Métodos Variacionais para Sistemas Lineares Esparsos : Uma Aplicação a Superfícies Livres de Capilaridade

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. Marcio Rodolfo Fernandes e aprovada pela Comissão Julgadora .

Campinas , 17 de setembro de 1993 .

Prof. Dr. Petronio Pulino

Petronio Pulino

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

Universidade Estadual de Campinas
Instituto de Matemática , Estatística e Ciências da Computação
Departamento de Matemática Aplicada

Métodos Variacionais para Sistemas Lineares Esparsos : Uma Aplicação a Superfícies Livres de Capilaridade

Marcio Rodolfo Fernandes

Prof. Dr. Petronio Pulino
(Orientador)

Campinas - São Paulo
Agosto de 1993

Aos meus pais , Antonio e Conceição ,
à minha irmã Morgana e
à minha futura esposa , Cristiane .

Agradecimentos

- *Aos amigos que cultivei nesta universidade .*

- *A Cristiane M. Alves Pissarra , pelas ilustrações deste trabalho e pelos bons momentos que tivemos em comum .*

- *Ao Prof. Petronio Pulino , pela orientação e pela infra-estrutura a mim confiada .*

- *Ao Prof. João Frederico (Joni) , pela atenção dedicada desde os meus primeiros dias nesta universidade .*

You Know the day destroys the night ,
Night divides the day ;
Tried to run , tried to hide
Break on through to the other side .

(Jim Morrison)

ÍNDICE

LISTA DE SÍMBOLOS E ABREVIAÇÕES vi

INTRODUÇÃO viii

CAPÍTULO 1 - Métodos de Direções Conjugadas para Sistemas Lineares 1

1.1 - Introdução 1

1.2 - O Método Generalizado de Direções Conjugadas (GCD) 1

1.2.1 - Método dos Gradientes Conjugados (CG) 4

1.2.2 - Métodos de Descida para Sistemas Não-Simétricos 6

1.2.3 - Método Generalizado de Resíduos Mínimos (GMRES) 9

1.3 - Convergência do Método Generalizado de Direções Conjugadas 12

1.4 - Algoritmo Generalizado de Direções Conjugadas 15

CAPÍTULO 2 - Gradientes Conjugados Quadrático (CGS) 18

2.1 - Uma Forma Polinomial Equivalente ao Método CG 18

2.2 - Gradientes Conjugados Quadrático (CGS) 25

CAPÍTULO 3 - Pré-Condicionadores 29

3.1 - Introdução 29

3.2 - Gradientes Conjugados com Pré-Condicionamento (PCG) 29

3.3 - CGS com Pré-Condicionamento (PCGS) 33

3.4 - Fatorações Incompletas 35

3.5 - Exemplos Numéricos : O Problema de Difusão-Convecção 37

CAPÍTULO 4 - Cálculo em Espaços Normados 40
4.1 - Introdução 40
4.2 - A Diferencial Forte (Fréchet) 40
4.3 - A Diferencial Fraca (Gateaux) 41
4.4 - O Vínculo entre as Diferenciações Forte e Fraca 42
4.5 - O Método de Newton 42
4.6 - O Método Inexato de Newton 48
4.7 - Equação de Poisson Não-Linear 50

CAPÍTULO 5 - Superfície Livre de Capilaridade 52
5.1 - Introdução 52
5.2 - O Método de Gauss : Caracterização das Energias 55
5.2.1 - Energia Superficial Livre (E_s) 56
5.2.2 - Energia de Molhabilidade (E_w) 57
5.2.3 - Energia Potencial Gravitacional (E_p) 58
5.2.4 - Restrição de Volume 58
5.3 - Princípio da Mínima Energia 59
5.4 - Tubo Capilar Cilíndrico Bidimensional 65
5.4.1 - Colocação do Problema 65
5.4.2 - Linearização 67
5.4.3 - Discretização e Resultados Numéricos 69
5.5 - Capilar Cilíndrico Tridimensional com Simetria Axial 75
5.5.1 - Colocação do Problema 75
5.5.2 - Linearização 77
5.5.3 - Discretização e Resultados Numéricos 78

CONCLUSÕES 81

APÊNDICE - Rotinas para o Método de Direções Conjugadas : Sistemas Lineares Esparsos 83
Introdução 83
Rotinas dos Principais Métodos Desenvolvidos 84
Rotinas para Manipulação Matricial 97

BIBLIOGRAFIA 111

LISTA DE SÍMBOLOS E ABREVIAÇÕES

$B(x,r)$: bola de centro em $x \in \mathbb{R}^n$ e raio r ;

BI-CG : método dos Gradientes Bi-Conjugados ;

BI-CGSTAB : método BI-CG Estabilizado ;

CG : método dos Gradientes Conjugados ;

CGNE : método CG aplicado às equações normais ;

CGS : método dos Gradientes Conjugados Quadrático ;

CR : método dos Resíduos Conjugados ;

$C(\Omega)$: espaço vetorial das funções contínuas no domínio Ω;

$C^2(\Omega)$: espaço vetorial das funções contínuas e com derivadas contínuas até segunda ordem , no domínio Ω ;

$\text{cond}_2(\mathbf{A})$: número de condição espectral da matriz $\mathbf{A}$;

$\cos(\delta)$: cosseno do argumento δ ;

$\text{cotan}(\delta)$: cotangente do argumento δ ;

dim : dimensão ;

div : divergente ;

$\partial\Omega$: fronteira do domínio Ω ;

δ_{ij} : Delta de Kronecker - $\delta_{ij} = \begin{cases} 1 \text{ se } i = j ; \\ 0 \text{ se } i \neq j ; \end{cases}$

E_p : Energia Potencial Gravitacional ;

E_s : Energia Superficial Livre ;

E_v : energia associada à restrição de volume ;

E_w : Energia de Molhabilidade ;

g : aceleração da gravidade ;

H : curvatura média ;

GCD : método Generalizado de Direções Conjugadas ;

GCR : método Generalizado de Resíduos Conjugados ;

GMRES : método Generalizado de Resíduos Mínimos ;

ILU : decomposição LU incompleta ;

$K_k(A,b)$: subespaço de Krylov associado à matriz A e ao vetor b

$K_k(A,b) = \text{span}\{ b , Ab , A^2b , \ldots , A^{k-1}b \}$;

λ : multiplicador de Lagrange ;

$L(X,Y)$: espaço vetorial dos operadores lineares limitados do espaço normado X no espaço normado Y ;

lim : limite ;

PCG : método CG pré-condicionado ;

PCGS : método CGS pré-condicionado ;

PCGNE : método CGNE pré-condicionado ;

pd : positiva-definida ;

Π^N : espaço dos polinômios reais de grau menor que N ;

ρ : densidade de massa ;

$\mathbb{R}^{n\times n}$: espaço vetorial das matrizes quadradas de ordem n ;

σ : tensão superficial ;

$\sin(\delta)$: seno do argumento δ ;

$\text{span}\{ v_1 , \ldots , v_n \} \equiv [v_1 , \ldots , v_n]$:
espaço vetorial gerado por todas as combinações lineares dos vetores v_1 ;

spd : simétrica e positiva-definida ;

$\langle x,y \rangle$: produto interno entre os vetores x e y ;

Δu : Laplaciano da função u ;

∇u : Gradiente da função u ;

$\nabla^2 u$: Hessiana da função u ;

$\| \cdot \|$: norma ;

$\| \cdot \|_H$: norma energia associada à matriz H ;

$\| \cdot \|_2$: norma-2 ;

INTRODUÇÃO

Os métodos para a resolução de sistemas lineares que requerem a fatoração da matriz **A** dos coeficientes são chamados Métodos Diretos . Estes podem se tornar impraticáveis se a matriz **A** for de grande porte e esparsa porque seus fatores , geralmente , serão matrizes cheias , isto é , sem estrutura esparsa . Uma exceção a este fato ocorre quando **A** tem estrutura de banda . Ainda assim , os algoritmos de fatoração podem tornar-se de difícil implementação computacional .

Uma das razões para o grande interesse em sistemas lineares esparsos é a importância da análise numérica aplicada às equações diferenciais . Sabe-se que as pesquisas nesta área têm sido responsáveis pela maioria das estratégias para o tratamento da esparsidade .

Mais detalhadamente , existem duas formas de atacar um problema esparso $\mathbf{Ax} = \mathbf{b}$. Uma delas é escolher um método direto e adaptar seu algoritmo de forma a explorar a esparsidade de **A** . Esta adaptação envolve o uso de estrutura de dados propícias ao seu problema e estratégias especiais de pivoteamento que minimizem o preenchimento da matriz .

Em contraste com os métodos diretos estão os métodos iterativos. Estes métodos geram uma sequência de soluções aproximadas $\{\mathbf{x}_k\}$ e essencialmente envolvem somente produtos do tipo matriz-vetor . Desta forma a estrutura de **A** não é alterada , isto é , não há o aparecimento de novos elementos na matriz tornando-se dispensável o uso de estrutura de dados dinâmica .

Nesta classe de métodos, o Método dos Gradientes Conjugados (CG), desenvolvido por Hestenes e Stiefel , tem sido amplamente usado para resolver sistemas lineares esparsos de grande porte onde a matriz **A** é real , quadrada de ordem n , simétrica e positiva-definida (*spd*) . Vide as referências [7],[8],[9],[12],[24] e [25] . O Método dos Gradientes Conjugados também pode ser visto como um método direto que , na ausência de erros de arredondamento , obtém a solução do sistema em n iterações , ou como um método iterativo que , em certas condições , fornece uma boa aproximação da solução em poucos passos .

Este método envolve basicamente um produto de matriz por vetor a cada passo . Esta característica torna-o particularmente vantajoso para sistemas lineares esparsos de grande porte pois economiza tempo de execução e memória para armazenamento de dados . Outra vantagem é que , ao contrário de alguns métodos , CG não exige nenhuma estimativa de parâmetros .

Entretanto , quando o sistema não tem simetria e positividade , caso da discretização de equações diferenciais elípticas não auto-adjuntas , o algoritmo não pode ser diretamente aplicado . Por isso muitos métodos têm sido pesquisados com a finalidade de resolver problemas deste tipo e que possuam propriedades semelhantes às do Método dos Gradientes Conjugados . São os chamados Métodos Tipo Gradientes Conjugados ou Direções Conjugadas (veja [7]).

No capítulo 1 esta família de métodos é apresentada . Entre seus integrantes encontra-se , logicamante , o método CG . É claro que CG pode também ser aplicado a um sistema linear qualquer , A não-singular, através da formação (não explícita) das equações normais

$$A^t A x = A^t b \quad ,$$

mas esta técnica tende a retardar a convergência do método que é conhecido como Gradientes Conjugados Aplicado às Equações Normais (CGNE) .

Ainda para sistemas não-singulares , sem a necessidade de positividade ou simetria , temos o Método Generalizado de Resíduos Mínimos (GMRES) que constrói uma base adequada para o espaço de solução , o que também é feito de maneira mais elegante pela maioria dos métodos desta família . Infelizmente a demanda de espaço de memória do método GMRES cresce linearmente com o número de iterações . A maioria dos métodos desta família podem ser vistos como variantes do método de aproximação de Petrov-Galerkin como pode ser conferido nas referências [20] , [21] e [22] .

No capítulo 2 , construímos um método derivado do Método dos Gradientes Conjugados e que também não exige simetria ou positividade da matriz : BI-CG ou Gradientes Bi-Conjugados ([8] , [24] e [26]) .

Este método tem a vantagem de não necessitar da formação , mesmo que implícita , das equações normais evitando , assim , o aumento do número de condição espectral da matriz . Por outro lado , exige praticamente o dobro do número de operações de CG , a cada iteração .

Uma variante de BI-CG , que tem se tornado bastante popular recentemente , também é apresentada neste capítulo . Trata-se do método CGS ou Gradientes Conjugados Quadrático encontrado nas referências [8] , [24] , [25] e [26] .

A avaliação de um método iterativo baseia-se invariavelmente na rapidez de sua convergência . Para acelerá-la , é indispensável o uso de técnicas de pré-condicionamento . No capítulo 3 , algumas destas técnicas são apresentadas , implementadas e testadas num problema de Difusão-Convecção , com base nas referências [12] , [18] e [24] .

No capítulo 5 , simulações numéricas e testes são executados em sistemas oriundos da discretização de equações diferenciais parciais do tipo elíptico que descrevem um problema de Superfícies Livres de Capilaridade (veja [1] , [3] , [4] e [11]) . Como este problema é não-linear , alguns resultados sobre linearização de operadores em espaços normados são apresentados no capítulo 4 (veja [5] , [10] , [15] , [16] , [19] e [23]) .

No apêndice encontramos o código Fortran dos principais métodos apresentados neste trabalho , bem como das rotinas mais importantes para a manipulação matricial da forma com a qual as matrizes foram armazenadas .

CAPÍTULO 1

MÉTODOS DE DIREÇÕES CONJUGADAS PARA SISTEMAS LINEARES

1.1 - INTRODUÇÃO

Apresentaremos a unificação de uma classe de métodos de direções conjugadas para a solução do sistema linear

$$\mathbf{Ax} = \mathbf{b} \tag{1.1}$$

com $\mathbf{A} \in \mathbb{R}^{n \times n}$ não-singular e esparsa .

A classe considerada consiste de métodos que , a cada passo , minimizam um funcional de erro sobre um subespaço afim . Nesta família estão inclu í dos o método dos Gradientes Conjugados , Resíduos Conjugados , ORTHOMIN e GMRES , entre outros .

1.2 - O MÉTODO GENERALIZADO DE DIREÇÕES CONJUGADAS (GCD)

O Método Generalizado de Direções Conjugadas é baseado na busca do minimizador da forma quadrática

$$q(\mathbf{x}) = \frac{1}{2}\,\mathbf{x}^t\mathbf{Hx} - \mathbf{h}^t\mathbf{x} ,$$

onde supomos a matriz $\mathbf{H}$ simétrica e positiva-definida . Desta forma , $q(\mathbf{x})$ tem um único minimizador e , encontrá-lo é equivalente a procurar a solução da equação

$$\nabla q(\mathbf{x}) \equiv \mathbf{Hx} - \mathbf{h} = 0 .$$

Para usar o método GCD na resolução do sistema linear $\mathbf{Ax} = \mathbf{b}$ podemos tomar $\mathbf{H} = \mathbf{A}$, $\mathbf{h} = \mathbf{b}$ no caso em que $\mathbf{A}$ for simétrica e positiva-definida (*spd*) , ou $\mathbf{H} = \mathbf{ZA}$, $\mathbf{h} = \mathbf{Zb}$ com $\mathbf{Z}$ não-singular e $\mathbf{ZA}$ *spd* quando isto não ocorrer .

Usaremos a notação

$$\bar{\mathbf{r}}_k = - \nabla q(\mathbf{x}_k) = \mathbf{h} - \mathbf{H}\mathbf{x}_k$$

e

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k \ .$$

O minimizador de $q(\mathbf{x})$ é

$$\mathbf{x}^* = \mathbf{H}^{-1}\mathbf{h} \ ,$$

e o valor mínimo de $q(\mathbf{x})$ é

$$q(\mathbf{x}^*) = - \frac{1}{2} \mathbf{h}^t\mathbf{H}^{-1}\mathbf{h} \ .$$

Definimos

$$\| \mathbf{x} - \mathbf{x}^* \|_H^2 \equiv (\mathbf{x} - \mathbf{x}^*)^t\mathbf{H}(\mathbf{x} - \mathbf{x}^*) \ ,$$

e observamos que

$$\| \mathbf{x} - \mathbf{x}^* \|_H^2 = \mathbf{x}^t\mathbf{H}\mathbf{x} - 2\mathbf{x}^t\mathbf{H}\mathbf{x}^* + \mathbf{x}^{*t}\mathbf{H}\mathbf{x}^* = 2 \ (\ q(\mathbf{x}) - q(\mathbf{x}^*) \ ) \ ,$$

de forma que minimizar $q(\mathbf{x})$ é equivalente a minimizar $\| \mathbf{x} - \mathbf{x}^* \|_H$.

O método GCD produz uma sequência $\mathbf{p}_1, \mathbf{p}_2, \ \ldots$ de direções linearmente independentes e H-ortogonais , isto é , $\mathbf{p}_i^t\mathbf{H}\mathbf{p}_j = 0$, $i \neq j$. Conceitualmente , não tomamos estas direções . Ao invés disso , a cada passo tomamos uma direção $\mathbf{d}_k$ que apenas não seja ortogonal ao gradiente de q em $\mathbf{x}_{k-1}$. Determinamos $\mathbf{x}_k$ como o minimizador de $q(\mathbf{x})$ no subespaço gerado por $\{ \mathbf{p}_1, \mathbf{p}_2, \ \ldots \ , \mathbf{p}_{k-1}, \mathbf{d}_k \}$ e definimos $\mathbf{p}_k \equiv \mathbf{x}_k - \mathbf{x}_{k-1}$. Obviamente a escolha de $\mathbf{d}_k$ determina $\mathbf{p}_k$.

Para facilitar a exposição , usaremos $\mathbf{x}_0 = 0$ como aproximação inicial . Isto não é restritivo . Se alguma melhor aproximação $\bar{\mathbf{x}}$ é conhecida , podemos resolver $\mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}) = \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$ com aproximação inicial nula .

Algoritmo GCD :

$\mathbf{x}_0 = 0 \quad , \quad \bar{\mathbf{r}}_0 = \mathbf{h} \quad , \quad k = 1$

Enquanto $\bar{\mathbf{r}}_{k-1} \neq 0$

tome $\mathbf{d}_k$ tal que $\mathbf{d}_k^t \bar{\mathbf{r}}_{k-1} \neq 0$

$\mathbf{x}_k = \arg \min q(\mathbf{x})$
$\mathbf{x} \in \text{span} \{ \mathbf{p}_1, \ldots, \mathbf{p}_{k-1}, \mathbf{d}_k \}$

$\mathbf{p}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$

$\bar{\mathbf{r}}_k = \bar{\mathbf{r}}_{k-1} - \mathbf{H}\mathbf{p}_k$

$k = k + 1$

fim

Na tabela 1.1 encontramos as várias escolhas de $\mathbf{H}$, $\mathbf{h}$ e $\mathbf{d}_k$ e o respectivo método obtido a partir do Método Generalizado de Direções Conjugadas . Uma visão destes métodos será dada a seguir .

Algoritmo	*Imposições*	$\mathbf{H}$	$\mathbf{h}$	$\mathbf{d}_k$
CG	$\mathbf{A}$ *spd*	$\mathbf{A}$	$\mathbf{b}$	$\mathbf{r}_{k-1}$
CGNE	$\mathbf{A}$ não-singular	$\mathbf{A}^t\mathbf{A}$	$\mathbf{A}^t\mathbf{b}$	$\bar{\mathbf{r}}_{k-1}$
CR	$\mathbf{A}$ *spd*	$\mathbf{A}^t\mathbf{A}$	$\mathbf{A}^t\mathbf{b}$	$\mathbf{r}_{k-1}$
PCG	$\mathbf{A}$ *spd*	$\mathbf{A}$	$\mathbf{b}$	$\mathbf{M}^{-1}\mathbf{r}_{k-1}$
GCR	$\mathbf{A} + \mathbf{A}^t$ *pd*	$\mathbf{A}^t\mathbf{A}$	$\mathbf{A}^t\mathbf{b}$	$\mathbf{r}_{k-1}$
ORTHOMIN	$\mathbf{A} + \mathbf{A}^t$ *pd*	$\mathbf{A}^t\mathbf{A}$	$\mathbf{A}^t\mathbf{b}$	$\mathbf{r}_{k-1}$
GMRES	$\mathbf{A}$ não-singular	$\mathbf{A}^t\mathbf{A}$	$\mathbf{A}^t\mathbf{b}$	$\in [\mathbf{v}_1, \ldots, \mathbf{v}_m]^*$

Tabela 1.1 : Família GCD .

(*) $[\mathbf{v}_1 , \ldots , \mathbf{v}_m]$ é uma base ortonormal para $K_m(A, \mathbf{r}_0)$

1.2.1 - MÉTODO DOS GRADIENTES CONJUGADOS (CG)

Este método é próprio para sistemas nos quais a matriz **A** é simétrica e positiva definida . Sem dúvida nenhuma , é o mais importante e eficiente dos métodos para a resolução de sistemas lineares esparsos e de grande porte . Por isso , desde a década passada , muitas generalizações do método dos Gradientes Conjugados para matrizes não-simétricas e sem positividade têm sido pesquisadas .

A partir de uma aproximação inicial $\mathbf{x}_0$ para a solução de (1.1) , ele gera uma sequência $\mathbf{x}_1, \mathbf{x}_2, \ldots$, de aproximações com as seguintes propriedades principais :

- $\mathbf{x}_k \in \mathbf{x}_0 + \mathbf{K}_k(\mathbf{A}, \mathbf{r}_0)$ (subespaço de afim) ;
- $\mathbf{x}_k$ é o argumento para o qual $\| \mathbf{x} - \mathbf{u} \|_A$ é mínima , para todo $\mathbf{u} \in \mathbf{x}_0 + \mathbf{K}_k$, onde $\| \mathbf{v} \|_A^2 = \langle \mathbf{v}, \mathbf{A}\mathbf{v} \rangle$;

Como consequência das propriedades acima temos que , na ausência de erros de arredondamento , o método converge para a solução do sistema $\mathbf{x}^*$ em n (ordem da matriz) passos , no máximo .

Os vetores $\mathbf{r}_k$ e $\mathbf{p}_k$ (veja o algoritmo abaixo) são conhecidos como resíduos e direções de busca , respectivamente , e têm as seguintes propriedades :

As direções de busca são **A**-conjugadas , isto é ,

$$\langle \mathbf{p}_i, \mathbf{A}\mathbf{p}_j \rangle = 0 , \quad i \neq j$$

e os resíduos são ortogonais , ou seja ,

$$\langle \mathbf{r}_i, \mathbf{r}_j \rangle = 0 , \quad i \neq j .$$

Algoritmo CG :

$k = 0$

$r_0 = b - Ax_0$

$p_1 = r_0$

Enquanto $\| r_k \| >$ tolerância

$k = k + 1$

$$\beta_k = \frac{r_{k-1}^t r_{k-1}}{r_{k-2}^t r_{k-2}} \qquad (\beta_1 \equiv 0)$$

$$p_k = r_{k-1} + \beta_k p_{k-1}$$

$$\alpha_k = \frac{r_{k-1}^t r_{k-1}}{p_k^t A p_k}$$

$$x_k = x_{k-1} + \alpha_k p_k$$

$$r_k = r_{k-1} - \alpha_k A p_k$$

fim

Para acelerar a convergência do Método dos Gradientes Conjugados, tomamos uma matriz não-singular e simétrica C^{-1} tal que $C^{-1}AC^{-1}$ seja melhor condicionada que A e , transformamos o sistema (1.1) em

$$(C^{-1}AC^{-1})(Cx) = C^{-1}b \ .$$

A matriz M que aparece na escolha de d_k , na tabela *1.1* , é igual a C^2 e é chamada de pré-condicionador .

Desta forma temos o Método dos Gradientes Conjugados com Pré-condicionamento (PCG) . Algumas formas de pré-condicionar um sistema linear serão vistas mais detalhadamente no capítulo 3 .

No caso em que a matriz A não for simétrica e positiva-definida , podemos aplicar este método às equações normais

$$A^tA\ x = A^tb \quad ,$$

sem a formação explícita do produto A^tA para evitar a perda da esparsidade da matriz do sistema . Desta forma , o método passa a ser conhecido por Gradientes Conjugados aplicado às Equações Normais (CGNE) . Esta prática , apesar de simples , geralmente não é eficiente pois contribui para o retardamento da convergência do método , como veremos no capítulo 3 .

1.2.2 - MÉTODOS DE DESCIDA PARA SISTEMAS NÃO-SIMÉTRICOS

Nesta seção , apresentaremos três variantes do Método dos Gradientes Conjugados e que requerem que a parte simétrica da matriz A seja positiva-definida , isto é ,

$$\frac{A + A^t}{2}$$

positiva-definida .

Eles se diferem uns dos outros pelo trabalho computacional e pelo uso da memória e têm a seguinte forma geral :

Tome x_0 (aproximação inicial)

$r_0 = b - Ax_0$

$p_0 = r_0$

$i = 0$ até convergir faça

$$\alpha_i = \frac{r_i^t A p_i}{p_i^t A^t A p_i}$$

$$x_{i+1} = x_i + \alpha_i p_i \qquad r_{i+1} = r_i - \alpha_i A p_i$$

Compute p_{i+1}

A escolha de α_i no algoritmo acima minimiza

$$\| r_{i+1} \|_2 = \| b - A(x_i + \alpha p_i) \|_2$$

como função de α , de forma que a norma Euclidiana do resíduo decresça a cada passo . Os métodos diferem na técnica usada para computar a nova direção p_{i+1} .

Uma boa escolha para p_{i+1} é aquela que resulta um significativo decréscimo da norma do resíduo , mas não exige muito trabalho computacional . Quando A é simétrica e positiva-definida , tais vetores podem ser computados pela relação

$$p_{i+1} = r_{i+1} + \beta_i p_i \quad , \tag{1.2}$$

onde

$$\beta_i = - \frac{r_{i+1}^t A^t A p_i}{p_i^t A^t A p_i} \, . \tag{1.3}$$

O método definido com a escolha de p_{i+1} por (1.2)-(1.3) é conhecido como Método dos Resíduos Conjugados (CR) . Os vetores direção produzidos são A^tA-ortogonais , isto é,

$$p_i^t A^t A p_j = 0 \text{ , para } i \neq j \text{ ,} \tag{1.4}$$

e x_{i+1} minimiza o funcional

$$E(w) = \| b - Aw \|_2$$

sobre o subespaço afim $x_0 + \text{span} \{ p_0 , p_1 , \ldots , p_i \}$.

Se A for não-simétrica , a relação de ortogonalidade (1.4) não vale , em geral . Entretanto , um conjunto de direções A^tA-ortogonais pode ser gerado usando-se todas as direções anteriores $\{ p_j \}_{j=0}^{i}$ para calcular p_{i+1} :

$$p_{i+1} = r_{i+1} + \sum_{j=0}^{i} \beta_j^{(i)} p_j \quad , \tag{1.5}$$

$$\beta_j^{(i)} = - \frac{r_{i+1}^t A^t A p_j}{p_j^t A^t A p_j} \quad , \quad j \leq i \; . \tag{1.6}$$

A iteração x_{i+1} gerada desta forma também minimiza $E(w)$ sobre o subespaço afim x_0 + span $\{ p_0 , p_1 , \ldots , p_i \}$. Nos referimos a este algoritmo como Método Generalizado de Resíduos Conjugados (GCR) . Na ausência de erros de arredondamento , o método GCR também converge para a solução de (1.1) em n iterações , no máximo .

O trabalho computacional e o armazenamento de variáveis por iteração podem ser proibitivamente grandes quando n (dimensão do sistema) for grande . Para evitar este problema , podemos propor uma modificação do Método Generalizado de Resíduos Conjugados , conhecida como ORTHOMIN, que tem um custo significativamente menor por iteração . Ao invés de fazer p_{i+1} A^tA-ortogonal a todas as direções anteriores, podemos fazer p_{i+1} A^tA-ortogonal somente aos k últimos vetores direção $\{ p_j \}_{j=i-k+1}^{i}$:

$$p_{i+1} = r_{i+1} + \sum_{j=i-k+1}^{i} \beta_j^{(i)} p_j \quad , \tag{1.7}$$

com $\{ \beta_j^{(i)} \}_{j=i-k+1}$ definidos como em (1.6) . Desta forma , somente k vetores direção precisam ser guardados .

Outra alternativa é recomeçar o método GCR a cada $k + 1$ iterações tomando a iteração corrente como nova aproximação inicial . Assim , também teremos um armazenamento máximo de k vetores direção , porém os custos por iteração serão menores desde que geralmente menos de k direções são usadas para calcular p_{i+1} .

1.2.3 - MÉTODO GENERALIZADO DE RESÍDUOS MÍNIMOS (GMRES)

O método GMRES constrói uma base ortonormal $\{ v_1, v_2, \ldots, v_m \}$ para o subespaço de Krylov $K_m(A,r_0)$ e, usando esta base, determina a solução de $Ax = b$ como o minimizador de $\| Ax - b \|_2$ em $K_m(A,r_0)$.

Seja x_0 uma aproximação inicial para a solução x^* do sistema (1.1), com $r_0 = b - Ax_0$.

Se x^* for decomposto em $x^* = x_0 + z$, então

$$A(x_0 + z) = b \Rightarrow Az = r_0 \quad . \tag{1.8}$$

Seja $V_m = [v_1, v_2, \ldots, v_m]$ uma matriz nxm cujas colunas formam uma base ortonormal para $K_m(A,r_0)$.

Procuraremos uma aproximação $z^{(m)}$ para z ($z^{(m)} \in K_m$) tal que

$$V_m^t A z^{(m)} - V_m^t r_0 = 0 \quad ,$$

mas $z^{(m)} = V_m y^{(m)}$, logo

$$V_m^t A \, V_m y^{(m)} - V_m^t r_0 = 0 \quad . \tag{1.9}$$

Então, a solução aproximada para o sistema linear $Ax = b$ será dada por

$$x^{(m)} = x_0 + z^{(m)} \quad ,$$

onde $z^{(m)} = V_m y^{(m)}$ e $y^{(m)}$ é a solução do sistema linear de ordem m (1.9).

Para construir uma base ortonormal para o subespaço de Krylov $K_m(A,r_0)$ usamos o seguinte algoritmo , conhecido como Método de Arnoldi , e que se utiliza do Método de Gram-Schmidt para encontrar V_m :

$$r_0 = b - Ax_0$$

$$v_1 = \frac{r_0}{\| r_0 \|_2}$$

k = 1 até m faça

$$w = Av_k - \sum_{l=1}^{k} h_{lk} v_l \quad \text{com} \quad h_{lk} = \langle Av_k, v_l \rangle$$

$$h_{k+1,k} = \| w \|_2$$

$$v_{k+1} = \frac{w}{h_{k+1,k}}$$

Para descrever o método GMRES , devemos notar que após *m* passos do Método de Arnoldi teremos um sistema ortonormal V_{m+1} e uma matriz H_m , $(m+1) \times m$, cujos únicos elementos não-nulos são os h_{ij} gerados pelo método . Os vetores v_1 e a matriz H_m satisfazem a importante relação

$$AV_m = V_{m+1}H \, . \tag{1.10}$$

Queremos resolver o problema de quadrados mínimos

$$\min_{z \in K_m} \| b - A(x_0 + z) \|_2 = \min_{z \in K_m} \| r_0 - Az \|_2 \, . \tag{1.11}$$

Se fizermos $z = V_m y$, podemos ver a norma a ser minimizada como a seguinte função de y

$$J(y) = \| \beta v_1 - AV_m y \|_2 ,$$

onde $\beta = \| r_0 \|_2$. Usando (1.10) temos

$$J(y) = \| V_{m+1}(\beta e_1 - H_m y) \|_2 .$$

Aqui , o vetor e_1 é a primeira coluna da matriz identidade de ordem $m+1$. Lembrando que V_{m+1} é ortogonal e que transformações ortogonais preservam a norma-2 , podemos ver que

$$J(y) = \| \beta e_1 - H_m y \|_2 . \qquad (1.12)$$

Portanto ,

$$x_m = x_0 + V_m y_m$$

onde y_m minimiza o funcional $J(y)$, definido em (1.12) , sobre todos os $y \in \mathbb{R}^m$.

Algoritmo GMRES :

- Método de Arnoldi

- $x^{(m)} = x_0 + V_m y^{(m)}$, onde $y^{(m)}$ minimiza

$$J(y) = \| \beta e_1 - H_m y \|_2 , \quad y \in \mathbb{R}^m .$$

Quando m cresce , o número de vetores que precisam ser armazenados cresce linearmente em relação a m . Para contornar esta dificuldade podemos recomeçar o método GMRES a cada k iterações .

1.3 - CONVERGÊNCIA DO MÉTODO GENERALIZADO DE DIREÇÕES CONJ

Teorema 1.1 : *Se* $\mathbf{H}$ *é simétrica e positiva-definida , então as seguintes afirmações sobre o método* GCD *são verdadeiras :*

i) *O algoritmo termina em , no máximo , n passos , e termina se, e somente se* $\mathbf{x}_{k-1} = \mathbf{x}^*$ *(solução do sitema linear)* ;

ii) $\mathbf{x}_k$ *minimiza* $q(\mathbf{x})$ *no subespaço* $S_k = \text{span} \{ \mathbf{p}_1, \ldots , \mathbf{p}_k \}$;

iii) *Todo* $\mathbf{p}_k$ *gerado é não-nulo* ;

iv) $\bar{\mathbf{r}}_k^t \mathbf{p}_j = 0 \ , \ 1 \le j \le k$;

v) $\mathbf{p}_i^t \mathbf{H} \mathbf{p}_j = 0 \ , \ i \ne j$;

vi) $\dim S_k = k$;

vii) $\mathbf{p}_i^t \mathbf{H} \mathbf{p}_i = \mathbf{p}_i^t \mathbf{h}$, *para todo* i .

Se , além disso , $\mathbf{d}_k \in \text{span} \{ \mathbf{d}_1, \mathbf{B}\mathbf{p}_1, \ldots , \mathbf{B}\mathbf{p}_{k-1} \}$ *para alguma matriz* $\mathbf{B}$ *, então*

viii) $S_k = \mathbf{K}_k(\mathbf{B},\mathbf{d}_1) \equiv \text{span} \{ \mathbf{d}_1, \mathbf{B}\mathbf{d}_1, \ldots , \mathbf{B}^{k-1}\mathbf{d}_1 \}$, *tal que* $\mathbf{x}_k$ *minimiza* $q(\mathbf{x})$ *em* $\mathbf{K}_k(\mathbf{B},\mathbf{d}_1)$;

ix) $\bar{\mathbf{r}}_k^t \mathbf{B} \mathbf{p}_i = 0 \ , \ 1 \le i \le k-1$.

Demonstração : O algoritmo termina só quando $\bar{\mathbf{r}}_{k-1} = 0$. Então $0 = \bar{\mathbf{r}}_{k-1} = - \nabla q(\mathbf{x}_{k-1})$, de forma que $\mathbf{x}_{k-1}$ é ponto crítico de $q(\mathbf{x})$. Como $\nabla^2 q(\mathbf{x}_{k-1}) = \mathbf{H}$ é positiva-definida , isto é necessário e suficiente para que $\mathbf{x}_{k-1} = \mathbf{x}^*$. Vamos mostrar que o algoritmo termina em n passos , no máximo , provando (ii) e (vi) para $k = n$,por indução .

Para $k = 1$, se $\bar{\mathbf{r}}_0 = \mathbf{h} = 0$ o algoritmo termina . Caso contrário , tomamos $\mathbf{d}_1$ tal que $\mathbf{d}_1^t \bar{\mathbf{r}}_0 \ne 0$. Podemos tomar , por exemplo, $\mathbf{d}_1 = \bar{\mathbf{r}}_0$.

Como $\mathbf{d}_1^t \bar{\mathbf{r}}_0 \ne 0$, span $\{ \mathbf{d}_1 \}$ contém uma direção de descida para q partindo de $\mathbf{x}_0$. Então ,

$$\mathbf{p}_1 = \mathbf{x}_1 - \mathbf{x}_0 \ne 0 \ ,$$

que torna (iii) e (vi) verdade para $k = 1$, e

$$\text{span } \{ \mathbf{p}_1 \} = \text{span } \{ \mathbf{d}_1 \} ,$$

tornando (ii) verdadeira .

Como $\mathbf{x}_1$ minimiza $q(\mathbf{x})$ em S_1 , S_1 não pode conter uma direção de descida partindo de $\mathbf{x}_1$. Então

$$0 = - \nabla q(\mathbf{x}_1)^t \mathbf{p}_1 = \bar{\mathbf{r}}_1^t \mathbf{p}_1 = (\mathbf{h} - \mathbf{H}\mathbf{x}_1)^t \mathbf{p}_1 = (\mathbf{h} - \mathbf{H}\mathbf{p}_1)^t \mathbf{p}_1 ,$$

tornando (iv) e (vii) verdadeiras . Além disso ,

$$\text{span } \{ \mathbf{p}_1 \} = \text{span } \{ \mathbf{d}_1 \} = \mathbf{K}_k(\mathbf{B}, \mathbf{d}_1)$$

para qualquer $\mathbf{B}$, sendo (viii) verdadeira .

Assumiremos (v) e (ix) verdadeiras .

Vamos supor que as afirmações (i) a (vii) sejam verdadeiras para $1 \leq j \leq k-1$, já tendo tomado cuidado com o caso $\bar{\mathbf{r}}_{k-1} = 0$.

Tomamos $\mathbf{d}_k$ tal que $\mathbf{d}_k^t \bar{\mathbf{r}}_{k-1} \neq 0$, por exemplo $\mathbf{d}_k = \bar{\mathbf{r}}_{k-1}$, tendo portanto span $\{ \mathbf{d}_k \}$ uma direção de descida para q , partindo de $\mathbf{x}_{k-1}$. Então

$$0 \neq \mathbf{x}_k - \mathbf{x}_{k-1} = \mathbf{p}_k , \qquad \text{(iii)} .$$

Como $\mathbf{x}_{k-1} \in \text{span } \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_{k-1} \}$, e $\mathbf{x}_k \in \text{span } \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_{k-1}, \mathbf{d}_k \}$ temos que $\mathbf{d}_k \in \text{span } \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_k \}$. Pela hipótese de indução , $\bar{\mathbf{r}}_{k-1}^t \mathbf{p}_j = 0$, para $1 \leq j \leq k-1$, de forma que $\mathbf{d}_k \notin \text{span } \{\mathbf{p}_1, \ \ldots \ , \mathbf{p}_{k-1}\}$. Portanto

$$\text{span } \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_k \} = \text{span } \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_{k-1}, \mathbf{d}_k \} , \qquad \text{(ii)}$$

e

$$\begin{aligned} \dim \{ \text{span } \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_k \} \} &= \dim \{ \text{span } \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_{k-1}, \mathbf{d}_k \} \} \\ &= \dim \{ \text{span } \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_{k-1} \} \} + 1 \\ &= k - 1 + 1 = k , \qquad \text{(vi)} . \end{aligned}$$

Como $\mathbf{x}_k$ minimiza q em S_k ,

$$0 = \nabla q(\mathbf{x}_k)^t \mathbf{p}_i = - \bar{\mathbf{r}}_k^t \mathbf{p}_i \ , \ 1 \le i \le k \ , \qquad \text{(iv)} \ .$$

Note que $\mathbf{x}_k = \sum_{j=1}^{k} \mathbf{p}_j$, tal que $\bar{\mathbf{r}}_k = \mathbf{h} - \sum_{j=1}^{k} \mathbf{H}\mathbf{p}_j$. Então para $i < k$,

$$0 = \mathbf{p}_i^t \bar{\mathbf{r}}_k = \mathbf{p}_i^t \mathbf{h} - \sum_{j=1}^{k} \mathbf{p}_i^t \mathbf{H}\mathbf{p}_j \ .$$

Pela hipótese de indução , isto é

$$0 = \mathbf{p}_i^t \mathbf{h} - \mathbf{p}_i^t \mathbf{H}\mathbf{p}_i - \mathbf{p}_i^t \mathbf{H}\mathbf{p}_k = - \mathbf{p}_i^t \mathbf{H}\mathbf{p}_k \ , \qquad \text{(v)} \ .$$

Ainda mais ,

$$0 = \mathbf{p}_k^t \bar{\mathbf{r}}_k = \mathbf{p}_k^t \mathbf{h} - \sum_{j=1}^{k} \mathbf{p}_k^t \mathbf{H}\mathbf{p}_j = \mathbf{p}_k^t \mathbf{h} - \mathbf{p}_k^t \mathbf{H}\mathbf{p}_k \ , \qquad \text{(vii)} \ .$$

Vamos assumir agora que (viii) é verdade para $1 \le j \le k\text{-}1$, e que $\mathbf{d}_k \in \text{span} \{ \mathbf{d}_1, \mathbf{B}\mathbf{p}_1, \ \ldots \ , \mathbf{B}\mathbf{p}_{k-1} \}$. Então existem escalares α e β_j , tais que

$$\mathbf{d}_k = \alpha \mathbf{d}_1 + \sum_{j=1}^{k-1} \beta_j \mathbf{B}\mathbf{p}_j = \left(\alpha \mathbf{d}_1 + \sum_{j=1}^{k-2} \beta_j \mathbf{B}\mathbf{p}_j \right) + \beta_{k-1} \mathbf{B}\mathbf{p}_{k-1} \ .$$

Pela hipótese de indução , para $1 \le i \le k\text{-}1$, $\mathbf{p}_i \in \mathbf{K}_{k-1}(\mathbf{B}, \mathbf{d}_1)$. Isto , junto com o fato que

$$\text{span} \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_k \} = \text{span} \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_{k-1}, \mathbf{d}_k \}$$

resulta

$$\text{span} \{ \mathbf{p}_1, \ \ldots \ , \mathbf{p}_k \} = \mathbf{K}_k(\mathbf{B}, \mathbf{d}_1) \ , \qquad \text{(viii)} \ .$$

Para (ix) , observe que , por (iv) e (viii) , $\bar{\mathbf{r}}_k$ é ortogonal $\mathbf{K}_k(\mathbf{B}, \mathbf{d}_1)$. Para $i \le k\text{-}1$, $\mathbf{p}_i \in \mathbf{K}_{k-1}(\mathbf{B}, \mathbf{d}_1)$, tal que $\mathbf{B}\mathbf{p}_i \in \mathbf{K}_k(\mathbf{B}, \mathbf{d}_1)$. Então ,

$$\bar{\mathbf{r}}_k^t \mathbf{B}\mathbf{p}_i = 0 \ , \ 1 \le i \le k\text{-}1 \ . \ \blacksquare$$

1.4 - ALGORITMO GENERALIZADO DE DIREÇÕES CONJUGADAS

A cada passo do algoritmo , $\mathbf{x}_k$ é determinado como

$$\mathbf{x}_k = \arg\min q(\mathbf{x}) \ .$$
$$\mathbf{x} \in \text{span}\ \{\ \mathbf{p}_1, \ \ldots\ , \mathbf{p}_{k-1}, \mathbf{d}_k\ \}$$

Seja $\mathbf{P}_k = [\ \mathbf{p}_1, \ \ldots\ , \mathbf{p}_{k-1}, \mathbf{d}_k\ ]$, $\mathbf{x} = \mathbf{P}_k\mathbf{c}$ e ,

$$\bar{q}(\mathbf{c}) = q(\mathbf{P}_k\mathbf{c}) = \frac{1}{2}\ \mathbf{c}^t(\mathbf{P}_k^t\mathbf{H}\mathbf{P}_k)\mathbf{c} - \mathbf{h}^t\mathbf{P}_k\mathbf{c} \ .$$

Então $\nabla\bar{q}(\mathbf{c}) = \mathbf{P}_k^t\mathbf{H}\mathbf{P}_k\mathbf{c} - \mathbf{P}_k^t\mathbf{h}$. Como $\mathbf{H}$ é positiva-definida e $\mathbf{P}_k$ tem posto completo , $\mathbf{P}_k^t\mathbf{H}\mathbf{P}_k$ é positiva-definida . Portanto $\mathbf{x}_k$ pode ser determinado resolvendo-se o sistema $\mathbf{P}_k^t\mathbf{H}\mathbf{P}_k\mathbf{c} = \mathbf{P}_k^t\mathbf{h}$ e fazendo $\mathbf{x}_k = \mathbf{P}_k\mathbf{c}$. De acordo com (v) e (vii) do teorema 1.1 , a solução do sistema acima é

$$\mathbf{c}_i = 1 - \alpha_k\ \frac{\mathbf{p}_i^t\mathbf{H}\mathbf{d}_k}{\mathbf{p}_i^t\mathbf{H}\mathbf{p}_i} \quad , \ i = 1,2,\ldots,k-1$$

$$\mathbf{c}_k = \alpha_k \ ,$$

onde

$$\alpha_k = \frac{\mathbf{d}_k^t\mathbf{h} - \sum_{j=1}^{k-1}\ \mathbf{d}_k^t\mathbf{H}\mathbf{p}_j}{\mathbf{d}_k^t\mathbf{H}\mathbf{d}_k - \sum_{j=1}^{k-1}\ \dfrac{(\mathbf{d}_k^t\mathbf{H}\mathbf{p}_j)^2}{\mathbf{p}_j^t\mathbf{H}\mathbf{p}_j}} \ .$$

O numerador de α_k é igual a $\mathbf{d}_k^t\bar{\mathbf{r}}_{k-1}$. Vamos definir, para $1 \le j \le k-1$,

$$\beta_j^{(k)} = -\ \frac{\mathbf{d}_k^t\mathbf{H}\mathbf{p}_j}{\mathbf{p}_j^t\mathbf{H}\mathbf{p}_j} \ .$$

Então ,

$$\alpha_k = \frac{d_k^t \bar{r}_{k-1}}{d_k^t H\left(d_k + \sum_{j=1}^{k-1} \beta_j^{(k)} p_j \right)}$$

e

$$x_k = P_k c = \sum_{j=1}^{k-1} p_j + \alpha_k \left(d_k + \sum_{j=1}^{k-1} \beta_j^{(k)} p_j \right)$$

$$= x_{k-1} + \alpha_k \left(d_k + \sum_{j=1}^{k-1} \beta_j^{(k)} p_j \right) .$$

Quando d_k é tomado de forma que $d_k^t H p_i = 0$ para $1 \leq i \leq k-2$, a solução tem a forma simplificada

$$\beta_k \equiv \beta_{k-1}^{(k)} = - \frac{d_k^t H p_{k-1}}{p_{k-1}^t H p_{k-1}} \quad , \quad \alpha_k = \frac{d_k^t \bar{r}_{k-1}}{d_k^t H\left(d_k + \beta_k p_{k-1} \right)} \quad ,$$

e $$x_k = x_{k-1} + \alpha_k (d_k + \beta_k p_{k-1}) \quad .$$

Os algoritmos CG , PCG , CR , entre outros , tomam d_k de forma que somente p_{k-1} e d_k são necessários para gerar p_k e x_k . Para CG e CR ,

$$d_k = r_{k-1} = b - A x_{k-1} = b - \sum_{j=1}^{k-1} A p_j \quad ,$$

tal que a matriz B que aparece nas hipóteses do teorema 1.1 é igual a A . Ainda no algoritmo CG ,

$$d_k^t H p_i = r_{k-1}^t A p_i = \bar{r}_{k-1}^t A p_i$$

e para CR

$$\mathbf{d}_k^t \mathbf{H}\mathbf{p}_i = \mathbf{r}_{k-1}^t \mathbf{A}^t \mathbf{A}\mathbf{p}_i = \bar{\mathbf{r}}_{k-1}^t \mathbf{A}\mathbf{p}_i ,$$

desde que **A** é simétrica . Então , para ambos ,

$$\mathbf{d}_k^t \mathbf{H}\mathbf{p}_i = 0 \text{ , para } 1 \leq i \leq k-2 ,$$

pela parte (ix) do teorema 1.1 . Para CGNE ,

$$\mathbf{d}_k = \bar{\mathbf{r}}_{k-1} = \mathbf{h} - \mathbf{H}\mathbf{x}_{k-1} = \mathbf{h} - \sum_{j=1}^{k-1} \mathbf{H}\mathbf{p}_j ,$$

tal que **B** é igual a **H** , e

$$\mathbf{d}_k^t \mathbf{H}\mathbf{p}_i = \bar{\mathbf{r}}_{k-1}^t \mathbf{H}\mathbf{p}_i = 0 \text{ , para } 1 \leq i \leq k-2 .$$

Para PCG ,

$$\mathbf{d}_k = \mathbf{M}^{-1}\mathbf{r}_{k-1} ,$$

e **B** é $\mathbf{M}^{-1}\mathbf{A}$. Como **M** é simétrica ,

$$\mathbf{d}_k^t \mathbf{H}\mathbf{p}_i = (\mathbf{r}_{k-1}^t \mathbf{M}^{-1})\mathbf{A}\mathbf{p}_i = \bar{\mathbf{r}}_{k-1}^t (\mathbf{M}^{-1}\mathbf{A})\mathbf{p}_i = 0 \quad \text{para } 1 \leq i \leq k-2 .$$

Para obtermos a convergência do método GCD precisamos que $\mathbf{d}_k^t \bar{\mathbf{r}}_{k-1} \neq 0$ sempre que $\bar{\mathbf{r}}_{k-1} \neq 0$. Mas esta condição é satisfeita para a maioria dos métodos , como pode ser conferido na *tabela 1.1* .

CAPÍTULO 2

GRADIENTES CONJUGADOS QUADRÁTICO (CGS)

2.1 - UMA FORMA POLINOMIAL EQUIVALENTE AO MÉTODO CG

Baseados no algoritmo GCD do capítulo anterior , apresentaremos dois métodos da família de direções conjugadas para sistemas lineares e que não exigem que a matriz seja simétrica ou positiva-definida . Trata-se do método dos Gradientes Bi-Conjugados (BI-CG) e do método dos Gradientes Conjugados Quadrático (CGS) que são variantes polinomiais do método dos Gradientes Conjugados . Vamos começar relembrando algumas propriedades e o algoritmo CG , já vistos no capítulo anterior .

Seja $\mathbf{A} \in \mathbb{R}^{nxn}$, simétrica e positiva-definida , e consideremos o sistema linear

$$\mathbf{Ax} = \mathbf{b} \quad , \quad \mathbf{x},\mathbf{b} \in \mathbb{R}^n \ .$$

Se tomarmos uma aproximação inicial $\mathbf{x}_0$ para a solução $\mathbf{x}^*$ do sistema linear , podemos escrever algoritmo CG da seguinte forma :

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

$\mathbf{p}_{-1} = 0$;

$\rho_{-1} = 1$;

$k = 0$;

Enquanto $\| \mathbf{r}_k \| >$ tolerância

$$\rho_k = \mathbf{r}_k^t \mathbf{r}_k \ ; \quad \beta_k = \frac{\rho_k}{\rho_{k-1}} \ ;$$

$$p_k = r_k + \beta_k p_{k-1} \ ;$$

$$\sigma_k = p_k^t A p_k \ ; \quad \alpha_k = \frac{\rho_k}{\sigma_k} \ ; \tag{2.1}$$

$$r_{k+1} = r_k - \alpha_k A p_k \ ;$$

$$x_{k+1} = x_k + \alpha_k p_k \ ;$$

$$k = k + 1 \ ;$$

fim ;

Já foi provado que este algoritmo converge , teoricamente , em n passos . Além disso , os resíduos são mutuamente ortogonais e as direções de busca são mutuamente A-conjugadas , ou seja ,

$$r_k^t r_m = \rho_k \delta_{km} \ , \qquad p_k^t A p_m = \sigma_k \delta_{km}$$

onde δ_{km} é o Delta de Kronecker .

Da estrutura do algoritmo (2.1) , notamos que r_k e p_k podem ser escritos como segue :

$$r_k = \varphi_k(A) r_0 \ , \qquad p_k = \psi_k(A) r_0$$

onde φ_k e ψ_k são polinômios de grau menor ou igual a k . Este fato pode ser provado por indução , se definirmos $\varphi_0(\tau) \equiv 1$, $\psi_{-1}(\tau) \equiv 0$, que é consistente com o algoritmo . Substituindo

$$r_k = \varphi_k(A) r_0 \ , \qquad p_{k-1} = \psi_{k-1}(A) r_0 \tag{2.2}$$

para algum $k \geq 0$, temos

$$p_k = \varphi_k(A) r_0 + \beta_k \psi_{k-1}(A) r_0 = \psi_k(A) r_0$$

com

$$\psi_k(\tau) \equiv \varphi_k(\tau) + \beta_k \psi_{k-1}(\tau) \ , \quad \forall \ \tau \in \mathbb{R} \ .$$

Substituindo (2.2) no cálculo de r_{k+1} , encontramos

$$r_{k+1} = \varphi_k(A) r_0 - \alpha_k A \psi_k(A) r_0 = \varphi_{k+1}(A) r_0$$

com

$$\varphi_{k+1}(\tau) \equiv \varphi_k(\tau) - \alpha_k \tau \psi_k(\tau) \ , \quad \forall \ \tau \in \mathbb{R} \ .$$

Portanto , a validade da hipótese (2.2) para k , implica na validade para k+1 . ■

Desta forma , (2.1) pode ser reinterpretado como um algoritmo para gerar uma sequência de polinômios "ortogonais" em relação à forma bilinear que definiremos a seguir .

Denotaremos o conjunto dos polinômios reais de grau menor que N por Π^N, e definiremos a seguinte forma bilinear e simétrica $a(.,.)$ em Π^N

$$a(.,.) : \Pi^N \times \Pi^N \longrightarrow \mathbb{R}$$

$$a(\varphi,\psi) = [\varphi(A) r_0]^t \psi(A) r_0 \ .$$

Obviamente temos $a(\varphi,\varphi) \geq 0$, para todo $\varphi \in \Pi^N$. Como A é simétrica , podemos também escrever

$$a(\varphi,\psi) = r_0^t \varphi(A) \psi(A) r_0 \ .$$

Ainda mais ,

$$a(\varphi\chi,\psi) = a(\varphi,\chi\psi)$$

para quaisquer polinômios φ , χ , ψ .

Portanto , esta forma bilinear só não é um produto interno no espaço Π^N porque a positividade não está garantida para todos os polinômios , desde que para algum polinômio não-nulo φ , $\varphi(A)r_0$ pode ser nulo , e então $a(\varphi,\varphi) = 0$.

Com base nas observações feitas até aqui , colocaremos o algoritmo (2.1) numa forma polinomial equivalente :

$\varphi_0 \equiv 1$;

$\psi_{-1} \equiv 0$;

$\rho_{-1} \equiv 1$;

Enquanto $\| r_k \| >$ tolerância

$$\rho_k = a(\varphi_k,\varphi_k) \; ; \quad \beta_k = \frac{\rho_k}{\rho_{k-1}} \; ;$$

$$\psi_k = \varphi_k + \beta_k \psi_{k-1} \; ;$$

$$\sigma_k = a(\psi_k,\vartheta\psi_k) \; ; \quad \alpha_k = \frac{\rho_k}{\sigma_k} \; ; \qquad (2.3)$$

$$\varphi_{k+1} = \varphi_k - \alpha_k \vartheta\psi_k \; ;$$

$k = k + 1$;

fim ;

onde

$$\vartheta(\tau) \equiv \tau \; . \qquad (2.4)$$

A ortogonalidade dos r_k e a A-ortogonalidade dos p_k permitem-nos entender (2.3) como um algoritmo de "ortogonalização" de polinômios , com respeito aos "produtos internos" $a(.,.)$ e $a(.,\vartheta.)$.

Teorema 2.1 : *Seja* $a[.,.]$ *uma forma bilinear simétrica definida em* Π^n , *satisfazendo*

$$a[\varphi\chi,\psi] = a[\varphi,\chi\psi] \quad , \quad \forall\ \varphi,\chi,\psi \in \Pi^n \ . \tag{2.5}$$

Sejam as sequências de polinômios φ_k e ψ_k *construídas de acordo com o algoritmo* (2.3) , *mas usando a forma bilinear* $a[.,.]$ *ao invés de* $a(.,.)$. *Então, se o algoritmo não for interrompido devido a uma divisão por zero , estes polinômios satisfazem*

$$a[\varphi_k,\varphi_m] = \rho_k \delta_{km} \ , \qquad a[\psi_k,\vartheta\psi_m] = \sigma_k \delta_{km} \qquad \forall\ k,m \tag{2.6}$$

com ϑ definido por (2.4) .

Demonstração : Mostraremos por indução o seguinte :

$$a[\varphi_k,\psi_j] = 0 \ , \quad a[\psi_{k-1},\vartheta\psi_j] = \sigma_{k-1}\delta_{k-1,j} \quad \forall\ k \geq 0 \ , \ 1 \leq j \leq k-1 \tag{2.7}$$

com $\sigma_{-1} = 0$. Se $k = 0$, (2.7) já é verdade porque $\psi_{-1} \equiv 0$. Vamos supor (2.7) verdadeiro para $k \leq m$, e seja $j < m$. De acordo com (2.3) temos

$$a[\psi_m,\vartheta\psi_j] = a[\varphi_m,\vartheta\psi_j] + \beta_m a[\psi_{m-1},\vartheta\psi_j] \ .$$

O segundo termo do lado direito se anula para $j < m-1$, por hipótese . No primeiro termo substituímos $\psi_j = (\varphi_j - \varphi_{j+1}) / \alpha_j \vartheta$, dando

$$a[\psi_m,\vartheta\psi_j] = \frac{a[\varphi_m,\varphi_j] - a[\varphi_m,\varphi_{j+1}]}{\alpha_j} + \beta_m \sigma_{m-1} \delta_{m-1,j} \qquad \forall\ j \leq m-1 \ .$$

Por hipótese , isto se anula para $j < m-1$, enquanto para $j = m-1$ temos

$$a[\psi_m,\vartheta\psi_{m-1}] = -\frac{\rho_m}{\alpha_{m-1}} + \beta_m \sigma_{m-1} = 0 \ ,$$

o qual prova a segunda parte de (2.7) para k = m+1 .

Procedendo da mesma forma para a primeira parte , escreveremos

$$a[\varphi_{m+1},\psi_j] = a[\varphi_m,\psi_j] - \alpha_m a[\psi_m,\theta\psi_j] \quad \forall\ j \le m\ .$$

Se $j \le m-1$ o lado direito se anula , por hipótese . Usando o algoritmo e tomando $j = m$ chegamos a

$$a[\varphi_{m+1},\psi_m] = a[\varphi_m,\varphi_m + \beta_m\psi_{m-1}] - \alpha_m a[\psi_m,\theta\psi_m] = \rho_m - \alpha_m\sigma_m = 0\ ,$$

que prova a primeira parte de (2.7) . Portanto (2.7) é válido para todo $k \in \mathbb{N}$.

Finalmente , escrevendo $\varphi_j = \psi_j - \beta_j\psi_{j-1}$, para todo $j \ge 0$ temos

$$a[\varphi_k,\varphi_j] = a[\varphi_k,\psi_j] - \beta_j a[\varphi_k,\psi_{j-1}] = 0 \quad ,\quad \forall\ j < k\ ,$$

que junto com a segunda parte de (2.7) prova o teorema . ■

Estamos prontos para generalizar o algoritmo dos Gradientes Conjugados para matrizes não-simétricas , definindo a forma bilinear a[.,.] convenientemente .

Sejam $\mathbf{Ax} = \mathbf{b}$ um sistema linear não-singular e $\mathbf{x}_0$ uma aproximação inicial para $\mathbf{x}^*$, com correspondente resíduo $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, e $\bar{\mathbf{r}}_0$ um dado vetor de $\mathbb{R}^n$. Definiremos a[.,.] por

$$a[\varphi,\psi] = \bar{\mathbf{r}}_0^t\varphi(\mathbf{A})\psi(\mathbf{A})\mathbf{r}_0 = (\varphi(\mathbf{A}^t)\bar{\mathbf{r}}_0)^t\psi(\mathbf{A})\mathbf{r}_0 \qquad (2.8)$$

e os vetores $\mathbf{r}_k$, $\bar{\mathbf{r}}_k$, $\mathbf{p}_k$ e $\bar{\mathbf{p}}_k$ por

$$\mathbf{p}_{-1} = \bar{\mathbf{p}}_{-1} = 0\ ,$$

$$\mathbf{r}_k = \varphi_k(\mathbf{A})\mathbf{r}_0\ ,\quad \bar{\mathbf{r}}_k = \varphi_k(\mathbf{A}^t)\bar{\mathbf{r}}_0\ , \qquad (2.9)$$

$$\mathbf{p}_k = \psi_k(\mathbf{A})\mathbf{r}_0\ ,\quad \bar{\mathbf{p}}_k = \psi_k(\mathbf{A}^t)\bar{\mathbf{r}}_0$$

com φ_k e ψ_k de acordo com (2.3) .

Estes vetores podem ser produzidos pelo seguinte algoritmo chamado Gradientes Bi-Conjugados (BI-CG) :

$$r_0 = b - Ax_0 \ ;$$

$$p_{-1} = \bar{p}_{-1} = 0 \ ;$$

$$\rho_{-1} = 1 \ ;$$

$$k = 0 \ ;$$

Enquanto $\| r_k \| >$ tolerância

$$\rho_k = \bar{r}_k^t r_k \ ; \quad \beta_k = \frac{\rho_k}{\rho_{k-1}} \ ;$$

$$p_k = r_k + \beta_k p_{k-1} \ ; \quad \bar{p}_k = \bar{r}_k + \beta_k \bar{p}_{k-1} \ ;$$

$$\sigma_k = \bar{p}_k^t A p_k \ ; \quad \alpha_k = \frac{\rho_k}{\sigma_k} \ ; \qquad (2.10)$$

$$r_{k+1} = r_k - \alpha_k A p_k \ ; \quad \bar{r}_{k+1} = \bar{r}_k - \alpha_k A^t \bar{p}_k \ ;$$

$$x_{k+1} = x_k + \alpha_k p_k \ ;$$

$$k = k + 1 \ ;$$

fim ;

De novo , se não ocorrer divisão por zero , r_k converge para zero implicando na convergência de x_k para a solução x^* . Na prática , o vetor $\bar{r}_0$ é tomado igual a r_0 . Então , se A não está muito "longe" de ser *spd* , as formas bilineares $a[.,.]$ e $a[.,\theta.]$ serão positivas semi-definidas , e o algoritmo convergirá da mesma forma e pelos mesmos argumentos do método CG .

2.2 - GRADIENTES CONJUGADOS QUADRÁTICO (CGS)

Assuma que o algoritmo BI-CG (2.10) esteja convergindo . Então $r_k \to 0$ quando k cresce , que pode ser interpretado como a seguinte propriedade das matrizes $\varphi_k(A)$: como $r_k = \varphi_k(A)r_0$, estas matrizes comportam-se como operadores contractantes . Então podemos esperar que $\varphi_k(A^t)$ sejam contractantes também , com a conseqüência de que $\bar{r}_k \to 0$. Embora a convergência dos "quase-resíduos" $\bar{r}_k$ não seja usada , precisamos computá-los pois estes estão envolvidos no cálculo de ρ_k e σ_k . Portanto , o trabalho computacional exigido pelo método BI-CG é o dobro do trabalho computacional do método CG .

Para contornar estes problemas vamos retornar ao algoritmo polinomial (2.3) com $\rho_k = a[\varphi_k,\varphi_k]$ e $\sigma_k = a[\psi_k,\vartheta\psi_k]$ e alterar a forma de calcular ρ_k e σ_k para evitar a construção dos vetores $\bar{r}_k$ e $\bar{p}_k$. Como a[.,.] tem a propriedade (2.5) , segue que ρ_k e σ_k podem ser calculados da seguinte forma :

$$\rho_k = a[\varphi_0,\varphi_k^2] \quad , \quad \sigma_k = a[\varphi_0,\vartheta\psi_k] \quad .$$

Isto sugere a possibilidade de um algoritmo que gere polinômios φ_k^2 e ψ_k^2 ao invés de φ_k e ψ_k . Conseqüentemente , estaremos tentando interpretar os vetores $\varphi_k^2(A)r_0$ como os resíduos da estimativa x_k de x^* . Desta forma , o cálculo de ρ_k pode ser feito por $\rho_k = \bar{r}_0^t r_k$ com $r_k = \varphi_k^2(A)r_0$. Deixamos , portanto , de precisar de um processo para o cálculo de $\bar{r}_k$ e $\bar{p}_k$, que é uma importante simplificação . Ainda mais , o efeito contractante das matrizes $\varphi_k(A)$ é duplicado e isto pode acelerar a convergência do algoritmo .

Para chegar ao algoritmo CGS tomaremos os quadrados de ψ_k e φ_{k+1} em (2.3) :

$$\psi_k^2 = \varphi_k^2 + 2\beta_k\varphi_k\psi_{k-1} + \beta_k^2\psi_{k-1}^2 \quad ,$$

$$\varphi_{k+1}^2 = \varphi_k^2 - 2\alpha_k\vartheta\varphi_k\psi_k + \alpha_k^2\vartheta^2\psi_k^2$$

com ϑ definido em (2.4) . Definiremos para $k \geq 0$:

$$\Phi_k = \varphi_k^2 \quad , \quad \theta_k = \varphi_k \psi_{k-1} \quad , \quad \Psi_{k-1} = \psi_{k-1}^2 \quad . \tag{2.11}$$

Desta forma , o seguinte algoritmo tem as propriedades que desejamos :

$\theta_0 \equiv 0$;

$\Psi_{-1} \equiv 0$;

$\Phi_0 \equiv 1$;

$\rho_{-1} \equiv 1$;

$k = 0$;

Enquanto $\| r_k \| >$ tolerância

$$\begin{aligned}
&\rho_k = a[1, \Phi_k] \; ; \quad \beta_k = \frac{\rho_k}{\rho_{k-1}} \; ; \\
&\Upsilon_k = \Phi_k + \beta_k \theta_k \; ; \\
&\Psi_k = \Upsilon_k + \beta_k (\theta_k + \beta_k \Psi_{k-1}) \; ; \\
&\\
&\sigma_k = a[1, \vartheta \Psi_k] \; ; \quad \alpha_k = \frac{\rho_k}{\sigma_k} \; ; \\
&\theta_{k+1} = \Upsilon_k - \alpha_k \vartheta \Psi_k \; ; \\
&\Phi_{k+1} = \Phi_k - \alpha_k \vartheta (\Upsilon_k + \theta_{k+1}) \; ; \\
&k = k + 1 \; ;
\end{aligned} \tag{2.12}$$

fim ;

Vamos obter uma forma vetorial deste algoritmo substituindo a matriz $\mathbf{A}$ nos polinômios e fazendo as partes agirem como operadores lineares no resíduo inicial r_0 . Definindo $r_k = \Phi_k(A) r_0$, $q_k = \theta_k(A) r_0$ e $p_k = \Psi_k(A) r_0$ temos o método CGS cujo algoritmo encontramos a seguir.

Seja x_0 uma aproximação inicial para a solução x^* do sistema linear $Ax = b$, não-singular , e $\bar{r}_0$ um vetor apropriado . Então o algoritmo CGS pode ser escrito como :

$r_0 = b - Ax_0$;

$p_{-1} = 0$;

$q_0 = 0$;

$\rho_{-1} = 1$;

$k = 0$;

Enquanto $\| r_k \|$ > tolerância

$$\rho_k = \bar{r}_0^t r_k \; ; \quad \beta_k = \frac{\rho_k}{\rho_{k-1}} \; ;$$

$$u_k = r_k + \beta_k q_k \; ;$$

$$p_k = u_k + \beta_k (q_k + \beta_k p_{k-1}) \; ;$$

$$v_k = Ap_k \; ;$$

$$\sigma_k = \bar{r}_0^t v_k \; ; \quad \alpha_k = \frac{\rho_k}{\sigma_k} \; ; \qquad (2.13)$$

$$q_{k+1} = u_k - \alpha_k v_k \; ;$$

$$r_{k+1} = r_k - \alpha_k A(u_k + q_{k+1}) \; ;$$

$$x_{k+1} = x_k + \alpha_k (u_k + q_{k+1}) \; ;$$

$k = k + 1$;

fim ;

Desde que $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ e $\mathbf{r}_{k+1} - \mathbf{r}_k = \mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k+1})$ para todo k, segue que $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$. Portanto os resíduos satisfazem à relação

$$\mathbf{r}_k = \varphi_k^2(\mathbf{A})\mathbf{r}_0$$

como esperávamos.

CAPÍTULO 3

PRÉ-CONDICIONADORES

3.1 - INTRODUÇÃO

A maior parte dos métodos apresentados até agora tem sua taxa de convergência relacionada com o número de condição espectral da matriz do sistema . Quanto menor o número de condição , mais rápida a convergência . Por isso , é comum o uso de técnicas de pré-condicionamento para diminuir o número de condição da matriz , e torná-los mais eficientes .

Daremos agora uma descrição geral de uma técnica de pré-condicionamento .

Sejam $\mathbf{P}_L$ e $\mathbf{P}_R$ dois operadores lineares sobre $\mathbb{R}^n$. Na prática , as técnicas de pré-condicionamento geralmente acrescentam a resolução de um ou mais sistemas lineares a cada iteração do método utilizado . As matrizes destes sistemas estão diretamente relacionadas com os operadores $\mathbf{P}_L$ e $\mathbf{P}_R$. Por isso , $\mathbf{P}_L$ e $\mathbf{P}_R$ são escolhidos como matrizes esparsas ou triangulares para facilitar a resolução destes sistemas lineares adicionais .

Pré-multiplicando o sistema linear $\mathbf{Ax} = \mathbf{b}$ por $\mathbf{P}_L$ e definindo $\tilde{\mathbf{x}}$ por $\mathbf{P}_R\tilde{\mathbf{x}} = \mathbf{x}$, temos o seguinte sistema linear modificado

$$\mathbf{B}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{3.1}$$

onde $\mathbf{B} = \mathbf{P}_L\mathbf{A}\,\mathbf{P}_R$ e $\tilde{\mathbf{b}} = \mathbf{P}_L\mathbf{b}$. Os operadores $\mathbf{P}_L$ e $\mathbf{P}_R$ são tomados de forma que $\mathbf{B}$ tenha propriedades espectrais mais adequadas e o produto $\mathbf{P}_R\mathbf{P}_L$ pode ser considerado como uma aproximação da inversa da matriz $\mathbf{A}$.

3.2 - GRADIENTES CONJUGADOS COM PRÉ-CONDICIONAMENTO (PCG)

Teorema 3.1 : *Se uma sequência* $\{ \mathbf{x}_k \}$ *é produzida pelo Método dos Gradientes Conjugados aplicado ao sistema linear* $\mathbf{Ax} = \mathbf{b}$, *com* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *simétrica e positiva-definida , então*

$$\| x^* - x_k \|_A \leq \| x^* - x_0 \|_A \left(\frac{k - 1}{k + 1} \right)^k ,$$

onde $\| w \|_A = \sqrt{w^t A w}$ *e* $k = \sqrt{cond_2(A)}$.

Demonstração : Referência [12] . □

Observando o teorema da página anterior , podemos concluir que o método CG converge rapidamente , na norma energia ($\| . \|_A$) , se $cond_2(A) \cong 1$, isto é , se os autovalores da matriz A estiverem próximos uns dos outros . Caso contrário , as curvas de nível da forma quadrática $q(x)$, estudada no capítulo 1 , serão hiperelipsóides muito alongados . Como minimizar $q(x)$ corresponde a descer ao ponto mais baixo do vale (figura 3.1) , as direções de busca p_k podem ser geradas de forma a atravessá-lo em zig-zag antes de descê-lo e isto tende a retardar a convergência do método .

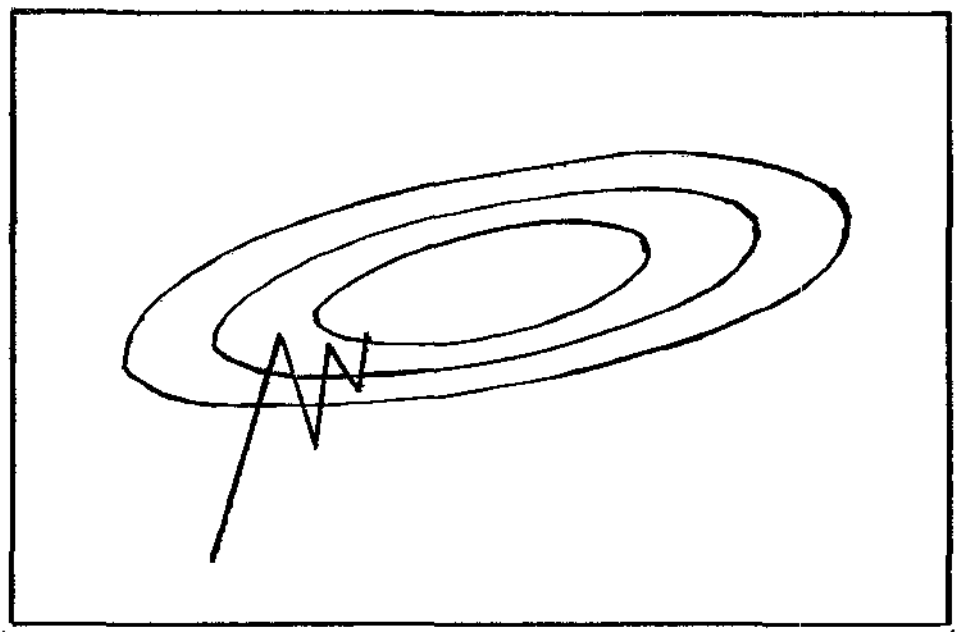

Figura 3.1

Seja o sistema linear $\mathbf{Ax} = \mathbf{b}$ com $\mathbf{A}$ simétrica e positiva-definida . Com o objetivo de obtermos uma redução do número de condição da matriz do sistema , aplicaremos o método CG ao sistema transformado $\mathbf{B}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$, onde $\mathbf{B} = \mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-1}$, $\tilde{\mathbf{x}} = \mathbf{C}\mathbf{x}$ e $\tilde{\mathbf{b}} = \mathbf{C}^{-1}\mathbf{b}$, com $\mathbf{C}$ simétrica e positiva-definida (note que $\mathbf{P}_L = \mathbf{P}_R = \mathbf{C}^{-1}$) :

$\tilde{r}_0 = C^{-1}(b - Ax_0)$;

$p_{-1} = 0$;

$\rho_{-1} = 1$;

$k = 0$;
Enquanto $\| \tilde{r}_k \| >$ tolerância

$$\rho_k = \tilde{r}_k^t \tilde{r}_k \; ; \quad \beta_k = \frac{\rho_k}{\rho_{k-1}} \; ;$$

$$\tilde{p}_k = \tilde{r}_k + \beta_k \tilde{p}_{k-1} \; ;$$

$$\sigma_k = \tilde{p}_k^t C^{-1} A C^{-1} \tilde{p}_k \; ; \quad \alpha_k = \frac{\rho_k}{\sigma_k} \; ; \qquad (3.2)$$

$$\tilde{r}_{k+1} = \tilde{r}_k - \alpha_k C^{-1} A C^{-1} \tilde{p}_k \; ;$$

$$\tilde{x}_{k+1} = \tilde{x}_k + \alpha_k \tilde{p}_k \; ;$$

$k = k + 1$;

fim ;

Fazendo $\tilde{p}_k = Cp_k$, $\tilde{x}_k = Cx_k$, $\tilde{r}_k = C^{-1}r_k$ em (3.2) , obtemos o algoritmo CG envolvendo apenas os vetores associados ao sistema linear original $\mathbf{Ax} = \mathbf{b}$:

$C^{-1}r_0 = C^{-1}(b - Ax_0)$;

$p_{-1} = 0$;

$\rho_{-1} = 1$;

$k = 0$;

Enquanto $\| C^{-1}r_k \| >$ tolerância

$$\rho_k = (C^{-1}r_k)^t C^{-1}r_k \; ; \quad \beta_k = \frac{\rho_k}{\rho_{k-1}} \; ;$$

$$Cp_k = C^{-1}r_k + \beta_k Cp_{k-1} \; ;$$

$$\sigma_k = (Cp_k)^t (C^{-1}AC^{-1})Cp_k \; ; \quad \alpha_k = \frac{\rho_k}{\sigma_k} \; ; \qquad (3.3)$$

$$C^{-1}r_{k+1} = C^{-1}r_k - \alpha_k (C^{-1}AC^{-1})Cp_k \; ;$$

$$Cx_{k+1} = Cx_k + \alpha_k Cp_k \; ;$$

$k = k + 1$;

fim ;

Definindo o pré-condicionador M como sendo $M = C^2$, também simétrica e positiva-definida , e z_k solução de $Mz_k = r_k$, chegamos à seguinte simplificação do algoritmo (3.3) :

$r_0 = b - Ax_0$;

$p_{-1} = 0$;

$\rho_{-1} = 1$;

$k = 0$;

Enquanto $\| r_k \| >$ tolerância

Resolver $Mz_k = r_k$;

$$\rho_k = r_k^t z_k \quad ; \quad \beta_k = \frac{\rho_k}{\rho_{k-1}} ;$$

$$p_k = z_k + \beta_k p_{k-1} ;$$

$$\sigma_k = p_k^t A p_k \quad ; \quad \alpha_k = \frac{\rho_k}{\sigma_k} ; \qquad (3.4)$$

$$r_{k+1} = r_k - \alpha_k A p_k ;$$

$$x_{k+1} = x_k + \alpha_k p_k ;$$

$$k = k + 1 ;$$

fim ;

Note que a cada iteração do algoritmo PCG (3.4) , é necessário a resolução de um sistema linear do tipo **Mz** = **r** . Portanto , a matriz **M** deve ser escolhida de forma que a resolução deste sistema não cause um aumento considerável na quantidade de memória usada para a armazenagem das variáveis , nem aumente consideravelmente a quantidade de operações em ponto flutuante em relação ao algoritmo CG sem pré-condicionamento . A forma de tomar a matriz **C** de modo que a matriz **M** obedeça as observações acima será vista na seção 3.4 .

3.3 - CGS COM PRÉ-CONDICIONAMENTO (PCGS)

Como já vimos no capítulo anterior , se a forma bilinear *a*[.,.] definida em (2.8) não estiver "longe" de ser um produto interno , isto é , de ser positiva-definida , a convergência do método CGS será bastante similar ao caso *spd* do método CG . Por este motivo é que aplicaremos técnicas de pré-condicionamento ao algoritmo CGS buscando uma melhor distribuição dos autovalores da matriz do sistema linear que nos propomos a resolver .

Existem algumas possibilidades de implementação de pré-condicionadores no algoritmo CGS . A variante usada em nossos experimentos numéricos pode ser escrita da seguinte forma :

$$r_0 = P_L(b - Ax_0) ;$$

$$p_{-1} = 0 ;$$

$$q_0 = 0 ;$$

$$\rho_{-1} = 1 ;$$

Enquanto $\| r_k \| >$ tolerância

$$\rho_k = \bar{r}_0^t r_k \quad ; \quad \beta_k = \frac{\rho_k}{\rho_{k-1}} ;$$

$$u_k = r_k + \beta_k q_k ;$$

$$p_k = u_k + \beta_k (q_k + \beta_k p_{k-1}) ;$$

$$v_k = P_L A P_R p_k ;$$

$$\sigma_k = \bar{r}_0^t v_k \quad ; \quad \alpha_k = \frac{\rho_k}{\sigma_k} ; \qquad (3.5)$$

$$q_{k+1} = u_k - \alpha_k v_k ;$$

$$r_{k+1} = r_k - \alpha_k P_L A P_R (u_k + q_{k+1}) ;$$

$$x_{k+1} = x_k + \alpha_k P_R (u_k + q_{k+1}) ;$$

$$k = k + 1 ;$$

fim ;

3.4 - FATORAÇÕES INCOMPLETAS

Os algoritmos CG e CGS com pré-condicionamento já foram vistos nas duas seções anteriores . Vamos , então , mostrar de que forma podemos escolher os operadores lineares $\mathbf{P}_L$ e $\mathbf{P}_R$ de modo que os algoritmos PCG e PCGS possam ser efetivamente implementados e testados .

Uma classe importante de métodos de pré-condicionamento é baseada em fatorações incompletas da matriz **A** . Uma fatoração incompleta consiste basicamente em fatorar a matriz **A** escolhendo um método de acordo com suas propriedades - LU ou Cholesky , por exemplo - e de forma que as matrizes triangulares resultantes da fatoração mantenham a mesma esparsidade da matriz **A** . Para isso , os elementos não nulos que aparecerem durante a fatoração e que modificarem a estrutura de esparsidade pré-definida devem ser desconsiderados . Desse modo, estaremos calculando uma aproximação para a fatoração da matriz **A** .

No caso do método iterativo não exigir que a matriz **A** seja simétrica e positiva-definida , CGS por exemplo , poderemos utilizar uma fatoração LU incompleta de **A** e usá-la como pré-condicionador deste método .

$$\mathbf{LU} = \mathbf{A} + \mathbf{E} \quad ,$$

onde **E** é a matriz que contém o erro da fatoração e também é esparsa . A fatoração incompleta LU pode ser obtida através de uma simples alteração do algoritmo de eliminação de Gauss . Se $NZ(\mathbf{A})$ denota o conjunto dos pares [i,j] para os quais o elemento a_{ij} da matriz **A** é não-nulo , a fatoração incompleta de Gauss pode ser escrita como :

```
para k = 1..n
    para i = k+1..n
        a_ik = a_ik / a_kk ;
        para j = k+1..n
            se [i,j] ∈ NZ(A)
                corr = - a_ik a_kj ;
                a_ij = a_ij + corr ;
            fim
        fim
    fim
fim ;
```
(3.6)

No caso em que a matriz A estiver armazenada de forma compactada , ou seja , numa estrutura que só guarde os elementos não-nulos , os inteiros *i* e *j* não assumem todos os valores de seus "laços". Mas programar isto é a parte mais difícil deste algoritmo . Nós denotamos o pré-condicionamento baseado no algoritmo anterior por ILU . Desta forma podemos definir os seguintes pré-condicionadores :

$$\mathbf{P_L = L^{-1}} \ , \quad \mathbf{P_R = U^{-1}}$$

$$\mathbf{P_L = U^{-1}L^{-1}} \ , \quad \mathbf{P_R = I}$$

$$\mathbf{P_L = I} \ , \quad \mathbf{P_R = U^{-1}L^{-1}} \ .$$

Todas estas alternativas originam matrizes **B** similares , isto é , com o mesmo conjunto de autovalores . A escolha de uma delas depende das facilidades de implementação .

No caso do método CG , podemos usar uma fatoração incompleta de Cholesky da matriz **A** . A idéia é calcular uma matriz triangular inferior **H** com uma estrutura de esparsidade semelhante a da parte triangular inferior de **A** e que seja uma aproximação do fator "exato" **G** de Cholesky . A partir daí o pré-condicionador **M** pode ser construído pelo produto $\mathbf{M = HH^t}$.

Se tomarmos a decomposição QR da matriz **C** de (3.2) :

$$\mathbf{C} = \mathbf{Q}\mathbf{H}^t \quad ,$$

com **Q** ortogonal , então

$$\mathbf{M} = \mathbf{C}^2 = \mathbf{C}^t\mathbf{C} = \mathbf{H}\mathbf{Q}^t\mathbf{Q}\mathbf{H}^t = \mathbf{H}\mathbf{H}^t$$

logo

$$\mathbf{B} = \mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-1} = \mathbf{C}^{-t}\mathbf{A}\mathbf{C}^{-1} = (\mathbf{H}\mathbf{Q}^t)^{-1}\mathbf{A}(\mathbf{Q}\mathbf{H}^t)^{-1} = \mathbf{Q}(\mathbf{H}^{-1}\mathbf{G}\mathbf{G}^t\mathbf{H}^{-t})\mathbf{Q}^t \cong \mathbf{I} \ .$$

Concluímos , portanto , que quanto melhor **H** aproxima **G** , menor o número de condição espectral da matriz **B** .

3.5 - EXEMPLOS NUMÉRICOS : O PROBLEMA DE DIFUSÃO-CONVECÇÃO

Considere o seguinte problema num domínio $\Omega \subset \mathbb{R}^2$:

$$\begin{aligned} -\alpha(x,y)\Delta u + \beta(x,y).\nabla u &= f(x,y) \quad \text{em } \Omega \\ u(x,y) &= 0 \quad \text{em } \partial\Omega \ , \end{aligned}$$

com $\alpha(x,y) > 0$. Este é um exemplo de um problema estacionário de difusão-convecção . O termo do Laplaciano corresponde a um fenômeno de difusão com coeficiente difusivo $\alpha(x,y)$ e as derivadas de primeira ordem correspondem ao fenômeno de convecção na direção do vetor $\beta = (\beta_1(x,y),\beta_2(x,y))^t$.

Testaremos alguns dos métodos vistos até agora na resolução do problema acima . Escolhemos o domínio $\Omega = [0,1] \times [0,1]$, $\alpha(x,y) = 2$, $f(x,y) = 1$ e $\beta = (12,12)^t$ de forma que o problema fosse bem comportado . A discretização foi feita através do método de Diferenças Finitas com uma malha regular contendo **npt** pontos em cada uma das direções (**x** e **y**) dando origem a um sistema linear esparso com dimensão **ndim** . Para testar a convergência dos métodos , usamos **eps** = 10^{-5} como tolerância .

Nas tabelas a seguir (3.1 e 3.2) encontramos os resultados obtidos . Na tabela 3.1 encontramos o número de iterações necessárias para os seguintes métodos :

CGNE : Método dos Gradientes Conjugados - Equações Normais ;

PCGNE : Método CGNE com pré-condicionamento pela diagonal da matriz do sistema linear ;

CGS : Método dos Gradientes Conjugados Quadrático ;

PCGS : Método CGS com pré-condicionamento tridiagonal .

npt	ndim	CGNE	PCGNE	CGS	PCGS
17	225	137	138	38	20
25	529	307	305	58	32
29	729	413	413	72	32
37	1225	677	675	94	44
41	1521	835	834	103	48

Tabela 3.1

Olhando os resultados do método CGNE podemos notar que a utilização das equações normais para a resolução de sistemas lineares que não têm simetria torna os resultados bastante pobres em relação aos métodos que não fazem esta exigência da matriz (CGS e PCGS) . Veja também como a estratégia de pré-condicionamento melhorou os resultados do método CGS e quase não alterou os resultados do método CGNE . Fato que deve se explicar por termos usado uma fatoração incompleta da parte tridiagonal no método PCGS e apenas diagonal no método PCGNE .

Na tabela 3.2 encontramos , para cada método , a relação entre a o número de iterações e a dimensão do sistema .

Note que esta relação diminui , em todos os métodos testados , quando aumentamos a dimensão do sistema linear . Isto mostra que estes métodos são mais apropriados para sistemas lineares esparsos e de grande porte . Note também que em todos eles a convergência se deu em menos de **ndim** iterações .

npt	ndim	CGNE	PCGNE	CGS	PCGS
17	225	0.61	0.61	0.17	0.09
25	529	0.58	0.58	0.11	0.06
29	729	0.57	0.57	0.10	0.04
37	1225	0.55	0.55	0.08	0.04
41	1521	0.55	0.55	0.07	0.03

Tabela 3.2

Na figura abaixo encontramos as curvas de nível da solução do problema de difusão-convecção que resolvemos (não são mostrados os pontos da fronteira de Ω) .

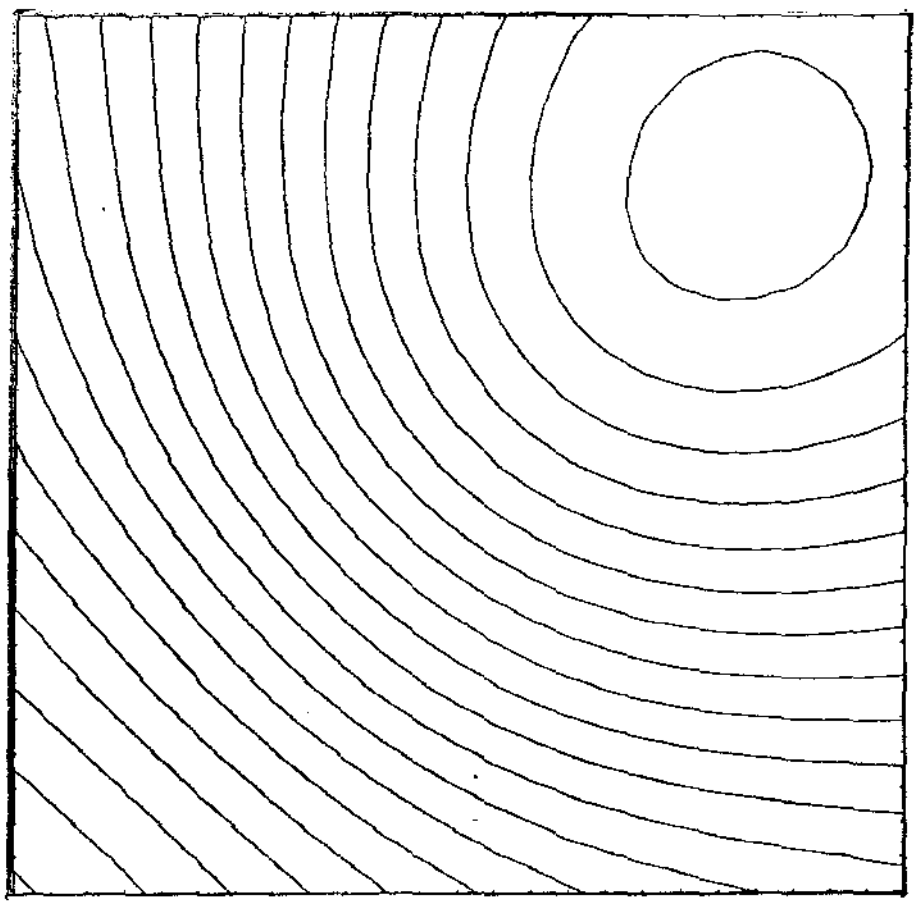

Figura 3.2

CAPÍTULO 4

CÁLCULO EM ESPAÇOS NORMADOS

4.1 - INTRODUÇÃO

No próximo capítulo , estaremos interessados em testar os métodos iterativos para sistemas lineares desenvolvidos até aqui num problema de valores de contorno não-linear . Desta forma , existem dois caminhos a serem seguidos : discretizamos a equação diferencial e então linearizamos o sistema algébrico não-linear resultante , ou começamos linearizando o operador diferencial para , posteriormante , discretizá-lo . Escolhemos a segunda forma por nos parecer mais elegante matematicamente . Para isso , necessitaremos de que o operador associado ao problema de valores de contorno seja diferenciável em algum senso generalizado . Este capítulo trata da extensão de algumas idéias básicas do cálculo diferencial a operadores em espaços normados . Este tipo de cálculo não somente é uma poderosa ferramenta para o estudo de operadores não-lineares com importantes aplicações , mas também é bastante interessante por si só .

4.2 - A DIFERENCIAL FORTE (FRÉCHET)

Sejam **X** e **Y** dois espaços normados e **F** uma aplicação definida sobre um aberto Ω de **X** ,

$$\begin{aligned} F : \Omega \subset X &\rightarrow Y \\ x &\rightarrow F(x) \quad . \end{aligned}$$

A aplicação F será dita diferenciável num ponto $x \in \Omega$, se existir um operador linear limitado $L \in \mathcal{L}(X,Y)$ gozando da seguinte propriedade : dado $\varepsilon > 0$, existe $\delta > 0$, dependendo de ε , tal que

$$\| F(x + h) - F(x) - L(h) \| < \varepsilon \| h \| \qquad (4.1)$$

sempre que $\| h \| < \delta$.
Ou mais brevemente ,

$$F(x + h) - F(x) - L(h) = o(h) , \qquad (4.2)$$

onde o(h) indica que

$$\lim_{h \to 0} \frac{F(x + h) - F(x) - L(h)}{h} = 0 .$$

De (4.1) resulta que uma aplicação diferenciável em x é contínua neste ponto . A expressão L(h) (que denota , obviamente , para qualquer $h \in X$, um elemento do espaço Y) chama-se diferencial forte (ou diferencial de Fréchet) da aplicação F em x . O operador L denomina-se derivada forte da aplicação F no ponto x que será denotada por F'(x) .

4.3 - A DIFERENCIAL FRACA (GATEAUX)

Sejam X e Y dois espaços normados e F uma aplicação definida sobre um aberto Ω de X ,

$$F : \Omega \subset X \to Y$$
$$x \to F(x) \quad .$$

Denomina-se diferencial fraca (ou diferencial de Gateaux) da aplicação F no ponto x , " na direção de h ", o limite

$$F'(x)[h] = \left. \frac{dF(x + th)}{dt} \right|_{t=0} = \lim_{t \to 0} \frac{F(x + th) - F(x)}{t} .$$

A diferencial fraca F'(x)[h] pode não ser linear em relação a h . Se tal linearidade se cumprir , isto é ,se $F'(x)[h] = F'_g(x).h$, onde $F'_g(x)$ é um operador linear limitado , então este último se chamará derivada fraca (ou derivada de Gateaux) .

4.4 - O VÍNCULO ENTRE AS DIFERENCIAÇÕES FORTE E FRACA

Os conceitos de derivada forte e derivada fraca não coincidem mesmo no caso de espaços de dimensão finita . No entanto , se uma aplicação F admitir a derivada forte , então admitirá igualmente a derivada fraca e ambas coincidirão . De fato , dada uma aplicação fortemente diferenciável , teremos

$$F(x + th) - F(x) = F'(x)(th) + o(th) = tF'(x)(h) + o(th) ,$$

logo ,

$$\frac{F(x + th) - F(x)}{t} = F'(x)(h) + \frac{o(th)}{t} \to F'(x)(h) ,$$

quando $t \to 0$.

Estabeleceremos certas condições que permitam , dada uma aplicação fracamente diferenciável F , concluir que esta é fortemente diferenciável .

Teorema 4.1 : *Se a derivada fraca* $F'_g(x)$ *de uma aplicação* F *existir em qualquer ponto de uma vizinhança* U *de* x_0 *e se a função que a cada* x *desta vizinhança faz corresponder o operador* $F'_g(x)$ *for contínua em* x_0 , *então a derivada forte* $F'(x_0)$ *existirá e coincidirá com a fraca .*

Demonstração : Referência [15] . □

4.5 - O MÉTODO DE NEWTON

Consideremos a equação

$$F(x) = 0 , \qquad (4.3)$$

onde F é uma aplicação de um espaço de Banach X num espaço de Banach Y. Admitiremos que a aplicação F é fortemente diferenciável numa bola $B(x_0,r)$, cujo centro x_0 será tomado a título de aproximação inicial da solução procurada . Substituindo F(x) pela sua parte linear $F(x_0)$ + + $F'(x_0)(x - x_0)$ na equação (4.3) temos

$$F(x_0) = F'(x_0)(x_0 - x)$$

que tem por solução

$$x_1 = x_0 - [\ F'(x_0)\ ]^{-1}F(x_0)\ .$$

Esta pode ser vista como uma nova aproximação da solução exata x^* da equação (4.3) , onde estamos admitindo a existência do operador $[F'(x_0)\ ]^{-1}$. Seguindo este raciocínio repetidas vezes obtemos a seguinte sequência

$$x_{n+1} = x_n - [\ F'(x_n)\ ]^{-1}F(x_n) \qquad (4.4)$$

das soluções aproximadas da equação (4.3) . A sequência (4.4) é conhecida como Método de Newton .

Daremos a seguir um teorema de convergência para o Método de Newton . Para demonstrá-lo , usaremos os resultados abaixo cujas provas poderão ser encontradas na referência [19] .

Lema de Banach : *Se* L *é um operador linear limitado num espaço de Banach* X , L^{-1} *existe se , e somente se , existe um operador linear limitado* M *em* X *tal que* M^{-1} *exista e*

$$\| I - M L \| < 1 \ .$$

Se L^{-1} *existe , então*

$$\| L^{-1} \| \leq \frac{\| M \|}{1 - \| I - ML \|} \quad . \quad \square$$

Para provar a convergência do Método de Newton torna-se conveniente estender o conceito de integral de Riemann para funções abstratas de uma variável real , isto é , operadores que levam $\mathbb{R}$ num espaço de Banach **Y** . Se P é um operador de um espaço de Banach **X** num espaço de Banach **Y** , então para

$$x(\theta) = \theta\, x_1 + (1 - \theta)\, x_0 \quad , \qquad 0 \leq \theta \leq 1 \quad ,$$

a função abstrata

$$P(\theta) = P(x(\theta)) = P(\theta\, x_1 + (1 - \theta)\, x_0)$$

levará o intervalo [0,1] num arco abstrato em **Y** , que tem como extremos $y_0 = P(x_0)$ e $y_1 = P(x_1)$.

Como na definição usual da integral de Riemann , particionamos o intervalo $0 \leq \theta \leq 1$ em n subintervalos de comprimento $\Delta\theta_i$, $i = 1,2,\ldots,n$, tomamos os pontos θ_i interiores ao subintervalo e formamos a soma

$$\sum_{\pi} P(\theta_i)\, \Delta\theta_i = \sum_{i=1}^{n} P(\theta_i)\, \Delta\theta_i \quad ,$$

onde π denota uma dada partição do intervalo . Para qualquer partição π , seja

$$| \pi | = \max_i \{ \Delta\theta_i \} \quad .$$

Se

$$J = \lim_{|\pi| \to 0} \sum_{\pi} P(\theta_i)\, \Delta\theta_i$$

existe , então será chamada de integral de Riemann de $P(\theta)$ de 0 a 1 , e será denotada por

$$J = \int_0^1 P(\theta)\, d\theta \quad .$$

A integral abstrata de Riemann tem várias propriedades em comum com a integral de Riemann em $\mathbb{R}^n$ que conhecemos , entre elas , a linearidade .

Uma função abstrata limitada $P(\theta)$ em $[0,1]$ que tenha um conjunto D_P de pontos de descontinuidade com medida nula é dita integrável em $[0,1]$.

Teorema 4.2 : *Se* $P'(\theta)$ *é integrável , então*

$$P(1) - P(0) = \int_0^1 P'(\theta)\, d\theta \quad . \quad \square$$

Com estes resultados em mente , passamos agora a enunciar e provar o seguinte teorema de convergência para o Método de Newton .

Teorema 4.3 : *Sejam* X *e* Y *dois espaços de Banach e* F *uma aplicação de* X *em* Y *fortemente diferenciável num aberto* $\Omega \subset X$. *Assuma que existam* $x^* \in X$ *e* $r,\beta > 0$, *tais que* $B(x^*,r) \subset \Omega$, $F(x^*) = 0$, $[F'(x^*)]^{-1}$ *exista com* $\| F'(x^*)^{-1} \| \leq \beta$, *e* $F'(x)$ *satisfazendo em* $B(x^*,r)$ *a condição de Lipschitz*

$$\| F'(x) - F'(y) \| \leq \gamma \| x - y \| \quad .$$

Então existe $\varepsilon > 0$ *tal que para todo* $x_0 \in B(x^*,\varepsilon)$, *a sequência* $x_1, x_2, \ldots$, *gerada por*

$$x_{k+1} = x_k - [F'(x_k)]^{-1} F(x_k) \qquad k = 0,1,\ldots \tag{4.5}$$

está bem definida , converge para x^* *e vale a relação*

$$\| x_{k+1} - x^* \| \leq \beta \gamma \| x_k - x^* \|^2 \quad k = 0,1,... \tag{4.6}$$

que garante que a taxa de convergência local é q-quadrática .

Demonstração : Escolheremos ε de forma que $[F'(x)]^{-1}$ exista para todo $x \in B(x^*,r)$.

Seja

$$\varepsilon = \min \left\{ r , \frac{1}{2\beta\gamma} \right\} . \tag{4.7}$$

Mostraremos por indução em k que em todo passo a relação (4.6) vale e também que

$$\| x_{k+1} - x^* \| \leq \frac{1}{2} \| x_k - x^* \|$$

e , portanto ,

$$x_{k+1} \in B(x^*,\varepsilon) . \tag{4.8}$$

Primeiro vamos mostrar que $[F'(x_0)]^{-1}$ existe . Lembrando que $\| x_0 - x^* \| \leq \varepsilon$, a condição de Lipschitz e (4.7) segue que

$$\| F'(x^*)^{-1}[F'(x_0) - F'(x^*)] \| \leq \| F'(x^*)^{-1} \| \| F'(x_0) - F'(x^*) \| \leq$$

$$\leq \beta \gamma \| x_0 - x^* \| \leq \beta \gamma \varepsilon \leq \frac{1}{2} .$$

Pelo Lema de Banach , $F'(x_0)^{-1}$ existe e

$$\| F'(x_0)^{-1} \| \leq \frac{\| F'(x^*)^{-1} \|}{1 - \| F'(x^*)^{-1}[F'(x_0)-F'(x^*)] \|} \tag{4.9}$$

$$\leq 2 \| F'(x^*)^{-1} \| \leq 2\beta .$$

Portanto x_1 está bem definido e

$$x_1 - x^* = x_0 - x^* - [F'(x_0)]^{-1}F(x_0)$$

$$= x_0 - x^* - [F'(x_0)]^{-1}[F(x_0) - F(x^*)]$$

$$= [F'(x_0)]^{-1}[\, F(x^*) - F(x_0) - F'(x_0)(x^* - x_0)\,] \,.$$

Note que o termo entre colchetes é a diferença entre $F(x^*)$ e a sua aproximação linear calculada em x^* .
Então

$$\| x_1 - x^* \| \leq \| F'(x_0)^{-1} \| \| F(x^*) - F(x_0) - F'(x_0)(x^* - x_0) \| \,.$$

Mas pelo teorema 4.2 e a condição de Lipschitz

$$\| F(x^*) - F(x_0) - F'(x_0)(x^* - x_0) \| =$$

$$= \left\| \int_0^1 [\, F'(\theta) - F'(x_0)]\, (x^* - x_0) d\theta \right\| =$$

$$= \left\| \int_0^1 [\, F'(\theta\, x^* + (1 - \theta)\, x_0) - F'(x_0)\,]\, (x^* - x_0)\, d\theta \right\| \leq \tag{4.10}$$

$$\leq \int_0^1 \| F'(\theta\, x^* + (1 - \theta)\, x_0) - F'(x_0) \| \, \| x^* - x_0 \| \, d\theta \leq$$

$$\leq \int_0^1 \gamma \, \| \theta\, (x^* - x_0) \| \, \| x^* - x_0 \| \, d\theta =$$

$$= \int_0^1 \gamma \, \| x^* - x_0 \|^2 \theta \, d\theta = \frac{\gamma \, \| x^* - x_0 \|^2}{2} \,.$$

Então , usando (4.9) temos que

$$\| x_1 - x^* \| \leq \beta \gamma \, \| x^* - x_0 \|^2 \,.$$

Isto prova (4.6) .

Como $\| x_0 - x^* \| \leq \frac{1}{2\beta\gamma}$,

$$\| x_1 - x^* \| \leq \frac{1}{2} \| x^* - x_0 \| ,$$

que mostra (4.8) e completa o caso $k = 0$. A prova segue identicamente para os passos seguintes . ∎

4.6 - O MÉTODO INEXATO DE NEWTON

Sejam **X** e **Y** dois espaços de Banach e F uma aplicação definida sobre um aberto Ω de X ,

$$\begin{aligned} F : \Omega \subset X &\rightarrow Y \\ x &\rightarrow F(x) \end{aligned}$$

fortemente diferenciável em Ω . Vamos considerar novamente a equação

$$F(x) = 0 . \tag{4.11}$$

Um dos algoritmos de maior importância para resolver (4.11) é , sem dúvida , o método de Newton . Este método tem como principal atrativo a rapidez com que converge . Dada uma aproximação inicial x_0, computamos uma sequência de passos $\{s_n\}$ e uma sequência de aproximações $\{x_n\}$ como segue :

$$\begin{aligned} F'(x_n)s_n &= - F(x_n) \\ x_{n+1} &= x_n + s_n , \end{aligned} \tag{4.12}$$

onde F'(x) denota a derivada de Fréchet do operador F calculada em x . Métodos de discretização para resolver (4.12) originarão um sistema linear de grande porte e esparso . Um dos problemas do método de Newton pode ser resolver este sistema , em cada iteração . Calcular a solução usando um método direto tal qual fatoração LU pode requerer grande esforço computacional . Além disso , quando x_n estiver longe de

x^* , todo este esforço pode não ser justificado .

Neste caso , pode ser razoável o uso de um método iterativo para resolver o sistema linear resultante da discretização de (4.12) apenas aproximadamente . Isto é , nas primeiras iterações de Newton , quando provavelmente estivermos mais longe da solução x^* , exigiremos menos do método iterativo para o sistema linear .

Uma condição de parada natural pode se basear no tamanho relativo do resíduo

$$\frac{\| r_n \|}{\| F(x_n) \|} ,$$

onde

$$r_n \equiv F'(x_n)s_n + F(x_n) .$$

Uma questão importante é que nível de precisão será necessária para preservar as propriedades de convergência local do método de Newton .

De forma mais geral , consideraremos uma classe de métodos Inexatos de Newton que calculam uma solução aproximada das equações de Newton de forma que

$$\frac{\| r_n \|}{\| F(x_n) \|} \leq \eta_n ,$$

onde a sequência não-negativa $\{\eta_n\}$ é usada para controlar o nível de precisão . Para ser mais preciso , um método Inexato de Newton é todo algoritmo que , dada uma aproximação inicial x_0 para a solução x^* de (4.11) , gera uma sequência $\{x_n\}$ de aproximações para x^* como segue :

- encontrar algum passo s_n satisfazendo

$$F'(x_n)s_n = - F(x_n) + r_n$$

$$x_{n+1} = x_n + s_n ,$$

onde

$$\frac{\| r_n \|}{\| F(x_n) \|} \leq \eta_n \ .$$

Aqui η_n pode depender de x_n . Tomando $\eta_n \equiv 0$ temos o método de Newton.

Na referência [5] , o comportamento local de tais métodos é analisado . Prova-se que estes são localmente convergentes com taxa q-linear , isto é , $\exists\ c \in [0,1)$ e $\bar{K}$ tal que para $k \geq \bar{K}$

$$\| x_{k+1} - x^* \| \leq c \| x_k - x^* \| \ ,$$

se a sequência $\{\eta_n\}$ for uniformemente menor que 1 . As demostrações foram feitas para F de $\mathbb{R}^n$ em $\mathbb{R}^n$, mas todos os resultados podem ser generalizados para o caso de um espaço de Banach qualquer . Alguns destes resultados podem ser encontrados na referência [16] .

4.7 - EQUAÇÃO DE POISSON NÃO-LINEAR

Para ilustrar as discussões deste capítulo , consideraremos o seguinte problema de valores de contorno não-linear

$$-\Delta u = \rho(u) \ ; \ x \in \Omega \tag{4.13}$$

$$u = g \ ; \ x \in \partial\Omega \tag{4.14}$$

onde $\Omega \subset \mathbb{R}^2$, $u = u(x_1,x_2)$ e $g \in C^2(\Omega)$ é uma função dada .

Seja F o operador diferencial

$$\begin{aligned} F : C^2(\Omega) &\longrightarrow C(\Omega) \\ u &\longrightarrow F(u) = \Delta u + \rho(u) \ . \end{aligned}$$

Desta forma , o problema (4.13)-(4.14) pode ser escrito como :

- encontrar $u \in C^2(\Omega)$ tal que

$$F(u) = 0 \quad ; \quad x \in \Omega$$

$$u = g \quad ; \quad x \in \partial\Omega \, .$$

O diferencial de Gateaux de F no ponto $\bar{u}$, " na direção de h " , é dado por

$$F'(\bar{u})[h] = \Delta h + h \, . \frac{\partial \rho(u)}{\partial u} \Bigg|_{u = \bar{u}} \, .$$

Note que o diferencial do operador F é linear em relação a h . Portanto ,

$$F'(\bar{u})[h] = F'_g(\bar{u}).h$$

onde

$$F'_g(\bar{u}) = \Delta + I \, . \frac{\partial \rho(u)}{\partial u} \Bigg|_{u = \bar{u}}$$

é um operador linear limitado de $C^2(\Omega)$ em $C(\Omega)$ e pode ser chamado de derivada fraca de F . Além disso , o diferencial de F é contínuo no ponto $\bar{u}$, desde que ρ seja continuamente diferenciável . Portanto , de acordo com o teorema 4.1 , a derivada forte do operador F existe e coincide com a sua derivada fraca .

$$F'(\bar{u}) \; : \; C^2(\Omega) \longrightarrow C(\Omega)$$

$$F'(\bar{u}) = \Delta + I \, . \frac{\partial \rho(u)}{\partial u} \Bigg|_{u = \bar{u}} \, .$$

CAPÍTULO 5

SUPERFÍCIE LIVRE DE CAPILARIDADE

5.1 - INTRODUÇÃO

Fenômenos de capilaridade estão presentes no nosso dia-a-dia .Todos nós já observamos uma gota de orvalho sobre a folha de uma planta ou já nos perguntamos como algumas árvores tão grandes podem transportar seiva e água de sua raiz até suas folhas . Mesmo assim ,até o século XVIII , não existia a percepção de que estes , e muitos outros fenômenos , são todos manifestações que ocorrem quando dois materiais diferentes são colocados em contato e não se misturam . Se pelo menos um dos materiais é um fluído , que forma com outro fluído (ou gás) uma superfície livre , então esta interface será chamada de superfície livre de capilaridade .

Um dos primeiros cientistas a se preocupar com a observação destes fenômenos foi Leonardo Da Vinci , tendo uma teoria mais consistente e capaz de fazer predições científicas sido desenvolvida apenas no início do século XIX nos trabalhos de Young e Laplace . Esta teoria foi colocada mais tarde com uma fundamentação teórica mais sólida por Gauss e tornou-se objeto de extensivos estudos por alguns dos mais renomados cientistas daquele século . Este problema caiu de moda durante a primeira metade deste século , entretanto , os novos desenvolvimentos matemáticos em Superfícies Mínimas e o interesse causado pela Medicina e pelas experiências com crescimento de cristais realizadas nos Programas Espaciais , reativaram a pesquisa dos fenômenos de capilaridade , donde podemos destacar principalmente os trabalhos de Emmer , Tamanini , Gonzalez , Robert Finn e Paul Concus .

Do ponto de vista da engenharia , problemas específicos têm sido tratados usando-se métodos tradicionais , devidos inicialmente e acidentalmente a Laplace , e também numericamente . Em geral , bons resultados são alcançados , porém , em algumas situações particulares, resultados incoerentes são obtidos .

Inicialmente , vamos apresentar alguns problemas clássicos de capilaridade :

i) *Gota Apoiada :* Considere uma gota de um líquido com um volume prescrito V apoiada num plano horizontal π sujeita a um campo de gravidade na direção perpendicular a π . Considere também que o plano tenha material homogêneo de forma que o ângulo de contacto δ entre a gota e o plano seja constante , $0 \leq \delta \leq \Pi$. Nosso problema é calcular a forma da gota .

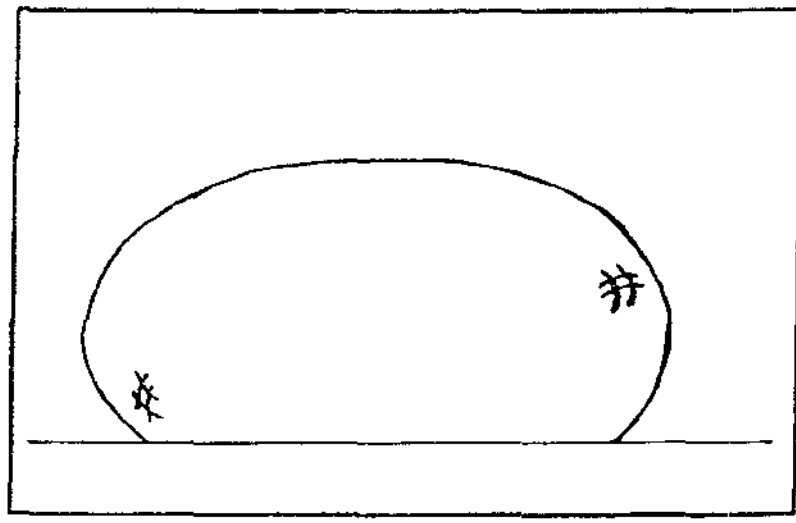

Figura 5.1 : Gota Apoiada

ii) *Gota Pendente :* Se , na configuração anterior , a direção da gravidade for invertida , obtemos um novo problema , isto é , determinar a forma da gota suspensa .

iii) *Ponte Líquida :* Um volume V de um líquido contido entre duas superfícies sólidas forma uma "ponte" entre os dois sólidos . Nosso problema é determinar a forma que este líquido toma .

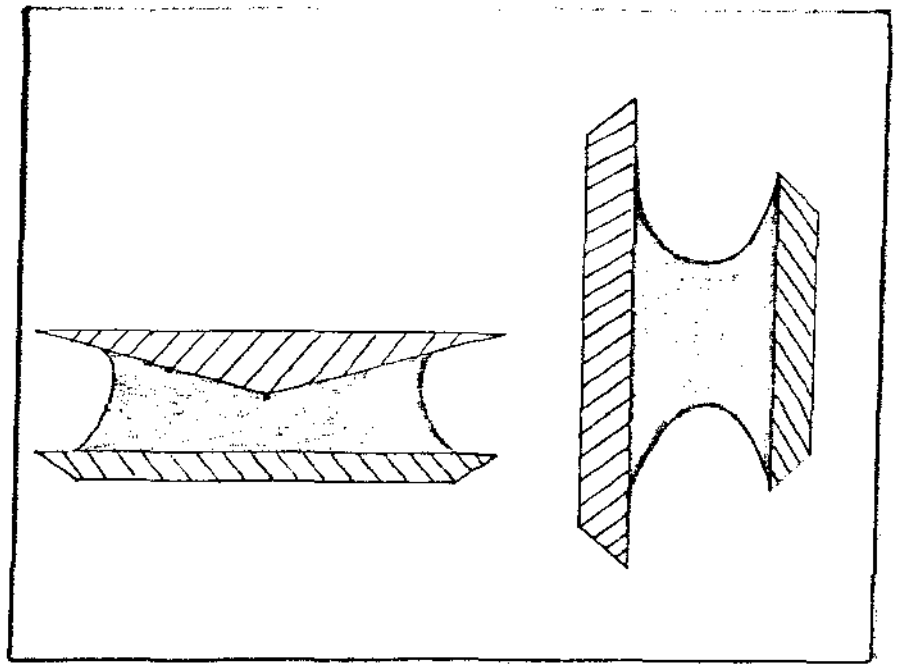

Figura 5.2 : Ponte Liquida

iv) *Transporte em um Capilar* : Dois fluídos imisciveis e com diferentes propriedades de adesão em relação a um tubo capilar e , se movendo dentro deste tubo . Este é mais um problema de capilaridade onde o que se pretende determinar é o efeito desta adesão na velocidade da interface (menisco) entre os dois fluídos .

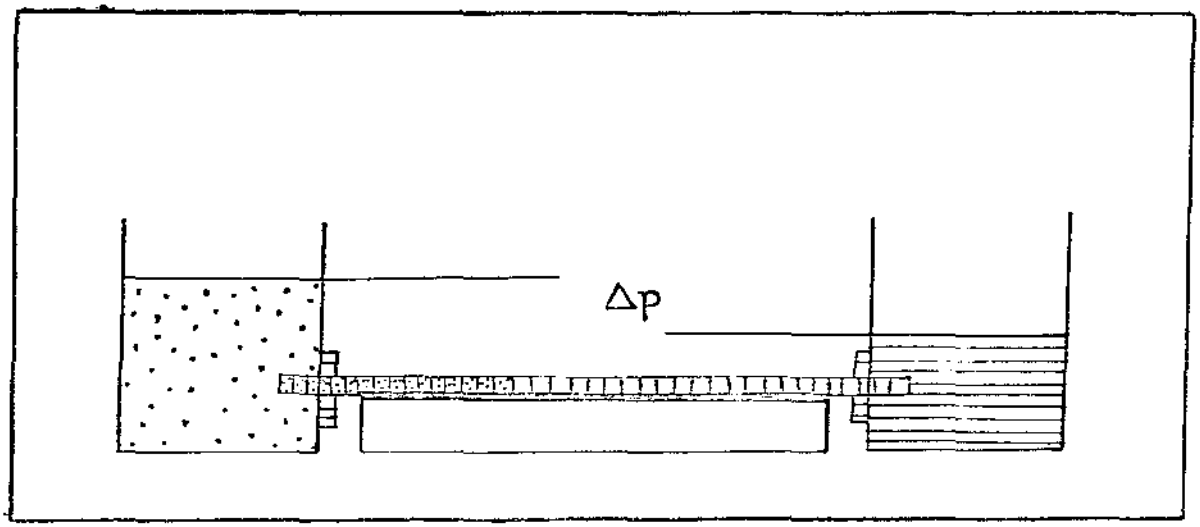

Figura 5.3 : Transporte em um capilar

v) *Tubo Capilar* : Dado um volume **V** de um líquido confinado em um recipiente girando em um campo gravitacional tendo o seu eixo de rotação paralelo ao campo , gostaríamos de determinar a superfície livre de separação entre o líquido e o ar .

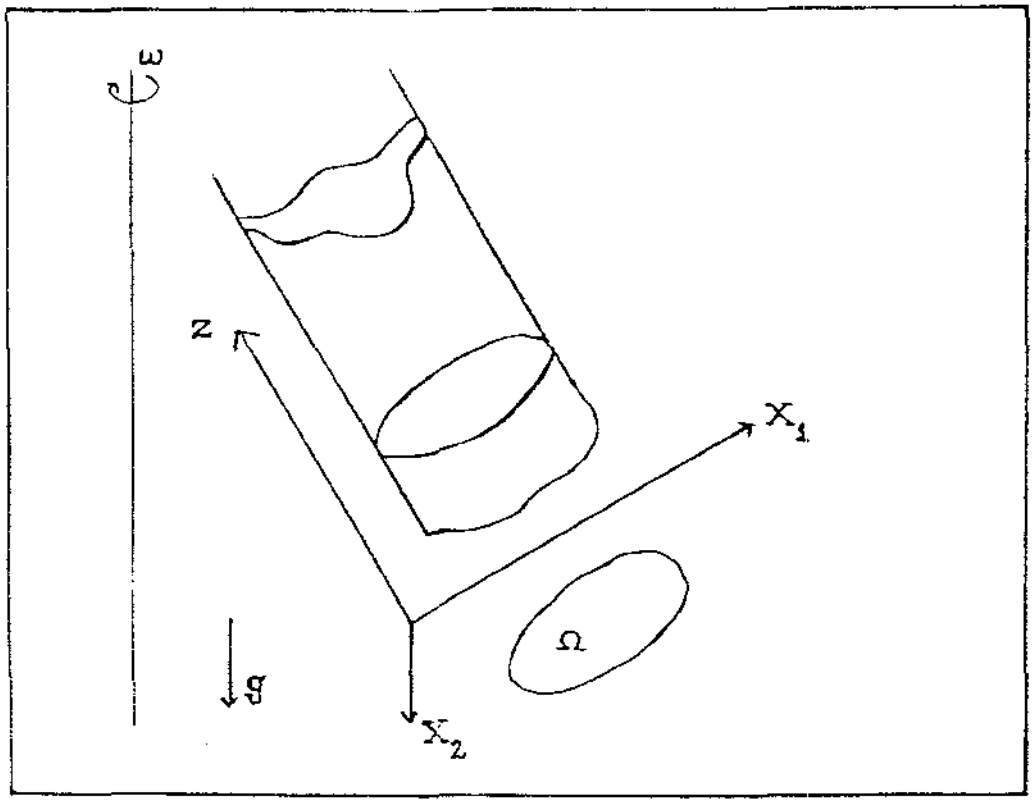

Figura 5.4 : Tubo capilar girando

5.2 - O MÉTODO DE GAUSS : CARACTERIZAÇÃO DAS ENERGIAS

Vamos considerar um sistema como o da figura 5.5 , composto por um fluído , um gás e um tubo capilar .

Laplace mostrou que a curvatura média H de uma superfície é proporcional à variação da pressão através da mesma . Então , usando as leis da Hidrostática obtemos

$$H = \frac{1}{2} Ku \tag{5.1}$$

onde u é a altura da superfície livre medida a partir do nível de pressão atmosférica e K é uma constante física que depende das características do fluído e do campo gravitacional local .

A equação (5.1) é conhecida como Equação de Young e Laplace .

Vamos considerar uma superfície S em $\mathbb{R}^3$ definida pela equação

$$z = u(x,y) \quad ; \quad (x,y) \in \Omega \subset \mathbb{R}^2 \quad ,$$

onde Ω é um domínio de $\mathbb{R}^2$.

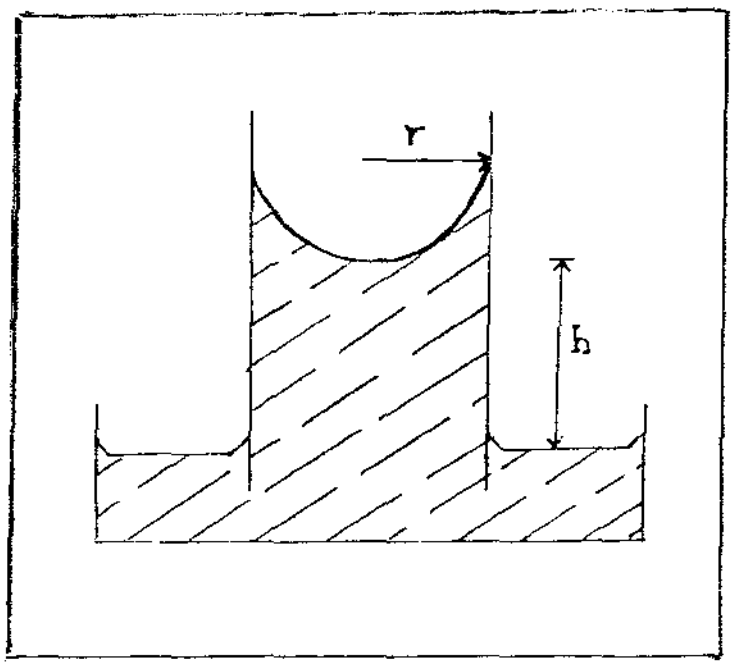

Figura 5.5

Para a situação física descrita acima , a energia é composta por quatro componentes , a saber :

5.2.1 - ENERGIA SUPERFICIAL LIVRE (E_s)

Quando em equilíbrio , as partículas de um fluído na superfície livre de separação dos dois meios devem atrair mais as partículas deste fluído que as do outro meio , caso contrário os dois meios se misturariam e a superfície desapareceria . A energia associada com esta atração dos fluídos deve ser proporcional à área da superfície de separação , isto é ,

$$E_S = \sigma A \tag{5.2}$$

onde

A : área da superfície de separação ;

σ : tensão superficial - trabalho necessário para aumentar a área de uma superfície em uma unidade - ([força]/[comprimento]) .

No caso em que a superfície S é descrita pela equação $z = u(x,y)$, sua área é dada por

$$A = \int\int_{\Omega} \left(1 + u_x^2 + u_y^2\right)^{1/2} dx\ dy \quad . \tag{5.3}$$

5.2.2 - ENERGIA DE MOLHABILIDADE (E_W)

A Energia de Molhabilidade está presente devido às forças de adesão entre o fluído e o bordo do tubo capilar .

$$E_W = -\sigma\beta A^* \tag{5.4}$$

onde

β : coeficiente relativo de adesão entre o fluído e as paredes do capilar ;

A^* : área de contato entre o fluído e o bordo do tubo capilar .

Para a superfície S descrita acima , a área de contato é dada por

$$A^* = \oint_{\partial\Omega} u\ ds \quad . \tag{5.5}$$

5.2.3 - ENERGIA POTENCIAL GRAVITACIONAL (E_p)

A Energia Potencial Gravitacional depende somente da posição da superfície em relação ao nível de pressão atmosférica (referencial na figura 5.6) .

$$E_p = \frac{1}{2} \rho g \int\int u^2 \, dx \, dy \qquad (5.6)$$

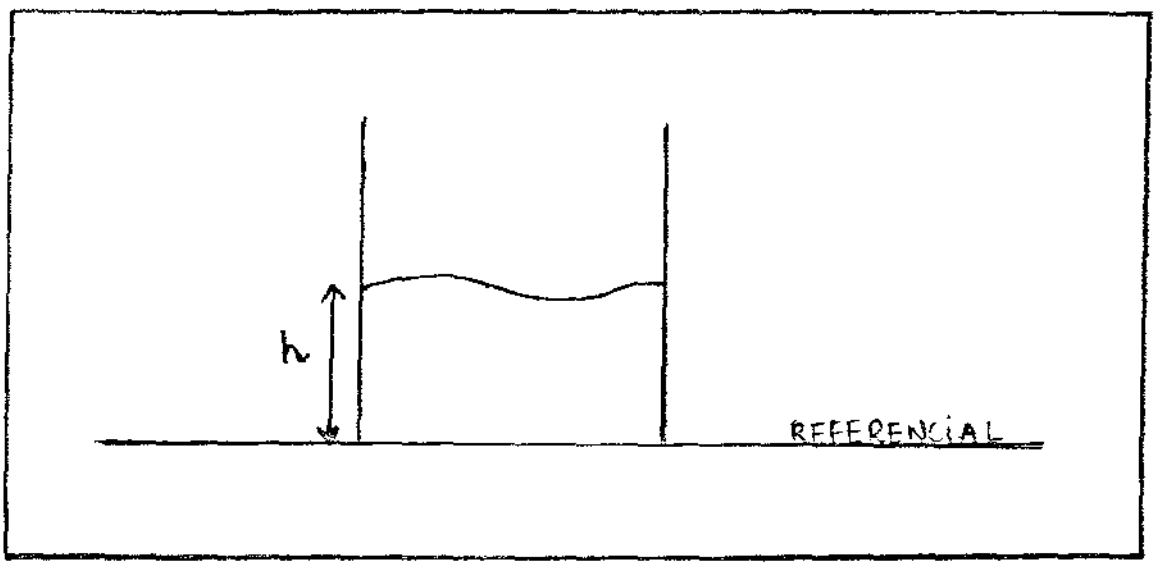

Figura 5.6

5.2.4 - RESTRIÇÃO DE VOLUME

Para muitos problemas , o volume do fluído deve permanecer constante . Uma forma natural de respeitar esta restrição é introduzir o volume V multiplicado por um parâmetro de Lagrange λ como um novo termo de energia .

$$E_v = \sigma \lambda V \qquad (5.7)$$

onde o multiplicador λ deve ser determinado .

No caso da superfície S descrita por $z = u(x,y)$, o volume de fluído é dado por

$$V = \int\int_{\Omega} u \, dx \, dy \, . \tag{5.8}$$

5.3 - PRINCÍPIO DA MÍNIMA ENERGIA

A origem do princípio físico de mínima energia para situações de equilíbrio data pelo menos do século XVII pois , nos trabalhos de Torricelli (por volta de 1644) já está presente o postulado de que um sistema de corpos sob a ação de um campo gravitacional será estável se o seu centro de gravidade ocupar a posição mais baixa possível . Neste caso , o que se requer é a minimização da energia potencial do sistema .

Em seu histórico trabalho sobre Mecânica de Partículas , Lagrange enunciou este princípio para campos conservativos .

A demonstração matemática deste princípio mecânico e a sua extensão , como hipótese , para o equilíbrio de corpos elásticos , é atribuída a Dirichlet - aluno de Gauss e Jacobi - que sugeriu a partir daí um método matemático de grande generalidade para o estudo das equações diferenciais parciais . Um de seus alunos , Riemann , atento às suas sugestões , iniciou o desenvolvimento de algumas idéias baseadas no que ele mesmo denominou de Princípio de Dirichlet . Estas idéias deram origem a uma parte indispensável da teoria contemporânea das equações diferenciais parciais e influenciaram decisivamente os caminhos da Matemática , do final do século XIX até hoje .

Para maiores detalhes consulte também a referência [2] .

Baseados nas discussões feitas até aqui , temos o seguinte funcional de energia para o sistema que estamos descrevendo

$$E(u) = E_s + E_w + E_p + E_v \, . \tag{5.9}$$

A superfície capilar $z = u(x,y)$ é a função que realiza o mínimo do funcional de energia (5.9) .

Para determinar condições necessárias para que $u(x,y)$ realize o mínimo de $E(u)$, consideraremos uma família de superfícies que contenha a superfície ótima .

Podemos reescrever o funcional (5.9) como :

$$E(u) = \sigma \left\{ \iint_{\Omega} \left(1 + u_x^2 + u_y^2 \right)^{1/2} dx\, dy + \lambda \iint_{\Omega} u\, dx\, dy + \right. \tag{5.10}$$

$$\left. + \frac{\rho g}{2\sigma} \iint_{\Omega} u^2\, dx\, dy - \beta \oint_{\partial\Omega} u\, ds \right\} .$$

Introduzimos uma perturbação à candidata $u(x,y)$ à superfície de equilíbrio fazendo $\tilde{u}(x,y,\varepsilon) = u(x,y) + \varepsilon\eta(x,y)$, onde η é uma função arbitrária de classe $C^1(\Omega)$. Substituindo $\tilde{u}$ em (5.10) , o funcional de energia $E(u)$ passa a depender unicamente de ε . Desta forma $E(u)$ será mínimo para $\varepsilon = 0$, por hipótese . Portanto ,

$$\left. \frac{\partial E}{\partial \varepsilon} \right|_{\varepsilon=0} = 0 .$$

$$\left. \frac{\partial E}{\partial \varepsilon} \right|_{\varepsilon=0} = \sigma \left\{ \iint_{\Omega} \frac{\langle \nabla u, \nabla \eta \rangle}{\left(1 + \| \nabla u \|^2 \right)^{1/2}} dxdy + \lambda \iint_{\Omega} \eta\, dx\, dy + \right.$$

$$\left. + \frac{\rho g}{\sigma} \iint_{\Omega} u\, \eta\, dx\, dy - \beta \oint_{\partial\Omega} \eta\, ds \right\} = 0 .$$

Usando o Teorema de Green obtemos

$$\int\int_{\Omega} \eta \left(- \operatorname{div} (Tu) + \frac{\rho g}{\sigma} u + \lambda \right) dx\, dy +$$

$$+ \oint_{\Omega} \eta \left(\langle Tu, n \rangle - \beta \right) ds = 0 \qquad (5.11)$$

onde

$$Tu = \frac{\nabla u}{\left(1 + \| \nabla u \|^2 \right)^{1/2}} ,$$

e n é o vetor normal à fronteira de Ω .

Se tivermos

$$- \operatorname{div}(Tu) + \frac{\rho g u}{\sigma} + \lambda \neq 0$$

para algum ponto p de Ω , e tomarmos a função η positiva e com suporte compacto numa pequena vizinhança centrada em p , a segunda integral em (5.11) se anula de forma que a igualdade em (5.11) não seja possível . Temos assim uma contradição . Portanto , a primeira integral de (5.11) deve se anular identicamente . Com isso a segunda integral de (5.11) deve também se anular . Devido a arbitrariedade de η , obtemos

$$\operatorname{div} \left(\frac{\nabla u}{\left(1 + \| \nabla u \|^2 \right)^{1/2}} \right) = \frac{\rho\, g\, u}{\sigma} + \lambda , \qquad (5.12)$$

sobre Ω .

E a relação

$$\beta = \langle Tu, n \rangle \tag{5.13}$$

em $\partial\Omega$.

A equação diferencial parcial não-linear (5.12) é do tipo elíptico . Alguns cálculos podem mostrar que a curvatura média da superfície S pode ser expressa como um divergente , veja a referência [13] . Assim , se compararmos as equações (5.1) e (5.12) , vemos que elas são semelhantes . A diferença entre elas se encontra no multiplicador λ , visto que no problema estudado por Laplace não havia restrição de volume .

Young e Laplace também observaram que o ângulo de contato δ entre a superfície capilar e as paredes do tubo se mantinha constante ao longo do bordo do capilar .

Introduziremos os vetores **N** e **n** para deduzirmos as condições de contorno do problema . **N** é um vetor unitário normal à superfície e voltado para fora do fluído , enquanto **n** é um vetor unitário normal à fronteira da região Ω (veja a figura 5.7) .

Seja também o ângulo Θ entre os vetores **N** e **n** . Da figura 6 concluímos que

$$\Theta + \delta = \Pi \ , \tag{5.14}$$

que implica em

$$\cos(\Theta) = - \cos(\delta) \ . \tag{5.15}$$

O vetor **N** pode ser dado por

$$\mathbf{N} = \frac{(-ux, -uy, 1)}{(1 + ux^2 + uy^2)^{1/2}} \ . \tag{5.16}$$

Da definição de produto interno , temos que

$$\langle N,n \rangle = \cos(\Theta) \quad , \tag{5.17}$$

então , usando (5.15) ,

$$\langle N,n \rangle = - \cos(\delta) \quad . \tag{5.18}$$

Como n está contido no plano xy , isto é , não tem componente na direção do eixo z , podemos escrever , usando também (5.13)

$$\langle N,n \rangle = - \frac{\langle n, \nabla u \rangle}{(1 + u_x^2 + u_y^2)^{1/2}} = - \langle Tu, n \rangle = -\beta \quad . \tag{5.19}$$

Comparando (5.18) e (5.19) , chegamos à seguinte condição de contorno

$$\frac{\langle n, \nabla u \rangle}{(1 + u_x^2 + u_y^2)^{1/2}} = \cos(\delta) \quad , \tag{5.20}$$

e à relação

$$\beta = \cos(\delta) \quad . \tag{5.21}$$

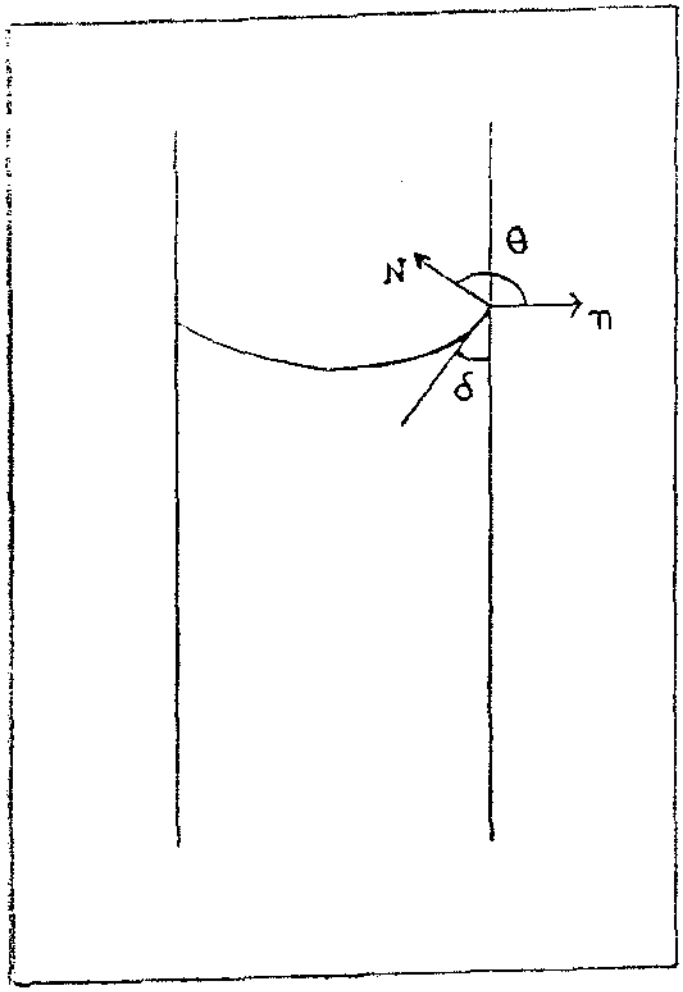

Figura 5.7

5.4 - TUBO CAPILAR CILÍNDRICO BIDIMENSIONAL

5.4.1 - COLOCAÇÃO DO PROBLEMA

Vamos considerar nesta seção , um problema de capilaridade estática onde um líquido está contido num capilar bidimensional V sob a ação de um campo gravitacional . Assumimos que o recipiente V é cilíndrico e está colocado num sistema de coordenadas cartesianas , isto é

$$V = \left\{ (x,y) \in \mathbb{R}^2 \ / \ y > \varphi_\Gamma(x) \ , \ 0 < x < L \right\} ,$$

onde L é a largura do capilar e ,

$$\varphi_\Gamma : x \rightarrow \varphi_\Gamma(x) \quad , \qquad 0 \leq x \leq L ,$$

é uma função que descreve o fundo do recipiente . O capilar é colocado de forma que o ângulo entre a direção negativa do eixo y e a direção da força de gravidade g seja igual a γ (veja figura 5.8) . Nesta situação , o único campo de forças externo presente é o devido à gravidade .

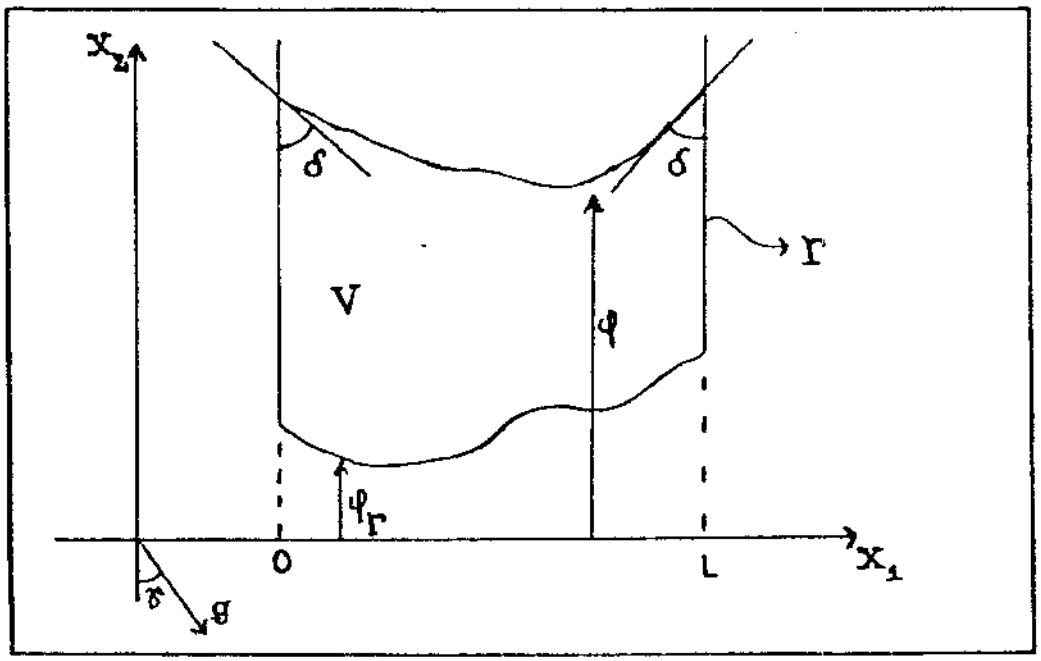

Figura 5.8

Assumimos que a interface liquido-gás admite uma função que a represente $\varphi(x)$ para $0 \leq x \leq L$. Esta situação pode ser considerada como um problema tridimensional invariante na direção perpendicular ao plano xy .

Nesta situação , a curvatura média pode ser expressa em termos da função φ ,

$$H = \frac{1}{2} \left[\frac{\varphi'}{(1 + (\varphi')^2)^{1/2}} \right]'$$

onde o símbolo " ' " indica derivação com relação à variável x .

As condições para o ângulo de contacto δ são dadas por

$$\varphi'(0) = - \operatorname{cotan}(\delta) \quad , \qquad \varphi'(L) = \operatorname{cotan}(\delta) \quad .$$

Se o volume do líquido no capilar é igual a $| V |$, então

$$\int_0^L \varphi(x)\, dx = | V | + \int_0^L \varphi_\Gamma(x)\, dx \quad .$$

O problema pode ser escrito , então , na seguinte forma adimensional com escala L :

- Encontrar uma função $x \to \varphi(x)$, $0 \leq x \leq 1$ e uma constante λ tal que

$$- \left[\frac{\varphi'}{(1 + (\varphi')^2)^{1/2}} \right]' + B_0 \varphi \cos(\gamma) = B_0 x \sin(\gamma) + \lambda \qquad \text{em } (0,1) \ ,$$

$$\varphi'(0) = - \operatorname{cotan}(\delta) \ , \qquad \varphi'(1) = \operatorname{cotan}(\delta) \ , \qquad (5.22)$$

$$\int_0^1 \varphi(x)\,dx = V_c\,, \quad V_c = \int_0^1 \varphi_\Gamma(x)\,dx + L^{-2}|V|$$

com $B_0 = \dfrac{\rho g L^2}{\sigma}$ Número de Ligação .

A constante λ pode ser expressa em termos dos dados do problema . Integrando a equação diferencial (5.22) de 0 a 1 e usando as condições de Neumann e de volume temos

$$\lambda = V_c B_0 \cos(\gamma) - \frac{1}{2} B_0 \sin(\gamma) - 2\cos(\delta)\,. \qquad (5.23)$$

Se em (5.22) as condições de contorno forem do tipo Dirichlet será impossível encontrar uma expressão analítica para a constante λ. Portanto , para o caso geral , λ é uma incógnita do problema que deve ser calculada como parte integrante da solução do mesmo .

5.4.2 - LINEARIZAÇÃO

O problema (5.22) é uma equação diferencial ordinária não-linear que , para nossas propostas numéricas , deve ser linearizada .

Consideramos a equação não-linear $F(\varphi) = 0$ onde

$$F(\varphi) = -\left[\frac{\varphi'}{(1 + (\varphi')^2)^{1/2}}\right]' + B_0\varphi\cos(\gamma) - B_0 x \sin(\gamma) - \lambda. \qquad (5.24)$$

Usaremos o Método de Newton para linearizar este problema :

i) Tomamos uma aproximação inicial φ_0 para a solução φ de (5.22).

ii) Geramos uma sequência $\varphi_1, \varphi_2, \ldots, \varphi_n, \varphi_{n+1}$ através do seguinte problema linear :

$$F'(\varphi_n)(\varphi_{n+1} - \varphi_n) = -F(\varphi_n) , \qquad (5.25)$$

onde $F'(\varphi)$ denota a derivada forte do operador F no ponto φ .

Vamos calcular , primeiro , a derivada de Gateaux de F no ponto φ e na direção de ψ .

$$F'(\varphi)[\psi] = \lim_{\varepsilon \to 0} \frac{F(\varphi + \varepsilon\psi) - F(\varphi)}{\varepsilon} .$$

$$F'(\varphi)[\psi] = - \left[\frac{\psi'}{(1 + (\varphi')^2)^{3/2}} \right]' + B_0 \psi \cos(\gamma) .$$

Como a derivada de Gateaux está de acordo com as hipóteses do teorema 4.1 concluímos que a derivada forte de F existe e coincide com a derivada fraca . Vamos , portanto , utilizá-la para gerar a sequência de Newton através de (5.25) . Pode ser mostrado que o termo geral φ_{n+1} da sequência satisfaz o seguinte problema de fronteira , linear

$$\begin{aligned} -\left[\frac{\varphi'_{n+1}}{(1 + (\varphi'^2_n))^{3/2}} \right]' + B_0 \varphi_{n+1} \cos(\gamma) - \lambda = \\ = B_0 x \sin(\gamma) + \left[\frac{(\varphi'_n)^3}{(1 + (\varphi'^2_n))^{3/2}} \right]' \quad \text{em } (0,1) . \end{aligned} \qquad (5.26)$$

$$\varphi'_{n+1}(0) = -\text{cotan}(\delta) , \qquad \varphi'_{n+1}(1) = \text{cotan}(\delta) ,$$

$$\int_0^1 \varphi_{n+1}(x)\, dx = V_c .$$

5.4.3 - DISCRETIZAÇÃO E RESULTADOS NUMÉRICOS

Para ilustrar as nossas discussões anteriores , apresentamos alguns experimentos numéricos .

Consideramos um cilindro V com fundo $\varphi_{\Gamma} \equiv 0$. O volume de liquido no mesmo foi tomado unitário , isto é , $V_c = 1$. Resolvemos o problema (5.26) através de uma discretização de Diferenças Finitas Centrais . Subdividimos o intervalo (0,1) em um número finito de subintervalos $0 = x_0 < x_1 < \ldots < x_{m-1} < x_m = 1$. Se $(\varphi_{n+1})_i$ denota uma aproximação de $\varphi_{n+1}(x_i)$, então o método de discretização dá origem a um sistema de equações algébricas lineares , com a estrutura representada em (5.7) .

$$\begin{bmatrix} x & x & & & & x \\ x & x & x & & & x \\ & x & x & x & & x \\ & & \ddots & \ddots & & \vdots \\ & & & x & x & x \\ & & & & x & x \\ x & x & \ldots & x & x & 0 \end{bmatrix} \begin{bmatrix} (\varphi_{n+1})_0 \\ \vdots \\ (\varphi_{n+1})_m \\ \lambda \end{bmatrix} = \begin{bmatrix} x \\ x \\ x \\ \vdots \\ x \\ x \\ x \\ x \end{bmatrix} . \qquad (5.27)$$

Note que a última coluna do sistema tem como incógnita a constante λ . Para determiná-la , acrescentamos a última linha do sistema que representa a discretização da restrição de volume pela Regra dos Trapézios de integração numérica :

$$\int_0^1 \varphi_{n+1}(x)\, dx \cong \sum_{i=1}^{m} (x_i - x_{i-1}) \left[\frac{(\varphi_{n+1})_i + (\varphi_{n+1})_{i-1}}{2} \right] = V_c .$$

Podemos também utlizar as condições de contorno e calcular λ analiticamente por (5.23) , eliminando esta incógnita do sistema , bem como , a sua última linha .

Trabalhando com a idéia do Método Inexato de Newton , resolvemos os sistemas lineares resultantes desta discretização através de vários métodos iterativos . Todos eles implementados com armazenamento da matriz do sistema por vetores buscando economizar memória .

Nas tabelas subseqüentes encontramos os resultados , isto é , o número de iterações necessárias para atingirmos uma precisão **eps** ou , no caso de não atingirmos tal precisão em um dado número máximo de iterações , a palavra **limax** .

Para o processo iterativo de Newton , comparamos aproximações sucessivas até que elas estivessem próximas . Isto é , até que a diferença entre as soluções de duas iterações consecutivas fosse menor que eps em todos os nós da partição x_i , $i = 0,1,\ldots,m$. No primeiro conjunto de testes tomamos como aproximação inicial $\varphi_0 \equiv 1$. Para todos eles fizemos m = 80 , **eps** = 10^{-6}, **limax** = 200 .

Nas tabelas abaixo , **CGNE** se refere ao método dos Gradientes Conjugados aplicado às equações normais , **CGS** ao método dos Gradientes Conjugados Quadrático e **PCGS** ao método CGS com pré-condicionamento pela parte tridiagonal da matriz .

Tabela 5.1 : $\gamma = 0^0$, $\delta = 45^0$, $B_0 = 1$, λ analítica

Iteração Newton	1	2	3	4	5	6
CGNE	limax	limax	limax	limax	limax	limax
CGS	47	limax	47	46	46	-

Tabela 5.2 : $\gamma = 0^0$, $\delta = 45^0$, $B_0 = 1$, λ numérica

Iteração Newton	1	2	3	4	5	6	7
CGNE	limax	limax	limax	limax	limax	limax	23
CGS	81	limax	limax	53	limax	09	-
PCGS	04	04	04	04	01	-	-

Tabela 5.3 : $\gamma = 0^0$, $\delta = 45^0$, $B_0 = 1000$, λ analítica

Iteração Newton	1	2	3	4
CGNE	88	83	82	83
CGS	25	19	14	05

Tabela 5.4 : $\gamma = 0^0$, $\delta = 45^0$, $B_0 = 1000$, λ numérica

Iteração Newton	1	2	3	4
CGNE	limax	95	88	84
CGS	44	80	46	05
PCGS	06	04	04	03

Tabela 5.5 : $\gamma = 45^0$, $\delta = 90^0$, $B_0 = 10$, λ analítica

Iteração Newton	1	2	3	4
CGNE	11max	11max	11max	11max
CGS	62	43	116	46

Tabela 5.6 : $\gamma = 45^0$, $\delta = 90^0$, $B_0 = 10$, λ numérica

Iteração Newton	1	2	3	4	5	6	7	8	9
CGNE	11max	11max	11max	11max	11max	11max	11max	11max	11max
CGS	11max	65	61	46	-	-	-	-	-
PCGS	05	04	04	02	-	-	-	-	-

A primeira observação a ser feita se refere ao condicionamento dos sistemas lineares . Estes têm matrizes com número de condição da ordem de 10^3. Isto torna o método CG , aplicado às equações normais , pouco competitivo , como pudemos observar nas tabelas acima em que vê-se que , na maioria das vezes , a propriedade de terminação finita do método , convergência em n passos , falha . Podemos observar também que para um mesmo conjunto de dados a inclusão do multiplicador λ como incógnita causa um retardamento na convergência . Seja através do aumento do número de iterações para resolver os sistemas lineares , seja pelo aumento do número de iterações do método de Newton . Mas estas dificuldades podem ser bem contornadas com o uso de um pré-condicionador do tipo tridiagonal no algoritmo CGS . Isto se deve basicamente à estrutura da matriz (5.27), tendo grande quantidade de informações na sua parte tridiagonal .

Tendo percebido uma maior eficiência do método CGS , vamos usá-lo juntamente com uma mudança na aproximação inicial de Newton , buscando acelerar nossos resultados . Para isto vamos considerar os seguintes dados :

teste 1 : $\gamma = 0^0$, $\delta = 45^0$, $B_0 = 10$.
teste 2 : $\gamma = 0^0$, $\delta = 45^0$, $B_0 = 100$.
teste 3 : $\gamma = 0^0$, $\delta = 45^0$, $B_0 = 1000$.
teste 4 : $\gamma = 45^0$, $\delta = 90^0$, $B_0 = 10$.
teste 5 : $\gamma = 90^0$, $\delta = 90^0$, $B_0 = 10$.

A seguir , encontramos os resultados do método CGS com $\varphi_0 \equiv 1$ e λ calculado analiticamente , para os testes 1 a 5 :

Iteração Newton	1	2	3	4	5
teste 1	42	Ilmax	88	84	23
teste 2	41	46	41	41	01
teste 3	25	19	14	05	-
teste 4	62	43	42	116	-
teste 5	75	45	45	45	01

Tabela 5.7

Se usarmos os resultados do teste 1 como aproximação inicial para o teste 2 , os resultados do teste 2 como aproximação para o teste 3 , os resultados do teste 4 como aproximação inicial para o teste 5 e vice-versa , obtemos :

Iteração Newton	1	2	3	4
teste 2	47	41	40	08
teste 3	25	19	14	05
teste 4	48	43	44	40
teste 5	44	45	46	52

Tabela 5.8

Note que esta alteração na escolha das aproximações iniciais causou uma melhoria significativa nos resultados .

Para encerrar este conjunto de testes , usamos novamente CGS com $\varphi_0 \equiv 1$, mas com um limite máximo de 20 iterações para cada sistema linear . Veja os testes que obtiveram os melhores resultados :

Iteração Newton	1	2	3	4	5
teste 2	20	20	20	20	18
teste 3	20	19	14	05	-

Tabela 5.9

No gráfico 5.1 encontramos as soluções para os seguintes dados :

- $\gamma = 0^0$
- $\delta = 45^0$
- $B_0 = 1$ (linha continua)
- $B_0 = 1000$ (linha tracejada) .

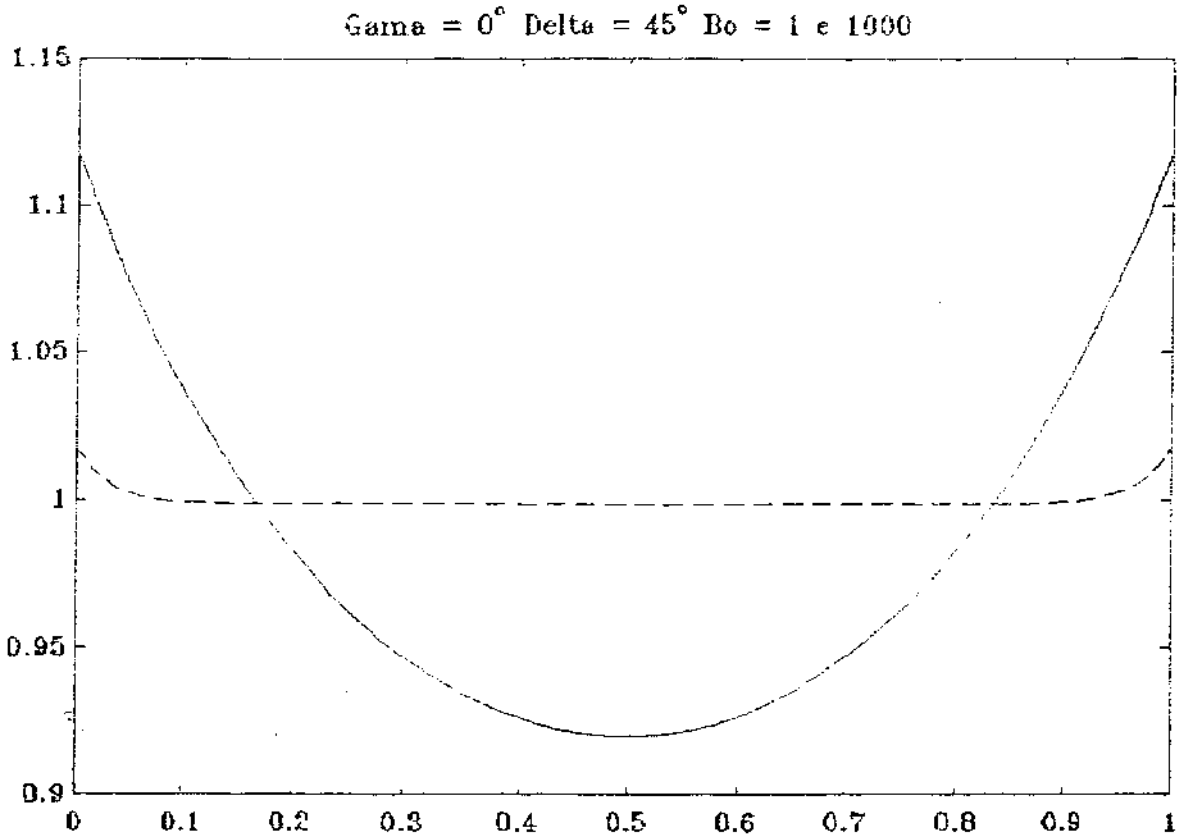

Gráfico 5.1

No gráfico 5.2 encontramos as soluções para os seguintes dados :

- $\delta = 90^0$
- $B_0 = 10$
- $\gamma = 45^0$ (linha continua)
- $\gamma = 60^0$ (linha tracejada) .

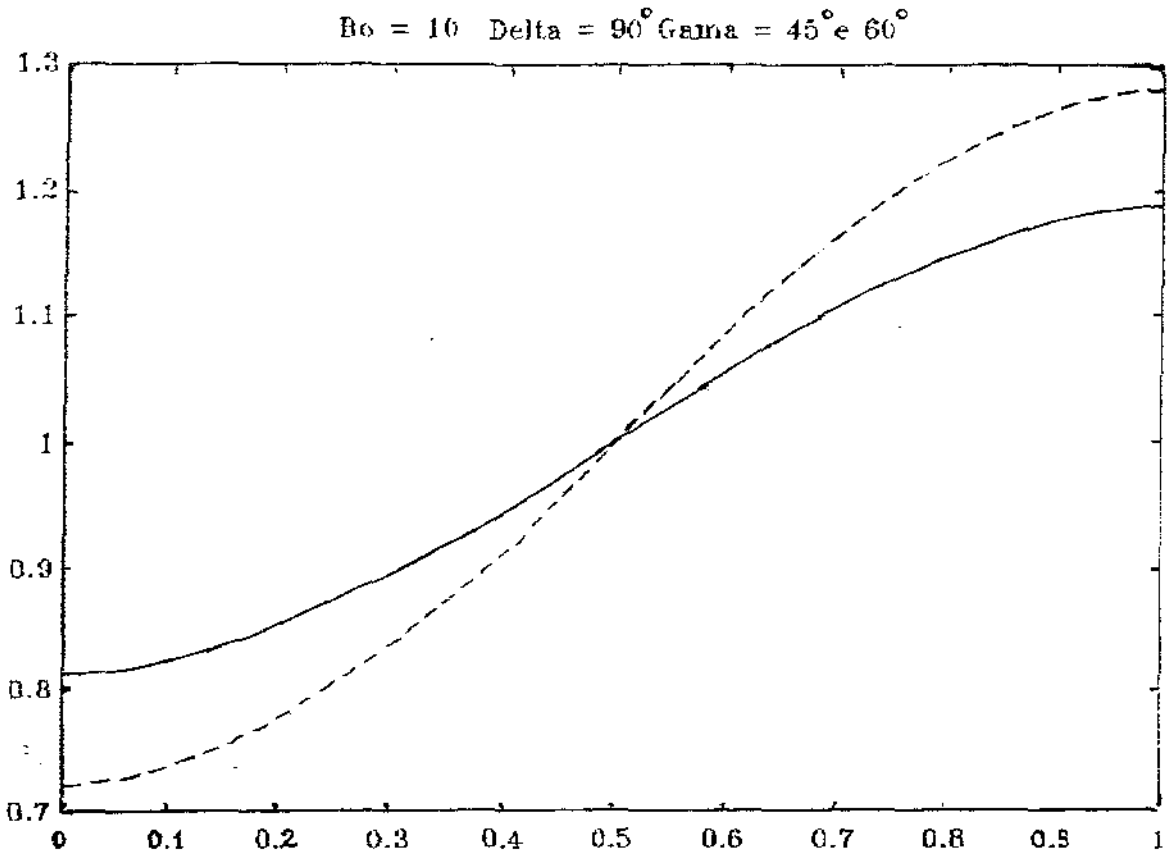

Gráfico 5.2

5.5 - CAPILAR CILÍNDRICO TRIDIMENSIONAL COM SIMETRIA AXIAL

5.5.1 - COLOCAÇÃO DO PROBLEMA

Vamos considerar um líquido contido num cilindro tridimensional V colocado em um campo gravitacional de forma que a direção do eixo de simetria do cilindro coincida com a direção do campo gravitacional . Ainda mais , consideraremos a situação na qual o cilindro gira em torno de seu eixo de simetria com velocidade angular ω . Seja o cilindro dado por

$$V = \left\{ (x,y) \in \mathbb{R}^2 \ / \ x^2 + y^2 < R^2, \ z > 0 \right\}$$

onde R denota o raio do cilindro .

Devido à simetria axial , o problema pode ser escrito em coordenadas cilíndricas $0 \leq r \leq R$ e $z \geq 0$.

A velocidade angular ω fará aparecer uma força adicional no experimento e , portanto , um novo termo de pressão dado por

$$\frac{1}{2} \rho \omega^2 r^2 .$$

Para este sistema de coordenadas a curvatura média pode ser escrita como

$$H = \frac{1}{2r} \left[\frac{r \varphi'}{(1 + (\varphi')^2)^{1/2}} \right]' ,$$

onde o símbolo " ' " denota a derivada com relação à variável r .

Impondo a condição de que o ângulo de contato δ permaneça constante , obtemos a seguinte condição de contorno para r = R

$$\varphi'(R) = \text{cotan}(\delta) .$$

Como φ é simétrica , é uma função par . Logo φ' é impar . Portanto , para $r = 0$, continuidade e simetria implicam em

$$\varphi'(0) = 0 \ .$$

Como o multiplicador λ é também uma incógnita do problema , devemos ter uma condição adicional sobre o volume , isto é

$$\int_0^{2\pi} \int_0^R r\ \varphi(r)\ dr = 2\Pi \int_0^R r\ \varphi(r)\ dr = | V |$$

que é o volume constante (dado) do liquido .

Muitas vezes é vantajoso escrever a equação diferencial de forma adimensional . Usando o comprimento R para esta finalidade , temos o seguinte problema adimensionalisado :

- Encontrar uma função $r \to \varphi(r)$, $0 \le r \le 1$ e uma constante λ tal que

$$-\frac{1}{r}\left[\frac{r\ \varphi'}{(\ 1 + (\ \varphi')^2\ )^{1/2}}\right]' + B_0\varphi = We\ r^2 + \lambda$$

$$\varphi'(0) = 0 \ , \qquad \varphi'(1) = \text{cotan}(\delta)$$

$$2\Pi \int_0^1 r\ \varphi(r)\ dr = Vc \quad , \qquad Vc = R^{-3}\ |\ V|$$

com

$$B_0 = \frac{\rho g R^2}{\sigma} \qquad \text{Número de ligação} \ ,$$

$$We = \frac{\rho \omega^2 R^3}{2\sigma} \qquad \text{Número de Weber} \ .$$

Se multiplicarmos a equação diferencial por r e a integrarmos entre 0 e 1 podemos encontrar , usando tambem as condições de fronteira , analiticamente o valor de λ

$$\lambda = \frac{Bo.Ve}{\pi} - \frac{1}{2} We - 2\cos(\delta) \quad . \tag{5.28}$$

5.5.2 - LINEARIZAÇÃO

Como o problema é não-linear , usaremos o Método de Newton para linearizá-lo , isto é , queremos resolver $F(\varphi) = 0$, onde

$$F(\varphi) = -\frac{1}{r}\left[\frac{r\,\varphi'}{(1+(\varphi')^2)^{1/2}}\right]' + B_0\varphi - We\,r^2 - \lambda \ .$$

A partir de uma aproximação inicial φ_0, geramos uma sequência $\varphi_1, \varphi_2, \ldots , \varphi_n$ através de

$$F'(\varphi_n)(\varphi_{n+1} - \varphi_n) = -F(\varphi_n) \tag{5.29}$$

onde $F'(\varphi)$ denota a derivada forte do operador F no ponto φ .

Vamos calcular , primeiro , a derivada de Gateaux de F no ponto φ e na direção de ψ .

$$F'(\varphi)[\psi] = \lim_{\varepsilon\to 0} \frac{F(\varphi + \varepsilon\psi) - F(\varphi)}{\varepsilon} \quad .$$

(5.30)

$$F'(\varphi)[\psi] = -\frac{1}{r}\left[\frac{r\,\psi'}{(1+(\varphi')^2)^{3/2}}\right]' + B_0\psi$$

Como (5.30) está de acordo com as hipóteses do teorema 4.1 , concluimos que a derivada forte de F existe e coincide com a derivada fraca . Vamos , portanto , utilizá-la para gerar a seqüência de Newton através de (5.29) .

Pode ser mostrado que o termo geral φ_{n+1} da seqüência satisfaz o seguinte problema de fronteira linear

$$- \frac{1}{r}\left[\frac{r\,\varphi'_{n+1}}{(\,1+(\,\varphi'_{n}\,)^2\,)^{3/2}}\right]' + B_0\varphi_{n+1} - \lambda = We\, r^2 +$$

$$+ \frac{1}{r}\left[\frac{r\,(\,\varphi'_{n}\,)^3}{(\,1+(\,\varphi'\,)^2_{n}\,)^{3/2}}\right]' \quad \text{em } (0,1) \quad , \tag{5.31}$$

$$\varphi'_{n+1}(0) = 0 \quad , \qquad \varphi'_{n+1}(1) = \operatorname{cotan}(\delta) \quad ,$$

$$2\Pi.\int_0^1 r\,\varphi_{n+1}(r)\,dr = V_c \quad .$$

5.5.3 - DISCRETIZAÇÃO E RESULTADOS NUMÉRICOS

Novamente resolvemos o problema (5.31) através de uma discretização por Diferenças Finitas Centrais . Este método de discretização dá origem a um sistema de equações algébricas com a seguinte estrutura :

$$\begin{bmatrix} x & x & & & & x \\ x & x & x & & & x \\ & x & x & x & & x \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & x & x & x \\ & & & & x & x \\ x & x & \cdots & x & x & 0 \end{bmatrix} \begin{bmatrix} (\varphi_{n+1})_0 \\ \vdots \\ (\varphi_{n+1})_m \\ \lambda \end{bmatrix} = \begin{bmatrix} x \\ x \\ x \\ \vdots \\ x \\ x \\ x \end{bmatrix} \quad . \tag{5.32}$$

Usando as condições de contorno , eliminamos λ do sistema através de (5.28) .

Inicialmente tomamos $\varphi_0 \equiv 1$ como aproximação inicial para o Método Inexato de Newton . As tabelas 5.10 e 5.11 contém os resultados - número de iterações - para os sistemas lineares em cada iteração de Newton . Para estes testes , fizemos m = 80 (número de partições da discretização) , eps = 10^{-6}, limax = 200 . Nas tabelas abaixo , PCGNE se refere ao método CGNE com pré-condicionamento pela diagonal da matriz .

Tabela5.10 : We = 200 , $\delta = 45^0$, $B_0 = 1000$, $V_c = \Pi$

Iteração Newton	1	2	3	4
CGNE	limax	limax	limax	limax
CGS	51	61	27	11
PCGNE	limax	limax	164	122

Tabela5.11: We = 800 , $\delta = 45^0$, $B_0 = 1000$, $V_c = \Pi$

Iteração Newton	1	2	3	4
CGNE	limax	limax	limax	limax
CGS	51	46	25	11
PCGNE	limax	182	115	49

Repare que o método PCGNE apresenta alguma melhoria em relação ao método CG , mas não consegue ter a mesma eficiência do método CGS .

Vamos , agora , limitar o número de iterações do algoritmo CGS . Na tabela 5.12 , limax = 25 e na tabela 5.13 limax = 30 . Teste 1 se refere aos mesmos dados usados pela tabela 5.10 e teste 2 aos dados usados para gerar a tabela 5.11 .

Iteração Newton	1	2	3	4	5
teste 1	25	25	25	25	03
teste 2	25	25	25	25	06

Tabela 5.12

Iteração Newton	1	2	3	4	5
teste 1	30	30	30	28	-
teste 2	30	30	30	10	-

Tabela 5.13

Repare que se somarmos o número total de iterações do método CGS nas tabelas 5.10 e 5.11 e comparamos com as tabelas 5.12 e 5.13 , veremos que obtivemos uma certa economia na quantidade de iterações .

No gráfico 5.3 encontramos a solução para os dados abaixo :

- $B_0 = 1000$
- $\delta = 45^0$
- $V_c = \Pi$
- We = 200 (linha contínua)
- We = 800 (linha tracejada) .

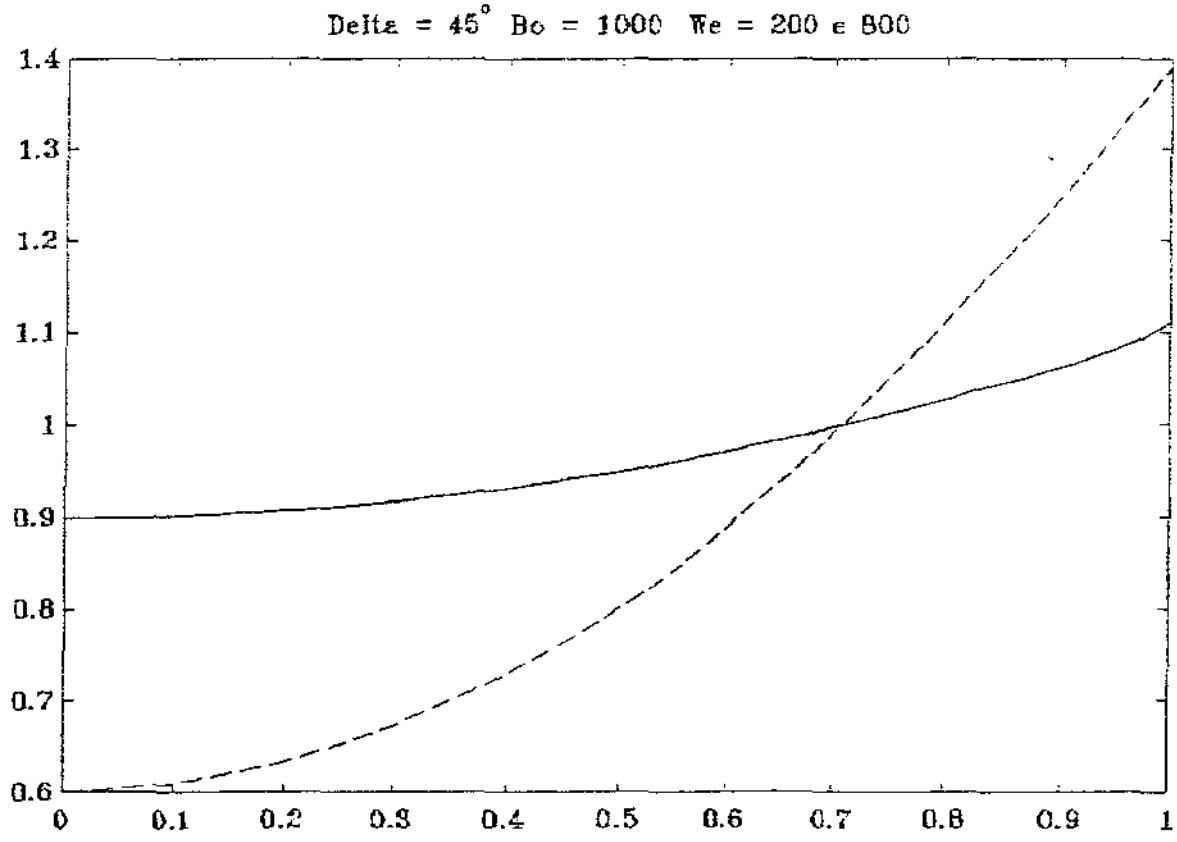

Gráfico 5.3

CONCLUSÕES

O armazenamento das matrizes , usado nos métodos (veja as explicações no Apêndice) , mostrou-se bastante simples e eficiente .

Todos os métodos testados na resolução do problema de difusão-convecção com as restrições quanto à escolha dos parâmetros obedecendo a referência [14] , apresentaram bons resultados . Em todos eles , a propriedade de terminação em *n* (ordem do sistema linear) passos , no máximo , foi alcançada com sobras . Ainda mais , a relação entre o número de iterações efetivamente usadas para a resolução dos sistemas lineares e a ordem das matrizes decresceu quando aumentamos a dimensão do sistema . Estes fatos acima fazem-nos concluir que os métodos desenvolvidos neste trabalho são próprios para sistemas lineares esparsos e de grande porte .

Nos problemas de capilaridade , os métodos testados não obtiveram o mesmo desempenho alcançado no problema de difusão-convecção . Neles, o método CGNE não se mostrou satisfatório nem mesmo no caso em que utilizamos uma versão com pré-condicionador diagonal (PCGNE) , isto é, com um reescalamento da matriz do sistema pela sua diagonal . Este comportamento talvez se explique pela maior complexidade dos problemas de capilaridade , principalmente , por sua não-linearidade .

A opção de linearizar os problemas do capítulo 5 antes de discretizá-los , além de mais elegante matematicamente , diminuiu os cálculos e facilitou o trabalho de resolvê-los .

A estratégia de limitar a quantidade de iterações dos métodos para sistemas lineares em cada iteração do método de linearização apresentou bons resultados .

O método CGS , principalmente na forma pré-condicionada (PCGS), mostrou-se como uma ferramenta bastante eficiente para a resolução de sistemas lineares esparsos . A sua velocidade , entretanto , depende fortemente da qualidade do pré-condicionador utilizado . Por esta razão , é importante desenvolver-se códigos para fatorações incompletas para matrizes mais complicadas e que permitiriam a resolução de outros tipos mais amplos de problemas .

O método CGS apresentou , em algumas situações , convergência irregular . Em particular , quando tomávamos uma aproximação inicial próxima da solução do problema , caso das iterações finais dos métodos de linearização . Para contornar este tipo de dificuldade , Van der Vorst (referência [26]) propôs o método BI-CGSTAB , Gradientes Bi-Conjugados Estabilizado . Segundo o autor , trata-se de uma outra variante dos Gradientes Bi-Conjugados (BI-CG) com convergência mais suave , mas mantendo a rapidez do método CGS nos melhores casos . Desta forma , nosso passo seguinte será estudá-lo e testá-lo .

APÊNDICE - ROTINAS PARA O MÉTODO DE DIREÇÕES CONJUGADAS : SISTEMAS LINEARES ESPARSOS

INTRODUÇÃO

Apresentamos , a seguir , o código em Fortran 77 de alguns dos principais algoritmos , incluídos neste trabalho , para a resolução de sistemas lineares esparsos . Dos métodos propriamente , bem como das rotinas que trabalham com a matriz do sistema da forma com que foi armazenada .

Dada a matriz A do sistema , fizemos seu armazenamento em 3 vetores :

ILIN e JCOL - vetores de inteiros ;

A - vetor de reais com dupla precisão .

Armazenamos em *A* todos os elementos não-nulos da matriz , sem a necessidade de uma ordenação qualquer , e em ILIN e JCOL suas respectivas linhas e colunas . Por exemplo :

Se na matriz do sistema tivermos $a_{13} = 5.2$ e este for o k-ésimo elemento da matriz a ser armazenado , faremos

$$ILIN(k) = 1$$
$$JCOL(k) = 3$$
$$A(k) = 5.2 \quad .$$

É importante notar que este tipo de estocagem não prevê o aparecimento de novos elementos na matriz durante a resolução do sistema , sendo portanto favorável ao tipo de métodos que escolhemos. Além disso , não necessita de nenhuma ordenação para o armazenamento da matriz , sendo bastante conveniente para a discretização de equações diferenciais porque permite a construção da matriz na ordem em que tivermos maior facilidade : discretização de pontos interiores , condições de contorno , condição de continuidade em interfaces , etc .

ROTINAS DOS PRINCIPAIS MÉTODOS DESENVOLVIDOS

Rotina : SCG

Objetivo : Método dos Gradientes Conjugados para matrizes simétricas e positiva-definidas .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

X : aproximação inicial para o vetor solução do sistema .

B : vetor do lado direito do sistema .

NDIM : dimensão do sistema .

NELEM : número de elementos não-nulos da matriz A .

R , VET : vetores de trabalho .

EPS : precisão desejada .

LIMAX : número máximo de iterações para o método .

Variáveis de saída :

X : vetor solução do sistema linear .

ITER : número de iterações realizadas pelo método .

```
      SUBROUTINE SCG(ILIN,JCOL,A,X,B,NDIM,NELEM,R,VET)
C
      REAL*8  A(*),X(*),B(*),R(*),EPS,XNEW,VET(*)
      REAL*8 STEP,BETA,GAMA,ERRO,AUX,ALFA,ERROR
      INTEGER*2 ILIN(*),JCOL(*),NDIM,NELEM
C
      COMMON /CONVERGENCIA/ EPS,LIMAX,ITER
C
C     INICIO DO PROCESSO ITERATIVO
C
      ITER = 0
C
      CALL RESIDUO(ILIN,JCOL,A,NELEM,NDIM,X,B,R)
C
      CALL PINNER(R,R,NDIM,ALFA)
```

```
C
      DO 5 I=1,NDIM
           VET(I) = R(I)
5     CONTINUE
C
20    ITER = ITER + 1
C
C     CALCULO DO        STEP = < R , R > / < A.P , P >
C
      CALL PXTAX(ILIN,JCOL,A,NELEM,VET,GAMA)
C
      STEP = ALFA / GAMA
C
C     NOVA APROXIMACAO  e  TESTE DE CONVERGENCIA
C
      ERRO = 0.D0
      DO 30 I=1,NDIM
           XNEW = X(I) + STEP*VET(I)
           AUX = DABS(XNEW-X(I))
           IF(ERRO.LT.AUX) ERRO = AUX
           X(I) = XNEW
30    CONTINUE
C
C     CALCULO DO  RESIDUO
C
      CALL RESIDUO(ILIN,JCOL,A,NELEM,NDIM,X,B,R)
C
      CALL SUPNOR(R,NDIM,ERROR)
C
C     CALCULO DA NOVA  DIRECAO  A-CONJUGADA
C
      BETA = ALFA

      CALL PINNER(R,R,NDIM,ALFA)
      BETA = ALFA / BETA
C
C
```

```
      DO 35 I=1,NDIM
           VET(I) = R(I) + BETA*VET(I)
35    CONTINUE
C
C
      WRITE(*,200) ITER,ERRO,ERROR
200   FORMAT(//,T5,'ITERACAO      :   ',I3.3,/,T5,
     *             'E R R O       :   ',E12.5,/,T5,
     *             'RESIDUO       :   ',E12.5)
C
      IF(ERRO.LE.EPS.AND.ERROR.LE.EPS) GO TO 70
      IF(ITER.LE.LIMAX) GO TO 20
C
70    RETURN
      END
C
C     ------------------------------------------------------------
```

Rotina : CONJGRD

Objetivo : Método dos Gradientes Conjugados - Equações Normais .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

X : aproximação inicial para o vetor solução do sistema .

B : vetor do lado direito do sistema .

NDIM : dimensão do sistema .

NELEM : número de elementos não-nulos da matriz A .

R , WORK , VET : vetores de trabalho .

EPS : precisão desejada .

LIMAX : número máximo de iterações para o método .

Variáveis de saída :

X : vetor solução do sistema linear .

ITER : número de iterações realizadas pelo método .

```
      SUBROUTINE CONJGRD(ILIN,JCOL,A,X,B,NDIM,NELEM,R,WORK,VET)
C
      REAL*8  A(*),X(*),B(*),R(*),EPS,XNEW,WORK(*),VET(*)
      REAL*8 STEP,BETA,GAMA,ERRO,AUX,ALFA,ERROR
      INTEGER*2 ILIN(*),JCOL(*),NDIM,NELEM
C
      COMMON /CONVERGENCIA/ EPS,LIMAX,ITER
C
C     INICIO DO PROCESSO ITERATIVO
C
      ITER = 0
      CALL GRADIENTE(ILIN,JCOL,A,NELEM,NDIM,NDIM,WORK,X,B,R)
      CALL PINNER(R,R,NDIM,ALFA)
C
      DO 5 I=1,NDIM
          VET(I) = R(I)
5     CONTINUE
C
20    ITER = ITER + 1
C
C     CALCULO DO        STEP = < R , R > / < A.R , A.R >
C
      CALL PRODAX(ILIN,JCOL,A,NELEM,NDIM,VET,WORK)
      CALL PINNER(WORK,WORK,NDIM,GAMA)
      STEP = ALFA / GAMA
C
C     NOVA APROXIMACAO  e  TESTE DE CONVERGENCIA
C
      ERRO = 0.D0
      DO 30 I=1,NDIM
          XNEW = X(I) + STEP*VET(I)
          AUX = DABS(XNEW-X(I))
          IF(ERRO.LT.AUX) ERRO = AUX
          X(I) = XNEW
30    CONTINUE
C
```

```
C          CALCULO DO  RESIDUO
C
           CALL GRADIENTE(ILIN,JCOL,A,NELEM,NDIM,NDIM,WORK,X,B,R)
           CALL SUPNOR(R,NDIM,ERROR)
C
C          CALCULO DA NOVA  DIRECAO  A-CONJUGADA
C
           BETA = ALFA
           CALL PINNER(R,R,NDIM,ALFA)
           BETA = ALFA / BETA
C
           DO 35 I=1,NDIM
                VET(I) = R(I) + BETA*VET(I)
35         CONTINUE
C
           WRITE(*,200) ITER,ERRO,ERROR
200        FORMAT(//,T5,'ITERACAO      :    ',I3.3,/,T5,
     *                  'E R R O      :    ',E12.5,/,T5,
     *                  'RESIDUO      :    ',E12.5)
C
           IF(ERRO.LE.EPS.AND.ERROR.LE.EPS) GO TO 70
           IF(ITER.LE.LIMAX) GO TO 20
C
70         RETURN
           END
C
C    ---------------------------------------------------------------
```

Rotina : SCGCOND

Objetivo : Método dos Gradientes Conjugados para matrizes simétricas e positiva-definidas com pré-condicionamento tridiagonal por fatoração incompleta de Cholesky .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

X : aproximação inicial para o vetor solução do sistema .

B : vetor do lado direito do sistema .

NDIM : dimensão do sistema .

NELEM : número de elementos não-nulos da matriz A .

R , VET , Z , DIAG , SUB : vetores de trabalho .

EPS : precisão desejada .

LIMAX : número máximo de iterações para o método .

Variáveis de saída :

X : vetor solução do sistema linear .

ITER : número de iterações realizadas pelo método .

```
      SUBROUTINE SCGCOND(ILIN,JCOL,A,X,B,NDIM,NELEM,R,VET,Z,DIAG,SUB)
C
      REAL*8  A(*),X(*),B(*),R(*),EPS,XNEW,VET(*)
      REAL*8  DIAG(*),SUB(*),Z(*)
      REAL*8 STEP,BETA,GAMA,ERRO,AUX,ALFA,ERROR
      INTEGER*2 ILIN(*),JCOL(*),NDIM,NELEM
C
      COMMON /CONVERGENCIA/ EPS,LIMAX,ITER
C
C     BUSCA DA PARTE TRIDIAGONAL DA MATRIZ
C
      CALL ARMAZENA(ILIN,JCOL,A,NELEM,DIAG,SUB,NDIM)
C
C     DECOMPOSIÇÃO CHOLESKY DA PARTE TRIDIAGONAL
C
      CALL CHOLTRI(DIAG,SUB,NDIM)
C
C     INICIO DO PROCESSO ITERATIVO
C
      ITER = 0
      CALL RESIDUO(ILIN,JCOL,A,NELEM,NDIM,X,B,R)
      CALL SOLVE(DIAG,SUB,R,Z,NDIM)
      CALL PINNER(R,Z,NDIM,ALFA)
```

```
C
      DO 5 I=1,NDIM
            VET(I) = Z(I)
5     CONTINUE
C
20    ITER = ITER + 1
C
C     CALCULO DO   STEP = < R , Z > / < A.P , P >
C
      CALL PXTAX(ILIN,JCOL,A,NELEM,VET,GAMA)
      STEP = ALFA / GAMA
C
C     NOVA APROXIMACAO  e  TESTE DE CONVERGENCIA
C
      ERRO = 0.D0
      DO 30 I=1,NDIM
            XNEW = X(I) + STEP*VET(I)
            AUX = DABS(XNEW-X(I))
            IF(ERRO.LT.AUX) ERRO = AUX
            X(I) = XNEW
30    CONTINUE
C
C     CALCULO DO  RESIDUO
C
      CALL RESIDUO(ILIN,JCOL,A,NELEM,NDIM,X,B,R)
      CALL SOLVE(DIAG,SUB,R,Z,NDIM)
      CALL SUPNOR(R,NDIM,ERROR)
C
C     CALCULO DA NOVA  DIRECAO  A-CONJUGADA
C
      BETA = ALFA
      CALL PINNER(R,Z,NDIM,ALFA)
      BETA = ALFA / BETA
C
      DO 35 I=1,NDIM
            VET(I) = Z(I) + BETA*VET(I)
35    CONTINUE
C
```

```
      WRITE(*,200) ITER,ERRO,ERROR
200   FORMAT(//,T5,'ITERACAO      :   ',I3.3,/,T5,
     *             'E R R O      :   ',E12.5,/,T5,
     *             'RESIDUO      :   ',E12.5)
C
      IF(ERRO.LE.EPS.AND.ERROR.LE.EPS) GO TO 70
      IF(ITER.LE.LIMAX) GO TO 20
C
70    RETURN
      END
C     ------------------------------------------------------------------
```

Rotina : PCTDCGS

Objetivo : Método dos Gradientes Conjugados Quadrático com pré-condicionamento tridiagonal por fatoração LU incompleta .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

X : aproximação inicial para o vetor solução do sistema .

B : vetor do lado direito do sistema .

MDIM : dimensão do sistema .

NELEM : número de elementos não-nulos da matriz **A** .

IFLAG : 'S' se desejar pré-condicionamento , 'N' caso contrário

R , RO , Q , P , DIAG , SUB , SUP : vetores de trabalho .

EPS : precisão desejada .

LIMAX : número máximo de iterações para o método .

Variáveis de saída :

X : vetor solução do sistema linear .

ITER : número de iterações realizadas pelo método .

```
      SUBROUTINE PCTDCGS(ILIN,JCOL,A,X,B,MDIM,NELEM,IFLAG,
     *                   R,R0,Q,P,DIAG,SUB,SUP)
C
      REAL*8  A(*),X(*),B(*)
      REAL*8  Q(*),P(*),R(*),R0(*)
      REAL*8 DIAG(*),SUP(*),SUB(*)
      REAL*8 ALFA,BETA,SIGMA,GAMA,ERRO,AUX,ERROR,EPS,XNEW
      INTEGER*2 ILIN(*),JCOL(*),MDIM,NELEM
      CHARACTER IFLAG*1
C
      COMMON  / CONVERGENCIA /  EPS,LIMAX,ITER
C
      IF(IFLAG.EQ.'S') THEN
C
C     BUSCA DA PARTE TRIDIAGONAL DA MATRIZ
C
          CALL TRIDIAG(ILIN,JCOL,A,NELEM,MDIM,DIAG,SUB,SUP)
C
C     DECOMPOSICAO LU DA PARTE TRIDIAGONAL DA MATRIZ
C
          CALL TRIDLU(DIAG,SUP,SUB,MDIM)
      END IF
C
C     INICIO DO PROCESSO ITERATIVO
C
      ITER = 0
      GAMA = 1.D0
C
      CALL RESIDUO(ILIN,JCOL,A,NELEM,MDIM,X,B,R)
      IF(IFLAG.EQ.'S') CALL SOLVTRID(DIAG,SUP,SUB,R,R,MDIM)
C
      DO 10 I = 1,MDIM
          Q(I) = 0.D0
          P(I) = 0.D0
          R0(I) = R(I)
10    CONTINUE
C
```

```
20        ITER = ITER + 1
          CALL PINNER(R,R0,MDIM,BETA)
          ALFA = BETA
          BETA = BETA / GAMA
C
          DO 25 I = 1,MDIM
               R(I) = R(I) + BETA*Q(I)
               P(I) = R(I) + BETA*( Q(I) + BETA*P(I) )
25        CONTINUE
C
          CALL PRODAX(ILIN,JCOL,A,NELEM,MDIM,P,Q)
C
          IF(IFLAG.EQ.'S') CALL SOLVTRID(DIAG,SUP,SUB,Q,Q,MDIM)
C
          CALL PINNER(R0,Q,MDIM,SIGMA)
          GAMA = ALFA
          ALFA = ALFA / SIGMA
C
C         NOVA APROXIMACAO  e  TESTE DE CONVERGENCIA
C
          ERRO = 0.D0

          DO 30 I = 1,MDIM
               Q(I) = R(I) - ALFA*Q(I)
               XNEW = X(I) + ALFA*( R(I) + Q(I) )
               AUX = DABS(XNEW-X(I))
               IF(ERRO.LT.AUX) ERRO = AUX
               X(I) = XNEW
30        CONTINUE
C
C         CALCULO DO  RESIDUO
C
          CALL RESIDUO(ILIN,JCOL,A,NELEM,MDIM,X,B,R)
          IF(IFLAG.EQ.'S') CALL SOLVTRID(DIAG,SUP,SUB,R,R,MDIM)
C
          CALL SUPNOR(R,MDIM,ERROR)
C
```

```
      WRITE(*,200) ITER,ERRO,ERROR
200   FORMAT(//,T5,'ITERACAO      :   ',I3.3,/,T5,
     *             'E R R O      :   ',E12.5,/,T5,
     *             'RESIDUO      :   ',E12.5)
C
      IF(ERRO.LE.EPS.AND.ERROR.LE.EPS) RETURN
      IF(ITER.LE.LIMAX) GO TO 20
C
C
      RETURN
      END
C     ------------------------------------------------------------------
```

Rotina : LSCGPD

Objetivo : Método dos Gradientes Conjugados - Equações Normais - com pré-condicionamento diagonal .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

X : aproximação inicial para o vetor solução do sistema .

B : vetor do lado direito do sistema .

MLIM : número de linhas do sistema .

NCOL : número de colunas do sistema .

NELEM : número de elementos não-nulos da matriz A .

R , VET , WORK , DIAG : vetores de trabalho .

EPS : precisão desejada .

LIMAX : número máximo de iterações para o método .

Variáveis de saída :

X : vetor solução do sistema linear .

ITER : número de iterações realizadas pelo método .

```
      SUBROUTINE LSCGPD(ILIN,JCOL,A,X,B,MLIN,NCOL,NELEM,
     *                  R,VET,WORK,DIAG)
C
      REAL*8  A(*),X(*),B(*)
      REAL*8 STEP,BETA,GAMA,ERRO,AUX,ALFA,ERROR,EPS,XNEW
      REAL*8 WORK(*),VET(*),R(*),DIAG(*)
      INTEGER*2 ILIN(*),JCOL(*),MLIN,NCOL,NELEM
C
      COMMON /CONVERGENCIA/ EPS,LIMAX,ITER
C
C     BUSCA DA PARTE DIAGONAL DE A^tA
C
      CALL DIAGATA(JCOL,A,NELEM,NCOL,DIAG)
C
C     INICIO DO PROCESSO ITERATIVO
C
      ITER = 0
      CALL GRADIENTE(ILIN,JCOL,A,NELEM,MLIN,NCOL,WORK,X,B,R)
      DO 10 K = 1,NCOL
         WORK(K) = R(K) / DIAG(K)
         VET(K) = WORK(K)
10    CONTINUE
C
      CALL PINNER(R,WORK,NCOL,ALFA)
20    ITER = ITER + 1
C
C     CALCULO DO   STEP = < R , R > / < A.R , A.R >
C
      CALL PRODAX(ILIN,JCOL,A,NELEM,MLIN,VET,WORK)
      CALL PINNER(WORK,WORK,MLIN,GAMA)
      STEP = ALFA / GAMA
C
C     NOVA APROXIMACAO  e  TESTE DE CONVERGENCIA
C
      ERRO = 0.D0
```

```
      DO 25 I=1,NCOL
           XNEW = X(I) + STEP*VET(I)
           AUX = DABS(XNEW-X(I))
           IF(ERRO.LT.AUX) ERRO = AUX
           X(I) = XNEW
25    CONTINUE
C
C     CALCULO DO  RESIDUO
C
      CALL GRADIENTE(ILIN,JCOL,A,NELEM,MLIN,NCOL,WORK,X,B,R)
C
      DO 30 K = 1,NCOL
           WORK(K) = R(K) / DIAG(K)
30    CONTINUE
C
      CALL SUPNOR(R,NCOL,ERROR)
C
C     CALCULO DA NOVA  DIRECAO  A-CONJUGADA
C
      BETA = ALFA
      CALL PINNER(R,WORK,NCOL,ALFA)
      BETA = ALFA / BETA
C
      DO 35 I=1,NCOL
           VET(I) = WORK(I) + BETA*VET(I)
35    CONTINUE
C
      WRITE(*,200) ITER,ERRO,ERROR
200   FORMAT(//,T5,'ITERACAO      :   ',I3.3,/,T5,
     *           'E R R O       :   ',E12.5,/,T5,
     *           'RESIDUO       :   ',E12.5)
C
      IF(ERRO.LE.EPS.AND.ERROR.LE.EPS) GO TO 70
      IF(ITER.LE.LIMAX) GO TO 20
C
70    RETURN
      END
```

ROTINAS PARA MANIPULAÇÃO MATRICIAL

Apresentaremos , a seguir , as rotinas responsáveis pela manipulaçãc matricial dos nossos métodos iterativos para sistemas lineares com o armazenamento através de vetores . Os procedimentos mais importantes serão listados e os mais comuns , isto é , que não variam com o armazenamento , serão apenas citados .

Rotina : PRODAX

Objetivo : Cálculo do produto **y = Ax** , de uma matriz quadrada por um vetor .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

NELEM : número de elementos não-nulos da matriz **A** .

MLIN : número de linhas da matriz **A** .

X : vetor a ser multiplicado pela matriz .

Variável de saída :

Y : vetor solução da multiplicação .

```
      SUBROUTINE PRODAX(ILIN,JCOL,A,NELEM,MLIN,X,Y)
C
      REAL*8  A(*),X(*),Y(*)
      INTEGER*2 ILIN(*),JCOL(*),NELEM,MLIN
C
C     Procedimento para Calcular  :  Y = A.X
C
      DO 10 I=1,MLIN
         Y(I) = 0.D0
10    CONTINUE
C
      DO 20 K=1,NELEM
         Y(ILIN(K)) = Y(ILIN(K)) + A(K)*X(JCOL(K))
20    CONTINUE
```

```
C
      RETURN
      END
C     ------------------------------------------------------------------
```

Rotina : PATX

Objetivo : Cálculo do produto $y = A^t x$, de uma matriz quadrada por um vetor .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

NELEM : número de elementos não-nulos da matriz A .

NCOL : número de colunas da matriz A .

X : vetor a ser multiplicado pela matriz .

Variável de saída :

Y : vetor solução da multiplicação .

```
      SUBROUTINE PATX(ILIN,JCOL,A,NELEM,NCOL,X,Y)
C
      REAL*8  A(*),X(*),Y(*)
      INTEGER*2 ILIN(*),JCOL(*),NELEM,NCOL
C
C     Procedimento para Calcular        Y = A'.X
C
      DO 10 I=1,NCOL
         Y(I) = 0.D0
10    CONTINUE
C
      DO 20 K=1,NELEM
         Y(JCOL(K)) = Y(JCOL(K)) + A(K)*X(ILIN(K))
20    CONTINUE
C
      RETURN
      END
```

Rotina : PXTAX

Objetivo : Cálculo da forma quadrática $x^t Ax$.

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

NELEM : número de elementos não-nulos da matriz A .

X : vetor x .

Variável de saída :

XTAX : resultado do cáculo de $x^t Ax$.

```
      SUBROUTINE PXTAX(ILIN,JCOL,A,NELEM,X,XTAX)
C
      REAL*8 A(*),X(*),XTAX
      INTEGER*2 ILIN(*),JCOL(*),NELEM
C
C     Procedimento para Calcular   <A.X,X>  =  X'.A.X

      XTAX = 0.D0
      DO 20 K=1,NELEM
         XTAX = XTAX  +  A(K)*X(JCOL(K))*X(ILIN(K))
20    CONTINUE
C
      RETURN
      END
```

C ----------

Rotina : PINNER

Objetivo : Cálculo do produto interno $x^t y$.

Variáveis de entrada :

X : vetor x .

Y : vetor y .

NDIM : dimensão dos vetores .

Variável de saída :

Rotina : SUPNOR

Objetivo : Cálculo da norma do sup do vetor **x** .

Variáveis de entrada :

X : vetor **x** .

NDIM : dimensão do vetor .

Variável de saída :

DIST : norma do sup do vetor **x** .

Rotina : EUCLNOR

Objetivo : Cálculo da norma-2 do vetor **x** .

Variáveis de entrada :

X : vetor **x** .

NDIM : dimensão do vetor .

Variável de saída :

DIST : norma-2 do vetor **x** .

Rotina : RESIDUO

Objetivo : Cálculo do resíduo **R = b - Ax** .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

B : vetor do lado direito do sistema .

NELEM : número de elementos não-nulos da matriz **A** .

NDIM : dimensão da matriz **A** .

X : vetor **x** .

B : vetor **b** .

Variável de saída :

R : vetor com o resíduo .

```
      SUBROUTINE RESIDUO(ILIN,JCOL,A,NELEM,NDIM,X,B,R)
C
      REAL*8   A(*),X(*),R(*),B(*)
      INTEGER*2 ILIN(*),JCOL(*),NELEM,NDIM

C
C     Procedimento para Calcular o Residuo     R  =  B  -  A.X
C
      CALL PRODAX(ILIN,JCOL,A,NELEM,NDIM,X,R)
C
      DO 10 I=1,NDIM
         R(I) = B(I) - R(I)
10    CONTINUE

      RETURN
      END
C     ------------------------------------------------------------
```

Rotina : GRADIENTE

Objetivo : Cálculo do resíduo para as equações normais .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

NELEM : número de elementos não-nulos da matriz **A** .

MLIN : número de linhas da matriz **A** .

NCOL : número de colunas da matriz **A** .

X : vetor **x** .

B : vetor **b** .

WORK : vetor de trabalho .

Variáveis de saída :

R : vetor com o resíduo .

ITER : número de iterações realizadas pelo método .

```
      SUBROUTINE GRADIENTE(ILIN,JCOL,A,NELEM,MLIN,NCOL,WORK,X,B,R)
C
      REAL*8  A(*),X(*),R(*),B(*),WORK(*)
      INTEGER*2 ILIN(*),JCOL(*),NELEM,MLIN,NCOL
C
C     Procedimento para Calcular o Residuo    R  =  A'.( B  -  A.X )
C
      CALL PRODAX(ILIN,JCOL,A,NELEM,MLIN,X,WORK)
C
      DO 10 I=1,MLIN
         WORK(I) = B(I) - WORK(I)
10    CONTINUE
C
      CALL PATX(ILIN,JCOL,A,NELEM,NCOL,WORK,R)
C
      RETURN
      END
C     ------------------------------------------------------------
```

Rotina : DIAGATA

Objetivo : Busca a diagonal da matriz $\mathbf{A}^t\mathbf{A}$ e armazena no vetor DIAG .

Variáveis de entrada :

JCOL , A : vetorização da matriz do sistema .

NELEM : número de elementos não-nulos da matriz **A** .

NCOL : número de colunas de **A** .

Variável de saída :

DIAG : vetor contendo a diagonal da matriz $\mathbf{A}^t\mathbf{A}$.

```
      SUBROUTINE DIAGATA(JCOL,A,NELEM,NCOL,DIAG)
C
```

```
C          Procedimento para Calcular  a  Diagonal de      A'.A
C
           DO 10 I = 1,NCOL
                DIAG(I) = 0.D0
10         CONTINUE
C
           DO 20 K = 1,NELEM
                DIAG(JCOL(K)) = DIAG(JCOL(K)) + A(K)*A(K)
20         CONTINUE

C
           RETURN
           END
C  ------------------------------------------------------------
```

Rotina : ARMAZENA

Objetivo : Busca a parte tridiagonal de uma matriz simétrica e armazena nos vetores DIAG e SUB .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

NDIM : dimensão da matriz **A** .

NELEM : número de elementos não-nulos da matriz **A** .

Variáveis de saída :

DIAG : vetor contendo a diagonal principal de **A** .

SUB : vetor contendo a diagonal inferior de **A** .

```
           SUBROUTINE ARMAZENA(ILIN,JCOL,A,NELEM,DIAG,SUB,NDIM)
C
           REAL*8  DIAG(*),SUB(*),A(*)
           INTEGER*2 ILIN(*),JCOL(*),NELEM,I,J,NDIM
C
C          ARMAZENA A PARTE TRIDIDAGONAL DA MATRIZ
C
```

```
      DO 5 K = 1,NDIM
           DIAG(K) = 0.D0
           SUB(K) = 0.D0
5     CONTINUE
C
      DO 10 K = 1,NELEM
           I = ILIN(K)
           J = JCOL(K)
           IF (I .EQ. J) THEN
                DIAG(I) = A(K)
           ELSE

                IF (I .EQ. J+1) SUB(I-1) = A(K)
           ENDIF
10    CONTINUE
C
      RETURN
      END
C     ------------------------------------------------------------------
```

Rotina : SRCHVLR

Objetivo : Busca um elemento na matriz A na posição INEW,JNEW .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

NELEM : número de elementos não-nulos da matriz A .

INEW , JNEW : localização do elemento procurado .

Variável de saída :

ANEW : valor do elemento procurado .

```
      SUBROUTINE SRCHVLR(ILIN,JCOL,A,NELEM,INEW,JNEW,ANEW)
C
      REAL*8 A(*),ANEW
      INTEGER*2 ILIN(*),JCOL(*),NELEM,INEW,JNEW
C
```

```
      DO 10 K = 1,NELEM
           IF(INEW.EQ.ILIN(K).AND.JNEW.EQ.JCOL(K)) THEN
                ANEW = A(K)
                RETURN
           END IF
C
      ANEW = 0.D0
C
10    CONTINUE
C
      RETURN
      END
```

Rotina : TRIDIAG

Objetivo : Busca a parte tridiagonal de uma matriz e armazena nos vetores DIAG , SUP e SUB .

Variáveis de entrada :

ILIN , JCOL , A : vetorização da matriz do sistema .

NELEM : número de elementos não-nulos da matriz A .

MDIM : dimensão da matriz A .

Variáveis de saída :

DIAG : vetor contendo a diagonal principal de A .

SUB : vetor contendo a diagonal inferior de A .

SUP : vetor contendo a diagonal superior de A .

```
      SUBROUTINE TRIDIAG(ILIN,JCOL,A,NELEM,MDIM,DIAG,SUB,SUP)
C
      REAL*8 A(*),DIAG(*),SUB(*),SUP(*),ANEW
      INTEGER*2 ILIN(*),JCOL(*),NELEM,MDIM,INEW,JNEW
C
C     DIAG : DIAGONAL PRINCIPAL DA MATRIZ  A
C     SUB : DIAGONAL INFERIOR DA MATRIZ  A
C     SUP : DIAGONAL SUPERIOR DA MATRIZ  A
```

```
C
C          DIAGONAL PRINCIPAL
C
           DO  10 I = 1,MDIM
               INEW  =  I
               JNEW  =  I
               CALL SRCHVLR(ILIN,JCOL,A,NELEM,INEW,JNEW,ANEW)
               DIAG(I) = ANEW
10         CONTINUE
C
C          DIAGONAL SUPERIOR
C
           DO  20 I = 1,(MDIM-1)
               INEW  =  I
               JNEW  =  I + 1
               CALL SRCHVLR(ILIN,JCOL,A,NELEM,INEW,JNEW,ANEW)
               SUP(I) = ANEW
20         CONTINUE
C
C          DIAGONAL INFERIOR
C
           DO  30 J = 1,(MDIM-1)
               INEW  =  J + 1
               JNEW  =  J
               CALL SRCHVLR(ILIN,JCOL,A,NELEM,INEW,JNEW,ANEW)
               SUB(J) = ANEW
30         CONTINUE
C
           RETURN
           END
C  ------------------------------------------------------------------
```

Rotina : TRIDLU

Objetivo : Decomposição LU para uma matriz tridiagonal **A** .

Variáveis de entrada :

DIAG : vetor contendo a diagonal principal de A .

SUB : vetor contendo a diagonal inferior de A .

SUP : vetor contendo a diagonal superior de A .

MDIM : dimensão da matriz A .

Variáveis de saída :

DIAG : vetor contendo a diagonal principal de U .

SUP : vetor contendo a diagonal superior de U .

SUB : vetor contendo a diagonal inferior de L .

```
      SUBROUTINE TRIDLU(DIAG,SUP,SUB,MDIM)
C
      REAL*8 DIAG(*),SUP(*),SUB(*),EPS
      INTEGER*2 MDIM,I
      DATA EPS / 1.D-20 /
C
C     DECOMPOSICAO L.U DE UMA MATRIZ TRIDIAGONAL
C
      IF(DABS(DIAG(1)).LE.EPS) GO TO 90
      SUP(1) = SUP(1)/DIAG(1)
C
      DO 10 I = 2,MDIM
         K = I - 1
         DIAG(I) = DIAG(I) - SUB(K)*SUP(K)
         IF(DABS(DIAG(I)).LE.EPS) GO TO 90
         IF(I.NE.MDIM)  SUP(I) = SUP(I)/DIAG(I)
10    CONTINUE
C
      GO TO 120
90    WRITE(*,100)
100   FORMAT(///,T10,'MATRIZ DO SISTEMA SINGULAR',///)
      STOP
120   RETURN
```

Rotina : SOLVTRID

Objetivo : Resolução dos sistemas triangulares da decomposição LU da matriz A .

Variáveis de entrada :

DIAG : vetor contendo a diagonal principal de U .

SUB : vetor contendo a diagonal inferior de L .

SUP : vetor contendo a diagonal superior de U .

B : vetor do lado direito do sistema .

MDIM : dimensão da matriz A .

Variável de saída :

X : vetor contendo a solução do sistema linear .

```
      SUBROUTINE SOLVTRID(DIAG,SUP,SUB,B,X,MDIM)
C
      REAL*8   DIAG(*),SUP(*),SUB(*),B(*),X(*)
      INTEGER*2 MDIM
C
C     RESOLUCAO DO SISTEMA TRIANGULAR INFERIOR
C
      X(1) = B(1)/DIAG(1)
      DO 10 I = 2,MDIM
         K = I -1
         X(I) = ( B(I) - SUB(K)*X(K))/DIAG(I)
10    CONTINUE
C
C     RESOLUCAO DO SISTEMA TRIANGULAR SUPERIOR
C
      DO 20 I = (MDIM-1),1,-1
         K = I + 1
         X(I) = X(I)  - SUP(I)*X(K)
20    CONTINUE
C
      RETURN
      END
```

Rotina : CHOLTRI

Objetivo : Decomposição de Cholesky de uma matriz tridiagonal **A** .

Variáveis de entrada :

DIAG : vetor contendo a diagonal principal de **A** .

SUB : vetor contendo a diagonal inferior de **A** .

N : dimensão da matriz **A** .

Variáveis de saída :

DIAG : vetor contendo a diagonal principal de **G** .

SUB : vetor contendo a diagonal inferior de **G** .

```
      SUBROUTINE CHOLTRI(DIAG,SUB,N)
C
      REAL*8 DIAG(*),SUB(*),AUX
      INTEGER*2 N
C
C     DECOMPOSICAO INCOMPLETA DE CHOLESKY DE UMA MATRIZ TRIDIAGONAL
C
      AUX = DIAG(1)
      IF (AUX .LT. 0.D0) AUX = -AUX
      DIAG(1) = DSQRT(AUX)
      SUB(1) = SUB(1)/DIAG(1)
C
      DO 10 I = 2,N-1
         AUX = DIAG(I) - SUB(I-1)*SUB(I-1)
         IF (AUX .LT. 0.D0) AUX = - AUX
         DIAG(I) = DSQRT(AUX)
         SUB(I) = SUB(I)/DIAG(I)
10    CONTINUE
C
      AUX = DIAG(N) - SUB(N-1)*SUB(N-1)
      IF (AUX .LT. 0.D0) STOP
      DIAG(N) = DSQRT(AUX)
C
      RETURN
      END
```

Rotina : SOLVE

Objetivo : Resolução dos sistemas triangulares da decomposição de Cholesky de **A** .

Variáveis de entrada :

DIAG ; vetor contendo a diagonal principal de G .

SUB : vetor contendo a diagonal inferior de G .

B : vetor contendo o lado direito do sistema linear .

N : dimensão da matriz **A** .

Variável de saída :

X : vetor solução do sistema .

```
      SUBROUTINE SOLVE(DIAG,SUB,B,X,N)
      REAL*8   DIAG(*),SUB(*),B(*),X(*)
      INTEGER*2  N
C
C     RESOLUCAO DO SISTEMA TRIANGULAR INFERIOR
C
      X(1) = B(1) / DIAG(1)
      DO 10 I=2,N
          K = I-1
          X(I) = ( B(I) - SUB(K)*X(K) ) / DIAG(I)
10    CONTINUE
C
C     RESOLUCAO DO SISTEMA TRIANGULAR SUPERIOR
C
      X(N) = X(N)/DIAG(N)
C
      DO 20 I=(N-1),1,-1
          K = I + 1
          X(I) = ( X(I)  - SUB(I)*X(K) )/DIAG(I)
20    CONTINUE
C
      RETURN
      END
```

BIBLIOGRAFIA

[1] A. W. Adamson - Physical Chemistry Of Surfaces , 1960 , Interscience Publishers .

[2] R. Bassanezi , W. Ferreira - Equações Diferenciais com Aplicações , 1988 , Editora Harbra .

[3] P. Concus - " Numerical Solution of the Minimal Surface Equation " , Maths of Comp. , volume 21 , 1967 , pags 340-350 .

[4] C. Cuvelier , R.M.S.M. Schulkes - " Some Numerical Methods For The Computation Of Capillary Free Boundaries Governed By The Navier-Stokes Equation " , SIAM Review , vol 32 , número 3 , pags 355-423 , Setembro de 1990 .

[5] R. S. Dembo , S. C. Eisentat , T. Steihaug - " Inexact Newton Methods " , SIAM J. Numer. Anal. , volume 19 , número 2 , pags 400-408 , Abril de 1982 .

[6] Demidovich , Maron - Calculo Numerico Fundamental , 1977 , Editora VAAP .

[7] J. E. Dennis , K. Turner - " Generalized Conjugate Directions ", Linear Algebra And Its Applications 88/89 , pags 187-209 , 1987 .

[8] J. Dongarra et al - Solving Linear Systems On Vector And Shared Memory Computers , SIAM .

[9] S. C. Eisenstat , H. C. Elman , M. H. Schultz - " Variational Iterative Methods For Nonsymmetric Systems Of Equations ", SIAM J. Numer. Anal. , volume 20 , número 2 , , pags 345-357 , Abril de 1983 .

[10] L. Elsgoltz - Ecuaciones Diferenciales y Cálculo Variacional , 1977 , Editora MIR .

[11] R. Finn - Equilibrium Capillary Surfaces , 1986 , Springer-Verlag .

[12] G. H. Golub , C. V. Loan - Matrix Computations , 1985 , The Johns Hopkins University Press .

[13] C. E. Harle - Geometria Diferencial , IX Colóquio Brasileiro de Matemática (1973) .

[14] C. Johnson - Numerical Solution of Partial Differential Equations by The Finite Element Method , 1987 , Cambridge University Press .

[15] A. N. Kolmogorov , S.V. Fomin - Elementos da Teoria das Funções e de Análise Funcional , 1982 , Editora MIR .

[16] M. A. Krasnosel'skii , Ya. B. Rutickii - " Some Approximate Methods Of Solving Nonlinear Operator Equations Based On Linearization ", Soviet Math. Dokl. , volume 2 , pags 1542-1546 , 1961 .

[17] Mitchell & Griffiths - The Finite Difference Method in Partial Differential Equations ,1987, John Wiley & Sons .

[18] G. Pini , G. Zilli - " On Vectorizing The Preconditioned Generalized Conjugate Residual Methods ", Intern. J. Computer Math , volume 33 , pags 195-207 , 1990 .

[19] L. B. Rall - Computacional Solution Of Nonlinear Operator Equations , 1969 , John Wiley & Sons .

[20] Y. Saad - " Krylov Subspace Methods For Solving Large Unsymmetric Linear Systems ", Mathematics Of Computation , volume 37 , número 155 , pags 105-126 , Julho de 1981 .

[21] Y. Saad , M. Schultz - " GMRES : A Generalized Minimal Residual Algorithm For Solving Nonsymmetric Linear Systems ", SIAM J. Sci. Stat. Comput. , volume 7, número 3 , pags 856-869 , Julho de 1986 .

[22] Y. Saad - " Krylov Subspace Methods On Supercomputers ", SIAM J. Sci. Stat. Comput. , volume 10 , número 6 , pags 1200-1232 , Novembro de 1989 .

[23] A. H. Sherman - " On Newton-Iterative Methods For The Solution Of Systems Of Nonlinear Equations ", SIAM J. Numer. Anal. , volume 15 , número 4 , pags 755-771 , Agosto de 1978 .

[24] P. Sonneveld - " CGS , A Fast Lanczos-Type Solver For Nonsymmetric Linear Systems ", SIAM J. Sci. Stat.Comput., volume 10, número 1 , pags 36-52 , Janeiro de 1989 .

[25] Van der Vorst - " The Convergence Behavior Of Preconditioned CG And CG-S In The Presence Of Rounding Errors ", Lecture Notes In Mathematics 1457 , pags 126-136 Springer-Verlag , 1990 .

[26] Van der Vorst - " BI-CGSTAB : A Fast And Smoothly Converging Variant Of BI-CG For The Solution Of Nonsymmetric Linear Systems ", SIAM J. Sci. Stat. Comput. , volume 13, número 2, pags 631-644 , Março de 1992 .