БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
Факультет прикладной математики и информатики

**Лабораторная работа №4**  
по теме  
«Итерационные методы решения проблемы собственных значений»

Выполнила   
Молочко Екатерина  
2 курс 7 группа

Преподаватель  
Горбачева Юлия Николаевна

Минск   
2021

## Постановка задачи:

1. Написать и отладить программу нахождения наибольшего по модулю собственного значения и соответствующему ему собственного вектора степенным методом
2. Привести теоретические сведения о степенном методе
3. Выполнить вычислительный эксперимент с отслеживанием итерации, на которой достигнута требуемая точность
4. Написать и отладить программу нахождения всех собственных значений и соответствующих им собственных векторов итерационным методом вращений (Якоби).
5. Привести теоретические сведения об итерационном методе Якоби
6. Выполнить вычислительный эксперимент итерационным методом вращений (Якоби) с отслеживанием итерации, на которой достигнута требуемая точность

## Листинг программы

**import** numpy **as** np

**import** sympy **as** sp

**from** math **import** cos**,** pi**,** sin**,** sqrt

EPS **=** 10 **\*\*** **(-**7**)**

N **=** **int(input(**'Degree of a matrix: '**))**

# Генерация диагонализируемой матрицы

A **=** np**.**full**((**N**,** N**),** 0**)**

**for** i **in** **range(len(**A**)):**

A**[**i**][**i**]** **=** np**.**random**.**randint**(-**100**,** 100**)**

**for** j **in** **range(len(**A**[**i**])):**

**if** **(**i **>** j**):**

A**[**i**][**j**]** **=** np**.**random**.**randint**(-**100**,** 100**)**

A**[**j**][**i**]** **=** A**[**i**][**j**]**

**print(**A**)**

# Генерация начального вектора

Y\_ **=** np**.**array**([**0**]** **\*** N**)**

Y\_**[**0**]** **=** 1

**print(**"Initial vector y: "**,** Y\_**)**

# Нормированный начальный вектор

U\_ **=** Y\_**/**np**.**linalg**.**norm**(**Y\_**)**

**print(**"Initial norm vector"**,** U\_**)**

**def** power\_iteration\_method**(**A\_**,** Y\_**,** U\_**):**

K **=** 0

Y\_ **=** np**.**matmul**(**A\_**,** U\_**)**

lam **=** np**.**dot**(**Y\_**,** U\_**)**

U\_ **=** Y\_ **/** np**.**linalg**.**norm**(**Y\_**)**

crit **=** np**.**matmul**(**A\_**,** U\_**)** **-** lam **\*** U\_

**while(**np**.**linalg**.**norm**(**crit**)** **>** EPS**):**

K **=** K**+**1

Y\_ **=** np**.**matmul**(**A\_**,** U\_**)**

lam **=** np**.**dot**(**Y\_**,** U\_**)**

U\_ **=** Y\_ **/** np**.**linalg**.**norm**(**Y\_**)**

crit **=** np**.**matmul**(**A\_**,** U\_**)** **-** lam **\*** U\_

**print(**"Maximal eigenvalue:"**,** lam**)**

**print(**"Result vector:"**,** U\_**)**

**print(**"Number of itertions:"**,** K**)**

**print(**"Test:"**,** crit**)**

**print(**"Test norm:"**,** np**.**linalg**.**norm**(**crit**))**

**def** find\_max**(**A\_**):**

mat **=** np**.abs(**np**.**copy**(**A\_**))**

**for** i **in** **range(len(**mat**)):**

mat**[**i**][**i**]** **=** 0

**for** j **in** **range(len(**mat**[**i**])):**

**if(**i **>** j**):**

mat**[**i**][**j**]** **=** 0

# print(mat)

indecies **=** np**.**unravel\_index**(**np**.**argmax**(**mat**),** mat**.**shape**)**

# print("Indecies", indecies)

**return** indecies

**def** sum\_of\_non\_diagonal**(**A**):**

**sum** **=** 0

**for** i **in** **range(**A**.**shape**[**0**]):**

**for** j **in** **range(**i **+** 1**,** A**.**shape**[**0**]):**

**sum** **+=** A**[**i**,** j**]** **\*\*** 2

**return** 2 **\*** **sum**

**def** calculate\_trig\_functions**(**matrix**,** row**,** column**):**

**if** matrix**[**row**,** row**]** **==** matrix**[**column**,** column**]:**

**return** sp**.**cos**(**pi**/**4**),** sp**.**sin**(**pi**/**4**)**

mu **=** 2**\***matrix**[**row**,** column**]** **/** **(**matrix**[**row**,** row**]** **-** matrix**[**column**,** column**])**

tmp **=** 1**/**sqrt**(**1**+**mu**\*\***2**)**

mult **=** **-**1

**if** mu **>** 0**:**

mult **\*=** **-**1

**return** sqrt**(**0.5**\*(**1**+**tmp**)),** mult **\*** sqrt**(**0.5**\*(**1**-**tmp**))**

**def** jacobi\_method**(**A\_**):**

number\_of\_iterations **=** 0

A\_k **=** np**.**copy**(**A\_**)**

T **=** np**.**matrix**(**np**.**diag**([**1.0 **for** i **in** **range(**A\_**.**shape**[**0**])])).**astype**(float)**

**while** **(**sum\_of\_non\_diagonal**(**A\_k**)** **>** EPS**):**

number\_of\_iterations **+=** 1

indecies **=** find\_max**(**A\_k**)**

k **=** indecies**[**0**]**

l **=** indecies**[**1**]**

I\_c **=** A\_**[**0**:][**k**]** # i column

I\_r **=** A\_**[**k**][**0**:]** # i row

J\_c **=** A\_**[**0**:][**l**]** # j column

J\_r **=** A\_**[**l**][**0**:]** # j row

cos**,** sin **=** calculate\_trig\_functions**(**A\_k**,** k**,** l**)**

# only columns with indexes k,l will change B = AkTkl

B **=** np**.**copy**(**A\_k**)**

T\_tmp **=** np**.**copy**(**T**)**

**for** i **in** **range(**A\_**.**shape**[**0**]):**

B**[**i**,** k**]** **=** A\_k**[**i**,** k**]** **\*** cos **+** A\_k**[**i**,** l**]** **\*** sin

B**[**i**,** l**]** **=** A\_k**[**i**,** l**]** **\*** cos **-** A\_k**[**i**,** k**]** **\*** sin

T\_tmp**[**i**,** k**]** **=** T**[**i**,** k**]** **\*** cos **+** T**[**i**,** l**]** **\*** sin

T\_tmp**[**i**,** l**]** **=** T**[**i**,** l**]** **\*** cos **-** T**[**i**,** k**]** **\*** sin

T **=** np**.**copy**(**T\_tmp**)**

# C = T^-1klB

# only rows k,l will change

C **=** np**.**copy**(**B**)**

**for** i **in** **range(**A\_**.**shape**[**0**]):**

C**[**k**,** i**]** **=** B**[**k**,** i**]** **\*** cos **+** B**[**l**,** i**]** **\*** sin

C**[**l**,** i**]** **=** B**[**l**,** i**]** **\*** cos **-** B**[**k**,** i**]** **\*** sin

A\_k **=** np**.**copy**(**C**)**

**print(**"Eigenvalues"**,** np**.**diagonal**(**A\_k**))**

**print(**"Number of iterations"**,** number\_of\_iterations**)**

**print(**"Matrix T"**,** T**.**T**)**

R **=** np**.**array**(**np**.**full**(**N**),**0**)**

**for** i **in** **range** **(**A\_**.**shape**[**0**]):**

R**[**i**]** **=** np**.**matmul**(**A\_**,**T**.**T**[**i**][**0**:])** **-** np**.**matmul**(**A\_**)**

**return** np**.**diagonal**(**A\_k**),** T**.**T**,** number\_of\_iterations

power\_iteration\_method**(**A**,** Y\_**,** U\_**)**

jacobi\_method**(**A**)**

jacobi\_result **=** jacobi\_method**(**A**)**

**print(**f"number of iterations: {jacobi\_result**[**2**]**}\n"**)**

**for** i **in** **range(**A**.**shape**[**0**]):**

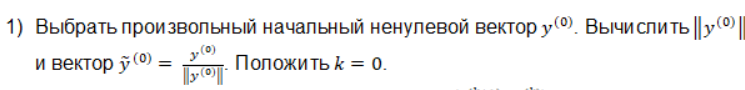
**print(**f"eigenvalue: {jacobi\_result**[**0**][**i**]**}"**)**

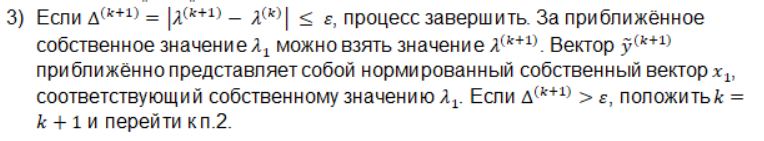
**print(**f"corresponding eigenvector:\n{jacobi\_result**[**1**][**i**]**}"**)**

**print(**f"the residual r\_i: {np**.**subtract**(**np**.**matmul**(**A**,** jacobi\_result**[**1**][**i**]),** jacobi\_result**[**0**][**i**]** **\***jacobi\_result**[**1**][**i**]** **)**}\n"**)**

## Теоретические сведения

## Степенной метод(метод скалярных произведений)

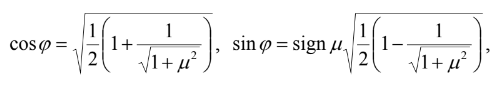




## Итерационный метод вращений Якоби:

Ищем оптимальный элемент матрицы(наибольший по модулю недиагональный элемент)

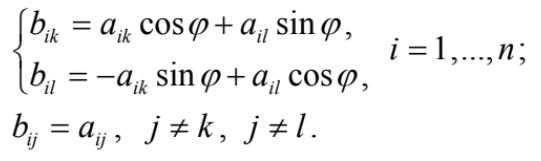
 где k и l – номер строки и столбца на, пересечении которых стоит наш оптимальный элемент.



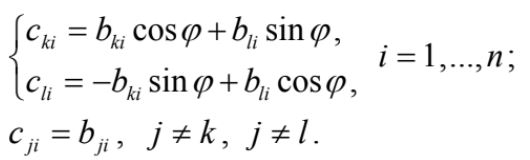
Потом составляем матрицу вращений, у которой на месте kk и ll стоит , на позиции lk стоит , а на kl стоит -.

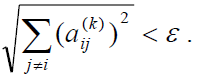
Оставшиеся диагональные элементы равны 1, а все остальные элементы 0.

Образуем матрицу B = A \* , которая отличается от А только k и l столбцами:



Аналогично образуем матрицу С = , которая отличается от B только k и l строками:



Если условие выхода: не выполнилось, то продолжаем действия но уже с матрицей A = C.

Если условие выхода выполнилось, то на диагонали матрицы A мы получим наши собственные значения, а столбцы матрицы T, которая равна произведению всех матриц вращения в том порядке, в котором мы их получали, равны собственным векторам, соответствующим собственным значениям .

## Вычислительный эксперимент

**Начальный вектор:** [ 1 0 0 0 0 0 0 0 0 0 ]

**Номер итерации, на которой достигнута требуемая точность:** 3856

**Приближенное наибольшее по модулю собственное значение :** 325.64790495007264

**Соответствующий ему собственный вектор :**  
[ 0.16611407 0.46677327 -0.20149657 0.6044673 -0.11190232 -0.4007341

0.22581618 0.13240336 0.08895862 -0.31464274 ]

**Вектор :**

[ 4.36788241e-08 -5.63693447e-09 2.63006399e-08 -1.84710416e-08  
 -4.71735078e-08 -3.44731177e-08 -3.23775566e-08 -3.72854174e-08   
-2.46796041e-08 -2.28520634e-08 ]

**Норма**  9.978006617700295e-08

## Вычислительный эксперимент

**Номер итерации, на которой достигнута требуемая точность:** 49

**Приближенные собственные значения соответствующие ему собственные векторы**eigenvalue: -84

corresponding eigenvector:

[ 0.36559722 0.31233819 0.36294637 -0.35235969 -0.05108888 0.16614453

0.12579412 -0.38470579 -0.5268698 -0.20314414]

the residual r\_i: [-8.97125131 -0.07962325 -0.5206283 -4.39895495 -5.24163455 -0.62178711

5.73676644 5.50224964 14.3871956 14.03783383]

eigenvalue: -70

corresponding eigenvector:

[-0.38716563 0.41145903 -0.37234947 -0.24046343 -0.1452244 0.01388779

0.05854655 0.47356454 -0.47386379 0.10400234]

the residual r\_i: [ 1.98627666 1.5114506 3.35928818 5.12511385 -0.87988112 5.60070834

-0.96945607 -2.75353387 -0.88346135 -0.19118863]

eigenvalue: -126

corresponding eigenvector:

[-0.33269876 0.03055487 0.46894383 0.13959723 0.18376643 0.46872489

0.25099326 0.00619869 -0.0198629 0.57626256]

the residual r\_i: [ 6.12441886 -2.36363968 -5.1946366 -5.4307767 -3.90997663

-10.83662795 0.95155145 -6.16560208 -13.84201211 -2.13722842]

eigenvalue: 38

corresponding eigenvector:

[ 0.36329315 0.28758876 -0.44218385 0.45603545 0.44837054 0.40076878

-0.09008541 -0.03723997 -0.10266994 0.01099263]

the residual r\_i: [ -6.15929152 2.60184462 -13.32049231 0.47134709 10.87449328

2.91727374 0.64579427 1.88631825 6.53749638 -4.41917876]

eigenvalue: 226

corresponding eigenvector:

[-0.09549958 -0.44145056 -0.20112133 -0.50353894 0.46588813 0.06829447

-0.39972718 -0.17300787 -0.20680834 0.21863409]

the residual r\_i: [ 0.21746342 -3.82013054 -0.44162013 -4.91081481 6.40146505

7.13078064 -16.80221697 -2.84031246 1.42134465 9.68996419]

eigenvalue: 159

corresponding eigenvector:

[ 0.35000225 -0.24906902 -0.08338718 -0.34832658 -0.27968494 0.60509534

0.1042167 0.40681892 0.24608852 -0.07675847]

the residual r\_i: [ 2.69622171 -2.77401725 -0.45580225 -6.0313334 -1.34677131

6.35236034 -1.78455686 2.84339782 -1.37055448 -11.0147709 ]

eigenvalue: -160

corresponding eigenvector:

[ 0.18787597 -0.12891399 -0.15596508 -0.17697257 0.39995544 -0.30448313

0.7911731 0.0978554 0.02422188 0.06039721]

the residual r\_i: [ 2.78733338 5.4635517 8.03057721 13.98602928 -9.75482935 6.06849453

-7.77504502 -3.16893013 -2.44595369 0.97232915]

eigenvalue: 298

corresponding eigenvector:

[ 0.10521588 -0.36329037 0.37326108 0.30248125 0.20775001 -0.08244033

-0.11194036 0.53790596 -0.44953655 -0.26873396]

the residual r\_i: [-3.48647106 -9.11291061 5.42766289 4.75551409 2.72878192 -3.47991045

0.63521012 17.09050788 1.89772338 -3.14500678]

eigenvalue: -141

corresponding eigenvector:

[ 0.07546945 0.49240049 0.31983056 -0.30352503 0.4333836 -0.12757355

-0.2581399 0.33090501 0.41315897 -0.08059813]

the residual r\_i: [ 2.12291978 -9.607141 -8.3003783 3.7379584 -4.17186356 3.38095354

-2.59346279 -7.05423596 -9.70010205 -8.33810718]

eigenvalue: 172

corresponding eigenvector:

[ 0.53859626 0.03917275 0.0291065 0.02719861 -0.23510075 -0.32449744

-0.18403934 0.14987616 -0.10112504 0.69257639]

the residual r\_i: [ 0.84278174 -4.0935386 6.6183381 -4.90540584 -3.22941712 -9.9534812

2.571893 -5.2107288 -6.74613877 15.03041853]  
  
**Векторы**[ 0.84278174 -4.0935386 6.6183381 -4.90540584 -3.22941712 -9.9534812

2.571893 -5.2107288 -6.74613877 15.03041853]

## Вывод

С помощью степенного метода(метода скалярных произведений) удалось найти максимальное собственное значение с точностью 1𝑒−7 для симметрической вещественной матрицы 𝐴. Как видно из результатов вычислений, скорость нахождения зависит от заданной точности, а также от начального вектора 𝑦(0).

С помощью итерационного метода вращений Якоби мы можем получить собственные значения матрицы с точностью 1е-7, а также все собственные векторы для данных значений. Из вычислений видно, что метод зависит от самой матрицы и её размерности.