PANEL '84
EXPODATA

DECIMA CONFERENCIA LATINOAMERICANA DE INFORMATICA

23 · 28 ABRIL '84
VIÑA DEL MAR - CHILE

• UNIVERSIDAD CATOLICA DE VALPARAISO
CENTRO LATINOAMERICANO DE ESTUDIOS EN INFORMATICA. CLEI

X CONFERENCIA LATINOAMERICANA de INFORMATICA

VIÑA DEL MAR ABRIL 1984

documentos de trabajo

" SOLUÇÃO DE GRANDES SISTEMAS LINEARES COM MINI-COMPUTADORES "

Paulo José Erlich
Therezinha Chaves

DEPARTAMENTO DE INFORMÁTICA
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
22453 - Rio de Janeiro, R.J. Brasil

I - INTRODUÇÃO

A solução de sistemas lineares de grandes dimensões e, principalmente banda ou tridiagonais é bastante frequente em uma grande quantidade de problemas de engenharia, princi - palmente os que envolvem a solução numérica de equações diferências parciais. Isso torna necessária a utilização de máquinas de grande porte para o tratamento numérico desses siste - mas. Entretanto, nos últimos anos, a quantidade de mini-computadores no mercado tem aumentado numa proporção muito superior a dos computadores de grande porte, crescendo também é número de usuários e de aplicações dessas máquinas. Considerando-se como as características fundamentais de um mini-computador a simplicidade de operação e o limitado tamanho da memó - ria - tipicamente 64 K bytes - verifica-se a incapacidade de se trabalhar neles com sistemas lineares de dimensões consideráveis, que exigiria, normalmente, uma quantidade muito maior de memória.

Existem, entretanto, ferramentas da própria Álgebra Linear que estão sendo desenvolvidas para permitir o uso de máquinas de pequeno e médio porte. Vamos, nesse trabalho, utilizar uma dessas idéias - a da partição de matrizes - para resolver com mini-computadores (em princípio o COBRA) grandes sistemas lineares.

II - PARTIÇÃO

Consideremos o sistema linear AX = B onde A é uma matriz quadrada n × n, que contém os coeficientes do sistema, X e B vetores de n componentes representando respectivamente o vetor solução e o vetor termo independente do sistema.

Se, dado o sistema linear AX = B realizarmos uma partição em A passaremos a considerá-la sob uma nova forma, isto é, sendo A dada por

$$A = \left(\begin{array}{cc|ccc} a_{11} & a_{12} & a_{13} & -a_{14}- & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ \hline a_{3b} & a_{32} & a_{33} & -a_{34} & a_{35} \\ a_{41} & a_{32} & a_{33} & -a_{44}- & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{array}\right)$$

e a separação dos elementos de A da forma indicada, teremos a nova matriz A_B dada por

$$A_B = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$$

onde agora os elementos de A_B são também matrizes, respectivamente $A_1(2 \times 2)$, $A_2(2 \times 3)$ $A_3(3 \times 2)$ e $A_4(3 \times 3)$.

III - MÉTODOS BLOCO-ITERATIVOS

a - Idéia

Uma das idéias de se usar a partição de matrizes para a solução de sistemas lineares é com métodos iterativos. Assim, métodos iterativos clássicos como Jacobi, Gauss-Seidel ou Relaxação ganham nova roupagem. Por exemplo, na aplicação do método de Gauss-Seidel a um sistema AX = B particionado , como no exemplo da forma

$$\begin{cases} A_1X_1 + A_oX_2 = B_1 \\ A_3X_1 + A_4X_2 = B_2 \end{cases}$$

onde X_1, X_2, B_1 e B_2 são partições iguais às feitas em A, com idênticas dimensões, isto é

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \qquad B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \qquad X_1 = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad B_1 = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$X_2 = \begin{pmatrix} x_3 \\ x_4 \\ x_5 \end{pmatrix} \qquad B_2 = \begin{pmatrix} b_3 \\ b_4 \\ b_5 \end{pmatrix}$$

o novo sistema será resolvido utilizando-se cada uma de suas linhas - que representam os sub-sistemas do sistema original - de maneira independente.

b - Aspectos computacionais

Desde que resolvemos isoladamente cada um dos subsistemas selecionados através da definição da partição do sistema, não necessitamos armazenar ao mesmo tempo todo o sistema original, bastando apenas que se armazene um sub - sistema de cada vez. Esse fato é o mais importante para a utilização computacional desses métodos, pois podemos reduzir drástica - mente a área de memória necessária.

Há, entretanto, problemas paralelos relacionados com a precisão da solução. Em geral, a limitação de memória implica também em perda de precisão se olharmos a eficiência das máquinas de grande porte. A manipulação de dupla precisão (ou até precisão extendida) requer uma duplicação (ou n-plicação) da memória utilizada.

Procuramos aproveitar da nelhor maneira possível, a característica desses métodos de modo a minimizar a área de memória necessária para armazenar o sistema, permitindo assim o trabalho com matrizes de ordem elevada. Mas como foi indicado antes, embora às matrizes mais freqüentes sejam do tipo banda e, algumas vezes, apenas tridiagonais, preferimos tratar do trabalho no seu pior caso, isto é, quando a matriz A do sistema é cheia e geral.

c - Problemas relativos à convergência

Como se sabe a aplicação de um método iterativo requer sempre um estudo detalhado da convergência que, em última análise, depende do raio espectral da matriz de iteração definida pelo método escolhido, cujo estudo é bastante complexo para automatizar. Entretanto,sabemos que, no caso mais frequente de discretização de equações diferenciais parciais, a matriz A do sistema original é tal que para seus elementos a_{ij}, i, j=1,...,n

a) $a_{ii} > \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} \qquad \forall\ i \leq n$

b) $a_{ii} > o$, $a_{ij} \leq o$, $i \neq j$, $i,j \leq n$

e que nesse caso a convergência dos métodos simples e bloco de Jacobi e Gauss-Seidel convergem sempre [1]. Assim, enquanto trabalhamos com essa classe de matrizes não teremos problemas de convergência.

Ainda por problemas de convergência demos preferência ao método de Gauss-Seidel pois sabemos que no caso das matrizes acima mencionadas, podemos garantir que o método de Gauss-Seidel converge assintoticamente melhor que o de Jacobi [1].

Vale ainda a pena ressaltar o fato de que um método bloco-iterativo é na realidade "híbrido" pois em sua primeira parte é utilizado um método direto (não iterativo) para determinar as expressões de $X_1^{(k)}$, $X_2^{(k)}$,..., $X_m^{(k)}$, independentemente um do outro. Numa segunda fase é utilizado um método itera

tivo que nos leva de $X_1^{(k)},\ldots,X_m^{(k)}$ para $X_1^{(k+1)},\ldots,X_m^{(k+1)}$
Dissemos que escolhemos o método de Gauss-Seidel pois para passar de um passo k para o seguinte k+1, utilizamos os novos valores intermediários já calculados nesse próprio passo.

IV - INVERSÃO POR PARTIÇÃO

a - Idéia

Dada a matriz A de um sistema AX=B, para determinar sua solução podemos simplesmente considerar $X=A^{-1}B$ (desde que A^{-1} exista isso é sempre possível). Assim podemos, particio - nar a matriz A convenientemente, invertê-la por um processo econômico e multiplicando pelo vetor independente, obter a so lução final.

b - Definição

Considerando-se a matriz $A=(a_{ij})$ i,j, = 1,...,n po demos reescrevê-la como

$$A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$$

onde A_1 e A_3 são matrizes quadradas $m_1 \times m_1$ e $m_2 \times m_2$, $m_1 + m_2 = n$.

De acordo com [ ] a inversa de A será dada por

$$A^{-1} = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}$$

onde

$$X_1 = (A_1 - A_2 A_4^{-1} A_3)^{-1}$$

$$X_2 = (A_4 - A_3 A_1^{-1} A_2)^{-1}$$

$$X_3 = - A_1^{-1} A_2 X_2$$

$$X_4 = - A_4^{-1} A_3 X_1$$

Como se pode notar vamos precisar da inversa das du as matrizes A_1 e A_4 cujas ordens m_1 e m_2 são tais que $m_1 + m_2 = n$. Assim reduzimos drasticamente a ordem da matriz a ser invertida.

Para se reduzir ainda mais essa ordem é possível a plicarmos recursivamente o processo da partição, para inver - ter cada uma das matriz A_1 e A_4. No nosso caso permitimos es se fato apenas mais uma vez. Entretanto é possível levar-se a tê que a ordem da matriz final a ser invertida seja 2×2.

V - PRINCÍPIOS DE PROGRAMAÇÃO

De acordo com as características básicas de programação, procuramos dar ao código presente uma estrutura que facilitasse sua utilização no maior número possível de máquinas de médio porte. Assim utilizamos a linguagem FORTRAN IV Standard, padrão AN5. Deixamos de aproveitar algumas pecularidades FORTRAN, por serem fora dos padrões internacionais, uma vez que consideramos de fundamental importância a adaptabilidade das rotinas a qualquer mini-computador.

Os mini-computadores têm como características básicas, a simplicidade de operação e o limitado tamanho da memória - tipicamente, 64 KBYTES. Com a nossa implementação, somos capazes de, por exemplo, inverter uma matriz 1ØØ × 1ØØ , necessitando de uma Área de Dados de, aproximadamente 36 KBYTES; enquanto que, se fossemos inverter a mesma matriz sem particioná-la, só de área para armazenar a matriz, seriam necessários 80 KBYTES, sem considerar as Áreas de Trabalho. Para conseguirmos implementar as rotinas que manuseiam matrizes particionadas, tivemos que criar uma Estrutura de Arquivos de Trabalho compatível com as rotinas. Esses arquivos têm o objetivo de armazenar resultados intermediários de diversas rotinas e servem de interface entre as mesmas. O problema de espaço também está presente na Estrutura dos Arquivos, uma vez que, tipicamente, os minis possuem uma área de 128 KBYTES reservados para Memórias Auxiliar. Cada arquivo lógico, além de armazenar a esturura correspondente (vetor a matriz do trabalho), armazenar a(s) dimensão(ões) da mesma, com o objetivo de poder recuperá-la sempre que necessário. Afim de otimizar o tempo de E-S, usamos os comandos 'READ' e 'WRITE' sem formato e, com isso, os arquivos têm que ser criados como sendo do tipo 'VBS' (Variable Blocked Spannd).

VI - DESEMPENHO

Foram obtidos os tempos de execuções para a solução de sistemas lineares e inversão de matrizes, considerando várias ordens de matriz (Tridiagonal e Cheia) e número de decimais de precisão distinto. Os resultados são mostrados nas tabelas I e II à seguir, onde na horizontal estão as ordens das matrizes e na vertical o número de decimais de precisão desejada para a Solução do Problema. Todos os tempos de Execução estão em segundos.

Ordem de matriz / Decimais de Precisão	1	25	26	5Ø	51	1ØØ
1	Ø.Ø2	Ø.4Ø	4.91	9.5Ø	16.73	3Ø.ØØ
2	Ø.Ø2	Ø.4Ø	5.Ø5	1Ø.16	16.84	3Ø.88
3	Ø.Ø2	Ø.6Ø	5.Ø8	11.ØØ	17.Ø5	32.56
4	Ø.Ø2	Ø.6Ø	5.29	12.Ø1	17.32	34.21
5	Ø.Ø2	Ø.7Ø	5.36	13.15	17.68	35.81
6	Ø.Ø2	Ø.8Ø	5.42	13.18	18.46	37.61

MATRIZES TRIDIAGONAIS - SOLUÇÃO TABELA - I.1

Ordem de matriz / de precisão	1	25	26	5Ø	51	1ØØ
1	Ø.Ø2	Ø.37				
2	Ø.Ø2	Ø.53				
3	Ø.Ø2	Ø.72				
4	Ø.Ø2	Ø.87				
5	Ø.Ø2	1.Ø6				
6	Ø.Ø2	1.22	3.55	12.48	67.34	197.Ø2

MATRIZES TRIDIAGONAIS - INVERSÃO TABELA - I.2

Ordem da matriz / Decimais de precisão	1	25	26	5Ø	51	1ØØ
1	Ø.Ø2	Ø.4Ø	6.ØØ	18.ØØ	117.ØØ	248.ØØ
2	Ø.Ø2	Ø.62	46.ØØ	138.ØØ	214.ØØ	544.ØØ
3	Ø.Ø2	Ø.85	118.ØØ	1158.ØØ	1745.ØØ	436Ø.ØØ
4	Ø.Ø2	1.Ø7	2Ø6.ØØ	2258.ØØ	3135.ØØ	8252.ØØ
5	Ø.Ø2	1.3Ø	314.ØØ	3411.ØØ	4723.ØØ	12475.ØØ
6	Ø.Ø2	1.52	434.ØØ	4813.ØØ	6669.ØØ	176Ø3.ØØ

MATRIZES CHEIAS - SOLUÇÃO TABELA II.1

Ordem da matriz / Decimais de precisão	1	25	26	5Ø	51	1ØØ
1	Ø.Ø2	2.11				
2	Ø.Ø2	7.28				
3	Ø.Ø2	12.98				
4	Ø.Ø2	18.71				
5	Ø.Ø2	24.3Ø				
6	Ø.Ø2	3Ø.Ø	6.58	61.75	87.49	433.35

MATRIZ CHEIAS - INVERSÃO TABELA II.2

VII - CONCLUSÕES

Observando os resultados obtidos nas Tabelas I e II, vemos que, para matrizes tridiagonais, os resultados foram altamente satisfatórios. É importante notar que o conjunto de rotinas implementadas visam ao usuário que, tipicamente, possuirá matrizes oriundas da discretização de equações diferenciais parciais parabólicas e elipticas, ou seja matrizes banda. Nesse caso, se torna altamente interessante a técnica de partição de matrizes. Ressaltamos aqui, que as rotinas foram implementadas levando em consideração que o usuário possuida matrizes não obrigatoriamente banda. Se tivessemos assumido o contrário, isto é, que todas as matrizes seriam banda, obteríamos um desempenho ainda melhor, uma vez que otimizaríamos o armazenamento e o acesso/recuperação de todas as estruturas envolvidas.

Para o caso de possuírmos matrizes cheias, os resultados mostram que a solução de sistemas pode se tornar extremamente lenta. Nesse caso é recomendável a utilização da inversa obtida pelo processo de partição, que se mostrou muitas vezes mais eficiente.

Como comentário final, cumpre observar que a diferença dos tempos de execução da inversão de matrizes tridiagonais para as matrizes cheias não é tão gritante quanto à solução, posto que a inversão de uma matriz tridiagonal é geralmente uma matriz cheia.

VIII - BIBLIOGRAFIA

a - Referência

[1] Métodos Interativos para Sistemas "Lineares e sua aplicação na Soluções Diferenciais Parciais" - Prof. Peter Albrecht, Série "Monografias em Ciências da Computação" nº 1/82. Departamento de Informática, PUC/RJ

b - Bibliografia complementar

" Matrix Iterative Analysis", R. Varga Prentice-Hall.