МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по курсу

«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 2

приложение 1-3, приложение 2(п. 2-6)

ОТЧЁТ

о выполненном задании

студента 205 учебной группы факультета ВМК МГУ

Феофилактова Андрея Дмитриевича

Москва, 2016

Содержание

1. Цель работы

Изучить классические итерационные методы (Зейделя и верхней релаксации), используемые для численного решения систем линейных алгебраических уравнений. Изучить скорость сходимости этих методов в зависимости от выбора итерационного параметра.

2. Постановка задачи

Дана система уравнений $Ax = f$ порядка $n \times n$ с невырожденной матрицей A. Написать программу, решающую систему линейных алгебраических уравнений заданного пользователем размера (n – параметр программы), использующую численный алгоритм итерационного метода Зейделя:

$$\left(D + A^{(-)}\right)\left(x^{k+1} - x^k\right) + Ax^k = f$$

где $D, A^{(-)}$ - соответственно диагональная и нижняя треугольные матрицы, k - номер текущей итерации.

в случае использования итерационного метода верхней релаксации итерационный процесс имеет вид:

$$\left(D + \omega A^{(-)}\right)\frac{x^{k+1} - x^k}{\omega} + Ax^k = f$$

где ω - итерационный параметр (при $\omega = 1$ метод верхней релаксации переходит в метод Зейделя).

3. Цели и задачи практической работы

1) Научиться решать заданную СЛАУ итерационным методом Зейделя;

2) Разработать критерий остановки итерационного процесса, гарантирующий получение приближённого решения исходной СЛАУ с заданной точностью;

3) Изучить скорость сходимости итераций к точному решению задачи;

4) Правильность решения СЛАУ подтвердить системой тестов.

4. Описание алгоритма решения

Пусть дана система $Ax = f$. $D, A^{(-)}$ - соответствующие диагональная и нижнетреугольная матрицы. Можем переписать итерационный шаг, как

$$x^{k+1} = x^k + \omega \left(D + \omega A^{(-)}\right)^{-1} \left(f - Ax^k\right)$$

Или, в поэлементной форме, для $i = 1, \dots n$:

$$x_i^{k+1} = (1 - \omega)x_i^k + \frac{\omega}{a_{ii}} \left(f_i - \sum_{j<i} a_{ij} x_j^{k+1} - \sum_{j>i} a_{ij} x_j^k \right)$$

Выберем начальное приближение x^0 и параметр ω, а также параметр допустимого приближения. Далее будем повторять для всех $i = 1, \dots n$ верхнее присваивание, пока евклидова норма вектора невязки не станет меньше заданного значения.

Особенностью данного алгоритма является то, что он позволяет реализацию со всего одним дополнительным вектором, поскольку необязательно хранить весь вектор с предыдущей итерации.

5. Описание программы

Программа написана на C++11.

Краткое содержание реализации, существенное в этой части работы (подробнее – см. код):

- Чисто виртуальный класс BaseElementGenerator, предоставляющий интерфейс для задания системы уравнений;
- Класс ElementGenerator, реализующий заданные в условии формулы для получения очередного элемента матрицы и вектора;
- Класс EquationsSystem, описывающий систему уравнений. В качестве параметра конструктора принимает имя файла, содержащего систему или объект типа BaseElementGenerator. Имеет метод SuccessiveOverRelaxation() с необязательным параметром ω (1 по умолчанию), который производит решение системы методом верхней релаксациии. Так же реализован метод RelaxationResidual(), подсчитывающий норму вектора невязки системы при её решении этим методом.
- Классы Matrix и Vector, представляющие из себя рализации квадратной матрицы и вектора соответственно. Оба класса отнаследованы от BaseRightEquationsSystemPart, а также представляют стандартный набор операций для этих алгебраических объектов (перегружены операторы $*, *=, +, +=, -, -=$)
- Также реализованы вспомогательные функции Equal(double, double) и Zero(double), проверяющие числа на равенства и на равенство нулю, соответственно.
- Можно решить систему, заданную матрицей m, непосредственно обратившись к её методу SuccessiveOverRelaxation(), принимающему в качестве параметра Vector свободных коэффициентов и ω

6. Тесты

Проверка проводилась с помощью приложенного скрпита на python3.4, с использованием библиотеки numpy Установленная точность вычислений 0.001

6.1. Небольшие тесты

Тест 1

$$\begin{cases} 2x_1 + 5x_2 + 4x_3 + x_4 = 20, \\ x_1 + 3x_2 + 2x_3 + x_4 = 11, \\ 2x_1 + 10x_2 + 9x_3 + 7x_4 = 40, \\ 3x_1 + 8x_2 + 9x_3 + 2x_4 = 37. \end{cases}$$

Результат: Рассчитано программой:

$$\begin{pmatrix} 1.22884 \\ 1.88988 \\ 2.00155 \\ 0.0900723 \end{pmatrix}$$

Рассчитано при помощи пакета numpy:

$$\begin{pmatrix} 1.0 \\ 2.0 \\ 2.0 \\ -3.7 \times 10^{-15} \end{pmatrix}$$

Тест 2

$$\begin{cases} 6x_1 + 4x_2 + 5x_3 + 2x_4 = 1, \\ 3x_1 + 2x_2 + 4x_3 + x_4 = 3, \\ 3x_1 + 2x_2 - 2x_3 + x_4 = -7, \\ 9x_1 + 6x_2 + x_3 + 3x_4 = 2. \end{cases}$$

Результаты: Матрица является вырожденной.

Тест 3

$$\begin{cases} 2x_1 + x_2 + x_3 = 2, \\ x_1 + 3x_2 + x_3 + x_4 = 5, \\ x_1 + x_2 + 5x_3 = -7, \\ 2x_1 + 3x_2 - 3x_3 - 10x_4 = 14. \end{cases}$$

Результат: Рассчитано программой:

$$\begin{pmatrix} 0.999765 \\ 2.00009 \\ -1.99996 \\ -3.33812 \times 10^{-05} \end{pmatrix}$$

Рассчитано при помощи пакета numpy:

$$\begin{pmatrix} 1.0 \\ 2.0 \\ -2.0 \\ -0.0 \end{pmatrix}$$

6.2. Объёмные тесты

Тестирование проводилось для матриц порядка $n = 100$, заданных формулой

$$A_{ij} = \begin{cases} q_M^{i+j} + 0.1 \cdot (j - i), i \neq j, \\ (q_M - 1)^{i+j}, i = j. \end{cases}$$

где $q_M = 1.001 - 2 * M * 10^{-3}$, $i, j = 1, \ldots n$. M было принято равным 6(по условию). Элементы вектора свободных коэффициентов задавались формулой:

$$x \cdot \exp \frac{x}{i} \cdot \cos \frac{x}{i}$$

Программа выполнялась для значений x из промежутка с 2 до 5 с шагом 0.5. Значение ω подбиралось экспериментально для быстрейшей сходимости. Для проверки решений использовался приложенный скрипт check.py, в котором считалась евклидова норма вектора невязки. Погрешность проверки 0.001. При данных настройках метод верхней релаксации даёт большую погрешность, чем метод Гаусса, поскольку для ограничения времени выполнения была искусственно уменьшена точность вычислений.

7. Выводы

В результате выполненной работы установлено, что метод верхней релаксации позволяет решать системы линейных уравнений. Но в классическом варианте для больших плохо обусловленных матриц от выбора параметра релаксации будет существенно зависеть сходимость.

Плюсом метода является то, что он хорошо векторизуется и параллелизуется. В условиях однопоточного выполнения единственным плюсом данного метода, как итерационного, является то, что он позволяет использовать минимум дополнительной памяти. По времени же метод ограничен тем, что на каждом шаге алгоритма приходится заново вычислять невязку, а эта операция довольно затратна. Использование других индикаторов сходимости (Например разность текущей гипотезы с предыдущей) приводит, на примере матрицы, заданной в 6.2, к скорейшей сходимости, но гораздо меньшей точности результата.

С другой стороны, данный метод довольно универсален и может быть применён в широком круге различных итерационных методов.