Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра 806 «Вычислительная математика и программирование»

Курсовая работа

по курсу «Численные методы».

Решение систем линейных алгебраических уравнений с симметричными разреженными матрицами большой размерности. Метод сопряженных градиентов.

Студент: Лукашкин К. В.
Группа: М8О-308Б

Москва, 2019

Задание

Решение систем линейных алгебраических уравнений с симметричными разреженными матрицами большой размерности. Метод сопряженных градиентов.

Теоретические сведения

Метод сопряженных градиентов — численный метод решения систем линейных алгебраических уравнений, является итерационным методом Крыловского типа. Пусть дана система линейных уравнений: $Ax=b$. Причём матрица системы — это симметричная положительно определённая матрица. Тогда процесс решения СЛАУ можно представить как минимизацию следующего функционала: $(Ax, x) - 2(b, x) \to min$

Алгоритм:

Подготовка перед итерационным процессом

1. Выберем начальное приближение x^0
2. $r^0 = b - Ax^0$
3. $z^0 = r^0$

k-я итерация метода

1. $\alpha_k = \dfrac{(r^{k-1}, r^{k-1})}{(Az^{k-1}, z^{k-1})}$
2. $x^k = x^{k-1} + \alpha_k z^{k-1}$
3. $r^k = r^{k-1} - \alpha_k A z^{k-1}$
4. $\beta_k = \dfrac{(r^k, r^k)}{(r^{k-1}, r^{k-1})}$
5. $z^k = r^k + \beta_k z^{k-1}$

Как и все методы на подпространствах Крылова, метод сопряженных градиентов от матрицы требует только возможность умножать её на вектор, что приводит к возможности использовать специальные форматы хранения матрицы (такие, как разреженный) и сэкономить память на хранении матрицы.

Реализация

Для представления разреженных матриц, использовался массив стандартных ассоциативных контейнеров Python3 – dict. При этом была введена дополнительная надстройка – если данного элемента нет в строке, то возвращается 0. Таким образом в памяти находятся только ненулевые значения.

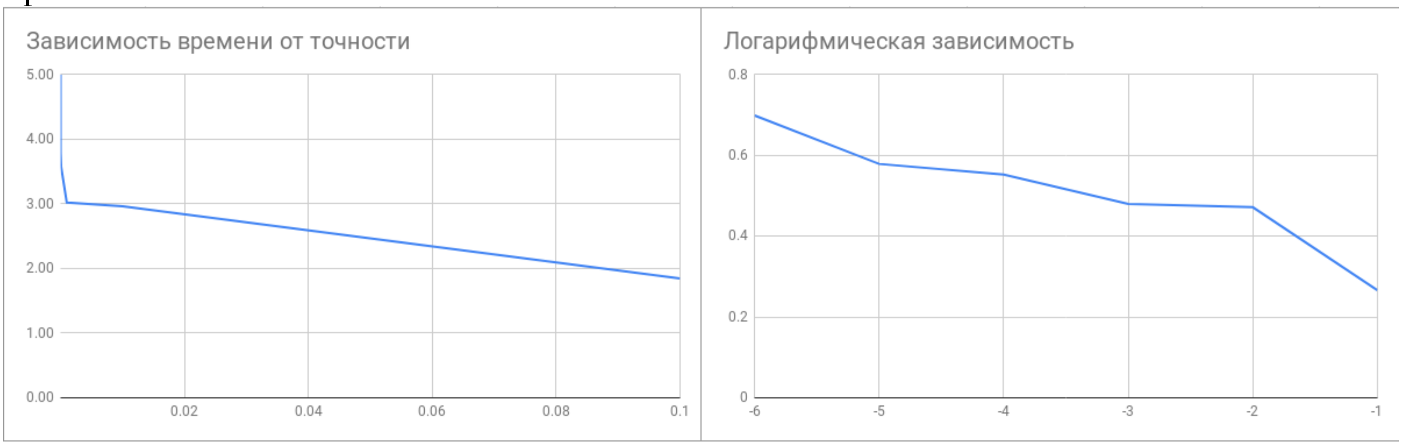
Код алгоритма для матрицы.

```
def conjugate_gradient(A, b, eps=0.0001):
    n = len(b)
    max_iter = 10 ** 4
    # проверка на симметричность
    for i in range(n):
        for j in range(n):
            if A[i][j] != A[j][i]:
                raise TypeError('Матрица не симметрична')
    b_product = scalar_product(b, b)
    # начальное приближение
    x = [0.2] * n
    #  Задаем начальное значение r и z.
```

```
    # r = b - A * x      r - градиент
    r = sub(b, mul(A, x))
    # z = r
    z = r.copy()
    iter_count = 0
    while True:
        iter_count += 1
        # alpha = (r,r)/(A*z,z) скалярный шаг -- смещение по заданному направлению
        alpha = scalar_product(r, r) / scalar_product(mul(A, z), z)
        # новое приближение: x = x + alpha * z
        x = add(x, mul(alpha, z))
        # градиент: r = r - alpha * A * z
        r_prev = r
        r = sub(r, mul(alpha, mul(A, z)))
        # коэфф бета для нового вектора спуска
        beta = scalar_product(r, r) / scalar_product(r_prev, r_prev)
        # вектор спуска  z = r+ beta * z-1
        z = add(r, mul(beta, z))
        if scalar_product(r, r) / b_product < eps \
                or iter_count > max_iter:
            break
return x
```

Оценка скорости работы.

Тесты программы для матрицы 186x186 – 34596 элементов из которых ненулевые приблизительно 1000.



В целом, для алгоритма ожидается линейная сложность, однако вследствие погрешности представления вещественных чисел, сложность получается существенно выше.

Вывод

Выполнив курсовую работу по курсу Численные методы, я приобрёл практические навыки в использовании знаний, полученных в течении курса и провел исследование в выбранной предметной области. Мною был реализован алгоритм сопряженных градиентов, в частности для разреженных матриц.