\frac{1}{2\sqrt{2}}(6\hat{z} - 3\hat{x} - 3\hat{y}) = \frac{2\hat{z} - \hat{x} - \hat{y}}{2\sqrt{2}}.$$

Визуализация. Матрицы 3×3. Проекция линейной оболочки

Пример
линейной оболочки столбцов для матрицы

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Построение линейной оболочки:

- генерировались
  числа $s_1, s_2, \ldots s_n$ из равномерного распределения на отрезке $[-||A||, ||A||]$,
  n — число столбцов матрицы A,
- получался вектор
  $z = (z_i), z_i = \max_j(a_{ij} + s_j), i = 1 \ldots n$,
- строилась проекция вектора z,
- алгоритм повторялся 1000 раз.

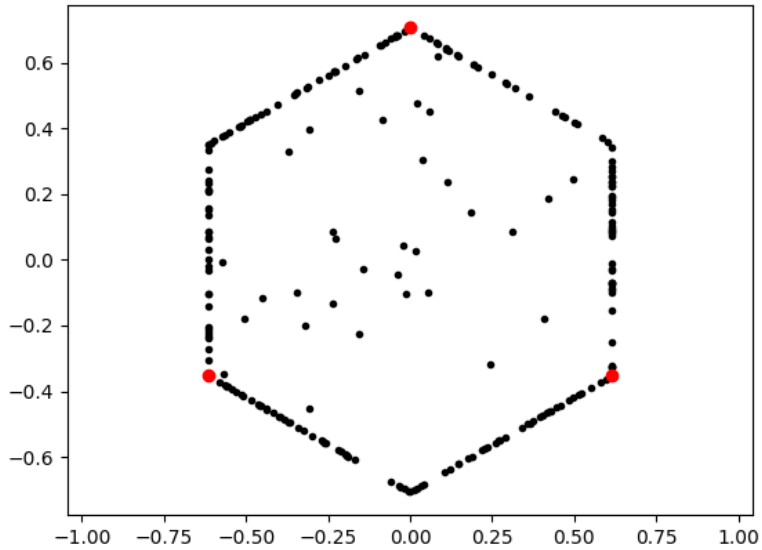

Рис. 2: Линейная оболочка

Красные точки — это проекции столбцов матрицы.
Черные точки — это сгенерированные значения линейной оболочки.

Визуализация. Матрицы 3×4 и 3×5. Проекции линейных оболочек

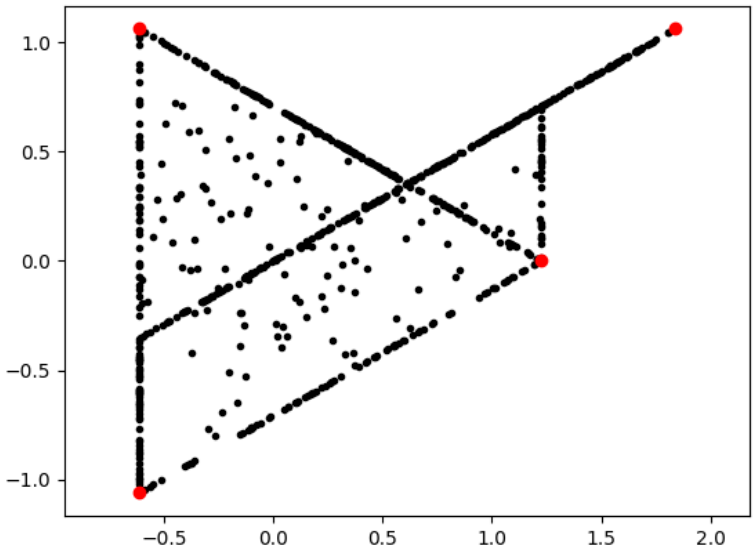

Рис. 3: Линейная оболочка столбцов матрицы

$$A = \begin{pmatrix} 1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 2 \\ 0 & -1 & 1 & 1 \end{pmatrix}$$

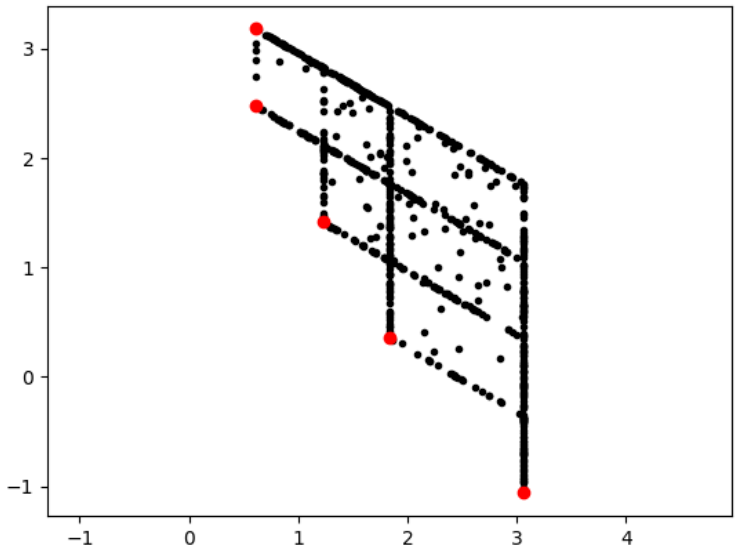

Рис. 4: Линейная оболочка столбцов матрицы

$$A = \begin{pmatrix} 1 & 1 & 2 & 3 & 5 \\ 0 & 0 & 0 & 0 & 0 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix}$$

Визуализация. Матрицы 2×2

Визуализация процесса поиска решения алгоритмом.
Для матриц 2×2 изображаются линейные оболочки столбцов матриц и приближения, найденные алгоритмом.
Для матриц 3×3 сперва необходимо спроецировать трехмерные вектора на двумерную плоскость, затем вывести результаты аналогично матрицам 2×2.
Алгоритм для матриц 2×2:

1. Выводим на изображение прямые, проходящие под углом 45° через координаты столбцов матриц. Это будут линейные оболочки столбцов матриц
2. Находим приближение Δ_i и y_{i+1}
3. Выводим на изображение точку с координатами By_{i+1}
4. Находим приближение Δ_i и x_{i+1}
5. Выводим на изображение точку с координатами Ax_{i+1}
6. Переходим к пункту (2)

Визуализация. Матрицы 2×2. Решение найдено

Входные данные:

$$x_0 = \begin{pmatrix} 29 \\ 88 \end{pmatrix}, A = \begin{pmatrix} 63 & 2 \\ 16 & 95 \end{pmatrix}, B = \begin{pmatrix} 78 & 47 \\ 23 & 65 \end{pmatrix}.$$

Ход работы алгоритма:

1. $i = 0,\ y_1 = (50.5, 81.5)^T,\ \Delta_0 = 73$
2. $i = 1,\ x_2 = (65.5, 51.5)^T,\ \Delta_1 = 0$

Решение и невязка:

$$x^* = \begin{pmatrix} 65.5 \\ 51.5 \end{pmatrix}, y^* = \begin{pmatrix} 50.5 \\ 81.5 \end{pmatrix}, \Delta_* = 0$$

Алгоритм закончил работу, так
как в точке (3) невязка стала равна нулю.
Решение выглядит верным, потому что
линейные оболочки матриц A и B пересекаются.

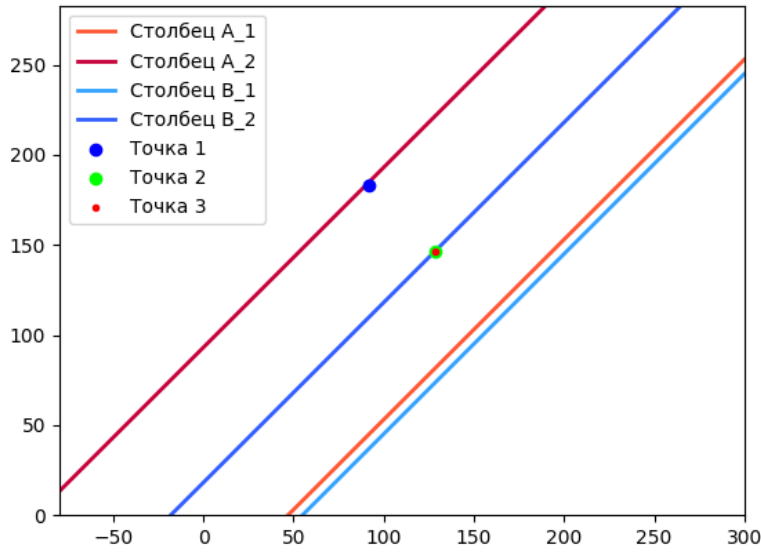

Рис. 5: Работа алгоритма

Визуализация. Матрицы 2×2. Решение не найдено

Входные данные:

$$x_0 = \begin{pmatrix} 36 \\ 18 \end{pmatrix}, A = \begin{pmatrix} 5 & 29 \\ 62 & 74 \end{pmatrix}, B = \begin{pmatrix} 49 & 93 \\ 22 & 88 \end{pmatrix}.$$

Оптимальные значения и невязка равны:

$$x^* = \begin{pmatrix} 33 \\ 21 \end{pmatrix}, y^* = \begin{pmatrix} 26 \\ -18 \end{pmatrix}, \Delta_* = 50$$

Алгоритм закончил
работу из-за совпадения точек (2) и (4).
Решение выглядит верным, потому что:

- линейные оболочки столбцов матриц A и B не пересекаются,
- точки (3) и (4) лежат на границах линейных оболочек.

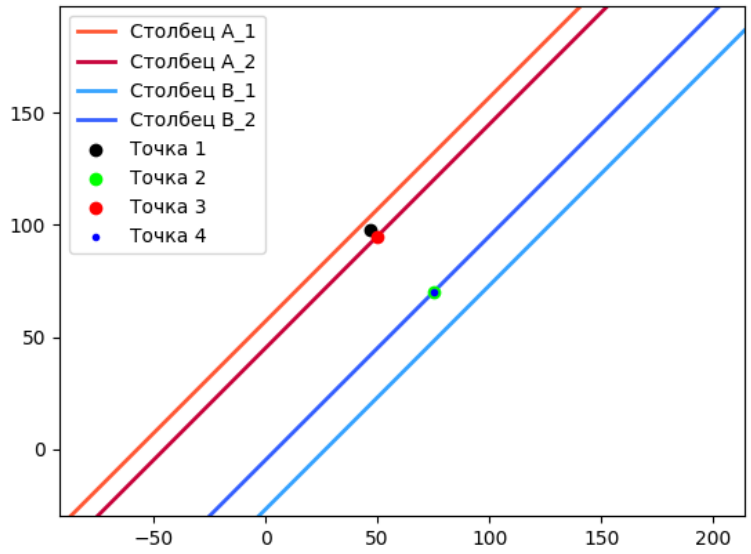

Рис. 6: Работа алгоритма

Визуализация. Матрицы 3×3. Решение найдено

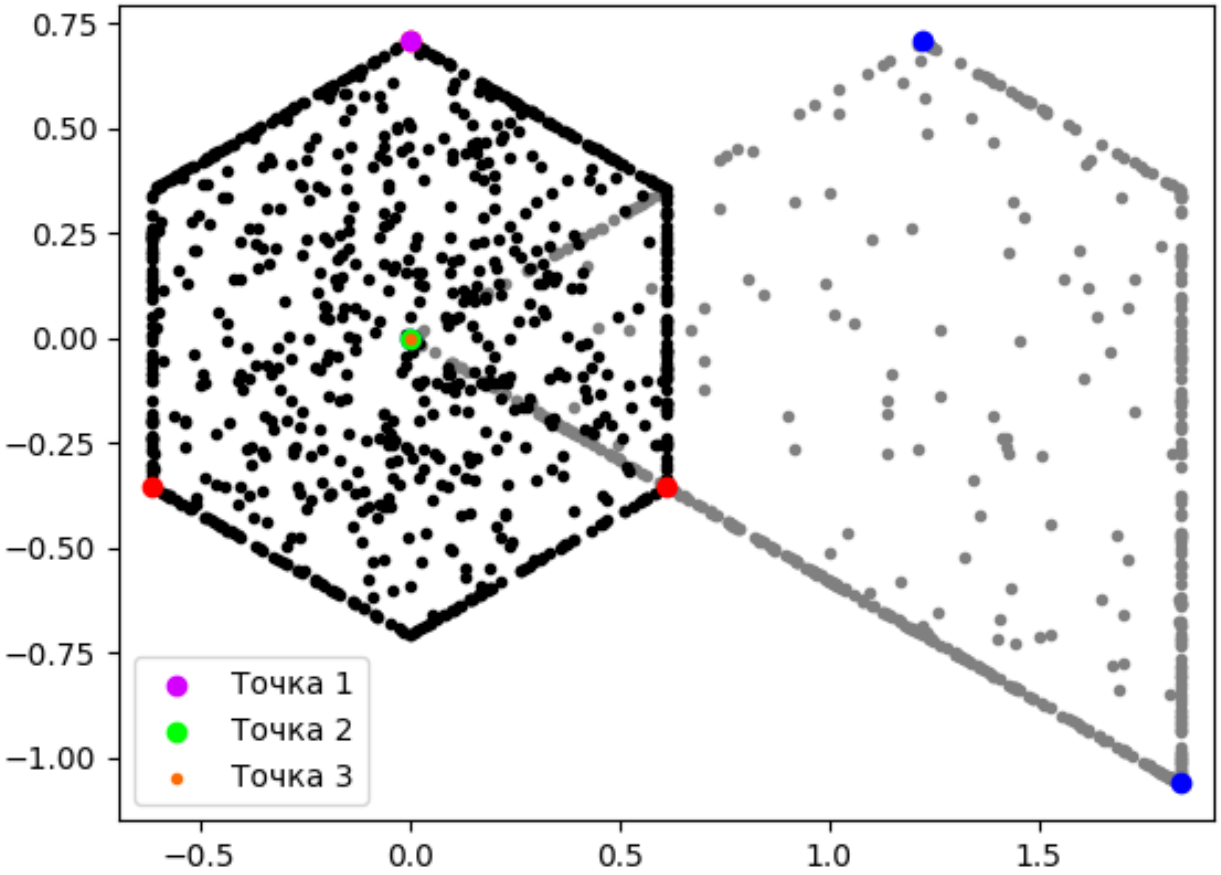

Рис. 7: Работа алгоритма

Визуализация. Матрицы 3×3. Решение найдено

Входные данные:

$$x_0 = \begin{pmatrix} 10 \\ 9 \\ 8 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 3 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Ход работы алгоритма:

1. $i = 0,\ y_1 = (7.5, 10.5, 10.5)^T,\ \Delta_0 = 1$
2. $i = 1,\ x_2 = (9.5, 9.5, 9.5)^T,\ \Delta_1 = 0$

Оптимальные значения и невязка равны:

$$x^* = x_0 = \begin{pmatrix} 9.5 \\ 9.5 \\ 9.5 \end{pmatrix}, \quad y^* = \begin{pmatrix} 7.5 \\ 10.5 \\ 10.5 \end{pmatrix}, \quad \Delta_* = 0$$

Алгоритм закончил работу так как в точке (3) невязка стала равна нулю.

Визуализация. Матрицы 3×3. Решение не найдено

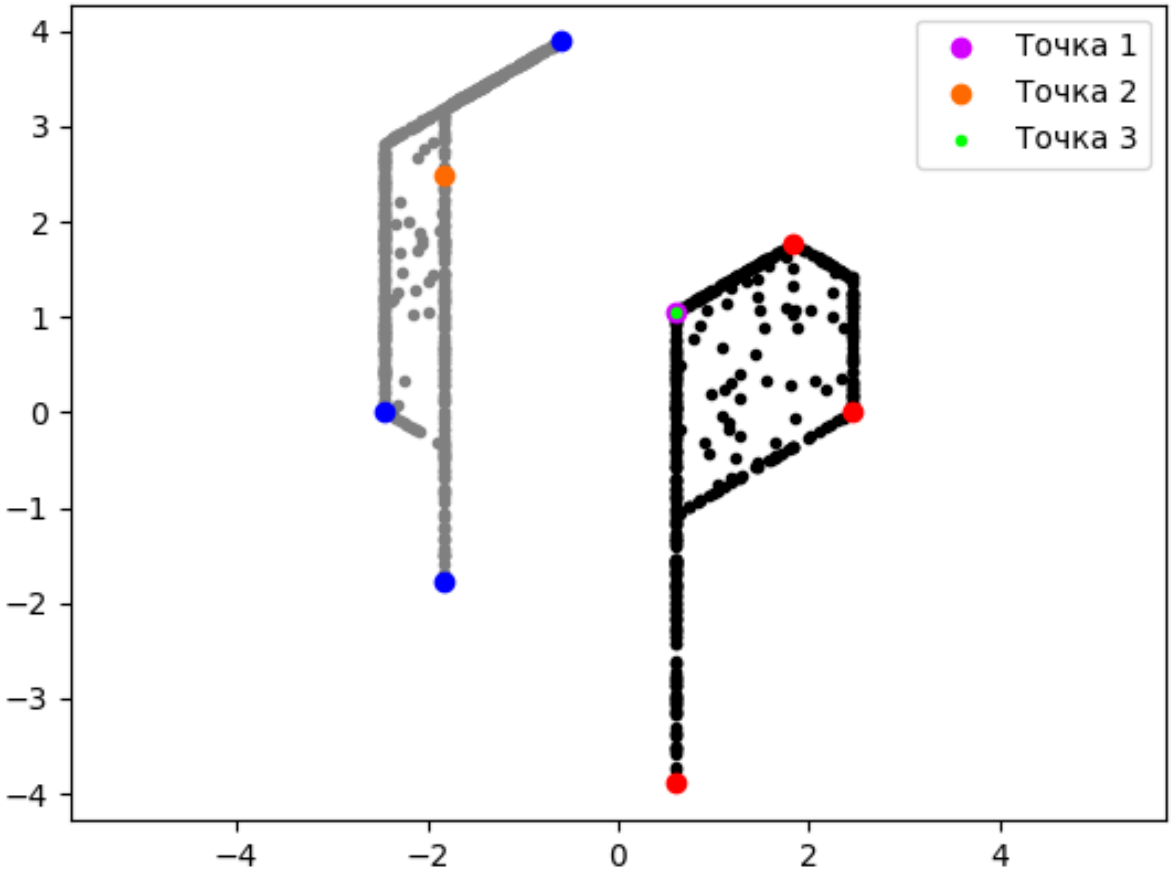

Рис. 8: Работа алгоритма

Визуализация. Матрицы 3×3. Решение не найдено

Входные данные:

$$x_0 = \begin{pmatrix} 10 \\ 9 \\ 8 \end{pmatrix}, \quad A = \begin{pmatrix} 3 & 2 & 1 \\ 0 & -2 & 0 \\ 4 & 0 & -5 \end{pmatrix}, \quad B = \begin{pmatrix} -3 & -2 & -1 \\ 0 & 2 & 0 \\ -4 & 0 & 5 \end{pmatrix}$$

Ход работы алгоритма:

1. $i = 0,\ y_1 = (12.0, 10.0, 9.0)^T,\ \Delta_0 = 4$
2. $i = 1,\ x_2 = (10.0, 9.0, 8.0)^T,\ \Delta_1 = 4$

Оптимальные значения и невязка равны:

$$x^* = \begin{pmatrix} 10 \\ 9 \\ 8 \end{pmatrix}, \quad y^* = \begin{pmatrix} 12 \\ 10 \\ 9 \end{pmatrix}, \quad \Delta_* = 4$$

Алгоритм закончил работу, так как значения в точках (1) и (3) совпали.

Визуализация. Матрицы 3×3. Большое количество итераций

Входные данные:

$$x_0 = \begin{pmatrix} 49 \\ 56 \\ 66 \end{pmatrix}, \quad A = \begin{pmatrix} 81 & 30 & 68 \\ 72 & 28 & 2 \\ 2 & 76 & 65 \end{pmatrix},$$

$$B = \begin{pmatrix} 35 & 18 & 39 \\ 45 & 24 & 33 \\ 32 & 48 & 85 \end{pmatrix}.$$

Оптимальные значения и невязка:

$$x^* = \begin{pmatrix} 47.5 \\ 98.5 \\ 60.5 \end{pmatrix}, \quad y^* = \begin{pmatrix} 81.5 \\ 102.5 \\ 89.5 \end{pmatrix}, \quad \Delta_* = 0$$

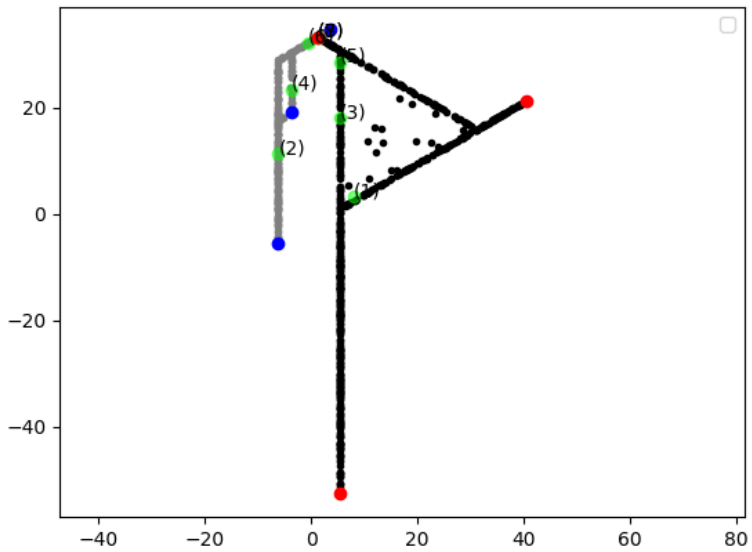

Рис. 9: Работа алгоритма

Заключение

Основные результаты работы:

1. изучены основы тропической математики,
2. рассмотрены алгоритмы решения уравнений,
3. найдено аналитическое решение уравнения $Ax = Bx$ для некоторых размерностей матриц,
4. выбран наиболее оптимальный алгоритм для дальнейшего исследования,
5. изучена зависимость количества итераций алгоритма от начальных параметров,
6. реализовано графическое представление линейных оболочек столбцов матриц,
7. реализовано графическое представление работы алгоритма.