МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Н. Э. БАУМАНА

Факультет информатики и систем управления

Кафедра теоретической информатики и компьютерных технологий

Лабораторная работа №2

по курсу «Моделирование»

«Сравнительный анализ методов решения СЛАУ (метод Гаусса, метод Зейделя)»

Выполнил:

студент ИУ9-91

Выборнов А. И.

Руководитель:

Домрачева А. Б.

Москва 2015

1. Постановка задачи

Провести сравнительный анализ методов Гаусса и Зейделя для решения СЛАУ. Реализовать оба метода.

Под решением СЛАУ подразумевается решение системы уравнений $Ax = b$ показанной на рисунке 1.

$$\begin{cases} a_{11}x_1 + \ldots + a_{1n}x_n & = & b_1 \\ \ldots \\ a_{m1}x_1 + \ldots + a_{mn}x_n & = & b_m \end{cases} \iff Ax = b, \quad A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \ldots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}.$$

Рисунок 1 — Постановка задачи решения СЛАУ

2. Теоретическая часть

2.1. Метод Гаусса

Метод Гаусса — классический метод решения системы линейных алгебраических уравнений. Метод заключается в последовательном исключении переменных и происходит в два этапа:

1. Прямой ход — путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме, либо устанавливают, что система несовместна.

   На рисунке — 2 показана СЛАУ, приведённая к ступенчатому виду. Количество базисных переменных равно рангу матрицы A, то есть будет r базисные переменных. Пусть базисные будут переменные $x_{j_1}, ..., x_{j_r}$, остальные будем называть свободными (небазисными).

$$\begin{cases} \alpha_{1j_1}x_{j_1} + \alpha_{1j_2}x_{j_2} + \ldots + \alpha_{1j_r}x_{j_r} + \ldots + \alpha_{1j_n}x_{j_n} & = & \beta_1 \\ \alpha_{2j_2}x_{j_2} + \ldots + \alpha_{2j_r}x_{j_r} + \ldots + \alpha_{2j_n}x_{j_n} & = & \beta_2 \\ & \ldots & \\ \alpha_{rj_r}x_{j_r} + \ldots + \alpha_{rj_n}x_{j_n} & = & \beta_r \\ 0 & = & \beta_{r+1} \\ & \ldots & \\ 0 & = & \beta_m \end{cases}, \qquad \alpha_{1j_1}, \ldots, \alpha_{rj_r} \neq 0.$$

   Рисунок 2 — СЛАУ приведённая к ступенчатому виду

   Если $\exists \beta_i \neq 0 \colon i > r$, то рассматриваемая система не совместна, то есть не имеет решений. Если рассматриваемая система совместна, то переходим к следующему этапу.

2. Обратный ход — необходимо выразить все получившиеся базисные переменные через небазисные и построить фундаментальную систему решений, либо, если все переменные являются базисными, то выразить в численном виде единственное решение системы линейных уравнений

   На рисунке 3 показана СЛАУ, в которой базисные переменные выражены через свободные.

$$\begin{cases} x_{j1} + \widehat{\alpha}_{1j_2}x_{j2} + \ldots + \widehat{\alpha}_{1j_r}x_{j_r} & = & \widehat{\beta}_1 - \widehat{\alpha}_{1j_{r+1}}x_{j_{r+1}} - \ldots - \widehat{\alpha}_{1j_n}x_{j_n} \\ x_{j2} + \ldots + \widehat{\alpha}_{2j_r}x_{j_r} & = & \widehat{\beta}_2 - \widehat{\alpha}_{2j_{r+1}}x_{j_{r+1}} - \ldots - \widehat{\alpha}_{2j_n}x_{j_n} \\ & \ldots & \\ x_{j_r} & = & \widehat{\beta}_r - \widehat{\alpha}_{rj_{r+1}}x_{j_{r+1}} - \ldots - \widehat{\alpha}_{rj_n}x_{j_n} \end{cases}, \qquad \widehat{\beta}_i = \frac{\beta_i}{\alpha_{ij_i}}, \quad \widehat{\alpha}_{ij_k} = \frac{\alpha_{ij_k}}{\alpha_{ij_i}}$$

Рисунок 3 — СЛАУ, в которой базисные переменные выражены через свободные

Если все переменные базисные — то получаем единственное решение системы путём последовательного нахождения переменных двигаясь по СЛАУ снизу вверх. Если присутствуют свободные переменные, то мы получаем бесконечное множество решений: перебираем всевозможные значения свободных переменных и находим базисные двигаясь снизу вверх.

2.1.1. Преимущества

- Для матриц ограниченного размера менее трудоёмкий по сравнению с другими методами.
- Позволяет однозначно установить, совместна система или нет, и если совместна, найти её решение.
- Позволяет найти ранг матрицы

2.1.2. Недостатки

- Не оптимален - работает за $O(n^3)$: существуют алгоритмы перемножения матриц работающие ассимптотически быстрее.
- Вычислительно неустойчив для плохо обусловленных матриц.

2.2. Метод Зейделя

Метод Зейделя — итерационный метод решения системы линейных уравнений.

Рассмотрим матрицу A размера $n \times n$. Полагаем, что диагональные коэффициенты не нулевые, то есть $\forall i \ a_{ii} \neq 0$. Теперь решаем i-ое уравнение относительно x_i:

$$x_i = \frac{b_i - (a_{i1}x_1 + ... + a_{i(i-1)}x_{i-1} + a_{i(i+1)}x_{i+1} + a_{in}x_n)}{a_{ii}}$$

Найдя x_i для каждого i получаем систему:

$$\begin{cases} x_1 = \frac{b_1 - (a_{12}x_2 + a_{13}x_3 + a_{1n}x_n)}{a_{11}}, \\ x_2 = \frac{b_2 - (a_{11}x_1 + a_{13}x_3 + a_{1n}x_n)}{a_{22}}, \\ ... \\ x_n = \frac{b_n - (a_{11}x_1 + a_{12}x_2 + a_{1(n-1)}x_{n-1})}{a_{nn}}. \end{cases}$$

Возьмём некоторый начальный вектор $x^{(0)}$. Подставив его в полученную систему мы получим $x^{(1)}$. Аналогично с помощью $x^{(1)}$ находим $x^{(2)}$. Если будем учитывать полученные ранее решения, то система примет вид:

$$\begin{cases} x_1^{(k+1)} = \frac{b_1 - (a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + a_{1n}x_n^{(k)})}{a_{11}}, \\ x_2^{(k+1)} = \frac{b_2 - (a_{11}x_1^{(k+1)} + a_{13}x_3^{(k)} + a_{1n}x_n^{(k)})}{a_{22}}, \\ ... \\ x_n^{(k+1)} = \frac{b_n - (a_{11}x_1^{(k+1)} + a_{12}x_2^{(k+1)} + a_{1(n-1)}x_{n-1}^{(k+1)})}{a_{nn}}. \end{cases}$$

Повторяя этот итерационный процесс получаем последовательность приближений: $(x^{(0)}, x^{(1)}, ..., x^{(k)}, ...)$. Если эта последовательность сходится, то $x = \lim_{k\to\infty} x^k$ - является решением системы.

Разложим матрицу A на сумму матриц $A = L + D + U$, где $D -$ диагональная матрица, а матрицы L и U соответственно нижне и верхне диагоальные матрицы. Тогда необходимое условие сходимости метода Зейделя можно записать в виде:
если $|| - (L + D)^{-1}U|| < 1$, то метод Зейделя сходится.

2.2.1. Преимущества

- Быстро сходится.
- Сходимость доказана, всегда можно проверить сходится ли он.
- Одна итерация работает за $O(n^2)$, что на больших матрицах даёт преимущество перед методом Гаусса.

2.2.2. Недостатки

- Может медленно сходиться или даже расходиться на некоторых системах.

3. Реализация

В рамках лабораторной работы была написана программа на языке python, которая реализует методы Гаусса и Зейделя

3.1. Метод Гаусса

Ниже представлена метод на Python, реализующий метод Гаусса. Метод принимает на вход матрицы A и b и возвращает вектор решения x.

```
def gaussian(A, b):
    n = len(A)
    x = [0]*n

    #A = A/b
    A = [A[i]+[b[i]] for i in range(n)]

    for k in range(1, n):
        for j in range(k, n):
            m = A[j][k-1]*1.0 / A[k-1][k-1]
            for i in range(n+1):
                A[j][i] -= m*A[k-1][i]

    b = [A[i][n] for i in range(n)]

    for i in range(n-1, -1, -1):
        b[i] -= sum(A[i][j]*x[j] for j in range(i+1, n))
        x[i] = b[i]*1.0 / A[i][i]

    return x
```

3.2. Метод Зейделя

Ниже представлена метод на Python, реализующий метод Зейделя. Метод принимает на вход матрицы A и b, а также число eps, характеризующее точность решения, и возвращает вектор решения x.

```
def seidel(A, b, eps):
    n = len(A)

    #Ax=b
    x = [0] * n
    converge = False
    while not converge:
        p = copy.copy(x)
        for i in range(n):
            s = sum(A[i][j] * x[j] for j in range(i))
            s += sum(A[i][j] * p[j] for j in range(i+1,n))
            x[i] = (b[i] - s)*1.0 / A[i][i]

        converge = sqrt(sum((x[i]-p[i])**2 for i in range(n))) < eps
    return x
```

4. Выводы

При реализации на ЭВМ метод Зейделя превосходит метод Гаусса по скорости работы, но требует дополнительной проверки на сходимость. Также, в отличие от метода Гаусса метод Зейделя позволяет находить значения с заданной точностью и уточнять полученные ранее решения.