

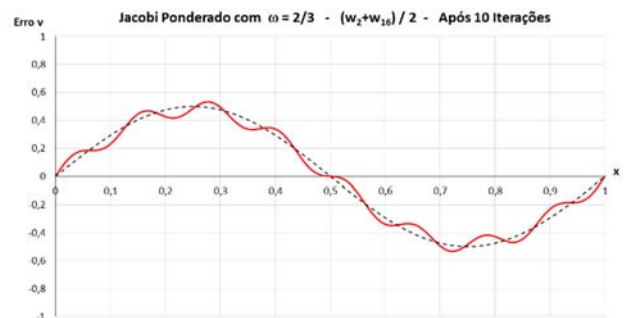
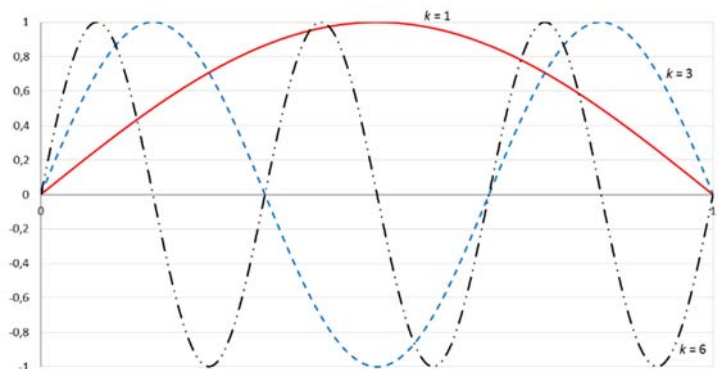
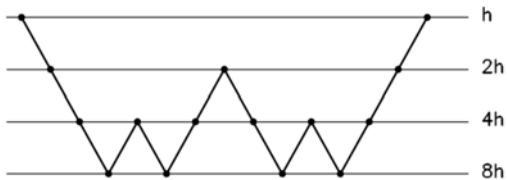
UNIVERSIDADE FEDERAL DO PARANÁ
PPGMNE

Método Multigrid

MNE-756

Professor Marcio Augusto Villela Pinto

NOTAS DE AULA



Fevereiro 2018

CAPÍTULO 1

PROBLEMAS-MODELO

Considere o problema de valor de contorno que descreve a distribuição de temperatura em regime permanente ao longo da barra uniforme:

$$\begin{cases} -u''(x) + \sigma u(x) = f(x), & 0 < x < 1, \quad \sigma \geq 0 \\ u(0) = u(1) = 0 \end{cases} \quad (1.1)$$

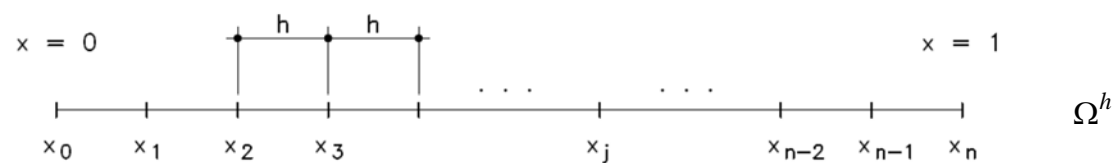
Observação: se $\sigma = 0$, temos a equação de Poisson. Se ainda, $f = 0$, temos a equação de Laplace.

Se $\sigma \neq 0$, temos a equação de Helmholtz.

Vamos considerar o método numérico “Método das Diferenças Finitas”.

O domínio do problema, $\{x: 0 \leq x \leq 1\}$ é particionado em n subintervalos, introduzindo a malha

$$x_j = j h \text{ com } h = \frac{1}{n}.$$



Considere v_j uma aproximação para a solução exata $u(x_j)$.

Considere $\mathbf{v} = (v_1, v_2, \dots, v_{n-1})^T$. Estas componentes satisfazem as $n-1$ equações lineares:

$$\begin{cases} \frac{-v_{j-1} + 2v_j - v_{j+1}}{h^2} + \sigma v_j = f(x_j), & 1 \leq j \leq n-1 \\ v(0) = v(n) = 0 \end{cases} \quad (1.2)$$

Definindo $\mathbf{f} = (f(x_1), f(x_2), \dots, f(x_{n-1}))^T = (f_1, f_2, \dots, f_{n-1})^T$, este sistema de equações lineares pode ser escrito na forma matricial.

Vejamos:

$$j=1 \quad \frac{1}{h^2}(-v_0 + 2v_1 - v_2) + \sigma v_1 = f_1 \quad (v_0 = 0)$$

$$j=2 \quad \frac{1}{h^2}(-v_1 + 2v_2 - v_3) + \sigma v_2 = f_2$$

...

$$j=n-1 \quad \frac{1}{h^2}(-v_{n-2} + 2v_{n-1} - v_n) + \sigma v_{n-1} = f_{n-1} \quad (v_n = 0)$$

$$\frac{1}{h^2} \begin{bmatrix} 2 + \sigma h^2 & -1 & & & & \\ -1 & 2 + \sigma h^2 & -1 & & & \\ & -1 & 2 + \sigma h^2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 + \sigma h^2 & -1 \\ & & & & -1 & 2 + \sigma h^2 \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{Bmatrix}$$

Ou ainda, $\mathbf{A} \mathbf{v} = \mathbf{f}$, onde \mathbf{A} é uma matriz $(n-1) \times (n-1)$, tridiagonal, simétrica e definida positiva.

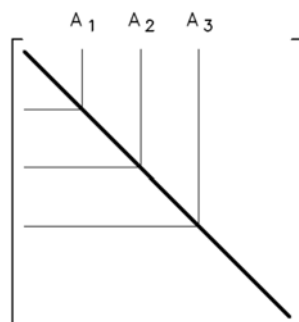
Observação: As matrizes geradas pela discretização de problemas de contorno, em geral têm propriedades desejáveis para muitos métodos numéricos. Ex.: simétrica, diagonal dominante, definida positiva, M-matriz, etc.

Lembrete:

Definição: Diagonal dominante: $\sum_{j \neq i}^n |a_{ij}| \leq |a_{ii}|, \quad 1 \leq i \leq n$

Definição: Definida positiva: $\mathbf{u}^T \mathbf{A} \mathbf{u} > 0, \quad \forall \mathbf{u} \neq \mathbf{0}$

Propriedade: Para matrizes pequenas: $\det \mathbf{A}_k > 0, \quad \forall k$, onde \mathbf{A}_k é submatriz líder de ordem k .



Teorema: Toda matriz simétrica definida positiva possui autovalores reais e positivos.

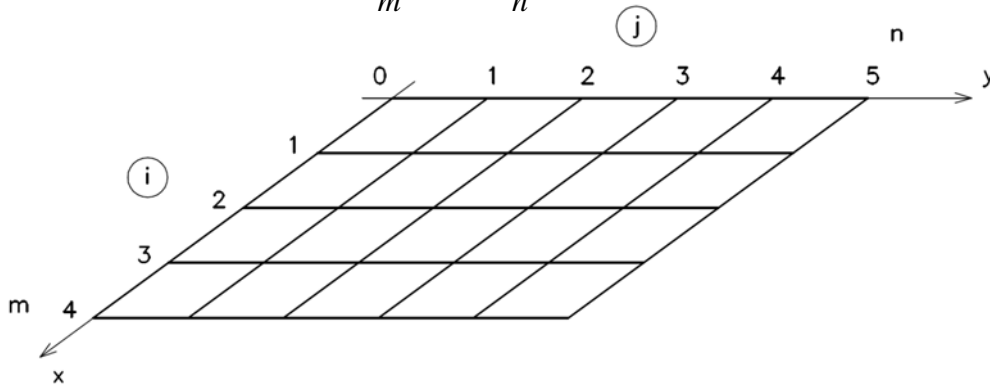
Teorema: Se a matriz é simétrica, diagonal dominante e os elementos da diagonal são positivos, então é simétrica definida positiva.

Definição: M-matriz: matriz simétrica definida positiva com diagonal positiva, e negativa fora da diagonal principal.

Vamos formular o problema (1.1) na versão bidimensional:

$$\begin{cases} -u_{xx} - u_{yy} + \sigma u = f(x, y), & 0 < x < 1, \quad 0 < y < 1, \quad \sigma \geq 0 \\ u = 0 & \text{na fronteira do quadrado unitário} \end{cases} \quad (1.3)$$

$(x_i, y_j) = (i h_x, j h_y)$ com $h_x = \frac{1}{m}$ e $h_y = \frac{1}{n}$, são os pontos da malha.



Isto nos leva ao sistema de equações lineares:

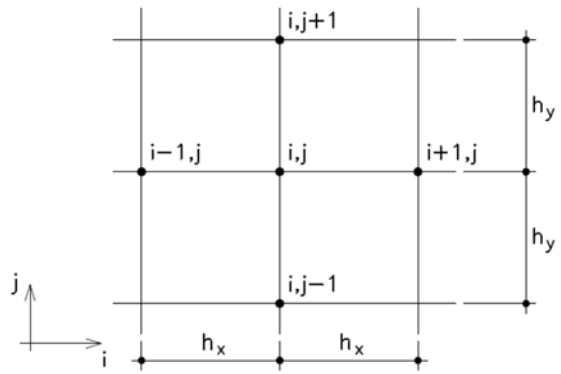
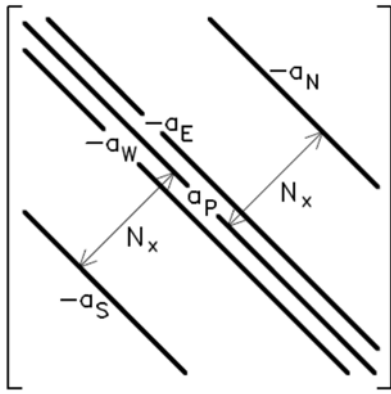
$$\begin{cases} \frac{-v_{i-1,j} + 2v_{ij} - v_{i+1,j}}{h_x^2} + \frac{-v_{i,j-1} + 2v_{ij} - v_{i,j+1}}{h_y^2} + \sigma v_{ij} = f_{ij} & 1 \leq i \leq m-1 ; 1 \leq j \leq n-1 \\ v_{i0} = v_{in} = v_{0j} = v_{mj} = 0 \end{cases} \quad (1.4)$$

v_{ij} é a aproximação da solução exata $u(x_i, y_j)$ e $f_{ij} = f(x_i, y_j)$.

Existem $(m-1) \times (n-1)$ pontos interiores (incógnitas) no problema.

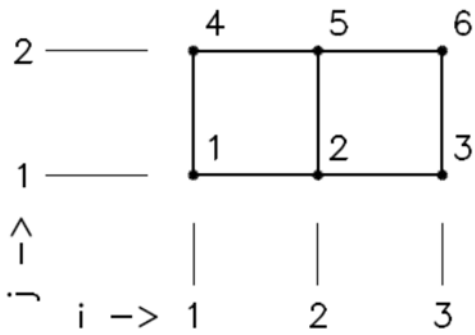
Considere a ordem lexicográfica $k = i + m(j-1)$.

Assim, temos, $A v = f$, onde A é uma matriz $(m-1)(n-1) \times (m-1)(n-1)$, pentadiagonal, simétrica, definida positiva.



Exemplo: ordem lexicográfica.

Se $m = 3$ e $n = 2$:



(i, j)

$$(1, 1) \rightarrow k = 1 + 3(1 - 1) \rightarrow k = 1$$

$$(2, 1) \rightarrow k = 2 + 3(1 - 1) \rightarrow k = 2$$

$$(3, 1) \rightarrow k = 3 + 3(1 - 1) \rightarrow k = 3$$

$$(1, 2) \rightarrow k = 1 + 3(2 - 1) \rightarrow k = 4$$

$$(2, 2) \rightarrow k = 2 + 3(2 - 1) \rightarrow k = 5$$

$$(3, 2) \rightarrow k = 3 + 3(2 - 1) \rightarrow k = 6$$

Notação estêncil

Para o problema 1D:

$$A = \frac{1}{h^2} \begin{pmatrix} -1 & 2 + \sigma h^2 & -1 \end{pmatrix}$$

Para o problema 2D:

$$A = \frac{1}{h^2} \begin{pmatrix} & -1 & \\ -1 & 4 + \sigma h^2 & -1 \\ & -1 & \end{pmatrix}$$

Estêncil 3D:

$$A = \frac{1}{h^2} \begin{matrix} & \text{botton} & \text{middle} & \text{top} \\ \begin{pmatrix} \bullet \end{pmatrix} & \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix} & \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix} & \begin{pmatrix} \bullet \end{pmatrix} \end{matrix}$$

Os métodos existentes para solucionar os sistemas lineares são de duas categorias: métodos diretos e métodos iterativos (ou métodos de relaxação).

Os métodos diretos (por exemplo, eliminação de Gauss) determinam a solução exata (a menos de erros de arredondamento) em um número finito de passos.

Os métodos iterativos (como Jacobi e Gauss-Seidel) começam com uma estimativa inicial para a solução, e melhoram esta aproximação através de sucessivas atualizações (ou iterações).

Observação: Métodos diretos são recomendados para sistemas densos e de pequeno porte; métodos iterativos para sistemas esparsos e de grande porte.

CAPÍTULO 2

MÉTODOS ITERATIVOS BÁSICOS

Seja $Au = f$ os sistemas lineares das Eqs. (1.2) e (1.4), onde u é a solução exata e v uma aproximação.

O erro (ou erro algébrico) é dado por

$$e = u - v,$$

com magnitude dada por alguma norma, por exemplo:

$$\text{norma } \ell_\infty : \|e\|_\infty = \max_{1 \leq j \leq n} |e_j| \quad \text{ou}$$

$$\text{norma } \ell_2 : \|e\|_2 = \left[\sum_{j=1}^n e_j^2 \right]^{1/2}$$

Observação: O erro é tão inacessível quanto a solução exata. $\|e\|_\infty$

O resíduo é dado por

$$r = f - Av$$

Pela unicidade da solução temos que

$$r = 0 \Leftrightarrow e = 0$$

Observação: $e \approx 0 \Rightarrow r \approx 0$, mas $r \approx 0$ não $\Rightarrow e \approx 0$

Exemplo:

$$Ax = b$$

$$\begin{bmatrix} 1 & 2 \\ 1,0001 & 2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 3 \\ 3,0001 \end{Bmatrix}$$

A solução exata é: $x = \{1, 1\}^T$

Suponha $\tilde{x} = \{3, 0\}^T$

O resíduo é então: $r = b - A \tilde{x} = \begin{Bmatrix} 0 \\ 0,002 \end{Bmatrix} \Rightarrow \|r\|_{\infty} = 0,002 \leftarrow \text{pequeno!}$

E o erro vale: $e = x - \tilde{x} = \begin{Bmatrix} 2 \\ 1 \end{Bmatrix} \Rightarrow \|e\|_{\infty} = 2 \leftarrow \text{grande!}$

Observação: isto está relacionado com o $\text{cond}(A)$.

Definição:

Se $\text{cond}(A) \approx 1$, então A é bem condicionada; e se $\text{cond}(A) \gg 1$, então A é mal condicionada.

No exemplo anterior, $\text{cond}(A) = 60,002$, ou seja, A é mal condicionada.

Definição:

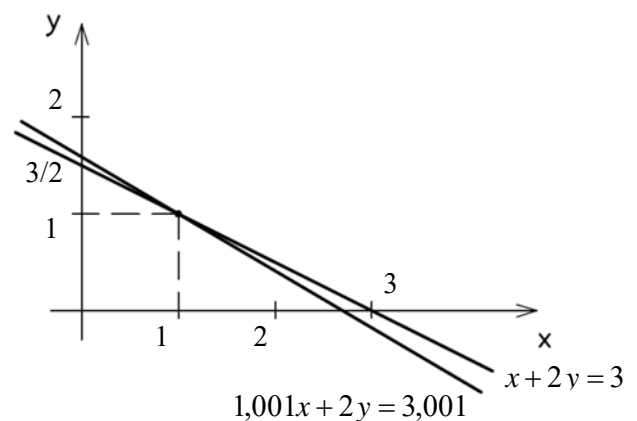
$$\text{cond}(A) = \|A\| \|A^{-1}\|$$

Interpretação geométrica:

$$\begin{cases} x + 2y = 3 \\ 1,001x + 2y = 3,001 \end{cases}$$

$$y = \frac{3}{2} - \frac{x}{2}$$

$$y = \frac{3,001}{2} - \frac{1,001x}{2}$$



Se $\text{cond}(A) \approx 1 \Rightarrow (r \approx 0 \Rightarrow e \approx 0)$

Uma relação importante entre erro e resíduo é a chamada equação residual $Ae = r$.

$$e = u - v \quad \therefore \quad u = v + e$$

Para melhorar a aproximação v , resolvemos a equação residual para e e calculamos uma nova aproximação $u = v + e$.

Esta é a ideia do refinamento iterativo:

$$\left\{ \begin{array}{l} Au = f \rightarrow v \rightarrow r = f - Av \\ Ae = r \rightarrow \bar{e} \rightarrow \bar{r} = r - A\bar{e} \\ AE = \bar{r} \rightarrow \bar{E} \dots \\ u^* = ? \end{array} \right.$$

2.1. Métodos básicos

Voltamos ao problema modelo 1D com $\sigma = 0$.

$$\left\{ \begin{array}{l} -u_{j-1} + 2u_j - u_{j+1} = h^2 f_j, \quad 1 \leq j \leq n-1 \\ u_0 = u_n = 0 \end{array} \right.$$

Método de Jacobi

$$v_j^{(1)} = \frac{1}{2} \left(v_{j-1}^{(0)} + v_{j+1}^{(0)} + h^2 f_j \right), \quad 1 \leq j \leq n-1$$

Atualiza depois da 1ª iteração.

Em forma matricial: $A = D - L - U$

onde $D =$ diagonal, $L =$ parte triangular inferior e $U =$ parte triangular superior.

Incluindo h^2 no vetor f , tem-se

$$Au = f \Rightarrow (D - L - U)u = f \Rightarrow Du = (L + U)u + f$$

$$u = D^{-1}(L + U)u + D^{-1}f$$

Fazendo $R_J = D^{-1}(L + U)$

Então, o método de Jacobi fica:

$$v^{(1)} = R_J v^{(0)} + D^{-1}f$$

Veja uma importante modificação no método de Jacobi:

$$v_j^* = \frac{1}{2} \left(v_{j-1}^{(0)} + v_{j+1}^{(0)} + h^2 f_j \right), \quad 1 \leq j \leq n-1$$

e a nova iterada é dada por:

$$v_j^{(1)} = (1 - \omega) v_j^{(0)} + \omega v_j^*$$

$$v_j^{(1)} = v_j^{(0)} + \omega (v_j^* - v_j^{(0)}), \quad 1 \leq j \leq n-1$$

$\omega \in \mathfrak{R}$ é o fator de ponderação.

Este é o método de Jacobi ponderado.

Observação: Note que $\omega = 1$, temos o método de Jacobi.

Em forma matricial,

$$v^{(1)} = R_\omega v^{(0)} + \omega D^{-1} f \quad \text{com} \quad R_\omega = (1 - \omega) I - \omega R_J$$

Observação: O método de Jacobi ponderado exige $O(2n)$ de memória.

Método de Gauss-Seidel

No método de Gauss-Seidel tem-se:

$$v_j^{(1)} = \frac{1}{2} (v_{j-1}^{(1)} + v_{j+1}^{(0)} + h^2 f_j), \quad 1 \leq j \leq n-1$$

Em forma matricial (com h^2 em f):

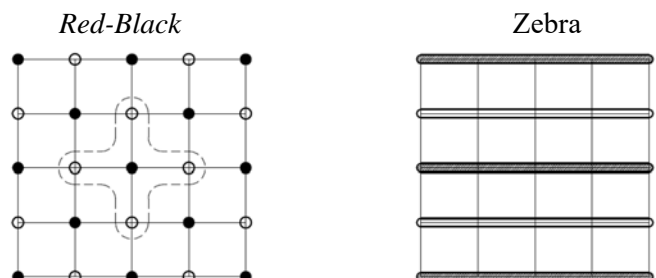
$$A u = f \Rightarrow (D - L - U) u = f \Rightarrow (D - L) u = U u + f$$

$$u = (D - L)^{-1} U u + (D - L)^{-1} f$$

Fazendo $R_G = (D - L)^{-1} U$

E o método se escreve:

$$v^{(1)} = R_G v^{(0)} + (D - L)^{-1} f$$



Observação: Para Gauss-Seidel simétrico ou Gauss-Seidel *red-black*, ver Briggs *et al.* (2000) e Weneling (1992).

2.2. Análise de Erros

Vejamos o desempenho dos métodos iterativos básicos.

É suficiente tratar do sistema linear homogêneo $Au = 0$ e usar uma estimativa inicial qualquer.

Neste caso, a solução exata é conhecida ($u = 0$) e o erro na aproximação v é $-v$.

$$e = u - v$$

$$e = 0 - v = -v$$

Então:

$$\begin{cases} -u_{j-1} + 2u_j - u_{j+1} = 0, & 1 \leq j \leq n-1 \\ u_0 = u_n = 0 \end{cases} \quad (2.1)$$

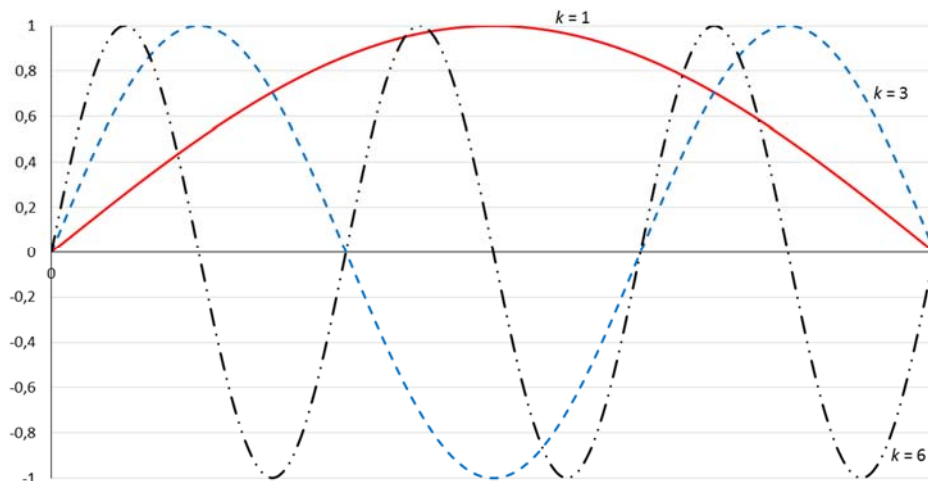
Vamos considerar a estimativa inicial do tipo (modos de Fourier)

$$v_j = \sin\left(\frac{jk\pi}{n}\right), \quad 0 \leq j \leq n, \quad 1 \leq k \leq n-1$$

onde j denota a componente do vetor v e k (número de ondas ou frequência) indica o número de “meios senos” que constituem v .

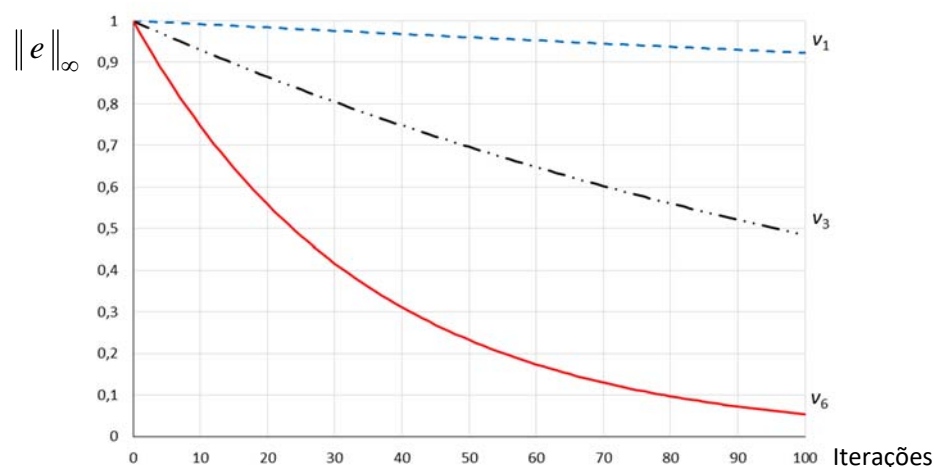
v_k denota o k -ésimo modo de Fourier, ou seja, o vetor v com o número de ondas iguais a k .

Exemplo:



Observação: Note que para k pequeno, tem-se ondas longas e suaves; e para k grande, tem-se ondas curtas e oscilatórias.

Vamos aplicar o método de Jacobi ponderado com $\omega = 2/3$ ao problema (2.1) com $n = 64$, estimativas iniciais v_1 , v_3 e v_6 , aplicados até 100 iterações.



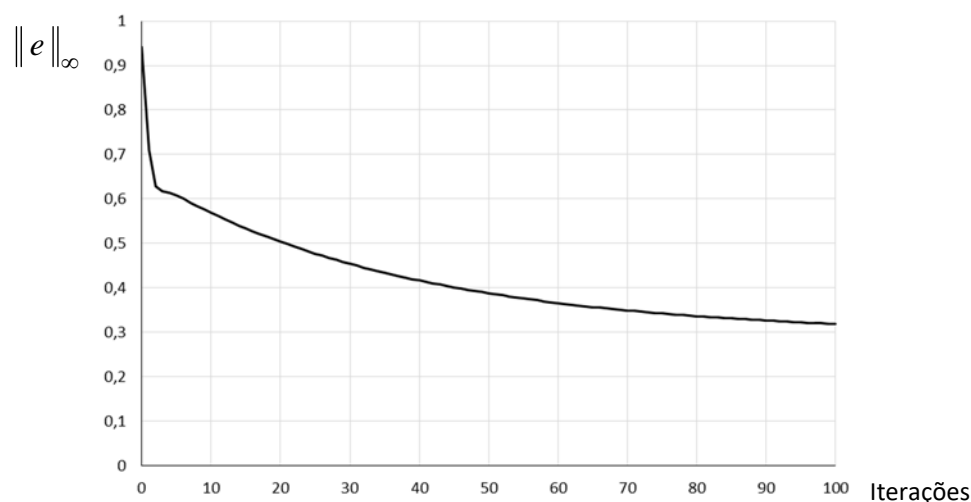
$$v_k = \sin\left(\frac{k j \pi}{64}\right)$$

Observação: Análogo para Gauss-Seidel.

Observação: O decréscimo do erro é maior para as altas frequências.

Considere uma estimativa inicial mais realística constituída de três modos com baixo, média e alta frequência (Jacobi ponderado, $\omega = 2/3$, $n = 64$, e 100 iterações):

$$v_j = \frac{1}{3} \left[\sin\left(\frac{j \pi}{64}\right) + \sin\left(\frac{6j \pi}{64}\right) + \sin\left(\frac{32j \pi}{64}\right) \right]$$



A redução drástica no início está relacionada às altas frequências, depois, ele não reduz o erro, devido às baixas frequências.

Observação: O decréscimo inicial é a rápida eliminação dos modos de alta frequência e depois o lento decréscimo é devido à persistência dos modos de baixa frequência.

Vejamos uma abordagem mais analítica:

Cada método iterativo pode ser dado por

$$v^{(1)} = R v^{(0)} + g$$

Lembrete:

Jacobi: $R = R_J$ e $g = D^{-1}f$

Jacobi Ponderado: $R = R_w$

Gauss-Seidel:

Estes métodos iterativos foram gerados tal que

$$\omega = R u + g$$

Subtraindo uma expressão da outra, temos:

$$u - v^{(1)} = R(u - v^{(0)}) + g - g$$

$$e^{(1)} = R e^{(0)}$$

Assim, $e^{(R)} = R e^{(1)} = R.R.e^{(0)} = R^2.e^{(0)}, \dots$

Após vários passos iterativos, tem-se

$$e^{(m)} = R^m e^{(0)}$$

Então:

$$\|e^{(m)}\| = \|R^m e^{(0)}\| \leq \|R^m\| \|e^{(0)}\| = \|R^m\| \|e^{(1)}\|$$

Definição: Normas de matriz

Norma induzida: $\|A\|_p = \sup_{x \neq 0} \frac{\|A x\|_p}{\|x\|_p}$

$$\|A\|_1 = \max_j \sum_{i=1}^n |a_{ij}|$$

$$\|A\|_{\infty} = \max_i \sum_{j=1}^n |a_{ij}|$$

$$\|A\|_2 = \sqrt{\rho(A^T A)} \quad \text{onde } \rho(A) = \max |\lambda(A)| \quad \text{e } \lambda(A) \text{ são os autovalores de } A.$$

Observação 1: Se A é simétrica, então

$$\|A\|_2 = \sqrt{\rho(A^T A)} = \sqrt{\rho(A^2)} = \sqrt{[\rho(A)]^2} = \rho(A)$$

Observação 2: Note de $\|e^{(m)}\| \leq \|R^m\| \|e^{(0)}\|$

que $\|R\| < 1$ força o erro ir para zero.

Teorema: $\lim_{m \rightarrow \infty} R^m = 0 \Leftrightarrow \rho(R) < 1$

Definição: $\rho(R)$ é chamado de fator de convergência assintótica. Ele representa o pior fator de redução do erro com o passo iterativo.

Quantas iterações são exigidas para reduzir o erro por um fator de 10^{-d} ?

Se $[\rho(R)]^m \leq 10^{-d}$ então aproximadamente

$$\frac{\|e^{(m)}\|}{\|e^{(0)}\|} \leq 10^{-d}$$

Vejamos:

$$[\rho(R)]^m = \left[\left(\lim_{m \rightarrow \infty} \|R\|^m \right)^{1/m} \right]^m = \lim_{m \rightarrow \infty} \|R\|^m \leq 10^{-d}$$

É razoável pensar que

$$\frac{\|e^{(m)}\|}{\|e^{(0)}\|} \leq \|R\|^m \approx \lim_{m \rightarrow \infty} \|R\|^m \leq 10^{-d}$$

ou seja,

$$\frac{\|e^{(m)}\|}{\|e^{(0)}\|} < \approx 10^{-d}$$

Então, $[\rho(R)]^m \leq 10^{-d}$, e assim,

$$m \log[\rho(R)] \leq -d \log 10$$

e como $\log[\rho(R)] < 0$

$$m \geq \frac{-d}{\log[\rho(R)]}$$

Definição: $-\log[\rho(R)]$ é chamado de razão de convergência assintótica.

Observação: Note que $\rho(R) = 1^- \Rightarrow \log \rho(R) \rightarrow 0^- \Rightarrow m \rightarrow \infty \leftarrow$ caso ruim!

$$\rho(R) = 0^+ \Rightarrow \log \rho(R) \rightarrow -\infty \Rightarrow m \rightarrow 0 \leftarrow$$
 caso ótimo!

Considerando o método de Jacobi ponderado aplicado ao problema 1D, temos:

$$R_\omega = (1 - \omega) I - \omega R_J$$

Então,

$$R_\omega = I - \frac{\omega}{2} A$$

e

$$\lambda(R_\omega) = 1 - \frac{\omega}{2} \lambda(A)$$

Observação: Note que o problema de procurar os autovalores de R_ω , reduz-se ao problema de procurar os autovalores de A .

Os autovalores de A são da forma

$$\lambda_k(A) = 4 \sin^2 \left(\frac{k \pi}{2n} \right), \quad 1 \leq k \leq n-1$$

e os autovalores de R_ω são

$$\lambda_k(R_\omega) = 1 - 2\omega \sin^2 \left(\frac{k \pi}{2n} \right), \quad -1 \leq k \leq n-1$$

Note que se $0 < \omega \leq 1$, então $|\lambda_k(R_\omega)| < 1$, e o método de Jacobi ponderado é convergente.

Definição: Os modos de Fourier na parte inferior do espectro, ou seja, $1 \leq k \leq n/2$, são chamados de modos suaves (ou de baixa frequência). Os modos na parte superior do espectro, ou seja, $n/2 \leq k \leq n-1$, são chamados de modos oscilatórios (ou de alta frequência).

Tome $k=1$ (o modo mais suave):

$$\lambda_k = 1 - 2\omega \sin^2\left(\frac{k\pi}{2n}\right)$$

então,

$$\lambda_1 = 1 - 2\omega \sin^2\left(\frac{\pi}{2n}\right) \quad \text{e} \quad h = \frac{1}{n}$$

$$\lambda_1 = 1 - 2\omega \sin^2\left(\frac{\pi h}{2}\right) \quad (\text{como } \sin x \approx x \text{ para } x \approx 0)$$

$$\lambda_1 \approx 1 - 2\omega \left(\frac{\pi h}{2}\right)^2 \quad \therefore \quad \lambda_1 \approx 1 - \omega \frac{\pi^2 h^2}{2}$$

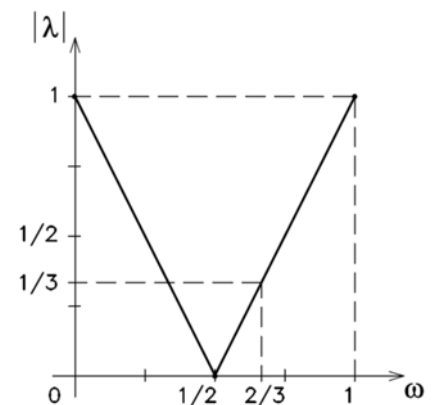
Portanto, o autovalor associado ao modo mais suave, estará sempre próximo de 1 (fator de convergência péssimo).

Tome $k=n-1$ (o modo mais oscilatório):

$$\lambda_{n-1} = 1 - 2\omega \sin^2\left(\frac{(n-1)\pi}{2n}\right) \quad (\text{como } n \text{ é grande, } \frac{n-1}{2n} \rightarrow \frac{1}{2})$$

$$\lambda_{n-1} = 1 - 2\omega \left[\sin\left(\frac{(n-1)\pi}{2n}\right) \right]^2$$

$$\lambda_{n-1} \approx 1 - 2\omega \sin^2\frac{\pi}{2} \quad \therefore \quad \lambda_{n-1} \approx 1 - 2\omega$$



Lembre-se que se $\omega=1$, temos Jacobi puro, ou seja, método ruim para modos oscilatórios.

Por exemplo, para $\omega=2/3$, $|\lambda_{n-1}| < 1/3$ (bom fator de convergência).

Pode-se mostrar ainda que, para $\omega = 2/3$, $|\lambda_k| < 1/3$ e $n/2 \leq k \leq n-1$.

Definição: $|\lambda_n|$, $n/2 \leq k \leq n-1$ é chamado de fator de suavização.

Observação: Note que o fator de suavização $n/2 \leq k \leq n-1$, não depende de h .

2.3. Resultados através de experimentação numérica

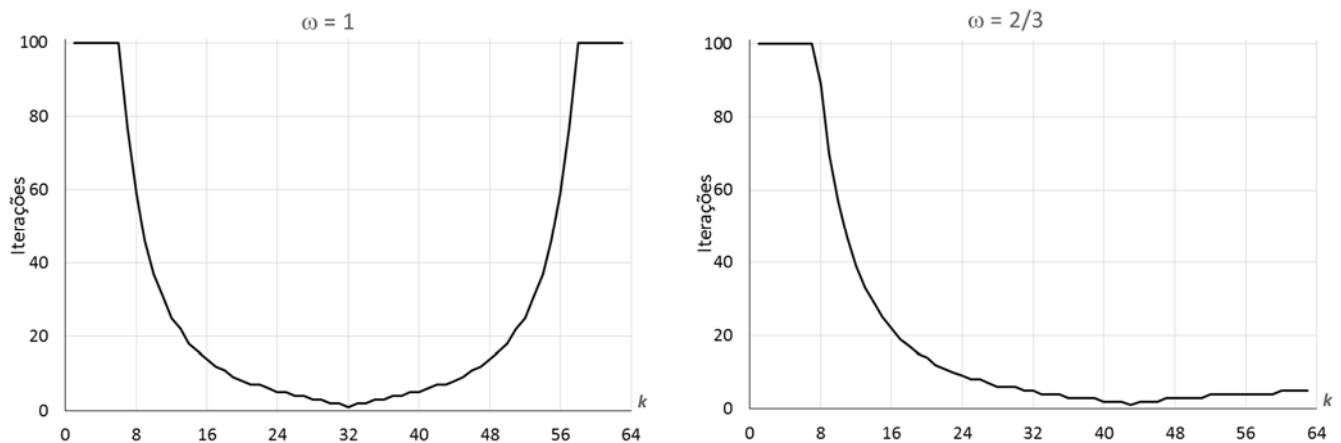
Vamos aplicar o método de Jacobi ponderado no problema modelo 1D homogêneo (2.1), ou seja,

$Au = 0$, com $n = 64$ pontos.

Vamos usar estimativas iniciais constituindo de modos de Fourier puros (não combina modos).

Usamos $\omega = 1$ (Jacobi) e $\omega = 2/3$ (Jacobi ponderado).

Os gráficos mostram o número de iterações necessárias para reduzir o erro por um fator de 10^{-2} para cada modo.



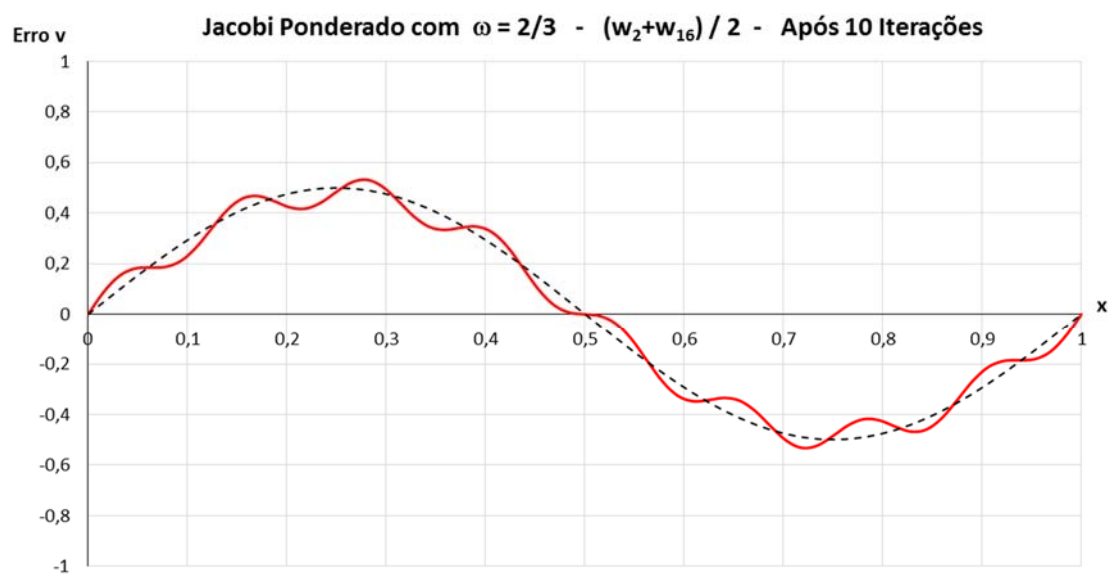
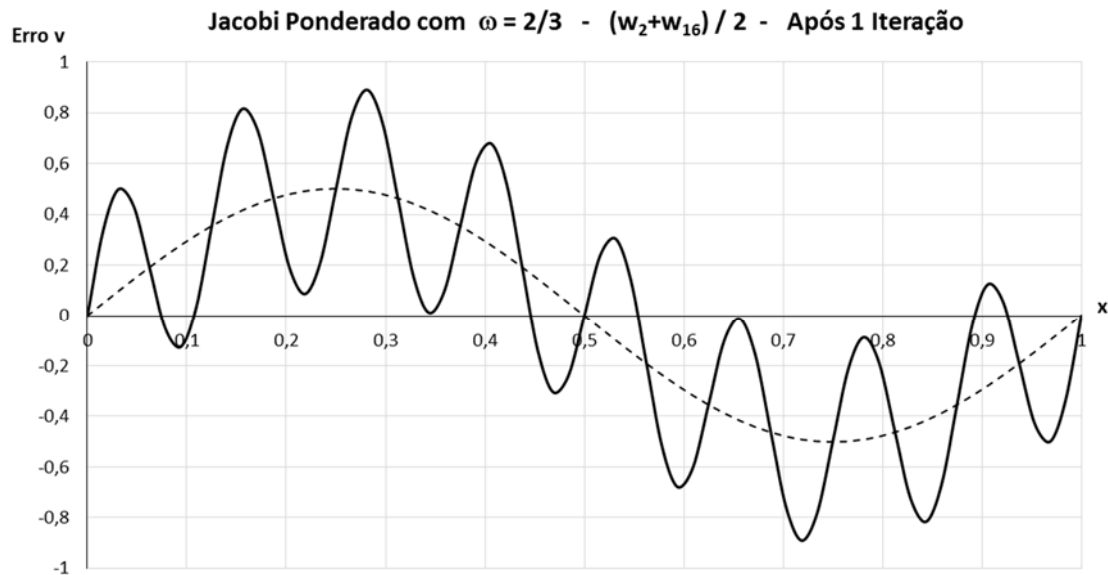
Observação: Para $\omega = 1$, as componentes do erro de alta e baixa frequência são amortecidos muito lentamente. Para $\omega = 2/3$ os modos suaves são amortecidos lentamente e os modos oscilatórios mais rapidamente.

Os gráficos a seguir mostram a seletividade da propriedade de amortecimento.

Para isso, usa-se uma estimativa inicial consistindo de dois modos $k = 2$ e $k = 16$, ou seja:

$$v_j = \frac{1}{2} \left[\sin \left(\frac{2\pi j}{n} \right) + \sin \left(\frac{16\pi j}{n} \right) \right]$$

com Jacobi ponderado, $\omega = 2/3$ e $n = 64$.



Note que os modos de alta frequência foram rapidamente suavizados e os modos de baixa frequência persistem.

2.4. Considerações

- Vimos que o rápido decréscimo do erro durante as iterações iniciais é devido à eficiente eliminação dos modos oscilatórios, mas uma vez estes modos tenham sido removidos, as iterações são pouco efetivas para reduzir os modos suaves.
- E nos modos suaves tem resíduos pequenos e o erro cai lentamente. E nos modos oscilatórios tem resíduos grandes e o erro cai mais lentamente.

Definição: A propriedade de eliminar os modos oscilatórios e deixar apenas modos suaves, e chamada **propriedade de suavização**.

Observação: Jacobi não possui tal propriedade; Jacobi ponderado sim.

CAPÍTULO 3

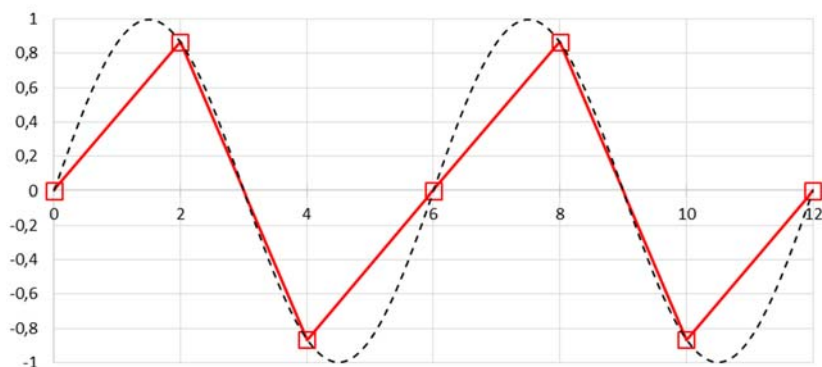
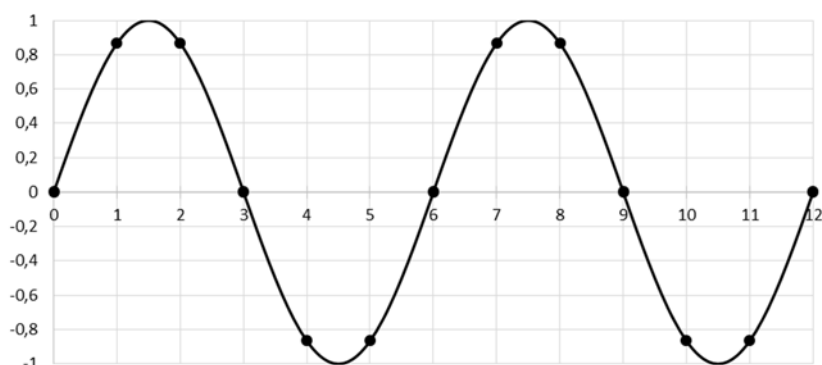
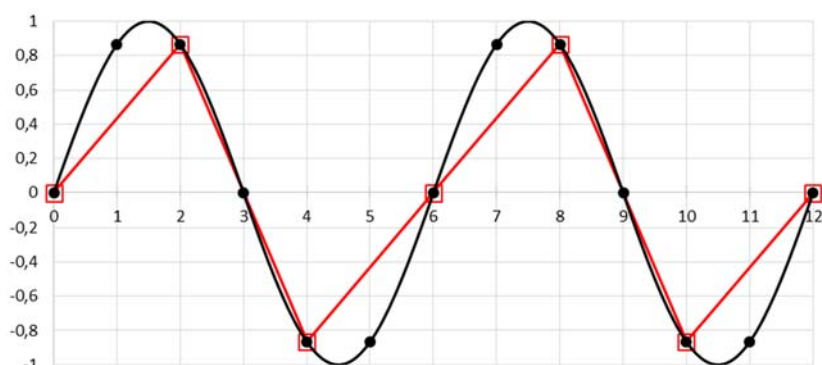
ELEMENTOS DO MÉTODO MULTIGRID

Muitos métodos iterativos possuem a propriedade de suavização (eliminar modos oscilatórios deixando somente modos suaves).

Esses métodos podem ser modificados para tratar efetivamente todos os campos de erro (suave e oscilatório)?

Assumimos que um esquema de relaxação tenha sido aplicado até restarem apenas modos de erro suaves.

Como estas componentes ficariam em uma malha mais grossa?



$$\Omega_{n=12}^h, \quad \frac{n}{2} = 6 > 4 = k \quad (\text{número de ondas}) \quad \leftarrow \text{modo suave}$$

$$\Omega_{n=6}^{2h}, \quad \frac{n}{2} = 3 < 4 = k \quad \leftarrow \text{modo oscilatório}$$

Modo suave em Ω^h foi projetado diretamente em Ω^{2h} , onde tornou-se mais oscilatório.

Considere o k -ésimo modo da malha fina Ω^h , avaliado nos pontos pares. Se $1 \leq k < \frac{n}{2}$ (modos suaves), pode-se escrever:

$$\omega_{k,j}^h = \sin\left(\frac{j k \pi}{n}\right)$$

Obs.: j faz papel de x em $\sin x$.

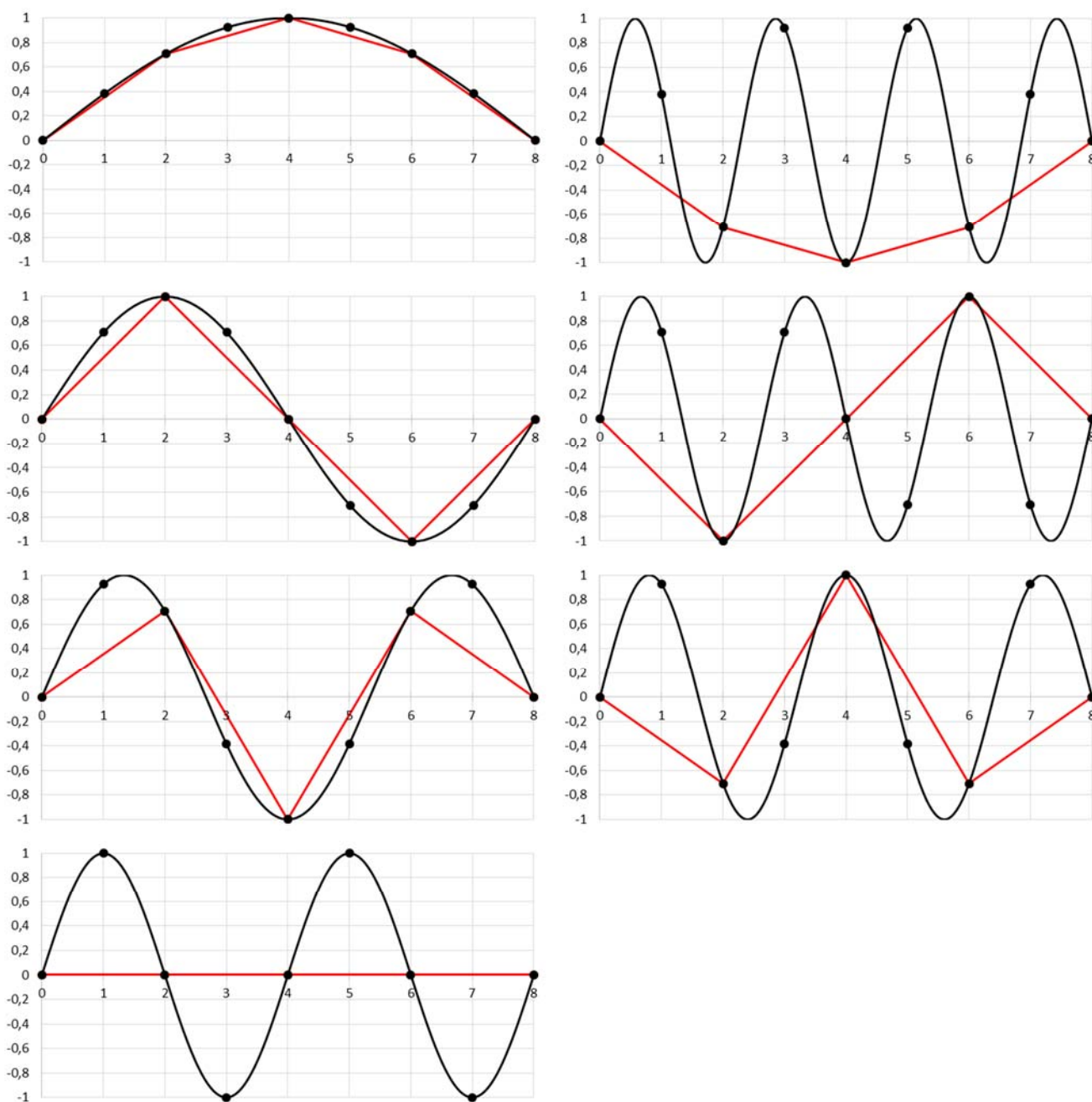
$$\omega_{k,2j}^h = \sin\left(\frac{2j k \pi}{n}\right) = \sin\left(\frac{j k \pi}{n/2}\right) = \omega_{k,j}^{2h}$$

Observação: O k -ésimo modo suave na malha fina Ω^h , torna-se o k -ésimo modo na malha grossa Ω^{2h} , tornando-se mais oscilatório.

Considere o k -ésimo modo $\left(k > \frac{n}{2}\right)$ na malha fina Ω^h . Este modo (oscilatório) sofre uma curiosa transformação: ele se torna o $(n - k)$ -ésimo modo na malha grossa Ω^{2h} .

Este fenômeno chama-se *aliasing*.

$$\varphi^k = \omega_{k,j}^h$$



Note que: $\varphi^7 = -\varphi^1$; $\varphi^6 = -\varphi^2$; $\varphi^5 = -\varphi^3$.

Observação: O k -ésimo modo oscilatório em Ω^h é representado pelo $(n - k)$ -ésimo modo suave em Ω^{2h} . (Exercício 1)

Importante: Modos suaves na malha fina tornam-se modos oscilatórios na malha grossa. Isto sugere que, quando o processo de relaxação começa a ficar lento, sinalizando a predominância dos modos suaves, é recomendável mudar para uma malha mais grossa, onde estes modos tornam-se mais oscilatórios e o processo de relaxação será mais efetivo.

Mas como passar o problema para uma malha mais grossa? Como suavizar os modos oscilatórios do erro?

3.1. Noções básicas

Se v é uma aproximação para a solução exata u , então $e = u - v$ e $Ae = r = f - Av$.

Então, podemos focar o processo de suavização das componentes oscilatórias do erro usando diretamente a equação residual na malha grossa. (Exercício 2)

Estratégia:

- Suavizar $Au = f$ em Ω^h e obter v^h .
- Calcular $r = f - Av^h$
 - Suavizar $Ae = r$ em Ω^{2h} e obter e^{2h} .
 - Obter e^h em Ω^h , dado e^{2h} .
- Corrigir a solução $\Omega^h : v^h \leftarrow v^h + e^h$

Esta estratégia é a base do algoritmo que é chamado “esquema de correção” (CS – *Correction Scheme*).

As questões a respeito da transferência de informação entre as malhas, ainda devem ser respondidas. Para isso consideramos o caso onde a malha mais grossa tem duas vezes o espaçamento da malha mais fina, ou seja, razão de engrossamento $q = 2$.

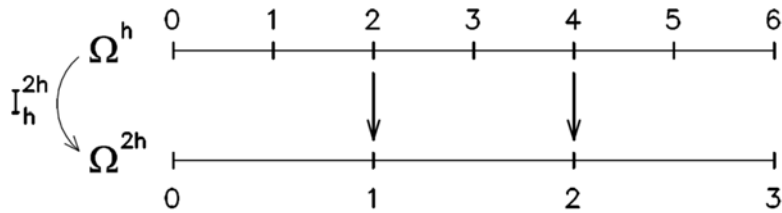
3.2. Restrição

Restrição é a operação que transfere informações de Ω^h para Ω^{2h} , e é denotada por I_h^{2h} .

Exemplo:

Injeção: $v^{2h} = I_h^{2h} v^h$, onde

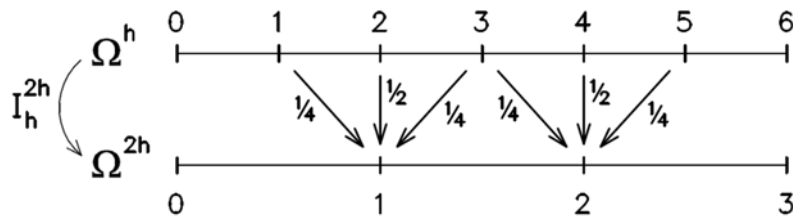
$$v_j^{2h} = v_{2j}^h$$



Note que: $1 \leq j \leq \frac{n}{2} - 1$

Ponderação completa: $v^{2h} = I_h^{2h} v^h$, onde

$$v_j^{2h} = \frac{1}{4} (v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h), \quad 1 \leq j \leq \frac{n}{2} - 1$$



O operador de restrição por ponderação completa é um operador linear de \mathfrak{R}^{n-1} para $\mathfrak{R}^{\frac{n}{2}-1}$.

Exemplo: para $n = 6 \rightarrow I_h^{2h} : \mathfrak{R}^5 \rightarrow \mathfrak{R}^2$

$$I_h^{2h} v^h = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{Bmatrix}_h = \begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix}_{2h} = v^{2h}$$

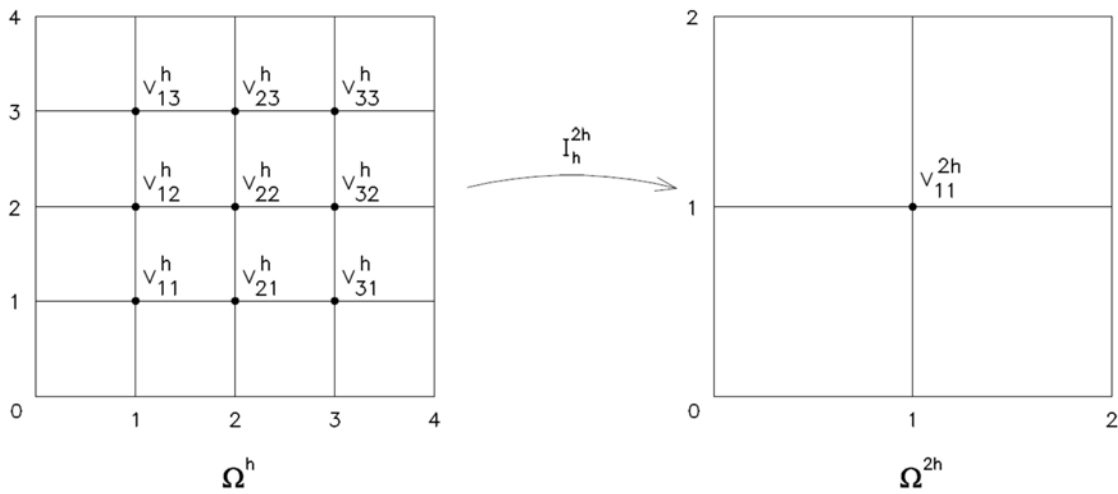
Para o caso 2D, o operador de restrição por ponderação completa é dado por,

$$v^{2h} = I_h^{2h} v^h$$

onde:

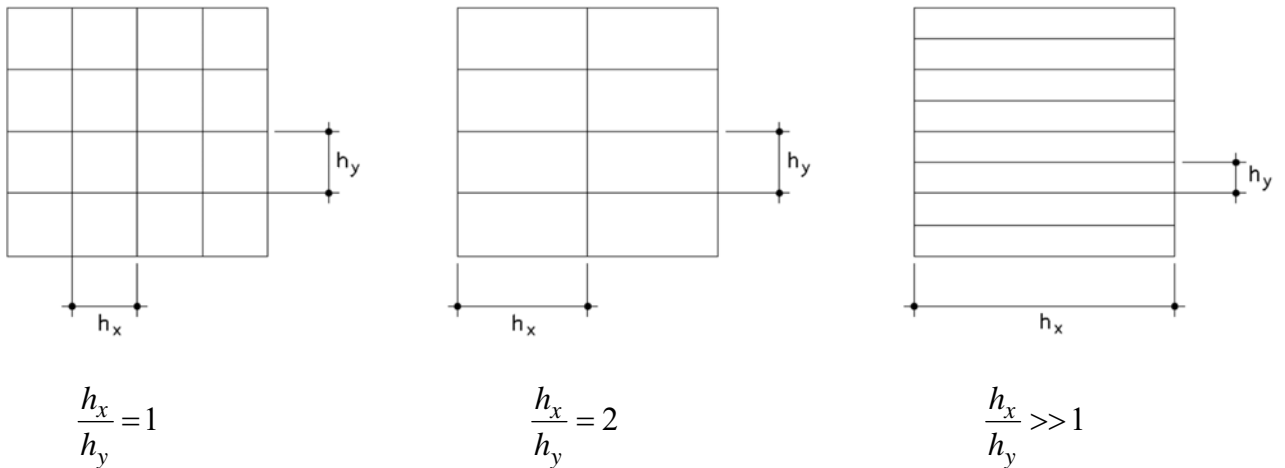
$$v_{ij}^{2h} = \frac{1}{16} \left[v_{2i-1,2j-1}^h + v_{2i-1,2j+1}^h + v_{2i+1,2j-1}^h + v_{2i+1,2j+1}^h + \right. \\ \left. + 2 \left(v_{2i,2j-1}^h + v_{2i,2j+1}^h + v_{2i-1,2j}^h + v_{2i+1,2j}^h \right) + 4 v_{2i,2j}^h \right], \quad 1 \leq i, j \leq \frac{n}{2} - 1$$

Exemplo:



$$v_{11}^{2h} = \frac{1}{16} \left[v_{11}^h + v_{13}^h + v_{31}^h + v_{33}^h + 2 \left(v_{21}^h + v_{23}^h + v_{12}^h + v_{32}^h \right) + 4 v_{22}^h \right]$$

Exemplo: Existe ainda a injeção (somente em P), a meia ponderação (N , S , E e W) e a ponderação parcial (N e S , ou E e W , dependendo da direção de anisotropia).



3.3. Prolongação

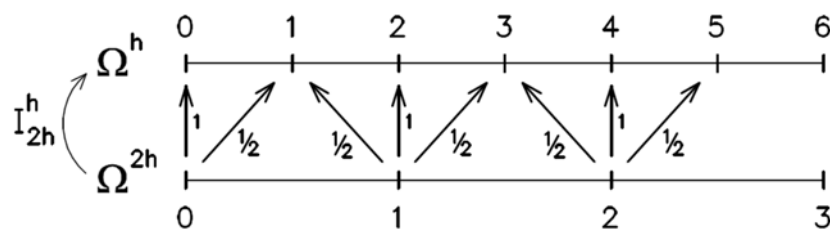
Prolongação (ou interpolação) é a operação que transfere informações da malha grossa Ω^{2h} para a malha fina Ω^h , e é denotada por I_{2h}^h .

A mais simples das interpolações é bastante efetiva, portanto, vamos considerar a interpolação linear.

Exemplo: Interpolação linear: $v^h = I_{2h}^h v^{2h}$

onde:

$$\begin{cases} v_{2j}^h = v_j^{2h} \\ v_{2j+1}^h = \frac{1}{2}(v_j^{2h} + v_{j+1}^{2h}), \quad 0 \leq j \leq \frac{n}{2} - 1 \end{cases}$$



$$j=0 \rightarrow \begin{cases} v_0^h = v_0^{2h} \\ v_1^h = \frac{1}{2}(v_0^{2h} + v_1^{2h}) \end{cases} ; \quad j=1 \rightarrow \begin{cases} v_2^h = v_1^{2h} \\ v_3^h = \frac{1}{2}(v_1^{2h} + v_2^{2h}) \end{cases}$$

O operador I_{2h}^h é um operador linear de $\mathfrak{R}^{\frac{n}{2}-1}$ para \mathfrak{R}^{n-1} .

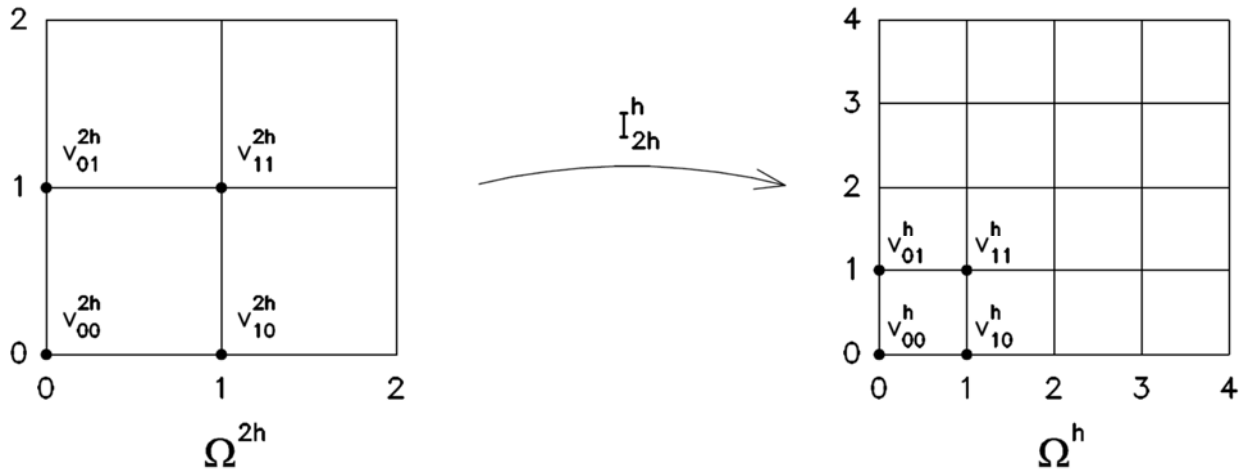
Exemplo: para $n = 6 \rightarrow I_{2h}^h : \mathfrak{R}^2 \rightarrow \mathfrak{R}^5$

$$I_{2h}^h v^{2h} = \frac{1}{2} \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 1 & 1 \\ 0 & 2 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix}_{2h} = \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{Bmatrix}_h = v^h$$

Lembrar que $v_0 = 0$.

Para o caso bidimensional, $v^h = I_{2h}^h v^{2h}$, onde

$$\text{Bi-linear} \left\{ \begin{array}{l} v_{2i,2j}^h = v_{i,j}^{2h} \\ v_{2i+1,2j}^h = \frac{1}{2} (v_{i,j}^{2h} + v_{i+1,j}^{2h}) \\ v_{2i,2j+1}^h = \frac{1}{2} (v_{i,j}^{2h} + v_{i,j+1}^{2h}) \\ v_{2i+1,2j+1}^h = \frac{1}{4} (v_{i,j}^{2h} + v_{i+1,j}^{2h} + v_{i,j+1}^{2h} + v_{i+1,j+1}^{2h}) \end{array} \right\} \quad 0 \leq i, j \leq \frac{n}{2} - 1$$



Exemplo:

$$i = j = 0 \Rightarrow \left\{ \begin{array}{l} v_{00}^h = v_{00}^{2h} \\ v_{10}^h = \frac{1}{2} (v_{00}^{2h} + v_{10}^{2h}) \\ v_{01}^h = \frac{1}{2} (v_{00}^{2h} + v_{01}^{2h}) \\ v_{11}^h = \frac{1}{4} (v_{00}^{2h} + v_{10}^{2h} + v_{01}^{2h} + v_{11}^{2h}) \end{array} \right.$$

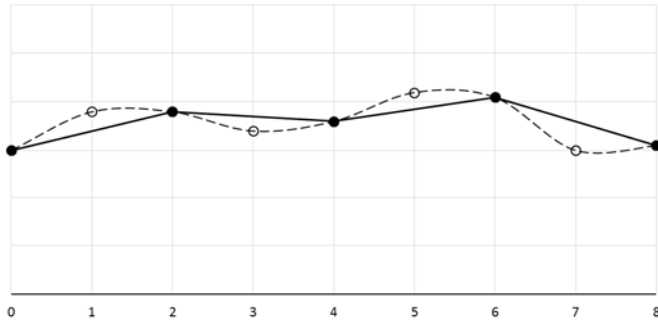
Quão bom é o processo de interpolação?

i) Se o erro é suave na malha fina

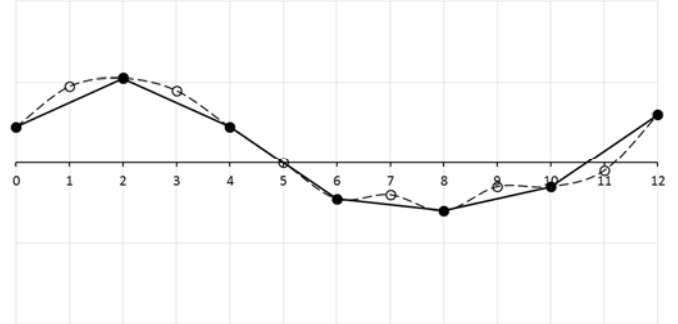
Quando o erro na malha grossa é interpolado para a malha fina, o interpolante é também suave e esperamos uma boa aproximação para o erro na malha fina. Ver exemplos (a) e (b) da Fig. 1.

ii) Se o erro é oscilatório na malha fina

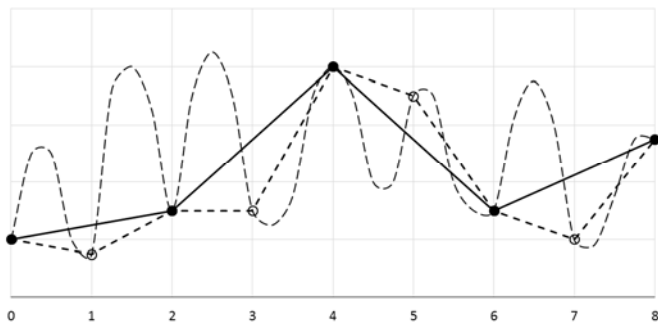
Quando o erro na malha grossa é interpolado para a malha fina, até mesmo boas aproximações na malha grossa podem produzir interpolantes que não são acurados.



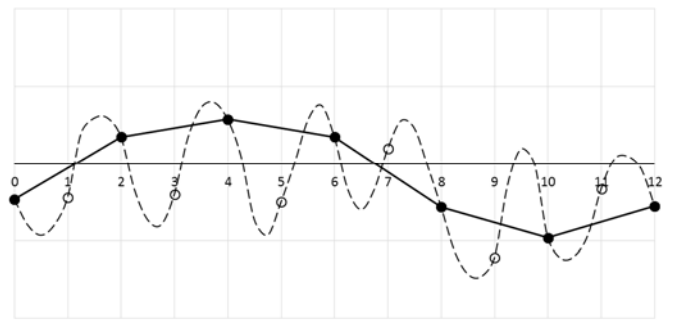
(a)



(b)



(c)

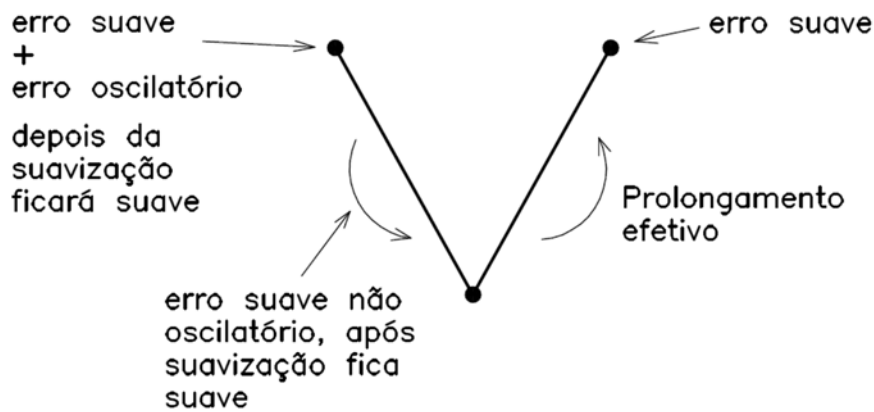


(d)

- malha grossa ————— interpolante do erro na malha grossa
- malha fina - - - - - erro na malha fina

Figura 1. (a) e (b) Se o erro exato em Ω^h (indicado por ○ e ●) é suave, um interpolante do erro e^{2h} na malha grossa (linha sólida conectando pontos ●) deve dar uma boa representação do erro exato. (c) e (d) Se o erro exato em Ω^h (indicado por ○ e ●) é oscilatório, um interpolante do erro e^{2h} na malha grossa (linha sólida conectando pontos ●) pode dar uma péssima representação do erro exato.

Observação: Podemos concluir que o processo de interpolação é mais efetivo quando o erro é suave.

Lembrete:

Propriedade variacional:

Dado I_h^{2h} o operador de restrição por ponderação completa, e I_{2h}^h o operador de prolongação linear,

vale que

$$I_{2h}^h = C \cdot (I_h^{2h})^T, \quad C \in \mathfrak{R} \quad (\text{Exercício 6})$$

Sendo A^{2h} a matriz A na malha grossa, A^{2h} pode ser tomada de duas formas:

- i) Por discretização do problema em Ω^{2h} .
- ii) Pela propriedade de Galerkin:

$$A^{2h} = I_h^{2h} A^h I_{2h}^h$$

3.4. Algoritmo

Esquema de Correção (CS) para duas malhas

$$v^h \leftarrow TG(v^h, f^h) \quad TG = \text{Two-Grid}$$

- Suavize $A^h u^h = f^h$, v_1 vezes em Ω^h com estimativa v^h
- Calcule $r^h = f^h - A^h v^h$
- Restrinja r^h para Ω^{2h} , ou seja, calcule $r^{2h} = I_h^{2h} r^h$
- Resolva $A^{2h} e^{2h} = r^{2h}$ em Ω^{2h}
- Interpole e^{2h} para Ω^h , ou seja, calcule $e^h = I_{2h}^h e^{2h}$
- Corrija a solução na malha fina, ou seja

$$v^h \leftarrow v^h + e^h$$

- Suavize $A^h u^h = f^h$, v_2 vezes em Ω^h com estimativa v^h

Observação: Na prática, v_1 e v_2 (pré e pós suavizações, respectivamente) são aproximadamente 1, 2 ou 3.

Importante: O processo de suavização na malha fina elimina componentes oscilatórios do erro, deixando apenas componentes suaves.

Supondo que a equação residual é resolvida acuradamente em Ω^{2h} , é importante a transferência do erro para a malha fina de uma forma precisa. Devido ao fato do erro ser suave em Ω^h , a interpolação trabalha bem e a correção na malha fina é efetiva.

Exemplo numérico:

Considere o método de Jacobi ponderado com $\omega = 2/3$ aplicado ao problema unidimensional

$A \cdot u = 0$ com $n = 64$ e usando como estimativa inicial

$$v_j^h = \frac{1}{2} \left[\sin\left(\frac{16j\pi}{n}\right) + \sin\left(\frac{40j\pi}{n}\right) \right]$$

Discutir a redução percentual da norma $\|e\|_2$, quando:

- 1 suavização (Ω^h) : 57%
- 3 suavizações (Ω^h) : 36%
- 1 suavização (Ω^{2h}) : 26%
- 3 suavizações (Ω^{2h}) : 8%
- 3 suavizações (Ω^h) : 3%

(ver gráficos) [Trabalho Computacional 2] (pag. 47): sugestão de armazenamento

Ideia Multigrid:

Qual a melhor forma de se resolver $A^{2h} e^{2h} = r^{2h}$?

Este problema em Ω^{2h} não é diferente do problema em Ω^h ($A^h u^h = f^h$). Então, podemos aplicar o esquema de correção de duas malhas para a equação residual em Ω^{2h} , ou seja, suavizar em Ω^{2h} e transferir o problema para Ω^{4h} e efetuar o passo de correção.

Nós podemos repetir este processo sucessivamente até a malha mais grossa possível ou a mais grossa desejada.

Vamos descrever um algoritmo para o esquema de correção para diversas malhas.

Para fins de implementação computacional, nós usaremos uma notação específica:

- O vetor do lado direito (vetor dos termos independentes) da equação residual será chamado f^{2h} (ao invés de r^{2h}), porque ele sempre será um vetor do lado direito.
- O vetor solução da equação residual será chamado u^{2h} (ao invés de e^{2h}), porque ele sempre será um vetor solução.
- v^{2h} é a aproximação de u^{2h} .

$$A^h u^h = f^h$$

$$A^{2h} e^{2h} = r^{2h} \Leftrightarrow A^{2h} u^{2h} = f^{2h}$$

E a estimativa para a equação residual será $v^{2h} = 0$ ($Ae = r$, ou seja, $e = 0$ teoricamente).

Assumimos que existem $\ell > 1$ malhas com espaçamentos $h, 2h, 4h, \dots, \ell h = 2^{\ell-1} h$.

Esquema de correção (CS) com ciclo v :

$$v^h \leftarrow MG(v^h, f^h) \quad MG = \text{Multigrid}$$

- Suavize $A^h u^h = f^h$, v_1 vezes com estimativa inicial v^h (1)

- Calcule $r^h = f^h - A^h v^h$

- Calcule $f^{2h} = I_h^{2h} r^h$ (2)

- Suavize $A^{2h} u^{2h} = f^{2h}$, v_1 vezes com estimativa inicial $v^{2h} = 0$ (3)

- Calcule $r^{2h} = f^{2h} - A^{2h} v^{2h}$

- Calcule $f^{4h} = I_{2h}^{4h} r^{2h}$ (4)

- Suavize $A^{4h} u^{4h} = f^{4h}$, v_1 vezes com estimativa inicial $v^{4h} = 0$ (5)

- Calcule $r^{4h} = f^{4h} - A^{4h} v^{4h}$

- Calcule $f^{8h} = I_{4h}^{8h} r^{4h}$ (6)

⋮

- Resolva $A^{\ell h} u^{\ell h} = f^{\ell h}$ (7)

⋮

- Calcule a correção $v^{4h} \leftarrow v^{4h} + I_{8h}^{4h} v^{8h}$ (8)

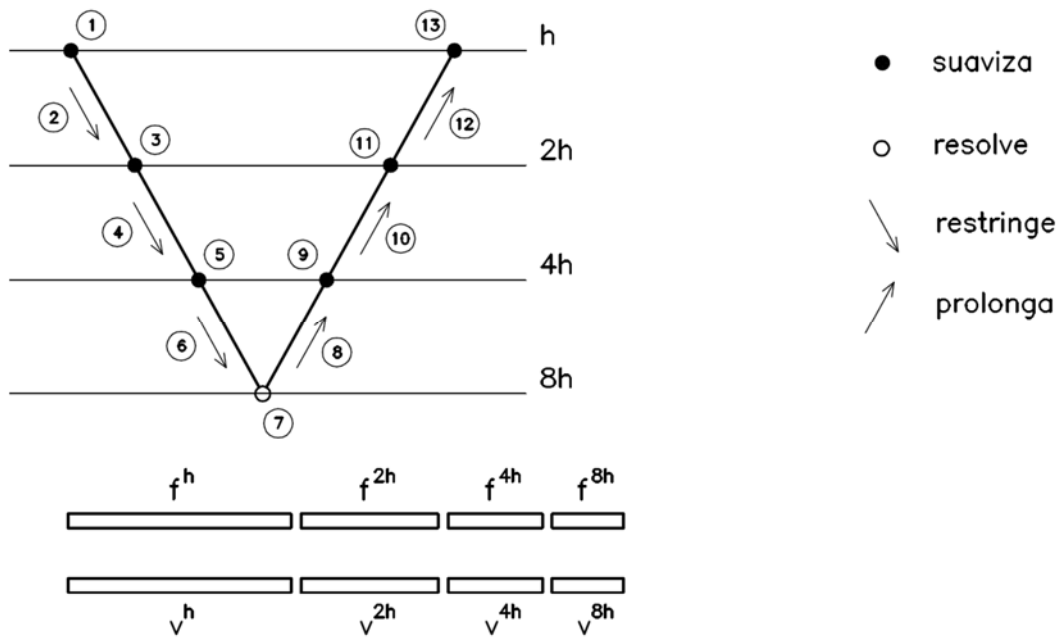
- Suavize $A^{4h} u^{4h} = f^{4h}$, ν_2 vezes com estimativa inicial v^{4h} (9)

- Calcule a correção $v^{2h} \leftarrow v^{2h} + I_{4h}^{2h} v^{4h}$ (10)

- Suavize $A^{2h} u^{2h} = f^{2h}$, ν_2 vezes com estimativa inicial v^{2h} (11)

- Calcule a correção $v^h \leftarrow v^h + I_{2h}^h v^{2h}$ (12)

- Suavize $A^h u^h = f^h$, ν_2 vezes com estimativa inicial v^h (13)



Devido ao formato deste diagrama (a ordem em que o algoritmo percorre as malhas), este algoritmo é chamado de ciclo V.

O ciclo V pertence a uma família de ciclos chamados de ciclo μ .

Esquema de correção (CS) com ciclo μ :

$$v^h \leftarrow MG_{\mu}(v^h, f^h)$$

- 1) Suavize $A^h u^h = f^h$, v_1 vezes com estimativa inicial v^h
- 2) Se Ω^h é a malha mais grossa

Vá para o passo 4

Caso contrário:

$$\text{Calcule } r^h = f^h - A^h v^h$$

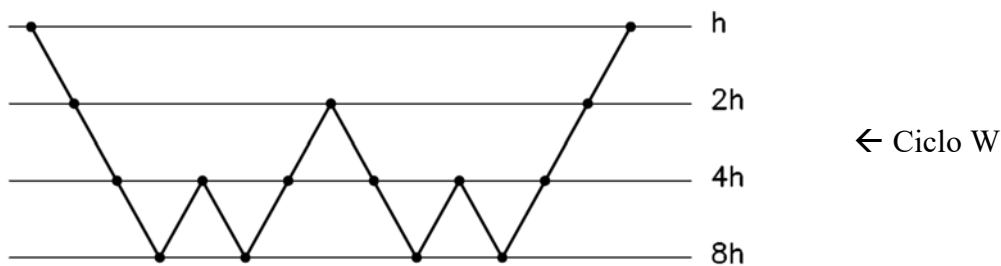
$$\text{Restrinja } f^{2h} = I_h^{2h} r^h$$

$$v^{2h} \leftarrow 0$$

$$v^{2h} \leftarrow MG_{\mu}(v^{2h}, f^{2h}) \mu \text{ vezes.}$$

- 3) Calcule a correção $v^h \leftarrow v^h + I_{2h}^h v^{2h}$
- 4) Suavize $A^h u^h = f^h$, v_2 vezes com estimativa inicial v^h

Observação: Se $\mu = 1$ temos o ciclo V; se $\mu = 2$ temos o ciclo W (estes são os dois ciclos mais usados).



- Como obter uma estimativa inicial melhor para o problema de suavização na malha Ω^h ?

Podemos para isso resolver o problema na malha Ω^{2h} e depois interpolar tal solução para Ω^h .

- Como obter uma estimativa inicial melhor para o problema na malha Ω^{2h} ?

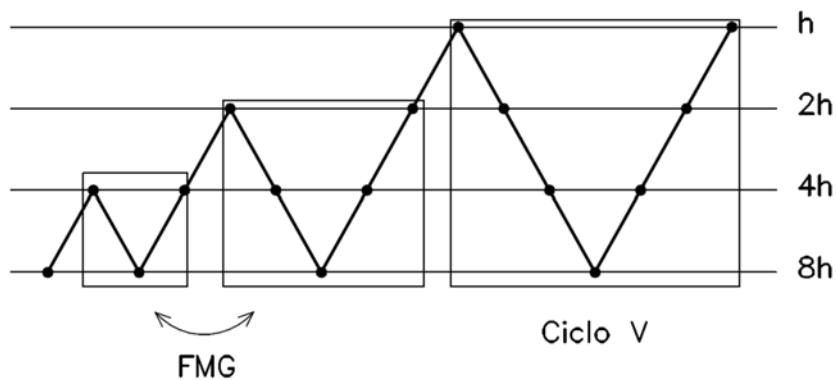
Resolvemos o problema na malha Ω^{4h} . E assim por diante, até que se atinja a malha mais grossa possível ou mais grossa desejada.

Esta é a ideia do Full Multigrid (FMG).

Observação: O trabalho extra do FMG não é caro computacionalmente, e se paga por si só.

(Exercício 8).

Exemplo: FMG



Exercícios:

Lista 3: Exercícios 1, 2, 6 e 8. (dia 23/nov)

Trabalho Computacional 2: Reproduzir os dados da Fig. 3.5 da página 39 para 2 níveis, para ajudar a ver a Fig. 4.1 da página 47 (adaptada para 2 níveis). Observação: restrição por injeção, e prolongação por interpolação linear. (dia 30/nov)

CAPÍTULO 4

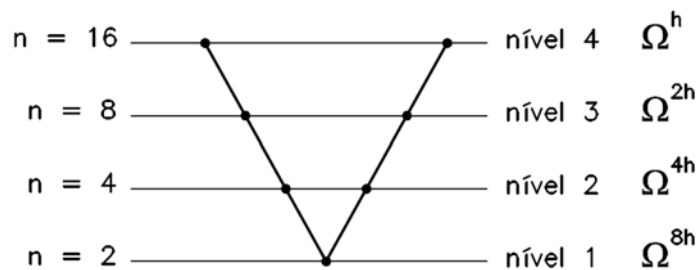
IMPLEMENTAÇÃO

Agora vamos recorrer a alguns aspectos práticos da escrita de programas Multigrid. Tais programas deveriam ser montados de forma modular (suavização, restrição, prolongação, etc.).

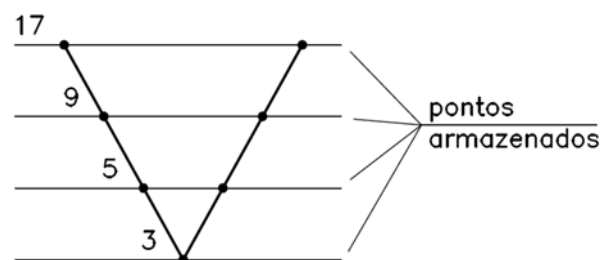
4.1 ESTRUTURAS DE DADOS

Vetores soluções e vetores dos termos independentes deveriam ser armazenados em um vetor da forma adjacente.

Considere um problema unidimensional com $n = 16$ pontos, ciclo V e quatro níveis.



Devido às condições de contorno, em geral, no nível ℓ , tem-se $2^\ell + 1$ pontos (diferenças finitas).




Necessitamos de dois vetores: um para guardar as soluções em cada malha e o outro para o vetor dos termos independentes em cada malha.

	Vetor das soluções v				Vetor termos independentes f (ou r)			
	Número de pontos da malha				Número de pontos da malha			
	3	5	9	17	3	5	9	17
Inicialização	0	0	0	0	0	0	0	f^h
Suavização em $\Omega^h : r^h \rightarrow \Omega^{2h}$	0	0	0	v^h	0	0	f^{2h}	x
Suavização em $\Omega^{2h} : r^{2h} \rightarrow \Omega^{4h}$	0	0	v^{2h}	x	0	f^{4h}	x	x
Suavização em $\Omega^{4h} : r^{4h} \rightarrow \Omega^{8h}$	0	v^{4h}	x	x	f^{8h}	x	x	x
Resolva em Ω^{8h}	v^{8h}	x	x	x	x	x	x	x
Corrija e suavize em Ω^{4h}	0	v^{4h}	x	x	x	x	x	x
Corrija e suavize em Ω^{2h}	0	0	v^{2h}	x	x	x	x	x
Corrija e suavize em Ω^h	0	0	0	v^h	x	x	x	x

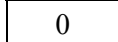
Suavização em $\Omega^h : r^h \rightarrow \Omega^{2h} : A^h v^h = f^h$

$$r^h = f^h - A^h v^h$$

$$r^{2h} = I_h^{2h} r^h \Rightarrow f^{2h}$$

 Dados da malha atual

 x Dados inalterados

 0 Zeros

Suavização em $\Omega^{2h} : r^{2h} \rightarrow \Omega^{4h} : A^{2h} v^{2h} = f^{2h}$

$$r^{2h} = f^{2h} - A^{2h} v^{2h}$$

$$r^{4h} = I_{2h}^{4h} r^{2h} \Rightarrow f^{4h}$$

Suavização em $\Omega^{4h} : r^{4h} \rightarrow \Omega^{8h} : A^{4h} v^{4h} = f^{4h}$

$$r^{4h} = f^{4h} - A^{4h} v^{4h}$$

$$r^{8h} = I_{4h}^{8h} r^{4h} \Rightarrow f^{8h}$$

Resolva em $\Omega^{8h} :$

$$A^{8h} v^{8h} = f^{8h}$$

4.2 ARMAZENAMENTO (MEMÓRIA)

Pergunta: quanto o método Multigrid custa em termo de armazenamento?

Considere um problema d -dimensional com n^d pontos.

Suponha n como potência de 2.

Dois vetores (v e f) devem ser armazenados a cada nível.

A malha mais fina, Ω^h , exige $2n^d$ de memória.

A malha imediatamente mais grossa, Ω^{2h} , exige 2^{-d} vezes a memória exigida por Ω^h .

$$\text{Exemplo: } d=1, \quad 2^{-1} = \frac{1}{2}; \quad d=2, \quad 2^{-2} = \frac{1}{4}$$

A malha Ω^{4h} exige 2^{-d} vezes a memória exigida por Ω^{2h} , ou seja, $2^{-d} \times 2^{-d} = 4^{-d}$.

A malha Ω^{ph} exige p^{-d} vezes a memória exigida por Ω^h .

Adicionando:

$$\begin{aligned} \text{Memória} &= 2n^d + 2^{-d} (2n^d) + 4^{-d} (2n^d) + \dots + p^{-d} (2n^d) = \\ &= 2n^d + 2^{-d} (2n^d) + 2^{-2d} (2n^d) + \dots + 2^{-nd} (2n^d) \quad (\text{pois } p = 2^n) \end{aligned}$$

Então:

$$\begin{aligned} \text{Memória} &= 2n^d \left(1 + 2^{-d} + 2^{-2d} + \dots + 2^{-nd} \right) \\ &= 2n^d \left(\underbrace{1 + \frac{1}{2^d} + \frac{1}{2^{2d}} + \dots + \frac{1}{2^{nd}}}_{< \frac{1}{1-2^{-d}}} \right) < \frac{2n^d}{1-2^{-d}} \quad (\text{PA e PG}) \end{aligned}$$

Exemplo: Para $d=1$ (unidimensional)

$$\text{Memória} < \frac{2n^1}{1-2^{-1}} = 4n$$

Para malha mais fina, Ω^h : Memória = $2n$

Memória $< 4n$ (no máximo, o dobro da malha fina).

Observação: O custo de armazenamento (memória) decresce quando a dimensão do problema cresce. (Exercício 3)

4.3 COMPLEXIDADE (CUSTO COMPUTACIONAL)

Vamos estimar o custo computacional do método Multigrid.

Uma medida conveniente é WU (*Work Unit*), que é o custo para desenvolver um passo de suavização na malha mais fina. Neste caso, desconsidera-se o custo associado à restrição e interpolação, que geralmente é cerca de 10 a 20% do custo total de um ciclo.

Considere um ciclo V com um passo de suavização em cada nível ($v_1 = v_2 = 1$). Cada nível é visitado duas vezes e a malha Ω^{ph} exige p^{-d} WU. Adicionando estes custos, tem-se:

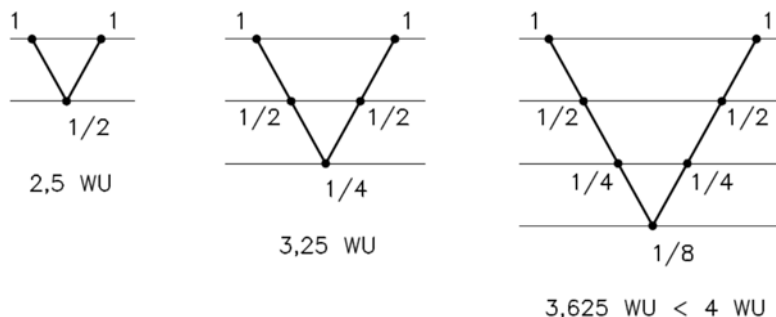
$$\begin{aligned} \text{Custo de um ciclo V} &= 2 + 2 \cdot 2^{-d} + 2 \cdot 4^{-d} + \dots + 2 \cdot p^{-d} = \\ &= 2 + 2 \cdot 2^{-d} + 2 \cdot 2^{-2d} + \dots + 2 \cdot 2^{-nd} \quad (\text{pois } p = 2^n) \end{aligned}$$

Então:

$$\text{Custo de um ciclo V} = 2 \underbrace{\left(1 + 2^{-d} + 2^{-2d} + \dots + 2^{-nd}\right)}_{< \frac{1}{1-2^{-d}}} < \frac{2}{1-2^{-d}} \text{ WU}$$

Exemplo: Para problemas 1D:

$$\text{Custo de um ciclo V} < \frac{2}{1-2^{-1}} = 4 \text{ WU}$$



Para outras dimensões, Exercício 4.

No caso de FMG, assumimos $\underbrace{v_0}_{FMG} = \underbrace{v_1 = v_2}_{\text{pré, pós}} = 1$.

Um ciclo completo iniciado de Ω^h custa aproximadamente $\frac{2}{1-2^{-d}}$ WU.

Iniciando de Ω^{2h} custa 2^{-d} de um ciclo completo.

Adicionando:

$$\text{Custo FMG} = \left(\frac{2}{1-2^{-d}} \right) \left(1 + 2^{-d} + 2^{-2d} + \dots + 2^{-nd} \right) < \frac{2}{(1-2^{-d})^2} \text{ WU}$$

Observação: Um ciclo FMG custa mais do que um ciclo V, mas a discrepância é menor para problemas com dimensões maiores. (Exercício 5)

Exercícios:

Lista 4: Exercícios 3, 4 e 5. (dia 23/nov)

Trabalho Computacional 3: Mescla dos Exercícios 13 e 14. (dia 14/dez)

Exerc. 13: $n = 32$, $\omega = 2/3$

Onde tiver ponderação completa, leia-se injeção.

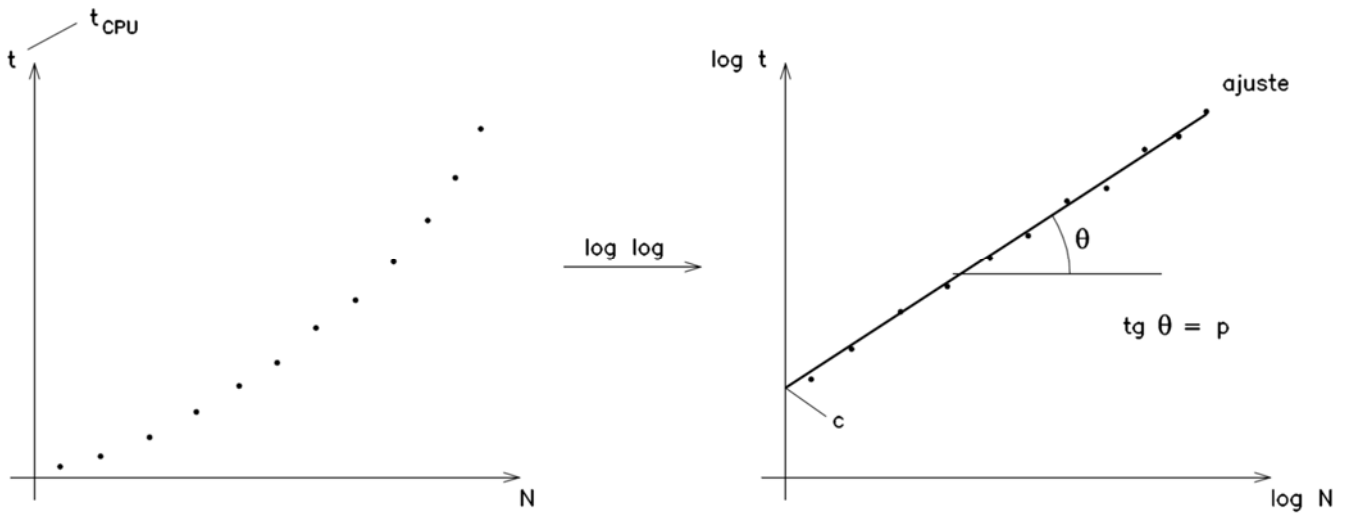
$C = 1$, $\sigma = 0$, $k = 2$

Exerc. 14: $\omega = 2/3$, retirar Gauss-Seidel *red-black*.

Trocar injeção por ponderação completa.

Formas alternativas de se medir a complexidade:

1) Tempo de processamento



$$t = c N^p \Rightarrow \log t = \log c N^p = \log c + \log N^p$$

$$\log t = p \log N + \log c$$

Se comparado com a reta $y = a x + b$, vemos que p é a inclinação da reta (coeficiente angular) e c é o coeficiente linear.

Mas note que temos que fazer um ajuste, pois os dados na escala bi-logarítmica nem sempre geram uma reta.

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}$$

$$\begin{bmatrix} n & \sum \log x_i \\ \sum \log x_i & \sum (\log x_i)^2 \end{bmatrix} \cdot \begin{bmatrix} a \\ \log c \end{bmatrix} = \begin{bmatrix} \sum \log y_i \\ \sum \log x_i \log y_i \end{bmatrix}$$

2) Contagem de *flops* (operações com números flutuantes)

Como medida para o desempenho do método Multigrid, nós podemos contar o número de operações com pontos flutuantes durante o processo iterativo, pois eles independem do *hardware*. Por simplicidade, contamos adição, subtração, multiplicação e divisão como sendo 1 *flop* cada.

Podemos ainda dividir o número de *flops* pelo número de pontos da malha mais fina.

Exemplo:

```
conta = 0
do i = 2, dim
  ii = i - 1
  fator = -a(i) / c(ii)
  c(i) = c(i) + fator * b(ii + 1)
  conta = conta + 5
end do
```

CAPÍTULO 5

CONVERGÊNCIA E EFICIÊNCIA (Trottenberg *et al*, 2001)

Frequentemente queremos determinar o fator de convergência q empiricamente, lembrando que a única quantidade disponível é o resíduo $r_{(m)}^h$ (número de iterações: $m = 1, 2, 3, \dots$).

Podemos medir, por exemplo: $q^{(m)} = \frac{\|r_{(m)}^h\|}{\|r_{(m-1)}^h\|}$

ou $\hat{q}^{(m)} = \sqrt[m]{\frac{\|r_{(m)}^h\|}{\|r_{(0)}^h\|}}$, em alguma norma apropriada, por exemplo, $\|\cdot\|_2$, onde $\hat{q}^{(m)}$ representa o fator

de redução médio do resíduo nas m iteradas.

As primeiras iteradas não refletem o comportamento assintótico das iterações do Multigrid, então podemos redefinir $\hat{q}^{(m)}$ como

$$\hat{q}^{(m)} = \sqrt[m-m_0]{\frac{\|r_{(m)}^h\|}{\|r_{(m_0)}^h\|}}$$

para m_0 pequeno (em geral, $2 \leq m_0 \leq 5$).

Exemplo: Equação de Poisson 2D no quadrado unitário, método Gauss-Seidel RB (*red-black*), ciclos $V(v_1, v_2)$ e $W(v_1, v_2)$, ponderação completa, interpolação bi-linear, $m_0 = 0$, $n = 256^2$ e $\varepsilon = 10^{-12}$.

Ciclo	$q^{(m)}$	$\hat{q}^{(m)}$	$\hat{q} < q$
V(1,1)	$q^{(12)} = 0,101$	$\hat{q}^{(12)} = 0,089$	$W < V$
W(1,1)	$q^{(11)} = 0,063$	$\hat{q}^{(11)} = 0,060$	

E a medida do fator de convergência é independente do tamanho da malha mais fina (h).

Exemplo: Mesmos dados, exceto n , considerando a malha mais grossa possível $h = 1/2$ e q .

Ciclo	$h = 1/512$	$h = 1/256$	$h = 1/128$	$h = 1/64$	$h = 1/32$	$h = 1/16$
V(1,1)	0,100	0,100	0,100	0,100	0,110	0,120
W(1,1)	0,063	0,063	0,063	0,063	0,063	0,067

Para escolher o Multigrid mais eficiente, é importante olhar na velocidade de convergência e na complexidade (seu custo). A velocidade de convergência pode ser medida pelo fator de convergência e seu custo pode ser medido pelo tempo de CPU.

Exemplo: Mesmos dados, exceto $n = 256^2$, redução de q por um fator de 10^{-12} , tempo em ms.

Ciclo	t_{CPU}	$q(V) > q(W)$
V(1,1)	759	
V(2,2)	799	$t_{CPU}(V) < t_{CPU}(W)$
W(1,1)	819	
W(2,2)	1469	

Observação: $q(V) > q(W)$, mas $t_{CPU}(V) < t_{CPU}(W)$, e portanto o ciclo V é mais eficiente que o ciclo W.