Лабораторная работа №2, вариант №23

РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ ПРЯМЫМИ МЕТОДАМИ. ТЕОРИЯ ВОЗМУЩЕНИЙ

Задания: 3.1.23, 3.3.6, 3.8.5

Студент:
Паршина Софья Романовна
3 курс, группа БПМ213

Содержание

1 Погрешность решения при изменение правой части(3.1.23)

1.1 Формулировка задачи

Задача 3.1. Дана система уравнений $Ax=b$ порядка n. Исследовать зависимость погрешности решения x от погрешностей правой части системы b.

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

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 $d=(d_1,\dots,d_n)^T$,

$d_i = \dfrac{\| x - x^i \|_\infty}{\| x \|_\infty}$, $i=1, \dots, n$, относительных погрешностей решений x^i систем $Ax^i = b^i$, $i=1, \dots, n$, где

компоненты векторов b^i вычисляются по формулам: $b_k^i = \begin{cases} b_k + \Delta, & k = i, \\ b_k, & k \neq i, \end{cases}$ $k=1, \dots, n$

($\Delta -$ произвольная величина погрешности).

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:

$\delta(x^m) \le cond(A) \cdot \delta(b^m)$. Сравнить значение $\delta(x^m)$ со значением практической погрешности d_m. Объяснить полученные результаты.

$$N = 23, n = 5$$

$$c_ij = 0.1 * N * i * j$$

$$a_ij = \frac{11.7}{(1+c)^7}$$

1.2 Код на Python

```
# -*- coding: utf-8 -*-
"""cond.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/19EXKt2o34TMTK8rp70kt9ZaBuEGISvwX
"""

N= 23
n = 5
C= np.zeros((n, n), dtype=float)
b = np.full(n, 23, dtype=float)

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

A = 11.7 /(1+C)**7
x = np.linalg.solve(A, b)
cond_A = np.linalg.cond(np.abs(A), p=np.inf)

delta = 0.1 #произвольная
x_new = np.empty((n, n))

for i in range(n):
    b_new = b.copy()
    b_new[i] += delta
    x_new[i] = np.linalg.solve(A, b_new)

d = np.array([np.linalg.norm(x - x_i, ord=np.inf) / np.linalg.norm(x, ord=np.inf)
              for x_i in x_new])

plt.figure(figsize=(6, 5))
plt.bar(['1', '2', '3','4','5'], d)
plt.xlabel('m')
plt.ylabel('d_m')
plt.savefig('cond_precision.png', dpi=300)

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


rel_delta = (np.linalg.norm(b_new - b, ord=np.inf)
             / np.linalg.norm(b, ord=np.inf))
print(f'm = {d_argmax + 1}')
print(f'd = {d}\n')
print(f'delta(x^m) = {d[d_argmax]}')
```

```
print(f'delta(b^m) = {rel_delta}')
print(f'cond(A) = {cond_A}\n')
print(f'{d[d_argmax]} <= {rel_delta * cond_A}')
print(f'delta(x^m) <= cond(A) * delta(b^m)')
```

1.3 Результат работы программы

```
m = 5
d = [3.62570934e-09 5.97569006e-06 3.68975948e-04 3.66129297e-03
 7.64611517e-03]

delta(x^m) = 0.007646115173505971
delta(b^m) = 0.004347826086956583
cond(A) = 53358232352997.17

0.007646115173505971 <= 231992314578.25186
delta(x^m) <= cond(A) * delta(b^m)
```

Практическая погрешность была взята $\Delta = 0.1$, а $\delta(x^m) = 0.0076...$ -> теоретическая погрешность решения x^m меньше. Из результатов работы программы можно увидеть, что формула для теоретической оценки верна, так как правая часть существенно больше левой.

1.4 График погрешности в зависимости от возмущения правой части

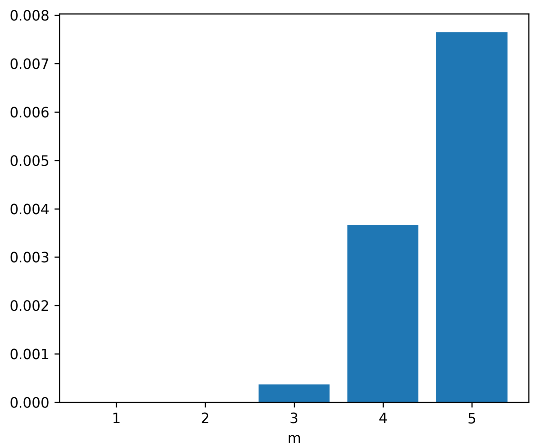

Рис. 1: Наибольшее влияние на погрешность решения вносит компонента b_m, $m = 5$

2 Число обусловленности матрицы(3.3.6)

2.1 Формулировка задачи

Задача 3.3. Дана матрица *A*. Найти число обусловленности матрицы, используя вычислительный эксперимент.

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Выбрать последовательность линейно независимых векторов $b^i, i = 1,\ldots k$. Решить *k* систем уравнений $Ax^i = b^i$, *i=1, ..., k*, используя встроенную функцию.

2. Для каждого найденного решения x^i вычислить отношение $\dfrac{\| x^i \|}{\| b^i \|}$, *i*=1, ...,*k*.

3. Вычислить норму матрицы A^{-1} по формуле $\| A^{-1} \| \approx \max\limits_{1 \le i \le k} \dfrac{\| x^i \|}{\| b^i \|}$, вытекающей из неравенства $\|x\| \le \left\|A^{-1}\right\| \cdot \|b\|$.

4. Вычислить число обусловленности матрицы *A* по формуле $cond(A) \approx \|A\| \cdot \left\|A^{-1}\right\|$.

$$\text{Матрица A} = \begin{bmatrix} 611 & 196 & -192 & 407 \\ 196 & 899 & 113 & -192 \\ -192 & 113 & 899 & 196 \\ 407 & -192 & 196 & 611 \end{bmatrix}$$

2.2 Код на Python

```
# -*- coding: utf-8 -*-
"""cond.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/19EXKt2o34TMTK8rp7Okt9ZaBuEGISvwX
"""

import numpy as np

A = np.array([[611, 196, -192, 407],
              [196, 899, 113, -192],
              [-192, 113, 899, 196],
              [407, -192, 196, 611]], dtype=float)

b = np.full((4,4), 0, dtype=float)
m = -np.Inf

for i in range(4):
  b[i, i] = 1
  x_i = np.linalg.solve(A, b[i])
  nor_i = np.linalg.norm(x_i)/np.linalg.norm(b[i])
  m = max(nor_i, m)
```

```
print(A)
norm_A = np.linalg.norm(A)
print(f'Экспериментальное число обусловленности матрицы:                {m*norm_A}')
print(f'Число обусловленности матрицы, полученное встроенной функцией: {np.linalg.cond
```

2.3 Результат работы программы

```
[[ 611.  196. -192.  407.]
 [ 196.  899.  113. -192.]
 [-192.  113.  899.  196.]
 [ 407. -192.  196.  611.]]
Экспериментальное число обусловленности матрицы:               109.93530906959117
Число обусловленности матрицы, полученное встроенной функцией: 102.00000000000013
```

Число обсуловленности, полученное экспериментально, близко к числу, полученному с помощью встроенной функции. Погрешность накапливается по ходу вычислений, поэтому в результатах есть различие.

3 Метод Гаусса - схема полного выбора(3.8.5)

3.1 Формулировка задачи

Задача 3.8.* Дана система уравнений $Az(x)=b(x)$ порядка n. Построить график функции $y(x)=\sum_{i=1}^{n} z_i(x)$ на отрезке $[a, b]$; здесь $z(x)=(z_1(x), z_2(x), \ldots, z_n(x))^T$ - решение системы. Для решения системы уравнений использовать метод Гаусса (схема полного выбора).

Элементы матрицы A вычисляются по формулам:

$$A_{ij}=\begin{cases} q_M^{i+j}+0.1\cdot(j-i), & i\neq j, \\ (q_M-1)^{i+j}, & i=j, \end{cases} \quad \text{где } q_M=1.001-2\cdot M\cdot 10^{-3},\ i,j=1,\ldots n.$$

Элементы вектора b задаются в индивидуальном варианте. Во всех вариантах отрезок $[a, b]$=[-5, 5].

$$M=5, n=100, b_i=|x-\frac{n}{10}|*i*\sin(x)$$

3.2 Код на Python

```
# -*- coding: utf-8 -*-
"""gauss.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/19EXKt2o34TMTK8rp70kt9ZaBuEGISvwX
"""

import numpy as np
```

```python
import math
import matplotlib.pyplot as plt

M = 5
n = 100
q = 1.001 - 2*M*10**(-3)
def search_b(x):
  b = np.zeros(n, dtype=float)
  for i in range(n):
    b[i] = abs(x-(n/10))*i*math.sin(x)
  return b

A = np.zeros((n,n), dtype=float)
for i in range(n):
  for j in range(n):
    if (i==j):
      A[i][j] = (q-1)**(i+j)
    else:
      A[i][j] = (q)**(i+j) + 0.1*(j-i)

def gauss_method(A, b):
    A = A.copy()
    b = b.copy()
    n = A.shape[0]
    b_tmp = []
    for i in range(n):
        max_val = abs(A[i, 0])
        max_i = i
        max_j = 0
        for j in range(i, n):
            for k in range(0, n):
                if abs(A[j, k])> max_val:
                    max_val = abs(A[j, k])
                    max_i = j
                    max_j = k
        b_tmp.append(max_j)
        if max_i!=i:
            tmp = A[i].copy()
            A[i] = A[max_i]
            A[max_i] = tmp
            tmp = b[i]
            b[i] = b[max_i]
            b[max_i] = tmp
        for k in range(i + 1, n):
            b[k] -= b[i] * A[k,max_j]/A[i,max_j]
            A[k] -= A[i] * A[k,max_j]/A[i,max_j]
    result = b.copy()
    for i in range(n-1, -1, -1):
        j = b_tmp[i]
        b[i]/=A[i,j]
```

```
        result[j] = b[i]
        for k in range(0, i):
            b[k] -= b[i] * A[k,j]
    return result

num = 50
point = np.linspace(-5.0, 5.0, num)
y = []
orig = []
for x in point:
  y.append(np.sum(gauss_method(A, search_b(x))))
  orig.append(np.sum(np.linalg.solve(A, search_b(x))))


plt.figure(figsize=(6, 5))
plt.plot(point, y)
plt.title("Реализованный метод Гаусса - схема полного выбора")
plt.xlabel('x')
plt.ylabel('y(x)')
plt.show()
plt.savefig('graphic.png', dpi=300)

plt.figure(figsize=(6, 5))
plt.plot(point, orig)
plt.title("Встроенный метод Гаусса")
plt.xlabel('x')
plt.ylabel('orig(x)')
plt.show()
plt.savefig('orig_graphic.png', dpi=300)
```

3.3 Результат работы программы

Было взято 50 точек x в интервале от [-5, 5], равномерно распределённых. Решение реализованное и решение из библиотеки Python совпадаen - это видно из графиков.

3.4 График реализованного метода Гаусса $y(x)$ и встроенного метода Гаусса $orig(x)$