MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE MATEMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM MATEMÁTICA APLICADA

A Transformada Wavelet Discreta Incompleta

Aplicada à

Resolução das Equações de Poisson

por

Simone de Fátima Tomazzoni Gonçalves

Dissertação submetida como requisito parcial para obter o grau
de Mestre em Matemática Aplicada

Prof. Dr. Oclide José Dotto
Orientador

Porto Alegre, novembro de 2002.

A TRANSFORMADA WAVELET DISCRETA INCOMPLETA APLICADA À RESOLUÇÃO DAS EQUAÇÕES DE POISSON

por

Simone de Fátima Tomazzoni Gonçalves

Dissertação submetida ao Programa de Pós-Graduação em Matemática Aplicada do Instituto de Matemática da Universidade Federal do Rio Grande do Sul, como requisito parcial para obtenção do grau de

Mestre em Matemática Aplicada

Linha de Pesquisa: Algoritmos Numéricos e Algébricos
Orientador: Prof. Dr. Oclide José Dotto
Banca Examinadora:

Prof. Dr. Jacques Aveline Loureiro da Silva
Prof. Dr. João Goedert
Prof. Dr. José Afonso Barrionuevo

Dissertação apresentada e aprovada
em 14 de novembro de 2002.

Prof. Dr. Vilmar Trevisan
Coordenador

Ao meu namorado Claitor Mazzochi,
que sonhou comigo essa realidade,
com amor, paciência e incentivo.

Somehow I cannot believe that there are any heights that cannot be scaled by a man who knows the secrets of making dreams come true. This special secret, it seems to me, can be summarized in four C's. They are: curiosity, confidence, courage, and constancy, and the greatest off all is confidence. When you believe in a thing, believe in it all way, implicitly and unquestionable.

by Walt Disney

... the development of wavelets is an example where ideas from many different fields combined to merge into a whole that is more than the sum of its parts – many of the applications that will described in the review articles in the remainder of this issue would not have been possible if this merging had not taken place. I have very much enjoyed being a part of this process. I wish there were many more examples of such interconnections – we would all benefit from them.

by Ingrid Daubechies

Agradecimentos

Gostaria de deixar um agradecimento especial ao meu orientador, prof. Dr. Oclide José Dotto, com quem tive o privilégio aprender e compartilhar muitas coisas sobre a Matemática e muitas coisas sobre a vida.

À minha amada família: meu pai Alziro que, embora longe, sei que ficará feliz por minha conquista; minha mãe Elizabete que sofreu minha ausência, especialmente nos últimos meses; aos meus maravilhosos irmãos Vana e Eri, dos quais me orgulho imensamente.

Aos meus professores do curso de Licenciatura Plena em Matemática, da Universidade de Caxias do Sul, que me incentivaram a ingressar no programa de Mestrado.

Agradeço também os professores do PPGMAP/UFRGS Trevisan, Jacques e Rudinei, pelos ensinamentos e pela disposição em ajudar.

Aos meus colegas de estudos José Caleffi, Lucile Jacoby, Márcia Lourenço , Sérgio Rech e Fábio Pinto, pela turma maravilhosa que foram e que hoje chamo amigos.

Aos pais de meu namorado, Castor e Jurema, pela colaboração, compreensão e torcida.

Gostaria também de agradecer alguns amigos que eu tinha antes e outros que ganhei no decorrer do caminho: Adalberto Dornelles, por compartilhar sua experiência; Silvana Kissman por ser uma grande colega e amiga; Helena Libardi, por sua doçura; Márcia e Vitor Onzi, pelo apoio e pelo empréstimo do computador; Marla Michelli, pela carona e por minha afilhada Mirela; Rejane Pergher, pelas caronas; Mauren Turra, Eduardo Nabinger e sua namorada, Milene, pelo empréstimo de livros e *papers*; Janecyr, pelas tentativas de conciliar meus horários. À minha tia Elizete, minha prima Ilma e às amiguinhas Gicele e Lidi, pelos momentos de descontração.

Sumário

Lista de figuras

Lista de Tabelas

Listas de notações e abreviaturas

Lista de notações

$\hat{f}(\omega)$	: transformada de Fourier da função $f(x)$
$\|f\|$	: norma da função f
$\|f\|_p$	: norma-p da função f
$\langle f, g \rangle$	: produto interno das funções f e g
$\mathbf{A}^t$	: transposta da matriz **A**
$\mathbf{A}^{-1}$	: inversa da matriz **A**
$\kappa(\mathbf{A})$	: número de condição da matriz **A**
$\oplus$	: soma direta de espaços vetoriais
$\otimes$	: produto direto ou produto de Kronecker de espaços vetoriais ou matrizes
g_k	: coeficientes da equação de refinamento da função wavelet
h_k	: coeficientes da equação de refinamento da função de escala
c_k^j	: coeficientes da função de escala
d_k^j	: coeficientes da função wavelet
L^2, $L^2(\mathbb{R})$	: espaço de Hilbert das funções de quadrado integrável
$W(s, \tau)$	: Transformada wavelet contínua
V_j	: subespaço de multi-resolução nível j
W_j	: subespaço de multi-resolução (detalhes) nível j
M	: número de momentos nulos
$\varphi(x)$	: função de escala
$\tilde{\varphi}(x)$	: função de escala dual

$\psi(x)$	: função wavelet
$\tilde{\psi}(x)$	: função wavelet dual
$\hat{\psi}(\omega)$	: transformada de Fourier da função $\psi(x)$

Lista de abreviaturas

AMR	: análise multi-resolução
GCCi	: método do gradiente conjugado precondicionado pela fatoração incompleta de Cholesky
MGC	: método do gradiente conjugado
MGCP	: método do gradiente conjugado precondicionado
SOR	: sucessive over relaxation method
SPD	: simétrica positiva definida
TF	: transformada de Fourier
TWC	: transformada wavelet contínua
TWD	: transformada wavelet discreta
TWDi	: transformada wavelet discreta incompleta
TWI	: transformada wavelet inversa

RESUMO

Apresentamos a transformada wavelet discreta incompleta, e a aplicamos no precondicionamento de sistemas de equações lineares, originados na discretização de problemas de contorno de Poisson. Esses sistemas podem ser resolvidos por algum método iterativo, mas a velocidade de convergência piora rapidamente com o aumento do número de nodos da malha de discretização. O precondicionamento mediante wavelets tem a propriedade de que, mediante uma mudança de escala pelo método da diagonal limita a variação do número de condição, vantagem aproveitada por G. Beylkin [03, 04, 05] na solução matricial do sistema linear. O método de Beylkin, no entanto, tem diversos problemas práticos e é computacionalmente difícil. A transformada wavelet discreta incompleta, que modifica o método de Berylkin, aproximando a transformada wavelet discreta (completa), resolve as dificuldades e é de fácil implementação computacional. Especificamente, mostraremos mediante estudos experimentais, que, com o precondicionamento decorrente da transformada wavelet discreta incompleta, aplicado ao método do gradiente conjugado, os resultados numéricos confirmam os efeitos e vantagens do método proposto.

ABSTRACT

We present the incomplete discrete wavelet transform and we apply it for preconditioning a system of linear equations, originated in the discretization of Poisson boundary problems. Such systems can be solved by some iterative method, but the convergence speed worsens quickly with the increase of the condition number of the coefficients matrix, and that number increases exponentially with the number of discretization mesh nodes. The wavelets preconditioning has the property that a diagonal rescaling bounds the condition number, and G. Beylkin [03, 04, 05] took advantage of that in a matrix solver. The Beylkin´s method, however, has several practical problems and is computationally difficult. The incomplete discrete wavelet transform, that modifies Beylkin's method approximating the (complete) discrete wavelet transform, solves the difficulties, and is of easy computational implementation. Specifically, we will show by experimental studies that the incomplete discrete wavelet transform preconditioning, applied to the method of conjugated gradient, produces numeric results that confirm the effects and advantages.

Introdução

A necessidade de usar simulações numéricas vem crescendo muito em muitos campos da engenharia. Uma das causas dessa tendência é o alto custo da realização de experimentos físicos e o desenvolvimento de computadores cada vez mais potentes que permitem rapidez nos resultados. Ainda assim, a maior parte do tempo consumido na análise desses fenômenos é consumida durante a resolução das equações que os descrevem. Essas equações são aproximadas e transformadas, por meio de um método de discretização adequado, em um grande sistema de equações lineares. Ao contrário do que ocorro com as equações integrais, a discretização de uma equação diferencial produz um sistema esparso $\mathbf{Ax} = \mathbf{y}$ com número de condição grande [03]. Entretanto, estudos mostram que é numericamente importante um tal sistema em outro bem condicionado. O número de condição da matriz **A** mede a estabilidade do sistema. A velocidade de convergência diminui com o crescimento do número de condição de **A**. Esse número cresce exponencialmente com o aumento do número de pontos da malha de discretização. Por esse motivo, mesmo com computadores de alto desempenho, se torna necessário o desenvolvimento de novos algoritmos que diminuam o custo computacional.

Em um estudo, Beyklin [03] propôs um novo algoritmo para resolver um sistema linear, usando a transformada wavelet discreta que controla o crescimento do número de condição da matriz do sistema. Outros pesquisadores [21, 36, 37] têm estudado o uso de bases wavelet na solução desses sistemas lineares. Como resultado desses estudos, foi provado [21] que o uso de wavelets produz um sistema linear esparso, devido ao suporte compacto das wavelets e que, após o precondicionamento, o número de condição torna-se aproximadamente independente do tamanho do problema, devido à estrutura de multi-resolução.

Neste trabalho, apresentamos o precondicionamento do método do gradiente conjugado, baseado em wavelets, como uma alternativa para a resolução de uma classe desses sistemas, comprovando que o uso da matriz precondicionadora com base wavelets

mantém a esparsidade da matriz e limita o número de condição, o que diminui o tempo computacional.

Organizamos nossa dissertação como segue. No capítulo 1, **Apresentação do Problema**, discutimos o sistema linear correspondente ao problema de contorno de Poisson unidimensional e bidimensional, com algum comentário a respeito da teoria do número de condição. No capítulo 2, **A Transformada Wavelet e Análise Multi-resolução**, o mais longo, apresentamos alguns aspectos da teoria de wavelets necessários ao desenvolvimento desse trabalho: transformada wavelet contínua, análise multi-resolução e transformada wavelet discreta. O capítulo 3, **Descrição do Método da Transformada Wavelet Discreta Incompleta e Resultados Numéricos**, contem uma descrição do precondicionamento e os resultados numéricos que confirmam os efeitos e vantagens do método proposto. Nosso maior trabalho e maior originalidade concentram-se precisamente nesse último capítulo, onde tivemos que lidar com sistemas lineares de grande porte, o que exigiu a estruturação de algoritmos adequados, que implementamos no MATLAB. Em particular, descobrimos que o produto de Kronecker (produto tensorial, ou produto direto) é muito adequado para construir o caso bidimensional do problema de Poisson, a partir de wavelets unidimensionais.

1 APRESENTAÇÃO DO PROBLEMA

Neste capítulo apresentamos principalmente os preliminares que interessam ao nosso objetivo. Abordaremos a equação de Poisson unidimensional e bidimensional, algumas considerações sobre número de condição e precondicionamento, uma breve justificativa sobre o uso de wavelets no precondicionamento do sistema linear correspondente a uma discretização do problema de Poisson, bem como um estudo dos autovalores das matrizes oriundas da discretização das equações de Poisson unidimensionais e bidimensionais.

A equação de Poisson e outra, relacionada com ela, a equação de Laplace, surgem em muitas aplicações, incluindo eletromagnetismo, mecânica dos fluidos, fluxo de calor, difusão e mecânica quântica, para enumerar algumas.

1.1 Problema de contorno de Poisson unidimensional

Seja a versão unidimensional da equação de Poisson,

$$-\frac{d^2v(x)}{dx^2} = f(x), \qquad 0 < x < 1, \tag{1.1}$$

onde f é uma função dada e v é a função incógnita que deve satisfazer também condições de fronteira, de Dirichlet por exemplo, $v(0) = v(1) = 0$. Para uma solução aproximada do problema, o discretizamos e calculamos valores aproximados de v em $N+2$ pontos igualmente espaçados x_i entre 0 e 1: $x_i = ih$, onde $h := 1/(N+1)$, e $i = 0, 1, \ldots, N+1$. Usaremos as notações $v_i := v(x_i)$ e $f_i := f(x_i)$. Para converter a equação diferencial (1.1) em um sistema linear aproximativo com incógnitas $v_1, v_2, \ldots, v_n$, usando diferenças finitas centrais, procedemos como segue.

$$\left.\frac{dv(x)}{dx}\right|_{x=(i-0.5)h} \approx \frac{v_i - v_{i-1}}{h},$$

$$\left.\frac{dv(x)}{dx}\right|_{x=(i+0.5)h} \approx \frac{v_{i+1} - v_i}{h};$$

subtraindo a segunda aproximação da primeira e dividindo por h, temos a derivada segunda por diferenças centrais

$$-\left.\frac{d^2v(x)}{dx^2}\right|_{x=x_i} = \frac{2v_i - v_{i-1} - v_{i+1}}{h^2} - \tau_i, \tag{1.2}$$

onde τ_i, chamado erro de truncamento, é, conforme podemos demonstrar, um

$$O\left(h^2 \left\| \frac{d^4v}{dx^4} \right\|_\infty\right).$$

Podemos escrever a equação (1.1) discretizada na forma

$$-v_{i-1} + 2v_i - v_{i+1} = h^2 f_i + h^2 \tau_i,$$

onde $i = 1, 2, \ldots, N$. O conhecimento dos valores da função na fronteira implica N equações lineares com N incógnitas $v_1, \ldots, v_N$, ou a equação matricial

$$\mathbf{T}_N \mathbf{v} := \begin{bmatrix} 2 & -1 & \cdots & 0 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ 0 & \cdots & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ \vdots \\ v_N \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ \vdots \\ \vdots \\ f_N \end{bmatrix} + h^2 \begin{bmatrix} \tau_1 \\ \vdots \\ \vdots \\ \tau_N \end{bmatrix}, \tag{1.3}$$

ou ainda, compactamente,

$$\mathbf{T}_N \mathbf{v} = h^2 \mathbf{f} + h^2 \boldsymbol{\tau}. \tag{1.4}$$

Para resolver esse sistema, ignoramos $\boldsymbol{\tau}$, pois é insignificante comparado a $\mathbf{f}$, e obtemos

$$\mathbf{T}_N\mathbf{v} = h^2\mathbf{f}. \tag{1.5}$$

A matriz dos coeficientes $\mathbf{T}_N$ é uma matriz tridiagonal e desempenha papel central em tudo que segue. No MATLAB 6.x é gerada com o comando `gallery('tridiag', N)`. Examinaremos essa matriz em detalhes, posteriormente.

1.2 Problema de contorno de Poisson bidimensional

Seja a equação de Poisson bidimensional,

$$-\frac{\partial^2 v(x,y)}{\partial x^2} - \frac{\partial^2 v(x,y)}{\partial y^2} = f(x,y) \tag{1.6}$$

sobre o quadrado unitário $\{(x, y): 0 < x, y < 1\}$ com condição de fronteira $v(x, y) = 0$. Tomamos uma malha de pontos (x_i, y_j), com $x_i = ih$, $y_j = ih$, $h = 1/(N+1)$ e $i, j = 0, 1, 2, \ldots, N+1$ (eventualmente o número dos pontos x_i pode ser diferente do número dos pontos y_j). Semelhantemente ao que fizemos no caso unidimensional, pomos $v_{ij} := v(ih, jh)$ e $f_{ij} := f(ih, jh)$, Fig. 1.1 (para $N = 3$).

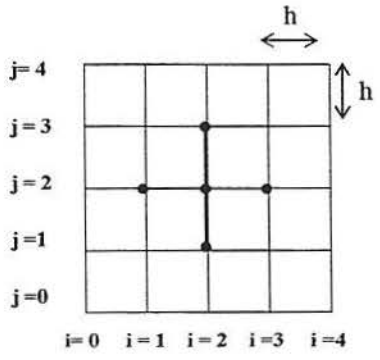

Fig. 1.1

A fórmula (1.2) nos permite escrever

$$-\left.\frac{\partial^2 v(x,y)}{\partial x^2}\right|_{x=x_i,y=y_j} \approx \frac{2v_{ij}-v_{i-1,j}-v_{i+1,j}}{h^2}, \tag{1.7}$$

e

$$-\left.\frac{\partial^2 v(x,y)}{\partial y^2}\right|_{x=x_i,y=y_j} \approx \frac{2v_{ij}-v_{i,j-1}-v_{i,j+1}}{h^2}. \tag{1.8}$$

Adicionando essas duas aproximações, vem

$$-\frac{\partial^2 v(x,y)}{\partial x^2}-\left.\frac{\partial^2 v(x,y)}{\partial y^2}\right|_{x=x_i,y=y_j} = \frac{4v_{ij}-v_{i-1,j}-v_{i+1,j}-v_{i,j-1}-v_{i,j+1}}{h^2}-\tau_{ij}, \tag{1.9}$$

onde τ_{ij} é o erro de truncamento, que é um O(h^2). A cruz no meio da Fig. 1.1 conecta 5 nodos da malha. Das condições de fronteira que tomamos resulta $v_{0j} = v_{N+1,j} = v_{i0} = v_{i,N+1} = 0$. Dessa forma, a equação (1.9) define um conjunto de N^2 equações lineares a N^2 incógnitas v_{ij}, vindas de fazer $i, j = 1, 2, \ldots, N$:

$$4v_{ij}-v_{i-1,j}-v_{i+1,j}-v_{i,j-1}-v_{i,j+1} = h^2 f_{ij}. \tag{1.10}$$

A fórmula (1.10), por razões óbvias, é conhecida como a *fórmula a diferenças de cinco pontos*.

Existem duas maneiras de escrever as N^2 equações representadas pela equação (1.10) com uma única equação matricial. A primeira maneira é dispor as incógnitas v_{ij} como numa matriz quadrada $\mathbf{V} = [\, v_{ij} \,]$ de ordem N, e o lado direito, como uma matriz quadrada $h^2\mathbf{F} = [\, h^2 f_{ij} \,]$, também de ordem N. A estratégia consiste em escrever a matriz com elementos $4v_{ij}-v_{i-1,j}-v_{i+1,j}-v_{i,j-1}-v_{i,j+1}$ na posição (i, j), de uma forma simples, em termos de $\mathbf{V}$ e $\mathbf{T}_N$. Observemos que

$$2v_{ij}-v_{i-1,j}-v_{i+1,j} = (\mathbf{T}_N\mathbf{V})_{ij},$$

$$2v_{ij}-v_{i,j-1}-v_{i,j+1} = (\mathbf{V}\mathbf{T}_N)_{ij}.$$

Adicionando essas duas equações, obtemos

$$(\mathbf{T}_N\mathbf{V} + \mathbf{V}\mathbf{T}_N)_{ij} = 4v_{ij} - v_{i-1,j} - v_{i+1,j} - v_{i,j-1} - v_{i,j+1} = h^2 f_{ij} = (h^2\mathbf{F})_{ij},$$

ou

$$\mathbf{T}_N\mathbf{V} + \mathbf{V}\mathbf{T}_N = h^2\mathbf{F}. \tag{1.11}$$

A segunda maneira de escrever as N^2 equações representadas por (1.10) como uma equação matricial única é obtida dispondo os N^2 elementos no segundo membro num único vetor-coluna. Isso requer a escolha de uma ordem para as incógnitas. Nossa escolha está ilustrada na Fig. 1.2.

$\mathbf{V} =$

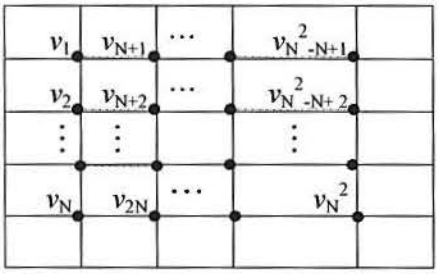

Fig. 1.2 – Número e ordenação das incógnitas na equação de Poisson

1.2.1 Exemplo. Se tomamos $N = 3$, adotando a segunda maneira, a equação (1.10) gera a equação matricial

$$\mathbf{T}_{N^2}\mathbf{v} = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ v_8 \\ v_9 \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ f_8 \\ f_9 \end{bmatrix}. \quad \square$$

Para um N geral, obtemos um sistema linear quadrado de ordem N^2,

$$\mathbf{T}_{N^2}\mathbf{v} = h^2\mathbf{f}, \tag{1.12}$$

onde $\mathbf{T}_{N^2}$ tem N blocos quadrados de ordem N do tipo $\mathbf{T}_N + 2\mathbf{I}_N$ na diagonal principal e as negativas $-\mathbf{I}_N$ das matrizes identidades nas subdiagonais inferior e superior:

$$\mathbf{T}_{N^2} = \begin{bmatrix} \mathbf{T}_N + 2\mathbf{I}_N & -\mathbf{I}_N & & \\ -\mathbf{I}_N & \ddots & \ddots & \\ & \ddots & \ddots & -\mathbf{I}_N \\ & & -\mathbf{I}_N & \mathbf{T}_N + 2\mathbf{I}_N \end{bmatrix}. \tag{1.13}$$

A matriz (1.13) é normalmente chamada de matriz de Poisson e, no MATLAB, é produzida com o comando `gallery('poisson', N)`.

1.3 Número de condição de uma matriz e convergência do método do gradiente conjugado

1.3.1 Número de condição. Teoricamente, para uma matriz quadrada **A**, existe uma única solução **x** do sistema linear **Ax** = **y**, para cada **y**, se e somente se **A** é uma matriz inversível. Entretanto, na prática, existem aspectos importantes que são cruciais. Um deles é o número de condição da matriz.

1.3.1.1 Exemplo. Consideremos o sistema linear **Ax** = **y**, onde $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$, e

$$\mathbf{A} = \begin{bmatrix} 5.95 & -14.85 \\ 1.98 & -4.94 \end{bmatrix}.$$

O determinante de **A** é 0.01, diferente de zero, logo **A** é inversível. Para $\mathbf{y} = [3.05 \quad 1.02]^t$, a solução $\mathbf{x} = [8 \quad 3]^t$ é única. Agora, tomando $\mathbf{y}' = [3 \quad 1]^t$, a nova solução é $\mathbf{x}' = [3 \quad 1]^t$. Observemos que **y** e **y'** são próximos, mas as soluções **x** e **x'** são muito diferentes. Um sistema linear para o qual isto acontece é dito *mal-condicionado*.

Para um sistema mal-condicionado, pequenos erros nos dados em **y** podem levar a grandes erros na solução **x**. É indesejável tomar isso em conta nas aplicações, porque

em todos os cálculos com dados reais existem erros, seja devido aos arredondamentos e truncamentos (visto que computadores podem calcular somente com um grau de precisão finito), seja devido às medições imperfeitas que produzem os dados. Uma solução de um sistema mal-condicionado pode ser não-significativa fisicamente.

Voltemos ao nosso exemplo acima. Diagonalizando **A**, temos indicativos do que ocorre. Vale $\mathbf{Ax'} = \mathbf{x'}$, isto é, $\mathbf{x'}$ é um autovetor de **A** associado ao autovalor 1. O vetor $\mathbf{x''} := \mathbf{x} - \mathbf{x'} = [5 \;\; 2]^t$ satisfaz $\mathbf{Ax''} = 0.01\mathbf{x''}$, de modo que $\mathbf{x''}$ é um autovetor de **A**, associado ao autovalor 0.01. Então **A** é semelhante à matriz diagonal

$$\begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix}.$$

Em particular, na direção $\mathbf{x'}$, **A** comporta-se como a matriz identidade: perturbando **y** na direção $\mathbf{x'}$, resulta a mesma perturbação na solução **x**. Entretanto, na direção $\mathbf{x''}$ **A** atua multiplicando por 0.01 os vetores; dessa forma, uma perturbação de **y** na direção $\mathbf{x''}$ produz uma perturbação na solução 100 vezes maior de **x**. Essa diferença de comportamento de **A** nas diferentes direções é a chave do problema. □

Uma conclusão que poderia ser tirada do exemplo anterior é que uma maneira de determinar se um sistema é mal-condicionado é olhar para o seu determinante ou para o seu menor autovalor. Entretanto, isso não é correto, porque esses números não representam escala apropriada. Por exemplo, se multiplicarmos a matriz **A** do Exemplo 1.3.1.1 por 10, multiplicaremos cada autovalor por 10 e o determinante por 100, mas a natureza básica da matriz não muda. Isso sugere que devemos olhar para quantidades que são invariantes com a escala. O número de condição satisfaz essa propriedade. Ele, juntamente com a *distribuição* dos autovalores da matriz e controla a velocidade de convergência dos algoritmos iterativos para resolver sistemas lineares.

1.3.1.2 Definição. *Seja* **A** *uma matriz quadrada inversível. Definimos e denotamos o número de condição de* **A** *por*

$$\kappa(\mathbf{A}) := \|\mathbf{A}\|\|\mathbf{A}^{-1}\|,$$

onde $\|\mathbf{A}\| = \sup\|\mathbf{Ax}\|$, *com* $\|\mathbf{x}\|$ *é a norma euclidiana em* $\mathbb{R}^n$, $\|\mathbf{x}\| = \sqrt{\langle x, x\rangle} = \sqrt{x_1^2 + \ldots x_n^2}$; *se* **A** *não é inversível, pomos* $\kappa(\mathbf{A}) = +\infty$.

Poderíamos usar qualquer norma matricial na Definição 1.3.1.2, e mesmo dar uma definição independente da norma, mas preferimos adotar a definição com a norma euclidiana. Para $c \neq 0$,

$$\kappa(c\mathbf{A}) = \|c\mathbf{A}\|\left\|(c\mathbf{A})^{-1}\right\| = \|c\mathbf{A}\|\left\|c^{-1}\mathbf{A}^{-1}\right\| = \kappa(\mathbf{A}),$$

isto é, o número de condição é invariante por multiplicação por escalar. Também, para qualquer matriz quadrada **A**, $\kappa(\mathbf{A}) \geq 1$. Para matrizes simétricas o número de condição pode ser calculado usando o Teorema seguinte [16] .

1.3.1.3 Teorema. *Seja* **A** *uma matriz quadrada simétrica inversível e*

$$|\lambda|_{\max} := \max\{|\lambda| : \lambda \text{ } e' \text{ } um \text{ } autovalor \text{ } de \text{ } \mathbf{A}\},$$
$$|\lambda|_{\min} := \min\{|\lambda| : \lambda \text{ } e' \text{ } um \text{ } autovalor \text{ } de \text{ } \mathbf{A}\}.$$

Então,

$$\kappa(\mathbf{A}) = \frac{|\lambda|_{\max}}{|\lambda|_{\min}}.$$

O Teorema 1.3.1.3 vale para matrizes mais gerais, matrizes complexas **A** que comutam com sua conjugada transposta, ditas matrizes normais, mas precisaremos apenas do caso especial acima.

1.3.2 Análise da convergência do MGC. O método do gradiente conjugado (MGC) é um dos métodos iterativos, que podemos utilizar para resolver o sistema linear proveniente da discretização da equação de Poisson, cujas matrizes são simétricas positivas definidas (SPD) (cf. Apêndice A) e esparsas. O MGC na verdade é um método direto, pois, teoricamente, se o sistema é quadrado de ordem N, após exatamente N iterações, apresenta a solução exata. Mas, por certas razões (erros de computador,

tamanho do sistema), é considerado iterativo. Métodos iterativos não produzem a resposta exata após um número finito de passos, porém o erro, até certo ponto, sofre redução de uma iteração para a seguinte. O processo iterativo termina quando o erro é menor do que um limite inferior pre-estabelecido. O erro final depende do número de iterações realizadas assim como das propriedades do método e do sistema linear.

Uma estimativa da razão de convergência do MGC é [25]

$$\|\mathbf{x}_k - \mathbf{x}\| \leq 2\alpha^k \|\mathbf{x}_0 - \mathbf{x}\|_{\mathbf{A}},$$

onde

$$\alpha := \frac{\sqrt{\kappa - 1}}{\sqrt{\kappa + 1}}, \quad \kappa := \kappa(\mathbf{A}) = \frac{\lambda_N}{\lambda_1},$$

e λ_1 e λ_N são o menor e o maior autovalores, respectivamente (os autovalores de uma matriz SPD são reais positivos).

Observamos que, se **A** é otimamente condicionada, isto é, se o número de condição $\kappa(\mathbf{A}) = 1$, então $\alpha = 0$; e, ao contrário, quando $\kappa(\mathbf{A}) \to \infty$, então $\alpha \to 1$. Portanto, quanto mais mal-condicionada é a matriz **A**, mais lenta é a convergência do MGC.

A análise do número de condição não explica totalmente o comportamento da convergência do MGC. O exemplo a seguir mostra que a distribuição dos autovalores da matriz dos coeficientes de um sistema linear influencia a convergência do MGC.

1.3.1.2 Exemplo. Seja $\mathbf{A}_{256\times256}$ a matriz oriunda da discretização da equação de Poisson bidimensional, e $\mathbf{D}_i$ uma matriz diagonal, com a mesma ordem de **A,** com *i* autovalores distintos, construída de tal forma que tenha o maior e o menor autovalores iguais, respectivamente, ao maior e ao menor autovalores de **A**, vindo então $\kappa(\mathbf{A}) = \kappa(\mathbf{D}_i)$. A Tabela 1.1 apresenta o número de iterações necessárias para a convergência do MGC.

Tabela 1. 1 – Número necessário de iterações para a convergência do MGC

	$\mathbf{D}_3$	$\mathbf{D}_5$	$\mathbf{D}_7$	$\mathbf{D}_9$	$\mathbf{D}_{11}$	$\mathbf{D}_{13}$	**A**
Número de iterações	3	5	7	9	11	25	25

Uma conclusão, a partir dos resultados da Tabela 1.1, é que a taxa de convergência do MGC não decorre apenas do número de condição da matriz; a distribuição dos autovalores também é fator importante nessa análise. □

1.4 Estudo dos autovalores da matriz $\mathbf{T}_N$

1.4.1 Lema. *Os autovalores de* $\mathbf{T}_N$ *são*

$$\lambda_j = 2\left(1-\cos\frac{j\pi}{N+1}\right),$$

e os correspondentes autovetores são $\mathbf{z}_j = [z_{kj}]$, *com*

$$z_{kj} = \sqrt{\frac{2\pi}{N+1}}\operatorname{sen}\left(\frac{jk}{N+1}\right), \quad k, j = 1, 2, \ldots, N.$$

Além disso, $\mathbf{z}_j$ *são vetores ortonormais.*

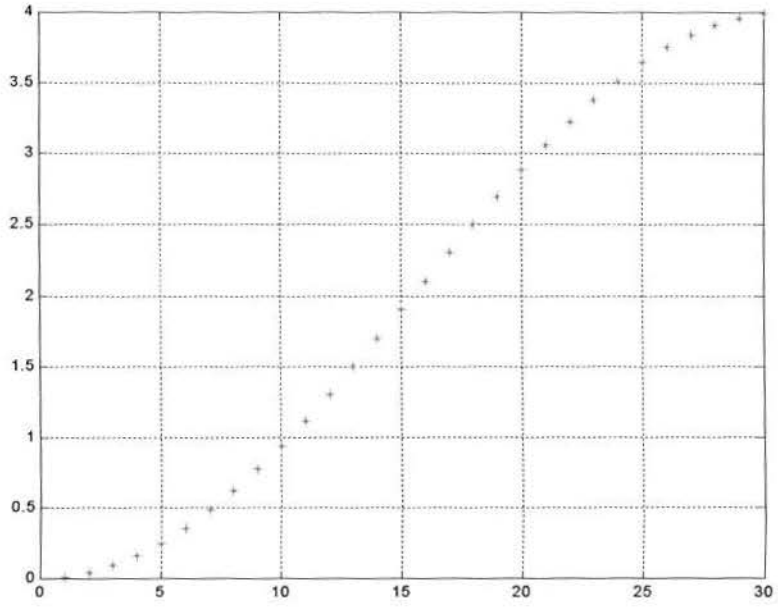

Fig. 1.3 – Autovalores da matriz T_{30}

Fig. 1.3 – Autovalores da matriz T_{30}

O Lema 1.4.1 pode ser facilmente demonstrado por meio de identidades trigonométricas. Em conseqüência desse lema, tomando as matrizes $\mathbf{Z} := [\mathbf{z}_1 \quad \mathbf{z}_2 \quad . \quad . \quad . \quad \mathbf{z}_N]$ e $\Lambda := \mathrm{diag}(\lambda_1,\ldots,\lambda_N)$, podemos escrever $\mathbf{T}_N = \mathbf{Z}\Lambda\mathbf{Z}^t$. O maior autovalor é

$$\lambda_N = 2\left(1-\cos\frac{N\pi}{N+1}\right) \approx 4, \text{ se é } N \text{ é grande.}$$

O menor autovalor é λ_1. Para valores de i pequenos,

$$\lambda_i = 2\left(1-\cos\frac{i\pi}{N+1}\right) \approx 2\left(1-\left(1-\frac{i^2\pi^2}{2(N+1)^2}\right)\right) = \left(\frac{i\pi}{N+1}\right)^2.$$

Assim, T_N é positiva definida com número de condição

$$\frac{\lambda_N}{\lambda_1} \approx \frac{4(N+1)^2}{\pi^2}, \text{ para } N \text{ grande.}$$

A Fig. 1.3 apresenta geometricamente os autovalores de $\mathbf{T}_{30}$. Os autovetores são senoidais, com freqüências baixas para j próximo de 1, e altas para j próximo de N, como mostra a Fig. 1.4 (caso unidimensional $N = 30$).

Examinemos agora o caso bidimensional no que toca à determinação dos autovalores e autovetores da matriz dos coeficientes do sistema linear que discretiza a equação de Poisson, usando-o na forma (1.11). Ainda que nessa forma basta determinar os autovalores e autovetores da matriz subjacente **A**, do sistema escrito na forma usual **Ax** = **b**, porque "**Ax** = λ**x**" é o mesmo que "$\mathbf{T}_N\mathbf{V}+\mathbf{V}\mathbf{T}_N = \lambda\mathbf{V}$". Sejam, então, $(\lambda_i,\mathbf{z}_i)$ e $(\lambda_j,\mathbf{z}_j)$ dois autopares de $\mathbf{T}_N$, isto é, $\mathbf{T}_N\mathbf{z}_i = \lambda_i\mathbf{z}_i$ e $\mathbf{T}_N\mathbf{z}_j = \lambda_j\mathbf{z}_j$, e seja $\mathbf{V} = \mathbf{z}_i\mathbf{z}_j^t$. Então

$$\begin{aligned}
\mathbf{T}_N\mathbf{V}+\mathbf{V}\mathbf{T}_N &= (\mathbf{T}_N\mathbf{z}_i)\mathbf{z}_j^t + \mathbf{z}_i(\mathbf{z}_j^t\mathbf{T}_N) \\
&= (\lambda_i\mathbf{z}_i)\mathbf{z}_j^t + \mathbf{z}_i(\mathbf{z}_j^t\lambda_j) \\
&= (\lambda_i+\lambda_j)\mathbf{z}_i\mathbf{z}_j^t \\
&= (\lambda_i+\lambda_j)\mathbf{V}, \qquad (1.14)
\end{aligned}$$

de forma que $\mathbf{V} = \mathbf{z}_i \mathbf{z}_j^t$ é um autovetor associado ao autovalor $\lambda_i + \lambda_j$. Como $\mathbf{V}$ possui N^2 elementos, haverá N^2 autopares, um para cada par de autovalores λ_i e λ_j (arranjos com repetição de N^2 elementos 2 a 2), de $\mathbf{T}_N$. Em particular o menor autovalor é $2\lambda_1$ e o maior

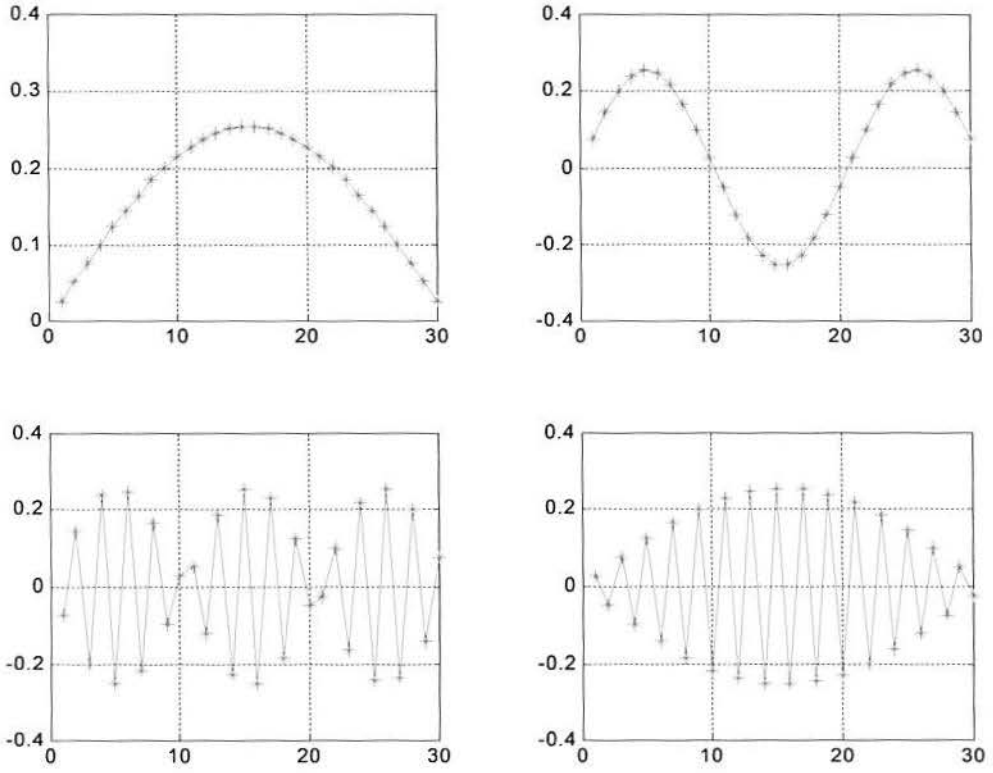

Fig. 1.4 – Autovetores de ordem 1, 3, 28 e 30 (ordem crescente dos autovalores associados) da matriz $\mathbf{T}_{30}$.

$2\lambda_N$, de maneira que o número de condição é o mesmo do caso unidimensional. Assim, como os autovalores e autovetores de $h^{-2}\mathbf{T}_N$ são boas aproximações para os autovalores e autofunções do operador diferencial de Poisson unidimensional (laplaciano), o mesmo acontece para o laplaciano bidimensional, cujos autovalores e autofunções são [13] revelados pela igualdade

$$\left(-\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2}\right)\text{sen}(i\pi x)\text{sen}(j\pi y) = (i^2\pi^2 + j^2\pi^2)\text{sen}(i\pi x)\text{sen}(j\pi y).$$

A Fig. 1.5 ilustra geometricamente o comportamento de dois autovetores do caso bidimensional $N = 10$, com as curvas de nível.

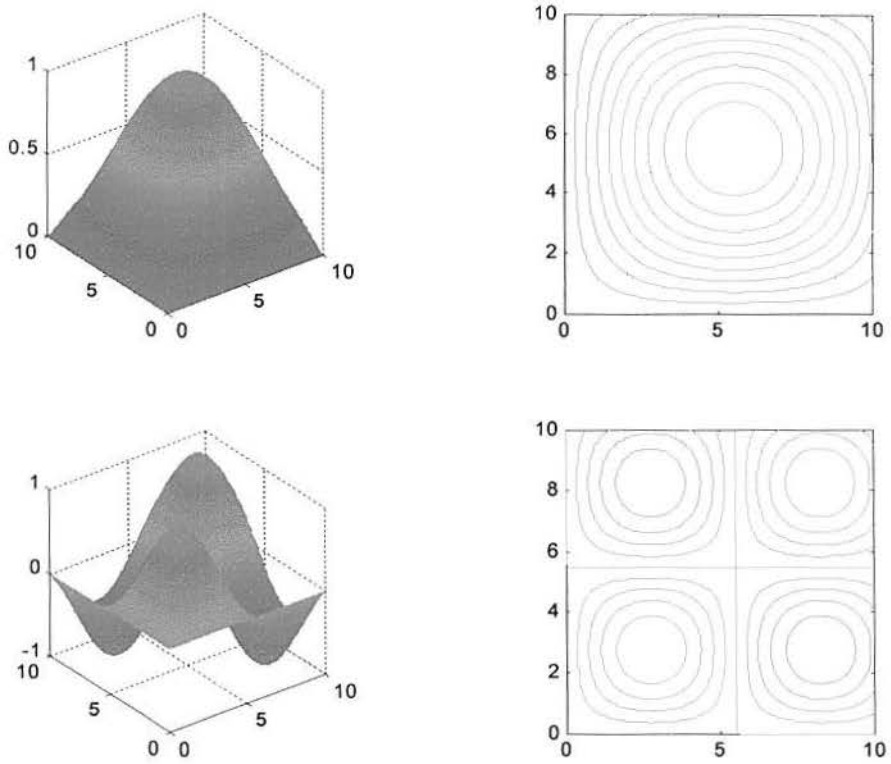

Fig. 1.5 – Dois autovetores de T_{10}^2.

1.5 Geração da matriz de Poisson por meio do produto de Kronecker

Apresentamos aqui uma maneira sistemática de deduzir as equações (1.12) e (1.13) bem como de calcular os autovalores e autovetores de $\mathbf{T}_{N^2}$.

1.5.1 Definição. *Seja* **X** *uma matriz de ordem m por n. Então* vec(**X**) *é um vetor coluna com m.n elementos, formado pelas colunas de* **X** *colocadas umas sobre as outras, tomadas da esquerda para a direita.*

Observemos que o vetor **v**, representado na Fig. 1.2, é o vetor **v** = vec(**V**).

1.5.2 Definição. *Seja* $\mathbf{A} = [\, a_{ij} \,]$ *uma matriz de ordem m por n e* $\mathbf{B} = [\, b_{ij} \,]$, *uma matriz de ordem p por q. O* produto de Kronecker *de* **A** *e* **B**, *nessa ordem, é a matriz de ordem*

mp por nq, denotada e definida por

$$\mathbf{A}\otimes\mathbf{B} := \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}.$$

1.5.3 Exemplo. Se

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{32} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix},$$

então

$$\mathbf{A}\otimes\mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & a_{13}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & a_{23}\mathbf{B} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{12}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{13}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \\ a_{21}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{22}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{23}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} & a_{13}b_{11} & a_{13}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} & a_{13}b_{21} & a_{13}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} & a_{23}b_{11} & a_{23}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} & a_{23}b_{21} & a_{23}b_{22} \end{bmatrix}. \square$$

O MATLAB realiza esse produto, também chamado *produto tensorial*, ou *produto direto*, através do comando `kron(A,B)`.

1.5.4 Lema. *São válidos os seguintes fatos sobre o produto de Kronecker.*

1. *Sejam os produtos* **AC** *e* **AB** *bem definidos. Então*

$$(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D}) = (\mathbf{AC})\otimes(\mathbf{BD}).$$

2. *Se* $\mathbf{A}$ *e* $\mathbf{B}$ são *inversíveis, então* $(\mathbf{A}\otimes\mathbf{B})^{-1} = \mathbf{A}^{-1}\otimes\mathbf{B}^{-1}$.

3. $(\mathbf{A}\otimes\mathbf{B})^{t} = \mathbf{A}^{t}\otimes\mathbf{B}^{t}$.

A prova do Lema 1.5.4 pode ser encontrada em [26]. O seguinte lema, demonstrado em [13], mostra-nos como escrever a equação de Poisson em termos do produto de Kronecker e do operador vec(.).

1.5.5 Lema. *Sejam* $\mathbf{A}$ e $\mathbf{B}$ *matrizes quadradas de ordem m e n, respectivamente, e* $\mathbf{X}$, *uma matriz de ordem por m. Vale o seguinte*:

1. $vec(\mathbf{AX}) = (\mathbf{I}_n\otimes\mathbf{A})\cdot vec(\mathbf{X})$;

2. $vec(\mathbf{XB}) = (\mathbf{B}^{t}\otimes\mathbf{I}_m)\cdot vec(\mathbf{X})$;

3. *a equação de Poisson discretizada,* $\mathbf{T}_N\mathbf{V} + \mathbf{V}\mathbf{T}_N = h^2\mathbf{F}$ *é equivalente à*

$$\mathbf{T}_{N^2}\,vec(\mathbf{V}) \equiv (\mathbf{I}_N\otimes\mathbf{T}_N + \mathbf{T}_N\otimes\mathbf{I}_N)\,vec(\mathbf{V}) = h^2\mathbf{F}. \tag{1.15}$$

Dessa forma, podemos confirmar que a expressão

$$\mathbf{T}_{N^2} = \mathbf{I}_N\otimes\mathbf{T}_N + \mathbf{T}_N\otimes\mathbf{I}_N$$

$$= \begin{bmatrix} \mathbf{T}_N & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \mathbf{T}_N \end{bmatrix} + \begin{bmatrix} 2\mathbf{I}_N & -\mathbf{I}_N & & \\ -\mathbf{I}_N & \ddots & \ddots & \\ & \ddots & \ddots & -\mathbf{I}_N \\ & & -\mathbf{I}_N & 2\mathbf{I}_N \end{bmatrix}$$

da igualdade (1.15) concorda com a igualdade (1.13).

Podemos calcular os autovalores das matrizes definidas pelo produto de Kronecker, como a matriz $\mathbf{T}_{N^2}$, com base no Lema 1.5.4 e na proposição a seguir.

1.5.6 Proposição *Seja* $\mathbf{T}_N = \mathbf{Z}\Lambda\mathbf{Z}^{t}$ *a decomposição espectral de* $\mathbf{T}_N$, *onde* $\mathbf{Z} := [\mathbf{z}_1 \ \ldots \ \mathbf{z}_N]$ *é a matriz ortogonal, cujas colunas são autovetores, e* $\Lambda = diag[\lambda_1 \ \ldots \ \lambda_N]$. *Então a decomposição espectral de* $\mathbf{T}_{N^2} = \mathbf{I}\otimes\mathbf{T}_N + \mathbf{T}_N\otimes\mathbf{I}$ *é*

$$\mathbf{I}\otimes\mathbf{T}_N+\mathbf{T}_N\otimes\mathbf{I}=(\mathbf{Z}\otimes\mathbf{Z})(\mathbf{I}\otimes\Lambda+\Lambda\otimes\mathbf{I})(\mathbf{Z}\otimes\mathbf{Z})^t. \qquad (1.16)$$

A matriz $\mathbf{I}\otimes\Lambda+\Lambda\otimes\mathbf{I}$ *é uma matriz diagonal, que tem como* $(N+j)^{ésimo}$ *elemento diagonal o* $(i, j)^{ésimo}$ *autovalor de* $\mathbf{T}_{N^2}$, *sendo este autovalor* $\lambda_{ij}=\lambda_i+\lambda_j$. *A matriz* $\mathbf{Z}\otimes\mathbf{Z}$ *é uma matriz ortogonal, cuja* $(iN+j)^{ésima}$ *coluna, o autovetor associado a* λ_{ij}, *é* $\mathbf{z}_i\otimes\mathbf{z}_j$.

Prova. Das partes 1 e 3 do Lema 1.5.4, temos que $\mathbf{Z}\otimes\mathbf{Z}$ é ortogonal, uma vez que $(\mathbf{Z}\otimes\mathbf{Z})(\mathbf{Z}\otimes\mathbf{Z})^t=(\mathbf{Z}\otimes\mathbf{Z})(\mathbf{Z}^t\otimes\mathbf{Z}^t)=(\mathbf{Z}\mathbf{Z}^t)\otimes(\mathbf{Z}\mathbf{Z}^t)=\mathbf{I}\otimes\mathbf{I}=\mathbf{I}$. Podemos, então, verificar a equação (1.16):

$$\begin{aligned}
&(\mathbf{Z}\otimes\mathbf{Z})(\mathbf{I}\otimes\Lambda+\Lambda\otimes\mathbf{I})(\mathbf{Z}\otimes\mathbf{Z})^t\\
&=(\mathbf{Z}\otimes\mathbf{Z})(\mathbf{I}\otimes\Lambda+\Lambda\otimes\mathbf{I})(\mathbf{Z}^t\otimes\mathbf{Z}^t)\\
&=(\mathbf{Z}\mathbf{I}\mathbf{Z}^t)\otimes(\mathbf{Z}\Lambda\mathbf{Z}^t)+(\mathbf{Z}\Lambda\mathbf{Z}^t)\otimes(\mathbf{Z}\mathbf{I}\mathbf{Z}^t)\\
&=\mathbf{I}\otimes\mathbf{T}_N+\mathbf{T}_N\otimes\mathbf{I}\\
&=\mathbf{T}_{N^2}.
\end{aligned}$$

A segunda igualdade resulta da parte 3 do Lema 1.6.4 e a penúltima, da parte 1 do mesmo lema. Também podemos verificar que $\mathbf{I}\otimes\Lambda+\Lambda\otimes\mathbf{I}$ é diagonal, com elementos diagonais na posição $(iN+j)$, dado por $\lambda_{ij}=\lambda_i+\lambda_j$, de forma que a equação (1.16) é realmente a decomposição espectral de $\mathbf{T}_{N^2}$. Finalmente, da definição do produto de Kronecker, podemos ver que a coluna $iN+j$ de $\mathbf{Z}\otimes\mathbf{Z}$ é $\mathbf{z}_i\otimes\mathbf{z}_j$. □

Confirmamos, então, a igualdade $\mathbf{z}_i\otimes\mathbf{z}_j=vec(\mathbf{z}_j\mathbf{z}_i^t)$, onde $\mathbf{z}_j\mathbf{z}_i^t$ é a expressão usada para um autovetor em (1.14).

De maneira similar, a equação de Poisson tridimensional conduz a

$$\mathbf{T}_{N^3}\equiv\mathbf{T}_N\otimes\mathbf{I}_N\otimes\mathbf{I}_N+\mathbf{I}_N\otimes\mathbf{T}_N\otimes\mathbf{I}_N+\mathbf{I}_N\otimes\mathbf{I}_N\otimes\mathbf{T}_N,$$

cujos autovalores são todas as possíveis somas de três autovalores de $\mathbf{T}_N$, e os autovetores são as colunas de $\mathbf{Z}\otimes\mathbf{Z}\otimes\mathbf{Z}$. Para dimensões maiores da equação de Poisson a representação é análoga.

1.6 Precondicionamento

A palavra precondicionamento originou-se com Turing, em 1948, e algumas das primeiras contribuições no contexto de iterações matriciais são devidas a Hestens, Engel, Wacgpress, Evans e Axelsson [25,40]. A idéia tornou-se famosa nos anos 70 com a introdução da fatoração incompleta por Mujirink e Van de Vorst [25].

Na Seção 1.3 vimos que a convergência do MGC depende do número de condição de **A**, ou, mais exatamente, da distribuição de seus autovalores. Conforme explicamos, um número de condição pequeno (próximo de 1) é desejável. Se o número de condição é grande, para melhorar o condicionamento de **A**, isto é, para *precondicionar* **A**, devemos substituir o sistema linear $\mathbf{Ax} = \mathbf{y}$ por um sistema $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{y}$, onde **M** é uma aproximação de **A** tal que

1. **M** é SPD,
2. $\mathbf{M}^{-1}\mathbf{A}$ é mais bem-condicionada que **A** ou possui poucos autovalores distintos,
3. $\mathbf{Mx} = \mathbf{y}$ é fácil de resolver.

Uma escolha criteriosa de **M** pode tornar o número de condição de $\mathbf{M}^{-1}\mathbf{A}$ muito menor do que o número de condição de **A** e assim acelerar grandemente a convergência. De fato, um bom precondicionador é freqüentemente necessário para que haja convergência de um método iterativo. Como conseqüência desse fato, muita pesquisa atual sobre métodos iterativos está direcionada à procura de melhores precondicionadores.

O objetivo principal de nosso trabalho é exatamente apresentar uma matriz precondicionadora, baseada em wavelets, para a equação de Poisson discretizada.

Wavelets constituem uma das mais importantes novas técnicas matemáticas para descrever funções e analisar dados contínuos, derivados dos mais diferentes tipos de sinais. Publicações sobre o assunto exibem uma grande variedade de aplicações da teoria de wavelets. Diversos estudos sobre wavelets demonstram sua aplicabilidade na análise numérica de problemas relacionados às equações diferenciais. Algumas das principais referências para esse ponto são [02, 03, 04, 05, 10, 21, 23, 36, 37]. Esses estudos apresentam a construção de eficientes precondicionadores para sistemas lineares com base na transformada wavelet discreta (TWD).

2. TRANSFORMADA WAVELET E ANÁLISE MULTI-RESOLUÇÃO

Neste capítulo abordamos wavelets e análise multi-resolução (AMR), tópico central da teoria de wavelets, necessária ao desenvolvimento deste trabalho. O capítulo está organizado da seguinte forma: inicialmente apresentamos um breve histórico sobre a teoria de wavelets e após introduzimos a transformada wavelet contínua; em seguida, discutimos algumas propriedades e definimos a função wavelet a partir do conceito de AMR. Também abordamos a TWD e as relações com bancos de filtros, bem como wavelets bidimensionais.

2.1 Breve histórico de wavelets

Historicamente, a teoria de wavelets apresenta muitas origens diferentes. Antes de 1930, o principal ramo da matemática que levou às wavelets teve início com Joseph Fourier (1807) em sua teoria sobre análise de freqüências, hoje referida como síntese de Fourier. Fourier afirmou (com palavras diferentes) que qualquer função f com período 2π é a soma

$$a_0 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \operatorname{sen} kx)$$

de sua série de Fourier. Os coeficientes a_0, a_k e b_k são calculados por

$$a_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)dx,\ \ a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(kx)dx \ \text{ e } \ b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\text{sen}(kx)dx.$$

A afirmação de Fourier teve um papel essencial na evolução das idéias que os matemáticos tinham sobre funções, abrindo a porta para um novo universo funcional. Após 1807, explorando o significado das funções, a convergência das séries de Fourier e sistemas ortogonais, os matemáticos foram, gradativamente, levados da noção inicial de análise de freqüências para a noção de análise de escalas.

A primeira menção sobre wavelets apareceu no apêndice da tese de A. Haar, em 1909. Uma propriedade das wavelets de Haar é ter suporte compacto, o que significa que desaparecem fora de um intervalo limitado. Infelizmente, as wavelets de Haar não são continuamente diferenciáveis, não são sequer diferenciáveis, o que limita suas aplicações.

A formalização da teoria de wavelets ocorreu recentemente, na década de 80, com base na generalização de conceitos já conhecidos, oriundos de diversos campos de pesquisa, como geofisica, análise e compressão de sinais, física e matemática. Desde então, a teoria de wavelets tem atraído a atenção de diversos pesquisadores, encontrando aplicações em diferentes áreas, particularmente em processamento computacional de imagens e visão.

O termo wavelets foi introduzido por J. Morlet, e a base matemática de suas idéias foi formalizada pelo físico teórico Alex Grossmann [01, 12, 20]. Os dados sísmicos estudados por Morlet exibiam conteúdos de freqüência que mudavam rapidamente com o tempo, para os quais a transformada de Fourier (TF) não era adequada como ferramenta de análise, pois essa não permite uma análise local do conteúdo de freqüência do sinal. Eventos que venham a ocorrer em intervalos de tempo distintos, e mesmo bastante remotos entre si, contribuem de maneira global para a transformada, afetando a representação como um todo. Wavelets têm demonstrado ser uma ferramenta potente para a representação e aproximação de funções e distribuições de Laurent. Bases de wavelets são particularmente bem adaptadas para aplicação na solução numérica de equações diferenciais parciais, especialmente quando as soluções apresentam singularidades, sendo assim uma alternativa para o clássico método de Fourier. Pesquisas recentes têm demonstrado que, em muitos casos, wavelets podem levar a algoritmos mais eficientes [03,04,21,36,37].

2.2 Transformada wavelet contínua

A transformada wavelet contínua (TWC) de uma função $f \in L^2(\mathbb{R})$ é definida por

$$W_{s\tau}(f(x)) = \int f(x)\overline{\psi}_{s\tau}(x)dx, \tag{2.1}$$

onde a barra denota complexo conjugado e as funções $\psi_{s\tau}$ são definidas abaixo. Entretanto, a maioria das wavelets é de valores reais. A transformada wavelet TW usa tipicamente translações e dilatações de uma função fixa $\psi \in L^2(\mathbb{R})$, a wavelet-mãe, ou simplesmente wavelet. No caso da TWC, os parâmetros de translação e dilatação variam continuamente. Especificamente, a transformada usa as funções $\psi_{s\tau}$, definidas por

$$\psi_{s\tau}(x) = \frac{1}{\sqrt{|s|}}\psi\left(\frac{x-\tau}{s}\right), \text{ com } s,\tau \in \mathbb{R},\ s \neq 0, \tag{2.2}$$

onde s é o fator de escala, τ é o fator de translação e $|s|^{-1/2}$ é o fator de normalização, cujo efeito é produzir em L^2 uma norma constante para essas funções. É importante observar que a função wavelet em (2.2) não é especificada. Essa é uma importante diferença entre a TW e a TF ou outras transformadas. A teoria da TW lida com propriedades gerais das wavelets e da TW, definindo uma estrutura na qual podemos construir bases wavelets com características específicas para cada aplicação.

2.2.1 Propriedades

2.2.1.1 Condição de admissibilidade. Podemos mostrar [30] que qualquer função ψ que satisfaz a *condição de admissibilidade*

$$C_\psi := \int_{-\infty}^{+\infty} \frac{|\hat{\psi}(\xi)|^2}{\xi} d\xi < \infty, \tag{2.3}$$

pode ser usada para decompor e, então, reconstruir uma função, sem que nenhuma informação seja perdida durante a transformação ((2.3) é o requisito para que a *resolução da identidade*, que mostra a conservação da energia nos domínios tempo e tempo-escala seja satisfeita). Essa condição implica que a TF $\hat{\psi}$ de ψ degenera para freqüência nula, isto é,

$$|\hat{\psi}(0)|^2 = 0. \tag{2.4}$$

Se $\hat{\psi}$ é contínua, então $\hat{\psi}(0) = 0$, isto é, o valor médio da wavelet no domínio temporal também deve ser zero,

$$\int_{-\infty}^{+\infty} \psi(x)dx = 0. \tag{2.5}$$

Geometricamente, essa condição estabelece que o gráfico da função ψ deve oscilar, para cancelar os valores positivos e negativos das áreas, de forma que a integral em (2.5) resulte nula.

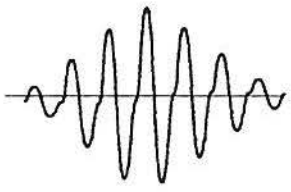

Fig. 2.1 – Exemplo do gráfico de uma wavelet

Da conservação de energia, dada pela condição de admissibilidade, temos que a TWC $W(s,\tau)$ é inversível, e a transformada inversa f é definida por

$$f(x) = \frac{1}{C_\psi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} W(s,\tau)\psi_{s\tau}(x) \frac{ds\, d\tau}{s^2}. \tag{2.6}$$

2.2.1.2 Condições de regularidade. A TW de uma função unidimensional é bidimensional; a TW de uma função bidimensional é quadridimensional. Conseqüentemente, temos uma alta densidade da *sampling*[1] no espaço tempo-escala (*redundância*). Com o objetivo de que o número de coeficientes da TW diminua rapidamente com o aumento de $1/s$, são impostas algumas *condições de regularidade* sobre a função ψ: a função ψ e sua TW devem ter alguma suavidade e boa localização em ambos os domínios de tempo e freqüência. Sabemos [30] que uma rápido decréscimo do número de coeficientes da TW corresponde à localização de $\hat{\psi}$ no domínio das freqüências. Para examinarmos esse fato, tomemos $\tau = 0$ e consideremos a convergência da seqüência dos coeficientes da TW para zero com o crescimento de $1/s$.

Se $f(x)$ foi expandida em termos da série de Taylor em $t = 0$ até a ordem n, teremos

$$W(s,0) = \frac{1}{\sqrt{s}}\left[\sum_{p=0}^{n} f^{(p)}(0)\int\frac{x^p}{p!}\psi\left(\frac{x}{s}\right)dx + \mathrm{O}(n+1)\right], \tag{2.7}$$

onde $\mathrm{O}(n+1)$ é o resto da expansão. Denotamos e definimos os momentos de ordem p da wavelet por

$$M_p = \int x^p \psi(x)dx. \tag{2.8}$$

Temos então o desenvolvimento finito de (2.7),

$$W(s,0) = \frac{1}{\sqrt{s}}\left[f(0)M_0 s + \frac{f'(0)}{1!}M_1 s^2 + \frac{f''(0)}{2!}M_2 s^3 + \ldots + \frac{f^{(n)}(0)}{n!}M_n s^{n+1} + O(s^{n+2})\right]. \tag{2.9}$$

[1] Não traduzimos a palavra *sample* e sua cognatas, como *sampling*, *downsampling*, etc.

De acordo com a condição de admissibilidade, temos que $M_0 = 0$, e por isso o primeiro termo da expansão dentro do colchete em (2.9) é zero. Assim a taxa de decaimento dos coeficientes da TW com o crescimento de $1/s$ é determinada pelo primeiro momento não-nulo da wavelet básica. Se os primeiros $n + 1$ momentos, ou seja, os momentos até a ordem n, são nulos,

$$M_p = \int x^p \psi(x) dx = 0, \quad p = 0, 1, 2, \ldots, n, \tag{2.10}$$

vem de (2.9) que o número de coeficientes da TW decai com a mesma rapidez que s^{n+2}, para uma função f suave. Então, a regularidade da wavelet leva à localização da TW no domínio da freqüência. Na literatura, isso é conhecido como *número de momentos nulos* ou *ordem de aproximação*. Em outras palavras, dizer que uma função wavelet possui n momentos nulos (os n primeiros) é o mesmo que dizer que a ordem de aproximação da TW é n.

A *ordem de regularidade* de uma wavelet é o número de suas derivadas contínuas; para ter regularidade maior que n, uma wavelet deve ter pelo menos $n + 1$ momentos nulos [20].

O efeito prático dos momentos nulos é concentrar a informação de uma função em um relativamente pequeno número de coeficientes. Conforme demonstrado por Daubechies [11], se uma wavelet possui M momentos nulos, então seu suporte deve ser, pelo menos, o intervalo [0 ; $2M$-1] e, aumentando o tamanho do suporte, aumenta o cálculo. O número de momentos nulos adequado depende da aplicação.

Entre as wavelets ortogonais, as wavelets de Daubechies são as que produzem o menor suporte possível para um dado número de momentos nulos. Uma relação ainda melhor entre o tamanho do suporte e o número de momentos nulos pode ser obtida com multiwavelets [20].

Em nosso estudo no Capítulo 3 utilizamos wavelets de Daubechies com até 3 momentos nulos, com o que obtemos os melhores resultados.

Como conseqüência da condição de admissibilidade, o gráfico de uma ψ deve ter a forma de uma onda, em inglês, *wave*. Das condições de regularidade temos o rápido decaimento do número de coeficientes não-nulos, daí o sufixo *let* do diminutivo. Essas características justificam o nome wavelet (pequena onda, em francês, *ondellete*).

2.3 Transformada wavelet discreta e frames

Conforme comentamos na seção anterior, para a maioria das aplicações, o objetivo é representar a função eficientemente com um mínimo de coeficientes possível. Então é interessante encontrar transformadas inversas que não utilizem $W(s,\tau)$ sobre toda a extensão do campo de variação de s e τ. Usando a teoria de *frames* [06], é possível estudar o caso onde somente valores discretos de s e τ são considerados. O uso de wavelets discretas reduz o produto tempo-escala (que produz no número de coeficientes) resultante da transformada wavelet.

Obtemos uma wavelet discreta tomando a wavelet contínua com valores discretos para os parâmetros de mudança de escala (dilatação) s e de translação τ, que podem ser expressos como $s = s_0^j$ e $\tau = k\tau_0 s_0^j$, onde i e k são inteiros e $s_0 > 1$ é um passo fixo de dilatação. O fator de translação $\tau = k\tau_0 s_0^j$ depende do passo de dilatação s_0^j. A wavelet discreta correspondente a (2.2) do caso contínuo é dada por

$$\begin{aligned}\psi_{jk}(x) &= s_0^{-j/2}\psi\left(s_0^{-j}(x - k\tau_0 s_0^j)\right) \\ &= s_0^{-j/2}\psi\left(s_0^{-j}x - k\tau_0\right).\end{aligned} \tag{2.11}$$

O comportamento das wavelets discretas depende dos parâmetros s_0 e τ_0. Quando o valor de s_0 é próximo de 1 e o valor de τ_0 é pequeno, a wavelet discreta é aproximadamente a wavelet contínua. A escolha mais comum é usar uma grade diádica: $s_0 = 2$ e $\tau_0 = 1$. Nesse caso, a igualdade (2.11) fica

$$\psi_{jk}(x) = 2^{-j/2}\psi(2^{-j}x - k). \tag{2.12}$$

Originalmente, é a transformada que usa somente valores diádicos para s e τ que foi denominada *transformada wavelet discreta*. Hoje, entretanto, esse termo é ambíguo, uma vez que também é usado para designar a operação, e o próprio resultado, que transforma a seqüência dos coeficientes da função de escala em coeficientes da wavelet.

2.3.1 Frames

A teoria de frames provê uma linguagem para descrever a completude, a estabilidade e a redundância de todo tipo de transformadas discretas – incluindo transformadas ortogonais e biortogonais como casos especiais. Foi originalmente desenvolvida em 1952 por Duffin e Schaeffer; mais recentemente, outros pesquisadores, notadamente Ingrid Daubechies, têm mostrado como essa teoria se aplica às wavelets [20].

Com base de wavelets discretas, uma função contínua f é decomposta em uma seqüência de coeficientes de wavelet

$$\int f(x)\psi_{jk}(x)dx = \langle f, \psi_{jk} \rangle. \tag{2.13}$$

Em (2.13) o símbolo $\langle \cdot, \cdot \rangle$ indica produto interno em L^2. Uma questão importante é saber quão bem uma função f pode ser reconstruída a partir dos coeficientes da wavelet discreta:

$$f(x) = A \sum_j \sum_k \langle f, \psi_{jk} \rangle \psi_{jk}(x), \tag{2.14}$$

onde A é uma constante que não depende de f. Como já comentamos, se o valor de s_0 é próximo de 1 e o valor de τ_0 é suficientemente pequeno, a wavelet discreta aproxima a wavelet contínua. A reconstrução é obtida, aproximadamente, pela Transformada Wavelet Inversa (TWI). Nesse caso, a reconstrução da função não apresenta nenhuma outra restrição, além da condição de admissibilidade da wavelet. Por outro lado, se a *sampling* é esparsa, como no caso da grade diádica, a reconstrução somente é obtida para escolhas especiais da wavelet.

A teoria de *frames* provê uma estrutura geral que cobre essas duas situações extremas. Ela permite um equilíbrio entre a *redundância* e as restrições sobre a wavelet, para trabalhar com esquemas de reconstrução. Se desejamos redundância grande, então são impostas restrições brandas sobre a wavelet básica. Se desejamos redundância pequena, então a base de funções wavelet é muito restrita.

Daubechies [11] mostra que a condição necessária e suficiente para uma reconstrução estável de uma função f a partir de seus coeficientes wavelet $\langle f, \psi_{j,k} \rangle$ é

que

$$A\|f\|^2 \leq \sum_{j,k\in\mathbb{Z}} \left|\langle f, \psi_{jk}\rangle\right|^2 \leq B\|f\|^2 .$$

O conjunto $\{\psi_{jk} : j,k \in \mathbb{Z}\}$ constitui uma *frame*. As constantes $A > 0$ e $B < +\infty$ são chamadas limitantes da *frame* e $\|f\|^2$ é a energia de f. A exatidão da reconstrução é controlada pelos limitantes da *frame* que, conforme demonstrado por Daubechies [11], podem ser calculados a partir de s_0 e τ_0 e da função básica ψ.

2.3.1.1 Proposição. *Num espaço de Hilbert, se* $\{\psi_{jk} : j, k \in \mathbb{Z}\}$ *é uma frame, e as funções* ψ_{jk} *são linearmente independentes, então* $\{\psi_{jk} : j, k \in \mathbb{Z}\}$ *constitui uma base de Riesz* [32]

2.3.1.2 Proposição. *Se* $\{\psi_{jk} : j, k \in \mathbb{Z}\}$ *é uma base de Riesz, com* $A = B = 1$ *e se* $\|\psi_{jk}\| = 1$ *para todo* $j, k \in \mathbb{Z}$, *então* $\{\psi_{jk} : j \in \mathbb{Z}\}$ *é uma base ortonormal* [11].

Com wavelets ortonormais conseguimos eliminar completamente a redundância da TWC. Apesar disso, a ortogonalidade não é essencial na representação de funções. Em algumas aplicações a redundância é desejável, pois pode ajudar a reduzir a sensibilidade ao ruído, ou melhorar a invariância por translação da transformada. Esta é uma das desvantagens das wavelets discretas: a transformada wavelet de um sinal e a transformada wavelet de uma versão da translação no tempo desse sinal não são simplesmente versões transladadas de cada uma.

2.4 Análise multi-resolução

Nossa percepção do mundo é feita em diferentes escalas, apropriadas a cada tipo de observação. A escala deve ser adequada para entendermos os diferentes detalhes que precisamos. Por exemplo, quando precisamos representar um objeto, tentamos usar

uma escala em que os detalhes importantes possam ser descritos na representação. Os mapas são exemplos típicos do uso de escalas. Com escalas pequenas podemos observar apenas características macroscópicas das regiões mapeadas. Escalas maiores apresentam mais detalhes, melhor definição dos contornos das regiões representadas. A representação multi-resolução é um modelo matemático adequado para representar o universo físico em escala.

Oriundos de diferentes áreas, matemática pura e visão computacional, Stéphane Mallat e Yves Meyer desenvolveram a teoria da multi-resolulção. Essa teoria popularizou a idéia de analisar imagens em diferentes níveis de resolução, onde cada resolução é diferenciada por um fator 2 daquela do nível imediatamente anterior. Outro resultado dessa teoria é a teoria matemática de wavelets ortogonais. Nessa abordagem, a wavelet é criada por outra função, a função de escala, que fornece uma série de representações da função. Numa direção, essas sucessivas representações aproximam a função original com precisão cada vez melhor. Na outra direção, aproximam o zero, contendo cada vez menos informação.

Uma AMR de $L^2(\mathbb{R})$ é definida como uma seqüência de subespaços encaixados $V_j \subset L^2(\mathbb{R})$ que satisfaz as seguintes propriedades:

I. $V_j \subset V_{j-1}$;

II. $\overline{\bigcup_{j\in\mathbf{Z}} V_j} = L^2(\mathbb{R})$;

III. $\bigcap_{j\in\mathbf{Z}} V_j = \{0\}$;

IV. $f(x) \in V_j \Leftrightarrow f(2^j x) \in V_0$;

V. $f(x) \in V_0 \Leftrightarrow f(x-k) \in V_0$;

VI. Existe uma função de escala $\varphi \in V_0$ tal que o conjunto $\{\varphi_{0k} : k \in \mathbb{Z}\}$ de funções forma uma base ortonormal de V_0.

Seguem algumas simples observações a respeito dessa definição de AMR. Definimos P_{V_j} como sendo o operador projeção ortogonal sobre V_j. Então, das condições I e II temos, para toda $f \in L^2(\mathbb{R})$,

$$\lim_{j \to -\infty} P_j f = f,$$

isto é, à medida que a escala se torna mais fina, obtemos uma melhor representação da função f. Na Fig. 2.2 visualizamos uma função f e suas aproximações nos espaços de escala V_{-1}, V_0 e V_1 da representação multi-resolução de Haar.

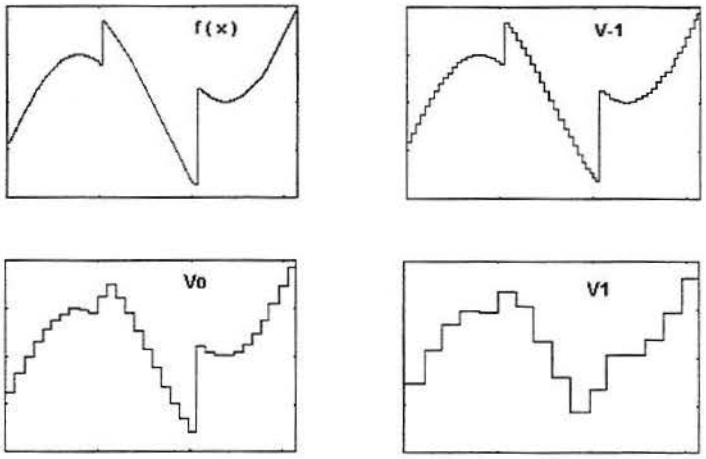

Fig. 2.2 – Aproximações de uma função em escalas diferentes

Por I os detalhes de uma função, que aparecem na escala 2^j, devem aparecer certamente quando representamos essa função usando uma escala mais fina 2^{j-1}. Dada uma função $f \in L^2(\mathbb{R})$, um requisito natural é que $f(x) \in V_j \Leftrightarrow f(2x) \in V_{j-1}$. De fato, mudando a escala por um fator 2, o suporte de f é reduzido por um fator ½ (cf. Fig.2.3). Aplicando sucessivamente esse procedimento, obtemos o requisito IV da AMR, que implica que todos os espaços são versões em diferentes escalas do espaço V_0. A condição V mostra a invariância de V_0 sob translações. Essa condição, junto com a condição IV, implica que $f \in V_j \Leftrightarrow f(x - 2^j k) \in V_j$, para todo $j \in \mathbb{Z}$.

2.4.1 A função de escala. Uma vez que $\varphi \in V_0 \subset V_{-1}$, e a seqüência $\left(\varphi_{-1,n}\right)$ forma uma base ortonormal em V_{-1}, então

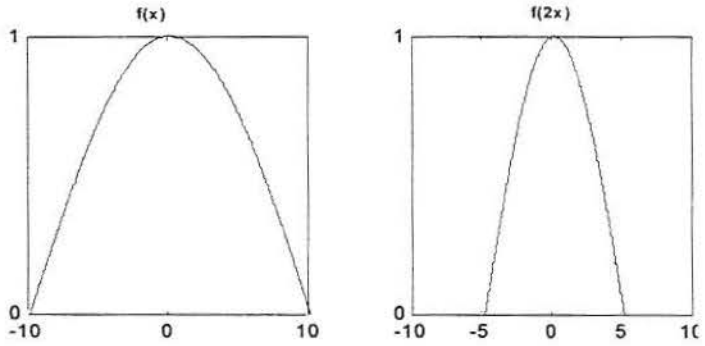

Fig. 2.3 – Mudança de escala por um fator 2

$$\varphi = \sum_n h_n \varphi_{-1,n}, \tag{2.15}$$

com

$$h_n = \left\langle \varphi, \varphi_{-1,n} \right\rangle \tag{2.16}$$

e

$$\sum_{n \in \mathbb{Z}} |h_n|^2 = 1, \tag{2.17}$$

que pode também ser escrita na forma

$$\varphi(x) = \sqrt{2} \sum_n h_n \varphi(2x - n), \tag{2.18}$$

ou

$$\hat{\varphi}(\xi) = \frac{1}{\sqrt{2}} \sum_n h_n e^{-in\xi/2} \hat{\varphi}(\xi/2). \tag{2.19}$$

A convergência das duas últimas séries é no sentido do L^2.

A formula (2.19) pode ser escrita assim:

$$\hat{\varphi}(\xi) = m_0(\xi/2)\hat{\varphi}(\xi/2), \tag{2.20}$$

onde

$$m_0(\xi) = \frac{1}{\sqrt{2}} \sum_n h_n e^{-in\xi}. \tag{2.21}$$

Em (2.20) a igualdade se verifica ponto a ponto quase em toda parte. Não é difícil ver que a m_0 é uma função periódica com período 2π.

A (2.18) é conhecida por diversos nomes: *equação de refinamento*, *equação de dilatação* e *equação a diferenças de escala dupla*. Usaremos a primeira designação.

Segue de IV e VI que $\{\varphi_{jk} : k \in \mathbb{Z}\}$, com

$$\varphi_{jk}(x) = 2^{-j/2} \varphi(2^{-j}x - k), \tag{2.22}$$

forma uma base ortonormal de V_j, para todo $j \in \mathbb{Z}$. Contudo essa condição pode ser relaxada: $\{\varphi_{jk} : k \in \mathbb{Z}\}$ forma uma base de Riesz de V_j, para todo $j \in \mathbb{Z}$, o que é uma condição menos restritiva.

A função de escala é, sob condições bem gerais, univocamente definida por sua equação de refinamento com a normalização

$$\int_{-\infty}^{+\infty} \varphi(x)dx = 1.$$

Em muitos casos, não temos uma expressão explícita para φ. Entretanto, existem algoritmos rápidos que usam a equação de refinamento para avaliar a função

de escala φ nos pontos diádicos $x = 2^{-j}k,\ j,k \in \mathbb{Z}$. Em muitas aplicações não há necessidade da função de escala; trabalhamos diretamente com a seqüência (h_k).

Os espaços V_j são usados para aproximar funções gerais. Isso é feito pela definição de projeções apropriadas sobre esses espaços. Uma vez que a união de todos os V_j é densa em $L^2(\mathbb{R})$, qualquer função de L^2 pode ser aproximada por tais projeções.

Para que possamos aproximar desde as funções mais simples por meio do conjunto $\{\varphi(x-k) : k \in \mathbb{Z}\}$, é natural supor que a função de escala e suas translações formem uma partição da unidade, ou, em outras palavras,

$$\forall x \in \mathbb{R},\ \sum_k \varphi(x-k) = 1.$$

Observemos que, pela fórmula de somação de Poisson [32]

$$\frac{1}{2\pi}\sum_{k=-\infty}^{\infty} G(2\pi k) = \sum_{k=-\infty}^{\infty} \widehat{G}(n) \tag{2.23}$$

a partição da unidade é equivalente a

$$\hat{\varphi}(2\pi k) = \delta_k,\ k \in \mathbb{Z}, \tag{2.24}$$

onde δ é o delta de Kronecker.

2.4.2 A função wavelet. Se as condições da AMR são satisfeitas, então existe uma base de wavelets ortonormal $\{\psi_{jk} : j,k \in \mathbb{Z}\}$ de $L^2(\mathbb{R})$, sendo

$$\psi_{jk}(x) = 2^{-j/2}\psi(2^{-j}x - k), \tag{2.25}$$

que pode ser construída por meio da AMR. Para todo $j \in \mathbb{Z}$, definimos W_j como sendo o complemento ortogonal de V_j em V_{j-1}, isto é,

$$V_{j-1} = V_j \oplus W_j. \tag{2.26}$$

O espaço W_j contém os detalhes de informação necessários para ir de uma aproximação de resolução j a uma aproximação de resolução j-1. É imediato verificar que W_j é ortogonal a W_k, se $j \neq k$. Conseqüentemente, fixando $J \in \mathbb{Z}$, $j < J$, temos

$$V_j = V_J \oplus \bigoplus_{k=0}^{J-j-1} W_{J-k}.$$

Todos os espaços envolvidos nessa soma são ortogonais e segue de II e III da definição da AMR que

$$L^2(\mathbb{R}) = \bigoplus_{j \in \mathbb{Z}} W_j, \tag{2.27}$$

isto é, obtemos uma decomposição de $L^2(\mathbb{R})$ em espaços mutuamente ortogonais.

O seguinte teorema, demonstrado em [11], mostra que é possível construir ψ por meio da AMR.

2.4.1 Teorema. *Se uma seqüência de subespaços $(V_j)_{j \in \mathbb{Z}}$ em $L^2(\mathbb{R})$ satisfaz as condições I – VI, então existe uma base ortonormal $\{\psi_{j,k} : j,k \in \mathbb{Z}\}$, de wavelets associada, de $L^2(\mathbb{R})$ tal que, para todo $f \in L^2(\mathbb{R})$,*

$$P_{j-1} = P_j + \sum_k \langle f, \psi_{jk} \rangle \psi_{jk}, \tag{2.28}$$

onde P_j é a projeção ortogonal sobre V_j. Uma possibilidade para a construção da wavelet ψ é

$$\hat{\psi}(\xi) = e^{i\xi/2} \overline{m_0(\xi/2 + \pi)} \hat{\varphi}(\xi/2), \tag{2.29}$$

com m_0 definido em (2.21). *Ou, de forma equivalente,*

$$\psi = \sum_n (-1)^{n-1} \overline{h_{-n-1}} \varphi_{-1,n}, \tag{2.30}$$

$$\psi(x) = \sqrt{2}\sum_n (-1)^{n-1}\overline{h_{-n-1}}\varphi(2x-n), \tag{2.31}$$

com convergência da última série no sentido de L^2.

Encontramos também (2.30) escrita na forma

$$\psi = \sum_n g_n \varphi_{-1,n}, \quad \text{com} \quad g_n = (-1)^n \overline{h_{-n+1}}, \text{ ou } g_n = (-1)^n h_{-n+1+2N},$$

para uma escolha apropriada de $N \in \mathbb{Z}$.

2.5 Classes de wavelets

2.5.1 Wavelets ortogonais. Na seção anterior definimos, a partir da AMR, as funções φ e ψ como sendo ortogonais. Wavelets ortogonais possuem a vantagem da reconstrução perfeita e, no caso da TWD rápida temos uma transformação unitária (sua transposta (adjunta) é igual à sua inversa) e, conseqüentemente, seu número de condição é igual a 1, o que indica estabilidade numérica [23].

Exemplos de wavelets ortogonais são a família de wavelets construídas por Daubechies [11] e a família de Coifflets (Daubechies-Coiffman). As ilustrações nas Fig. 2.4 e 2.5 foram geradas no MATLAB por meio do comando `wavefun`.

2.5.2 Wavelets biortogonais. Wavelets biortogonais constituem uma generalização de wavelets ortogonais. No contexto da AMR ortogonal, definimos os operadores projeção sobre os subespaços V_j e W_j, respectivamente, por

$$P_{V_j} f := \underbrace{\sum_k \overbrace{\langle f, \varphi_{jk} \rangle}^{\text{análise}} \varphi_{jk}}_{\text{síntese}} \quad \text{e} \quad P_{W_j} f := \underbrace{\sum_k \overbrace{\langle f, \psi_{jk} \rangle}^{\text{análise}} \psi_{jk}}_{\text{síntese}},$$

onde as funções φ e ψ desempenham uma dupla tarefa, isto é, elas são usadas para:

a) **análise:** cálculo dos coeficientes da representação de f em termos das bases,

geradas por φ e ψ, dos espaços V_j e W_j, isto é, $c_j^k = \langle f, \varphi_{jk} \rangle$ e $d_k^j = \langle f, \psi_{jk} \rangle$;

b) **síntese:** reconstrução da projeção de f sobre V_j e W_j, a partir dos coeficientes da representação, isto é, $P_{V_j} f = \sum_k c_k^j \varphi_{jk}$ e $P_{W_j} f = \sum_k d_k^j \psi_{jk}$.

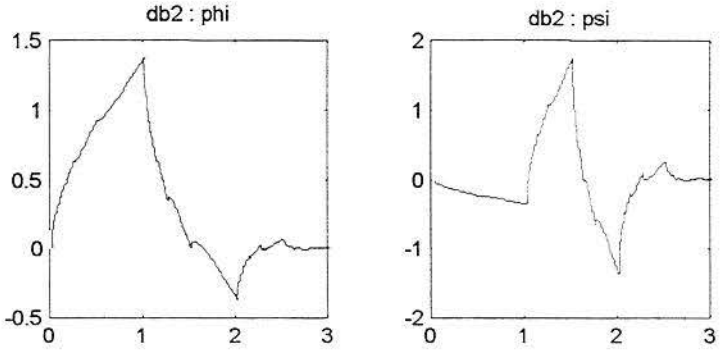

Fig. 2.4 – Função de escala φ e wavelet ψ ortogonais com 2 momentos nulos, construídas por Daubechies

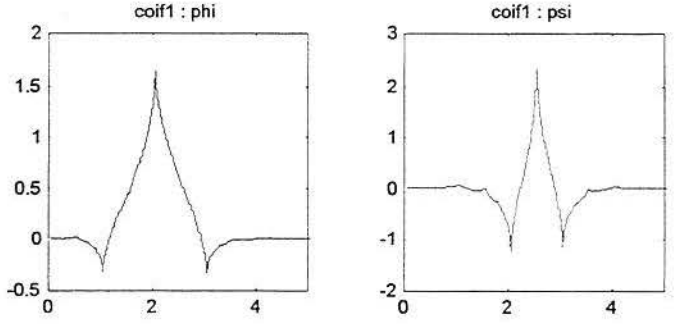

Fig. 2 5 – Exemplo de função de escala φ e wavelet ψ ortogonais, construídas por Daubechies-Coiffman (Coifflet)

A estrutura mais geral de AMR biortogonal utiliza operadores de projeção semelhantes:

$$P_{V_j} f = \sum_k \langle f, \varphi_{jk} \rangle \tilde{\varphi}_{jk} \quad \text{e} \quad P_{W_j} f = \sum_k \langle f, \psi_{jk} \rangle \tilde{\psi}_{jk},$$

onde os pares de funções φ, $\tilde{\varphi}$ e ψ, $\tilde{\psi}$ são utilizados para dividir o trabalho: uma função do par age como função de análise, enquanto a outra age como função de reconstrução. As funções φ e ψ são chamadas *função de escala primal* e *função wavelet primal*, respectivamente, e as funções $\tilde{\varphi}$ e $\tilde{\psi}$ são ditas *função de escala dual* e *wavelet dual*, respectivamente. Embora outras convenções sejam possíveis, supomos que as funções primais são usadas para análise e as funções duais, para síntese. Em termos da AMR, esse esquema leva a uma família biortogonal de funções de escala e wavelets, que são bases duais dos espaços de aproximação e de detalhes.

Mais precisamente, definimos funções φ_j e $\tilde{\varphi}_j$ que formam, respectivamente, bases de Riesz dos subespaços V_j e $\tilde{V}_j$. Da mesma forma, definimos o par de funções wavelet ψ_j e $\tilde{\psi}_j$ que formam, respectivamente, bases de Riesz dos subespaços W_j e $\tilde{W}_j$. Essas funções geram escadas duais de AMR

$$\cdots \subset V_1 \subset V_0 \subset V_{-1} \subset \cdots$$

$$\cdots \subset \tilde{V}_1 \subset \tilde{V}_0 \subset \tilde{V}_{-1} \subset \cdots$$

onde $V_0 = \text{gerado}\{\varphi_{0,k} : k \in \mathbb{Z}\}$ e $\tilde{V}_0 = \text{gerado}\{\tilde{\varphi}_{0,k} : k \in \mathbb{Z}\}$. Os espaços W_j e $\tilde{W}_j$, gerados por ψ_{jk} e $\tilde{\psi}_{jk}$ são, respectivamente, os complementares de V_j em V_{j-1} e de $\tilde{V}_j$ em $\tilde{V}_{j-1}$. Em outras palavras, $V_{j-1} = V_j \oplus W_j$ e $\tilde{V}_{j-1} = \tilde{V}_j \oplus \tilde{W}_j$. A intersecção desses espaços é o espaço nulo, isto é, $V_j \cap W_j = \{0\}$ e $\tilde{V}_j \cap \tilde{W}_j = \{0\}$, mas os espaços V_j e W_j, e também $\tilde{V}_j$ e $\tilde{W}_j$, não são ortogonais, em geral.

Para compensar a perda da ortogonalidade dos espaços de aproximação e de detalhes, impomos uma relação de biortogonalidade entre as escadas de multi-resolução primal e dual,

$$V_j \perp \tilde{W}_j \text{ e } \tilde{V}_j \perp W_j,$$

e, conseqüentemente, para $l \neq j$,

$$W_j \perp \tilde{W}_l.$$

Os primeiros exemplos de wavelets biortogonais foram desenvolvidos independentemente por Cohen *et al.* (1992) e Vetterli & Herley (1992) [17]. É possível construir wavelets biortogonais de forma que as bases primal e dual gerem uma única AMR ortogonal. Nesse caso, a função de escala e a wavelet são ditas *semi-ortogonais*). Os espaços são ortogonais, $V_0 \perp W_0$ e $V_0 \oplus W_0 = V_{-1}$, mas as bases que os formam, não. A designação *pré-wavelet* também é utilizada para esse tipo de wavelets [32].

A Fig. 2.6 traz um exemplo da função de escala e da wavelet, e suas duais, para um elemento da família de wavelets biortogonais, construída por Cohen-Daubechies-Feveau. Essas são, talvez, as wavelets biortogonais mais amplamente usadas, já que a função de escala e a wavelet são simétricas e têm suportes semelhantes. O elemento representado na Fig. 2.6 é usando na compressão de impressões digitais usuais na FBI [06].

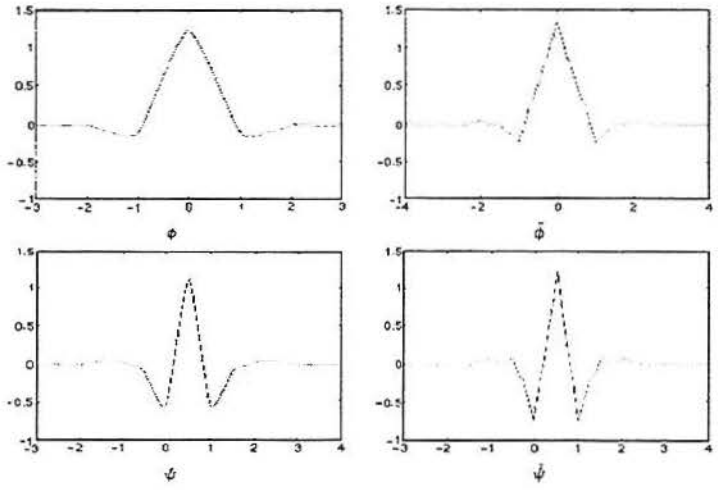

Fig. 2.6 – Função e escala e wavelet, e duais para uma wavelet da família Cohen-Daubechies-Feauveau

A biortogonalidade resulta em maior flexibilidade na escolha de funções com propriedades desejáveis. Wavelets biortogonais podem ser simétricas e ter suporte compacto. Simetria é incompatível com ortogonalidade, exceto no caso da wavelet de Haar, ou no caso de multiwavelets, onde usamos mais de uma função de escala. Para algumas aplicações, como em análise numérica, simetria não é importante. Como é

usual, esses benefícios têm um preço. Bases wavelets biortogonais não reproduzem a energia da função exatamente e a reconstrução da função a partir dos coeficientes pode amplificar qualquer erro introduzido nos coeficientes.

2.6 O Algoritmo de Mallat

Outro resultado da AMR, desenvolvido por Stéphane Mallat, foi a transformada wavelet rápida (TWR) também chamada de *algoritmo de Mallat*. Inspirado no algoritmo da pirâmide, descrito nos anos 80 por Burt e Andelson [20], esse algoritmo consiste em um método simples e rápido para a decomposição de uma função (sinal) em componentes que diferem entre si por um fator 2. O resultado da transformação tem o mesmo número de pontos (coeficientes) da função original e o processo tem complexidade linear.

Como conseqüência da AMR (2.26) e (2.27), temos que qualquer função $f \in L^2(\mathbb{R})$ pode ser decomposta como a soma de suas projeções sobre os subespaços de wavelets:

$$f = \sum_{j \in \mathbb{Z}} P_{W_j} f. \tag{2.32}$$

Denotaremos as projeções de f sobre V_j e W_j por $f_j := P_{V_j} f$ e $o_j := P_{W_j} f$. Como $V_j = V_{j+1} \oplus W_{j+1}$, segue que qualquer função $f_j \in V_j$ pode ser expressa por

$$f_j = f_{j+1} + o_j. \tag{2.33}$$

Essa é a principal relação recursiva para construir uma representação de uma função usando decomposição wavelet. Para implementar a decomposição e a reconstrução wavelet, precisamos calcular as projeções sobre os espaços V_j e W_j. Sabemos que os conjuntos $\{\varphi_{jn} : n \in \mathbb{Z}\}$ e $\{\psi_{jn} : n \in \mathbb{Z}\}$, cujos elementos são definidos por (2.22) e (2.25), são bases ortonormais de V_j e W_j. Portanto, os operadores de projeção P_{V_j} e P_{W_j} são dados pelo produto interno de f com os elementos dessas bases.

$$P_{V_j}(f) = \sum_n \langle f, \varphi_{jn} \rangle \varphi_{j,n} = \sum_n \left(\int f(x) \overline{\varphi_{jn}(x)} dx \right) \varphi_{jn}, \tag{2.34}$$

$$P_{W_j}(f) = \sum_n \langle f, \psi_{jn} \rangle \psi_{jn} = \sum_n \left(\int f(x) \overline{\psi_{jn}(x)} dx \right) \psi_{jn}. \tag{2.35}$$

Veremos que não precisamos calcular as integrais em (2.34) e (2.35) explicitamente. Podemos aproveitar a recursividade dos processos de decomposição e reconstrução, que requerem apenas projeções entre subespaços consecutivos da escada da multi-resolução. A interdependência entre dois subespaços consecutivos em uma AMR é formulada pelas *equações de refinamento* e wavelet,

$$\varphi(x) = \sum_k h_k \varphi_{-1,k}(x), \tag{2.36}$$

$$\psi(x) = \sum_k g_k \varphi_{-1,k}(x). \tag{2.37}$$

Como $V_{j-1} = V_j \oplus W_j$ e, conseqüentemente, $V_j \subset V_{j-1}$ e $W_j \subset V_{j-1}$, usando as relações (2.36) e (2.37), podemos expressar as funções de escala e wavelet, básicas para os espaços V_j e W_j, no nível j, em termos das funções básicas para os espaço subseqüente V_{j-1}, no espaço de resolução mais fina j-1. De fato, usando (2.36) em (2.22), obtemos

$$\left.\begin{aligned}
\varphi_{j,k}(x) &= 2^{-j/2}\varphi(2^{-j}x-k) \\
&= 2^{-j/2}\sum_n h_n 2^{1/2}\varphi(2^{-j+1}x-2k-n) \\
&= \sum_n h_n \sum_n h_n 2^{-j+1/2}\varphi(2^{-j+1}x-(2k+n)) \\
&= \sum_n h_n \varphi_{j-1,2k+n}(x). \\
&= \sum_n h_{n-2k}\varphi_{j-1,n}(x)
\end{aligned}\right\} \qquad (2.38)$$

Da mesma forma, usando (2.37) em (2.25), resulta

$$\left.\begin{aligned}
\psi_{jk}(x) &= 2^{-j/2}\psi(2^{-j}x-k) \\
&= 2^{-j/2}\sum_n g_n 2^{1/2}\varphi(2^{-j+1}x-2k-n) \\
&= \sum_n g_n \sum_n g_n 2^{-j+1/2}\varphi(2^{-j+1}x-(2k+n)) \\
&= \sum_n g_n \varphi_{j-1,2k+n}(x). \\
&= \sum_n g_{n-2k}\varphi_{j-1,n}(x)
\end{aligned}\right\} \qquad (2.39)$$

Podemos, então, usar as seqüências $(h_n)_{n\in\mathbb{Z}}$ e $(g_n)_{n\in\mathbb{Z}}$ para calcular recursivamente os produtos internos em (2.34) e (2.35):

$$\langle f, \varphi_{jk}\rangle = \left\langle f, \sum_n \overline{h_{n-2k}}\varphi_{j-1,k}\right\rangle = \sum_n \overline{h_{n-2k}}\langle f, \varphi_{j-1,k}\rangle, \qquad (2.40)$$

$$\left\langle f, \psi_{jk} \right\rangle = \left\langle f, \sum_n \overline{g_{n-2k}} \varphi_{j-1,k} \right\rangle = \sum_n \overline{g_{n-2k}} \left\langle f, \varphi_{j-1,k} \right\rangle. \tag{2.41}$$

As igualdades (2.40) e (2.41) mostram como os coeficientes da representação de uma função na escala 2^{j-1} estão relacionados com os coeficientes da representação na próxima escala (menos fina) e com os coeficientes do espaço wavelet complementar.

Também mostramos que, a partir dos produtos internos da função f com as funções da base de V_{j-1}, podemos obter os produtos internos com as bases de V_j e de W_j, sem calcular explicitamente as integrais. Esse é o resultado crucial para o desenvolvimento dos métodos recursivos da decomposição e reconstrução wavelet.

2.6.1 Decomposição. Sem perda de generalidade, supomos que o processo de decomposição wavelet inicie com a representação de uma função f no espaço V_0. Dada a função f, representada pelos coeficientes (c_k) de sua seqüência de representação no espaço de escala V_0, isto é,

$$P_{W_0} f = \sum_k \langle f, \varphi_{0k} \rangle \varphi_{0k}(x) = \sum_k c_k^0 \varphi_{0k}. \tag{2.42}$$

No caso de termos apenas *samples* da função f, distribuídas uniformemente, isto é, $f(k)$, $k \in \mathbb{Z}$, os coeficientes (c_k^j) podem ser obtidos a partir desses valores, por meio de uma operação de convolução [17]. O objetivo da decomposição é tomar a seqüência inicial de coeficientes $\left(c_k^0\right)_{k\in\mathbb{Z}}$ e transformá-la na seqüência dos coeficientes da representação wavelet da função. O processo é feito por meio da aplicação recursiva da regra da decomposição (2.33). O processo inicia com $f^0 \in V_0 = V_1 \oplus W_1$ e, no primeiro passo, f^0 é decomposta em $f^1 + o^1$. O processo recursivo age sobre f^j, decompondo-a em $f^j = f^{j+1} + o^{j+1}$, $j = 0, 1, 2, \ldots, N$. Aplicando essa relação recursivamente, temos a representação wavelet

$$f = f^{j+N} + o^{j+N} + \ldots + o^{j+2} + o^{j+1}, \tag{2.43}$$

o que expressa que uma função $f \in V_j$ é decomposta em suas projeções sobre os espaços W_{j+1}, ..., W_{j+N} e um resíduo, dado por sua projeção sobre o espaço V_{j+N}. Esse processo recursivo está esquematizado na Fig. 2.7.

O processo de decomposição divide a seqüência $\left(c_k^j\right)$ dos coeficientes da função de

$$\begin{array}{ccccccc} f^j & \rightarrow & f^{j+1} & \rightarrow & f^{j+2} & \rightarrow & f^{j+N} \\ & \searrow & & \searrow & & \searrow & \\ & & o^{j+1} & & o^{j+2} & & o^{j+N} \end{array}$$

Fig. 2.7 – Esquema da decomposição wavelet de *f*

função de escala, associados com f^j, em duas seqüências $\left(c_k^{j+1}\right)$ e $\left(d_k^j\right)$, a primeira de coeficientes da função de escala e a segunda de coeficientes da função wavelet, associados, respectivamente. Podemos visualizar esse processo como uma transformação, onde realizamos mudança de base,

$$\left(\varphi_{jk}\right)_{k\in\mathbb{Z}} \rightarrow \left(\varphi_{j+1,k}, \psi_{j+1,k}\right)_{k\in\mathbb{Z}}.$$

As igualdades (2.40) e (2.41) dão as fórmulas para fazer a transformação sobre os coeficientes das funções de escala e das wavelets de nível j e posição k, respectivamente:

$$c_k^{j+1} = \sum_n \overline{h_{n-2k}} c_n^j, \tag{2.44}$$

$$d_k^{j+1} = \sum_n \overline{g_{n-2k}} c_n^j. \tag{2.45}$$

Observemos que as seqüências $\left(c_k^{j+1}\right)$ e $\left(d_k^{j+1}\right)$ são calculadas por meio de

convoluções discretas com as seqüências (h_n) e (g_n), respectivamente. Observemos também que estamos retendo apenas os coeficientes pares para o próximo passo recursivo (devido ao fator $2k$ no índice). Essa é uma operação de *dizimação*.

Em resumo, iniciamos com a seqüência $\left(c_n^0\right)$, contendo $n = 2^J$ coeficientes, que é decomposta nas seqüências $\left(d_{n/2}^1\right)$, $\left(d_{n/4}^2\right)$, $\cdots$, $\left(d_{n/2^J}^J\right)$ e $\left(c_{n/2^J}^J\right)$. O processo de decomposição resulta em uma representação wavelet com o mesmo número de coeficientes da representação original.

2.6.2 Reconstrução. No processo de reconstrução, os coeficientes da representação de escala são gerados a partir dos coeficientes da representação wavelet. No caso de bases ortogonais, temos uma reconstrução exata, onde a entrada da reconstrução é igual á saída da decomposição.

Para desenvolver as relações recursivas do processo de reconstrução, lembramos que um passo da decomposição consiste em tomar uma representação f^{j} da função e separá-la em suas componentes f^{j} e o^{j}:

$$\begin{aligned} f^{j-1}(x) &= f^{j}(x) + o^{j}(x) \\ &= \sum_k c_k^j \varphi_{jk}(x) + \sum_k d_k^j \psi_{jk}(x). \end{aligned} \tag{2.46}$$

Precisamos recuperar a seqüência dos coeficientes $\left(c_n^{j-1}\right)$ a partir das $\left(c^j\right)$ e $\left(d^j\right)$.

Temos

$$c_n^{j-1} = \left\langle f^{j-1}, \varphi_{j-1,n} \right\rangle. \tag{2.47}$$

Usando (2.46) em (2.47), obtemos

$$c_n^{j-1} = \left\langle \sum_k c_k^j \varphi_{jk} + \sum_k d_k^j \psi_{jk},\ \ \varphi_{j-1,n} \right\rangle$$

$$= \sum_k c_k^j \left\langle \varphi_{jk},\ \varphi_{j-1,n} \right\rangle + \sum_k d_k^j \left\langle \psi_{jk},\ \varphi_{j-1,n} \right\rangle. \tag{2.48}$$

Como $\varphi_0, \psi_0 \in V_{-1}$, estas podem ser representadas como uma combinação linear de elementos da base $\{\varphi_{j-1,n} : n \in \mathbb{Z}\}$. Logo, $\varphi_0 = \sum_n \langle \varphi_0, \varphi_{-1,n} \rangle \varphi_{-1,n}$ e $\psi_0 = \sum_n \langle \psi_0, \varphi_{-1,n} \rangle \varphi_{-1,n}$. Como essa representação é única, usando as relações (2.36) e (2.37), vem

$$h_n = \langle \varphi_0, \varphi_{-1,n} \rangle \quad \text{e} \quad g_n = \langle \psi_0, \varphi_{-1,n} \rangle.$$

Esses resultados produzem uma fórmula de reconstrução para os coeficientes c_n^{j-1}, a partir das seqüências de decomposição no nível j:

$$c_n^{j-1} = \sum_k h_{n-2k} c_k^j + \sum_k g_{n-2k} d_k^j$$

$$= \sum_k \left[h_{n-2k} c_k^j + g_{n-2k} d_k^j \right]. \tag{2.49}$$

O processo de reconstrução constrói a representação final $\left(c_n^0\right)$, de baixo para cima. Em cada passo, são combinadas as seqüências $\left(c_n^j\right)$ e $\left(d_n^j\right)$ para recuperar a seqüência intermediária $\left(c_n^{j-1}\right)$, para $j = J, \ldots, 1$.

A implementação direta do método descrito nesta Secção 2.6 consiste no algoritmo recursivo, desenvolvido por Mallat. A implementação das fórmulas (2.44) e (2.45) conduz à transformada, e a da fórmula (2.49) conduz à transformada inversa.

2.7 Multi-resolução e bancos de filtros

Além de ter introduzido o conceito de AMR, descrito na seção anterior, Mallat

também desenvolveu a relação entre a AMR e bancos de filtros [17]. Em muitas aplicações, não precisamos lidar diretamente com as funções de escala ou wavelet. Somente os coeficientes h_k e g_k dessas funções e os coeficientes c_k e d_k da TWD precisam ser considerados. Esses coeficientes podem ser vistos como um filtro e um sinal digital, respectivamente.

2.7.1 – Bancos de filtros.

Nesta secção veremos que a AMR define um banco de filtros com reconstrução perfeita. O banco de filtros tem uma estrutura de pirâmide e sua complexidade computacional é linear.

2.7.1.1 Sistemas e filtros. Em processamento de sinais, a palavra sistema tem um significado bastante amplo. Um sistema pode modificar um sinal (entrada) de diferentes maneiras, produzindo outro sinal (saída). Um filtro é um tipo de sistema que altera somente algumas freqüências de um sinal de entrada. Ou seja, é feita uma seleção e algumas bandas de freqüência são alteradas: atenuadas ou acentuadas.

No universo matemático, um sinal pode ser representado por uma função, e um filtro por uma transformação $S: F_1 \longrightarrow F_2$ entre dois espaços de funções:

$$f \mapsto \boxed{\text{sistema}} \mapsto S(f).$$

O sistema ilustrado acima constitui-se no modelo matemático para estudar as várias operações envolvendo processamento de sinais no universo físico. Se o sistema S é linear e F_1, F_2 têm representação de dimensão finita, isto é, $(R)F_1 = \mathbb{R}^m$ e $(R)F_2 = \mathbb{R}^n$, então o sistema S discretizado, $\overline{S}: \mathbb{R}^m \to \mathbb{R}^n$, é uma transformação linear e pode ser representado por uma matriz de ordem $n \times m$,

$$\begin{bmatrix} a_{11} & \cdots & a_{1m} \\ a_{21} & \cdots & a_{2} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}.$$

2.7.1.2 Operador *downsampling*. O operador *downsampling* de ordem 2, $\downarrow 2 : l^2 \to l^2$, é definido por

$$(\downarrow 2)(x_n) = (x_{2n}).$$

Esse operador, como podemos observar, descarta todos os termos da seqüência, com exceção daqueles, cujos índices são múltiplos de 2. Na literatura, esse operador também é conhecido como *operador de dizimação de ordem 2*. A matriz desse operador é

$$\begin{bmatrix} 1 & & & & & & \\ 0 & 0 & 1 & & & & \\ & & 0 & 0 & 1 & & \\ & & & & 0 & & \\ & & & & & \cdots & \end{bmatrix}.$$

Essa matriz é obtida da matriz identidade **I**, incluindo colunas de zeros, alternadamente. O operador *downsampling* não é inversível, mas possui uma inversa à esquerda, que é o operador *upsampling*.

2.7.1.3 Operador *upsampling*. O operador *upsampling* de ordem 2 é denotado e definido por

$$(\uparrow 2)(x_n)(k) = \begin{cases} x(k), & \text{se} \quad n = 2k \\ 0, & \text{se} \quad n = 2k+1. \end{cases}$$

Esse operador intercala zeros entre os elementos da seqüência de representação:

$$(\uparrow 2)(\ldots, x_{-1}, x_0, x_1, \ldots) = (\ldots, x_{-1}, 0, x_0, 0, x_1, 0, \ldots).$$

A matriz desse operador é

$$\begin{bmatrix} 1 & 0 & & & \\ & 0 & 0 & & \\ & 1 & 0 & 0 & \\ & & 0 & 0 & \\ & & 1 & 0 & \\ & & & & \cdots \end{bmatrix}.$$

Vemos que a matriz do operador *upsampling* é a transposta da matriz do operador *downsampling*. Podemos estabelecer a relação entre esses operadores, usando matrizes:

$$(\downarrow 2)(\uparrow 2) = \mathbf{I}.$$

Definimos um *banco de filtros de análise S*, usando dois filtros L e H, com operadores *downsampling*, conforme ilustrado na Fig. 2.8.

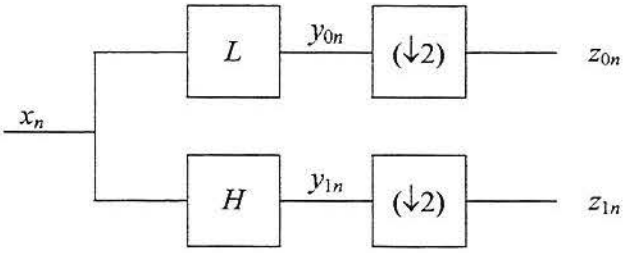

Fig. 2.8 – Esquema de um banco de filtros de análise (decomposição) com dois canais.

O sinal (x_n) é processado pelo filtro L para obter as componentes de baixa freqüência (y_{0n}), e pelo filtro H, para obter as componentes de alta freqüência (y_{1n}). No segundo nível de processamento, as seqüências (y_{0n}) e (y_{1n}), que representam o sinal filtrado, constituem, juntas, a representação do sinal original (x_n), mas com o dobro de *samples*. Para que o número de coeficientes dessas duas seqüências, juntas, seja o mesmo que o do sinal original, precisamos descartar termos. Tal operação é realizada através de *downsampling*. O sinal de saída $S(x_n)$ do banco de análise é, conseqüentemente, uma representação do sinal original em termos de suas componentes de freqüências altas e baixas, com o mesmo número de coeficientes que o sinal original.

Definimos também um banco de filtros de síntese $\tilde{S}$ pelo esquema da Fig. 2.9. Na síntese, aplicamos uma operação de *upsampling* para as componentes dos dois canais, passagem baixa e passagem alta, e, então, usamos o par de filtros L_1 e H_1 de modo que a combinação das componentes reconstrua o sinal original. A combinação desses dois bancos de filtros S e $\tilde{S}$ forma um banco de filtros de análise e síntese. O banco de filtros S é chamado *banco de análise*, porque produz a representação do sinal em termos de suas freqüências. O banco de filtros $\tilde{S}$ é chamado *banco de síntese*, porque reconstrói o sinal a partir de sua representação. Quando o sinal de saída é o mesmo que o sinal de entrada, isto é, se

$$(\hat{x}_n) = \tilde{S}S(x_n) = (x_n),$$

dizemos que o banco de filtros satisfaz a propriedade da *reconstrução perfeita*.

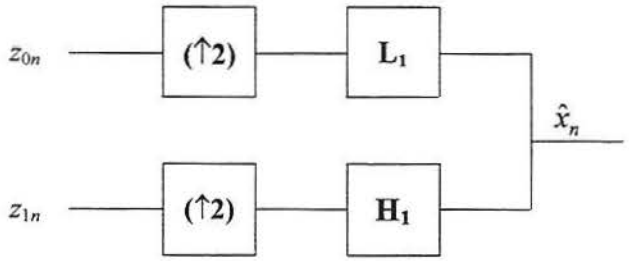

Fig. 2.9 – Esquema de um banco de filtros de síntese (reconstrução)

2.7.2 Representação matricial

2.7.2.1 Exemplo. Suponhamos que os filtros L e H sejam definidos como operadores de convolução

$$L(x_n) = \sum_{k=0}^{3} c(k)x(n-k)$$

e

$$H(x_n) = \sum_{k=0}^{3} g(k)x(n-k).$$

Então, o banco de análise é dado pela matriz

$$\mathbf{S} = \begin{bmatrix} \downarrow L \\ \downarrow H \end{bmatrix},$$

ou, usando os resultados da aplicação do operador ($\downarrow 2$),

$$\mathbf{S} = \begin{bmatrix} c(3) & c(2) & c(1) & c(0) & & \\ & & c(3) & c(2) & c(1) & c(0) \\ g(3) & g(2) & g(1) & g(0) & & \\ & & g(3) & g(2) & g(1) & g(0) \end{bmatrix}. \quad \square$$

A propriedade da reconstrução perfeita está relacionada com a inversibilidade da matriz **S**. Se **S** é inversível, o banco de síntese é $\tilde{\mathbf{S}} = \mathbf{S}^{-1}$, e temos reconstrução exata. Um caso particular importante ocorre quando o sistema linear é ortogonal, no caso de termos a matriz **S** ortogonal. Esse sistema possui a propriedade da reconstrução perfeita e o filtro de síntese, isto é, a matriz inversa $\tilde{\mathbf{S}}$, é determinada pela transposta da matriz de análise **S**: $\tilde{\mathbf{S}} = \mathbf{S}^{t}$.

Dada uma AMR, a função de escala associada é um filtro de passagem baixa e a wavelet correspondente é um filtro de passagem alta. Veremos que, no domínio discreto, a AMR define um banco de filtros semelhante ao banco de filtros $\tilde{S}$ e S, que possui a propriedade de reconstrução perfeita .

Se a função φ está associada à AMR, definida na Seção 2.4, então, como já vimos, $V_{j-1} = V_j \oplus W_j$, onde W_j é o complemento ortogonal de V_j. Dada uma função $f \in L^2(\mathbb{R})$, temos

$$P_{V_{j-1}} f = P_{V_j} f + P_{W_j} f,$$

onde $P_{V_j} f$ representa a projeção de f na escala 2^j e $P_{W_j} f$ representa a projeção de f, que dá os detalhes perdidos quando f é representada nessa escala, ambas definidas pela transformada wavelet. Usando a notação

$$L_j f := P_{V_j} f \quad \text{e} \quad H_j f := P_{W_j} f,$$

temos

$$f = L_j f + H_j f.$$

Aplicando sucessivamente essa decomposição para a componente de baixa freqüência $L_j f$ da função sinal, resulta

$$f = L_j^k f + H_k f + H_{k-1} f + \cdots + H_j f. \tag{2.50}$$

Em resumo, a função f é decomposta em uma componente de baixa freqüência (na escala que não captura detalhes) $L_j^k f$ e componentes de alta freqüência $H_n f$, com $n = k, k\text{-}1, \ldots, j$, que contêm os detalhes perdidos, quando vamos para uma representação de menor resolução de f.

A fórmula (2.50) estabelece que podemos reconstruir a função f exatamente a partir da componente de baixa resolução, adicionando as componentes de alta resolução adequadamente. Conseqüentemente, temos um banco de filtros de dois canais com reconstrução perfeita. As operações de reconstrução do banco de filtros de análise e de decomposição do banco de filtros de síntese são ilustradas nas Fig. 2.10 e Fig. 2.11.

$$\begin{array}{ccccccccc} & & L_j f & & L_j^2 f & & & & L_j^k f \\ & \nearrow & & \nearrow & & \nearrow & \cdots & \nearrow & \\ f & \rightarrow & H_j f & \rightarrow & H_{j+1} f & \rightarrow & & \rightarrow & H_k f \end{array}$$

Fig. 2.10 – Diagrama de decomposição

$$\begin{array}{ccccccccc} L_k f & \rightarrow & L_{k+1} f & \rightarrow & & & \rightarrow & L_j f & \rightarrow & f \\ & \nearrow & & \nearrow & \cdots & \nearrow & & \nearrow & & \\ H_k f & & H_{k+1} f & & & & H_j f & & & \end{array}$$

Fig. 2.11 – Diagrama de reconstrução

2.7.3 Análise multi-resolução discreta. Nosso objetivo, agora, é estudar o banco de filtros acima, que corresponde à AMR, no domínio discreto. Todos os ingrediente para a discretização de uma AMR, e a revisão do banco de filtros associado, no domínio discreto, já são de nosso conhecimento.

Podemos escrever as expressões (2.40) e (2.41) usando o operador filtro (convolução e *downsampling*), que pode ser identificado com um banco de filtros. Então segue que

$$\left\langle f,\ \varphi_{jk}(x)\right\rangle = (\downarrow 2)\left[\left(\left\langle f,\ \varphi_{j-1,n}\right\rangle\right) * \overline{h_{-n}}\right], \tag{2.51}$$

$$\left\langle f,\ \psi_{jk}(x)\right\rangle = (\downarrow 2)\left[\left(\left\langle f,\ \psi_{j-1,n}\right\rangle\right) * \overline{g_{-n}}\right], \tag{2.52}$$

Essas são as fórmulas de decomposição e reconstrução do banco de filtros, associado com a AMR. Estudaremos essas fórmulas mais detalhadamente.

Sem perda de generalidade, podemos supor que a função f seja definida

inicialmente em alguma escala como f^0. Supomos que essa escala esteja associada a alguma freqüência na qual f pode ser representada por uma seqüência de *samples*. Supomos também que $f^0 = P_{V_0} f$, isto é, que f^0 seja a discretização da função f no espaço de escala V_0. Como $V_0 = V_1 + W_1$, temos

$$f^0 = f^1 + o^1.$$

Além disso,

$$f^0 = \sum_n c_n^0 \varphi_{on}, \quad f^1 = \sum_n c_n^1 \varphi_{1n}, \quad o^1 = \sum_n d_n^1 \varphi_{1n}.$$

De (2.40) e (2.41) obtemos

$$c_k^1 = \sum_n h_{n-2k} c_n^0, \quad d_k^1 = \sum_n g_{n-2k} c_n^0.$$

Essas igualdades permitem-nos obter uma representação de f na escala mais fina V_{-1} a partir da seqüência inicial $\left(c_n^0\right)_{n\in\mathbb{Z}}$ na escala V_0. Do ponto de vista da álgebra linear, estamos apenas fazendo uma mudança de base, da base $\{\varphi_{0n}\}_{n\in\mathbb{Z}}$ do espaço V_0 para as bases $\{\varphi_{1n}\}_{n\in\mathbb{Z}}$ e $\{\psi_{1n}\}_{n\in\mathbb{Z}}$.

Indicando por **L** a matriz do operador em (2.52) e por **H** a matriz do operador em (2.51), podemos escrever

$$\left(c_n^1\right)^t = \mathbf{L}\left(c_n^0\right)^t \quad \text{e} \quad \left(d_n^1\right)^t = \mathbf{H}\left(c_n^0\right)^t.$$

Da mesma forma,

$$f^1 \in V_{-1} = V_{-2} \oplus W_{-2}$$

e, conseqüentemente,

$$f^1 = f^2 + o^2, \quad f^2 \in V_{-2}, \quad o^2 \in W_{-2}.$$

Então

$$f^2 = \sum_n c_n^2 \varphi_{2,n}, \quad o^2 = \sum_n d_n^2 \psi_{2,n},$$

com

$$\left(c_n^2\right)^t = \mathbf{L}\left(c_n^1\right)^t \quad \text{e} \quad \left(d_n^2\right)^t = \mathbf{H}\left(c_n^1\right)^t.$$

Temos então um banco de filtros de análise, que é dado pelo diagrama da Fig. 2.12. Esse banco de filtros é um caso particular de um banco de filtros chamado *banco da pirâmide*. De fato, a estrutura de pirâmide é a melhor forma para representar esse tipo de banco de filtros em um diagrama.

$$\begin{array}{ccccccccc} & & \left(d_n^1\right) & & \left(d_n^2\right) & & & & \left(d_n^k\right) \\ & & H\nearrow & & H\nearrow & & \cdots & & H\nearrow & \\ \left(c_n^0\right) & L\rightarrow & \left(c_n^1\right) & L\rightarrow & \left(c_n^2\right) & & & L\nearrow & \left(c_n^k\right) \end{array}$$

Fig. 2.12 – Banco de filtros de análise

2.7.3.1 Exemplo. Consideremos um sinal (função), dado por uma seqüência de *samples* com 8 elementos, isto é: $f^0 = \left(c_0^0, c_1^0, c_2^0, \ldots, c_7^0\right)$. As análises sucessivas de f^0 pelo banco de filtros são representadas na pirâmide invertida do diagrama da Fig. 2.13. No primeiro nível temos a seqüência inicial da função f; no segundo nível, os coeficientes $\left(c_n^1\right)$ de baixa resolução e os coeficientes dos detalhes $\left(d_n^1\right)$. Continuamos a operação do banco de filtros, seguindo nível por nível da pirâmide, até alcançar uma seqüência com um único elemento $\left(c_0^3\right)$.

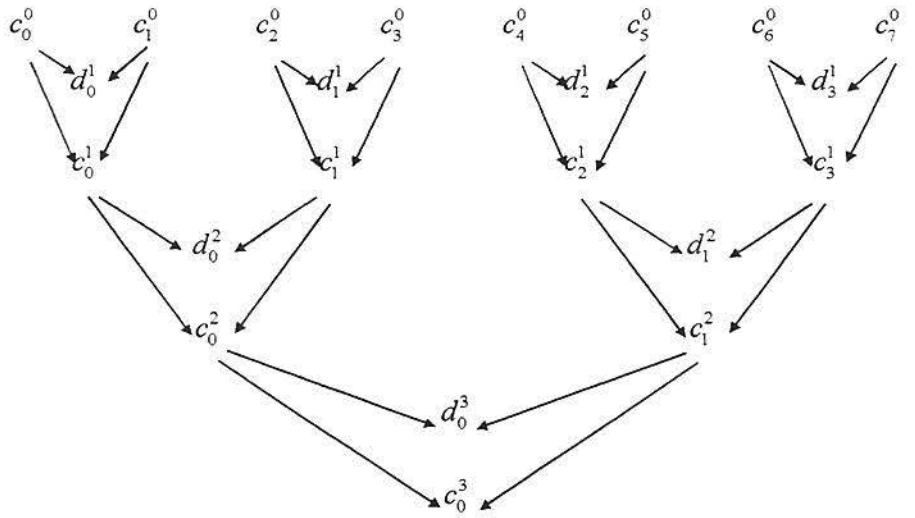

Fig. 2.13 – Diagrama da Pirâmide

Em notação matricial, o filtro é representado pela estrutura

$$\mathbf{S} = \begin{bmatrix} L_k \\ H_k \end{bmatrix} \begin{bmatrix} \\ \mathbf{I} \end{bmatrix} \cdots \begin{bmatrix} L_1 \\ H_1 \end{bmatrix} \begin{bmatrix} \\ \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{L} \\ \mathbf{H} \end{bmatrix}. \tag{2.53}$$

A ordem de cada bloco $\begin{bmatrix} \mathbf{L}_k \\ \mathbf{H}_k \end{bmatrix}$ é metade da ordem do bloco anterior $\begin{bmatrix} \mathbf{L}_{k-1} \\ \mathbf{H}_{k-1} \end{bmatrix}$. Desse modo, a ordem do bloco da identidade **I** dobra de tamanho de uma matriz para outra. Este produto de matrizes é a representação do banco de filtros da AMR. Quando essa matriz é aplicada ao vetor inicial $\left(c_n^0\right)$, produz decomposição de multi-resolução completa. O vetor resultante $S\left(\left(c_n^0\right)\right)$ dessa operação é

$$\left(c^{J_0},\ d^{J_0},\ d^{J_0-1},\ \ldots,\ d^0\right),$$

onde

$$d^k = \left(d_0^k,\ d_1^k,\ \ldots,\ d_m^k\right), \qquad k = J_0,\ J_0\text{-}1\ ,\ldots,\ 0.\ \square$$

2.7.3.2 Exemplo. As equações de refinamento e wavelet da AMR de Haar são, respectivamente, $\varphi(x) = \varphi(2x) + \varphi(2x-1)$ e $\psi(x) = \psi(2x) - \psi(2x-1)$. Para um sinal representado por 4 *samples*, a matriz do banco de filtros, correspondente ao primeiro nível, tem ordem 4 e é dada por

$$\begin{bmatrix} \mathbf{L} \\ \mathbf{H} \end{bmatrix} = \begin{bmatrix} 1 & 1 & & \\ & & 1 & 1 \\ 1 & -1 & & \\ & & 1 & -1 \end{bmatrix}.$$

Para que essa matriz seja um operador ortogonal devemos multiplicá-la por $r = 1/\sqrt{2}$, obtendo a matriz

$$\begin{bmatrix} r & r & & \\ & & r & r \\ r & -r & & \\ & & r & -r \end{bmatrix}.$$

No segundo nível a matriz é

$$\begin{bmatrix} r & r \\ r & -r \end{bmatrix}.$$

Conseqüentemente, a matriz do banco de filtros resulta do produto

$$\begin{bmatrix} r & r & & \\ r & -r & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} r & r & & \\ & & r & r \\ r & -r & & \\ & & r & -r \end{bmatrix}.\ \square$$

Se as funções de escala e wavelet formam bases ortonormais, os operadores envolvidos no banco de filtros de análise e síntese também são ortonormais. Assim, a operação inversa para o banco de síntese é dada pela matriz adjunta (transposta conjugada) e o banco de filtros resulta em reconstrução perfeita. A expressão da reconstrução é dada pela igualdade (2.49) e é dita a igualdade de síntese do banco de filtros, associado à multi-resolução.

2.8 Transformada wavelet bidimensional

Os comentários abaixo se estendem para o caso n-dimensional. Consideraremos apenas o caso em que as bases são ortonormais.

Existem duas maneiras de construir wavelets em $\mathbb{R}^2$: a primeira é baseada no produto tensorial de bases wavelets unidimensionais, que é uma maneira direta e leva a uma transformada wavelet na forma (2.54) [11]. A segunda maneira é por meio de uma AMR bidimensional, gerada por uma única função de escala a duas variáveis em $\mathbb{R}^2$ [17]. Abordamos antes a primeira maneira.

De forma trivial, podemos construir uma base ortonormal para $L^2(\mathbb{R}^2)$, a partir de uma base de wavelets ortonormal $L^2(\mathbb{R})$. Basta tomar a função gerada pelo produto tensorial de duas bases unidimensionais:

$$\Psi_{j_1k_1j_2k_2}(x,y) = \psi_{j_1k_1}(x)\psi_{j_2k_2}(y). \tag{2.54}$$

As funções resultantes são, de fato, wavelets, e $\left\{\Psi_{j_1k_1j_2k_2} : j_1, j_2, k_1, k_2 \in \mathbb{Z}\right\}$ é uma base ortonormal de $L^2(\mathbb{R}^2)$ [11]. Nessa base, as variáveis x e y são dilatadas separadamente.

Em outra construção, usamos o produto tensorial da AMR bidimensional, ao invés do produto de wavelets básicas. Definimos os espaços V_j, $j \in \mathbb{Z}$, por

$$V_0^{(x,y)} = V_0^x \otimes V_0^y = \overline{\text{gerado}\{F(x,y) = f(x)g(y) : f, g \in V_0\}},$$

sendo

$$F(x,y) \in V_j^{(x,y)} \Leftrightarrow F(2^j x,\ 2^j y) \in V_0^{(x,y)}.$$

Então $V_j(x,y)$ forma uma escada multi-resolução em $L^2(\mathbb{R}^2)$,

$$\cdots V_2^{(x,y)} \subset V_1^{(x,y)} \subset V_0^{(x,y)} \subset V_{-1}^{(x,y)} \subset V_{-2}^{(x,y)} \cdots,$$

que satisfaz

$$\bigcap_{j\in\mathbb{Z}} V_j^{(x,y)} = \{0\} \text{ e } \overline{\bigcup_{j\in\mathbb{Z}} V_j^{(x,y)}} = L^2(\mathbb{R}^2).$$

Como $\{\varphi(x-n): n\in\mathbb{Z}\}$ é uma base ortonormal para $V_0^{(x,y)}$, os produtos

$$\Phi_{0n_1n_2}(x,y) = \varphi(x-n_1)\varphi(x-n_2),\ \ n_1,n_2 \in \mathbb{Z}$$

formam uma base ortonormal para $V_0^{(x,y)}$, gerada por translações de uma única função Φ em $\mathbb{Z}^2$, por valores inteiros nos dois argumentos x e y. Da mesma forma,

$$\Phi_{jn_1n_2}(x,y) = \varphi_{jn_1}(x)\varphi_{jn_2}(y)$$

$$= 2^{-j}\Phi(2^{-j}x-n_1,\ 2^{-j}y-n_2),\ \ n_1,n_2 \in \mathbb{Z}$$

constitui uma base ortonormal para $V_j^{(x,y)}$. De maneira semelhante ao caso unidimensional, definimos, para cada espaço $W_j^{(x,y)}$, $j\in\mathbb{Z}$, o complemento ortogonal de $V_j^{(x,y)}$ em $V_{j-1}^{(x,y)}$. Temos

$$V_{J-1}^{(x,y)} = V_{j-1}^{(x)} \otimes V_{j-1}^{(y)}$$

$$= \left(V_j^{(x)} \oplus W_j^{(x)}\right) \otimes \left(V_j^{(y)} \oplus W_j^{(y)}\right)$$

$$= \left(V_j^{(x)} \otimes V_j^{(y)}\right) \oplus \left[\left(V_j^{(x)} \otimes W_j^{(y)}\right) \oplus \left(W_j^{(x)} \otimes V_j^{(y)}\right) \oplus \left(W_j^{(x)} \otimes W_j^{(y)}\right)\right]$$

$$= V_j^{(x,y)} \oplus W_j^{(x,y)}.$$

Observemos que $W_j^{(x,y)}$ consiste em três partes, com bases ortonormais dadas por

$$\varphi_{jn_1}(x)\ \psi_{jn_2}(y), \text{ para } V_j^x \otimes W_j^y,$$

$$\psi_{jn_1}(x)\varphi_{jn_2}(y), \text{ para } W_j^x \otimes V_j^y,$$

e

$$\psi_{jn_1}(x)\psi_{jn_2}(y), \text{ para } W_j^x \otimes W_j^y.$$

Então, definimos três wavelets bidimensionais

$$\Psi^1(x,y) = \varphi(x)\psi(y),$$

$$\Psi^2(x,y) = \psi(x)\varphi(y),$$

e

$$\Psi^3(x,y) = \psi(x)\psi(y).$$

Resulta que

$$\left\{\Psi^{\beta}_{jn_1,n_2} : n_1, n_2 \in \mathbb{Z},\ \beta = 1,2 \text{ ou } 3\right\}$$

é uma base ortonormal de $W_j^{(x,y)}$, e

$$\left\{\Psi_{jn}^{\beta}:\ j\in\mathbb{Z},\ \beta=1,\ 2\text{ ou }3\right\}$$

é uma base ortonormal de $\overline{\oplus W_j^{(x,y)}} = L^2(\mathbb{R}^2)$. Mais do que isso, a interpretação em termos de bancos de filtros com relação a uma base ortonormal com suporte compacto se estende para duas dimensões. As seqüências dos coeficientes são dadas por

$$\left.\begin{aligned} c_{lm}^{j} &= \sum_{i,k} h_i h_k c_{2l+1,2m+k}^{j-1}, \\ d_{lm}^{j(x)} &= \sum_{i,k} g_i h_k c_{2l+1,2m+k}^{j-1}, \\ d_{lm}^{j(y)} &= \sum_{i,k} h_i g_k c_{2l+1,2m+k}^{j-1}, \\ d_{lm}^{j(xy)} &= \sum_{i,k} g_i g_k c_{2l+1,2m+k}^{j-1}. \end{aligned}\right\} \qquad (2.55)$$

3 DESCRIÇÃO DO MÉTODO DA TRANSFORMADA WAVELET DISCRETA INCOMPLETA E RESULTADOS NUMÉRICOS

Neste capítulo apresentamos a construção da matriz de precondicionamento baseada em wavelets, o método da diagonal para mudança de escala, um estudo sobre as condições de fronteira e sobre a esparsidade da matriz precondicionada, bem como os resultados sobre o número de condição da matriz precondicionada e a eficiência do MGC precondicionado com wavelets.

3.1 Construção da matriz precondicionadora $\mathbf{W}_{(J)}$ e precondicionamento

Reconsideraremos o sistema linear (1.5), onde substituímos as notações $\mathbf{T}_N$ por **A**, **v** por **u** e $h^2\mathbf{f}$ por **f**:

$$\mathbf{Au} = \mathbf{f}. \tag{3.1}$$

Observamos também que trocamos a notação v da função incógnita do problema (1.1) por u. Como estabelecido na Seção 1.1, a ordem de (3.1) será sempre do tipo $N = 2^n$ e o passo da malha, $h = 1/(N+1)$.

Comentamos na Seção 1.3 que um número de condição grande da matriz **A** torna lenta a convergência do MGC, para resolver um sistema linear **Au** = **f**. Por isso, convém verificar a possibilidade de diminuir previamente, isto é, antes de aplicar o método, esse número de condição, o que fazemos mediante um bom precondicionamento, o que é

importante por duas razões: primeiro porque um mau condicionamento leva a instabilidades numéricas e, segundo, porque, quando usamos métodos iterativos para resolver um sistema linear, um mau condicionamento leva a uma convergência muito lenta. Há muitas maneiras de escolher uma matriz precondicionadora. O objetivo central neste trabalho é exatamente mostrar como podemos construir uma matriz precondicionadora eficiente bascada em wavelets.

Antes de apresentar o precondicionamento, discutimos um pouco mais sobre wavelets e fixamos a notação. Usaremos as seguintes funções de escala e wavelets de Daubechies da j$^{\text{ésima}}$ ordem de resolução, respectivamente, ao invés daquelas definidas em (2.12) e (2.22),

$$\varphi_{jk}(x) = 2^{(n-j)/2}\varphi(2^{n-j}x - k), \tag{3.2}$$

$$\psi_{jk} = 2^{(n-j)/2}\psi(2^{n-j}x - k), \tag{3.3}$$

onde n é tal que a ordem do sistema (3.1) é 2^n. Vimos em (2.18) que o espaço V_{j-1} é representado como a soma direta dos espaços V_j e W_j. Como $\{\varphi_{jk} : j,k \in \mathbb{Z}\}$ é uma base ortonormal de V_j e $\{\psi_{jk} : j,k \in \mathbb{Z}\}$ é uma base ortonormal de W_j, para todo j, a função $u \in L^2(\mathbb{R})$ pode ser dada pela expansão

$$u(x) = \sum_k c_k^J \varphi_{Jk}(x) + \sum_{j=1}^{J} \sum_k d_k^j \psi_{jk}(x), \tag{3.4}$$

onde c_k^j e d_k^j são os coeficientes da TWD e são dados por

$$c_k^j = \int_{\mathbb{R}} u(x)\varphi_{jk}(x)dx \tag{3.5}$$

e

$$d_k^j = \int_{\mathbb{R}} u(x)\psi_{jk}(x)dx. \tag{3.6}$$

Das equações (3.2) e (3.5), sendo o suporte de φ o intervalo $[0 ; 2M-1]$, obtemos

$$c_k^0 = 2^{n/2}\int_0^{(2M-1)/2^n} u\left(\frac{k}{2^n}+x\right)\varphi\left(2^n x\right)dx \tag{3.7}$$

Expandimos u em séries de Taylor em torno do ponto $x = k/2^n$ para reescrever a igualdade (3.7) na forma

$$\left.\begin{aligned} 2^{-n/2}c_k^0 = {} & u\left(\frac{k}{2^n}\right)\int_0^{(2M-1)/2^n}\varphi\left(2^n x\right)dx \\ & +u'\left(\frac{k}{2^n}\right)\int_0^{(2M-1)/2^n} x\varphi\left(2^n x\right)dx \\ & +u''\left(\frac{k}{2^n}\right)\int_0^{(2M-1)/2^n} x^2\varphi\left(2^n x\right)dx \\ & +\cdots \end{aligned}\right\} \tag{3.8}$$

Como vale [36]

$$\int_0^{(2M-1)/2^n}\varphi\left(2^n x\right)dx = 2^{-n}, \tag{3.9}$$

e

$$\int_0^{(2M-1)/2^n}\left[\varphi\left(2^n x\right)\right]^2 dx = 2^{-n} \tag{3.10}$$

implica o seguinte, pela desigualdade de Schwarz,

$$\left.\begin{aligned} 0 \le \int_0^{(2M-1)/2^n} x\varphi\left(2^n x\right)dx &\le \left(\int_0^{(2M-1)/2^n} x^2 dx\right)^{1/2}\left(\int_0^{(2M-1)/2^n}\varphi(2^n x)^2 dx\right)^{1/2} \\ &= \frac{1}{\sqrt{3}}\left(\frac{2M-1}{2^n}\right)^{3/2}\cdot 2^{-n/2} \\ &= \frac{1}{\sqrt{3}}(2M-1)^{3/2}\cdot 2^{-2n}, \end{aligned}\right\} \tag{3.11}$$

podemos finalmente obter

$$c_k^0 = 2^{-n/2}u\left(\frac{k}{2^n}\right) + O\left(2^{-3n/2}\right). \tag{3.12}$$

A igualdade (3.12) gera a seqüência $\left(c_k^0\right)$ a partir dos valores de u (e f) nos nodos da malha de discretização. Conseqüentemente, podemos calcular todos os

coeficientes da equação (3.4) a partir de (3.12), mediante o algoritmo da pirâmide

$$c_k^j = \sum_{i=0}^{2M-1} h_i c_{2k+i}^{j-1}, \qquad d_k^j = \sum_{i=0}^{2M-1} g_i c_{2k+i}^{j-1}, \tag{3.13}$$

onde (h_i) e (g_i) são os coeficientes nas bases de wavelets.

Mas a implementação direta do algoritmo da pirâmide, descrito na Seção 2.6, traz dificuldades para manejar os pontos de fronteira, uma vez que são requeridos diversos dados, que são desconhecidos porque se originam fora das fronteiras. Existem algumas técnicas para superar tais dificuldades, como ilustra a Fig.3.1.

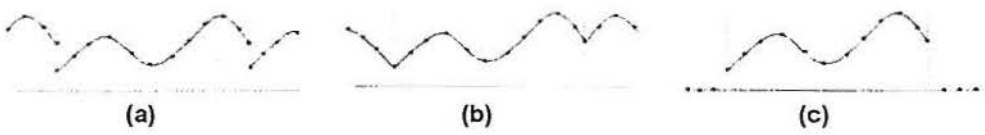

**Fig. 3.1 – (a) periodização por translação,
(b) periodização por reflexão,
(c) extensão com zeros**

Beylkin [03,04,05] apresenta seus estudos, utilizando condições de fronteira periódicas. Mas transformar problemas de fronteira em problemas de condições de fronteira periódicas não é uma coisa prática em geral. O método se torna muito complicado para resolver problemas gerais. Nosso trabalho mostra como evitar tais dificuldades, simplificando o método, de acordo com a proposta de Tanaka [36,37], que iguala a zero todos os valores obtidos fora da fronteira, produzindo o que chamamos de método da *transformada wavelet discreta incompleta* (TWDi). Essa designação foi inspirada pela expressão "fatoração *incompleta* de Cholesky". Por oposição, designamos *transformada wavelet discreta completa* (TWDc) a usada por Beylkin. Embora resulte da TWDi que as transformações não são exatamente ortonormais, o algoritmo é mais acessível e eficaz para implementações numéricas do que o proposto por Beylkin, além de muito mais simples de implementar, e produz aproximações da solução de (3.1) muito boas.

Seja $\mathbf{W}_{(J)}$ a matriz da TWDi que transforma os coeficientes da resolução básica para a J$^{\text{ésima}}$ ordem (nível) de resolução. Então

$$\mathbf{W}_{(J)} = \mathbf{W}_{J-1}\mathbf{W}_{J-2}\cdots\mathbf{W}_0,$$

onde $\mathbf{W}_j$ é a matriz quadrada de ordem N, que transforma dados da $j^{\text{ésima}}$ para a $(j+1)^{\text{ésima}}$ ordem de resolução, isto é,

$$\mathbf{W}_j = \begin{bmatrix} \begin{bmatrix} \mathbf{H}_j \\ \mathbf{G}_j \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N-2^{n-j}} \end{bmatrix}.$$

Aqui $\mathbf{I}_i$ é a matriz identidade de ordem i, e $\mathbf{H}_j$ e $\mathbf{G}_j$ são as matrizes bandas de ordem 2^{n-j-1} por 2^{n-j}, definidas assim:

$$\mathbf{H}_j := \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{2M-1} & 0 & \cdots & & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & \cdots & h_{2M-1} & 0 & & 0 \\ \vdots & & & \ddots & & & & \ddots & & \vdots \\ 0 & & \cdots & 0 & h_0 & h_1 & & \cdots & & h_{2M-1} \\ \vdots & & & & & \ddots & & & & \vdots \\ 0 & & & & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ 0 & 0 & & & \cdots & & 0 & 0 & h_0 & h_1 \end{bmatrix}$$

e

$$\mathbf{G}_j := \begin{bmatrix} g_0 & g_1 & 0 & 0 & 0 & & \cdots & & 0 & 0 \\ g_{-2} & g_{-1} & g_0 & g_1 & 0 & 0 & & & & 0 \\ \vdots & & & & \ddots & & & & & \vdots \\ g_{-2M} & & \cdots & & g_0 & g_1 & 0 & \cdots & & 0 \\ 0 & \ddots & & & & & \ddots & & & \vdots \\ \vdots & & 0 & g_{-2M} & \cdots & g_{-1} & g_0 & g_1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & g_{-2M} & \cdots & g_{-1} & g_0 & g_1 \end{bmatrix}.$$

A equação matricial (3.1), resultante da discretização do problema unidimensional de Poisson, pode ser expressa em termos das matrizes $\mathbf{W}_{(J)}$ na forma

$$\mathbf{W}_{(J)}\mathbf{A}\mathbf{W}_{(J)}^{-1}\vec{\mathbf{u}}_j = \vec{\mathbf{f}}_j,$$

onde

$$\vec{\mathbf{u}}_j := 2^{-n/2}\mathbf{W}_{(J)}\mathbf{u} \quad \text{e} \quad \vec{\mathbf{f}}_j := 2^{-n/2}\mathbf{W}_{(J)}\mathbf{f}.$$

Os vetores $\vec{\mathbf{u}}_j$ e $\vec{\mathbf{f}}_j$ são constituídos pelos coeficientes da expansão em wavelets de u e f. Visto que $\mathbf{W}_{(J)}$ aproxima a TWDc, que é uma transformação ortonormal, temos

$$\mathbf{W}_{(J)}\mathbf{W}_{(J)}^{t} \approx \mathbf{I}_N,$$

e, daí,

$$\mathbf{W}_{(J)}\mathbf{A}\mathbf{W}_{(J)}^{t}\vec{\mathbf{u}}_j \approx \vec{\mathbf{f}}_j. \tag{3.14}$$

3.2 Método da diagonal para mudança de escala

Quando o sistema linear (3.1) é transformado, em relação a um espaço que tem wavelets como funções de uma base, a matriz **A**, que é a forma discreta do Laplaciano, é transformada na matriz $\mathbf{W}_{(J)}\mathbf{A}\mathbf{W}_{(J)}^{t}$. Sendo a TWDi aproximadamente ortonormal, o número de condição dessa nova matriz mantém-se praticamente o mesmo que o de **A**. Entretanto, o método da diagonal para mudança de escala é muito eficaz, nesse caso, para reduzir o número de condição da matriz transformada. Como explicado em [15], as autofunções e os autovalores de um operador L em $\mathbf{V}_j$ e $\mathbf{W}_j$ são, respectivamente,

$$V_j: \quad E_\xi^j = \sum_k e^{ik\xi}\varphi_{jk} \quad \text{e} \quad \lambda_\xi^j = a_j \sum_k p_k e^{-ik\xi},$$

$$W_j: \quad E_\xi^j = \sum_k e^{ik\xi}\psi_{jk} \quad \text{e} \quad \lambda_\xi^j = b_j \sum_k q_k e^{-ik\xi},$$

onde

$$p_k := \int_\Omega \varphi(x-k)L\varphi(x)dx,$$

$$q_k := \int_\Omega \psi(x-k) L\psi(x) dx,$$

$$a_j := \frac{\langle \varphi_0^j, L\varphi_{jk} \rangle}{p_k}, \qquad b_j := \frac{\langle \psi_0^j, L\psi_{jk} \rangle}{q_k}.$$

De acordo com [36], para $\Omega = \mathbb{R}$, $0 \le \xi < 2\pi$, o índice k corre sobre todos os inteiros. Quando as funções são periódicas em $\mathbb{R}$, ξ assume valores discretos no conjunto

$$\{(2\pi k)/2^n : k = 0, 1, 2, \ldots, 2^n - 1\},$$

e a matriz diagonal de mudança de escala, escolhida para melhorar a distribuição dos autovalores, tem a forma

$$\mathbf{P}_j := \begin{bmatrix} \mathbf{I}_{2^{n-j+1}} / \sqrt{a_j} & 0 & 0 & \cdots & 0 \\ 0 & \mathbf{I}_{2^{n-j+1}} / \sqrt{b_j} & 0 & & 0 \\ 0 & 0 & \mathbf{I}_{2^{n-j+2}} / \sqrt{b_{j-1}} & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & 0 & & 0 & \mathbf{I}_{2^{n-1}} / \sqrt{b_1} \end{bmatrix}, \text{ para } j > 1,$$

e

$$\mathbf{P}_j = \mathbf{I}_{2^n} / \sqrt{a_j}, \text{ para } j = 1.$$

Por exemplo, para $L := d^{2m}/dx^{2m}$, temos $a_j = b_j = 2^{-2mj}$. Em nosso problema, temos $m = 1$ e, portanto, $a_j = b_j = 2^{-2j}$, e a matriz $\mathbf{P}_j$, nesse caso, é

$$\mathbf{P}_j := \begin{bmatrix} 2^j \mathbf{I}_{2^{n-j+1}} & 0 & 0 & \cdots & 0 \\ 0 & 2^{j-1} \mathbf{I}_{2^{n-j+1}} & 0 & & 0 \\ 0 & 0 & 2^{j-2} \mathbf{I}_{2^{n-j+2}} & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & 0 & & 0 & 2\mathbf{I}_{2^{n-1}} \end{bmatrix}, \text{ para } j > 1,$$

e

$$\mathbf{P}_j = 2\mathbf{I}_{2^n}, \text{ para } j = 1.$$

Aplicamos a matriz $\mathbf{P}_j$ ao sistema (3.14) para obter

$$\mathbf{P}_J\mathbf{W}_{(J)}\mathbf{A}\mathbf{W}_{(J)}^t\mathbf{P}_J\overline{\mathbf{u}}_J \approx \overline{\mathbf{f}}_J, \tag{3.15}$$

onde

$$\overline{\mathbf{u}}_J := 2^{-n/2}\mathbf{P}_J^{-1}\mathbf{W}_{(J)}\mathbf{u} \quad \text{e} \quad \overline{\mathbf{f}}_J := 2^{-n/2}\mathbf{P}_J\mathbf{W}_{(J)}\mathbf{f}.$$

Agora o MGC precondicionado é numericamente aplicável ao sistema (3.15). Como a mudança de escala melhora o número de condição, o crescimento do número de iterações do MGC associado ao crescimento do número de pontos da malha é suprimido.

3.2.1 Exemplo. Na Fig. 3.2 trazemos um exemplo da matriz diagonal **P** (N = 32), usada para mudança de escala. □

3.3 O MGC precondicionado pela TWDi

3.3.1 Caso unidimensional. No MGC precondicionado aplicamos a TWDi, transformando (3.1) para a forma

$$\mathbf{P}_J\mathbf{W}_{(J)}\mathbf{A}\mathbf{W}_{(J)}^t\mathbf{P}_J\hat{\mathbf{u}}_J = \hat{\mathbf{f}}_J, \tag{3.16}$$

onde

$$\hat{\mathbf{u}}_J := 2^{-n/2}\mathbf{P}_J^{-1}\mathbf{W}_{(J)}^t\mathbf{u} \quad \text{e} \quad \hat{\mathbf{f}}_J := 2^{-n/2}\mathbf{P}_J\mathbf{W}_{(J)}\mathbf{f}.$$

Observemos que, enquanto (3.15) aproxima (3.1), a equação (3.16) é equivalente à (3.1). Para resolver (3.16) pelo MGC precondicionado, ponhamos $\mathbf{V} := \mathbf{P}_j\mathbf{W}_{(j)}$ e tomemos a matriz precondicionadora

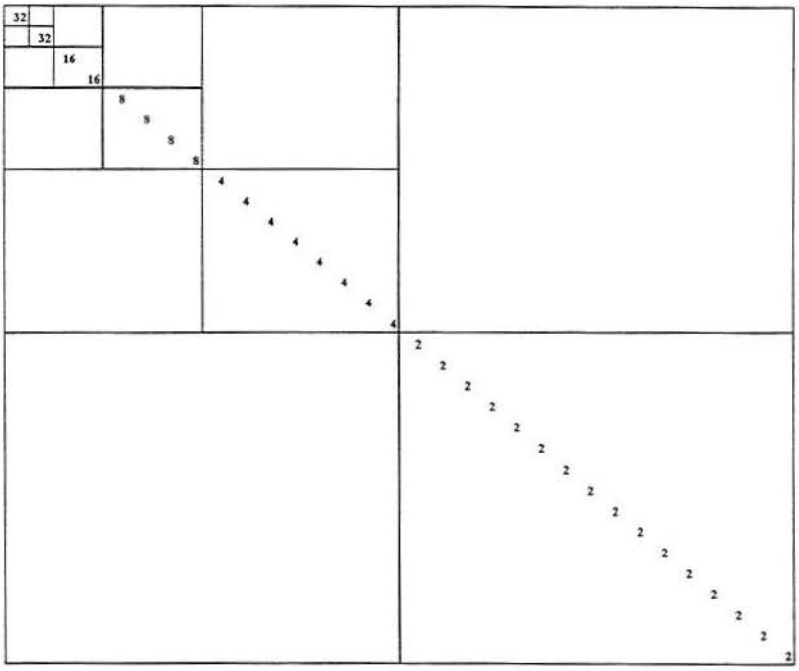

Fig. 3.2 – Matriz P diagonal de mudança de escala para N =32

$$\mathbf{K} := \mathbf{V}^t\mathbf{V} = \mathbf{W}_{(J)}^t\mathbf{P}_J^2\mathbf{W}_{(J)} = \left[\mathbf{P}_J\mathbf{W}_{(J)}^t\right]^t\left[\mathbf{P}_J\,\mathbf{W}_{(J)}\right]. \tag{3.17}$$

No caso da equação unidimensional de Poisson, o cálculo de $\mathbf{P}_j^2$ pode ser simplificado por meio da mudança nos filtros,

$$\left(h_k^{'}\right) := 2\left(h_k\right) \quad \text{e} \quad \left(g_k^{'}\right) := 2\left(g_k\right),$$

que dispensa o cálculo explícito de $\mathbf{P}_J^2$ em (3.17). Isso decorre do seguinte fato (a segunda igualdade é obtida por indução):

$$\mathbf{P}_{(j)}\mathbf{W}_{(j)} = \mathbf{W}_{sj\text{-}1}\mathbf{P}_{(j\text{-}1)}\mathbf{W}_{(j\text{-}1)}$$

$$= \mathbf{W}_{sj\text{-}1}\mathbf{W}_{sj\text{-}2}\cdots\mathbf{W}_{s1}\mathbf{W}_{s0},$$

com

$$\mathbf{W}_{s,j} := \begin{bmatrix} 2\begin{pmatrix} \mathbf{H}_j \\ \mathbf{G}_j \end{pmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N-2^{n-j}} \end{bmatrix}.$$

O resíduo é calculado usando essa mudança nos coeficientes dos filtros:

$$\tilde{\mathbf{r}} = \mathbf{W}_{(J)}^{t} \mathbf{P}_J^2 \mathbf{W}_{(J)} \mathbf{r}.$$

Abaixo visualizamos a estrutura da matriz **A** antes (Fig. 3.3) e depois (Fig. 3.4) do precondicionamento baseado em wavelets para $n = 8$. Nessa representação, em azul estão os elementos não-nulos da matriz e em branco os elementos nulos da matriz.

3.3.2 Caso bidimensional. Para resolver o problema bidimensional numericamente, usamos um esquema de diferenças finitas com uma malha de pontos uniforme sobre a região $\Omega := [0\ ;\ 1] \times [0\ ;\ 1]$, e o produto de Kronecker, definido em 1.5.2, de duas bases unidimensionais para estender a TWDi para duas dimensões, isto é, a matriz wavelet para o caso bidimensional é obtida por

$$\mathbf{W}_{(J)\text{bidimensional}} = \mathbf{W}_{(J)} \otimes \mathbf{W}_{(J)}.$$

Nesse caso, a função $u(x, y) \in L^2(\Omega)$ é expandida assim:

$$u(x,y) = \sum_{l,m} c_{lm}^J \varphi_{Jl} \varphi_{Jm} + \sum_{j=1}^{J} \sum_{lm} \left(d_{lm}^{(x),j} \psi_{jl} \varphi_{jm} + d_{lm}^{(y),j} \varphi_{jl} \psi_{jm} + d_{lm}^{(xy),j} \psi_{jl} \psi_{jm} \right),$$

onde os coeficientes c_{lm}^J, $d_{jm}^{(x),j}$, $d_{jm}^{(y),j}$ e $d_{jm}^{(x,y),j}$ são dados em (2.55).

Os autovalores e as autofunções do operador Laplaciano bidimensional

$$\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

são

$$E_{\xi\eta}^J = \sum_{l,m} e^{i(l\xi + m\eta)} \varphi_{jl} \psi_{jm}$$

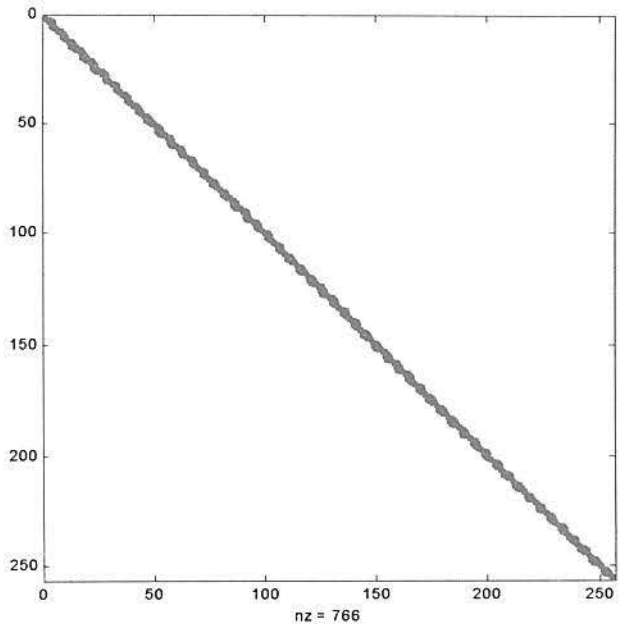

Fig. 3.3 – Estrutura da matriz de A, caso unidimensional, antes do precondicionamento para N = 256

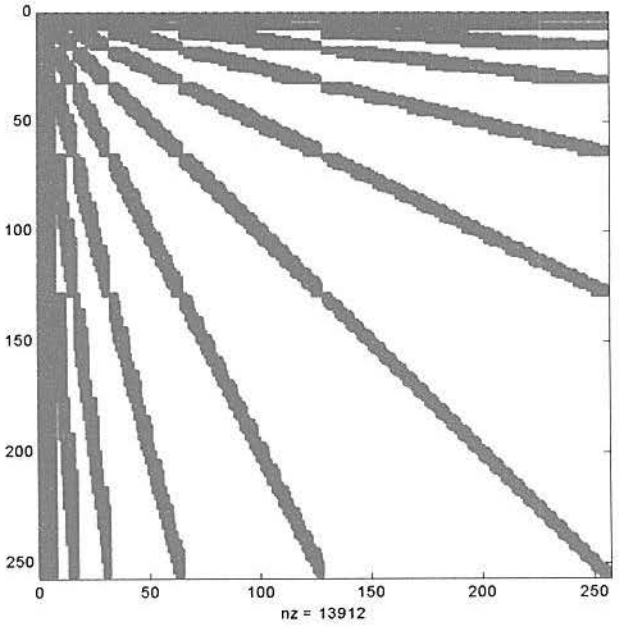

Fig. 3.4 – Estrutura da matriz de A, caso unidimensional, após o precondicionamento para N = 256

e $$\lambda_{\xi\eta}^{j} = 2^{2(n-j)} \sum_{k} \left(p_k e^{-ik\xi} + q_k e^{-ik\eta} \right),$$

respectivamente, onde

$$p_k = \int_{\Omega} \varphi(x-k) \frac{d^2}{dx^2} \varphi(x) dx \quad \text{e} \quad q_k = \int_{\Omega} \psi(x-k) \frac{d^2}{dx^2} \psi(x) dx.$$

Como no caso unidimensional, o método da diagonal para mudança de escala é efetuada por uma transformação nos coeficientes dos filtros em cada coordenada por

$$\left(h_i^{'}\right) = \sqrt{2}\left(h_i\right) \quad \text{e} \quad \left(g_i^{'}\right) = \sqrt{2}\left(g_i\right).$$

Uma vantagem dos métodos de diferenças finitas é a esparsidade das matrizes oriundas dos problemas discretizados. À primeira vista, pode parecer que o precondicionamento destrói a esparsidade da matriz original. No entanto muitos autores afirmam que a matriz precondicionada é esparsa. Em [21] temos uma prova sobre a esparsidade da matriz precondicionada (cf. Fig. 3.5 e 3.6). É um fato que a esparsidade aumenta com a ordem da matriz.

Computamos alguns resultados, que apresentamos nas Tabelas 3.1 e 3.2, onde podemos observar que a esparsidade da matriz dos coeficientes, após o precondicionamento, aumenta com a ordem da matriz.

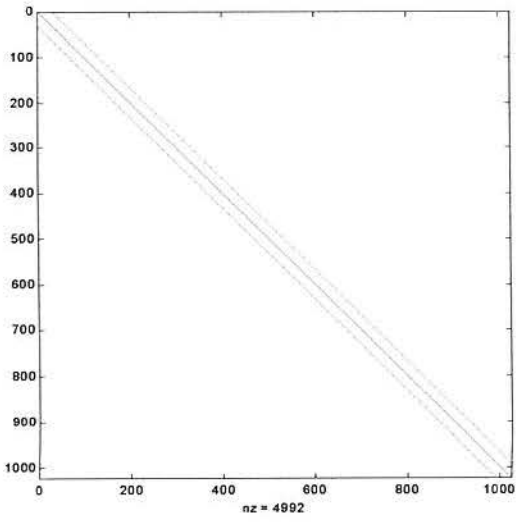

Fig.3.5 – Estrutura da matriz A, caso bidimensional, antes do precondicionamento para uma matriz de ordem N^2 =1024

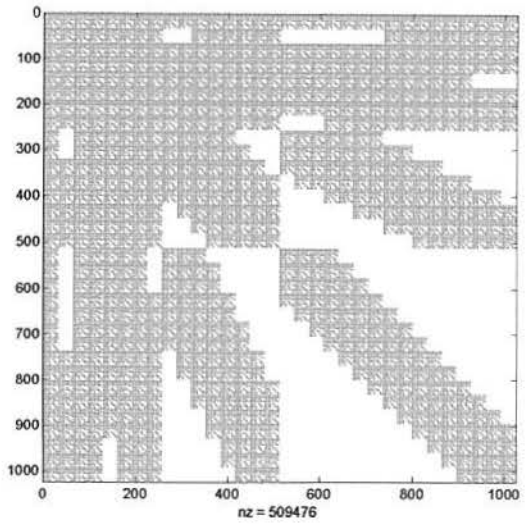

Fig. 3.6 – Estrutura da matriz A com ordem N^2 =1024, caso bidimensional, após o precondicionamento

Tabela 3.1 – Esparsidade no caso unidimensional

Ordem da matriz	Percentual de zeros	Percentual de coeficientes $< 10^{-3}$	Percentual de coeficientes $< 10^{-2}$
16 x 16	9,38	9,38	14,06
32 x 32	27,34	29,10	35,35
64 x 64	48,05	50,05	56,84
128 x 128	65,82	67,77	73,38
256 x 256	78,77	80,45	84,44
512 x 512	87,33	88,66	91,24
1024 x 1024	92,65	93,64	95,22
2048 x 2048	95,82	96,52	97,46

Tabela 3.2 – Esparsidade no caso bidimensional

Ordem da matriz	Percentual de zeros	Percentual de coeficientes $< 10^{-3}$	Percentual de coeficientes $< 10^{-2}$
16 x 16	14,45	25,00	25,00
32 x 32	5,76	37,89	41,41
64 x 64	9,06	54,00	57,13
128 x 128	13,96	65,78	70,69
256 x 256	24,60	77,27	81,70
512 x 512	37,65	85,29	88,53
1024 x 1024	51,41	91,42	93,47
2048 x 2048	64,16	95,05	96,30

3.4 Condições de fronteira

Em nossos experimentos utilizamos, para o caso unidimensional, condições de fronteira de Dirichlet, Neumann e mistas. Para o caso bidimensional, utilizamos apenas condições de Dirichlet. A Tabela 3.3 apresenta uma descrição de cada uma dessas condições. Como explicaremos na secção 3.4.1, das condições de fronteira dependem, na matriz da discretização, os valores de a e b nas posições (1, 1) e (N, N):

$$\begin{bmatrix} a & & \\ & \ddots & \\ & & b \end{bmatrix}.$$

Tabela 3.3 – Condições de Fronteira

Tipo	Descrição	Exemplos
Dirichlet	É conhecido o valor da função f na fronteira	$u(0) = 0$ e $u(1) = 0$
Neumann	É conhecido o valor da derivada da função f	$u'(0) = 0$ e $u'(0) = 0$
Mistas	É conhecida uma relação entre o valor da função f e de sua derivada	$u'(0) + \alpha u(0) = \beta$ e $u'(1) + \alpha u(1) = \beta$

3.4.1 Condições de fronteira de Dirichlet

3.4.1.1 Caso unidimensional. Voltemos às diferenças centrais que produzem a discretização da equação diferencial, $-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i + h^2 \tau_i$, ou, ignorando τ,

$$-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i. \tag{3.18}$$

Seja $u(0) = u(1) = 0$ (Dirichlet), o mesmo que $u_0 = u_{N+1} = 0$. Para $i = 1$, a equação (3.18) produz

$$2u_1 - u_2 = h^2 f_1,$$

e, para $i = N$, a equação (3.18) implica

$$-u_{N-1} + 2u_N = h^2 f_N.$$

Logo $a = b = 2$. Obtemos os mesmos valores de a e b com as condições de fronteira $u(0) = u(1) = 1$ (ainda Dirichlet), ou seja $u_0 = u_{N+1} = 1$, como é fácil ver com apoio em (3.18).

3.4.1.2 Caso bidimensional. A forma discretizada do problema de Poisson bidimensional é

$$4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{ij}. \tag{3.19}$$

Considerando $u(x, y)$ constante na fronteira de $[0 ; 1] \times [0 ; 1]$, procedendo como no caso unidimensional e usando (3.19), com $i = j = 1$ obtemos a equação $4u_{11} - u_{21} - u_{12} =$ constante, e com $i = j = N$ resulta uma equação da forma $4u_{NN} - u_{N-1,N} - u_{N,N-1} =$ constante. Concluímos que $a = b = 4$.

3.4.2 Condições de fronteira de Neumann

3.4.2.1 Caso unidimensional. As condições unidimensionais de fronteira de Neumann podem ser $u'(0) = u'(1) = 0$. Usando a aproximação da derivada $u'(1)$,

$$u'(1) \approx \frac{u_2 - u_0}{2h} = 0, \tag{3.20}$$

vem $u_0 = u_2$. Então, para $i = 1$, a equação (3.18) se torna

$$u_1 - u_2 = \frac{h^2 f_1}{2}.$$

Com $i = N$, escrevemos a equação correspondente à equação (3.20), que acarreta a igualdade $u_{N-1} = u_{N+1}$, e esta, em vista da (3.18), nos leva à equação

$$-u_{N-1} + u_N = \frac{h^2 f_N}{2}.$$

Para as condições de Neumann acima, então, $a = b = 1$.

Se as condições de fronteira de Neumann forem $u'(0) = u'(1) = 1$, seguindo os mesmos passos acima, obtemos os mesmos valores para a e b.

3.4.3 Condições de Fronteira Mistas

3.4.3.1 Caso unidimensional. Para condições mistas, conforme descrito em [28], teremos $a = (1 + h\alpha)$ e $b = (1 + h\beta)$, onde h é o passo. Adotando $\alpha = 1/h$ e $\beta = 0$, teremos os valores $a = 2$ e $b = 1$.

3.5 Resultados Numéricos

Nesta secção mostramos os resultados numéricos da implementação do método proposto. O objetivo principal dos experimentos numéricos é verificar se realmente o número de condição da matriz **A** dos coeficientes do sistema linear (3.1) se torna limitado, independentemente do tamanho do problema, ou seja, não varia muito com o número de pontos da malha de discretização. Também verificamos, através de experimentos, se a convergência do MGC se torna mais rápida mediante o precondicionamento baseado em wavelets.

3.5.1 Caso unidimensional. Os resultados para o caso unidimensional foram calculados em um computador Pentium III, com processador INTEL de 900 MHz e 256 Mb de RAM.

3.5.1.1 Número de condição. As Tabelas 3.4, 3.5 e 3.6 e a Fig. 3.7 comparam o número de condição $\kappa(\mathbf{A})$ da matriz **A** com o número de condição $\kappa(\mathbf{C})$ da matriz **C**, onde $\mathbf{C} := \mathbf{P}_J \mathbf{W}_{(J)} \mathbf{A} \mathbf{W}_{(J)}^T \mathbf{P}_J$ é a matriz **A** precondicionada pela TWDi, para os casos Dirichlet $(a = b = 2)$, Neumann $(a = b = 1)$ e misto $(a = 2 \text{ e } b = 1)$, respectivamente. Foram usadas wavelets ortonormais de Daubechies com 3 momentos nulos e $J = n$. Como critério de parada, usamos

$$\|\mathbf{Ax} - \mathbf{b}\| \leq 10^{-10} \|\mathbf{b}\|.$$

Para o cálculo do número de condição utilizamos o programa `cond.m` do MATLAB e também a fórmula a seguir, que pode ser encontrada em [13,36]

$$\kappa(\mathbf{A}) = \frac{1+\cos\left(\frac{\pi}{(n+1)}\right)}{1-\cos\left(\frac{\pi}{(n+1)}\right)}.$$

Tabela 3.1 – Número de condição da matriz A com e sem precondicionamento, para o problema de Poisson unidimensional com condições de fronteira de Dirichlet (a = b = 2)

N	κ(A)	κ(C)
16	116,46	13,78
32	440,69	26,99
64	1 711,70	53,72
128	6 743,70	107,63
256	26 768,00	216,03
512	106 660,00	433,60
1024	425 800,00	869,65
2048	1 701 500,00	1 742,90

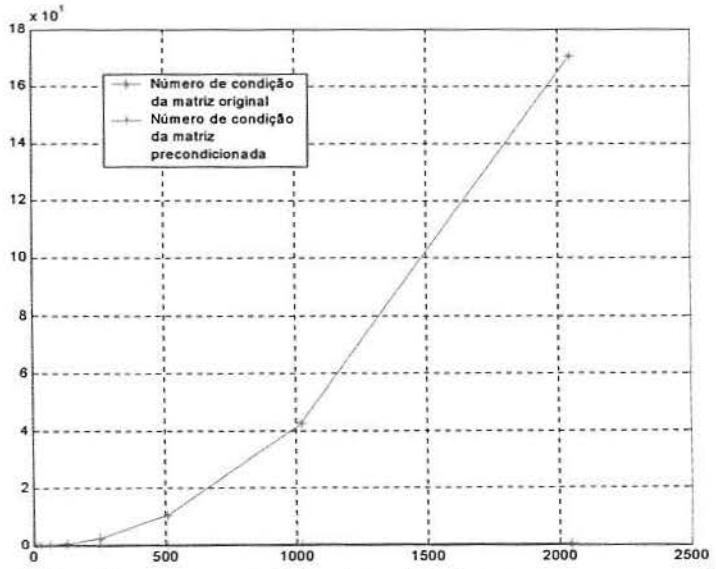

Fig. 3.7 – Número de condição da matriz A, com e sem precondicionamento, para o problema de Poisson unidimensional com condições de fronteira de Dirichlet (a = b = 2)

Para o caso Neumann, usamos como parâmetro de comparação, a razão do maior autovalor e o segundo menor autovalor de **A**, porque o menor autovalor é zero; a matriz é positiva semidefinida. Embora essa razão não seja o verdadeiro número de condição, que, no caso é definido como sendo ∞, presta-se muito bem como parâmetro de comparação do condicionamento do sistema linear.

Tabela 3.5 – Número de condição da Matriz A com e sem precondicionamento, para o problema de Poisson unidimensional com condições de fronteira de Neumann (a = b = 1)

N	κ(A)	κ(C)
16	103,09	19,68
32	414,34	28,90
64	1 659,40	37,59
128	6 639,50	45,32
256	26 560,00	51,99
512	106 240,00	57,74
1024	424 970,00	62,70

Tabela 3.6 – Comparação do número de condição, com e sem precondicionamento, para o problema de Poisson unidimensional com condições de fronteira mistas (a =2 e b = 1)

N	κ(A)	κ(C)
16	437,70	26,58
32	1 708,70	33,96
64	6 740,70	40,92
128	26 765,00	47,37
256	106 650,00	53,20
512	425 800,00	58,43
1024	1 701 500,00	63,09

3.5.1.2 Comparação do MGC precondicionado pela TWDi com o MGC e SOR com parâmetro ótimo. As Tabelas 3.7 e 3.8 e a Fig. 3.8 comparam o método em estudo, isto é, do gradiente conjugado precondicionado pela TWDi, com o MGC sem precondicionamento, e com o método SOR com parâmetro ótimo com respeito ao número de iterações.

Usamos wavelets ortonormais de Daubechies com 3 momentos nulos e número de níveis $J = n$. Como critério de parada, adotamos

$$\|\mathbf{Ax} - \mathbf{b}\| \leq 10^{-10}\mathbf{b}. \tag{3.21}$$

Para executar os cálculos, utilizamos o programa `idwts.m`, que automatiza os 3 métodos com recurso das sub-rotinas do `CG.m` do MATLAB, do `jasor.m` [08] e do `contrgh.m`, construídos por nós (cf. Apêndice B).

O número de iterações para a ordem 2048 da matriz **A** e método SOR foi colocado em vermelho pelo seguinte: constatamos que o critério de parada (3.21) que adotamos não é atingido nunca, o que deve ocorrer porque o erro de arredondamento do computador supera, em cada iteração, a tolerância sobre o resíduo e as iterações entram em círculo; contudo para a tolerância $10^{-9}\|\mathbf{b}\|$, o computador nos apresenta aquele número de iterações, que serve para nos dar uma boa idéia do desempenho do SOR.

Tabela 3.7 – Comparação dos métodos GC, SOR e GC precondicionado pela TWDi, no problema de Poisson unidimensional, com condições de fronteira de Dirichlet (a = b = 2)

N	Número de Iterações		
	GC	SOR	TWDi
16	8	75	16
32	16	147	25
64	32	289	33
128	64	574	39
256	128	1144	46
512	256	2286	51
1024	512	4962	56
2048	1024	8411	62

Da mesma forma que para o caso da fronteira de Dirichlet, os resultados na Tabela 3.8 mostram que o método da TWDi supera os outros dois também com fronteira mista. Os dois números em vermelho nessa tabela têm o mesmo significado que para o caso Dirirchlet.

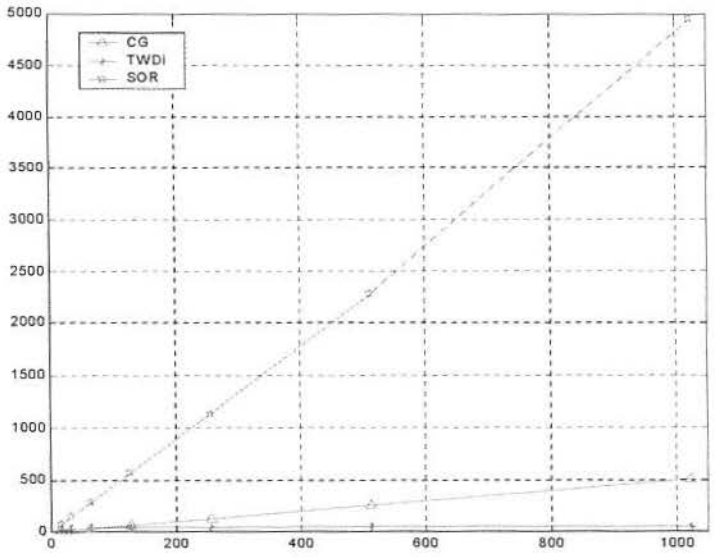

Fig. 3.8 – Comparação dos métodos GC, SOR e GC precondicionado pela TWDi, no problema de Poisson unidimensional com condições de fronteira de Dirichlet (a = b = 2)

Tabela 3.1 – Comparação dos métodos GC, SOR e GC precondicionado pela TWDi, no problema de Poisson unidimensional, com condições de fronteira mistas (a = 3, b = 1).

N	Número de Iterações		
	GC	SOR	TWDi
16	16	142	16
32	32	284	26
64	64	571	38
128	128	1153	46
256	256	2329	55
512	512	5279	62
1024	1024	8828	70
2048	2048	63563	77

3.5.2 Caso bidimensional. Para o caso bidimensional os resultados foram obtidos em um computador Pentium IV, com processador INTEL com 1.7 GHz de velocidade e 256 Mb de RAM.

3.5.2.1 Número de condição. A Tabela 3.9 e a Fig.3.9 comparam o número de condição $\kappa(\mathbf{A})$ da matriz $\mathbf{A}$ com o número de condição $\kappa(\mathbf{C})$ da matriz $\mathbf{C} := \mathbf{P}_J \mathbf{W}_{(J)} \mathbf{A} \mathbf{W}_{(J)}^t \mathbf{P}_J$, para o caso de Dirichlet $(a = b = 2)$. Usamos wavelets ortonormais de Daubechies com 3 momentos nulos e número de níveis $J = n$.

Tabela 3.9 – Número de condição da matriz A, com e sem precondicionamento com a TWDi, para o Problema de Poisson bidimensional com condições de fronteira de Dirichlet (a = b = 4)

N	$\kappa(\mathbf{A})$	$\kappa(\mathbf{C})$
8	4,79	5,01
16	9,47	10,22
32	14,92	7,39
64	32,16	14,32
128	50,72	10,05
256	116,46	19,57
512	184,57	16,56
1024	440,69	31,50
2048	701,27	33,10

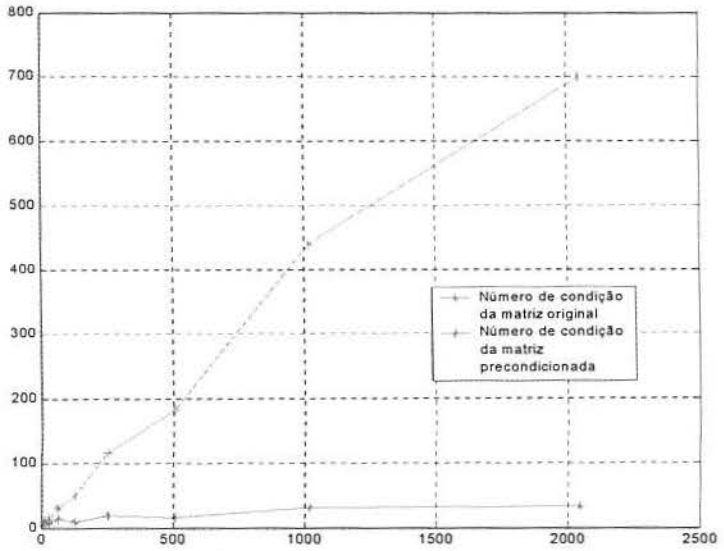

Fig. 3.9 – Número de condição da matriz A, com e sem precondicionamento com a TWDi, para o Problema de Poisson bidimensional com condições de fronteira de Dirichlet (a = b = 4)

3.5.2.2 Comparação do MGC precondicionado pela TWDi com os métodos GC, SOR e GCCi. A Tabela 3.10 e a Fig. 3.10 comparam o método em estudo, MGC com precondicionamento com TWDi, com o MGC sem precondicionamento, com o método SOR com parâmetro ótimo, e com MGC precondicionado com o uso da fatoração incompleta de Cholesky (ICGC), em relação ao número de iterações, para o caso das condições de fronteira de Dirichlet ($a = b = 4$). Usamos wavelets ortonormais de Daubechies com 3 momentos nulos e $J = n$. Como critério de parada aqui também é o (3.21).

Para a execução dos cálculos utilizamos os programas `pcg.m` e `cholinc.m` do MATLAB, sendo que o primeiro foi modificado por nós para adequação às sub-rotinas `precond2D.m` e `precond2Dt.m`, construídas por nós, bem como o algoritmo `jasor.m` com `poisparot.m` e, ainda, `poissonbi.m` (cf. Apêndice B).

A Fig. 3.10 ilustra os resultados apresentados na Tabela 3.10, que confirmam a eficiência do método TWDi para problemas de altas ordens. Observamos que, com relação ao número de iterações, o TWDi supera o método SOR para todas as ordens, supera o MGC a partir da ordem 256×256 e, a partir de 4096×4096, passa a superar também o GCCi, o que demonstra que, para problemas de altas ordens, o método TWDi é mais eficiente do que os demais.

Tabela 3.10 – Comparação dos métodos GC, SOR, GCCi e GC precondicionado pela TWDi, para o problema de Poisson com condições de fronteira de Dirichlet (a = b = 4)

N	GC	SOR	ICGC	TWDi
	Iterações	iterações	iterações	iterações
16	3	16	8	10
32	8	22	9	18
64	10	32	11	21
128	25	40	14	25
256	28	58	17	28
512	51	74	24	34
1024	59	109	29	37
2048	104	139	44	50
4096	119	208	52	53
8192	208	266	81	72
16384	239	398	100	78
32768	417	511	144	105
65536	470	759	470	113

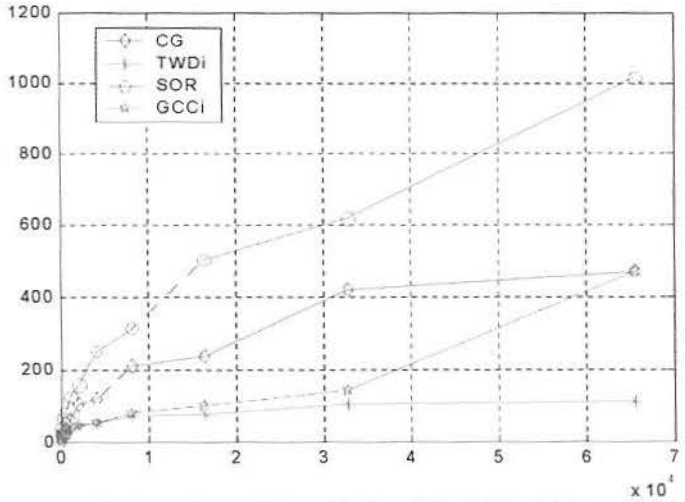

Fig. 3.10 – Comparação dos métodos GC, SOR, GCCi e GC precondicionado pela TWDi, para o problema de Poisson com condições de fronteirade Dirichlet (a = b = 4)

Conclusão

Apresentamos ao longo desse trabalho a descrição da resolução de sistemas lineares oriundos da discretização das equações de Poisson unidimensional e bidimensional, propondo a TWDi para construir uma matriz precondicionadora. Para a resolução desses sistemas, utilizamos o MGC e estudamos alguns aspectos da matriz dos coeficientes precondicionada pela TWDi: sua esparsidade, seu número de condição e, relacionado com isso, o número de iterações necessárias para a convergência do MGCP. Os resultados obtidos em nossos experimentos evidenciam que a esparsidade da matriz dos coeficientes precondicionada aumenta com a ordem da matriz, tanto no caso unidimensional quanto no caso bidimensional, o número de condição dessa matriz fica limitado e, o método proposto diminui o número de iterações necessárias para a convergência do MGC. Comparamos os resultados obtidos com métodos clássicos, como o MGC básico, SOR e GCCi e verificamos que o novo método, de modo geral, supera a todos, principalmente no caso de sistemas de altas ordens.

Há, ainda, muito a ser abordado, apontando para estudos futuros, pois, a otimização do algoritmo do GC por meio do precondicionamento é uma das técnicas que podem ser usadas para controlar o crescimento do custo computacional. Essa técnica, como já comentamos, melhora a taxa de convergência do método, diminuindo assim o número de iterações necessárias para o cálculo da solução com uma tolerância preestabeleciada. Outra técnica que vêm sendo estudada com o mesmo objetivo é usar o processamento paralelo para a execução do MGC precondicionado com a TWDi [36].

REFERÊNCIAS BIBLIOGRÁFICAS

[01] ASEKA, I. B. – *Aproximações multi-resolução e bases ortonormais wavelet de* $L^2(\mathbb{R})$, Dissertação de Mestrado, Florianópolis, 1995.

[02] BACRY, E. & MALLAT, S., & PAPANICOLAOU, G. – *A wavelet based space-time adaptative numerical method for partial differential equations*, Mathematical Modelling and Numerical Analysis, 26, 793-834, 1992.

[03] BEYLKIN, G. – *On wavelet-based algorithms for solving differential equations*, 1993, ftp://amath.colorado.edu/pub/wavelets/papers/index.html.

[04] BEYLKIN, G. – *Wavelets and fast numerical algorithms*, Proceedings of Symposia in Applied Mathematics, 47, 89-117, 1993, ftp://amath.colorado.edu/pub/wavelets/papers/index.html

[05] BEYLKIN, G. & KEISER, J. M. – *On the adaptive numerical solution of nonlinear partial differential equations in wavelet bases*, Journal of Computational Physics, 132, 233-259, 1997, ftp://amath.colorado.edu/pub/wavelets/papers/index.html

[06] BURRUS, C. & SIDNEY, G. & RAMESH, A. & GUO, H. – *Introduction to wavelets and wavelets transforms: a primer*, Prentice-Hall, 1998.

[07] BURKE-HUBBARD, B. – *The world according to wavelets*, A. K. Peters, Wellesley Mass, 1996.

[08] CALEFFI, J. – *Otimização do método SOR para matrizes p-cíclicas consistentemente ordenadas*, Dissertação de Mestrado, Porto Alegre, 2000.

[09] CHAPLAIS, F. – *A wavelet tour of signal processing by Stéphane Mallat – a short presentation*, 1998, http://cas.ensmp.fr/^chaplais/Wavetour_presentation/Wavetour_presentation_US.html

[10] CHEN, K. – *Discrete wavelet transforms accelerated sparse preconditioners for dense boundary element system*, Electronic Transactions on Numerical Analysis, 8, 138-153, 1999.

[11] DAUBECHIES, I. – *Ten lectures on wavelets*, Philadelphia, PA: SIAM\ Books, 1992.

[12] DAUBECHIES, I. – *Where do wavelets come from? - a personal point of view*, Proceedings of de IEEE, 84(4), 510-513, 1996, http://www.princeton.edu/~icd/publications/

[13] DEMMEL, J. W. – *Applied Numerical Linear Algebra* – SIAM, Philadelphia, 1997.

[14] DEVORE, R. A. & LUCIER, B. J. – *Wavelets Paper*, Acta Numerica, A. Iserles, Cambridge University Press, v.1, pp.1-56, 1992.

[15] GLOWINSKI, R. et AL. – *Wavelet solution of linear and non-linear elliptic, parabolic and hyperbolic problems in one space dimension*, Computing Methods in Applied Science and Engeneering, R. Glowinski (ed.), SIAM, Philadelphia, PA, 1990.

[16] FRAZIER, M. W. – *An introduction to wavelets through linear algebra*, Springer, 1999.

[17] GOMES, J. & VELHO, L. – *From Fourier Analysis to wavelets*, Course notes Siggraph 98. IMPA, 1998.

[18] GOLUB, G. H. & Van Loan, C. F. – *Matriz Computations*, The John Hopkins University Press, Baltimore, 1996.

[19] GRAPS, A. –*An introduction to wavelets*, IEEE Comnputational Science and Engineering, 2(2), 1995, http://www.amara.com/

[20] HUBBARD, B. B. – *The world according to wavelets: the story of a mathematical technique in the making*, 1998.

[21] JAFFARD, S. – *Wavelet methods for fast resolution of elliptic problems*, SIAM Journal on Numerical Analysis, 29(4), 965-986, 1992.

[22] JAMESON, L. – *On the wavelet optimised finite difference method*, http://www.mathsoft.com/wavelets.html

[23] JAWERT, B. & SWELDENS, W. – *An overview of wavelet base multiresolution analysis*, SIAM\ Rev., 36(3), 377-412, 1994, http://www.math.ntu.edu.tw/~yccen/wavelet/sweldens/sweldens.html

[24] JAWERT, B & SWELDENS, W. – *Wavelet multiresolution analyses adapted for the fast solution of boundary value ordinary differential equations*, Sixt Copper Mountain Conference on Multigrid Methods, NASA Conference Publication 3224, 259-273, 1993.

[25] KELLEY, C. T. – *Iterative methods for linear and non-linear equations*. Frontiers in applied mathematics vol.16.SIAM, 1995.

[26] MARCUS, M. – *Matrices and MATLAB a tutorial*. Prentice-Hall, New Jersey, 1993.

[27] MATHEWS, J. H. *Numerical methods for mathematics, science and engineering*, Prentice Hall, New Jersey, 1992.

[28] NAKAMURA, S. – *Numerical Analysis and Graphic Visualization with MATLAB*, Prentice Hall, Inc, Upper Saddle River, NJ, 1996.

[29] POLIKAR, R. – *The Engineer's Ultimate Guide to Wavelet Analysis – The wavelet tutorial*, 1996. http://www.public.iastate.edu/~rpolikar/WAVELETS/waveletindex.html

[30] POULARIKAS, A. – *The transforms and applications Handbook*, CRC Press Boca Raton Flórida, 1996.

[31] PRESS, WILLIAM H. ET AL – *Numerical recipes in Fortran 77 - The art of scientific computing* Second Edition, Cambridge, 1996.

[32] STRANG, G. & NGUYEN, T. – *Wavelets and filter banks*. Wellesley, MA, Wellesley-Cambridge Press, 1996.

[33] STUART, R. D. – *Introduccion al Analisis de Fourier*, UTEHA, México, 1965.

[34] SWELDENS, W. – *Wavelets: What next?* http://www.math.ntu.edu.tw/~yccen/wavelet/sweldens/sweldens.html

[35] SWELDENS, W. – *The construction and application of wavelets in numerical analysis*, Katholieke Universiteit Leuven, Belgium, 1994, http://www.math.ntu.edu.tw/~yccen/wavelet/sweldens/sweldens.html

[36] TANAKA, N. & TERASAKA, H. & SHIMIZU, T. & TAKIGAWA, Y. – *Incomplete discrete wavelet transform and applications to a Poisson equation solver*, Journal of Nuclear Science and Technology, 33(7) 555-561, 1996.

[37] TANAKA, N. – *Parallel processing of Haar-wavelet-preconditioned conjugate gradient methods*, Japan Society for Computational Engineering and Science, 2001, http://www.mech.ibaraki.ac.jp/~tanaka/

[38] UYTTERHOEVEN, G. – *Wavelets: software and applications*, Tese de Doutorado, Katholieke Universiteit Lewen, Heverlle, Belgium, 1999, http://home.tvd.be/cr26864/

[39] VALENS, C. – *A really friendly guide to wavelets, Tutorial*, http://www.perso.wanadoo.fr/polyvalens/clemens/wavelets/wavelets.html

[40] TREFETHEN, LLOYD N, & BAU III, DAVID – *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[41] VARGAS, R. S. – *Matrix Iterative Analysis*, Springer, Berlin, 2000.

[42] WOJTASZCZYK, P. – *A mathematical introduction to wavelets*, Cambridge University Press, Cambridge, 1997.

a.1 Método do Gradiente Conjugado

O Método do Gradiente Conjugado (MGC) foi desenvolvido nos anos 50 por Hestenes e Stiefel. Embora teoricamente seja considerado um método direto, nos últimos 20 anos tem sido amplamente usado como um método iterativo para resolver sistemas lineares grandes e esparsos, cuja matriz dos coeficientes é simétrica positiva definida (SPD), superando as famílias dos métodos de Jacobi-Gauss-Seidel-SOR.

Lembremos que uma matriz $\mathbf{A}$ é simétrica se $\mathbf{A} = \mathbf{A}^t$, e positiva definida se $\mathbf{x}^t\mathbf{A}\mathbf{x} > \mathbf{0}$, para todo $\mathbf{x} \neq \mathbf{0}$. Quando $\mathbf{A}$ é SPD, podemos definir a norma

$$\|\mathbf{x}\|_A := \sqrt{\mathbf{x}^t\mathbf{A}\mathbf{x}}.$$

O MGC determina uma solução numérica $\mathbf{x}$ de um sistema de n equações simultâneas,

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A} \in \mathbb{R}^{n\times n}, \quad \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{b} \in \mathbb{R}^n, \tag{a.0}$$

por meio de sucessivos cálculos da melhor tentativa no espaço n-dimensional, que minimiza a função erro.

Seja $\mathbf{x}_k$ uma solução-teste do sistema (a.0). O vetor-resíduo associado é

$$\mathbf{r}_k := \mathbf{b} - \mathbf{A}\mathbf{x}_k. \tag{a.1}$$

Se $\mathbf{A}$ é SPD, então também sua inversa $\mathbf{A}^{-1}$ é SPD, e a função erro é

$$h := \mathbf{r}_k^t\mathbf{A}^{-1}\mathbf{r}_k > 0, \quad \text{para} \quad \mathbf{x}_k \neq \mathbf{0}. \tag{a.2}$$

Substituindo $\mathbf{r}_k$, dado por (a.1), em (a.2), obtemos

$$h = \mathbf{x}_k^t\mathbf{A}\mathbf{x}_k - 2\mathbf{b}^t\mathbf{x}_k + \mathbf{b}^t\mathbf{A}^{-1}\mathbf{b}. \tag{a.3}$$

Dada uma aproximação $\mathbf{x}_k$ da solução de $\mathbf{Ax} = \mathbf{b}$, encontramos uma aproximação melhor,

$$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha \mathbf{p}_k,$$

onde $\mathbf{p}_k$ é um vetor de $\mathbb{R}^n$, dito vetor de direção, escolhido convenientemente, e α é um escalar. A substituição de $\mathbf{x}_k$ por $\mathbf{x}_{k+1}$ em (a.3) produz

$$h = \alpha^2 \mathbf{p}_k^t \mathbf{A} \mathbf{p}_k - 2\alpha \mathbf{p}_k^t \mathbf{r}_k + \mathbf{x}_k^t \mathbf{A} \mathbf{x}_k + \mathbf{b}^t \mathbf{A}^{-1} \mathbf{b}.$$

Um mínimo local de h (como função de α) pode ser determinado igualando a primeira derivada a zero, isto é,

$$\frac{dh}{d\alpha} = 2\alpha \mathbf{p}_k^t \mathbf{A} \mathbf{p}_k - 2\mathbf{p}_k^t \mathbf{r}_k = \mathbf{0}. \qquad \text{(a.4)}$$

É imediato verificar que a solução

$$\alpha_k = \frac{\mathbf{p}_k^t \mathbf{r}_k}{\mathbf{p}_k^t \mathbf{A} \mathbf{p}_k}$$

da equação (a.4) é o único ponto de mínimo (global) de h. Usamos o correspondente mínimo como a nova aproximação $\mathbf{x}_{k+1}$ da solução de $\mathbf{Ax} = \mathbf{b}$ no passo seguinte, isto é,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k,$$

No MGC, o vetor $\mathbf{p}_k$ é escolhido de forma que tenha a direção que minimize a função erro h tanto quanto possível, e ao mesmo tempo satisfaça a condição

$$\mathbf{p}_i^t \mathbf{A} \mathbf{p}_j = 0, \quad \text{para} \quad i \neq j.$$

O algoritmo para o MGC é descrito no quadro abaixo. Nesse algoritmo, $\langle \cdot , \cdot \rangle$ é a função produto interno usual em $\mathbb{R}^n$. Detalhes adicionais sobre o método podem ser encontrado em [13, 25,40].

MGC

Inicialização

(a) valor inicial $\mathbf{x}_0$

(b) $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

(c) $\mathbf{p}_1 = \mathbf{r}_0$

Processo Iterativo

(d) $\alpha_k = \dfrac{\langle \mathbf{p}_k, \mathbf{r}_k \rangle}{\langle \mathbf{p}_k, \mathbf{A}\mathbf{p}_k \rangle}$

(e) $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$

(f) $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$

(g) Confere a convergência

(h) $\beta_k = \dfrac{\langle \mathbf{r}_{k+1}, \mathbf{r}_{k+1} \rangle}{\langle \mathbf{r}_k, \mathbf{r}_k \rangle}$

(i) $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$

a.2 Método do Gradiente Conjugado Precondicionado

Quanto **A** possui um número de condição grande, o precondicionamento surge como forma de determinar uma aproximação numérica da solução **x** de um sistema de *n* equações simultâneas (a.0), através da modificação do sistema original, por meio de transformações lineares, tornando-o mais fácil de resolver, isto é, multiplicando ambos os lados do sistema por uma *matriz precondicionadora* **K**. A matriz **K** deve ser SPD e tal que reduza muito o número de condição de **A**,

$$\kappa\|\mathbf{A}\| \gg \kappa\|\mathbf{K}\mathbf{A}\|.$$

A notação $\gg$ significa *muito maior que*. Como **K** é SPD, podemos usar a fatoração de Cholesky [13]

$$\mathbf{K} = \mathbf{V}^T\mathbf{V}.$$

Então o sistema $\mathbf{Ax} = \mathbf{b}$ é transformado pela matriz $\mathbf{V}$ em

$$\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}},$$

com $\hat{\mathbf{A}} := \mathbf{VAV}^t$, $\hat{\mathbf{x}} := \mathbf{V}^{-t}\mathbf{x}$ e $\hat{\mathbf{b}} := \mathbf{Vb}$. Para resolver o novo sistema linear, apenas aplicamos a ele o algoritmo do MGC acima. Um resumo do algoritmo para o MGCP é dado a seguir.

MGCP

Inicialização

(a) valor inicial $\mathbf{u}_0$

(b) $\mathbf{r}_0 = \mathbf{f} - \mathbf{Au}_0$

(c) $\mathbf{p}_0 = \mathbf{r}_0$

Processo Iterativo

(d) $\alpha_k = \dfrac{\langle \vec{\mathbf{r}}_k, \mathbf{r}_k \rangle}{\langle \mathbf{p}_k, \mathbf{Ap}_k \rangle}$

(e) $\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k \mathbf{p}_k$

(f) $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{Ap}_k$

(g) Confere a convergência

(h) $\vec{\mathbf{r}}_{k+1} = \mathbf{Kr}_{k+1}$

(i) $\beta_k = \dfrac{\langle \vec{\mathbf{r}}_{k+1}, \mathbf{r}_{k+1} \rangle}{\langle \vec{\mathbf{r}}_k, \mathbf{r}_k \rangle}$

(j) $\mathbf{p}_{k+1} = \vec{\mathbf{r}}_{k+1} + \beta_k \mathbf{p}_k$

IMPLEMENTAÇÃO DE ALGORITMOS

Reunimos neste apêndice os algoritmos por nós desenvolvidos na linguagem MATLAB para a realização deste trabalho. Nesses algoritmos foram utilizadas sub-rotinas do MATLAB como `cg.m`, `pcg.m`, `dbaux.m`, `kron.m`, `diagmx.m`, e o programa `jasor.m`, desenvolvido por Caleffi [08].

Rotina principal: `idwts.m`

```
function [x,t,iter,y,z]=idwts(n,m,a,b,vetorb,p)

%-------------------------------------------------------------
% IDWTS resolve o sistema linear Ax = b como segue.
%
% Entrada:
% n indica que 2^n é a ordem do sistema linear;
% m é o número de momentos nulos;
% a e b são números correspondente às condições de fronteira
% usadas no problema de Poisson (Dirichlet, Newmann ou mis-
%     tas)
% vetorb é o vetor dos termos independentes.
%% p = 1, para o GC sem precondicionamento;
% p = 2, para o GC precondicionado com wavelets;
% p = 3, para o SOR;
%
% Saída:
% x é vetor solução do sistema;
% t é o tempo de execução do programa;
% iter é o número de iterações realizadas;
% y,z são os números de condição de A e C:=PWAW'P;
% matrizes A e C e geração do número de condição dessas
% matrizes
% -------------------------------------------------------------

xo = zeros(2^n,1);
N = 2^n;
A = full(gallery('tridiag',N));
A(1,1) = a;
A(N,N) = b;
[C,y,z] = conw(n,m,a,A)    %opção completa
```

```
% Resolução do Sistema Linear

% GC sem precondicionamento
if p == 1
   K=eye(2^n);
   1e-10);

% CG precondicionado com wavelets
elseif p==2
   WJ = 1;
   for k = 0:n-1
      [W] = contrgh(n,k,m,2); %f=2 p/evitar o cálculo de P^2
       WJ = W*WJ;
   end
   K = WJ'*WJ;
   [x, error, iter, flag,t]=cg(A,xo,vetorb,K,...
   2^n,1e-10);

% SOR
elseif p == 3
   wo=2/(1+sqrt(1-cos(pi/(N+1))^2));
   [x,error,iter,t,c] = jasor(A,vetorb,xo,wo,2^n,1e-10,2);
end
```

Sub-rotina: conw.m

```
function [C,y,z]=conw(n,m,a,b,A)

%-------------------------------------------------------------
% CONW constrói as matrizes WJ, PJ, C e calcula o número de
% condição de A e C.
%
% Entrada:
% n indica que ordem da matriz A é 2^n;
% m é o número de momentos nulos;
% a e b são os números correspondentes às condições de fron-
%     teira,usadas no problema de Poisson;
% A é a matriz dos coeficientes dos sistema Ax = b.
%
% Saída:
% C é a matriz precondicionadora de A pelo método da TWDi;
% y é o número de condição de A;
% z é o número de condição de C.
%-------------------------------------------------------------

% Construção da matriz WJ=W(n-1)*...*W1*W0
WJ = 1;
for k = 0:n-1
   W = contrgh(n,k,m,1);
```

```
   WJ = W*WJ;
end

% Construção da matriz PJ: 2^n = ordem da matriz e n =
% número de níveis

PJ = matrizp(n,n);

% Construção da matriz C e cálculo dos números de condição
% de A e C

C = PJ*WJ*A*WJ'*PJ;

if a == 1
   a = sort(eig(A));
   y = a(N)/a(2);
   c = sort(eig(C));
   z = c(N)/c(2);
else
   y = cond(A);
   z = cond(C);
end
```

Sub-rotina: contrgh.m

```
function W=contrgh(n,k,m,f)

%-------------------------------------------------------------
% CONTRGH constrói as submatrizes H e G e a matriz W.
%
% Entrada:
% n indica que a ordem de W é 2^n;
% k+1 é o número de níveis de resolução TWD;
% m é número de momentos nulos.
% f é o fator para mudança de escala (evita o cálculo de P);
%
% Saída:
% W = matriz da TWDi
%-------------------------------------------------------------

% Construção da matriz H
N = 2^n;
N1 = 2^(n-k);
N2 = 2^(n-k-1);
L = 2*m;
H = zeros(N2,N1);
h = sqrt(2)*dbaux(m);
i = 1;
for j = 1:2:N1-1
```

```
   for p = 1:L
      if j+p-1 < N1+1
         H(i,j+p-1) = h(p);
      end
   end
   i = i+1;
end

% Construção da matriz G
G = zeros(N2,N1);

% Construção do vetor g
for i = 1:L
   g(i)= (-1)^i*h(i);
   for i = L+1:L+2
      g(i) = 0;
   end
end

i = N2;
for j = N1:-2:1
   for p = 1:L+2
      if j-p+1 > 0
      G(i,j-p+1) = g(p);
   end
end
I = i-1;
end

% Construção da matriz Wj (W)
W = diagmx(f*[H;G],eye(N-N1));
```

Sub-rotina: `matrizp.m`

```
function P = matrizp(n,J)
%-------------------------------------------------------
% MATRIZP constrói a matriz-mudança-de-escala P pelo proces-
% so da diagonal.
%
% Entrada:
% n indica que a ordem da matriz P é 2^n;
% J-1 indica o número de níveis de resoluição da TWD.
%
% Saída:
% P é a matriz diagonal de mudança de escala.
%-------------------------------------------------------

if J == 1
   P = 2*eye(2^n);
```

```
else
   P = 2^J*eye(2^(n-J+1));
end
for m = J-1:-1:1
   P = diagmx(P,2^m*eye(2^(n-m)));
end
```

Sub-rotina: precond2D.m

```
function x = precond2D(x,N)

%-------------------------------------------------------------
% PRECOND2D constrói a matriz WJ=Wn-2*Wn-1*...W1*Wo, do caso
% bidimensional e precondiciona o vetor x, utilizando as
% sub-rotinas CONTRGH e KRON.
%
% Entrada:
% x é o vetor a ser precondicionado pela matriz WJ.
% N é ordem da matriz WJ do caso unidimensional;
%
% Saída:
% vetor x precondicionado pela matriz WJ
%-------------------------------------------------------------

n = log2(sqrt(N));

WJ = x;
f = sqrt(2);
for k = 0:n-2
   W = contrgh(n,k,3,f);
   x = kron(W,W)*(WJ);
end
```

Sub-rotina: precond2Dt.m

```
function xt = precond2Dt(x,N)

%-------------------------------------------------------------
% PRECOND2DT constrói a matriz WJt=Wn-2*Wn-1*...W1*Wo do
% caso bidimensional e precondiciona o vetor x, utilizando a
% sub-rotina CONTRGH.
%
% Entrada:
% xt = vetor a ser precondicionado
% N é ordem da matriz WJt do caso unidimensional;
%
```

```
% Saída:
% vetor xt precondicionado pela matriz WJt
%-------------------------------------------------------

n = log2(sqrt(N));
WJt = x;
f = sqrt(2);
for k=n-2:-1:0
   W=contrgh(n,k,3,f);
   WJt=kron(W',W')*(WJt);
   end
```

Outros programas

poissonbi.m

```
function A = poissonbi(m,n)

%-------------------------------------------------------------
% POISSONBI calcula a matriz de Poisson, que é a matriz dos
% coeficientes do sistema linear, resultante da discretiza-
% ção, com o uso de diferenças centrais, do problema de con-
% torno bidimensional de Poisson no retângulo [0;1]x[0;1],
% com condições de fronteira de Dirichlet. Utiliza a sub-
% rotina KRON.
%
% Entrada:
% m indica o número de nodos equidistantes no 1o intervalo
%     [0;1];
% n indica o número de nodos equidistantes no 2o intervalo
%     [0;1].
%
% Saída:
% A é matriz de Poisson de ordem n*m, na forma esparsa.
%-------------------------------------------------------------
if nargin < 2,
    n = m;
    T = gallery('tridiag',n);
    I = eye(n);
    A = kron(I,T) + kron(T,I);
end

T1 = gallery('tridiag',m);
I1 = eye(m);
T2 = gallery('tridiag',n);
I2 = eye(n);
if m > n,
     A = kron(I1,T2) + kron(T1,I2);
else
     A = kron(I2,T1) + kron(T2,I1);
```

```
end
```

---

poisparot.m

```
function w = poisparot(M,N)

%-------------------------------------------------------------
% POISPAROT calcula o parâmetro ótimo w do SOR para resolver
% o sistema Ax = b, onde A é a matriz de Poisson de ordem
% MxN. Essa matriz A provém da discretização do problema de
% contorno de Poisson sobre um retângulo com as condições de
% fronteira de Dirichlet, tomando M+1 nodos eqüiespaçados
% sobre um eixo e N+1 nodos sobre o outro eixo.
%
%-------------------------------------------------------------

if nargin < 2,
    N = M;
end

w = 4/(2 + sqrt(4 - (cos(pi/(M+1)) + cos(pi/(N+1)))^2));
```
