02/06732-9
Relatório Parcial
set/2002 a fev/2003

# Introdução aos métodos multigrid para resolução de sistemas lineares e não lineares

Aluno: Gabriel Haeser - RA: 001744
Orientadora: Prof. Dra. Márcia A. Gomes-Ruggiero

## RESUMO DO PLANO INICIAL

A proposta deste projeto é o estudo de técnicas de Multigrid para resolução dos sistemas lineares que resultam da discretização de Problemas de Valor de Contorno. O objetivo do trabalho é uma introdução ao tema que permita ao aluno entrar em contato com esta técnica e desenvolver programas em MatLab para resolver problemas de valor de contorno que modelam alguns problemas aplicados. Desta forma, o projeto inclui etapas que envolvem estudo teórico, elaboração de programas computacionais e resolução de problemas aplicados, possibilitando ao aluno o desenvolvimento completo de um trabalho em Matemática Aplicada.

## RESUMO DO PROJETO

Neste peróodo realizamos os seguintes estudos: elementos de multigrid; algoritmos multigrid para sistemas lineares; método gmres para resolução de sistemas lineares; implementação computacional dos ciclos multigrid.

# 1 Introdução

Os métodos Multigrid foram inicialmente aplicados em problemas de valor de contorno que aparecem em muitos problemas físicos. Como exemplo, considere o seguinte problema de valor de contorno, que poderia descrever a distribuição de temperatura ao longo de uma corda uniforme.

$$-u''(x) + \sigma u(x) = f(x), \qquad 0 < x < 1, \quad \sigma \geq 0 \tag{1}$$

sujeito às condições iniciais $u(0) = u(1) = 0$.

Embora possamos trabalhar com este problema analiticamente, é instrutivo que desenvolvamos métodos numéricos para sua solução. O método mais simples é provavelmente um método de diferenças finitas, onde o domínio do problema $x : 0 \leq x \leq 1$ é particionado em $N$ subintervalos, introduzindo uma malha com os pontos $x_j = jh$, onde $h = 1/N$ é o comprimento de cada intervalo. Isto estabelece uma malha de tamanho $h$, que denotamos por $\Omega^h$.

A cada um dos $N - 1$ pontos interiores à malha, a equação diferencial original (1) é substituída por uma aproximação por diferenças finitas de segunda ordem.

Neste trabalho, as derivadas serão aproximadas por diferenças finitas centradas, isto é:

$$u''(x) \simeq \frac{u(x_j - h) - 2u(x_j) + u(x_j + h)}{h^2}$$

$$u'(x) \simeq \frac{u(x_j + h) - u(x_j - h)}{2h}$$

e, se $v_j$ representa uma aproximação para a solução exata $u(x_j)$ teremos:

$$u''(x_j) \simeq \frac{v_{j-1} - 2v_j + v_{j+1}}{h^2}$$

$$u'(x_j) \simeq \frac{v_{j+1} - v_{j-1}}{2h}$$

O Problema (1) com $\sigma = 0$, após a discretização, resulta na resolução do sistema linear $(N-1)$x$(N-1)$:

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

com $v_0 = 0 = v_N$.

Ou na forma matricial: $\mathbf{Av} = \mathbf{f}$ onde:

$$\mathbf{A} = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & & & \ddots & \\ & & & & -1 & 2 \end{pmatrix} \quad \text{e} \quad \mathbf{f} = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_{N-1}) \end{pmatrix}$$

**A** é tridiagonal, simétrica, definida positiva.

Analogamente podemos formular uma versão em duas dimensões do problema. Considere a equação diferencial parcial de segunda ordem advinda do problema de convecção - difusão.

$$-\epsilon\left(u_{xx} + u_{yy}\right) + au_x + bu_y = f(x,y), \quad 0 < x < 1, \quad 0 < y < 1 \tag{2}$$

com $u(x,y) = 0$ na fronteira do quadrado unitário.

Como antes, este problema pode ser escrito de uma maneira discretizada, bastando definir os pontos $(x_i, y_j) = (ih, jh)$, onde $h = 1/N$. Esta malha bi-dimensional é também denominada $\Omega^h$. Substituindo as fórmulas de diferenças finitas centradas de segunda ordem na equação (2) temos o seguinte sistema linear:

$$-\epsilon(\frac{v_{i-1,j} - 2v_{ij} + v_{i+1,j}}{h^2} + \frac{v_{i,j-1} - 2v_{ij} + v_{i,j+1}}{h^2}) + a(\frac{u_{i+1,j} - u_{i-1,j}}{2h}) + b(\frac{u_{i,j+1} - u_{i,j-1}}{2h}) = f_{ij} \tag{3}$$

$u_{ij} = 0 \quad \text{se} \quad i = 0 \quad \text{ou} \quad i = N \quad \text{ou} \quad j = 0 \quad \text{ou} \quad j = N.$

## 2 Considerações iniciais

Suponhamos que se queira resolver o sistema linear $\mathbf{Au} = \mathbf{f}$ de modo iterativo, para tal denotamos por **u** a solução exata deste sistema e por **v** uma solução aproximada do sistema, gerada por um método iterativo.

Há duas importantes medições a respeito de **v** como uma aproximação de **u**. Uma é o *erro* que é dado por $\mathbf{e} = \mathbf{u} - \mathbf{v}$. Infelizmente o erro é tão inacessível quanto a solução exata propriamente dita. Entretanto, uma medida computável de quão bem **v** aproxima **u** é o *resíduo*, dado por $\mathbf{r} = \mathbf{f} - \mathbf{Av}$.

O resíduo **r** é simplesmente a quantidade pela qual a aproximação **v** falha em satisfazer o problema original $\mathbf{Au} = \mathbf{f}$.

Da definição do resíduo e da definição do erro podemos escrever

$$\mathbf{Av} = \mathbf{f} - \mathbf{r} \Rightarrow \mathbf{A}(\mathbf{u} - \mathbf{e}) = \mathbf{f} - \mathbf{r} \Rightarrow \mathbf{Ae} = \mathbf{r} + \mathbf{Au} - \mathbf{f} \Rightarrow \mathbf{Ae} = \mathbf{r}$$

que é uma importante relação entre o erro e o resíduo.

# 3 Introdução aos métodos Multigrid

Para iniciarmos uma abordagem aos métodos Multigrid, primeiramente é importante entendermos a performance das primeiras iterações, para tal, devemos proceder tanto analiticamente quanto experimentalmente.

É interessante trabalharmos com sistemas homogêneos, da forma $\mathbf{Au} = \mathbf{0}$ e utilizar aproximações iniciais arbitrárias para a solução pois desta forma a solução $\mathbf{u} = \mathbf{0}$ é conhecida, e o erro cometido com a aproximação $\mathbf{v}$ é simplesmente $-\mathbf{v}$.
Como aproximação inicial, $\mathbf{v_0}$, vamos considerar um vetor da forma

$$v_j = \sin\left(\frac{jk\pi}{N}\right)$$

para $0 \leq j \leq N$ e $1 \leq k \leq N-1$, denominado Fourier modes, onde $j$ denota a componente (ou ponto da malha) do vetor $\mathbf{v_0}$. O parâmetro $k$ é chamado número de onda, que indica o número de meias ondas de seno que constituem $\mathbf{v}$. A notação $\mathbf{v_k}$ será usada para denotar o vetor $\mathbf{v}$ desta forma, com parâmetro $k$. Note que valores pequenos de $k$ correspondem a ondas longas e suaves, enquanto valores maiores de $k$ correspondem a ondas altamente oscilatórias.

Ao utilizarmos um método iterativo convencional para tentarmos encontrar a solução do sistema, utilizando como aproximação inicial um vetor $\mathbf{v_k}$ da forma descrita anteriormente, vemos que o método consegue rapidamente diminuir a norma do erro se a onda for oscilatória ($k$ grande), mas o método falha ao tentarmos remover as componentes suaves do erro ($k$ pequeno).

A idéia então é adaptar os métodos iterativos convencionais para que consigam eliminar tanto as componentes suaves do erro quanto as oscilatórias.

Uma maneira de melhorar o resultado das iterações é usar uma boa aproximação inicial. Uma técnica para obtermos uma boa aproximação inicial é realizarmos algumas iterações em uma malha mais grossa, $\Omega^{2h}$, (com menos pontos) daí usamos o resultado obtido como uma aproximação inicial para a solução na malha fina, $\Omega^h$.

Iterações em uma malha mais grossa são mais baratas computacionalmente pois há menos variáveis e demonstra-se ainda que o fator de convergência se comporta como $1 - O\left(h^2\right)$, o que torna a convergência mais rápida na malha mais grossa.

Com a idéia da malha mais grossa em mente, podemos pensar com mais cuidado sobre suas aplicações. Assuma que um certo sistema tenha sido iterado de forma que o erro oscilatório tenha sido removido e o que resta são as componentes suaves do erro, afirmamos que esta onda parecerá mais oscilatória se estivermos trabalhando em uma malha mais grossa, digamos $\Omega^{2h}$.

Quando as iterações em uma malha $\Omega^h$ começam a ficar estáveis, significa que o que resta ainda são os erros suaves, o que sugere que mudemos para uma malha mais grossa, onde os erros suaves parecem mais oscilatórios e as iterações se tornam mais eficientes.

Uma primeira maneira de aplicarmos esta idéia é utilizando a técnica chamada de *nested iteration*, descrita da seguinte maneira:

Itere $\mathbf{Au} = \mathbf{f}$ em uma malha muito grossa.
$\vdots$
Itere $\mathbf{Au} = \mathbf{f}$ em $\Omega^{4h}$ para obter uma aproximação inicial em $\Omega^{2h}$.
Itere $\mathbf{Au} = \mathbf{f}$ em $\Omega^{2h}$ para obter uma aproximação inicial em $\Omega^{h}$.
Itere $\mathbf{Au} = \mathbf{f}$ em $\Omega^{h}$ para obter uma aproximação final para a solução.

Para podermos aplicar este método, devemos de alguma maneira descrever o problema original em uma malha grossa, o que será visto em seguida.
Uma segunda estratégia chamada de *correção em malha grossa* incorpora a idéia de usarmos a equação residual para iterarmos sobre o erro:

Itere $\mathbf{Au} = \mathbf{f}$ em $\Omega^{h}$ para obter uma aproximação inicial $\mathbf{v^h}$.
Calcule o resíduo $\mathbf{r} = \mathbf{f} - \mathbf{Av^h}$.
Itere na equação residual $\mathbf{Ae} = \mathbf{r}$ em $\Omega^{2h}$ para obter uma aproximação inicial para o erro $\mathbf{e^{2h}}$.
Corrija a aproximação obtida em $\Omega^{h}$ com o erro estimado obtido em $\Omega^{2h}$: $\mathbf{v^h} \leftarrow \mathbf{v^h} + \mathbf{e^{2h}}$.
Tendo iterado na malha fina até que os erros oscilatórios tenham sido removidos, iteramos então a equação residual na malha grossa para obtermos uma aproximação do erro, daí voltamos para a malha fina para corrigirmos a primeira aproximação obtida.

Para que possamos realmente implementar estes métodos, precisamos de mecanismos para transferir informações de uma malha para outra. Este é um procedimento comum em análise numérica que é geralmente denominado de *interpolação* ou *prolongação*. Há diversas maneiras de fazermos interpolações, mas felizmente para este propósito a maneira mais simples é bastante eficiente, por esta razão utilizaremos apenas interpolações lineares.

O operador de interpolação linear denotado por $\mathbf{I_{2h}^h}$, transforma um vetor de malha grossa em um vetor de malha fina de acordo com a regra $\mathbf{I_{2h}^h v^{2h}} = \mathbf{v^h}$, onde

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

$$v_{2j+1}^{h} = \frac{v_{j}^{2h} + v_{j+1}^{2h}}{2}$$

$$0 \leq j \leq \frac{N}{2} - 1.$$

Para o caso $N = 8$, este operador tem a forma

$$\mathbf{I_{2h}^h v^{2h}} = \frac{1}{2}\begin{pmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \\ & & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}_{2h} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{pmatrix}_h = \mathbf{v^h}$$

Note que esta interpolação só deve ser executada quando a parte oscilatória do erro já tiver sido removida.

Para problemas em duas dimensões, o operador de interpolação pode ser definido de uma maneira similar. Seja $\mathbf{I_{2h}^h}$ tal que $\mathbf{I_{2h}^h v^{2h}} = \mathbf{v^h}$, onde as componentes de $\mathbf{v^h}$ são dadas por

$$v^h_{2i,2j} = v^{2h}_{i,j}$$

$$v^h_{2i+1,2j} = \frac{v^{2h}_{i,j} + v^{2h}_{i+1,j}}{2}$$

$$v^h_{2i,2j+1} = \frac{v^{2h}_{i,j} + v^{2h}_{i,j+1}}{2}$$

$$v^h_{2i+1,2j+1} = \frac{v^{2h}_{i,j} + v^{2h}_{i+1,j} + v^{2h}_{i,j+1} + v^{2h}_{i+1,j+1}}{4}$$

$$0 \leq i, j \leq \frac{N}{2} - 1.$$

A segunda classe de operadores são aqueles que transferem informações da malha mais fina para a malha mais grossa, tais operadores são chamados *operadores de restrição* e são denotados por $\mathbf{I_h^{2h} v^h} = \mathbf{v^{2h}}$. O operador de restrição mais óbvio é a *injeção*, onde as componentes de $\mathbf{v^{2h}}$ são dadas por

$$v^{2h}_j = v^h_{2j}.$$

Um operador de restrição alternativo (chamado de *full weighting operator*), pode ser definido por

$$v^{2h}_j = \frac{v^h_{2j-1} + 2v^h_{2j} + v^h_{2j+1}}{4}, \quad 1 \leq j \leq \frac{N}{2} - 1.$$

Para o caso $N = 8$ este operador pode ser escrito como

$$\mathbf{I_h^{2h}v^h} = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{pmatrix}_h = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}_{2h} = \mathbf{v^{2h}}$$

Uma das razões para preferirmos o operador de restrição do tipo *full weighting* ao invés do operador injeção é que

$$\mathbf{I_{2h}^h} = c\left(\mathbf{I_h^{2h}}\right)^T, \quad c \in I\!R.$$

No caso de duas dimensões temos que o operador *full weighting* de restrição pode ser escrito como

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

# 4 Algoritmos

**Esquema de correção em malha grossa**

$$\mathbf{v^h} \leftarrow \mathrm{CG}\left(\mathbf{v^h}, \mathbf{f^h}\right)$$

1. Itere $n_1$ vezes em $\mathbf{A^h u^h = f^h}$ na malha $\Omega^h$ com aproximação inicial $\mathbf{v^h}$.
2. Calcule $\mathbf{r^{2h}} = \mathbf{I_h^{2h}}\left(\mathbf{f^h} - \mathbf{A^h v^h}\right)$.
3. Resolva $\mathbf{A^{2h} e^{2h} = r^{2h}}$ na malha $\Omega^{2h}$.
4. Transfira o erro da malha grossa para a fina e atualize $\mathbf{v^h}$ : $\mathbf{v^h} \leftarrow \mathbf{v^h} + \mathbf{I_{2h}^h e^{2h}}$.

Note que para podermos implementar este algoritmo, devemos primeiro definir $\mathbf{A^{2h}}$, o que será feito em seguida.

Na implementação deste algoritmo, podemos utilizar o mesmo método para resolver $\mathbf{A^{2h} e^{2h} = r^{2h}}$, bastando trabalhar na malha $\Omega^{4h}$, daí obtemos a equação do erro neste

nível, que também pode ser resolvido da mesma maneira. Claramente este algoritmo pode ser escrito recursivamente.

**Algoritmo recursivo do ciclo em V**

$$\mathbf{v^h} \leftarrow \mathrm{MV}^{\mathbf{h}}\left(\mathbf{v^h}, \mathbf{f^h}\right)$$

1. Itere $n_1$ vezes em $\mathbf{A^h u^h} = \mathbf{f^h}$ na malha $\Omega^h$ com aproximação inicial $\mathbf{v^h}$.
2. Se $\Omega^h$ = malha mais grossa então vá para 4.
   Senão
   (a) $\mathbf{f^{2h}} \leftarrow \mathbf{I_h^{2h}}\left(\mathbf{f^h} - \mathbf{A^h v^h}\right)$
   (b) $\mathbf{v^{2h}} \leftarrow \mathbf{0}$
   (c) $\mathbf{v^{2h}} \leftarrow \mathrm{MV}^{\mathbf{2h}}\left(\mathbf{v^{2h}}, \mathbf{f^{2h}}\right)$
3. Corrija $\mathbf{v^h} \leftarrow \mathbf{v^h} + \mathbf{I_{2h}^h v^{2h}}$
4. Itere $n_2$ vezes em $\mathbf{A^h u^h} = \mathbf{f^h}$ com aproximação inicial $\mathbf{v^h}$.

O ciclo em V é apenas um dos ciclos de uma família mais geral denominada ciclo $\mu$ definido recursivamente como:

$$\mathbf{v^h} \leftarrow \mathrm{M}\mu^{\mathbf{h}}\left(\mathbf{v^h}, \mathbf{f^h}\right)$$

1. Itere $n_1$ vezes em $\mathbf{A^h u^h} = \mathbf{f^h}$ na malha $\Omega^h$ com aproximação inicial $\mathbf{v^h}$.
2. Se $\Omega^h$ = malha mais grossa então vá para 4.
   Senão
   (a) $\mathbf{f^{2h}} \leftarrow \mathbf{I_h^{2h}}\left(\mathbf{f^h} - \mathbf{A^h v^h}\right)$
   (b) $\mathbf{v^{2h}} \leftarrow \mathbf{0}$
   (c) $\mathbf{v^{2h}} \leftarrow \mathrm{M}\mu^{\mathbf{2h}}\left(\mathbf{v^{2h}}, \mathbf{f^{2h}}\right)$ $\mu$ vezes
3. Corrija $\mathbf{v^h} \leftarrow \mathbf{v^h} + \mathbf{I_{2h}^h v^{2h}}$
4. Itere $n_2$ vezes em $\mathbf{A^h u^h} = \mathbf{f^h}$ com aproximação inicial $\mathbf{v^h}$.

Note que para $\mu = 1$ o algoritmo se reduz ao ciclo em V e para $\mu = 2$ o algoritmo é chamado de ciclo em W.

Ao olharmos para estes algoritmos vemos que é preciso obter uma boa aproximação inicial para cada malha, para isto, vamos utilizar as idéias das *nested iterations*. O algoritmo resultante é o chamado

**ciclo em V completo**

$$\mathbf{v^h} \leftarrow \mathrm{FMV^h}\left(\mathbf{v^h}, \mathbf{f^h}\right)$$

Inicialize $\mathbf{f^h}, \mathbf{f^{2h}}, \ldots; \mathbf{v^h}, \mathbf{v^{2h}}, \ldots$

Resolva ou itere na malha mais grossa.

$\vdots$

$\mathbf{v^{4h}} \leftarrow \mathbf{v^{4h}} + \mathbf{I_{8h}^{4h}}\mathbf{v^{8h}}$

$\mathbf{v^{4h}} \leftarrow \mathrm{MV^{4h}}\left(\mathbf{v^{4h}}, \mathbf{f^{4h}}\right)$

$\mathbf{v^{2h}} \leftarrow \mathbf{v^{2h}} + \mathbf{I_{4h}^{2h}}\mathbf{v^{4h}}$

$\mathbf{v^{2h}} \leftarrow \mathrm{MV^{2h}}\left(\mathbf{v^{2h}}, \mathbf{f^{2h}}\right)$

$\mathbf{v^h} \leftarrow \mathbf{v^h} + \mathbf{I_{2h}^h}\mathbf{v^{2h}}$

$\mathbf{v^h} \leftarrow \mathrm{MV^h}\left(\mathbf{v^h}, \mathbf{f^h}\right)$

Expresso recursivamente, o algoritmo tem a forma

$$\mathbf{v^h} \leftarrow \mathrm{FMV}\mu^{\mathbf{h}}\left(\mathbf{v^h}, \mathbf{f^h}\right)$$

1. Se $\Omega^h$ = malha mais grossa então vá para 3.
   Senão
   (a) $\mathbf{f^{2h}} \leftarrow \mathbf{I_h^{2h}}\left(\mathbf{f^h} - \mathbf{A^h}\mathbf{v^h}\right)$
   (b) $\mathbf{v^{2h}} \leftarrow \mathbf{0}$
   (c) $\mathbf{v^{2h}} \leftarrow \mathrm{FMV^{2h}}\left(\mathbf{v^{2h}}, \mathbf{f^{2h}}\right)$ $\mu$ vezes
2. Corrija $\mathbf{v^h} \leftarrow \mathbf{v^h} + \mathbf{I_{2h}^h}\mathbf{v^{2h}}$
3. $\mathbf{v^h} \leftarrow \mathrm{MV^h}\left(\mathbf{v^h}, \mathbf{f^h}\right)$ $n_0$ vezes.

**Cálculo de $\mathbf{A}^{2h}$**

Adotaremos a notação $\Omega^h$ como sendo não apenas a malha com espaçamento $h$, mas também o espaço dos vetores definidos nesta malha. Assuma que o erro $\mathbf{e^h} = \mathbf{u^h} - \mathbf{v^h}$ está inteiramente no espaço imagem de $\mathbf{I_{2h}^h}$, que será denotado por $\mathbf{Im}\left(\mathbf{I_{2h}^h}\right)$. Isto significa que para algum vetor $\mathbf{u^{2h}} \in \Omega^{2h}$, $\mathbf{e^h} = \mathbf{I_{2h}^h}\mathbf{u^{2h}}$.

Portanto, a equação residual em $\Omega^h$ pode ser escrita como

$$\mathbf{A^h}\mathbf{e^h} = \mathbf{A^h}\mathbf{I_{2h}^h}\mathbf{u^{2h}} = \mathbf{r^h}.$$

aplicando o operador de restrição $\mathbf{I_h^{2h}}$ em ambos os lados da equação temos:

$$\mathbf{I_h^{2h}}\mathbf{A^h}\mathbf{I_{2h}^h}\mathbf{u^{2h}} = \mathbf{I_h^{2h}}\mathbf{r^h}.$$

ou

$$\mathbf{A^{2h}u^{2h}} = \mathbf{r^{2h}}$$

O que nos dá uma definição plausível para a matriz em malhas grossas, chamado de *condição de Galerkin*, dado por

$$\mathbf{A^{2h}} = \mathbf{I_h^{2h}A^hI_{2h}^h}.$$

O argumento acima se baseia no fato de que $\mathbf{e^h}$ pertence ao espaço imagem de $\mathbf{I_{2h}^h}$ , o que não é sempre verdade. Caso fosse, ao resolver a equação residual em $\Omega^{2h}$ exatamente e corrigir a solução na malha fina, teríamos a solução exata. Entretanto a argumentação nos dá uma boa definição de $\mathbf{A^{2h}}$.

# 5 Experimentos Computacionais

Para começarmos a implementar os métodos multigrid foi preciso antes criar procedimentos para efetuar a troca de malha de um dado vetor. Para transferir vetores da malha grossa para a malha fina foi usado o operador de interpolação e para transferir vetores da malha fina para a malha grossa foi utilizado o operador *full weighting*, ambos descritos no capítulo 4 acima. Tais procedimentos foram feitos em MatLab (versão 6.0.0.88 Release 12), inicialmente utilizando o comando de repetição FOR para cada vetor a ser transferido de malha, o que tornou mais fácil a implementação, porém os resultados não foram bons, pois o tempo de espera era muito e a dimensão máxima era pequena. Outro fato que retardava o programa era o fato de que os vetores eram tratados como matrizes, isto foi feito para facilitar a programação porém era necessário outra rotina para transformar esta matriz em um vetor.

Em seguida abandonamos esta idéia e nos empenhamos em criar os operadores para troca de malha em forma de matriz, o que melhorou e muito a velocidade de execução das operações e ainda aumentou o tamanho máximo da malha em que podemos trabalhar, isto pois as matrizes com que trabalhamos são facilmente transformadas em esparsas pelo MatLab. Além disso podemos utilizar o fato de que $\mathbf{I_{2h}^h} = 4\left(\mathbf{I_h^{2h}}\right)^T$, e também podemos utilizar estas matrizes para criar uma matriz na malha grossa que representa a matriz original na malha fina, desta forma não é necessário que a matriz do sistema possa ser construída através de um procedimento em função do espaçamento da malha, isto é, podemos tentar utilizar os métodos multigrid para resolver qualquer tipo de sistema linear e não apenas os oriundos da discretização de equações diferenciais.

Após os procedimentos terem sido criados e testados, começamos a trabalhar com a implementação dos métodos multigrid. O programa foi implementado utilizando o GMRES com restart do MatLab para iterar no sistema linear, o método foi estudado pelas referências [9] e [10].

## 5.1 Primeira tentativa de Implementação em duas malhas

O algoritmo utilizado consite em:

1. Iterar na malha fina, $\Omega^h$, até a precisão de $10^{-1}$.
2. Trazer a solução obtida para a malha grossa, $\Omega^{2h}$, e iterar no sistema até a precisão de $10^{-7}$.
3. Voltar para a malha fina e iterar até a precisão de $10^{-8}$.

Aqui comparamos também a eficácia do método quando não iteramos inicialmente na malha fina.

Os resultados obtidos podem ser vistos na tabela 1, onde o problema utilizado para teste foi o problema (2) do capítulo 2 com, $a = 0$, $b = 0$, $\epsilon = 1$ e $f(x, y)$ escolhido tal que a solução exata seja $u(x, y) = 1000xy(1 - x)(1 - y)e^{x^{4.5}}$.

A discretização por diferenças centrais, usando $h = 0.03125$ , resultou num sistema linear de dimensão 31, que corresponde à malha fina. Neste caso, o problema na malha grossa implica na resolução de um sistema linear de dimensão 15.

Testamos ainda para $h = 0.015625$, o que resulta num sistema linear de dimensão 63 na malha fina e dimensão 31 na malha grossa. Fixamos a aproximação inicial como sendo o vetor nulo.

Na tabela indicamos a dimensão da malha fina, na primeira coluna, o total de iterações usadas no restart do GMRES, na segunda coluna, o total de iterações no primeiro ciclo na malha fina, malha grossa e malha fina novamente, respectivamente nas colunas 3, 4 e 5 e finalmente a última coluna o total de iterações realizadas resolvendo o sistema linear somente na malha fina.

Como pode ser verificado na tabela 1, a utilização do método multigrid é mais vantajosa para sistemas de maior dimensão, e o método é ainda mais interessante quando não iteramos inicialmente na malha fina. Para podermos comparar a eficiência do método devemos lembrar que iterações na malha grossa são mais baratas computacionalmente do que iterações na malha fina, na figura 1 mostramos os resultados das iterações em cada malha.

| Número de pontos da malha | restart | Iterações iniciais na malha fina | Iterações na malha grossa | Iterações na malha fina | Iterações na malha fina com $x_0 = 0$ |
|---|---|---|---|---|---|
| 15x15=225 | 30 | 11 | 16 | 51 | 43 |
| 15x15=225 | 30 | - | 18 | 51 | 43 |
| 15x15=225 | 50 | 11 | 16 | 45 | 40 |
| 15x15=225 | 50 | - | 18 | 45 | 40 |
| 31x31=961 | 30 | 23 | 36 | 117 | 195 |
| 31x31=961 | 30 | - | 40 | 117 | 195 |
| 31x31=961 | 50 | 23 | 34 | 106 | 118 |
| 31x31=961 | 50 | - | 37 | 106 | 118 |
| 63x63=3969 | 30 | 68 | 93 | 399 | 639 |
| 63x63=3969 | 30 | - | 162 | 399 | 639 |
| 63x63=3969 | 50 | 48 | 71 | 283 | 429 |
| 63x63=3969 | 50 | - | 88 | 283 | 429 |
| 127x127=16129 | 30 | 202 | 484 | 1403 | sem memória |
| 127x127=16129 | 30 | - | 556 | 1196 | sem memória |

Tabela 1 - Resultados obtidos trocando uma vez de malha

## 5.2 *Nested iterations* em mais de duas malhas

Implementamos o algoritmo *nested iterations* com mais de uma malha no MatLab tanto iniciando as iterações do GMRES na malha inicial, quanto iniciando diretamente na malha mais grossa, lembrando que a idéia deste método é trazer da malha grossa uma boa aproximação inicial para a malha fina.

Os resultados apresentados na tabela 2 foram obtidos iterando inicialmente na malha mais grossa com precisão de $10^{-7}$ em cada malha, a não ser na malha final (mais fina) onde a precisão é de $10^{-8}$.

A justificativa de não efetuarmos as iterações iniciando na malha fina são os resultados pouco satisfatórios que podem ser vistos na tabela 1.

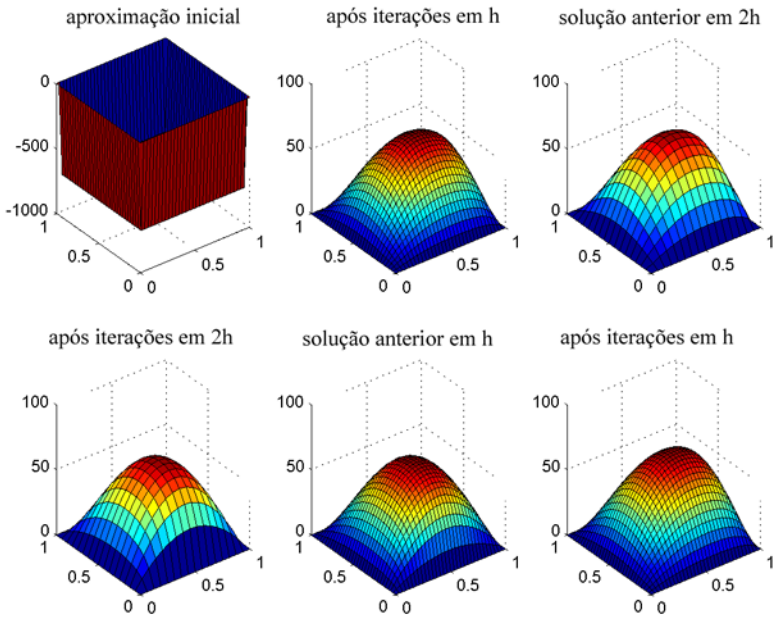


Figura 1: Resolução do problema (2) do capítulo 2 com $a = 0$, $b = 0$, $\epsilon = 1$ e $f(x, y)$ escolhido tal que a solução exata seja $1000xy(1-x)(1-y)e^{x^{4.5}}$, discretizado com 961 pontos e trocando uma vez de malha. A primeira figura representa a aproximação inicial constante utilizada, a segunda é a solução obtida quando iterarmos na malha fina (h) até a precisão de $10^{-1}$, a terceira figura é esta mesma solução transferida para a malha grossa (2h), a quarta é a solução obtida pelo GMRES ao iterarmos no sistema até a precisão de $10^{-7}$ com aproximação inicial obtida anteriormente, a quinta figura é esta solução transportada para a malha fina e a sexta figura é a solução obtida após iterar no sistema até a precisão de $10^{-8}$ com aproximação inicial obtida anteriormente.

Na tabela 2 abaixo o sistema utilizado foi o problema (2) do capítulo 2 com $a = 0$, $b = 0$, $\epsilon = 1$, com a função $f(x, y)$ escolhida tal que a solução exata seja $1000xy(1-x)(1-y)e^{x^{4.5}}$.

A última coluna da tabela 2 é o número de iterações realizadas ao utilizarmos uma aproximação inicial nula diretamente na malha mais fina.

Na tabela 2, o número de iterações realizadas quando a dimensão do sistema é 127x127=16129 na malha mais fina (h) é do GMRES com restart=30, já a resolução do sistema iterando diretamente na malha fina não foi possível de ser realizada por falta de capacidade com-

putacional, o que mostra a eficiência do método multigrid empregado.

| aproximação inicial nula com solução exata $1000xy(1-x)(1-y)e^{x^{4.5}}$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| n | 32h | 16h | 8h | 4h | 2h | h | direto |
| 15x15=225 | | | | | 18 | 45 | 40 |
| 15x15=225 | | | | 5 | 20 | 45 | 40 |
| 31x31=961 | | | | | 37 | 82 | 81 |
| 31x31=961 | | | | 18 | 41 | 82 | 81 |
| 31x31=961 | | | 5 | 20 | 41 | 82 | 81 |
| 63x63=3969 | | | | | 75 | 150 | 164 |
| 63x63=3969 | | | | 37 | 72 | 150 | 164 |
| 63x63=3969 | | | 18 | 41 | 72 | 150 | 164 |
| 63x63=3969 | | 5 | 20 | 41 | 72 | 150 | 164 |
| 127x127=16129 | | | | | 151 | 1196 | - |
| 127x127=16129 | | | | 75 | 132 | 1196 | - |
| 127x127=16129 | | | 37 | 72 | 132 | 1196 | - |
| 127x127=16129 | | 18 | 41 | 72 | 132 | 1196 | - |
| 127x127=16129 | 5 | 20 | 41 | 72 | 132 | 1196 | - |

Tabela 2 - Aproximação inicial nula e solução exata $1000xy(1-x)(1-y)e^{x^{4.5}}$

Na tabela 3 abaixo mostramos os resultados iterando inicialmente na malha mais grossa ao escolhermos uma aproximação inicial aleatória para resolvermos o problema (2) do capítulo 2 com $a = 0$, $b = 0$, $\epsilon = 1$ e $f(x, y)$ escolhido tal que a solução exata seja constante, igual a 5.

Na dimensão de 127x127=16129 as iterações na malha mais fina (colunas h e direto) foram feitas com restart=30 do GMRES

| aproximação inicial aleatória entre -5 e 15 com solução exata constante, igual a 5. | | | | | | | |
|---|---|---|---|---|---|---|---|
| n | 32h | 16h | 8h | 4h | 2h | h | direto |
| 15x15=225 | | | | | 22 | 29 | 50 |
| 15x15=225 | | | | 5 | 9 | 29 | 50 |
| 31x31=961 | | | | | 46 | 59 | 99 |
| 31x31=961 | | | | 22 | 27 | 59 | 97 |
| 31x31=961 | | | 5 | 9 | 27 | 59 | 99 |
| 63x63=3969 | | | | | 91 | 111 | 185 |
| 63x63=3969 | | | | 46 | 54 | 111 | 188 |
| 63x63=3969 | | | 22 | 27 | 54 | 111 | 186 |
| 63x63=3969 | | 5 | 9 | 27 | 54 | 111 | 186 |
| 127x127=16129 | | | | | 172 | 751 | 1183 |
| 127x127=16129 | | | | 91 | 103 | 751 | 1183 |
| 127x127=16129 | | | 46 | 54 | 103 | 751 | 1183 |
| 127x127=16129 | | 22 | 27 | 54 | 103 | 751 | 1183 |
| 127x127=16129 | 5 | 9 | 27 | 54 | 103 | 751 | 1183 |

Tabela 3 - Aproximação inicial aleatória entre -5 e 15 com solução exata constante, igual a 5.

Na tabela 4 abaixo, estamos tratando do mesmo problema anterior, apenas utilizando como aproximação inicial os vetores *fourier mode*.

| aproximação inicial *fourier mode* com solução exata constante, igual a 5. | | | | | | | |
|---|---|---|---|---|---|---|---|
| n | k | 32h | 16h | 8h | 4h | 2h | h |
| 15x15=225 | 220 | | | | | 21 | 29 |
| 15x15=225 | 22 | | | | | 21 | 29 |
| 15x15=225 | 220 | | | | 5 | 9 | 29 |
| 15x15=225 | 22 | | | | 5 | 9 | 29 |
| 31x31=961 | 220 | | | | | 32 | 59 |
| 31x31=961 | 22 | | | | | 43 | 59 |
| 31x31=961 | 220 | | | | 21 | 27 | 59 |
| 31x31=961 | 22 | | | | 21 | 27 | 59 |
| 31x31=961 | 220 | | | 5 | 9 | 27 | 59 |
| 31x31=961 | 22 | | | 5 | 9 | 27 | 59 |
| 63x63=3969 | 220 | | | | | 63 | 111 |
| 63x63=3969 | 22 | | | | | 73 | 111 |
| 63x63=3969 | 220 | | | | 32 | 54 | 111 |
| 63x63=3969 | 22 | | | | 43 | 54 | 111 |
| 63x63=3969 | 220 | | | 21 | 27 | 54 | 111 |
| 63x63=3969 | 22 | | | 21 | 27 | 54 | 111 |
| 63x63=3969 | 220 | | 5 | 9 | 27 | 54 | 111 |
| 63x63=3969 | 22 | | 5 | 9 | 27 | 54 | 111 |
| 127x127=16129 | 220 | | | | | 125 | 751 |
| 127x127=16129 | 22 | | | | | 136 | 751 |
| 127x127=16129 | 220 | | | | 63 | 103 | 751 |
| 127x127=16129 | 22 | | | | 73 | 103 | 751 |
| 127x127=16129 | 220 | | | 32 | 54 | 103 | 751 |
| 127x127=16129 | 22 | | | 43 | 54 | 103 | 751 |
| 127x127=16129 | 220 | | 21 | 27 | 54 | 103 | 751 |
| 127x127=16129 | 22 | | 21 | 27 | 54 | 103 | 751 |
| 127x127=16129 | 220 | 5 | 9 | 27 | 54 | 103 | 751 |
| 127x127=16129 | 22 | 5 | 9 | 27 | 54 | 103 | 751 |

Tabela 4 - Aproximação inicial *fourier mode* com solução exata constante, igual a 5.

Com base nos dados das tabelas 3 e 4 podemos ver que o método começa a fazer sentido, pois quanto mais malhas utilizamos, menos iterações estamos fazendo (lembrando que iterações em malhas mais grossas são mais baratas computacionalmente).

## 5.3 Ciclo em V

A idéia do ciclo em V é utilizar tanto a correção em malha grossa quanto as *nested iterations*. O método consiste em iterar inicialmente na malha fina, em seguida calcular o

resíduo e iterar na equação do erro-resíduo: $\mathbf{Ae} = \mathbf{r}$ um número $v_1$ de vezes. Calculamos novamente o resíduo, passamos para a próxima malha grossa e iteramos novamente um número $v_1$ de vezes, continuamos este processo até chegar à malha mais grossa, que no nosso caso é a malha com 3 pontos ($h = 0.25$), nesta malha resolvemos o sistema exatamente, e levamos a solução até a próxima malha mais fina, que será somada à solução já obtida anteriormente nesta malha e iteramos um número $v_2$ de vezes, passamos esta solução para a próxima malha mais fina e repetimos o processo até chegarmos a malha fina inicial.

As tabelas 5, 6 e 7 abaixo mostram os resultados obtidos ao empregarmos o ciclo em V no problema (2) do capítulo 2 com $a = 0$, $b = 0$, $\epsilon = 1$, com a função $f(x, y)$ escolhida tal que a solução exata seja $1000xy(1 - x)(1 - y)e^{x^{4.5}}$. A idéia deste teste é comparar o algoritmo do ciclo em V com iterações diretas na malha fina com aproximação inicial nula, para tal fixamos o número máximo de iterações em cada malha e calculamos o resíduo, em seguida utilizamos o GMRES para verificar quantas iterações na malha fina com aproximação inicial nula seriam necessárias para obter o mesmo resíduo obtido pelo ciclo em V.

| ciclo em V com dimensão 15x15=225 | | | | |
|---|---|---|---|---|
| malha | $v_1 = 10$<br>$v_2 = 10$ | $v_1 = 20$<br>$v_2 = 20$ | $v_1 = 30$<br>$v_2 = 30$ | $v_1 = 40$<br>$v_2 = 40$ |
| h | 10 | 20 | 30 | 40 |
| 2h | 10 | 20 | 23 | 25 |
| 4h | exata | exata | exata | exata |
| 2h | 10 | 20 | 3 | 3 |
| h | 10 | 20 | 30 | 40 |
| resíduo | $1.7\text{x}10^{-3}$ | $1.9\text{x}10^{-6}$ | $1.7\text{x}10^{-10}$ | $2.6\text{x}10^{-15}$ |
| direto | 21 | 33 | 45 | 56 |
| resíduo | $1.1\text{x}10^{-3}$ | $9.8\text{x}10^{-7}$ | $9.2\text{x}10^{-11}$ | $1.9\text{x}10^{-14}$ |

Tabela 5 - Ciclo em V com dimensão 15x15=225

| ciclo em V com dimensão 31x31=961 | | | | |
|---|---|---|---|---|
| malha | $v_1 = 10$ $v_2 = 10$ | $v_1 = 30$ $v_2 = 30$ | $v_1 = 50$ $v_2 = 50$ | $v_1 = 70$ $v_2 = 70$ |
| h | 10 | 30 | 50 | 70 |
| 2h | 10 | 30 | 50 | 64 |
| 4h | 10 | 23 | 25 | 25 |
| 8h | exata | exata | exata | exata |
| 4h | 10 | 4 | 1 | 3 |
| 2h | 10 | 30 | 20 | 3 |
| h | 10 | 30 | 50 | 70 |
| resíduo | $3.2\text{x}10^{-3}$ | $3.0\text{x}10^{-5}$ | $2.6\text{x}10^{-9}$ | $3.2\text{x}10^{-14}$ |
| direto | 42 | 56 | 86 | 111 |
| resíduo | $2.1\text{x}10^{-3}$ | $2.8\text{x}10^{-5}$ | $2.1\text{x}10^{-9}$ | $1.4\text{x}10^{-13}$ |

Tabela 6 - Ciclo em V com dimensão 31x31=961

| ciclo em V com dimensão 63x63=3969 | | | | | |
|---|---|---|---|---|---|
| malha | $v_1 = 30$ $v_2 = 30$ | $v_1 = 60$ $v_2 = 60$ | $v_1 = 80$ $v_2 = 80$ | $v_1 = 120$ $v_2 = 120$ | $v_1 = 180$ $v_2 = 180$ |
| h | 30 | 60 | 80 | 120 | 150 |
| 2h | 30 | 60 | 80 | 120 | 129 |
| 4h | 30 | 60 | 63 | 67 | 68 |
| 8h | 23 | 25 | 25 | 25 | 25 |
| 16h | exata | exata | exata | exata | exata |
| 8h | 1 | 2 | 4 | 4 | 1 |
| 4h | 30 | 4 | 3 | 4 | 3 |
| 2h | 30 | 60 | 80 | 3 | 4 |
| h | 30 | 60 | 80 | 120 | 150 |
| resíduo | $4.2\text{x}10^{-4}$ | $8.8\text{x}10^{-6}$ | $4.6\text{x}10^{-7}$ | $6.3\text{x}10^{-12}$ | $3.0\text{x}10^{-14}$ |
| direto | 93 | 122 | 142 | 200 | 219 |
| resíduo | $3.9\text{x}10^{-4}$ | $8.1\text{x}10^{-6}$ | $4.2\text{x}10^{-7}$ | $5.7\text{x}10^{-12}$ | $1.5\text{x}10^{-12}$ |

Tabela 7 - Ciclo em V com dimensão 63x63=3969

Como podemos ver nas tabelas 5, 6 e 7 acima, em malhas mais grossas, o GMRES não consegue iterar o número de vezes pedido, isto ocorre pois a precisão da solução atual já está próxima da precisão da máquina. Podemos notar também que para $v_1$ e $v_2$ grandes o mesmo resíduo obtido pelo ciclo em V não consegue ser obtido pelas iterações diretas na malha fina, isto está de acordo com a teoria, pois com o ciclo em V conseguimos remover as componentes do erro que são menos oscilatórias, tornando o resíduo menor.

## 5.4 Ciclo em V completo

A idéia do ciclo em V completo é buscar uma boa aproximação inicial para o ciclo em V, para isto, iniciamos resolvendo o sistema na malha mais grossa possível e levamos a solução para a próxima malha fina, em seguida aplicamos o procedimento "ciclo em V" descrito acima e a solução obtida é levada para a próxima malha mais fina. Repetimos este procedimento até chegarmos a malha fina original, daí utilizamos a solução obtida até o momento como aproximação inicial para o ciclo em V.

Os resultados obtidos podem ser vistos nas tabelas 8, 9 e 10 abaixo onde o ciclo em V completo foi aplicado ao problema (2) do capítulo 2 com $a = 0$, $b = 0$, $\epsilon = 1$, com a função $f(x, y)$ escolhida tal que a solução exata seja $1000xy(1-x)(1-y)e^{x^{4.5}}$. As duas últimas linhas são o número de iteração obtidas, com aproximação inicial nula, ao iterarmos no sistema diretamente na malha mais fina até que o resíduo esteja próximo do obtido pelo ciclo em V completo.

| ciclo em V completo com dimensão 15x15=225 | | | | |
|---|---|---|---|---|
| malha | $v_1 = 10$ $v_2 = 10$ | $v_1 = 20$ $v_2 = 20$ | $v_1 = 30$ $v_2 = 30$ | $v_1 = 40$ $v_2 = 40$ |
| 4h | exata | exata | exata | exata |
| 2h | 10 | 20 | 21 | 21 |
| 4h | exata | exata | exata | exata |
| 2h | 10 | 15 | 1 | 1 |
| h | 10 | 20 | 30 | 40 |
| 2h | 10 | 20 | 23 | 25 |
| 4h | exata | exata | exata | exata |
| 2h | 10 | 14 | 2 | 4 |
| h | 10 | 20 | 30 | 28 |
| resíduo | $1.6\text{x}10^{-4}$ | $3.0\text{x}10^{-6}$ | $7.4\text{x}10^{-11}$ | $1.2\text{x}10^{-14}$ |
| direto | 25 | 32 | 46 | 56 |
| resíduo | $1.1\text{x}10^{-4}$ | $2.3\text{x}10^{-6}$ | $4.0\text{x}10^{-11}$ | $1.9\text{x}10^{-14}$ |

Tabela 8 - Ciclo em V completo com dimensão 15x15=225

| ciclo em V completo com dimensão 31x31=961 | | | | |
|---|---|---|---|---|
| malha | $v_1 = 10$<br>$v_2 = 10$ | $v_1 = 30$<br>$v_2 = 30$ | $v_1 = 50$<br>$v_2 = 50$ | $v_1 = 70$<br>$v_2 = 70$ |
| 8h | exata | exata | exata | exata |
| 4h | 10 | 21 | 21 | 21 |
| 8h | exata | exata | exata | exata |
| 4h | 10 | 1 | 1 | 1 |
| 2h | 10 | 30 | 50 | 56 |
| 4h | 10 | 23 | 25 | 25 |
| 8h | exata | exata | exata | exata |
| 4h | 10 | 2 | 3 | 2 |
| 2h | 10 | 30 | 9 | 2 |
| h | 10 | 30 | 50 | 70 |
| 2h | 10 | 30 | 50 | 61 |
| 4h | 10 | 23 | 25 | 25 |
| 8h | exata | exata | exata | exata |
| 4h | 10 | 1 | 2 | 3 |
| 2h | 10 | 30 | 16 | 2 |
| h | 10 | 30 | 50 | 70 |
| resíduo | $1.0\text{x}10^{-4}$ | $2.0\text{x}10^{-6}$ | $4.8\text{x}10^{-10}$ | $3.3\text{x}10^{-14}$ |
| direto | 52 | 66 | 89 | 111 |
| resíduo | $8.2\text{x}10^{-5}$ | $1.8\text{x}10^{-6}$ | $4.7\text{x}10^{-10}$ | $1.4\text{x}10^{-13}$ |

Tabela 9 - Ciclo em V completo com dimensão 31x31=961

| ciclo em V completo com dimensão 63x63=3969 | | | | | | |
|---|---|---|---|---|---|---|
| malha | $v_1 = 30$ $v_2 = 30$ | $v_1 = 60$ $v_2 = 60$ | $v_1 = 80$ $v_2 = 80$ | $v_1 = 120$ $v_2 = 120$ | $v_1 = 180$ $v_2 = 180$ | $v_1 = 350$ $v_2 = 350$ |
| 16h | exata | exata | exata | exata | exata | exata |
| 8h | 21 | 21 | 21 | 21 | 21 | 21 |
| 16h | exata | exata | exata | exata | exata | exata |
| 8h | 1 | 1 | 1 | 1 | 1 | 1 |
| 4h | 30 | 56 | 56 | 56 | 56 | 56 |
| 8h | 23 | 25 | 25 | 25 | 25 | 25 |
| 16h | exata | exata | exata | exata | exata | exata |
| 8h | 2 | 2 | 2 | 2 | 2 | 2 |
| 4h | 30 | 2 | 2 | 2 | 2 | 2 |
| 2h | 30 | 60 | 80 | 113 | 113 | 113 |
| 4h | 30 | 60 | 65 | 71 | 71 | 71 |
| 8h | 23 | 25 | 25 | 25 | 25 | 25 |
| 16h | exata | exata | exata | exata | exata | exata |
| 8h | 1 | 3 | 2 | 4 | 4 | 4 |
| 4h | 30 | 3 | 2 | 3 | 3 | 3 |
| 2h | 30 | 60 | 54 | 3 | 3 | 3 |
| h | 30 | 60 | 80 | 120 | 180 | 212 |
| 2h | 30 | 60 | 80 | 120 | 132 | 145 |
| 4h | 30 | 60 | 64 | 68 | 69 | 69 |
| 8h | 23 | 25 | 25 | 25 | 25 | 25 |
| 16h | exata | exata | exata | exata | exata | exata |
| 8h | 2 | 3 | 3 | 2 | 1 | 2 |
| 4h | 30 | 3 | 2 | 4 | 2 | 2 |
| 2h | 30 | 60 | 80 | 4 | 4 | 5 |
| h | 30 | 60 | 80 | 120 | 28 | 2 |
| resíduo | $1.5\text{x}10^{-6}$ | $1.4\text{x}10^{-7}$ | $1.3\text{x}10^{-8}$ | $4.1\text{x}10^{-13}$ | $1.3\text{x}10^{-13}$ | $4.8\text{x}10^{-14}$ |
| direto | 135 | 149 | 162 | 219 | 219 | 219 |
| resíduo | $1.3\text{x}10^{-6}$ | $1.3\text{x}10^{-7}$ | $1.2\text{x}10^{-8}$ | $1.5\text{x}10^{-12}$ | $1.5\text{x}10^{-12}$ | $1.5\text{x}10^{-12}$ |

Tabela 10 - Ciclo em V completo com dimensão 63x63=3969

Tendo em vista os dados das tabelas 5, 6, 7, 8, 9 e 10 podemos ver que os métodos multigrid empregados nem sempre resolvem o sistema com menos iterações, mas sim conseguem resolver o sistema com uma melhor precisão.

A implementação dos ciclos multigrid está em fase de ajustes, pois serão realizados testes com outras equações, e a partir daí serão melhor analisados e comparados os diferentes algoritmos (ciclos) e parâmetros $v_1$ e $v_2$ do algoritmo.

## 5.5 O cálculo de $\mathbf{A^{2h}}$

Nos experimentos acima a matriz sempre foi criada através de um procedimento em função do espaçamento, mas podemos ver que para o problema (2) do capítulo 2 com $a = 0$, $b = 0$, $\epsilon = 1$, podemos criar a matriz na malha $\Omega^{2h}$ através da matriz na malha $\Omega^h$, bastando aplicar a condição de Galerkin, descrita no capítulo 5 dada por:

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

Neste capítulo vamos verificar se a matriz $\mathbf{A^{2h}}$ descrita acima representa bem a matriz $\mathbf{A^h}$ original, para tal, consideramos $h = 0.03125$ o que corresponde a uma discretização de 31 pontos. Criamos então a matriz $\mathbf{A^h}$ pelo procedimento descrito acima em função do espaçamento e criamos as matrizes $\mathbf{I_h^{2h}}$ e $\mathbf{I_{2h}^h}$, para em seguida criarmos a matriz $\mathbf{A^{2h}}$ através da condição de Galerkin. Em seguida criamos o vetor $\mathbf{f^{2h}}$ escolhida tal que a solução exata do sistema seja $1000xy(1-x)(1-y)e^{x^{4.5}}$. Resolvemos então o sistema $\mathbf{A^{2h}u^{2h}} = \mathbf{f^{2h}}$ exatamente e em seguida plotamos o vetor $\mathbf{u^h} = \mathbf{I_{2h}^h u^{2h}}$, inserindo as condições de contorno nulas, que pode ser visto no primeiro gráfico da figura 2. Criamos então a matriz $\mathbf{A_p^{2h}}$ através do precedimento em função do espaçamento, resolvemos o sistema $\mathbf{A_p^{2h}v^{2h}} = \mathbf{f^{2h}}$ e em seguida plotamos o vetor $\mathbf{v^h} = \mathbf{I_{2h}^h v^{2h}}$, inserindo as condições de contorno nulas, que pode ser visto no segundo gráfico da figura 2. Resolvemos também o sistema $\mathbf{A^h w^h} = \mathbf{f^h}$ exatamente, cuja solução pode ser vista no terceiro gráfico da figura 2.

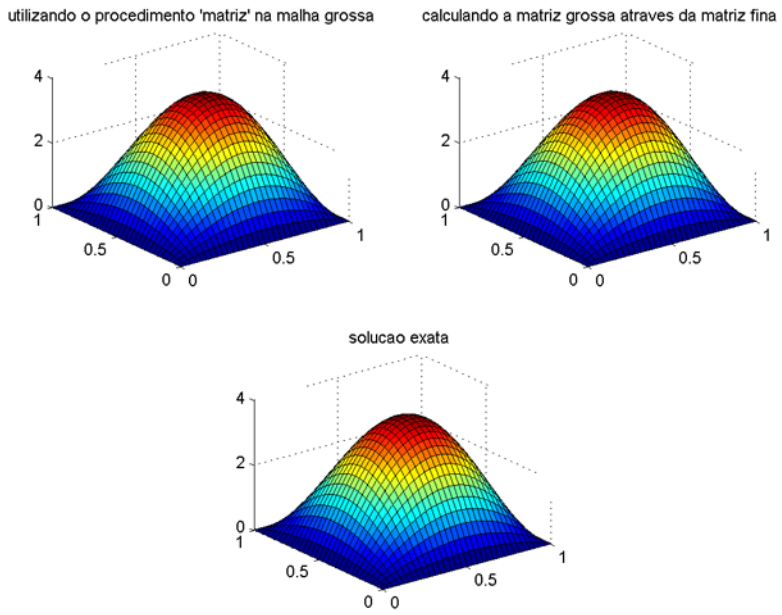


Figura 2: verificação da condição de Galerkin

Podemos ver pela figura 2 acima que a matriz $\mathbf{A^{2h}}$ criada pela condição de Galerkin representa muito bem o sistema.

# Referências

[1] Brandt, A., *Multi-Level Adaptive Solutions to Boundary-Value Problems*, Mathematics of Computation, Vol.31, 138, pp.:333–390.

[2] Briggs, William L., *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

[3] Gomes–Ruggiero, M.A., Kozakevich D. N. e Martínez, J. M. : *A Numerical Study on Large–Scale Nonlinear Solver*, Computers and Mathematics with Applications, Vol.32, nº 3, págs.: 1–13, 1996.

[4] Friedlander, A., Gomes–Ruggiero, M. A., Kozakevich, D.N, Martínez, J. M., Santos, S. A.: *Solving Nonlinear Systems of Equations by Means of Quasi–Newton Methods with a Nonmonotone Strategy* – Optimization Methods and Software, Vol.8, págs.:25-51, 1997.

[5] Hackbusch, W. and Trottenberg, U (editors), *Multigrid Methods*, Lecture Notes in Mathematics, 960, Springer, 1982.

[6] Knoll, D.A. and Rider, W. J., *A Multigrid Preconditioned Newton-Krylov Method*, SIAM J. Sci. Comput., vol.21, no.2, pp.:691–710.

[7] Wesseling, P., *An Introduction to Multigrid Methods*, John Wiley & Sons, 1991.

[8] Adams L., J.L. Nazareth, *Linear and Nonlinear Conjugate Gradient - Related Methods*, SIAM

[9] Shewchuk, J. R., *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, http://www-2.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.ps

[10] Saad Y., Schultz M., *GMRES a generalized minimal residual algorithm for solving nonsymmetric linear sistems*, SIAM J. Sci Statist. Comput., 7(1986), pp.:856–869.

## DETALHAMENTO DOS PROGRESSOS

Nesta etapa do projeto fizemos um estudo sobre as técnicas multigrid para resolução de sistemas lineares e implementamos o ciclo em V e o ciclo em V completo para resolvermos a equação diferencial parcial de segunda ordem advinda do problema de convecção - difusão descrita abaixo:

$$-\epsilon\left(u_{xx}+u_{yy}\right)+au_x+bu_y=f(x,y),\quad 0<x<1,\quad 0<y<1 \tag{4}$$

com $u(x,y)=0$ na fronteira do quadrado unitário.

Discretizamos o problema utilizando diferenças finitas e utilizamos o gmres para iterarmos no sistema. Encontramos dificuldades para ajustar os parâmetros do algoritmo, como o número de iterações em cada malha e o restart do gmres. No início da próxima etapa será realizada uma análise computacional mais detalhada sobre a escolha dos parâmetros para os ciclos Multigrid, uma vez que este é o ponto essencial para a eficiência das técnicas Multigrid. O algoritmo foi feito em Matlab versão 6.0.0.88 Release 12 e os arquivos podem ser obtidos no endereço `www.ime.unicamp.br/~ghaeser/fapesp/ciclos`.

## PLANO DE TRABALHO E CRONOGRAMA

fevereiro/2003 à agosto/2003: Técnica multigrid aplicadas a sistemas não lineares:

1. análise computacional da eficiência dos ciclos multigrid para sistemas lineares;
2. pesquisar alguns problemas clássicos de Valor de Contorno: Problema de Bratu e Convecção–Difusão, entender o significado físico das equações e propor saídas do programa computacional através de gráficos da superfície e de curvas de nível que possibilitem a interpretação da solução obtida;
3. estudo e aplicação das idéias Multigrid estudadas na primeira etapa para o caso de sistemas não lineares, usando o método de Newton como o método iterativo para o sistema não linear;
4. implementação computacional dos ciclos multigrid para sistemas não lineares;
5. análise dos resultados computacionais e adaptação de parâmetros do algoritmo;
6. elaboração do relatório final.

As etapas descritas serão elaboradas seguindo o seguinte cronograma:

| fev | mar | abr | maio | jun | jul | ago |
|---|---|---|---|---|---|---|
| 1 | 1 | | | | | |
| | | 2 | | | | |
| | | 3 | | | | |
| | | | 4 | 4 | 4 | |
| | | | | 5 | 5 | |
| | | | | | 6 | 6 |