Вычислительный практикум

Отчет по третьему заданию

Автор: Павел Соболев

29 мая 2021 г.

Часть 1

Теория

Дана одна или несколько функций a_{ij}, задающих элементы симметричной матрицы. Для этой матрицы необходимо найти

- наибольшее по модулю собственное число λ_{max} степенным методом;
- наименьшее по модулю собственное число λ_{min} обратным степенным методом;
- все собственные числа, используя классический метод Якоби.

1.1 Степенной метод

Выбрав начальный вектор x_0, проводим итерации следующего вида

$$\tilde{x}_{n+1} = Ax_n,\ x_{n+1} = \frac{\tilde{x}_{n+1}}{\{\tilde{x}_{n+1}\}_1}. \tag{1.1}$$

В процессе $\{\tilde{x}_{n+1}\}_1 \longrightarrow \lambda_{max}$.

1.2 Обратный степенной метод

Выбрав начальный вектор x_0, проводим итерации следующего вида

$$A\tilde{x}_{n+1} = x_n,\ x_{n+1} = \frac{\tilde{x}_{n+1}}{\{\tilde{x}_{n+1}\}_1}. \tag{1.2}$$

Решая системы линейных уравнений, в процессе $\{\tilde{x}_{n+1}\}_1 \longrightarrow \lambda_{min}$.

1.3 Классический метод Якоби

Пусть A – симметричная матрица, а $G = G(i,j,\theta)$ – матрица вращения. Тогда матрица $A' = G^\top AG$ симметрична и подобна матрице А. Вращение выражается как

$$A'_{ii} = c^2 A_{ii} - 2sc\,A_{ij} + s^2 A_{jj}, \tag{1.3}$$
$$A'_{jj} = s^2 A_{ii} + 2sc\,A_{ij} + c^2 A_{jj}, \tag{1.4}$$
$$A'_{ij} = A'_{ji} = (c^2 - s^2)A_{ij} + sc\,(A_{ii} - A_{jj}), \tag{1.5}$$
$$A'_{ik} = A'_{ki} = c\,A_{ik} - s\,A_{jk},\ k \neq i,j, \tag{1.6}$$
$$A'_{jk} = A'_{kj} = s\,A_{ik} + c\,A_{jk},\ k \neq i,j, \tag{1.7}$$
$$A'_{kl} = A_{kl},\ k,l \neq i,j, \tag{1.8}$$

где $s = \sin\theta$ и $c = \cos\theta$, причем можем выбрать θ так, что $A'_{ij} = 0$:

$$\operatorname{tg}(2\theta) = \frac{2A_{ij}}{A_{jj} - A_{ii}}. \tag{1.9}$$

Если $A_{jj} = A_{ii}$, то

$$\theta = \frac{\pi}{4}. \tag{1.10}$$

Для оптимального эффекта A_{ij} выбирается как наибольший по модулю внедиагональный элемент. Вращения продолжаются до тех пор, пока матрица не приобретет диагональный вид.

Часть 2

Реализация

Алгоритм реализован на языке программирования `Julia` в виде локального модуля `A3` и расположен в GitHub репозитории `paveloom-p/P12` в папке `A3`. Для воспроизведения результатов следуй инструкциям в файле `README.md`. Далее приводятся только сниппеты кода.

2.1 Степенной метод

Листинг 1: Поиск наибольшего по модулю собственного числа

```julia
# Get the matrix
A = Symmetric(Array{Float64}(undef, n, n))
for i = 1:n, j = i:n
    A.data[i,j] = aᵢⱼ(i, j)
end

# Get a vector of initial values
x₀ = get_x₀(A)

# Find the maximum modulo eigenvalue

xₙ = copy(x₀)
λₘₐₓ = 0
λₚᵣₑᵥ = 1

while abs(1 - λₘₐₓ / λₚᵣₑᵥ) > ϵₚ
    λₚᵣₑᵥ = λₘₐₓ
    x̃ₙ₊₁ = A * xₙ
    λₘₐₓ = x̃ₙ₊₁[1]
    xₙ = x̃ₙ₊₁ / λₘₐₓ
end
```

Конструктор типа `Symmetric` позволяет получить вид на верхний или нижний треугольник симметричной матрицы и предоставляет доступ к соответствующим оптимизациям. Интерфейс, однако, ожидает передачу всей матрицы, хотя и необязательно полностью инициализированной.

Функция `get_x₀` реализует получение вектора-столбца начальных значений. По умолчанию это вектор из единиц.

2.2 Обратный степенной метод

Листинг 2: Поиск наименьшего по модулю собственного числа

```
# Find the minimum modulo eigenvalue

xₙ = copy(x₀)
λ⁻¹ₘᵢₙ = 0
λ⁻¹ₚᵣₑᵥ = 1

while abs(1 - λ⁻¹ₘᵢₙ / λ⁻¹ₚᵣₑᵥ) > ϵₚ
    λ⁻¹ₚᵣₑᵥ = λ⁻¹ₘᵢₙ
    x̃ₙ₊₁ = A \ xₙ
    λ⁻¹ₘᵢₙ = x̃ₙ₊₁[1]
    xₙ = x̃ₙ₊₁ / λ⁻¹ₘᵢₙ
end
```

2.3 Классический метод Якоби

Листинг 3: Поиск всех собственных чисел (часть 1)

```
# Use the Jacobi eigenvalue algorithm to find all eigenvalues
λ = Vector{Float64}(undef, n)

for _ in 1:iterₘₐₓ

    aₘₐₓ = k = l = 0

    for i in 1:n-1, j in i+1:n
        if abs(A[i,j]) ≥ aₘₐₓ
            aₘₐₓ = abs(A[i,j])
            k = i; l = j
        end
    end

    if aₘₐₓ < ϵⱼ
        λ = Diagonal(A).diag
        break
    end

    Δ = A[l,l] - A[k,k]

    t = if abs(A[k,l]) < abs(Δ) * 1e-36
        A[k,l] / Δ
    else
        ϕ = Δ / (2 * A[k,l])
        if ϕ < 0
            -1 / (abs(ϕ) + √(ϕ^2 + 1))
        else
            1 / (abs(ϕ) + √(ϕ^2 + 1))
        end
    end

    <...>
```

Листинг 4: Поиск всех собственных чисел (часть 2)

```
    <...>

    c = 1 / √(t^2 + 1)
    s = t * c
    τ = s / (1 + c)

    a_temp = A[k,l]
    A.data[k,l] = 0
    A.data[k,k] -= t * a_temp
    A.data[l,l] += t * a_temp

    for i in 1:k-1
        a_temp = A[i,k]
        A.data[i,k] = a_temp - s * (A[i,l] + τ * a_temp)
        A.data[i,l] += s * (a_temp - τ * A[i,l])
    end

    for i in k+1:l-1
        a_temp = A[k,i]
        A.data[k,i] = a_temp - s * (A[i,l] + τ * A[k,i])
        A.data[i,l] += s * (a_temp - τ * A[i,l])
    end

    for i in l+1:n
        a_temp = A[k,i]
        A.data[k,i] = a_temp - s * (A[l,i] + τ * a_temp)
        A.data[l,i] += s * (a_temp - τ * A[l,i])
    end

end
```

Изменение данных типа `Symmetric` в текущей версии требует доступа к полю `data`. Изменение одного элемента, однако, автоматически изменяет и симметричный ему элемент.

2.4 Пример

Симметричная матрица задана функцией

$$a_{ij} = \frac{(-1)^{i+j}}{|i-j|+1}. \tag{2.1}$$

Листинг 5: Определение и решение проблемы

```
include("src/A3.jl")
using .A3

λₘₐₓ, λₘᵢₙ, λ = solve(
    Problem(
        function(i, j)
            return (-1)^(i+j) / (abs(i-j) + 1)
        end,
    ),
)
```

Листинг 6: Результат

```
λₘₐₓ = 3.478950198080321
λₘᵢₙ = 0.3915295599251142

λ:
1.653468417452173
0.48334088187775437
0.6592071525661884
0.4078747606964994
1.1122936734459035
0.43707220669552194
0.39152955992511285
0.8237636815142848
3.478950198080321
0.5524994677462397
```