**Формулировка задания:**

Решить СЛАУ методом градиентного спуска и исследовать зависимость числа итераций от точности и от определителя(уменьшая определитель следить за числом обусловленности)

**Алгоритм:**

Шаг 1: сгенерировать матрицу А размера n *x* n, выбрать начальное приближение - nмерный вектор , ввести nмерный вектор b - вектор правой части матричного уравнения

Шаг 2: вычислить kый вектор невязки:

Шаг 3: вычислить итерационный параметр:

Шаг 4: получаю следующее приближение:

Шаг 5: проверить норму разности двух последних приближений, если их разность меньше заданной точности, то ответ - последнее приближение, иначе перейти к шагу 2

**Анализ задачи + проверка условий:**

Для решения СЛАУ методом градиентного спуска необходимо генерировать положительно определенную симметричную матрицу, чтобы удовлетворить этим условиям генерируется сначала диагональная и ортогональная матрица и конечная матрица получается перемножением: , где С - ортогональная матрица и D - диагональная матрица. Чтобы понимать, что метода работает правильно(находит правильное решение) сначала буду генерировать вектор решение и перемножая вектор решений и данную матрицу буду получать вектор правых частей

**Тестовый пример:**

A = [1 1 1; 1 2 2; 1 2 4]

b = [6; 7; 13]

ε = 0.1

первая итерация:

Шаг 1: = [0; 0; 0]

Шаг 2: = [6; 7; 13]

Шаг 3: = -0.179632

Шаг 4: = [1.077793; 1.257425; 2.335219]

Шаг 5: перейти к шагу 2

вторая итерация:

Шаг 2: = [1.329561; -1.263083; 0.066478]

Шаг 3: = 2.309091

Шаг 4: = [4.147871; -1.659148; 2.488723]

Шаг 5: перейти к шагу 2

третья итерация:

Шаг 2: = [1.022553; 1.192979 2.215532]

Шаг 3: = 0.179632

Шаг 4: = [4.331555; -1.444851; 2.886704]

Шаг 5: перейти к шагу 2

четвертая итерация:

Шаг 2: = [0.226591; -0.215261; 0.011329]

Шаг 3: = 2.309091

Шаг 4: = [4.854775; -1.941910; 2.912865]

Шаг 5: перейти к шагу 2

пятая итерация:

Шаг 2: = [0.174269; 0.203314; 0.377583]

Шаг 3: = 0.179632

Шаг 4: = [4.886079; -1.905388; 2.980691]

Шаг 5: перейти к шагу 2

шестая итерация:

Шаг 2: = [0.038616; -0.036686; 0.001930]

Шаг 3: = 2.309091

Шаг 4:= [4.975250; -1.990100; 2.985150]

Шаг 5: x = [4.975250; -1.990100; 2.985150]

**Перечень контрольных тестов:**

Чтобы получить зависимости на графиках ниже(***численный анализ решения***):

в первом случае матрица оставалась неизменной, менялась только точность(матрица генерировалась соответственно правилам описанным в **анализ задачи + проверка условий**), во втором случае матрица для того, чтобы число обусловленности сохранялось диагональная матрица умножалась на , где n - порядок матрицы(это делалось для сохранения числа обусловленности матрицы неизменным, при этом уменьшая определитель)

**Модульная структура программы:**

программа состоит из библиотеки работы с матрицами(определение типов: матрица, вектор, работа с памятью: выделение, удаление памяти, выделенной матрицы, вектора, простые операции: перемножение матриц, перемножение матрицы и вектора, генерация ортогональной нормированной матрицы и т.д.)

matrix\_t genA(**int** size);

size размер матрицы

возвращает матрицу с подходящими условиями

solution\_t solve(matrix\_t matrix, vector\_t b, **int** size, **double** accuracy)

решает СЛАУ с матрицей - matrix размера - size с точностью - accuracy и вектором правых частей - b методом градиентного спуска и возвращает структуру, состоящую из количества итераций и вектора решений

**void** iterations\_by\_accuracy();

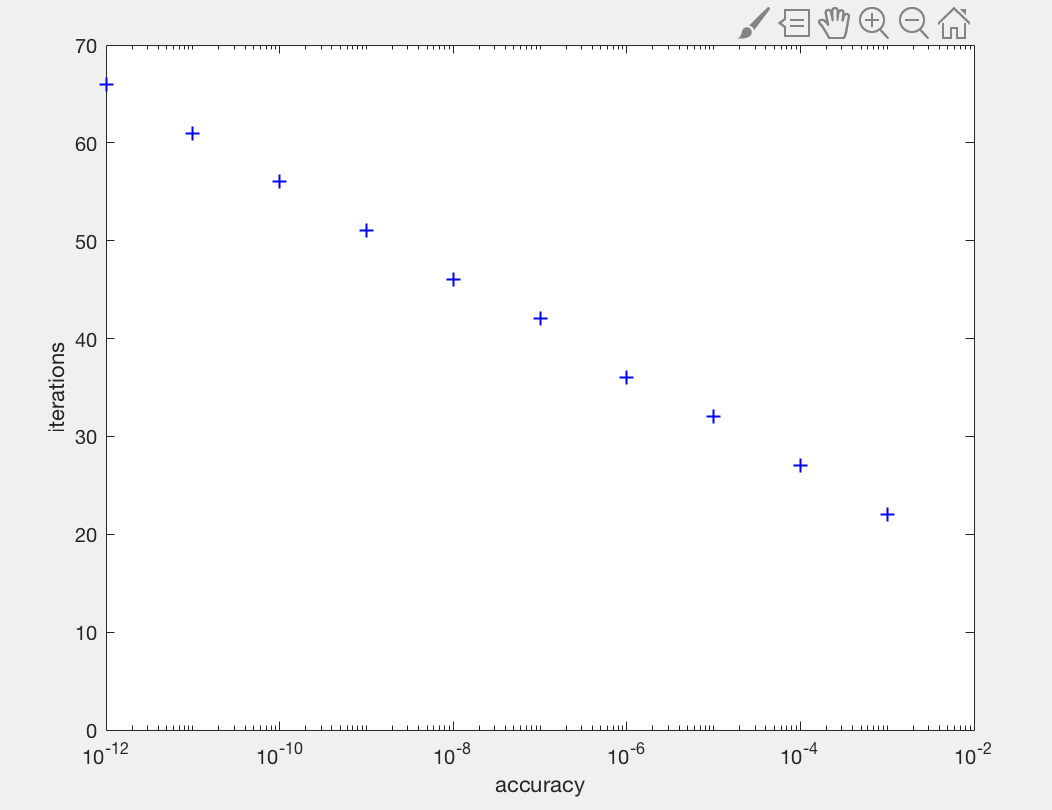
**void** iterations\_by\_determinant();

записывают данные, полученные в ходе тестов, в свой файл

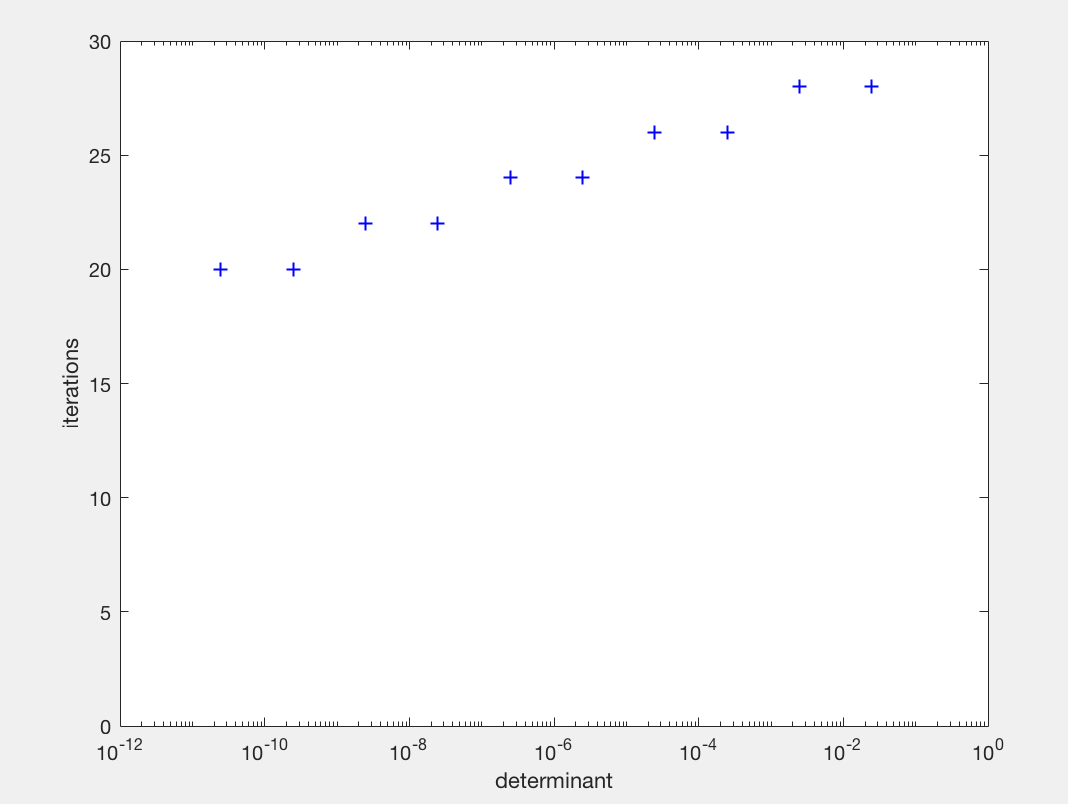
function main()

считывает данные из файлов и выводит на экран зависимость

**Численный анализ решения:**



экспоненциальная зависимость количества итераций от точности



как ни странно: чем меньше определитель - тем меньшее количество итераций требуется

**Вывод:**

не очень эффективный метод(пример более эффективного метода - метод сопряженных градиентов), но важный для понимания