PROPAGACÃO DE ONDAS EM DOMÍNIOS NÃO LIMITADOS - GALERKIN COM DIRECÕES ALTERNADAS

UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E CIÊNCIA DA COMPUTAÇÃO

UNICAMP

Campinas - São Paulo

Brasil

PROPAGACÃO DE ONDAS EM DOMÍNIOS NÃO LIMITADOS - GALERKIN COM DIRECÕES ALTERNADAS

Este exemplar corresponde a redação final da tese devidamente corrigida e defendida pela Sra. ANDRÉA MARIA PEDROSA VALLI e aprovada pela Comissão Julgadora.

Campinas, 6 de março de 1990.

Profa. Dra. MARIA CRISTINA CUNHA BEZERRA
(Orientadora)

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para obtenção do Título de Mestre em Matemática Aplicada.

Insubordinacao

Questione-se o metodo,
a regra
o convencionado.
Que a ordem da cartilha se corrompa,
e se curvem o juizo e a razao,
que se rompa o que ficou estipulado.
Afinal, nao e preciso que um botao
permaneca o tempo todo abotoado.

Flora Figueiredo

AGRADECIMENTOS

Gostaria de agradecer àqueles que, direta ou indiretamente, colaboraram para a realização deste trabalho :

- à Cristina, minha orientadora, pela confiança depositada e pelo apoio e incentivo durante toda a execução deste trabalho;

- à minha família pelo total apoio demostrado ao longo de minha vida;

- aos colegas do IMECC pela agradável convivência e, em especial, ao Petrônio e ao Edivaldo que muito me auxiliaram na realização deste trabalho;

- aos meus professores do IMECC, em especial, ao Boldrini e ao Zago pelo apoio oferecido;

- às amigas Angela, Ignês, Roseana e Cristina pelos grandes momentos que passamos juntas;

- aos colegas do Departamento de Matemática da UFES

que assumiram as minhas atividades durante o curso de Mestrado;

- à CAPES, via PICD-UFES, pelo apoio financeiro.

ÍNDICE

1 - INTRODUCÃO :

O estudo em simulação numérica de ondas propagando em meios acústicos não limitados tem sido largamente desenvolvido devido a sua relevância em problemas de interesse geofísico. Este fenômeno físico é modelado matematicamente por uma equação diferencial parcial.

Na busca de soluções para esta equação é preciso introduzir fronteiras artificiais por causa da capacidade finita do computador. Com a introdução destas bordas ao modelo são necessárias condições de fronteira para garantir uma solução única. A utilização de condições do tipo Dirichlet ou Neumann geram reflexões na fronteira, fazendo com que as soluções geradas não sejam compativeis com o fenômeno físico simulado.

Existem distintas formas de tratar o problema de reduzir estas reflexões. James Sochack et al [19] sugerem a adição de um choque absorvente na equação de ondas na na região vizinha ao modelo, de modo que condições do tipo Dirichlet ou Neumann podem ser usadas.

Motivados por esta sugestão e vislumbrando uma possível solução do problema acrescentando posteriormente na equação o termo de amortecimento, partimos para o estudo

numérico da equação de ondas restrita a um domínio retangular e com condições de fronteira do tipo Dirichlet.

Na investigação de técnicas numéricas usadas nas aproximações das soluções de equações diferenciais parciais encontramos os métodos variacionais dos Resíduos Ponderados, aplicáveis a uma extensa classe de equações.

Nesta técnica o problema é posto em uma forma variacional equivalente e a solução aproximada é assumida como uma combinação de funções testes. Assim, o problema é substituído por uma sequência de problemas de dimensão finita de resolução automática, gerando funções que supostamente convergem para a equação original.

Na literatura especializada é possível encontrar resultados que fornecem critérios para a avaliação das soluções aproximadas obtidas por distintos métodos variacionais, entre os quais podemos destacar : o método de Galerkin, o método de Rayleigh-Ritz, o método dos Quadrados Mínimos e o método da Colocação.

A meta inicial que adotamos foi estudar com mais cuidado o método de Galerkin juntamente com a técnica dos Elementos Finitos, que usa polinômios por partes como escolha das funções testes.

Apesar da eficiência comprovada deste método, o objetivo deste estudo é auxiliar na compreensão da técnica que

combina o método de Galerkin com Direções Alternadas [06], escolhida para o tratamento numérico do problema.

Esta escolha é devida principalmente aos resultados obtidos por este método em [10], na resolução do problema usando elementos lineares na base dos Elementos Finitos. Segundo Fernandes [10], o método de Galerkin apresenta dificuldades quanto ao espaço de memória disponível, tanto a nível de micro-computadores pessoais tipo IBM-PC quanto no computador VAX/VMS versão 4.5 da UNICAMP; os métodos da Colocação Natural com B-Splines e Colocação Ortogonal nos nós gaussianos da partição apresentaram problemas de instabilidade numérica.

Isto posto, partimos para a implementação computacional do método de Galerkin com Direções Alternadas usando polinômios de Hermite Cúbico na base do Elemento Finito, no intuito de obtermos melhores aproximações.

Em seguida, modificamos adequadamente esta implementação de maneira a incluir o termo de amortecimento na equação, conforme os objetivos iniciais.

Mostramos também os testes realizados e os resultados obtidos, assim como as conclusões e sugestões a trabalhos futuros.

2 - MÉTODO DE GALERKIN

2.1 - INTRODUCÃO

Neste capítulo deduziremos a equação de propagação de ondas, para o problema das vibrações de uma membrana horizontal distendida, e mostraremos as idéias fundamentais sobre como obter aproximações para a solução desta equação, usando o método de Galerkin. O objetivo do estudo deste método é ajudar na compreensão do tratamento numérico aqui usado - que é uma combinação do método de Galerkin com o de Direções Alternadas.

Na seção 2.3 encontra-se a formulação a partir da qual se constroi uma solução discreta bem como os espaços vetoriais necessários à correta definição do problema. As demonstrações das proposições contidas nesta seção estão desenvolvidas em [14]. Em seguida, apresentaremos os métodos de Galerkin contínuo e discreto no tempo, sugeridos em [09], e os espaços de aproximação normalmente utilizados. Para o método de Galerkin discreto no tempo será feita uma estimativa de erro.

2.2 - EQUACÃO DE PROPAGACÃO DE ONDAS

Considere uma membrana horizontal distendida em todas as direções por meio de uma tensão T (por unidade de comprimento). Vamos supor que os pontos desta membrana podem ter somente deslocamentos transversais (verticais) e que, além da tensão T, não haja nenhuma outra força horizontal atuando sobre a membrana. Suponha também que a membrana não ofereça resistência ao deslocamento de modo que o vetor de tensão T é função de x e y.

Seja u(x,y,t) o deslocamento da membrana e C a projeção de certa parte da membrana no plano (x,y) com fronteira S. Para a dedução da equação das oscilações da membrana, aplique o teorema do incremento da quantidade de movimento : a variação da quantidade de movimento é igual ao impulso das componentes verticais das forças de tensão e das forças externas, F(x,y,t). Desta forma, obtemos a equação das oscilações da membrana na forma integral [22] :

$$\int_S \left[\frac{\partial u(u,y,t_2)}{\partial t} - \frac{\partial u(u,y,t_1)}{\partial t} \right] \rho(x,y)\, dxdy =$$

$$= \int_{t_1}^{t_2}\int_C T \frac{\partial u}{\partial n}\, dsdt + \int_{t_1}^{t_2}\int_S F\, dxdydt \qquad (2.1)$$

onde $\rho(x,y)$ é a densidade superficial da membrana e $T\frac{\partial u}{\partial n}$ a

componente vertical da tensão.

Para passar para a equação diferencial da membrana, é preciso supor que a função u(x,y,t) possua derivadas segundas contínuas. Usando o teorma da Divergência [12], a integral de contorno que aparece em (2.1) se transforma em uma integral de superfície, ou seja,

$$\int_C T \frac{\partial u}{\partial n} \, ds = \int_S \nabla.(T\nabla u) \, dxdy.$$

Consequentemente, a equação das oscilações da membrana se reduz a seguinte forma :

$$\int_{t_1}^{t_2} \int_S \left[\rho \frac{\partial^2 u}{\partial t^2} - \nabla.(T\nabla u) - F \right] dxdydt = 0 \qquad (2.2)$$

Aplicando o teorema do valor médio e tendo em conta a arbitrariedade nas escolhas de S e do segmento de tempo (t_1,t_2), podemos concluir que a expressão entre chaves em (2.2) é identicamente nula. Assim, chegamos a equação de propagação de ondas :

$$\rho \frac{\partial^2 u}{\partial t^2} = \nabla.(T\nabla u) - F$$

onde $\rho(x,y)$ é a densidade de superfície da membrana e F(x,y,t)

a densidade das forças externas. No nosso trabalho, estudaremos os casos onde a equação de propagação de ondas pode ser escrita na forma :

$$\frac{\partial^2 u}{\partial t^2} = \nabla.(c\nabla u) - f \qquad (2.3)$$

onde $c(x,y) = T(x,y)/\rho$, com densidade superficial ρ constante, e $f(x,y,t)$ é a densidade de força.

A função $u = u(x,y,t)$ deve satisfazer certas condições no contorno. Portanto, tomaremos a condição de Dirichlet nula, $u = 0$ no contorno para qualquer t, o que significa que as bordas da membrana estão fixas. Finalmente, o movimento da membrana será determinado pelas condições iniciais, que indicam o deslocamento inicial da membrana e a velocidade inicial :

$$\begin{cases} u(x,y,0) = u_0(x,y) \\ \dfrac{\partial u(x,y,0)}{\partial t} = u_1(x,y) \end{cases}$$

2.3 - FORMULACÃO FRACA

A equação de propagação de ondas foi obtida a partir de uma equação integral, dada pela expressão (2.1). O fenômeno físico está matematicamente representado por esta equação e,

portanto, seria possível tentar resolvê-la diretamente para obter a solução u. Entretanto costuma-se supor que u tenha mais regularidade para que seja possível construir a formulação clássica da equação diferencial equivalente (2.3). Soluções com tal regularidade, e que satisfaçam a equação pontualmente, requerem hipóteses adicionais e artificiais sobre os dados do problema. Tais soluções são denominadas clássicas e foram estudadas por D'Alembert [11]. Porém existem situações físicas relevantes em que tais condições de regularidade são violadas. Para contorná-las, reformula-se o problema de modo a admitir condições fracas sobre a solução e suas derivadas. Tal formulação é chamada Fraca ou Variacional e foi, provavelmente, motivada pela forma integral da equação. É precisamente esta formulação que é usada para construir as aproximações da solução usando o método de Galerkin.

Para tornar mais clara a noção de solução fraca da equação diferencial hiperbólica de segunda ordem são necessárias algumas considerações sobre Distribuições Vetoriais e a definição de espaços de Sobolev.

2.3.1 - DISTRIBUICÕES

Antes de introduzir o conceito de Distribuição,

define-se a noção de convergência para sucessões de funções em $C_o^\infty(\Omega)$, que é o espaço vetorial das funções infinitamente deriváveis em Ω, com suporte compacto contido em Ω, um aberto do $\mathbb{R}^2$.

NOÇÃO DE CONVERGÊNCIA EM $C_o^\infty(\Omega)$:

Diz-se que uma sucessão $\langle u_n \rangle$ de funções de $C_o^\infty(\Omega)$ converge para zero, quando as seguintes condições forem satisfeitas :

- todas as funções u_n possuem seus suportes contidos em um compacto fixo de Ω;
- a sucessão $\langle u_n \rangle$ converge uniformemente para zero em Ω, juntamente com todas as suas derivadas.

Uma sucessão $\langle u_n \rangle$ de funções de $C_o^\infty(\Omega)$ converge para uma função u de $C_o^\infty(\Omega)$, quando a sucessão $\langle u_n - u \rangle$ converge para zero no sentido acima definido.

ESPAÇOS DE DISTRIBUIÇÕES :

O espaço vetorial $C_o^\infty(\Omega)$ com esta noção de convergência, denomina-se o espaço das funções testes, representado por $\mathbb{D}(\Omega)$.

Uma Distribuição T sobre Ω é um funcional linear e

contínuo no sentido da convergência definida em $\mathbb{D}(\Omega)$. O espaço das Distribuições sobre Ω, $\mathbb{D}^*(\Omega)$, é o dual de $\mathbb{D}(\Omega)$. Como exemplo, seja Ω um aberto do $\mathbb{R}^2$ e $u \in L^1_{Loc}(\Omega)$, espaço vetorial das funções localmente integráveis em Ω. A forma linear T_u definida em $\mathbb{D}(\Omega)$ por :

$$T_u(\varphi) = \int_\Omega u(x,y)\varphi(x,y)\,dxdy$$

é uma Distribuição sobre Ω. Neste espaço vetorial $D^*(\Omega)$, uma sucessão (T_n) de Distribuições sobre Ω converge para uma Distribuição T sobre Ω, quando para toda função $\varphi \in \mathbb{D}(\Omega)$, a sucessão numérica $T_n(\varphi)$ converge para o número real $T(\varphi)$.

Seja T uma Distribuição sobre um aberto Ω do $\mathbb{R}^2$. Denomina-se derivada primeira de T em relação a x_i, $i=1,2$, ao funcional representado por $\frac{\partial T}{\partial x_i}$ e definido em $\mathbb{D}(\Omega)$ do seguinte modo :

$$\frac{\partial T}{\partial x_i}(\varphi) = -T\left(\frac{\partial \varphi}{\partial x_i}\right)$$

para toda $\varphi \in \mathbb{D}(\Omega)$. Pode-se mostrar que toda Distribuição é derivável e sua derivada é uma nova Distribuição. Seja $k=(k1,k2)$ um vetor de números inteiros não negativos e $|k|=k1+k2$. Define-se a derivada de ordem $|k|$ de uma Distribuição T sobre um aberto Ω, como sendo o funcional D^kT definido em $\mathbb{D}(\Omega)$ por :

$$D^k T = \frac{\partial^{|k|}}{\partial x_1^{k1} \partial x_2^{k2}} T(\varphi) = (-1)^{|k|} T\left(\frac{\partial^{|k|}}{\partial x_1^{k1} \partial x_2^{k2}} \varphi\right)$$

para toda φ em $\mathbb{D}(\Omega)$.

2.3.2 - ESPACOS DE SOBOLEV

Seja Ω um aberto do $\mathbb{R}^2$ e $L^p(\Omega)$ o espaço das funções mensuráveis f, definidas em Ω com valores reais tais que

$$\int_\Omega |f|^p \, dxdy < +\infty$$

Se $f \in L^p(\Omega)$, define-se a norma de f em $L^p(\Omega)$ da seguinte forma :

$$||f||_{L^p} = \left(\int_\Omega |f|^p \, dxdy\right)^{1/p}$$

Os espaços $L^p(\Omega)$, $1 \leq p < +\infty$, são de Banach e quando p=2, $L^2(\Omega)$ é um espaço de Hilbert com o seguinte produto interno :

$$\langle f, g \rangle = \int_\Omega f.g \, dxdy \qquad f, g \in L^2(\Omega)$$

ESPAÇOS DE SOBOLEV DE ORDEM UM :

Denomina-se o espaço de Sobolev de ordem um, e

representa-se por $H^1(\Omega)$, ao espaço das funções $v \in L^2(\Omega)$ tais que todas derivadas primeiras $\partial v/\partial x_i$, $i=1,2$, pertencem a $L^2(\Omega)$. As derivadas devem ser entendidas no sentido das Distribuições. Define-se o seguinte produto interno em $H^1(\Omega)$:

$$< u , v >_{H^1} = \int_\Omega u.v \, dxdy + \int_\Omega \nabla u . \nabla v \, dxdy,$$

para todo par $u,v \in H^1(\Omega)$. O espaço vetorial $H^1(\Omega)$ com esse produto interno é um espaço de Hilbert. Um subespaço de $H^1(\Omega)$, que será necessário às condições de contorno, é $H_o^1(\Omega)=\{ f \in L^2$, $\partial f/\partial x_i \in L^2(\Omega)$, $i=1,2$, e que se anulam na fronteira $\}$. Pode-se mostrar que $H_o^1(\Omega)$ é um espaço de Hilbert. O espaço vetorial das formas lineares contínuas sobre $H_o^1(\Omega)$, denomina-se o dual de $H_o^1(\Omega)$, e escreve-se H^{-1}; este também um espaço de Hilbert.

ESPAÇOS DE SOBOLEV DE ORDEM r :

O espaço de Sobolev de ordem r sobre Ω, $H^r(\Omega)$, é o espaço de todas as funções cujas derivadas, no sentido de distribuições, de ordem menor ou iqual a r pertencem a $L^2(\Omega)$. Define-se a norma de u em $H^r(\Omega)$ da seguinte forma :

$$|| u ||_r = \left[\sum_{|| j || \leq r} || D^j u ||^2_{L^p} \right]^{1/2}$$

Define-se o espaço de Sobolev $H^r_o(\Omega)$ como sendo o fecho de $C^\infty_o(\Omega)$ com respeito a norma $|| u ||_r$. Pode-se mostrar que esta norma em $H^r(\Omega)$ é equivalente a norma

$$|| u ||_{r,o} = \left[\sum_{|| j || = r} || D^j u ||^2_{L^p} \right]^{1/2}$$

2.3.3 - DISTRIBUIÇÕES COM VALORES VETORIAIS

Considere $1 \leq p < +\infty$ e um espaço de Hilbert real $\mathcal{M}$. Representa-se com $L^p(0,T;\mathcal{M})$, $0 < T < +\infty$, o espaço das funções vetoriais $v: (0,T) \longrightarrow \mathcal{M}$, mensuráveis, limitadas e tais que

$$\begin{array}{rccc} || v || : & (0,T) & \longrightarrow & \mathbb{R} \\ & t & \longrightarrow & || v(t) ||_{\mathcal{M}} \end{array}$$

pertence a $L^p(\Omega)$. Define-se em $L^p(0,T;\mathcal{M})$ a norma :

$$|| v ||_{L^p(0,T;\mathcal{M})} = \left[\int_0^T || v(s) ||^p_{\mathcal{M}} \, ds \right]^{1/p}$$

Caso $p = +\infty$, a norma em $L^{\infty}(0,T;\mathcal{M})$ é dada por

$$\| v \|_{L^{\infty}(0,T;\mathcal{M})} = \sup_{0<s<T} \text{ess } \| v(s) \|_{\mathcal{M}}$$

e pode-se provar que $L^{p}(0,T;\mathcal{M})$, $1 \leq p \leq +\infty$, é um espaço de Banach.

Ao espaço das aplicações lineares de $\mathcal{D}(0,T)$ em $\mathcal{M}$, contínua no sentido da convergência em $\mathcal{D}(0,T)$, $L(\mathcal{D}(0,T);\mathcal{M})$, dá-se o nome de Espaço das Distribuições Vetoriais sobre $(0,T)$. Como exemplo seja $v \in L^{p}(0,T;\mathcal{M})$; uma Distribuição Vetorial em $\mathcal{D}(0,T)$ em $\mathcal{M}$ pode ser definida da seguinte forma :

$$Y_v(\varphi) = \int_0^T v(s)\varphi(s)\,ds \qquad , \varphi \in \mathcal{D}(0,T)$$

Dada qualquer Distribuição Vetorial Y, define-se sua derivada de ordem k por :

$$\frac{d^k Y}{dt^k}(\varphi) = (-1)^k\, Y\left(\frac{d^k \varphi}{dt^k}\right),$$

para toda $\varphi \in \mathcal{D}(0,T)$.

2.3.4 - SOLUCÃO FRACA PARA A EQUACÃO DA ONDA

Considere o problema de determinar uma função $u=u(x,y,t)$ definida em $\Omega \times (0,T)$, Ω aberto do $\mathbb{R}^2$, que

satisfaça as seguintes condições :

$$(2.4)\quad \begin{cases} \dfrac{\partial^2 u}{\partial t^2} - \nabla.(c\nabla u) = f & \text{em } \Omega \times (0,T) \\ u(x,y,t) = 0 & \text{em } \partial\Omega \times (0,T) \\ u(x,y,0) = u_0(x,y) & \text{em } \Omega \\ \partial u/\partial t(x,y,0) = u_1(x,y) & \text{em } \Omega \end{cases}$$

onde $0 < c_{min} \leq c(x,y) \leq c_{max} < +\infty$ e também são conhecidas as funções f, u_0 e u_1. O objetivo agora é introduzir o conceito de Solução Fraca da equação diferencial. Multiplicando-se a equação (2.4) por $v \in H_0^1(\Omega)$ e integrando-se com respeito a variável espacial em Ω :

$$\int_\Omega \frac{\partial^2 u}{\partial t^2}\, v \; dxdy - \int_\Omega \nabla.(c\nabla u)\, v \; dxdy = \int_\Omega f\, v \, dxdy$$

Como $\nabla.(c\nabla u)v = \nabla.(vc\nabla u) - c\nabla u.\nabla v$, pelo teorema da Divergência [12] temos que :

$$\int_\Omega \nabla.(c\nabla u)v \; dxdy = \int_{\partial\Omega} c\frac{\partial u}{\partial \eta}.v \; ds - \int_\Omega c\nabla u.\nabla v \, dxdy$$

A primeira integral do lado direito é nula pois $v \in H_0^1(\Omega)$. Logo, a equação torna-se :

$$\int_\Omega \frac{\partial^2 u}{\partial t^2}\, v\, dxdy + \int_\Omega c\, \nabla u.\nabla v\, dxdy = \int_\Omega f\, v\, dxdy$$

Observe que esta equação pode ser reescrita como,

$$\frac{\partial}{\partial t}\int_\Omega \frac{\partial u}{\partial t}\, v\, dxdy + \int_\Omega c\, \nabla u.\nabla v\, dxdy = \int_\Omega f\, v\, dxdy \qquad (2.5)$$

Defina agora uma forma bilinear $a(u,v)$ em $H_o^1(\Omega)xH_o^1(\Omega)$,

$$a(u,v) = \int_\Omega c\nabla u.\nabla v\, dxdy$$

e, $\langle f,g\rangle = \int_\Omega fg\, dxdy$, um produto interno em $L^2(\Omega)$.

Logo, a equação (2.5) se escreve sob a forma :

$$\frac{\partial}{\partial t}\left\langle \frac{\partial}{\partial t}u(t),v\right\rangle + a(u(t),v) = \langle f(t),v\rangle \qquad \forall\, v \in H_o^1(\Omega),\ t \in (0,T)$$

Portanto, podemos entender por Solução Fraca do problema hiperbólico, uma função $u: (0,T) \longrightarrow H_o^1(\Omega)$, tal que :

$$\frac{d}{dt}\left\langle \frac{d}{dt}u(t),v\right\rangle + a(u(t),v) = \langle f(t),v\rangle,\ \forall\, v \in H_o^1(\Omega),$$

sendo a igualdade entendida no sentido das Distribuições sobre

$(0,T)$, ou seja, $\forall\, v \in H_0^1(\Omega)$ e $\forall\, \theta(t) \in \mathbb{D}(0,T)$,

$$\int_0^T \frac{d}{dt}\left\langle \frac{d}{dt} u(t), v\right\rangle \theta(t)\,dt + \int_0^T a(u(t), v)\theta(t)\,dt = \int_0^T \langle f(t), v\rangle \theta(t)\,dt$$

Nesta notação, define-se o problema em sua Formulação Fraca como :

Dados $f \in L^2(0,T;L^2(\Omega))$, $u_0 \in H_0^1(\Omega)$, $u_1 \in L^2(\Omega)$, achar $u : (0,T) \longrightarrow H_0^1(\Omega)$ satisfazendo as seguintes condições :

(1) $u \in L^\infty(0,T;H_0^1(\Omega))$

(2) $u' \in L^\infty(0,T;L^2(\Omega))$

(3) $u'' \in L^2(0,T;H^{-1}(\Omega))$

(4) $d/dt\ \langle u'(t), v\rangle + a(u(t)\ v) = \langle f(t), v\rangle$, $\forall\, v \in H_0^1(\Omega)$ no sentido de $\mathbb{D}^*(0,T)$

(5) $u(0) = u_0(x,y)$

(6) $u'(0) = u_1(x,y)$

com $u'(t) = \dfrac{d}{dt} u(t)$ $\qquad u''(t) = \dfrac{d^2}{dt^2} u(t)$

Podemos demonstrar que este problema possui solução única. O método da demonstração, que está apresentada na referência [14], consiste em aproximar a solução que se deseja encontrar por soluções de problemas análogos, porém de dimensão finita. Para tal, usa-se o fato de $H_0^1(\Omega)$ ser um

espaço de Hilbert separável. A dificuldade reside em se provar que esta sucessão de soluções obtidas em dimensão finita, converge para a solução do problema colocado na sua formulação fraca. A existência da solução aproximada resulta do fato de que ela está definida por um sistema linear de equações diferenciais ordinárias com valores iniciais, nas condições do teorema de existência de Cauchy [20]. Para se demonstrar a unicidade, procede-se da maneira usual.

2.4 - MÉTODO DE GALERKIN CONTÍNUO NO TEMPO

Este método permite aproximar soluções de equações diferenciais parciais, postas em sua forma Variacional ou Fraca. Ele consiste em definir formulações discretas, sobre subespaços de dimensão finita, que geram sequências de funções que supostamente convergem à solução da equação diferencial. O método de Galerkin será aplicado para a equação hiperbólica (2.4) cuja formulação fraca pode ser resumida por :

$$(2.6)\quad \begin{cases} \left\langle \dfrac{d^2u}{dt^2}, v \right\rangle + \langle c\nabla u, \nabla v \rangle = \langle f, v \rangle, \quad \forall\, v(t) \in H_o^1(\Omega) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 0 \le t < T \\ u(0) = u_o(x,y) \\ u'(0) = u_1(x,y) \end{cases}$$

A existência dos subespaços de aproximação $\mathcal{M}_h \subset \mathcal{M}$ vem do fato de $\mathcal{M}$, $H_o^1(\Omega)$ no nosso problema, ser um espaço de Hilbert separável, ou seja, admite uma base $\{W_i\}_{i=1}^{\infty}$ enumerável.

Seja $\mathcal{M}_h$ gerado pelos N primeiros vetores de $\mathcal{M}$, $\{W_1, W_2, \ldots, W_N\}$. O método de Galerkin contínuo no tempo consiste em encontrar uma função $U(t)$: $[0,T] \longrightarrow \mathcal{M}_h$, duas vezes diferenciável em relação a t, satisfazendo :

$$(2.7)\quad \begin{cases} \left\langle \dfrac{d^2U}{dt^2}, V \right\rangle + \langle c\nabla U, \nabla V \rangle = \langle f, V \rangle, & \forall\, V \in \mathcal{M}_h \\ & 0 \le t < T \\ U(0) = U_{0m}(x,y), \quad \lim\limits_{m \to \infty} U_{0m} = u_0 & \text{em } H_o^1(\Omega) \\ U'(0) = U_{1m}(x,y), \quad \lim\limits_{m \to \infty} U_{1m} = u_1 & \text{em } L^2(\Omega) \end{cases}$$

Escrevendo $U(t)$ em função da base de $\mathcal{M}_h$,

$$U(t) = \sum_{i=1}^{N} \eta_i(t) W_i$$

e substituindo esta expressão em (2.7) com $V = W_i$, $i=1,\ldots,N$, obtém-se o seguinte sistema de equações diferenciais ordinárias,

$$A \frac{d^2}{dt^2} \beta(t) + B \beta(t) = g(t) \qquad (2.8)$$

onde, A e B são matrizes definidas por

$$\begin{cases} A = [\ a_{i,j}\ ]_{NXN} & a_{i,j} = \langle W_i, W_j \rangle \\ B = [\ b_{i,j}\ ]_{NXN} & b_{i,j} = \langle c\nabla W_i, \nabla W_j \rangle \end{cases}$$

e, $g(t)$ e $\beta(t)$ vetores da forma

$$\begin{cases} g(t) = [\ f_i(t)\ ]_{NX1} & f_i(t) = \langle f, W_i \rangle \\ \beta(t) = [\ \beta_i(t)\ ]_{NX1}\ , \end{cases}$$

Pode-se mostrar que o sistema acima possui solução única para cada escolha de $U(0)$ e $U'(0)$ [20]. A questão da convergência das funções $U(t)$, quando $N \to \infty$, é tratada em vários livros especializados,[09] e [21]. Também é importante estabelecer estimativas para o erro $||U-u||$ em normas convenientes; em [09] pode-se encontrar um estudo a respeito.

2.5 - MÉTODO DE GALERKIN DISCRETO NO TEMPO

A idéia do método é obter soluções aproximadas para o conjunto de equações diferencias ordinárias (2.8), discretizando a variável t. Seja $t_M = M\Delta t$, onde $\Delta t = T/\hat{M}$, e $U^M(x,y)=U(x,y,t_M)$ a solução discreta da equação (2.7) no nível de tempo t_M. Escrevendo $U^M(x,y)$ em função da base de $\mathcal{M}_h$, tem-se :

$$U^M(x,y) = \sum_{i=1}^{N} \eta_i^M \, W_i(x,y)$$

Como aproximação para a derivada segunda em relação a t, usa-se, por exemplo, diferenças finitas centrais da seguinte forma :

$$\frac{d^2U}{dt^2} = \frac{U^{M+1} - 2U^M + U^{M-1}}{(\Delta t)^2}$$

Pode-se agora estabelecer o método de Galerkin discreto no tempo :

Achar $U^M \in \mathcal{M}_h$ tal que, $\forall\, V \in \mathcal{M}_h$,

$$< \frac{U^{M+1} - 2U^M + U^{M-1}}{(\Delta t)^2}, V > + <c\nabla U^{MK}, \nabla V > = < f^{MK}, V> \qquad (2.9)$$

onde,
$$\begin{cases} U^{MK} = K(U^{M+1} + U^{M-1}) + (1-2K)U^M \\ f^M = f(x, t_M) \\ f^{MK} = K(f^{M+1} + f^{M-1}) + (1-2K)f^M \end{cases}$$

Em [09], temos que para $K \geq 1/4$ este método é estável independentemente de Δt e $\mathcal{M}_h$. Substituindo em (2.9) as expressões para U^{M-1}, U^M e U^{M+1} em função da base de $\mathcal{M}_h$, com $V=W_i$, $i=1,\ldots,N$, obtém-se um sistema de equações lineares, em cada passo de tempo.

Para que esse método seja competitivo com o método de diferenças finitas, é preciso que ele apresente menos dificuldades na obtenção das soluções dos sistemas. A construção dos espaços $\mathcal{M}_h$ usando polinômios por partes garante o sucesso do método pois fornece sistemas esparsos de equações, cujas soluções são facilmente obtidas por técnicas próprias. De fato, uma escolha apropriada das funções de base de $\mathcal{M}_h$, produz em (2.9) sistemas que possuem estruturas de banda. Quando o espaço de aproximação $\mathcal{M}_h$ é gerado por funções que são polinômios por partes, o método de Galerkin é denominado de método do Elemento Finito. Outros métodos discretos no tempo podem ser vistos na referência [09] e, no próximo capítulo, descreveremos o método que combina direções alternadas com o processo de Galerkin.

Vamos apresentar agora os espaços conhecidos por Splines ou polinômios de Hermite por partes que têm sido amplamente utilizados pois possuem propriedades de aproximações e possibilitam a construção de bases locais. Seja $\Pi : a=x_1<x_2<\ldots<x_N<x_{N+1}=b$ uma partição regular do intervalo $I=[a,b]$, com $h=(b-a)/N$ e $I_j=[x_j,x_{j+1}]$, $j=1,\ldots,N$. Considere $P_s(E)$ o conjunto dos polinômios em E de grau no máximo s. Definimos o espaço linear, $M_k^s(\Pi)$, dos polinômios por partes como :

$$M_k^{\bullet}(\Pi) = \left\{ v \in C^k(I) : \quad v|_{I_j} \in P_{\bullet}(I_j) \quad ; \; j=1,\ldots,N \right\}$$

onde $0 \leq k < s$, $v|_{I_j}$ denota a restrição de v ao intervalo I_j e $C^k(I)$ representa o conjunto das funções com derivada até a ordem k contínuas em I. Como exemplo, considere o espaço dos polinômios de Hermite por partes definido em Π, $H^{(m)}(\Pi)$, como sendo o seguinte conjunto de polinômios por partes :

$$M_{m-1}^{2m-1}(\Pi) = \left\{ v \in C^{m-1}(I) : \quad v|_{I_j} \in P_{2m-1}(I_j) \quad ; j=1,\ldots,N \right\}$$

Em particular, para m=2 temos o espaço que foi utilizado nos nossos experimentos computacionais, denominado espaço de Hermite Cúbico. Um elemento de $H^{(2)}(\Pi)$ é uma função $p \in C^1(I)$ definida em cada subintervalo $[x_j, x_{j+1}]$ de Π por um polinômio de grau 3 tal que :

$$\begin{cases} \dfrac{d^k}{dx^k} p_j(x_j) = d_j^{(k)} \\ \dfrac{d^k}{dx^k} p_j(x_{j+1}) = d_{j+1}^{(k)} \end{cases}$$

onde k=0,1 e $d_j^{(k)}$ são parâmetros interpolantes. É fácil verificar que existe em cada subintervalo $[x_j, x_{j+1}]$ um único polinômio interpolante nestas condições [09]. Denotaremos por $H_o^{(m)}(\Pi)$ o espaço

$$H^{(m)}(\Pi) \cap \{ v \; / \; v \text{ se anula na fronteira} \}.$$

Para estabelecer o resultado básico que fornece a estimativas para o erro do esquema (2.9), discreto no espaço e no tempo, torna-se necessário apresentar o seguinte resultado da teoria de aproximações por Splines [09]: seja $\mathcal{M}_h$ um subespaço de dimensão finita de $H^k(\Omega)$. Diz-se que $\mathcal{M}_h$ é de classe $S_{k,r}(\Omega)$ se para qualquer $u \in H^j(\Omega)$, existir $\tilde{u} \in \mathcal{M}_h$ e uma constante c, independente de h e u, tal que

$$|| u - \tilde{u} ||_\ell \leq c\, h^{j-\ell} || u ||_j \qquad (2.10)$$

com $0 \leq \ell \leq k$ e $\ell \leq j \leq r$. Assim, como exemplo, se $\mathcal{M}_h$ é de classe $S_{1,r}(\Omega)$, tem-se :

$$\ell = 0 \qquad || u - \tilde{u} ||_2 \leq c\, h^j || u ||_j$$

$$\ell = 1 \qquad || u - \tilde{u} ||_1 \leq c\, h^{j-1} || u ||_j ,$$

ou seja, as aproximações são da ordem de h^r, $O(h^r)$, para funções $u \in H^r(\Omega)$. Analogamente, $\mathcal{M}_h \subset H^k(\Omega) \cap H_0^1(\Omega)$ é de classe $S^0_{k,r}(\Omega)$ se (2.10) acontece para $u \in H^j(\Omega) \cap H_0^1(\Omega)$. Portanto, estas propriedades dos espaços de aproximação $\mathcal{M}_h$ se relacionam com o estudo da ordem de precisão do método de Galerkin.

Os espaços $H_0^{(m)}(\Pi)$ possuem a propriedade de que para qualquer função $u \in H^j(I) \cap H_0^1(I)$ existe $\tilde{u} \in H_0^{(m)}(\Pi)$ e uma constante c, independente de h e u, tal que

$$|| D^{\ell}(u - \tilde{u}) ||_{L^2} \leq c\, h^{j-\ell} || D^{j}u ||_{L^2} \qquad (2.11)$$

para todo $0 \leq \ell \leq m$ e $\ell \leq j \leq 2m$. Portanto, para cada m, $H_o^{(m)}(\Pi)$ é uma família de espaços de classe $S^o_{m,2m}(I)$. Para os espaço de Hermite Cúbico (m=2) temos :

$\ell = 0$ $\qquad || u - \tilde{u} ||_{L^2} \leq c\, h^{j} || u ||_{L^2}$

ou seja, as aproximações são da ordem de h^4, $O(h^4)$, para funções $u \in H^4(I) \cap H_o^1(\Omega)$,

$\ell = 1$ $\qquad || D(u - \tilde{u}) ||_{L^2} \leq c\, h^{j-1} || Du ||_{L^2}$

isto é, a aproximação da derivada é da ordem de h^3, $O(h^3)$, para funções $u \in H^4(I) \cap H_o^1(\Omega)$. No teorema apresentado abaixo encontram-se as estimativas para o erro do esquema discreto no tempo, (2.8).

TEOREMA :

Suponha que $u \in L^{\infty}(0,T;H^r(\Omega))$, $\partial u/\partial t \in L^2(0,T;H^r(\Omega))$, $\partial^K u/\partial t^K \in L^2(0,T;L^2(\Omega))$, para k=3,4, e

$$|| \partial_t (U-W)^o ||_{L^2} + || (U-W)^{1/2} ||_{L^2} = O(h^r + (\Delta t)^2) \qquad (2.12)$$

onde $W : [0,T] \longrightarrow \mathcal{M}_h$ é tal que $\langle c(\nabla(W-u), \nabla V\rangle + \langle W-u, V\rangle = 0$,

$\forall\ V \in \mathcal{M}_h$. Então $\exists$ uma constante c tal que, para $0 \leq M \leq \hat{M}-1$,

$$|| \partial_t (u-U)^M ||_{L^2} + || (u-U)^{M+1/2} ||_{L^2} \leq c(h^r + (\Delta t)^2) \qquad (2.13)$$

sendo $\mathcal{M}_h$ um subespaço de classe $S_{1,r}(\Omega)$, $r \geq 2$ e

$$\partial_t U^M = \frac{U^{M+1} - U^M}{\Delta t} \quad \text{e} \quad U^{M+1/2} = \frac{U^M + U^{M+1}}{2} .$$

As aproximações iniciais U^0 e U^1 devem ser escolhidas de tal forma que (2.12) é satisfeita. Em [09], encontram-se, além deste teorema, algumas técnicas para a escolha desses valores.

3 - GALERKIN COM DIRECÕES ALTERNADAS

3.1 - INTRODUCÃO

Neste capítulo veremos como o método de Direções Alternadas pode ser aplicado na formulação de Galerkin, para obtenção da solução discreta da equação de propagação de ondas em meios acústicos (2.4). Os resultados apresentados aqui foram desenvolvidos por Douglas e Dupont em [06] e, se limitam aos domínios retangulares do $\mathbb{R}^2$.

A idéia utilizada por Douglas e Dupont foi modificar a equação (2.4), introduzindo dois termos, de modo que o problema, colocado na sua Formulação Fraca, possa ser fatorado num produto tensorial. Fazendo uma escolha apropriada do subespaço de aproximação no processo de Galerkin, é possível separar, neste produto tensorial, as variáveis espaciais. Consegue-se, portanto, como no método de Direções Alternadas, reduzir o problema bidimensional a um conjunto de problemas unidimensionais.

No próximo capítulo apresentaremos os detalhes da implementação computacional deste algoritmo quando usamos Hermite Cúbico como espaço de aproximação.

3.2 - PROBLEMA APROXIMADO

Seja $\Omega = [0,1] \times [0,1]$. Considere a seguinte modificação da equação (2.4) :

$$(3.1) \quad \begin{cases} \dfrac{\partial^2 u}{\partial t^2} - \lambda (\Delta t)^2 \nabla^2 . \left(\dfrac{\partial^2 u}{\partial t^2} \right) + \\ + \lambda^2 (\Delta t)^4 \dfrac{\partial^4}{\partial x^2 \partial y^2} \left(\dfrac{\partial^2 u}{\partial t^2} \right) = \nabla . (c \nabla u) + f \end{cases}$$

onde λ é um parâmetro de estabilidade que deve assumir valores maiores que $(1/4)c_{max}$, onde c_{max} é o valor máximo da função $c(x,y)$ [06].

Na construção das aproximações da solução de (3.1), usando o método de Galerkin, é necessária, como vimos no capítulo 2, a sua Formulação Fraca. Para tal, multiplica-se esta equação por $v \in H_o^1(\Omega)$ e integra-se com respeito a variável espacial em Ω, ou seja,

$$\int_\Omega \frac{\partial^2 u}{\partial t^2} v \, dxdy - \lambda (\Delta t)^2 \int_\Omega \nabla^2 . \left(\frac{\partial^2 u}{\partial t^2} \right) v \, dxdy +$$

$$+ \lambda^2 (\Delta t)^4 \int_\Omega \frac{\partial^4}{\partial x^2 \partial y^2} \left(\frac{\partial^2 u}{\partial t^2} \right) v \, dxdy =$$

$$= \int_\Omega \nabla . (c \nabla u) \, v \, dxdy + \int_\Omega f \, v \, dxdy \qquad (3.2)$$

Como foi feito em 2.3.4, utilizando o fato de

$$\nabla.(c\nabla u)v = \nabla.(vc\nabla u) - c\nabla u.\nabla v,$$

e $v \in H_o^1(\Omega)$, pelo teorema da Divergência temos que :

$$\int_\Omega \nabla.(c\nabla u)\, v \, dxdy = -\int_\Omega c\nabla u.\nabla v \, dxdy$$

Assim, tomando na equação acima $c = 1$ e $u = (\partial^2 u/\partial t^2)$, temos :

$$\int_\Omega \nabla^2.\left(\frac{\partial^2 u}{\partial t^2}\right) v \, dxdy = -\int_\Omega \nabla\left(\frac{\partial^2 u}{\partial t^2}\right).\nabla v \, dxdy$$

Integrando por partes a terceira integral de (3.2), uma vez em cada direção x e y, obtemos :

$$\int_\Omega \frac{\partial^4}{\partial x^2 \partial y^2}\left(\frac{\partial^2 u}{\partial t^2}\right) v \, dxdy = \int_\Omega \frac{\partial^2}{\partial x \partial y}\left(\frac{\partial^2 u}{\partial t^2}\right)\frac{\partial^2 v}{\partial x \partial y} \, dxdy$$

Substituindo estes resultados em (3.2), ficamos com a seguinte equação :

$$\int_\Omega \frac{\partial^2 u}{\partial t^2} v \, dxdy + \lambda (\Delta t)^2 \int_\Omega \nabla\left(\frac{\partial^2 u}{\partial t^2}\right).\nabla v \, dxdy +$$

$$+ \lambda^2 (\Delta t)^4 \int_\Omega \frac{\partial^2}{\partial x \partial y}\left(\frac{\partial^2 u}{\partial t^2}\right)\frac{\partial^2 v}{\partial x \partial y} \, dxdy =$$

$$= -\int_\Omega c\nabla u.\nabla v \, dxdy + \int_\Omega f\, v \, dxdy$$

Colocando a equação acima na notação do produto interno definido na seção 2.3.4, temos a Formulação Fraca da equação (3.1) :

Dados f, u_o e u_1, determinar $u : (0,T) \longrightarrow H_o^1(\Omega)$ satisfazendo :

$$
(3.3)\quad \begin{cases}
(1)\ \left\langle \dfrac{d^2u}{dt^2}, v \right\rangle + \lambda\,(\Delta t)^2 \left\langle \nabla\left(\dfrac{d^2u}{dt^2} \right), \nabla v \right\rangle + \\
\qquad + \lambda^2\,(\Delta t)^4 \left\langle \dfrac{\partial^2}{\partial x \partial y}\left(\dfrac{d^2u}{dt^2} \right), \dfrac{\partial^2 v}{\partial x \partial y} \right\rangle = \\
\qquad = -\langle c\nabla u, \nabla v \rangle + \langle f, v \rangle, \quad \forall\, v(t) \in H_o^1(\Omega) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 0 \le t < T \\
\qquad \text{no sentido de } \mathcal{D}^*(0,T). \\
(2)\ u(0) = u_o(x,y) \\
(3)\ u'(0) = u_1(x,y)
\end{cases}
$$

Seja $\mathcal{M}_h$ um subespaço de aproximação de $H_o^1(\Omega)$ gerado por $\{ W_1, W_2, \ldots, W_N \}$. O primeiro problema aproximado para (3.3) consiste em encontrar uma função $U(t): [0,T] \longrightarrow \mathcal{M}_h$, duas vezes diferenciável em relação a t, satisfazendo :

$$
(3.4)\quad
\begin{cases}
\left\langle \dfrac{d^2U}{dt^2}\,,\, V \right\rangle + \lambda\,(\Delta t)^2 \left\langle \nabla\!\left(\dfrac{d^2U}{dt^2}\right),\, \nabla V \right\rangle + \\[2ex]
\quad + \lambda^2\,(\Delta t)^4 \left\langle \dfrac{\partial^2}{\partial x \partial y}\left(\dfrac{d^2U}{dt^2}\right),\, \dfrac{\partial^2 V}{\partial x \partial y} \right\rangle = \\[2ex]
\quad = -\langle c\nabla U\,,\, \nabla V\rangle + \langle f\,,\, V\rangle, \qquad \forall\, V(t) \in \mathcal{M}_h \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 0 \le t < T \\[2ex]
U(0) = U_{0M}(x,y), \qquad \lim_{M\to\infty} U_{0M} = u_0 \qquad \text{em } H_0^1(\Omega) \\[2ex]
U'(0) = U_{1M}(x,y), \qquad \lim_{M\to\infty} U_{1M} = u_1 \qquad \text{em } L^2(\Omega)
\end{cases}
$$

Escrevendo $U(t)$ em função da base de $\mathcal{M}_h$, temos :

$$U(t) = \sum_{i=1}^{N} \eta_i(t) W_i$$

e substituindo esta expressão em (3.4) com $V = W_i$, $i=1,\ldots,N$, obtém-se um sistema de equações ordinárias.

O procedimento agora é procurar soluções aproximadas para este sistema, discretizando a variável t. Seja $t_M = M\,\Delta t$, onde $\Delta t = T/\hat{M}$, e $U^M(x,y)=U(x,y,t_M)$ a solução da equação (3.4) no nível de tempo t_M. Escrevendo $U^M(x,y)$ em função da base de $\mathcal{M}_h$, ficamos com a expressão :

$$U^M(x,y) = \sum_{i=1}^{N} \eta_i^M\, W_i(x,y)$$

Como aproximação para a derivada segunda em relação a t, usaremos, como na seção 2.5, diferenças finitas centrais. Ficamos, portanto, com o seguinte problema aproximado :

$$
(3.5)\quad \begin{cases}
\left\langle \dfrac{U^{M+1} - 2U^{M} + U^{M-1}}{(\Delta t)^2} , V \right\rangle + \\
+ \lambda \left\langle \nabla(U^{M+1} - 2U^{M} + U^{M-1}) , \nabla V \right\rangle + \\
+ \lambda^2 (\Delta t)^2 \left\langle \dfrac{\partial^2}{\partial x \partial y} (U^{M+1} - 2U^{M} + U^{M-1}), \dfrac{\partial^2 V}{\partial x \partial y} \right\rangle = \\
= - \langle c \nabla U^{M} , \nabla V \rangle + \langle f^{M}, V \rangle \qquad \forall\, V \in \mathcal{M}_h \\
\text{onde,} \quad f^{M} = f(x, y, t_{M})
\end{cases}
$$

Vamos considerar $\mathcal{M}_h = \mathcal{M}_{hx} \otimes \mathcal{M}_{hy}$, onde $\mathcal{M}_{hx}$ e $\mathcal{M}_{hy}$ são subespaços de dimensão finita de $H_0^1([0,1])$, cujas bases são $\{ \alpha_i(x),\ i=1,\ldots,N \}$ e $\{ \beta_j(y),\ j=1,\ldots,N \}$, respectivamente. Então, $\mathcal{M}_h$ é gerado por $\{ \alpha_1\beta_1,\ \alpha_1\beta_2,\ \alpha_1\beta_3,\ \ldots,\ \alpha_N\beta_N \}$. Escrevendo $U^{M}(x,y)$ em função desta base temos

$$
U^{M}(x,y) = \sum_{i=1}^{N} \sum_{j=1}^{N} \eta_{ij}^{M} \ \alpha_i(x)\beta_j(y) \qquad (3.6)
$$

e substituindo esta expressão em (3.5) com $V=\alpha_K\beta_L$, obtemos :

$$\frac{1}{(\Delta t)^2} < \sum_{i,j} (\eta_{ij}^{M+1} - 2\eta_{ij}^{M} + \eta_{ij}^{M-1})\alpha_i\beta_j \, , \, \alpha_K\beta_L > \; +$$

$$+ \; \lambda < \sum_{i,j} (\eta_{ij}^{M+1} - 2\eta_{ij}^{M} + \eta_{ij}^{M-1})(\alpha_i'\beta_j , \alpha_i\beta_j') \, , \, (\alpha_K'\beta_L , \alpha_K\beta_L') > +$$

$$+ \; \lambda^2 (\Delta t)^2 < \sum_{i,j} (\eta_{ij}^{M+1} - 2\eta_{ij}^{M} + \eta_{ij}^{M-1})\alpha_i'\beta_j' \, , \, \alpha_K'\beta_L' > \; =$$

$$= \; - < c\nabla U^M \, , \, \nabla\alpha_K\beta_L > \; + \; < f^M \, , \, \alpha_K\beta_L > \qquad (3.7)$$

Definimos em $L^2([0,1])$ os seguintes produtos internos:

$$< f \, , \, g >_x \; = \int_0^1 f \, g \, dx$$

$$< f \, , \, g >_y \; = \int_0^1 f \, g \, dy$$

Usando esta definições em (3.7) e as propriedades do produto interno, pode-se obter :

$$\sum_{i,j} < \alpha_i , \alpha_K >_x < \beta_L , \beta_j >_y \frac{Z_{ij}}{(\Delta t)^2} \; +$$

$$+ \; \lambda \sum_{i,j} (< \alpha_i' , \alpha_K' >_x < \beta_L , \beta_j >_y + < \alpha_i , \alpha_K >_x < \beta_L' , \beta_j' >_y) Z_{ij} +$$

$$+ \; \lambda^2 (\Delta t)^2 \sum_{i,j} < \alpha_i' , \alpha_K' >_x < \beta_L' , \beta_j' >_y Z_{ij} \; =$$

$$= \; - < c\nabla U^M \, , \, \nabla\alpha_K\beta_L > \; + \; < f^M \, , \, \alpha_K\beta_L > \qquad (3.8)$$

onde, $Z_{ij} = \eta_{ij}^{M+1} - 2\eta_{ij}^{M} + \eta_{ij}^{M-1}$.

Definimos o produto tensorial entre duas matrizes quaisquer A_{MxN} e B_{RxS} como,

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1N}B \\ \vdots & & & \\ a_{M1}B & a_{M2}B & \cdots & a_{MN}B \end{bmatrix} \qquad (3.9)$$

onde $A \otimes B$ tem MR linhas e NS colunas. Em [02], encontramos as seguintes propriedades do produto tensorial :

$$(3.10) \begin{cases} 1) \ (A + B) \otimes (C + D) = A \otimes C + A \otimes D + B \otimes C + B \otimes D \\ 2) \ (A \otimes B)(C \otimes D) = (AC) \otimes (BD) \end{cases}$$

Sejam C^X, C^Y, A^X e A^Y matrizes definidas por :

$$C_{ij}^{X} = \int_0^1 \alpha_i(x)\alpha_j(x)\,dx \qquad C_{ij}^{Y} = \int_0^1 \beta_i(y)\beta_j(y)\,dy \qquad (3.11)$$

$$A_{ij}^{X} = \int_0^1 \alpha_i'(x)\alpha_j'(x)\,dx \qquad A_{ij}^{Y} = \int_0^1 \beta_i'(y)\beta_j'(y)\,dy$$

Substituindo estas definições em (3.8), temos :

$$C^X \otimes C^Y \frac{Z}{(\Delta t)^2} + \lambda (A^X \otimes C^Y + C^X \otimes A^Y) Z +$$

$$+ \lambda^2(\Delta t)^2 A^X \otimes A^Y Z = - < c\nabla U^M , \nabla\alpha_K\beta_L > + < f^M , \alpha_K\beta_L >$$

onde, $Z = [Z_{ij}]_{N^2 \times 1}$ com $Z_{ij} = \eta_{ij}^{M+1} - 2\eta_{ij}^{M} + \eta_{ij}^{M-1}$ é um vetor de comprimento N^2 construído fixando i, $i=1,\ldots,N$, e variando $j=1,\ldots,N$.

Multiplicando esta equação por $(\Delta t)^2$ e usando a propriedade 1 do produto tensorial, é possível separar as variáveis espaciais que aparecem neste produto, como mostra a equação abaixo :

$$(3.12)\quad \begin{cases} (C^X + \lambda(\Delta t)^2 A^X) \otimes (C^Y + \lambda(\Delta t)^2 A^Y) Z = (\Delta t)^2 \Phi^M \\ \text{onde,} \\ \begin{cases} Z = \eta^{M+1} - 2\eta^M + \eta^{M-1}, \quad M \geq 1 \\ (\Phi^M)_{KL} = - \langle c\nabla U^M , \nabla \alpha_K \beta_L \rangle + \langle f^M, \alpha_K \beta_L \rangle \end{cases} \end{cases}$$

Podemos observar em (3.12) que a matriz do sistema é constituída de um produto tensorial entre matrizes cujos elementos só envolvem uma variável espacial. Além disto, notamos que para iniciar o processo é preciso especificar η^0 e η^1, ou seja,

$$U^0(x,y) = \sum_{i=1}^{N} \sum_{j=1}^{N} \eta_{ij}^{0} \alpha_i(x)\beta_j(y) \qquad (3.13)$$

$$U^1(x,y) = \sum_{i=1}^{N} \sum_{j=1}^{N} \eta_{ij}^{1} \alpha_i(x)\beta_j(y) \qquad (3.14)$$

para, então, obter η^2, η^3,..., η^M usando (3.12). Veremos na

seção 3.3 os procedimentos adotados para o cálculo das aproximações iniciais.

Vamos apresentar agora a expressão que fornece uma estimativa de erro para as aproximações calculadas pelo esquema (3.12), sem os detalhes de sua demonstração, que podem ser encontrados na referência [06]. Definimos $e^M = u^M - U^M$, onde $u^M = u(x,y,t_M)$ e U^M é a aproximação da solução no nível de tempo t_M, com $t_M = M\Delta t$ e $\Delta t = T/\hat{M}$. Sejam,

$$|| v ||^2_{\mathcal{M}xL^2} = \sum_{M=0}^{\hat{M}} || v^M ||^2_{\mathcal{M}} \Delta t$$

$$|| v ||_{\mathcal{M}xL^\infty} = \max_{0 \le t_M \le T} || v^M ||_{\mathcal{M}}$$

Fazendo uma escolha apropriada para a função teste V em (3.5), manipulações algébricas e, usando a condição de estabilidade $\lambda > (1/4)c_{max}$ e o lema discreto de Gronwall [09], Douglas e Dupont mostram que :

$$|| \Delta_t e ||^2_{L^2xL^\infty} + || e ||^2_{H_0^1xL^\infty} + (\Delta t)^4 || \frac{\partial^2 (\Delta_t e)}{\partial x \partial y} ||^2_{L^2xL^\infty}$$

$$\le C \left[(\Delta t)^4 + || e^0 ||^2_{1,0} + || e^1 ||^2_{1,0} + || \Delta_t e^0 ||^2_{L^2} + \right.$$

$$\left. + (\Delta t)^4 || \frac{\partial^2 (\Delta_t e^0)}{\partial x \partial y} ||^2_{L^2} + || \partial \delta u ||^2_{H^1xL^2} + \right.$$

$$+ \; || \; \partial \; \delta u \; ||^2_{L^2xL^\infty} \; + \; || \; \Delta_t^2 \; \delta u \; ||^2_{L^2xL^2} \; +$$

$$+ \; (\Delta t)^4 \left\{ || \; \frac{\partial^2}{\partial x \partial y} (\; \partial \; \delta u \;) \; ||^2_{L^2xL^\infty} + || \; \frac{\partial^2}{\partial x \partial y} (\; \Delta_t^2 \; \delta u \;) \; ||^2_{L^2xL^2} \right\} \Big]$$

onde, $\delta u = \tilde{U} - u$, $\forall \; \tilde{U}(t) \in \mathcal{M}_h$ e u é solução de (2.4).

$$\Delta_t \; e^M = \frac{e^{M+1} - e^M}{\Delta t} \qquad \partial \; \delta u^M = \frac{\delta u^{M+1} - \delta u^{M-1}}{2\Delta t}$$

$$\Delta_t \; \delta u^M = \frac{\delta u^{M+2} - \delta u^{M+1} - \delta u^M + \delta u^{M-1}}{2(\Delta t)^2} \qquad (3.15)$$

Desta expressão vemos que os valores das aproximações iniciais U^0 e U^1 devem ser escolhidos de modo que

$$|| \; e^0 \; ||^2_{1,0} \; + \; || \; e^1 \; ||^2_{1,0} \; +$$

$$+ \; || \; \Delta_t e^0 \; ||^2_{L^2} \; + \; (\Delta t)^4 \; || \; \frac{\partial^2}{\partial x \partial y} (\; \Delta_t e^0 \;) \; ||^2_{L^2} \; \leq$$

$$\leq \; C \; [\; (\Delta t)^4 + \text{os erros das aproximações }] \qquad (3.16)$$

para preservar a ordem de grandeza da estimativa. Os termos

$$|| \; \partial \; \delta u \; ||^2_{H^1xL^2} \; + \; || \; \partial \; \delta u \; ||^2_{L^2xL^\infty} \; + \; || \; \Delta_t^2 \; \delta u \; ||^2_{L^2xL^2}$$

são delimitados através da expressão (2.11) uma vez que estamos usando $H_0^{(2)}(\Pi)$ e admitindo como hipótese que a solução analítica está em $H^4(\Omega) \cap H_0^1(\Omega)$. Assim, podemos concluir que a

delimitação final é :

$$\| \Delta_t e \|_{L^2xL^\infty} + \| e \|_{H^1xL^\infty} + (\Delta t)^4 \| \frac{\partial^2}{\partial x \partial y} (\Delta_t e) \|_{L^2xL^\infty}$$

$$\leq \varepsilon \left\{ (\Delta t)^2 + h^4 \right\}$$

3.3 - APROXIMACÕES INICIAIS

3.3.1 - CÁLCULO DE U^o

No cálculo da aproximação da solução no tempo t=0, U^o, pode-se usar, por exemplo, Quadrados Mínimos [15]. Para tal, projeta-se $u^o = u(x,y,0) = u_o$ no espaço de aproximação $\mathcal{M}_h$, da seguinte forma :

$$< U^o - u^o, \alpha_K \beta_L > = 0 \qquad K,L = 1,2,\ldots,N \qquad (3.17)$$

onde $\alpha_K \beta_L$ são os elementos da base de $\mathcal{M}_h$. Logo,

$$< U^o , \alpha_K \beta_L > = < u_o , \alpha_K \beta_L > \qquad K,L = 1,2,\ldots,N \qquad (3.18)$$

Substituindo (3.13) em (3.18) e utilizando as propriedades do produto interno, obtemos

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \langle \alpha_i \beta_j , \alpha_K \beta_L \rangle \eta^o_{ij} = \langle u_o , \alpha_K \beta_L \rangle$$

que pode ser escrito, usando as definições dadas em (3.11), da sequinte forma :

$$(3.19) \quad \begin{cases} (C^X \otimes C^Y) \eta^o = \phi^o \\ \text{onde,} \quad (\phi^o)_{KL} = \langle u_o , \alpha_K \beta_L \rangle \end{cases}$$

Resolvendo este sistema linear podemos determinar η^o. No entanto, o procedimento adotado por Douglas e Dupont em [06] foi tomar a seguinte projeção,

$$\langle U^o - u_o , \alpha_K \beta_L \rangle + \langle \nabla(U^o - u_o) , \nabla \alpha_K \beta_L \rangle +$$

$$+ \langle \frac{\partial^2}{\partial x \partial y} (U^o - u_o) , \frac{\partial^2}{\partial x \partial y} \alpha_K \beta_L \rangle = 0$$

de modo a obter a estimativa desejada em (3.16). Substituindo a expressão de U^o na equação acima e usando as definições dadas em (3.11), determinaremos abaixo o esquema que será usado para o cálculo de η^o.

$$(3.20) \quad \begin{cases} (C^X + A^X) \otimes (C^Y + A^Y) \eta^o = \phi^o \\ \text{onde,} \quad (\phi^o)_{KL} = \langle u_o , \alpha_K \beta_L \rangle + \langle \nabla u_o , \nabla \alpha_K \beta_L \rangle + \\ \qquad + \langle \frac{\partial^2}{\partial x \partial y} u_o , \frac{\partial^2}{\partial x \partial y} \alpha_K \beta_L \rangle \end{cases}$$

Douglas e Dupont demonstram em [06] que :

$$\| e^0 \|^2_{1,0} + \| \frac{\partial^2}{\partial x \partial y} e^0 \|^2_{L^2} \leq C \left[\| \delta u^0 \|^2_{1,0} + \right.$$

$$\left. + \| \frac{\partial^2}{\partial x \partial y} \delta u^0 \|^2_{L^2} + (\Delta t)^4 \right] \qquad (3.21)$$

Usar o esquema (3.20) para o cálculo de η^0 diminui o esforço computacional pois possibilita, com pequenas modificações, aproveitar o procedimento do caso geral (3.12).

3.3.2 - CÁLCULO DE U^1

Seja $u = u(x,y,t)$ solução de (2.4). O desenvolvimento em série de Taylor de u em relação a t, em torno do ponto t=0, é dado por :

$$u(x,y,\Delta t) = u(x,y,0) + \Delta t \frac{\partial}{\partial t} u(x,y,0) +$$

$$+ \frac{(\Delta t)^2}{2} \frac{\partial^2}{\partial t^2} u(x,y,0) + \Theta(\Delta t^3)$$

Substituindo nesta equação as condições dadas em (2.4), consegue-se uma aproximação para $u^1 = u(x,y,\Delta t)$, ou seja,

$$u^1 = u_0 + \Delta t u_1 + \frac{(\Delta t)^2}{2} (\nabla.(c \nabla u_0) + f^0) + O(\Delta t^3)$$

Considere $s = u^1 - u_o + O(\Delta t^3)$. Vamos tomar a seguinte projeção de s em $\mathcal{M}_h$:

$$\langle (U^1-U^0)-s \, , \, \alpha_K\beta_L \rangle + \Delta t \langle \nabla(U^1-U^0)-\nabla s \, , \, \nabla\alpha_K\beta_L \rangle +$$

$$+ (\Delta t)^4 \left\langle \frac{\partial^2}{\partial x\partial y}(U^1-U^0) - \frac{\partial^2}{\partial x\partial y} s \, , \, \frac{\partial^2}{\partial x\partial y}\alpha_K\beta_L \right\rangle = 0$$

onde $\alpha_K\alpha_L$ são os elementos da base de $\mathcal{M}_h$ e $s = \Delta t u_1 + \frac{(\Delta t)^2}{2}(\nabla(c\nabla u_o)+f^o)$, de maneira a obter a a estimativa desejada em (3.16). Substituindo nesta equação as expressões de U^0 e U^1 dadas em (3.13) e (3.14), respectivamente, e as definições apresentadas em (3.11), obtemos :

(3.22)
$$\begin{cases} \left[C^X \otimes C^Y + \Delta t (A^X \otimes C^Y + C^X \otimes A^Y) + \right. \\ \qquad \left. + (\Delta t)^4 A^X \otimes A^Y \right] (\eta^1 - \eta^0) = \phi^1 \\ \text{onde,} \\ \qquad (\phi^1)_{KL} = \langle s, \alpha_K\beta_L \rangle + \Delta t \langle \nabla s, \nabla\alpha_K\beta_L \rangle \\ \qquad\qquad + (\Delta t)^4 \left\langle \frac{\partial^2}{\partial x\partial y} s, \frac{\partial^2}{\partial x\partial y}\alpha_K\beta_L \right\rangle \end{cases}$$

Em seguida, apresentamos a estimativa de erro calculada por Douglas e Dupont em [06] para o esquema (3.22) :

$$\| \Delta_t e^0 \|^2_{L^2} + (\Delta t)^{-1} \| e^1 - e^0 \|^2_{1,0} +$$

$$+ (\Delta t)^4 \| \frac{\partial^2}{\partial x \partial y} (\Delta_t e^0) \|^2_{1,0} \leq$$

(3.23)

$$\leq C \left[\| \Delta_t \delta u^0 \|^2_{1,0} + \right.$$

$$\left. + (\Delta t)^4 \| \frac{\partial^2}{\partial x \partial y} (\Delta_t \delta u^0) \|^2_{L^2} + (\Delta t)^4 \right]$$

Com as estimativas dadas em (3.21) e (3.23) podemos concluir que η^0 e η^1 são aproximações iniciais satisfatórias, ou seja, estão de acordo com condição exigida em (3.16).

Note que (3.22) não pode ser fatorado como em (3.20), logo, vamos modificá-la da seguinte forma :

$$\left[C^x \otimes C^y + \Delta t (A^x \otimes C^y + C^x \otimes A^y) + \right.$$

$$\left. + (\Delta t)^2 A^x \otimes A^y \right] (\eta^1 - \eta^0) = \left[-(\Delta t)^4 A^x \otimes A^y + \right.$$

$$\left. + (\Delta t)^2 A^x \otimes A^y \right] (\eta^1 - \eta^0) + \phi^1$$

Agrupando os termos da equação, ficamos com o seguinte processo iterativo para o cálculo de η^1 :

$$
(3.24)\quad
\begin{cases}
\left(C^X + (\Delta t)A^X \right) \otimes \left(C^Y + (\Delta t)A^Y \right) Z^{(q+1)} = \\
\qquad = (\Delta t)^2(1-(\Delta t)^2)\, A^X \otimes A^Y Z^{(q)} + \phi^1 \\
\text{onde,} \quad
\begin{cases}
Z = \eta^1 - \eta^0 \\
\left(\phi^1 \right)_{KL} = \langle s, \alpha_K \beta_L \rangle + \Delta t \langle \nabla s, \nabla \alpha_K \beta_L \rangle \\
\qquad\qquad + (\Delta t)^4 \langle \dfrac{\partial^2}{\partial x \partial y} s, \dfrac{\partial^2}{\partial x \partial y} \alpha_K \beta_L \rangle \\
\text{com} \quad s = \Delta t\, u_1 + \dfrac{(\Delta t)^2}{2} \left(\nabla (c \nabla u_0) + f^0 \right)
\end{cases}
\end{cases}
$$

Se $e^{(q)} = \left(\eta^1 - \eta^0 \right) - \left(\eta^1 - \eta^0 \right)^{(q)}$, é possível mostrar que :

$$
\| e^{(q+1)} \|^2_{L^2} + \Delta t \, \| e^{(q+1)} \|^2_{1,0} +
$$

$$
+ \frac{1}{2} (\Delta t)^2 (1+(\Delta t)^2) \left\| \frac{\partial^2}{\partial x \partial y} e^{(q+1)} \right\|^2_{L^2} \leq
$$

$$
\leq \frac{1}{2} (\Delta t)^2 (1-(\Delta t)^2) \left\| \frac{\partial^2}{\partial x \partial y} e^{(q)} \right\|^2_{L^2}
$$

portanto, a iteração converge.

3.4 - PROCEDIMENTO GERAL

Nesta seção apresentaremos resumidamente o algoritmo usado para determinar as soluções aproximadas para a equação de propagação de ondas, (2.4). Em cada passo de tempo, é

preciso resolver o seguinte sistema linear :

$$(C^X + \mu A^X) \otimes (C^Y + \mu A^Y) Z = B$$

onde,

no tempo t = 0, temos :

$$(3.25)\quad \begin{cases} \mu = 1 \\ Z = \eta^0 \\ (B)_{KL} = < u_0 , \alpha_K \beta_L > + < \nabla u_0 , \nabla \alpha_K \beta_L > + \\ \qquad + < \dfrac{\partial^2}{\partial x \partial y} u_0 , \dfrac{\partial^2}{\partial x \partial y} \alpha_K \beta_L > \end{cases}$$

no tempo t = Δt, temos :

$$(3.26)\quad \begin{cases} \mu = \Delta t \\ Z = (\eta^1 - \eta^0)^{(q+1)} \\ (B)_{KL} = (\Delta t)^2 (1-(\Delta t)^2) A^X \otimes A^Y (\eta^1 - \eta^0)^{(q)} + (\phi^1)_{KL} \\ \text{com,} \quad \begin{cases} (\phi^1)_{KL} = < s, \alpha_K \beta_L > + \Delta t < \nabla s, \nabla \alpha_K \beta_L > \\ \qquad + (\Delta t)^4 < \dfrac{\partial^2}{\partial x \partial y} s, \dfrac{\partial^2}{\partial x \partial y} \alpha_K \beta_L > \\ s = \Delta t \, u_1 + \dfrac{(\Delta t)^2}{2} (\nabla (c \nabla u_0) + f^0) \end{cases} \end{cases}$$

nos tempos $t = i \Delta t$, $i = 2, 3, \ldots, \hat{M}$, temos :

$$(3.27)\quad \begin{cases} \mu = \lambda (\Delta t)^2 \\ Z = \eta^{M+1} - 2\eta^M + \eta^{M-1}, \quad M \geq 1 \\ (B)_{KL} = (\Delta t)^2 (- < c \nabla U^M, \nabla \alpha_K \beta_L > + < f^M, \alpha_K \beta_L >) \end{cases}$$

Este procedimento é, portanto, um esquema de passo múltiplo, ou seja, a solução em um determinado tempo depende das soluções nos dois tempos imediatamente anteriores. Para iniciarmos o processo são necessárias as condições iniciais dadas pelos esquemas (3.25) e (3.26).

4 - IMPLEMENTACÃO COMPUTACIONAL

4.1 - INTRODUCÃO

Neste capítulo descreveremos como elaboramos um programa que resolve os procedimentos descritos em (3.25),(3.26) e (3.27) que implementado em um computador fornece a solução da equação de propagação de ondas (2.4). Primeiramente, apresentaremos a base para o espaço de aproximação utilizado no nosso trabalho - espaço de Hermite Cúbico.

Nosso programa foi desenvolvido na linguagem FORTRAN e implementado, inicialmente, em microcomputadores do tipo IBM-PC. Foi possível testar nestes aparelhos somente os cálculos e a ordem de aproximação das soluções aproximadas nos tempos iniciais, U^0 e U^1. Passamos, então, a usar os computador VAX da UNICAMP, onde obtemos os resultados que serão apresentados no próximo capítulo.

4.2 - BASE PARA O ELEMENTO FINITO

No nosso trabalho escolhemos elementos cúbicos em cada direção como base para $\mathcal{M}$,ou seja, $\mathcal{M}_{hx}=\mathcal{M}_{hy}=H^{(2)}(\Delta)$. Uma base conveniente para o espaço de Hermite Cúbico, $H^{(2)}(\Delta)$, é :

$$\left\{ \alpha_{i,k}(x),\ i=1,\ldots,N+1 \quad e \quad K=0,1 \right\}$$

onde os elementos $\alpha_{i,K}(x)$ possuem as seguintes propriedades :

$$\begin{cases} \alpha_{i,0}(x_j) = \delta_{ij} & i,j = 1,\ldots,N+1 \\ \alpha'_{i,0}(x_j) = 0 \end{cases}$$

$$\begin{cases} \alpha_{i,1}(x_j) = 0 & i,j = 1,\ldots,N+1 \\ \alpha'_{i,1}(x_j) = \delta_{ij} \end{cases}$$

com $\delta_{ij} = \begin{cases} 1 & \text{se } i=j \\ 0 & \text{se } i \neq j \end{cases}$

e $\alpha_{i,k}(x)$ é zero fora do intervalo $[x_{i-1}, x_{i+1}]$. Na fig. 4.1 temos os gráficos das funções de base $\alpha_{i,0}$ e $\alpha_{i,1}$ definidas em $[x_{i-1}, x_{i+1}]$.

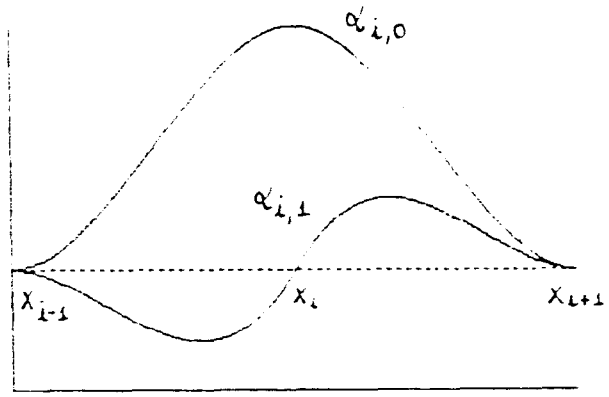


FIGURA 4.1

Devido à condição de solução nula na fronteira (2.4),

não tomaremos na base as funções $\alpha_{1,0}$ e $\alpha_{N+1,0}$. Este novo conjunto gera o espaço das funções $u \in H^{(2)}(\Delta)$ que se anulam na fronteira, denominado $H_0^{(2)}(\Delta)$. Se $f(x) \in H_0^{(2)}(\Delta)$, temos :

$$f(x) = \sum_{i=2}^{N} \eta_i^0 \, \alpha_{i,0}(x) + \sum_{i=1}^{N+1} \eta_i^1 \, \alpha_{i,1}(x) \qquad (4.1)$$

Portanto, a dimensão do espaço $H_0^{(2)}(\Delta)$ é 2N. Agora, considere a seguinte renumeração dos elementos da base :

$$(4.2) \quad \begin{cases} \tilde{\alpha}_{2i-1}(x) = \alpha_{i,1}(x) & , \; i = 1,2,\ldots,N \\ \tilde{\alpha}_{2N}(x) = \alpha_{N+1,1}(x) & \\ \tilde{\alpha}_{2i-2}(x) = \alpha_{i,0}(x) & , \; i = 2,3,\ldots,N \end{cases}$$

Substituindo (4.2) em (4.1) e usando as propriedades das funções de base, podemos escrever f(x) como

$$f(x) = \sum_{i=1}^{2N} \tilde{\eta}_i \, \tilde{\alpha}_i(x) \qquad (4.3)$$

$$\text{com} \quad \begin{cases} f(x_1) = f(x_{N+1}) = 0 & \\ f(x_i) = \tilde{\eta}_{2i-2} & , i = 2,3,\ldots,N \\ f'(x_i) = \tilde{\eta}_{2i-1} & , i = 1,2,\ldots,N \\ f'(x_{N+1}) = \tilde{\eta}_{2N} & \end{cases}$$

Normalmente, o procedimento adotado nos cálculos que envolvem as funções de base é tomá-las definidas no elemento padrão $[-1,1]$ e usar uma mudança de variável que leva no elemento genérico $[x_i, x_{i+1}]$.

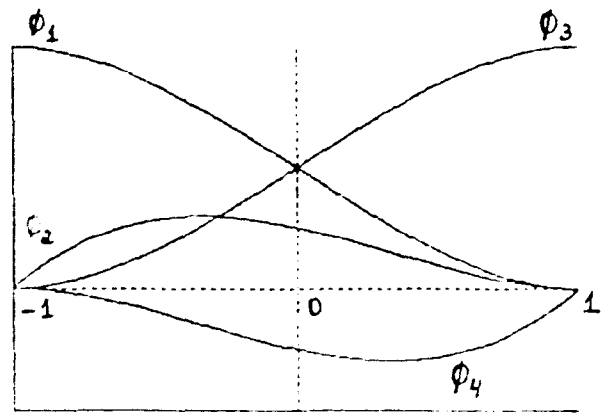


FIGURA 4.2 - Base Elemento Padrão

As definições das funções de base no elemento padrão, denominadas base local, são dadas por :

$$\begin{cases} \phi_1(t) = 0.25(2+t)(1-t)^2 \\ \phi_2(t) = 0.25(1+t)(1-t)^2 \\ \phi_3(t) = 0.25(2-t)(1+t)^2 \\ \phi_4(t) = 0.25(t-1)(1+t)^2 \end{cases}$$

Seja $x(t)$ a transformação que faz a seguinte mudança de variável :

$$\begin{array}{rcl} x(t): [-1,1] & \longrightarrow & [x_i, x_{i+1}] \\ t & \longrightarrow & x(t) \end{array} \qquad (4.4)$$

onde $x(t) = \frac{1}{2}((x_i + x_{i+1}) + (x_{i+1} - x_i)t)$. Portanto, as relações entre as funções de base definidas no elemento genérico $[x_i, x_{i+1}]$ e as funções de base local são dadas por :

$$(4.5)\qquad \begin{cases} \tilde{\alpha}_{2i-2}(x(t)) = \phi_1(t) \\ \tilde{\alpha}_{2i}(x(t)) = \phi_3(t) \end{cases} \qquad \begin{cases} \tilde{\alpha}_{2i-1}(x(t)) = \frac{h}{2}\,\phi_2(t) \\ \tilde{\alpha}_{2i+1}(x(t)) = \frac{h}{2}\,\phi_4(t) \end{cases}$$

No nosso trabalho optamos pelo conjunto de funções $\{\alpha_i\}_{i=1}^{2N}$, definido por

$$(4.6)\qquad \begin{cases} \alpha_1 = \frac{2}{h}\,\tilde{\alpha}_1 & \\ \alpha_{2i-2} = \tilde{\alpha}_{2i-2} & i = 2,3,\ldots,N \\ \alpha_{2i-1} = \frac{2}{h}\,\tilde{\alpha}_{2i-1} & i = 1,2,\ldots,N \\ \alpha_{2N} = \frac{2}{h}\,\tilde{\alpha}_{2N} & \end{cases}$$

como base para $H_0^{(2)}(\Delta)$. Portanto, substituindo (4.6) em (4.5) obtemos agora a seguinte relação entre as funções de base definidas no elemento genérico $[x_i, x_{i+1}]$ e no elemento padrão:

$$(4.7)\qquad \begin{cases} \alpha_{2i-2}(x(t)) = \phi_1(t) \\ \alpha_{2i}(x(t)) = \phi_3(t) \end{cases} \qquad \begin{cases} \alpha_{2i-1}(x(t)) = \phi_2(t) \\ \alpha_{2i+1}(x(t)) = \phi_4(t) \end{cases}$$

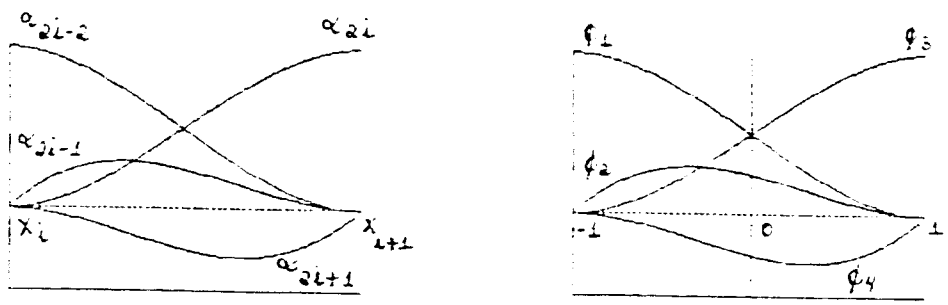


FIGURA 4.3

Escrevendo $f(x) \in H_o^{(2)}(\Delta)$ em função desta nova base e usando (4.3) e (4.6), temos :

$$f(x) = \sum_{i=1}^{2N} \eta_i \, \alpha_i(x)$$

com
$$\begin{cases} f(x_1) = f(x_{N+1}) = 0 \\ f(x_i) = \eta_{2i-2} & ,i = 2,3,\ldots,N \\ f'(x_i) = \frac{2}{h}\,\eta_{2i-1} & ,i = 1,2,\ldots,N \\ f'(x_{N+1}) = \frac{2}{h}\,\eta_{2N} \end{cases}$$

Das relações acima podemos observar que o vetor η, contém os valores de $f(x)$ e de $\frac{2}{h}\,f'(x)$ calculados nos pontos da partição em x.

Tomando elementos cúbicos em cada direção, ou seja, $\mathcal{M}_{hx}=\mathcal{M}_{hy}=H_o^{(2)}(\Delta)$, o espaço de aproximação $\mathcal{M}_h=\mathcal{M}_{hx}\otimes\mathcal{M}_{hy}$ é gerado por $\{\alpha_1\alpha_1,\alpha_1\alpha_2,\ldots,\alpha_1\alpha_{2N},\ldots,\alpha_{2N}\alpha_{2N}\}$. Escrevendo a solução aproximada a cada nível de tempo em função desta base, temos :

$$U^M(x,y) = \sum_{i=1}^{2N}\sum_{j=1}^{2N} \eta_{ij}^M \, \alpha_i(x)\alpha_j(y) \qquad (4.8)$$

onde o vetor η, de comprimento $(2N)^2$, contém os valores da solução aproximada e suas derivadas parciais calculadas nos pontos da rede, ou seja,

$$U^M(x_i,y_j) = \eta^M_{2i-2,2j-2} \qquad i,j=2,3,\ldots,N$$

$$U_x^M(x_i,y_j) = \frac{2}{h}\eta^M_{2i-1,2j-2} \qquad \begin{cases} i=1,2,\ldots,N \\ j=2,3,\ldots,N \end{cases}$$

$$U_x^M(x_{N+1},y_j) = \frac{2}{h}\eta^M_{2N,2j-2} \qquad j=2,3,\ldots,N$$

$$U_y^M(x_i,y_j) = \frac{2}{h}\eta^M_{2i-2,2j-1} \qquad \begin{cases} i=2,3,\ldots,N \\ j=1,2,\ldots,N \end{cases}$$

$$U_y^M(x_i,y_{N+1}) = \frac{2}{h}\eta^M_{2j-2,2N} \qquad i=2,3,\ldots,N$$

$$U_{xy}^M(x_i,y_j) = \frac{4}{h^2}\eta^M_{2i-1,2j-1} \qquad i,j=1,2,\ldots,N$$

$$U_{xy}^M(x_{N+1},y_j) = \frac{4}{h^2}\eta^M_{2N,2j-1} \qquad j=1,2,\ldots,N$$

$$U_{xy}^M(x_i,y_{N+1}) = \frac{4}{h^2}\eta^M_{2i-1,2N} \qquad i=1,2,\ldots,N$$

$$U_{xy}^M(x_{N+1},y_{N+1}) = \frac{4}{h^2}\eta^M_{2N,2N}$$

onde $U_x^M = \frac{\partial U^M}{\partial x}$, $U_y^M = \frac{\partial U^M}{\partial y}$ e $U_{xy}^M = \frac{\partial^2 U^M}{\partial x \partial y}$.

Com estas relações podemos localizar no vetor η os resultados que desejarmos.

4.3 - PROGRAMA COMPUTACIONAL

4.3.1 - FLUXOGRAMA

Apresentaremos agora o fluxograma que resolve numericamente a equação de propagação de ondas (2.4), usando os procedimentos propostos em (3.25), (3.26) e (3.27).

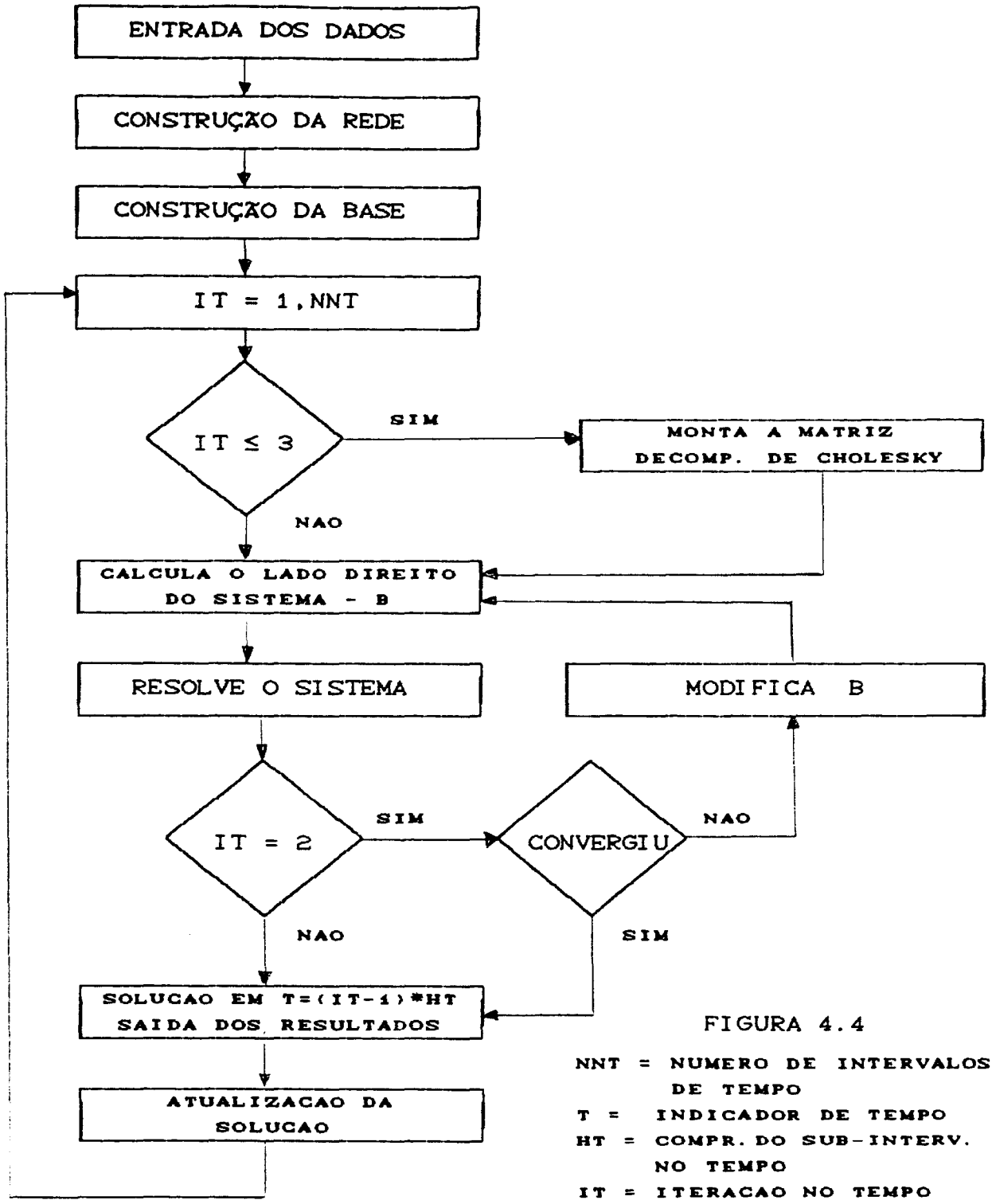


FIGURA 4.4

NNT = NUMERO DE INTERVALOS DE TEMPO
T = INDICADOR DE TEMPO
HT = COMPR. DO SUB-INTERV. NO TEMPO
IT = ITERACAO NO TEMPO

Mostramos abaixo as sub-rotinas que realizam os procedimentos citados na figura 4.4. Apresentaremos cada sub-rotina detalhadamente no decorrer deste capítulo.

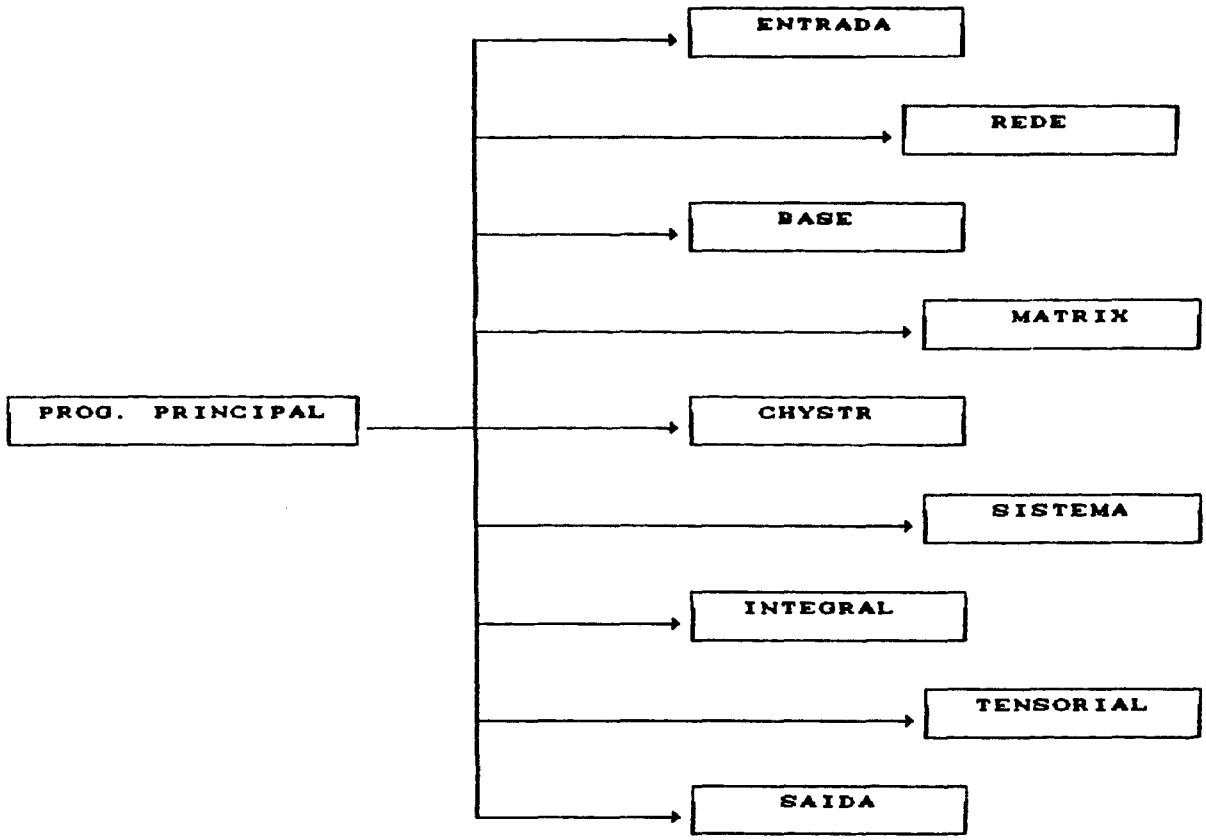


FIGURA 4.5

4.3.2 - VARIÁVEIS COMUNS

Alguns elementos do nosso programa estão organizados em blocos comuns, listados abaixo :

** COMMON / ELEMENTO / N,M

N → Número de elementos por direção

M → 2*N

** COMMON / DOMINIO / X1,X2,Y1,Y2,T1,T2

[X1,X2]x[Y1,Y2] → Domínio de definição das variáveis espaciais

[T1,T2] → Intervalo de definição da variável t

** COMMON / PCENTR / XO,YO

(XO,YO) → Ponto central do domínio das variáveis espaciais

** COMMON / PARTIÇÃO / H,HT

H → Comprimento so sub-intervalo no espaço

HT → Comprimento do sub-intervalo no tempo

** COMMON / PTEMPO / IT

IT → Indicador do nível de tempo

** COMMON / NPINTEG / NPI

NPI → Número de pontos de integração por direção na fórmula da Quadratura Gaussiana

** COMMON / PARAMETRO / LAMBDA

LAMBDA → Parâmetro de estabilidade

** COMMON / PESO / T,Z

T → Vetor com os zeros do polinômio de Legendre

Z → Vetor com os pesos da fórmula da Quadratura Gaussiana

** COMMON / BANDA / IWB

IWB → Comprimento da semi-banda da matriz do algoritmo

** COMMON / FUNCOES / FI,DFIX,DFIY,D2FIXY

FI → Matriz com o produto entre as funções de base

DFIX → Matriz com o produto entre as funções de base e suas derivadas em relação a variável x

DFIY → Matriz com o produto entre as funções de base e suas derivadas em relação a variável y

D2FIXY → Matriz com o produto entre as derivadas, em relação a x e y, das funções de base

4.3.3 - PROGRAMA PRINCIPAL

Vamos apresentar inicialmente as variáveis que aparecem no programa principal e que não estão organizadas nos blocos comuns definidos na seção anterior. Na construção da

rede, temos :

X ⟶ Vetor que contém a partição em x

Y ⟶ Vetor que contém a partição em y

TP ⟶ Vetor que contém a partição em t

NT ⟶ Número de intervalos de tempo

ISE ⟶ Indica na sub-rotina ENTRADA se o problema possui solução exata

As constantes ITERMAX e EPS1 são os critérios de parada do processo iterativo no cálculo da solução em $t=\Delta t$. As matrizes U0, U1, e UN guardam as soluções aproximadas nos tempos $t=0$, $t=\Delta t$ e $t=M\Delta t$ $M\geq 2$, respectivamente. Em UPPER armazenaremos a matriz do sistema em cada iteração do tempo. As matrizes BS e Z1 são usadas no cálculo iterativo da solução aproximada no tempo $t=\Delta t$ e o vetor U aparece na sub-rotina que constroi o arquivo para saída gráfica.

```
C     ***************************************************************
C     *  PROGRAMA QUE RESOLVE A EQUACAO DA ONDA BIDIMENSIONAL       *
C     *  PELO METODO DE GALERKIN COM DIRECOES ALTERNADAS COM        *
C     *  CONDICOES DE FRONTEIRA TIPO DIRICHLET                      *
C     ***************************************************************
      PARAMETER ( NMIN = 40 )
      PARAMETER ( NEXP = NMIN*NMIN )
      PARAMETER ( NMAX = 80 )
      PARAMETER ( NTEMP = 130 )
      REAL*8 X1,X2,Y1,Y2,T1,T2,H,HT,X(NMIN),Y(NMIN),TP(NTEMP)
      REAL*8 U0(NMAX,NMAX),U1(NMAX,NMAX),UN(NMAX,NMAX)
      REAL*8 UPPER(NMAX,4),B(NMAX,NMAX),U1I(NMAX,NMAX),U(NEXP)
      REAL*8 T(5),Z(5),FI(5,5,4,4),DFIX(5,5,4,4)
      REAL*8 DFIY(5,5,4,4),D2FIXY(5,5,4,4)
      REAL*8 EPS1,LAMBDA,PR,EMAX,ER,X0,Y0
```

```
      INTEGER N,M,,NT,NNT,NPI,IWB,NITER,ITERMAX,IT,IMOD,ISE
      CHARACTER RES*1


      COMMON / ELEMENTO / N,M
      COMMON / DOMINIO / X1,X2,Y1,Y2,T1,T2
      COMMON / PCENTR / XO,YO
      COMMON / PARTICAO / H,HT
      COMMON / NPINTEG / NPI
      COMMON / BANDA / IWB
      COMMON / PESO / T,Z
      COMMON / FUNCOES / FI,DFIX,DFIY,D2FIXY
      COMMON / PTEMPO / IT
      COMMON / PARAMETRO / LAMBDA


C     **  ENTRADA DOS DADOS  **

      CALL ENTRADA(NT,EPS1,ITERMAX,ISE)

      IWB = 3
      M = 2*N
      NNT = NT + 1
      NITER = 0

C     **  CONSTRUCAO DA REDE  **

      CALL REDE(NT,X,Y,TP)

C     **  DADOS PARA A INTEGRACAO  **

      CALL BASE

C     **  APROXIMACAO INICIAL (PROCESSO ITERATIVO EM T=HT)  **

      DO 10 I=1,M
      DO 10 J=1,M
      U1I(I,J) = 0.DO
10    CONTINUE

C     **  ITERACAO NO TEMPO  **

      DO 20 IT=1,NNT

C     **  MONTAGEM DA MATRIZ E SUA DECOMPOSICAO DE CHOLESKY  **

      IF(IT.LT.4) THEN
      IF(IT.EQ.1) THEN
```

```
      PR = 1.DO
      CALL MATRIX(NMAX,PR,UPPER)
      CALL CHKYSTR(NMAX,UPPER)
      ELSE IF(IT.EQ.2) THEN
      PR = HT
      CALL MATRIX(NMAX,PR,UPPER)
      CALL CHKYSTR(NMAX,UPPER)
      ELSE
      PR = LAMBDA*HT*HT
      CALL MATRIX(NMAX,PR,UPPER)
      CALL CHKYSTR(NMAX,UPPER)
      END IF
      END IF

C     **  CALCULO DO LADO DIREITO DO SISTEMA - B  **

      CALL INTEGRAL(NMAX,X,Y,TP,U1,B)

C     **  RESOLUCAO DO SISTEMA  **

      DO 30 I=1,M
      DO 30 J=1,M
      UN(I,J) = B(I,J)
30    CONTINUE

80    CALL SISTEMA(NMAX,UPPER,UN)

C     **  PROCESSO ITERATIVO - SOLUCAO NO TEMPO T=HT  **

      IF(IT.EQ.2) THEN

C     **  CRITERIO DE PARADA  **

      EMAX = 0.DO
      DO 40 I=1,M
      DO 40 J=1,M
      ER = DABS( UN(I,J) - U1I(I,J) )
      IF(ER.GT.EMAX) EMAX = ER
40    CONTINUE

      IF(EMAX.LE.EPS1) GO TO 50

      NITER = NITER + 1

      IF(NITER.GE.ITERMAX) GO TO 50

C     **  ATUALIZACAO DA SOLUCAO NO PROCESSO ITERATIVO  **
```

```
      DO 60 I=1,M
      DO 60 J=1,M
      U1I(I,J) = UN(I,J)
60    CONTINUE

C     ** CALCULO DO PRODUTO TENSORIAL **

      CALL TENSORIAL(NMAX,NITER,U1I,UN)

      DO 70 I=1,M
      DO 70 J=1,M
      UN(I,J) = UN(I,J) + B(I,J)
70    CONTINUE

      GO TO 80

      END IF

C     ** CALCULO DA SOLUCAO NO TEMPO T=(IT-1)*HT **

50    IF(IT.LT.3) THEN
      IF(IT.EQ.1) THEN
      DO 90 I=1,M
      DO 90 J=1,M
      UO(I,J) = UN(I,J)
90    CONTINUE
      ELSE
      DO 100 I=1,M
      DO 100 J=1,M
      U1(I,J) = UN(I,J) + UO(I,J)
100   CONTINUE
      END IF
      ELSE
      DO 110 I=1,M
      DO 110 J=1,M
      UN(I,J) = UN(I,J) + 2.DO*U1(I,J) - UO(I,J)
110   CONTINUE
      END IF

C     ** SAIDA DOS RESULTADOS **

      IF(IT.LT.4) THEN
      CALL SAIDA(NMAX,X,Y,TP,NT,NITER,ISE,U,UN)
      ELSE
      IMOD = MOD(IT,4)
      IF(IMOD.EQ.0.OR.IT.EQ.NNT) THEN
      CALL SAIDA(NMAX,X,Y,TP,NT,NITER,ISE,U,UN)
      END IF
```

```
      END IF

C     ** ATUALIZACAO DA SOLUCAO **

      IF(IT.GT.2) THEN
      DO 120 I=1,M
      DO 120 J=1,M
      UO(I,J) = U1(I,J)
      U1(I,J) = UN(I,J)
120   CONTINUE
      END IF

20    CONTINUE

      STOP
      END
```

4.3.4 - ENTRADA DOS DADOS

Na sub-rotina ENTRADA, descrita abaixo, fornecemos os dados necessários ao problema. Acreditamos que devido a sua simplicidade não são necessárias maiores explicações.

```
C     ****************************************
C     *        ENTRADA  DOS  DADOS           *
C     ****************************************

      SUBROUTINE ENTRADA(NT,EPS1,ITERMAX,ISE)
      REAL*8 X1,X2,Y1,Y2,T1,T2,EPS1,LAMBDA,XO,YO
      INTEGER N,M,NT,NPI,ITERMAX,ISE

      COMMON / DOMINIO / X1,X2,Y1,Y2,T1,T2
      COMMON / ELEMENTO / N,M
      COMMON / NPINTEG / NPI
      COMMON / PCENTR / XO,YO
      COMMON / PARAMETRO / LAMBDA

      WRITE(*,10)
```

```
10      FORMAT(/,T10,'PROGRAMA QUE TESTA A SOLUCAO EM T = 0',
     *  //,T20,'ENTRADA DOS DADOS : ',//,T5,'DOMINIO DE',
     *  ' DEFINICAO : [X1,X2]x[Y1,Y2]',/,T5,'ENTRE COM O',
     *  ' VALOR DE :',/,T10,'X1 = ',$)
        READ(*,*) X1
        WRITE(*,20)
20      FORMAT(/,T10,'X2 = ',$)
        READ(*,*) X2
        WRITE(*,30)
30      FORMAT(/,T10,'Y1 = ',$)
        READ(*,*) Y1
        WRITE(*,40)
40      FORMAT(/,T10,'Y2 = ',$)
        READ(*,*) Y2
        WRITE(*,50)
50      FORMAT(/,T5,'ENTRE COM O PONTO CENTRAL DO DOMINIO ',
     *  '[XO,YO] :',/,T5,'ENTRE COM O VALOR DE : ',T10,'XO = '
     *  ,$)
        READ(*,*) XO
        WRITE(*,60)
60      FORMAT(/,T10,'YO = ',$)
        READ(*,*) YO
        WRITE(*,70)
70      FORMAT(/,T5,'NUMERO DE ELEMENTOS POR DIRECAO : ',$)
        READ(*,*) N
        WRITE(*,80)
80      FORMAT(/,T5,'INTERVALO DE TEMPO : [T1,T2]',/,T5,
        'ENTRE COM O VALOR DE : ',/,T10,'T1 = ',$)
        READ(*,*) T1
        WRITE(*,90)
90      FORMAT(/,T10,'T2 = ',$)
        READ(*,*) T2
        WRITE(*,100)
100     FORMAT(/,T5,'NUMERO DE INTERVALOS DE TEMPO : ',$)
        READ(*,*) NT
        WRITE(*,110)
110     FORMAT(/,T5,'NUMERO DE PONTOS DE INTEGRACAO (2,3 OU',
     *  ' 4)' : ',$)
        READ(*,*) NPI
        WRITE(*,120)
120     FORMAT(//,T5,'PROCESSO ITERATIVO NO CALCULO DA',
     *  ' SOLUCAO NO TEMPO T=1 :',//,T5,'ENTRE COM O',
     *  ' NUMERO MAXIMO DE ITERACOES : ',$)
        READ(*,*) ITERMAX
        WRITE(*,130)
130     FORMAT(/,T5,'ENTRE COM A PRECISAO DESEJADA : ',$)
        READ(*,*) EPS1
        WRITE(*,140)
```

```
140     FORMAT(//,T5,'ENTRE COM O PARAMETRO DE ESTABILIDADE',
     *   ' : ',$)
        READ(*,*) LAMBDA
        WRITE(*,150)
150     FORMAT(//,T5,'POSSUI SOLUCAO EXATA (SIM=1) ?',$)
        READ(*,*) ISE

        RETURN
        END
```

4.3.5 - CONSTRUCÃO DA REDE

Aqui serão construídos os elementos da rede, ou seja, as partições com os comprimentos dos sub-intervalos no espaço e no tempo. Como vimos, os vetores X e Y guardam as coordenadas no espaço e o vetor TP as coordenadas no tempo.

```
C       ******************************************
C       *          CONSTRUCAO  DA  REDE          *
C       ******************************************

        SUBROUTINE REDE(NT,X,Y,TP)
        REAL*8 X1,X2,Y1,Y2,T1,T2,H,HT,X(*),Y(*),TP(*)
        INTEGER N,M,NT

        COMMON / ELEMENTO / N,M
        COMMON / DOMINIO / X1,X2,Y1,Y2,T1,T2
        COMMON / PARTICAO / H,HT

C       N = NUMERO DE ELEMENTOS POR DIRECAO
C       NT = NUMERO DE INTERVALOS DE TEMPO
C       H = COMPRIMENTO DO SUB-INTERVALO NO ESPACO
C       HT = COMPRIMENTO DO SUB-INTERVALO NO TEMPO

C       **  PARTICAO NO ESPACO  **

        H = (X2 - X1)/DBLE(FLOAT(N))
        X(1) = X1
```

```
      Y(1) = Y1
      DO 10 I=2,N
      X(I) = X(I-1) + H
      Y(I) = Y(I-1) + H
10    CONTINUE
      X(N+1) = X2
      Y(N+1) = Y2

C     ** PARTICAO NO TEMPO **

      HT = (T2 - T1)/DBLE(FLOAT(NT))
      TP(1) = T1
      DO 20 I=2,NT
      TP(I) = TP(I-1) + HT
20    CONTINUE
      TP(NT+1) = T2

      RETURN
      END
```

4.3.6 - RESOLUCÃO DOS SISTEMAS LINEARES

Em primeiro lugar vamos determinar a estrutura da matriz dos coeficientes $K = C^x+\mu A^x = C^y+\mu A^y$ para o caso de $\mathcal{M}_{hx}=\mathcal{M}_{hy}=H_0^{(2)}(\Delta)$. Usando a numeração das funções de base dada em (4.2) e o fato de α_{2i-2} e α_{2i-1} se anularem fora do intervalo $[x_{i-1},x_{i+1}]$, temos

$$C^x_{kl} = C^y_{kl} = 0$$

para $|k-l| > 3$. Portanto, as matrizes C^x e A^x são de banda sete. Usando as relações dadas (4.7) podemos calcular as seguintes integrais, definidas no elemento genérico $[x_i,x_{i+1}]$,

$$\int_{x_i}^{x_{i+1}} \alpha_k(x)\alpha_l(x)\,dx = \frac{h}{2}\int_{-1}^{1} \phi_i(t)\phi_j(t)\,dt \qquad \begin{cases} k,l=2i-1,\ldots,2i \\ i,j=1,2,\ldots,4 \end{cases}$$

$$\int_{x_i}^{x_{i+1}} \alpha'_k(x)\alpha'_l(x)\,dx = \frac{h}{2}\int_{-1}^{1} \phi'_i(t)\phi'_j(t)\,dt \qquad \begin{cases} k,l=2i-1,\ldots,2i \\ i,j=1,2,\ldots,4 \end{cases}$$

onde $\alpha'_k(x) = \frac{d}{dx}\alpha_k(x)$ e $\phi'_k(t) = \frac{d}{dt}\phi_k(t)$. Desta forma, constroi-se as matrizes

$$(C^x)^e = (C^y)^e = \frac{h}{210}\begin{bmatrix} 78 & 22 & 27 & -13 \\ 22 & 8 & 13 & -6 \\ 27 & 13 & 78 & -22 \\ -13 & -6 & -22 & 8 \end{bmatrix}$$

$$(A^x)^e = (A^y)^e = \frac{1}{210h}\begin{bmatrix} 252 & 42 & -252 & 42 \\ 42 & 112 & -42 & -28 \\ -252 & -42 & 252 & -42 \\ 42 & -28 & -42 & 112 \end{bmatrix}$$

onde $(C^x+\mu A^x)^e$ é denominada matriz de rigidez por elemento e μ é o parâmetro definido em (3.25),(3.26) e (3.27). Fazendo adequadamente o agrupamento dos elementos de $(C^x+\mu A^x)^e$ obtém-se a matriz dos coeficientes, $K = (C^x+\mu A^x)$, onde

$$C^x = C^y = \frac{h}{210}\begin{bmatrix} 8 & 13 & -6 & 0 & & & & & 0 & \\ 13 & 156 & 0 & 27 & -13 & & & & & \\ -6 & 0 & 16 & 13 & -6 & 0 & & & & \\ 0 & 27 & 13 & 156 & 0 & . & . & & & \\ & -13 & -6 & 0 & . & . & . & -13 & & \\ & & 0 & . & . & . & . & -6 & 0 & \\ & & & . & . & . & 156 & 0 & -13 & \\ & 0 & & & . & -6 & 0 & 16 & -6 & \\ & & & & & 0 & -13 & -6 & 8 & \end{bmatrix}$$

(4.9)

$$A^x = A^y = \frac{1}{210h}\begin{bmatrix} 112 & -42 & -28 & 0 & & & & & 0 & \\ -42 & 504 & 0 & -252 & 42 & & & & & \\ -28 & 0 & 224 & -42 & -28 & 0 & & & & \\ 0 & -252 & -42 & 504 & 0 & . & . & & & \\ & 42 & -28 & 0 & . & . & . & 42 & & \\ & & 0 & . & . & . & . & -28 & 0 & \\ & & & . & . & . & 504 & 0 & 42 & \\ & 0 & & & . & -28 & 0 & 224 & -28 & \\ & & & & & 0 & 42 & -28 & 112 & \end{bmatrix}$$

Portanto, a matriz K é de ordem 2N, simétrica, de banda 7 e positiva-definida [16]. Aproveitando esta estrutura, K será armazenada em uma matriz retangular UPPER, de dimensão 2Nx4, contendo a diagonal principal na primeira coluna e as demais diagonais superiores nas colunas restantes. Observe que a montagem da matriz dos coeficientes K só é necessária nas três primeiras iterações, uma vez que K não se modifica nas demais. Apresentaremos agora a sub-rotina que monta a matriz K, que depende do parâmetro μ, denominado PR, do número de elementos em cada direção e de H.

```
C     ************************************************************
C     *   SUB-ROTINA QUE MONTA A MATRIZ DOS COEFICIENTES         *
C     ************************************************************
      SUBROUTINE MATRIX(NMAX,PR,A)
      REAL*8 A(NMAX,*),H,HT,PR,AUX,AUX1,AUX2,HH
      INTEGER N,M,KK

      COMMON / ELEMENTO / N,M
      COMMON / PARTICAO / H,HT

      HH = H*H
      AUX = 210.D0*H

C     **  ZERA A MATRIZ  **

      DO 10 J=1,4
      DO 10 I=1,M
      A(I,J) = 0.D0
10    CONTINUE

C     **  DIAGONAL PRINCIPAL  **

      A(1,1) = (8.D0*HH + 112.D0*PR)/AUX
      A(M,1) = A(1,1)
      AUX1 = (156.D0*HH + 504.D0*PR)/AUX
      AUX2 = (16.D0*HH + 224.D0*PR)/AUX
      KK = 2
      DO 20 I=1,(N-1)
      A(KK,1) = AUX1
      A(KK+1,1) = AUX2
      KK = KK + 2
20    CONTINUE

C     **  PRIMEIRA DIAGONAL SUPERIOR  **

      A(M-1,2) = (-6.D0*HH - 28.D0*PR)/AUX
      AUX1 = (13.D0*HH - 42.D0*PR)/AUX
      KK = 1
      DO 30 I=1,(N-1)
      A(KK,2) = AUX1
      A(KK+1,2) = 0.D0
      KK = KK + 2
30    CONTINUE

C     ** SEGUNDA DIAGONAL SUPERIOR  **

      AUX1 = A(M-1,2)
      AUX2 = (27.D0*HH - 252.D0*PR)/AUX
```

```
      KK = 1
      DO 40 I=1,(N-1)
      A(KK,3) = AUX1
      A(KK+1,3) = AUX2
      KK = KK + 2
40    CONTINUE
      A(M-2,3) = (-13.DO*HH + 42.DO*PR)/AUX

C     ** TERCEIRA DIAGONAL SUPERIOR **

      AUX1 = A(M-2,3)
      A(1,4) = 0.DO
      KK = 2
      DO 50 I=1,(N-2)
      A(KK,4) = AUX1
      A(KK+1,4) = 0.DO
      KK = KK + 2
50    CONTINUE

      RETURN
      END
```

Vimos na seção 3.4 que o sistema que queremos resolver possui a seguinte estrutura :

$$(C^X + \mu A^X) \otimes (C^Y + \mu A^Y)\, Z = B \qquad (4.10)$$

Usando em (4.10) a propriedade 2 do produto tensorial, dada em (3.10), com $A = (C^X + \mu A^X)$, $D = (C^Y + \mu A^Y)$ e $C = B = I$, obtemos um sistema equivalente,

$$[(C^X + \mu A^X) \otimes I\,][\, I \otimes (C^Y + \mu A^Y)]\, Z = B,$$

que pode ser decomposto em dois sistemas mais simples :

$$\begin{cases} [(C^X + \mu A^X) \otimes I\,]\,\gamma = B \\ [\, I \otimes (C^Y + \mu A^Y)]\, Z = \gamma \end{cases}$$

Resumindo, temos que resolver

$$M\,\gamma = B \qquad (4.11)$$

e

$$N\,Z = \gamma \qquad (4.12)$$

com $\begin{cases} M = (C^X + \mu\, A^X) \otimes I \\ N = I \otimes (C^Y + \mu\, A^Y) \end{cases}$

Veremos, em primeiro lugar, a estratégia usada para resolver o segundo sistema, (4.12). Fazendo o produto tensorial, definido em (3.9), entre I e $K=(C^Y+ \mu\, A^Y)$, obtemos a matriz N,

$$N = \begin{bmatrix} K & & 0 \\ & K & \\ & & \ddots \\ 0 & & K \end{bmatrix}$$

diagonal por blocos e de ordem $(2N)^2$. Considere os vetores Z e γ, de comprimentos $(2N)^2$, divididos em 2N blocos de tamanho 2N, ou seja,

$$Z = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_{2N} \end{bmatrix} \qquad \gamma = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{2N} \end{bmatrix}$$

Assim, resolver $N\,Z = \gamma$ é o mesmo que resolver, para cada ι, o sistema

$$K\,Z_\iota = \gamma_\iota \qquad \iota=1,2,\ldots,2N \qquad (4.13)$$

onde K é uma matriz simétrica, de banda 7 e positiva-definida. Para obtermos a solução deste sistema, para cada i, faremos inicialmente a decomposição de Cholesky [04] da matriz K, ou seja, escreveremos K na forma

$$K = L\,L^T$$

onde L é triangular inferior e L^T a transposta da matriz L. Nesta decomposição estaremos aproveitando a estrutura de banda 7 da matriz; entraremos com a matriz UPPER, que foi montada na sub-rotina MATRIX, e sairemos com sua decomposição de Cholesky armazenada na própria matriz UPPER.

```
C       ************************************************************
C       *  SUB-ROTINA QUE FAZ A DECOMPOSIÇÃO DE CHOLESKY           *
C       *  SEM PIVOTEAMENTO DE UMA MATRIZ UPPER, SIMETRICA         *
C       *  E DE SEMI-BANDA IWB.                                    *
C       ************************************************************
        SUBROUTINE CHKYSTR(NMAX,UPPER)
        REAL*8 UPPER(NMAX,*),EPS,SOMA,ARG
        INTEGER N,M,IWB,NN,MM,II,JJ,KJ

        COMMON / ELEMENTO / N,M
        COMMON / BANDA / IWB

        DATA EPS / 1.D-16 /

C       **  CALCULO DA PRIMEIRA LINHA DA MATRIZ UPPER  **

        IF(UPPER(1,1).LT.0.D0) GO TO 70
        UPPER(1,1) = DSQRT(UPPER(1,1))
        IF(DABS(UPPER(1,1)).LE.EPS)  GO TO 60

        DO 10 J=2,(IWB+1)
        UPPER(1,J) = UPPER(1,J)/UPPER(1,1)
10      CONTINUE
```

```
      DO 50 I=2,M
      NN = I + IWB
      SOMA = 0.DO
      MM = 1
      IF(I.GT.(IWB+1)) MM = I - IWB
      DO 20 K = MM,(I-1)
      II = (I-K) + 1
      SOMA = SOMA + UPPER(K,II)*UPPER(K,II)
20    CONTINUE

C     ** CALCULO DA DIAGONAL PRINCIPAL **

C     ** VERIFICA SE A MATRIZ E DEFINIDA-POSITIVA **

      ARG = UPPER(I,1) - SOMA
      IF(ARG.LT.0.DO) GO TO 70

      UPPER(I,1) = DSQRT(ARG)

C     ** CALCULO ACIMA DA DIAGONAL PRINCIPAL **

      IF((M-I).LT.IWB)  NN = M
      DO 40 J = (I+1),NN
      SOMA = 0.DO
      JJ = (J-I) + 1
      DO 30 K = MM,(I-1)
      II = I - K + 1
      KJ = J - K + 1
      IF(KJ.GT.(IWB+1)) GO TO 30
      SOMA = SOMA + UPPER(K,II)*UPPER(K,KJ)
30    CONTINUE

C     ** VERIFICA SE A MATRIZ E SINGULAR **

      IF(DABS(UPPER(I,1)).LE.EPS) GO TO 60

      UPPER(I,JJ) = ( UPPER(I,JJ) - SOMA )/UPPER(I,1)

40    CONTINUE
50    CONTINUE

      RETURN

60    WRITE(*,100)
100   FORMAT(5(/),T10,'NAO E POSSIVEL A DECOMPOSICAO',///)
      STOP

70    WRITE(*,200)
```

```
200     FORMAT(5(/),T10,'MATRIZ NAO  E DEFINIDA-POSITIVA',///)
        STOP

        END
```

Feita a decomposição de Cholesky de K, o sistema (4.13), para cada i, assume a forma $L\,L^T\,Z_i = \gamma_i$, que pode ser representada por dois sistemas mais simples,

$$L\,\tilde{Z}_i = \gamma_i \qquad (4.14)$$

e

$$L^T\,Z_i = \tilde{Z}_i \qquad (4.15)$$

onde Z_i, $\tilde{Z}_i$ e γ_i são vetores de comprimento 2N. Em primeiro lugar determinamos $\tilde{Z}_i$ em (4.14) e depois Z_i em (4.15).

```
C       ************************************************
C       *  SUB-ROTINA QUE RESOLVE   L L^T Z = B ,      *
C       *  ONDE L É UMA MATRIZ TRIANGULAR INFERIOR     *
C       ************************************************

        SUBROUTINE  SOLVECHKY(NMAX,UPPER,B)
        REAL*8  UPPER(NMAX,*),B(*),SOMA
        INTEGER  N,M,IWB,II,MM,JI

        COMMON / ELEMENTO / N,M
        COMMON / BANDA / IWB

C       **  RESOLUCAO DO SISTEMA TRIANGULAR INFERIOR  **

        B(1) = B(1)/UPPER(1,1)

        II = IWB + 1
        DO 20 I=2,M
        SOMA = 0.DO
        MM = 1
        IF(I.GT.II)  MM = I - IWB
        DO 10 J = MM,(I-1)
        JI = I - J + 1
```

```
          SOMA = SOMA + UPPER(J,JI)*B(J)
10        CONTINUE

          B(I) = ( B(I) - SOMA )/UPPER(I,1)

20        CONTINUE

C         ** RESOLUCAO DO SISTEMA TRIANGULAR SUPERIOR **

          B(M) = B(M)/UPPER(M,1)

          DO 40 I=(M-1),1,-1
          SOMA = 0.D0
          MM = M
          IF((M-I).GT.IWB) MM = I + IWB
          DO 30 J = (I+1),MM
          JI = J - I + 1
          SOMA = SOMA + UPPER(I,JI)*B(J)
30        CONTINUE

          B(I) = ( B(I) - SOMA )/UPPER(I,1)

40        CONTINUE

          RETURN
          END
```

Agora, substituindo a decomposição de Cholesky de K na matriz M do primeiro sistema, (4.11), temos :

$$M = (L\ L^T) \otimes I$$

Usando a propriedade 2 do produto tensorial, dada em (3.10), com A = L, B = L^T e C = D = I, obtemos um sistema equivalente,

$$(L \otimes I)\ (L^T \otimes I)\ \gamma = B,$$

que pode ser decomposto em dois sistemas mais simples,

$$(L \otimes I)\,\sigma = B \qquad (4.16)$$

$$(L^T \otimes I)\,\gamma = \sigma \qquad (4.17)$$

Fazendo o produto tensorial que aparece em (4.16), da forma como foi definido em (3.9), encontramos uma matriz triangular inferior por blocos, onde cada bloco (i,j) é uma matriz de dimensão 2N, com a seguinte estrutura

$$\begin{bmatrix} \ell_{ij} & & 0 \\ & \ddots & \\ 0 & & \ell_{ij} \end{bmatrix} \qquad (4.18)$$

onde ℓ_{ij} é um elemento de L. Considere os vetores σ e B, de comprimentos $(2N)^2$, divididos em 2N blocos de tamanhos 2N. Para resolver (4.16) proceda como em (4.14), tratando cada elemento da matriz como um bloco da forma (4.18). Portanto, cada operação deste processo será repetida 2N vezes. Analogamente, resolvemos (4.17).

Resumindo, para obtermos a solução do sistema $(L \otimes I)\,(L^T \otimes I)\,\gamma = B$, podemos aproveitar, com pequenas modificações, a sub-rotina SOLVECCHKY que resolve um sistema da forma $L\,L^T\,\tilde{\gamma} = \tilde{B}$.

```
C     ************************************************************
C     *  SUB-ROTINA QUE RESOLVE (L ⊗ I)(L^T ⊗ I) γ = B.          *
C     *  ONDE L É UMA MATRIZ TRIANGULAR INFERIOR                 *
C     ************************************************************
```

```
      SUBROUTINE  SOLVE(NMAX,UPPER,B)
      REAL*8  UPPER(NMAX,*),B(NMAX,*),SOMA
      INTEGER  N,M,IWB,II,MM,JI

      COMMON / ELEMENTO / N,M
      COMMON / BANDA / IWB
C     ** RESOLUCAO DO SISTEMA TRIANGULAR INFERIOR  **

      DO 1 I=1,M
      B(1,I) = B(1,I)/UPPER(1,1)
1     CONTINUE

      II = IWB + 1
      DO 20 I=2,M
      DO 2 K=1,M
      SOMA = 0.DO
      MM = 1
      IF(I.GT.II) MM = I - IWB
      DO 10 J = MM,(I-1)
      JI = I - J + 1
      SOMA = SOMA + UPPER(J,JI)*B(J,K)
10    CONTINUE

      B(I,K) = ( B(I,K) - SOMA )/UPPER(I,1)

2     CONTINUE
20    CONTINUE

C     **  RESOLUCAO DO SISTEMA  TRIANGULAR  SUPERIOR  **

      DO 3 I=1,M
      B(M,I) = B(M,I)/UPPER(M,1)
3     CONTINUE

      DO 40 I=(M-1),1,-1
      DO 4 K=1,M
      SOMA = 0.DO
      MM = M
      IF((M-I).GT.IWB) MM  = I + IWB
      DO 30 J = (I+1),MM
      JI = J - I + 1
      SOMA = SOMA + UPPER(I,JI)*B(J,K)
30    CONTINUE

      B(I,K) = ( B(I,K) - SOMA )/UPPER(I,1)

4     CONTINUE
```

```
40      CONTINUE

        RETURN
        END
```

Vimos que para resolvermos os sistemas (4.11) e (4.12) é preciso inicialmente a decomposição de Cholesky da matriz K, que é feita na sub-rotina CHKYSTR. Como a montagem de K só é necessária nas 3 primeiras iterações, também a sua decomposição de Cholesky não será necessária nas demais iterações.

Concluindo, para obtermos a solução do sistema (4.10), conhecidos o vetor B e a decomposição de Cholesky de K, temos a sub-rotina SISTEMA, descrita abaixo, que chama os procedimentos SOLVECHKY E SOLVE.

```
C       ******************************************
C       *   SUB-ROTINA QUE RESOLVE O SISTEMA     *
C       ******************************************

        SUBROUTINE SISTEMA(NMAX,UPPER,B)
        REAL*8 UPPER(NMAX,*),B(NMAX,*)
        REAL*8 BB(80)
        INTEGER N,M

        COMMON / ELEMENTO / N,M

C       **  RESOLUCAO DO PRIMEIRO SISTEMA  **

        CALL SOLVE(NMAX,UPPER,B)

C       **  RESOLUCAO DO SEGUNDO SISTEMA  **

        DO 10 I=1,M
        DO 20 J=1,M
        BB(J) = B(I,J)
```

```
20      CONTINUE
        CALL SOLVECHKY(NMAX,UPPER,BB)
        DO 30 K=1,M
        B(I,K) = BB(K)
30      CONTINUE
10      CONTINUE

        RETURN
        END
```

Observe que, apesar do produto tensorial entre as matrizes $(C^X + \mu A^X)$ e $(C^Y + \mu A^Y)$ de (4.10) fornecer sistemas de $(2N)^2$ equações, iremos resolver sistemas cujas matrizes são de dimensão 2Nx4. Portanto, a escolha do método de Elementos Finitos com Direções Alternadas significou um ganho quanto ao espaço de memória requerido.

4.3.7 - MODIFICA B

Vimos que no tempo $t=\Delta t$ temos um processo iterativo onde o termo do lado direito do sistema, B, é dado por

$$B = (\Delta t)^2(1-(\Delta t)^2)\, A^X \otimes A^Y (\eta^1 - \eta^0)^{(q)} + (\phi^1)_{kl}$$

e $(\phi^1)_{kl}$ não se altera em cada iteração. Nosso interesse agora é mostrar como foi construído o primeiro termo de B. Para isto, considere o vetor $(\eta^1 - \eta^0)^{(q)} = Z$, de comprimento $(2N)^2$, com a seguinte estrutura de bloco :

$$Z = \begin{bmatrix} Z_1 \\ Z_2 \\ . \\ . \\ . \\ Z_{2N} \end{bmatrix}$$

Usando (4.9) e fazendo o produto $(A^x \otimes A^y)$ Z obtemos o vetor $v = \mathcal{K}(A^x \otimes A^y)$ Z, onde $\mathcal{K} = (\Delta t)^2(1-(\Delta t)^2)$, com a seguinte estrutura de bloco :

$$v = \frac{\mathcal{K}}{210h} \begin{bmatrix} A^y(112Z_1 - 42Z_2 - 28Z_3) \\ A^y(-42Z_1 + 504Z_2 - 252Z_4 + 42Z_5) \\ A^y(-28Z_1 + 224Z_3 - 42Z_4 - 28Z_5) \\ A^y(-252A_2 - 42Z_3 + 504Z_4 - 252Z_6 + 42Z_7) \\ A^y(42Z_2 - 28Z_3 + 224Z_5 - 42Z_6 - 28Z_7) \\ . \\ . \\ . \\ A^y(42Z_{(2N-2)} - 28Z_{(2N-1)} + 112Z_{2N}) \end{bmatrix} \quad (4.19)$$

Para construirmos o vetor acima é necessária inicialmente a montagem da matriz $A^x = A^y$, definida em (4.9). No programa esta matriz é denominada MAT e possui, devido a sua estrutura, 2N linhas e 4 colunas.

```
C      ************************************************
C      *  SUB-ROTINA QUE CONSTROI A MATRIZ (210H A^X)  *
C      ************************************************

       SUBROUTINE MATRIG(A)
       REAL*8 A(80,*)
```

```
      INTEGER N,M,KK

      COMMON / ELEMENTO / N,M

C     ** DIAGONAL PRINCIPAL **

      A(1,1) = 112.D0
      A(M,1) = 112.D0
      KK = 2
      DO 10 I=1,(N-1)
      A(KK,1) = 504.D0
      A(KK+1,1) = 224.D0
      KK = KK + 2
10    CONTINUE

C     ** PRIMEIRA DIAGONAL SUPERIOR **

      A(M-1,2) = -28.D0
      KK = 1
      DO 20 I=1,(N-1)
      A(KK,2) = -42.D0
      A(KK+1,2) = 0.D0
      KK = KK + 2
20    CONTINUE

C     ** SEGUNDA DIAGONAL SUPERIOR **

      KK = 1
      DO 30 I=1,(N-1)
      A(KK,3) = -28.D0
      A(KK+1,3) = -252.D0
      KK = KK + 2
30    CONTINUE
      A(M-2,3) = 42.D0

C     ** TERCEIRA DIAGONAL SUPERIOR **

      A(1,4) = 0.D0
      KK = 2
      DO 40 I=1,(N-2)
      A(KK,4) = 42.D0
      A(KK+1,4) = 0.D0
      KK = KK + 2
40    CONTINUE

      RETURN
      END
```

Os produtos $A^y Z_i$, $i=1,2,\ldots,2N$, que aparecem na definição do vetor v em (4.19), são feitos separadamente na sub-rotina PRODUTO, descrita abaixo. IWB representa o comprimento da semi-banda da matriz simétrica A^y.

```
C       ************************************************************
C       *    SUB-ROTINA QUE CALCULA O PRODUTO DE UMA               *
C       *    MATRIZ A, SIMETRICA COM ESTRUTURA DE BANDA,           *
C       *    POR UM VETOR Z :   A*Z = B                            *
C       ************************************************************

        SUBROUTINE PRODUTO(A,Z,B)
        REAL*8 Z(*),B(*),SOMA,EPS
        REAL*8 A(80,*)
        INTEGER N,M,IWB,MM,KI

        COMMON / ELEMENTO / N,M
        COMMON / BANDA / IWB

        DATA EPS / 1.D-18 /

        DO 1  I=1,M
        IF(DABS(Z(I)).LE.EPS) Z(I) = 0.DO
1       CONTINUE

C       **  CALCULO DE B(1)  **

        SOMA = 0.DO
        MM = 1 + IWB
        DO 10 J=1,MM
        SOMA = SOMA + A(1,J)*Z(J)
10      CONTINUE
        B(1) = SOMA

        DO 20 I=2,M

C       ** CALCULO DOS ELEMENTOS DA DIAGONAL E ACIMA DELA  **

        SOMA = 0.DO
        MM = I + IWB
        IF((M-I).LT.IWB) MM = M
        DO 30 J=I,MM
        KI = J - I + 1
```

```
          SOMA = SOMA + A(I,KI)*Z(J)
30        CONTINUE

C         ** CALCULO DOS ELEMENTOS ABAIXO DA DIAGONAL **

          MM = I - IWB
          IF((I-IWB).LE.0) MM = 1
          DO 40 J=MM,(I-1)
          KI = I - J + 1
          SOMA = SOMA + A(J,KI)*Z(J)
40        CONTINUE
          B(I) = SOMA
20        CONTINUE

          RETURN
          END
```

Observe que a montagem da matriz A^y para o cálculo de v é necessária apenas no primeiro passo do processo iterativo, ou seja, quando NITER = 1. Na sub-rotina TENSORIAL, descrita abaixo, efetivamos o cálculo de v.

```
C         ************************************************
C         *  SUB-ROTINA QUE CALCULA O PRIMEIRO TERMO     *
C         *  DE B NO PROCESSO ITERATIVO NO TEMPO T=ΔT    *
C         ************************************************
          SUBROUTINE TENSORIAL(NMAX,NITER,Z1,Z2)
          REAL*8 H,HT,Z1(NMAX,*),Z2(NMAX,*),AUX,AUX1,AUX2,EPS
          REAL*8 MAT(80,4),V1(80),V2(80)
          INTEGER N,M,NITER,II,JJ

          COMMON / NPINTEG / NPI
          COMMON / ELEMENTO / N,M
          COMMON / PARTICAO / H,HT

          DATA EPS / 1.D-18 /

          IF(NITER.EQ.1) THEN
          CALL MATRIG(MAT)
          END IF

          AUX1 = HT*HT
```

```
      AUX2 = 210.D0*210.D0*H*H
      AUX = AUX1*(1.D0 - AUX1)/AUX2

      DO 1 I=1,M
      DO 1 J=1,M
      IF(DABS(Z1(I,J)).LE.EPS) Z1(I,J) = 0.D0
1     CONTINUE

C     ** CALCULO DE Z2(1,I), I=1,M **

      DO 10 I=1,M
      V1(I) = ( 112.D0*Z1(1,I)-42.D0*Z1(2,I)-28.D0*Z1(3,I) )
     * *AUX
10    CONTINUE
      CALL PRODUTO(MAT,V1,V2)
      DO 20 I=1,M
      Z2(1,I) = V2(I)
20    CONTINUE

C     ** CALCULO DE Z2(2,I), I=1,M **

      DO 30 I=1,M
      V1(I) = ( -42.D0*Z1(1,I)+504.D0*Z1(2,I)-252.D0*Z1(4,I)
     *          +42.D0*Z1(5,I) )*AUX
30    CONTINUE
      CALL PRODUTO(MAT,V1,V2)
      DO 40 I=1,M
      Z2(2,I) = V2(I)
40    CONTINUE

C     ** CALCULO DE Z2(3,I), I=1,M **

      DO 50 I=1,M
      V1(I) = ( -28.D0*Z1(1,I)+224.D0*Z1(3,I)-42.D0*Z1(4,I)-
     *           28.D0*Z1(5,I) )*AUX
50    CONTINUE
      CALL PRODUTO(MAT,V1,V2)
      DO 60 I=1,M
      Z2(3,I) = V2(I)
60    CONTINUE

C     ** CALCULO DE Z2(J,I), J=4,(M-3) E I=1,M **

      II = 2
      JJ = 4
      DO 70 K=1,(N-3)

C     ** CALCULO DE Z2(2*J,I), J=2,(N-2) E I=1,M **
```

```
      DO 80 I=1,M
      V1(I) = ( -252.D0*Z1(II,I)-42.D0*Z1(II+1,I)+504.D0*
     *    Z1(II+2,I)-252.D0*Z1(II+4,I)+42.D0*Z1(II+5,I) )*AUX
80    CONTINUE
      CALL PRODUTO(MAT,V1,V2)
      DO 90 I=1,M
      Z2(JJ,I) = V2(I)
90    CONTINUE

C     ** CALCULO DE Z2(2*J+1,I), J=2,(N-2) E I=1,M **

      DO 100 I=1,M
      V1(I) = ( 42.D0*Z1(II,I)-28.D0*Z1(II+1,I)+224.D0*
     *    Z1(II+3,I)-42.D0*Z1(II+4,I)-28.D0*Z1(II+5,I) )*AUX
100   CONTINUE
      CALL PRODUTO(MAT,V1,V2)
      DO 110 I=1,M
      Z2(JJ+1,I) = V2(I)
110   CONTINUE

      II = II + 2
      JJ = JJ + 2
70    CONTINUE

C     ** CALCULO DE B(M-2,I), I=1,M **

      DO 120 I=1,M
      V1(I) = ( -252.D0*Z1(M-4,I)-42.D0*Z1(M-3,I)+504.D0*
     *          Z1(M-2,I)+42.D0*Z1(M,I) )*AUX
120   CONTINUE
      CALL PRODUTO(MAT,V1,V2)
      DO 130 I=1,M
      Z2(M-2,I) = V2(I)
130   CONTINUE

C     ** CALCULO DE B(M-1,I), I=1,M **

      DO 140 I=1,M
      V1(I) = ( 42.D0*Z1(M-4,I)-28.D0*Z1(M-3,I)+224.D0*
     *          Z1(M-1,I)-28.D0*Z1(M,I) )*AUX
140   CONTINUE
      CALL PRODUTO(MAT,V1,V2)
      DO 150 I=1,M
      Z2(M-1,I) = V2(I)
150   CONTINUE

C     ** CALCULO DE B(M,I), I=1.,M **
```

```
      DO 160 I=1,M
      V1(I) = ( 42.D0*Z1(M-2,I)-28.D0*Z1(M-1,I)+112.D0*
     *            Z1(M,I) )*AUX
160   CONTINUE
      CALL PRODUTO(MAT,V1,V2)
      DO 170 I=1,M
      Z2(M,I) = V2(I)
170   CONTINUE

      RETURN
      END
```

4.3.8 - CÁLCULO DAS INTEGRAIS

Para obtermos o lado direito do sistema linear, B, é preciso calcular integrais dos seguintes tipos :

- nos tempos iniciais :

$$1 - \langle G(x,y), \alpha_K \alpha_L \rangle \qquad (4.20)$$

$$2 - \langle \nabla G(x,y), \nabla \alpha_K \alpha_L \rangle \qquad (4.21)$$

$$3 - \langle \frac{\partial^2}{\partial x \partial y} G(x,y), \frac{\partial^2}{\partial x \partial y} \alpha_K \alpha_L \rangle \qquad (4.22)$$

onde,

$$t = 0 \quad (4.23) \quad \begin{cases} G(x,y) = u_o(x,y) \\ (B)_{KL} = \langle u_o , \alpha_K \beta_L \rangle + \langle \nabla u_o , \nabla \alpha_K \beta_L \rangle + \\ \qquad + \langle \frac{\partial^2}{\partial x \partial y} u_o , \frac{\partial^2}{\partial x \partial y} \alpha_K \beta_L \rangle \end{cases}$$

$$t = \Delta t \quad (4.24) \quad \begin{cases} G(x,y) = s(x,y) = \Delta t\, u_1 + \dfrac{(\Delta t)^2}{2}\left(\nabla(c\nabla u_0) + f^0 \right) \\ (B)_{KL} = (\Delta t)^2(1-(\Delta t)^2)A^x \otimes A^y(\eta^1 - \eta^0)^{(q)} + (\phi^1)_{KL} \\ \text{onde,} \quad (\phi^1)_{KL} = \langle s, \alpha_K\beta_L \rangle + \Delta t \langle \nabla s, \nabla\alpha_K\beta_L \rangle \\ \qquad + (\Delta t)^4 \left\langle \dfrac{\partial^2}{\partial x\partial y} s, \dfrac{\partial^2}{\partial x\partial y}\alpha_K\beta_L \right\rangle \end{cases}$$

- nos demais tempos :

$$\langle G(x,y), \alpha_K\alpha_L \rangle$$
$$4 - \langle c(x,y)\nabla U^M, \nabla\alpha_K\alpha_L \rangle \qquad (4.25)$$

onde,

$$t = i\Delta t \quad (4.26) \quad \begin{cases} G(x,y) = f^M = f(x,y,t_M) \\ (B)_{KL} = (\Delta t)^2 \left(- \langle c\nabla U^M, \nabla\alpha_K\beta_L \rangle + \langle f^M, \alpha_K\beta_L \rangle \right) \end{cases}$$

A integral em (4.25) é diferente da integral do tipo dois, (4.21), pois ∇U^M é um vetor, ou seja, o gradiente da solução numérica no tempo imediatamente anterior, e não uma expressão algébrica como no caso de G(x,y). Portanto, existem basicamente quatro tipos de integral que precisamos resolver,(4.20),(4.21),(4.22) e (4.25). O método numérico adotado, em qualquer iteração no tempo, foi Quadratura Gaussiana com a opção de 2,3 ou 4 nós em cada direção por

elemento [05].

A título de exemplo, suponhamos que estejamos interessados em calcular as integrais para o caso de k=3 e L=4. As funções de base $\alpha_3(x)$ e $\alpha_4(y)$ possuem seus suportes definidos em $[x_1,x_3]$ e $[y_2,y_4]$, logo o produto $\alpha_3(x)\alpha_4(y)$ é diferente de zero somente no retângulo $[x_1,x_3]\times[y_2,y_4]$, como mostra as figuras abaixo.

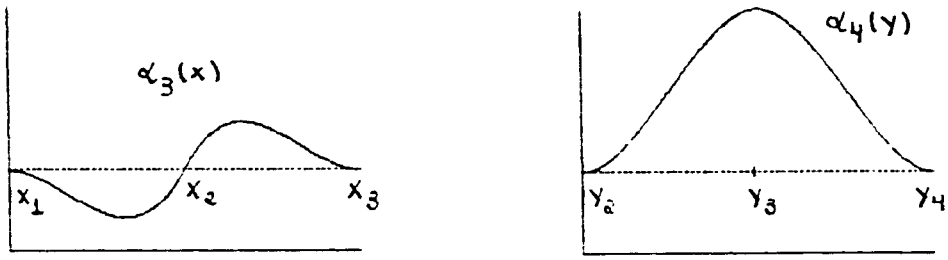


FIGURA 4.6

Determinaremos as expressões para o cálculo das integrais em II = $[x_1,x_2]\times[y_3,y_4]$, lembrando que o valor total da integral é dado pela soma nos quatro quadrantes. Em II, as funções de base locais correspondentes a $\alpha_3(x)$ e $\alpha_4(y)$ são $\phi_4(t)$ e $\phi_1(t)$, respectivamente, e as relações entre elas dadas em (4.7). Para a integral do tipo 1, (4.20), temos :

$$\iint_{II} G(x,y)\alpha_3(x)\alpha_4(y)\,dxdy = \int_{x_1}^{x_2}\int_{y_3}^{y_4} G(x,y)\alpha_3(x)\alpha_4(y)\,dxdy$$

$$= \frac{h^2}{4}\int_{-1}^{1}\int_{-1}^{1} G(x(t),y(z))\phi_4(t)\phi_1(z)\,dtdz$$

onde,
$$\begin{cases} x(t) = \frac{1}{2}\,(\,(x_1 + x_2) + h\,t) \\ y(z) = \frac{1}{2}\,(\,(y_3 + y_4) + h\,z) \end{cases} \qquad (4.27)$$

A função que faz estas mudanças de variáveis é definida no nosso programa da seguinte forma :

```
C       ************************************************
C       *  FUNCAO QUE FAZ A MUDANCA DE VARIAVEL        *
C       ************************************************

        REAL*8 FUNCTION TRANSF(T,A,B)
        REAL*8 T,A,B

        TRANSF = (A + B + (B - A)*T)*0.5D0

        RETURN
        END
```

Usando Quadratura Gaussiana com NPI pontos por elemento em cada direção, obtemos :

$$\iint_{II} G(x,y)\alpha_3(x)\alpha_4(y)\,dxdy = \qquad (4.28)$$

$$= \frac{h^2}{4}\sum_{i=1}^{NPI} Z_i \sum_{j=1}^{NPI} Z_j\, G(x(t_i),y(t_j))\,\phi_4(t_i)\phi_1(t_j)$$

onde t_i são os zeros do polinômio de Legendre e Z_i, os pesos da fórmula da Quadratura Gaussiana. Usando o mesmo procedimento, conseguimos as expressões para as outras integrais.

$$\iint_{II} \nabla G(x,y) \nabla \alpha_3(x) \alpha_4(y)\, dxdy \;=$$

(4.29)

$$= \frac{h}{2} \sum_{i=1}^{NPI} Z_i \sum_{j=1}^{NPI} Z_j \left(G_x(x(t_i),y(t_j))\, \phi_4'(t_i)\phi_1(t_j) + G_y(x(t_i),y(t_j))\, \phi_4(t_i)\phi_1'(t_j) \right)$$

$$\iint_{II} \frac{\partial^2}{\partial x \partial y} G(x,y)\; \frac{\partial^2}{\partial x \partial y} \alpha_3(x)\alpha_4(y)\, dxdy \;=$$

(4.30)

$$= \sum_{i=1}^{NPI} Z_i \sum_{j=1}^{NPI} Z_j\, G_{xy}(x(t_i),y(t_j))\, \phi_4'(t_i)\phi_1'(t_j)$$

$$\iint_{II} c(x,y) \nabla U^M\, \nabla \alpha_3(x)\alpha_4(y)\, dxdy \;=$$

(4.31)

$$= \frac{h}{2} \sum_{i=1}^{NPI} Z_i \sum_{j=1}^{NPI} Z_j\, c(x(t_i),y(t_j)) \left(U_x^M(x(t_i),y(t_j))\phi_4'(t_i)\phi_1(t_j) + U_y^M(x(t_i),y(t_j))\, \phi_4(t_i)\phi_1'(t_j) \right)$$

Observe que nas expressões para o cálculo das integrais é necessário o produto das funções de base locais e suas derivadas calculadas nos nós gaussianos. Estes valores são construídos na sub-rotina BASE e armazenados nas matrizes FI,DFIX,DFIY,D2FIXY, definidas na seção 4.3.2. Também fornecemos em BASE os pesos da fórmula da Quadratura Gaussiana.

PHI(I,II) representa a função de base local II calculada no nó gaussiano T(I).

DPHI(J,JJ) representa a derivada da função de base local JJ calculada no nó gaussiano T(J).

```
C       ************************************************
C       *  SUB-ROTINA QUE FORNECE OS DADOS PARA A      *
C       *  INTEGRACAO E CONSTROI AS BASES LOCAIS       *
C       ************************************************

        SUBROUTINE BASE
        REAL*8 T(5),Z(5),PHI(5,4),DPHI(5,4),FI(5,5,4,4)
        REAL*8 DFIX(5,5,4,4),DFIY(5,5,4,4),D2FIXY(5,5,4,4)
        INTEGER NPI,II,JJ

        COMMON / NPINTEG / NPI
        COMMON / PESO / T,Z
        COMMON / FUNCOES / FI,DFIX,DFIY,D2FIXY

C       **  FORNECE OS PONTOS E OS PESOS PARA A INTEGRACAO  **

        IF(NPI.EQ.2) THEN

        T(1) = -0.57735027D0
        T(2) = -T(1)
        Z(1) = 1.D0
        Z(2) = 1.D0

        ELSE IF(NPI.EQ.3) THEN
```

```
      T(1) = -0.7745966692414 83D0
      T(2) = 0.D0
      T(3) = -T(1)
      Z(1) = 5.D0/9.D0
      Z(2) = 8.D0/9.D0
      Z(3) = Z(1)

      ELSE IF(NPI.EQ.4) THEN

      T(1) = -0.86113631D0
      T(2) = -0.33998104D0
      T(3) = -T(2)
      T(4) = -T(1)
      Z(1) = 0.34785485D0
      Z(2) = 0.65214515D0
      Z(3) = Z(2)
      Z(4) = Z(1)

      ELSE IF(NPI.EQ.5) THEN

      T(1) = -0.90617985D0
      T(2) = -0.53846931D0
      T(3) = 0.D0
      T(4) = -T(2)
      T(5) = -T(1)
      Z(1) = 0.23692688D0
      Z(2) = 0.47862868D0
      Z(3) = 0.56888889D0
      Z(4) = Z(2)
      Z(5) = Z(1)

      ELSE

      WRITE(*,10)
10    FORMAT(//,T5,'ENTROU COM O NUMERO DE PONTOS DE GAUSS
     *  ERRADO ')
      STOP

      END IF

C     **   CALCULA AS FUNCOES DE BASE LOCAL E SUAS     **
C     **   DERIVADAS EM T(I), I=1,...,NPI              **

      DO 20 I=1,NPI

C     ** CALCULO DAS FUNCOES NO ELEMENTO PADRAO  **
```

```
      PHI(I,1) = 0.25D0*(2.D0 + T(I))*(1.D0 - T(I))*
     *  (1.D0 - T(I))
      PHI(I,2) = 0.25D0*(1.D0 + T(I))*(1.D0 - T(I))*
     *  (1.D0 - T(I))
      PHI(I,3) = 0.25D0*(2.D0 - T(I))*(1.D0 + T(I))*
     *  (1.D0 + T(I))
      PHI(I,4) = 0.25D0*(T(I) - 1.D0)*(1.D0 + T(I))*
     *  (1.D0 + T(I))

C     ** CALCULO DAS DERIVADAS **

      DPHI(I,1) = -0.75D0*(1.D0 - T(I)*T(I))
      DPHI(I,2) = -0.25D0*(1.D0 - T(I))*(1.D0 + 3.D0*T(I))
      DPHI(I,3) = 0.75D0*(1.D0 - T(I)*T(I))
      DPHI(I,4) = 0.25D0*(1.D0 + T(I))*(3.D0*T(I) - 1.D0)

20    CONTINUE

C     ** CALCULO DOS PRODUTOS **

      DO 30 II=1,4
      DO 30 JJ=1,4
      DO 30 I=1,NPI
      DO 30 J=1,NPI
      FI(I,J,II,JJ) = PHI(I,II)*PHI(J,JJ)
      DFIX(I,J,II,JJ) = DPHI(I,II)*PHI(J,JJ)
      DFIY(I,J,II,JJ) = PHI(I,II)*DPHI(J,JJ)
      D2FIXY(I,J,II,JJ) = DPHI(I,II)*DPHI(J,JJ)
30    CONTINUE

      RETURN
      END
```

As expressões dos somatórios que definem as integrais podem ser resumidos da seguinte forma :

$$C \sum_{I} Z_I \sum_{J} Z_J \, FF(I,J,II,JJ) \qquad (4.32)$$

onde, C é uma constante e

$$FF = \begin{cases} D1*FI & \text{em (4.28)} \\ D2X*DFIX + D2Y*DFIY & \text{em (4.29) e (4.31)} \\ D3*D2FIXY & \text{em (4.30)} \end{cases}$$

$$\text{com,} \quad \begin{cases} D1 = G \\ \begin{cases} D2X = G_x \\ D2Y = G_y \end{cases} & \text{em (4.29)} \\ \begin{cases} D2X = C\ U^M_x \\ D2Y = C\ U^M_y \end{cases} & \text{em (4.31)} \\ D3 = G_{xy} \end{cases}$$

O segundo somatório em (4.32) é feito na sub-rotina SOMAY, fornecendo as somas parciais, SY1,SY2,SY3 e SY4, das integrais dos tipos 1,2,3 e 4, respectivamente. As definições das funções D1,D2X,D2Y e D3 em cada iteração no tempo podem ser encontradas em (4.23), (4.24) e (4.26).

No vetor XT temos os valores $x(t_i)$, i=1,...,NPI, e em YT os valores de $y(t_j)$, j=1,...,NPI. Nas matrizes DUNX e DUNY encontramos as derivadas da solução U^M imediatamente anterior, avaliadas em (XT,YT). Veremos depois como obter estes valores.

```
C       ************************************************************
C       * SUB-ROTINA QUE CALCULA AS SOMAS EM Y DAS INTEGRAIS *
C       ************************************************************

        SUBROUTINE SOMAY(K,I,XT,YT,TP,II,JJ,DUNX,DUNY,SY1,SY2,
     *                   SY3,SY4)
        REAL*8 H,HT,XT,YT(*),TP(*),SY1,SY2,SY3,SY4,TRANSF
        REAL*8 D1,D2X,D2Y,D3,UO,DUOX,DUOY,D2UOXY,S,DSX,DSY
        REAL*8 D2SXY,VALFN,VALFC,FC,FI(5,5,4,4),DFIX(5,5,4,4)
        REAL*8 DFIY(5,5,4,4),D2FIXY(5,5,4,4)
```

```
      REAL*8 T(5),Z(5),DUNX(40,5,*),DUNY(40,5,*),
      INTEGER IT,I,NPI,II,JJ,K

      COMMON / PARTICAO / H,HT
      COMMON / NPINTEG / NPI
      COMMON / PESO / T,Z
      COMMON / FUNCOES / FI,DFIX,DFIY,D2FIXY
      COMMON / PTEMPO / IT

C     SUB-ROTINA QUE CALCULA AS SOMAS EM Y
C     DAS SEQUINTES INTEGRAIS :
C        1) <F,PHI(i)PHI(j)>
C        2) <GRAD(F),GRAD(PHI(i)PHI(j))>
C        3) <D2(F)/DXDY,D2(PHI(i)PHI(j)/DXDY>
C        4) <GRAD(UN),GRAD(PHI(i)PHI(j)>

C     II,JJ : NUMERO DA FUNCAO DE BASE
C     II,JJ = 1,2,3 ou 4
C     INTERVALO DE INTEGRACAO EM Y : [Y1,Y2]

      DO 10 J=1,NPI

      IF(IT.EQ.1) THEN
        CALL VALF(XT,YT(J),UO,DUOX,DUOY,D2UOXY)
        D1 = UO
        D2X = DUOX
        D2Y = DUOY
        D3 = D2UOXY
      ELSE IF(IT.EQ.2) THEN
        CALL VALF1(XT,YT(J),S,DSX,DSY,D2SXY)
        D1 = S
        D2X = DSX
        D2Y = DSY
        D3 = D2SXY
      ELSE
        D1 = VALFN(XT,YT(J),TP(IT-1))
        FC = VALFC(XT,YT(J))
        D2X = FC*DUNX(K,I,J)
        D2Y = FC*DUNY(K,I,J)
      END IF

      IF(IT.LT.3) THEN
        SY1 = SY1 + Z(J)*D1*FI(I,J,II,JJ)
        SY2 = SY2 + Z(J)*( D2X*DFIX(I,J,II,JJ) +
     *                     D2Y*DFIY(I,J,II,JJ) )
        SY3 = SY3 + Z(J)*D3*D2FIXY(I,J,II,JJ)
      ELSE
        SY1 = SY1 + Z(J)*D1*FI(I,J,II,JJ)
```

```
          SY4 = SY4 + Z(J)*( D2X*DFIX(I,J,II,JJ) +
     *                       D2Y*DFIY(I,J,II,JJ) )
        END IF

10      CONTINUE

        RETURN
        END
```

Na sub-rotina SOMAX fazemos a soma dos diferentes tipos de integral que aparecem em cada iteração do tempo, para a construção do lado direito do sistema.

```
C       ************************************************
C       *  SUB-ROTINA QUE FAZ A SOMA DAS INTEGRAIS     *
C       ************************************************

        SUBROUTINE SOMAX(SX1,SX2,SX3,SX4,SOMA)
        REAL*8 H,HT,SX1,SX2,SX3,SX4,SOMA
        INTEGER IT

        COMMON / PARTICAO / H,HT
        COMMON / PTEMPO / IT

        IF(IT.EQ.1) THEN
          SOMA = SX1*H*H*0.25D0 + SX2*H*0.5D0 + SX3
        ELSE IF(IT.EQ.2) THEN
          SOMA = SX1*H*H*0.25D0 + SX2*H*0.5D0*HT +
     *                SX3*HT*HT*HT*HT
        ELSE
          SOMA = (SX1*H*H*0.25D0 - SX4*H*0.5D0)*HT*HT
        END IF

        RETURN
        END
```

Nas deduções das fórmulas para os cálculos das integrais podemos notar a necessidade de relacionarmos as funções de base envolvidas, α_p, p=1,...,2N, com as funções de

base local, ϕ_i, i=1,2,3,4. Como exemplo, vamos mostrar, nas tabelas 1 e 2, estas relações para o caso das integrais que envolvem os produtos $\alpha_{2i-1}\alpha_q$ e $\alpha_{2i-2}\alpha_q$, q = 1,...,2N. Estas relações são mantidas para i=2,3,...,N alterando-se, é claro, os intervalos de integração. Quando i=1 temos somente a segunda coluna da tab.1 com o intervalo em x igual a $[x_1,x_2]$, e para i=N, a primeira coluna de tab.2 com $[x_N,x_{N+1}]$. O procedimento geral usado para o cálculo das integrais foi caminhar na rede da esquerda para a direita e em faixas da forma $[x_{i-1},x_{i+1}]x[y_1,y_{N+1}]$, i=1,...,N. Assim, o lado direito do sistema, B, será calculado por blocos e obedecerá o produto tensorial envolvido. Na sub-rotina BLOCO obtemos cada um destes blocos de B.

Nos vetores XT1 e YT1 temos os valores $x(t_i)$ e $y(t_j)$, i,j=1,...,NPI, quando estes pontos pertencem à faixa $[x_{k-1},x_k]x[y_1,y_{N+1}]$ e, XT2 e YT2 quando pertencem à faixa seguinte, $[x_k,x_{k+1}]x[y_1,y_{N+1}]$, k=2,...,N. JJ e KK indicam quais funções de base local entram no cálculo das integrais. Se JJ=0, significa que estamos calculando o primeiro bloco de B e, para KK=0, o último bloco. Em DUNX1 e DUNY1 temos os valores de U_x^M e U_y^M nos pontos (XT1,YT1) e em DUNX2 e DUNY2 os valores de U_x^M e U_y^M nos pontos (XT2,YT2).

	PRODUTOS : $\alpha_{2i-1}\ \alpha_q$		
	$[x_{i-1}, x_i]$	$[x_i, x_{i+1}]$	
$[y_1, y_2]$	$\phi_4\ \phi_2$	$\phi_2\ \phi_2$	$q = 1$
$[y_1, y_2]$	$\phi_4\ \phi_3$	$\phi_2\ \phi_3$	$q = 2$
$[y_2, y_3]$	$\phi_4\ \phi_1$	$\phi_2\ \phi_1$	
$[y_1, y_2]$	$\phi_4\ \phi_4$	$\phi_2\ \phi_4$	$q = 3$
$[y_2, y_3]$	$\phi_4\ \phi_2$	$\phi_2\ \phi_2$	
. . .			
$[y_N, y_{N+1}]$	$\phi_4\ \phi_4$	$\phi_2\ \phi_4$	$q = 2N$

TABELA 1

	PRODUTOS : $\alpha_{2i-2}\ \alpha_q$		
	$[x_{i-1}, x_i]$	$[x_i, x_{i+1}]$	
$[y_1, y_2]$	$\phi_3\ \phi_2$	$\phi_1\ \phi_2$	$q = 1$
$[y_1, y_2]$	$\phi_3\ \phi_3$	$\phi_1\ \phi_3$	$q = 2$
$[y_2, y_3]$	$\phi_3\ \phi_1$	$\phi_1\ \phi_1$	
$[y_1, y_2]$	$\phi_3\ \phi_4$	$\phi_1\ \phi_4$	$q = 3$
$[y_2, y_3]$	$\phi_3\ \phi_2$	$\phi_1\ \phi_2$	
. . .			
$[y_N, y_{N+1}]$	$\phi_3\ \phi_4$	$\phi_1\ \phi_4$	$q = 2N$

TABELA 2

```
C       ******************************************
C       *  SUB-ROTINA QUE CALCULA UM BLOCO DO    *
C       *  LADO DIREITO DO SISTEMA LINEAR        *
C       ******************************************

        SUBROUTINE BLOCO(XP,Y,TP,JJ,KK,DUNX1,DUNY1,
     *  DUNX2,DUNY2,B)
        REAL*8 XP,Y(*),TP(*),B(*),H,HT,XI,XM,XF,YI,YM,YF
        REAL*8 SX1,SX2,SX3,SX4,SX11,SX22,SX33,SX44
        REAL*8 SY1,SY2,SY3,SY4,SY11,SY22,SY33,SY44,SOMA
        REAL*8 T(5),Z(5),XT1(5),XT2(5),YT1(5),YT2(5),TRANSF
        REAL*8 DUNX1(40,5,*),DUNY1(40,5,*)
        REAL*8 DUNX2(40,5,*),DUNY2(40,5,*)
        INTEGER IT,JJ,KK,N,M,NPI,II

        COMMON / ELEMENTO / N,M
        COMMON / PARTICAO / H,HT
        COMMON / NPINTEG / NPI
        COMMON / PESO / T,Z
        COMMON / PTEMPO / IT

        XI = XP - H
        XM = XP
        XF = XP + H
        YI = Y(1)
        YM = Y(2)
        YF = Y(3)

C       SX1,SY1 = SOMAS EM X E EM Y NO CALCULO
C                 DA PRIMEIRA INTEGRAL
C       SX2,SY2 = SOMAS EM X E EM Y NO CALCULO
C                 DA SEGUNDA INTEGRAL
C       SX3,SY3 = SOMAS EM X E EM Y NO CALCULO
C                 DA TERCEIRA INTEGRAL
C       SX4,SY4 = SOMAS EM X E EM Y NO CALCULO DA
C                 QUARTA INTEGRAL
C       SUB-ROTINA SOMAY : CALCULA AS SOMAS EM Y DAS INTEGRAIS

C       **  CALCULO DE B(1)  **

        DO 10 I=1,NPI
        XT1(I) = TRANSF(T(I),XI,XM)
        XT2(I) = TRANSF(T(I),XM,XF)
        YT1(I) = TRANSF(T(I),YI,YM)
10      CONTINUE

        SX1 = 0.D0
        SX2 = 0.D0
```

```
      SX3 = 0.DO
      SX4 = 0.DO
      DO 20 I=1,NPI

      SY1 = 0.DO
      SY2 = 0.DO
      SY3 = 0.DO
      SY4 = 0.DO
      IF(JJ.NE.0) THEN
      CALL SOMAY(1,I,XT1(I),YT1,TP,JJ,2,DUNX1,DUNY1,
     *  SY1,SY2,SY3,SY4)
      END IF

      IF(KK.NE.0) THEN
      CALL SOMAY(1,I,XT2(I),YT1,TP,KK,2,DUNX2,DUNY2,
     *  SY1,SY2,SY3,SY4)
      END IF

      SX1 = SX1 + Z(I)*SY1
      SX2 = SX2 + Z(I)*SY2
      SX3 = SX3 + Z(I)*SY3
      SX4 = SX4 + Z(I)*SY4

20    CONTINUE

      CALL SOMAX(SX1,SX2,SX3,SX4,SOMA)
      B(1) = SOMA

C     SX1,SX2,SX3,SX4 = SOMAS EM X DAS INTEGRAIS, PARA
C                     O CALCULO DE B(I), I=2,N
C     SY1,SY2,SY3,SY4 = SOMAS EM Y DAS INTEGRAIS, PARA
C                     O CALCULO DE B(I), I=2,N
C     SX11,SX22,SX33,SX44 = SOMAS EM X DAS INTEGRAIS, PARA
C                       O CALCULO DE B(I+1), I=2,N
C     SY11,SY22,SY33,SY44 = SOMAS EM Y DAS INTEGRAIS, PARA
C                       O CALCULO DE B(I+1), I=2,N

C     **  CALCULO DE B(I) E B(I+1), I=2,N  **

      II = 2
      DO 30 K=2,N

      DO 40 I=1,NPI
      YT2(I) = TRANSF(T(I),YM,YF)
40    CONTINUE

      SX1 = 0.DO
      SX2 = 0.DO
```

```
      SX3 = 0.DO
      SX4 = 0.DO

      SX11 = 0.DO
      SX22 = 0.DO
      SX33 = 0.DO
      SX44 = 0.DO
      DO 50 I=1,NPI
      SY1 = 0.DO
      SY2 = 0.DO
      SY3 = 0.DO
      SY4 = 0.DO

      IF(JJ.NE.0) THEN
      CALL SOMAY(K-1,I,XT1(I),YT1,TP,JJ,3,DUNX1,DUNY1,
     *  SY1,SY2,SY3,SY4)
      CALL SOMAY(K,I,XT1(I),YT2,TP,JJ,1,DUNX1,DUNY1,
     *  SY1,SY2,SY3,SY4)
      END IF


      IF(KK.NE.0) THEN
      CALL SOMAY(K-1,I,XT2(I),YT1,TP,KK,3,DUNX2,DUNY2,
     *  SY1,SY2,SY3,SY4)
      CALL SOMAY(K,I,XT2(I),YT2,TP,KK,1,DUNX2,DUNY2,
     *  SY1,SY2,SY3,SY4)
      END IF

      SX1 = SX1 + Z(I)*SY1
      SX2 = SX2 + Z(I)*SY2
      SX3 = SX3 + Z(I)*SY3
      SX4 = SX4 + Z(I)*SY4

      SY11 = 0.DO
      SY22 = 0.DO
      SY33 = 0.DO
      SY44 = 0.DO

      IF(JJ.NE.0) THEN
      CALL SOMAY(K-1,I,XT1(I),YT1,TP,JJ,4,DUNX1,DUNY1,
     *  SY11,SY22,SY33,SY44)
      CALL SOMAY(K,I,XT1(I),YT2,TP,JJ,2,DUNX1,DUNY1,
     *  SY11,SY22,SY33,SY44)
      END IF

      IF(KK.NE.0) THEN
      CALL SOMAY(K-1,I,XT2(I),YT1,TP,KK,4,DUNX2,DUNY2,
     *  SY11,SY22,SY33,SY44)
```

```
      CALL SOMAY(K,I,XT2(I),YT2,TP,KK,2,DUNX2,DUNY2,
     *  SY11,SY22,SY33,SY44)
      END IF

      SX11 = SX11 + Z(I)*SY11
      SX22 = SX22 + Z(I)*SY22
      SX33 = SX33 + Z(I)*SY33
      SX44 = SX44 + Z(I)*SY44
50    CONTINUE

      CALL SOMAX(SX1,SX2,SX3,SX4,SOMA)
      B(II) = SOMA

      CALL SOMAX(SX11,SX22,SX33,SX44,SOMA)
      B(II+1) = SOMA

      II = II + 2
      YM = YF
      YF = YF + H

      DO 60 I=1,NPI
      YT1(I) = YT2(I)
60    CONTINUE

30    CONTINUE

C     **  CALCULO DE B(2ND)  **

      SX1 = 0.D0
      SX2 = 0.D0
      SX3 = 0.D0
      SX4 = 0.D0
      DO 70 I=1,NPI

      SY1 = 0.D0
      SY2 = 0.D0
      SY3 = 0.D0
      SY4 = 0.D0

      IF(JJ.NE.0) THEN
      CALL SOMAY(N,I,XT1(I),YT1,TP,JJ,4,DUNX1,DUNY1,
     *  SY1,SY2,SY3,SY4)
      END IF

      IF(KK.NE.0) THEN
      CALL SOMAY(N,I,XT2(I),YT1,TP,KK,4,DUNX2,DUNY2,
     *  SY1,SY2,SY3,SY4)
      END IF
```

```
      SX1 = SX1 + Z(I)*SY1
      SX2 = SX2 + Z(I)*SY2
      SX3 = SX3 + Z(I)*SY3
      SX4 = SX4 + Z(I)*SY4

70    CONTINUE

      CALL SOMAX(SX1,SX2,SX3,SX4,SOMA)
      B(M) = SOMA

      RETURN
      END
```

Para o cálculo das integrais do tipo 4, (4.31), precisamos do valor das derivadas da solução imediatamente anterior, U_x^M e U_y^M, nos pontos $x(t_i)$, onde t_i, $i=2,3$ ou 4, são os nós gaussianos. No entanto, vimos na seção 4.2 que o algoritmo fornece estas derivadas nos pontos da rede. Portanto, é preciso fazer uma interpolação para obtermos estes valores. Como exemplo, veremos como calcular $U_x^M(x(t_i),y(t_j))$ para $(x(t_i),y(t_j)) \in [x_1,x_2]\times[y_2,y_3]$. Temos abaixo os gráficos das funções de base definidas neste retângulo.

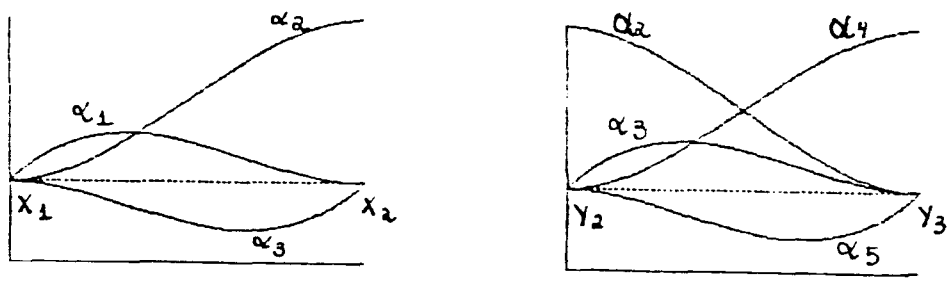


FIGURA 4.7

Usando a definição de U^M dada em (4.8), a transformação de variáveis (4.4) e as relações (4.7), temos :

$$U_x^M(x(t_i),y(t_j)) = \sum_{k=1}^{3}\sum_{L=2}^{5} \eta^M_{k,L}\,\alpha'_k(x(t_i))\,\alpha_L(y(t_j))$$

$$= \frac{2}{h}\sum_{p=2}^{4}\sum_{q=1}^{4} \eta^M_{\nu x(p),\nu y(q)}\,\phi'_p(x(t_i))\,\phi_q(y(t_j))$$

onde, $\nu_x = (0,1,2,3)^T$ e $\nu_y = (2,3,4,5)^T$. Nestes vetores guardamos a numeração global das derivadas e das funções de base.

As sub-rotinas VDUNF1, VDUNFI e VDUNFN fornecem U_x^M e U_y^M nos pontos pertencentes às faixas da rede, $[x_1,x_2]\times[y_1,y_{N+1}]$, $[x_i,x_{i+1}]\times[y_1,y_{N+1}]$ $i=2,\ldots,N$ e $[x_N,x_{N+1}]\times[y_1,y_{N+1}]$, respectivamente. Temos em DUNX1 e DUNY1 estas derivadas, U_x^M e U_y^M, calculadas na faixa $[x_{i-1},x_i]\times[y_1,y_{N+1}]$, e em DUNX2 e DUNY2 estes valores na faixa seguinte $[x_i,x_{i+1}]\times[y_1,y_{N+1}]$, $i=2,\ldots,N$. Nos vetores NGX e NGY encontramos a numeração global das derivadas e das funções de base.

```
C       ************************************************
C       *  SUB-ROTINA QUE CALCULA AS DERIVADAS DE U^M   *
C       *  NA FAIXA  [X_1,X_2]x[Y_1,Y_N+1] DO DOMINIO   *
C       *                                               *
C       ************************************************
```

```
      SUBROUTINE VDUNF1(NMAX,NGX,UN,DUNX,DUNY)
      REAL*8 H,HT,UN(NMAX,*),SOM1,SOM2
      REAL*8 FI(5,5,4,4),DFIX(5,5,4,4)
      REAL*8 DFIY(5,5,4,4),D2FIXY(5,5,4,4)
      REAL*8 DUNX(40,5,*),DUNY(40,5,*)
      INTEGER N,M,NPI,NGX(*),NGY(4),KK
      COMMON / ELEMENTO / N,M
      COMMON / PARTICAO / H,HT
      COMMON / NPINTEG / NPI
      COMMON / FUNCOES /  FI,DFIX,DFIY,D2FIXY

C     ** CALCULOS EM [X(1),X(2)]x[Y(1),Y(2)] **

      NGY(1) = 0
      DO 10 K=2,4
      NGY(K) = NGY(K-1) + 1
10    CONTINUE

      DO 20 I=1,NPI
      DO 20 J=1,NPI
      DUNX(1,I,J) = 0.D0
      DUNY(1,I,J) = 0.D0
      DO 30 K=2,4
      SOM1 = 0.D0
      SOM2 = 0.D0
      DO 40 L=2,4
      SOM1 = SOM1 + UN(NGX(K),NGY(L))*DFIX(I,J,K,L)
      SOM2 = SOM2 + UN(NGX(K),NGY(L))*DFIY(I,J,K,L)
40    CONTINUE
      DUNX(1,I,J) = DUNX(1,I,J) + SOM1
      DUNY(1,I,J) = DUNY(1,I,J) + SOM2
30    CONTINUE
      DUNX(1,I,J) = 2.D0*DUNX(1,I,J)/H
      DUNY(1,I,J) = 2.D0*DUNY(1,I,J)/H
20    CONTINUE

C     ** CALCULOS EM [X(1),X(2)]x[Y(I),Y(I+1)], I=2,N-1  **

      DO 50 KK=2,(N-1)

      DO 60 K=1,4
      NGY(K) = NGY(K) + 2
60    CONTINUE

      DO 70 I=1,NPI
      DO 70 J=1,NPI
      DUNX(KK,I,J) = 0.D0
```

```
      DUNY(KK,I,J) = 0.D0
      DO 80 K=2,4
      SOM1 = 0.D0
      SOM2 = 0.D0
      DO 90 L=1,4
      SOM1 = SOM1 + UN(NGX(K),NGY(L))*DFIX(I,J,K,L)
      SOM2 = SOM2 + UN(NGX(K),NGY(L))*DFIY(I,J,K,L)
90    CONTINUE
      DUNX(KK,I,J) = DUNX(KK,I,J) + SOM1
      DUNY(KK,I,J) = DUNY(KK,I,J) + SOM2
80    CONTINUE
      DUNX(KK,I,J) = 2.D0*DUNX(KK,I,J)/H
      DUNY(KK,I,J) = 2.D0*DUNY(KK,I,J)/H
70    CONTINUE

50    CONTINUE

C     ** CALCULOS EM [X(1),X(2)]x[Y(N),Y(N+1)] **

      NGY(1) = M - 2
      NGY(2) = NGY(1) + 1
      NGY(4) = NGY(2) + 1

      DO 100 I=1,NPI
      DO 100 J=1,NPI
      DUNX(N,I,J) = 0.D0
      DUNY(N,I,J) = 0.D0
      DO 110 K=2,4
      SOM1 = 0.D0
      SOM2 = 0.D0
      DO 120 L=1,4
      IF(L.EQ.3) GO TO 120
      SOM1 = SOM1 + UN(NGX(K),NGY(L))*DFIX(I,J,K,L)
      SOM2 = SOM2 + UN(NGX(K),NGY(L))*DFIY(I,J,K,L)
120   CONTINUE
      DUNX(N,I,J) = DUNX(N,I,J) + SOM1
      DUNY(N,I,J) = DUNY(N,I,J) + SOM2
110   CONTINUE
      DUNX(N,I,J) = 2.D0*DUNX(N,I,J)/H
      DUNY(N,I,J) = 2.D0*DUNY(N,I,J)/H
100   CONTINUE

      RETURN
      END
```

```
C     ******************************************************
C     *  SUB-ROTINA QUE CALCULA AS DERIVADAS DE U^M        *
C     *  NAS FAIXAS  [X_i , X_i+1 ]x[Y_1 , Y_N+1 ] DO DOMINIO *
C     *  i=2,...,N.                                        *
C     ******************************************************

      SUBROUTINE VDUNFI(NMAX,NGX,UN,DUNX,DUNY)
      REAL*8 H,HT,UN(NMAX,*),SOM1,SOM2
      REAL*8 FI(5,5,4,4),DFIX(5,5,4,4)
      REAL*8 D2FIX(5,4,4),D2FIXY(5,5,4,4)
      REAL*8 DUNX(40,5,*),DUNY(40,5,*)
      INTEGER N,M,NPI,NGX(*),NGY(4),KK

      COMMON / ELEMENTO / N,M
      COMMON / PARTICAO / H,HT
      COMMON / NPINTEG / NPI
      COMMON / FUNCOES /  FI,DFIX,DFIY,D2FIXY

C     SUB-ROTINA QUE CALCULA OS VALORES DAS DERIVADAS PARCIAIS DA
C     SOLUCAO APROXIMADA, UN(X,Y), NOS PONTOS DE GAUSS, PERTENCENTE
C     A FAIXA DO DOMINIO [X(I),X(I+1)]x[Y(1),Y(N+1)], I=2,N-1

C     **  CALCULOS EM [X(I),X(I+1)]x[Y(1),Y(2)] **

      NGY(1) = 0
      DO 10 K=2,4
      NGY(K) = NGY(K-1) + 1
10    CONTINUE

      DO 20 I=1,NPI
      DO 20 J=1,NPI
      DUNX(1,I,J) = 0.D0
      DUNY(1,I,J) = 0.D0
      DO 30 K=1,4
      SOM1 = 0.D0
      SOM2 = 0.D0
      DO 40 L=2,4
      SOM1 = SOM1 + UN(NGX(K),NGY(L))*DFIX(I,J,K,L)
      SOM2 = SOM2 + UN(NGX(K),NGY(L))*DFIY(I,J,K,L)
40    CONTINUE
      DUNX(1,I,J) = DUNX(1,I,J) + SOM1
      DUNY(1,I,J) = DUNY(1,I,J) + SOM2
30    CONTINUE
      DUNX(1,I,J) = 2.D0*DUNX(1,I,J)/H
      DUNY(1,I,J) = 2.D0*DUNY(1,I,J)/H
20    CONTINUE

C     ** CALCULOS EM [X(I),X(I+1)]x[Y(J),Y(J+1)], J=2,N-1 **
```

```
      DO 50 KK=2,(N-1)

      DO 60 K=1,4
      NGY(K) = NGY(K) + 2
60    CONTINUE

      DO 70 I=1,NPI
      DO 70 J=1,NPI
      DUNX(KK,I,J) = 0.DO
      DUNY(KK,I,J) = 0.DO
      DO 80 K=1,4
      SOM1 = 0.DO
      SOM2 = 0.DO
      DO 90 L=1,4
      SOM1 = SOM1 + UN(NGX(K),NGY(L))*DFIX(I,J,K,L)
      SOM2 = SOM2 + UN(NGX(K),NGY(L))*DFIY(I,J,K,L)
90    CONTINUE
      DUNX(KK,I,J) = DUNX(KK,I,J) + SOM1
      DUNY(KK,I,J) = DUNY(KK,I,J) + SOM2
80    CONTINUE
      DUNX(KK,I,J) = 2.DO*DUNX(KK,I,J)/H
      DUNY(KK,I,J) = 2.DO*DUNY(KK,I,J)/H
70    CONTINUE

50    CONTINUE

C     ** CALCULOS EM [X(I),X(I+1)]x[Y(N),Y(N+1)] **

      NGY(1) = M - 2
      NGY(2) = NGY(1) + 1
      NGY(4) = NGY(2) + 1

      DO 100 I=1,NPI
      DO 100 J=1,NPI
      DUNX(N,I,J) = 0.DO
      DUNY(N,I,J) = 0.DO
      DO 110 K=1,4
      SOM1 = 0.DO
      SOM2 = 0.DO
      DO 120 L=1,4
      IF(L.EQ.3) GO TO 120
      SOM1 = SOM1 + UN(NGX(K),NGY(L))*DFIX(I,J,K,L)
      SOM2 = SOM2 + UN(NGX(K),NGY(L))*DFIY(I,J,K,L)
120   CONTINUE
      DUNX(N,I,J) = DUNX(N,I,J) + SOM1
      DUNY(N,I,J) = DUNY(N,I,J) + SOM2
110   CONTINUE
```

```
      DUNX(N,I,J) = 2.D0*DUNX(N,I,J)/H
      DUNY(N,I,J) = 2.D0*DUNY(N,I,J)/H
100   CONTINUE

      RETURN
      END


C     ***************************************************
C     *  SUB-ROTINA QUE CALCULA AS DERIVADAS DE U^M     *
C     *  NA FAIXA  [X_N,X_N+1]x[Y_1,Y_N+1] DO DOMINIO   *
C     *                                                 *
C     ***************************************************

      SUBROUTINE VDUNFN(NMAX,NGX,UN,DUNX,DUNY)
      REAL*8 H,HT,UN(NMAX,*),SOM1,SOM2
      REAL*8 FI(5,5,4,4),DFIX(5,5,4,4)
      REAL*8 DFIY(5,5,4,4),D2FIXY(5,5,4,4)
      REAL*8 DUNX(40,5,*),DUNY(40,5,*)
      INTEGER N,M,NPI,NGX(*),NGY(4),KK

      COMMON / ELEMENTO / N,M
      COMMON / PARTICAO / H,HT
      COMMON / NPINTEG / NPI
      COMMON / FUNCOES /  FI,DFIX,DFIY,D2FIXY

C     **  CALCULOS EM [X(N),X(N+1)]x[Y(1),Y(2)] **

      NGY(1) = 0
      DO 10 K=2,4
      NGY(K) = NGY(K-1) + 1
10    CONTINUE

      DO 20 I=1,NPI
      DO 20 J=1,NPI
      DUNX(1,I,J) = 0.D0
      DUNY(1,I,J) = 0.D0
      DO 30 K=1,4
      IF(K.EQ.3) GO TO 30
      SOM1 = 0.D0
      SOM2 = 0.D0
      DO 40 L=2,4
      SOM1 = SOM1 + UN(NGX(K),NGY(L))*DFIX(I,J,K,L)
      SOM2 = SOM2 + UN(NGX(K),NGY(L))*DFIY(I,J,K,L)
40    CONTINUE
      DUNX(1,I,J) = DUNX(1,I,J) + SOM1
      DUNY(1,I,J) = DUNY(1,I,J) + SOM2
30    CONTINUE
```

```
      DUNX(1,I,J) = 2.D0*DUNX(1,I,J)/H
      DUNY(1,I,J) = 2.D0*DUNY(1,I,J)/H
20    CONTINUE

C     ** CALCULOS EM [X(N),X(N+1)]x[Y(J),Y(J+1)], J=2,N-1 **

      DO 50 KK=2,(N-1)

      DO 60 K=1,4
      NGY(K) = NGY(K) + 2
60    CONTINUE

      DO 70 I=1,NPI
      DO 70 J=1,NPI
      DUNX(KK,I,J) = 0.D0
      DUNY(KK,I,J) = 0.D0
      DO 80 K=1,4
      IF(K.EQ.3) GO TO 80
      SOM1 = 0.D0
      SOM2 = 0.D0
      DO 90 L=1,4
      SOM1 = SOM1 + UN(NGX(K),NGY(L))*DFIX(I,J,K,L)
      SOM2 = SOM2 + UN(NGX(K),NGY(L))*DFIY(I,J,K,L)
90    CONTINUE
      DUNX(KK,I,J) = DUNX(KK,I,J) + SOM1
      DUNY(KK,I,J) = DUNY(KK,I,J) + SOM2
80    CONTINUE
      DUNX(KK,I,J) = 2.D0*DUNX(KK,I,J)/H
      DUNY(KK,I,J) = 2.D0*DUNY(KK,I,J)/H
70    CONTINUE

50    CONTINUE

C     ** CALCULOS EM [X(N),X(N+1)]x[Y(N),Y(N+1)] **

      NGY(1) = M - 2
      NGY(2) = NGY(1) + 1
      NGY(4) = NGY(2) + 1

      DO 100 I=1,NPI
      DO 100 J=1,NPI
      DUNX(N,I,J) = 0.D0
      DUNY(N,I,J) = 0.D0
      DO 110 K=1,4
      IF(K.EQ.3) GO TO 110
      SOM1 = 0.D0
      SOM2 = 0.D0
      DO 120 L=1,4
```

```
      IF(L.EQ.3) GO TO 120
      SOM1 = SOM1 + UN(NGX(K),NGY(L))*DFIX(I,J,K,L)
      SOM2 = SOM2 + UN(NGX(K),NGY(L))*DFIY(I,J,K,L)
120   CONTINUE
      DUNX(N,I,J) = DUNX(N,I,J) + SOM1
      DUNY(N,I,J) = DUNY(N,I,J) + SOM2
110   CONTINUE
      DUNX(N,I,J) = 2.D0*DUNX(N,I,J)/H
      DUNY(N,I,J) = 2.D0*DUNY(N,I,J)/H
100   CONTINUE

      RETURN
      END
```

Podemos agora apresentar a sub-rotina INTEGRAL que, chamando todas as outras sub-rotinas descritas neste capítulo, efetiva o cálculo do lado direito do sistema, B, em cada iteração do tempo.

```
C     ****************************************
C     *  SUB-ROTINA QUE CALCULA O LADO       *
C     *  DIREITO DO SISTEMA, B               *
C     ****************************************

      SUBROUTINE INTEGRAL(NMAX,X,Y,TP,UN,B)
      REAL*8 X(*),Y(*),TP(*),UN(NMAX,*),B(NMAX,*)
      REAL*8 DUNX1(40,5,5),DUNY1(40,5,5)
      REAL*8 DUNX2(40,5,5),DUNY2(40,5,5),BB(80)
      INTEGER N,M,IT,NPI,NGX(4),II,KK

      COMMON / ELEMENTO / N,M
      COMMON / NPINTEG / NPI
      COMMON / PTEMPO / IT

C     A SUB-ROTINA CALCULA AS SEQUINTES INTEGRAIS PARA
C     CONSTRUIR O LADO DIREITO DO SISTEMA :
C       1) <F,PHI(i)PHI(j)>
C       2) <GRAD(F),GRAD(PHI(i)PHI(j))>
C       3) <D2(F)/DXDY,D2(PHI(i)PHI(j)/DXDY>
C       4) <GRAD(UN),GRAD(PHI(i)PHI(j))>

C     ** CALCULO DE B(1,J), J=1,M **
```

```
      IF(IT.GT.2) THEN
      NGX(1) = 0
      DO 10 I=2,4
      NGX(I) = NGX(I-1) + 1
10    CONTINUE
      CALL VDUNF1(NMAX,NGX,UN,DUNX1,DUNY1)
      END IF
      CALL BLOCOC(X(2),Y,TP,2,0,DUNX1,DUNY1,DUNX2,DUNY2,BB)
      DO 20 I=1,M
      B(1,I) = BB(I)
20    CONTINUE

C     ** CALCULO DE B(I,J) E B(I+1,J), I=2,N E J=1,M **

      II = 2
      DO 30 I=2,N

      IF(IT.GT.2) THEN
      IF(I.LT.N) THEN
      DO 40 J=1,4
      NGX(J) = NGX(J) + 2
40    CONTINUE
      CALL VDUNFI(NMAX,NGX,UN,DUNX2,DUNY2)
      ELSE
      NGX(1) = M - 2
      NGX(2) = M - 1
      NGX(4) = M
      CALL VDUNFN(NMAX,NGX,UN,DUNX2,DUNY2)
      END IF
      END IF

      CALL BLOCOC(X(I),Y,TP,3,1,DUNX1,DUNY1,DUNX2,DUNY2,BB)
      DO 50 J=1,M
      B(II,J) = BB(J)
50    CONTINUE

      CALL BLOCOC(X(I),Y,TP,4,2,DUNX1,DUNY1,DUNX2,DUNY2,BB)
      DO 60 J=1,M
      B(II+1,J) = BB(J)
60    CONTINUE

      IF(IT.GT.2) THEN
      DO 70 KK=1,N
      DO 70 K=1,NPI
      DO 70 J=1,NPI
      DUNX1(KK,K,J) = DUNX2(KK,K,J)
      DUNY1(KK,K,J) = DUNY2(KK,K,J)
```

```
70      CONTINUE
        END IF

        II = II + 2

30      CONTINUE

C       ** CALCULO DE B(2N,J), J=1,M **
        CALL BLOCO(X(N),Y,TP,0,4,DUNX1,DUNY1,DUNX2,DUNY2,BB)
        DO 80 I=1,M
        B(M,I) = BB(I)
80      CONTINUE

        RETURN
        END
```

4.3.9 - FUNCÕES

Mostraremos agora como definimos no programa as condições iniciais, a fonte e as outras funções utilizadas nos cálculos das integrais. Como exemplo, apresentaremos as funções para a equação de propagação de ondas (2.4), cuja fonte é dada pela expressão,

$$f(x,y,t) = 100\exp(-50((x-xo)^2+(y-yo)^2) - 40t^2)$$

e as condições iniciais, por :

$$u_o(x,y,t) = 0$$

$$u_1(x,y,t) = 0$$

Resolvemos numericamente este problema e os resultados obtidos encontram-se no próximo capítulo.

```
C       ************************************************
C       *  SUB-ROTINA QUE SAI COM O VALORES INICIAIS   *
C       *  DA SOLUCAO E DE SUAS DERIVADAS              *
C       ************************************************

        SUBROUTINE VALF(X,Y,UO,DUOX,DUOY,D2UOXY)
        REAL*8 X,Y,UO,DUOX,DUOY,D2UOXY

        UO =  0.DO

        DUOX = 0.DO

        DUOY = 0.DO

        D2UOXY = 0.DO

        RETURN
        END


C       ********************************************
C       *  SUB-ROTNA QUE SAI COM OS VALORES DA     *
C       *  FUNCAO S E DE SUAS DERIVADAS            *
C       ********************************************

        SUBROUTINE VALF1(X,Y,S,DSX,DSY,D2SXY)
        REAL*8 H,HT,X,Y,FO,DFOX,DFOY,D2FOXY,AUX
        REAL*8 S,DSX,DSY,D2SXY,XO,YO,AUX1

        COMMON / PARTICAO / H,HT
        COMMON / PCENTR / XO,YO

C       FONTE NO TEMPO ZERO = FO
C       DFOX = DERIVADA DE FO EM RELACAO A X
C       DFOY = DERIVADA DE FO EM RELACAO A Y
C       D2FOXY = DERIVADA DE FO EM RELACAO A X E Y

        AUX1 = 5.D1*( (X-XO)*(X-XO) + (Y-YO)*(Y-YO) )
        IF(AUX1.LT.3.D1) THEN

        FO = 1.D2*DEXP(-AUX1)
        DFOX = -1.D4*(X-XO)*DEXP(-AUX1)
        DFOY = -1.D4*(Y-YO)*DEXP(-AUX1)
        D2FOXY = 1.D6*(X-XO)*(Y-YO)*DEXP(-AUX1)

        ELSE

        FO = 0.DO
```

```
      DFOX = 0.D0
      DFOY = 0.D0
      D2FOXY = 0.D0

      END IF

      AUX = 0.5D0*HT*HT

      S = AUX*FO
      DSX = AUX*DFOX
      DSY = AUX*DFOY
      D2SXY = AUX*D2FOXY

      RETURN
      END


C     ***************************
C     *  VELOCIDADE DO MEIO  *
C     ***************************

      REAL*8 FUNCTION VALFC(X,Y)
      REAL*8 X,Y

      VALFC = 1.D0

      RETURN
      END


C     ************************************************
C     *  FONTE DO PROBLEMA EM TODOS OS TEMPOS  *
C     ************************************************

      REAL*8 FUNCTION VALFN(X,Y,TP)
      REAL*8 X,Y,TP,XO,YO,AUX1,AUX2,ESPACO,TEMPO

      COMMON / PCENTR / XO,YO

      AUX1 = 5.D1*( (X-XO)*(X-XO) + (Y-YO)*(Y-YO) )
      IF(AUX1.LT.3.D1) THEN
      ESPACO = 1.D2*DEXP(-AUX1)
      ELSE
      ESPACO = 0.D0
      END IF

      AUX2 = 4.D1*TP*TP
      IF(AUX2.LT.3.D1) THEN
```

```
TEMPO = DEXP(-AUX2)
ELSE
TEMPO = 0.DO
END IF

VALFN = ESPACO*TEMPO

RETURN
END
```

4.3.10 - SAÍDA DOS RESULTADOS E DO ERRO

A sub-rotina SAÍDA fornece as seguintes opções na saída dos resultados : avaliação da medida do erro no caso de se conhecer a solução exata, a construção de arquivos que contém as soluções aproximadas para saídas gráficas ou, simplesmente, a apresentação das soluções em determinadas iterações no tempo. Na avaliação do erro usamos a norma do SUP,

$$|| u - \tilde{u} ||_{\infty} = \max_{1 \le i, j \le N} | u(x_i, y_j, 0) - \tilde{u}(x_i, y_j) |$$

e a norma L^2, definida na seção 2.3.2 e cuja integração foi resolvida pelo método de Simpson [05]. Caso o problema tenha solução exata, devemos colocar em SOL a definição desta solução. Na sub-rotina GRAFICO é feita a vetorização da solução aproximada, para ser utilizada em GRAF3D na construção do arquivo para saída gráfica. Usamos o aplicativo

ENERTRONICS GRAFIT para obtermos os gráficos.

```
C       ************************************************************
C       *  SUB-ROTINA QUE SAI COM OS RESULTADOS E O ERRO           *
C       ************************************************************

        SUBROUTINE SAIDA(NMAX,X,Y,TP,NITER,ISE,U,UN)
        REAL*8 H,HT,X(*),Y(*),TP(*),U(*),UN(NMAX,*),ERRO,SOL
        REAL*8 L2NOR,SUPNOR,XO,YO
        INTEGER N,M,NN,NITER,ISE,NTNOS,KI,KJ
        CHARACTER NOME*15,RES*1

        COMMON / ELEMENTO / N,M
        COMMON / PARTICAO / H,HT
        COMMON / PTEMPO / IT
        COMMON / PCENTR / XO,YO

        WRITE(*,10) IT
10      FORMAT(//,T15,'ITERACAO = ',I2)

C       * ANALISA O ERRO SE POSSUIR SOLUCAO EXATA **

        IF(ISE.EQ.1) THEN

50      WRITE(*,20)
20      FORMAT(/,T5,'DESEJA A SOLUCAO E O ERRO (S/N) ?',$)
        READ(*,30) RES
30      FORMAT(A1)
        IF(RES.EQ.'N') GO TO 40
        IF(RES.NE.'S') GO TO 50

        WRITE(*,60)
60      FORMAT(/,T5,'ENTRE COM O NOME DO ARQUIVO :  ',$)
        READ(*,70) NOME
70      FORMAT(A15)

        OPEN(UNIT=80,FILE=NOME,STATUS='NEW')

        WRITE(80,90)
90      FORMAT(/,T4,'X',T15,'Y',T28,'SOL. EXATA',T42,
     *  'SOL.APROX.',T56,'ERRO',//)

        SUPNOR = 0.D0
        L2NOR = 0.D0
        DO 100 I=2,N
        KI = 2*(I-1)
```

```
      DO 100 J=2,N
      KJ = 2*(J-1)

      SOL =

      ERRO = DABS(SOL - UN(KI,KJ))
      L2NOR = L2NOR + ERRO*ERRO
      IF(SUPNOR.LT.ERRO) SUPNOR = ERRO

      WRITE(80,110) X(I),Y(J),SOL,UN(KI,KJ),ERRO
110   FORMAT(T2,F9.5,T13,F9.5,T24,D12.5,T38,D12.5,T52,D12.5)

100   CONTINUE
      L2NOR = H*DSQRT(L2NOR)

      WRITE(*,120) SUPNOR,L2NOR
120   FORMAT(///,T15,'ERRO NA NORMA DO SUP = ',E12.5,//,T15,
     *  'ERRO NA NORMA L2 (TRAPEZIO) = ',E12.5,///)

      IF(IT.EQ.2) THEN
      WRITE(*,130) NITER
130   FORMAT(T15,'NUMERO DE ITERACOES = ',I2)
      END IF

      CLOSE(UNIT=80)

      ELSE

C     ** SAI COM A SOLUCAO **

170   WRITE(*,140)
140   FORMAT(/,T5,'DESEJA A SOLUCAO NUMERICA (S/N) ?',$)
      READ(*,150) RES
150   FORMAT(A1)
      IF(RES.EQ.'N') GO TO 160
      IF(RES.NE.'S') GO TO 170

      WRITE(*,180)
180   FORMAT(/,T5,'ENTRE COM O NOME DO ARQUIVO : ',$)
      READ(*,190) NOME
190   FORMAT(A15)

      OPEN(UNIT=200,FILE=NOME,STATUS='NEW')

      WRITE(200,210)
210   FORMAT(/,T5,'X',T20,'Y',T33,'SOL. APROX',//)

      DO 220 I=2,N
```

```
      KI = 2*(I-1)
      DO 220 J=2,N
      KJ = 2*(J-1)
      WRITE(200,230) X(I),Y(J),UN(KI,KJ)
230   FORMAT(T3,F9.5,T18,F9.5,T29,D12.5)
220   CONTINUE

      CLOSE(UNIT=200)

C     ** MONTA O ARQUIVO PARA SAIDA GRAFICA **

160   NN = N + 1
      NTNOS = NN*NN
      CALL GRAFICO(NMAX,TP,NN,NTNOS,UN,U)

      END IF

40    RETURN
      END


C     ************************************************************
C     *  SUB-ROTINA QUE FAZ A VETORIZACAO DA SOLUCAO E  *
C     *  CHAMA A SUB-ROTINA GRAF3D                      *
C     ************************************************************

      SUBROUTINE GRAFICO(NMAX,TP,NN,NTNOS,UN,U)
      REAL*8 TP(*),UN(NMAX,*),U(*)
      INTEGER N,M,NN,NTNOS,IT,KI,KJ,FREQ
      CHARACTER RES*1

      COMMON / ELEMENTO / N,M
      COMMON / PTEMPO / IT

50    WRITE(*,20)
20    FORMAT(/,T5,'DESEJA O GRAFICO (S/N) ? ',$)
      READ(*,30) RES
30    FORMAT(A1)

      IF(RES.EQ.'N') GO TO 40
      IF(RES.NE.'S') GO TO 50

      WRITE(*,60)
60    FORMAT(//,T5,'FREQUENCIA PARA SAIDA GRAFICA : ',$)
      READ(*,*) FREQ

C     ** VETORIZACAO DA SOLUCAO **
```

```
      DO 70 I=1,NTNOS
      U(I) = 0.DO
70    CONTINUE

      DO 80 I=2,N
      KI = 2*(I-1)
      DO 80 J=2,N
      KJ = 2*(J-1)
      K = J + (I-1)*NN
      U(K) = UN(KI,KJ)
80    CONTINUE

C     ** CHAMADA DA SUB-ROTINA QUE MONTA O ARQUIVO **

      CALL GRAF3D(TP,NN,FREQ,U)

40    RETURN
      END


C     ************************************************************
C     * SUB-ROTINA QUE MONTA O ARQUIVO PARA SAIDA GRAFICA        *
C     ************************************************************

      SUBROUTINE GRAF3D(TP,NN,FREQ,U)
      REAL*8 TP(*),U(*)
      INTEGER N,M,NN,FREQ,IT
      CHARACTER NOME*15

      COMMON / ELEMENTO / N,M
      COMMON / PTEMPO / IT

      WRITE(*,10)
10    FORMAT(//,T5,'ARQUIVO PARA SAIDA GRAFICA : ',$)
      READ(*,20) NOME
20    FORMAT(A15)

      OPEN(UNIT=30,FILE=NOME,STATUS='NEW')

      WRITE(30,*) 'IT = ',IT,'******','TP(IT) = ',TP(IT)
      WRITE(30,*) '      '
      WRITE(30,*) '      '
      WRITE(30,*) '      '
      WRITE(30,*) '      '
      WRITE(30,*) '33,33,-1,1,-1,1'
      WRITE(30,*) '2,2,1,200,70'
      WRITE(30,*) '45,45,3,0'
```

```
      DO 40 I=1,NN,FREQ
      DO 40 J=1,NN,FREQ
      K = J + (I-1)*NN
      WRITE(30,50) U(K)
50    FORMAT(D12.5)
40    CONTINUE

      CLOSE(UNIT=30)

      RETURN
      END
```

5 - RESULTADOS E APLICACÕES

INTRODUÇÃO :

Os primeiros testes que fizemos procuram avaliar a eficiência do nosso programa na obtenção das soluções aproximadas nos tempos iniciais, $t=0$ e $t=\Delta t$, e os cálculos nas demais iterações no tempo. Para isto, construímos exemplos com soluções analíticas conhecidas, sem o cuidado de simular situações físicas reais. A seguir, procuramos um exemplo que tivesse um sentido físico. Tomamos uma fonte que se aproxima de uma função delta de Dirac [11] localizada no centro do domínio e mostramos, através de gráficos, o comportamento da solução no decorrer do tempo.

No final deste capitulo apresentamos uma sugestão de como resolver o problema em simulação numérica de ondas propagando na terra, ou seja, quando o domínio de interesse em (2.4) é infinito, $-\infty < x < \infty$, $y \geq 0$ e $t \geq 0$. Mostraremos como foi possível usar, com algumas modificações, o nosso programa e os resultados que obtivemos.

TESTE 1 :

O primeiro teste objetiva analisar o comportamento da solução aproximada no tempo t=0, ou seja, testar a parte do programa que resolve o algoritmo descrito em (3.25). Mostraremos também, como exemplo, o comportamento da derivada em relação a x da solução aproximada. Tomamos

$$u(x,y,t) = 10x(x-1)y(y-1)$$

como solução exata de (2.4) em $[0,1]\times[0,1]$ e obtemos os seguintes resultados:

N	NPI	
	2	3
8	5.1127 E-06	1.0002 E-16
16	3.3700 E-07	2.8853 E-16
32	2.1500 E-08	9.4652 E-16

TABELA 1 - Avaliação da solução aproximada
(Erro na norma L^2)

onde, NPI = Número de pontos de integração em cada direção

N = Número de elementos em cada direção

N	NPI	
	2	3
8	6.2160 E-05	2.0359 E-16
16	7.6210 E-06	1.1786 E-16
32	9.2882 E-07	9.0401 E-17

TABELA 2 - Avaliação da derivada
(Erro na norma L^2)

Como neste exemplo a solução pertence ao espaço de aproximação, o erro é devido a integração numérica. Quando usamos três pontos de integração na fórmula da Quadratura Gaussiana, temos a integral exata para polinômios de grau cinco. Observe que precisamos resolver neste problema integrais de polinômios de graus no máximo cinco em cada direção, portanto, tomando NPI = 3 encontramos somente erros de arredondamento, como confirma a tabela 1.

TESTE 2 :

Agora estamos interessados em verificar a ordem de aproximação que obtemos neste procedimento que calcula a

solução no tempo t=0. Teoricamente, como estamos usando Quadrados Mínimos e funções de Hermite Cúbico, a ordem de aproximação deve ser igual a h^4 [15].

Para isto, precisamos determinar o valor de α na relação

$$|| u - \tilde{u} || \leq K h^{\alpha},$$

onde u é a solução exata e $\tilde{u}$ a solução aproximada. Para isto, considere

$$E_1 = || u - \tilde{u}_1 || \cong K h_1^{\alpha}$$

$$E_2 = || u - \tilde{u}_2 || \cong K h_2^{\alpha}$$

com $h_2 = h_1/2$, $\tilde{u}_1$ e $\tilde{u}_2$ duas soluções aproximadas. Assim, dividindo E_1 por E_2 e aplicando o logaritmo, obtemos :

$$\alpha \cong \ln (E_1/E_2) / \ln 2 \qquad (5.1)$$

Tomando como solução exata da equacão (2.4) a função

$$u(x,y,t) = \text{sen}(\Pi x)\cos(\Pi(y+0.5))$$

no domínio $\Omega=[0,1]\times[0,1]$, construímos a tabela 3. Usando estes resultados na equação (5.1) calculamos os valores de α que, como podemos observar na tabela 4, confirmam a ordem de aproximação teórica.

N	NPI	
	2	3
8	1.0036 E-04	3.1437 E-05
16	6.4452 E-06	2.0650 E-06
32	3.8155 E-07	1.5694 E-07

TABELA 3 - Erro na norma L^2

N	NPI	
	2	3
8 / 16	3.96	3.93
16 / 32	4.08	3.72

TABELA 4 - Valores de α

TESTE 3 :

Este experimento verifica os cálculos feitos no processo iterativo que determina a solução aproximada no tempo $t=\Delta t$, descrito em (3.26). Tomemos a fonte de modo que

$$u(x,y,t) = 100x(x-1)y(y-1)t^2,$$

pertencente ao espaço de aproximação, seja solução exata da equação em $[0,1]\times[0,1]$. Como usamos no esquema geral diferenças finitas centrais na discretização do tempo e a solução exata é um polinômio em t^2, o erro obtido neste exemplo deve ser da ordem da precisão desejada. Os resultados apresentados na tabela 5 foram obtidos para um número fixo de pontos em cada direção, N=16, e para NPI = 3. Os critérios de parada que usamos neste processo iterativo são dados por

ITERMAX = Número máximo de iterações

EPS = Precisão desejada

e, definimos ITER como o número de iterações obtidas.

	NT			
EPS=1.D-07 ITERMAX=20	16	32	64	128
ERRO	1.1968E-07	5.1059E-08	1.2541E-08	2.9248E-09
ITER	19	8	4	2

TABELA 5

TESTE 4 :

Agora estamos interessados em investigar a ordem de aproximação que obtemos neste processo iterativo que calcula a

solução aproximada no tempo $t=\Delta t$. Para isto, tomamos a função

$$u(x,y,t) = x(x-1)y(y-1)(1-\exp(-t^2))$$

como solução exata da equação em $[0,1]\times[0,1]$. Fixando o número de pontos em cada direção , N = 16 , e variando os intervalos em t obtemos os resultados dados na tabela 6. Com estes valores construimos, como anteriormente, a tabela que fornece β em

$$|| u - \tilde{u} || \leq K(\Delta t)^{\beta}.$$

Experimentalmente, o erro de aproximação obtido neste processo iterativo para o cálculo da solução em $t=\Delta t$ é da ordem de $(\Delta t)^4$, como mostra a tabela 7.

	NT			
EPS=1.D-07 ITERMAX=60 NPI=9	16	32	64	128
ERRO	2.5398E-04	1.5889E-05	9.9350E-07	6.2922E-08
ITER	55	20	8	3

TABELA 6

	N		
EPS=1.D-07 ITERMAX=60 NPI = 9	16/32	32/64	64/128
	4.00	4.00	3.98

TABELA 7 - Valores de β

TESTE 5 :

Visa verificar o procedimento (3.27) que calcula as soluções aproximadas nos demais tempos, $t=i\Delta t$ $i \geq 3$. Tomamos como solução analítica conhecida da equação (3.1), a função

$$u(x,y,t) = 100x(x-1)y(y-1)t^2$$

e iniciamos o processo com as soluções exatas. Usando um número pequeno de pontos em cada direção, obtivemos os erros na ordem de precisão da máquina, como mostra a tabela 8, significando que os cálculos estão sendo feitos corretamente. Neste exemplo tivemos as seguintes entradas :

$\Omega = [0,1]\times[0,1]$ $T = [0,1]$

N = 8 (Número de intervalos em cada direção)

NT = 16 (Número de intervalos no tempo)

NPI = 3 (Número de pontos de integração em cada direção)

$\lambda = 0.26$ (parâmetro de estabilidade)

TEMPO	ERRO NA NORMA L^2
0.25 seg.	4.0654 E-18
0.50 seg.	1.0579 E-17
0.75 seg.	3.2361 E-17
1.00 seg.	9.6398 E-17

TABELA 8

TESTE 6 :

Objetiva testar o programa para o problema cuja solução exata é

$$u(x,y,t) = \text{sen}(\pi x)\text{sen}(\pi y)\cos(\sqrt{2}\,\pi t)$$

no domínio $\Omega = [0,1]\times[0,1]$. Na referência [10], temos este exemplo para o caso de usarmos elementos lineares em cada direção, ou seja, $\mathcal{M}_{hx} = \mathcal{M}_{hy} = H^{(1)}(\Pi)$. Tomando os dados de entrada usados em [10] e reduzindo o número de elementos por direção de N = 40 para N = 8, conseguimos resultados com a mesma ordem de aproximação.

Entradas :

T = [0,1]

N = 8 (Número de intervalos em cada direção)

NT = 100 (Número de intervalos no tempo)

NPI = 3 (Número de pontos de integração em cada direção)

λ = 10 (Parâmetro de estabilidade)

Resultados :

TEMPO	ERRO NA NORMA L^2
0.00 seg	3.1410 E-05
0.10 seg	6.9586 E-04
0.20 seg	2.9058 E-03
0.30 seg	5.7866 E-03
0.40 seg	8.0785 E-03
0.50 seg	8.5345 E-03
0.60 seg	6.3150 E-03
0.70 seg	1.2937 E-03
0.80 seg	5.8112 E-03
0.90 seg	1.3528 E-02
1.00 seg	2.0428 E-02

TABELA 9

TESTE 7 :

Mostraremos um exemplo que procura simular uma situação física real. Tomamos uma fonte que se aproxima de um delta de Dirac,

$$f(x,y,t) = 100 \exp(-50(x^2 + y^2) - 40t^2),$$

localizada no centro do domínio, $[-3,3]\times[-3,3]$, e as seguintes condições iniciais :

$$u_0(x,y,0) = 0$$

$$u_1(x,y,0) = 0.$$

Lembre-se que estamos resolvendo o problema com condição de Dirichlet nula. Com as entradas dadas abaixo contruímos os gráficos que aparecem a seguir.

$T = [0,6]$

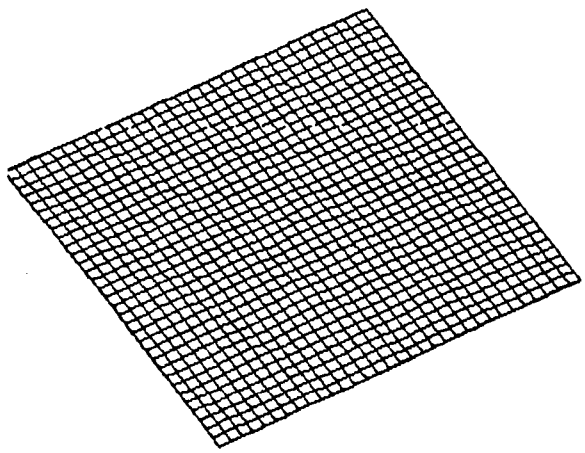
$N = 32$

$NT = 64$

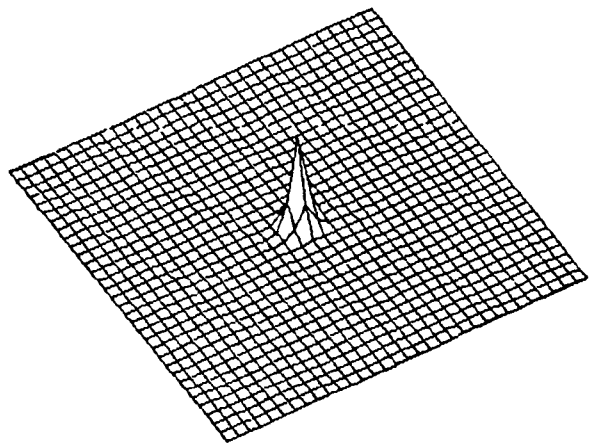
$NPI = 2$

$C = 2$

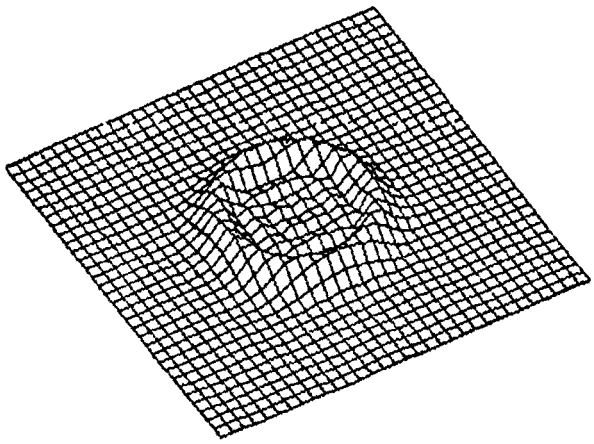
$\lambda = 0.6$



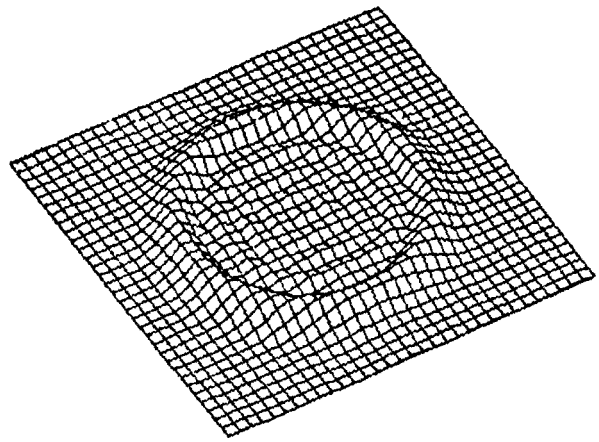
T=0.000000



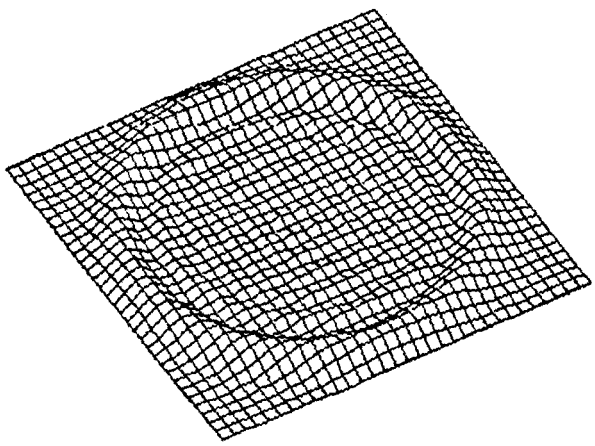
T=0.18750



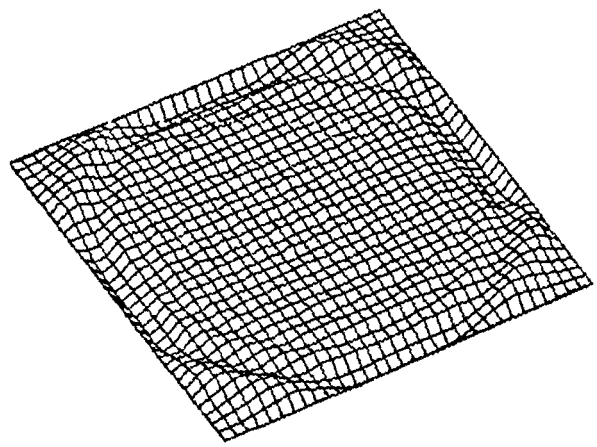
T=1. 40625



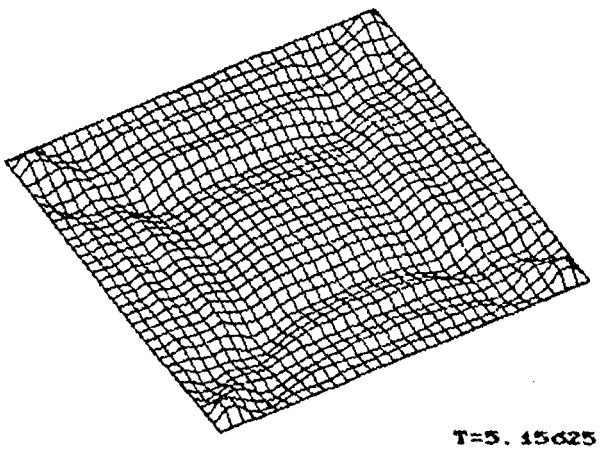
T=2. 15625



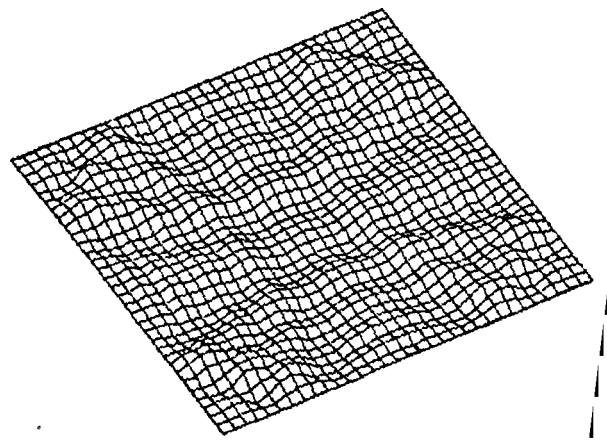
T=2. 90625



T=8. 65625



T=5. 15625



T=6. 00000

APLICAÇÃO :

Estamos interessados agora no problema de determinar uma função $u=u(x,y,t)$ definida em $-\infty < x < +\infty$, $y > 0$ e $t \geq 0$, que satisfaça as seguintes condições :

$$(5.2)\quad \begin{cases} \dfrac{\partial^2 u}{\partial t^2} - c\,\nabla^2.u = f & -\infty < x < \infty\,,\quad y > 0 \\ & t \geq 0 \\ B\,u(x,0,t) = p(x,y) & t \geq 0 \\ u(x,y,0) = u_0(x,y)\,,\quad \dfrac{\partial u}{\partial t}(x,y,0) = u_1(x,y) & -\infty < x < \infty \\ & y > 0 \end{cases}$$

onde B é a condição de superfície e $p(x,y)$ é a fonte de superfície.

A primeira consideração que deve ser feita nos problemas em simulação numérica de ondas propagando na terra é que, como o computador tem capacidade de memória finita, o modelo estudado tem que ser reduzido em tamanho. Portanto, a nova região de definição das variáveis espaciais é da seguinte forma :

$$\begin{cases} -M \leq x \leq M \\ 0 \leq y \leq L \end{cases}$$

como mostra a figura 5.1. Consequentemente, condições de fronteira devem ser impostas nas bordas desta nova região para

produzir soluções únicas. Condições do tipo Dirichlet nula causariam reflexões indesejadas na fronteira [17]. Portanto, muitas das condições de fronteira geralmente usadas em soluções numéricas de ondas são aproximações de condições de fronteira perfeitamente absorventes e dependem do ângulo de incidência da onda de chegada.

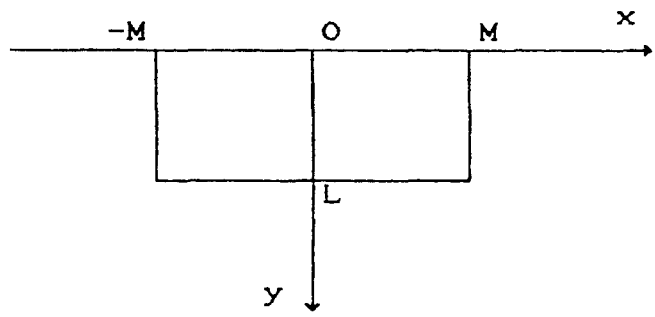


Figura 5.1

Clayton e Engquist [03], Engquist e Majda [07] e [08], Reynolds [17] e Smith [18] encontraram aproximações para as condições de fronteira absorventes que trabalham muito bem para ondas que batem na fronteira artificial com pequeno ângulo de incidência. Outra forma de reduzir as reflexões indesejadas é considerar a região limitada muito grande, de forma a retardar as reflexões [01] e [13]. O método descrito em [19] e que usaremos neste trabalho sugere a adição de um choque absorvente na equação da onda na região vizinha ao modelo, de modo que condições do tipo Dirichlet podem ser usadas no lugar de fronteiras absorventes.

Introduziremos no modelo o termo de amortecimento $2A(x,y)u_t$, onde

$$A(x,y) = 0 \quad \text{para} \quad -M < x < M \ \text{e} \ 0 < y < L$$

$A(x,y)$ é uma função positiva na região vizinha, como mostra a figura 5.2. A equação agora para o modelo se torna :

$$(5.3) \quad \begin{cases} u_{tt} + 2A(x,y)u_t = c\,\nabla^2.u + f \\ u(-M-m,y,t) = u(M+m,y,t) = u(x,L+\ell,t) = 0 \\ B\,u(x,0,t) = p(x,t) \\ u(x,y,0) = u_0(x,y) \\ u_t(x,y,0) = u_1(x,y) \end{cases}$$

Em [19] encontramos as propriedades que a função $A(x,y)$ deve possuir para que as ondas refletidas tenham decaimento suficientemente próximo de zero nas fronteiras assim como algumas definições para esta função de amortecimento.

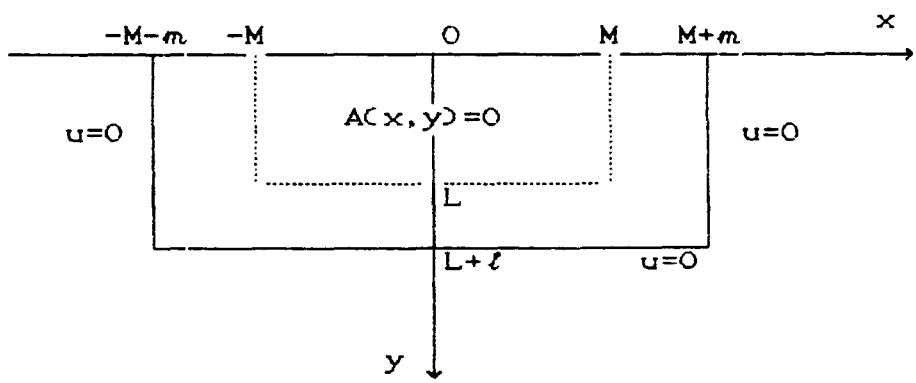


Figura 5.2

Como o termo de amortecimento é parte da equação, uma variedade de técnicas numéricas pode se usada para resolver este modelo. No nosso trabalho vamos utilizar o método de Galerkin com Direções Alternadas aproveitando com algumas modifições o programa desenvolvido no capítulo 4.

Seguindo os procedimentos descritos na seção 3.2 e a notação utilizada, podemos obter o seguinte problema aproximado para a equação (5.3) :

$$(C^x + \lambda(\Delta t)^2 A^x) \otimes (C^y + \lambda(\Delta t)^2 A^y) Z = (\Delta t)^2 \Phi^M$$

onde,

$$Z = \eta^{M+1} - 2\eta^M + \eta^{M-1}, \quad M \geq 1$$

$$(\Phi^M)_{KL} = - < c\nabla U^M , \nabla \alpha_K \beta_L > + < f^M, \alpha_K \beta_L > -$$

$$- < 2A (\frac{U^M - U^{M-1}}{\Delta t}), \alpha_K \beta_L >$$

Como aproximação para a derivada primeira de U em relação a t, usamos diferenças finitas :

$$\frac{dU}{dt} = \frac{U^M - U^{M-1}}{\Delta t}$$

Desta forma, o termo de amortecimento que acrescentamos à equação (2.4) modifica somente o lado direito do sistema. Sendo assim, é preciso alterar no nosso programa apenas a sub-rotina INTEGRAL, somando o termo

$$- < 2A \left(\frac{U^M - U^{M-1}}{\Delta t} \right), \alpha_K \beta_L >,$$

ao lado direito do sistema. Observe que esta integral é do tipo calculado em (4.31).

Para o cálculo da solução aproximada no tempo t=0, U^0, podemos usar o mesmo procedimento apresentado na seção 3.3.1, ou seja, as mesmas definições dadas em (3.25). A expressão de s em (3.26), para o caso da equação com amortecimento (5.3), é da seguinte forma :

$$s = \Delta t u_1 + \frac{(\Delta t)^2}{2} [c \nabla^2 . u_0 + f^0 - 2A u_1] \qquad (5.4)$$

Portanto, para o cálculo da solução aproximada no tempo $t=\Delta t$, U^1, usaremos o mesmo procedimento descrito em (3.26), porém com a expressão de s dada por (5.4).

TESTE 8 :

Tomamos a função

$$f(x,y,t) = 100 \exp(-50(x^2 + y^2) - 40t^2)$$

como fonte do problema (5.3), localizada no centro do domínio $[-3,3]\times[-3,3]$, com as condições iniciais e função de amortecimento definidas abaixo.

Condições Iniciais :

$$u_0(x,y) = 0$$

$$u_1(x,y) = 0$$

Função de Amortecimento :

$$A(x,y) = \begin{cases} 0 & x^2+y^2 \leq R_1^2 \\ C_2 \exp\left(C_1(x^2+y^2) \right) & R_1^2 < x^2+y^2 \leq R_2^2 \\ U_{max} & x^2+y^2 > R_2^2 \end{cases}$$

onde,

$$\begin{cases} V_1 = U_{max} / U_{min} \\ V_2 = R_2^2 - R_1^2 \\ C_1 = \log(V_1) / V_2 \\ C_2 = U_{min} \exp(-C_1 / R_1^2) \end{cases}$$

Com as entradas dadas abaixo construímos os graficos que aparecem a seguir.

$T = [0,6]$

$N = 32$

$NT = 128$

$NPI = 2$

$C = 1$

$\lambda = 0.26$

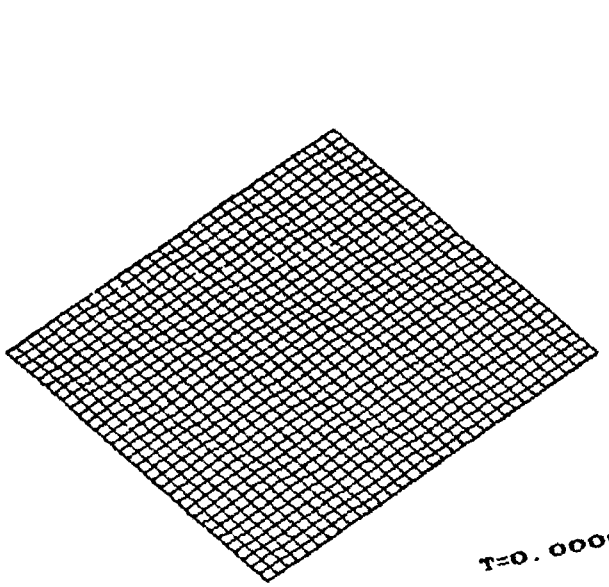
Dados para a definição da função de amortecimento :

$R_1 = 1.5$

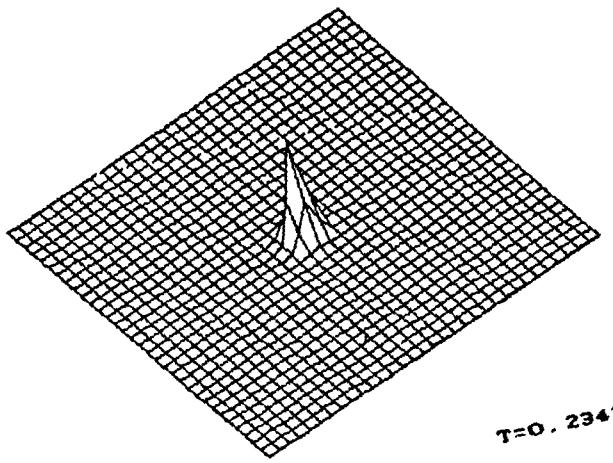
$R_2 = 3.0$

$U_{min} = 0.01$

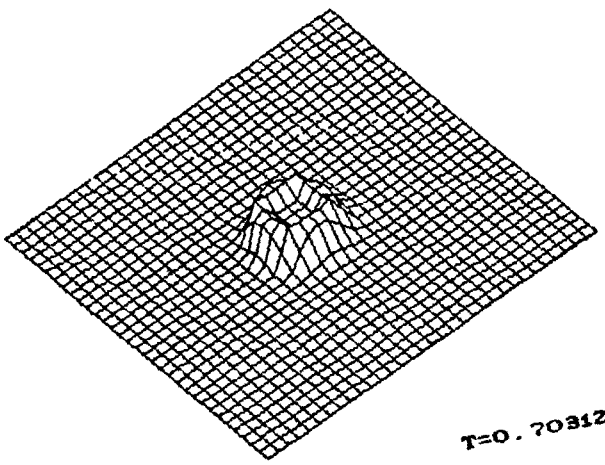
$U_{max} = 10$



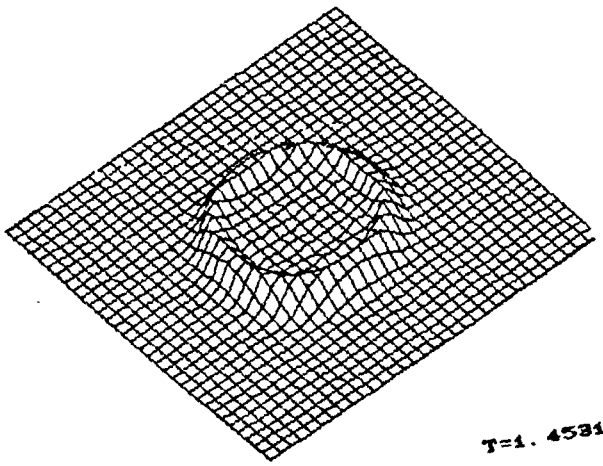
T=0. 000000



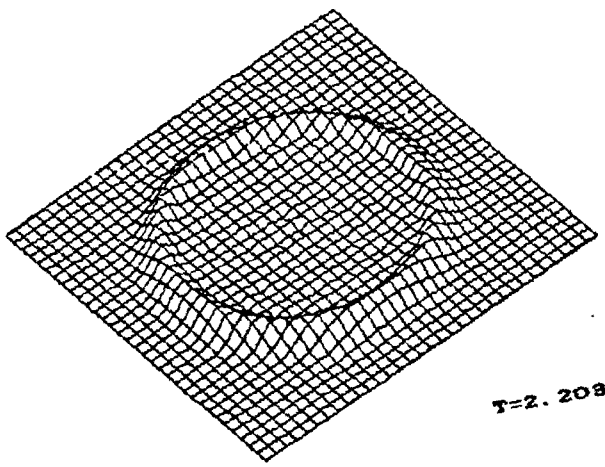
T=0. 234375



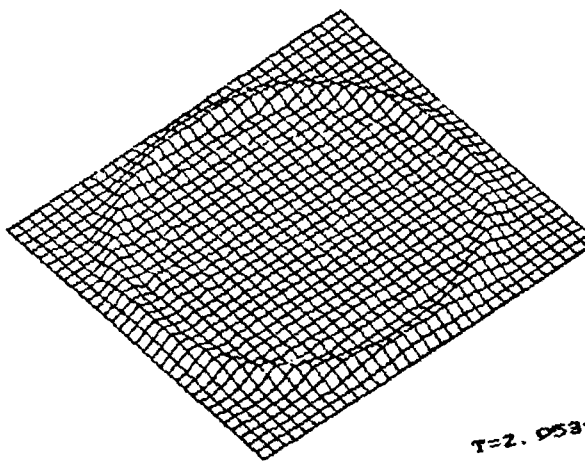
T=0. 703125



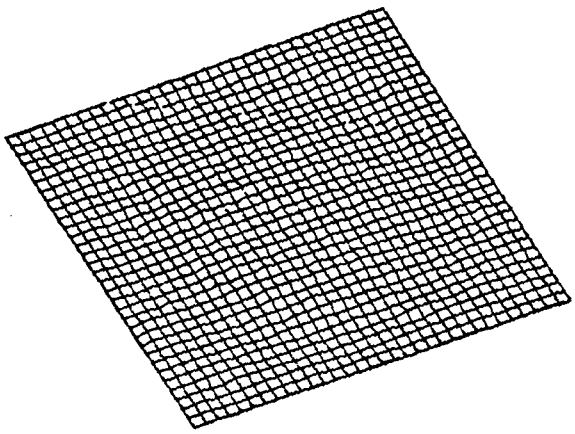
T=1. 453125



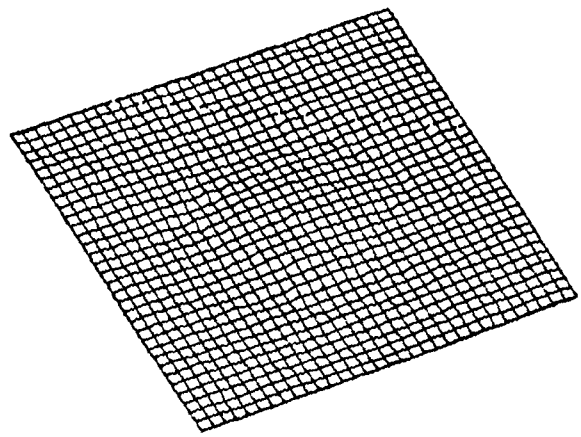
T=2. 203125



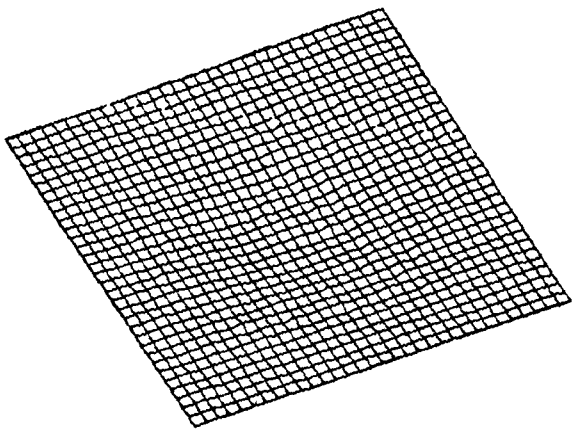
T=2. 953



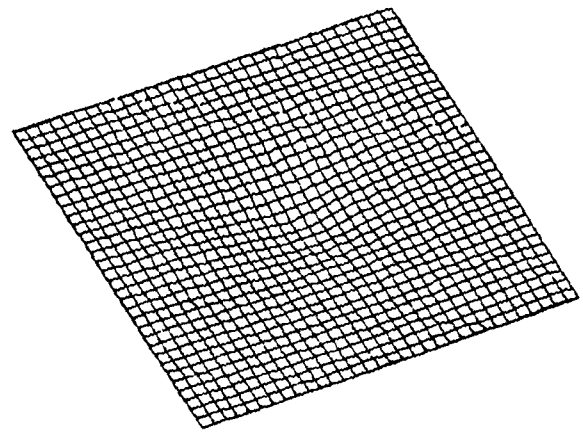
T=3. 703125



T=4. 453125



T=5. 203125



T=6. 000000

CONCLUSÕES E SUGESTÕES

Os estudos realizados comprovam que o método de Galerkin com Direções Alternadas fornece boas aproximações para a simulação numérica de propagação de ondas em meios limitados.

O uso dos polinômios de Hermite Cúbico significou um aprimoramento nas respostas obtidas pela aplicação de Galerkin com Direções Alternadas tomando como base elementos lineares [10], sem no entando tornar a implementação por demais complicada.

Podemos observar também que a combinação da estratégia de Direções Alternadas com o método de Galerkin reduziu significamente a memória necessária à sua implementação. Este fato é importante pois permite que a técnica seja aplicável em equipamentos computacionais compatíveis com a realidade da maioria dos centros de pesquisas nacionais (ex: a maioria das Universidades brasileiras).

Quanto ao problema de eliminar ou pelo menos reduzir as reflexões consequentes da limitação do domínio, consideramos este trabalho como uma reflexão inicial. Acreditamos ser importante e sugerimos sua continuidade, o que certamente trará grandes benefícios a diversas áreas de

aplicação.

Também sugerimos a comparação dos resultados obtidos neste trabalho com a utilização de condições de contorno absorventes como é recomendado nos trabalhos [03], [07], [08], [17] e [18]. Outra alternativa é misturar as duas técnicas propostas na biblografia especializada, isto é, usar uma camada absorvente aliada a condições de fronteira também absorventes; embora seja uma alternativa mais cara, esta seria uma solução otimizada.

Na nossa opinião, no caso de aplicações práticas repetitivas, o programa computacional poderia sofrer algumas modificações de maneira a otimizá-lo; apesar de termos investido na direção de elaborar o programa da maneira mais eficiente, não somos especialistas na área. Esta seria uma sugestão na direção de um trabalho com enfoque computacional como prioridade que poderia inclusive analisar sua implementação em equipamentos com arquitetura em paralelo.

CONVENCÕES

A numeração dos tópicos e subtópicos em cada capítulo obedece à seguinte ordem : número do capítulo, seguido do número do tópico e posteriormente do número do subtópico.

As fórmulas em cada capítulo são numeradas tomando inicialmente o número do capítulo e depois o número que indica a ordem de aparição da fórmula dentro do capítulo, isto para aquelas que necessitam ser mencionadas posteriormente.

As figuras e tabelas são numeradas como as fórmulas, precedidas dos nomes FIGURA E TABELA.

Usaremos neste trabalho, além da notação usual da teoria matemática, as seguintes convenções :

Ω = aberto do $\mathbb{R}^2$

$\partial\Omega$ = fronteira de Ω

$I = [a,b]$ = intervalo fechado $\{ x;\ a \leq x \leq b \}$

$[0,T]$ = intervalo fechado no tempo $\{ t;\ 0 \leq t \leq T \}$

Π : $a=x_1<x_2<\ldots<x_N<x_{N+1}=b$, partição regular de I

$h = x_i - x_{i-1}$

$\Delta t = t_i - t_{i-1}$

$L^p(\Omega)$ = espaço das funções p integráveis sobre Ω.

$H^r(\Omega)$ = espaço de Sobolev de ordem r

$H^{(m)}(\Pi)$ = espaço dos polinômios de Hermite de grau 2m-1

∇u = vetor gradiente de u

$\nabla . u$ = divergente de u

$\nabla^2 . u$ = laplaciano de u

$[a_{ij}]_{NXN}$ = matriz de ordem N

$[b_{ij}]_{Nx1}$ = vetor de comprimento N

$K \otimes L$ = produto tensorial entre as matrizes K e L

$\int_C f \, ds$ = integral sobre o contorno C

$\int_S f \, dxdy$ = integral sobre a superfície S

BIBLIOGRAFIA

[01] ALFORD, R.M., KELLY, K.R. and BOORE, D.M. *Accuracy of Finite-Difference Modeling of the Acoustic Wave-Equation.* **Geophysics (39) : 834-842**, 1974.

[02] BELLMAN, R. Introduction to Matrix Analysis. **McGraw-Hill**, New York, 1970.

[03] CLAYTON, R.W. & ENGQUIST, B. *Absorbing Boundary Conditions for Wave-Equation Migration.* **Geophysics (45) : 895-904**, 1980.

[04] CONTE, S.D. Elementos de Análise Numérica. **Editora Globo**, Porto Alegre, 1977.

[05] DEMIDOVICH, B.P. & MARON, I.A. Computation Mathematics. **Mir Publishers**, 1987.

[06] DOUGLAS J., J. & DUPONT, T. *Alternating-Direction Galerkin Methods on Retangles* in Numerical Solution of Partial Differential Equations. **B. Hubbard (2)**, Academic Press, New York, 1971.

[07] ENGQUIST, B. & MAJDA, A. *Absorbing Boundary Conditions for Numerical Simulation of Waves.* **Mathematics of Computation (31) : 629-651**, 1977.

[08] ENGQUIST, B. & MAJDA, A. *Radiation Boundary Conditions for Acoustic and Elastic Wave Calculations.*

Communications on Pure and Applied Mathematics (32) : 313-357, 1979

[09] FAIRWEATHER, G. Finite Element Galerkin Methods of Differential Equations. **MARCELL DEKKER, INC.**, New York and Basel, 1978.

[10] FERNANDES, J.A.N. *Onda Elastica-Galerkin com Direcoes Alternadas*. **IMECC-UNICAMP**, Campinas, 1988.

[11] FIGUEIREDO, D.G. Análise de Fourier e Equações Diferenciais Parciais. **IMPA-CNPq**, Rio de Janeiro, 1977.

[12] IÓRIO J., R. & IÓRIO, V.M. Equações Diferenciais Parciais : uma introdução. **IMPA-CNPq**, Rio de Janeiro, 1988.

[13] KELLY, K.R. et al. *Absorbing Boundary Conditions and Surface Waves*. **Geophysics (52) : 60-71**, 1987.

[14] MEDEIROS, L.A. Iniciação aos Espaços de Sobolev e Aplicações. **Instituto de Matematica-UFRJ**, Rio de Janeiro, 1983.

[15] MOURA, C.A., KUBRUSLY, R.S. and KRITZ, M.V. Elementos Finitos e Aplicações à Mecânica dos Fluidos. **LNCC**, Rio de Janeiro, 1984.

[16] NOBLE, B. & DANIEL, J.W. Álgebra Linear Aplicada. **Prentice-Hall do Brasil**, Rio de Janeiro, 1986.

[17] REYNOLDS, A.C. *Boundary Conditions for the Numerical*

Solution of Wave Propagation Problems. **Geophysics (43) : 1099-1110**, 1978.

[18] SMITH, W.D. *A Nonreflecting Plane Boundary for Wave Propagation Problems.* **Journal of Computation Physics (15) : 492-503**, 1974.

[19] SOCHACKI, J. at al. *Absorbing Boundary Conditions and Surface Waves.* **Geoghysics (52) : 60-71**, 1987.

[20] SOTOMAYOR, J. Lições de Equações Diferenciais Ordinárias. **IMPA-CNPq**, 1979.

[21] STRANG, G. & FIX, G.J. An Analysis of the Finite Element Method. **Prentice-Hall, Inc.**, Englewood Cliffs, 1973.

[22] TIJONOV, A.N. & SAMARSKY, A.A. Ecuaciones de la Fisica Matematica. **Editorial Mir**, Moscu, 1980.