Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе № 2**

Дисциплина: Вычислительная математика

Выполнил студент гр. 3530901/10001 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Л. Симоновский

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Н. Цыган

(подпись)

“17” февраль 2023 г.

Санкт-Петербург

2023

Оглавление

[Задание: 2](#_Toc127542887)

[Инструменты: 2](#_Toc127542888)

[Ход выполнения работы: 2](#_Toc127542889)

[***Порядок действий:*** 2](#_Toc127542890)

[***Первая задача:*** 2](#_Toc127542891)

[Полная формулировка: 2](#_Toc127542892)

[Разбиваем на задачи: 2](#_Toc127542893)

[Пункт 1: 3](#_Toc127542894)

[Пункт 2: 3](#_Toc127542895)

[***Вторая задача:*** 4](#_Toc127542896)

[Полная формулировка: 4](#_Toc127542897)

[Решение: 4](#_Toc127542898)

[Результат: 6](#_Toc127542899)

[***Анализ:*** 7](#_Toc127542900)

[Вывод: 7](#_Toc127542901)

[Листинг кода: 8](#_Toc127542902)

[Ссылки: 9](#_Toc127542903)

# Задание:

Вариант 11:

Семейство линейных систем, представленных в следующем виде зависит от *p*:  
Решить линейные системы, используя программы DECOMP и SOLVE, при   
*p* = . Сравнить решение системы с решением системы  
, полученной из исходной, левой трансформацией Гаусса.  
Проанализировать связь числа обусловленности *cond* и величины

# Инструменты:

Для работы был выбран язык программирования Python версии 3.11 по причине удобства его использования для поставленной задачи. Были выбраны следующие библиотеки:

* NumPy – для большей скорости расчетов, для функций расчета нормы и обусловленности матрицы
* SciPy – для функции расчета решения матрицы
* PrettyTable – для красивого вывода в консоль таблицы

# Ход выполнения работы:

## ***Порядок действий:***

Поставленное задание легко можно разбить на две глобальные задачи:

1. Вычислительная математика.   
   Проанализировать связь между системами и .   
   Проанализировать теоретическую связь между *cond* и величиной
2. Выполнить расчеты для проверки теоретических выкладок, а также решить систему при *p* =

## ***Первая задача:***

### Полная формулировка:

Сравнить решение системы с решением системы  
, полученной из исходной, левой трансформацией Гаусса.  
Проанализировать связь числа обусловленности *cond* и величины .

### Разбиваем на задачи:

В поставленной задаче явно есть несколько этапов:

1. Сравнить решение системы с решением системы
2. Проанализировать связь числа обусловленности *cond* и величины .

### Пункт 1:

Первый пункт является очевидным, поэтому опустим доказательство и перейдем сразу к выводу.

Система уравнений эквивалентна системе с точностью до погрешности вычислений.

### Пункт 2:

Этот пункт я бы хотел вывести подробнее т.к. зависимость не столь очевидна:

Умножим второе уравнение на , таким образом получим , однако необходимо не забыть про вычислительную погрешность, для её симуляции добавим вектор r:

Вычтем из первой системы уравнений, вторую (r для простоты запишем со знаком плюс т.к. это не влияет на ход решения):

Далее воспользуемся свойством нормы матрицы :

Теперь используем свойство нормы матрицы :

Поделим обе части неравенства на :

Умножим правую часть неравенства на дробь :

(1)

Сделаем преобразования над первым уравнением в исходной системе, используя описанные выше свойства:

Подставим полученное в (1):

Как мы знаем, это определение числа обусловленности cond(A):

Таким образом мы получили отношение величины и cond(A)

## ***Вторая задача:***

### Полная формулировка:

Семейство линейных систем, представленных в следующем виде зависит от *p*:  
Решить линейные системы, используя программы DECOMP и SOLVE, при   
*p* = .

### Решение:

В основе задания лежат матрица A и вектор b, зависящие от константы p. Напишем функции для получения этих объектов:

**def** get\_matrix\_A(p):  
 *"""Для получения значения матрицы A по p"""* **return** np.array([  
 [p - 29, 6, -6, -4, -3, -8, -5, 5],  
 [6, -13, -3, 5, 4, 3, 1, 7],  
 [5, -5, -1, 7, 2, 0, 7, 1],  
 [5, -5, 5, 6, 4, -7, 4, 0],  
 [4, 4, 7, -4, 9, -8, -8, -4],  
 [-4, 5, -4, 1, 0, 12, 0, 6],  
 [-3, -2, -4, 2, -8, -3, 16, 4],  
 [7, 5, 0, 2, 0, -6, 8, -12]  
 ])  
  
  
**def** get\_vector\_b(p):  
 *"""Для получения значения матрицы b по p"""* **return** np.vstack(np.array([4 \* p - 175, 133, 110, 112, 17, 32, 13, -18]))

Далее напишем цикл для p = , используем для этого for-each по массиву из требуемых элементов, в начале выведем на экран переменную p, чтоб было понятно, для какого этапа производятся вычисления, а также получим матрицу A и вектор b:

**for** p **in** [1.0, 0.1, 0.01, 0.0001, 0.000001]:  
 print(**f'p={**p**}'**)matrix\_A = get\_matrix\_A(p)vector\_b = get\_vector\_b(p)

Далее необходимо получить транспонированную матрицу A для второй системы уравнений, сделать это можно вызвав соответствующий метод на матрице A:

matrix\_A\_transpose = matrix\_A.transpose()

Теперь посчитаем произведение матрицы на матрицу A и матрицы на вектор b:

*# Получение матрицы A\_trans \* А*matrix\_A\_transpose\_A = matrix\_A\_transpose.dot(matrix\_A)  
*# Получение вектора A\_trans \* b*vector\_A\_ransponse\_b = matrix\_A\_transpose.dot(vector\_b)

На данный момент у нас были получены все компоненты для решения систем уравнений , осталось вызвать для них DECOMP и SOLVE, однако в python есть метод scipy.linalg.solve, он принимает на вход матрицу A и матрицу b, в том числе в общем виде, и выдает результирующую матрицу. Таким образом нет необходимости раскладывать матрицу на LU, можно сразу передать полученные выше матрицы для системы в ответ:

*# Решение уравнения Ax=b*result = solve(matrix\_A, vector\_b)  
*# Решение уравнения A\_t \* A \* x = A\_t \* b*result\_transform = solve(matrix\_A\_transpose\_A, vector\_A\_ransponse\_b)

Выведем на экран информацию об обусловленности матрицы A и матрицы . В качестве функции для вычисления нормы будем использовать норму Фробениуса. Для этого используем numpy.linalg.cond, передавая в качестве параметра p="fro":

print(f'Обусловленность матрицы A: {cond(matrix\_A, p="fro")} '  
 f'A\_T \* A: {cond(matrix\_A\_transpose\_A, p="fro")}')

Так же необходимо вывести полученные результаты решения систем уравнений и и разницу между их решениями, для этого напишем функцию, воспользовавшись библиотекой prettytable:

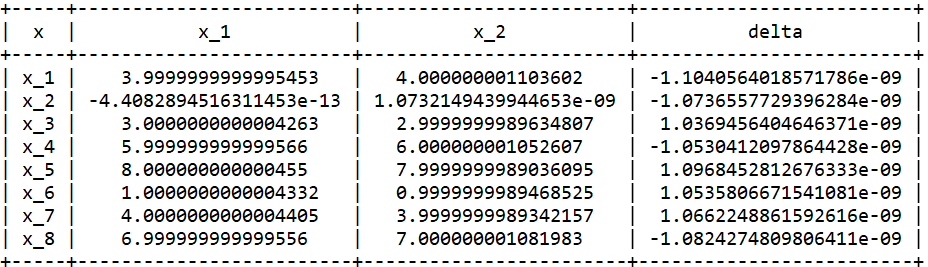
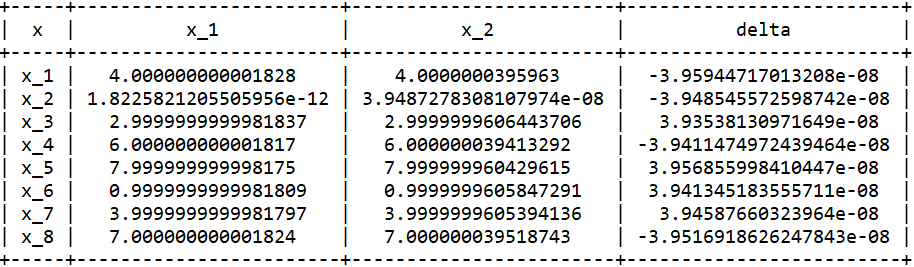
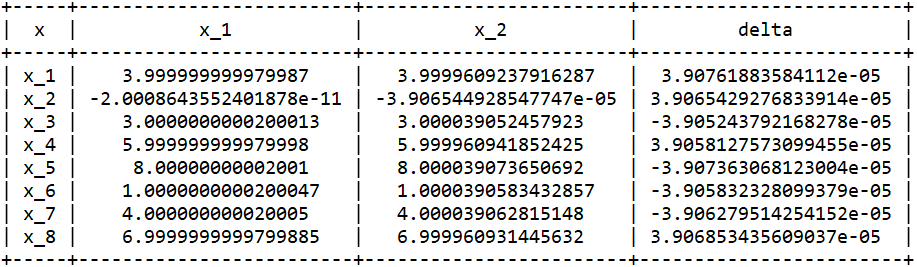
**def** print\_answer(x\_1, x\_2):  
 *"""Выводит на экран два решения и разницу между ними"""* pt = PrettyTable()  
 pt.add\_column(**'x'**, [**f'x\_{**i + 1**}' for** i **in** range(0, len(x\_1))])  
 pt.add\_column(**'x\_1'**, [x[0] **for** x **in** x\_1])  
 pt.add\_column(**'x\_2'**, [x[0] **for** x **in** x\_2])  
 pt.add\_column(**'delta'**, [x[0] **for** x **in** x\_1 - x\_2])  
 print(pt)

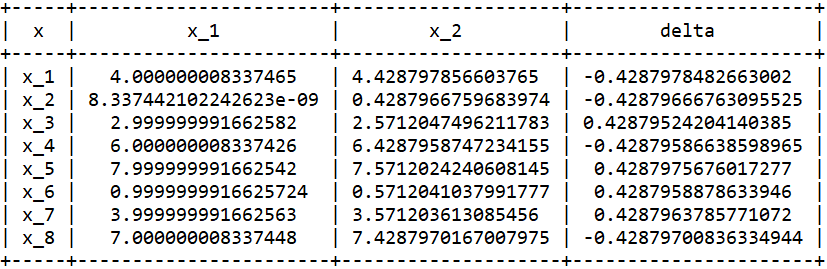
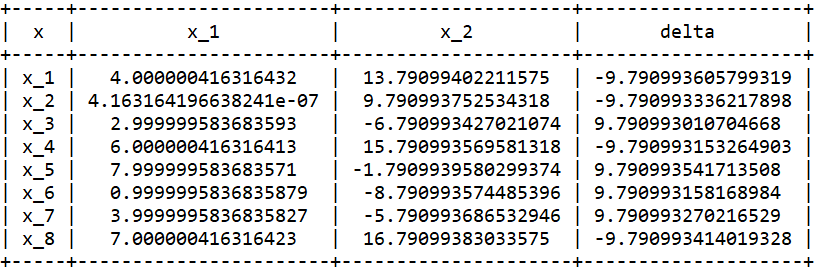
Ну и последнее – выведем на экран значение , чтоб проверить теоретические выкладки из пункта 1. Для вычисления нормы будем использовать numpy.linalg.norm и передавать в качестве аргумента ord="fro", дабы в качестве нормы функция использовала норму Фробениуса:

print(**f'||x\_1-x\_2||/||x\_1||='  
 f'{**norm(result - result\_transform, ord=**"fro"**) / norm(result, ord=**"fro"**)**}'**)

### Результат:

В итоге программа выдала такие значения:

p=1.0   
Обусловленность матрицы A: 8519.140985499118 A\_T \* A: 39385779.397530325  
  
||x\_1-x\_2||/||x\_1||=2.1920150022324254e-10  
===========================================================  
p=0.1   
Обусловленность матрицы A: 88317.35680165827 A\_T \* A: 4263883451.436119  
  
||x\_1-x\_2||/||x\_1||=8.07895675426055e-09  
===========================================================  
p=0.01  
Обусловленность матрицы A: 886347.6011370448 A\_T \* A: 429776614560.4676  
  
||x\_1-x\_2||/||x\_1||=7.9948412920894e-06  
===========================================================

p=0.0001  
Обусловленность матрицы A: 88669727.7088329 A\_T \* A: 4520269532764875.0  
  
||x\_1-x\_2||/||x\_1||=0.08775656200579741  
===========================================================  
p=1e-06  
Обусловленность матрицы A: 8867008514.835018 A\_T \* A: 1.591928701060035e+17  
  
||x\_1-x\_2||/||x\_1||=2.0038031860726955  
===========================================================

Так же в ходе работы было выдано предупреждение:

LinAlgWarning: Ill-conditioned matrix (rcond=2.92164e-18): result may not be accurate.

## ***Анализ:***

Из результата работы программы можно сделать ряд выводов:

1. Чем меньше переменная p, тем больше обусловленность матрицы A
2. Чем больше обусловленность матрицы A, тем больше обусловленность матрицы
3. Чем больше обусловленность матрицы A, тем больше разброс между решениями системы уравнений и
4. Чем больше обусловленность матрицы A, тем больше

Все эти пункты соответствуют математическим ожиданиям.

Так же хотелось бы отдельно отметить LinAlgWarning. Из его описания понятно, что он говорит о том, что обусловленность матрицы A слишком большая, что приведет к неточностям.

# Вывод:

В ходе работы я ознакомился с аналогом подпрограммы SOLVE, поработал с функциями для вычисления нормы матрицы и обусловленности в языке Python. По результатам работы видно, что зависит от обусловленности матрицы A, а также, что при увеличении этой обусловленности погрешность решения сильно возрастает.

# Листинг кода:

**import** numpy **as** np  
**from** prettytable **import** PrettyTable  
**from** scipy.linalg **import** solve  
**from** numpy.linalg **import** cond, norm  
  
  
**def** get\_matrix\_A(p):  
 *"""Для получения значения матрицы A по p"""* **return** np.array([  
 [p - 29, 6, -6, -4, -3, -8, -5, 5],  
 [6, -13, -3, 5, 4, 3, 1, 7],  
 [5, -5, -1, 7, 2, 0, 7, 1],  
 [5, -5, 5, 6, 4, -7, 4, 0],  
 [4, 4, 7, -4, 9, -8, -8, -4],  
 [-4, 5, -4, 1, 0, 12, 0, 6],  
 [-3, -2, -4, 2, -8, -3, 16, 4],  
 [7, 5, 0, 2, 0, -6, 8, -12]  
 ])  
  
  
**def** get\_vector\_b(p):  
 *"""Для получения значения матрицы b по p"""* **return** np.vstack(np.array([4 \* p - 175, 133, 110, 112, 17, 32, 13, -18]))  
  
  
**def** print\_answer(x\_1, x\_2):  
 *"""Выводит на экран два решения и разницу между ними"""* pt = PrettyTable()  
 pt.add\_column(**'x'**, [**f'x\_{**i + 1**}' for** i **in** range(0, len(x\_1))])  
 pt.add\_column(**'x\_1'**, [x[0] **for** x **in** x\_1])  
 pt.add\_column(**'x\_2'**, [x[0] **for** x **in** x\_2])  
 pt.add\_column(**'delta'**, [x[0] **for** x **in** x\_1 - x\_2])  
 print(pt)  
  
  
**def** main():  
 **for** p **in** [1.0, 0.1, 0.01, 0.0001, 0.000001]:  
 print(**f'p={**p**}'**)  
 *# Получение матрицы A* matrix\_A = get\_matrix\_A(p)  
 *# Получение вектора b* vector\_b = get\_vector\_b(p)  
 *# Получение транспонированной матрицы A* matrix\_A\_transpose = matrix\_A.transpose()  
 *# Получение матрицы A\_trans \* А* matrix\_A\_transpose\_A = matrix\_A\_transpose.dot(matrix\_A)  
 *# Получение вектора A\_trans \* b* vector\_A\_ransponse\_b = matrix\_A\_transpose.dot(vector\_b)  
 *# Решение уравнения Ax=b* result = solve(matrix\_A, vector\_b)  
 *# Решение уравнения A\_t \* A \* x = A\_t \* b* result\_transform = solve(matrix\_A\_transpose\_A, vector\_A\_ransponse\_b)  
 *# Выводим обусловленность матриц* print(**f'Обусловленность матрицы A: {**cond(matrix\_A, p=**"fro"**)**} '  
 f'A\_T \* A: {**cond(matrix\_A\_transpose\_A, p=**"fro"**)**}'**)  
 *# Выводим вектора ответа и дельту* print\_answer(result, result\_transform)  
 *# Выведем ||x\_1-x\_2||/||x\_1||* print(**f'||x\_1-x\_2||/||x\_1||='  
 f'{**norm(result - result\_transform, ord=**"fro"**) / norm(result, ord=**"fro"**)**}'**)  
 print(**'='** \* 80)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

# Ссылки:

Листинг код: [github.com](https://github.com/DafterT/comp_math_2)

Документация по SciPy: [docs.scipy.org](https://docs.scipy.org/doc/scipy/index.html)