# EFEITO DE ROTEIROS DO MÉTODO *MULTIGRID* SOBRE O TEMPO DE CPU PARA A EQUAÇÃO DE LAPLACE 2D

**Fabiane de Oliveira**
**Marcio Augusto Villela Pinto**
*faboliveira@uepg.br*
*marcio_villela@yahoo.com.br*
Universidade Estadual de Ponta Grossa (UEPG)
Departamento de Matemática e Estatística
Ponta Grossa, PR, Brasil

**Carlos Henrique Marchi**
*marchi@ufpr.br*
Universidade Federal do Paraná (UFPR)
Departamento de Engenharia Mecânica
Curitiba, PR, Brasil

***Resumo***. *O objetivo deste trabalho é minimizar o tempo de CPU necessário para resolver a equação de Laplace bidimensional (2D) com condições de contorno de Dirichlet. Utiliza-se: método de diferenças finitas com diferença central; malhas uniformes; e método multigrid geométrico com esquema de aproximação completa (FAS), ciclo V, prolongação por interpolação bilinear, e engrossamento padrão com razão dois. Estuda-se a influência de: diversos roteiros para o número de iterações internas ($\nu$) nos solvers; três solvers (Gauss-Seidel, MSI e ADI); dois tipos de esquema FAS; restrição por injeção, meia ponderação e ponderação completa; número de níveis de malha (L); e número de incógnitas (N) para malhas de 129x129 a 2049x2049 nós. Verificou-se que: (1) o menor tempo de CPU é obtido com o roteiro que usa um número fixo de iterações (três) no solver MSI, restrição por meia ponderação e FAS modificado; (2) L pode afetar significativamente o tempo de CPU, mas ao se usar o máximo L possível para uma dada malha tem-se em média um aumento de 2,5% no tempo de CPU em relação ao L ótimo, que é igual ao máximo L – 2; e (3) a ordem de p no algoritmo otimizado é de 1,08.*

***Palavras-chave:*** *algoritmos, otimização, diferenças finitas, iterações internas, solvers, CFD.*

## 1. INTRODUÇÃO

Para reduzir o erro de discretização em problemas de dinâmica dos fluidos computacional (CFD), são necessárias malhas muito refinadas que geram sistemas de equações muito grandes. A resolução destes sistemas através de métodos iterativos (*solvers*) básicos requer grande tempo de CPU. Isso ocorre porque no início do processo iterativo a taxa de convergência é grande, passando a decair sensivelmente à medida que o número de iterações aumenta.

Existem muitos trabalhos de pesquisa que visam aumentar a taxa de convergência dos métodos iterativos. Para isso, um método que pode ser usado é o *multigrid* (Briggs et al., 2000; Wesseling, 1992). Ele consiste no uso de malhas auxiliares mais grossas (com menor número de nós) do que a malha na qual se quer resolver o problema. São usados processos de restrição e prolongação para transferir informações entre as diversas malhas. Podem ser usados diversos *solvers* no processo de relaxação ou suavização. A seqüência com que as diversas malhas são visitadas denomina-se ciclo *multigrid* (ciclo V, W, F entre outros). Em cada tipo de ciclo pode-se partir da malha mais grossa, no esquema chamado de *full multigrid* (FMG), ou da malha mais fina, no esquema chamado padrão. Podem ser usados dois tipos de esquemas com o método *multigrid*: CS (*Correction Storage*) e FAS (*Full Aproximation Storage*). Eles são indicados respectivamente para problemas lineares e não-lineares. Finalmente, podem ser distinguidos os métodos *multigrid* geométrico e algébrico, respectivamente indicados para malhas estruturadas e não-estruturadas.

A eficiência do método *multigrid* não tem sido totalmente alcançada em aplicações práticas da engenharia na área de CFD. Com a crescente complexidade das aplicações, é crescente também a demanda por métodos mais eficientes e robustos. Espera-se que esses métodos tenham uma boa redução do tempo computacional, usem pouca memória e possam abordar não-linearidades e acoplamentos sem grandes prejuízos em seu desempenho.

Wesseling e Oosterlee (2001) fizeram uma revisão dos desenvolvimentos, na década de 90, do método *multigrid* geométrico em CFD, mostrando o estado-da-arte para escoamentos incompressíveis e compressíveis. Segundo eles o *multigrid* geométrico permanece um tópico ativo de pesquisa em CFD e é um dos mais significativos desenvolvimentos em análise numérica na segunda metade do século XX. Teoricamente, os algoritmos atuais usados com o *multigrid* geométrico ainda podem ser muito otimizados em relação ao tempo computacional necessário para resolver um problema de CFD; por exemplo, para resolver as equações de Navier-Stokes, o tempo de CPU pode ser reduzido ainda de 10 a 100 vezes (Brandt et al., 2002) do atual. A redução do tempo computacional para resolver um mesmo problema resulta na redução de custos de projetos. Um aumento na eficiência do método também permite, no mesmo tempo computacional, resolver um problema em uma malha mais refinada, isto é, com maior número de nós; isto significa obter uma solução numérica com menor erro de discretização (Roache, 1998), melhorando a qualidade e a confiabilidade dos projetos.

Segundo Trottenberg et al. (2001), experiências com o método *multigrid* mostram que seus parâmetros (número de malhas, o suavizador ou *solver*, o número de iterações no *solver*, ciclos, roteiros e os esquemas de restrição e interpolação) podem ter uma forte influência na eficiência do algoritmo. Não há regras gerais na escolha destes parâmetros, porém certos valores podem ser recomendados para determinadas situações. A taxa de convergência depende das escolhas feitas. Uma simples escolha do número de malhas a usar pode afetar significativamente o tempo computacional. Isso justifica a importância de estudar os diversos parâmetros do método *multigrid*. Na literatura encontram-se alguns trabalhos sobre a influência de parâmetros no método *multigrid*, por exemplo: Pinto et al. (2005) estudaram os parâmetros ótimos do *multigrid* para as equações 1D de difusão, advecção-difusão e Burgers; Rabi e De Lemos (2001) estudaram os parâmetros ótimos para um problema 2D de advecção-

difusão; e Santiago e Marchi (2007) fizeram uma análise dos parâmetros para o problema de Navier 2D que envolve duas equações.

O objetivo do presente trabalho é determinar qual é o roteiro, para o número de iterações do *solver*, dentre os mais usados na literatura, que resulta no menor tempo de CPU. Denomina-se roteiro às diferentes formas de percorrer os ciclos. A palavra roteiro foi usada como uma tradução para *schedule*. Tem-se o valor ótimo de um parâmetro quando a solução do problema é obtida no menor tempo de CPU para valores fixos dos demais parâmetros. No presente trabalho, usa-se tempo de CPU em vez de *work units* devido às razões apontadas por Larsson et al. (2005) e Trottenberg et al. (2001). Os roteiros considerados para o número de iterações internas ($\nu$) são: totalmente constante, dinâmico, constante na restrição e na prolongação, Hortmann e suas variações, e dente-de-serra e suas variações. Estes roteiros estão descritos na seção 3. É analisado também o efeito de alguns *solvers* (MSI, Gauss-Seidel e ADI) e tipos de restrição (injeção, meia ponderação e ponderação completa). Não se encontra na literatura um estudo tão amplo e detalhado sobre o número de iterações internas e roteiros quanto o apresentado no presente trabalho. Chisholm (1997) realizou um estudo sobre alguns parâmetros do *multigrid* para um *solver* implícito aproximadamente fatorado para a equação de Navier-Stokes (*approximately-factored implicit Navier-Stokes solver*) entre eles o número de iterações internas para os ciclos V, W e dente-de-serra.

O presente trabalho envolve um problema bidimensional linear de condução de calor, governado pela equação de Laplace com condições de contorno de Dirichlet. O método numérico utilizado é o de diferenças finitas (Golub e Ortega, 1992; Tannehill et al., 1997) com aproximação CDS (diferença centrada) e malhas uniformes.

Este artigo está organizado da seguinte forma: na seção 2, são apresentados os modelos matemático e numérico, incluindo detalhes do método *multigrid*. Na seção 3 são apresentados os roteiros para a variação das iterações internas. Na seção 4 são apresentados os resultados. E na seção 5 é apresentada a conclusão do trabalho.

## 2. MODELOS MATEMÁTICO E NUMÉRICO

### 2.1 Equação de Laplace 2D e sua discretização

O problema linear de condução de calor bidimensional numa placa plana, em regime permanente, com propriedades constantes e condições de contorno de Dirichlet é modelada matematicamente pela equação de Laplace.: (Incropera e DeWitt, 1998)

$$\begin{cases} \dfrac{\partial^2 T}{\partial x^2} + \dfrac{\partial^2 T}{\partial y^2} = 0, \qquad 0 < x, y < 1, \\ T(x,1) = sen(\pi x),\ T(x,0) = T(0,y) = T(1,y) = 0, \end{cases} \tag{1}$$

onde $x$ e $y$ são as direções coordenadas; e $T$ representa a temperatura. A solução analítica deste problema é dada por

$$T(x,y) = sen(\pi x)\frac{senh(\pi y)}{senh(\pi)}. \tag{2}$$

A discretização do domínio é feita com malhas uniformes, com um número de incógnitas (ou número de pontos) dado por $N = N_x\, N_y$, onde $N_x$ e $N_y$ são os números de pontos nas

direções coordenadas *x* e *y* respectivamente, incluindo os contornos. Para cada um dos $(N_x - 2)$ por $(N_y - 2)$ pontos interiores da malha, a Eq. (1) é discretizada com o método das diferenças finitas e diferença central (CDS) (Burden e Faires, 2003; Tannehill et al., 1997), resultando em

$$\frac{\left(T_{i-1,j} - 2T_{i,j} + T_{i+1,j}\right)}{h^2} + \frac{\left(T_{i,j-1} - 2T_{i,j} + T_{i,j+1}\right)}{h^2} = 0, \tag{3}$$

onde $h$ é a distância entre dois nós consecutivos da malha.

A Eq. (3) pode ser representada por um sistema de equações algébricas do tipo

$$AT = b, \tag{4}$$

onde a matriz dos coeficientes *A* é pentadiagonal *N* por *N*, simétrica e definida positiva (Briggs et al., 2000).

### 2.2 Método *multigrid* e detalhes numéricos

O sistema de equações algébricas representado pela Eq. (4) é resolvido com o método *multigrid* geométrico, conforme descrito por Wesseling (1992), usando-se dois esquemas: o FAS padrão e uma modificação dele dada por Yan e Thiele (1998). Pinto e Marchi (2006) verificaram que, para a equação de Laplace bidimensional, o esquema FAS é mais rápido que o esquema CS. O tipo de ciclo utilizado é o V com *multigrid* padrão. O ciclo V foi escolhido porque o W é cerca de 50% mais caro (Hirsch, 1988). O processo de restrição é feito por injeção e posteriormente comparado com meia ponderação e ponderação completa. A prolongação é feita através de interpolação bilinear (Briggs et al., 2000; Trottenberg et al., 2001). A razão de engrossamento de malhas (*r*) usada é a padrão cujo valor é dois.

Os suavizadores ou *solvers* considerados serão comparados são: o método de Gauss-Seidel (GS) descrito em Burden e Faires (2003); o *Modified Strongly Implicit Method* (MSI) (Schneider e Zedan, 1981); e o *Alternated Direction Implicit* (ADI) (Ferziger e Peric, 1999; Tannehill et al., 1997). No presente trabalho, o número de iterações internas feitas em cada malha é representado por $\nu$. Na nomenclatura de Trottenberg et al. (2001), para o ciclo V, tem-se $V(\nu_1, \nu_2)$ sendo $\nu_1$ o número de pré-suavizações no *solver* no processo de restrição, e $\nu_2$ o número de pós-suavizações no *solver* no processo de prolongação. Quando $\nu_1 = \nu_2$, representa-se simplesmente por $\nu$.

O número de vezes que o ciclo V é repetido é denominado de iterações externas. O critério de convergência usado para interromper as iterações externas é baseado na média da norma $l_1$ do erro de iteração, definida por

$$\bar{l}_1[E_n(\phi)]_k = \frac{\sum_{p=1}^{N} \left|(\phi_{k\to\infty} - \phi_k)_i\right|}{N}, \tag{5}$$

onde $\phi_{k\to\infty}$ é a solução exata do sistema de equações, que é obtido ao se iterar até ser atingido o erro de arredondamento de máquina; $\phi_k$ é a solução na iteração *k*; $N$ é o número total de nós na malha; $E_n$ é o erro na iteração *k*; $\bar{l}_1[E_n(\phi)]_k$ denota a média da norma do erro na

iteração *k*; e *p* denota cada nó da malha. Este critério já foi utilizado por Santiago e Marchi (2007). O processo iterativo é interrompido quando $\bar{l}_1[E_n(\phi)]_k$ é menor ou igual à tolerância (*Tol*) de $10^{-10}$. Utilizou-se o valor nulo como estimativa inicial da solução de cada problema.

Em todas as simulações, considerou-se um número de níveis de malha *L* tal que $1 \le l \le L \le L_{max}$, onde *l* é o número do nível de uma malha em particular; e $L_{max}$ representa o número máximo possível de malhas que se pode usar para uma dada malha mais fina, com a malha mais grossa tendo apenas um nó interno. Por exemplo, se *N* = 513 x 513 nós com $r = 2$, as malhas são de 513x513, 257x257, 129x129, 65x65, 33x33, 17x17, 9x9, 5x5 e 3x3 nós; neste exemplo, portanto, $L_{max} = 9$.

O foco deste trabalho é a minimização do tempo de CPU. Entende-se por tempo de CPU o tempo gasto para realizar a geração de malhas, atribuir a estimativa inicial, calcular os coeficientes, e resolver o sistema linear representado pela Eq. (4) até atingir a tolerância estabelecida com base no critério de convergência. Este tempo é medido em segundos (s) usando-se a sub-rotina CPU_TIME do Fortran 95.

Tem-se o valor ótimo de um parâmetro quando a solução do problema é obtida no menor tempo de CPU para valores fixos dos demais parâmetros. Assim, denota-se por $\nu_{ótimo}$ o número ótimo de iterações internas no *solver*, e por $L_{ótimo}$ o número ótimo de níveis de malhas.

Os algoritmos foram implementados na linguagem Fortran 95, versão 9.1 da Intel com precisão dupla. As simulações foram realizadas num microcomputador com processador Intel Core 2 Duo de 2,66 GHz, 8 GB de RAM e sistema operacional Windows xp 64 bits.

## 3. ROTEIROS

Briggs et al. (2000), Hortmann et al. (1990), Mesquita e De- Lemos (2004), Trottenberg et al. (2001), Yan et al. (2007) e Wesseling (1992) apresentam diversas estratégias para determinar o momento de se mudar de malha no método *multigrid*, usando critérios do tipo dinâmico ou de ciclo. No presente trabalho, estas estratégias estão incluídas dentro dos roteiros. O critério dinâmico consiste em monitorar a taxa de convergência da solução numérica, a qual pode, por exemplo, ser determinada pela razão entre as normas dos resíduos de duas iterações sucessivas. Algumas propostas do critério dinâmico podem ser encontradas em Brandt (1977) e Donohue et al. (1998) e uma aplicação em Ghia et al., (1982).

O critério de ciclo consiste em especificar o número de iterações internas em cada nível de malha. A literatura, em geral, utiliza o critério de ciclo. Algumas aplicações do critério de ciclo podem ser encontradas em: Hortmann et al. (1990), onde se define o roteiro de Hortmann; Chisholm (1997), Gerolymos e Vallet (2005) e Wesseling (1992) onde é utilizado o roteiro dente-de-serra; Pinto et al. (2005) utilizaram um roteiro que considera o número de iterações internas totalmente constante. Alguns autores referem-se ao critério dinâmico como "*adaptative schedule*". A seguir tem-se a descrição dos roteiros utilizados no presente trabalho.

### 3.1 Número de iterações internas totalmente constante ($\nu = \nu_1 = \nu_2$)

O roteiro com $\nu$ totalmente constante é o mais utilizado na literatura (Briggs et al., 2000; Mesquita e De-Lemos, 2004; Wesseling e Oosterlee, 2001). Ele consiste em utilizar o mesmo número de iterações internas em todos os níveis de malha, tanto na restrição ($\nu_1$) como na prolongação ($\nu_2$), isto é, $\nu_1 = \nu_2 = \nu$. A Fig. 1 exemplifica dois ciclos V para $\nu$ = 4 e *L* = 6,

onde $L$ representa o número de níveis de malha usado. O símbolo • representa as suavizações realizadas em cada malha; e os traços que os unem, os operadores de transferência entre malhas (operadores de restrição e de prolongação).

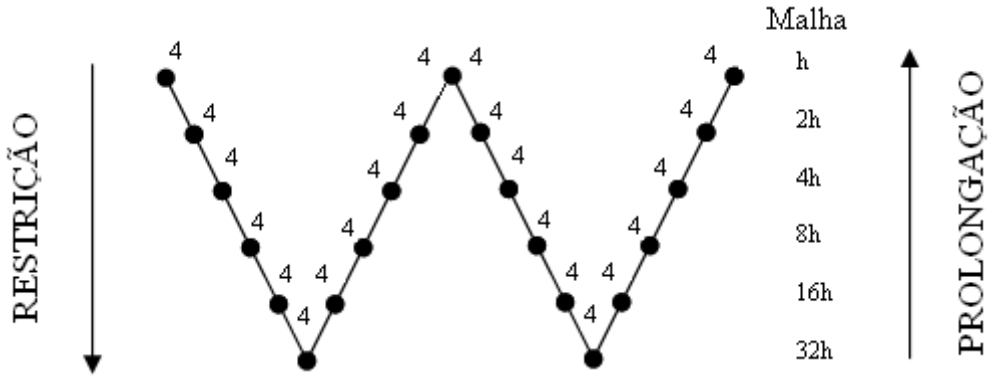


Figura 1 – Roteiro $\nu$ totalmente constante para $\nu = 4$ e $L = 6$.

### 3.2 Número de iterações internas dinâmico

O roteiro com $\nu$ dinâmico segue o critério de Brandt (1977). Este roteiro consiste em realizar um número de iterações internas em cada nível de malha até atingir uma tolerância interna estabelecida, representada por *Tol_d.* A Fig. 2 representa dois ciclos V do roteiro para $\nu$ dinâmico.

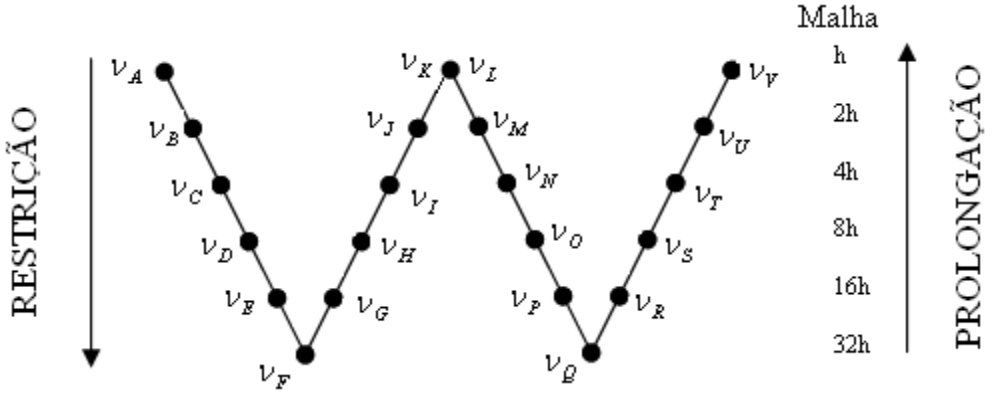


Figura 2 – Roteiro $\nu$ dinâmico para $L = 6$.

Na Fig. 2 a variável $\nu$ com $\nu = \nu_A,...,\nu_V$ representa o número de iterações internas necessárias em cada nível de malha, e em cada ciclo V, para se atingir a tolerância estabelecida, ou seja, *Tol_d.* Observa-se que não existe um $\nu$ fixo, mas sim um $\nu$ determinado através do critério citado acima. Desta forma, pretende-se verificar o comportamento de $\nu$ nos diferentes níveis de malha. Uma aplicação deste roteiro pode ser encontrada em Ghia et al. (1982).

### 3.3 Número de iterações internas constante na restrição e na prolongação

Através das simulações realizadas com o roteiro $\nu$ dinâmico, verificou-se que o $\nu$ é maior na restrição e que na prolongação este valor varia entre 1 e 3, como poderá ser visto na seção 4.2. O roteiro que diferencia o número de iterações internas para a restrição ($\nu_1$) do

número de iterações internas para a prolongação ($\nu_2$) já foi utilizado por Trottenberg et al., (2001). Considera-se $\nu_1$ e $\nu_2$ sempre constantes em cada um destes dois processos, sendo em geral diferentes entre si. A Fig. 3 representa dois ciclos V deste roteiro.

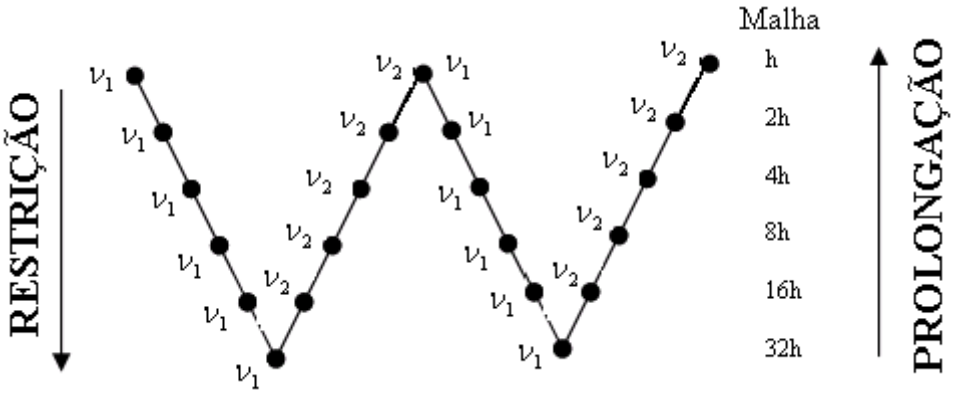


Figura 3 – Roteiro $\nu$ constante na restrição e na prolongação para $L$ = 6.

### 3.4 Roteiros tipo Hortmann

***Hortmann tipo I***. Este roteiro foi proposto por Hortmann et al. (1990) para o *full multigrid*, visando resolver um problema de escoamento laminar, sem o objetivo de otimizar o roteiro. No presente trabalho, o roteiro de Hortmann foi adaptado para o ciclo V. Ele consiste em especificar o número de iterações internas para a restrição e prolongação que varia com o nível de cada malha ($l$). Inicia-se com $\nu$ = 2. Nas malhas subseqüentes $\nu$ tem um acréscimo de uma unidade a cada nível de malha. Na malha mais grossa $\nu$ é igual a $L$+1. Dois ciclos deste roteiro para um problema com $L$ = 6 estão representados na Fig. 4.

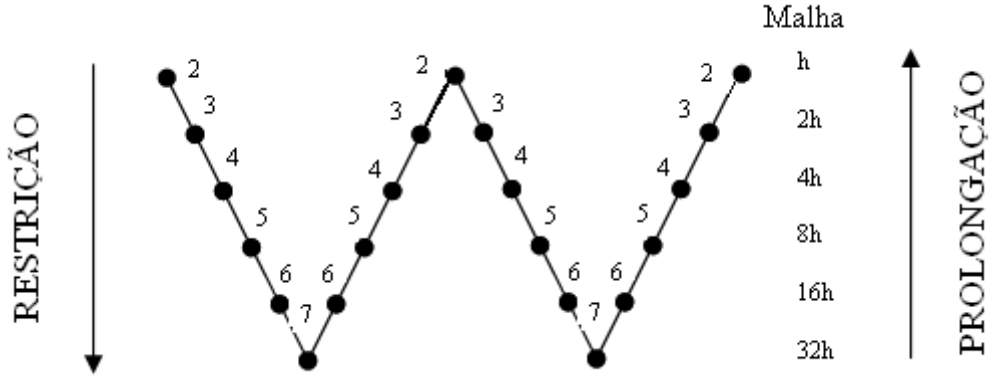


Figura 4 – Roteiro Hortmann tipo I para $L$ = 6.

***Hortmann tipo II***. É uma modificação no roteiro de Hortmann tipo I. Observa-se na Fig. 4 que ao término do primeiro ciclo V as suavizações são realizadas somente na prolongação e o ciclo seguinte não suaviza na restrição. No roteiro de Hortmann tipo II, suaviza-se tanto na restrição como na prolongação, ou seja, o processo completo de um ciclo V é repetido.

***Hortmann tipo II inverso***. Nos testes realizados com $\nu$ dinâmico, observou-se que para ser atingido *Tol_d* é necessário um número maior de iterações internas nas malhas mais finas. No roteiro de Hortmann tipo I, apresentado na Fig. 4, o número de iterações internas é maior nas malhas mais grossas. O roteiro de Hortmann tipo II inverso, proposto no presente

trabalho, utiliza um $\nu$ maior nas malhas mais finas, iniciando com $\nu = L+1$. Nas malhas subseqüentes este $\nu$ tem um decréscimo de uma unidade. Na malha mais grossa o $\nu$ é igual a 2. Dois ciclos V deste roteiro, para um problema com $L = 6$, estão representados na Fig. 5.

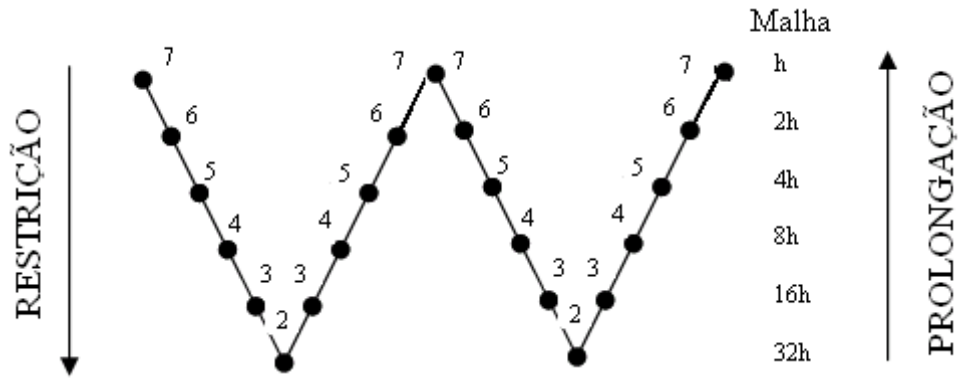


Figura 5 – Roteiro Hortmann tipo II inverso para $L = 6$.

***Hortmann tipo II com $\nu_2$ constante***. Neste roteiro, proposto no presente trabalho, o roteiro de Hortmann tipo II é aplicado somente na restrição. Na prolongação, o $\nu_2$ é fixo. A variável $\nu_2$ foi baseada em dados obtidos com o $\nu$ dinâmico, onde se observa que o número de iterações internas na prolongação varia entre 1 e 3.

### 3.5 Roteiros tipo dente-de-serra

***Dente-de-serra tipo I***. Conforme Wesseling (1992), o dente-de-serra (tipo I) é um caso especial do ciclo V, no qual a suavização é feita somente na prolongação. Este roteiro está representado na Fig. 6. Mais detalhes sobre o dente-de-serra tipo I podem ser encontrados em Wesseling (1992). Aplicações do dente-de-serra tipo I podem ser encontradas em Zeeuw (1996).

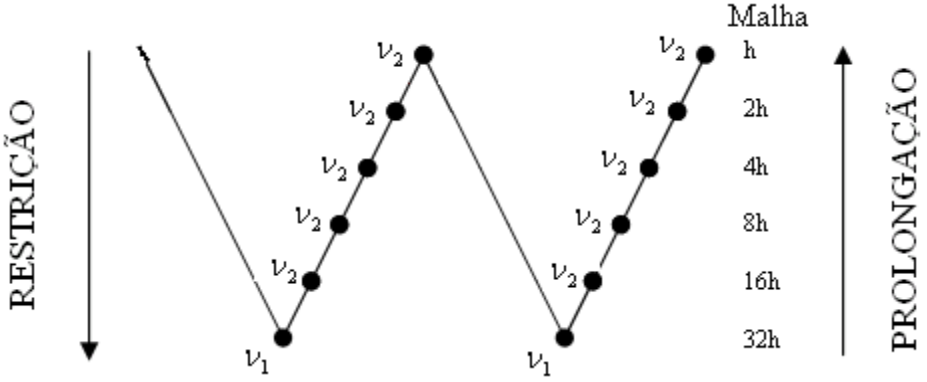


Figura 6 – Roteiro dente-de-serra tipo I para $L = 6$.

***Dente-de-serra tipo II***. O roteiro dente-de-serra tipo II suaviza somente na restrição. Na prolongação é feita apenas a transferência de informações. A suavização é realizada somente na malha mais fina. Pode-se encontrar este roteiro nos trabalhos de Manzano (1999) e Gerolymos e Vallet (2005).

***Dente-de-serra tipo II modificado***. O roteiro dente-de-serra tipo II modificado, proposto no presente trabalho, é uma variação do dente-de-serra tipo II. A diferença entre eles se encontra nos três últimos níveis das malhas mais grossas onde $\nu_1 = 1$. Este fato foi observado em testes realizados com o roteiro $\nu$ dinâmico, onde nos três últimos níveis das malhas mais grossas *Tol_d* era atingida com uma única suavização.

## 4. RESULTADOS E DISCUSSÃO

Foram realizadas aproximadamente 1000 simulações com os seguintes intervalos para cada parâmetro: malhas = 129x129 a 2049x2049 que correspondem ao número de incógnitas $N$ = 16.641 a 4.198.401; número de malhas $L$ entre 1 e 11; e número de iterações internas $\nu$ = 1 a 20.

Nas seções 4.1 a 4.5 utilizou-se o método *multigrid* com esquema FAS, *solver* MSI e restrição por injeção. Nas seções 4.1 e 4.10 utilizou-se $L = L_{max}$ a $L_{max}$-5, e nas demais seções $L = L_{max}$.

Para todas as malhas cujo tempo de CPU foi inferior a 100 segundos, acrescentou-se ao programa principal um ciclo mais externo, fazendo com que o número de simulações se repetisse até ser obtido tempo de CPU igual ou superior a 100 segundos. Nestes casos, o tempo de CPU de uma simulação é uma média do tempo obtido em todas as repetições. Esse procedimento foi adotado para reduzir a incerteza da sub-rotina CPU_TIME na medida do tempo de CPU. São apresentados a seguir somente os resultados mais representativos do estudo realizado.

### 4.1 $\nu$ totalmente constante

A Fig. 7 mostra a influência do número de iterações internas $(\nu)$ sobre o tempo de CPU, para as malhas 129x129 a 2049x2049, com $L = L_{max}$. Verifica-se que o número ótimo de iterações internas é igual a 3 para todas estas malhas, isto é, $\nu_{ótimo} = 3$. Nesta figura e nas seguintes, cada ponto nos gráficos representa uma simulação totalmente independente das demais.

A Fig. 8 mostra a influência do número de níveis de malha ($L$) sobre o tempo de CPU. Para cada curva da Fig. 8, parte-se do valor máximo que $L$ pode assumir para dado $N$, denotado por $L_{max}$. Por exemplo, para a malha 513x513, $L_{max} = 9$. Este valor de $L$ é diminuído até $L_{max}$-5. Para cada curva da Fig. 8 tem-se o valor de $L_{ótimo}$, que é o $L$ que resulta no menor tempo de CPU para se resolver o problema. Verificou-se que para $\nu = 3$, o número ótimo de malhas é $L_{ótimo} = L_{max}$ - (0 a 2). Este resultado concorda com os obtidos por Pinto e Marchi (2006). Nas seções 4.2 a 4.9 utiliza-se $L_{max}$ para cada $N$.

Nota-se nas Figs. 7 e 8 que o número de iterações internas $(\nu)$ e o número de níveis de malha ($L$) podem influenciar significativamente o tempo de CPU, que é mais sensível ao valor de $L$ do que de $\nu$.

### 4.2 $\nu$ dinâmico

Para o roteiro com $\nu$ dinâmico, descrito na seção 3.2, foram realizadas simulações para *Tol_d* = 0,01; 0,1; e 0,2. Para todos os valores de *Tol_d* analisou-se a variação de $\nu$ em cada

nível de malha, tanto na restrição como na prolongação. Com base nos resultados obtidos e para todos os valores de *Tol_d*, verificou-se que:

- O $\nu$ é maior na restrição do que na prolongação. Para a restrição, $\nu$ varia entre 1 e o número máximo de iterações internas permitido em cada nível.
- O $\nu$ na prolongação varia entre 1, 2 e 3.
- $\nu$ = 1 nas três malhas mais grossas, tanto na restrição como na prolongação.
- O cálculo do resíduo demanda muito tempo de CPU.

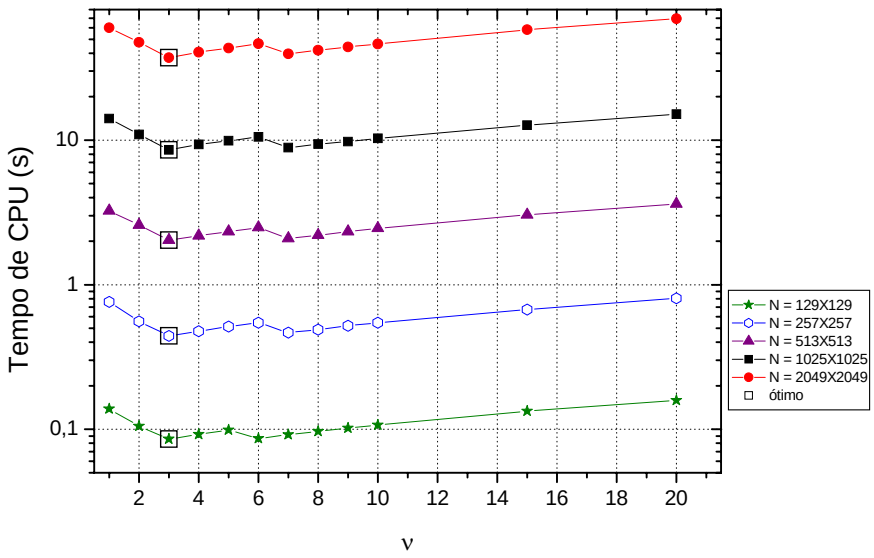


Figura 7 – Tempo de CPU versus número de iterações internas $(\nu)$ para o roteiro com $\nu$ totalmente constante e $L = L_{max}$.

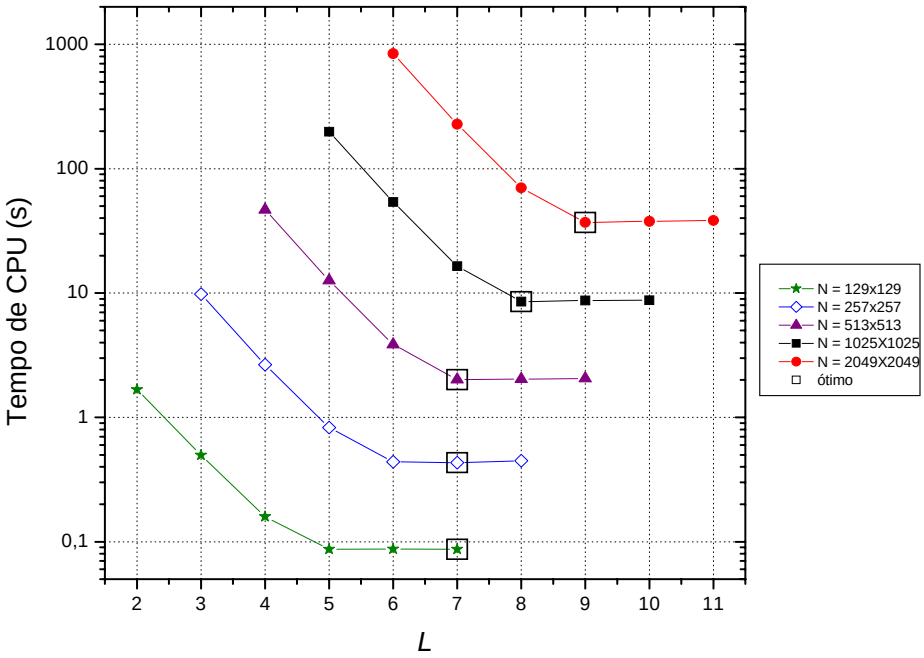


Figura 8 – Tempo de CPU versus número de níveis de malha ($L$) para o roteiro com $\nu = 3$ totalmente constante.

A Fig. 9 compara o tempo de CPU obtido entre os roteiros de $\nu$ dinâmico, com três valores de tolerância, e o $\nu$ totalmente constante. Observa-se que o roteiro com $\nu$ dinâmico para *Tol_d* = 0,01 obteve o menor tempo de CPU para problemas envolvendo malhas mais finas (1025x1025 e 2049x2049). Para *N* = 2049x2049 e *Tol_d* = 0,01, o tempo de CPU é cerca de 15% menor em relação a *Tol_d* = 0,2, e 22% menor em relação a *Tol_d* = 0,1. Para *N* menor (129x129, 257x257 e 513x513) e *Tol_d* = 0,1, obteve-se menor tempo de CPU.

Para a malha *N* = 2049x2049, o roteiro com $\nu$ totalmente constante é 771% mais rápido que $\nu$ dinâmico com *Tol_d* = 0,01. Esta diferença no tempo de CPU é devido ao fato de que o $\nu$ dinâmico necessita de mais iterações internas e externas, e também do cálculo da norma a cada suavização realizada.

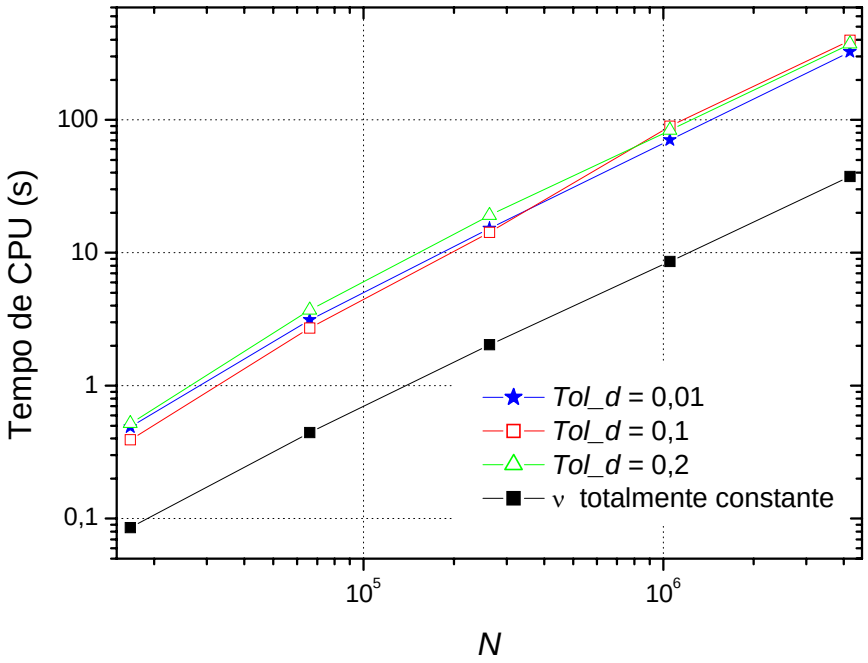


Figura 9 – Comparação entre $\nu$ totalmente constante e $\nu$ dinâmico (*Tol_d* = 0,01; 0,1; e 0,2).

### 4.3 $\nu$ constante na restrição e na prolongação

Nestes testes fixou-se um valor para $\nu_2$ (variando entre 1 e 4). Para cada valor de $\nu_2$, variou-se $\nu_1$ (1 a 20) com o objetivo de determinar os valores ótimos de $\nu_1$ e $\nu_2$. A Tabela 1 a seguir apresenta o número ótimo de iterações internas para a restrição ($\nu_1$) dado um $\nu_2$ fixo na prolongação.

Tabela 1. Número ótimo de $\nu_1$ para $\nu_2$ fixo

| $L = L_{max}$ | $N$ | $\nu_2 = 1$ | $\nu_2 = 2$ | $\nu_2 = 3$ | $\nu_2 = 4$ |
|---|---|---|---|---|---|
| 7 | 129 x 129 | 5 | 4 | 3 | 2 |
| 8 | 257 x 257 | 5 | 4 | 3 | 2 |
| 9 | 513 x 513 | 5 | 4 | 3 | 2 |
| 10 | 1025 x 1025 | 5 | 4 | 3 | 2 |
| 11 | 2049 x 2049 | 5 | 4 | 3 | 2 |

Verificou-se que, para um dado $\nu_2$ fixo, a soma do número de iterações internas ótimo para a restrição ($\nu_1$) com o valor de $\nu_2$ é igual a 6. Além disso, a combinação $\nu_1 = \nu_2 = 3$ resultou no menor tempo de CPU entre os roteiros de $\nu_1$ e $\nu_2$ fixos. Vale lembrar que quando $\nu_1 = \nu_2$ obtém-se o roteiro com $\nu$ totalmente constante. Portanto, o ponto ótimo do roteiro $\nu$ constante na restrição e na prolongação é o roteiro com $\nu$ totalmente constante.

### 4.4 Roteiros tipo Hortmann

Para o roteiro de Hortmann tipo II com $\nu_2$ constante foram realizados testes variando-se $\nu_2$ (1, 2 e 3) e aplicando Hortmann tipo II apenas na restrição. O melhor resultado obtido foi para $\nu_2 = 3$. Realizaram-se testes também para os demais roteiros tipo Hortmann descritos na seção 3, variando-se o tamanho da malha.

A Fig. 10 faz uma comparação entre o tempo de CPU obtido com estes tipos de roteiros. Verificou-se que o melhor roteiro Hortmann é o tipo II, seguido do tipo II com $\nu_2 = 3$, tipo II inverso e por último o tipo I. Em termos de erros percentuais o roteiro de Hortmann tipo II é em média 61% mais rápido que o roteiro Hortmann tipo I.

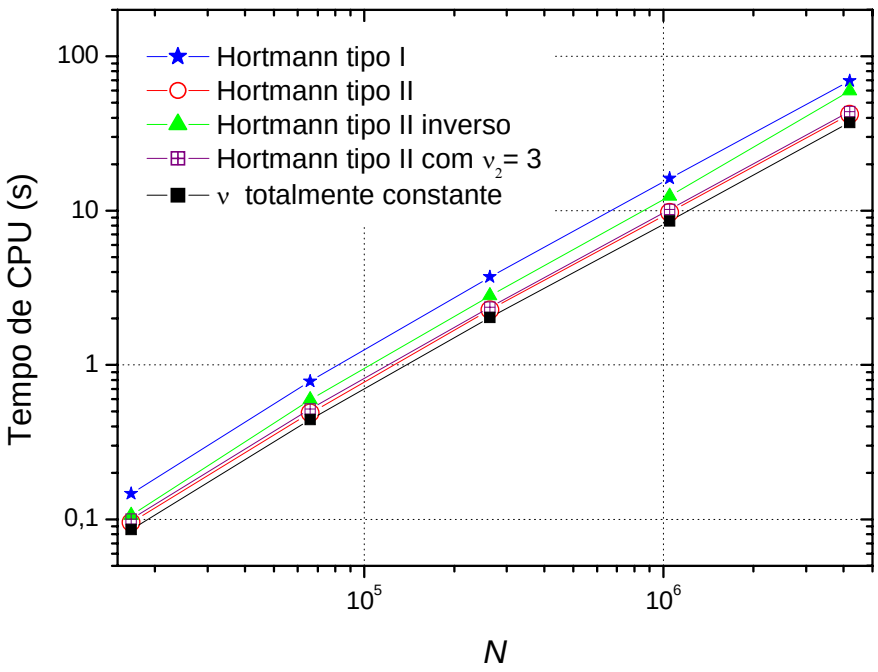


Figura 10 – Comparações entre os roteiros tipo Hortmann.

### 4.5 Roteiros tipo dente-de-serra

Para os roteiros tipo dente-de-serra, descritos na seção 3.5, foram realizadas as seguintes simulações: para o dente-de-serra tipo I foi feita uma análise de *N*, e através da variação de $\nu_2$, determinou-se $\nu_2$ ótimo para cada malha; no dente-de-serra tipo II foi realizada a análise de *N* e $\nu_1$; para o dente-de-serra tipo II modificado fixou-se nos três últimos níveis $\nu_1 = 1$ e determinou-se o $\nu_1$ ótimo para os demais níveis.

A Fig. 11 apresenta os resultados das simulações realizadas com os roteiros tipo dente-de-serra. Observa-se nela que o tempo de CPU obtido para os roteiros tipo dente-de-serra são muito próximos. O melhor roteiro dente-de-serra é o tipo II modificado com $\nu_1 = 4$, perdendo

somente no problema 2049x2049 para o tipo I. Para a malha 1025x1025, o dente-de-serra tipo II modificado é cerca de 1,8% mais rápido que o dente-de-serra tipo I.

O roteiro dente-de-serra tipo II modificado apresenta um menor tempo de CPU em relação aos roteiros dente-de-serra tipo I descrito em Wesseling (1992) e dente-de-serra tipo II aplicado por Gerolymos e Vallet (2005). Para a malha 1025x1025, o dente-de-serra tipo II modificado é 1,8% e 1,5% mais rápido que o dente-de-serra tipo I e tipo II, respectivamente.

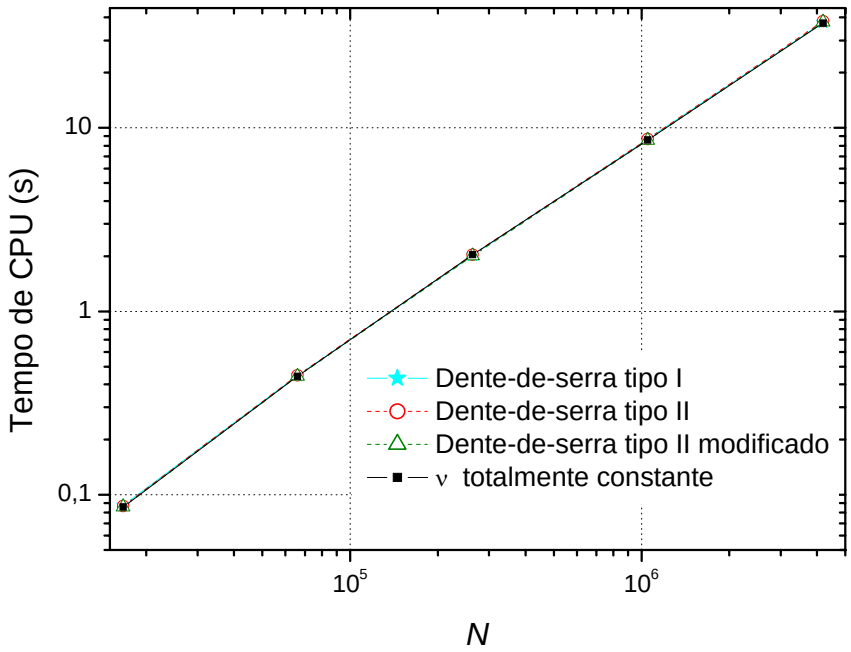


Figura 11 - Tempo de CPU para os roteiros tipo dente-de-serra.

### 4.6 Comparação entre os melhores roteiros de cada classe

A Fig. 12 faz uma comparação entre os melhores roteiros de cada classe utilizados neste trabalho. Não é mostrado o resultado do roteiro com $\nu$ constante na restrição e na prolongação porque o seu menor tempo de CPU é obtido para $\nu_1 = \nu_2$, que coincide com o roteiro $\nu$ totalmente constante.

Observa-se na Fig. 12 que o roteiro com $\nu$ totalmente constante apresenta o menor tempo computacional entre todos os tipos de roteiro estudados. Em termos percentuais o algoritmo com $\nu$ totalmente constante para a malha 2049x2049 é 0,11% mais rápido que o dente-de-serra tipo II modificado, 13% mais rápido que Hortmann tipo II e 771% em relação ao $\nu$ dinâmico com *Tol_d* = 0,01. Portanto, recomenda-se o uso do $\nu$ totalmente constante, pois este roteiro apresenta o menor tempo de CPU e é de mais fácil programação.

### 4.7 Análise de *solvers*

A Fig. 13 apresenta o tempo de CPU para resolver o problema em diversas malhas com os *solvers* Gauss-Seidel, MSI e ADI. Estes resultados foram obtidos com o melhor roteiro descrito na seção 4.6 ($\nu$ totalmente constante) e restrição por injeção.

A Tabela 2 apresenta o número ótimo de iterações internas para os *solvers*. Verifica-se nela que:

- O $\nu_{ótimo}$ para o MSI é igual a 3, em todos os tamanhos de malhas.

- O $\nu_{ótimo}$ para o Gauss-Seidel varia entre 3 e 5. Ao se usar $\nu = 4$, o tempo de CPU aumenta 8,8% e 7,0% respectivamente para $N$ = 513x513 e 1025x1025, em relação ao $\nu_{ótimo}$.
- O $\nu_{ótimo}$ para o ADI varia entre 3 e 5. Ao se usar $\nu = 5$, o tempo de CPU aumenta 0,8% e 0,1% respectivamente para $N$ = 513x513 e 1025x1025, em relação ao $\nu_{ótimo}$.

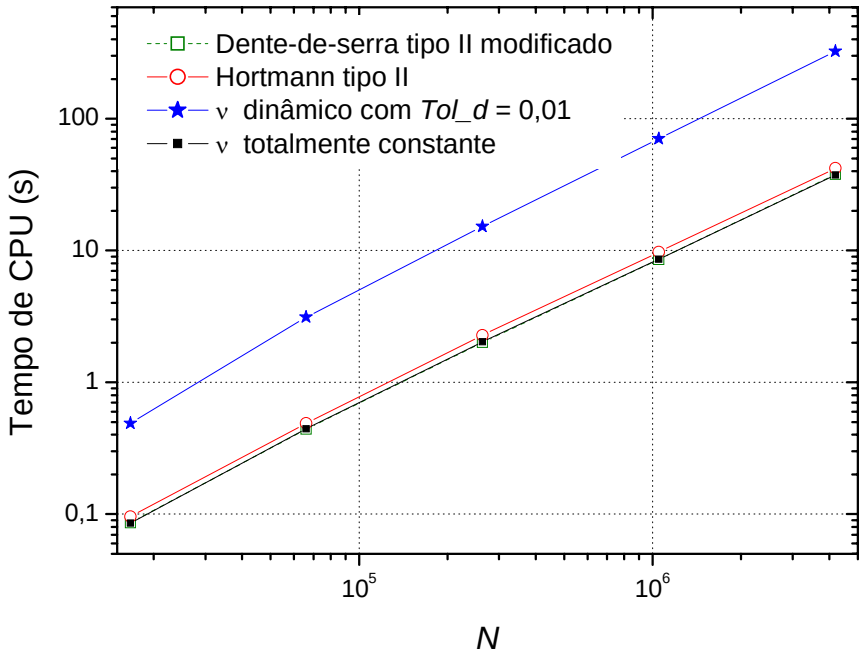


Figura 12 – Comparação entre os melhores roteiros de cada classe.

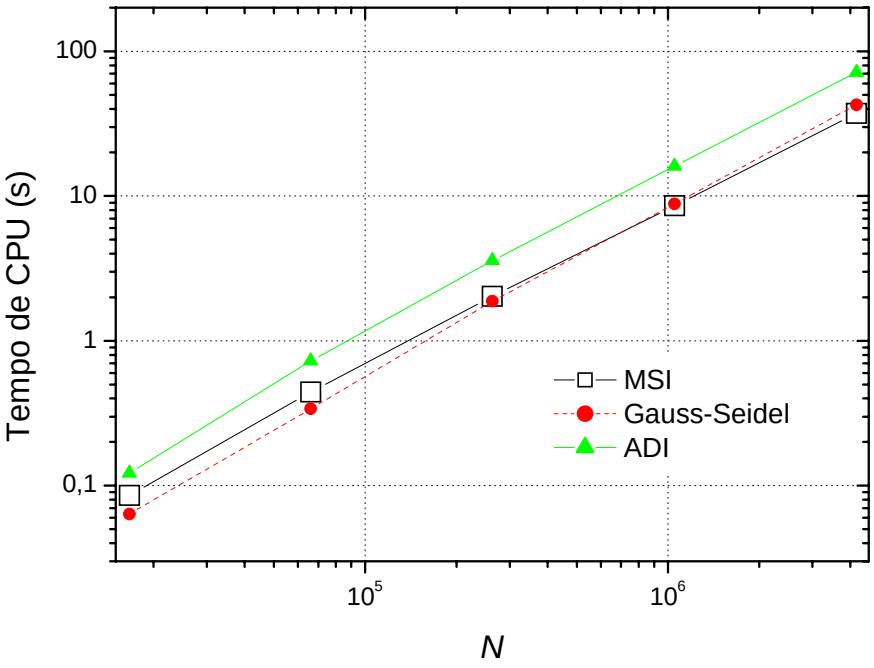


Figura 13 – Comparação entre os *solvers* para o roteiro $\nu$ totalmente constante.

Com base nos resultados obtidos, verificou-se que:

- O ADI apresenta o maior tempo de CPU em comparação com Gauss-Seidel para todos os tamanhos de malha. Por exemplo, para a malha 2049x2049, o ADI é 67% mais lento que o Gauss-Seidel e 92% que o MSI.
- Para problemas menores (129x129 a 513x513), o Gauss-Seidel apresenta o menor tempo de CPU. E para problemas de tamanho 1025x1025 e 2049x2049, o MSI apresenta o menor tempo de CPU.
- Utilizar o MSI significa uma economia no tempo de CPU com um pequeno acréscimo de memória RAM.
- Para a malha 2049x2049, o MSI é 13% mais rápido que o Gauss-Seidel.

Portanto, entre os *solver*s estudados, recomenda-se utilizar o MSI, visto que é o *solver* mais rápido para problemas maiores e mais comportado no que diz respeito ao número de iterações internas.

Foram realizadas análises para verificar a quantidade de memória RAM necessária para os *solvers* MSI e Gauss-Seidel. Através do gerenciador de tarefas do Windows, verificou-se qual a memória máxima utilizada para cada *solver* em cada uma das malhas. O *solver* MSI usa de 37% a 79% a mais de memória em relação ao Gauss-Seidel respectivamente para as malhas de 129x129 a 2049x2049 nós. Portanto, em computadores com limitação de memória, talvez seja mais indicado usar o *solver* Gauss-Seidel. Se compararmos a memória utilizada pelo método *multigrid* com o método de uma malha (*singlegrid*), verifica-se que o acréscimo na memória é de 13% a 18% respectivamente para as malhas de 129x129 a 2049x2049 nós.

Tabela 2. Número ótimo de iterações internas para os *solvers*

| $L = L_{max}$ | $N$ | MSI | Gauss-Seidel | ADI |
|---|---|---|---|---|
| 7 | 129x129 | 3 | 4 | 5 |
| 8 | 257x257 | 3 | 4 | 5 |
| 9 | 513x513 | 3 | 3 | 3 |
| 10 | 1025x1025 | 3 | 5 | 5 |
| 11 | 2049x2049 | 3 | 4 | 3 |

### 4.8 Análise dos tipos de restrição

Utilizando-se o melhor roteiro ($\nu$ totalmente constante, seção 4.6) e o melhor *solver* (MSI, seção 4.7), realizou-se uma análise de tipos de restrição: injeção, meia ponderação e ponderação completa (Trottenberg et al., 2001; e Hirsch, 1988). Os resultados são mostrados na Fig. 14.

Verificou que:

- A restrição por meia ponderação obteve o menor tempo de CPU, seguida da restrição por injeção e da ponderação completa.
- A restrição por meia ponderação é mais rápida em média 0,7% em relação à restrição por injeção.
- Quanto maior o número de níveis de malha, maior é a diferença entre o tempo de CPU obtido por ponderação completa em relação à injeção. Para problemas envolvendo malhas de 2049x2049, a restrição por ponderação completa é 20% mais lenta que a restrição por injeção.
- O tipo de restrição não afeta o número ótimo de iterações internas, ou seja, $\nu_{ótimo} = 3$, para os três tipos de restrição estudados.

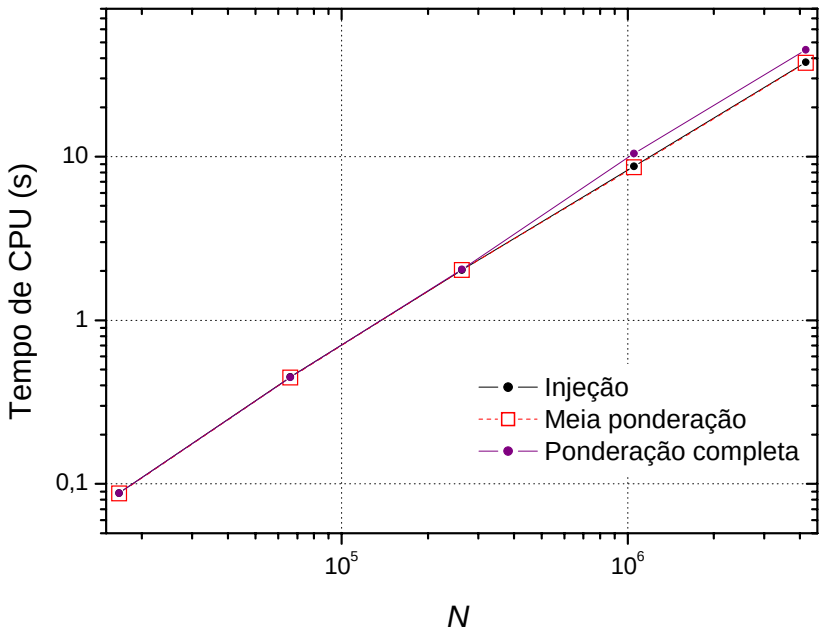


Figura 14 – Comparação entre os tipos de restrição para o roteiro $\nu$ totalmente constante.

### 4.9 Algoritmo de Yan e Thiele

Yan e Thiele (1998) propuseram um algoritmo FAS modificado, em que somente o resíduo é restrito entre as malhas, e as soluções, em cada nível, são tomadas do ciclo anterior. O esquema FAS padrão restringe o resíduo e a solução.

Yan et al. (2007) fizeram uma validação geral do algoritmo apresentado em 1998, envolvendo esquemas de convecção de alta ordem, várias geometrias, diferentes tipos de malhas, modelos turbulentos e laminares 2D e 3D. No presente trabalho este algoritmo foi implementado no ciclo V para o roteiro com $\nu$ totalmente constante (seção 3.1), *solver* MSI e restrição por meia ponderação (restrição que obteve o menor tempo de CPU, conforme a seção 4.8).

Através dos testes realizados, verificou-se que para a malha 513x513 nós, o algoritmo de Yan e Thiele é 2,4% mais rápido em comparação com o FAS padrão. Para a malha 129x129 nós, o algoritmo de Yan e Thiele é 0,7% mais lento. Em média, o algoritmo de Yan e Thiele é cerca de 1,1% mais rápido que o FAS padrão.

### 4.10 Algoritmo ótimo

Um novo algoritmo foi elaborado unindo as melhores componentes algorítmicas encontradas neste trabalho. Suas principais características são:

- Roteiro: $\nu$ totalmente constante (Seção 4.1).
- *Solver*: MSI (Seção 4.7).
- Restrição: meia ponderação (Seção 4.8).
- Algoritmo de Yan e Thiele (Seção 4.9).

A Fig. 15 apresenta uma comparação entre o tempo de CPU obtido com o algoritmo ótimo e o FAS padrão (com o roteiro com $\nu$ totalmente constante, *solver* MSI e restrição por injeção). Observa-se que o algoritmo ótimo é um pouco mais rápido que o FAS padrão. Para a malha 2049x2049 nós, o tempo de CPU do algoritmo ótimo é cerca de 5,5% menor.

Para cada curva da Fig. 15, calculou-se o expoente *p*, obtido pelo do método dos mínimos quadrados, para a função dada por

$$t_{CPU}(N) \quad = \quad k\,N^{p}\,. \tag{6}$$

Para o método *multigrid* ideal, $p=1$, significando que o esforço computacional cresce linearmente com o tamanho da malha (Brandt, 1977; Trottenberg et al., 2001; e Hirsch, 1988). Assim, quanto menor for *p*, mais eficiente são o algoritmo, *hardware* e compilador utilizados. Obteve-se *p* = 1,09 e 1,08 respectivamente para o FAS padrão e o algoritmo ótimo.

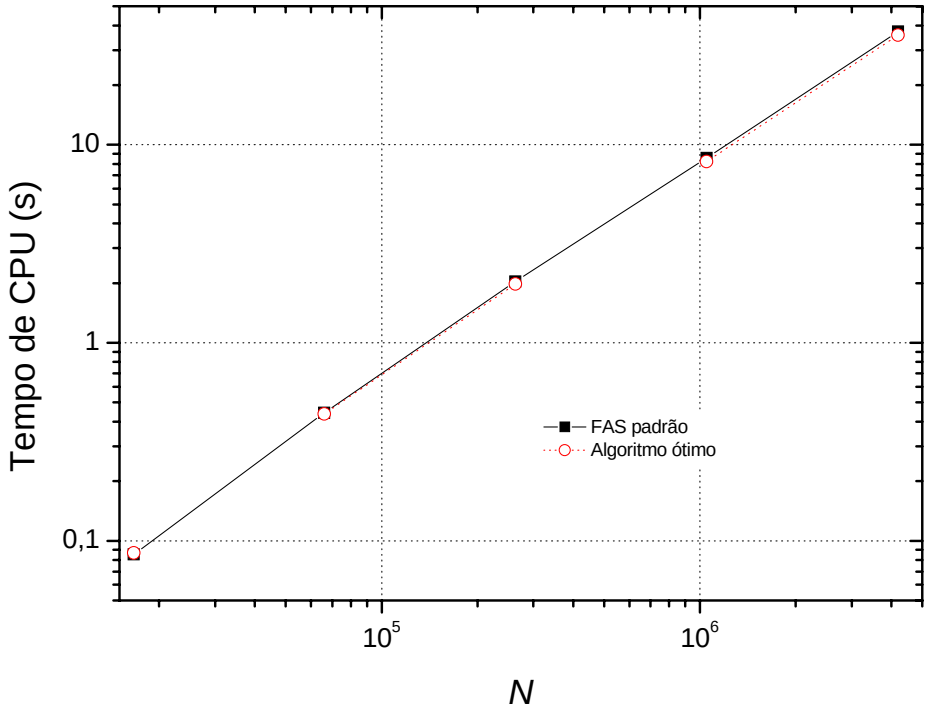


Figura 15 – Comparação entre o algoritmo ótimo e o FAS padrão.

Para o algoritmo ótimo, a Fig. 16 mostra a influência do número de iterações internas $(\nu)$ sobre o tempo de CPU nas malhas *N* = 129x129 a 2049x2049, com $L = L_{max}$. Verifica-se que o número ótimo de iterações internas é igual a 3 para todos os tamanhos de malha em estudo, isto é, $\nu_{ótimo} = 3$. Observa-se que este $\nu$ ótimo é igual ao obtido na seção 4.1, Fig. 7, para o algoritmo padrão.

Para o algoritmo ótimo, a Fig. 17 mostra a influência do número de níveis de malha (*L*) sobre o tempo de CPU. Para cada curva da Fig. 17, parte-se do valor máximo que *L* pode assumir para dado *N*, denotado por $L_{max}$; por exemplo, para a malha 513x513, $L_{max} = 9$. Este valor de *L* é diminuído até $L_{max}$-5. Para cada curva da Fig. 17, mostra-se o valor de $L_{ótimo}$, que é o *L* que resulta no menor tempo de CPU para se resolver o problema. Nota-se que o número de malhas pode influenciar significativamente o tempo de CPU. Verificou-se que para $\nu = 3$, o número ótimo de malhas é $L_{ótimo} = L_{max}$-2.

## 5. CONCLUSÃO

Neste trabalho estudou-se o efeito de diversos roteiros do método *multigrid* sobre o tempo de CPU para a equação de Laplace bidimensional. Os roteiros utilizados foram: $\nu$ totalmente constante, $\nu$ dinâmico, $\nu$ constante na restrição e na prolongação, roteiros tipo

Hortmann e roteiros tipo dente-de-serra. Também foi estudada a influência dos *solvers* (MSI, Gauss-Seidel e ADI), das restrições (injeção, meia ponderação e ponderação completa) e dois esquemas FAS (o padrão e o de Yan e Thiele). A equação foi discretizada através do método das diferenças finitas com aproximações numéricas de 2ª ordem e malhas uniformes. Usou-se o método *multigrid* geométrico com ciclo V e razão de engrossamento padrão.

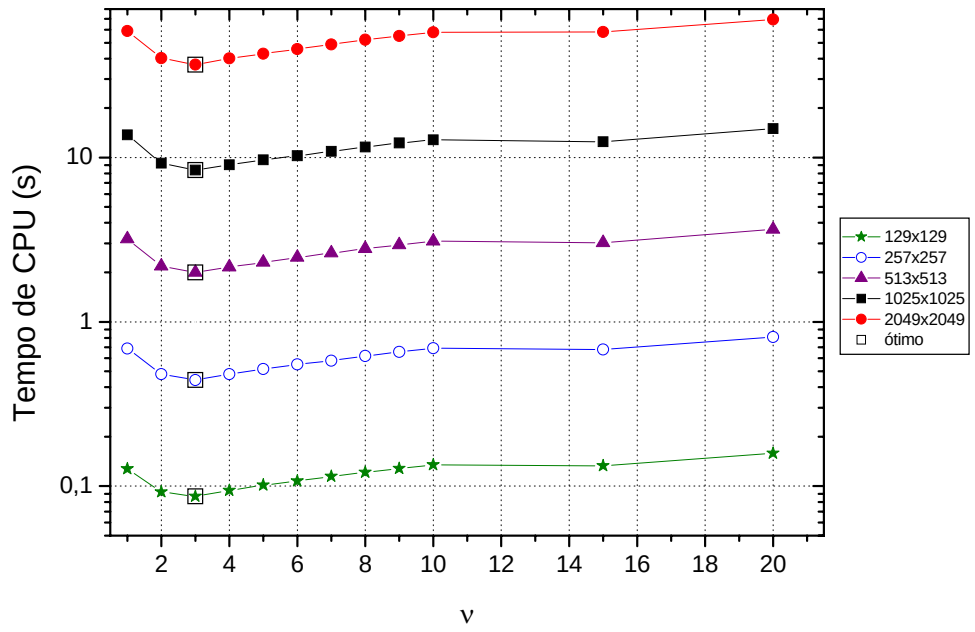


Figura 16 – Tempo de CPU versus número de iterações internas $(\nu)$ para o algoritmo ótimo com $L = L_{max}$.

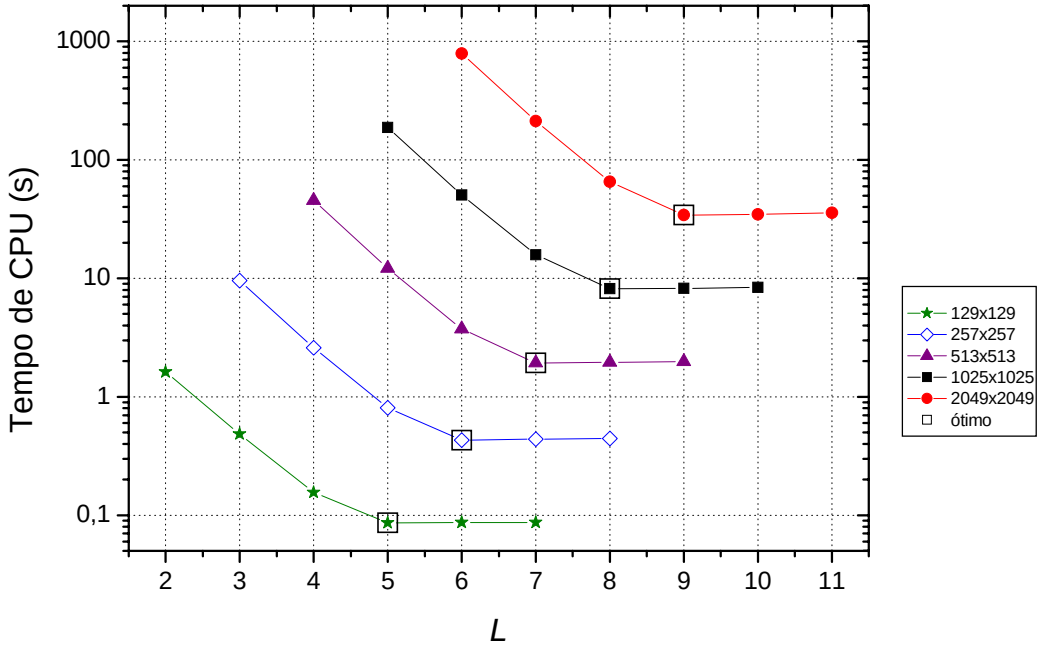


Figura 17 – Tempo de CPU versus número de níveis de malha ($L$) para o algoritmo ótimo com $\nu = 3$.

Com base nos resultados obtidos, verificou-se que:

1) O número ótimo de iterações internas ($\nu_{ótimo}$) no roteiro $\nu$ totalmente constante, com esquema FAS, *solver* MSI e restrição por injeção é igual a 3.
2) O roteiro $\nu$ totalmente constante, com $\nu = 3$ e $L = L_{max}$, apresenta o menor tempo de CPU.
3) O *solver* MSI é o mais rápido para problemas de grande porte e mais bem comportado no que diz respeito ao número de iterações internas.
4) A restrição por meia ponderação apresenta o menor tempo de CPU em relação à restrição por injeção e por ponderação completa. O tempo de CPU da restrição por meia ponderação é em média 0,7% menor que a restrição por injeção.
5) Em média, o algoritmo FAS de Yan e Thiele é cerca de 1,1% mais rápido que o FAS padrão.
6) O algoritmo (ótimo) que resulta no menor tempo de CPU é resultado das seguintes componentes *multigrid*: $\nu$ totalmente constante, *solver* MSI, restrição por meia ponderação, $\nu = 3$, $L = L_{max} - 2$, e o esquema FAS de Yan e Thiele.
7) O número de níveis de malha ($L$) pode afetar significativamente o tempo de CPU. Para o algoritmo ótimo, obteve-se $L_{ótimo} = L_{max} - 2$; este resultado é válido para qualquer tamanho de malha ($N$). Ao se utilizar $L_{max}$ em vez de $L_{ótimo}$, tem-se em média um aumento de 2,5% no tempo de CPU.
8) O expoente $p$ da Eq. (6) é de 1,09 para o esquema FAS padrão e de 1,08 para o algoritmo ótimo.

***Agradecimentos***

Os dois primeiros autores agradecem ao Laboratório de Experimentação Numérica (LENA), do Departamento de Engenharia Mecânica, da UFPR, por disponibilizar sua infra-estrutura. Eles também agradecem à Universidade Estadual de Ponta Grossa pelo suporte financeiro e aos amigos do LENA. Os autores agradecem ao MCT/CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, do Brasil) pelo apoio financeiro. O terceiro autor é bolsista do CNPq.

## REFERÊNCIAS

Brandt, A., 1977. Multi-Level Adaptive Solutions to Boundary-Value Problems, *Mathematics of Computation*, vol. 31, pp. 333-390.

Brandt A., Diskin B., & Thomas J.L., 2002. Recent advances in achieving textbook multigrid efficiency for computational fluid dynamics simulations. Hampton, VA, USA: ICASE/NASA; 2002. NASA/CR-211656.

Briggs, W. L., Henson, V.E., & Mccormick, S.F., 2000. *A Multigrid Tutorial*, 2ª ed., SIAM.

Burden, R. L., & Faires, J. D., 2003. *Análise Numérica*, Tradutor: Ricardo Lenzi Tombi. São Paulo: Pioneira Thomson Learning.

Chisholm, T., 1997. *Multigrid Acceleration of an Approximately-Factored Algorithm for Steady Aerodynamic Flows*, Dissertação de mestrado, University of Toronto.

Donohue, D. J., Ku, H-C., & Thompson D.R. S., 1998. Application of Iterative Moment-Method Solutions to Ocean Surface Radar Scattering, *IEEE Transactions on Atennas and propagation*, vol. 46, n. 1, January.

Ferziger, J. H., & Peric, M., 1999. *Computational Methods for Fluid Dynamics*, 2 ed., Springer.

Gerolymos, G.A., & Vallet I.., 2005. Mean-Flow-Multigrid for Implicit Reynolds-Stress-Model Computations, *AIAA Journal*. vol. 43, n. 9, September.

Ghia, U., Ghia, N., & Shin, C.T., 1982. High-Re solutions for incompressible flow using the Navier-Stokes equations and a Multigrid method, *Journal of Computational Physics*, vol.48, pp. 387-411.

Golub, G. H. & Ortega, J. M., 1992. *Scientific Computing and Differential Equations: An Introduction to Numerical Methods*, Academic Press, Inc.

Hirsch, C., 1988. *Numerical Computational of Internal and External Flows*, v.1, Wiley.

Hortmann, M., Peric. M., & Scheuerer, G., 1990. Finite Volume Multigrid Prediction of Laminar Natural Convection: Bench-Mark Solutions, *International Journal for Numerical methods in Fluid*. vol.11, 189-207.

Incropera, F. P., & DeWitt, D. P., 1998. *Fundamentos de Transferência de Calor e Massa*. 4 ed. Rio de Janeiro:LTC Editora.

Larsson, J., Lien, F. S., & Yee, E., 2005. Conditional Semicoarsening Multigrid Algorithm for the Poisson Equation on Anisotropic Grids, *Journal of Computational Physics*, vol. 208, pp. 368-383.

Manzano, L., 1999. *Implementation of Multigrid for Aerodynamic Computations on Multi-Block Grids*, Dissertação de mestrado. Department of aerospace science and engineering, University of Toronto.

Mesquita, M. S., & De Lemos, M. J. S., 2004. Optimal Multigrid Solutions of Two-dimensional Convection-Conduction Problems, *Applied Mathematics and Computation*, vol. 152, pp. 725-742.

Pinto, M. A. V., & Marchi, C. H., 2006. Efeito dos Parâmetros do Método Multigrid CS e FAS sobre o tempo de CPU para a Equação de Laplace Bidimensional, *Proceedings of ENCIT*.

Pinto, M. A. V., Santiago, C. D., & Marchi, C.H., 2005. Effect of Parameters of a Multigrid Method on CPU Time for One-dimensional Problems, *Proceedings of COBEM*.

Rabi, J.A., & De Lemos, M.J.S., 2001. Optimization of convergence acceleration in multigrid numerical solutions of conductive-convective problems, *Applied Mathematics and Computation*, vol. 124, pp.215-226.

Roache, P. J., 1998. *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers.

Santiago, C.D., & Marchi, C.H., 2007. Optimum Parameters of a Geometric Multigrid for a Two-Dimensional Problem of Two-Equations, Proceedings of COBEM.

Schneider, G. E., & Zedan, M., 1981. A Modified Strongly Implicit Procedure for Numerical Solution of Field Problems, *Numerical Heat Transfer*, vol. 4, pp. 1-19.

Tannehill, J. C., Anderson, D. A., & Pletcher, R. H., 1997. *Computational Fluid Mechanics and Heat Transfer*, 2 ed., Washington: Taylor & Francis.

Trottenberg, U., Oosterlee, C., & Schüller, A., 2001. *Multigrid*, Academic Press.

Wesseling, P., 1992. *An Introduction to Multigrid Methods*, John Wiley & Sons.

Wesseling, P., & Oosterlee, C. W., 2001. Geometric Multigrid with Applications to Computational Fluid Dynamics, *Journal of Computation and Applied Mathematics*, vol. 128, pp. 311-334.

Yan, J., & Thiele, F. A., 1998. Modified Full Multigrid Algorithm for the Navier-Stockes Equations. *Numerical heat transfer*, Part B, 34:323-338.

Yan, J. Thiele, F., & Xue L., 2007. Performance and Accuracy of an Modified Full Multigrid Algorithm for Fluid Flow and Heat Trasfer. *Computer & Fluids* 36, pp. 445-454.

Zeeuw, P. M., 1996. Development of Semi-Coarsening Techniques. *Applied Numerical Mathematics*, pp. 433-465.