Universidade Estadual Paulista Julio de Mesquista Filho - UNESP
Instituto de Biociências
Departamento de Bioestatística, Biologia Vegetal, Parasitologia e Zoologia
Programa de Pós-Graduação em Biometria
Métodos Numéricos e Computacionais

Trabalho Computacional 2 - Métodos Iterativos para Sistemas Lineares

Aluno(a): Richard Castro Júnior

Professor: Daniela Renata Cantane

Botucatu-SP
2022

1 Introdução

1.1 Sistemas lineares

Um sistema linear de m equações e n incógnitas é um conjunto de m equações lineares onde cada uma possui n variáveis. Seja A = $[a_{ij}]$ uma matriz de ordem m × n definida sobre $\mathbb{R}$, isto é, seus elementos $a_{i,j} \in \mathbb{R}$ para $1 \leq i \leq m$ e $1 \leq j \leq n$. Consideremos o problema de encontrar escalares $x_1, , x_n \in \mathbb{R}$ satisfazendo simultaneamente o seguinte sistema de equações lineares (PULINO, 2012):

$$\begin{cases} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & y_1 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{i1}x_1 & + & a_{i2}x_2 & + & \cdots & + & a_{in}x_n & = & y_i \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & y_m \end{cases}$$

conhecendo os escalares $y_1, , y_n \in \mathbb{R}$.

O sistema linear pode ser representado em sua forma matricial $AX = Y$, onde

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{in} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} \quad \text{e} \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_m \end{bmatrix}$$

sendo a matriz A de ordem $m \times n$ a matriz dos coeficientes do sistema linear; o vetor coluna X de ordem $n \times 1$, o vetor de incógnitas; e o vetor coluna Y de ordem $m \times 1$, o vetor do lado direito do sistema linear (PULINO, 2012).

1.2 Métodos iterativos

Um método iterativo fornece uma sequência de soluções aproximadas de modo que cada solução aproximada é obtida da anterior pela aplicação de um mesmo procedimento. Um processo iterativo

é definido conforme a sequência de vetores $x^{(0)}$, $x^{(1)}$, $x^{(2)}$, . . . , $x^{(k)}$ produzida que deve convergir para a solução $\bar{x}$ ($\lim_{k\to\infty} x^{(k)} = \bar{x}$), independentemente da escolha do chute inicial $x^{(0)} \in \mathbb{R}^n$.

De modo geral, a construção do método iterativo considera a tranformação do sistema original $Ax = b$ para a forma equivalente $x = Cx + d$ e, posteriormente, a partir desta nova forma e de uma solução aproximada inicial $x^{(0)}$, determinamos a sequência de soluções aproximadas considerando o processo iterativo (MELO; SANTOS, 2013):

$$x^{(k+1)} = Cx^{(k)} + d \qquad k = 0, 1, 2, , n \quad \text{onde}$$

$$C \to \text{matriz iterativa}(n \times n) \quad \text{e} \quad d \to \text{vetor}(n \times 1)$$

O processo iterativo é repetido até que o vetor $x^{(k)}$ esteja suficientemente próximo do vetor $x^{(k-1)}$. Medindo a distância $d^{(k)} = \dfrac{max|X^{(n)} - X^{(n-1)}|}{max|X^{(n)}|} < \epsilon$, com $X^{(n)}$ uma solução aproximada do sistema. Assim, dada uma precisão ϵ, o vetor $x^{(k)}$ será escolhido como $\bar{x}$, solução aproximada da solução exata, se $d^{(k)} < \epsilon$ (MELO; SANTOS, 2013).

1.2.1 Método Iterativo de Gauss-Jacobi

Utilizando-se da forma algébrica, o método de Gauss-Jacobi transforma o sistema linear $Ax = b$ em $x = Cx + d$ é a seguinte (MELO; SANTOS, 2013):

Considere o sistema linear original:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \cdots a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \cdots a_{2n}x_n = b_2 \\ \quad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 \cdots a_{nn}x_n = b_n \end{cases}$$

e supondo $a_{ii} \neq 0, i = 1, \cdots, n$, isolamos $x_1, x_2, x_3, \cdots x_n$ em cada equação (MELO; SANTOS, 2013):

$$\begin{cases} x_1 = \dfrac{1}{a_{11}} \cdot (b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n) \\ x_2 = \dfrac{1}{a_{22}} \cdot (b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n) \\ \vdots \\ \vdots \\ x_n = \dfrac{1}{a_{nn}} \cdot (b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1}) \end{cases}$$

Seja $X = CX + d$:

$$C = \begin{bmatrix} 0 & -\dfrac{a_{12}}{a_{11}} & -\dfrac{a_{13}}{a_{11}} & \cdots & \dfrac{a_{1n}}{a_{11}} \\ -\dfrac{a_{21}}{a_{22}} & 0 & -\dfrac{a_{23}}{a_{22}} & \cdots & \dfrac{a_{2n}}{a_{22}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\dfrac{a_{n1}}{a_{nn}} & -\dfrac{a_{n2}}{a_{nn}} & -\dfrac{a_{n3}}{a_{nn}} & \cdots & 0 \end{bmatrix} \quad e \quad d = \begin{bmatrix} \dfrac{b_1}{a_{11}} \\ \dfrac{b_2}{a_{22}} \\ \vdots \\ \dfrac{b_n}{a_{nn}} \end{bmatrix}$$

O método de Gauss-Jacobi consiste em, dado $x^{(0)}$, aproximação inicial, obter $x^{(1)} \cdots, x^{(k)} \cdots$, através da relação recursiva $x^{(k+1)} = Cx^{(k)} + d$ (MELO; SANTOS, 2013)

$$\begin{cases} x_1^{(k+1)} = \dfrac{1}{a_{11}} \cdot (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)}) \\ x_2^{(k+1)} = \dfrac{1}{a_{22}} \cdot (b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)}) \\ \vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ x_n^{(k+1)} = \dfrac{1}{a_{nn}} \cdot (b_n - a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \cdots - a_{n,n-1}x_{n-1}^{(k)}) \end{cases}$$

Seja o sistema linear Ax $= b$ e seja $\alpha_k = \dfrac{\left(\sum_{j=1;j\neq k}^{n} |a_{kj}|\right)}{|a_{kk}|}$. Se $\alpha = \max_k < 1$ com $1 \leq k \leq n$, então o método de Gauss-Jacobi gera uma sequência $x^{(k)}$ convergente para a solução do sistema doda, independentemente da escolha da aproximação inicial (MELO; SANTOS, 2013):

$$\alpha_k = \sum_{\substack{j=1 \\ j\neq k}}^{n} \frac{|a_{kj}|}{a_{kk}}, i = 1, \ldots, n \quad \text{se } \alpha_{\max} < 1 \text{ então o sistema é convergente.}$$

1.2.2 Método Iterativo de Gauss-Seidel

O método de Gauss-Seidel é um método aperfeiçoamento Gauss-Jacobi, como forma de diminuir o tempo de resolução. O método de Gauss-Seidel é um método aperfeiçoamento Gauss-Jacobi, como forma de diminuir o tempo de resolução.

Utilizando-se da forma algébrica, o método de Gauss-Seidel transforma o sistema linear $Ax = b$ na forma equivalente $x = Cx + d$ por separação da diagonal. Este método se assemelha ao método Gauss-Jacobi (MELO; SANTOS, 2013).

O processo iterativo consiste em, sendo $x^{(0)}$ uma aproximação inicial, calcular $x^{(1)}, x^{(2)}, \cdots x^{(k)}, \cdots$ por:

$$\begin{cases} x_1^{(k+1)} = \dfrac{1}{a_{11}} \cdot \left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots a_{1n}x_n^{(k)}\right) \\ x_2^{(k+1)} = \dfrac{1}{a_{22}} \cdot \left(b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} - \cdots a_{2n}x_n^{(k)}\right) \\ x_3^{(k+1)} = \dfrac{1}{a_{33}} \cdot \left(b_3 - a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)} - \cdots a_{3n}x_n^{(k)}\right) \\ \vdots \\ x_n^{(k+1)} = \dfrac{1}{a_{nn}} \cdot \left(b_3 - a_{n1}x_1^{(k+1)} - a_{n2}x_2^{(k+1)} - \cdots a_{n,n-1}x_{n-1}^{(k+1)}\right) \end{cases}$$

Portanto, no processo iterativo de Gauss-Seidel, no momento de se calcular $x_j^{(k+1)}$ usando todos os valores $x_1^{(k+1)}, \cdots, x_{j-1}^{(k+1)}$ que já foram calculados e os valores $x_{j+1}^{(k)}, \cdots, x_n^{(k)}$ restante.

O Critério das linhas estudado no método de Gauss-Jacobi pode ser aplicado no estudo da convergência do método de Gauss-Seidel:

$$\beta_k = \frac{\sum_{j=1}^{i-1} \beta_j \left|a_{kj}\right| + \sum_{j=i+1}^{n} \left|a_{kj}\right|}{\left|a_{kk}\right|}, i = 1, \ldots, n \quad \text{se } \beta_{\max} \leq 1 \text{ então o sistema é convergente.}$$

1.2.3 Método Iterativo SOR

Métodos SOR, ou relaxação sucessíva, é uma técnica de aceleração da convergência dos métodos iterativos. Neste caso, define-se a aproximação na iteração $k+1$ como uma média entre o valor de x_k obtido na iteração k e o valor de x_{k+1} que seria obtido pelo método de Gauss-Seidel (GS). As iterações são definidas da seguinte forma:

$$x_{SOR}^{(k+1)} = (1-\omega)x_{SOR}^{k} + \omega x_{GS}^{k+1},$$

sendo que SOR e GS referem-se as aproximações obtidas pelos métodos SOR e Gauss-Seidel, respectivamente, e ω é o parâmetro de sobre-relaxação.

Quando $\omega = 1$ temos o método de Gauss-Seidel. Quando $1 < \omega < 2$ temos os métodos de sobre-relaxação, e quando $0 < \omega < 1$ temos os métodos de sub-relaxação. Em alguns casos particulares é possível estabelecer um valor ótimo para ω. Por exemplo, se A é uma matriz bloco tridiagonal e ρ é o maior autovalor da matriz de iteração do Método de Gauss-Jacobi, tem-se que:

$$\omega = \frac{2}{\sqrt{1 + (1 - \rho^2)}}.$$

Em relação à convergência do Método SOR, o método só converge se $0 < \omega < 2$. Os Métodos de Gauss-Jacobi e Gauss-Seidel convergem se A é uma matriz definida positiva e, seja A é simétrica e definida positiva, o método SOR converge para qualquer $0 < \omega < 2$.

1.2.4 Método do Gradiente

No caso do método do gradiente (ou declive máximo - steepest descent), a direção de $r^{(n)}$ é (MÉTODO... 2022)

$$r^{(n)} = -f(x^{(n)}) = b - Ax^{(n)} \ ,$$

que neste caso de sistemas lineares é também designado por resíduo.

Resta encontrar o valor an que minimiza f, de entre os possíveis valores $x^{(n)} + \alpha r^{(n)}$. Encontramos o ponto de mínimo, derivando f (nesses pontos) em ordem a α, ou seja, (MÉTODO... 2022)

$$\begin{aligned} \frac{d}{d\alpha} f(x^{(n)} + \alpha r^{(n)}) &= f(x^{(n)} + \alpha r^{(n)}).r^{(n)} \\ &= (b - A(x^{(n)} + \alpha r^{(n)})) * r^{(n)} \end{aligned}$$

$$= (b - Ax^{(n)}).r^{(n)} - \alpha Ar^{(n)} * r^{(n)}$$
$$= r^{(n)} * r^{(n)} - \alpha Ar^{(n)} * r^{(n)}.$$

Assim, o valor mínimo an será obtido com o zero da derivada, $r^{(n)} * r^{(n)} - anAr^{(n)} * r^{(n)}) = 0 \Leftrightarrow an = r^{(n)} * \frac{r^{(n)}}{Ar^{(n)} * r^{(n)}}$ (MÉTODO... 2022).

Em conclusão, dado um vector inicial $x^{(0)}$, o método do gradiente resume-se à iteração

$$x^{(n+1)} = x^{(n)} + \frac{r^{(n)} * r^{(n)}}{*} r^{(n)}, com r^{(n)} = b - Ax^{(n)}.$$

Um critério de paragem consiste em exigir que $||r^{(n)}||^{(2)} = r^{(n)} * r^{(n)} < \epsilon$, com ϵ pequeno, notando que isso implica que $Ax^{(n)}$ é próximo de b (MÉTODO... 2022).

1.2.5 Método do Gradiente Conjugado

Uma vez que o Método dos Gradientes normalmente tem convergência lenta, o Método dos Gradientes Conjugados é usado como uma alternativa. Trata-se de métodos de primeira ordem que minimizam a mesma função quadrática em, no máximo, n iterações.

Da mesma forma como no Método dos Gradientes, para encontrar a solução do sistema $Ax = b$ utilizaremos o problema equivalente:

$$\text{Minimizar } \phi(x) = \frac{1}{2} x^t Ax - x^t b, x \in \Re^n,$$

em que a matriz A é simétrica e definida positiva.

Sendo

$$\nabla\phi(x) = Ax - b = 0 \text{ e } \nabla^2\phi(x) = A,$$

encontrar a solução do sistema linear $Ax = b$ é equivalente a encontrar o mínimo da função quadrática.

Seja um produto interno real (euclidiano ou canônico) de dois vetores x e y em $\Re^n$ como:

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i = y^t x = x^t y$$

Dada uma matriz A simétrica e definida positiva, definimos um produto interno induzido por A como:

$$\langle x, y \rangle_A = y^t Ax = x^t Ay$$

no qual possui as mesmas propriedades do produto interno usual.

Agora, considerando $\phi(x) = \frac{1}{2} x^t Ax - x^t b$, podemos reescrever ϕ como:

$$\phi(x) = \frac{1}{2} \langle x, x \rangle_A - \langle b, x \rangle.$$

Seja $A \in \Re^{n \times n}$ uma matriz definida positiva. Dizemos que os vetores $d^0, d^1, \ldots, d^k \in \Re^n - \{0\}$ são A - ortogonais ou $A-$ conjugados se

$$\left\langle d^i, d^j \right\rangle_A = \left(d^i\right)^t A d^j = 0,$$

para qualquer $i, j = 0, 1, \ldots, k, \text{com } i \neq k$.

Seja $A \in \Re^n \times n$ uma matriz definida positiva. Um conjunto qualquer de vetores A-ortogonais é linearmente independente.

Definindo n direções A - ortogonais com A definida positiva, a solução ótima pode ser escrita como combinação linear dessas n direções (dimensão de A) mais a direção de b.

Dado um ponto inicial x_0, para encontrar a solução da função ϕ quadrática, toma-se um conjunto de direções $\{d_0, d_1, \ldots, d_{k-1}\}$, em que a primeira direção é a mesma direção de descida obtida pela busca linear exata do método do Gradiente e as demais são $A-$ conjugadas entre si.

A primeira iteração ($k = 1$) do Método dos Gradientes Conjugados é igual a primeira iteração do Método do Gradiente:

- Calcula-se o resíduo: $r_0 = Ax_0 - b$;
- Define-se a direção: $p_1 = -r_0$
- O novo ponto é dado por:

$$x_1 = x_0 + \beta p_1, \text{ em que } \beta = q_1 = -\frac{r_0^t p_1}{\left(Ap_1\right)^t p_1} = \frac{r_0^t r_0}{\left(Ar_0\right)^t r_0}.$$

Portanto,

$$x_1 = x_0 - \frac{r_0^t r_0}{\left(Ar_0\right)^t r_0} r_0 \qquad (1)$$

Para cada iteração $k > 1$, toma-se uma direção p_k que seja conjugada à direção p_{k-1} : $(Ap_k)^t p_{k-1} = p_k^t A p_{k-1} = 0$. Considera-se também,

$$p_k = -r_{k-1} + \alpha_{k-1} p_{k-1} \qquad (2)$$

ou seja, p_k é uma combinação linear de r_{k-1} e p_{k-1} (o coeficiente de r_{k-1} não pode ser nulo e foi definido como -1).

Sabemos que

$$\begin{aligned} p_k^t A p_{k-1} = 0 &\Rightarrow \left(-r_{k-1} + \alpha_{k-1} p_{k-1}\right)^t A p_{k-1} = 0 \\ &\Rightarrow -r_{k-1}^t A p_{k-1} + \alpha_{k-1} p_{k-1}^t A p_{k-1} = 0 \end{aligned}$$

Portanto,

$$\alpha_{k-1} = \frac{r_{k-1}^t A p_{k-1}}{p_{k-1}^t A p_{k-1}}, \text{ para } k > 1. \qquad (3)$$

Calculada a direção p_k, precisamos definir o tamanho do passo q_k para encontrar o próximo ponto:

$$x_k = x_{k-1} + q_k p_k \qquad (4)$$

O tamanho do passo q_k é o minimizador da função ϕ na direção p_k :

$$q_k = -\frac{r_{k-1}^t p_k}{(Ap_k)^t p_k} \qquad (5)$$

Para o resíduo r_k temos que:

$$\begin{aligned} &r_k = Ax_k - b = A\left(x_{k-1} + q_k p_k\right) - b = Ax_{k-1} - b + q_k A p_k \\ &\Rightarrow r_k = r_{k-1} + q_k A p_k. \end{aligned} \qquad (6)$$

As Equações (1)-(6) compõem o Método dos Gradientes Conjugados. Vamos usar propriedades de r_k para simplificar os cálculos de α_k e q_k.

Os resíduos das iterações k e $k-1$ são ortogonais:

$$r_k^t r_{k-1} = 0.$$

Na iteração k, o resíduo e a direção são ortogonais:

$$r_k^t p_k = 0.$$

O resíduo na iteração k é ortogonal à direção da iteração $k-1$:

$$r_k^t p_{k-1} = 0.$$

Assim, temos que:

$$\begin{aligned} -r_{k-1}^t p_k &= -r_{k-1}^t \left(-r_{k-1} + \alpha_{k-1} p_{k-1}\right)^t = \\ &= r_{k-1}^t - r_{k-1} - \alpha_{k-1} r_{k-1}^t p_{k-1} = r_{k-1}^t r_{k-1} \end{aligned}$$

Substituindo na Equação (5), temos:

$$q_k = -\frac{r_{k-1}^t r_{k-1}}{\left(A p_k\right)^t p_k}$$

Da Equação (6) temos:

$$r_k = r_{k-1} + q_k A p_k.$$

E podemos escrever:

$$A p_{k-1} = \frac{1}{q_{k-1}} \left(r_{k-1} - r_{k-2}\right)$$

Usando as propriedades indicadas, temos que:

$$\begin{aligned} r_{k-1}^t A p_{k-1} &= r_{k-1}^t \left(\frac{1}{q_{k-1}} \left(r_{k-1} - r_{k-2}\right)\right) \\ \Rightarrow r_{k-1}^t A p_{k-1} &= \frac{1}{q_{k-1}} r_{k-1}^t r_{k-1}. \end{aligned}$$

E ainda:

$$p_{k-1}^t A p_{k-1} = p_{k-1}^t \left(\frac{1}{q_{k-1}} \left(r_{k-1} - r_{k-2} \right) \right)$$

$$\Rightarrow p_{k-1}^t A p_{k-1} = \frac{1}{q_{k-1}} p_{k-1}^t r_{k-1} - \frac{1}{q_{k-1}} p_{k-1}^t r_{k-2}$$

$$= -\frac{1}{q_{k-1}} p_{k-1}^t r_{k-2} = -\frac{1}{q_{k-1}} \left(-r_{k-2} + \alpha_{k-2} p_{k-2} \right)^t r_{k-2}$$

$$= -\frac{1}{q_{k-1}} \left(-r_{k-2}^t r_{k-2} + \alpha_{k-2} p_{k-2}^t r_{k-2} \right)$$

$$= \frac{1}{q_{k-1}} r_{k-2}^t r_{k-2} - \frac{\alpha_{k-2}}{q_{k-1}} p_{k-2}^t r_{k-2}$$

$$\Rightarrow p_{k-1}^t A p_{k-1} = \frac{1}{q_{k-1}} r_{k-2}^t r_{k-2}$$

Substituindo na Equação (3), temos:

$$\alpha_{k-1} = \frac{r_{k-1}^t A p_{k-1}}{p_{k-1}^t A p_{k-1}} = \frac{r_{k-1}^t r_{k-1}}{r_{k-2}^t r_{k-2}},$$

para $k > 1$

2 Objetivos

Implementar computacionalmente os métodos: Gauss-Seidel, Gauss-Jacobi, SOR, Gradiente e Gradiente Conjugado; fazer os três ordenamentos das matrizes vistos e comparando todos os casos em relação à densidade/esparsidade da matriz e quantidade de iterações.

3 Procedimento

3.1 Métodos iterativos

Os métodos iterativos (Método de Gauss-Seidel, Método de Gauss-Jacobi, Método do Gradiente e Método do Gradiente Conjugado) foram implementados utilizando o MATLAB ®. Trata-se de uma linguagem de programação que expressa diretamente matrizes e arranjos matemáticos, combinando um ambiente de desktop ajustado para análise iterativa e processos de design com uma linguagem de programação.

Em seguida, foram implementados os reordenamentos: Mínimo Grau, Método Forma Bloco Triangular e Método de Triangularização de Björck. A densidade/esparsidade da matriz com e sem reordenamento foram, então, analisados e comparados.

Por fim, um pré-condicionador foi implementado para o Método do Gradiente Conjugado e os resultados do método com e sem o uso de pré-condicionador foram comparados.

3.1.1 Método Iterativo de Gauss-Jacobi

Considere $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^{n\times 1}, x^{(0)} \in \mathbb{R}^{n\times 1}, \varepsilon > 0$.

1. Teste de convergência: verificar se $\alpha < 1$ utilizando o critério das linhas.

   Seja o sistema linear $Ax = b$ e seja

   $$\alpha_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{|a_{ij}|}{a_{ii}}, i = 1, \ldots, n.$$

   Se $\alpha = \max_{1\leq i\leq n} \alpha_i < 1$, então o método de Gaus- Jacobi gera uma sequência $\left\{x^{(k)}\right\}$ que converge para a solução do sistema dado, independente da escolha da aproximação inicial $x^{(0)}$.

   Observação: Se $\alpha > 1$ não podemos concluir que a sequência diverge. Deve-se tentar uma permutação de linhas de forma que a matriz satisfaça o critério das linhas. A convergência do método não depende da solução inicial.

2. Construção de C e g, se possível $(a_{ii} \neq 0 i = 1, \ldots, n)$

   - Para $i = 1, \ldots, n$
   - $g_i = b_i/a_{ii}$
   - Para $j = 1, \ldots, n$

     $$C_{ij} = \begin{cases} -a_{ij}/a_{ii}, & \text{se } i \neq j, i, j = 1, \ldots, n \\ 0, & \text{se } i = j \end{cases}$$

3. Determinação da sequência de aproximações:

   - Para $k = 0, 1, \ldots$
   - Para $i = 1, \ldots, n$
   - $x_i^{(k+1)} = \sum_{\substack{j=1 \\ j\neq i}}^{n} C_{ij} x_j^{(k)} + g_i, \quad i = 1, 2, \cdots, n.$

4. Teste de Parada: erro $= \dfrac{\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|_\infty}{\left\|\mathbf{x}^{(k+1)}\right\|_\infty} < \varepsilon.$

3.1.2 Método Iterativo de Gauss-Seidel

Considere $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^{n\times 1}, x^{(0)} \in \mathbb{R}^{n\times 1}, \varepsilon > 0$.

1. Teste de convergência: verificar se $\beta < 1$ utilizando o critério de Sassenfeld.

   Sejam o sistema linear $Ax = b$ e β_i definido pela seguinte fórmula de recorrência

   $$\beta_i = \frac{\sum_{j=1}^{i-1} \beta_j \left|a_{ij}\right| + \sum_{j=i+1}^{n} \left|a_{ij}\right|}{\left|a_{ii}\right|}, i = 1, \ldots, n$$

   Se $\beta = \max\limits_{1\leq i\leq n} \beta_i < 1$, então a sequência $\left\{x^{(k)}\right\}$ gerada pelo método iterativo de Gauss-Seidel converge para a solução do sistema dado.

2. Construção de C e g, se possível $(a_{ii} \neq 0 i = 1, \ldots, n)$

   - Para $i = 1, \ldots, n$
   - $g_i = b_i/a_{ii}$
   - Para $j = 1, \ldots, n$

   $$C_{ij} = \begin{cases} -a_{ij}/a_{ii}, & \text{se } i \neq j, i, j = 1, \ldots, n \\ 0, & \text{se } i = j \end{cases}$$

3. Determinação da sequência de aproximações:

   - Para $k = 0, 1, \ldots$
   - Para $i = 1, \ldots, n$

   $$x_i^{(k+1)} = \sum_{j=1}^{i-1} C_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} C_{ij} x_j^{(k)} + g_i, \quad i = 1, 2, \cdots, n.$$

4. Teste de Parada: erro $= \dfrac{\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|_\infty}{\left\|\mathbf{x}^{(k+1)}\right\|_\infty} < \varepsilon$.

3.1.3 Método Iterativo SOR

Considere $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^{n\times 1}, x^{(0)} \in \mathbb{R}^{n\times 1}, \omega, k_{\max}, \varepsilon > 0$.

- Para $k = 0, \ldots, k_{\max}$
- Para $i = 1, \ldots, n$

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right)$$

- Teste de Parada: erro $= \dfrac{\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|_\infty}{\left\|\mathbf{x}^{(k+1)}\right\|_\infty} < \varepsilon$.

3.1.4 Método do Gradiente

Dado o sistema linear $Ax = b, A \in \Re^{n\times n}$ simétrica definida positiva, $b \in \Re^n$, o ponto inicial x_0, a precisão $\epsilon > 0$ e o número máximo de iterações $k_{\max}$, o algoritmo do Método do Gradiente que encontra uma aproximação x_k da solução do sistema é dado por:

- P1: Faça $k \leftarrow 0$;
- P2: Faça $r_k \leftarrow b - Ax_k$
- P3: Se $\|r_k\| < \epsilon$, então x_k é solução e pare;
- P4: Faça $\alpha_k \leftarrow \dfrac{r_k^t r_k}{r_k^t A r_k}$
- P5: Atualize $x_{k+1} \leftarrow x_k + \alpha_k r_k$
- P6: Critério de parada: se $\dfrac{\|x_{k+1} - x_k\|}{\|x_{k+1}\|} < \epsilon$ então x_{k+1} é solução e pare;
- P7: Se $k > k_{\max}$ então escreva: "O método não encontrou solução após $k_{\max}$ iterações"e pare.
- P8: Faça $k \leftarrow k + 1$ e vá para P2.

3.1.5 Método do Gradiente Conjugado

Dado o sistema linear $Ax = b, A \in \Re^{n\times n}$ simétrica definida positiva, $b \in \Re^n$, o ponto inicial x_0 e a precisão $\epsilon > 0$ e o número máximo de iterações $k_{\max}$, o algoritmo do Método do Gradiente Conjugado que encontra uma aproximação x_k da solução do sistema é dado por:

- P1: Faça $r_0 \leftarrow Ax_0 - b$
- P2: Se $\|r_0\| < \epsilon$, então x_0 é solução e pare;
- P3: Faça $p_1 \leftarrow -r_0$ e $q_1 = -\dfrac{r_0^t r_0}{(Ar_0)^t r_0}$;
- P4: Atualize $x_1 \leftarrow x_0 + q_1 p_1$
- P5: Critério de parada: se $\dfrac{\|x_1 - x_0\|}{\|x_1\|} < \epsilon$, então x_1 é solução e pare;
- P6: Faça $r_1 \leftarrow r_0 + q_1 A p_1$ e $k \leftarrow 2$
- P7: Enquanto $k \leq k_{\max}$, faça:
- P8: Se $\|r_{k-1}\| < \epsilon$, então x_{k-1} é solução e pare;
- P9: Faça $\alpha_{k-1} \leftarrow \dfrac{r_{k-1}^t r_{k-1}}{r_{k-2}^t r_{k-2}}$
- P10: Faça $p_k \leftarrow -r_{k-1} + \alpha_{k-1} p_{k-1}$;
- P11: Faça $q_k \leftarrow -\dfrac{r_{k-1}^t r_{k-1}}{(Ap_k)^t p_k}$
- P12: Atualize $x_k \leftarrow x_{k-1} + q_k p_k$;
- P13: Critério de parada: se $\dfrac{\|x_{k+1} - x_k\|}{\|x_{k+1}\|} < \epsilon$ então x_{k-1} é solução e pare;
- P14: Faça $r_k \leftarrow r_{k-1} + q_k A p_k$ e $k \leftarrow k + 1$;
- P15: Escreva: "O método não encontrou solução após $k_{\max}$ iterações"e pare.

3.2 Matriz de Hilbert

Uma função que receba um inteiro n e que retorne uma matriz de Hilbert de ordem n foi implementada e testados para todos os programas implementados utilizando a matriz de Hilbert para resolver o sistema linear com

$$a_{ij} = \frac{1}{i + j - 1} \qquad b = \frac{1}{3} \sum_{j=1}^{n} a_{ij}$$

Foram considerados $\epsilon = 10^{-5}$, $\epsilon = 10^{-10}$ e $\epsilon = 10^{-14}$, cujos resultados foram analisados e comparados.

A função *hilb(n)* do MATLAB ® foi utlizada para comparação da matriz de Hilbert.

4 Resultados e discussão

4.1 Matriz de Hilbert

Implementados os cálculos para as matrizes de Hilbert, os resultados foram organizados de acordo com o método implementados. A partir da escolha da precisão ϵ, foram comparados os resultados para o código do *script* e a função hilb(n) do MATLAB ®. Para este estudo, foi escolhida a ordem 12 para a matriz de Hilbert, já que esta é a maior matriz definida positiva possível e que tem determinante diferente de zero. Foi analisado o tempo computacional, o número de iterações da resolução do sistema, ou seja, quantas iterações foram necessárias para obter a solução aproximada, e a precisão da resolução, a partir do erro relativo da última iteração (ϵ).

Utilizando o vetor $x^{(0)}$ nulo, o método de Gauss-Jacobi gerou uma sequência $x^{(k)}$ que converge para a solução do sistema dado, independente da escolha da aproximação inicial $x^{(0)}$, tal que $\alpha = 0.9545$ pelo critério de linhas. A partir de $\epsilon = 10^{-15}$ o tempo computacional aumentava abruptamente. Assim, os dados da Tabela 1 foram analisados para $\epsilon = 10^{-5}$, $\epsilon = 10^{-10}$ e $\epsilon = 10^{-14}$:

Tabela 1: Implementação do método Gauss-Jacobi.

	Código implementado			Função *hilb(n)*		
ϵ	Tempo computacional (s)	Número de iterações	Erro relativo final	Tempo computacional (s)	Número de iterações	Erro relativo final
10^{-5}	5.185480e-02	260	9.9720e-06	4.952940e-02	260	9.9720e-06
10^{-10}	5.170990e-02	508	9.7351e-11	4.222620e-02	508	9.7351e-11
10^{-14}	4.368910e-02	706	9.6304e-15	4.215760e-02	706	9.6304e-15

Podemos notar que a aplicação da função *hilb(n)* foi mais eficiente que a implementação do código, apresentando maiores diferenças de tempo computacional para $\epsilon = 10^{-5}$ e $\epsilon = 10^{-10}$. No entanto, não é tão significativa para $\epsilon = 10^{-14}$, tendo em vista que essa diferença é de, aproximadamente, 3,5%. Os dados da tabela também indicam que, para maiores precisões, o número de iterações aumenta consideravelmente, entretanto o tempo computacional apresenta uma relação inversamente proporcional a este fator.

Implementando o código e a função *hilb(n)* para o método Gauss-Seidel, foi constatado que, para matriz de Hilbert de ordem 12, pelo critério de Critério de Sassenfeld, $\beta = 8.7895$, ou seja, a convergência do método não depende da solução inicial e não poderíamos concluir que a sequência diverge. No entanto, isto não acontece somente para a matriz desta ordem, mas também para todas as matrizes cuja ordem fosse diferente de 2 ($\beta = 0.5$). Ainda sim, o teste falhou para esta matriz,

pois o tempo computacional. Possivelmente, o fato de a matriz de Hilbert ser mal condiocionada, isto é, está próxima de ser singular, não permite que o sistema estabelecido encontre uma solução razoavelmente precisa. Além do mais, os próprios métodos iterativos fornecem uma sequência de soluções aproximadas de modo que cada solução aproximada é obtida da anterior pela aplicação de um mesmo procedimento.

Analisamos também os métodos SOR (ou relaxação sucessiva), uma técnica de aceleração da convergência dos métodos iterativos. Considerando o parâmetro de sobre-relaxação ω, tal que o método só converge se $0 < \omega < 2$, foi escolhido $\omega = 0.9$ para este estudo. Conforme as aplicações testes do algoritmo, para $\omega > 1$, o algoritmo exigiu um número muito maior de iterações e o tempo computacional é maior em comparação a $\omega < 1$. Para $0 < \omega < 0.9$, progressivamente, o número de iterações vai diminuindo, assim como o tempo computacional, embora de modo mais contido. Para $0.9 < \omega \geq 1$, há um crescimento do número de interações e do tempo computacional. Deste modo, a aplicação do algortimo para este método foi feita também para $\epsilon = 10^{-5}$, $\epsilon = 10^{-10}$ e $\epsilon = 10^{-14}$, segundo a tabela abaixo:

Tabela 2: Implementação dos métodos de SOR.

	Código implementado			Função *hilb(n)*		
ϵ	Tempo computacional (s)	Número de iterações	Erro relativo final	Tempo computacional (s)	Número de iterações	Erro relativo final
10^{-5}	3.107650e-02	17	6.6020e-06	2.462160e-02	17	6.6020e-06
10^{-10}	2.401450e-02	32	9.2455e-11	2.478720e-02	32	9.2455e-11
10^{-14}	2.193990e-02	45	5.6968e-15	2.321940e-02	45	5.6968e-15

Os dados mostram que este método apresenta um comportamento igual à aplicação do método Gauss-Jacobi: a função *hilb(n)* foi mais eficiente que a implementação do código, apresentando maiores diferenças de tempo computacional para $\epsilon = 10^{-5}$ e $\epsilon = 10^{-10}$, apesar de ter ocorrido um aumento no tempo computacional para $\epsilon = 10^{-14}$, mas o aumento não se mostrou significativo, já que essa diferença é de, aproximadamente, 5,8%; do mesmo o número de iterações aumenta consideravelmente com o aumento das precisões e o tempo computacional apresenta uma relação inversamente proporcional a este fator. Não obstante, o método SOR necessitou de menos iterações, com $\omega = 0.9$, para obter uma solução, tal que a mais expressiva mudança neste número é para $\epsilon = 10^{-10}$ (93,7%). O tempo computacional também apresentou uma considerável diminuição, mas um comportamento diferente, pois, para Gauss-Jacobi, entre $\epsilon = 10^{-5}$ e $\epsilon = 10^{-10}$, a diferença não foi significativa, enquanto, em métodos de SOR ocorreu entre $\epsilon = 10^{-10}$ e $\epsilon = 10^{-14}$, ou seja, a

mudança do tempo computacional, nestes métodos, é mais relevante para menores precisões.

Foram também feitas implementações para os métodos de gradiente (Gradiente e Gradiente Conjugado), cujos resultados estão dispostos nas tabelas 3 e 4:

Tabela 3: Implementação do método Gradiente.

	Código implementado			Função *hilb(n)*		
ϵ	Tempo computacional (s)	Número de iterações	Erro relativo final	Tempo computacional (s)	Número de iterações	Erro relativo final
10^{-5}	1.500890e-02	753	9.9514e-06	1.452690e-02	753	9.9514e-06
10^{-8}	4.408470e-01	120487	9.9998e-09	4.409370e-01	120487	9.9998e-09
10^{-10}	7.303357e+00	2658423	1.0000e-10	7.264142e+00	2658423	1.0000e-10

Tabela 4: Implementação do método Gradiente Conjugado.

	Código implementado			Função *hilb(n)*		
ϵ	Tempo computacional (s)	Número de iterações	Erro relativo final	Tempo computacional (s)	Número de iterações	Erro relativo final
10^{-4}	5.392510e-02	5039	9.9986e-05	5.134290e-02	5039	9.9986e-05
10^{-5}	3.831881e-01	50121	9.9999e-06	3.870070e-01	50121	9.9999e-06
10^{-6}	2.790727e+00	500380	1.0000e-06	3.004729e+00	500380	1.0000e-06

Para estes métodos, observamos novamente o mesmo comportamento vistos anteriormente nos métodos Gauss-Jacobi e de SOR. O que se destaca aos métodos de gradiente é que não demonstraram melhora, nem eficiência quando comparados aos métodos antes analisados. Poder-se-ia considerar uma melhor no tempo computacional para as menores precisões, mas não mostram ser significativas as diferenças temporais, principlamente se levado em conta os altos números de iterações a partir dos métodos de gradiente. Ainda, pelo método Gradiente, depois de $\epsilon = 10^{-10}$, o tempo computacional cresce abruptamente, como para o método Gradiente Conjugado, a partir de $\epsilon = 10^{-7}$. Em outras palavras, estes métodos foram ineficientes para aplicação da matriz de Hilbert nos algoritmos implementados, além de, quanto maior for a precisão, menos favorável é a implementação deles. De fato, método Gradiente tem essa desvantagem de, normalmente, ter convergência lenta. Por isso, usar o método de Gradiente Conjugados é uma alternativa, pois este métodos é mais rápido de primeira ordem e minimizama mesma função quadrática em, no máximo, n iterações.

4.2 Aplicação do métodos de reordenamento

Os próximos cálculos também foram operados com os métodos anteriormente utilizados, mas, desta vez, foram acrescentados cálculos com reordenamentos aplicados também. Foi utilizada uma matriz extraída do site https://math.nist.gov/MatrixMarket/, que fornece acesso a um repositório de dados de teste para uso em estudos comparativos de algoritmos para álgebra linear numérica. A matriz escolhida foi bcsstk22.mtx, real simétrica indefinida, de ordem 138 e com 417 entradas (disponível em: https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/bcsstruc3/bcsstk22.html). Para os resultados à mostra, foi estipulado um vetor nulo $x^{(0)}$, uma precisão $\epsilon = 10^{-5}$ e número de iterações máximo $k_{max} = 10^6$.

Através dos dados das tabelas 5 e 6, podemos analisar e comparar as implemetações dos métodos Gauss-Jacobi e Gauss-Seidel:

Tabela 5: Implementação do método Gauss-Jacobi.

Método de reordenamento	Tempo Computacional (s)	Esparsidade	Grau de esparsidade	Número de iterações
Sem reordenamento	1.753561e+02	174405663	9.99996e-01	O método não encontrou solução após k_{max} iterações
Mínimo Grau	Não podemos concluir que a sequência diverge			
Bloco Triangular	Não podemos concluir que a sequência diverge			
Triangularização de Björck	Não podemos concluir que a sequência diverge			

Tabela 6: Implementação do método Gauss-Seidel.

Método de reordenamento	Tempo Computacional (s)	Esparsidade	Grau de esparsidade	Número de iterações	Erro relativo final
Sem reordenamento	1.271179e+00	174405663	9.99996e-01	5	9.4507e-07
Mínimo Grau	Não podemos concluir que a sequência diverge				
Bloco Triangular	Não podemos concluir que a sequência diverge				
Triangularização de Björck	Teste falha				

Em ambos os métodos, não foi possível implementar os métodos de reordenamento, uma vez que a convergência do método não depende da solução inicial, tanto para um vetor nulo ou um vetor aleatório, de ordem *n x 1* (sendo n o número de linhas da matriz A), $x^{(0)}$, exceto para a aplicação do método de reordenamento da Triangularização de Björck na implementação do método de Gauss-Seidel. Neste último caso, o método de Gauss-Seidel gera uma sequência $x^(k)$ que converge para a solução do sistema dado, independente da escolha da aproximação inicial $x^{(0)}$,

mas a implementação falhou na construão da matriz iterativa C_{nxn}.

Apesar destas irresolução, em ambos os métodos, a matriz A convergiu ($\alpha = 0.50$ e $\beta = 0.25$ para o critério das linhas e para o critério de Sassenfeld, respectivamente) e o método Gauss-Seidel mostrou-se mais eficiente. Primeiramente, o método de Gauss-Jacobi não conseguiu encontrar uma solução aproximação até k_{max} e, enquanto o método de Gauss-Seidel resolveu o sistema em, aproximadamente, 1 segundo, a indeterminação do sistema levou cerca de 3 minutos. Além do baixo tempo de operação do algoritmo, o método de Gauss-Seidel precisou de apenas 5 iterações para a resolução do sistema, o que aponta uma vantagem dos métodos iterativos em relações aos métodos diretos que é rapidez em resolver os sistemas lineares.

A resolução dos métodos de SOR apresentaram o problema em relação aos métodos de reordenamento. Deste modo, o estudo destes métodos voltaram-se para uma comparação dos resutados comparados entre parâmetros de sobre-relaxação ω, conforme a tabela abaixo:

Tabela 7: Implementação dos métodos de SOR.

ω	Tempo Computacional (s)	Esparsidade	Grau de esparsidade	Número de iterações	Erro relativo final
0.1	2.241513e+02	174405663	9.99996e-01	89	9.6059e-06
0.3	8.091746e+01	174405663	9.99996e-01	32	9.8236e-06
0.5	4.312227e+01	174405663	9.99996e-01	17	8.1982e-06
0.9	2.272580e+01	174405663	9.99996e-01	8	2.1242e-06
1	1.557519e+01	174405663	9.99996e-01	6	5.3764e-07
1.25	2.039866e+01	174405663	9.99996e-01	10	9.3315e-06
1.5	1.557519e+01	174405663	9.99996e-01	8	2.1242e-06
1.9	5.722372e+01	174405663	9.99996e-01	20	6.3524e-06

Sabemos que, em relação à convergência do Método SOR, o método só converge se $0 < \omega < 2$. Por isso, este foi o intervalo escolhido para a análise dos resultados. Além disso, se a matriz for simétrica e definida positiva, então o método converge para qualquer $0 < \omega < 2$, o que não representa este estudo, tendo em vista que a matriz é real simétrica indefinida e, por isso, a matriz após os métodos de reordenamento não convergiram (ao menos, não pudemos concluir se a sequência diverge).

Nota-se que, para os métodos de sub-relaxação ($0 < \omega < 1$), os resultados configuram uma

função decrescente para o tempo computacional (apesar de um "ponto fora da curva", para $\omega = 0.3$), número de iterações e erro relativo final, enquanto, para os métodos de sobre-relaxação ($1 < \omega < 2$), a função que a caracterizaria seria crescente, tal que $\omega = 1$ seria o ponto mínimo desta função, onde há o menor tempo computacional, menor número de iteração e menor erro relativo (maior precisão). Assim, para $\omega = 1$, temos o método de Gauss-Seidel e é apresentado os melhores resultados para a matriz sem reordenamento. Isto confirma o que foi posto previamente pela comparação dos resultados das tabelas 5 e 6, dos métodos de Gauss-Jacobi e de Gauss-Seidel, respectivamente.

O método Gradiente, sob as mesmas condições dos demais cálculos, foi testado, mas apresentou falha por não encontrar solução após k_{max} iterações, com os reordenamentos e sem nenhum deles também. Além disso, outras condições e outras matrizes foram postas para testar e obter resultados aproximados, mas todos falharam. Isto ocorre, certamente, porque a escolha do método a utilizar depende do problema concreto que se pretende resolver. Além disso, os erros de cálculo podem ocorrer ao escolhermos um intervalo de tempo muito grande, destacam-se outra desvantagem dos métodos iterativos que é a dificuldade de se estimar o tempo de convergência do sistema. Isto faz com que, muitas vezes, este método alcance convergência em um número elevado de iterações, sendo que isto torna o método ineficiente.

Assim sendo, podemos usar o método alternativo, o método dos Gradientes Conjugados. As implementações do algoritmo com e sem os reordenamentos e pré-condicionamento ocorreram com sucesso e os resultados estão dispostos na Tabela 8:

Tabela 8: Implementação do método Gradiente Conjugado.

Método de Reordenamento	Tempo computacional (s)	Esparsidade	Grau de esparsidade	Número de iterações	Erro relativo final
Sem reordenaento	5.514700e-02	174405663	9.99996e-01	3	4.1676e-08
Mínimo Grau	5.143410e-02	174405663	9.99996e-01	3	3.6504e-09
Bloco Triangular	5.198790e-02	174405663	9.99996e-01	3	1.5902e-09
Triangularização de Björck	5.307240e-02	174405663	9.99996e-01	3	2.4956e-08
Pré-condionamento	2.352001e-01	174405663	9.99996e-01	4	8.4727e-11

Destaca-se por estes resultados que o método apresentou uma estabilidade no tempo computacional e no número de iterações. A aplicação dos reordenamento é medida, neste caso, pelos erros relativos, ou seja, pelo precisão de cada operação. Apesar de não ter o menor tempo computacional,

a implementação com o método de reordenamento de Bloco Triangular apresentou, dentre esses métodos, a melhor precisão. Em comparação, a precisão teve uma melhora de 61,84%. Sob esta perspectiva, a aplicação dos métodos de reordenamento não traria menores gastos computacionais, mas teríamos mais precisão na solução a partir dos métodos iterativos.

O pré-condicionamento trouxe os melhores resultados de todo estudo. Além da redução de mais de 57% do tempo computacional, o aumento do número de iterações não foi significativo e a precisão aumento consideravelmente, tal que ela é bem menor do que a estipulada previamente ao cálculo do algoritmo lançado no programa MATLAB ®. Em comparação à precisão do método de reordenamento de Bloco Triangular, a melhora é de quase 95%.

5 Conclusão

Os métodos iterativos foram implementados computacionalmente e foram feitos os reordenamentos de matriz, comparando todos os casos em relação ao número de iterações, tempo computacional e quantidade de iterações. O método de Gradiente Conjugado apresentou os melhores resultado, já que, em todos os cálculos, todos as matrizes convergiram e apresentaram uma solução aproximada rapidamente e com erros relativos pequenos. O pré-condicionamento apresentou as principais e mais relevantes melhoras na implementação do algoritmo. Os métodos iterativos mostraram-se, de fato, mais rápidos do que os métodos diretos, inclusive necessitaram de menos iterações para a obtenção de soluções aproximadamente, mas ainda há desvantagens na utilização deles, dentre elas: a dependência do problema e da estrutura da matriz pode levar há não convergir a matriz e à lentidão da resolução do sistema; deste modo, a precisão nem sempre é tão confiável, tendo em vista que o número de iterações máximas nem sempre é suficiente para a resolução, que fica muito suscetível a erros.

Comparando os métodos Gauss-Jacobi e Gauss-Seidel, ambas implementações demonstraram que os reordenamentos não foram aplicações cabíveis, entretando os cálculos para matrizes sem reordenamento mostram que o método de Gauss-Seidel tem resultados mais precisos e menores gastos computacionais, o que foi confirmado pela análise dos métodos de SOR. O mesmo ocorre entre os métodos Gradiente e Gradiente Conjugado, onde, na análise das matrizes de Hilbert, o método de Gradiente Conjugado foi mais eficiente e o método de Gradiente apresentou falhas ao aplicar matrizes do banco de dados utilizado.

6 Referências

MELO, Edelson dos Santos; SANTOS, Paulo Izidio dos. MÉTODOS NUMÉRICOS: APLICAÇÕES EM SISTEMAS LINEARES E APROXIMAÇÕES DE FUNÇÕES. 2013. 73 f. TCC (Graduação) - Curso de Matemática, Universidade Federal do Amapá, Macapá, 2013.

MÉTODO do Gradiente e Gradientes Conjugados. Disponível em: https://www.math.tecnico.ulisboa.pt/ calves/AN/Eng2004/GradConj.html. Acesso em: 07 out. 2022.

PULINO, Petronio. **Álgebra Linear e suas Aplicações**. Campinas: Pulinus, 2012.