Московский авиационный институт
(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Информационные технологии и прикладная математика»

Курсовая работа по курсу «Численные методы»

Тема: «Нахождение собственных значений и собственных векторов несимметричных разреженных матриц большой размерности. Метод Арнольди.»

Студент: Суров В.О.
Группа: М8О-402Б-20
Преподователь: Пивоваров Д.Е.

Москва, 2024

1. Условие

Написать программу для нахождение собственных значений и собственных векторов симметричных разреженных матриц большой размерности методом простых итераций Арнольди.

2. Описание алгоритма

Метод Арнольди относится к алгоритмам линейной алгебры, которые позволяют получить частичное решение после небольшого количества итераций, в отличие от так называемых прямых методов, которые должны полностью завершиться для получения каких-либо удовлетворительных результатов.

Итуитивным методом нахождения наибольшего (по модулю) собственного значения данной матрицы A размерами m x m является степенной метод: начать с произвольного начального вектора b, вычислить $Ab, A^2b, A^3b, \dots$, нормируя результат после каждого вычисления. Эта последовательность сходится к собственному вектору соответствующего собственного значения λ с максимальным модулем. Vного вычислений тратится впустую, т.к. в итоге используется лишь конечный результат $A^{n-1}b$. Тогда составим матрица Крылова: $K_n = [b\ Ab\ A^2b \dots A^{n-1}b]$. Столбцы этой матрицы в общем случае не являются ортогональными, но мы можем получить из них ортогональный базис с помощью ортогонализации Грама-Шмидта. Полученное множество векторов будет являться ортогональным базисом подпространства Крылова.

Итерация Арнольди использует стабилизированный процесс Грама-Шмидта для получения последовательности ортонормированных векторов $q_1, q_2, q_3, \dots$, называемых векторами Арнольди таких, что для каждого n векторы $q_1, q_2, q_3, \dots q_n$ 1,…, являются базисом подпространства Крылова.

Алгоритм выглядит следующим образом:

Начинаем с произвольного вектора q1 с нормой 1.

Повторить для k = 2, 3, ...

```
qk := A qk−1
for j from 1 to k − 1
    hj,k−1 := qj* qk
    qk := qk − hj,k−1 qj
hk,k−1 := ‖qk‖
qk := qk / hk,k−1
```

Цикл по j проецирует компоненту q_k на $q_1, \dots, q_{k-1}$. Это обеспечивает ортогональность всех построенных векторов. Алгоритм останавливается, когда q_k является нулевым вектором. Это происходит, когда минимальный многочлен матрицы A будет степени k. Каждый шаг цикла по k производит одно умножение матрицы на вектор и около $4mk$ операций с дробными числами.

3. Результат

```
Введите размер матрицы: 6
Введите элементы матрицы построчно:
1 3 1 0 2 3
7 4 0 0 1 2
5 3 0 4 6 1
4 5 2 8 1 9
1 2 2 1 6 6
0 0 7 4 4 2

Матрица:
[[1. 3. 1. 0. 2. 3.]
 [7. 4. 0. 0. 1. 2.]
 [5. 3. 0. 4. 6. 1.]
 [4. 5. 2. 8. 1. 9.]
 [1. 2. 2. 1. 6. 6.]
 [0. 0. 7. 4. 4. 2.]]

Все собственные значения:
[17.90679078 -2.95208073 -2.99010248 -1.88676162  5.46107702  5.46107703]

Приближенные собственные векторы:
[[ 1.00000000e+00  5.08181722e-18  4.31066232e-18 -4.15279144e-17
  -1.43366330e-17 -1.86762487e-17]
 [-3.66124987e-18  1.00000000e+00  1.69149126e-17 -2.70735533e-17
   8.91971387e-17  1.70243734e-17]
 [-3.15680429e-17 -5.06214738e-17  1.00000000e+00 -3.31199855e-17
   1.09827581e-17 -3.78544658e-17]
 [-8.93506488e-18 -5.66952451e-17  6.85339924e-17  1.00000000e+00
 -9.00661223e-17 -8.39021869e-17]
[-3.21955322e-18  1.08556090e-16  4.47509513e-17 -5.41178818e-17
  1.00000000e+00  3.49668407e-17]
[-3.89815870e-17  2.85666459e-17 -4.08607746e-17 -3.84070003e-17
  1.01088866e-16  1.00000000e+00]]
```

4. Вывод

В ходе исследования метода итераций Арнольди, основанного на использовании подпространств Крылова, для анализа собственных значений и векторов разреженной несимметричной матрицы. Я приобрел глубокое понимание этого эффективного численного метода. В процессе работы над курсовой работой, я освоил ключевые принципы формирования подпространств Крылова, их взаимодействия с матрицей, а также специфику работы с разреженными несимметричными матрицами.

5. Код

```
6. import numpy as np


   def QR(matrix, tol=1e-12, max_iter=1000):
       n = matrix.shape[0]
       eigenvalues = np.zeros(n, dtype=complex)
       for i in range(max_iter):
           shift = matrix[-1, -1]
           Q, R = custom_qr(matrix - shift * np.eye(n))
           matrix = R @ Q + shift * np.eye(n)
           if frobenius_norm_upper_triangle(matrix) < tol:
               break
           eigenvalues = np.diag(matrix)
       return eigenvalues


   def QR_dop(matrix):
       n = matrix.shape[0]
       Q = np.eye(n)
       R = matrix.copy()
       for i in range(n - 1):
           x = R[i:, i].copy()
           norm_x = np.linalg.norm(x)
           v = x - norm_x * np.eye(len(x))[:, 0]
           v = v / np.linalg.norm(v)
           H = np.eye(len(x)) - 2 * np.outer(v, v)
           R[i:, i:] = H @ R[i:, i:]
           Q[:, i:] = Q[:, i:] @ H.T
       return Q, R


   def fFrobenius(matrix):
       return np.linalg.norm(np.triu(matrix, k=1), ord='fro')

   def Arnoldi(matrix, k):
       n = len(matrix)
       q = np.random.rand(n)
       q = q / np.linalg.norm(q)
       Q = np.zeros((n, k + 1))
       H = np.zeros((k + 1, k))
       Q[:, 0] = q

       for j in range(k):
           v = np.dot(matrix, Q[:, j])
           for i in range(j + 1):
               H[i, j] = np.dot(Q[:, i], v)
               v = v - H[i, j] * Q[:, i]
           H[j + 1, j] = np.linalg.norm(v)
           Q[:, j + 1] = v / H[j + 1, j]
       eigenvalues_H = qr_alg(H[:k, :k])
       eigenvectors_A = np.dot(Q[:, :k], np.linalg.inv(Q[:, :k].T @ Q[:, :k]) @
   Q[:, :k].T)

       return eigenvalues_H, eigenvectors_A


   n = int(input("Введите размер матрицы: "))
   matrix = np.zeros((n, n))
   print("Введите элементы матрицы построчно:")

   for i in range(n):
       matrix[i, :] = list(map(float, input().split()))
```

```
k = n
eigenvalues, eigenvectors = arnoldi_method(matrix, k)

print("\nМатрица:")
print(matrix)
print("\nВсе собственные значения:")
print(eigenvalues)
print("\nПриближенные собственные векторы:")
print(eigenvectors)
```