Федеральное государственное автономное образовательное учреждение высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ "ВЫСШАЯ ШКОЛА ЭКОНОМИКИ"

Московский институт электроники и математики имени А. Н. Тихонова
Программа "Прикладная математика"

Нигматуллин Роман Максимович

Лабораторная работа

Решение систем алгебраических линейных уравнений прямыми методами, теория возмущений.

3 курс, группа БПМ203

Преподаватель:
Брандышев Петр Евгеньевич

Москва, 2021 г.

Содержание

1 Погрешность решения при изменении правой части (3.1.16)

1.1 Формулировка задачи

Дана система уравнений $Ax = b$ порядка n. Исследовать зависимость погрешности решения x от погрешностей правой части системы b. 1. Задать матрицу системы A и вектор правой части b. Используя встроенную функцию, найти решение x системы $Ax = b$ с помощью метода Гаусса. 2. С помощью встроенной функции вычислить число обусловленности матрицы A. 3. Принимая решение x, полученное в п. 1, за точное, вычислить вектор $d = (d_1, ..., d_n)^T$:

$$d_i = \frac{||x - x^i||}{||x||}, i = 1, ..., n$$

относительных погрешностей решений x^i систем $Ax^i = b^i, i = 1, ..., n$, где компоненты векторов определяются по формулам:

$$b_k^i = b_k + \Delta, k = i$$

$$b_k^i = b_k, k \neq i$$

4. На основе вычисленного вектора d построить гистограмму. По гистограмме определить компоненту b^m вектора b, которая оказывает наибольшее влияние на погрешность решения. 5. Оценить теоретически погрешность решения x^m по формуле, сравнить значение $\delta(x^m)$ со значением практической погрешности d_m. Объяснить полученные результаты.

$$\delta(x^m) \leq cond(A) \times \delta(b^m)$$

$$N = 16, n = 5$$

$$c_{ij} = 0.1Nij$$

$$a_{ij} = \frac{100}{(3 + 0.3c)^5}$$

1.2 Код на Python

```python
import numpy as np  # type: ignore
import matplotlib.pyplot as plt  # type: ignore


N, n = 16, 5
C, b = np.zeros((n, n), dtype=float), np.full(n, fill_value=N, dtype=float)

for i in range(n):
    for j in range(n):
        C[i, j] = 0.1 * N * (i + 1) * (j + 1)

A = 100 / (3 + 0.3 * C) ** 5

x = np.linalg.solve(A, b)

cond_value = np.linalg.cond(np.abs(A), p=np.inf)
delta = 0.1

x_modified = np.empty((n, n))
for i in range(n):
    b_modified = b.copy()
    b_modified[i] += delta
    x_modified[i] = np.linalg.solve(A, b_modified)
d = np.array([np.linalg.norm(x - x_i, ord=np.inf) / np.linalg.norm(x, ord=np.inf)
              for x_i in x_modified])

plt.figure(figsize=(6, 5))
plt.bar(range(n), d)
plt.xlabel('')
plt.savefig('plots/cond_precision.png', dpi=300)

d_argmax = np.argmax(d)
b_modified = b.copy()
b_modified[d_argmax] += delta


with np.printoptions(precision=5):
    rel_delta = (np.linalg.norm(b_modified - b, ord=np.inf)
                 / np.linalg.norm(b, ord=np.inf))
    print(f'm = {d_argmax + 1}')
    print(f'd = {d}')
    print(f'delta(x^m) = {d[d_argmax]}')
```

```
print(f'delta(b^m) = {rel_delta}')
print(f'cond(A) = {cond_value}')
cmp_sign = '<=' if d[d_argmax] <= rel_delta * cond_value else '>'
print(f'{d[d_argmax]} {cmp_sign} {rel_delta * cond_value}')
print(f'delta(x^m) {cmp_sign} cond(A) * delta(b^m)')
```

1.3 Результат работы программы

Для теоретической оценки использовался расчет внутри кода, который рассчитал необходимые величины и вывел итоговое неравенство.

```
m = 4
d = [0.00012 0.00372 0.02673 0.06374 0.04686]
delta(x^m) = 0.06373935809151422
delta(b^m) = 0.006250000000000089
cond(A) = 1312465.3402483896
0.06373935809151422 <= 8202.908376552552
delta(x^m) <= cond(A) * delta(b^m)
```

Выявление максимальной погрешности сделано также при помощи кода, но визуально заметно, что $m = 4$. Заметно, что теоретическая оценка погрешности (правая часть неравенства) сильно больше, чем полученное на практике значение, так как готовые методы довольно точны.

1.4 График погрешности в зависимости от возмущения правой части

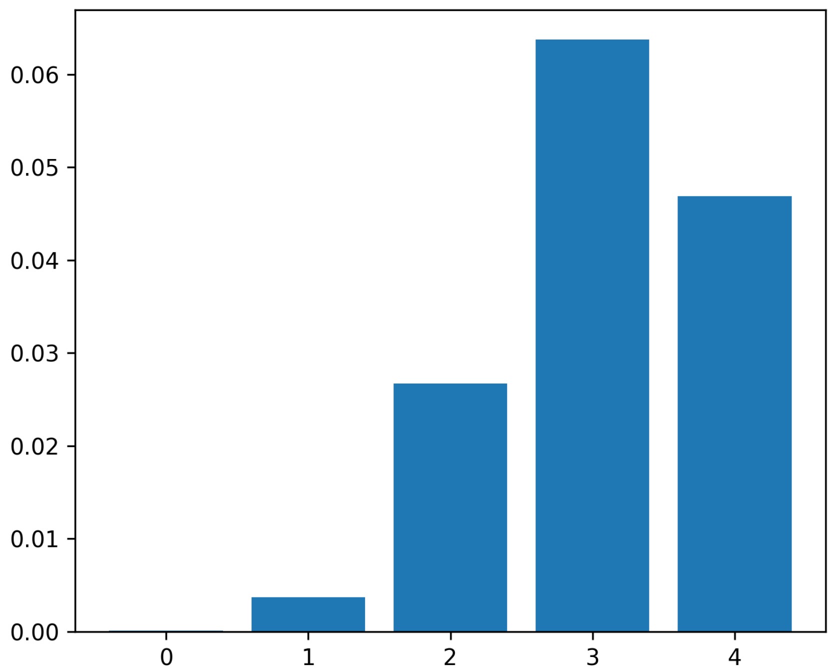

Рис. 1: Значения d_m

2 Погрешность решения при изменении матрицы A (3.2)

2.1 Формулировка задачи

Для системы уравнений $Ax = b$ из задачи №1 исследовать зависимость погрешности решения системы от погрешностей коэффициентов матрицы A (аналогично задаче 3.1). Теоретическая оценка погрешности в этом случае имеет вид:

$$\delta(x^h) \leq cond(A) \times \delta(A^h)$$

Где x^h - решение системы с возмущенной матрицей A^h.

2.2 Код на Python

```
import numpy as np  # type: ignore
import matplotlib.pyplot as plt  # type: ignore


N, n = 16, 5
C, b = np.zeros((n, n), dtype=float), np.full(n, fill_value=N, dtype=float)

for i in range(n):
    for j in range(n):
        C[i, j] = 0.1 * N * (i + 1) * (j + 1)

A = 100 / (3 + 0.3 * C) ** 5
x = np.linalg.solve(A, b)
cond_value = np.linalg.cond(A, p=np.inf)

delta = 0.1
x_modified = {}
for i in range(n):
    for j in range(n):
        A_modified = A.copy()
        A_modified[i, j] += delta
        x_modified[(i, j)] = np.linalg.solve(A_modified, b)
d = {key: np.linalg.norm(x - x_i, ord=np.inf) / np.linalg.norm(x, ord=np.inf)
     for key, x_i in x_modified.items()}

plt.figure(figsize=(10, 5))
plt.bar([str(el) for el in d.keys()], d.values())
plt.xlabel('')
plt.xticks(rotation=90)
plt.savefig('plots/matrix_precision.png', dpi=300)

d_i, d_j = max(d, key=d.get)  # type: ignore
A_modified = A.copy()
A_modified[d_i, d_j] += delta
rel_delta = (np.linalg.norm(A_modified - A, ord=np.inf)
             / np.linalg.norm(A, ord=np.inf))
```

```
cmp_sign = '<=' if d[(d_i, d_j)] <= rel_delta * cond_value else '>'

print(f'i, j = {d_i, d_j}')
print(f'delta(x^h) = {d[(d_i, d_j)]}')
print(f'delta(A^h) = {rel_delta}')
print(f'cond(A) = {cond_value}')
print(f'{d[(d_i, d_j)]} {cmp_sign} {rel_delta * cond_value}')
print(f'delta(x^h) {cmp_sign} cond(A) * delta(A^h)')
```

2.3 Результат работы программы

```
i, j = (1, 0)
delta(x^h) = 3.068157134210475
delta(A^h) = 0.24210681618175434
cond(A) = 1312465.3402483896
3.068157134210475 <= 317756.80487644055
delta(x^h) <= cond(A) * delta(A^h)
```

Аналогично выявление максимальной погрешности сделано также при помощи кода, но максимум заметен и визуально. Также аналогично предыдущему заданию реальная погрешность получилась на несколько порядков меньше верхней теоретической оценки.

2.4 График погрешности в зависимости от возмущения правой части

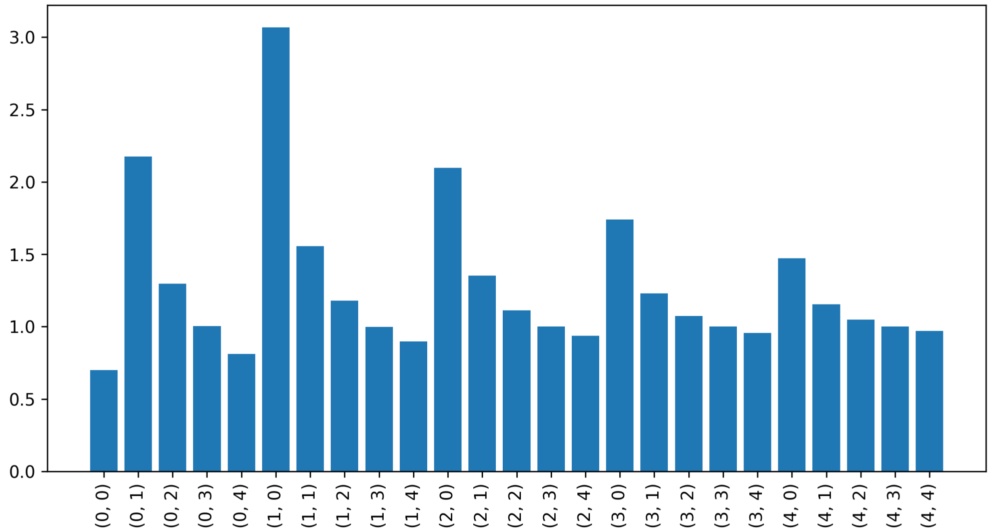

Рис. 2: Значения d_{ij}

3 Метод прогонки (3.10.4)

3.1 Формулировка задачи

Дана система уравнений $Ax = b$ порядка n с разреженной матрицей A. Решить систему методом прогонки. Размер матрицы $n = 50$, на главной диагонали элементы равны 100, на первой наддиагонали элементы равны 1, на 1 поддиагонали элементы равны 2.

$$b_i = i * e^{10/i} cos(9/i)$$

3.2 Код на Python

```
import numpy as np  # type: ignore


def tridiagonal_solve(a: np.array,
                      b: np.array,
                      c: np.array,
                      d: np.array) -> np.array:
    for it in range(1, len(d)):
        mc = a[it - 1] / b[it - 1]
        b[it] = b[it] - mc * c[it - 1]
        d[it] = d[it] - mc * d[it - 1]
    xc = b
    xc[-1] = d[-1] / b[-1]
    for il in range(len(d) - 2, -1, -1):
        xc[il] = (d[il] - c[il] * xc[il + 1]) / b[il]
    return xc


# for matrix from task itself
n = 50
A = np.full(n, fill_value=2, dtype=float)
A[0] = 0
C = np.full(n, fill_value=1, dtype=float)
C[n-1] = 0
B = np.full(n, fill_value=100, dtype=float)
D = (np.arange(1, n+1) * np.exp(10 / np.arange(1, n+1))
     * np.cos(9 / np.arange(1, n+1)))

print('Main diagonal B:')
print(B)
print('Lower subdiagonal A:')
print(A[1:])
print('Upper subdiagonal C:')
print(C[:-1])
print('Coefficients D:')
print(D)
print('Solution X:')
print(tridiagonal_solve(A, B, C, D))

# for smaller matrix, example and proof
n = 3
A = np.full(n, fill_value=4, dtype=float)
A[0] = 0
```

```python
C = np.full(n, fill_value=3, dtype=float)
C[n-1] = 0
B = np.full(n, fill_value=5, dtype=float)
D = np.arange(1, n+1)
print('Solution for smaller matrix')
print('Main diagonal B:')
print(B)
print('Lower subdiagonal A:')
print(A[1:])
print('Upper subdiagonal C:')
print(C[:-1])
print('Coefficients D:')
print(D)
print('Solution X:')
print(tridiagonal_solve(A, B, C, D))
```

3.3 Результат работы программы

```
Main diagonal B:
[100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100.
 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100.
 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100. 100.
 100. 100. 100. 100. 100. 100. 100. 100.]
Lower subdiagonal A:
[2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2.]
Upper subdiagonal C:
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1.]
Coefficients D:
[-2.00689795e+04 -6.25697410e+01 -8.32532949e+01 -3.06108855e+01
 -8.39404512e+00  2.24710446e+00  8.21466622e+00  1.20396314e+01
  1.47716414e+01  1.68971108e+01  1.86627560e+01  2.02031814e+01
  2.15963248e+01  2.28895882e+01  2.41130141e+01  2.52863204e+01
  2.64228428e+01  2.75318378e+01  2.86198783e+01  2.96917258e+01
  3.07508916e+01  3.18000060e+01  3.28410682e+01  3.38756167e+01
  3.49048499e+01  3.59297108e+01  3.69509488e+01  3.79691649e+01
  3.89848448e+01  3.99983842e+01  4.10101077e+01  4.20202830e+01
  4.30291327e+01  4.40368427e+01  4.50435689e+01  4.60494429e+01
  4.70545764e+01  4.80590642e+01  4.90629877e+01  5.00664164e+01
  5.10694103e+01  5.20720212e+01  5.30742940e+01  5.40762678e+01
  5.50779767e+01  5.60794503e+01  5.70807149e+01  5.80817933e+01
  5.90827057e+01  6.00834700e+01]
Solution X:
[-2.00683620e+02 -6.17524484e-01 -8.17292669e-01 -2.88979069e-01
 -7.83932080e-02  2.32338238e-02  8.05084970e-02  1.17348877e-01
  1.43726608e-01  1.64282788e-01  1.81378844e-01  1.96306101e-01
  2.09813553e-01  2.22357342e-01  2.34226872e-01  2.45612245e-01
```

```
  2.56642124e-01  2.67405866e-01  2.77966952e-01  2.88371388e-01
  2.98653112e-01  3.08837557e-01  3.18944056e-01  3.28987492e-01
  3.38979458e-01  3.48929082e-01  3.58843617e-01  3.68728881e-01
  3.78589575e-01  3.88429532e-01  3.98251893e-01  4.08059254e-01
  4.17853770e-01  4.27637240e-01  4.37411175e-01  4.47176852e-01
  4.56935351e-01  4.66687592e-01  4.76434362e-01  4.86176336e-01
  4.95914095e-01  5.05648141e-01  5.15378909e-01  5.25106779e-01
  5.34832080e-01  5.44555101e-01  5.54276088e-01  5.63995876e-01
  5.73653523e-01  5.89361630e-01]
Solution for smaller matrix
Main diagonal B:
[5. 5. 5.]
Lower subdiagonal A:
[4. 4.]
Upper subdiagonal C:
[3. 3.]
Coefficients D:
[1 2 3]
Solution X:
[0.09846154 0.16923077 0.38461538]
```