Отчёт

1 Постановка и решение задачи

Пусть задана матрица $A \in Mat_{n\times n}, A = A^T > 0$ и вектор $b \in \mathbb{R}^n$, и поставлена задача решить систему линейных уравнений $Ax = b$. Для того, чтобы сделать это максимально быстро, воспользуемся итерационным методом решения СЛУ, задаваемым уравнением:

$$\frac{x^{k+1} - x^k}{\tau_{k+1}} + Ax^k = b \tag{1}$$

Теорема 1. *Пусть* $A = A^T > 0,\ \lambda(A) \in [m; M],\ m > 0$. *Тогда циклический итерационный процесс (1) с параметрами* $\tau_1, \dots \tau_N$ *будет сходиться наилучшим образом при выборе*

$$\tau_k^{-1} = \frac{M+m}{2} + \frac{M-m}{2} \cdot \cos\frac{\pi(2k-1)}{2N},\ k = 1, \dots, N,\ \textit{причём}$$

$$\|x^N - x\|_2 \le \frac{2q_1^N}{1+q_1^{2N}} \cdot \|x^0 - x\|_2,\ \textit{где}\ q_1 = \frac{\sqrt{M}-\sqrt{m}}{\sqrt{M}+\sqrt{m}} < 1,$$

и сходимость будет при любом начальном условии.

Необходимое уточнение: величины τ_k необходимо взять в другом порядке, потому что иначе из-за вычислительной погрешности решение не будет сходиться. Делать это надо исходя из того, что норма ошибки на каждом этапе не должна возрастать:

$$\tau_1 : \max_\lambda |1 - \tau_1\lambda| \le 1$$

$$\tau_2 : \max_\lambda |(1 - \tau_1\lambda)(1 - \tau_2\lambda)| \le 1$$

$$\dots$$

$$\tau_N : \max_\lambda |(1 - \tau_1\lambda)\dots(1 - \tau_N\lambda)| \le 1$$

Также отдельно найдём решение системы с помощью метода Жордана с выбором наибольшего элемента по строке, чтобы оценивать решение, полученное итерационным методом.

2 Описание работы программы

При запуске программа считывает с клавиатуры способ задания матрицы A: 1 - из файла, 2 - по формуле, и размерность задачи **n**. После этого вызывается функция

int input(double* A, double* b, int input_method, int n, FILE* in);

которая заполняет матрицу A и вектор свободных членов b указанным способом. Далее запускается функция

int solve_jordan(int n, double* A, double* b, double* x, int* N);

которая находит решение СЛУ методом Жордана. Следующим шагом функция

void tau_filling(double* tau, double m, double M, int N);

заполняет массив **tau** значениями τ_k в правильном порядке. И, наконец, функция

int solve_iteratively(int n, int N, ..., double* help_vector);

находит решение описанным в первом пункте методом.