Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Московский государственный технический университет имени Н.Э. Баумана

Лабораторная работа №1
по дисциплине
«Численные методы»

Студент группы ИУ9-62 Белогуров А.А.

Преподаватель Домрачева А.Б.

Москва, 2017

Постановка задачи:

Дано: $A\overline{x} = \overline{d}$, где $A \in \mathbb{R}^{n\times n}$ и $\overline{x}, \overline{d} \in \mathbb{R}^n$. Найти решение СЛАУ с помощью методов: Гаусса и Зейделя. В качестве исходных данных взять следующие матрицы (1):

$$A\overline{x} = \overline{d} \quad \Leftrightarrow \quad \begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 5 \end{pmatrix} \tag{1}$$

Теоретические сведения:

Решение СЛАУ методом Гаусса:

1. Составить расширенную матрицу $(A|d)$ и привести её к ступенчатому виду(2):

$$(\widetilde{A}|\widetilde{d}) = \left(\begin{array}{ccccc|c} 1 & \widetilde{a_{12}} & \widetilde{a_{13}} & \dots & \widetilde{a_{1n}} & \widetilde{d}_1 \\ 0 & 1 & \widetilde{a_{23}} & \dots & \widetilde{a_{2n}} & \widetilde{d}_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & \widetilde{d}_n \end{array}\right) \tag{2}$$

2. Найти приближенный вектор $\overline{x}^*$. При вычислении $\widetilde{a_{ij}}$ и $\widetilde{d}_i$ с плавающей точкой возникает погрешность $\overline{e}$ (3):

$$- \quad \begin{array}{c} A\overline{x}^* = \overline{d}^* \\ A\overline{x} = \overline{d} \\ \hline A(\overline{x}^* - \overline{x}) = (\overline{d}^* - \overline{d}) \end{array}$$

или:

$$A\overline{e} = \overline{r} \quad \Leftrightarrow \quad \boxed{\overline{e} = A^{-1}\overline{r}} \tag{3}$$

3. Тогда исходный вектор $\overline{x} = \overline{x}^* - \overline{e}$.

Решение СЛАУ методом Зейделя:

1. Преобразовать систему $A\overline{x} = \overline{d}$ к виду $x = B\overline{x} + \overline{c}$, где

$$b_{ij} = -\frac{a_{ij}}{a_{ii}}, b_{ii} = 0, i \neq j$$

$$c_i = \frac{d_i}{a_{ii}}, a_{ii} \neq 0$$

2. Задать начальное приближение решения $x^{(0)}$ произвольно, определить k и малое положительное число ε:

$$x^{(0)} = 0, k = 0, \varepsilon = 0.01$$

3. Произвести расчеты по формуле (4) и найти $x^{(k+1)}$.

$$\begin{cases} x_1^{(k+1)} = b_{12}x_2^{(k)} + ... + b_{1n}x_n^{(k)} + c_1 \\ x_2^{(k+1)} = b_{21}x_1^{(k+1)} + ... + b_{2n}x_n^{(k)} + c_2 \\ \vdots \\ x_n^{(k+1)} = b_{n1}x_1^{(k+1)} + ... + b_{nn-1}x_{n-1}^{(k+1)} + c_1 \end{cases} \tag{4}$$

4. Если выполнено условие окончания $\|x^{(k-1)} - x^{(k)}\| \leq \varepsilon$ (5), то завершить процесс и в качестве приближенного решения принять $x_* \cong x^{(k)}$. Иначе положить $k = k + 1$ и перейти к пункту 3.

$$\|x^{(k-1)} - x^{(k)}\| = \sqrt{\sum_{i=1}^{n}(x_i^{(k-1)} - x_i^{(k)})^2} \leq \varepsilon \tag{5}$$

Практическая реализация:

Листинг 1: Решение СЛАУ методом Гаусса

```
import numpy
```

```

matrix_f = [[1,2,0,0], [2,-1,1,0], [0,1,-1,1], [0,0,1,1]]
matrix = [[1,2,0,0], [2,-1,1,0], [0,1,-1,1], [0,0,1,1]]

d_1 = [5,3,3,7]
d = [5,3,3,7]

def printMatrix():
  for j in range(len(matrix)):
    print(str(matrix[j]) + "   " + str(d[j]))
  print()

def maxElement(j_cur, i_cur):
  j = j_cur
  max = j_max = -1
  for j in range(j_cur, len(matrix)):
    if abs(matrix[j][i_cur]) > max and matrix[j][i_cur] != 0:
      max = matrix[j][i_cur]
      j_max = j

  if j_max != j_cur:
    matrix[j_max], matrix[j_cur] = matrix[j_cur], matrix[j_max]
    d[j_max], d[j_cur] = d[j_cur], d[j_max]


# ---------------------------------------------------
# reduction to triangular form
j_cur = i_cur = 0

while True:
  if j_cur == len(matrix) or i_cur == len(matrix[j_cur]):
    break

  maxElement(j_cur, i_cur)
  printMatrix()


  k = (matrix[j_cur][i_cur])
  if k == 0:
    i_cur += 1
    continue
  for i in range(i_cur, len(matrix[j_cur])):
```

```
    matrix[j_cur][i] /= k
  d[j_cur] /= k


  for j in range(j_cur + 1, len(matrix)):
    k = - matrix[j][i_cur] / matrix[j_cur][i_cur]
    for i in range(i_cur, len(matrix[j_cur])):
      matrix[j][i] += matrix[j_cur][i] * k
    d[j] += d[j_cur] * k

  printMatrix()

  j_cur += 1
  i_cur += 1

# ---------------------------------------------------
# the calculation of the vector x_2
x_2 = [0 for i in range(0, len(d))]

j_cur = len(x_2)-1
i_cur = len(matrix[j_cur]) - 1

for j in range(j_cur, -1, -1):
  sum = 0
  k = j + 1
  for i in range(i_cur + 1, len(matrix[j])):
    sum += matrix[j][i] * x_2[k]
    k += 1
  try:
    x_2[j] = (d[j] - sum)/matrix[j][i_cur]
  except ZeroDivisionError:
    print("Coefficient X[" + j + "] may be any")
    exit(0)
  i_cur -= 1

print("X_2: " + str(x_2))

# ---------------------------------------------------
# matrix product to find D_2
d_2 = [0 for i in range(0, len(d))]

for j in range(0, len(matrix_f)):
```

```
  sum = 0
  k = 0
  for i in range(0, len(matrix_f[j])):
    sum += matrix_f[j][i] * x_2[k]
    k += 1
  d_2[j] = sum

print("D_2: " + str(d_2))

# ---------------------------------------------------
# finding the difference D_2 - D = r

r = [d_2[i] - d_1[i] for i in range(0, len(matrix_f))]
print("r: " + str(r))
print()

# ---------------------------------------------------
# inverse matrix matrix_rev
matrix_rev = numpy.matrix(matrix_f).I
matrix_rev = matrix_rev.tolist()

# ---------------------------------------------------
# error e = matrix_rev * r
e = [0 for i in range(0, len(d))]

for j in range(0, len(matrix_rev)):
  sum = 0
  k = 0
  for i in range(0, len(matrix_rev[j])):
    sum += matrix_rev[j][i] * r[k]
    k += 1
  e[j] = sum

print("e: " + str(e))

# ---------------------------------------------------
# X = X_2 - e
x = [x_2[i] - e[i] for i in range(0, len(matrix_f))]
print("X: " + str(x))
```

Листинг 2: Решение СЛАУ методом Зейделя

```python
from math import *

matrix_a = [[4, 1, 0, 0],
            [1, 4, 1, 0],
            [0, 1, 4, 1],
            [0, 0, 1, 4]]
matrix_b = [[0 for i in range(len(matrix_a))]
  for j in range(len(matrix_a[0]))]

d = [5, 6, 6, 5]
x = [0 for i in range(len(d))]
x_n = [0 for i in range(len(d))]
c = [0 for i in range(len(d))]

e = 0.001

def test():
  sum = 0
  for i in range(len(x_n)):
    sum += (x_n[i] - x[i]) ** 2
  if sqrt(sum) <= e:
    return True
  return False


for j in range(len(matrix_b)):
  for i in range(len(matrix_b[j])):
    if i != j:
      matrix_b[j][i] = - matrix_a[j][i] / matrix_a[j][j]
  c[j] = d[j] / matrix_a[j][j]

#print(matrix_b)

x_test = 0
iteration = 0
while True:
  x_n = x.copy()
  iteration += 1
  for k in range(len(matrix_b)):
    j_cur = k
    sum = 0
    for i in range(len(matrix_b[j_cur])):
```

```
      if k > i:
        x_test = x_n[i]
      else:
        x_test = x[i]
      sum += matrix_b[j_cur][i] * x_test
    x_n[k] = sum + c[k]
  print(x_n)
  if test():
    break

  x = x_n

print(iteration)
```

Результаты:

Решение СЛАУ методом Гаусса.

В качестве исходных данных возьмем матрицы (1). Тогда в результате работы программы (Листинг 1) получены следующие значения:

$$e = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{6}$$

Как видно выше (6), не всегда вектор $\overline{e}$ может содержать погрешность. Для того, чтобы вектор $\overline{e}$ был ненулевым, возьмем другие исходные значения(7) и получим следующие результаты(8):

$$A\overline{x} = \overline{d} \quad \Leftrightarrow \quad \begin{pmatrix} 1 & 2 & 0 & 0 \\ 2 & -1 & -1 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ 3 \\ 7 \end{pmatrix} \tag{7}$$

$$e = \begin{pmatrix} 3.947459643111668e^{-16} \\ -1.973729821555834e^{-16} \\ -9.86864910777917e^{-16} \\ 9.86864910777917e^{-16} \end{pmatrix} \quad x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \tag{8}$$

Решение СЛАУ методом Зейделя.

Для тестирования работы программы (Листинг 2) была взята система (1).

Для $\varepsilon = 0.01$ было выполнено 5 итераций. Результаты практически аналогичны методу Гаусса (9):

$$x = \begin{pmatrix} 1.0003395080566406 \\ 0.9997768402099609 \\ 1.0000903606414795 \\ 0.9999774098396301 \end{pmatrix} \quad (9)$$

Выводы:

В ходе выполнения лабораторной работы были рассмотрены итерационные методы решения СЛАУ: метод Гаусса и метод Зейделя, так же для каждого из них была написана реализация на языке программирования Python.

Для метода Гаусса можно отметить довольно высокую вычислительную сложность и высокое накопление погрешности в результате приведения матрицы к треугольному виду и преобразование матрицы A к обратной A^{-1}.

Для метода Зейделя характерно довольное малое количество итераций для нахождения решения и простота вычислений. Но самое главное то, что данный метод уступает по точности методу Гауса.