Universidade Federal Rural de Pernambuco
Licenciatura em Computação
Programação Paralela e Distribuída

# Projeto PPD

## Paralelização do Algoritmo de Jacobi

**Alunos:** **Jean Batista Ouriques**
**Joseir Alves de Oliveira**

# Paralelização do algoritmo sequencial Jacobi
## Projeto e Resumos

Jean Batista Ouriques[1], Joseir Alves de Oliveira[1]

[1]Licenciatura em Computação – Universidade Federal Rural de Pernambuco (UFRPE)
Recife - PE - Brazil

***Abstract*** *- the objective of this work is to comment regarding the iterative method to solve systems of linear equations - algorithm of Jacobi. We intend to describe its sequential algorithm, as also, we suggest a way to parallel this sequential implementation.*

***Resumo*** *- o objetivo deste trabalho é comentar a respeito do método iterativo para resolver sistemas de equações lineares - algoritmo de Jacobi. Nós pretendemos descrever o seu algoritmo sequencial, como também, nós sugerimos uma maneira de paralelizar essa implementação sequencial.*

### 1. Introdução

Os métodos numéricos para a resolução de sistemas lineares ***n*** x ***n*** podem ser divididos em dois grupos: métodos diretos e métodos iterativos.

***Métodos diretos*** - são aqueles que, a menos de erros de arredondamento, fornecem a solução exata do sistema linear, caso ela exista, após um número finito de operações.

***Métodos iterativos*** - são caracterizados pela busca da solução de um sistema linear através de etapas. Para cada etapa o método fornece uma aproximação para a solução do sistema. Quando o número de etapas tende a infinito a seqüência de aproximações tende para a solução exata do sistema.

Comentaremos nesse trabalho acerca da implementação sequencial de um dos métodos iterativos - o método Jacobi. Utilizaremos um sistema como exemplo para descrever o seu funcionamento.

### 2. Desenvolvimento

Uma implementação sequencial do método de Jacobi pode ser como segue:

| | |
|---|---|
| Entradas: | |
| **n** | (tamanho do sistema linear) |
| **ε** | (critério de convergência, tolerância) |
| a[1...n][1...n] | (coeficientes das equações lineares) |
| b[1...n] | (termos independentes) |
| Saída: | |
| ***x***[1...n] | (cálculos antigos do vetor solução) |
| Globais: | |
| new***x***[1...n] | (novos cálculos do vetor solução) |
| diff | (máxima escolha de alguns elementos da solução) |
| | |
| i, j | (índices dos loops) |

```
begin
   for  i = 1  to   n   do
      x[i] = b[i]/ a[i][i]
   endfor

   do            (Refina os valores de x até que o critério de convergência seja satisfeito)
      diff = 0
      for i = 1 to n do
         newx[i] = b[i]
         for j = 1 to  n  do
            if  j  ≠  i  then
               newx[i] = newx[i]  -  a[i][j]  x  x[j]
            endif
         endfor
         newx[i] = newx[i]/a[i][i]
      endfor
      for i = 1 to  n  do
         diff = max (diff, | x[i] - newx[i] | )
         x[i] = newx[i]
      endfor
   while diff > ε
end
```

Laço a ser paralelizado

Fig 1 - Algoritmo sequencial de Jacobi

Examinemos como o algoritmo funciona (apenas uma iteração) utilizando para isso o sistema abaixo:

$$\begin{cases} 2x_1 - x_2 = 1 \\ x_1 + 2x_2 = 3 \end{cases}$$

Entradas:

n = 2 $\quad a_{11} = 2 \quad a_{12} = -1 \quad b_1 = 1$

ε = 0.05 $\quad a_{21} = 1 \quad a_{22} = 2 \quad b_2 = 3$

```
begin
   de  i = 1  to   2   faça      (calculando os valores dos elementos de x)
      x[i] = b[i]/ a[i][i]
```

( p/ i = 1 )

$$x_1 = \frac{b_1}{a_{11}} \qquad \left( x_1 = \frac{1}{2} \right)$$

( p/ i = 2 )

$$x_2 = \frac{b_2}{a_{22}} \qquad \left( x_2 = \frac{3}{2} \right)$$

fim do laço

faça (refina os valores de ***x*** até que o critério de convergência seja satisfeito)

diff = 0 (variável diff começa valendo 0)
de i = 1 até **2** faça
( p/ i = 1 )
new$x_1$ = $b_1$ $\left( \text{new}x_1 = 1 \right)$

de j = 1 até **2** faça
( p/ j = 1 )
se j $\neq$ i então
new***x***[i] = new***x***[i] - a[i][j] x ***x***[j] ( i = j = 1 => não executa essa instrução )
fim se

( p/ j = 2 )
se j $\neq$ i então
new***x***[i] = new***x***[i] - a[i][j] x ***x***[j] ( i = 1 , j = 2 => executa essa instrução)

$\left( \text{new}x_1 = \text{new}x_1 - a_{12} \times x_2 \right)$ <=> $\left( \text{new}x_1 = \frac{5}{2} \right)$

fim se
fim laço j

( p/ i = 2 )
new$x_2$ = $b_2$ $\left( \text{new}x_2 = 3 \right)$

de j = 1 até **2** faça
( p/ j = 1 )
se j $\neq$ i então
new***x***[i] = new***x***[i] - a[i][j] x ***x***[j] ( i = 2 , j = 1 => executa essa instrução)

$\left( \text{new}x_2 = \text{new}x_2 - a_{21} \times x_1 \right)$ <=> $\left( \text{new}x_2 = \frac{5}{2} \right)$

fim se
( p/ j = 2 )
se j $\neq$ i então
new***x***[i] = new***x***[i] - a[i][j] x ***x***[j] ) ( i = j = 2 => não executa essa instrução )

fim se
fim laço j

fim laço i

de i = 1 to **2** faça

( p/ i = 1 )

diff = maior entre ( diff, | ***x***[i] - new***x***[i] | )

diff = maior entre ( $\underbrace{\text{diff}}_{0}$, | $\underbrace{x_1 - \text{new}x_1}$ | )

$$\left| \frac{1}{2} - \frac{5}{2} \right| = \left| -2 \right| = \mathbf{2}$$

( => diff = 2 )

$x_1 = \text{new}x_1 \qquad \left( x_1 = \frac{5}{2} \right)$

Encontra o valor atual de diff em cada iteração

( p/ i = 2 )

diff = maior entre ( diff, | ***x***[i] - new***x***[i] | )

diff = maior entre ( $\underbrace{\text{diff}}_{2}$, | $\underbrace{x_2 - \text{new}x_2}$ | )

$$\left| \frac{3}{2} - \frac{5}{2} \right| = \left| -1 \right| = \mathbf{1}$$

( => diff = 2 )

$x_2 = \text{new}x_2 \qquad \left( x_2 = \frac{5}{2} \right)$

fim do laço i

repete enquanto diff > **ε** ( critério de convergência, tolerância )

fim do algoritmo

Nesse exemplo o programa não para nessa primeira iteração. O valor atual da variável diff (diff = 2) não é igual nem é menor do que a tolerância declarada no início (**ε** = 0.05).

<u>Paralelização do algoritmo de Jacobi (Sequencial)</u>

Sugestão: paralelizar o laço **i** do algoritmo sequencial.
O processador **p0** é o responsável pela ativação dos demais processadores.

Os índices dos processadores corresponderão aos valores da variável **i** (laço i) do algoritmo sequencial, ou seja, **p**[1...n] (ver fig. 1 fl. 2).

**Processador central p0** inicia calculando os valores de ***x***[i]

```
begin
   for  i = 1  to   n   do
      x[i] = b[i]/ a[i][i]
   endfor
 do
      diff = 0
```

**p0** distribui a parte do código abaixo para todos os **n** processadores

```
    for all pi where  1  ⩽  i  ⩽   n
        newx[i] = b[i]
        for j = 1 to  n  do
           if  j  ≠  i  then
              newx[i] = newx[i]  -  a[i][j]  x   x[j]
           endif
        endfor
        newx[i] = newx[i]/a[i][i]
```

Cada processador **p*i*** executará essa parte do código individualmente e "ao mesmo tempo"

**p0** recebe todos os valores new***x***[i] de cada um dos processadores **p*i***

**p0** calcula a variável **diff** com base nos valores new***x***[i] de cada proc. **p*i***

```
      for i = 1 to  n  do
         diff = max (diff, | x[i] - newx[i] | )
         x[i] = newx[i]
      endfor
```

p0 verifica se **diff** > **ε** : caso afirmativo continua .../ caso negativo termina.

```
   while diff > ε    ( condição de parada )
end
```

**p0** armazena a solução que são os últimos valores assumidos por ***x*[i]**

## 3. Conclusão

Os métodos iterativos, quando há convergência garantida, são bastante vantajosos na resolução de sistemas de grande porte com a matriz de coeficientes do tipo esparço (grande proporção de zeros entre seus elementos). Esse trabalho procurou mostrar uma das formas de se conseguir a paralelização do algoritmo sequencial de Jacobi. O próximo passo seria a implementação numa linguagem de programação paralela, por exemplo PVM (Parallel Virtual Machine).

## 4. Referências

BARROSO, Leônidas Conceição; BARROSO, Magali Maria de Araújo; CAMPOS FILHO, Frederico Ferreira; DE CARVALHO, Márcio Luiz Bunte; MAIA, Miriam Lourenço . Cálculo Numérico ( Com aplicações), 2º edição. São Paulo, Ed Harbra ltda.1987

Quinn, Michael J.. Parallel computing: Theory and practice - Solving Linear Systems - Chapter 9, 239. Second Edition. McGraw-Hill, 1994