Курсовая работа

Кузнецов М.А.

24 мая 2012

Вводный слайд

Тензоры широко используются в

- физике,
- дифференциальной геометрии,
- многофакторном анализе,
- психометрике,
- хемометрики.

Каноническое разложение

Определение

Тензором A размерности d назовем многомерный массив, элементы которого $A(i_1, i_2, \ldots, i_d)$ имеют d индексов. $1 \leq i_k \leq n_k$; n_k называются модовыми размерами (размерами мод)

Определение

Каноническим разложением многомерного массива (*тензора*) называется представление вида

$$A(i_1, i_2, \ldots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha) \ldots U_d(i_d, \alpha), \qquad (1)$$

где U_k называются *факторами* канонического разложения, а r — каноническим рангом.

Мотивировка

Конечная цель: задача об интерполяции многомерных данных
Есть набор точек (x_i, y_i); $x_i = \{ x_1, \ldots, x_d \}$ $y_i = f(x_i)$ и нужно построить интерполяцию функции $f(x)$

Цель курсовой работы

Целью курсовой работы является изучение метода переменных направлений для канонической аппроксимации тензора, и написание его эффективной реализации. К программе предъявляются следующие требования:

1. Она должна работать для любой размерности тензора.
2. Реализация на Python

Простейший функционал

Пусть задан тензор A с элементами $A_{i_1 \ldots i_d}$. Задача состоит в том, чтобы найти его каноническое приближение, а именно найти такие матрицы $U_1, \ldots, U_d$

$$A_{i_1,\ldots,i_d} \approx \sum_{\alpha=1}^{r} U_1(i_1,\alpha) U_2(i_2,\alpha) \ldots U_d(i_d,\alpha). \qquad (2)$$

Математическая постановка задачи состоит в том, чтобы решить задачу (2) в смысле наименьших квадратов

$$F = \sum_{i,j,k=1} (A_{ijk} - \sum_{\alpha=1}^{r} U_{i\alpha} V_{j\alpha} W_{k\alpha})^2.$$

Метод ALS: идея

Основная идея алгоритма, состоит в том, чтобы фиксировать все факторы, кроме одного, канонического разложения и искать минимум функционала

$$F = \sum_{i,j,k=1} (A_{ijk} - \sum_{\alpha=1}^{r} U_{i\alpha} V_{j\alpha} W_{k\alpha})^2.$$

только по нему. Путем циклических перестановок, используя уже полученные факторы, строятся последующие, до тех пор, пока не будет достигнута требуемая точность аппроксимации или, пока не сработают другие критерии остановки алгоритма

Формулы метода ALS

Найдем частную производную функционала F по $U_{\hat{i}\hat{\alpha}}$ и приравняем ее к 0:

$$\frac{\partial F}{\partial U_{\hat{i}\hat{\alpha}}} = 2\Big(\sum_{i,j,k}(A_{ijk} - \sum_{\alpha} U_{i\alpha}V_{j\alpha}W_{k\alpha})\Big)\Big(-\sum_{\check{\alpha}}(V_{j\check{\alpha}}W_{k\check{\alpha}})\frac{\partial U_{i\check{\alpha}}}{\partial U_{\hat{i}\hat{\alpha}}}\Big) = 0;$$

$$\frac{\partial U_{i\check{\alpha}}}{\partial U_{\hat{i}\hat{\alpha}}} = \delta_{i,\hat{i}}\delta_{\check{\alpha}\hat{\alpha}};$$

Окончательно, получаем следующие соотношения:

$$\sum_{j,k} A_{\hat{i}jk}V_{j\hat{\alpha}}W_{k\hat{\alpha}} = \sum_{j,k,\alpha} U_{\hat{i}\alpha}V_{j\alpha}W_{k\alpha}V_{j\hat{\alpha}}W_{k,\hat{\alpha}},$$

Формулы ALS

Обозначим через $M_{\alpha\hat{\alpha}}$ матрицу с элементами

$$M_{\alpha,\hat{\alpha}} = (\sum_j V_{j,\alpha} V_{j\hat{\alpha}})(\sum_k W_{k\alpha} W_{k\hat{\alpha}}); \quad (3)$$

тогда

$$\sum_\alpha U_{\hat{i},\alpha} M_{\alpha,\hat{\alpha}} = \sum_{j,k} A_{\hat{i},j,k} V_{j,\hat{\alpha}} W_{k,\hat{\alpha}}; \quad (4)$$

Через $F_{i,\hat{\alpha}}$ обозначим правую часть. Тогда, имеем

$$\sum_\alpha U_{\hat{i}\alpha} M_{\alpha\hat{\alpha}} = F_{i\hat{\alpha}}. \quad (5)$$

В виде системы

или в виде системы линейных уравнений

$$UM = F. \tag{6}$$

где $M \in \mathbb{R}^{r \times r}$.

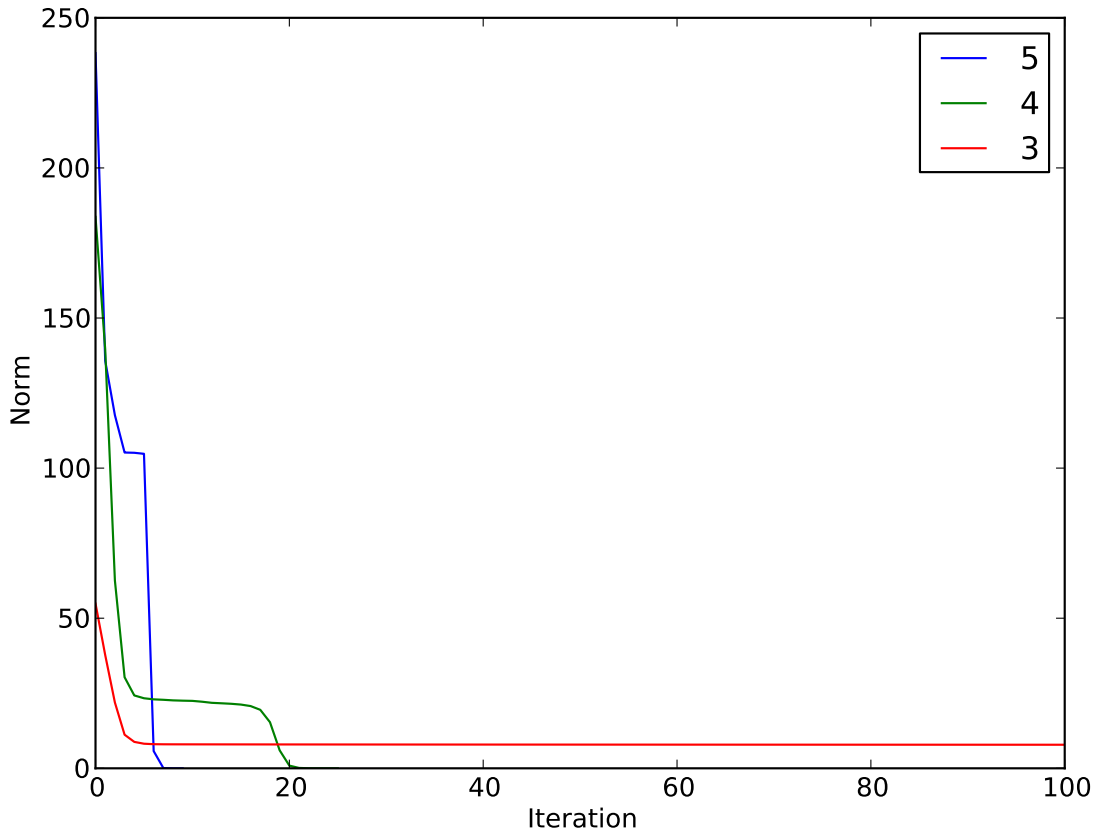
Реализация на Python

Поставленная задача реализации алгоритма ALS на Python предполагает:

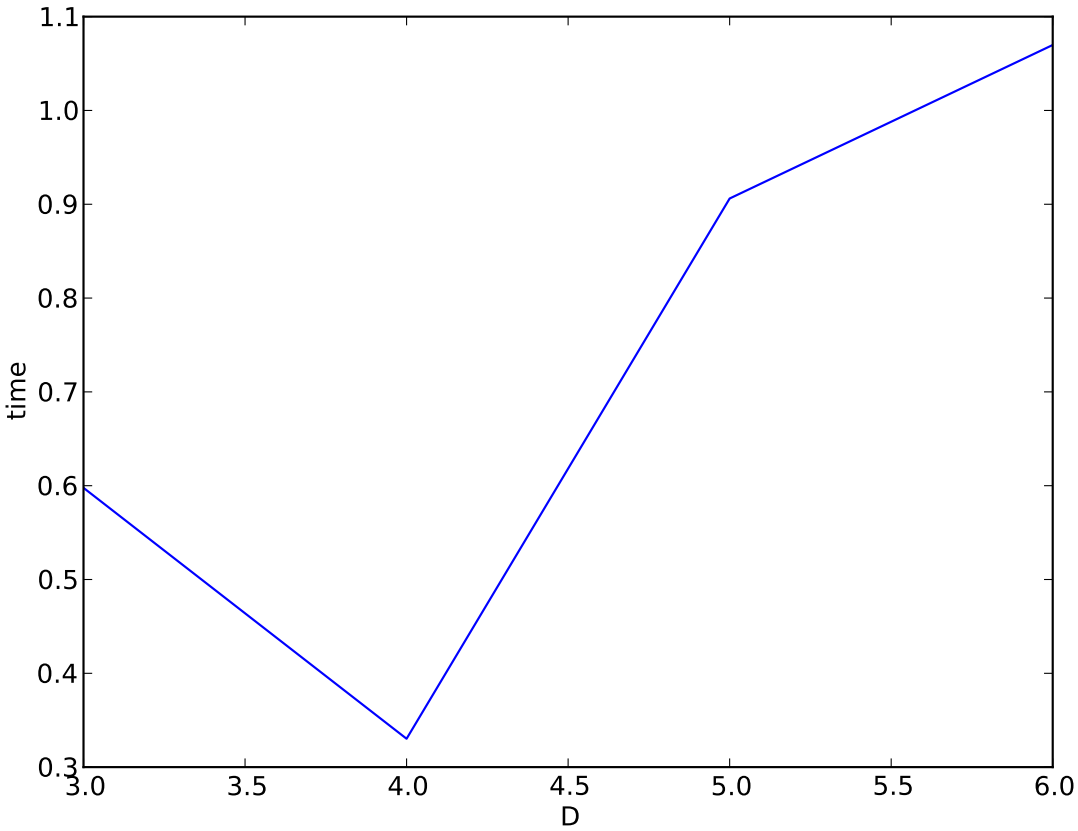
- Реализацию алгоритма в виде единой процедуры для любой размерности
- Реализацию функций вычисления правой и левой частей системы (6), используя математические ухищрения и возможности Python, для того чтобы обойти проблему неопределенной размерности, так как предыдущий пункт эту проблему ставит.
- Ограничение интрументария стандартными функциями библиотек (довольно богатых),чтобы избежать потерь в скорости, так как Python интерпретируемый скриптовый язык.

Численные эксперименты

Поведение невязки при разных размерностях тензора

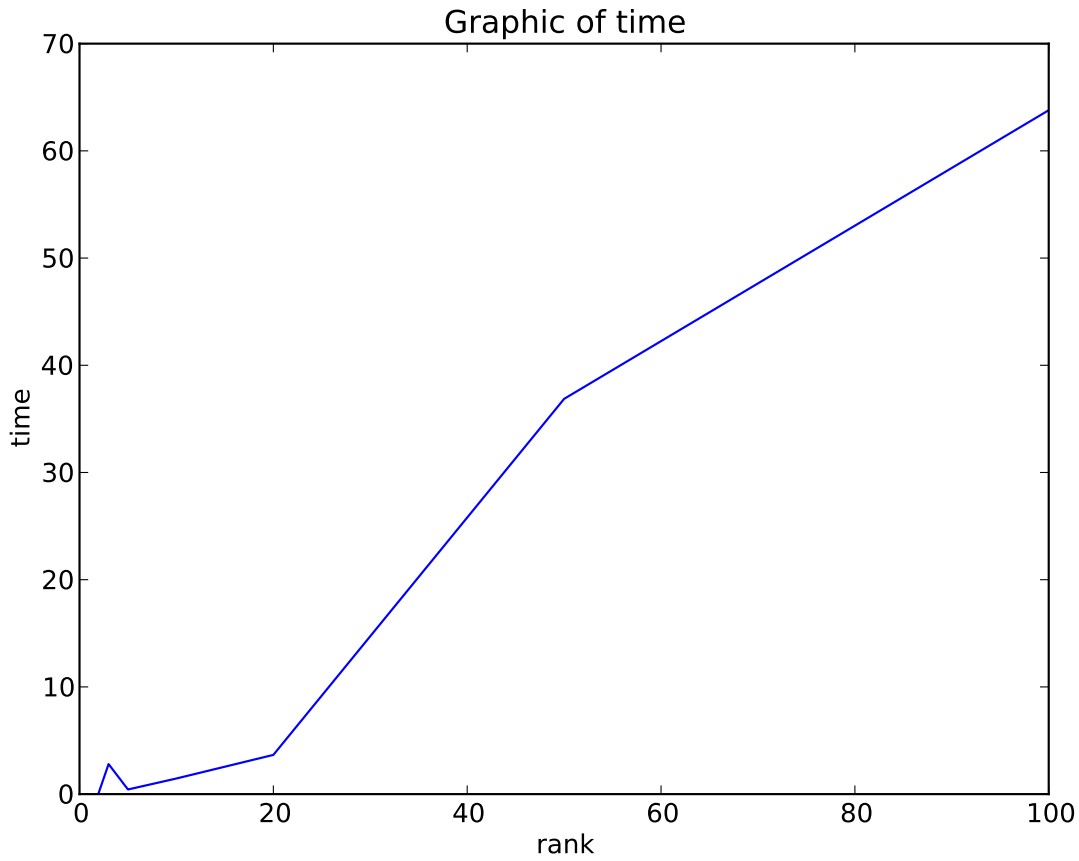


Зависимость времени выполнения программы от размерностей тензора

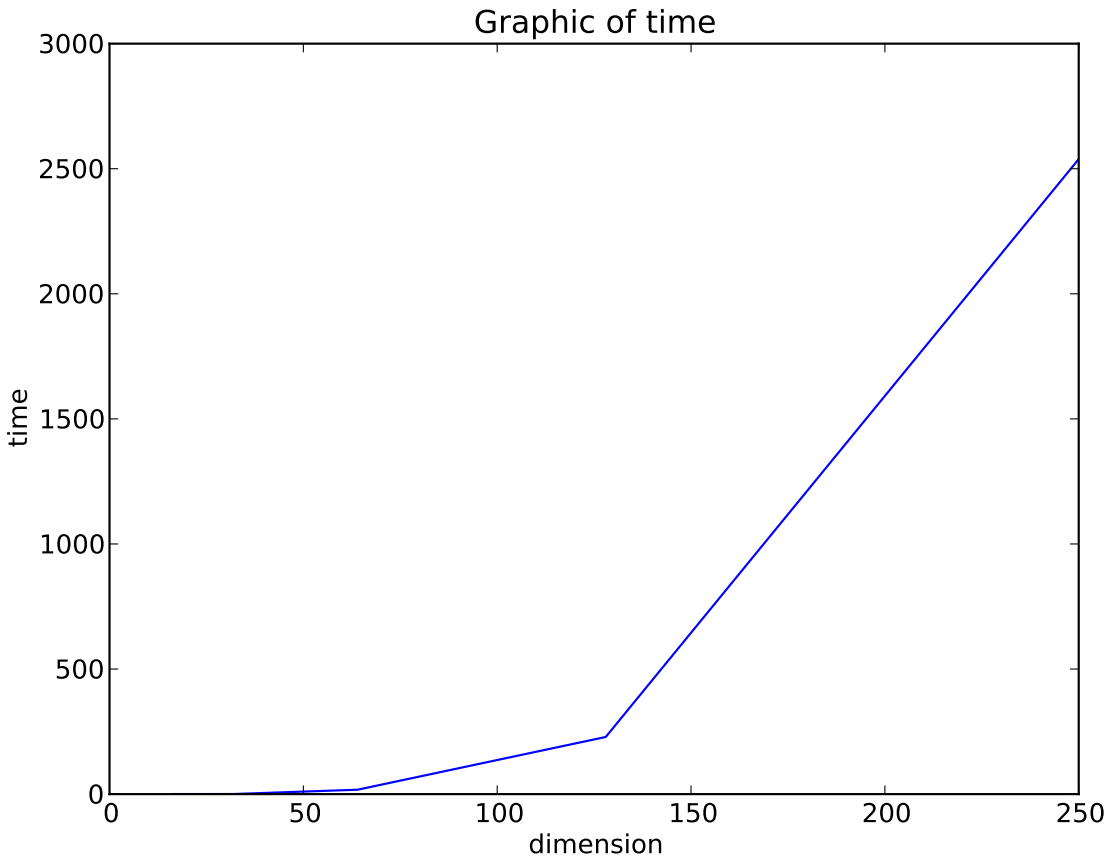


Численные эксперименты2

Зависимость времени выполнения программы от различных рангов



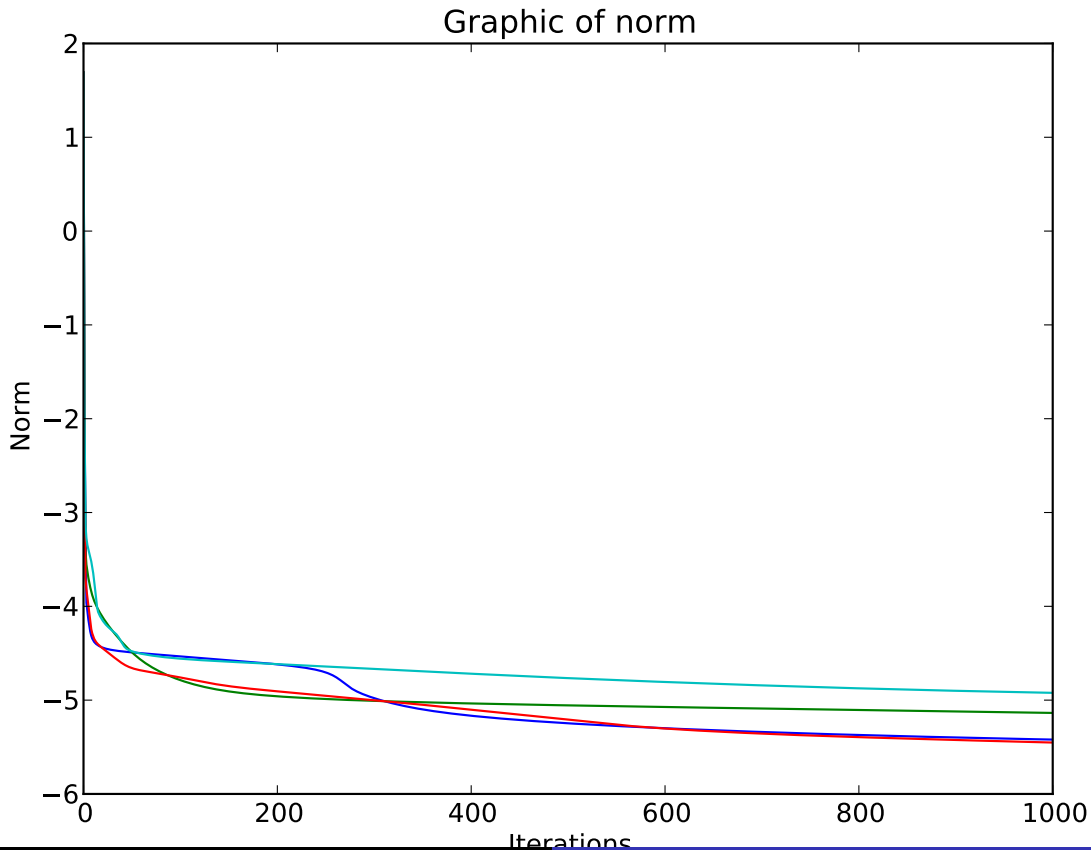
Зависимость времени выполнения программы от различных размеров мод



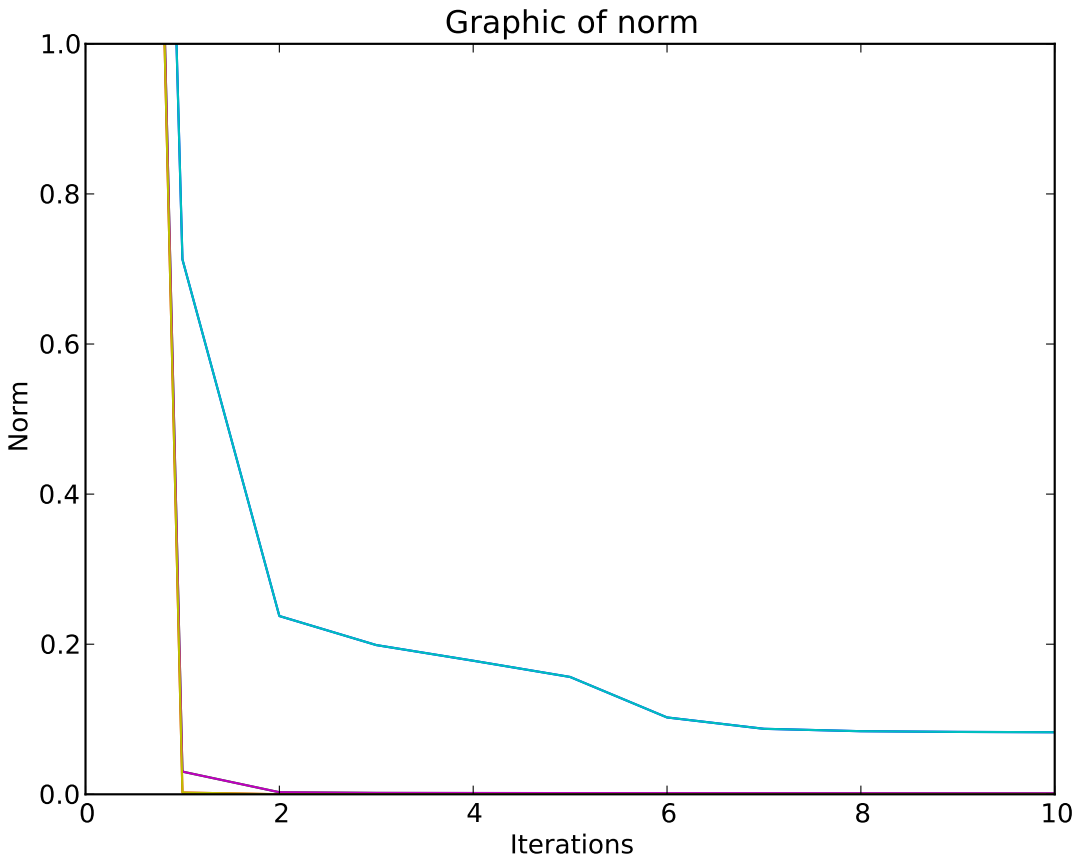
Численные эксперименты3

$$A[i,j,k] = \frac{1}{i+j+k+1}, i,j,k = 1,2,\ldots,n-1$$

Поведение невязки с разных стартов

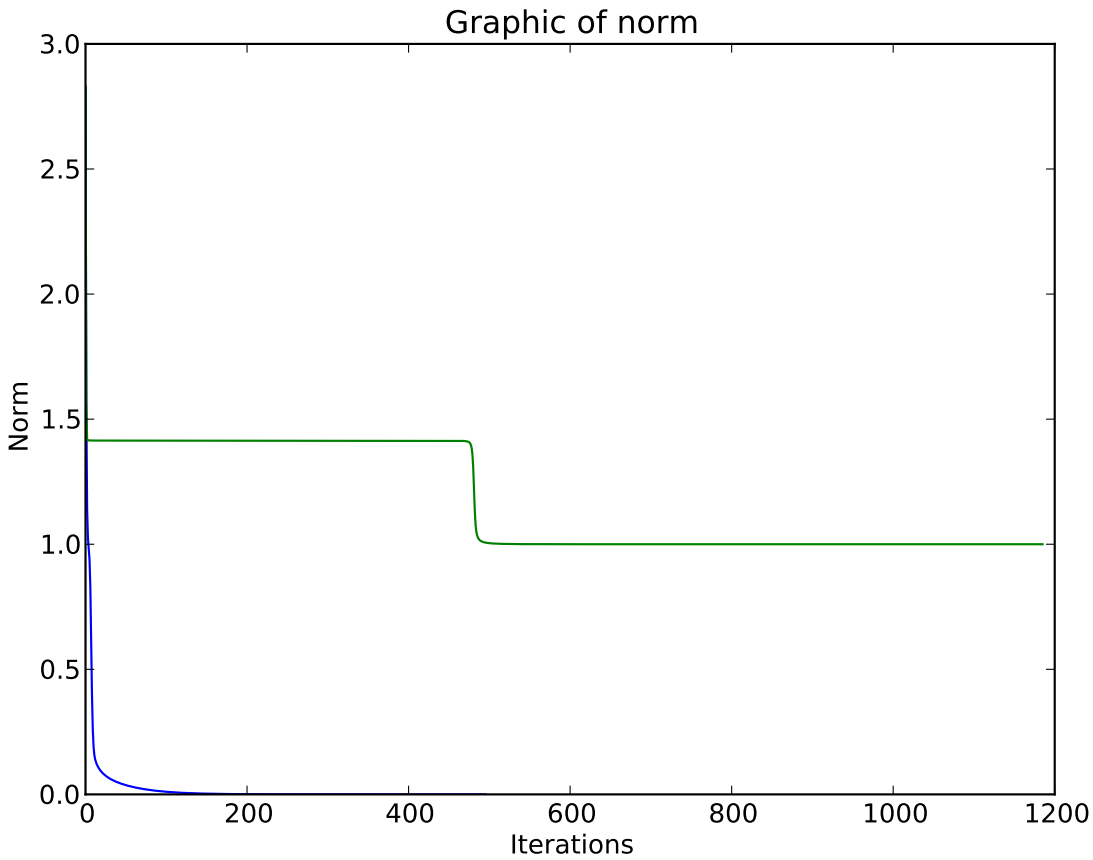


Поведение невязки при извменении ранга 5,20,37



Тензор матричного умножения

Поведение невязки тензора матричного умножения при ранге 7 и 6



Выводы и планы

В ходе выполнения работы была получена реализация алгоритма ALS, удовлетворяющая требованиям:

1. Независимости от размерности тензора
2. Реализации с помощью библиотек и стандартных средств языка Python

Результаты расчетов по реализации программы на Python согласуются с известными результатами.
В дальнейшем полученный опыт планируется распространить другие форматы и функционалы: TT (TensorTrain) и на задачу интерполяции многомерных функций