Resolução de Sistemas Esparsos

Daniel Cunha – 1512920

## Introdução

A resolução de sistemas lineares da forma *Ax = b* é um desafio corriqueiro em áreas da engenharia. Sendo assim o método mais frequente para resolução deste problema é o Método dos Gradiente Conjugados, altamente difundido por usar somente 4 tipos de operações: multiplicação de matriz por vetor, adição de vetores, multiplicação de escalar por vetor, produto escalar entre vetores. Limitando-se a essas operações, temos um custo computacional da ordem de *O(n2)* para cada iteração, sendo n a dimensão do sistema. Caso haja a conversão em k iterações, o custo total passa a ser O(kn2). Ponto interessante de ser analisado, pois caso k seja consideravelmente menor que n, se torna computacionalmente mais interessante que a outra técnica também utilizada conhecida como Eliminação de Gauss, porque teria sua complexidade menor que a técnica citada, que apresenta complexidade O(n3).

Fechando mais o escopo, temos a particularidade dos Sistemas Esparsos, que apresentam matrizes consideravelmente grandes, porém nem todas células possuem valores. Com essa situação, o objetivo é solucionar de forma mais eficientes esses sistemas, desconsiderando as células nulas e executando menos tarefas computacionais que são irrelevantes ao resultado final do sistema.

Para reduzir o custo por iteração, objetivo é fazer a operação de multiplicação de matriz por vetor processando apenas os elementos não nulos. Para atingir esse objetivo, foram definidos os tipos estruturados *matrix* e lista. Sendo que *matrix* possuiu um ponteiro para uma lista com todas as colunas não nulas da matriz tratada no sistema e também uma variável que armazena o tamanho da linha. Dessa forma foi possível desconstruir a matriz em sua formação original e reestrutura-la de forma mais eficiente visando o problema. A representação da matriz passa a ser um vetor de estruturas *matrix*, onde cada nó representa o início de cada linha, apontando para uma lista com os valores não nulos da linha representada.

Com a reestruturação da matriz foi necessário alterar as funções de multiplicação vetorial e matricial para que usassem a nova representação *matrix*.

O método dos Gradientes Conjugados na sua forma original, apresenta problemas de instabilidade numérica. Para amenizar o problema, usa-se pré-condicionadores M, transformando o sistema:

*Ax = b → M -1Ax = M -1 b*

O objetivo é analisar a eficiência do método descrito acima, utilizando os seguintes

pré-condicionadores:

• 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.

## Desenvolvimento

Para desenvolver as técnicas e atingir os resultados citados a cima foram usadas duas matrizes diferentes, porém semelhantes. Os primeiros testes foram realizados com o sistema *Ax = b*, sendo A preenchida nos modelos:

* A(i, i) = i
* A(i, i + 1) = 0.5
* A(i + 1, i) = 0.5
* A(i, i + 2) = 0.5
* A(i + 2, i) = 0.5

Preenchendo então somente a diagonal principal e seus três vizinhos diretos a cima da diagonal principal e a baixo, como mostra na figura 1.



Figura 1: Demosntração da matriz A1

Já para a execução de teste em outro sistema, a matriz A1 sofreu uma pequena alteração para observar como iria impactar em desempenho e precisão do algoritmo. Além dos modelos da matriz acima acrescentou-se também a seguinte regra:

* A(i, 2i) = 0.5
* A(2i, i) = 0.5

Aplicada somente quando i < , sendo n o número de linhas da matriz.

Com essa alteração, simplesmente expandimos a matriz, substituído por 0.5 em alguns campos e adicionando em outros como visto na figura 2.

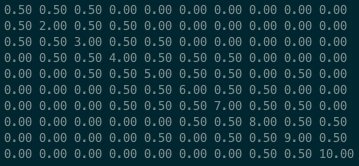


Figura 2: Demosntração da matriz A2

Além das matrizes, para cada sistema foi usado um vetor x com todos seus valores iguais a 1, que seria a resposta esperada, um vetor b com uma solução inicial, sendo todos seus valores iguais a 0 e um vetor com as respostas encontradas depois do processamento.

Para resolução do sistema e análise foi proposto a resolução por 4 métodos diferentes, como já citado acima a fim de comparação. Para cada método foi analisado o comportamento com cada uma das matrizes A1 e A2. Para avaliação e eficiência e precisão também foi levado em conta o número de iterações, o tempo gasto e o erro avaliado.

Para a execução do código na nova estrutura da matriz foram necessárias algumas alterações nos códigos originais de resolução por Gradiente Conjugado, Jacobi, Gauss-Seidel e SSQR. Os códigos que originalmente utilizavam matrizes em sua forma completa sofreram alterações para utilizar a nova estrutura *matrix.*

## Resultados e Análise

O resultado da execução do programa não o esperado, para alguns casos obtivemos resultados, porém em outros nem resultados foram expressos. Tentei descobrir o motivo do erro que retornou NaN no erro que vem de um NaN no vetor de resultado encontrado, que por sua vez foi gerado de algum erro anterior e acabou propagando o erro em cascata. Tentando encontrar a origem do erro com auxílio de um debbuger, porém ainda sim não foi possível encontrar a causa de tal comportamento.

Para os casos que obtive respostas, todos tiveram a mesma quantidade de iterações que a dimensão da matriz. A única circunstância que não obtive nenhum tipo de resposta foi com *n* = 10000, que mesmo passados 30 minutos de execução não foram encontradas respostas para nenhum dos métodos.

Apesar da tentativa de otimização do problema, as ações não se mostraram muito eficientes, pois o salto de consumo de recursos gastos como processamento e memória.

No geral, os erros obtidos foram relativamente altos, em casos chegando 0.36. Usando o n = 100 foram obtidos erro de 0.33 para o método que não faz uso de pré-condicionadores e 0.36 para o método de Jacobi. Para o pré-condicionador SSOR, sendo (Gauss-Seidel) não obtive resultado, retornando NaN no erro, impossibilitando a avaliação do método.

Com n = 1000, para ambas as matrizes, original e estendida, apenas os métodos de Jacobi e o normal atingiram um resultado. Passado um prazo também de 30 minutos, não obtive resposta para Gauss-Seidel e SSOR. Sendo Jacobi o melhor entre eles, atingindo 0.42 e sem pré-condicionador 0.43. Não houve também sequer diferença no valores para a matriz original e estendida, atingiram os mesmo resultados.

Para n = 10000 nenhum método conseguiu terminar a computação, mesmo insistindo por tempos acima de 60 minutos. Nota-se que possivelmente o problema não está no processamento de dados em sim, pois o tempo de espera foi bem acima do esperado. Não consegui atingir nenhuma resultado para n = 10000, não sendo possível notar também diferença entre a matriz original e a estendida.

Como análise final, o projeto teve falhas para a execução de todos os métodos em todas as situações, impossibilitando uma análise mais profunda de todos os métodos e suas variações. Mas era esperado que os pré-condicionadores SSOR e de Gauss-Seidel saíssem com melhores resultados em ambas a matrizes, principalmente quando fosse analisada a sobre-relaxação, com }.

Esperava também que o método de Jacobi apresentasse um erro menor que o método sem pré-condicionador para a matriz original, mesmo que para diferenças não muito expressivas para n = 100, mas nunca maior. Já no caso da matriz estendida, por não ser uma matriz diagonalmente forte, o erro maior para o método de Jacobi é plausível.

Em resumo:

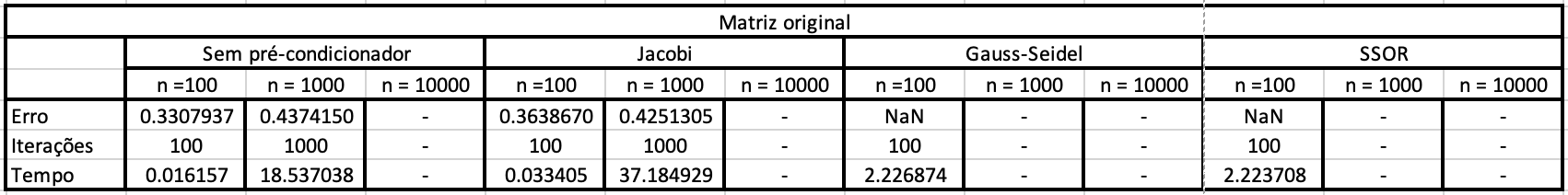


Figura 3: Tabela com os dados da matriz original

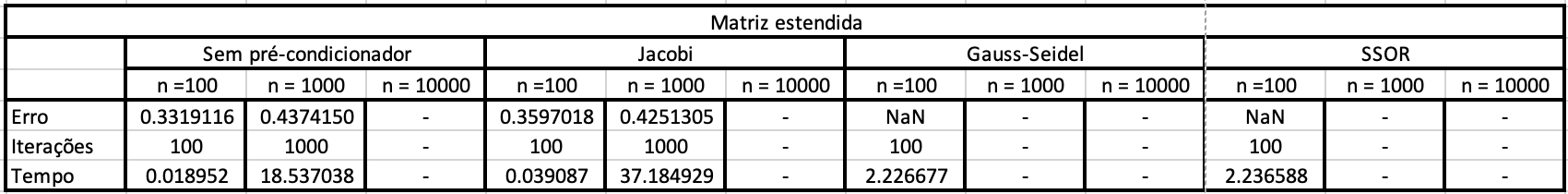


Figura 4: Tabela com os dados da matriz estendida

## Referência

1. Amauri Antunes Filho, Cesar Candido Xavier, “Solução de sistemas lineares esparsos utilizando CUDA. Uma comparação de desempenho em sistemas windows e linux”, http://pgsskroton.com.br/seer/index.php/rcext/article/viewFile/2255/2157, 01/12/2018