Universidade Federal de Juiz de Fora - UFJF
Jonatas Dias Machado Costa - 202165077AC
Lucas Marins Ramalho de Lima - 202165555C

1. Seja o sistema linear:

$$\mathbf{Ax} = \mathbf{b}$$

onde,

$$\mathbf{A} = A_{i,j} = \frac{1}{i+j+1} \quad \text{e} \quad \mathbf{b} = b_i = \frac{1}{i+n+1}.$$

Supondo a matriz $\mathbf{A}_{n\times n}$, com diferentes dimensões n (ex. $n = 10, 100, 500, 1000$), obtem-se os seguintes resultados:

– **Gauss sem pivoteamento**:

Algorithm 1 Gauss sem pivoteamento

```
for i = 0, ..., n do
    if A[i][i] = 0 then
        Divisão por zero detectada
        Sair do programa
    end if
    for j = 0, ..., n + 1 do
        razao ← A[j][i]/A[i][i]
        for k = 0, ..., n + 1 do
            A[j][k] ← A[j][k] − razao ∗ A[i][k]
        end for
    end for
end for
solucao[n − 1] = A[n − 1][n]/A[n − 1][n − 1]
for i = n − 2, ..., −1 do
    solucao[i] ← A[i][n]
    for j = i + 1, ..., n do
        solucao[i] ← solucao[i] − A[i][j] ∗ solucao[j]
    end for
    solucao[i] ← solucao[i]/A[i][i]
end for
return solucao
```

– **Gauss (pivoteamento)**:

Algorithm 2 Gauss (pivoteamento)

```
1: for k = 0, ..., n do
2:     i_max ← k
3:     pivo_max = A[i_max][k]
4:     for i = k + 1, ..., n do
5:         if |A[i][k]| > pivo_max then
6:             pivo_max ← A[i][k]
7:             i_max ← i
8:         end if
9:     end for
10:    if A[k][i_max] = 0 then
11:        Divisãoporzerodetectada
12:        Sairdoprograma
13:    end if
14:    if i_max! = k then
15:        troca_linha(A, k, i_max, n)
16:    end if
17:    for i = k + 1, ..., n do
18:        f ← A[i][k]/A[k][k]
19:        for j = k + 1, ..., n + 1 do
20:            A[i][j] ← A[i][j] − f ∗ A[k][j]
21:        end for
22:        A[i][k] = 0
23:    end for
24: end for
25: for i = n − 1, ..., −1 do
26:    solucao[i] = A[i][n]
27:    for j = i + 1, ..., n do
28:        solucao[i] ← solucao[i] − A[i][j] ∗ solucao[j]
29:    end for
30:    solucao[i] ← solucao[i]/A[i][i]
31: end for
32: return solucao
```

– **Decomposição LU**:

Algorithm 3 Decomposição LU

```
1: L, U ← decompoeLU(matriz)
2: //ResolucaoL * y = b
3: y[0] ← B[0]/L[0][0]
4: for i = 1, ..., n do
5:     soma ← 0
6:     for j = 0, ..., i do
7:         soma ← soma + L[i][j] * y[j]
8:     end for
9:     y[i] ← (B[i] − soma)/L[i][i]
10: end for
11: //ResolucaoU * x = y
12: x[n − 1] ← y[n − 1]/U[n − 1][n − 1]
13: for i = n − 1, ..., −1 do
14:     soma ← y[i]
15:     for j = i + 1, ..., n do
16:         soma ← soma − U[i][j] * x[j]
17:     end for
18:     x[i] ← soma/U[i][i]
19: end for
20: return x
```

– **Cholesky**:

Algorithm 4 Cholesky

```
1: for i = 0, ..., n do
2:     for j = 0, ..., i + 1 do
3:         S ← Soma de L[i][k] * L[j][k] para todo k = i, ..., j
4:         if i == j then
5:             L[i][j] ← RaizQuadrada(A[i][i] − s)
6:         end if
7:         if i! = j then
8:             1.0/L[j][j] * (A[i][j] − s
9:         end if
10:    end for
11:    Lt ← transposta(L)
12:    Y ← substituicaoRegressiva(L, B)
13:    X ← substituicaoRegressiva(Lt, Y)
14: return X
```

Tabela 1: Tempo de execução (em segundos) de cada método para uma matriz de ordem n

n	Gauss Pivoteado	Gauss sem Pivoteamento	Decomposição LU	Cholesky
10	0.002158880233765	0.004033088684082	0.001725912094116	0.000510692596436
100	0.820391654968262	0.766083955764771	0.243905067443848	-
500	64.0267498493195	85.0565690994263	27.3007416725159	-
1000	494.948547840118	756.900232791901	217.466570138931	-

- Na tabela acima, observa-se que o método de eliminação de Gauss com pivoteamento é mais rápido que o mesmo método sem o pivoteamento da matriz. O pivoteamento acelera o processo de convergência do método de Gauss porque garante que o elemento pivô seja o mais próximo possível de zero em cada iteração. Isso significa que o número de iterações necessárias para resolver o sistema é menor, o que leva a uma solução mais rápida.
- Além disso, observa-se que o método de Cholesky pôde ser aplicado apenas para $n = 10$. A eliminação de Cholesky só pode ser aplicada em matrizes positivas definidas, e de acordo com o critério de Sylvester, uma matriz $A \in R$ n×n e positiva definida, se e somente se:
  det(Ak) ¿ 0, k = 1, 2, . . . , n

  $$det(A_k) > 0, \quad \forall k = 1, 2, \ldots, n$$

  onde Ak e a matriz menor principal de ordem k (a matriz k × k formada pelas k primeiras linhas e pelas k primeiras colunas). Na matriz do problema estudado, é possível observar que a condição não se satisfaz a partir de $n = 14$, onde a determinante da matriz assume valor negativo $det(A_{14}) = -8.16944e - 107$
- O método mais rápido para a resolução da matriz em estudo é a decomposição LU, devido a característica de simetria da matriz estudada que a mais fácil de ser decomposta em matrizes triangulares superiores e inferiores. Dessa forma, a decomposição LU faz menos operações que as demais e performa melhor nessa matriz.
- Determinando o erro cometido, por cada um dos métodos utilizados, através do resíduo calculado na norma do máximo, dado por:

  $$\|\mathbf{Ax} - \mathbf{b}\|_\infty = \max_{1 \leq i \leq n} |A_{i,j} x_i - b_i|, \quad \forall j \in [1, n]$$

  onde x_i é o vetor solução:

Tabela 2: Erro cometido pelos métodos na matriz de ordem n

n	Gauss Pivoteado	Gauss sem Pivoteamento	Decomposição LU	Cholesky
10	1.08E-19	8.13E-20	1.36E-19	2.35E-17
100	1.02E-19	1.08E-19	3.66E-19	-
500	9.95E-21	8.68E-21	2.99E-20	-
1000	1.37E-20	1.61E-20	3.39E-20	-

2. A equação diferencial bidimensional

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 10 \quad \text{em} \quad [0,1]^2,$$
$$u = 0 \quad \text{sobre o contorno do domínio,}$$

quando resolvida pelo método de Diferenças Finitas dá origem a um sistema $\mathbf{Au} = \mathbf{b}$, conforme representado na matriz abaixo:

com $\mathbf{b} = \mathbf{1}$ e $\mathbf{A}$ a matriz heptadiagonal

$$\mathbf{A} = -(\sqrt{n}-1)^2 \begin{bmatrix} T & I & & & & \\ I & T & I & & & \\ & I & T & I & & \\ & & \ddots & \ddots & \ddots & \\ & & & I & T & I \\ & & & & I & T \end{bmatrix}, \quad \text{com} \quad T = \begin{bmatrix} -4 & 1 & & & & \\ 1 & -4 & 1 & & & \\ & 1 & -4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -4 & 1 \\ & & & & 1 & -4 \end{bmatrix}$$

Tomando $n = 81, 289, 1089, 4225, 16641$ podemos montar uma tabela comparando o tempo de execução dos métodos diretos, implementados no item anterior, com os métodos iterativos de Jacobi e Gauss-Seidel adotando diferentes valores da tolerância ε.

Tabela 3: Tempo de execução (em segundos) com $\epsilon = 10^{-16}$ de todos os métodos estudados

n	Gauss (pivoteamento)	Gauss	Decomposição LU	Cholesky	Gauss-Seidel	Jacobi
81	0.228669	0.336494	0.102443	0.0403025	0.896717	0.940109
289	10.816022	15.390144	3.345512	0.801468	40.745250	42.643204
1089	64.026750	616.698538	209.756098	45.185066	1070.693471*	1052.132564*
4225	318.123947	6122.712390	1049.772314	221.945216	986.812487*	976.732689*

Tabela 4: Tempo de execução (em segundos) com $\epsilon = 10^{-8}$ para os métodos iterativos

n	Gauss-Seidel	Jacobi
81	0.506811	0.508443
289	19.862020	19.841964
1089	1069.492442	1047.515233
4225	955.597634*	945.961754*

O cálculo de erro adotado como critério de parada dos métodos iterativos foi erro absoluto com a norma do máximo dado por:

$$\|\mathbf{Ax} - \mathbf{b}\|_\infty = \max_{1 \le i \le n} |A_{i,j}x_i - b_i|, \quad \forall j \in [1, n]$$

O método de eliminação de Cholesky foi o mais rápido dentre os métodos testados, porém, o erro cometido pela solução obtida ao aplicar esse método é muito maior que os demais, fazendo com que a sua aproximação não compense a velocidade de seu cálculo.
Já com $n = 1089$, o Cholesky apresentou erro 9.66 enquanto o Gauss com pivoteamento obteve $9.95 * 10^{-21}$, sendo Cholesky 41% mais rápido.

A escolha do valor de ε como $\varepsilon = 10^{-16}$ tem por objetivo aproximar a resposta dos métodos iterativos com os métodos diretos, que possuem erros nessa grandeza. Essa aproximação, por sua vez, torna possível a comparação dos tempos de execução entre um método direto e um iterativo. Já a escolha de $\varepsilon = 10^{-8}$ busca uma solução com menos iterações.

O número de iterações feitas para alcançar o critério de parada ε pelos métodos iterativos para cada ordem de matriz n e cada valor de erro obtido podem ser observados nas tabelas abaixo:

Tabela 5: Número de iterações dos métodos Gauss Seidel e Jacobi e os tempos de execução para cada n com $= 10-16$

n	N° Iterações (Gauss)	Tempo (Gauss)	N° Iterações (Jacobi)	Tempo(Jacobi)
81	395	0.896717	372	0.940109
289	1220	40.745250	1220	42.643204
1089	2212*	1070.693471*	2212*	1052.132564*
4225	100*	986.812487*	100*	976.732689*

Tabela 6: Número de iterações dos métodos Gauss Seidel e Jacobi e tempos de execução para cada n com $= 10-8$

n	N° Iterações (Gauss)	Tempo(Gauss)	N° Iterações (Jacobi)	Tempo(Jacobi)
81	189	0.506811	189	0.508443
289	618	19.862020	618	19.841964
1089	2211	1069.492442	2212	1047.515233
4225	100*	955.597634*	100*	945.961754*

*** Número de iterações máximas atingidas. O número de iterações máximas foi escolhido de modo a impedir a execução dos métodos iterativos por um tempo muito extenso, tendo em vista o custo computacional e o erro obitdo.**

Nota-se que a razão (tempo de execução/quantidade de iteração) aumenta bastante de acordo com o aumento da ordem da matriz. Isso faz com que o tempo de execução de 100 iterações para uma matriz de ordem 4225 se aproxime do tempo de execução de 2212 iterações para uma matriz de ordem 1089.