SOLUÇÃO DE SISTEMA ALGÉBRICO

O conjunto de equações algébricos

$$a_p \phi_p = \Sigma a_{nb} \phi_{nb} + b$$

dá origem a um sistema algébrico que pode ser representado pelo produto de uma matriz de coeficiente **A** e um vetor incógnita f igual a um vetor conhecido b

$$\mathbf{A}\, \phi = b$$

- Maior parte do tempo de uma simulação por volumes finitos, elementos finitos, diferenças finitas ou outro método numérico é gasto na resolução do sistema de equações obtido com a discretização.

  - Necessidade de métodos robustos e rápidos

  - Para resolver esse sistema pode-se utilizar
    - MÉTODOS DIRETOS
    - MÉTODOS ITERATIVOS

1. Métodos Diretos

Consiste em manipular a matriz de coeficientes **A** e obter de uma vez o campo da variável de interesse.

$$\mathbf{A}\,\phi = b \qquad\qquad \phi = \mathbf{A}^{-1}\,b$$

A solução exata (a menos de erros de truncamento do computador) é determinada após um número finito de operações

Para situações 1-D, a matriz de coeficientes **A** com muita freqüência só possui três diagonais diferentes de zero, sendo possível desenvolver algoritmos eficientes, que requerem pouco espaço de memória, como por exemplo o *Algoritmo de Thomas* (**TDMA**).

Para situações multi-dimensionais, em geral os algoritmos de inversão da matriz **A** de coeficientes são muito caros e requerem muito espaço de memória e tempo de execução.

- Requer mais memória de armazenamento
- Grande esforço computacional
- Mais robusto e Mais rápido

2. Métodos Iterativos

Fornece uma seqüência de soluções aproximadas que convergem quando o número de passos tende a infinito

Os problemas típicos de Fenômenos de transporte são problemas não lineares e portanto um procedimento iterativo deve ser utilizado. Já que o sistema de equações algébricas deverá ser resolvido diversas vezes atualizando os coeficientes, não vale a pena o esforço para inverter a matriz de coeficientes **A** e obter diretamente o campo de temperatura, pois o mesmo deverá ser corrigido, já que os coeficientes estarão errados.

- Menor necessidade de memória de armazenamento

- Problemas de convergência

MÉTODOS DIRETOS

SISTEMAS TRIANGULARES

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n-1} & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n-1} & u_{2n} \\ 0 & 0 & \cdots & u_{3n-1} & u_{3n} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

Se $u_{ii} \neq 0,\ i = 1,2,\ldots,n$ as incógnitas podem ser facilmente calculadas

linha n : $x_n = \dfrac{b_n}{u_{nn}}$;

linha n -1 : $u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n = b_{n-1} \quad \rightarrow \quad x_{n-1} = \dfrac{b_{n-1} - u_{n-1,n}x_n}{u_{n-1,n-1}}$;

$\vdots$

linha i : $x_i = \dfrac{b_i - \sum_{k=i+1}^{n} u_{i,k}x_k}{u_{ii}}$

RETROSUBSTITUIÇÃO

Se a matriz for triangular inferior:

$$\begin{bmatrix} l_{11} & 0 & \cdots & 0 & 0 \\ l_{21} & l_{22} & \cdots & 0 & 0 \\ l_{31} & l_{32} & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn-a} & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

A solução é calculada da seguinte forma:

$$\text{linha } 1: x_1 = \frac{b_1}{l_{11}};$$

$$\text{linha } 2: u_{2,1}x_1 + u_{2,2}x_2 = b_2 \quad \rightarrow \quad x_2 = \frac{b_2 - u_{2,1}x_1}{u_{22}};$$

$$\vdots$$

$$\text{linha i}: x_i = \frac{b_i - \sum_{k=i}^{i-1} l_{i,k}x_k}{l_{ii}}$$

SUBSTITUIÇÃO A FRENTE

NÚMERO DE OPERAÇÕES: $n + \sum_{i=1}^{n}(i-1) = n + \frac{1}{2}n(n-1) \approx \frac{1}{2}n^2$

Eliminação de Gauss

- Consiste em rearranjar a Matriz **A** em uma matriz triangular superior. A solução é obtida por substituição regressiva

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix}$$

- Exemplo do procedimento:

<u>Primeiro passo</u>

- linha 2 – linha 1 multiplicada por (a_{21} / a_{11})
- Procedendo de forma análoga para as outras linhas, elimina-se todos os coeficientes da primeira coluna com exceção da primeira linha.

$$m_{i1} = \frac{a_{i1}}{a_{11}} \; ; \quad i = 2,3,\ldots,n$$

$$a_{ij}^{(2)} = a_{ij} - m_{i1} a_{1j} \; ; \quad i = 2,3,\ldots,n$$

$$b_i^{(2)} = b_i - m_{i1} b_1 \; ; \quad i = 2,3,\ldots,n$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ \underbrace{\left(a_{21}-\frac{a_{21}}{a_{11}}a_{11}\right)}_{0} & \left(a_{22}-\frac{a_{21}}{a_{11}}a_{12}\right) & & \cdots & \left(a_{2n}-\frac{a_{21}}{a_{11}}a_{1n}\right) \\ \underbrace{\left(a_{31}-\frac{a_{31}}{a_{11}}a_{11}\right)}_{0} & \left(a_{32}-\frac{a_{31}}{a_{11}}a_{12}\right) & & & \left(a_{3n}-\frac{a_{31}}{a_{11}}a_{1n}\right) \\ \vdots & & & \cdots & \\ \underbrace{\left(a_{m1}-\frac{a_{m1}}{a_{11}}a_{11}\right)}_{0} & & & & \left(a_{mn}-\frac{a_{m1}}{a_{11}}a_{1n}\right) \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \left(b_2-\frac{a_{21}}{a_{11}}b_1\right) \\ \left(b_3-\frac{a_{31}}{a_{11}}b_1\right) \\ \vdots \\ \left(b_m-\frac{a_{n1}}{a_{11}}b_1\right) \end{bmatrix}$$

- A seguir, manipula-se as linhas de forma análoga para eliminar os coeficientes da segunda coluna com exceção da primeira e segunda linha e assim por diante

$$m_{i2} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}}\ ; \quad i = 3,4,\ldots,n$$

$$a_{ij}^{(3)} = a_{ij}^{(2)} - m_{i2}a_{2j}^{(2)}\ ; \quad i = 3,4,\ldots,n$$

$$b_i^{(3)} = b_i^{(2)} - m_{i2}b_2^{(2)}\ ; \quad i = 3,4,\ldots,n$$

- A matriz resultante será do seguinte tipo

$$\begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{pmatrix} = \begin{pmatrix} \otimes \\ \otimes \\ \otimes \\ \otimes \\ \otimes \end{pmatrix}$$

- ϕ pode ser obtido por substituição regressiva.

Os elementos $a_{11}^{(1)}, a_{22}^{(2)}, \ldots, a_{n-1,n-1}^{(n-1)}$ são denominados de *Pivots*

$\Rightarrow$ O lado direito do sistema de equações é modificado da mesma forma que os coeficientes das equações

$\Rightarrow$ Melhor tratar o sistema na forma matricial, com o lado direito do sistema sendo a coluna n+1 da matriz, conforme mostrado a seguir

$$a_{i,n+1}^{(k)} = b_i^{(k)},\ i = 1,2,\ldots,n$$

$$\left[\begin{array}{ccccc|c} a_{11} & a_{12} & \cdots & a_{1n-1} & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n-1} & a_{2n} & b_2 \\ a_{31} & a_{32} & \cdots & a_{3n-1} & a_{3n} & b_3 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn-a} & a_{nn} & b_n \end{array}\right]$$

ALGORÍTMO

ELIMINAÇÃO

For $k = 1, n-1$

  For $i = k+1, n$

    $$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

    For $j = k+1, n+1$

      $$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$$

    end

  end

end

RETROSUBSTITUIÇÃO

For $i = n,1,-1$

  $sum = 0$

  For $k = i+1, n$

    $sum = sum + a_{ik}^{(i)} * \phi_k$

  end

  $$\phi_i = \frac{a_{i,n+1}^{(i)} - sum}{a_{ii}^{(i)}}$$

end

NÚMERO DE OPERAÇÕES:

ELIMINAÇÃO

$$\sum_{k=1}^{n-1}(n-k)(n-k+1) \approx \tfrac{1}{3}n^3$$

RETROSUBSTITUIÇÃO

$$\sum_{i=1}^{n}(i-1) = \tfrac{1}{2}n(n-1) \approx \tfrac{1}{2}n^2$$

- O maior custo computacional ocorre no processo de eliminação
- Supor que o tempo de cada operação seja de 1 microsegundo $t = 10^{-6}\,s$

O tempo em segundos de cada parte do algoritmo é mostrado abaixo

n	***Eliminação***	***Retrosubstituição***
10	0.0050 s	0.0008 s
100	5 s	0.075 s
1000	5000 s	7.5 s

- número de multiplicações é proporcional a N^3
- erros de truncamento se acumulam para sistemas grandes

❑Caso o coeficiente da diagonal principal (pivot) vier a se anular, cuidados especiais precisam ser tomados:

❑ - uma solução é trocar a ordem das equações

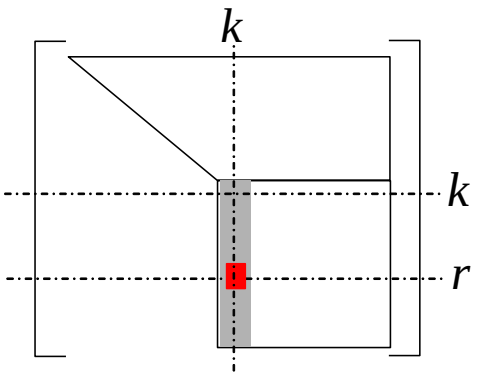
➹ Para evitar falha catastrófica (divisão por zero) ou resultados errados é necessário fazer uma escolha criteriosa dos PIVOTS usados na eliminação

PIVOTAMENTO PARCIAL
PIVOTAMENTO COMPLETO

PIVOTAMENTO PARCIAL

No passo k do processo de eliminação

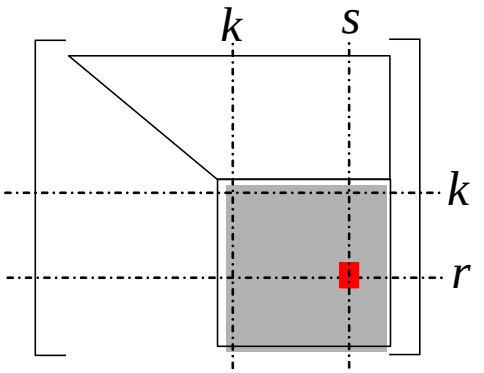
- Escolher r como o menor inteiro tal que
$$\left|a_{rk}^{(k)}\right| = \max\left|a_{ik}^{(k)}\right|, \; k \le i \le n$$
- Trocar linhas k e r



PIVOTAMENTO COMPLETO

No passo k do processo de eliminação

- Escolher r e s como os menores inteiros tal que
$$\left|a_{rs}^{(k)}\right| = \max\left|a_{ij}^{(k)}\right|, \; k \le i, j \le n$$
- Trocar linhas k e r, e colunas k e s



- A Eliminação Gaussiana deve ser feita sempre com PIVOTAMENTO para garantir estabilidade do método

- Na grande maioria dos casos, PIVOTAMENTO PARCIAL é suficiente e deve ser usada no lugar de PIVOTAMENTO COMPLETO

- PIVOTAMENTO COMPLETO não é muito usado devido ao grande tempo computacional gasto no processo de busca do pivot.

- PIVOTAMENTO não é necessário em dois casos particulares

  - *MATRIZ DIAGONAL DOMINANTE*

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad i = 1,2,\ldots,n.$$

  - *MATRIZ SIMÉTRICA E POSITIVA-DEFINIDA*

$$\mathbf{A}^T = \mathbf{A}\ (a_{ij} = a_{ji}) \quad \text{e} \quad \mathbf{x}^T \mathbf{A}\mathbf{x} > 0, \ \forall\ \mathbf{x} \neq \mathbf{0}$$

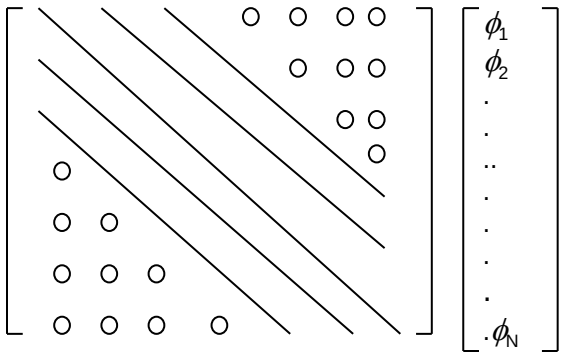
➢O método apresentado para resolver as matrizes com 3 diagonais adjacentes (Algoritmo de Thomas) é um caso particular do método acima, otimizado para não se efetuar contas desnecessárias com os coeficientes nulos. Para o caso de 5 diagonais adjacentes, também pode-se otimizar o método de eliminação de Gauss de forma análogo ao utilizado para a matriz tri-diadonal.

SOLUÇÃO DE SISTEMA ALGÉBRICO PENTA-DIAGONAL

$$a_i \varphi_i = b_i \varphi_{i+1} + c_i \varphi_{i-1} + e_i \varphi_{i+2} + f_i \varphi_{i-2} + d_i \quad \text{(I)}$$

$$\text{sendo } c_1 = 0 \;;\; f_1 = 0 \;;\; f_2 = 0$$
$$b_N = 0 \;;\; e_N = 0 \;;\; e_{N-1} = 0$$

A matriz de coeficientes penta-diagonal é



Para derivar este algoritmo deve-se proceder da mesma forma que procedemos para derivar o algoritmo para matrizes tri-diagonais.

- Vamos assumir que podemos determinar o valor de φ_i em função de φ_{i+1} e φ_{i+2}

$$\varphi_i = R_i\ \varphi_{i+2} + P_i\ \varphi_{i+1} + Q_i \qquad \text{(II)}$$

Rescrevemos a equação (II) para i-1 e i-2

$$\varphi_{i-1} = R_{i-1}\ \varphi_{i+1} + P_{i-1}\ \varphi_i + Q_{i-1} \qquad \text{(III)}$$

$$\varphi_{i-2} = R_{i-2}\ \varphi_i + P_{i-2}\ \varphi_{i-1} + Q_{i-2} \qquad \text{(IV)}$$

Vamos agora substituir (III) em (IV)

$$\varphi_{i-2} = R_{i-2}\ \varphi_i + P_{i-2}\ (R_{i-1}\ \varphi_{i+1} + P_{i-1}\ \varphi_i + Q_{i-1}) + Q_{i-2} \qquad \text{(V)}$$

Agora vamos substituir a equação (V) e (III) em (I)

$$a_i\ \varphi_i = b_i\ \varphi_{i+1} + c_i\ (R_{i-1}\ \varphi_{i+1} + P_{i-1}\ \varphi_i + Q_{i-1}) + e_i\ \varphi_{i+2} +$$
$$+ f_i\ (R_{i-2}\ \varphi_i + P_{i-2}\ (R_{i-1}\ \varphi_{i+1} + P_{i-1}\ \varphi_i + Q_{i-1}) + Q_{i-2}\ ) + d_i$$

Rearrumando a equação acima obtemos explicitando φ_i e comparando com a equação (II)

$$\varphi_i = R_i\ \varphi_{i+2} + P_i\ \varphi_{i+1} + Q_i$$

temos que

$$P_i = \frac{b_i + R_{i-1}\ (c_i + f_i\ P_{i-2})}{a_i - c_i\ P_{i-1} - f_i\ (P_{i-2}\ P_{i-1} + R_{i-2})}$$

$$Q_i = \frac{d_i + c_i\ Q_{i-1} + f_i\ (Q_{i-2} + P_{i-2}\ Q_{i-1})}{a_i - c_i\ P_{i-1} - f_i\ (P_{i-2}\ P_{i-1} + R_{i-2})}$$

$$R_i = \frac{e_i}{a_i - c_i\ P_{i-1} - f_i\ (P_{i-2}\ P_{i-1} + R_{i-2})}$$

Procedimento para matriz penta-diagonal

calcula-se $P_1 = \frac{b_1}{a_1}$; $Q_1 = \frac{d_1}{a_1}$; $R_1 = \frac{e_1}{a_1}$

$$P_2 = \frac{b_2 + c_2 R_1}{a_2 - c_2 P_1} \quad ; \quad Q_2 = \frac{d_2 + c_2 Q_1}{a_2 - c_2 P_1} \quad ; \quad R_2 = \frac{e_2}{a_2 - c_2 P_1}$$

Usar as relações recursivas para obter P_i ; Q_i e R_i *para* $3 \le i \le N$

$$P_i = \frac{b_i + R_{i-1}(c_i + f_i P_{i-2})}{a_i - c_i P_{i-1} - f_i (P_{i-2} P_{i-1} + R_{i-2})}$$

$$Q_i = \frac{d_i + c_i Q_{i-1} + f_i (Q_{i-2} + P_{i-2} Q_{i-1})}{a_i - c_i P_{i-1} - f_i (P_{i-2} P_{i-1} + R_{i-2})}$$

$$R_i = \frac{e_i}{a_i - c_i P_{i-1} - f_i (P_{i-2} P_{i-1} + R_{i-2})}$$

especificar $\varphi_N = Q_N$ e $\varphi_{N-1} = P_{N-1} \varphi_N + Q_{N-1}$

Usando $\varphi_i = R_i \varphi_{i+2} + P_i \varphi_{i+1} + Q_i$, obter φ_{N-2} ; ..... ; φ_2 ; φ_1

Decomposição LU

Resolver o sistema $\Rightarrow$ **[A] ϕ = b**

Todo matriz **não singular** pode ser decomposta como o produto de uma triangular inferior **L** e uma triangular superior **U**

[A] = [L] [U] então **[L] [U] ϕ = b**

$$\mathbf{A} = \mathbf{LU}; \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & & & & \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{pmatrix}; \quad \mathbf{U} = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & & & & \\ 0 & 0 & \cdots & 0 & u_{nn} \end{pmatrix}$$

Uma vez feita a decomposição, a solução do sistema fica reduzida a solução de dois sistemas triangulares:

Define-se um vetor auxiliar ***z***

$$[\mathbf{L}]\underbrace{[\mathbf{U}]\,\phi}_{z} = \mathbf{b}$$

Procedimento:

1º. Resolver **[L] z = b** por substituição progressiva, determinando **z**

$$\begin{pmatrix} \times & 0 & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{pmatrix} = \begin{pmatrix} \otimes \\ \otimes \\ \otimes \\ \otimes \\ \otimes \end{pmatrix}$$

2º. Resolver **[U]** ϕ **= z** por substituição regressiva, determinando

$$\begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{pmatrix} = \begin{pmatrix} \otimes \\ \otimes \\ \otimes \\ \otimes \\ \otimes \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 & 0 \\ l_{41} & l_{42} & l_{43} & 1 & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} \\ 0 & 0 & u_{33} & u_{34} & u_{35} \\ 0 & 0 & 0 & u_{44} & u_{45} \\ 0 & 0 & 0 & 0 & u_{55} \end{pmatrix}$$

$u_{11} = a_{11}$, $u_{12} = a_{12}$, ...

$l_{21} = \frac{a_{21}}{u_{11}}$, $l_{31} = \frac{a_{31}}{u_{11}}$, ...

$u_{22} = a_{22} - l_{21}\, u_{12}$, $u_{23} = a_{23} - l_{21}\, u_{13}$, ...

$l_{32} = \frac{a_{31} - l_{31}\ u_{12}}{u_{22}}$, ...

$$l_{i\,j} = \frac{a_{i\,j} - \sum_{k=1}^{j-1} l_{i\,k}\ u_{k\,j}}{u_{j\,j}} \quad para \quad i > j \qquad u_{i\,j} = a_{i\,j} - \sum_{k=1}^{i-1} l_{i\,k}\ u_{k\,j} \quad para \quad i \le j$$

abaixo da diagonal **acima da diagonal**

Observação:

1) Note que nem sempre podemos decompor uma matriz **A** em uma matriz triangular inferior e outra superior. Logo nem sempre o método de decomposição **L U** pode ser utilizado.

2) Assim como o método de Eliminação de Gauss também pode ter problemas, quando os coeficientes da diagonal principal são pequenos, e artifícios especiais devem ser introduzidos para evitar problemas

3) Os problemas típicos de fenômenos de transporte são problemas não lineares e portanto um procedimento iterativo deve ser utilizado. Já que o sistema de equações algébricas deverá ser resolvido diversas vezes atualizando os coeficientes, muitas vezes não vale a pena o esforço para inverter a matriz de coeficientes **A** e obter diretamente o campo de temperatura, pois o mesmo deverá ser corrigido, já que os coeficientes estarão errados.

Métodos Iterativos

Os métodos iterativos consistem em resolver o sistema algébrico utilizando duas etapas:

(a) prever,

(b) corrigir.

Os métodos iterativos podem ser:

- **ponto-a-ponto**: Os métodos iterativos ponto-a-ponto consistem em visitar um ponto nodal do domínio de cálculo de cada vez e estimar o valor da temperatura para o ponto nodal em função da temperatura estimada dos pontos vizinhos. Repete-se o procedimento para todos os pontos nodais. Como exemplo, pode-se citar o método de Gauss Seidel.
- **em blocos**: Os métodos iterativos em blocos consistem em resolver simultaneamente um conjunto de pontos nodais do domínio, em função da temperatura estimada dos pontos vizinhos. O procedimento é repetido até que todos os blocos do domínio tenham sido resolvidos. Como exemplo, pode-se citar o algoritmo TDMA linha por linha.

Método de Jacobi

Para $a_i \phi_i = \sum a_{nb} \phi_{nb} + b$ use $\phi_i = \dfrac{\sum a_{nb} \phi^*_{nb} + b}{a_i}$

onde ϕ^*_{nb} é o valor dos ϕ vizinhos da última iteração. Todos os pontos nodais são percorridos dessa maneira, atualizando os valores de ϕ. Este método apresenta uma convergência muito lenta, não sendo muito utilizado.

✓ **Exemplo:**

T1 = 0,5 T2 + 1,5 T2 = 0,25 T1 + 0,5

T_1	0 →	1,5	1,750	1,9375	.....	2,0
T_2	0 ↗	0,5	0,875	0,9375	.....	1,0

Converge lentamente para a solução correta

Método de Gauss-Seidel

Este é um algoritmo iterativo ponto-a-ponto para resolver o sistema de equações algébricas de situações uni- ou multi-dimensionais.

Para $a_i\phi_i = \sum a_{nb}\phi_{nb} + b$ use $\phi_i = \dfrac{\sum a_{nb}\phi^*_{nb} + b}{a_i}$

Onde ϕ^*_{nb} é o último valor disponível na memória do computador do ponto vizinho. Todos os pontos nodais são percorridos dessa maneira, atualizando os valores de ϕ.

✓ **Exemplo:**

T1 = 0,5 T2 + 1,5 T2 = 0,25 T1 + 0,5

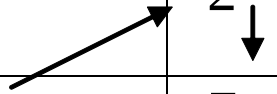
T_1	0	1,5	1,938	1,990	.....	2,0
T_2	0	0,875	0,984	0,998	.....	1,0

Converge para a solução correta

✓ **Exemplo:**

$T_1 = 4\ T_2 - 2$ $\qquad$ $T_2 = 2\ T_1 - 3$

T_1	0	-2	-30	-254	-2046	.....
T_2	0	-7	-63	-511	-4095	.....



a solução **diverge**.

Note que as equações são as mesmas que o exemplo anterior, porém rearranjadas

O Critério de Scarborough

- O método de Gauss-Seidel não converge sempre.
- As possibilidades de convergência do método de Gauss Seidel podem ser determinadas com referência ao *critério de Scarborough*, que é uma condição *suficiente* para a convergência do método de Gauss Seidel. É preciso que:

$$\frac{\sum |a_{nb}|}{|a_P|} \leq 1$$ para todas as equações e

< 1 para pelo menos uma equação

- **Exemplo:**

- ✓ As quatro regras básicas apresentadas, dão origem a equações de discretização que satisfazem o Critério de Scarborough, uma vez que os coeficientes são positivos e $ap \geq \Sigma a_{nb}$.

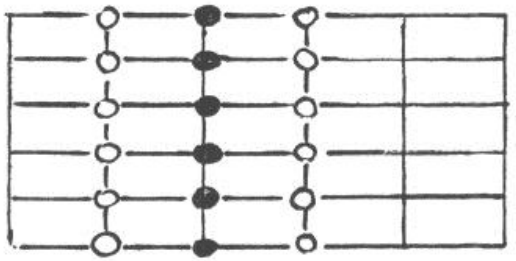
- ✓ Um S_P positivo pode fazer com que $ap < \Sigma a_{nb}$

- ✓ Um a_{nb} negativo pode levar a $ap = (\Sigma a_{nb}) < \Sigma |a_{nb}|$ violando o critério

- ✓ A condição de desigualdade do critério em pelo menos um ponto nodal é garantida pelas condições de contorno.

Algoritmo TDMA linha por linha

➢ Este é um algoritmo iterativo por blocos para resolver o sistema de equações algébricas de situações multi-dimensionais.



- Selecione uma linha da malha.
- Suponha que os valores de ϕ ao longo da linhas vizinhas são conhecidos, a partir de seus últimos valores.
- Resolva os valores de ϕ ao longo da linha selecionada pelo método direto TDMA.

$$a_p \phi_P = a_N \phi_N + a_S \phi_S + \underbrace{a_W \phi^*_W + a_E \phi^*_E + b}_{b^*}$$

- Repita este procedimento para todas as linhas em uma direção, repita, se desejado, para as linhas nas outras direções.

$$a_p \phi_P = a_W \phi_W + a_E \phi_E + \underbrace{a_N \phi^*_N + a_S \phi^*_S + b}_{b^*}$$

- No método de Gauss Seidel, as informações viajam um ponto nodal por iteração. No método TDMA linha-linha, a informação do contorno é transmitida de uma vez para o interior do domínio; consequentemente a convergência é mais rápida.

- Se os coeficientes em uma direção são muito maiores que aqueles nas outras direções, a TDMA ao longo da direção dos coeficientes dominantes levará a uma convergência muito mais rápida.

- Se $\Delta x >> \Delta y$ e $dx >> dy \rightarrow a_N$ e $a_S >>> a_W$ e a_E logo os valores ao longo da coluna terão um peso muito maior na obtenção da solução

$$a_p \phi_P = a_N \phi_N + a_S \phi_S + \underbrace{a_W \phi^*_W + a_E \phi^*_E + b}_{b^*}$$

- A direção da varredura também é importante. (Para escoamentos, a varredura deve ser de montante para jusante).

$$a_p \phi_P = a_W \phi_W + a_E \phi_E + \underbrace{a_N \phi^*_N + a_S \phi^*_S + b}_{b^*}$$

Método ADI

Alternating - Direction - Implicit

(Peaceman - Rachford) - 1955.

Este método adota uma formulação explícita em uma direção utilizando meio passo de tempo (Δt/2) ($a_P^o = \rho \Delta \forall / \Delta t$) e formulação implícita na outra, utilizando meio passo de tempo (Δt/2). Alterna-se as direções de uma iteração para o seguinte.

$$\left(a_E + a_W + 2\, a_P^o - S_P\, \Delta \forall \right) \phi_P^{n+1/2} = a_E\, \phi_E^{n+1/2} + a_W \phi_W^{n+1/2} + \underbrace{+ a_N\, \phi_N^n + a_S \phi_S^n + \left(2\, a_P^o + S_C\, \Delta \forall - a_N - a_S \right) \phi_P^n}_{b}$$

$$\left(a_N + a_S + 2\, a_P^o - S_P\, \Delta \forall \right) \phi_P^{n+1} = a_N\, \phi_N^{n+1} + a_S \phi_S^{n+1} + \underbrace{+ a_E\, \phi_E^{n+1/2} + a_W \phi_W^{n+1/2} + \left(2\, a_P^o + S_C\, \Delta \forall - a_E - a_W \right) \phi_P^{n+1/2}}_{b}$$

Método Fortemente Implícito

(SIP = Strongly Implicit Procedure, Stone 1968)

- Para ilustrar este método, considere um sistema de equações algébricas com

$$[A]\, u = C$$

[A] matriz de coeficientes ; u vetor desconhecido

C vetor conhecido

- Método Fortemente Implícito é uma técnica de fatorização da matriz

- O objetivo é substituir [A] por uma matriz [A + P] tal que a matriz modificada possa ser decomposta em matrizes triangulares, uma superior [U] e uma inferior [L].

Método Fortemente Implícito

✓ $[A] \Rightarrow [A + P]$ tal que $[A + P] = [L]\,[U]$

$$[A]\,u = C$$

$$[A + P]\,u^{n+1} = C + [P]\,u^{n}$$

$$[A + P] = [L]\,[U]$$

$$[L]\,[U]\,u^{n+1} = C + [P]u^{n}$$

definindo $V^{n+1} = [U]\,u^{n+1}$. Obtemos um algoritmo de 2 passos

1º passo $[L]\,V^{n+1} = C + [P]\,u^{n}$

2º passo $[U]\,u^{n+1} = V^{n+1}$

passo 1 consiste de uma substituição para frente e o passo 2 para trás.

repetir até convergir

Sobre-Relaxação e Sub-Relaxação

Se $ap\,\tilde{\phi}_p = \sum a_{nb}\,\phi_{nb} + b$

Definimos: $\phi_p = \alpha\,\tilde{\phi}_p + (1-\alpha)\,\phi_p^*$

onde ϕ_p^* é o valor de ϕ_p da iteração anterior, e α é o fator de relaxação.

$\alpha < 1$: sub-relaxação.

$\alpha > 1$: sobre-relaxação.

- A sobre-relaxação pode ser utilizada para acelerar a velocidade de convergência. Em geral, é utilizada junto com o método de Gauss-Seidel, o qual por ser um método iterativo ponto-a-ponto, apresenta taxas de convergência muito baixas. Este método é muitas vezes denominado de SOR (Sucessive Over Relaxation).

- Em problemas não lineares, é desejável reduzir as variações da variável dependente, então, sub-relaxação é recomendável.

- O fator de relaxação pode ser introduzido diretamente nas equações de discretização,

$$\phi_p = \alpha \frac{\sum a_{nb}\, \phi_{nb} + b}{a_p} + (1 - \alpha)\ \phi_p^*$$

- resultando em

$$\frac{a_p}{\alpha} \phi_p = \sum a_{nb}\, \phi_{nb} + b + (1 - \alpha) \frac{a_p}{\alpha} \phi_p^*$$

Observações:

$\Rightarrow$ Não existe regra geral para determinação de α.

$\Rightarrow$ Depende de:

- Natureza do problema número de pontos nodais
- espaçamento
- método iterativo utilizado
- etc.

$\Rightarrow$ Pode-se variar a durante o processo iterativo.

$\Rightarrow$ Pode-se também adotar a diferentes para cada ponto nodal (não é recomendado)

RELAXAÇÃO POR INÉRCIA

Rescreve-se a equação de discretização como

$$(a_P + i)\,\phi_P = \sum a_{nb}\,\phi_{nb} + b + i\,\phi_P^*$$

$i \Rightarrow$ inércia Se $i > 0 \Rightarrow$ sub-relaxação

Se $i < 0 \Rightarrow$ sobre-relaxação

$\Rightarrow$ Mesmos comentários feitos para α são válidos

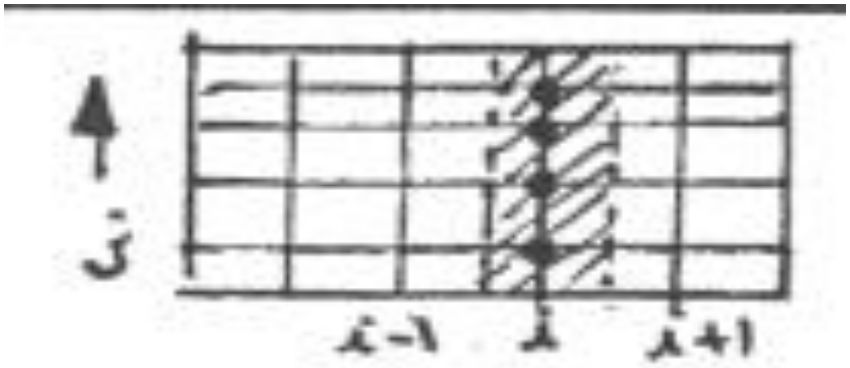
FORMULAÇÃO TRANSIENTE

$$\left(\sum a_{nb} + a_P^o - S_p\,\Delta\forall\right)\phi_P = \sum a_{nb}\,\phi_{nb} + S_c\,\Delta\forall + a_P^o\,\phi_P^o$$

a_P^o se comporta como a inércia i

$\Rightarrow$ Pode-se resolver um problema de regime permanente, utilizando a formulação transiente. Cada passo de tempo corresponde a uma iteração. Menor passo de tempo, maior sub-relaxação.

Algoritmo de Correção por Blocos



- Este é um algoritmo para acelerar a convergência. Consiste em resolver as equações de conservação para um conjunto de blocos de forma a fornecer o nível correto da temperatura mais rapidamente para o interior do domínio.

- Um domínio **3-D** é aproximado por um domínio **2-D** e um domínio **2-D** é aproximado por um domínio **1-D**. A solução do domínio de ordem inferior é mais fácil de ser obtida, fornecendo uma primeira estimativa para o domínio de ordem superior.

- Sem perda de generalidade, vamos considerar uma situação bi-dimensional.

$$a_{i,j}\,T_{i,j} = b_{i,j}T_{i+1,j} + c_{i,j}T_{i-1,j} + e_{i,j}T_{i,j+1} + f_{i,j}T_{i,j-1} + d_{i,j}$$

- ✓ Considere que

$$T_{i,j} = \overline{T}_i + T^*_{i,j}$$

- ✓ Substitua a definição acima na equação de discretização e some todos os blocos ao longo da coluna *i*.

$$\sum_j \left[a_{i,j}(\overline{T}_i + T^*_{i,j}) \right] = \sum_j \left[b_{i,j}\left(\overline{T}_{i+1} + T^*_{i+1,j} \right) \right] +$$
$$\sum_j \left[c_{i,j}\left(\overline{T}_{i-1} + T^*_{i-1,j} \right) \right] + \sum_j \left[e_{i,j}\left(\overline{T}_i + T^*_{i,j+1} \right) \right] +$$
$$\sum_j \left[f_{i,j}\left(\overline{T}_i + T^*_{i,j-1} \right) \right] + \sum_j \left[d_{i,j} \right]$$

- ✓ A equação resultante será

$$\bar{T}_i \sum_j \left[a_{i,j} - e_{i,j} - f_{i,j}\right] = \bar{T}_{i+1} \sum_j b_{i,j} + \bar{T}_{i-1} \sum_j c_{i,j} +$$

$$\sum_j \left[b_{i,j} T^*_{i+1,j} + c_{i,j} T^*_{i-1,j} + e_{i,j} T^*_{i,j+1} + f_{i,j} T^*_{i,j-1} + d_{i,j} - a_{i,j} T^*_{i,j} \right]$$

logo

$$BL\ \bar{T}_i = BLP\ \bar{T}_{i+1} + BLM\ \bar{T}_{i-1} + BLC$$

onde

$$BL = \sum_J \left[a_{i,j} - e_{i,j} - f_{i,j}\right]$$

$$BLP = \sum_j b_{i,j} \quad ; \quad BLM = \sum_j c_{i,j}$$

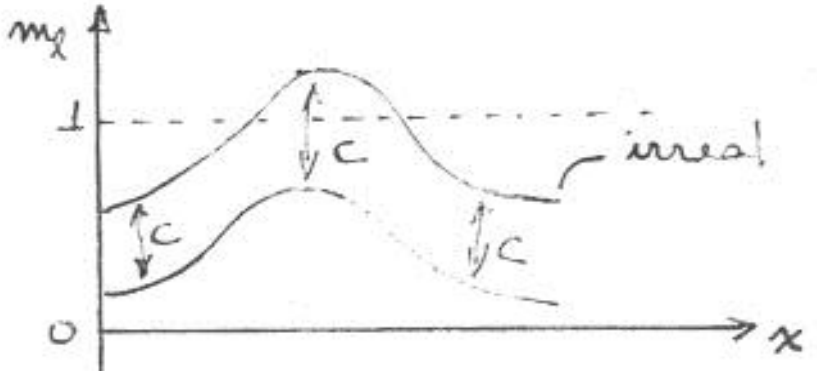
$$BLC = \sum \left[b_{i,j}\ T^*_{i+1,j} + c_{i,j} T^*_{i-1,j} + e_{i,j} T^*_{i,j+1} + f_{i,j} T^*_{i,j-1} + d_{i,j} - a_{i,j} T^*_{i,j} \right]$$

- A equação resultante para os blocos é **1-D**, podendo ser resolvida pelo método direto TDMA.
- Note que o termo BLC corresponde ao resíduo da equação original, logo, se a solução estiver convergida, o resíduo será nulo, a solução do sistema algébrico fornecerá valor nulo para , consequentemente, nenhuma nova correção será efetuada.
- O algoritmo de correção por blocos, só fornece uma correção média para o conjunto de blocos, portanto, após sua utilização o algoritmo TDMA linha por linha deverá ser utilizado para determinar a distribuição da temperatura ao longo dos blocos.
- Pode-se inverter a direção do algoritmo de correção por blocos.

$$T_{ij} = \overline{T}_j + T_{ij}^*$$

- Como o algoritmo de correção por blocos corrige da mesma maneira todos os volumes de controle dentro de um bloco, em algumas situações particulares, quando a variável dependente possui restrições físicas, este algoritmo poderá fornecer resultados indesejáveis, não devendo ser utilizado.

- Exemplo: fração em massa de uma espécie química, m_ℓ . De acordo com sua definição, temos que entre zero e um. Se durante o processo iterativo, está condição for violada, o processo irá divergir. O algoritmo de correção por bloco garante que a conservação para o bloco (conjunto de volumes de controle) seja satisfeita, mas um volume de controle pode estar mais errado do que outro. Logo, ao corrigir todos os volumes de controle de um bloco com a mesma constante, a condição acima poderá ser violada.



MÉTODO DE MULTIGRID

- Métodos iterativos ponto a ponto (Gauss Seidel) , ou bloco por bloco (TDMA linha por linha) removem rapidamente os erros locais (alta freqüência) da solução, porém os erros globais (baixa freqüência) são reduzidos a uma taxa inversamente proporcional ao tamanho da malha. Conseqüentemente, para um número elevado de nós, a taxa de redução do resíduo torna-se extremamente baixa.

- Técnicas de Multigrid permitem que o erro global seja tratado utilizando uma seqüência de malhas mais grosseiras. O método é baseado no princípio que o erro global (baixa freqüência) existente em um malha fina pode ser representado numa malha grosseira, onde este torna-se acessível como um erro local (de alta freqüência).

O conceito básico do Método de Multigrid

- Considere o conjunto de equações discretizadas lineares

$$A\,\phi_e + b = 0$$

- onde ϕ_e é a solução exata. Se ϕ é a solução aproximada, antes da solução convergir haverá um resíduo r

$$A\,\phi + b = r$$

- Deseja-se corrigir ψ com ϕ, tal que a solução exta seja dada por

$$\phi_e = \phi + \psi$$

- Substituindo na equação de discretização

$$A(\phi + \psi) + b = 0$$

$$A\psi + (A\phi + b) = 0$$

logo

$$A\psi + r = 0$$

- A qual é a equação para o termo de correção em função do operado *A* do nível fino original e do resíduo *r.*

- Assumindo que os erros locais (alta freqüência) foram suficientemente amortecidos pelo esquema de relaxação do nível fino, a correção ψ será suave e portanto mais eficientemente resolvida no próximo nível de malha grosseira.

Restrição e Prolongação

- Resolver as correções no nível grosseiro, requer transferir o resíduo do nível fino (restrição), computar as correções, e então transferir de volta do nível grosseiro (prolongação).

- Podemos escrever as equações para as correções do nível grosseiro ψ^H como

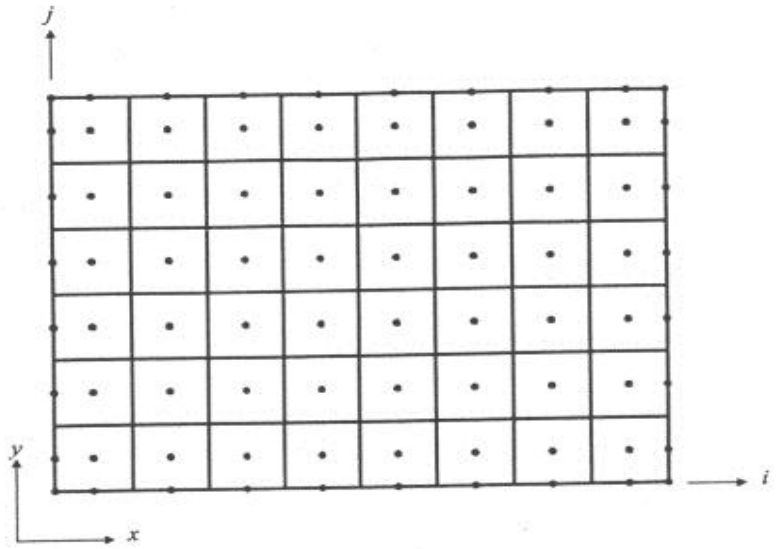
$$A^H \psi^H + R\, r = 0$$

- onde A^H é o operador no nível grosseiro e R é o operador de restrição responsável por transferir o resíduo do nível fino para o nível grosseiro. Após a solução da equação acima, a atualização da solução no nível fino é dada por

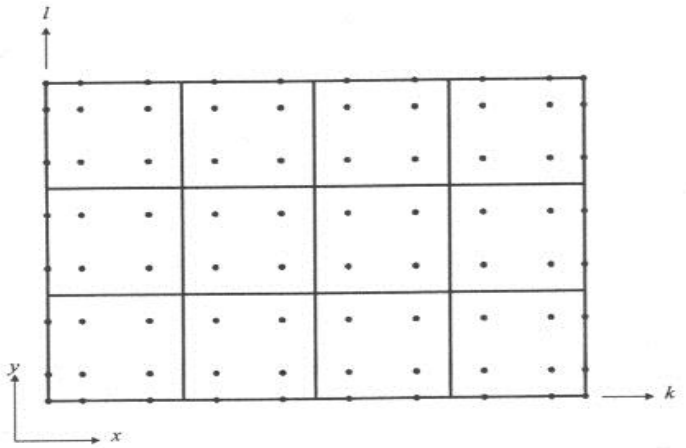
$$\phi^{new} = \phi + P\, \psi^H$$

- sendo P o operador de prolongação usado para transferir as correções do nível grosseiro para o nível fino.

MÉTODO DE MULTIGRID



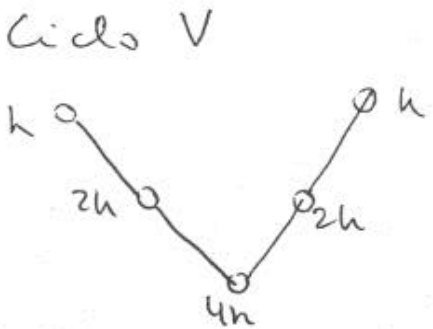
(a) Malha Fina original

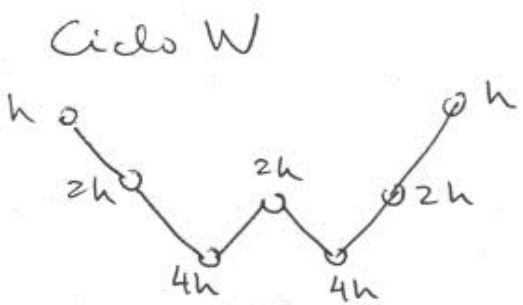


(b) Malha Grosseira

Figura 3.3 - Método "multigrid".

Ciclos do Multigrid





Pode-se ainda combinar o ciclos V e W (ciclo flexível)

Métodos Multigrid

$$ap\phi_p = \sum anb\ \phi_{nb} + b$$

$$a_{nb} = \text{dependem} \quad \text{h}$$

$\rightarrow$Multigrid tradicional

- resolve equação para h, $2h$, $4h$

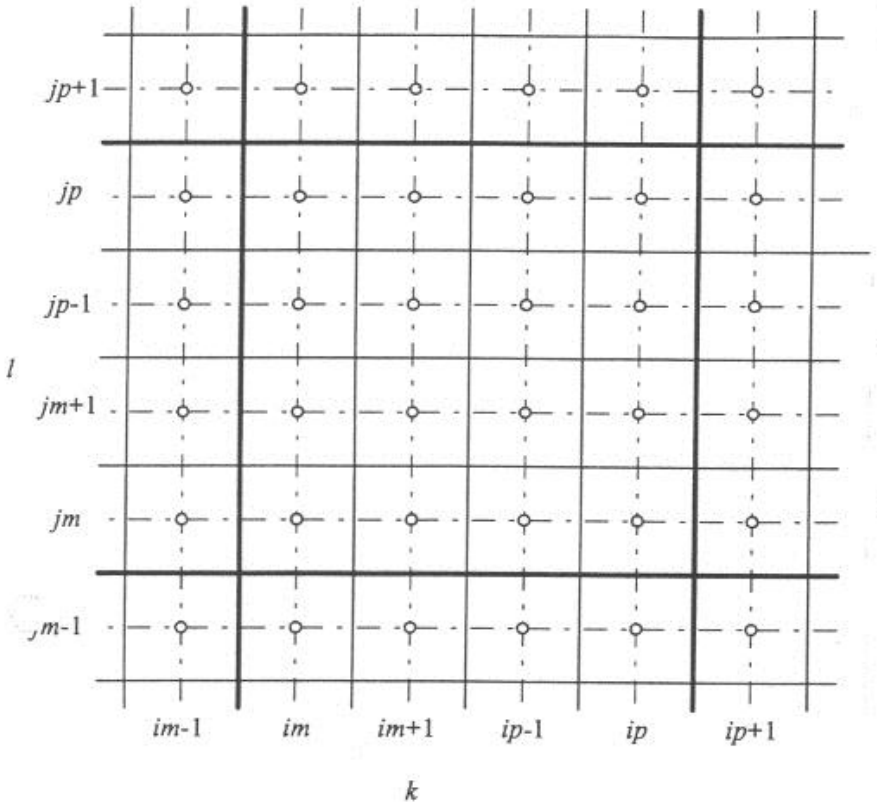
- interpola e extrapola as soluções para troca de níveis

$\rightarrow$Multigrid Correção Aditiva (Multigrid Algébrico)

- utiliza solução em malha grosseira para corrigir solução da malha mais fina

MULTIGRID DE CORREÇÃO ADITIVA (Hutchinson, Galpin, Raithby, 1988)

Relação entre malha fina e malha grosseira.



$$T_{ij} = T_{ij}^{*} + \delta_{k\ell}$$

$$\hat{a}_{kl}^{P}\delta_{kl} = \hat{a}_{kl}^{E}\delta_{k+1,\,l} + \hat{a}_{kl}^{W}\delta_{k\text{-}l,\,\ell} + \hat{a}_{kl}^{N}\delta_{k,\,\ell+1}\,\hat{a}_{k,\,\ell+1}^{s}\,\delta_{k,\,\ell\text{-}1} + \hat{b}_{kl}$$

$$\hat{a}_{kl}^{E} = \sum_{j} a_{ipj}^{E}\,,\ j = jm,jp \qquad \hat{a}_{kl}^{W} = \sum_{j} a_{imj}^{W}\,,\ j = jm,jp$$

$$\hat{a}_{kl}^{N} = \sum_{i} a_{ijp}^{N}\,,\ i = im,\ ip \qquad \hat{a}_{kl}^{S} = \sum_{i} a_{ijm}^{S}\,,\ i = im,\ ip$$

$$\hat{a}_{k\,l}^{P} = \sum_{j}\sum_{i} a_{ij}^{P} - \mathrm{SUM1} - \mathrm{SUM2} - \mathrm{SUM3} - \mathrm{SUM4},\ i = im,\ ip,\ j = jm,\ jp$$

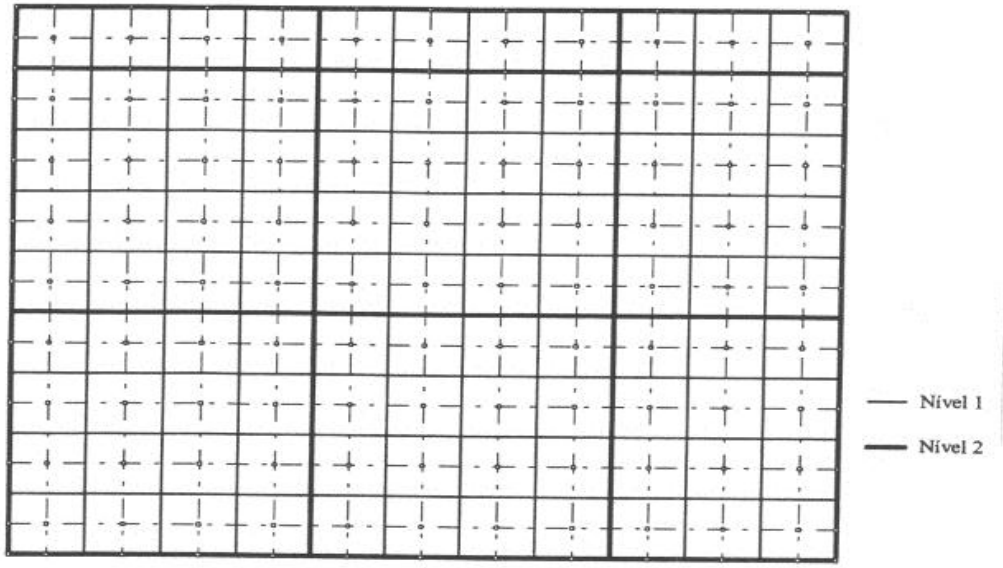
$$SUM1 = \sum_{j}\sum_{i} a_{ij}^{E},\ i = im, ip-1,\ j = jm, jp$$

$$SUM2 = \sum_{j}\sum_{i} a_{ij}^{W},\ i = im+1,\ ip,\ j = jm, jp$$

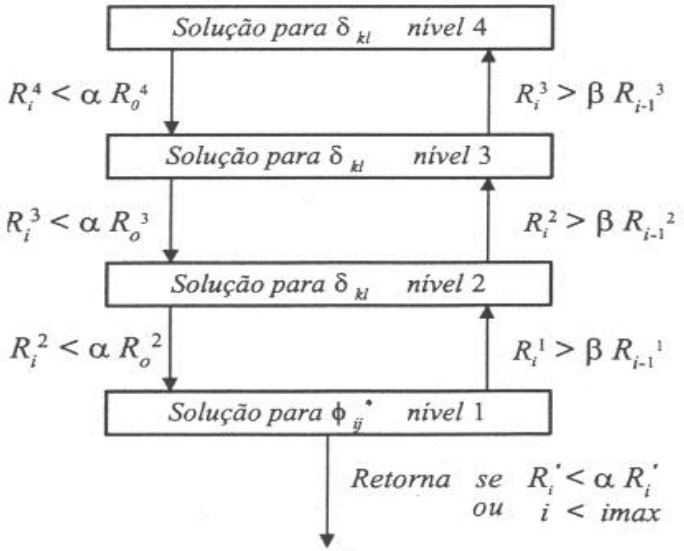
$$SUM3 = \sum_{j}\sum_{i} a_{ij}^{N},\ i = im,\ ip,\ j = jm, jp-1$$

$$SUM4 = \sum_{j}\sum_{i} a_{ij}^{S},\ i = im,\ ip,\ j = jm+1, jp$$

$$\hat{b}_{kl} = \sum_{j}\sum_{i} (b_{ij} + a_{ij}^{E}\phi^{*}_{i+1\,j} + a_{ij}^{W}\phi^{*}_{i-1\,j} + a_{ij}^{N}\phi^{*}_{ij+1} + a_{ij}^{S}\phi^{*}_{ij-1} - a_{ij}^{P}\phi^{*}_{ij})$$



- Nível 1 não contém um múltiplo integral de 4 volumes de controles.



- Procedimento lógico de controle de correção no "multigrid".

$$R_i > \beta R_{i-1}$$

$$R_i = \sum \left| Res_{ij} \right|$$

$$Res_{ij} = a_{nb}\,\varphi_{nb} + b - a_p \varphi_P$$