### 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: Эсмедляев Е.Р.

Группа: М8О-409Б-20

Проверил: Пивоваров Д.Е.

Дата:

Оценка:

### Москва, 2024

Оглавление

[Условие 3](#_bookmark0)

[Метод решения 3](#_bookmark1)

[Результаты 4](#_bookmark2)

[Выводы 7](#_bookmark3)

[Приложение: 8](#_bookmark4)

# 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*:



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, ...

*qk* := *A qk*−1

**for** *j* **from** 1 **to** *k* − 1

*hj*,*k*−1 := *q* \* *qk*

*j*

*qk* := *qk* − h*j*,*k*−1 *qj hk*,*k*−1 := ‖*qk*‖

*qk* := *qk* / *hk*,*k*−1

Данный цикл j проецирует компоненту ***qk*** на ***q1….qk-1***.

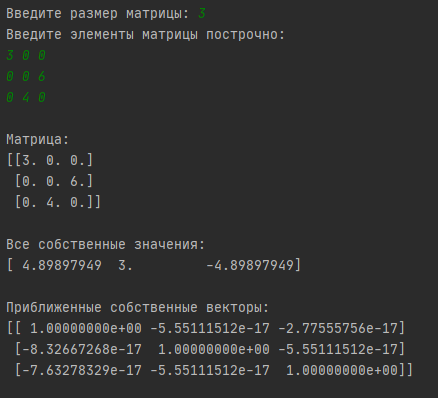
Это обеспечивает ортогональность всех генерируемых векторов.

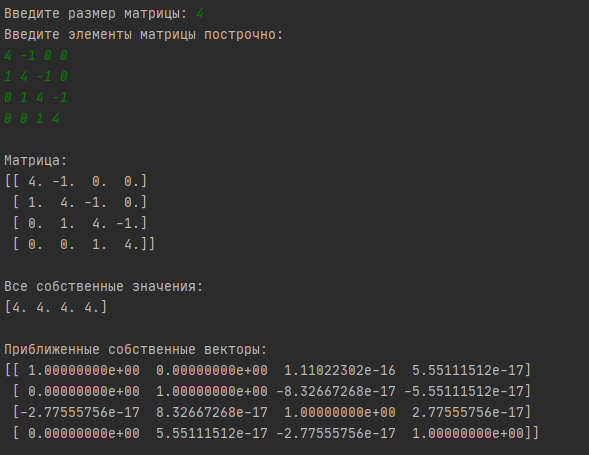
Алгоритм останавливается, когда ***qk*** - нулевой вектор. Это происходит, когда минимальный многочлен A имеет степень k. В большинстве приложений итерации Арнольди , алгоритм сходится в этой точке. Каждый шаг k-цикла занимает одно матрично-векторное произведение и примерно ***4mk*** операций с плавающей точкой.

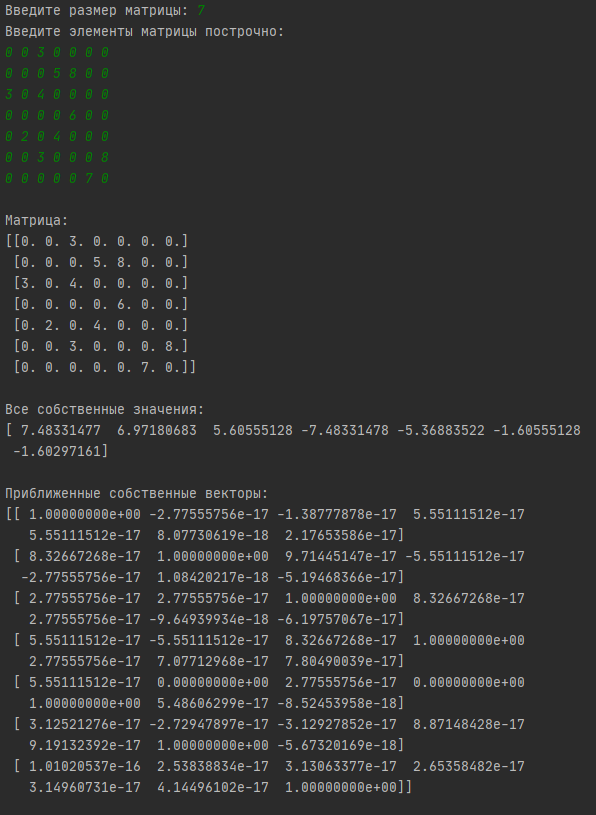
Идея итерации Арнольди как алгоритма вычисления собственных значений заключается в вычислении собственных значений в подпространстве Крылова. Поскольку конечная матрица H - матрица Гессенберга малого размера, ее собственные значения могут быть вычислены эффективно, например, с помощью QR-алгоритма.

# 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. :







# 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*

*the*

*Returns*

*Q: n x (k + 1) array, the columns are an orthonormal basis of*

*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)*