МГТУ им. Н. Э. Баумана
Факультет ФН «Фундаментальные Науки»
Кафедра ФН-12 «Математическое моделирование»

Численные методы решения задач теории управления

Домашнее задание №2

«Метод Хаусхолдера»

Вариант №4

Группа: ФН12-61Б

Студент: Дорохов М. А.

Преподаватель: Тверская Е. С.

Москва, 2024

Постановка задачи

1. Реализовать указанным методом решение СЛАУ.
2. Решить две заданные СЛАУ указанными методами, вычислить нормы невязок полученных приближенных решений, их абсолютные и относительные погрешности (использовать $\|\cdot\|_1$ и $\|\cdot\|_\infty$.
3. С использованием реализованных методов найти A_1^{-1} и A_2^{-1}. Проверить выполнение равенства $A_i^{-1}A_i = E$.
4. Для каждой системы оценить порядок числа обусловленности её матрицы и сделать вывод о его влиянии на точность полученного приближенного решения и отвечающую ему невязку.
5. Сравнить реализованный метод с решениями, полученными с помощью других прямых методов (например, Гаусса и Гивенса).

Исходные данные

A - матрица системы, b - правый столбец

Тест 1

$$A = \begin{pmatrix} -62.8000 & -6.9700 & 7.7300 & 0.0000 \\ 0.0000 & 49.4000 & -3.5600 & -4.5700 \\ -7.1100 & -5.9700 & 97.6000 & -7.5800 \\ 6.8100 & -1.2900 & -7.9500 & 42.2000 \end{pmatrix}, \quad b = \begin{pmatrix} -221.3100 \\ -523.8800 \\ 1121.6300 \\ 294.6700 \end{pmatrix}.$$

Тест 2

$$A = \begin{pmatrix} 0.0500 & -10.4550 & -5.3550 & 1.7850 \\ 0.0000 & 12.6410 & 6.3210 & -2.1070 \\ 0.3600 & 15.4650 & 7.0850 & -2.3750 \\ 1.0800 & 120.4410 & 58.2810 & -19.4670 \end{pmatrix}, \quad b = \begin{pmatrix} 26.2450 \\ -29.4990 \\ -24.5950 \\ -246.5790 \end{pmatrix}.$$

Ход работы

Описание метода Хаусхолдера

Метод Хаусхолдера - метод QR-разложения матриц, который в качестве ортогональных преобразований использует преобразования симметрии. Рассмторим матрицу $A \in \mathbb{R}^{n\times n}$:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}.$$

Найдем матрицу H_1 такую, что

$$H_1 A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}.$$

Теперь найдём H_2, чтобы занулить поддиагональные элементы второго столбца и так далее до H_{n-1}. Тогда получим верхнетругольную матрицу:

$$H_{n-1}\dots H_2H_1A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{nn} \end{pmatrix} = R.$$

Матрицы H_i называются матрицами Хаусхолдера. По определению,

$$Hx = x - 2(x,\omega)\omega = (E - 2\omega\omega^T)x, \quad \omega, x \in \mathbb{R}^n, \quad \|\omega\| = 1.$$

Можно также записать в виде:

$$Hx = (E - 2\omega\omega^T)x = \left(E - \frac{2\upsilon\upsilon^T}{\upsilon^T\upsilon}\right)x, \quad \omega = \frac{\upsilon}{\|\upsilon\|}.$$

Таким образом, мы преобразовываем СЛАУ:

$$Ax = b \;\leftrightarrow\; QRx = b \;\leftrightarrow\; Rx = Q^Tb,$$

где R - верхнетреугольная матрица, после преобразования x находится обратной подстановкой.

Результаты расчётов

Номер теста	1	2
Решение	$x = (6,\ -9,\ 12,\ 8)^T$	$x = (20,\ -6,\ 1,\ 2.00000001)^T$
Невзяка	$x = \begin{pmatrix} 2.84217094e-14 \\ 0.00000000e+00 \\ -4.54747351e-13 \\ -5.68434189e-14 \end{pmatrix}$	$x = \begin{pmatrix} 0.00000000e+00 \\ -3.55271368e-15 \\ -3.55271368e-15 \\ 0.00000000e+00 \end{pmatrix}$
$\|Ax-b\|_1$	5.400124791776761e-13	7.105427357601002e-15
$\|Ax-b\|_\infty$	4.547473508864641e-13	3.552713678800501e-15
Число обусловленности $\|\cdot\|_1$	3.3379041290652394	401918895.2910938
Число обусловленности $\|\cdot\|_\infty$	3.49769768291856	552318536.5954505
Абсолютная ошибка	$x = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1.77635684e-15 \end{pmatrix}$	$x = \begin{pmatrix} 7.99300182e-10 \\ -1.57293290e-10 \\ 2.41149856e-09 \\ 6.29081365e-09 \end{pmatrix}$
Относительная ошибка	$x = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}$	$x = \begin{pmatrix} 0.11778219 \\ -0.02317821 \\ 0.35535032 \\ 0.92699314 \end{pmatrix}$

Рассмотрим подробнее тестовый пример 2 (с плохо обусловленной матрицей). Внесем нарочно в вектор b погрешность 0.01, изменив каждую компоненту вектора на эту величину:

$$b = (26.2450 + 0.01,\ -29.4990 - 0.01,\ -24.5950 + 0.01,\ -246.5790 - 0.01)^T.$$

Решая изменённую систему методом Хаусхолдера, получили решение:

$$x = (683.19999749,\ -129.46666617,\ 1995.6666591,\ 5224.26664694)^T.$$

Как видно, решение системы с плохо обусловленной матрицей довольно затруднительно, поскольку малейшая погрешность входных данных очень сильно влияет на погрешность решения.

Код программы

```
import numpy as np
import math

#Вариант 4

def LoadTest(num):
  if num == 1: #хорошо обусловленная
    A = np.array([
        [-62.8000, -6.9700, 7.7300, 0.0000],
        [0.0000, 49.4000, -3.5600, -4.5700],
        [-7.1100, -5.9700, 97.6000, -7.5800],
        [6.8100, -1.2900, -7.9500, 42.2000]
    ], dtype = "float64")
    b = np.array([-221.3100, -523.8800, 1121.6300, 294.6700], dtype = "float64")
    sol = np.array([6, -9, 12, 8], dtype = "float64")
  if num == 2: #плохо обусловленная
    A = np.array([
        [0.0500, -10.4550, -5.3550, 1.7850],
        [0.0000, 12.6410, 6.3210, -2.1070],
        [0.3600, 15.4650, 7.0850, -2.3750],
        [1.0800, 120.4410, 58.2810, -19.4670]
    ], dtype = "float64")
    b = np.array([26.2450+0.01, -29.4990-0.01, -24.5950+0.01, -246.5790-0.01], dtype = "
    sol = np.array([20, 1, -6, 2], dtype = "float64")
  return A, b, sol

def house(x):
  n=len(x); mu=np.linalg.norm(x); v=np.copy(x)
  if mu:
    beta=x[0]+(1.0 if x[0] >= 0 else -1.0)*mu
    v[1:]/=beta
  v[0]=1.0
  return v

def row_house(A,v):
  beta = -2.0 / np.dot(v,v)
  w = beta * np.matmul(A.T,v)
  A+=np.outer(v,w)

def col_house(A,v):
  beta = -2.0 / np.dot(v,v)
  w = beta * np.matmul(A,v)
  A+=np.outer(w,v)

def house_QR(A):
  m,n=A.shape
  for j in range(n):
    v=house(A[j:,j])
    row_house(A[j:,j:],v)
    if j<m-1:
      A[j+1:,j]=v[1:]
```

```
def rev_subs(U,b):
  n=len(b)
  b[n-1]/=U[n-1,n-1]
  for i in range(n-2,-1,-1):
    b[i]=(b[i]-np.dot(U[i,i+1:],b[i+1:]))/U[i,i]

def row_house_vec(x,v):
  beta = -2.0 * np.dot(v,x) / np.dot(v,v)
  x += beta * v

def solve_house(A,b):
  house_QR(A)
  _,n=A.shape
  v=np.zeros(n)
  for j in range(n):
    v[j]=1.0; v[j+1:]=A[j+1:,j]
    row_house_vec(b[j:],v[j:])
  rev_subs(A,b)

A, b, sol = LoadTest(1)

solve_house(A, b)
print("Решение: ", b)
print("Абсолютная ошибка: ", b - sol)
print("Относительная ошибка: ", (b - sol) / np.linalg.norm(b-sol))
A, f, sol = LoadTest(1)

print("\nНевязка: ", A@b - f)
print("1-норма: ", np.linalg.norm(A@b - f, ord = 1))
print("inf-норма: ", np.linalg.norm(A@b - f, ord = math.inf))
print("\nЧисло обусловленности системы: ")
print("1-норма: ", np.linalg.cond(A, p = 1))
print("inf-норма: ", np.linalg.cond(A, p = math.inf))

A, b, sol = LoadTest(2)

solve_house(A, b)
print("\nРешение: ", b)
print("Абсолютная ошибка: ", b - sol)
print("Относительная ошибка: ", (b - sol) / np.linalg.norm(b-sol))

A, f, sol = LoadTest(2)

print("\nНевязка: ", A@b - f)
print("1-норма: ", np.linalg.norm(A@b - f, ord = 1))
print("inf-норма: ", np.linalg.norm(A@b - f, ord = math.inf))
print("\nЧисло обусловленности системы: ")
print("1-норма: ", np.linalg.cond(A, p = 1))
print("inf-норма: ", np.linalg.cond(A, p = math.inf))

A, _, _ = LoadTest(2)
```

```
x = []
def house_inverse(A):
  _,n = A.shape
  temp = A
  e = np.eye(n)
  temp_e = e
  for i in range(n):
    A = np.copy(temp)
    e = np.copy(temp_e)
    solve_house(A, e[i])
    x.append(e[i])
  return np.transpose(np.array(x))

A_inv = house_inverse(A)
A, _, _ = LoadTest(2)

display(A_inv.dot(A))
```