https://eventos.utfpr.edu.br//sicite/sicite2019

Comparação dos métodos iterativos Gauss-Jacobi e Gauss-Seidel

Comparison of iterative methods Gauss-jacobi and Gauss-Seidel

Felipe Gimenez Da Silva
felipegimenezsilva@gmail.com
Universidade Técnológica Federal do Paraná ,Ponta Grossa, Paraná, Brasil

Erikson Freitas de Morais
emorais@utfpr.edu.br
Universidade Técnológica Federal do Paraná , Ponta Grossa, Paraná, Brasil

Iara da Cunha Ribeiro da Silva
iarasilva@utfpr.edu.br
Universidade Técnológica Federal do Paraná , Ponta Grossa, Paraná, Brasil

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 de apresentar 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. Também contém os passos que foram seguidos para a elaboração de cada um dos testes, e seus respectivos resultados, 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.

PALAVRAS-CHAVE: Paralelismo. Métodos numéricos. Método híbrido paralelo.

Recebido: 19 ago. 2019.
Aprovado: 01 out. 2019.

ABSTRACT

This paper aims to analyze and compare the response time difference of the numerical methods known as Gauss-Seidel and Gauss-Jacobi, using sequential and parallel programming approaches in their implementations, and to present a hybrid numerical method as a parallelization option. Gauss-Seidel method, proposed for solving problems of linear systems that satisfy the line criterion. It also contains the steps that were followed for the elaboration of each of the tests, and their respective results, 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.

KEYWORDS: Paralelism. Numerical Methods. Hybrid parallel method.

INTRODUÇÃO

Esse trabalho está focado na comparação dos métodos numéricos de resolução de sistemas lineares, conhecidos como método de Gauss Seidel e método de Gauss Jacobi. Os métodos são abordados utilizando as implementações sequenciais e paralelas de cada método. O objetivo é sugerir uma forma de implementação paralela para o Gauss Seidel, e evidenciar casos onde a implementação sequencial dos métodos números se demonstra mais efetiva. A pesquisa limita-se a utilizar apenas matrizes que satisfazem o critério das linhas, geradas a partir de duas distribuições randômicas, como é explicado posteriormente. Há diversas pesquisas de comparações de métodos numéricos paralelos, utilizando outras abordagens, como o artigo feito por De Paula(2013), utilizando a tecnologia CUDA.

MATERIAIS E MÉTODOS

Os métodos iterativos, são fundamentados na ideia de aproximações sucessivas para resolução de problemas lineares AX=B, onde A é uma matriz de ordem n, X é um vetor de incógnitas de n linhas por 1 coluna, e B é o vetor de termos independentes, de n linhas por 1 coluna. Para obter uma aproximação que satisfaça o problema, deve-se informar um valor inicial, chamado de X_0. Com isso, temos que para cada iteração k, haverá uma nova aproximação para o vetor de incógnitas X_k. Quando os valores de X_k se aproximam da solução do sistema, é dito que o método converge.

Existem alguns critérios que garantem a convergência de problemas lineares, independente dos valores contidos no vetor inicial X_0. O critério utilizado para os testes realizados é conhecido como critério das linhas, dado pela equação 2:

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}| \tag{2}$$

onde a_{ij} representa um elemento da matriz A. Para que a matriz A obedeça ao critério, sua criação seguiu os passos:

$$a_{ij} = R_1 \tag{3}$$

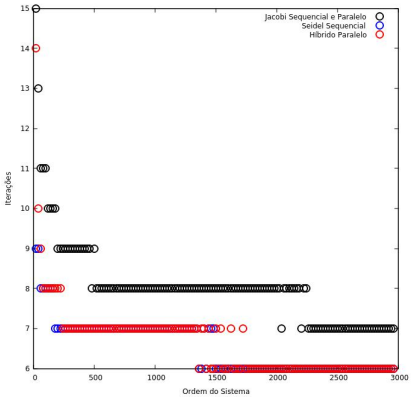
onde R_1 é um número randômico qualquer. Após randomizar todos os elementos de uma linha i, exceto o elemento pertencente a diagonal principal, é aplicado a equação 4 referente a linha i:

$$a_{ii} = \left(\sum_{j=1, j\neq i}^{n} |a_{ij}| \right) * \left| \frac{1}{R_2} \right| \tag{4}$$

onde R_2 é um número randômico maior que zero e menor que um.

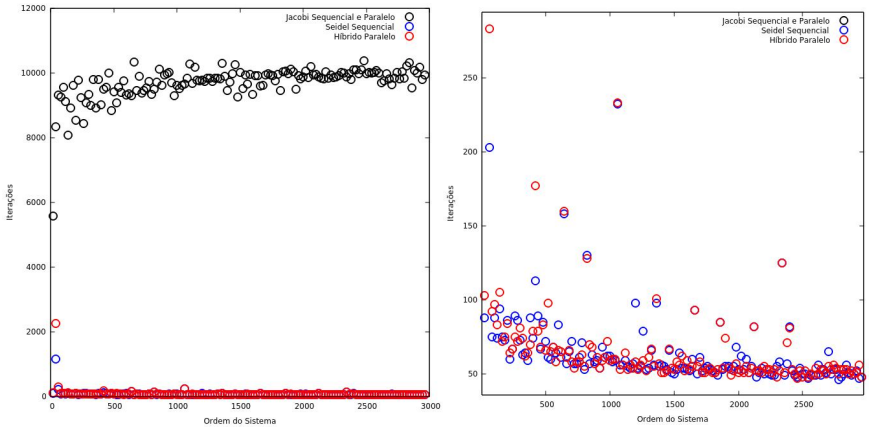
A escolha da distribuição randômica e do valor de R_2 afeta diretamente a quantidade de iterações realizadas. Ao utilizar a distribuição normal, não houve grandes alterações entre a quantidade de iterações necessárias para resolver os problemas lineares com os métodos numéricos de Gauss-Seidel e o Gauss-Jacobi,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

Figura 1 – Iteração para matrizes geradas pela distribuição normal



Fonte: produção própria

Figura 2 – Iteração para matrizes geradas pela distribuição de Weibull



Fonte: produção própria

Para que os algoritmos de Gauss-Jacobi, Gauss-Seidel e Híbrido determinem se o vetor de incógnitas X_k está com uma aproximação aceitável, deve-se estabelecer qual o máximo erro pretendido. O modo escolhido para o cálculo do erro é chamado de valor residual, dado pela equação 5:

$$e > || A * X_k - b || \quad (5)$$

onde e é o máximo erro aceitável.

Um dos algoritmos iterativos utilizados para resolver sistemas lineares é o Gauss-Jacobi. Seu funcionamento se dá para cada linha i, como observado na equação 6:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{n} a_{ij} * x_j^{(k)} \right) \quad (6)$$

para i = 1, 2, 3, ..., n.

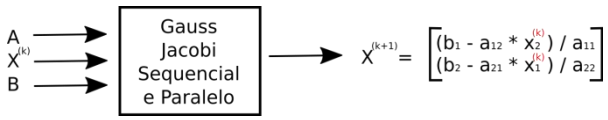
A implementação do método de Gauss jacobi é dada pela figura 3, e seu comportamento característico para um sistema de ordem 2 é dado pela figura 4.

Figura 3 – Implementação sequencial e paralela do método de Gauss Jacobi

```
int jacobi_sequencial ( gsl_matrix *m, gsl_vector *b,
                        gsl_vector *xk,gsl_vector *tm )
{
   int ord = m->size1 ;
   int k = 0;
   do
   {
      // ao incluir a linha 10, o algoritmo
      // torna-se paralelo (Gauss-Jacobi Paralelo)
      // #pragma omp parallel for
      for(int i = 0 ; i < ord ; i++)
      {
         long double x = gsl_vector_get(b,i);
         for(int j = 0 ; j < ord ; j++)
            if(i!=j)
               x -= gsl_matrix_get(m,i,j) * gsl_vector_get(xk,j) ;
         x /= gsl_matrix_get(m,i,i);
         gsl_vector_set(tm,i,x);
      }
      k++;
      gsl_vector_memcpy(xk,tm);
   }while(erro(m,b,xk) > MAX_ERRO && k < MAX_ITERATION );
   return k;
}
```

Fonte: produção própria.

Figura 4 – Comportamento do método de Gauss Jacobi para um sistema de ordem 2



Fonte: produção própria.

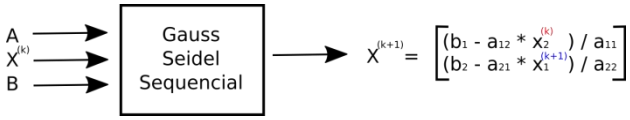
Além do método de Gauss-Jacobi, problemas de sistemas lineares também podem ser resolvidos pelo método de Gauss-Seidel. Esse método tem sua forma muito semelhante a de Gauss-Jacobi, porém parte do princípio de sempre aproveitar a melhor aproximação calculada anteriormente $x_j^{(k+1)}$, como pode ser visto na equação 7:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} * x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} * x_j^{(k)}\right) \quad (7)$$

para i = 1, 2, 3, ..., n

Seu comportamento para um sistema de ordem 2, dado pela figura 5, contém uma pequena variação em relação ao comportamento do método de Gauss Jacobi (figura 4).

Figura 5 – Comportamento do método de Gauss Seidel para um sistema de ordem 2



Fonte: produção própria.

O método híbrido paralelo foi criado através da implementação paralela do Gauss Seidel. Ao paralelizar o método

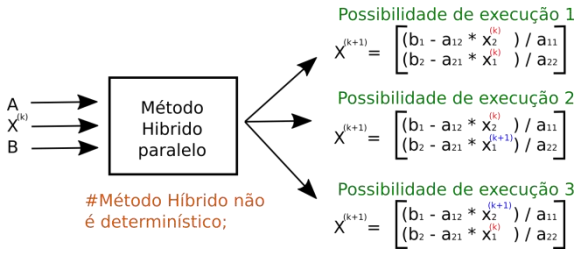
de Gauss Seidel do mesmo modo que ocorre no método de Gauss Jacobi, ocorrerá uma condição de corrida. Porém a condição gerada não afeta negativamente o funcionamento do método, mas cria novas possibilidades de aproximações de modo não determinístico, como pode ser visto na figura 7. A implementação dos métodos é dado pela figura 6.

Figura 6 – Implementação do método de Gauss Seidel sequencial e Híbrido paralelo

```
1  int seidel_sequencial( gsl_matrix *m,  gsl_vector *b,
2                         gsl_vector *xk,gsl_vector *tm )
3  {
4     int ord = m->size1 ;
5     int k = 0;
6     do
7     {
8        // ao incluir a linha 12, o algoritmo
9        // se tornará paralelo, porém haverá condição
10       // de corrida. Então, o Gauss-Seidel sequencial torna-se
11       // o método Híbrido não determinístico tratado na pesquisa.
12       // #pragma omp parallel for
13       for(int i = 0 ; i < ord ; i++)
14       {
15          long double x = gsl_vector_get(b,i);
16          for(int j = 0 ; j < ord ; j++)
17             if(i!=j)
18                x -= gsl_matrix_get(m,i,j) * gsl_vector_get(xk,j) ;
19          x /= gsl_matrix_get(m,i,i);
20          gsl_vector_set(xk,i,x);
21       }
22       k++;
23    }while(erro(m,b,xk) > MAX_ERRO && k < MAX_ITERATION );
24    return k;
25 }
```

Fonte: produção própria.

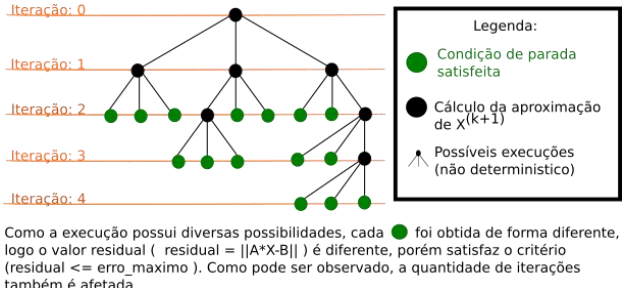
Figura 7 – Possíveis respostas do método Híbrido paralelo para um sistema de ordem 2



Fonte: produção própria.

Com a implementação do método híbrido paralelo, foi observado um melhor tempo de resposta para os casos avaliados. Como não é determinístico, o método pode convergir de diferentes formas, para diferentes soluções, como exemplificado na figura 8.

Figura 8 – Exemplo de possibilidades de convergência do método híbrido para um sistema de ordem 2

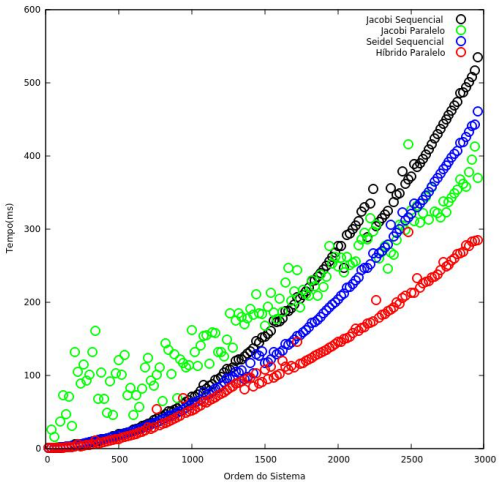


Fonte: produção própria.

RESULTADOS E DISCUSSÕES

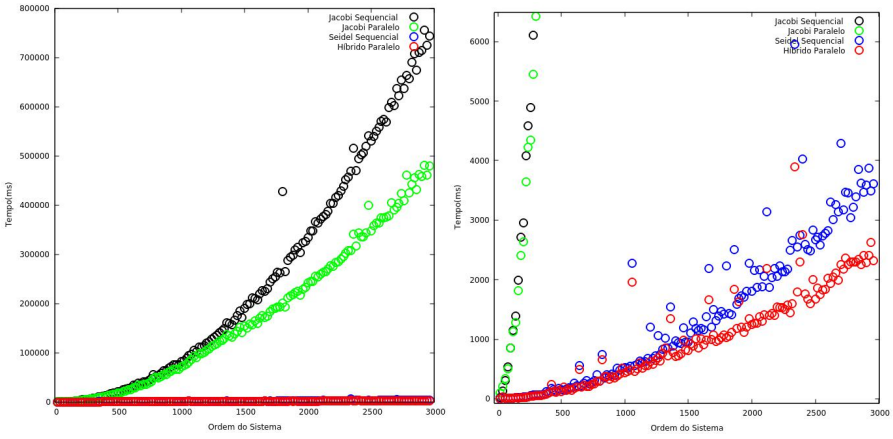
Os sistemas lineares foram gerados na linguagem GNU Octave de forma pseudo randômica, utilizando a distribuição normal e de Weibull. O valor residual máximo permitido é de 10^{-6}. Os testes foram realizados em um computador de 12 núcleos, processadores Intel(R) Core(TM) i7-8700. Os métodos foram escritos na linguagem C, utilizando a biblioteca Gnu Scientific Library e Open Multi-Processing. As figuras 9 e 10 apresentam o tempo gasto para cada um dos testes realizados, e a figura 11 apresenta o valor residual de cada teste.

Figura 9 – Tempo de processamento para matrizes randomizadas com a distribuição normal



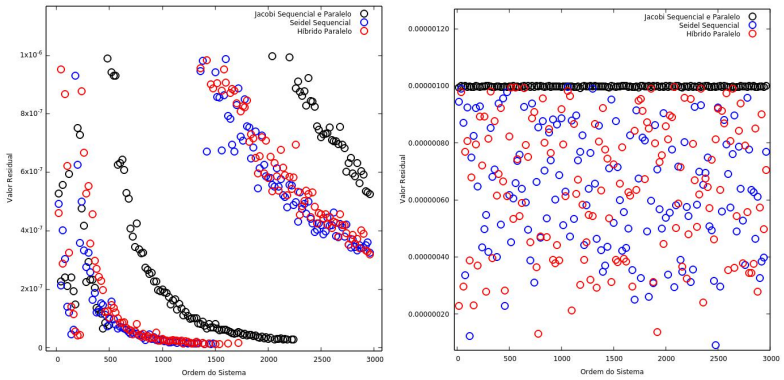
Fonte: produção própria.

Figura 10 – Tempo de processamento para matrizes randomizadas com a distribuição de Weibull



Fonte: produção própria.

Figura 11 – Valor Residual dos sistemas gerados pela distribuição normal e de Weibull



Fonte: produção própria.

CONCLUSÕES

Os testes realizados evidenciam que existem casos onde o uso de paralelismo não se faz necessário, como ocorre quando as matrizes não tem uma ordem suficientemente grande, pois o custo temporal para o sistema operacional gerenciar as *threads* geradas pelo algoritmo se torna superior.

Outro ponto observado é que há momentos onde a diferença entre a quantidade necessária de iterações utilizadas pelo Gauss-Jacobi com a abordagem sequencial e a abordagem paralela é muito alta em relação ao Gauss-Seidel sequencial, deixando seu processamento inviável.

O método híbrido paralelo implementado, mostrou um desempenho mais eficiente, porém não há previsibilidade de comportamento, já que pode haver formas diferentes de convergência de uma mesma matriz a cada execução.

REFERÊNCIAS

DE PAULA, Lauro. (2013). **Paralelização e Comparação de Métodos Iterativos na Solução de Sistemas Lineares Grandes e Esparsos**. ForScience: Revista Científica do IFMG. 1. 10.29069/forscience.2013v1n1.e18. Acesso em: 13 set. 2019

GAUTSCHI, Walter. **Numerical Analysis**. 2011. Disponível em: http://www.ikiu.ac.ir/public-files/profiles/items/090ad_1410599906.pdf . Acesso em: 15 nov. 2018.

Marc Lars Lipson e Seymour Lipschutz. **Linear Algebra**. 2009. Disponível em: http://www.astronomia.edu.uy/progs/algebra/Linear_Algebra,_4th_Edition__(2009)Lipschutz-Lipson.pdf . Acesso em: 08 mar. 2019.