МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Курсовая работа
по курсу «Численные методы»
на тему «Нахождение собственных значений и собственных векторов несимметричных разреженных матриц большой размерности. Метод Арнольди.»

Выполнил: Велесов Д.И.
Группа: 8О-408Б-20

Москва, 2023

Оглавление

Условие

Написать программу, которая будет считать собственные значения разреженной несимметричной матрицы, а также её собственные вектора, используя Метод итераций Арнольди.

Метод решения

В численной линейной алгебре **итерация Арнольди** является алгоритмом вычисления собственных значений. Арнольди находит приближение собственных значений и собственных векторов матриц общего вида с помощью построения ортонормированного базиса подпространства Крылова.

Матрица Крылова:

$$K_n = [b \quad Ab \quad A^2b \quad \cdots \quad A^{n-1}b].$$

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*. Можно ожидать, что вектора этого базиса будут хорошим приближением к векторам, соответствующим наибольшим по модулю собственным значениям.

Итерация Арнольди использует стабилизированный процесс Грама-Шмидта для получения последовательности ортонормированных векторов ***q1,q2,q3.***., называемых *векторами Арнольди*, таких, что для каждого ***n*** векторы ***q1…qn*** являются базисом подпространства Крылова. Алгоритм выглядит следующим образом:

Начинаем с произвольного вектора q_1 с нормой 1.
Повторить для $k = 2, 3, \ldots$
  $q_k := A\, q_{k-1}$
  for j **from** 1 **to** $k - 1$
    $h_{j,k-1} := q_j^* q_k$
    $q_k := q_k - h_{j,k-1}\, q_j$
  $h_{k,k-1} := \|q_k\|$
  $q_k := q_k / h_{k,k-1}$

Данный цикл j проецирует компоненту ***qk*** на ***q1….qk-1***.
Это обеспечивает ортогональность всех генерируемых векторов.
Алгоритм останавливается, когда ***qk*** - нулевой вектор. Это происходит, когда минимальный многочлен A имеет степень k. В большинстве приложений итерации Арнольди , алгоритм сходится в этой точке. Каждый шаг k-цикла занимает одно матрично-векторное произведение и примерно ***4mk*** операций с плавающей точкой.
Идея итерации Арнольди как алгоритма вычисления собственных значений заключается в вычислении собственных значений в подпространстве Крылова. Поскольку конечная матрица H - матрица Гессенберга малого размера, ее собственные значения могут быть вычислены эффективно, например, с помощью QR-алгоритма.

Результаты

Программа написана на Python, использует функцию итерации Арнольди для получения верхней треугольной матрицы, из которой мы получаем нужный результат с помощью QR разложения. Программе необходимо подать на вход размер матрицы, а также её элементы. После ввода всех её элементов программа выведем результат в виде собственных значений, а также собственных векторов. :

```
Введите размер матрицы: 3
Введите элементы матрицы построчно:
3 0 0
0 0 6
0 4 0

Матрица:
[[3. 0. 0.]
 [0. 0. 6.]
 [0. 4. 0.]]

Все собственные значения:
[ 4.89897949  3.         -4.89897949]

Приближенные собственные векторы:
[[ 1.00000000e+00 -5.55111512e-17 -2.77555756e-17]
 [-8.32667268e-17  1.00000000e+00 -5.55111512e-17]
 [-7.63278329e-17 -5.55111512e-17  1.00000000e+00]]
```

```
Введите размер матрицы: 4
Введите элементы матрицы построчно:
4 -1 0 0
1 4 -1 0
0 1 4 -1
0 0 1 4

Матрица:
[[ 4. -1.  0.  0.]
 [ 1.  4. -1.  0.]
 [ 0.  1.  4. -1.]
 [ 0.  0.  1.  4.]]

Все собственные значения:
[4. 4. 4. 4.]

Приближенные собственные векторы:
[[ 1.00000000e+00  0.00000000e+00  1.11022302e-16  5.55111512e-17]
 [ 0.00000000e+00  1.00000000e+00 -8.32667268e-17 -5.55111512e-17]
 [-2.77555756e-17  8.32667268e-17  1.00000000e+00  2.77555756e-17]
 [ 0.00000000e+00  5.55111512e-17 -2.77555756e-17  1.00000000e+00]]
```

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

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

Все собственные значения:
[ 7.48331477  6.97180683  5.60555128 -7.48331478 -5.36883522 -1.60555128
 -1.60297161]

Приближенные собственные векторы:
[[ 1.00000000e+00 -2.77555756e-17 -1.38777878e-17  5.55111512e-17
   5.55111512e-17  8.07730619e-18  2.17653586e-17]
 [ 8.32667268e-17  1.00000000e+00  9.71445147e-17 -5.55111512e-17
  -2.77555756e-17  1.08420217e-18 -5.19468366e-17]
 [ 2.77555756e-17  2.77555756e-17  1.00000000e+00  8.32667268e-17
   2.77555756e-17 -9.64939934e-18 -6.19757067e-17]
 [ 5.55111512e-17 -5.55111512e-17  8.32667268e-17  1.00000000e+00
   2.77555756e-17  7.07712968e-17  7.80490039e-17]
 [ 5.55111512e-17  0.00000000e+00  2.77555756e-17  0.00000000e+00
   1.00000000e+00  5.48606299e-17 -8.52453958e-18]
 [ 3.12521276e-17 -2.72947897e-17 -3.12927852e-17  8.87148428e-17
   9.19132392e-17  1.00000000e+00 -5.67320169e-18]
 [ 1.01020537e-16  2.53838834e-17  3.13063377e-17  2.65358482e-17
   3.14960731e-17  4.14496102e-17  1.00000000e+00]]
```

Выводы

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.

Приложение:

```
import numpy as np



def qr_alg(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)

        # Check for convergence
        if frobenius_norm_upper_triangle(matrix) < tol:
            break

        eigenvalues = np.diag(matrix)

    return eigenvalues



def custom_qr(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)

        # Construct the Householder matrix
        H = np.eye(len(x)) - 2 * np.outer(v, v)

        # Update R and Q
        R[i:, i:] = H @ R[i:, i:]
        Q[:, i:] = Q[:, i:] @ H.T

    return Q, R



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

#Метод Арнольди
def arnoldi_method(matrix, k):
    """Computes a basis of the (k + 1)-Krylov subspace of matrix

    Arguments
      matrix: n × n array
      k: dimension of Krylov subspace

    Returns
      Q: n x (k + 1) array, the columns are an orthonormal basis of
the
        Krylov subspace.
      h: (n + 1) x n array, A on basis Q. It is upper Hessenberg.
    """
    n = len(matrix)

    # Инициализация начального вектора
    q = np.random.rand(n)
    q = q / np.linalg.norm(q)
```

```
    # Массивы для хранения ортогональных векторов и верхнетреугольной матрицы H
    Q = np.zeros((n, k + 1))
    H = np.zeros((k + 1, k))

    # Первый столбец матрицы Q - начальный вектор
    Q[:, 0] = q

    for j in range(k):
        # Вычисление нового вектора
        v = np.dot(matrix, Q[:, j])

        # Процесс ортогонализации по методу Грама-Шмидта
        for i in range(j + 1):
            H[i, j] = np.dot(Q[:, i], v)
            v = v - H[i, j] * Q[:, i]

        # Нормализация нового вектора
        H[j + 1, j] = np.linalg.norm(v)
        Q[:, j + 1] = v / H[j + 1, j]

    # Compute eigenvalues without using np.linalg.eigvals()
    eigenvalues_H = qr_alg(H[:k, :k])

    # Вычисление приближенных собственных векторов матрицы A
    eigenvectors_A = np.dot(Q[:, :k], np.linalg.inv(Q[:, :k].T @ Q[:, :k]) @ Q[:, :k].T)

    return eigenvalues_H, eigenvectors_A



# Получаем матрицу от пользователя
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

# Вызываем функцию метода Арнольди
eigenvalues, eigenvectors = arnoldi_method(matrix, k)

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