

Санкт-Петербургский политехнический университет
Высшая школа прикладной математики
и вычислительной физики, ФизМех

Направление подготовки
«Прикладная математика и информатика»

Отчет по лабораторной работе №3
«Решение СЛАУ итерационными методами»

Дисциплина: «Численные методы»

Выполнил студент гр. 5030102/00003

Анищенко М.Д.

Преподаватель:

Курц В.В.

Санкт-Петербург 2021

Оглавление

1 Формулировка задачи и её формализация	3
1.1 Формализация задачи	3
1.2 Постановка задачи	3
2 Алгоритм метода и условия его применимости	4
2.1 Условия применимости метода	4
2.2 Алгоритм метода	4
3 Анализ задачи	5
4 Тестовый пример	5
5 Контрольные тесты	6
6 Модульная структура программы	6
7 Численный анализ	7
7.1 Зависимость числа итераций от степени диагонального преобладания	7
7.2 Сравнение времени работы итерационного и прямого метода	8
8 Общие выводы	8

Глава 1

Формулировка задачи и её формализация

Большинство расчётных математических задач сводится к решению систем линейных алгебраических уравнений. Существует два типа методов решения таких задач: прямые и итерационные. В данной работе будет рассмотрен итерационный метод, т.е. метод, который приводит к приближенному ответу за некоторое количество итераций, которое зависит от заданной точности.

1.1 Формализация задачи

Пусть дана система из n линейных уравнений с n неизвестными. $Ax = B$ – её матричная форма записи. Требуется найти вектор X , удовлетворяющий данному матричному уравнению и являющийся решением данной СЛАУ. В этой работе буде использован метод Якоби.

1.2 Постановка задачи

1. Исследовать условия применимости метода и алгоритм его работы.
2. Решить СЛАУ данным методом.
3. Построить график для анализа выбранного метода, а именно: зависимость числа итераций от степени диагонального преобладания матрицы.

Глава 2

Алгоритм метода и условия его применимости

2.1 Условия применимости метода

Достаточное условие сходимости: диагональное преобладание матрицы.

$$\sum_{j=1, j \neq i}^n |a_{ij}| < |a_{ii}|, i = 1, \dots, n$$

2.2 Алгоритм метода

Пусть имеется матрица $Ax = b$. Для метода простых итераций имеем формулу

$$x = (E - \alpha B^{-1} A)x + \alpha B^{-1} b$$

Метод Якоби - можно рассматривать как частный случай метода простых итераций, когда $\alpha = 1, B = D = \text{diag}(A)$.

Тогда получаем итерационную формулу

$$x^{(k+1)} = (E - D^{-1} A)x^{(k)} + D^{-1} b$$

Применяя здесь форму записи $x = Cx + g$, принимаем $C = (E - D^{-1} A)$, $g = D^{-1} b$. Матрицу C можно посчитать следующим образом:

$$C = \begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & -\frac{a_{23}}{a_{22}} & \dots & -\frac{a_{2n}}{a_{22}} \\ \dots & \dots & \dots & \dots & \dots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & -\frac{a_{n3}}{a_{nn}} & \dots & 0 \end{bmatrix}$$

Общее условие остановки:

$$\|x^{(k)} - x^{(k-1)}\| < \frac{1 - \|C\|}{\|C\|} \epsilon$$

При хорошо обусловленной матрице ($\|C\| \leq \frac{1}{2}$):

$$\|x^{(k)} - x^{(k-1)}\| < \epsilon$$

Глава 3

Анализ задачи

Существование СЛАУ зависит от введенной матрицы A . Для построения матрицы зададим её как $A = QDQ^T$, где Q – ортогональная матрица, D – диагональная. Затем разделим все элементы каждой строки (кроме диагонального) на сумму модулей элементов этой строки (опять же, в сумму не будет входить элемент главной диагонали). Таким образом мы гарантируем диагональное преобладание матрицы.

Глава 4

Тестовый пример

В качестве тестового примера возьмём следующую систему линейных алгебраических уравнений:

$$\begin{cases} 5x_1 + x_2 - 2x_3 = 4 \\ x_1 + 4x_2 - 1x_3 = 4, \\ -2x_1 - 1x_2 - 3x_3 = 0 \end{cases} \quad A = \begin{pmatrix} 5 & 1 & -2 \\ 1 & 4 & -1 \\ -2 & -1 & 3 \end{pmatrix}, B = \begin{pmatrix} 4 \\ 4 \\ 0 \end{pmatrix}, C = \begin{pmatrix} 0 & -0.2 & 0.4 \\ -0.25 & 0 & 0.25 \\ 0.667 & 0.333 & 0 \end{pmatrix}$$

Нетрудно сосчитать, что корни $x = (1, 1, 1)$. Проведем несколько первых итераций согласно методу Якоби и принимая точность равную 0.1:

$$\begin{aligned} x &= (0.8, 1, 0) \\ x &= (0.6, 0.8, 0.8667) \\ x &= (0.9867, 1.0667, 0.6667) \\ x &= (0.8533, 0.92, 1.0133) \\ x &= (1.0213, 1.04, 0.8756) \\ x &= (0.9422, 0.9635, 1.0275) \end{aligned}$$

Как мы видим, корни сходятся к требуемым значениям. На данной итерации погрешность составляет 0.1, но тем не менее, видно, что корни будут найдены с необходимой точностью спустя еще несколько итераций.

Глава 5

Контрольные тесты

Создадим 100 матриц размером 10×10 с числом обусловленности 10 и будем менять степень диагонального преобладания с помощью деления элементов не просто на сумму их модулей, а на сумму модулей, умноженную на некоторый коэффициент. Этот коэффициент будет принимать 100 разных значений, равномерно распределенных на отрезке $[0.1; 10]$

Глава 6

Модульная структура программы

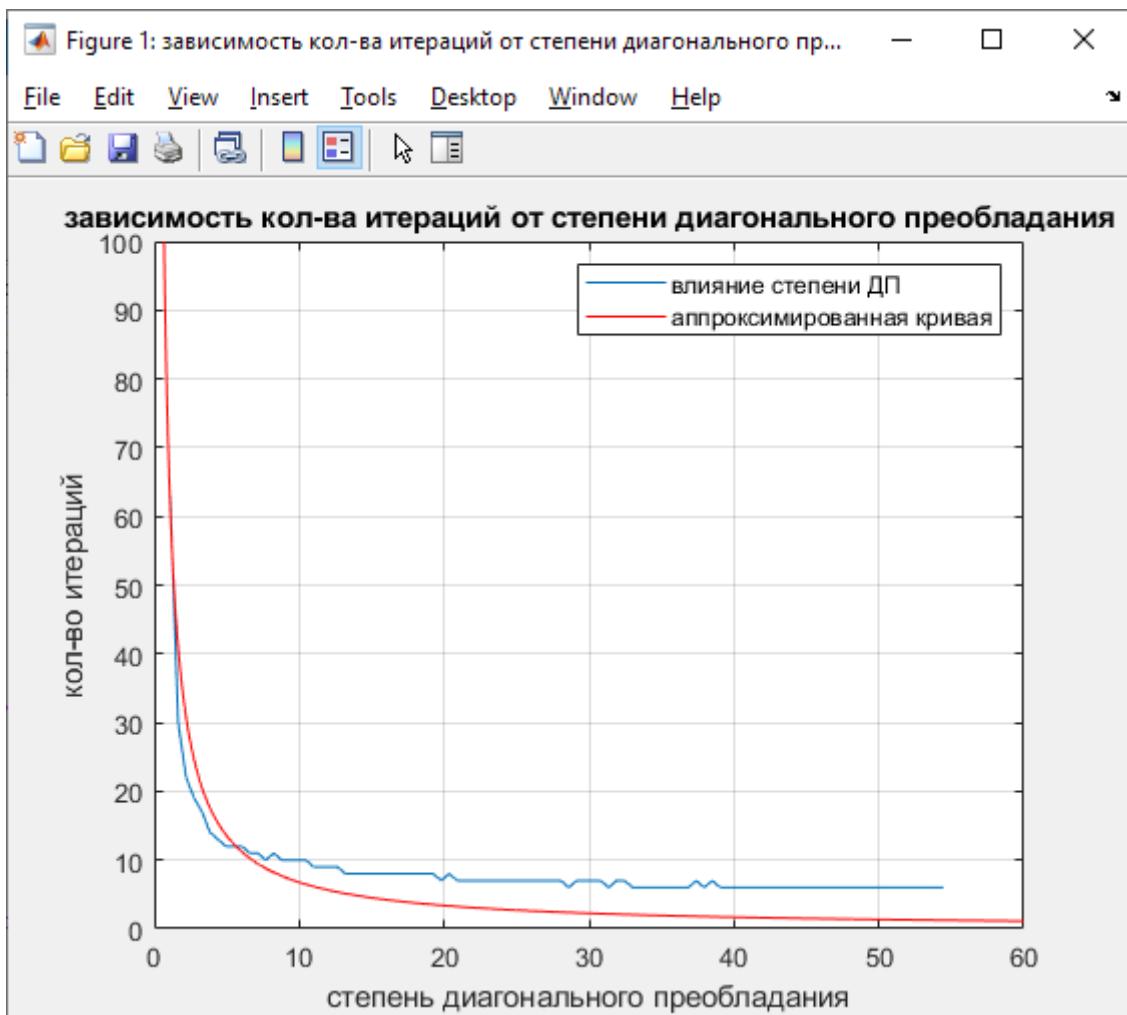
- `double** CreateMatrix(int n, int m)` – функция, выделяющая память для матрицы заданного размера.
- `void DestroyMatrix(double** A, int n)` - функция очистки памяти.
- `double** ReadMatrix(FILE* matrixfile, FILE* freekoefffile, int n)` – функция для чтения матрицы из файла.
- `void PrintVector(FILE* rootsfileforrec, double* arr, int n)` – функция для записи вектора корней в файл.
- `double** GetInverseDiagMatrix(double** diagMatrix, int n)` – функция получения матрицы, обратной к диагональной.
- `double* MxV(double** M, double* V, int n)` – функция умножения матрицы на вектор.
- `double** MxM(double** A, double** B, int n)` – функция матричного умножения.
- `double* VectorsSum(double* V1, double* V2, int n)` – функция для вычисления суммы векторов.
- `double** DiagMatrix(double** A, int n)` – функция для составления диагональной матрицы.
- `double NormVecDif(double* V1, double* V2, int n)` – функция для вычисления нормы вектора.
- `double* Jacobi(double** matrix, int n, double* average, int* iters)` – функция, реализующая метод Якоби
- `int main(void)` – основное тело программы.

Глава 7

Численный анализ

7.1 Зависимость числа итераций от степени диагонального преобразования

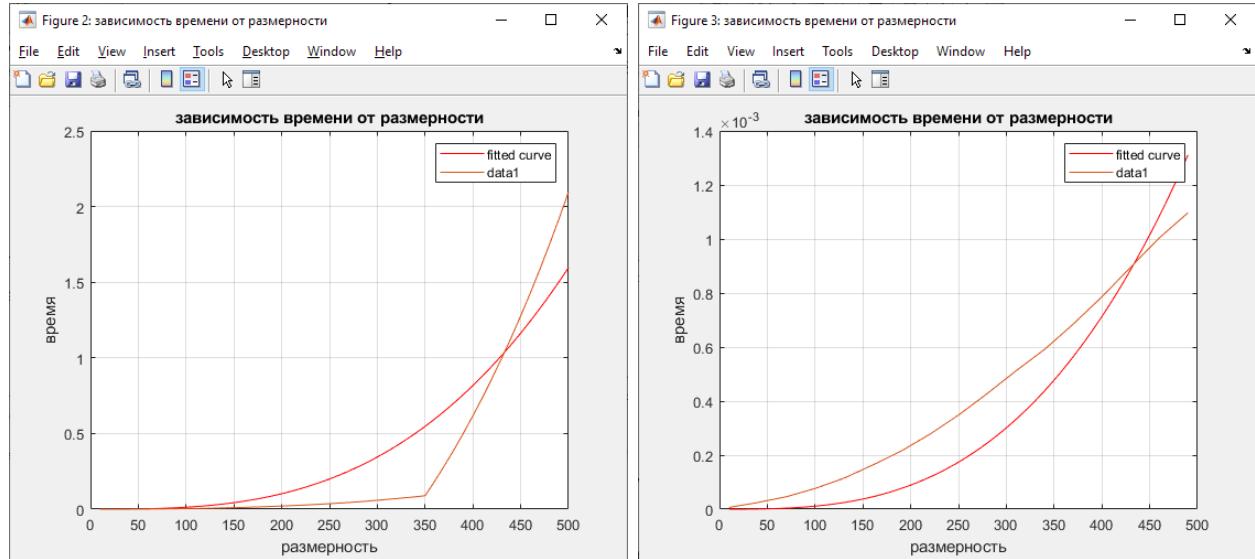
Воспользуемся пакетом MATLAB для построения данного графика.



По графику мы можем видеть, что количество итераций имеет обратную зависимость и приближается к кривой вида $y = \frac{k}{x}$.

7.2 Сравнение времени работы итерационного и прямого метода

Сравним методы прогонки и Якоби для одинаковых СЛАУ. Для метода Якоби зададим точность 10^{-15} . Для этого используем СЛАУ размерности от 10×10 до 500×500 с шагом в 10, замерим время за 100 запусков и усредним его.



Слева мы наблюдаем график для метода Якоби, а справа для метода прогонки. Как ни странно, но метод прогонки сработал быстрее, хотя обычно итерационные методы работают быстрее прямых. Возможно это связано с тем, что метод прогонки применим для трехдиагональных СЛАУ, поэтому он не требует больших объемов вычислений. В то время как метод прогонки взаимодействует с каждым элементом матрицы.

Глава 8

Общие выводы

Метод Якоби достаточно быстро находит решение СЛАУ с заданной точностью. При этом для хорошо обусловленной матрицы возрастание степени преобладания диагонального преобразования хорошо сказывается на количестве итераций. Однако метод Якоби проигрывает методу прогонки, вероятно, из-за специфики второго.