Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа №4.1

по курсу «Численные методы линейной алгебры»

«Вычисление собственных значений и собственных векторов симметричной матрицы методом А.Н. Крылова»

Студент группы ИУ9-71Б Окутин Д. А.

Преподаватель Посевин Д. П.

Москва 2024

1 Цель

Реализовать метод вычисления собственных значений и собственных векторов симметричной матрицы методом А.Н. Крылова.

2 Задание

1) Реализовать метод поиска собственных значений действительной симметричной матрицы A размером 4x4.

2) Проверить корректность вычисления собственных значений по теореме Виета.

3) Проверить выполнение условий теоремы Гершгорина о принадлежности собственных значений соответствующим объединениям кругов Гершгорина.

4) Вычислить собственные вектора и проверить выполнение условия ортогональности собственных векторов.

5) Проверить решение на матрице приведенной в презентации.

6) Продемонстрировать работу приложения для произвольных симметричных матриц размером n x n с учетом выполнения пунктов приведенных выше.

3 Реализация

Исходный код представлен в листинге 1 - 6.

Листинг 1: Генерация необходимых данных

```

using Random
using LinearAlgebra

function euclidean_norm(vec::Vector)
    return sqrt(sum(vec.^2))
end

function generate_symmetric_matrix(l::Int, r::Int, n::Int)
    A = rand(n, n) .* (r - l) .+ l
    A = (A + A') / 2
    return A
end

```

```
    function identity_matrix(n::Int)
        return Matrix{Float64}(I, n, n)
    end

    n = 3
    symmetric_matrix = generate_symmetric_matrix(-10,10,n)

    println("                                              :")
    println(symmetric_matrix)

    println(identity_matrix(3))
```

Листинг 2: Алгоритм Крылова

```

function krylov_algo(matrix::Matrix)
    p = []
    n = size(matrix, 1)
    D = copy(matrix)
    y = ones(n+1,n)
    A = zeros(n,n)
    for i in 2:n+1
        y[i,1:end] = D*y[i - 1,1:end]
    end
    for i in n:-1:1
        A[n-i+1,1:end]=y[i,1:end]
    end
    A = A'
    f = y[n+1,1:end]
    res_p = A\f

    p = [1.0]
    for pp in res_p
        push!(p,-pp)
    end

    return y, p
end
```

Листинг 3: Нахождение интервалов Гершгорина

```

function union_intervals(ints::Vector)
    union = []
    ints  = sort(copy(ints), by=x->x[1])
    for int in ints
        if length(union)>0 && union[end][2]>=int[1]-1
```

```
            union[end][2] = max(union[end][2], int[2])
        else
            push!(union,[int[1],int[2]])
        end
    end

    return union
end

function gershgorin_intervals(A::Matrix)
    n = size(A, 1)
    intervals = []

    for i = 1:n
        radius = sum(abs.(A[i, :])) - abs(A[i, i])
        center = A[i, i]
        push!(intervals, (center - radius, center + radius))
    end

    intervals = union_intervals(copy(intervals))

    return intervals
end

A = [1.0 2.0 3.0; 4.0 5.0 10.0; 7.0 8.0 9.0]
intervals = gershgorin_intervals(A)

println("                                                              A
    :")
intervals
```

Листинг 4: Вычисление собственных векторов и собственных значений

```

    function find_equatation_coeffs(P::Matrix)
    equatationCoeffs = copy(P[1,1:end]).*(-1)
    equatationCoeffs = [1.0; equatationCoeffs]

    return equatationCoeffs
end

function equatation_polynomy(a::Vector, x)
    val = 0
    n = length(a)
    for i in n-1:-1:0
        val += a[n - i] * (x^i)
    end
```

```

    return val
end


function find_eigen_values(eqCoeffs::Vector, intervals::Vector, step)
    values = []
    eps = 10e-7
    for interval in intervals
        left = interval[1]
        right = interval[2]
        l = Int(floor((right - left) / step))

        for i in 0:l
            x_left = left + i * step
            x_right = x_left + step
            y_left = equatation_polynomy(eqCoeffs, x_left)
            y_right = equatation_polynomy(eqCoeffs, x_right)
            alpha = y_left * y_right

            if alpha < 0
                while x_right - x_left >= eps
                    x_middle = (x_right + x_left) / 2
                    y_middle = polynomy(eqCoeffs, x_middle)
                    beta = y_left * y_middle
                    if beta < 0
                        x_right = x_middle
                    else
                        x_left = x_middle
                    end
                end
                push!(values,(x_right + x_left) / 2)
            elseif y_left == 0
                push!(values,x_left)
            elseif y_right == 0
                push!(values,x_right)
            end
        end
    end

    return values
end


function find_eigen_vectors_krylov(y, vals, p)
    n = length(p) - 1
```

```
    xs = zeros(n,n)
    for i in 1:n
        x = y[n,1:end]
        q_i = [1.0]
        for j in 2:n
            push!(q_i,vals[i] * q_i[j - 1] + p[j])
            x = x + q_i[j]* y[n - j+1,1:end]
        end
        xs[i,1:end]= x./ euclidean_norm(x)
    end

  return xs
end
```

Листинг 5: Функции проверки результатов

```

    function check_vieta(A::Matrix, vals::Vector)
    sum_eigs=sum(vals)
    sp = 0
    n = size(A,1)
    for i in 1:n
        sp += A[i,i]
    end
    if abs(sum_eigs-sp)>0.1
        println("\nVieta‘s theorem doesn‘t work")
    else
        println("\nVieta‘s theorem works")
    end
end

function check_gershorin(vals::Vector, intervals::Vector)
    for interval in intervals
        for val in vals
            if val > interval[2] || val < interval[1]
                println("Gershgorin‘s theorem error\n")
                return
            end
        end
    end

    println("Gershgorin‘s theorem works\n")
end

function check_ortogonal(n::Int, vecs::Matrix)
    for i in 1:n-1
        for j in i + 1:n
```

```
            scal = dot(vecs[i,1:end], vecs[j,1:end])
            if abs(scal) > 0.1
                println("Eigen vectors are not orthogonal\n")
                println(abs(scal))
                return
            end
        end
    end

    println("\nEigen vectors are orthogonal")
end
```

Листинг 6: Пример работы программы

```

n = 4
A = [2.2 1 0.5 2; 1 1.3 2 1; 0.5 2 0.5 1.6; 2 1 1.6 2]
println("matrix: $A\n")

intervals = gershgorin_intervals(A)
println("intervals: $intervals\n")

println("\nKRYLOV")
Ys, P = krylov_algo(A)
println("P: $P\n")
println("Y: $Ys\n")


eig_vals = find_eigen_values(P, intervals,10e-3)
println("eigvals: $eig_vals")

check_vieta(A, eig_vals)
check_gershorin(eig_vals, intervals)

# julia library
# eig_vects = eigvects(P)
# println("eigvects: $eig_vects")

eig_vects = find_eigen_vectors_krylov(Ys, eig_vals, P)
for i in 1:size(eig_vects,1)
    vec = eig_vects[i,1:end]
    println("eigvect_$i: $vec")
end

check_ortogonal(n, eig_vects)
```

4 Результаты

Результат представлен на рисунке 1.

```
matrix: [2.2 1.0 0.5 2.0; 1.0 1.3 2.0 1.0; 0.5 2.0 0.5 1.6; 2.0 1.0 1.6 2.0]

intervals: Any[[-3.5999999999999996, 6.6]]


KRYLOV
P: [1.0, -5.999999999999894, -0.2000000000005567, 12.734999999999483, -2.761599999998538]

Y: [1.0 1.0 1.0 1.0; 5.7 5.3 4.6 6.6; 33.34 28.39 26.31 37.26; 189.413 160.12699999999998 146.221 211.68599999999998; 1073.3181 901.7060999999999 826.7686 1196.2786]

eigvals: Any[-1.4200863647460937, 0.22263580322265686, 1.5454183959960939, 5.652032165527345]

Vieta`s theorem works
Gershgorin`s theorem works

eigvect_1: [-0.222042593227019, 0.5159106786075803, -0.7572739835488279, 0.33327071928355184]
eigvect_2: [0.5219206423771644, 0.45486963802490893, -0.15344683758462913, -0.7050861816111366]
eigvect_3: [0.6289298182269153, -0.5725741727668471, -0.48565374919670573, 0.20185771304709482]
eigvect_4: [0.5317360709371982, 0.44619411921045116, 0.40881553103618407, 0.5924841098543349]

Eigen vectors are orthogonal
```

Рис. 1 — Полученные результаты

5 Выводы

В результте выполнения данной лабораторной работы был реализован алгоритм, позволяющий анализировать матрицы с использованием метода Крылова, вычислять интервалы Гершгорина и находить их собственные значения и собственные вектора на языке программирования Julia. Результаом работы является успешное нахождение собственных значений и векторов матрицы, что подтверждает корректность алгоритмов. Также была проверена теорема Виета для собственных значений и ортогональность собственных векторов.