Estudo Comparativo entre a Implementação Sequencial e Paralela dos Métodos Gauss-Jacobi e Gauss-Seidel

Felipe G. Silva[1], **Iara C. R. Silva**[2], **Erikosn F. Morais**[1]

[1]Departamento de Ciência da Computação – Universidade Tecnológica Federal do Paraná
Ponta Grossa– PR – Brazil

[2]Departamento de Matemática – Universidade Tecnológica Federal do Paraná
Ponta Grossa– PR – Brazil

felipegimenezsilva@gmail.com,{iarasilva,emorais}@utfpr.edu.br

Abstract. *This paper aims to analyze and compare the response time difference of the Gauss-Seidel and Gauss-Jacobi numerical methods using sequential and parallel programming approaches in their implementations.Furthermore, a hybrid numerical method will be presented as a parallelization option of the Gauss-Seidel method for solving problems of linear systems that satisfy the line criterion. Tests were performed to show that there are cases where the sequential Gauss-Seidel method may be more efficient than the parallel execution of the Gauss-Jacobi method.*

Resumo. *Este artigo tem como objetivo analisar e comparar a diferença de tempo de resposta dos métodos numéricos conhecidos como Gauss-Seidel e Gauss-Jacobi, utilizando abordagens de programação sequencial e paralela em suas implementações. Além disso, será apresentado um método numérico híbrido como opção de paralelização do método Gauss-Seidel, proposto para a resolução de problemas de sistemas lineares que satisfazem o critério das linhas. Testes foram realizados para evidenciar que existem casos em que o método de Gauss-Seidel sequencial pode ser mais eficiente que a execução paralela do método de Gauss-Jacobi.*

1. Introdução

Muitos problemas reais como circuitos elétricos, problema de transportes e de redes e dentre outros, podem ser modelados por sistemas lineares. Os métodos para resolvê-los dividem-se em dois grupos: diretos e iterativos. Os métodos de Eliminação Gaussiana e Fatoração LU são diretos e, além disso, o número de passos depende da dimensão do problema. Já os métodos iterativos, como o Gauss-Jacobi e Gauss-Seidel, precisam de uma solução inicial (chamada também de chute inicial) para gerarem aproximações suficientemente próximas da solução do problema. No entanto, dependendo da estrutura da matriz dos coeficientes, esses métodos podem não convergir.

Os métodos diretos precisam da informação do sistema linear e dependendo da dimensão do problema, os algoritmos dessa classe ficam inviáveis de serem utilizados. Já os métodos iterativos permitem particionar o sistema de tal modo que os cálculos podem ser realizados em processadores diferentes, viabilizando o cálculo da solução aproximada.

Neste trabalho foram implementados os métodos Gauss-Jacobi e Gauss-Seidel paralelos. O objetivo dessa pesquisa é sugerir uma forma de implementação paralela

para o método de Gauss-Seidel e evidenciar casos onde a implementação sequencial dos métodos numéricos se demostra mais efetiva. Há diversas pesquisas de comparações de métodos numéricos paralelos utilizando outras abordagens, como o artigo feito por [De Paula 2013] utilizado CUDA, para essa pesquisa, utilizou-se a biblioteca para a programação multi-processo Open Multi-Processing.

2. Sistemas Lineares

Nesta sessão será apresentada as informações utilizadas para a geração das matrizes de teste, além das explicações de ocorrências básicas necessárias para o entendimento da pesquisa.

Um sistema linear é um conjunto de m equações lineares com n incógnitas [Hoffman and Kunze 2004]:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m \end{aligned} \tag{1}$$

onde $a_{ij} \in \mathbb{R}, i = 1, \ldots, m,\ j = 1, \ldots, n,\ b_i, i = 1, \ldots, m$ são constantes e $x_j, j = 1, \ldots, n$ são as incógnitas.

O sistema (1) pode ser reescrito na forma matricial:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \Leftrightarrow Ax = b \tag{2}$$

onde $A = [a_{ij}]$ é uma matriz de dimensão $(m \times n)$, $x = [x_i]$ é um vetor de dimensão $(n \times 1)$ e $b = [b_i]$ é um vetor de dimensão $(m \times 1)$.

Os sistemas lineares podem ser classificados como: possível e determinado, possível e indeterminado e impossível, se tiver uma única solução, infinitas soluções ou nenhuma solução, respectivamente [Hoffman and Kunze 2004].

Os métodos numéricos Gauss-Seidel e Gauss-Jacobi, são chamados de iterativos e possuem critérios que garantem a convergência dos mesmos.

2.1. Métodos iterativos

Segundo Paul Zimmermann [Brent and Zimmermann 2010], métodos iterativos são fundamentados na ideia de aproximações sucessivas. Dado um sistema linear $Ax = b$, onde A é uma matriz quadrada inversível, o método inicia-se com uma aproximação $x^{(0)}$ e de forma iterativa gera uma sequência de aproximações $x^{(0)}, x^{(1)}, \ldots, x^{(k)}$, tal que quando $k \to \infty$ tem-se que $x^{(k)} \to A^{-1}b = x^*$, ou seja, $x^{(k)}$ converge à solução do sistema.

Um método iterativo linear geral para a solução de $Ax = b$, pode ser escrito na forma matricial como:

$$x^{(k+1)} = Hx^{(k)} + C, \quad k = 0, 1, 2, \ldots \tag{3}$$

onde H é a matriz iteração e possui informações da matriz A e o vetor coluna C possui informações da matriz A e do vetor b. Para garantir que a sequência de vetores, construídos pelas iterações (3) , seja convergente para a solução do sistema, é suficiente que o sistema satisfaça algum critério de convergência [Gautschi 2011].

2.1.1. Critério de convergência

Os critérios de convergência, como o nome já diz, garantem que o método converge. Um dos critérios para verificar se a sequência gerada pela fórmula (3) converge é o critério das linhas:

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} \left|a_{ij}\right|, \quad \forall i = 0, 1, 2, \ldots, n. \tag{4}$$

Uma matriz estritamente diagonal dominante tem o critério (4) satisfeito. Essa condição é suficiente para a sequência definida pelo método convirja para a solução, porém não é necessária [Gautschi 2011], ou seja, existem sistemas em que a matriz dos coeficientes não satisfaz o critério (4) mas a sequência (3) converge para a solução.

Como nesse trabalho, o intuito é estudar a performace dos métodos iterativos Gauss Jacobi e Gauss Seidel paralelos, então foi criado um gerador de testes onde as matrizes dos coeficientes possuem o critério das linhas satisfeito.

2.1.2. Critério de parada

Quando se encontra uma aproximação aceitável para a solução do sistema, os métodos iterativos precisam parar, sendo assim, estima-se uma tolerância ε (um valor próximo de zero) tal que:

$$\frac{\left\|x^{(k+1)} - x^{(k)}\right\|_\infty}{\left\|x^{(k+1)}\right\|_\infty} \leq \varepsilon \tag{5}$$

ou

$$\left\|Ax^{(k+1)} - b\right\|_\infty \leq \varepsilon. \tag{6}$$

Com isso se define qual é o valor de erro que se pode obter com a aproximação. Quanto menor o erro máximo estabelecido, maior será a quantidade de iterações feitas pelos métodos para encontrar uma aproximação para a solução do sistema $Ax = b$. Os algoritmos implementados, nesse trabalho, utilizam o critério de parada definida por (6).

2.2. Testes gerados

Para criar um problema que convirja foi gerada uma matriz pseudo-aleatória satisfazendo o critério das linhas. Lembrando que, segundo Walter Gautschi [Gautschi 2011], caso a matriz não satisfaça um dos critérios de convergência, ainda sim o método poderá convergir.

Para que a matriz A satisfaça o critério das linhas, foi seguido os seguintes passos:

(1) $a_{ij} = R_1 \; \forall i, j$ tal que $i \neq j$. R_1 é um número real e pseudo- randômico qualquer;
(2) calcula-se os valores necessários para a diagonal principal da matriz, fazendo

$$a_{ii} = \left(\sum_{j=1, j \neq i}^{n} |a_{ij}| \right) \left| \frac{1}{R_2} \right|, \; \forall i \text{ e com } 0 < |R_2| < 1$$

onde R_2 é um número real e pseudo-randômico.

Com esses dois passos é garantida a convergência da matriz A para qualquer chute inicial $x^{(0)}$. Para a escolha do valor randomizado R_1, foi utilizado duas distribuições, a normal e a de Weibull [Howell and Rheinfurth 1982], para gerarem valores randômicos, pois dependendo da distribuição que se use para obter R_1, a quantidade de iterações necessárias para calcular as aproximações é afetada significativamente. A medida que R_2 se aproxima do valor um, os testes mostram um aumento na quantidade de iterações de cada método.

Ao utilizar a distribuição normal, não houve grandes alterações entre a quantidade de iterações necessárias para resolver os sistemas lineares com os métodos numéricos de Gauss-Seidel e o Gauss-Jacobi, conforme a figura 1. Porém, ao utilizar a distribuição de Weibull, houve um grande impacto, como mostrado na figura 2. A escolha da distribuição randômica e do valor de R_2 afeta diretamente a quantidade de iterações realizadas.

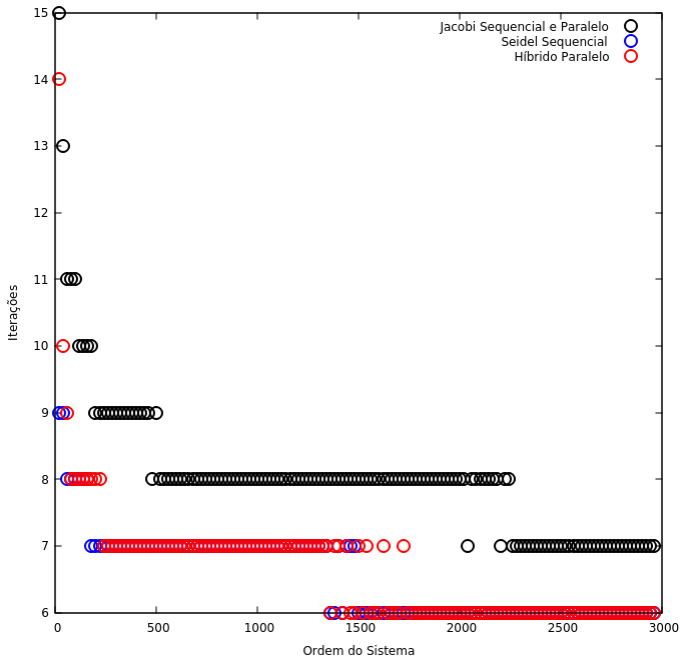

Figura 1. Influência da distribuição normal na quantidade de iterações.

2.3. Método iterativo Gauss-Jacobi

Seja $Ax = b$ um sistema linear. Assumindo que A é uma matriz quadrada de ordem n e que os elementos $a_{ii} \neq 0, i = 1, \ldots, n$, o método iterativo de Gauss-Jacobi pode ser

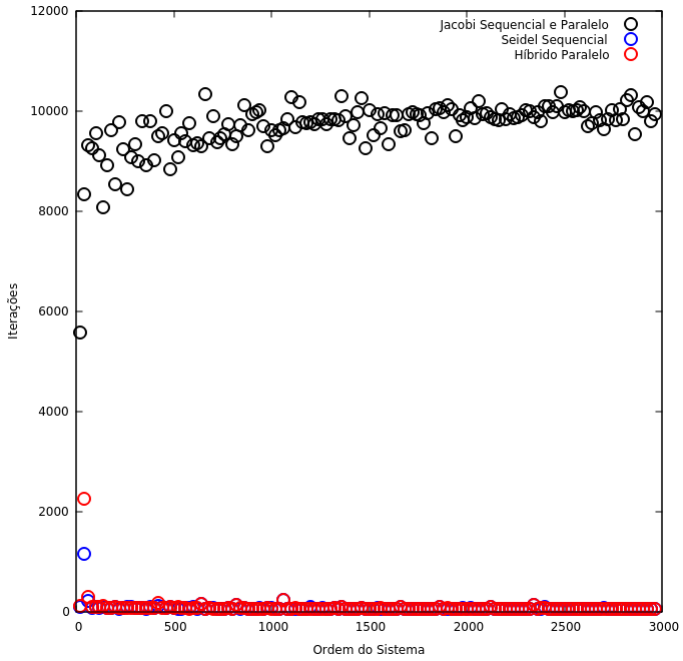

Figura 2. Influência da distribuição de Weibull na quantidade de iterações.

escrito como:

$$
\begin{aligned}
x_1^{(k+1)} &= (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} \ldots - a_{1n}x_n^{(k)})/a_{11}, \\
x_2^{(k+1)} &= (b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} \ldots - a_{2n}x_n^{(k)})/a_{22}, \\
&\vdots \\
x_n^{(k+1)} &= (b_n - a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} \ldots - a_{n,n-1}x_{n-1}^{(k)})/a_{nn}.
\end{aligned}
$$

A implementação algorítmica do método de Gauss-Jacobi, pode ser realizada com duas abordagens diferentes: a execução sequencial, como mostrado na figura 3, onde o programa realiza todas as tarefas em apenas um núcleo, e a execução em paralelo, no qual o algoritmo informa que certos trechos de código podem ocorrer em núcleos separados, ilustrado na figura 4.

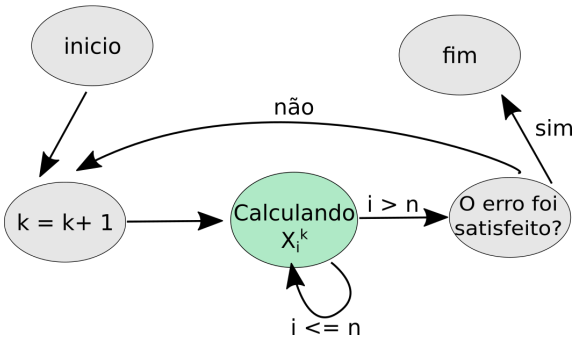

Figura 3. Lógica sequencial de métodos iterativos.

O Método de Gauss-Jacobi é um algoritmo naturalmente paralelizável. Isso indica que há formas de implementação paralela em que não há ocorrências de problemas

comuns de algoritmos paralelos, como a necessidade de uma ordem de processamento específica, ou pontos críticos, que devem ser executado por um núcleo por vez.

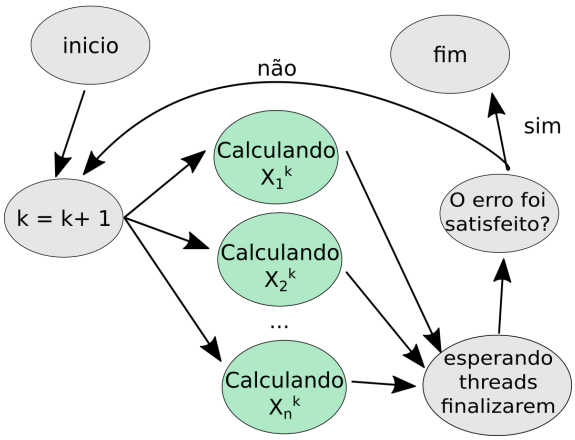

Figura 4. Lógica paralela natural de métodos iterativos

Com isso, para matrizes de uma certa ordem, o tempo de processamento do algoritmo é reduzido se executado paralelamente, mesmo que a complexidade do algoritmo permaneça a mesma.

Para que o uso da implementação paralela retorne num tempo de resposta melhor, deve ser considerado o tamanho da matriz, pois quanto maior, mais operações terão de ser feitas. Para matrizes pequenas, o paralelismo não se demostra eficiente, já que o sistema operacional gastará tempo gerenciando quais operações serão feitas e em quais núcleos, causando maior tempo de processamento até obter a aproximação final. Outro fator que afeta o desempenho em paralelo é a execução de outros programas ao mesmo tempo, já que haverá disputa pelos núcleos disponíveis.

2.4. Método iterativo Gauss-Seidel

Esse método é muito semelhante ao Gauss-Jacobi, porém ao se calcular $x_i^{(k+1)}$, usa-se todas as aproximações $x_j^{(k+1)},\ j < i$ calculadas na iteração corrente. Seja $Ax = b$ um sistema linear. Assumindo que A é uma matriz quadrada de ordem n e que seus elementos são $a_{ii} \neq 0, i = 1, \ldots, n$, o método iterativo de Gauss-Seidel pode ser escrito como:

$$
\begin{aligned}
x_1^{(k+1)} &= (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} \ldots - a_{1n}x_n^{(k)})/a_{11}, \\
x_2^{(k+1)} &= (b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} \ldots - a_{2n}x_n^{(k)})/a_{22}, \\
&\vdots \\
x_n^{(k+1)} &= (b_n - a_{n1}x_1^{(k+1)} - a_{n2}x_2^{(k)} \ldots - a_{n,n-1}x_{n-1}^{(k+1)})/a_{nn}.
\end{aligned}
$$

Comparando as versões sequenciais dos métodos de Gauss-Jacobi e Gauss-Seidel, esse último, na iteração corrente $k + 1$, já utiliza as coordenadas do vetor x calculadas nessa iteração. Essa mudança faz com que o método Gauss-Seidel retorne a solução com menos iterações que o Gauss-Jacobi [Khojasteh Salkuyeh 2007]. Porém, na perspectiva de programação paralela, a mudança cria dependências de acontecimentos, e assim o algoritmo deixa de ser naturalmente paralelizável, mas permite ser dividido em problemas menores para separar os trechos paralelos dos trechos sequenciais.

Um exemplo desse acontecimento pode ocorrer da seguinte forma: já que não se sabe ao certo qual conjunto de instruções do cálculo de uma determinada incógnita x_i^{k+1} terminará primeiro e supondo que a matriz seja de ordem 2, temos as seguintes possibilidades para cada iteração:

- $x_1^{(k+1)}$ e $x_2^{(k+1)}$ foram calculados com os valores não atualizados;
- $x_1^{(k+1)}$ foi calculado com a melhor aproximação enquanto $x_2^{(k+1)}$ foi calculado com valor não atualizado;
- $x_1^{(k+1)}$ foi calculado com o valor não atualizado enquanto x_2^{k+1} foi calculado com a melhor aproximação.

Esse problema não ocorre no Gauss-Jacobi, já que sempre ao realizar o processamento é utilizado o valor da iteração passada.

2.5. Método híbrido

O método de Gauss-Seidel paralelo, do mesmo modo que ocorre no método de Gauss-Jacobi, atualiza os valores das coordenadas de x de forma simultânea e não determinística. No método de Gauss-Jacobi, não há problemas na ordem das coordenadas do vetor $x^{(k+1)}$ que serão atualizadas, por exemplo, se forem atualizadas simultâneamente na $(k+1)$-ésima iteração as coordenadas $x_1^{(k+1)}$ e $x_3^{(k+1)}$, os valores utilizados para computar essas coordenadas estão armazenados no vetor x^k da iteração anterior. Usando o mesmo exemplo, mas para o método de Gauss-Seidel, a atualização de $x_1^{(k+1)}$ é feita como no método de Gauss-Jacobi, mas para atualizar $x_3^{(k+1)}$ é preciso do $x_2^{(k+1)}$ e não há certeza se essa coordenada já foi computada. Na figura 5, tem todas as possibilidades das atualizações dos coordenadas do vetor x para um sistema de ordem dois.

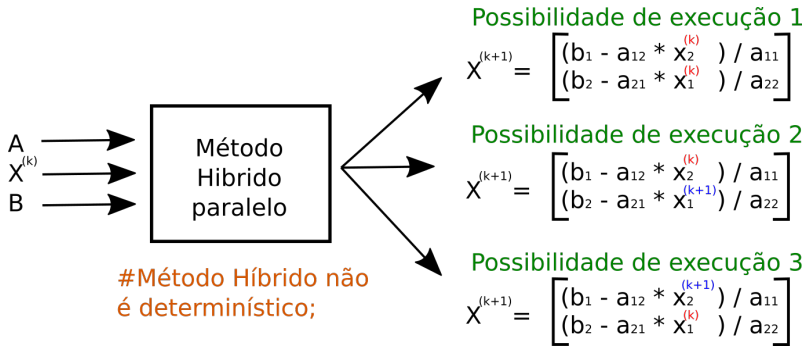

Figura 5. Possíveis respostas do método híbrido paralelo para um sistema de ordem 2.

Para contornar esse problema foi implementado o método híbrido paralelo, que obteve um melhor tempo de resposta para os casos avaliados comparando com o método de Gauss-Seidel paralelo. Como não é determinístico, o método pode encontrar uma aproximação satisfatória de diferentes formas, como exemplificado na figura 6.

O método paralelo implementado utiliza as propriedades das duas formas de resolução, porém com pequenas mudanças. O Gauss-Jacobi, sempre utiliza o valor não atualizado $x^{(k)}$, e por isso é naturalmente paralelo e com uma maior quantidade de iterações, enquanto o Gauss-Seidel, sempre utiliza o último valor de aproximação, $x^{(k+1)}$, diminuindo a quantidade de iterações e perdendo seu aspecto paralelizável.

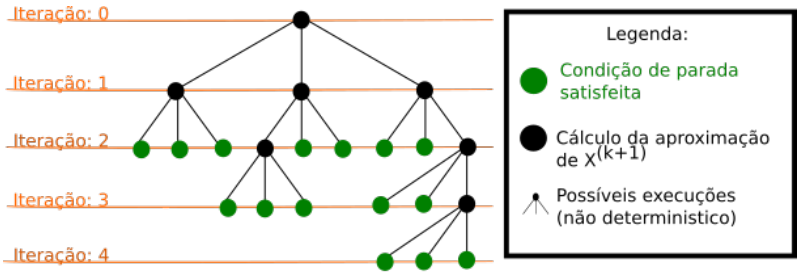

Figura 6. Exemplo de possibilidades de convergência do método híbrido para um sistema de ordem 2.

Ao paralelizar o método de Gauss-Seidel é perceptível que haverá tempos em que a informação mais atualizada $x^{(k+1)}$ não estará disponível, mas quando isso ocorre, ainda existe a informação do valor não atualizado $x^{(k)}$, que também pode ser utilizado, já que ambos os valores são uma aproximação de um determinado x_i.

Portanto, o algoritmo paralelo testado utiliza apenas o valor que possui no tempo do cálculo, quebrando assim a dependência de acontecimentos que ocorre com o Gauss-Seidel. Caso o algoritmo utilize sempre o valor desatualizado $x^{(k)}$, seu comportamento será igual ao Gauss-Jacobi e caso utilize sempre o valor atualizado $x^{(k+1)}$, o comportamento estará próximo ao Gauss-Seidel sequencial, porém não será igual já que não há uma ordem específica de linhas a ser processadas como no caso do Gauss-Seidel sequencial. Os testes realizados mostram que é comum que as aproximações troquem a cada execução, já que cada processo pode ser realizado em tempos diferentes, atualizando valores em ordem diferentes e processando linhas em ordens definidas pelo escalonador de processos, mas mesmo com essas ocorrências, o algoritmo sempre produz um resultado esperado de acordo com o erro estipulado.

Como o algoritmo pode ser influenciado por uma série de fatores com caráter randômico, não há como dizer que para uma matriz A, sempre será feito a mesma quantidade k de iterações, obtendo exatamente o mesmo valor para o vetor x. Porém, pode se dizer que, para uma matriz A que satisfaça o critério das linhas, o algoritmo satisfaz a tolerância estabelecida.

3. Resultados

Para a comparação entre os métodos, foi utilizado a linguagem C para a implementação, junto com a biblioteca Open Multi-Processing [ope 2019] para a paralelização dos algoritmos e a biblioteca Gnu Scientific Library [gsl 2019] para a estrutura de dados que contém matrizes e vetores, além das funções próprias de cálculo numérico. Os testes foram realizados com matrizes que satisfazem o critério das linhas, geradas pela linguagem Gnu Octave.

O computador utilizado para o teste contém 12 núcleos, com processador Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, com RAM 16GB, executando no sistema Linux Mint .

Os testes onde a matriz A foi gerada com a distribuição normal, demonstra casos onde a variação da quantidade de iterações entre os dois métodos não é grande, como visto na figura 1. Com isso, foi representado o tempo gasto por cada método, disponível

na figura 7.

Para a verificação do resultado, foi utilizado o cálculo da norma do vetor residual dado por $r = \|Ax - b\|$. Logo se $r \leq \varepsilon$, podemos afirmar que o sistema atingiu a aproximação desejada para o vetor coluna x. O resultado do valor residual de cada teste obtido está contido na figura 8.

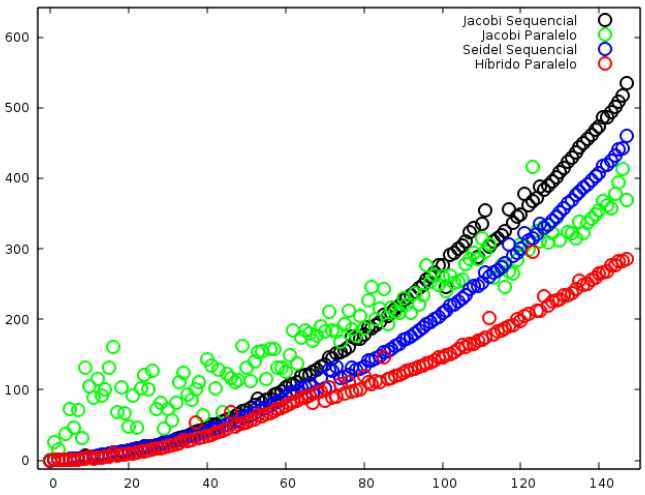

Figura 7. Tempo decorrido de cada método para matrizes gerada a partir da distribuição normal.

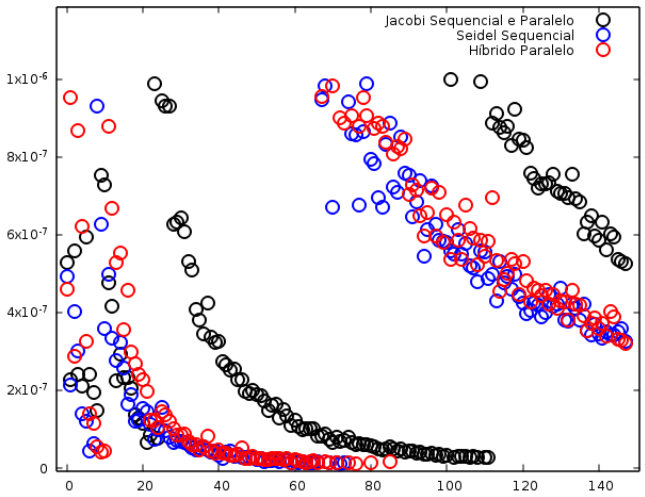

Figura 8. Residual das aproximações resultantes de cada método, para uma matriz A gerada pela distribuição normal.

Foi realizado também, a comparação em que a variação da quantidade de iterações entre os dois métodos era maior, que ocorreu ao utilizar a distribuição de Weibull, conforme apresentado na figura 2. Com essa distribuição, ocorreu um aumento na quantidade de iterações em todos os métodos, porém, a quantidade feita pelo Gauss-Seidel continuou sendo muito inferior que a do Gauss-Jacobi sequencial e paralelo, que causa uma grande diferença de tempo de resposta, conforme visto na figura 9. Esse teste demonstra que há

casos onde a vantagem do Gauss-Seidel sequencial pode ser grande a ponto de superar o Gauss-Jacobi paralelo.

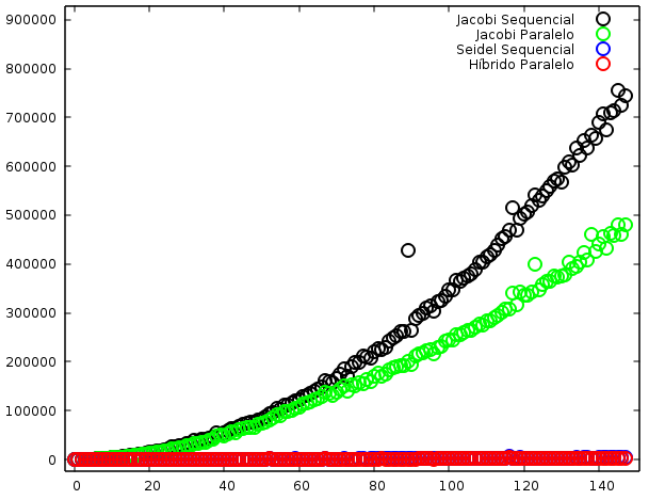

Figura 9. Tempo decorrido de cada método para matrizes gerada a partir da distribuição Weibull.

O cenário muda ao comparar o Gauss-Seidel com o método híbrido implementado, já que a quantidade de iterações não contém tanta variação entre os dois. Como o método híbrido ocorre em paralelo, em média seu tempo de resposta se mostra menor, representado na figura 10, mas ainda é um fator que depende do tamanho da matriz.

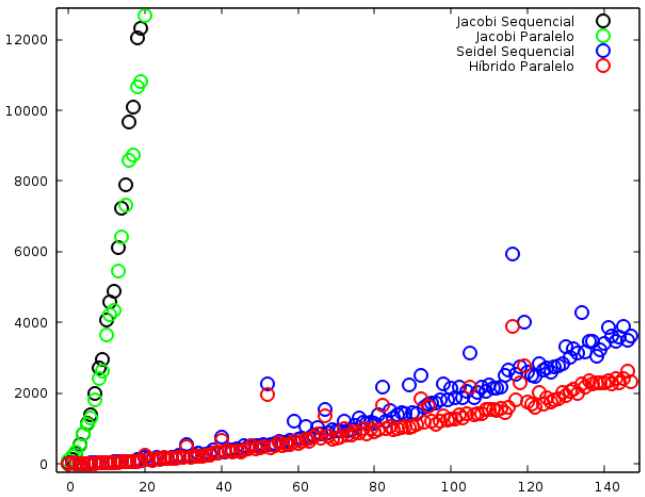

Figura 10. Tempo decorrido para o método de Gauss-Seidel e Híbrido.

Para os testes realizados com a distribuição de Weibull, o residual para cada teste pode ser visto na figura 11.

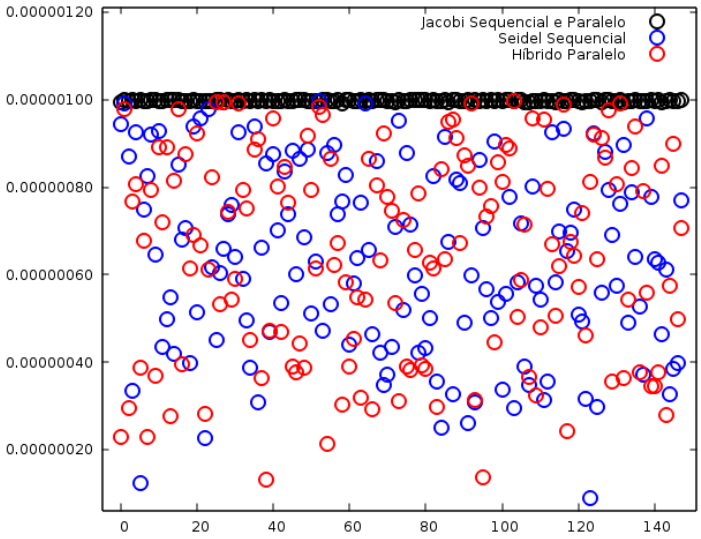

Figura 11. Residual das aproximações resultantes de cada método, para uma matriz A gerada ela pela distribuição de Weibull.

4. Conclusões

Após a análise dos testes, foi percebido que em média há uma economia de tempo de resposta para os métodos paralelos. As vantagens obtidas pelo Gauss-Seidel sequencial sobre o método de Gauss-Jacobi paralelo nem sempre é garantida, já que nesse caso depende de uma grande diferença entre a quantidade de iterações realizadas por cada método. Outro ponto observado é que o método híbrido implementado possui uma resposta rápida, porém influenciada por diversos aspectos enquanto está sendo executado, tornando seu comportamento imprevisível. Logo, para aplicações na qual necessite de respostas aproximadas de forma rápida, o método híbrido se torna vantajoso, enquanto que, para aplicações nas quais necessitem de maior previsibilidade, o método de Gauss-Jacobi paralelo apresenta uma estabilidade maior.

Referências

(2019). Gnu scientific library. https://www.gnu.org/software/gsl/.

(2019). Openmp. https://www.openmp.org/.

Brent, R. P. and Zimmermann, P. (2010). *Modern Computer Arithmetic*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.

De Paula, L. (2013). Paralelização e comparação de métodos iterativos na solução de sistemas lineares grandes e esparsos. 1(1).

Gautschi, W. (2011). *Numerical Analysis*. Birkhäuser.

Hoffman, K. and Kunze, R. A. (2004). *Linear Algebra*. PHI Learning, second edition.

Howell, L. W. and Rheinfurth, M. H. (1982). Generation of pseudo-random numbers. Technical report, NASA.

Khojasteh Salkuyeh, D. (2007). Generalized jacobi and gauss-seidel methods for solving linear system of equations. 16.