



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 для решения СЛАУ»**

Студент группы ИУ9-72Б Шемякин В.А.

Преподаватель Посевин Д. П.

*Mosква 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.

```
1 import LinearAlgebra
2 import Random
3 using Printf
4 using LinearAlgebra
5
6 function generate_matrix(l::Float64, r::Float64, n::Int)
7     return rand(n, n) .* (r - l) .+ l
8 end
9
10 function generate_vector(l::Float64, r::Float64, n::Int)
11     return rand(n) .* (r - l) .+ l
12 end
13
14 function make_strictly_diagonally_dominant(A::AbstractMatrix;
15     margin::Float64=1.0)
16     n, m = size(A)
17     @assert n == m "
18         "
19     = copy(Matrix{Float64}(A))
20     for i in 1:n
21         row_sum_off = sum(abs, [i, :]) - abs([i, i])
22         [i, i] = sign([i, i] == 0 ? 1.0 : [i, i]) * (
23             row_sum_off + margin)
24     end
25     return
26 end
27
28 function has_diagonal_dominance(A::AbstractMatrix)
29     n, m = size(A); @assert n == m "
30         "
31     for i in 1:n
32         s = sum(abs, A[i, :]) - abs(A[i, i])
33         if abs(A[i, i]) < s
34             return false
35         end
36     end
37     return true
38 end
```

```

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

```

```

74         return (x, k, inc)
75     end
76 end
77 return (x, maxiter, norm(DL \ (b - U * x) - x))
78
79
80 function seidel_solver(A::AbstractMatrix, b::AbstractVector; x0=
81           nothing,
82           tol::Float64=1e-8, maxiter::Int=100
83           _000)
84     x, k, inc = seidel(A, b; x0=x0, tol=tol, maxiter=maxiter)
85     return (x, k, inc)
86 end
87
88 time_s() = time_ns() / 1e9
89
90
91 function run_and_time(solver, A, b; kwargs...)
92     t0 = time_s()
93     out = solver(A, b; kwargs...)
94     t1 = time_s()
95     x, k, res = out[1], out[2], out[3]
96     return (x=x, k=k, res=res, t=t1 - t0)
97 end
98
99
100 function print_matrix(A; name::AbstractString="A", preview::Int=6,
101           print_full::Bool=false)
102     n, m = size(A)
103     println("$name: $n $m")
104     if print_full || (n <= preview && m <= preview)
105         show(stdout, "text/plain", A); println()
106     else
107         p = min(preview, n, m)
108         println(" $p $p :")
109         show(stdout, "text/plain", A[1:p, 1:p]); println()
110     end
111 end
112
113
114 function print_vector(v; name::AbstractString="b", preview::Int=10,
115           print_full::Bool=false)
116     n = length(v)
117     println("$name: $n")
118     if print_full || n <= preview
119         show(stdout, "text/plain", v); println()
120     else
121         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,
124                      print_full::Bool=false)
125
126   if seed !== nothing
127     Random.seed!(seed)
128   end
129
130   Araw = generate_matrix(l, r, n)
131   A = make_strictly_diagonally_dominant(Araw; margin=1.0)
132   b = generate_vector(l, r, n)
133   x0 = zeros(n)
134
135   if print_data
136     println("          ")
137     print_matrix(A; name="A", preview=preview, print_full=
138     print_full)
139     print_vector(b; name="b", preview=max(10, preview),
140     print_full=print_full)
141     println()
142   end
143
144   jac = run_and_time(jacobi, A, b; x0=x0, tol=tol, maxiter=
145   maxiter)
146   gs = run_and_time(seidel_solver, A, b; x0=x0, tol=tol, maxiter=
147   maxiter)
148
149   println("          ")
150   @printf("%-16s  %-12s  %-12s  %-12s\n", "          ", "          ",
151           "          ", "          ", "          ")
152   @printf("%-16s  %-12d  %-12.3e  %-12.6f\n", "          ", "          ",
153           jac.k, jac.res, jac.t)
154   @printf("%-16s  %-12d  %-12.3e  %-12.6f\n", "          ", "          ",
155           gs.k, gs.res, gs.t)
156   println()
157
158   return nothing
159 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::
158           Bool=false)
159   compare_run(n; l=l, r=r, tol=tol, maxiter=maxiter, seed=seed,
160               print_data=print_data, preview=preview, print_full=
161               print_full)
162 end
163
164 main()
```

Результат запуска представлен на рисунке 1.

```
Данные задачи
A: размер 100x100
Первые 6x6 элементы:
6x6 Matrix{Float64}:
 49.3513   -0.821549   -0.807574   -0.609557    0.768537   -0.548979
 -0.394013    49.6394   -0.271208    0.288782   -0.685713   -0.97255
  0.00430311   0.447136   -52.1138   -0.98236   -0.074886    0.536784
  0.674579    0.60136    0.873915    52.7108    0.347897   -0.516475
  0.0771567   -0.420051    0.65027    0.430504   -47.938   -0.610576
  0.00445068   -0.913021    0.144843    0.515757   -0.633382   44.6796
b: длина 100
Первые 10 элементов:
10-element Vector{Float64}:
 0.6618451980171207
 -0.6007618151054841
  0.13164737679293492
 -0.5663951934557674
 -0.5492480586038608
 -0.10474061066308615
 -0.6717929848846778
 -0.5677441779982335
 -0.12847709617647585
 -0.803477366343758

Сравнение методов
Метод      Итераций      Приращение      Время, с
Якоби        11           3.243e-11     0.114111
Зейдель       9            1.013e-11    0.000097
```

Рис. 1 — Результат