Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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 № 5

по курсу «Численные методы линейной алгебры»

«Методы Якоби и Зейделя дял решения СЛАУ»

Студент группы ИУ9-72Б Шемякин В.А.

Преподаватель Посевин Д. П.

Москва 2025

1 Задание

Реализовать методы Якоби и Зейделя для решения СЛАУ, провести сравнение методов по количеству итераций и времени работы.

2 Результаты

Исходный код программы представлен в листинге 1.

```
import LinearAlgebra
import Random
using Printf
using LinearAlgebra

function generate_matrix(l::Float64, r::Float64, n::Int)
    return rand(n, n) .* (r - l) .+ l
end

function generate_vector(l::Float64, r::Float64, n::Int)
    return rand(n) .* (r - l) .+ l
end

function make_strictly_diagonally_dominant(A::AbstractMatrix; margin::Float64=1.0)
    n, m = size(A)
    @assert n == m "Matrix must be square"
    A = copy(Matrix{Float64}(A))
    for i in 1:n
        row_sum_off = sum(abs, A[i, :]) - abs(A[i, i])
        A[i, i] = sign(A[i,i] == 0 ? 1.0 : A[i,i]) * (row_sum_off + margin)
    end
    return A
end

function has_diagonal_dominance(A::AbstractMatrix)
    n, m = size(A); @assert n == m "Matrix must be square"
    for i in 1:n
        s = sum(abs, A[i, :]) - abs(A[i,i])
        if abs(A[i,i]) < s
            return false
```

```
31          end
32      end
33      return true
34  end
35
36  function split_LDU(A::AbstractMatrix)
37      n, m = size(A); @assert n == m "
                                       "
38      L = Matrix(tril(A, -1))
39      D = Diagonal(diag(A))
40      U = Matrix(triu(A,  1))
41      return L, D, U
42  end
43
44  function jacobi(A::AbstractMatrix, b::AbstractVector; x0=nothing,
45                  tol::Float64=1e-8, maxiter::Int=100_000)
46      n, m = size(A); @assert n == m == length(b)
47      x = x0 === nothing ? zeros(Float64, n) : copy(x0)
48      L, D, U = split_LDU(A)
49      Dinv = Diagonal(1.0 ./ diag(D))
50      P = -(Dinv * (L + U))
51      g = Dinv * b
52      for k in 1:maxiter
53          xnew = P * x + g
54          inc = norm(xnew - x) / max(1.0, norm(xnew))
55          x = xnew
56          if inc < tol
57              return (x, k, inc)
58          end
59      end
60      return (x, maxiter, norm(P*x + g - x))
61  end
62
63  function seidel(A::AbstractMatrix, b::AbstractVector; x0=nothing,
64                      tol::Float64=1e-8, maxiter::Int=100_000)
65      n, m = size(A); @assert n == m == length(b)
66      x = x0 === nothing ? zeros(eltype(b), n) : copy(x0)
67      L = Matrix(tril(A, -1)); D = Diagonal(diag(A)); U = Matrix(triu
        (A, 1))
68      DL = LowerTriangular(D + L)
69      for k in 1:maxiter
70          xnew = DL \ (b - U * x)
71          inc = norm(xnew - x) / max(1.0, norm(xnew))
72          x = xnew
73          if inc < tol
```

```
74              return (x, k, inc)
75          end
76      end
77      return (x, maxiter, norm(DL \ (b - U * x) - x))
78  end
79
80  function seidel_solver(A::AbstractMatrix, b::AbstractVector; x0=
        nothing,
81                                      tol::Float64=1e-8, maxiter::Int=100
        _000)
82      x, k, inc = seidel(A, b; x0=x0, tol=tol, maxiter=maxiter)
83      return (x, k, inc)
84  end
85
86  time_s() = time_ns() / 1e9
87
88  function run_and_time(solver, A, b; kwargs...)
89      t0 = time_s()
90      out = solver(A, b; kwargs...)
91      t1 = time_s()
92      x, k, res = out[1], out[2], out[3]
93      return (x=x, k=k, res=res, t=t1 - t0)
94  end
95
96  function print_matrix(A; name::AbstractString="A", preview::Int=6,
        print_full::Bool=false)
97      n, m = size(A)
98      println("$name:                  $n $m")
99      if print_full || (n <= preview && m <= preview)
100         show(stdout, "text/plain", A); println()
101     else
102         p = min(preview, n, m)
103         println("                $p $p                    :")
104         show(stdout, "text/plain", A[1:p, 1:p]); println()
105     end
106 end
107
108 function print_vector(v; name::AbstractString="b", preview::Int=10,
        print_full::Bool=false)
109     n = length(v)
110     println("$name:              $n")
111     if print_full || n <= preview
112         show(stdout, "text/plain", v); println()
113     else
114         p = min(preview, n)
```

```
115         println("              $p                    :")
116         show(stdout, "text/plain", v[1:p]); println()
117     end
118 end
119
120 function compare_run(n::Int=5; l::Float64=-1.0, r::Float64=1.0,
121                      tol::Float64=1e-10, maxiter::Int=200_000,
122                      seed::Union{Int,Nothing}=nothing,
123                      print_data::Bool=true, preview::Int=6,
    print_full::Bool=false)
124
125     if seed !== nothing
126         Random.seed!(seed)
127     end
128
129     Araw = generate_matrix(l, r, n)
130     A = make_strictly_diagonally_dominant(Araw; margin=1.0)
131     b = generate_vector(l, r, n)
132     x0 = zeros(n)
133
134     if print_data
135         println("                              ")
136         print_matrix(A; name="A", preview=preview, print_full=
    print_full)
137         print_vector(b; name="b", preview=max(10, preview),
    print_full=print_full)
138         println()
139     end
140
141     jac = run_and_time(jacobi, A, b; x0=x0, tol=tol, maxiter=
    maxiter)
142     gs  = run_and_time(seidel_solver, A, b; x0=x0, tol=tol, maxiter
    =maxiter)
143
144     println("                                    ")
145     @printf("%-16s  %-12s  %-12s  %-12s\n", "          ", "
                  ", "                        ", "          ,   ")
146     @printf("%-16s  %-12d  %-12.3e  %-12.6f\n", "          ",  jac
    .k, jac.res, jac.t)
147     @printf("%-16s  %-12d  %-12.3e  %-12.6f\n", "              ",
    gs.k,  gs.res,  gs.t)
148     println()
149
150     return nothing
151 end
```

```
152
153
154 function main(; n::Int=100, l::Float64=-1.0, r::Float64=1.0,
155                  tol::Float64=1e-10, maxiter::Int=200_000,
156                  seed::Union{Int,Nothing}=nothing,
157                  print_data::Bool=true, preview::Int=6, print_full::
    Bool=false)
158     compare_run(n; l=l, r=r, tol=tol, maxiter=maxiter, seed=seed,
159                  print_data=print_data, preview=preview, print_full=
    print_full)
160 end
161
162 main()
```

Результат запуска представлен на рисунке 1.

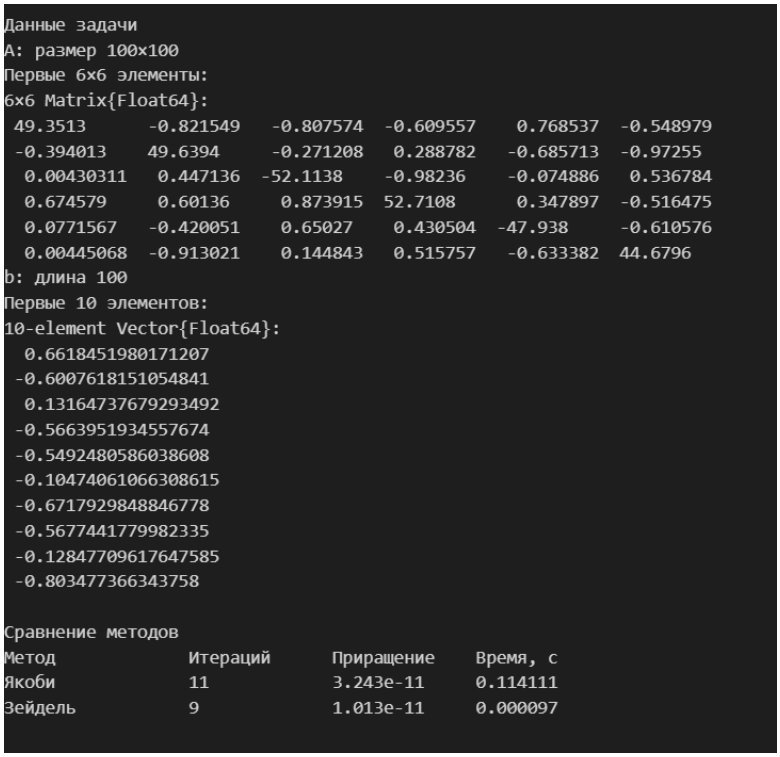

Рис. 1 — Результат

.