Trabalho Final - INF1608

Lucas Ferraço - 1513012

Pedro G. Branco - 1113366

PUC

RIO

Introdução

O método dos Gradientes Conjugados é um método iterativo para resolução de sistemas lineares na forma Ax = b. O método é largamente empregado na resolução de problemas de engenharia. Uma vantagem do método é que, em cada iteração, usa-se praticamente 4 tipos de operações: multiplicação de matriz por vetor, adição de vetores, multiplicação de escalar por vetor e produto escalar entre vetores. Portanto, cada iteração tem um custo computacional de $O(n^2)$, onde n é a dimensão do sistema. Se o método convergir em k iterações, o custo total é $O(kn^2)$. Se $k \ll n$, o método se torna muito atrativo em relação ao uso de Eliminação de Gauss (que tem complexidade $O(n^3)$).

O método dos Gradientes Conjugados na sua forma direta, no entanto, apresenta problemas de instabilidade numérica. Para amenizar o problema, usa-se pré-condicionadores M, transformando o sistema:$Ax = b \rightarrow M^{-1}Ax = M^{-1}b$

O pré-condicionador de Jacobi é a matriz diagonal de *A: M = D*. Este pré-condicionador é bom apenas quando A é estritamente diagonal dominante. O pré-condicionador SSOR (*symmetric successive over-relaxation*) apresenta melhores resultados e é dado por: $A = L + D + U$, $M = (D + wL)D^{-1}(D + wU)$, $w \in [0,2]$.

Com w = 1, temos o pré-condicionador Gauss-Seidel; com w > 1, temos um condicionador com sobre-relaxamento, que tende a aumentar a taxa de convergência do sistema.

O objetivo deste trabalho é testar a eficiência do método dos Gradientes Conjugados com pré-condicionadores para a resolução de sistemas esparsos. Considere o sistema esparso Ax = b, onde $A(i, i) = i, A(i,i+1) = A(i+1,i) = 0.5$, $A(i, i + 2) = A(i + 2, i) = 0.5$. A solução do sistema é o vetor x com todos os valores iguais a 1. Com b = Ax e usando o método de Gradientes Conjugados vamos achar uma solução x dentro de uma certa tolerância. Para tal, temos o vetor com valores todos iguais a zero como estimativa inicial.

Para valores de n variando entre 100, 1000, 10000, iremos verificar o desempenho em número de iterações e tempo de processamento para o método convergir, considerando:

- O método sem pré-condicionador: $M = I$
- O método com pré-condicionador de Jacobi: $M = D$
- O método com pré-condicionador de Gauss-Seidel: $M = (D + wL)D^{-1}(D + wU)$, com $w = 1.0$.
- O método com pré-condicionador SSOR: $M = (D + wL)D^{-1}(D + wU)$, com $w > 1.0$. e com o pré-condicionador Jacobi.

Como exemplo adicional, estendemos a matriz acima atribuindo $A(i, 2i) = A(2i, i) = 0.5$ para $i < n/2$.

Desenvolvimento

Começamos o desenvolvimento do trabalho definindo a melhor maneira de como criar os sistemas esparsos. Definimos então por criar uma estrutura chamada de *sparse* contendo um inteiro que seria o valor da coluna e um *double* que seria o valor do elemento. Para essa estrutura definimos algumas funções para o desenvolvimento do projeto como criação, remoção, acesso, multiplicação por vetor e multiplicação por matriz. Também criamos funções para vetores para criação, cópia, multiplicação por vetor, multiplicação por escalar, soma, subtração e norma.

Para um melhor desempenho não armazenamos os valores nulos da matriz, para isso criamos uma matriz de *sparse* que funciona da seguinte forma: cada linha da matriz é um vetor que possui para cada elemento o valor do número (diferente de zero) e a sua coluna. Essas implementações foram feitas no arquivo space.c com sua interface disponível no arquivo sparce.h.

Após essa etapa, desenvolvemos o arquivo gradconj.h que tem o protótipo da função do métodos dos Gradientes Conjugados, e no arquivo gradconj.c desenvolvemos a implementação.

Concluído o desenvolvimento dessa função, começamos a desenvolver as funções para o cálculo dos outros métodos: Jacobi, Gauss-Seidel e SSOR. Esses métodos estão em methods.c.

O método Jacobi retornava o pré-condicionador, uma matriz de sparse em que cada linha tinha o valor de um elemento da diagonal, e qual sua coluna já que para esse método o pré-condicionador é a matriz diagonal. E desenvolvemos também o método que retornava o pré-condicionador para os métodos de Gauss-Seidel e SSOR.

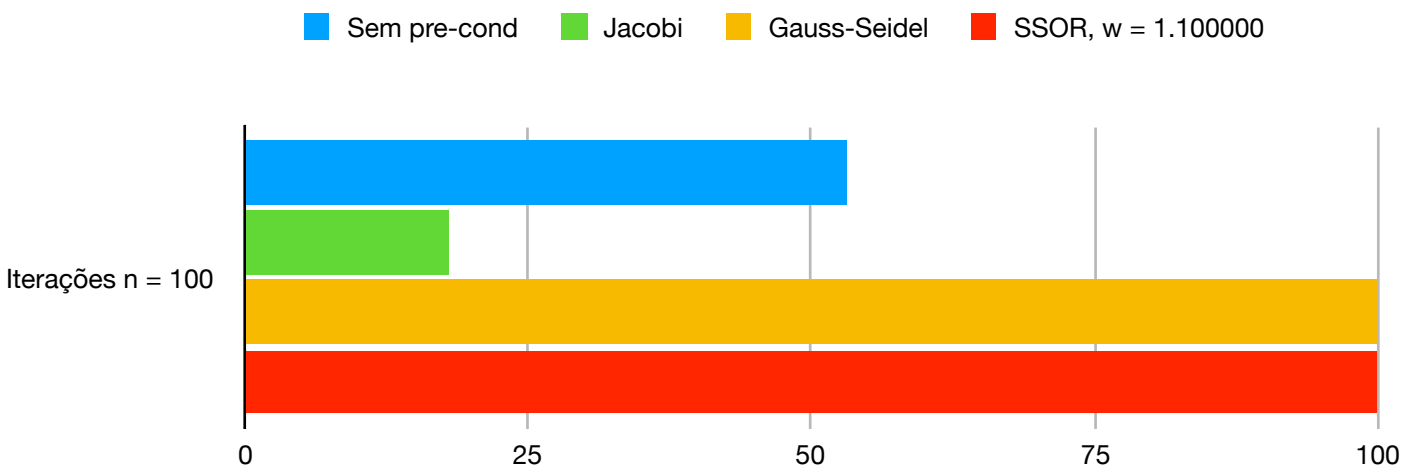
A última etapa do desenvolvimento foi testar a nossa solução conforme os exemplos e condições propostos. Criamos duas funções que geravam os sistemas esparsos propostos; utilizando nossa implementação da estrutura sparse, criamos uma matriz que continha em cada linha todos os seus valores diferentes de zero e a coluna que pertencem no sistema esparso. A última coluna da matriz era composta por elementos de *sparses* cuja coluna tinha o valor -1 para facilitar a implementação de diversos métodos no projeto.

Criamos funções para gerar a solução com valores 1 e usamos uma tolerância de 0.00001 e criamos uma função de teste para calcular o erro médio de cada método e o número de iterações para o método convergir. Calculamos o tempo que cada método demorava para o convergir na própria função main.

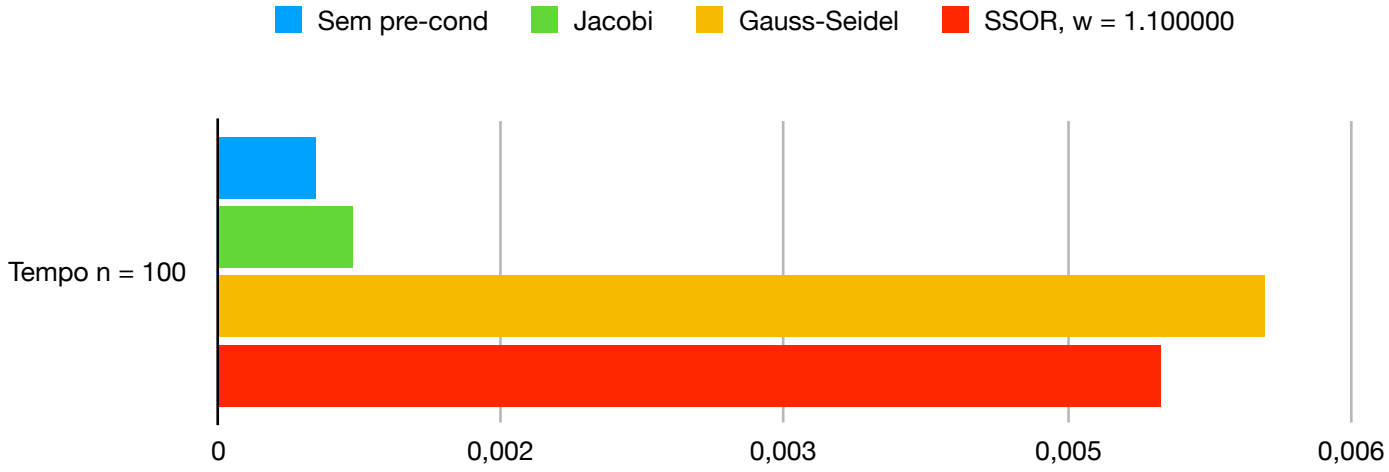
Resultados e Análise

Ao desenvolver nosso projeto e testamos como as configurações de pré-condicionamento afetam o desemprenho em relação ao número de iterações e tempo do método para cada um dos sistemas esparsos propostos fazendo um comparativo com o valor de n = 100, 1000 e 1000 e também fizemos avaliações sobre o efeito do sobre-relaxamento para valores de w ∈ [1.0, 2.0].

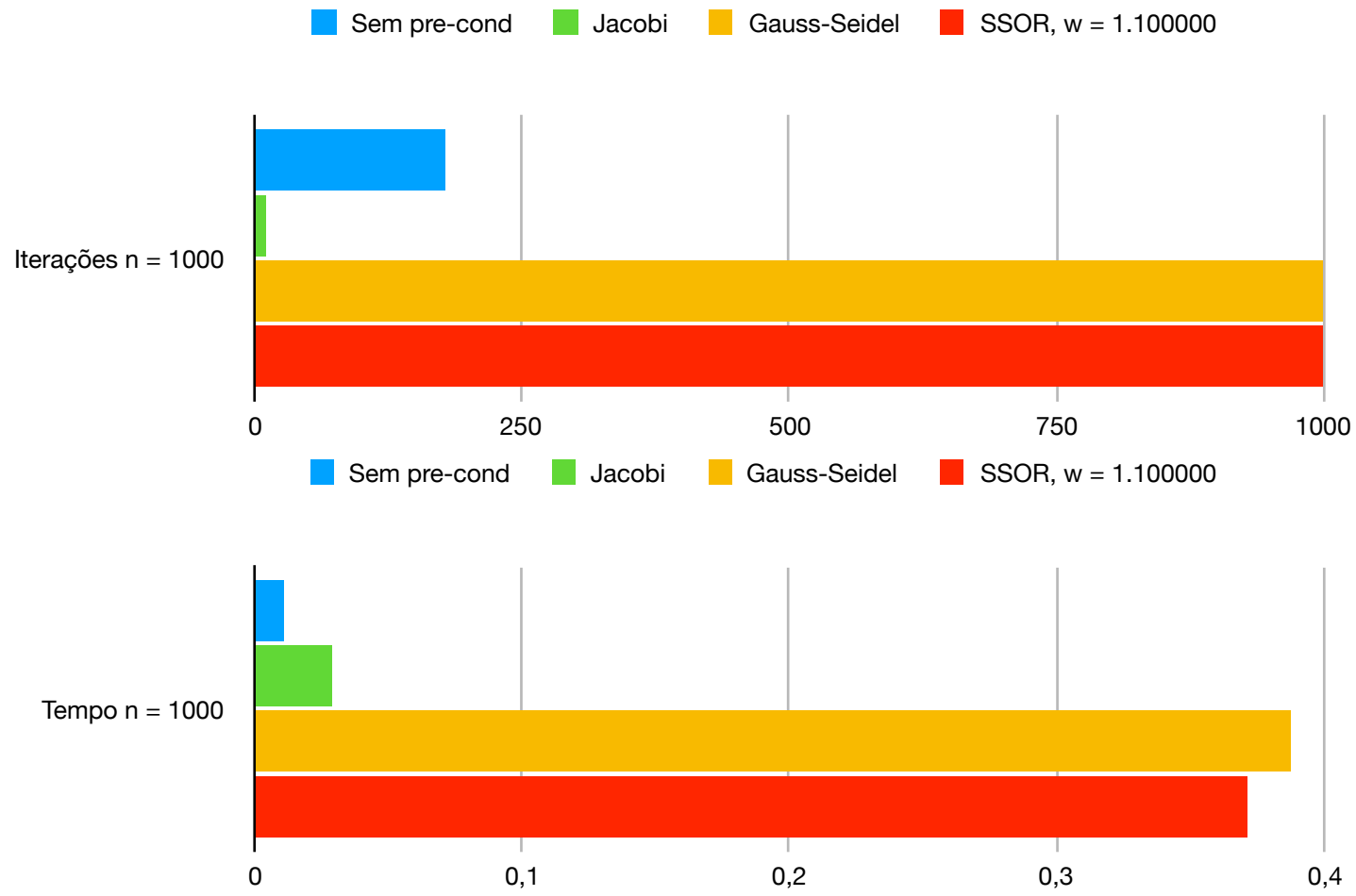
Para o primeiro sistema: $A(i, i) = i, A(i,i+1) = A(i+1,i) = 0.5$, $A(i, i + 2) = A(i + 2, i) = 0.5$, variamos o valor de n com n = 100, 1000 e 10000. Com o valor de n = 100 obtivemos os seguintes resultados:



Podemos observar que o método Jacobiano é o que faz menos iterações para convergir.

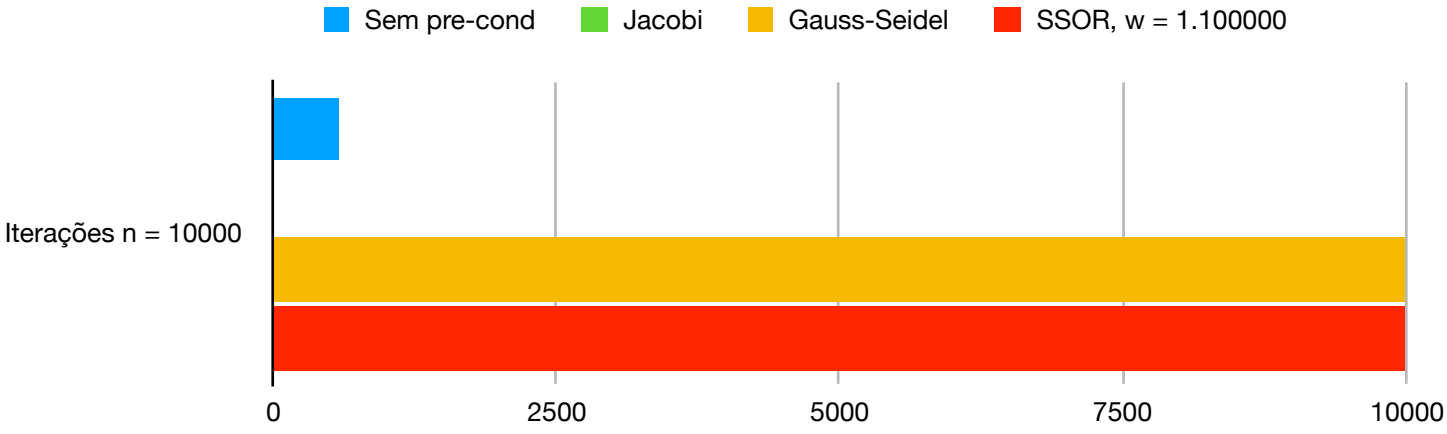


Em relação ao tempo de processamento, o método sem pré-condicionador é mais rápido que o Jacobiano embora precise fazer mais iterações como visto no gráfico anterior. Para n = 1000 os resultados foram semelhantes:

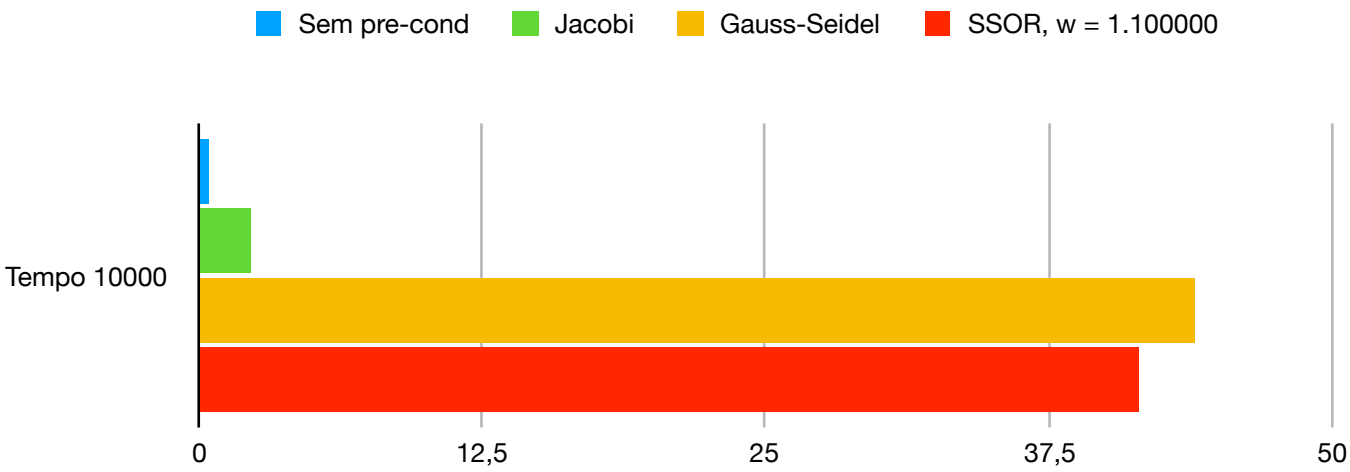


O método Jacobiano mais uma vez precisou fazer menos iterações que todos, mas novamente demorou um pouco mais do que o método sem pré-condicionador.

Já para n = 10000:



Podemos perceber que o método com pré-condicionador Jacobiano apresenta o melhor resultado em número de iterações mas com tempo um pouco mais elevado, e também podemos verificar que os métodos de Gauss-Seidel e SSOR são os mais custosos para a solução.



Para esse primeiro sistema, variamos o valor de w de 1 até 2 adicionando 0.1 a cada teste para ver como o pré-condicionador SSOR se comportava. Pudemos observar que mudando o valor de w, o número de iterações permanece o mesmo, porém existem variações no tempo de processamento de cada solução. Então dependendo do valor de w, o sobre-relaxamento pode ser mais eficiente do que o método de Gauss-Seidel.

Para o segundo sistema, podemos observar que os resultados foram bem semelhantes ao da primeira matriz, porém mais custosos, pois aumentamos o número de valores não nulos. Mais uma vez, o método com pré condicionador Jacobiano se mostrou o mais eficiente em termos de iterações, com uma pequena diferença de tempo de execução e relação ao método sem pré-condicionador.

Podemos concluir que o uso do método com pré-condicionador Jacobiano é o mais eficiente para solucionar esse problema. Quanto maior o número de n, menos iterações são necessárias para o método Jacobiano.