Performance dos Métodos de Jacobi, Gauss-Siedel e SRS para Resolver Sistemas de Equações de Problemas de Fluxo em Meio Poroso Incompressíveis*

Tiago C. A. Amorim[2]

[a]*Petrobras, Av. Henrique Valadares, 28, Rio de Janeiro, 20231-030, RJ, Brasil*

Abstract

Uma simulação de fluxo em meio poroso demanda a resolução de grandes sistemas de equações não-lineares a cada passo de tempo. Em geral estes sistemas são resolvidos com métodos que fazem a resolução de sucessivos sistemas de equações lineares. Este trabalho avaliou a utilização de métodos iterativos na resolução de problemas de fluxo em meio poroso incompressível. Os Métodos de Jacobi, Gauss-Seidel e SRS tiveram performance pior que a do Método de Eliminação de Gauss, e se mostraram inadequados para o problema proposto.

Keywords: Método de Eliminação de Gauss, Método de Jacobi, Método de Gauss-Siedel, Método SRS, Fluxo em Meio Poroso

1. Introdução

Os sistemas de equações lineares que aparecem na resolução de um problema de fluxo em meio poroso em geral são caracterizadas por terem muitas variáveis. É comum os modelos terem mais de 100 mil parâmetros (podendo chegar a alguns milhões). Outra característica é a esparsidade da matriz de coeficientes.

Este trabalho continua a avaliação de métodos de resolução de sistemas de equações lineares. O estudo anterior verificou que o Método da Eliminação de Gauss gera bons resultados, mas é computacionalmente custoso[1]. Nesta segunda etapa são avaliados três métodos iterativos: Jacobi, Gauss-Siedel e SRS.

2. Metodologia

A descrição do Método de Eliminação de Gauss e do fluxo em meio poroso incompressível foi feita no relatório anterior[1], e não será repetida neste relatório.

2.1. Método de Jacobi

Dado um sistema de n equações lineares na forma $A\vec{x} = \vec{b}$:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{2.1}$$

O Método de Jacobi consiste em, primeiramente, explicitar cada uma das variáveis do sistema de equações ($x_i = f_i(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$). A partir de uma estimativa inicial ($\vec{x}_0 = \{x_1^0, x_n^0, \dots, x_n^0\}$) são realizadas sucessivos cálculos com estas funções, até ser atingido algum critério de convergência[2].

A partir de 2.1 é possível construir equações para cada uma das variáveis do problema da seguinte forma:

$$x_i = \frac{1}{a_{i,i}}(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j}x_j) \tag{2.2}$$

Como o termo $a_{i,i}$ aparece no divisor de 2.2, é uma condição necessária para o método não existirem valores nulos na diagonal da matriz de coeficientes. Para isso é preciso fazer trocas de linhas. Uma estratégia para melhorar o método é buscar ter valores altos na diagonal, ou buscar colocar valores na diagonal *relativamente* altos com relação aos demais termos da mesma linha (maximizar $\frac{|a_{i,i}|}{\max_j |a_{i,j}|}$).

O algoritmo de Jacobi é apresentado em 1.

Diferentes critérios de convergência podem ser utilizados para definir quando a convergência é atingida. Definindo duas normas: L^2 (2.3) e L^∞ (2.4):

*Relatório número 11 como parte dos requisitos da disciplina IM253: Métodos Numéricos para Fenômenos de Transporte.

**Atualmente cursando doutorado no Departamento de Engenharia de Petróleo da Faculdade de Engenharia Mecânica da UNICAMP (Campinas/SP, Brasil).

Email address: t100675@dac.unicamp.br (Tiago C. A. Amorim)

Algorithm 1 Método de Jacobi

Entrada: $a_{i,j}$, b_i e x_i^0 *para* $i = 1, 2, \ldots, n$ e $j = 1, 2, \ldots, n$;
$\epsilon_{conv} > 0$ e $k_{max} \in \mathbb{Z}, k_{max} > 0$

Para todos $i \in \{1, \ldots, n\}$ **faça**
  Se $a_{i,i} = 0$ **então**
    Retorna: Erro: termo nulo na diagonal.
$k \leftarrow 1$
Enquanto $k \leq k_{max}$ **faça**
  Para todos $i \in \{1, \ldots, n\}$ **faça**
    $x_i^k \leftarrow \frac{1}{a_{i,i}}(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j} x_j^{k-1})$
  Se $||\vec{x}_k - \vec{x}_{k-1}|| < \epsilon_{conv}$ **então**
    Retorna: x_i^k *para* $i = 1, 2, \ldots, n$
  $k \leftarrow k + 1$
Avisar que método não atingiu convergência
Retorna: x_i^k *para* $i = 1, 2, \ldots, n$

Algorithm 2 Método de Gauss-Siedel

Entrada: $a_{i,j}$, b_i e x_i^0 *para* $i = 1, 2, \ldots, n$ e $j = 1, 2, \ldots, n$;
$\epsilon_{conv} > 0$ e $k_{max} \in \mathbb{Z}, k_{max} > 0$

Para todos $i \in \{1, \ldots, n\}$ **faça**
  Se $a_{i,i} = 0$ **então**
    Retorna: Erro: termo nulo na diagonal.
$k \leftarrow 1$
Enquanto $k \leq k_{max}$ **faça**
  Para todos $i \in \{1, \ldots, n\}$ **faça**
    $x_i^k \leftarrow \frac{1}{a_{i,i}}(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k-1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k)$
  Se $||\vec{x}_k - \vec{x}_{k-1}|| < \epsilon_{conv}$ **então**
    Retorna: x_i^k *para* $i = 1, 2, \ldots, n$
  $k \leftarrow k + 1$
Avisar que método não atingiu convergência
Retorna: x_i^k *para* $i = 1, 2, \ldots, n$

$$||x||_2 := \sqrt{\sum_i x_i^2} \tag{2.3}$$

$$||x||_\infty := \max_i |x_i| \tag{2.4}$$

Podemos utilizar diferentes critérios de convergência, como:

$$\epsilon = ||x_i^k - x_i^{k-1}||_2 \tag{2.5}$$

$$\epsilon = ||x_i^k - x_i^{k-1}||_\infty \tag{2.6}$$

$$\epsilon = \frac{||x_i^k - x_i^{k-1}||_2}{||x_i^k||_2} \tag{2.7}$$

$$\epsilon = \frac{||x_i^k - x_i^{k-1}||_\infty}{||x_i^k||_\infty} \tag{2.8}$$

$$\tag{2.9}$$

2.2. Método de Gauss-Siedel

O Método de Gauss-Siedel parte da ideia do Método de Jacobi, mas utiliza a estimativa *mais atual* de cada variável. Ou seja, ao invés de usar os valores de x_i^{k-1} em cada passo da iteração, é utilizado o valor de x_i^k, quando disponível.

As discussões sobre os termos da diagonal e critérios de convergência feitas para o Método de Jacobi seguem válidas para este método. O algoritmo do método é apresentado em 2.

2.3. Métodos SRS

Podemos calcular o residual da m-ésima variável na k-ésima iteração do Método de Gauss-Siedel como:

$$r_i^k = b_i - \sum_{j=1}^{i} a_{i,j} x_j^{k-1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k \tag{2.10}$$

Maniplando 2.10 e substituindo na equação de atualização de x_i do Método de Gauss-Siedel, temos:

$$r_i^k = b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k-1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k$$

$$r_i^k + a_{i,i} x_i^{k-1} = b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k-1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k$$

$$\frac{r_i^k}{a_{i,i}} + x_i^{k-1} = \frac{1}{a_{i,i}}(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k-1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k)$$

$$x_i^k = x_i^{k-1} + \frac{r_i^k}{a_{i,i}} \tag{2.11}$$

A equação 2.11 pode ser encarada como uma sequência. Uma forma de ajudar na convergência de x_i é aplicar um modificador no termo que atualiza x_i:

$$x_i^k = x_i^{k-1} + \omega \frac{r_i^k}{a_{i,i}}, \text{ com } \omega > 0 \tag{2.12}$$

Quando $0 < \omega < 1$ temos métodos de sub-relaxação, e quando $\omega > 1$ temos métodos de sobre-relaxação. A equação 2.12 ainda pode ser manipulada para explicitar x_i^{k-1} do lado direito e chegar ao Método SRS (sobre-relaxação sucessiva):

$$\begin{aligned} x_i^k =& (1 - \omega) x_i^{k-1} \\ &+ \omega \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k-1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k \right) \end{aligned} \tag{2.13}$$

O algoritmo do Método SRS é parecido com o algoritmo de Gauss-Seidel. O algoritimo é apresentado em 3.

Algorithm 3 Método SRS

Entrada: $a_{i,j}$, b_i e x_i^0 *para* $i = 1, 2, \ldots, n$ e
$j = 1, 2, \ldots, n$;
$\omega > 0$, $\epsilon_{conv} > 0$ e $k_{max} \in \mathbb{Z}, k_{max} > 0$
Para todos $i \in \{1, \ldots, n\}$ **faça**
Se $a_{i,i} = 0$ **então**
Retorna: Erro: termo nulo na diagonal.
$k \leftarrow 1$
Enquanto $k \leq k_{max}$ **faça**
Para todos $i \in \{1, \ldots, n\}$ **faça**
$x_{i,GS} \leftarrow \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k-1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k \right)$
$x_i^k \leftarrow (1 - \omega) x_i^{k-1} + \omega x_{i,GS}$
Se $||\vec{x}_k - \vec{x}_{k-1}|| < \epsilon_{conv}$ **então**
Retorna: x_i^k *para* $i = 1, 2, \ldots, n$
$k \leftarrow k + 1$
Avisar que método não atingiu convergência
Retorna: x_i^k *para* $i = 1, 2, \ldots, n$

2.4. Problema proposto

O problema proposto é a resolução de uma das iterações do Método do Ponto Fixo que foi implementado para resolver um modelo de fluxo em meio poroso incompressível. Foram gerados arquivos com a matriz de coeficientes e o vetor de constantes para dois tipos de modelo (uni e bidimensional) e diferentes refinamentos de malha.

O relatório anterior[1] descreve em mais detalhes o problema a ser resolvido.

3. Implementação

Todo o código utilizado nesta análise foi desenvolvido em C++. As principais funções são:

readCSV Função que recebe um *string* com o camimnho de um arquivo CSV e faz a sua leitura. É assumido que é utilizado vírgula como separador. A função retorna uma matrix de *double*.

SolveGauss Função que recebe uma matriz de *double* e um vetor de *double*, e resolve o sistema de equações lineares usando Eliminação de Gauss com Pivotamento Parcial. Existe a opção de realizar o pivotamento com e sem uso de escala.

SolveSRS Função que recebe uma matriz de *double* e um vetor de *double*, e resolve o sistema de equações lineares usando o Método SRS. Ao definir $\omega = 1$ o algoritmo coincide com o Método de Gauss-Siedel. Existe a opção de solicitar que sejam sempre utilizados os valores de x_j^{k-1} nos cálculos de x_i^k, que junto com $\omega = 1$ leva ao Método de Jacobi.

4. Resultados

Uma primeira dificuldade encontrada na implementação foi a definição do critério de convergência. Existe uma diferença significativa entre a ordem de grandeza das variáveis do problema proposto (saturações e pressões), de forma que um critério do tipo 2.8 não é muito adequado. Foi preciso adotar um critério de convergência ligeiramente diferente dos listados anteriormente. A proposta foi de utilizar o máximo erro relativo por variável:

$$\epsilon = \left|\left| \frac{x_i^k - x_i^{k-1}}{x_i^k} \right|\right|_\infty \tag{4.1}$$

Em 4.1 a divisão é feita elemento a elemento (*piecewise*).

O primeiro teste realizado foi o de verificar a qualidade das respostas das resoluções de diferentes sistemas de equações lineares de modelos unidimensionais. Foi adotado um valor de convergência baixo (10^{-5}) e um número máximo iterações alto (2000). Foram testados os três métodos expostos, e o Método SRS foi testado com $\omega = 0.8$ e $\omega = 1.2$.

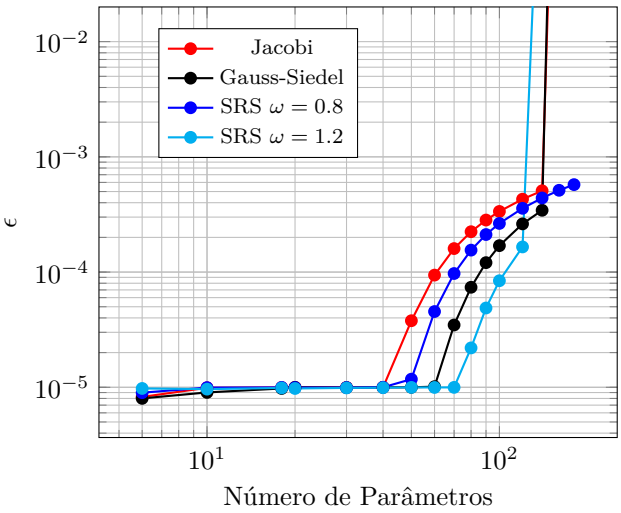

Figura 1: Valor da norma de convergência após até duas mil iterações na resolução de problemas unidimensionais.

Observa-se na Figura 1 que alguns dos métodos não conseguiram atingir o critério de convergência dentro do número de iterações estabelecidas. Foi reaizada uma segunda tentativa, agora com até 10 mil iterações.

Na Figura 2 *faltam* marcadores dos resultados com maior número de parâmetros. Vários métodos divergiram nos casos com maior número de parâmetros. Nem mesmo o Método SRS com $\omega = 0.8$ conseguiu gerar resultados para todos os problemas testados, pois divergiu nos testes com 160, 180 e 200 parâmetros.

Todos os métodos foram muito rápidos na resolução (convergindo ou divergindo) dos problemas unidimensionais testados, em geral tomando menos de 1 segundo. Na Figura 3 estão os resultados para o limite de 10 mil iterações, e, mesmo sendo um método lento, o Método da Eliminação de Gauss foi o mais rápido. A queda no tempo

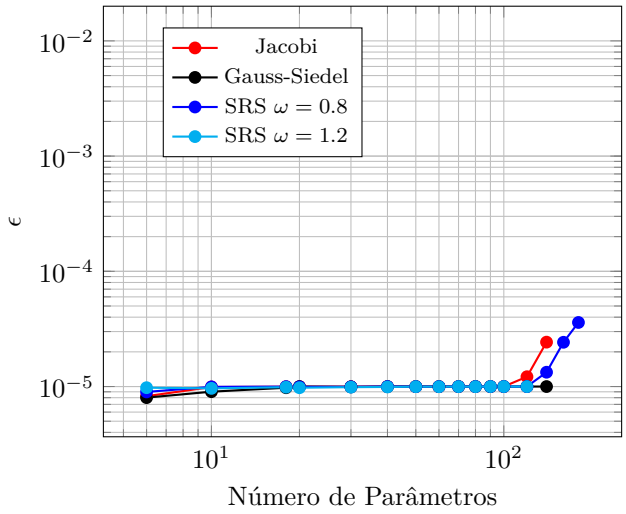

Figura 2: Valor da norma de convergência após até dez mil iterações na resolução de problemas unidimensionais.

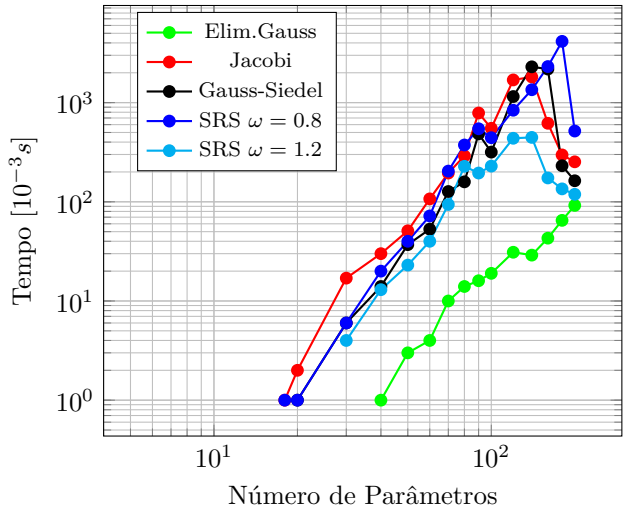

Figura 3: Tempo de resolução de problemas unidimensionais.

de resolução de alguns casos é reflexo da divergência das respostas.

O pacote para Python *Numpy* foi utilizado para resolver os mesmos sistemas de equações[3]. Estes resultados foram utilizados como uma aproximação das respostas exatas. Os Gráficos 4 e 5 mostram a máxima diferença absoluta entre os valores *exatos* e os resultados de cada método, para as pressões e saturações das células, respectivamente.

O Método da Eliminação de Gauss foi ocultado do gráfico de erros na pressão porque seus resultados são muito melhores que os dos outros métodos ($||p_{i,exato} - p_i||_\infty \approx 10^{-10}$). Os demais métodos apresentam uma relação linear entre o logarítmo do número de parâmetros e o logarítmo do erro. Os métodos iterativos tiveram dificuldade para encontrar bons resultados. Os resultados são especialmente ruins para as pressões. Em alguns casos o método da Eliminação de Gauss teve erro zero para as saturação de água, e por isso *faltam* algumas marcas no gráfico semilog.

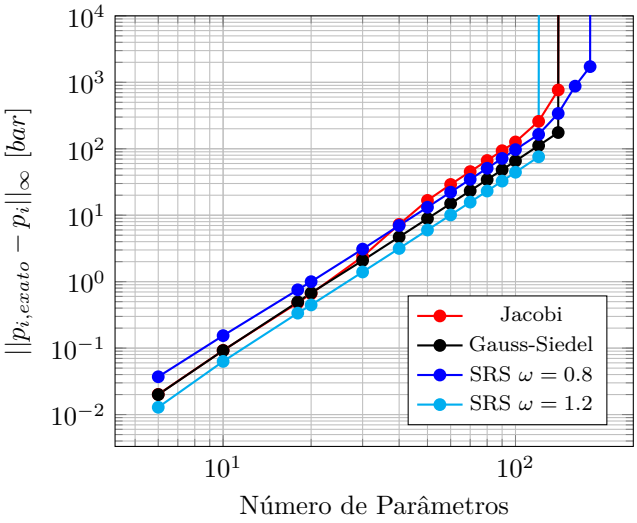

Figura 4: Máximo erro absoluto de pressão na resolução de problemas unidimensionais.

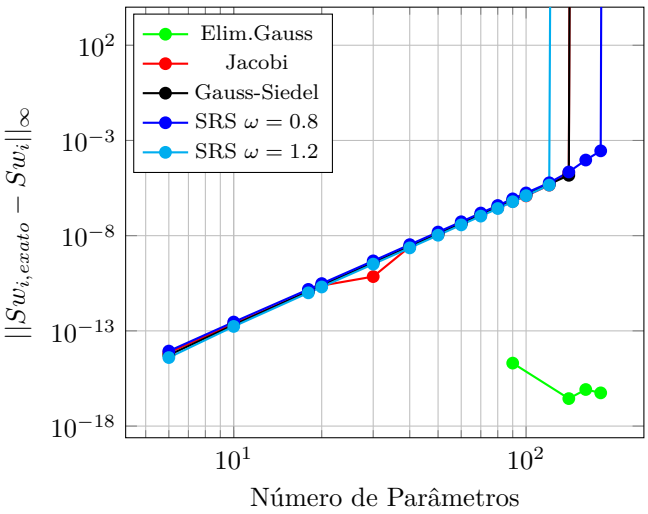

Figura 5: Máximo erro absoluto de saturação de água na resolução de problemas unidimensionais.

As avaliações foram repetidas para o caso de modelos bidimensionais (Figuras 6 a 9). Como o número de parâmetros aumenta consideravelmente, o número máximo de iterações foi de 2 mil. O destaque é o Método SRS com $\omega = 0.8$, que não divergiu nas avaliações feitas. Contudo, todos os métodos iterativos tiveram desempenho muito ruim quando o número de parâmetros aumentou muito. Para os problemas com mais de 500 variáveis o erro máximo na pressão foi de 20 a quase 200 bar (Figura 8).

Um outro efeito interessante é que, para o limite de 2 mil iterações, os métodos iterativos foram mais rápidos que o Método de Eliminação de Gauss quando o sistema de equações tinha mais de 1000 parâmetros (Figura 7). Contudo, este resultado mais rápido dos métodos iterativos não foi aceitável.

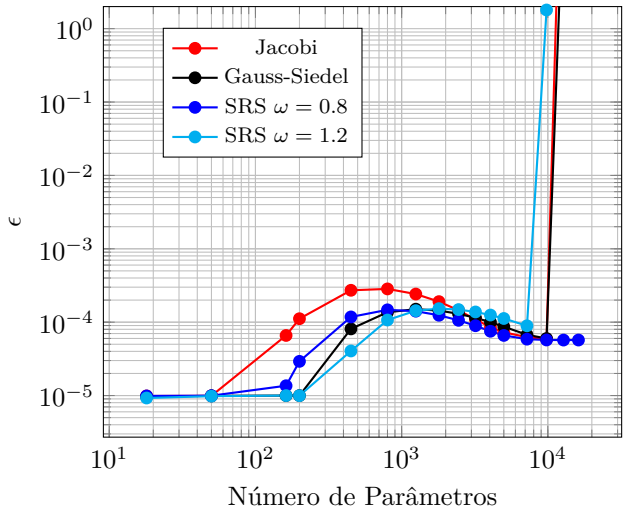

Figura 6: Valor da norma de convergência após até duas mil iterações na resolução de problemas bidimensionais.

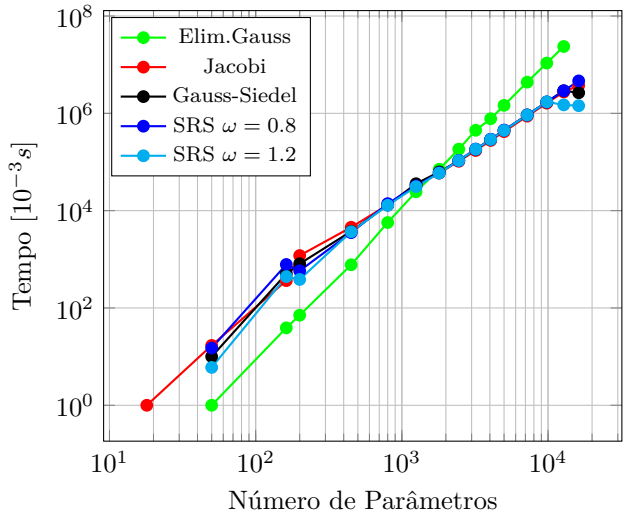

Figura 7: Tempo de resolução de problemas unidimensionais.

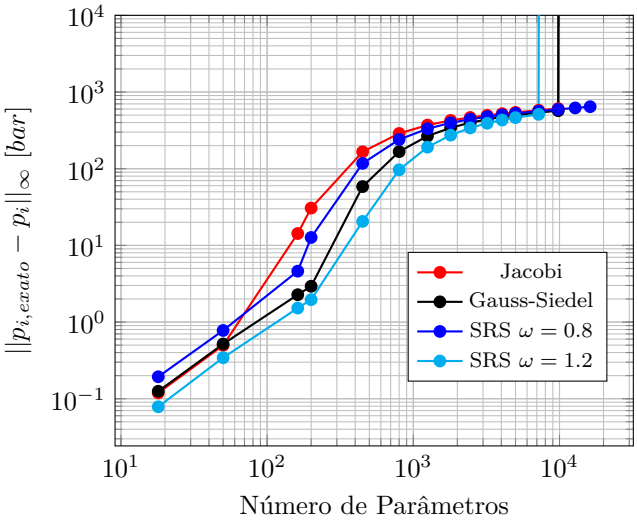

Figura 8: Máximo erro absoluto de pressão na resolução de problemas bidimensionais.

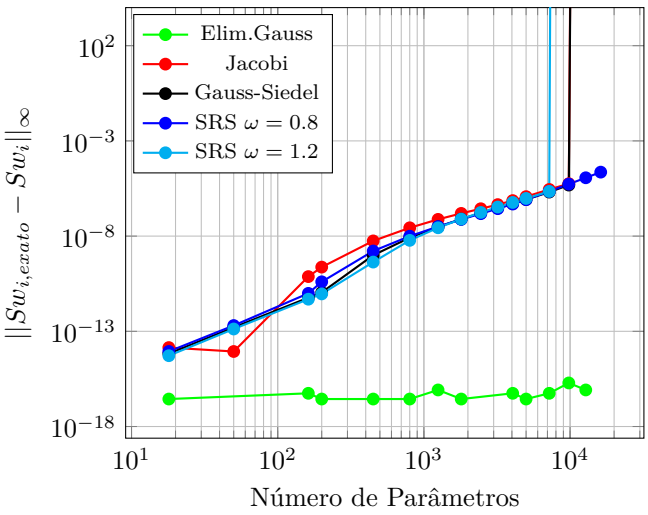

Figura 9: Máximo erro absoluto de saturação de água na resolução de problemas bidimensionais.

O código foi implementado em C++ e em um único arquivo. Pode ser encontrado em https://github.com/TiagoCAAmorim/numerical-methods.

5. Conclusão

Os testes realizados mostram que, entre os métodos testados, apenas o Método de Eliminação de Gauss chega a bons resultados. Mesmo com um incremento no número de iterações e baixo critério de convergência, os métodos iterativos avaliados tivem resultados pobres. Nos casos com um número maior de parâmetros estes métodos não convergiram.

Como o método da Eliminação de Gauss é computacionalmente muito demandante, é preciso buscar outros métodos de resolução de sistemas de equações para resolver o problema proposto. Uma possibilidade é buscar entre algoritmos que tenham boa performance com matrizes esparsas, como o Método do Gradiente Conjugado.

Referências

[1] T. C. A. Amorim, Performance do método de eliminação de gauss para resolver sistema de equações de problemas de fluxo em meio poroso incompressíveis, Relatório número 10 da disciplina IM253: Métodos Numéricos para Fenômenos de Transporte (12 2023).

[2] R. L. Burden, J. D. Faires, A. M. Burden, Análise numérica, Cengage Learning, 2016.

[3] J. J. Dongarra, J. W. Demmel, S. Ostrouchov, Lapack: a linear algebra library for high-performance computers, in: Computational Statistics: Volume 1: Proceedings of the 10th Symposium on Computational Statistics, Springer, 1992, pp. 23–28.