XXI ENMC
ENCONTRO NACIONAL DE MODELAGEM COMPUTACIONAL
IX ECTM
ENCONTRO DE CIÊNCIA E TECNOLOGIA DE MATERIAIS

08 a 11 de Outubro de 2018
Instituto Federal Fluminense
Búzios - RJ

COMPUTAÇÃO PARALELA DO MÉTODO DOS GRADIENTES CONJUGADOS APLICADO À RESOLUÇÃO DE EQUAÇÕES DIFERENCIAIS PARCIAIS: UM ESTUDO USANDO OPENMP

Fábio Silva de Souza[1] - fsilvadesouza@gmail.com
Breno Tiago Souza Mota[2] - brenotsm1@gmail.com
Luiz Nélio Henderson Guedes de Oliveira[3] - neliohenderson@gmail.com
[1]Universidade do Estado do Rio de Janeiro, Instituto Politécnico - Nova Friburgo, RJ, Brasil
[2]Universidade do Estado do Rio de Janeiro, Instituto Politécnico - Nova Friburgo, RJ, Brasil
[3]Universidade do Estado do Rio de Janeiro, Instituto Politécnico - Nova Friburgo, RJ, Brasil

Abstract. *Este trabalho almeja apresentar um método para resolução de equações diferenciais. Neste sentido, introduziremos o método dos gradientes conjugados através de seus fundamentos matemáticos. Em seguida, desenvolveremos um pseudo-código para a implementação computacional e, por fim, mostraremos os resultados de tal método aplicado a uma equação diferencial parcial de segunda ordem. Para isto, consideraremos o processo de paralelização do algorítmo apresentado. Assim, analisaremos o tempo computacional gasto para a obtenção da solução com uma dada precisão.*
Para tanto, utilizaremos o OpenMP. Esta interface de programação de aplicativos de memória compartilhada permite a subdivisão do domínio de discretização em regiões menores para que cada processador de um computador calcule as imagens desta sub-região, ao invés de exigir que os cômputos necessários sejam feitos por um único núcleo. Tal procedimento revela-se importante na diminuição do tempo computacional gasto para obtermos os resultados, conforme veremos no decorrer deste trabalho.

Keywords: *Métodos de Krylov, Equações Diferenciais Parciais, Gradiente Conjugado, Computação Paralela*

1. INTRODUÇÃO

Durante a discretização de equações diferenciais, é comum nos depararmos com a resolução de sistemas algébricos. Contudo, este pode ser um problema considerável se for preciso realizar operações com a matriz dos coeficientes na determinação de soluções.
O método dos gradientes conjugados é um processo iterativo utilizado para resolução de sistemas de equações algébricas. Tal técnica é oriunda dos métodos iterativos de Krylov, os quais

possuem um diferencial em relação aos métodos usuais visto que aqueles não envolvem o armazenamento de matrizes. Desta forma, estes métodos são computacionalmente mais viáveis. É preciso observar que, ao adotarmos uma das técnicas de Krylov, estaremos, na prática, trabalhando com um problema de otimização. De fato, dada uma equação matricial na forma:

$$Ax = b \tag{1}$$

onde x e b são vetores de ordem n x 1 e A é uma matriz de ordem n, $n > 0$; o que se almeja é minimizar o resíduo desta equação, isto é, procuraremos uma aproximação da solução x^* tal que o valor de

$$b - Ax^* \tag{2}$$

seja o menor possível.

2. MÉTODO DOS GRADIENTES CONJUGADOS

Nesta seção descreveremos alguns dos métodos de espaços de Krylov para equações lineares. Ao contrário dos métodos iterativos estacionários, este não usa uma matriz de iteração. Para desenvolver esta técnica, considere a Eq. (1) e o seguinte espaço vetorial:

$$\mathcal{K}_k =< r_0, Ar_0, ..., A^{k-1}r_0 >, k \geq 1 \tag{3}$$

O espaço $\mathcal{K}_k$ é chamado de k - ésimo subespaço de Krylov. Agora, sejam x_0 e r_0 a solução e resíduo iniciais da Eq. (1), respectivamente, e consideremos o espaço afim:

$$x_0 + \mathcal{K}_k \tag{4}$$

A sequência $\{r_k\}_{k\geq 0}$ denotará a sequência de residuos:

$$r_k = b - Ax_k \tag{5}$$

Suponhamos ainda que a matriz A, de ordem n é simétrica positiva-definida e definiremos a seguinte norma:

$$||x||_A = \sqrt{x^T Ax} \tag{6}$$

O método do gradiente conjugado consiste na minimização da seguinte função:

$$\phi(x) = \frac{1}{2}x^T Ax - x^T b \tag{7}$$

sobre $x_0 + \mathcal{K}_k$. Note que, se x^{**} minimiza a Eq. (7) então:

$$\bigtriangledown\phi(x^{**}) = Ax^{**} - b = 0 \tag{8}$$

e, portanto, $x^{**} = x^*$. Desta forma, minimizar a Eq. (7) é o mesmo que minimizar $||x^{**} - x^*||_A$. Para resolver tal problema, enunciaremos alguns resultados cuja demonstrações encontram-se

em (Kelley, 1995).

Lema 2.1: Seja $S \subset \mathbb{R}^n$. Se x_k minimiza ϕ sobre S então x_k minimiza $||x^* - x||_A = ||r||_{A^{-1}}$ sobre S.

Teorema 2.1: Sejam A uma matriz simétrica positiva-definida e $\{x_k\}$ as iterações do método do Gradiente Conjugado. Sejam ainda k um inteiro não - negativo dado e $\{p_k\}$ um polinômio de grau k tal que $p_k(0) = 1$. Então:

$$\frac{||x_k - x^*||_A}{||x_0 - x * ||_A} \leq \max_{z \in \sigma(A)} |p_k(z)| \tag{9}$$

onde $\sigma(A)$ é o conjunto dos autovalores de A.

Definição 2.1: O conjunto dos polinômios residuais de grau k é dado por:

$$P_k = \{p | p \text{ é um polinômio de grau } k \text{ e } p(0) = 1\} \tag{10}$$

Desta forma, o método dos gradientes conjugados pode ser visto como a análise de uma sequência de polinômios residuais sobre $\sigma(A)$. Além disso, este método pode ser visto como direto e, portanto, finito. Isto pode ser visto nos seguintes resultados:

Teorema 2.2: Seja A uma matriz simétrica positiva-definida de ordem n. Então, o método dos gradientes conjugados encontra a solução em n iterações.

Teorema 2.3: Seja A uma matriz simétrica positiva-definida com autovetores $\{u_i\}_{i=1,2,...,n}$. Seja b uma combinação linear de k autovetores de A:

$$b = \sum_{i=1}^{n} \gamma_l u_{il} \tag{11}$$

Então, o método dos gradientes conjugados para $Ax = b$ considerando $x_0 = 0$ converge após k iterações.

Teorema 2.4: Seja A uma matriz simétrica-positiva definida de ordem n. Suponha que existam exatamente k autovalores de A, $k \leq n^2$. Então, o método dos gradientes conjugados converge no máximo após k iterações.

É bom lembrarmos, neste momento, que um algoritmo computacional, em geral, não roda até encontrarmos uma solução exata. Neste sentido, é preciso estabelecer um critério de parada. Usualmente, pediremos que tenhamos resíduos muito pequenos. Isto significa que o algoritmo encerra com a seguinte condição:

$$||b - Ax_k|| \leq \eta ||b|| \tag{12}$$

para η suficientemente pequeno. Neste caso, $||.||$ denota a norma euclidiana. Observe que, neste ponto, devemos estabelecer uma relação entre a norma euclidiana e a norma definida na Eq. (6). Os lemas a seguir mostram como estas normas se relacionam:

Lema 2.2: Seja A uma matriz simétrica positiva-definida com autovalores $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$. Então, para todo $z \in \mathbb{R}^n$,

$$||A^{1/2}z|| = ||z||_A \tag{13}$$

e

$$\sqrt{\lambda_n}||z||_A \leq ||Az|| \leq \sqrt{\lambda_1}||z||_A \tag{14}$$

Lema 2.3:

$$\frac{||b||}{||r_0||}\frac{||b - Ax_k||}{||b||} = \frac{||b - Ax_k||}{||b - Ax_0||} \leq \sqrt{K_2(A)}\frac{||x_k - x^*||_A}{||x^* - x_0||_A} \tag{15}$$

e

$$\frac{||b - Ax_k||}{||b||} \leq \frac{\sqrt{K_2(A)}||r_0||}{||b||}\frac{||x_k - x^*||_A}{||x^* - x_0||_A} \tag{16}$$

onde $K_2(A)$ é chamado número de condição da matriz A.

Os lemas acima mostram que o número de condição da matriz A deve estar próximo de 1 para que o método convirja rapidamente. A busca por hipóteses que sejam necessárias para que tal número seja aproximadamente o número de condição ideal é chamado pré-condicionamento.

3. IMPLEMENTAÇÃO DO MÉTODO DOS GRADIENTES CONJUGADOS

O método dos gradientes conjugados depende do fato de que, uma vez conhecido x_k, ou $x_k = x^*$ ou uma direção p_{k+1} não nula deve ser encontrada de forma que $x_{k+1} = x_k + \alpha_{k+1}p_{k+1}$ para algum escalar α_{k+1}. O resultado a seguir mostra como tal direção pode ser encontrada.

Teorema 3.1: Seja A uma matriz simétrica positiva-definida e suponha que r_k é não nulo. Defina $p_0 = 0$. Então:

$$p_{k+1} = r_k + \beta_{k+1}p_k \text{ para algum } \beta_{k+1} \text{ e } k \geq 0 \tag{17}$$

Pode-se mostrar ainda que, para fins de implementação computacional, é interessante usar os parâmetros dados pelo lema a seguir.

Lema 3.1: Seja A uma matriz simétrica positiva-definida e suponha que r_k é não nulo. Então:

$$\alpha_{k+1} = \frac{||r_k||^2}{p_{k+1}^T A p_{k+1}} \tag{18}$$

e

$$\beta_{k+1} = \frac{||r_k||^2}{||r_{k-1}||^2} \tag{19}$$

Postos estes resultados todos, passemos a descrever um algoritmo para utilizarmos o método do gradiente conjugado:

1. $r = b - Ax, p_0 = ||r_k||^2, k = 1$
2. Faça enquanto $\sqrt{\rho_{k-1}} > \epsilon||b||$ e $k < kmax$
3. Se $k = 1$ então $p = r$; caso contrário $\beta = \rho_{k-1}/\rho_{k-2}$ e $p = r + \beta\rho$
4. $w = Ap$
5. $\alpha = \rho_{k-1}/p^T w$
6. $x = x + \alpha p$
7. $r = r - \alpha w$
8. $\rho_k = ||r||^2$
9. $k = k + 1$

Note que a matriz A não precisa ser construída ou armazenada pela memória do computador. Neste caso, precisaremos apenas implementar alguma rotina que calcule o produto de uma matriz por um vetor.

4. OPENMP

Em diversos problemas como, previsão do estado futuro da atmosfera para prever fenômenos meteorológicos (furacões, tempestades e etc), análise do comportamento de ações no mercado, mineração de dados para fins militares, além da acurácia nas soluções é primordial que o tempo gasto para se chegar aos resultados seja o menor possível. Outras inúmeras simulações numéricas também demandam um tempo computacional elevado devido a natureza complexa do problema matemático conforme visto em (Chandra, 2001).

Nesse contexto, a computação de alto desempenho fornece recursos que se destinam a otimizar o processamento de simulações numéricas. Uma das técnicas mais importantes é a computação em paralelo de clusters e supercomputadores conforme apresentando em (Chapman, 2008). Atualmente as metodologias mais empregadas para processamento em paralelo são as de memória distribuída e memória compartilhada.

O processamento em paralelo com arquitetura de memória compartilhada resume-se na divisão de tarefas entre os processadores que compartilham a mesma memória. Neste sentido, (Chapman, 2008) refere-se a máquinas com essa arquitetura pela sigla SMPs (*Symetric Multi-Processor*). As ferramentas de programação paralela mais comuns em SMPs são aquelas com threading explícito e as baseadas em diretivas de compilação. Entre os procedimentos que utilizam diretivas de compilação um dos padrões mais difundidos está o *OpenMP*, que especifica um conjunto de diretivas, funções e variáveis de ambiente para tornar o programa paralelo.

Segundo (Chapman, 2008), o *OpenMP* é uma interface de programação de aplicativos de memória compartilhada (API), não sendo uma nova linguagem de programação. Em vez disso, são instruções que podem ser adicionadas a programas sequenciais (*Fortram*, *C* ou *C++*) para descrever de que maneira o trabalho deve ser compartilhado entre as threads que serão executados em diferentes processadores (ou núcleos) e para solicitar acessos a dados compartilhados (se necessário). A métrica pela qual se mede o quanto o tempo despendido pode ser melhorado,

em comparação com o uso de um único processador, é chamada, segundo (Chandra, 2001), de Speedup e é calculado da seguinte forma:

$$S = \frac{T(1)}{T(N)} \tag{20}$$

onde S é o Speedup e $T(N)$ é o tempo gasto para N processadores.
O speedup de um programa usando múltiplos processadores em computação paralela é limitado pelo tempo da fração sequencial. Ao estudar o algoritmo do método Gradiente Conjugado é necessário identificar as partes que podem ser paralelizadas, nesse caso, com o uso do *OpenMP*. Dessa forma, todas as operações que envolvem vetores e matrizes podem ser paralelizadas com uso da diretiva de compilação: pragma omp parallel for. Essa diretiva atua sobre estruturas "for" (tanto simples como encadeadas). Por exemplo, no caso específico das operações $r = r - \alpha w$, $x = x + \alpha p$ do algoritmo, temos respectivamente os seguintes códigos:

```
#pragma omp parallel for shared(alfa)
for (int i = 0; i ¡ N; ++i)
{
r[i] = r[i] – alfa*w[i];
}

#pragma omp parallel for shared(alfa)
for (int i = 0; i ¡ N; ++i)
{
x[i] = x[i] + alfa*p[i];
}
```

5. Exemplo Numérico

Nesta seção, analisaremos um exemplo-teste para ilustrar como o método dos gradientes conjugados pode nos auxiliar a determinar soluções utilizando pouco tempo computacional. O algorítmo foi implementado em *C++* e o gráfico foi gerado em *Calc-LibreOffice*. Considere a seguinte equação diferencial:

$$-\nabla(cos(x,y)\nabla u) = f(x,y) \tag{21}$$

Neste caso, $f(x,y) = -10e^{x^{4.5}}[y(1-y)(a(x,y)sen(x) - b(x,y)cos(x)) + 2x(1-x)cos(x)]$, onde:

$$a(x,y) = 1 + 4.5x^{4.5} - 2x - 4.5x^{5.5} \tag{22}$$

E

$$b(x,y) = 24.75x^{3.5} + 20.25x^{8} - 20.25x^{9} - 33.75x^{4.5} - 2 \tag{23}$$

Além disso, as condições dadas para esta equação são:

$$u(x,0) = u(x,1) = u(0,y) = u(1,y) = 0, 0 < x, y < 1 \tag{24}$$

Em particular, a Eq. (21) admite como solução analítica a função:

$$u(x,y) = 10xy(1-x)(1-y)e^{x^{4.5}} \tag{25}$$

Para a discretização, utilizamos o método das diferenças finitas centradas, conforme descrito em (Subramaniam Gilat, 2008). Desta forma, obtivemos um sistema matricial da forma $Ax = b$, onde A é uma matriz simétrica positiva-definida, o que possibilita utilizar o método dos gradientes conjugados. Além disso, procuramos uma solução aproximada com erro inferior a 10^{-6} em relação à solução real. Neste caso, resolvemos o problema em 21 horas, ao usar uma malha suficientemente refinada. Registramos ainda que rodamos o algoritmo em um computador dotado de processadores Intel Xeon E5-2420.
Em um segundo momento, aplicamos no algoritmo o pré-condicionamento de Jacobi, conforme descrito em (Kelley, 1995). Novamente, resolvemos o mesmo problema com as mesmas condições. Porém, neste caso, obtivemos o resultado desejado em cerca de 23 horas. Devemos ressaltar que o pré-condicionamento garante a convergência do método sempre que o determinante da matriz A não for nulo. Desta forma, esta técnica costuma ser adotada na maioria dos casos. Posteriormente, aplicamos processos de paralelização em partes do algoritmo pré-condicionado usando o *OpenMP*, o que permitiu a ampliação da quantidade de processadores utilizados para resolver o problema posto. Mais precisamente, aplicamos diretivas de compilação nos passsos 4 a 7 do algoritmo descrito na Seção 3. Tal ferramenta pôde ser aplicada, visto que o algoritmo apresentado trata o método dos gradientes conjugados como um problema de otimização. Com isso, conseguimos determinar a mesma solução aproximada em cerca de sete horas e meia.

6. Análise de Resultados

A Figura 1 mostra a curva de speed-up, ou seja, a relação entre o tempo gasto para solucionar numericamente a Eq. (21) com um único processador e o tempo gasto com N processadores. Por exemplo, ao observarmos tal curva, verificamos que com seis processadores obtivemos o valor 2 para speed-up. Logo, utilizando apenas um processador gastamos o dobro do tempo necessário para determinarmos a mesma solução usando seis processadores. Analogamente, podemos dizer que com seis processadores reduzimos pela metade o tempo gasto em relação àquele dispensado com apenas um processador.
Deste modo, o gráfico exibido constata que, para reduzirmos o tempo de obtenção da solução em sessenta e seis porcento, são necessários vinte e dois processadores. A partir deste ponto, verificamos que o processo se torna obsoleto, isto é, a quantidade de processadores se torna irrelevante para a diminuição do tempo gasto.

Além disso, observemos que o algoritmo pré-condicionado gasta duas horas a mais para obter a solução em relação ao algoritmo sem pré-condicionamento. Porém, ao paralelizarmos aquele, reduzimos em aproximadamente um terço o tempo necessário para resolvermos numericamente a Eq. (21) ao compararmos com este. Perceba que, neste caso, estamos utilizando uma ferramenta que nos garante a convergência para a solução em um tempo consideravelmente menor.

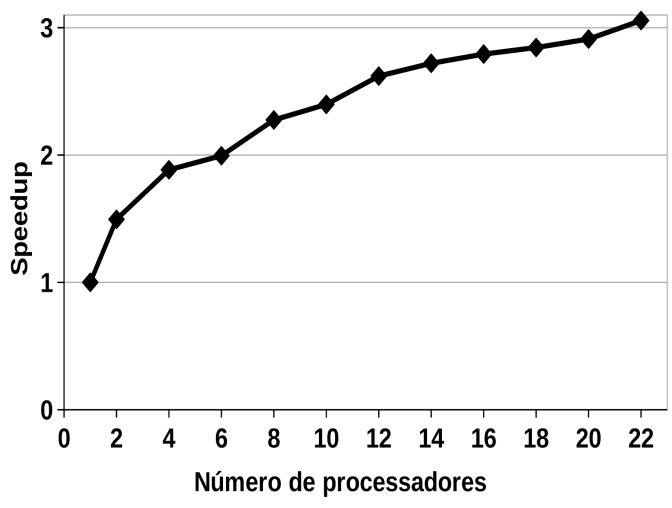

Figure 1- Curva de Speed Up.

7. CONCLUSÕES

Neste trabalho, apresentamos um algoritmo para método dos gradientes conjugados que dispensa o armazenamento de matrizes, sendo apenas necessária a inserção de uma rotina que faça o produto de uma matriz por um vetor. Isto já é um diferencial importante sobre outros métodos de resolução de sistemas lineares. Em geral, ao construir-se tal rotina, inserem-se apenas as equações que surgem no momento em que se discretiza a equação. Contudo, nem sempre temos garantia de que tal método irá convergir. Assim, muitos autores se dedicam a apresentar métodos de pré-condicionamento para garantir a convergência da solução com o custo do aumento do tempo computacional gasto. Porém, ao tratarmos o método dos gradientes conjugados como um problema de otimização, conseguimos paralelizar parte do algoritmo, o que nos permitiu observar uma redução significativa do tempo gasto. Mesmo tendo aplicado um pré-condicionamento simples, os resultados indicam que a computação paralela é uma ferramenta importante para tornar mais ágil a aplicação de tais técnicas. Sendo assim, podemos adaptá-la para outras formas de pré-condicionamento dependendo do problema adotado.

REFERENCES

Axelsson, O. (1994),*"Iterative Solution Methods"*, 4º ed., Cambridge University Press, Cambridge.
Chandra, R. (2001),*"Parallel Programming in OpenMP"*, 2º ed., Morgan Kaufmann, San Francisco.
Chapman, R., Jost, G., Van der Pas, R. (2008),*"Using OpenMP: Portable Shared Memory Parallel Programming"*, 1º ed., MIT Press, Cambridge.
Golub, G. H., Van Loan, C. F. (1996),*"Matrix Computations"*, 1º ed., The John Hopkins University Press, Baltimore.
Hestenes,M.R.; Stiefel, E. (1952), Methods of Conjugate Gradientes for Solving Linear Systems. Journal of Research of the National Bureau of Standards, 49, 409-436.
Kelley, C. T. (1995),*"Iterative Methods for Linear and Nonlinear Equations"*, 1º ed., Society for Industrial and Applied Mathematics, Philadelphia.
Subramaniam, V.; Gilat, A. (2008),*"Métodos Numéricos para Engenheiros e Cientistas: Uma introdução com aplicações usando o Matlab"*, 1º ed., Bookman, Porto Alegre.

TO THE RESOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS: A STUDY USING OPENMP

Abstract. *This work aims to present a method for solving differential equations. In this sense, we introduce the method of conjugate gradients through its mathematical foundations. Next, we will develop a pseudo-code for the computational implementation and, finally, we will show the results of such method applied to a second order partial differential equation. For this, we will consider the process of parallelization of the presented algorithm. Thus, we will analyze the time to obtain the solution with a given precision.*
For this, we will use OpenMP. This shared memory application programming interface allows subdivision of the discretization domain into smaller regions so that each processor in a computer computes the images of this sub-region, rather than requiring that the necessary computations be done by a single core. This procedure proves to be important in decreasing the computational time spent to obtain the results, as we will see in the course of this work.

Keywords: *Krylov Methods, Partial Differential Equations, Conjugate Gradient, Parallel Computing*