PROBLEMAS LINEARES COMPLEMENTARES MONÓTONOS

FERNANDA MARÍLIA DANIEL PIRES

Abril de 1994

PROBLEMAS LINEARES COMPLEMENTARES MONÓTONOS

FERNANDA MARÍLIA DANIEL PIRES

Dissertação para obtenção do Grau de Doutor em Ciências Exactas, na especialidade de Matemática, na Universidade do Algarve

Abril de 1994

1

AGRADECIMENTOS

Durante os anos que dediquei a este trabalho muitos foram aqueles que de algum modo contribuíram para a sua realização. Referir aqui todos os nomes seria impossível. Há, no entanto, algumas pessoas a quem não posso deixar de expressar a minha gratidão pois o seu contributo foi demasiado valioso para que seja deixado no anonimato.

Ao Professor Doutor Joaquim João Júdice, do Departamento de Matemática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, que orientou desde a primeira hora o desenvolvimento deste trabalho bem como a demais investigação entretanto produzida e com quem espero ter o previlégio de continuar a trabalhar no futuro. Além da sua lúcida orientação quero agradecer ao Professor Júdice a sua amizade, a sua paciência e as palavras de encorajamento que sempre lhe ouvi.

Ao Professor Doutor Afonso Serra Neves, do Departamento de Engenharia Civil da Faculdade de Engenharia da Universidade do Porto, pela colaboração prestada na obtenção dos modelos de Análise de Estruturas.

À Professora Doutora Ana Maria Faustino, do Departamento de Engenharia Civil da Faculdade de Engenharia da Universidade do Porto, colega e amiga, pelas nossas longas conversas de que normalmente surgiram boas ideias para continuar o trabalho.

Ao Professor Doutor Joseph Conboy, da Unidade de Ciências Exactas e Humanas da Universidade do Algarve, por sempre ter tentado proporcionar-me boas condições de trabalho nos tempos difíceis que se seguiram à minha mudança para a Universidade do Algarve.

NOTAÇÕES

Espaços	designados por letras maiúsculas
R	a recta real
R^n	espaço vectorial real de dimensão n
$R^{n\times m}$	espaço vectorial real das matrizes de n linhas e m colunas
Vectores	designados por letras minúsculas
z	vector identificado com uma matriz coluna
z^T	o transposto de z (identificado com uma matriz linha)
$\{ z^k \}$	sucessão de vectores $z^1, z^2, \ldots$
z_i	componente de ordem i do vector z
z_i^k	componente de ordem i do vector z^k
e	vector com todas as componentes iguais a 1
$\| z \|$	norma do vector z (qualquer norma em R^n)
$x \geq y$	a relação de ordem usual ($x_i \geq y_i$, $i = 1, \ldots, n$)
$x > y$	a relação de ordem estrita usual ($x_i > y_i$, $i = 1, \ldots, n$)
$z^T w$	produto interno usual dos vectores z e w
$\tilde{q}$	vector tranformado de q por uma operação pivotal principal
Matrizes	designadas por letras maiúsculas romanas ou gregas
det M	o determinante de M
M^{-1}	a matriz inversa de M
$\| M \|$	uma norma da matriz M
M^T	matriz transposta de M

I	matriz identidade
$diag(a_1, a_2, \ldots, a_n)$	matriz diagonal cujos elementos são $a_1, a_2, \ldots, a_n$
M_{FT}	submatriz de M cujos elementos são m_{ij}, $i \in F$ e $j \in T$
$M_{.k}$	coluna k da matriz M
$M_{k.}$	linha k da matriz M
$M_{F.}$	todas as linhas com índice em F da matriz M
$M_{.F}$	todas as colunas com índice em F da matriz M
M_{FF}^{-1}	$(M_{FF})^{-1}$ a inversa de uma submatriz principal de M
$\bar{M}$	transformada principal de M

Conjuntos	designados por letras maiúsculas gregas ou romanas
$\in$	pertence
$\notin$	não pertence
$\varnothing$	conjunto vazio
$\subset$	inclusão de conjuntos (não estrita)
$\cup$	reunião de conjuntos
$\cap$	intersecção de conjuntos
A-B	$\{ x \in A \text{ e } x \notin B \}$, conjunto diferença de A e B
# A	número elementos do conjunto A
[a, b]	intervalo fechado em R
] a, b [	intervalo aberto em R

Funções	designadas por letras gregas ou romanas
∇f	função gradiante da função f , $\nabla f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$
H ' (z, d)	derivada direccional de H no ponto z na direcção do vector d

ÍNDICE

INTRODUÇÃO

Apesar da sua origem relativamente recente, o Problema Linear Complementar (LCP) tem sido objecto de muitos e variados estudos, existindo actualmente um elevado número de algoritmos mais ou menos eficientes para a sua resolução. Além disso, este problema tem encontrado variadissimas aplicações nas diferentes áreas da ciência, engenharia e economia, nomeadamente em problemas de selecções de portfolios, equilíbrio económico, teoria dos jogos e análise elástica e elastoplástica de estruturas.

A existência e unicidade de solução para o LCP está relacionada com a classe a que pertence a sua matriz. Os LCPs monótonos são caracterizados pela sua matriz ser positiva semi-definida (PSD). Nestas condições é possível concluir que o LCP tem solução desde que as suas restrições lineares sejam compatíveis. Por outro lado, o LCP diz-se estritamente monótono se a sua matriz pertence à classe P, que é caracterizada pelo facto de todas as suas submatrizes principais terem determinantes positivos. Um LCP estritamente monótono é monótono se a sua matriz é simétrica. Contudo, isso não é necessariamente verdadeiro no caso da matriz ser não simétrica. Além disso, um LCP estritamente monótono tem solução única para cada vector independente.

Na maior parte das suas aplicações, a matriz do LCP é de ordem elevada e estrutura esparsa. Tal como é referido nos livros mais importantes sobre a complementaridade linear [12, 65], existe uma grande falta de estudos computacionais sobre a eficiência dos algoritmos existentes para o LCP. A primeira intenção desta tese foi

exactamente o preenchimento dessa lacuna. Nesse sentido, apresentamos uma revisão exaustiva dos principais algoritmos existentes para o LCP monótono. Esses processos podem ser directos ou iterativos. Os processos do segundo tipo podem ser implementados de uma maneira muito simples, mas não são considerados robustos, por dependerem bastante da condição da matriz. Por outro lado, os métodos directos necessitam de implementações mais elaboradas, principalmente quando se pretende resolver problemas de grandes dimensões. Neste trabalho desenvolvemos implementações eficientes para esse tipo de algoritmos.

Experiência computacional apresentada nesta tese com os algoritmos directos existentes para o LCP implicou a necessidade do desenvolvimento de técnicas pivotais por blocos. Nesta tese é introduzido um algoritmo deste tipo que não só tem convergência finita mas também se mostrou muito eficiente para a resolução de LCPs monótonos de grandes dimensões.

O Problema Linear Complementar com Limites (BLCP) é talvez a extensão mais natural e mais importante do LCP. Este problema pode ser sempre reduzido a um LCP e isso tem vindo a ser sugerido na maior parte dos trabalhos sobre complementaridade linear. Contudo, uma tal redução conduz a uma duplicação da ordem da matriz do LCP, com todos os custos daí inerentes. Daí surgir a necessidade de se encontrarem extensões dos algoritmos directos do LCP que possam ser aplicados directamente ao BLCP. Nesta tese são desenvolvidos e testados algoritmos pivotais principais simples e por blocos para o BLCP. As implementações desses processos são semelhantes às dos métodos directos do LCP. Experiência computacional com esses processos é também incluída e permite concluir que o método pivotal por blocos é muito eficiente para BLCPs de grandes dimensões.

É perfeitamente conhecido que todo o programa quadrático convexo se pode reduzir a um LCP ou BLCP monótono. Contudo tal redução conduz a problemas complementares com matrizes não simétricas, para os quais o uso de métodos pivotais principais não é muito recomendado. Se o programa quadrático for de variáveis separadas, então é possível reduzi-lo a um LCP ou BLCP monótono com uma matriz simétrica. Nesse caso os métodos pivotais principais por blocos podem ser utilizados. Por outro lado, esse tipo de processo pode também ser útil na resolução de um programa quadrático estritamente convexo com apenas uma restrição de igualdade além de limites nos valores das variáveis. Esses assuntos são plenamente discutidos nesta tese, sendo comparadas as eficiências desses métodos pivotais por blocos com as técnicas que têm vindo a ser normalmente mencionadas para o efeito. Esses estudos computacionais permitem concluir da superioridade das técnicas pivotais por blocos neste tipo de problemas.

Como referimos anteriormente, a complementaridade linear encontra grandes aplicações em alguns problemas de análise de estruturas. Nesta tese debruçamo-nos também sobre este aspecto, apresentando dois problemas de estruturas que são formulados em termos de problemas lineares complementares. Além disso, mostramos que as técnicas pivotais principais podem ser, mais uma vez, muito eficientes na resolução dos correspondentes problemas lineares complementares.

A tese está dividida em oito capítulos. No capítulo 1 introduzimos os Problemas Lineares Complementares LCP e BLCP e referimos sucintamente algumas das suas propriedades e equivalências a outros tipos de problemas. No capítulo 2 discutimos os métodos pivotais simples para o LCP, sendo os métodos pivotais principais por blocos apresentados no capítulo 3. Por sua vez, nos capítulos 4 e 5 são apresentadas extensões de alguns dos algoritmos dos capítulos 2 e 3 para a resolução de BLCPs monótonos. No capítulo 6 são descritos métodos de resolução do LCP e do BLCP que exploram a equivalência destes problemas a problemas de Optimização Não Linear. Alguns desses

algoritmos exploram a redução do LCP a um programa quadrático, enquanto que outros usam a equivalência do LCP a um sistema de equações não lineares. No capítulo 7 são apresentados algoritmos para a resolução de programas quadráticos estritamente convexos mencionados anteriormente. Finalmente, a aplicação da complementaridade linear a problemas de análise de estruturas é discutida no último capítulo desta tese.

CAPÍTULO I

PROBLEMA LINEAR COMPLEMENTAR

1 DEFINIÇÃO E PROPRIEDADES

Dados um vector $q \in R^n$ e uma matriz $M \in R^{n \times n}$, o Problema Linear Complementar (LCP) consiste em encontrar dois vectores (ou concluir que não existem) $w \in R^n$ e $z \in R^n$ tais que:

$$\begin{aligned} & w = q + Mz \\ & z \geq 0 \\ & w \geq 0 \\ & z^T w = 0 \end{aligned} \qquad (1.1)$$

Como o LCP depende do vector e da matriz dadas, representa-se normalmente por LCP(q, M) o LCP com vector q e matriz M.

Este problema tem sido muito estudado quer do ponto de vista prático quer do ponto de vista teórico desde que foi formulado pela primeira vez, há cerca de vinte anos [12, 65]. Inicialmente o problema foi proposto como um meio de resolver o jogo de duas matrizes e o problema de programação quadrática convexa [9]. Actualmente conhecem-se numerosas aplicações nas áreas da física, engenharia [26, 66, 67, 76] e economia [71, 73, 74]. A existência de solução para o LCP continua um problema em aberto, a não ser que a matriz M pertença a determinadas classes, que têm aparecido em algumas das suas aplicações. Em muitas dessas

aplicações a ordem da matriz é muito elevada o que justifica a necessidade de se desenvolverem técnicas eficientes para lidar com esses LCPs. Normalmente essas matrizes são esparsas, com maior ou menor percentagem de elementos não nulos conforme a natureza do problema. Em alguns casos concretos, as matrizes têm um mesmo padrão de não zeros especial que possibilita o uso de técnicas muito específicas para a sua solução. Estão neste caso os problemas de resolução numérica de equações de derivadas parciais com condições fronteira [14]. No entanto, em geral é difícil prever a estrutura específica da matriz do problema, não havendo sequer, na generalidade dos casos, qualquer estrutura definida. Assim, torna-se necessário desenvolver técnicas gerais, adaptáveis a qualquer caso, que não tirem partido de qualquer estrutura especial.

Vários métodos têm sido propostos para resolução do LCP. Conforme as diferentes abordagens do problema podem ser classificados em directos, iterativos e enumerativos. Além disso, alguns autores têm procurado resolver o LCP por redução a outros problemas de optimização [12].

Os métodos directos são baseados em operações pivotais e tentam obter uma solução exacta do problema (a menos de uma certa precisão numérica, dependente da precisão dos dados e da precisão da máquina utilizada). Nos métodos iterativos usam-se extensões de técnicas normalmente empregues na resolução de sistemas, convenientemente adaptadas para lidarem com a relação de complementaridade $z^T w = 0$. Tal como acontece para a resolução de sistemas, a solução vai sendo melhorada de iteração em iteração, segundo um determinado critério, até que se atinja uma certa precisão. Recentemente têm sido propostos métodos de ponto interior (assim chamados devido à interpretação geométrica do processo de solução), que podem ser considerados iterativos na concepção e directos no funcionamento [50, 51].

Qualquer algoritmo pertencente a um destes tipos só resolve o LCP, isto é, encontra uma solução ou determina a sua não existência , quando a matriz pertence a certas classes.

Alguns autores têm procurado resolver o LCP substituindo a relação de complementaridade $z^T w = 0$ pela optimização de uma função que lhe seja equivalente. Assim, em lugar de resolver um LCP, resolvem um problema de optimização. A dificuldade neste tipo de abordagem está no facto da função a minimizar ser não convexa em geral e, portanto, o problema ser de difícil solução.

Os métodos enumerativos podem resolver o LCP, para qualquer classe de matriz, explorando o caracter combinatório de que o LCP se reveste. São processos bastante complexos, que, de um modo geral, utilizam uma grande quantidade de memória e de tempo de computador. Por isso devem ser considerados como métodos de recurso para usar quando os métodos directos e iterativos não puderem ser utilizados.

Num LCP temos a distinguir as relações lineares, que definem o conjunto admissível

$$K = \{ (z,w) : w=q+Mz \; ; \; w \geq 0 \; , \; z \geq 0 \} \qquad (1.2)$$

e a condição de complementaridade $z^T w = 0$. Uma vez que os vectores z e w não têm componentes negativas, então essa relação é equivalente à verificação simultânea das n condições:

$$z_i w_i = 0 \quad , \quad i = 1, \ldots , n \qquad (1.3)$$

Qualquer par de vectores (z,w) pertencente ao conjunto K diz-se admissível, e é complementar se satisfizer as condições (1.3). Portanto, uma solução é complementar

se no máximo uma das variáveis do par (z_i ,w_i) é não nula. As variáveis desse par dizem-se complementares uma da outra. É evidente que a solução do LCP é simultaneamente complementar e admissível.

2 Classes de Matrizes

Os principais resultados teóricos relativos à existência de solução para o LCP são formulados em termos de classes de matrizes [33]. As classes mais gerais nesse sentido são as das matrizes Q e Q_0 e satisfazem as seguintes equivalências:

$M \in Q \Leftrightarrow$ LCP(q, M) tem solução para qualquer $q \in R^n$

$M \in Q_0 \Leftrightarrow$ LCP(q, M) tem solução para qualquer $q \in R^n$ tal que $K \neq \emptyset$

onde K é o conjunto admissível do LCP definido por (1.2).

Dado um LCP(q, M), decidir se a sua matriz pertence a uma destas classes é tarefa difícil se não mesmo impossível. No entanto existem algumas subclasses dessas matrizes que são facilmente identificáveis. Seguidamente apresentamos algumas dessas classes.

$M \in P \Leftrightarrow$ todos os menores principais são positivos

$M \in PD \Leftrightarrow x^TMx > 0$ para todo o $x \in R^n - \{0\}$

$M \in PSD \Leftrightarrow x^TMx \geq 0$ para todo o $x \in R^n$

$M \in Z \Leftrightarrow m_{ij} \leq 0$ para todo o $i \neq j$

$M \in K \Leftrightarrow M \in Z$ e $M \in P$

Além disso são conhecidas as seguintes relações entre estas classes:

$$K \subset Z \subset Q_o \quad ; \quad K \subset P \subset Q$$

$$PD \subset PSD \subset Q_o \quad ; \quad PD \subset P \subset Q$$

É ainda possível provar [65] que se $M \in P$ então a solução do LCP(q, M) é única para cada q. Nesta tese estuda-se o LCP quando M pertence às classes P e PSD. Nesses casos o LCP diz-se estritamente monótono e monótono respectivamente.

Considere-se o caso em que $M \in PSD$ e é simétrica. Então o Programa Quadrático (QP) definido por

$$\begin{array}{ll} \text{minimize} & q^T z + \tfrac{1}{2} z^T M z \\ \text{sujeito a} & z \geq 0 \end{array} \qquad (1.4)$$

é convexo. As suas condições de Kuhn-Tucker [53]

$$\begin{aligned} & w = q + Mz \\ & w \geq 0 \quad , \quad z \geq 0 \\ & w^T z = 0 \end{aligned} \qquad (1.5)$$

são necessárias e suficientes para a optimalidade. Portanto, qualquer solução do LCP (1.5) é solução óptima do programa quadrático (1.4). Se $M \notin PSD$ e é simétrica então a solução z do LCP é um ponto estacionário do programa (1.4).

Seja agora M uma matriz não simétrica e consideremos o seguinte programa quadrático:

$$\begin{aligned} &\text{minimizar} \quad z^T(q + Mz) = q^T z + \tfrac{1}{2} z^T (M + M^T) z \\ &\text{sujeito a} \quad q + Mz \geq 0 \\ &\qquad\qquad z \geq 0 \end{aligned} \tag{1.6}$$

Facilmente se conclui que qualquer solução do LCP (1.5) é um mínimo global do programa quadrático (1.6) mas o recíproco não é verdadeiro. Se se resolver o programa quadrático (1.6), então três casos podem acontecer:

(i) O programa é não admissível (conjunto de restrições vazio) e o mesmo acontece ao LCP.

(ii) O programa tem mínimo global e o seu valor óptimo é positivo. Então o LCP é admissível mas não tem solução.

(iii) O programa tem mínimo global e o seu valor é nulo. Então esse mínimo é uma solução do LCP.

Em geral este programa é não convexo, pelo que é difícil a determinação do mínimo global do programa quadrático (1.6). Contudo, se $M \in PSD$, então o programa é convexo e a sua resolução é muito mais simples.

Neste trabalho explorar-se-ão essas duas reduções do LCP quando a matriz M é PSD simétrica ou não simétrica. Conforme referimos atrás o LCP diz-se monótono nesse caso.

3 Operações Pivotais

Considere-se um sistema de equações lineares

$$A\,x = b \tag{1.7}$$

com $A \in R^{n\times m}$, $x \in R^{m}$ e $b \in R^{n}$ e suponha-se que $n < m$ e a característica da matriz A é n. Uma solução $\bar{x}$ de (1.7) diz-se básica se existir um conjunto de índices

$$J = \{ j_1, j_2, \ldots, j_n \} \subset \{ 1, 2, \ldots, m \}$$

tal que:

(i) $\bar{x}_j = 0 \quad , \quad j \notin J$

(ii) $B = \{ A_{\cdot j1}, A_{j2}, \ldots, A_{\cdot jn} \}$ é não singular.

onde $A_{\cdot j}$ representa a coluna j de A. As variáveis x_j com $j \in J$ dizem-se básicas enquanto que as restantes variáveis são chamadas não básicas. A matriz B é denominada base.

Reordenando as colunas da matriz A de modo a colocar em primeiro lugar as colunas cujos índices pertencem a J, o sistema (1.7) pode ser escrito na forma:

$$[\,B \quad N\,] \begin{bmatrix} x_J \\ x_I \end{bmatrix} = b$$

com $I = \{ 1, \ldots, m \} - J$ e N a matriz constituída pelas colunas de A cujos índices pertencem a I.

A solução básica associada a esta base será pois

$$x_J = B^{-1}b \;\; e \;\; x_I = 0 \tag{1.8}$$

Uma operação pivotal consiste em transformar a solução básica (1.8) noutra solução básica por troca de uma variável básica por uma variável não básica de modo a obter um sistema equivalente a $Ax = b$.

Pode-se associar à solução básica ($x_J = B^{-1}b$, $x_I = 0$) um quadro da forma:

$$w = \begin{array}{|c|c|} \hline q & M \\ \hline \end{array} \qquad z \tag{1.9}$$

ou seja

$$\begin{array}{c|c|cccccc} & & z_1 & z_2 & \dots & z_s & \dots & z_k \\ \hline w_1 & q_1 & m_{11} & m_{12} & \dots & m_{1s} & \dots & m_{1k} \\ w_2 & q_2 & m_{21} & m_{22} & \dots & m_{2s} & \dots & m_{2k} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ w_r = & q_r & m_{r1} & m_{r2} & \dots & m_{rs} & \dots & m_{rk} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ w_n & q_n & m_{n1} & m_{n2} & \dots & m_{ns} & \dots & m_{nk} \\ \hline \end{array}$$

com $k = m - n$, $w = x_J$, $z = x_I$, $q = B^{-1}b$ e $M = - B^{-1}N$. Se na operação pivotal se trocar a variável não básica z_s com a variável básica w_r, então obtém-se um quadro da forma:

$$\bar{w} = \begin{array}{|c|c|} \hline \bar{q} & \bar{M} \\ \hline \end{array} \qquad \bar{z}$$

$$\text{com} \quad \bar{w}_i = \begin{cases} w_i & \text{se} \quad i \neq r \\ z_s & \text{se} \quad i = r \end{cases} \quad , \quad \bar{z}_i = \begin{cases} z_i & \text{se} \quad i \neq s \\ w_r & \text{se} \quad i = s \end{cases}$$

e os elementos de $\bar{q}$ e $\bar{M}$ dados por :

$$\bar{q}_r = -\frac{q_r}{m_{rs}} \quad ; \quad \bar{q}_i = q_i - m_{is}\frac{q_r}{m_{rs}} \quad (\text{se } i \neq r)$$

$$\bar{m}_{rs} = \frac{1}{m_{rs}} \quad ; \quad \bar{m}_{is} = \frac{m_{is}}{m_{rs}} \quad (\text{se } i \neq r) \quad ; \quad \bar{m}_{rj} = -\frac{m_{rj}}{m_{rs}} \quad (\text{se } j \neq s)$$

$$\bar{m}_{ij} = m_{ij} - m_{is}\frac{m_{rj}}{m_{rs}} \quad (\text{se } i \neq r \text{ e } j \neq s)$$

É de notar que $m_{rs} \neq 0$ para que a operação pivotal possa ter lugar. Esse elemento é chamado pivot. Se r = s e M é uma matriz quadrada, então o pivot é um elemento da diagonal de M e a operação diz-se pivotal principal.

Para que a relação de complementaridade seja verificada numa solução básica não degenerada não se pode ter simultaneamente uma variável e a sua complementar básicas. Portanto, se uma tal solução básica é complementar, então a sua matriz base não pode conter uma coluna da matriz M e a coluna com o mesmo índice da matriz identidade. Nestas condições qualquer solução complementar tem associados os seguintes conjuntos:

$$F = \{ i : z_i \text{ é básica} \} \qquad T = \{ i : w_i \text{ é básica} \}$$

$$F \cap T = \varnothing \quad \text{e} \quad F \cup T = \{ 1, \ldots, n \}$$

Escrevendo o sistema (1.9) na forma

$$\begin{array}{ccc|cc} & & & z_F & z_T \\ w_F & = & q_F & M_{FF} & M_{FT} \\ w_T & = & q_T & M_{TF} & M_{TT} \end{array} \qquad (1.10)$$

então M_{FF} tem de ser não singular e uma operação pivotal com pivot M_{FF} transforma o sistema (1.10) no seguinte sistema equivalente:

$$\begin{array}{ccc|cc} & & & w_F & z_T \\ z_F & = & \bar{q}_F & \bar{M}_{FF} & \bar{M}_{FT} \\ w_T & = & \bar{q}_T & \bar{M}_{TF} & \bar{M}_{TT} \end{array} \qquad (1.11)$$

com

$$\bar{q}_F = - M_{FF}^{-1}\ q_F\ ;\ \bar{q}_T = q_T - M_{TF}\, M_{FF}^{-1} q_F ;$$

$$\bar{M}_{FF} = M_{FF}^{-1}\ ;\ \bar{M}_{FT} = - M_{FF}^{-1} M_{FT}\ ;\ \bar{M}_{TF} = M_{TF} M_{FF}^{-1}\ ;$$

$$\bar{M}_{TT} = M_{TT} - M_{TF} M_{FF}^{-1} M_{FT}$$

Os valores da solução básica associada a esse quadro são:

$$z_F = \bar{q}_F\ ;\ w_T = \bar{q}_T \quad \text{e} \quad z_T = w_F = 0$$

É de notar que se # F > 1 , então a operação pivotal diz-se por bloco e é equivalente a # F operações pivotais simples principais ou não. A matriz

$$\bar{M} = \begin{bmatrix} \bar{M}_{FF} & \bar{M}_{FT} \\ \bar{M}_{TF} & \bar{M}_{TT} \end{bmatrix}$$

diz-se uma transformada principal de M. Além disso,

$$\bar{M}_{TT} = M_{TT} - M_{TF} M_{FF}^{-1} M_{FT}$$

é denominado o Complemento de Schur de M_{FF} em M [8] e representa-se por (M | M_{FF}).

Os métodos directos para o LCP são métodos pivotais que, partindo de uma solução básica inicial (normalmente $w = q$, $z = 0$) e usando sempre soluções complementares, obtêm uma solução básica com $z \geq 0$ e $w \geq 0$. A diferença fundamental entre os métodos consiste no modo como em cada iteração se escolhe o pivot da operação pivotal a efectuar. Como o número de soluções básicas complementares é finito, então os algoritmos terminam num número finito de iterações se não houver repetição de soluções básicas complementares.

4 Algumas Reduções Importantes

Nesta secção ir-se-à mostrar que alguns problemas de optimização se podem reduzir a LCPs monótonos.

(i) Considere-se o problema de programação linear:

$$\begin{array}{ll} \text{Min} & c^T x \\ \text{sujeito a} & Ax \geq b \\ & x \geq 0 \end{array} \qquad (1.12)$$

O seu dual tem a forma:

$$\begin{array}{lll} \text{Max} & b^T y & \\ \text{sujeito a} & A^T y \leq c & (1.13) \\ & y \geq 0 & \end{array}$$

Pela teoria da dualidade da programação linear, $\bar{x}$ é solução óptima do primal (1.12) se e só se existe uma solução óptima do dual (1.13) e, além disso, se verifica a condição de complementaridade das variáveis de folga em relação às variáveis do primal e do dual [65]. Portanto o problema linear (1.12) é equivalente ao LCP :

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c \\ -b \end{bmatrix} + \begin{bmatrix} O & -A^T \\ A & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$u^T x = v^T y = 0 \ ; \ u \geq 0 \ ; \ v \geq 0 \ ; \ x \geq 0 \ ; \ y \geq 0$$

É de notar que esse LCP é monótono, pois a sua matriz é PSD. Recorrer à complementaridade para resolver o programa linear tem sido pouco usual até recentemente, por a dimensão do LCP ser bastante superior à do programa linear. O desenvolvimento de técnicas de pontos interiores veio alterar essa situação.

(ii) Considere-se o programa quadrático

$$\begin{array}{lll} \text{Min} & q^T z + \tfrac{1}{2} z^T M z & \\ \text{sujeito a} & Az \geq b & (1.14) \\ & z \geq 0 & \end{array}$$

em que M é simétrica quadrada de ordem n e $A \in R^{m \times n}$.

Se $\bar{z}$ é solução de (1.14), então também é solução do programa linear

$$\begin{aligned} \text{Min} \quad & (q + M\bar{z})^T z \\ \text{sujeito a} \quad & Az \geq b \\ & z \geq 0 \end{aligned} \tag{1.15}$$

O dual de (1.15) pode ser escrito na forma:

$$\begin{aligned} \text{Max} \quad & b^T y \\ \text{sujeito a} \quad & A^T y \leq q + M\bar{z} \\ & y \geq 0 \end{aligned}$$

Então, pela teoria da dualidade linear, $\bar{z}$ é solução do LCP:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} q \\ -b \end{bmatrix} + \begin{bmatrix} M & -A^T \\ A & O \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} \tag{1.16}$$

$$u^T z = v^T y = 0 \quad ; \quad u \geq 0 \; ; \; v \geq 0 \; ; \; z \geq 0 \; ; \; y \geq 0$$

Portanto qualquer solução óptima do programa (1.14) fornece uma solução para o LCP (1.16). No entanto o recíproco não é em geral verdadeiro. Contudo, tal acontece se $M \in PSD$, isto é, se o programa é convexo. Portanto, um programa quadrático convexo com apenas desigualdades é equivalente a um LCP monótono.

Um caso particular importante deste programa quadrático é aquele em que apenas se impoem limites inferiores e superiores aos valores das variáveis. Esse programa será estudado com bastante detalhe nesta tese e tem a forma

$$\begin{aligned} \text{Min} \quad & q^T z + \tfrac{1}{2} z^T M z \\ \text{sujeito a} \quad & a \leq z \leq b \end{aligned} \tag{1.17}$$

em que a e b são vectores reais com n componentes tais que $a_i < b_i$ $i = 1, \ldots, n$. Sem perda de generalidade pode-se considerar que o vector a é nulo. Então, procedendo

como anteriormente conclui-se que as condições de optimalidade desse programa têm a forma:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} q \\ b \end{bmatrix} + \begin{bmatrix} M & I \\ -I & O \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} \qquad (1.18)$$

$$u^T z = v^T y = 0 \;\; ; \;\; u \geq 0 \; ; \; v \geq 0 \; ; \; z \geq 0 \; ; \; y \geq 0$$

em que I é a matriz identidade de ordem n. Se $M \in PSD$ então os dois problemas (1.17) e (1.18) são equivalentes.

É possivel ainda demonstrar [25] que as condições de optimalidade do programa quadrático (1.17) se podem escrever na seguinte forma:

$$w = q + Mz$$

$$\left.\begin{array}{r} a_i \leq z_i \leq b_i \\ z_i = a_i \Rightarrow w_i \geq 0 \\ z_i = b_i \Rightarrow w_i \leq 0 \\ a_i < z_i < b_i \Rightarrow w_i = 0 \end{array}\right\} \quad i = 1, \ldots, n \qquad (1.19)$$

Este problema, denotado por BLCP, reduz-se a um LCP (1.1) se $b_i = +\infty$ e $a_i = 0$ para todo o $i = 1, \ldots, n$ e ao LCP (1.18) se $a_i = 0$ e $b_i < +\infty$ para todo o $i = 1, \ldots, n$.

Nesta tese o BLCP será estudado com bastante detalhe. Tal como no LCP, o BLCP diz-se monótono se a matriz M é PSD. Se M é simétrica PSD então o BLCP (1.19) é equivalente ao programa quadrático (1.17), onde algumas componentes de a_i e b_i podem ser infinitas.

(iii) Considere-se finalmente o problema de desigualdades variacionais lineares (LVI) na sua forma mais simples:

Encontrar $\bar{z} \in K$ tal que:

$$(q + M \bar{z})^T (z - \bar{z}) \geq 0 \quad \text{para todo o } z \in K$$

onde $$K = \{ z : a_i \leq z_i \leq b_i \ , \ i = 1 , \ldots , n \}$$

O LVI é equivalente ao BLCP (1.19) qualquer que seja a classe da matriz M e as componentes a_i e b_i [34]. Se $a_i = 0$ e $b_i = + \infty$ para todo i = 1, ... , n, então o LVI é equivalente ao LCP (1.1).

CAPÍTULO II

MÉTODOS PIVOTAIS SIMPLES PARA O PROBLEMA LINEAR COMPLEMENTAR

1 INTRODUÇÃO

Consideremos novamente o Problema Linear Complementar (LCP):

$$\begin{aligned} & w = q + Mz \\ & z \geq 0 \\ & w \geq 0 \\ & z^T w = 0 \end{aligned} \tag{2.1}$$

Como foi referido no capítulo anterior, os métodos directos são algoritmos pivotais que usam soluções complementares básicas até que uma solução admissível seja encontrada. Os algoritmos têm diferentes regras segundo as quais são escolhidos os pivots das várias operações pivotais.

Neste capítulo discutiremos os principais métodos para a resolução do LCP monótono. Começaremos por referir sumariamente o conhecido método de Lemke para depois apresentarmos os métodos pivotais principais simples de Murty, Criss-Cross, Keller e Graves. Os algoritmos pivotais por bloco serão discutidos num capítulo posterior. Apresentamos também implementações dos algoritmos pivotais principais para a resolução de LCPs de grandes dimensões com $M \in P$ ou $M \in PSD$ simétrica. Finalmente alguma experiência computacional será incluída que mostra as vantagens e desvantagens destes algoritmos.

Para uma melhor clarificação dos algoritmos deste capítulo é conveniente apresentar algumas propriedades das matrizes P e PSD. Para uma demonstração desses resultados sugere-se [12, 65].

Teorema 2.1

(i) Se $M \in P$ (PD, PSD) então qualquer submatriz principal é P (PD, PSD) .

(ii) Se $M \in P$ (PD) então $m_{ii} > 0$ para todo o $i = 1 , \ldots , n$.

(iii) Se $M \in$ PSD então $m_{ii} \geq 0$ para todo o $i = 1 , \ldots , n$.

(iv) Se $M \in$ PSD e $m_{ii} = 0$ então $m_{ij} = - m_{ji}$ para todo o $j \neq i$.

Se M é simétrica PSD e $m_{ii} = 0$ então $m_{ij} = m_{ji} = 0$ para todo o $j \neq i$.

(vi) Se $M \in P$ (PD, PSD) e $\tilde{M}$ é uma transformada principal de M, então $\tilde{M} \in P$ (PD, PSD).

2 Método de Lemke

Este algoritmo [56] é considerado como o mais conhecido para a resolução do LCP e é também aquele que pode ser utilizado para um maior número de classes de matrizes [12, 65]. Nesse processo são introduzidos uma variável artificial λ e um vector positivo p tais que $q + \lambda p \geq 0$, de modo a obter um LCP alargado da forma

$$w = q + \lambda p + M z \qquad (2.2)$$
$$w \geq 0 , z \geq 0 , z^T w = 0$$

Uma solução que satisfaz (2.2) com $\lambda > 0$ diz-se quase complementar. Se uma solução básica é quase complementar, então existe um par de variáveis

complementares não básicas. É evidente que qualquer solução do LCP (2.1) satisfaz (2.2) com $\lambda = 0$.

O quadro inicial tem assim o aspecto:

		λ	z_1	z_2	...	z_s	...	z_n
w_1		q_1	p_1	m_{11}	m_{12}	... m_{1s}	...	m_{1n}
w_2		q_2	p_2	m_{21}	m_{22}	... m_{2s}	...	m_{2n}
...		...	...	...	...	... ...	...	...
w_r	=	q_r	p_r	m_{r1}	m_{r2}	... m_{rs}	...	m_{rn}
...		...	...	...	...	... ...	...	...
w_n		q_n	p_n	m_{n1}	m_{n2}	... m_{ns}	...	m_{nn}

O algoritmo começa por fazer λ básica por troca com uma variável w_r de modo a obter uma primeira solução quase complementar. A escolha do índice r é feita de acordo com o seguinte critério:

$$-\frac{q_r}{p_r} = \text{máx} \left\{ -\frac{q_i}{p_i} : q_i < 0 \right\}$$

Se se efectuar uma operação pivotal com pivot p_r então obtém-se um quadro da forma

$$\bar{w} = \bar{q} + \bar{A}\,\bar{z}$$

com $\bar{w}_i = w_i$, $i \neq r$, $\bar{w}_r = \lambda$, $\bar{z}_1 = w_r$, $\bar{z}_{i+1} = z_i$, $i = 1, \ldots, n$ e $\bar{q}$ e $\bar{A}$ o vector e a matriz transformados de q e de [p | M] pela operação pivotal.

Na iteração seguinte a variável z_r irá ser tornada básica e a variável a tornar-se não básica vai ser escolhida por um critério de quociente mínimo típico do método simplex:

$$-\frac{\bar{q}_s}{\bar{a}_{sr}} = \min \left\{ -\frac{\bar{q}_i}{\bar{a}_{ir}} : \bar{a}_{ir} < 0 \right\} \qquad (2.3)$$

Se na aplicação do critério (2.3) for detectado que $\bar{a}_{ir} \geq 0$ para $i = 1, \ldots, n$, então a variável não básica de ordem r pode tomar qualquer valor positivo e obtém-se a chamada terminação em aresta ilimitada. De outro modo efectua-se uma operação pivotal com pivot $\bar{a}_{rs}$ e, ou obtém-se uma solução do LCP ($\lambda = 0$), ou uma nova solução quase complementar. No primeiro caso o algoritmo termina, enquanto que no segundo caso uma nova iteração tem de ser efectuada.

O algoritmo pode ser esquematizado na seguinte forma:

Algoritmo de Lemke

PASSO INICIAL: Se $q \geq 0$ então $w = q$, $z = 0$ é a solução.
Caso contrário determine r tal que:

$$-\frac{q_r}{p_r} = \text{máx} \left\{ -\frac{q_i}{p_i} : q_i < 0 \right\}$$

Efectue operação pivotal com pivot p_r.

PASSO GERAL: Escolha a variável não básica que é complementar da última que se tornou não básica. Seja r o seu índice.
i) Se $\bar{a}_{ir} \geq 0$ para $i = 1, \ldots, n$ termine em aresta ilimitada.
ii) Caso contrário determine s a partir de:

$$-\frac{\bar{q}_s}{\bar{a}_{sr}} = \min \left\{ -\frac{\bar{q}_i}{\bar{a}_{ir}} : \bar{a}_{ir} < 0 \right\}$$

Efectue uma operação pivotal com pivot $\bar{a}_{sr}$
a) Se $\lambda = 0$ termine com a solução ($\bar{w} = \bar{q}$, $\bar{z} = 0$).
b) Se $\lambda > 0$ repita o Passo Geral.

Em geral nada se pode concluir sobre a existência de solução quando o método termina em aresta ilimitada. Contudo [11] tem-se

(i) Se $M \in P$ ($M \in PD$) então a terminação em aresta ilimitada não pode ocorrer.

(ii) Se $M \in PSD$ ou $M \in Z$ então a terminação em aresta ilimitada implica que o LCP é não admissível.

Como se pode facilmente constatar o método de Lemke utiliza sempre operações pivotais não principais. Tal como o método simplex para programas lineares, o método de Lemke deve ser implementado usando em cada iteração a factorização LU da matriz base associada à solução básica correspondente. É também necessário usar um processo de actualização dessa factorização sempre que um único elemento é alterado na base [19, 80]. Alguns autores recomendam o uso de transformações ortogonais [79] por razões de precisão numérica. No entanto, não é um processo recomendável para problemas de grande porte, pois tais transformações destroem rapidamente a esparsidade das matrizes.

Se a matriz do LCP é PD ou PSD simétrica o método de Lemke é menos eficiente do que os algoritmos pivotais principais simples que se descrevem nas secções seguintes deste capítulo. Com efeito, contrariamente ao método de Lemke, estes últimos algoritmos têm a vantagem de ser implementáveis usando um processo que explora a simetria da matriz do LCP. Na última secção deste capítulo apresenta-se experiência computacional comparativa da eficiência do método de Lemke e de um dos métodos pivotais principais, que atesta a superioridade desses últimos processos sobre o método de Lemke. Daí não discutirmos o método de Lemke mais detalhadamente nesta tese.

3 MÉTODO DE MURTY

Este algoritmo é o primeiro exemplo de um método pivotal principal simples. Como veremos mais adiante é apenas aplicável a LCPs estritamente monótonos, isto é, a LCPs em que $M \in P$.

Como só se efectuam operações pivotais principais, então, em qualquer iteração, os valores das variáveis básicas são obtidos a partir das expressões:

$$\bar{q}_F = - M_{FF}^{-1} q_F \; ; \; \bar{q}_T = q_T - M_{TF} M_{FF}^{-1} q_F \qquad (2.4)$$

em que, como anteriormente,

$$F = \{ i : z_i \text{ é básica} \}$$

$$T = \{ i : w_i \text{ é básica} \}$$

Diz-se que i é uma inadmissibilidade da solução básica $z = (\bar{q}_F , 0)$, $w = (0 , \bar{q}_T)$ se $\bar{q}_i < 0$, $i \in \{ 1 , \ldots , n \}$. O conjunto H de inadmissibilidades associado à solução básica é definido por $H = \{ i \in F \cup T : \bar{q}_i < 0 \}$. Uma solução básica é não admissível (admissível) se $H \neq \varnothing$ ($H = \varnothing$). O método de Murty utiliza, em cada iteração, soluções básicas não admissíveis até obtenção de uma solução básica admissível que é a solução do LCP.

A escolha da inadmissibilidade a remover é feita de acordo com a regra de Bland, isto é, deve-se escolher o primeiro elemento de H. Como é discutido em [64] uma tal escolha evita que haja repetição de soluções básicas e que, portanto, o método possa entrar em ciclo.

Seja r o índice da inadmissibilidade a remover. Então efectua-se uma operação pivotal principal com pivot $\bar{m}_{rr}$. Com esta operação vão-se permutar os papéis das variáveis z_r e w_r, passando a variável que é básica a não básica e reciprocamente. Esta operação pivotal é sempre possível pois uma matriz P tem elementos diagonais sempre positivos.

A variável básica de ordem r (z_r ou w_r) passará a ter o valor $-\bar{q}_r / \bar{m}_{rr}$ enquanto que os valores das outras variáveis básicas serão modificados a partir de

$$\bar{q}_i - \frac{\bar{m}_{ir}}{\bar{m}_{rr}} \bar{q}_r$$

Este tipo de processo pode ser facilmente apresentado em termos dos conjuntos F e T e das fórmulas (2.4). Com efeito, como apenas operações pivotais principais são usadas e $M \in P$ então, pelo teorema 2.1, há a garantia de que $\bar{m}_{rr} > 0$ e, portanto, não é necessário calcular esse elemento. Por outro lado, cada operação pivotal com pivot $\bar{m}_{rr}$ consiste simplesmente em trocar o índice r de F para T ou inversamente. Tendo em conta estas considerações, podemos escrever os passos do método de Murty na seguinte forma:

Algoritmo de Murty

PASSO INICIAL : Faça $F = \varnothing$, $T = \{ 1, \ldots, n \}$ e $\bar{q} = q$

PASSO GERAL : Seja $H = \{ i : \bar{q}_i < 0 \}$.

Se $H = \varnothing$ a solução é $\begin{cases} z_F = \bar{q}_F & z_T = 0 \\ w_F = 0 & w_T = \bar{q}_T \end{cases}$

Caso contrário, seja:

$$r = \min \{ i : i \in H \}$$

Se $r \in F$ faça $F = F - \{ r \}$, $T = T \cup \{ r \}$.

Se $r \in T$ faça $T = T - \{ r \}$, $F = F \cup \{ r \}$.

Calcule $\bar{q}$ a partir de (2.4) e repita o Passo Geral.

Como é discutido em [64], este algoritmo é convergente se $M \in P$. Como se verá na experiência computacional trata-se de um método que, normalmente, efectua um número de iterações muito superior à dimensão da matriz do LCP. Uma modificação possível será escolher a inadmissiblidade a remover como sendo aquela que corresponde ao valor mais negativo de $\bar{q}_i$. Tal como acontece na resolução de programas lineares pelo método simplex, esta escolha conduz normalmente a uma grande economia no número de iterações. No entanto, apesar de ser raro, o algoritmo pode entrar em ciclo nesse caso [69].

4 Método Criss-Cross

Este processo [49] pode ser visto como uma extensão do método de Murty capaz de resolver LCPs monótonos não estritamente monótonos (isto é, $M \in PSD$ e $M \notin P$). Apesar do algoritmo poder ser aplicado a matrizes PSD não simétricas, tem especial interesse quando M é simétrica. Por essa razão apenas nos iremos debruçar sobre esse caso. No final desta secção apresentaremos algumas considerações breves sobre o uso do algoritmo para matrizes não simétricas PSD.

Suponhamos então que $M \in PSD$ e é simétrica. Se $m_{rr} = 0$, para um determinado índice $r \in \{ 1 , \ldots , n \}$, então, pelo teorema 2.1, $m_{rj} = 0$ para todo o $j = 1 , \ldots , n$. Portanto, se $q_r < 0$ o LCP é não admissível. Por outro lado, se $q_r \geq 0$ então podemos fixar $w_r = q_r$ e $z_r = 0$, retirar a linha e a coluna de ordem r da matriz M e resolver o LCP de ordem n-1 que resta. Podemos assim supor, sem perda de

generalidade, que todos os elementos diagonais de M são positivos. Assim, uma operação pivotal simples pode ser sempre efectuada na iteração inicial. Nas outras iterações é preciso verificar se é positivo o elemento diagonal $\bar{m}_{rr}$ associado à inadmissibilidade r escolhida. Como apenas operações pivotais principais são usadas, então, nessa iteração, tem-se um quadro na forma:

$$\begin{array}{ccc|cc} & & & w_F & z_T \\ z_F & = & \bar{q}_F & \bar{M}_{FF} & \bar{M}_{FT} \\ w_T & = & \bar{q}_T & \bar{M}_{TF} & \bar{M}_{TT} \end{array}$$

onde

$$\bar{M} = \begin{bmatrix} \bar{M}_{FF} & \bar{M}_{FT} \\ \bar{M}_{TF} & \bar{M}_{TT} \end{bmatrix}$$

é a transformada principal de M com pivot M_{FF} não singular.

Se a matriz M é simétrica PSD então $\bar{M}_{FF}$ é simétrica PD. Com efeito, toda a matriz PSD simétrica não singular é PD [12]. Além disso $\bar{M}_{TT}$ é simétrica PSD e $\bar{M}_{TF} = -\bar{M}_{FT}^{T}$. Portanto $\bar{M}$ só poderá ter um elemento diagonal nulo quando $r \in T$.

Seja agora $r \in T$ tal que $\bar{m}_{rr} = 0$. Como $\bar{M}_{TT}$ é simétrica PSD, então a linha e a coluna r de $\bar{M}_{TT}$ são nulas. Além disso a linha r da matriz $\bar{M}_{TF}$ é simétrica da coluna r da matriz $\bar{M}_{FT}$. Dois casos podem agora acontecer e são apresentados a seguir.

(i) Se $\bar{m}_{ir} \geq 0$ para todo o $i \in F$, então $\bar{M}$ tem a seguinte forma

$$\bar{M} = \left[\begin{array}{c|c} & \oplus \\ \bar{M}_{FF} & \cdots \\ & \oplus \\ \hline & 0 \\ & \cdots \\ \Theta\Theta\cdots\Theta & 0\cdots 0\cdots 0 \\ & \cdots \\ & 0 \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ \leftarrow r \\ \\ \\ \end{array}$$

$$\begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ r \end{array}$$

em que $\oplus$ representa um elemento não negativo e Θ representa um elemento não positivo. Então, a variável não básica de ordem r pode tomar qualquer valor positivo. Por outro lado, como a linha r de $\bar{M}_{TF}$ é não positiva, não há qualquer hipótese da variável básica de ordem r se tornar não negativa. Então o LCP não tem solução e o algoritmo termina.

(ii) Se o caso (i) não ocorrer, então podemos escolher o menor índice $s \in T$ tal que $\bar{m}_{sr} < 0$. A matriz

$$\begin{bmatrix} \bar{m}_{ss} & \bar{m}_{sr} \\ \bar{m}_{rs} & \bar{m}_{rr} \end{bmatrix} = \begin{bmatrix} \bar{m}_{ss} & \bar{m}_{sr} \\ \bar{m}_{rs} & 0 \end{bmatrix}$$

é não singular e pode servir como pivot numa operação pivotal. É de notar que essa operação pivotal consiste em modificar os conjuntos F e T a partir de

$$F = F - \{ s \} \cup \{ r \} \quad , \qquad T = T - \{ r \} \cup \{ s \}$$

Uma nova solução complementar é então obtida.

Por outro lado, se $\bar{m}_{rr} > 0$ e $r \in T$ ou $r \in F$, então pode ser efectuada uma operação pivotal principal simples com pivot $\bar{m}_{rr}$, obtendo-se uma nova solução complementar.

Estas considerações conduzem ao seguinte processo:

Algoritmo Criss-Cross

PASSO 0: Faça $F = \varnothing$, $T = \{ 1, \ldots, n \}$ e $\bar{q} = q$.

PASSO 1 : Seja $H = \{ i : \bar{q}_i < 0 \}$.

Se $H = \varnothing$ a solução é $\begin{cases} z_F = \bar{q}_F & z_T = 0 \\ w_F = 0 & w_T = \bar{q}_T \end{cases}$

Caso contrário, seja:

$r = \min \{ i : i \in H \}$.

PASSO 2 : (i) Se $r \in F$, faça:

$F = F - \{ r \}$ e $T = T \cup \{ r \}$.

Calcule $\bar{q}$ e volte ao Passo 1.

(ii) se $r \in T$ calcule $\bar{m}_{rr}$ e $\bar{M}_{Fr}$ e vá para o Passo 3.

PASSO 3 : (i) Se $\bar{m}_{rr} > 0$ faça $F = F \cup \{ r \}$ $T = T - \{ r \}$,

calcule $\bar{q}$ e volte ao Passo 1.

(ii) Se $\bar{m}_{rr} = 0$ vá para o Passo 4.

PASSO 4 : Se $\bar{M}_{Fr} \geq 0$ pare. O LCP é não admissível e não tem solução.

De outro modo seja $s = \min \{ i : \bar{m}_{ir} < 0 \}$.

Faça $F = F - \{ s \} \cup \{ r \}$ e $T = T - \{ r \} \cup \{ s \}$,

calcule $\bar{q}$ e volte a Passo 1.

Seguidamente mostramos que há que resolver um sistema em cada iteração do algoritmo. Se $r \in F$ não é necessário calcular o elemento diagonal de $\bar{M}$. De acordo com (2.4) o vector $\bar{q}$ na próxima iteração é calculado a partir de

$$\begin{array}{ll} \text{Resolva} & M_{FF}\, \bar{q}_F = -\, q_F \\ \text{Calcule} & \bar{q}_T = q_T + M_{TF}\, \bar{q}_F \end{array}$$

Se $r \in T$, então, de acordo com as fórmulas (1.11), $\bar{m}_{rr}$ e $\bar{M}_{Fr}$ podem ser calculados a partir de

$$\begin{array}{ll} \text{Resolva} & M_{FF}\, \bar{M}_{Fr} = -\, M_{Fr} \\ \text{Calcule} & \bar{m}_{rr} = m_{rr} + M_{rF}\, \bar{M}_{Fr} \end{array}$$

onde M_{rF} ($\bar{M}_{Fr}$) é o vector linha (coluna) constituído pelos elementos da linha r de M (coluna r de $\bar{M}$) com índices pertencentes a F. É de notar que, nesse caso, o vector $\bar{q}_F$ da próxima iteração pode ser obtido a partir da actualização

$$\bar{q}_F \leftarrow \bar{q}_F + \bar{M}_{Fr}\, \bar{q}_r$$

Portanto, cada iteração do algoritmo requer a resolução de um só sistema com matriz M_{FF}.

Da descrição do algoritmo conclui-se que ou se obtém uma solução do LCP ou então o LCP é não admissível. Isso confirma o resultado teórico da existência de solução para o LCP monótono.

Se a matriz M não é simétrica, o facto de se encontrar um elemento $r \in T$ tal que $\bar{m}_{rr} = 0$ com todos os elementos da coluna r, em linhas de F, não negativos não garante a não existência de solução. O algoritmo pode no entanto ser modificado para tratar este caso. Contudo, o método de Lemke é considerado mais eficiente para matrizes não simétricas PSD. Por isso não nos iremos debruçar sobre os passos do

algoritmo de Criss-Cross para matrizes não simétricas PSD. Finalmente notemos que se $M \in P$ então o algoritmo Criss-Cross reduz-se ao método de Murty.

5 MÉTODO DE KELLER

O método de Keller [48] foi desenvolvido com o objectivo de obter um ponto estacionário (solução das condições de Kuhn - Tucker associadas) de um programa quadrático com restrições lineares de desigualdade ou concluir que o programa não tem solução óptima. Se o programa quadrático é convexo então esse ponto estacionário é um mínimo global.

Como as condições de Kuhn-Tucker do programa quadrático com apenas valores não negativos nas variáveis (1.4) se reduz ao LCP simétrico (2.1), então o método de Keller permite obter uma solução de um LCP simétrico monótono desde que tal solução exista.

Tal como no método de Murty tenta-se remover uma inadmissibilidade por iteração. A diferença fundamental reside no facto de, em cada iteração, se forçar as variáveis básicas z_i a manterem valores não negativos. Assim sendo, em cada iteração, tem-se $H \subset T$, onde T é o conjunto das variáveis básicas w_i.

Seja então $r \in H$ ($r \in T$). Se se efectuar uma operação pivotal com pivot $\bar{m}_{rr}$ então o novo valor da variável z_r será dado por $-\bar{q}_r / \bar{m}_{rr}$. Isso não traria qualquer problema à variável z_r pois ela tornar-se-ia básica com valor positivo. Contudo, há que ter a garantia que as outras variáveis básicas z_i se mantêm não negativas. Mas, após a pivotagem tem-se:

$$i \in F \Rightarrow z_i = \bar{\bar{q}}_i = \bar{q}_i - \frac{\bar{m}_{ir}}{\bar{m}_{rr}} \bar{q}_r = \bar{q}_i + \bar{m}_{ir} z_r \qquad (2.5)$$

Se acontecer que todas estas quantidades sejam não negativas então pode-se efectuar a operação pivotal. Caso contrário, deve-se retirar da base a primeira variável z_i ($i \in F$) que se vá tornar negativa. Como z_r era nula, então à medida que o seu valor aumenta as outras varáveis z_i $i \in F$ vão variando de acordo com (2.5). Se $\bar{m}_{ir} \geq 0$ para todo o $i \in F$, então as variáveis z_i , $i \in F$, mantêm-se não negativas. Se existir $i \in F$ tal que $\bar{m}_{ir} < 0$ e $-\frac{\bar{q}_i}{\bar{m}_{ir}} < -\frac{\bar{q}_r}{\bar{m}_{rr}}$ então a variável z_i ficaria negativa se tal operação pivotal fosse efectuada. Seja então s o índice definido por

$$s = \min \{ i \in F : -\frac{\bar{q}_i}{\bar{m}_{ir}} = \min \{ -\frac{\bar{q}_j}{\bar{m}_{jr}} : \bar{m}_{jr} < 0 \} \} \qquad (2.6)$$

Como $\bar{M}_{FF}$ é positiva definida, então $\bar{m}_{ss} > 0$ e pode ser efectuada uma operação pivotal principal com esse pivot $\bar{m}_{ss}$. Após essa operação pivotal $F = F - \{ s \}$ e $\bar{q}_i \geq 0$, $i \in F$. Além disso $\bar{q}_r$ mantém-se negativo, pelo que há que investigar de novo se a variável z_r pode ser aumentada ou não. Como F reduz em um elemento, este processo tem de terminar num número finito de iterações.

Tal como já foi visto, se a matriz é PSD pode acontecer que $\bar{m}_{rr} = 0$. Nesse caso é necessário retirar um elemento s a F de tal modo que o índice r possa ser acrescentado sem que se obtenha uma matriz singular. O elemento a retirar deve ser novamente determinado a partir da fórmula (2.6). Se a matriz for simétrica, tal como se viu para o método Criss-Cross, o facto de não existir nenhum elemento negativo na coluna r implica que o problema não tem solução. Contudo, se a matriz não é simétrica já nada se pode concluir. De facto, pode acontecer essa terminação e na

realidade existir solução [48]. Portanto o algoritmo pode ser utilizado para os LCPs cuja matriz é simétrica PSD ou não simétrica P.

Utilizando os conjuntos de indices F e T este algoritmo pode ser descrito sucintamente do seguinte modo:

ALGORITMO DE KELLER

PASSO INICIAL : Faça $F = \varnothing$, $T = \{ 1, \ldots , n\}$ e $\bar{q} = q$.

PASSO 1 : Seja $H = \{ i \in T : \bar{q}_i < 0 \}$.

Se $H = \varnothing$ a solução é $\begin{cases} z_F = \bar{q}_F & z_T = 0 \\ w_F = 0 & w_T = \bar{q}_T \end{cases}$

Caso contrário seja:

$\bar{q}_r = \min \{ \bar{q}_i : i \in H \}$ e vá para o Passo 2.

PASSO 2 : Calcule

$$\theta_1 = \begin{cases} - \bar{q}_r / \bar{m}_{rr} & \text{se} \quad \bar{m}_{rr} > 0 \\ + \infty & \text{se} \quad \bar{m}_{rr} = 0 \end{cases}$$

$$\theta_2 = \begin{cases} - \bar{q}_s / \bar{m}_{sr} = \min \{ - \bar{q}_i / \bar{m}_{ir} : \bar{m}_{ir} < 0 \quad e \quad i \in F \} \\ + \infty \quad \text{se} \quad \bar{m}_{ir} \geq 0 \quad \text{para todo o } i \in F \end{cases}$$

Se $\theta_1 = \theta_2 = + \infty$ o LCP não tem solução e pare.

Se $\theta_1 < \theta_2$ faça $F = F \cup \{ r \}$ $T = T - \{ r \}$.

Calcule $\bar{q}$ e volte ao Passo 1.

Se $\theta_1 \geq \theta_2$ faça $F = F - \{ s \}$ $T = T \cup \{ s \}$.

Calcule $\bar{q}_F$ e $\bar{q}_r$ e repita o Passo 2.

Da descrição do algoritmo verifica-se que, em cada iteração, é necessário resolver dois sistemas com matriz M_{FF}, nomeadamente, um para determinar $\bar{M}_{Fr}$ e outro para actualizar $\bar{q}_F$. No entanto, na maior parte das iterações, pode-se aproveitar o facto de já se conhecer $\bar{M}_{Fr}$ para actualizar $\bar{q}_F$. Com efeito basta usar a relação $\bar{\bar{q}}_F = \bar{q}_F + \bar{M}_{Fr} z_r$, com $z_r = \theta_1$. Este modo de actualizar $\bar{q}_F$ deve ser usado com cuidado quando se retira uma variável da base. Deve-se pensar que tal é equivalente a fazer $z_r = \theta_2$ e, portanto, na iteração seguinte a coluna $\bar{M}_{Fr}$ deve ser multiplicada não por z_r mas pelo seu acréscimo, isto é $z_r - \theta_2$.

6 Método de Graves

Este algoritmo [29] pode ser visto como uma versão pivotal principal do algoritmo de Lemke. Tal como no algoritmo de Lemke é introduzido um vector $p \geq 0$ e um parâmetro $\lambda \geq 0$ tais que $q + \lambda p \geq 0$. A diferença fundamental reside no facto de λ ser aqui tratado como um parâmetro enquanto que no método de Lemke é uma variável que é tornada básica logo na primeira iteração.

O algoritmo de Graves é formulado de tal modo que, em cada iteração, o valor de λ não aumenta e é sempre decrescente desde que não haja soluções degeneradas. Se o LCP tem uma solução ela será encontrada quando se obtiver uma solução básica com $\lambda = 0$.

Como é discutido em [12] a escolha do vector p tem influência no número de iterações necessário para encontrar a solução do LCP. A escolha mais simples é fazer $p_i = 1$, $i = 1, \ldots, n$. Uma outra possibilidade é

$$\begin{cases} p_i = 1 & \text{se} \quad q_i < 0 \\ p_i = 0 & \text{se} \quad q_i \geq 0 \end{cases}$$

Contudo a experiência computacional mostra que esta escolha conduz normalmente a um maior número de iterações, pelo que, em geral, é vantajoso escolher todas as componentes de p iguais a um.

O algoritmo de Graves é assim um processo paramétrico onde se resolve uma sequência de LCP(λ) da forma:

$$w = (q + \lambda p) + Mz$$
$$w \geq 0 , z \geq 0 , z^T w = 0$$

com valores de λ não crescentes, até se obter uma solução do LCP(0). O valor inicial λ_o de λ é tal que $z = 0$ é solução do LCP(λ_o). Portanto $w = q + \lambda_o p \geq 0$ ou seja $\lambda_o = \max \{ -\frac{q_i}{p_i} : q_i < 0 \}$. A este valor vai corresponder o anulamento de uma variável básica . Então efectua-se uma operação pivotal que troca essa variável com a sua complementar.

Em cada iteração é necessário efectuar uma operação pivotal principal com pivot $\tilde{m}_{rr}$, com r o índice da variável básica que se anula quando λ toma o menor valor possível.

Tal como anteriormente $\tilde{m}_{rr} = 0$ pode ocorrer apenas se $r \in T$. Se tal acontecer é impossível realizar a operação pivotal. Neste caso deve-se retirar um

elemento a F, tal como já foi referido para os dois métodos anteriormente expostos. A escolha do elemento a retirar é feita de modo análogo ao anteriormente descrito no método de Keller, mas tendo em atenção a existência do vector p e do parâmetro λ. Assim, o índice s da variável que vai trocar com a complementar será dado por:

$$\frac{\bar{q}_s + \lambda\, \bar{p}_s}{-\bar{m}_{sr}} = \min \left\{ \frac{\bar{q}_i + \lambda\, \bar{p}_i}{-\bar{m}_{ir}} : \bar{m}_{ir} < 0 \ , \ i \in F \right\}$$

Tal como anteriormente, se $M \in PSD$ e simétrica e todos os elementos $\bar{m}_{ir}$ são não negativos, então não há qualquer processo de tornar positiva a variável de ordem r e o problema não tem solução. De outro modo o processo de trocar as variáveis s e r com as complementares é equivalente a efectuar a operação pivotal principal com pivot

$$\begin{bmatrix} 0 & -\bar{m}_{sr} \\ \bar{m}_{sr} & \bar{m}_{ss} \end{bmatrix}$$

Recorrendo mais uma vez aos conjuntos de índices F e T, o algoritmo pode ser esquematizado da forma seguinte.

Algoritmo de Graves

PASSO INICIAL: Faça: $F = \varnothing$, $T = \{ 1, \ldots , n \}$, $\bar{q} = q$ e $\bar{p} = p$.

PASSO 1: Se $\bar{q} \geq 0$ a solução é $\begin{cases} z_F = \bar{q}_F & z_T = 0 \\ w_F = 0 & w_T = \bar{q}_T \end{cases}$

Caso contrário seja:

$$\lambda = -\frac{\bar{q}_r}{\bar{p}_r} = \max \left\{ -\frac{\bar{q}_i}{\bar{p}_i} : \bar{q}_i < 0 \right\}$$

(i) Se $r \in F$ faça $F = F - \{ r \}$ $T = T \cup \{ r \}$,

actualize $\bar{q}$ e $\bar{p}$ e repita o Passo 1.

(ii) Se $r \in T$

- se $\bar{m}_{rr} > 0$ faça $T = T - \{ r \}$ $F = F \cup \{ r \}$ actualize $\bar{q}$ e $\bar{p}$ e repita o passo 1
- se $\bar{m}_{rr} = 0$ vá para Passo 2.

PASSO 2: Se $\bar{M}_{Fr} \geq 0$ o problema não tem solução.

Caso contrário determine s tal que:

$$\frac{\bar{q}_s + \lambda \bar{p}_s}{- \bar{m}_{sr}} = \min \left\{ \frac{\bar{q}_i + \lambda \bar{p}_i}{- \bar{m}_{ir}} : \bar{m}_{ir} < 0 \, , i \in F \right\}$$

Faça $F = F \cup \{ r \} - \{ s \}$, $T = T \cup \{ s \} - \{ r \}$,

actualize $\bar{q}$ e $\bar{p}$ e volte ao Passo 1.

Neste algoritmo há necessidade de resolver dois sistemas por iteração, para actualizar $\bar{q}$ e $\bar{p}$ ou para determinar $\bar{M}_{Fr}$ quando $r \in T$. O cálculo de $\bar{m}_{rr}$ é feito a partir de $\bar{m}_{rr} = m_{rr} + M_{rF} \bar{M}_{Fr}$.

A actualização do vector q é feita de modo semelhante ao explicado nos algoritmos de Criss-Cross e Keller. Além disso, a actualização do vector p é efectuada do mesmo modo que a do vector q, isto é, usando as expressões (2.4) com p em vez de q.

O algoritmo é capaz de processar qualquer LCP monótono (obter solução ou mostrar que o LCP não tem solução) desde que as soluções sejam todas não degeneradas. No caso de haver soluções básicas degeneradas, as regras de ultrapassar

este fenómeno usadas em programação linear podem também aqui ser aplicadas para tornar o algoritmo capaz de processar o LCP também nesse caso.

Da descrição do algoritmo nota-se que cada iteração tem sensivelmente o dobro do trabalho dos outros processos. Daí se induz uma menor eficiência deste método em relação aos anteriores. Isso será confirmado pela experiência computacional apresentada mais adiante neste capítulo.

7 Tratamento do Caso de Variáveis sem Restrição de Sinal

Em muitas aplicações o LCP aparece como as condições de optimalidade do programa quadrático (PQ)

$$\begin{aligned} &\text{Min} && q^T z + \tfrac{1}{2} z^T M z \\ &\text{sujeito a} && z \geq 0 \end{aligned}$$

Em alguns casos, algumas das variáveis z_i não têm restrição de sinal. As condições de Kuhn-Tucker de um tal PQ tomam o aspecto seguinte

$$\left.\begin{array}{l} w = q + M z \\ z_i \geq a_i \ , \ a_i \in \{ -\infty , 0 \} \\ w_i \geq 0 \\ z_i > a_i \ \Rightarrow w_i = 0 \end{array}\right\} \ i = 1, \ldots , n \qquad (2.7)$$

Os métodos directos descritos neste capítulo continuam a resolver este LCP desde que M seja da classe adequada. Para simplificar a exposição definem-se os conjuntos de índices G e K a partir de

$$G = \{ i : a_i = -\infty \} \qquad K = \{ 1 , \ldots , n \} - G$$

Por permutação de linhas e colunas e usando os conjuntos de índices G e K, o LCP (2.7) pode ser escrito na forma

$$\begin{aligned} 0 &= q_G + M_{GG}\, z_G + M_{GK}\, z_K \\ w_K &= q_K + M_{KG}\, z_G + M_{KK}\, z_K \\ z_K &\geq 0\ ,\ w_K \geq 0\ ,\ z_K^T w_K = 0 \end{aligned} \tag{2.8}$$

Resolver a primeira equação em ordem a z_G e substituir na segunda é equivalente a efectuar uma operação pivotal principal com pivot M_{GG}. O LCP (2.8) pode, assim, ser transformado em

$$\begin{aligned} y &= p + A\, x \\ y &\geq 0\ ,\ x \geq 0\ ,\ y^T x = 0 \end{aligned} \tag{2.9}$$

com

$$\begin{aligned} p &= q_K - M_{KG}\, M_{GG}^{-1}\, q_G \\ A &= M_{KK} - M_{KG}\, M_{GG}^{-1}\, M_{GK} = (\, M \setminus M_{KK}\,) \\ y &= w_K\ ,\ x = z_K \end{aligned}$$

Uma vez obtido z_K, o valor de z_G é calculado através de $z_G = -\, M_{GG}^{-1}\, (\, q_G + M_{GK}\, z_K\,)$. Se M é P ou PSD o mesmo acontece com a matriz A e o LCP (2.9) pode ser resolvido pelos algoritmos descritos neste capítulo. Torna-se, no entanto, muito moroso efectuar as transformações que permitem passar de (2.8) a (2.9), além de que, a matriz A, normalmente, será bastante mais densa do que M. Contudo não é difícil constatar que resolver o LCP (2.9) é equivalente a resolver o LCP (2.8) fazendo, na primeira iteração, F = G. No caso de M ser simétrica PSD singular a matriz M_{GG} pode, eventualmente, ser singular. Nesse caso deve-se escolher o maior conjunto $F \subseteq G$ tal que M_{FF} seja não singular. As variáveis z_i, $i \in G - F$, podem ser fixadas em qualquer valor real e, consequentemente, desprezadas, diminuindo assim a dimensão do LCP (2.8). Caso tal não seja possível o LCP não tem solução admissível [35].

Outra abordagem do problema consiste em introduzir uma variável $z_{n+1} \geq 0$ e efectuar a mudança de variáveis $z_G \leftarrow z_G - z_{n+1}\, e_G$, com e_G um vector com #G

componentes todas iguais a 1 e as novas variáveis z_G todas não negativas. O LCP (2.8) toma então o aspecto

$$y = p + A\,x$$
$$y \geq 0\,,\; x \geq 0\,,\; x^T y = 0$$

$$\text{com } p = \begin{bmatrix} q_G \\ q_K \\ q_G^T e_G \end{bmatrix},\quad A = \begin{bmatrix} M_{GG} & M_{GK} & -M_{GG}\,e_G \\ M_{KG} & M_{KK} & -M_{KG}\,e_G \\ -e_G^T M_{GG} & -e_G^T M_{GG} & e_G^T M_{GG}\,e_G \end{bmatrix},\quad y = \begin{bmatrix} w_G \\ w_K \\ w_{n+1} \end{bmatrix},\quad x = \begin{bmatrix} z_G \\ z_K \\ z_{n+1} \end{bmatrix}.$$

ou seja

$$A = \left[\begin{array}{c|c} M & c \\ \hline c^T & \alpha \end{array}\right]$$

com $\alpha = e_G^T M_{GG}\, e_G = \sum_{i,j \in G} m_{ij}$

$$c = -M_{.G}\, e_G \;\Rightarrow\; c_i = -\sum_{j \in G} m_{ij}$$

Cada matriz A_{FF} é da forma $\left[\begin{array}{c|c} M_{FF} & c_F \\ \hline c_F^T & \alpha \end{array}\right]$ ou M_{FF} conforme z_{n+1} seja ou não básica respectivamente. No primeiro caso a determinação de $\tilde{p}_F$ é feita a partir de

Resolva $\quad M_{FF}\, x = -\,p_F$

Resolva $\quad M_{FF}\, \beta = c_F$

Calcule $\quad \tilde{p}_{n+1} = \dfrac{-p_{n+1} - c_F^T x}{\alpha - c_F^T \beta}$

Calcule $\quad \tilde{p}_F = x - \tilde{p}_{n+1}\, \beta$

Este último método pode ser sempre usado mas tem, em relação ao anterior, a principal desvantagem de aumentar o esforço computacional para actualizar a solução

do LCP quando z_{n+1} é básica. Além disso, a matriz A é singular, pelo que se passa de um LCP estritamente monótono para um LCP monótono.

Em [39] apresentamos experiência computacional comprovativa do fraco desempenho deste último processo. Por isso, optámos por usar o primeiro processo sempre que há variáveis sem restrição de sinal.

8 Implementação Para Matrizes Esparsas Gerais

Da descrição dos algoritmos desta secção chega-se à conclusão que a parte mais importante de cada iteração se resume à resolução de um ou dois sistemas de equações lineares com a matriz não singular M_{FF}. Se M é simétrica positiva definida ou semi-definida, então M_{FF} é simétrica positiva definida. Por outro lado se $M \in P$ não simétrica o mesmo acontece a M_{FF}. Nesta secção vamos debruçar-nos sobre o processo de implementação dos algoritmos directos quando M é uma matriz esparsa simétrica PD ou PSD [35, 37]. Algumas considerações sobre o caso de $M \in P$ não simétrica serão apresentadas na parte final desta secção.

Se $A = M_{FF}$ é simétrica PD, então não é necessário efectuar escolha parcial de pivot para garantir a estabilidade da decomposição, podendo qualquer elemento diagonal ser tomado como pivot. Usa-se então uma variante simétrica da eliminação de Gauss para obter a factorização de Cholesky $A = LDL^T$, em que L é uma matriz triangular inferior com elementos diagonais iguais a 1 e D é uma matriz diagonal de elementos diagonais positivos. A vantagem desta factorização sobre a conhecida factorização LL^T reside no facto de se evitar o cálculo de raízes quadradas, que tornam o processo de decomposição mais demorado e mais sujeito a erros de arredondamento.

Uma vez efectuada a factorização, a resolução do sistema $A\,x = b$ reduz-se à resolução de dois sistemas triangulares (um superior e outro inferior) e um diagonal. Com efeito tem-se

$$A\,x = b \quad <=> \quad LDL^T\,x = b \quad <=> \quad (Lz = b \;\wedge\; Dy = z \;\wedge\; L^T x = y)$$

Se M é uma matriz esparsa, então também $A = M_{FF}$ o é e a decomposição LDL^T tem de ser efectuada de modo a tirar partido desse facto.

Antes propriamente de descrever o processo de implementação para a resolução do sistema $A\,x = b$, necessitamos de explicar como é armazenada a matriz M. Existem vários processos de armazenagem de matrizes esparsas que aparecem descritos em [24] e [77]. Como neste tipo de algoritmos tem especial importância o valor do elemento diagonal e também porque se vão efectuar factorizações LDL^T em que os elementos diagonais são tratados de modo diferente dos outros, optou-se por armazenar a matriz M com a diagonal à parte. Os outros elementos não nulos são guardados por colunas havendo um vector que dá o índice da linha ocupada por cada elemento. Um vector de ponteiros, com a indicação da posição ocupada pelo primeiro elemento de cada coluna, completa a estrutura necessária para guardar a matriz M do LCP.

Seja nonz o número de elementos não nulos da matriz M de dimensão n. Para guardar todos os dados do LCP(q, M) são necessários os seguintes vectores:

2 vectores reais de dimensão n (para o vector q e para a diagonal de M)

1 vector inteiro de dimensão n+1 (ponteiro para o início das colunas)

1 vector inteiro de dimensão nonz (índices de linhas)

1 vector real de dimensão nonz (valores numéricos de M)

Esta estrutura de dados é ilustrada com o seguinte exemplo, onde se considera o LCP(q, M) com matriz M e vector q seguintes:

$$M = \begin{bmatrix} 6 & 0 & 0 & -4 & 1 \\ 0 & 6 & 1 & 0 & 0 \\ 0 & 1 & 3 & 0 & 0 \\ -4 & 0 & 0 & 10 & 1 \\ 1 & 0 & 0 & 1 & 10 \end{bmatrix} \qquad q = \begin{bmatrix} -2 \\ -5 \\ 2 \\ 0 \\ -3 \end{bmatrix}$$

Essa matriz e vector são armazenados usando os seguintes vectores:

$$Q = (\ -2.\ ,\ -5.\ ,\ 2.\ ,\ 0.\ ,\ -3.\ ) \qquad D = (\ 6.\ ,6.\ ,\ 3.\ ,\ 10.\ ,\ 10.\ )$$

$$PONT = (\ 1\ ,\ 3\ ,\ 4\ ,\ 5\ ,\ 7\ ,\ 9\ ) \qquad IL = (\ 4\ ,\ 5\ ,\ 3\ ,\ 2\ ,\ 1\ ,\ 5\ ,\ 1\ ,\ 4\ )$$

$$M=(\ -4.\ ,\ 1.\ ,\ 1.\ ,\ 1.,\ -4.,\ 1.\ ,\ 1.\ ,\ 1.\ )$$

Como se trata de matrizes simétricas poder-se-ia pensar em armazenar apenas uma das partes triangulares e a diagonal da matriz. Contudo, esse tipo de armazenagem conduziria a algumas complicações na construção das matrizes M_{FF} e M_{FT} que são necessárias para calcular o vector $\bar{q}$ em cada iteração a partir de

$$M_{FF}\ \bar{q}_F = -\ q_F\ ,\ \ \bar{q}_T = q_T + M_{TF}\,\bar{q}_F$$

Assim, optámos por gastar um pouco mais de espaço e guardar a matriz completa do problema, não tirando, portanto, partido da sua simetria, mas conseguindo assim economias muito significativas no tempo de resolução dos problemas.

Voltemos agora à resolução do sistema com matriz $A = M_{FF}$. Quando se fala em factorização de matrizes esparsas a principal preocupação é a de reduzir os chamados enchimentos (fill-in), isto é, reduzir o mais possível o número de elementos que, sendo zeros na matriz inicial, se vão tornar não nulos devido às operações de factorização. Considere-se por exemplo a seguinte matriz:

$$A = \begin{bmatrix} + & * & * & * & * \\ * & + & 0 & 0 & 0 \\ * & 0 & + & 0 & 0 \\ * & 0 & 0 & + & 0 \\ * & 0 & 0 & 0 & + \end{bmatrix}$$

em que + representa o elemento da diagonal e * representa uma posição ocupada por um elemento não nulo. Efectuada a factorização ir-se-à obter o triângulo inferior L constituído por elementos todos não nulos, isto é, são criados 6 enchimentos. Parte-se do princípio que não vão existir elementos que se anulem no decorrer da factorização, pois essa circunstância é tão rara que não merece ser considerada.

Como uma matriz quadrada de permutação é ortogonal ($P^{-1} = P^T$ e $P^TP = I$), então resolver o sistema $Ax = b$ é equivalente a resolver o sistema $PAP^T(Px) = Pb$ com P uma matriz de permutação. Aplicando a matriz de permutação:

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

à matriz A obtém-se:

$$PAP^T = \begin{bmatrix} + & 0 & 0 & 0 & * \\ 0 & + & 0 & 0 & * \\ 0 & 0 & + & 0 & * \\ 0 & 0 & 0 & + & * \\ * & * & * & * & + \end{bmatrix}$$

e a factorização desta nova matriz não vai criar quaisquer enchimentos.

Este exemplo mostra a importância da reordenação das linhas e colunas da matriz A na resolução de sistemas de equações lineares, com o fim de reduzir os enchimentos durante a factorização. Várias heurísticas de ordenação têm sido propostas para esse fim [24]. Neste trabalho optou-se pelo algoritmo de ordenação de grau mínimo [24] por ser uma técnica que mostra um bom funcionamento para o

caso de matrizes esparsas gerais sem qualquer distribuição especial dos elementos não nulos. Este algoritmo pode ser encarado como a variante simétrica do esquema de Markowitz [60]. O algoritmo vai ordenando a matriz ao mesmo tempo que vai prevendo os enchimentos que irão ocorrer quando a linha/coluna ordenada for eliminada.

O algoritmo de grau mínimo pode ser descrito facilmente em termos de ordenação de um grafo simétrico. Seja $G_o = (X_o , E_o)$ o grafo simétrico associado à matriz M. Este grafo tem n nós, cada um correspondendo a uma linha/coluna de M e nonz arestas correspondentes aos elementos não nulos de M. O algoritmo pode ser descrito do seguinte modo:

Algoritmo de Grau Mínimo

PASSO 1 : $i = 1$.

PASSO 2 : $G_{i-1} = (X_{i-1} , E_{i-1})$.

PASSO 3 : Seja x_i o nó de grau mínimo em G_{i-1} .

PASSO 4 : Eliminar x_i de X_{i-1} e formar o grafo $G_i = (X_i , E_i)$.

PASSO 5 : $i = i + 1$, se $i > \# X$ pára ; caso contrário volta a Passo 2.

Escolher um nó de grau mínimo significa escolher a linha/coluna da matriz com menor número de elementos não nulos. Como cada par de elementos não nulos dá origem a uma aresta, os conceitos de grau mínimo e de menor número de elementos por linha são equivalentes.

Eliminar um nó do grafo significa retirar esse nó do conjunto dos nós, retirar do conjunto das arestas as arestas incidentes a esse nó e acrescentar arestas de modo a que os nós entre os quais existia um caminho de comprimento 2, passando pelo nó

eliminado, continuem ligados, agora directamente. O número de arestas que vão ser acrescentadas corresponde exactamente aos enchimentos que vão sendo obtidos no decorrer da operação de pivotagem com o elemento diagonal correspondente ao nó eliminado. Com efeito seja

$$M = \begin{bmatrix} m_{11} & v_1^T \\ v_1 & M_1 \end{bmatrix} = \begin{bmatrix} 1 & O \\ \frac{v_1}{m_{11}} & I_{n-1} \end{bmatrix} \begin{bmatrix} m_{11} & O \\ O & \bar{M}_1 \end{bmatrix} \begin{bmatrix} 1 & \frac{v_1^T}{m_{11}} \\ O & I_{n-1} \end{bmatrix}$$

onde I_{n-1} representa a matriz identidade de ordem n-1 e $\bar{M}_1 = M_1 - \frac{1}{m_{11}} v_1 v_1^T$. A matriz $\bar{M}_1$ terá um elemento não nulo na posição (k,j) se $m_{kj} \neq 0$ ou se $(v_1)_k \neq 0$ e $(v_1)_j \neq 0$. A ocorrência de enchimento dá-se na situação de ser $m_{kj} = 0$ e simultaneamente $(v_1)_k \neq 0$ e $(v_1)_j \neq 0$. Mas isto significa que existem as arestas (x_1, x_j) e (x_1, x_k) e que não existia a aresta (x_j, x_k). Assim por eliminação do nó x_1 é criada a aresta (x_j, x_k). Repetindo o processo com a matriz $\bar{M}_1$ obtém-se um resultado idêntico ao alcançado com o algoritmo anteriormente descrito.

Da descrição dos métodos pivotais principais conclui-se que há uma matriz M_{FF} que joga um papel fundamental, por ser usada para resolver um ou dois sistemas por iteração. Na iteração seguinte, esta matriz é alterada ficando com mais ou com menos uma linha e coluna. Quando a alteração é feita por inserção de linha e coluna convém saber em que ordem devem ser estas linhas e colunas acrescentadas à matriz. Em termos de actualização da factorização seria muito mais simples se se acrescentasse no fim. Mas esse procedimento conduziria, a curto prazo, a um número incomportável de enchimentos. Poder-se-ia pensar em executar um algoritmo de ordenação de cada vez que um novo elemento é acrescentado a F. Além da ordenação ser um procedimento demorado, haveria sempre necessidade de criar uma nova estrutura para conter a factorização de cada actualização. Além disso, como se verá

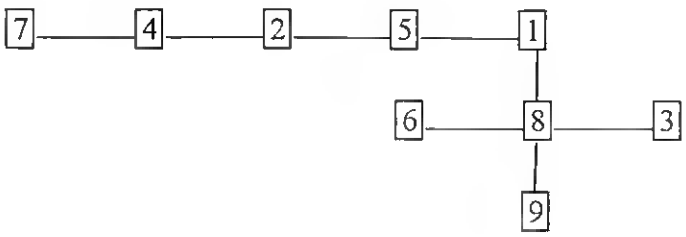
mais adiante, seria difícil aproveitar a decomposição correspondente a uma iteração para obter a decomposição da iteração seguinte.

À matriz M_{FF} corresponde um grafo G_o' que é subgrafo de G_o formado pelos nós de X_o cujo índice pertence a F e pelas arestas de E_o que unem esses nós. O facto de um nó ter grau mínimo em G_o não significa que ele tenha também grau mínimo em G_o' , mas tem boas probabilidades de o ter. Então, mantendo em M_{FF} a ordem já definida para a matriz M, pode não se obter a "melhor" factorização, em termos de redução de enchimentos para M_{FF}, mas tem-se boas hipóteses de obter uma "boa" factorização. Além disso, se no início a matriz M for factorizada simbolicamente e for reservado espaço para essa factorização, esse espaço será suficiente para conter as factorizações das submatrizes M_{FF} de cada iteração.

Como ilustração do que acabámos de dizer, consideremos a matriz de dados:

$$M = \begin{bmatrix} + & 0 & 0 & 0 & * & 0 & 0 & * & 0 \\ 0 & + & 0 & * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & + & 0 & 0 & 0 & 0 & * & 0 \\ 0 & * & 0 & + & 0 & 0 & * & 0 & 0 \\ * & * & 0 & 0 & + & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & * & + & 0 & * & 0 \\ 0 & 0 & 0 & * & 0 & 0 & + & 0 & 0 \\ * & 0 & * & 0 & 0 & * & 0 & + & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & + \end{bmatrix}$$

que tem associado o seguinte grafo:



A parte simbólica dessa matriz é armazenada na seguinte estrutura

PONT = (1 , 3 , 5 , 6 , 8 , 11 , 13 , 14 , 18 , 19)

IL = (5 , 8 , 4 , 5 , 8 , 2 , 7 , 1 , 2 , 6 , 5 , 8 , 4 , 1, 3 , 6 , 9, 8)

Como existe um caminho do nó 5 para o nó 8 passando pelo nó 1, então, ao eliminar este nó, vai aparecer uma aresta ligando directamente os nós 5 e 8. Em termos da matriz transformada significa que vão ser criados elementos não nulos na linha 5, coluna 8 e também na linha 8, coluna 5. Prosseguindo, ao eliminar o nó 2, vão aparecer elementos não nulos na linha 5, coluna 4 e na coluna 4, linha 5. Continuando este processo obtém-se uma matriz L com o seguinte padrão de não zeros:

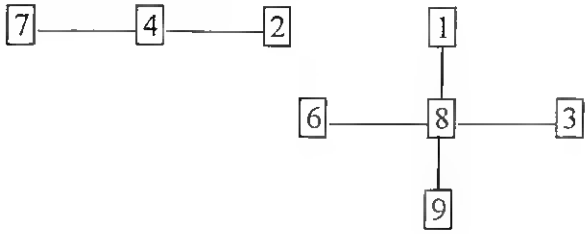
$$L = \begin{bmatrix} + & & & & & & & \\ 0 & + & & & & & & \\ 0 & 0 & + & & & & & \\ 0 & * & 0 & + & & & & \\ * & * & 0 & \circledast & + & & & \\ 0 & 0 & 0 & 0 & * & + & & \\ 0 & 0 & 0 & * & \circledast & \circledast & + & \\ * & 0 & * & 0 & \circledast & * & \circledast & + \\ 0 & 0 & 0 & 0 & 0 & 0 & * & + \end{bmatrix}$$

onde $\circledast$ representa um elemento não nulo que vai ser criado na factorização.

Suponha-se agora que M_{FF} é obtida de M por supressão da linha e coluna 5:

$$M_{FF} = \begin{bmatrix} + & 0 & 0 & 0 & 0 & * & 0 \\ 0 & + & 0 & * & 0 & 0 & 0 \\ 0 & 0 & + & 0 & 0 & * & 0 \\ 0 & * & 0 & + & 0 & * & 0 \\ 0 & 0 & 0 & 0 & + & 0 & * \\ 0 & 0 & 0 & * & 0 & + & 0 \\ * & 0 & * & 0 & * & 0 & + \\ 0 & 0 & 0 & 0 & 0 & * & + \end{bmatrix}$$

A esta matriz corresponde o grafo desconexo:



A eliminação de qualquer nó deste grafo não vai levar à criação de quaisquer novas arestas. Assim a matriz L correspondente à factorização não vai ter quaisquer enchimentos.

Estes exemplos ilustram que o conjunto de enchimentos que ocorre na factorização de qualquer submatriz principal M_{FF} de M é um subconjunto dos enchimentos associados à factorização de M, desde que a ordenação para M_{FF} seja a mesma de M. Este resultado é uma consequência do seguinte teorema:

Teorema 2.2 [24]: Seja $G_L = (X_0 , E_L)$ o grafo associado à matriz $L + L^T$. Se $\Gamma (x_i , S_{i-1})$ é o conjunto dos nós y para os quais existe um caminho que os une a x_i passando exclusivamente por nós de $S_{i-1} = \{ x_1, x_2, \ldots , x_{i-1} \}$, então

$$E_L = \{ (x_i , x_j) : x_j \in \Gamma (x_i , S_{i-1}) \}.$$

Seja agora $M_{FF} = L_F D_F L_F^T$ e $G_F = (X_F , E_F)$ o grafo associado à matriz $L_F + L_F^T$. Então

$$E_F = \{(x_i , x_j) : x_i \in F \text{ e } x_j \in \Gamma (x_i , S_{i-1}^F) \}.$$

Se se mantiver a ordenação do grau mínimo para a matriz M, então $S_{i-1}^F \subset S_{i-1}$ e, também, $E_F \subset E_L$.

Este resultado mostra que é apenas necessário determinar uma ordenação de grau mínimo para a matriz M, o que pode ser feito de acordo com a implementação descrita em [24]. Após esse processo, é fácil de construir a estrutura de dados que vai armazenar os elementos não nulos de M_{FF} e os enchimentos que irão ocorrer durante a factorização dessa matriz (de acordo com a ordem obtida pelo algoritmo de grau mínimo).

Se nl representa o número de não zeros de L (não zeros de M_{FF} e enchimentos) então a estrutura de dados é constituída pelos seguintes vectores:

1 vector real de dimensão n para guardar os elementos diagonais de D

1 vector inteiro de dimensão n+1 para indicar o início de cada coluna

1 vector inteiro de dimensão nl para guardar os índices de linha

1 vector real de dimensão nl para guardar os elementos de L

Além disso, em cada iteração, é necessário manter a informação sobre a ordem estabelecida para as linhas e colunas da matriz, sendo para isso usado um vector de dimensão n que, definido no início, nunca mais é alterado. É também importante conhecer em cada iteração quais os índices de { 1, ... , n } que pertencem a F e a T. Para isso usa-se um vector que guarda os elementos de F na ordem dada inicialmente. Para que as factorizações sejam mais rápidas convém ter também disponível um vector com a informação inversa do anterior, isto é, um vector que dê a informação sobre a ordem de uma dada coluna. Aproveita-se este vector para saber rapidamente se um elemento é de F ou T. Se esse elemento pertence a F a informação é o índice da coluna em M_{FF} . Por outro lado há um 0 na posição correspondente se esse elemento pertence a T.

Em cada iteração é acrescentado ou retirado um elemento a F. É assim essencial encontrar um processo expedito de encontrar a factorização LDL^T da matriz de uma dada iteração à custa da factorização obtida na iteração anterior.

Considere-se primeiramente o caso de se inserir um elemento em F. Tal procedimento resulta em acrescentar uma linha e uma coluna à matriz . Escrevamos $A = M_{FF}$ na seguinte forma

$$A = \begin{bmatrix} A_1 & A_3^T \\ A_3 & A_2 \end{bmatrix}$$

em que A_1 e A_2 são matrizes quadradas. Seja B a matriz que se obtém de A acrescentando uma linha e uma coluna:

$$B = \begin{bmatrix} A_1 & b_1 & A_3^T \\ b_1^T & \alpha & b_2^T \\ A_3 & b_2 & A_2 \end{bmatrix}$$

Suponha-se conhecida a decomposição LDL^T da matriz A, dada por

$$A = LDL^T = \begin{bmatrix} L_1 & \\ E & L_2 \end{bmatrix} \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} \begin{bmatrix} L_1^T & E^T \\ & L_2^T \end{bmatrix} \qquad (2.10)$$

Pretende-se a factorização de B, ou seja,

$$B = \bar{L}\,\bar{D}\,\bar{L}^T = \begin{bmatrix} \bar{L}_1 & & \\ c_1^T & 1 & \\ \bar{E} & c_2 & \bar{L}_2 \end{bmatrix} \begin{bmatrix} \bar{D}_1 & & \\ & \beta & \\ & & \bar{D}_2 \end{bmatrix} \begin{bmatrix} \bar{L}_1^T & c_1 & \bar{E}^T \\ & 1 & c_2^T \\ & & \bar{L}_2^T \end{bmatrix} \qquad (2.11)$$

Efectuando os produtos em (2.10) e em (2.11) tem-se

$$A = \begin{bmatrix} L_1 D_1 L_1^T & L_1 D_1 E^T \\ E D_1 L_1^T & E D_1 E^T + L_2 D_2 L_2^T \end{bmatrix}$$

$$B = \begin{bmatrix} \bar{L}_1 \bar{D}_1 \bar{L}_1^T & \bar{L}_1 \bar{D}_1 c_1 & \bar{L}_1 \bar{D}_1 \bar{E}^T \\ c_1 \bar{D}_1 \bar{L}_1^T & c_1^T \bar{D}_1 c_1 + \beta & c_1 \bar{D}_1 \bar{E}^T + \beta c_2^T \\ \bar{E} \bar{D}_1 \bar{L}_1^T & \bar{E} \bar{D}_1 c_1 + \beta c_2 & \bar{E} \bar{D}_1 \bar{E}^T + \beta c_2 c_2^T + \bar{L}_2 \bar{D}_2 \bar{L}_2^T \end{bmatrix}$$

Como a decomposição LDL^T de A_1 é única, então $\bar{L}_1 = L_1$ e $\bar{D}_1 = D_1$. Então também é $\bar{E} = E$. Além disso, igualando as expressões equivalentes, vem:

$L_1 D_1 c_1 = b_1$

$$\beta + c_1^T D_1 c_1 = \alpha \qquad \Rightarrow \quad \beta = \alpha - c_1^T D_1 c_1$$

$$E D_1 c_1 + \beta\, c_2 = b_2 \quad \Rightarrow \quad c_2 = (b_2 - E D_1 c_1) / \beta$$

$$E D_1 E^T + \beta\, c_2\, c_2^T + \bar{L}_2 \bar{D}_2 \bar{L}_2^T = A_2 = E D_1 E^T + L_2 D_2 L_2^T \quad \Rightarrow$$

$$\Rightarrow \quad \bar{L}_2 \bar{D}_2 \bar{L}_2^T = L_2 D_2 L_2^T - \beta\, c_2\, c_2^T$$

Estas fórmulas permitem calcular as quantidades c_1, c_2, β, $\bar{L}_2$ e $\bar{D}_2$ e assim obter a decomposição LDL^T de B.

É muito semelhante o processo de actualizar a factorização quando se retira uma linha e uma coluna à matriz cuja factorização é conhecida. Com efeito seja B a matriz obtida de A por supressão de uma linha e de uma coluna. Então

$$A = \begin{bmatrix} A_1 & b_1 & A_3^T \\ b_1^T & \alpha & b_2^T \\ A_3 & b_2 & A_2 \end{bmatrix} \quad , \quad B = \begin{bmatrix} A_1 & A_3^T \\ A_3 & A_2 \end{bmatrix} \tag{2.12}$$

e

$$A = LDL^T = \begin{bmatrix} L_1 & & \\ c_1^T & 1 & \\ E & c_2 & L_2 \end{bmatrix} \begin{bmatrix} D_1 & & \\ & \beta & \\ & & D_2 \end{bmatrix} \begin{bmatrix} L_1^T & c_1 & E^T \\ & 1 & c_2^T \\ & & L_2^T \end{bmatrix} \tag{2.13}$$

A decomposição LDL^T de B é dada por:

$$B = \bar{L}\, \bar{D}\, \bar{L}^T = \begin{bmatrix} \bar{L}_1 & \\ \bar{E} & \bar{L}_2 \end{bmatrix} \begin{bmatrix} \bar{D}_1 & \\ & \bar{D}_2 \end{bmatrix} \begin{bmatrix} \bar{L}_1^T & \bar{E}^T \\ & \bar{L}_2^T \end{bmatrix} \tag{2.14}$$

Se, tal como anteriormente, efectuarmos os produtos em (2.13) e em (2.14) e compararmos a matriz resultante com as matrizes (2.12) obtemos as relações que permitem calcular $\bar{L}_i$, $\bar{D}_i$ e $\bar{L}_i^T$ i = 1, 2 e E. Assim tem-se:

$$\bar{L}_1 = L_1$$
$$\bar{D}_1 = D_1$$
$$\bar{E} = E$$
$$\bar{L}_2\bar{D}_2\bar{L}_2^T = L_2D_2L_2^T + \beta\, c_2c_2^T$$

Podemos assim concluir que, quando se acrescenta uma linha e coluna, a actualização da decomposição LDL^T consiste em:

(i) Resolver um sistema triangular

(ii) Calcular um produto interno

(iii) Calcular um produto matriz por vector

(iv) Actualizar uma factorização de uma matriz a que foi feita uma correcção de característica um (rank-one).

Quando se retira uma linha e coluna é necessário apagar a informação respeitante à linha que desapareceu da estrutura que guarda a matriz L. Como se armazena L por colunas é preciso apagar um elemento em cada coluna. Para isso resolve-se simbolicamente o sistema $L_1\, b_1 = c_1$ para se saber quais as colunas que contêm os elementos a apagar. Além disso é necessário também efectuar a actualização de uma factorização quando se faz uma correcção de característica um à matriz de que se conhece a factorização.

Para efectuar a correcção de característica um utiliza-se o algoritmo de Bennett [4], seguindo a forma específica para matrizes esparsas desenvolvida em [54]. Esse processo é descrito a seguir.

Seja $B = A + \beta\, c\, c^T$ com A e B matrizes quadradas de dimensão n, β um número real não nulo e c um vector com n componentes e suponha-se conhecida a decomposição LDL^T da matriz A. O algoritmo de Bennett permite obter a factorização da matriz $B = \bar{L}\bar{D}\bar{L}^T$ a partir da factorização de A sem ser necessário obter explicitamente a matriz B. O processo é recursivo e vai acumulando num vector a contribuição do vector c para as actualizações que vão sendo feitas coluna a coluna.

Como as fórmulas que permitem obter $A = LDL^T$ têm a forma:

$$\text{Para } j = 1, \ldots, n \quad \begin{cases} d_j = a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 d_k \\ l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{jk}\, l_{ik}\, d_k \right) / d_j \quad i > j \end{cases}$$

então, analisando o efeito que produz nestas fórmulas a correcção $\beta\, c\, c^T$, obtemos o seguinte processo

Algoritmo de Bennett

$$\text{Para } j = 1, \ldots, n \quad \begin{cases} p = \beta\, c_j \\ d_j \leftarrow d_j + c_j\, p \\ p \leftarrow p / d_j \\ \left.\begin{array}{l} c_i \leftarrow c_i - l_{ij}\, c_j \\ l_{ij} \leftarrow l_{ij} + c_i\, p \end{array}\right\} \quad i > j \\ \beta \leftarrow \beta - p^2\, d_j \end{cases}$$

Quando a matriz A é esparsa e o vector c tem um grande número de elementos nulos, a matriz $\beta\, c\, c^T$ vai ter um grande número de linhas e colunas nulas. Assim as matrizes B e A só vão diferir num pequeno número de elementos e, é de esperar, que o mesmo aconteça com L e $\bar{L}$. Repare-se que se $c_j = 0$ nada acontece

para a coluna j. Como o vector c vai sendo alterado à medida que o algoritmo prossegue tem que se prever no início quais as componentes do vector que se irão tornar não nulas. Analizando a fórmula de alteração das componentes de c ($c_i \leftarrow c_i - l_{ij} c_j$, i = j+1 , ... , n) verifica-se que ela é semelhante à utilizada para resolver o sistema Ly = c. Começa-se então por, simbolicamente, detectar quais as componentes de y que vão ser não nulas e, ao executar o algoritmo de Bennett, percorrem-se só as colunas e linhas cujo índice corresponde a uma componente não nula de y. Para resolver simbolicamente o sistema Ly = c é necessário considerar um vector LINK em que a componente de ordem j indica qual o índice da primeira linha onde existe um elemento não nulo na coluna j que não é o elemento da diagonal. Se $c_j \neq 0$ as componentes

$$j\ ,\ LINK(j)\ ,\ LINK(LINK(j))\ ,\ LINK(LINK(...(LINK(j))...)) \qquad (2.15)$$

do vector y serão todas não nulas. Sempre que uma coluna não tenha elementos não diagonais não nulos, a informação correspondente no vector LINK deve ser zero. Assim, a sucessão de índices construída em (2.15) termina quando um zero for encontrado. Deste modo constrói-se o conjunto de índices $NZ = \{ i : y_i \neq 0 \}$.

Seguindo esta estratégia, podemos apresentar a seguinte forma do algoritmo de Bennett para matrizes esparsas:

$$\text{Para } j \in NZ \begin{cases} p = \beta\, c_j \\ d_j \leftarrow d_j + c_j\, p \\ p \leftarrow p / d_j \\ \left.\begin{array}{l} c_i \leftarrow c_i - l_{ij}\, c_j \\ l_{ij} \leftarrow l_{ij} + c_i\, p \end{array}\right\} \quad i > j \text{ e } i \in NZ \\ \beta \leftarrow \beta - p^2 d_j \end{cases}$$

Este processo é devido a Law e Fenves [55]. Alguns autores [23] apresentam variantes do algoritmo de Bennett que procuram evitar que os elementos diagonais da matriz D se tornem muito próximos de zero. As implementações de tais

variantes são bastante mais complexas. Por isso, optámos por controlar os valores dos sucessivos d_j durante a actualização da decomposição. Sempre que, numa iteração, um desses elementos diagonais se tornar muito pequeno, então obtém-se a decomposição LDL^T da correspondente matriz M_{FF} directamente em vez de usar essa actualização.

Se a matriz M é P não simétrica, o processo de obter as sucessivas factorizações de M_{FF} tem de ser alterado. Por outro lado, o algoritmo de grau mínimo só é aplicável a matrizes simétricas. Assim, para estabelecer a ordenação, deve-se considerar a estrutura da matriz $M + M^T$. A estrutura de dados a construir para conter as decomposições de cada iteração também deverá ter por base a factorização simbólica dessa matriz. Desta forma, qualquer decomposição $M_{FF} = LDU$ terá espaço para ser guardada.

Em cada iteração, quando é acrescentado ou retirado um elemento a F, a decomposição é actualizada de modo idêntico ao descrito para o caso simétrico. Para uma descrição sumária do processo de actualização, seja B uma matriz que se obtém de A por inserção de uma linha e coluna

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \qquad B = \begin{bmatrix} A_1 & b_1 & A_2 \\ a_1^T & \alpha & a_2^T \\ A_3 & b_2 & A_4 \end{bmatrix} \tag{2.16}$$

Suponha-se conhecida a decomposição de A

$$A = LDU = \begin{bmatrix} L_1 & \\ E & L_2 \end{bmatrix} \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} \begin{bmatrix} U_1 & F \\ & U_2 \end{bmatrix} \tag{2.17}$$

e que queremos obter a decomposição de B, ou seja,

$$B = \bar{L}\,\bar{D}\,\bar{U} = \begin{bmatrix} \bar{L}_1 & & \\ u_1^T & 1 & \\ \bar{E} & v_2 & \bar{L}_2 \end{bmatrix} \begin{bmatrix} \bar{D}_1 & & \\ & \beta & \\ & & \bar{D}_2 \end{bmatrix} \begin{bmatrix} \bar{U}_1 & v_1 & \bar{F} \\ & 1 & u_2^T \\ & & \bar{U}_2 \end{bmatrix} \quad (2.18)$$

Efectuando os produtos em (2.17) e (2.18), comparando com (2.16) e, atendendo a que a decomposição de A_1 é única, obtêm-se as seguintes igualdades:

$$\bar{L}_1 = L_1 \quad ; \quad \bar{U}_1 = U_1 \quad ; \quad \bar{D}_1 = D_1 \quad ; \quad \bar{E} = E \quad ; \quad \bar{F} = F$$

$$L_1 D_1 v_1 = b_1$$

$$U_1^T D_1 u_1 = a_1$$

$$\beta = \alpha - u_1^T D_1 v_1$$

$$v_2 = (b_2 - E D_1 v_1) / \beta$$

$$u_2 = (a_2 - F^T D_1 u_1) / \beta$$

$$\bar{L}_2 \bar{D}_2 \bar{U}_2 = L_2 D_2 U_2 - \beta\, v_2\, u_2^T$$

Verifica-se assim que tudo se passa de modo análogo ao caso simétrico com a agravante de haver o dobro das operações para executar.

Para a determinação de $\bar{L}_2$, $\bar{D}_2$ e $\bar{U}_2$ continua a ser possível a utilização do algoritmo de Bennett [4]. A sua versão original para matrizes não simétricas tem a seguinte forma

$$\text{Para } j = 1, \ldots, n \quad \begin{cases} p = \beta\, u_j \\ q = \beta\, v_j \\ d_j \leftarrow d_j + v_j\, p \\ p \leftarrow p / d_j \\ q \leftarrow q / d_j \\ \left.\begin{array}{l} u_i \leftarrow u_i - l_{ij}\, u_j \\ v_i \leftarrow v_i - u_{ji}\, v_j \\ l_{ij} \leftarrow l_{ij} + u_i\, q \\ u_{ji} \leftarrow u_{ji} + v_i\, p \end{array}\right\} \; i > j \\ \beta \leftarrow \beta - p\, q\, d_j \end{cases}$$

Tal como no caso simétrico, para tirar partido da esparsidade da matriz, devem-se resolver simbolicamente os sistemas $L\, y = u$ e $U^T\, t = v$. Se considerarmos os conjuntos de índices $NZL = \{ i: y_i \neq 0 \}$, $NZU = \{ i: t_i \neq 0 \}$ e se $NZ = NZL \cup NZU$, então obtém-se a seguinte forma do algoritmo de Bennett para matrizes esparsas

$$\text{Para } j \in NZ \quad \begin{cases} p = \beta\, u_j \\ q = \beta\, v_j \\ d_j \leftarrow d_j + v_j\, p \\ p \leftarrow p / d_j \\ q \leftarrow q / d_j \\ \left.\begin{array}{l} u_i \leftarrow u_i - l_{ij}\, u_j \\ l_{ij} \leftarrow l_{ij} + u_i\, q \end{array}\right\} \; i > j\,;\, i \in NZL \\ \left.\begin{array}{l} v_i \leftarrow v_i - u_{ji}\, v_j \\ u_{ji} \leftarrow u_{ji} + v_i\, p \end{array}\right\} \; i > j\,;\, i \in NZU \\ \beta \leftarrow \beta - p\, q\, d_j \end{cases}$$

Tal como anteriormente é também fácil de descrever o processo de actualização da decomposição LDU quando uma linha e uma coluna são retiradas.

É importante notar que o processo de actualização da decomposição LDU acabado de descrever pode ser sempre efectuado para matrizes P não simétricas. Contudo a estabilidade do processo é apenas garantida para matrizes diagonalmente dominantes [81] ou matrizes K [1]. No caso geral poderá haver problemas numéricos, principalmente se a matriz M é mal condicionada [27].

9 Implementação Para Matrizes Tridiagonais

Em algumas aplicações do LCP [14] a matriz M é simétrica PD e tridiagonal. Nestes casos há a possibilidade de desenvolver uma implementação bastante mais eficiente do que a apresentada na secção anterior [14, 37]. Antes de mais não há necessidade de qualquer ordenação, pois basta seguir a ordem natural das linhas e colunas de M para que, ao serem tomados pivots na diagonal, não se produzam quaisquer novos elementos.

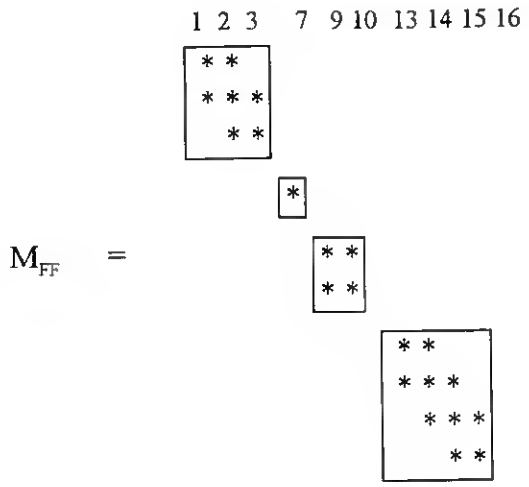
Para guardar os dados respeitantes ao LCP são necessários somente três vectores de dimensão n a seguir apresentados.

1 vector real para a diagonal principal de M

1 vector real para a diagonal secundária

1 vector real para o termo independente q

Cada submatriz M_{FF} é diagonal por blocos, sendo cada bloco uma matriz tridiagonal, podendo reduzir-se a um elemento simples ou uma matriz quadrada de dimensão 2. Para ilustrar essa afirmação, considere-se uma matriz M de ordem 20 e seja F = { 1 , 2 , 3 , 7 , 9 , 10, 13, 14 , 15 , 16 }. Então a matriz M_{FF} tem o aspecto seguinte:



Para factorizar esta matriz basta usar em cada subbloco diagonal as fórmulas da factorização LDL^T

$$i = 1, \ldots, n \quad \begin{cases} d_{ii} = a_{ii} - l^2_{i,i-1} \;\; d_{i-1,i-1} = a_{ii} - l_{i,i-1}\, a_{i,i-1} \\ l_{i+1,i} = a_{i+1,i} / d_{i\,i} \end{cases}$$

A actualização da factorização, quando um elemento é acrescentado ou retirado a F, só necessita de ser feita nos blocos onde houve alterações. Assim, no exemplo anterior, se se acrescentar o índice 4 ao conjunto F, só o primeiro bloco precisa de ser alterado. Se se acrescentasse o índice 8 isso iria fazer com que o segundo e o terceiro blocos se juntassem num só e seria exclusivamente nesse novo bloco que a factorização necessitava de ser refeita.

Ao acrescentar um elemento a F vários casos podem suceder:

i) criar um bloco com um só elemento

ii) acrescentar um elemento no fim de um bloco

iii) acrescentar um elemento no início de um bloco

iv) acrescentar um elemento entre dois blocos unindo-os

Na primeira hipótese não é preciso qualquer factorização. No segundo caso toda a factorização pode ser aproveitada bastando calcular os dois novos elementos criados (um em D e outro em L). No terceiro caso todo o bloco deve ser factorizado. Finalmente no último caso pode-se aproveitar a factorização do primeiro bloco e completá-la até ao fim do novo bloco.

Ao retirar um elemento a F pode acontecer uma das seguintes situações:

i) retirar um bloco com um só elemento

ii) retirar um elemento do fim de um bloco

iii) retirar um elemento do início de um bloco

iv) retirar um elemento do meio de um bloco partindo-o em dois

Nos dois primeiros casos não há qualquer actualização a realizar. No terceiro caso o bloco deve ser todo refactorizado. No último caso só o segundo bloco necessita de ser factorizado.

A actualização do vector dos termos independentes também é simplificada. O sistema $M_{FF}\bar{q}_F = - q_F$ resolve-se em cada bloco separadamente, pois, como a matriz M_{FF} é diagonal por blocos, o que se passa num bloco não afecta o bloco vizinho. Por essa mesma razão, quando um elemento é acrescentado ou retirado a F só se torna necessário resolver o sistema no(s) bloco(s) onde houve alterações.

Por seu turno a matriz M_{TF} tem poucos elementos não nulos, pois só há um elemento na linha i de T se i-1 ou i+1 for elemento de F. Por essa razão ao determinar $\bar{q}_T = q_T + M_{TF}\,\bar{q}_F$ nem todas as componentes do vector serão alteradas. De igual modo, quando o conjunto F é alterado em um só elemento, só é necessário actualizar as componentes cujo índice seja contíguo ao(s) bloco(s) alterado(s).A determinação do elemento diagonal actualizado $\bar{m}_{ss} = m_{ss} + M_{SF}\,\bar{M}_{FS}$ pode ser feita em alguns casos sem necessidade de resolver um sistema. Com efeito, a não ser que s-1 ou s+1 seja

elemento do conjunto F, o vector M_{FS} será nulo e, portanto, $\tilde{m}_{ss} = m_{ss}$. De qualquer modo, mesmo que tal não aconteça, só será necessário resolver o sistema no bloco que termina em s-1 ou no bloco que começa em s+1 ou em ambos, consoante s-1, s+1 ou ambos são elementos de F respectivamente.

Para cada bloco da matriz M_{FF} é necessário guardar dois números inteiros. Um deles representa o número de elementos do bloco e o outro aponta para o início do próximo bloco. Como não pode haver dois blocos a começar em elementos consecutivos (senão seria um só bloco), esta informação pode ser armazenada usando apenas um vector com n componentes que funciona como uma lista ligada.

O caso não simétrico tem também um tratamento muito simples. Deve-se usar a factorização $M_{FF} = LDU$ o que vai exigir mais um vector de dimensão n para guardar a informação respeitante a U. De resto tudo se passa como no caso simétrico tendo-se em atenção que as fórmulas para a decomposição passam a ser

$$i = 1, \ldots, n \begin{cases} d_{ii} = a_{ii} - l_{i,i-1} u_{i-1,i} d_{i-1,i-1} \\ l_{i+1,i} = a_{i+1,i} / d_{ii} \\ u_{i,i+1} = a_{i,i+1} / d_{ii} \end{cases}$$

Como se conclui do exposto nas duas últimas secções as implementações dos algoritmos pivotais principais para matrizes esparsas gerais e para matrizes tridiagonais são muito simples. Tal facto possibilita resolver LCPs de grandes dimensões. Na próxima secção iremos apresentar alguma experiência desse tipo.

10 EXPERIÊNCIA COMPUTACIONAL

Nesta secção é apresentado um estudo computacional comparativo da eficiência dos métodos pivotais principais descritos neste capítulo com LCPs monótonos e estritamente monótonos. Os resultados foram obtidos usando um computador CDC CYBER 180-830 da Universidade do Porto. A precisão da máquina é 10^{-16}.

Começamos por descrever sumariamente os problemas teste que usámos nas nossas experiências. Assim, a matriz do problema TP1 corresponde a uma aplicação do LCP no problema de análise elastoplastica de estruturas [67]. Os problemas TP2, TP3, TP4 e TP5 foram gerados usando uma técnica descrita em [83] que permite obter matrizes simétricas PD com número de condição conhecido. Para se obter uma matriz com condição c deve-se proceder como se descreve a seguir.

1) Gerar uma matriz diagonal D com condição $\alpha = \sqrt{c}$, fazendo $d_{11} = \sqrt{\alpha}$, $d_{22} = \frac{1}{\sqrt{\alpha}}$ e $d_{jj} = (\sqrt{\alpha})^{u_j}$, $j > 2$, com u_j uniformemente distribuído em [-1,1].

A condição de D é $\frac{d_{11}}{d_{22}} = \alpha$, pois d_{11} é o maior valor próprio e d_{22} é o menor.

2) Gerar uma matriz quadrada A tal que a_{ij} obedeça a uma distribuição normal com média 0 e variância 1.

3) Obter a factorização ortogonal de A = QR, usando transformações Householder e fazer B = Q.

4) Calcular $N = (BD)(BD)^T$. Como a matriz B é ortogonal o seu número de condição é igual a 1 e, por isso, a matriz N tem número de condição c.

É fácil de constatar que uma matriz N assim obtida é densa. É, no entanto, possível construir uma matriz A diagonal por blocos constituída por matrizes N_i. Se agora definirmos aleatoriamente uma matriz de permutação P e considerarmos a matriz $M = PAP^T$, então esta última matriz tem número de condição c e é esparsa. Deste modo é possível obter uma colecção de matrizes esparsas por variação das dimensões dos blocos N_i.

As matrizes dos problemas TP6, TP7 e TP8 foram geradas aleatoriamente seguindo a técnica descrita em [71] e contêm estruturas semelhantes às dos modelos económicos de selecção de investimentos. Assim são matrizes da forma $M = GG^T + D$, em que D é uma matriz diagonal com todos os elementos diagonais iguais a 2 e G é uma matriz rectangular de dimensão n×m com elementos no intervalo [-1., 1.]. Os problemas que gerámos foram obtidos com

TP6	n = 200	m = 5
TP7	n = 200	m = 20
TP8	n = 600	m = 2

Os problemas TP9, TP10 têm matrizes simétricas da classe K geradas completamente ao acaso.

As matrizes dos problemas TP11 e TP12 são pentadiagonais e foram construídas de acordo com [57] . As matrizes dos problemas TP13 e TP14 são tridiagonais simétricas PD geradas completamente ao acaso.

As matrizes dos problemas TP15 e TP16 são simétricas PSD singulares que foram geradas por um processo semelhante ao descrito em [83] que obriga alguns elementos diagonais de D a serem iguais a zero.

Para ter uma melhor ideia da capacidade dos algoritmos em obter a solução do LCP resolvemos gerar os vectores q dos vários problemas teste segundo um processo proposto em [78] que consiste em construir o vector q de modo a que a solução do LCP seja dada pelo utilizador. Esse processo consiste nos seguintes passos

1) Decide-se quantas componentes positivas tem o vector z. (#F).

2) Gera-se aleatoriamente #F componentes iguais a um inteiro uniformemente distribuído entre 2 e 5 inclusive. Iguala-se a zero as restantes componentes de z.

3) Gera-se o vector w tal que

$$i = 1, \ldots, n \begin{cases} w_i = 0 \quad \text{se} \quad z_i > 0 \\ w_i = \left\{ \begin{array}{l} 2 \text{ se } M_{.i}\, z \geq 0 \\ 1 \text{ se } M_{.i}\, z < 0 \end{array} \right\} \quad \text{se } z_i = 0 \end{cases}$$

4) Define-se $q = w - M z$

Na tabela 2.1 apresentam-se algumas características das matrizes dos LCPs. Para dar uma ideia do trabalho efectuado em cada iteração dos algoritmos pivotais principais acrescentámos a essa tabela a informação do tempo e do número de produtos e quocientes necessários para resolver um só sistema com a matriz M e o termo independente q. Tal não aconteceu nos problemas TP15 e TP16 pois as suas matrizes são singulares. Para se ter uma ideia da precisão das soluções obtidas, apresentamos também o resíduo da solução desse sistema:

$$\| q - M z \|_2 = \left[\sum_{i=1}^{n} \left(q_i - \sum_{j=1}^{n} m_{ij} z_j \right) \right]^{1/2}.$$

	n	nonz	stor	rsm	TO	TF	NO	T	RES
TP1	484	9920	17898	20.5	3.8	0.39	11.1	2.5	2-11
TP2	600	11442	17463	19.1	3.7	0.21	6.32	1.51	5-12
TP3	800	10920	16780	13.7	3.1	0.22	4.83	1.28	4-12
TP4	1000	16386	25073	16.4	5.1	0.32	8.14	2.03	4-12
TP5	1500	9824	15486	6.5	4.3	0.23	2.99	1.01	3-12
TP6	200	1922	3035	9.6	0.5	0.05	1.07	0.27	1-12
TP7	200	8072	16083	40.1	3.0	0.44	27.6	5.27	5-12
TP8	600	9878	15135	15.5	6.9	0.24	12.9	2.59	3-12
TP9	500	2156	8332	4.3	2.9	0.33	14.7	2.92	7-12
TP10	1000	2842	6217	2.8	1.7	0.16	2.59	0.71	6-12
TP11	500	2494	3991	5.	—	—	0.6	0.26	3-12
TP12	1000	4994	7991	5.	—	—	1.2	0.51	4-12
TP13	1000	2998	1998	3.	—	—	0.49	0.08	1-13
TP14	5000	14998	9998	3.	—	—	2.49	0.41	3-13
TP15	500	8224	12586	16.4	7.5	0.6	—	—	—
TP16	1000	11518	17577	19.2	12.4	0.9	—	—	—

Tabela 2.1

Os parâmetros que aparecem na tabela 2.1 têm o seguinte significado:

n = dimensão do LCP = ordem de M.

nonz = número de elementos não nulos da matriz M.

stor = espaço de armazenagem requerido para os valores numéricos de M, D e L.

rsm = esparsidade relativa de $M = \frac{nonz}{n}$

TO = tempo, em segundos de CPU, gasto pelo algoritmo de grau mínimo a ordenar a matriz.

TF = tempo, em segundos de CPU, gasto na fase simbólica de construção da estrutura de L.

T = tempo, em segundos de CPU gasto a resolver um sistema com matriz M.

NO = número de multiplicações e divisões, multiplicado por 10^{-4}, na resolução do sistema.

RES = resíduo da solução do sistema. Sendo α e β números reais, usamos $\alpha - \beta$ para representar $\alpha \times 10^{-\beta}$.

Na primeira experiência resolvemos comparar os métodos de Murty, Keller e Graves para LCPs estritamente monótonos. Como, neste caso, o método de Criss-Cross se reduz ao método de Murty, não foi feita qualquer experiência com este método. Para o método de Murty foi também considerada a hipótese de se escolher, em cada iteração, a maior inadmissibilidade. Essa variante é referida por Murty 2. Os resultados desse estudo são apresentados na tabela 2.2. Essa tabela contém as seguintes notações adicionais

NI = número de iterações.

T = tempo em segundos de CPU gasto a executar o algoritmo.

Pro-blema	N	VA	MURTY				MURTY 2				KELLER				GRAVES			
			NI	NO	T	RES	NI	NO	T	RES	NI	NO	T	RES	NI	NO	T	RES
TP1	484	339	553	116.4	27.6	8-12	191	39.9	10.	8-12	195	24.7	7.5	1-11	175	67.4	13.2	7-12
		97	637	325.8	74.7	1-11	411	230.4	54.7	1-11	411	109.3	37.7	2-11	401	417.9	81.9	2-11
TP2	600	420	554	85.4	22.5	2-12	210	41.3	11.	2-12	210	29.3	9.1	2-12	242	102.3	20.4	2-12
		120	836	343.	88.9	4-12	520	249.9	64.2	4-12	538	128.1	48.7	4-12	544	531.1	109.3	4-12
TP3	800	560	602	86.	25.8	1-12	278	50.2	15.3	1-12	278	36.5	12.9	2-12	294	110.4	24.2	1-12
		160	1004	402.8	112.7	3-12	674	300.9	84.4	4-12	684	156.2	65.1	3-12	724	675.5	147.8	4-12
TP4	1000	700	790	178.4	49.6	2-12	334	88.3	25.1	2-12	334	66.	21.5	2-12	354	198.7	41.1	2-12
		200	1476	837.9	226.3	4-12	838	559.1	149.9	4-12	858	286.3	114.6	4-12	886	1213.	256.6	4-12
TP5	1500	1050	920	117.4	190.2	1-12	500	78.2	33.7	1-12	502	61.4	31.	1-12	526	185.4	53.3	1-12
		300	1600	532.9	209.8	3-12	1250	485.6	174.7	3-12	1260	278.9	147.1	3-12	1306	1086.	289.9	3-12
TP6	200	140	108	2.81	1.09	5-13	60	1.88	0.65	7-12	60	1.61	0.58	5-13	60	4.19	0.99	6-13
		40	174	7.3	2.7	1-12	160	15.9	4.5	1-12	160	10.2	3.3	1-12	160	32.1	7.1	1-12
TP7	200	140	312	113.3	24.9	2-12	60	10.2	2.3	2-12	60	9.5	2.2	2-12	60	19.9	3.6	2-12
		40	242	132.5	28.2	4-12	160	128.2	27.9	4-12	160	112.5	25.4	5-12	160	210.4	40.2	5-12
TP8	600	420	278	29.5	9.7	1-12	180	30.4	7.8	2-12	180	25.7	6.8	3-12	182	67.5	12.8	2-12
		120	494	66.2	21.9	5-12	480	216.8	51.8	3-12	480	125.7	34.9	1-11	480	430.1	82.6	3-12
TP9	500	350	150	4.1	2.33	3-12	150	6.7	3.25	2-12	150	5.	3.	1-11	150	13.7	4.6	2-12
		100	400	41.6	14.7	7-12	400	133.5	35.1	7-12	400	92.4	28.9	4-11	400	190.7	45.5	8-12
TP10	1000	700	300	10.1	7.9	2-12	300	15.8	10.5	2-12	300	13.3	10.	2-12	300	35.6	14.9	2-12
		200	800	72.9	40.8	5-12	800	143.8	61.2	7-12	800	74.6	49.9	3-10	800	277.	90.8	6-12
TP11	500	350	386	14.9	6.9	1-12	150	5.7	3.09	1-12	150	5.1	3.08	1-11	150	13.	4.5	1-12
		100	714	71.2	28.1	3-12	400	41.6	20.	3-12	400	26.9	18.2	1-11	404	88.1	30.1	3-12
TP12	1000	700	802	61.6	27.9	1-12	300	22.6	12.1	1-12	300	20.	12.1	1-12	300	51.9	17.6	1-12
		200	1488	275.3	107.9	5-12	802	163.5	77.5	4-12	802	102.9	69.8	5-12	816	355.3	120.8	4-12

Tabela 2.2

RES = resíduo da solução óptima dado por

$$\| w - q - M z \|_2 = \left[\sum_{i=1}^{n} \left(w_i - q_i - \sum_{j=1}^{n} m_{ij} z_j \right)^2 \right]^{1/2}.$$

VA = número de variáveis z_i nulas na solução óptima.

A análise dos resultados leva-nos a concluir que o método de Keller é, em geral, o mais eficiente, tanto em número de iterações como, principalmente, em tempo de execução. O método de Murty tem um bom comportamento quando se utiliza o critério de escolha da maior inadmissibilidade, que contudo não oferece garantias teóricas de convergência. A versão teoricamente convergente do método de Murty não é, em geral, competitiva com o método de Keller. Finalmente, apesar do método de Graves ser competitivo com o método de Keller em termos do número de iterações, os seus tempos de execução são nitidamente superiores. Como referimos anteriormente, tal é devido ao maior esforço computacional de cada uma das suas iterações.

Se a matriz do LCP é K, então todas as variáveis z_i que se tornam básicas mantêm o seu estatuto, pelo que ambas as versões do método de Murty executam o mesmo número de iterações, que é igual ao número de elementos do conjunto F da solução do LCP. Contudo, as duas versões têm tempos de execução bastante diferentes. Tal é devido à maneira como se actualiza a decomposição LDL^T das matrizes M_{FF}. Com efeito, na versão original do método de Murty a actualização da decomposição requer muito menos trabalho do que no segundo, por as linhas e colunas se inserirem sempre no fim. É importante notar que, nestes casos, o método de Murty é mais eficiente que o método de Keller.

Numa última experiência com matrizes esparsas gerais fizemos uma pequena comparação entre o método de Keller e o método de Graves para o caso de LCPs monótonos e não estritamente monótonos. Os resultados dessas comparações

aparecem na tabela 2.3 e mostram que, tal como anteriormente, o método de Graves é menos eficiente do que o algoritmo de Keller. Como o método Criss-Cross é a extensão do método de Murty para matrizes PSD, então é de esperar uma menor eficiência desse processo em relação ao algoritmo de Keller. Daí não apresentarmos resultados computacionais comparativos nesta secção. No capítulo 3 iremos estudar o funcionamento desse algoritmo em outro contexto.

Problema		KELLER	GRAVES
TP15	NI	293	297
	NO*10^{-2}	1.0	1.8
	T	80.	103.
TP16	NI	562	572
	NO*10^{-2}	3.	6.2
	T	318.	451.

Tabela 2.3

A experiência apresentada leva-nos a recomendar a utilização do método de Keller para a resolução de LCPs monótonos com matrizes simétricas esparsas. Contudo o método de Murty (ou Criss-Cross) é, nalguns casos, competitivo com esse processo e tem a vantagem de poder iniciar-se com qualquer conjunto F. Essa característica será explorada mais adiante no desenvolvimento do algoritmo pivotal principal a apresentar no próximo capítulo.

A tabela 2.4 ilustra os resultados das experiências efectuadas na resolução de LCPs estritamente monótonos com matriz simétrica PD tridiagonal. Nessas experiências resolvemos apenas usar os métodos de Murty e Keller, pois o algoritmo de Graves foi considerado menos eficiente nas experiências anteriores.

	Problema	TP15		TP16	
	N	1000		5000	
	VA	700	200	3500	1000
MURTY	NI	414	878	2174	4456
	NO	0.2	1.58	1.06	7.84
	T	0.41	0.89	8.2	13.2
	RES	4-13	9-13	8-13	2-12
MURTY2	NI	299	799	1501	4007
	NO	0.12	0.93	0.59	4.79
	T	2.4	6.5	57.1	156.6
	RES	4-13	9-13	8-13	2-12
KELLER	NI	299	802	1501	4009
	NO	0.1	0.71	0.41	3.92
	T	2.1	5.6	49.	120.
	RES	4-13	9-13	8-13	2-12

Tabela 2.4

É interessante verificar que o método de Murty (versão original) é vantajoso em relação ao método de Keller e mesmo à versão 2 do método de Murty. Com efeito, apesar do número de iterações ser mais elevado, assiste-se a um decréscimo no tempo de execução. Esta aparente contradição deve-se por um lado ao facto de as colunas serem introduzidas em M_{FF} pela ordem natural, o que vai fazer com que as actualizações sejam menos demoradas. A implementação especifica para matrizes tridiagonais implica algum trabalho com ponteiros sempre que haja inserção de elementos no meio da lista, mas esse trabalho é mínimo quando os elementos são introduzidos no fim. Por outro lado, quer o método Murty 2 quer o método de Keller, devem, em cada iteração, determinar o menor elemento de um vector com n elementos, o que para valores de n muito elevados leva bastante tempo. Conclui-se assim que o algoritmo de Murty é altamente recomendável, neste caso especial. Esta experiência também mostra que a estrutura da matriz do LCP pode ser determinante

na escolha do algoritmo a utilizar na sua resolução. Como foi referido anteriormente, o método de Lemke pode ser utilizado para a resolução de LCPs monótonos, estritamente ou não. Contudo a sua implementação não tira partido da simetria da matriz M do LCP, o que o torna menos interessante para a resolução de LCPs monótonos com matrizes simétricas. Os resultados apresentados na tabela 2.5 confirmam exactamente esta afirmação. Contudo o método de Lemke é certamente competitivo para LCPs monótonos com matrizes PSD não simétricas e pode ser utilizado para alguns LCPs não monótonos, para os quais os algoritmos pivotais principais não têm possibilidade de ser utilizados.

	Problema	TP1	TP2	TP8	TP11
	N	484	600	600	500
	VA	114	301	300	240
LEMKE	NI	289	406	303	369
	T	77.8	87.3	57.5	46.5
MURTY2	NI	308	355	302	360
	NO	12.1	11.3	7.5	3.3
	T	28.9	28.7	18.1	14.6
	RES	1-11	5-12	3-12	9-12
KELLER	NI	314	367	302	360
	NO	12.3	11.7	7.1	3.3
	T	31.8	32.7	18.6	17.1
	RES	1-11	6-12	2-12	9-12

Tabela 2.5

CAPÍTULO III

MÉTODOS PIVOTAIS POR BLOCOS PARA O PROBLEMA LINEAR COMPLEMENTAR

1 INTRODUÇÃO

Os métodos directos discutidos no capítulo anterior permitem apenas alterações de um elemento do conjunto F das variáveis básicas z_i em cada iteração. Assim o número de iterações pode vir a ser bastante elevado se o conjunto F que corresponde à solução do LCP for muito diferente do inicial. Algumas tentativas têm sido feitas no sentido de remover várias inadmissibilidades por iteração, tendo aparecido ultimamente alguns métodos deste tipo que foram elaborados de modo a explorar as propriedades de certas classes de matrizes ou a geometria do problema. A característica principal de tais métodos pivotais por blocos é a de alterarem os conjuntos de índices F e T, referidos no capítulo 2, em vários elementos de cada vez. Deste modo, procura-se diminuir o número de iterações, aumentando, algumas vezes, o trabalho por iteração. No caso geral torna-se bastante difícil estabelecer a convergência de tais algoritmos e algumas precauções devem ser tomadas para evitar possíveis ciclos, como se verá mais adiante. Neste capítulo iremos discutir alguns algoritmos pivotais por blocos e mostraremos a eficiência desses processos para problemas de grandes dimensões.

Quando a matriz do LCP pertence à classe K, algumas das suas propriedades podem ser aproveitadas para estabelecer a convergência de um algoritmo pivotal por blocos. Essas propriedades podem ser apresentadas no seguinte teorema.

Teorema 3.1

(i) Se $M \in K$, então o complemento de Schur de uma sua submatriz principal em M pertence a K.

(ii) A inversa de M é não negativa.

Seja M uma matriz K de ordem n e $F \subseteq \{ 1, \ldots , n \}$. Então, por permutação principal de linhas e colunas, podemos escrever

$$M = \begin{bmatrix} M_{FF} & M_{FT} \\ M_{TF} & M_{TT} \end{bmatrix}$$

Efectuando uma operação pivotal principal com pivot M_{FF} , obtém-se:

$$\bar{M} = \begin{bmatrix} \bar{M}_{FF} & \bar{M}_{FT} \\ \bar{M}_{TF} & \bar{M}_{TT} \end{bmatrix}$$

com $\bar{M}_{FF} = M_{FF}^{-1}$, $\bar{M}_{TT} = M_{TT} - M_{TF} M_{FF}^{-1} M_{FT}$, $\bar{M}_{TF} = M_{TF} M_{FF}^{-1}$ e $\bar{M}_{FT} = - M_{FF}^{-1} M_{FT}$. Do teorema 3.1 conclui-se que $\bar{M}_{FF}$ é uma matriz de elementos não negativos, $\bar{M}_{TT} \in K$, $\bar{M}_{TF}$ é uma matriz de elementos não positivos e todos os elementos de $\bar{M}_{FT}$ são não negativos.

Seja agora H o conjunto das inadmissibilidades na primeira iteração, isto é $H = \{ i : q_i < 0 \}$. Fazendo $F = H$ ter-se-à $\bar{q}_F = - M_{FF}^{-1} q_F$. Como $q_F < 0$ e $M_{FF}^{-1} \geq 0$ será $\bar{q}_F > 0$. Portanto, todas as variáveis z básicas serão positivas. Na iteração seguinte

o conjunto das inadmissibilidades só terá elementos de T. Analise-se o que acontece aos valores das variáveis z básicas quando se efectua uma operação bloco pivotal com pivot $\bar{M}_{HH}$ e $H \subset T$, ou seja, quando se faz $F = F \cup H$.

As variáveis z_H passarão a ser básicas sendo o seu valor dado por $\bar{M}_{HH}^{-1}\ \bar{q}_H$. Como $\bar{q}_H < 0$ e $\bar{M}_{HH}$ é uma matriz K, então $z_H > 0$. Por outro lado as variáveis z que já eram básicas ficam com o valor $\bar{q}_F - \bar{M}_{FH}\ \bar{M}_{HH}^{-1}\ \bar{q}_H > \bar{q}_F$, ou seja continuarão a ser positivas. Então, ao fazer $F = F \cup H$, nenhuma variável z_i básica se tornará negativa. Deste modo o conjunto F aumenta em cada iteração obtendo-se assim um processo convergente da seguinte forma:

ALGORITMO DE CHANDRASEKARAN [5]

PASSO INICIAL: Faça $F = \{ i : q_i < 0 \}$ e $T = \{ 1, \ldots, n \} - F$

PASSO GERAL: Seja $H = \{ i \in T : \bar{q}_i < 0 \}$.

Se $H = \varnothing$ a solução é ($z_F = \bar{q}_F$, $w_T = \bar{q}_T$, $z_T = w_F = 0$).

Caso contrário faça $F = F \cup H$, $T = T - H$

e repita o Passo Geral.

Como afirmámos anteriormente este algoritmo é convergente, pois o conjunto F aumenta a cada iteração e tem no máximo n elementos. Isso também mostra que o algoritmo terá no máximo n iterações e que, por isso, é um processo polinomial.

Se o algoritmo de Chandrasekaran for aplicado a um LCP com uma matriz que não seja Z, não há garantia que $F \cap H = \varnothing$ e o algoritmo não funciona. Uma extensão simples considera as seguintes alterações dos conjuntos F e T:

$$F = F - (F \cap H) \cup (T \cap H) \quad e \quad T = T - (T \cap H) \cup (F \cap H) \tag{3.1}$$

Este tipo de processo foi primeiramente referido em Kostreva [52], e não oferece qualquer garantia de convergência para uma matriz $M \in P$. A experiência computacional com este processo mostrou que, em geral, o algoritmo requer um número muito pequeno de iterações, principalmente para LCPs estritamente monótonos com matrizes esparsas de dimensão elevada. Além disso raramente ocorrem ciclos durante o processo.

A experiência também mostrou que os casos de entrada em ciclo ocorrem normalmente quando o número de inadmissibilidades é muito pequeno. Por outro lado, o método de Murty é convergente qualquer que seja o conjunto F inicial e terá poucas iterações se a solução inicial tiver poucas inadmissibilidades. Estas constatações levaram-nos [36] a introduzir uma constante anti-ciclo, atc, de tal modo que se fazem pivotagens da forma (3.1) enquanto o número de inadmissibilidades seja superior a atc e pivotagens simples quando o número de inadmissibilidades seja inferior a essa constante. Deste modo podemos introduzir o seguinte processo:

ALGORITMO BLOCO 1

PASSO 0: Seja atc um número inteiro positivo

e considere-se $F = \varnothing$ e $T = \{ 1, \ldots, n \}$.

PASSO 1: Calcule $\bar{q}$ e seja $H = \{ i : \bar{q}_i < 0 \}$.

Se $\# H \leq$ atc vá para Passo 2. Caso contrário faça:

$$F = F - (F \cap H) \cup (T \cap H) \quad \text{e} \quad T = T - (T \cap H) \cup (F \cap H)$$

e repita o Passo 1.

PASSO2: Se $\# H = 0$ a solução é : $z_F = \bar{q}_F$; $z_T = 0$; $w_F = 0$; $w_T = \bar{q}_T$.

Caso contrário seja $s = \min \{ i \in H \}$.

$$\begin{cases} \text{se } s \in F \text{ faça } F = F - \{ s \} \text{ e } T = T \cup \{ s \} \\ \text{se } s \in T \text{ faça } F = F \cup \{ s \} \text{ e } T = T - \{ s \} \end{cases}$$

Vá para o Passo 1.

O primeiro assunto a tratar neste algoritmo reside na escolha de atc. Como será discutido na última secção, a experiência computacional indicou que atc deve ser escolhido muito pequeno. Normalmente atc = 3 é um bom valor mas a eficiência do processo é pouco afectada por um aumento pequeno dessa constante. Além disso o método conseguiu sempre obter uma solução do LCP em todos os testes. No entanto não há qualquer garantia que não possam ocorrer ciclos neste processo, que, portanto, deve ser considerado como heurístico. Em relação ao funcionamento propriamente dito do algoritmo a experiência indicou os seguintes factos:

(i) Sempre que o método funciona bem o número de inadmissibilidades é reduzido em cada grupo de $p \geq 1$ iterações, com p muito pequeno.

(ii) Sempre que ocorrem ciclos o número de inadmissibilidades mantém-se ou aumenta.

(iii) O número de pivotagens em bloco é normalmente muito pequeno.

Estas conclusões sugeriram-nos uma pequena alteração deste algoritmo de modo a evitar a ocorrência de ciclos. Para isso introduzimos uma constante que

designámos por nmaxpv e que representa o número máximo de iterações que se permite ao algoritmo bloco. Esta quantidade é redefinida durante a execução, tendo essa redefinição a ver com o decréscimo do número de inadmissibilidades. Assim, usamos pivotagens em bloco enquanto o número de inadmissibilidades for decrescente. Se o número de inadmissibilidades aumenta, permitem-se ainda (p - 1) pivotagens em bloco durante as quais o número de inadmissibilidades deve decrescer abaixo do último valor ninf que se obteve antes desse número começar a não decrescer. Se tal não acontecer então o método de Murty deve ser usado até que o número de inadmissibilidades seja menor do que ninf. Deste modo construímos o seguinte processo [44]

ALGORITMO BLOCO 2

PASSO 0: Seja p um número inteiro positivo.

Faça $k = 1$, $ninf = nmaxpv = n$, $F = \varnothing$ e $T = \{ 1 , \ldots , n \}$.

PASSO 1: Calcule $\bar{q}$ e seja $H = \{ i : \bar{q}_i < 0 \}$.

(i) Se $H = \varnothing$, $z = (\bar{q}_F , 0)$, $w = (0 , \bar{q}_T)$ é a solução única do LCP e páre.

(ii) Se $\#H < ninf$, faça $ninf = \#H$, $nmaxpv = k + p$ e vá para Passo 2.

(iii) Se $\#H \geq ninf$ então $\begin{cases} k \leq nmaxpv \Rightarrow \text{vá para Passo 2.} \\ k > nmaxpv \Rightarrow \text{vá para Passo 3.} \end{cases}$

PASSO 2: Faça $F = F - (F \cap H) \cup (T \cap H)$ e $T = \{ 1 , \ldots , n \} - F$.

Faça $k = k + 1$ e volte a Passo 1.

PASSO 3: Seja $s = \min \{ i \in H \}$

$$\begin{cases} \text{se } s \in F \text{ faça } F = F - \{ s \} \text{ e } T = T \cup \{ s \} \\ \text{se } s \in T \text{ faça } F = F \cup \{ s \} \text{ e } T = T - \{ s \} \end{cases}$$

Faça $k = k + 1$ e volte a Passo 1.

A escolha do parâmetro p tem um efeito muito importante na eficiência do algoritmo. Por um lado deve ser escolhido suficientemente pequeno para que não ocorram muitas iterações do método bloco sem melhoria do número de inadmissibilidades. Por outro lado não deve ser demasiado pequeno para que o método de Murty não seja chamado cedo demais e vezes demais. O valor $p = 3$ é geralmente uma boa escolha [44].

Este algoritmo termina num número finito de iterações com a solução única do LCP. Contudo não foi possível ainda estabelecer a complexidade do algoritmo. O seu desempenho tem muito a ver com a natureza do problema em estudo. Como veremos na última secção, o número de iterações do processo é normalmente bastante pequeno mesmo para problemas de grandes dimensões. Além disso, como será discutido na última secção, os famosos problemas de Murty [65] e Fathi [18], que ilustram o comportamento exponencial dos métodos pivotais simples, são resolvidos num número de iterações sensivelmente igual à dimensão do LCP.

Se a matriz do LCP é simétrica PSD este algoritmo deve ser ligeiramente modificado para prever a hipótese de se encontrarem submatrizes singulares. Isso é conseguido se substituirmos o método de Murty pelo método Criss-Cross na parte respeitante às pivotagens simples. Durante a fase bloco deve-se ter em atenção que os elementos a acrescentar ao conjunto F devem ser tais que a matriz M_{FF} seja sempre não singular. Para isso no passo 2 do algoritmo Bloco-2 a transformação $F = F - (F \cap H) \cup (T \cap H)$ deve ser substituída por:

Faça $F = F - (F \cap H)$

para cada $i \in (T \cap H)$ faça:

se $\begin{bmatrix} M_{FF} & M_{Fi} \\ M_{iF} & m_{ii} \end{bmatrix}$ é não singular faça $F = F \cup \{ i \}$ (3.2)

Caso contrário não altere F.

Faça $T = \{ 1 , \ldots , n \} - F$.

Imediatamente se constata que a matriz M_{FF} é sempre não singular e que o método Bloco 2 pode ser utilizado sem alterações. Tal como no método Criss-Cross, o algoritmo Bloco pode ter, neste caso, dois tipos de terminações, nomeadamente, obter a solução ou mostrar que o LCP é não admissível. Este tipo de terminação só pode ocorrer no passo respeitante às operações pivotais simples.

4 Implementação Para Matrizes Esparsas Gerais

A implementação de este tipo de algoritmos pode ser feita de modo muito semelhante ao descrito no capítulo anterior. Tal como para os algoritmos pivotais simples, a fase mais importante em cada iteração consiste na resolução de um sistema com uma submatriz principal M_{FF} de M. Tal como foi descrito no capítulo anterior há que começar por efectuar uma fase simbólica que forneça a ordenação das linhas e colunas de M que irá ser respeitada ao longo de todo o processo. Em seguida constrói-se a estrutura capaz de conter a informação respeitante às sucessivas factorizações LDL^T das matrizes M_{FF}. Sempre que um só elemento é alterado em F o processo de actualização já descrito no capítulo anterior deve ser usado. Tal ocorre sempre que o método de Murty é chamado. Durante as iterações do método bloco,

vários elementos são retirados e inseridos simultaneamente no conjunto F e a actualização da decomposição, tal como foi descrita, deixa de ser eficiente. Neste último caso torna-se mais vantajoso calcular as matrizes L e D directamente a partir de cada nova matriz M_{FF}.

A implementação apresentada não tira partido da decomposição da iteração anterior na obtenção da decomposição de uma dada iteração. Com efeito, a opção de orientar o processo de decomposição por colunas torna isso bastante difícil de fazer. Seguidamente iremos descrever uma outra implementação dos métodos pivotais por blocos orientada por linhas e que tira partido desse facto.

Seja M_{FF} a matriz associada a uma determinada iteração da qual se conhece a factorização LDL^T. Na iteração seguinte elementos vão ser retirados e acrescentados a F. Sejam r e s os índices do primeiro e do último elementos de F que são alterados de acordo com a ordem obtida na Fase de Análise preliminar. Considerem-se agora os seguintes conjuntos de índices

$$A = \{ i \in F : i < r \}, \quad B = \{ i \in F : r \leq i \leq s \}, \quad C = \{ i \in F : i > s \}$$

Então a matriz M_{FF} pode ser escrita na forma:

$$M_{FF} = \begin{bmatrix} M_{AA} & M_{AB} & M_{AC} \\ M_{BA} & M_{BB} & M_{BC} \\ M_{CA} & M_{CB} & M_{CC} \end{bmatrix}.$$

Durante a iteração seguinte o conjunto B vai ser substituído por outro conjunto de índices E e a nova matriz M_{FF} vem dada por:

$$M_{\overline{FF}} = \begin{bmatrix} M_{AA} & M_{AE} & M_{AC} \\ M_{EA} & M_{EE} & M_{EC} \\ M_{CA} & M_{CE} & M_{CC} \end{bmatrix}.$$

Considere-se agora a factorização de ambas as matrizes seguindo as mesmas partições dos índices mencionadas atrás.

$$L = \begin{bmatrix} L_{AA} & & \\ L_{BA} & L_{BB} & \\ L_{CA} & L_{CB} & L_{CC} \end{bmatrix}, \qquad D = \begin{bmatrix} D_{AA} & & \\ & D_{BB} & \\ & & D_{CC} \end{bmatrix} \tag{3.3}$$

e

$$\bar{L} = \begin{bmatrix} \bar{L}_{AA} & & \\ \bar{L}_{EA} & \bar{L}_{EE} & \\ \bar{L}_{CA} & \bar{L}_{CE} & \bar{L}_{CC} \end{bmatrix}, \qquad \bar{D} = \begin{bmatrix} \bar{D}_{AA} & & \\ & \bar{D}_{EE} & \\ & & \bar{D}_{CC} \end{bmatrix}. \tag{3.4}$$

Efectuando os produtos LDL^T e $\bar{L}\bar{D}\bar{L}^T$, comparando com M_{FF} e $M_{\bar{F}\bar{F}}$ e atendendo a que a factorização é única obtêm-se as seguintes relações:

$$\bar{L}_{AA} = L_{AA} \; ; \quad \bar{D}_{AA} = D_{AA} \; ; \quad \bar{L}_{CA} = L_{CA} \, .$$

Além disso B e E têm elementos em comum e, para essas linhas, os elementos que pertençam a colunas de A não são alterados.

Tal como referimos anteriormente, para que estas relações possam ser completamente exploradas é necessário utilizar uma estrutura por linhas para L em vez da estrutura por colunas que foi usada na implementação descrita anteriormente. Para isso usa-se o esquema por linhas descrito em [24] que consiste em ir calculando a factorização linha a linha do modo que a seguir se descreve.

Suponha-se conhecida a factorização de uma matriz $M = LDL^T$ e seja $\bar{M}$ a matriz que se obtém de M por acréscimo de uma linha e uma coluna na última posição de M. Então

$$\bar{M} = \begin{bmatrix} M & c \\ c^T & \alpha \end{bmatrix} = \begin{bmatrix} L & \\ d^T & 1 \end{bmatrix} \begin{bmatrix} D & \\ & \beta \end{bmatrix} \begin{bmatrix} L^T & d \\ & 1 \end{bmatrix}$$

Tendo em conta que $M = LDL^T$ e a unicidade da decomposição, facilmente se obtêm as seguintes relações:

$$\bar{L} = L \; ; \quad \bar{D} = D \; ;$$

$$LD\, d = c \; ; \quad \beta = \alpha - d^T D\, d \tag{3.5}$$

Assim a nova linha de L obtém-se resolvendo um sistema triangular cuja matriz consiste nas linhas de L já determinadas e o novo elemento diagonal de D é determinado efectuando um produto interno.

Quando se faz a actualização da factorização (3.3) para obter (3.4) começa-se por conservar as linhas cujos índices pertencem a A. Seguidamente suprimem-se as linhas de índice em B-E. Para as restantes linhas vão-se utilizar as relações (3.5), tendo em conta que os elementos pertencentes a colunas de A e a linhas de C ou de $B \cap E$ não vão ser alterados.

Esta implementação é também vantajosa no tratamento de LCP com matrizes simétricas PSD. Com efeito, o critério (3.2) pode ser implementado averiguando se o valor de β determinado por (3.5) é inferior a uma dada tolerância. Se tal acontecer esse índice não é acrescentado a F.

Tal como na implementação por colunas anteriormente descrita, também neste caso é necessário efectuar no início uma fase de análise simbólica destinada a construir a estrutura capaz de conter todas as matrizes L das sucessivas factorizações. A ordenação continuará a ser feita pelo algoritmo do grau mínimo já descrito no capítulo 2. Quanto à factorização simbólica de M tem que se adoptar um novo esquema que forneça a matriz L por linhas. No estudo das construções para esse efeito, começámos por tentar adaptar o processo de Law e Fenves [55] para uma orientação por linhas. A experiência demonstrou que este processo não era

competitivo com o esquema por colunas descrito anteriormente. Por isso, resolvemos usar a mesma fase simbólica inicial para obter a estrutura por colunas e depois calcular a transposta dessa matriz usando a técnica descrita por Gustavson [30]. Apesar do trabalho extra da transposição, este processo é mais rápido do que a adaptação do esquema de Law e Fenves [55].

A actualização da decomposição deve ser feita sempre que são efectuadas operações pivotais simples. Como o algoritmo de Bennett é orientado por colunas, então a matriz L deve ser primeiramente actualizada por colunas e depois transposta de modo a armazená-la por linhas.

Para terminar esta secção iremos tratar brevemente o caso das matrizes tridiagonais. O processo de factorização continua com o mesmo aspecto anteriormente descrito no capítulo 2. De notar que só os blocos de F com índices pertencentes ao conjunto B devem ser considerados. Além disso deve-se ter em conta que os elementos de L e de D só serão alterados a partir do primeiro índice alterado. Quanto à actualização dos termos independentes, só será necessário resolver o sistema $M_{FF}\ \bar{q}_F = -\ q_F$ para os blocos onde houve alterações e só será necessário recalcular as componentes de $\bar{q}_T$ que fiquem adjacentes a tais blocos.

5 Experiência Computacional

Nesta secção é apresentada alguma experiência computacional com os métodos pivotais por blocos que foram discutidos neste capítulo. Para atestar da eficiência desses processos apresentaremos também algumas comparações com os algoritmos pivotais simples que foram discutidos no capítulo anterior.

Os resultados da tabela 3.7 foram obtidos usando uma Wokstation SUN 3-60 do Departamento de Matemática da Universidade de Coimbra e os restantes foram obtidos usando um computador CDC CYBER 180-830 da Universidade do Porto.

Os problemas teste que usámos na nossa experiência são alguns dos já descritos no capítulo anterior e outros obtidos de modo análogo. Para possibilitar a comparação entre os resultados que agora apresentamos e os referentes aos métodos descritos no capítulo anterior, mantivemos a numeração para os LCPs em que a matriz é a mesma que foi utilizada atrás. Apresentamos seguidamente uma descrição sumária do modo como foram obtidos os problemas resolvidos neste capítulo.

(i) TP1: problema de análise elastoplástica de estruturas [67].

(ii) TP2, TP3, TP4, TP5: LCPs gerados aleatoriamente usando a técnica de Stewart [83].

(iii) TP6, TP7, TP8: LCPs que ocorrem em modelos económicos de acordo com [71].

(iv) TP17, TP18, TP19: LCPs com matrizes de classe K geradas aleatoriamente.

(v) TP20, TP21, TP22: LCPs com matrizes geradas de acordo com a técnica de Stewart [83] mas a que se somou uma correcção diagonal de modo a que sejam diagonal dominantes.

(vi) TP11, TP12, TP23: LCPs com matrizes pentadiagonais referenciadas em Lin e Pang [57], de dimensões 500, 1000 e 1500 respectivamente.

(vii) TP24, TP25: LCPs com matrizes PSD geradas aleatoriamente seguindo a técnica de Stewart com a modificação descrita no capítulo anterior.

(ix) TP26, TP27: LCPs com matrizes pertencentes à colecção de Harwell [16], de dimensões 1086 e 1919.

Em todos os casos em que há indicação do número de variáveis básicas na solução, o vector q foi obtido de acordo com o descrito no capítulo anterior. Nos outros casos ele foi gerado aleatoriamente de modo a que os seus elementos obedeçam a uma distribuição uniforme.

Na tabela 3.1 apresentamos as principais características de alguns dos problemas testes utilizados nesta secção. Os parâmetros descritos nessa tabela são os mesmos que já foram utilizados no capítulo anterior.

Problema	n	nonz	stor	rsm	TO
TP17	500	1658	4131	3.3	1.2
TP18	800	2314	4971	2.9	1.32
TP19	1000	3342	10263	3.3	3.28
TP20	500	3096	4914	6.2	0.88
TP21	800	4904	7198	6.1	1.66
TP22	1000	4098	6822	4.1	1.68
TP24	1000	6314	9971	6.3	2.41
TP25	1200	6000	9600	5.0	3.0

Tabela 3.1

Começámos por efectuar uma comparação entre as duas implementações descritas para os algoritmos bloco, que são orientadas por colunas e por linhas. Apenas matrizes PD foram consideradas nesta experiência, pois para as matrizes PSD singulares a implementação por colunas é impraticável. Com efeito torna-se extremamente demorada a aplicação do critério (3.2) se se usar a implementação por colunas. Na tabela 3.2 apresentamos o resultado dessa comparação. A implementação por colunas é referenciada por IMPL1 e a implementação por linhas por IMPL2.

Nestas comparações foi usado o método Bloco 1 com atc = 3. Neste caso, incluímos na tabela 3.2 as colunas respeitantes ao tempo de construção da estrutura de dados (TF), uma vez que ele difere nas duas implementações. Como seria de esperar esse tempo é ligeiramente superior para a implementação por linhas. No entanto, essa implementação mostra-se superior no funcionamento do algoritmo, quer no número de operações quer no tempo de execução, pelo que no computo geral se torna mais eficiente. De notar que o número de operações é sempre bastante menor, enquanto que a diferença no tempo de execução não é tão acentuada. Tal facto deve-se à necessidade de sempre que o algoritmo de Murty é utilizado se transpor a factorização para se obter o esquema por colunas necessário ao uso do algoritmo de Bennett.

Como referimos atrás, a escolha dos parâmetros p e atc condiciona a eficiência dos algoritmos bloco. Por isso, fizemos também alguma experiência que nos permitisse concluir quais os valores mais aconselháveis para esses parâmetros. Começámos por fazer variar o valor de atc no algoritmo Bloco 1. Repare-se que se for usado o valor atc = n, isso é equivalente à resolução do problema pelo método de Murty. Incluímos esse valor para simultaneamente dar uma ideia da eficiência do algoritmo bloco quando comparado com o método de Murty. Os resultados dessa comparação aparecem na tabela 3.3.

Como seria de esperar a eficiência do algoritmo decresce à medida que atc aumenta, pois o método de Murty vai ser chamado mais cedo e vai executar um maior número de iterações. No entanto, como se pode observar, mesmo para atc = 10 os resultados obtidos são muito melhores do que os obtidos com a aplicação simples do método de Murty.

	IMPL1			IMPL2		
Problema	TF	NO	T	TF	NO	T
TP1	0.39	20.3	5.04	0.42	18.5	4.33
TP2	0.21	7.24	2.16	0.24	6.58	1.69
TP3	0.22	5.83	1.9	0.24	5.23	1.42
TP4	0.32	11.1	3.46	0.37	9.98	2.59
TP5	0.23	4.33	1.78	0.26	3.81	1.27
TP6	0.05	1.62	0.44	0.06	1.14	0.34
TP7	0.40	13.6	2.83	0.48	12.9	2.69
TP8	0.24	18.1	3.74	0.26	17.4	3.59
TP17	0.12	2.01	0.75	0.15	1.68	0.65
TP18	0.12	1.52	0.76	0.15	1.39	0.62
TP19	0.35	3.87	1.26	0.41	2.39	1.23
TP20	0.08	0.7	0.31	0.1	0.62	0.23
TP21	0.13	1.05	0.48	0.16	0.93	0.34
TP22	0.13	1.35	0.68	0.16	1.23	0.52
TP11	0.08	0.7	0.39	0.1	0.65	0.27
TP12	0.17	1.26	0.68	0.21	1.18	0.52

Tabela 3.2

Problema		MURTY	Bloco 1			
		atc=n	atc=1	atc=2	atc=5	atc=10
TP1	NI	590	6	6	9	9
	NO	283.7	20.3	20.3	20.2	20.2
	T	63.17	5.04	5.04	5.07	5.07
TP3	NI	658	6	6	6	6
	NO	155.0	5.83	5.83	5.83	5.83
	T	43.65	1.9	1.9	1.9	1.9
TP7	NI	298	3	4	4	4
	NO	190.8	62.9	46.6	46.6	46.6
	T	37.5	11.76	8.92	8.92	8.92
TP17	NI	288	6	7	9	9
	NO	12.4	2.19	2.01	2.05	2.05
	T	6.54	0.81	0.75	0.78	0.78
TP22	NI	484	5	5	9	9
	NO	37.9	1.35	1.35	1.96	1.96
	T	19.41	0.68	0.68	1.05	1.05
TP11	NI	451	4	5	5	5
	NO	24.2	0.66	0.7	0.7	0.7
	T	9.67	0.37	0.39	0.39	0.39

Tabela 3.3

Nas nossas experiências comparámos também a eficiência do algoritmo de Chandrasekaran (algoritmo Bloco para LCPs com matrizes K) com o método de Murty. O resultado dessa comparação aparece na tabela 3.4 e indica uma grande superioridade do algoritmo bloco. Repare-se também que, enquanto que o número de iterações do método de Murty aumenta com o número de variáveis z_i básicas na solução, esse facto não parece afectar o número de iterações do método Bloco. Neste último caso só se nota um acréscimo do número de operações e do tempo de execução porque, em cada iteração, as matrizes M_{FF} têm dimensões maiores à medida que se aumenta a dimensão do conjunto F correspondente à solução do LCP.

		MURTY			CHANDRASEKARAN		
Problema	VA	NI	NO	T	NI	NO	T
TP17	417	83	1.02	0.95	3	0.097	0.08
	183	317	15.69	7.48	4	1.85	0.57
TP18	694	106	1.41	1.63	4	0.141	0.15
	332	468	26.55	14.6	3	0.79	0.42
TP19	847	153	3.79	3.39	4	0.263	0.22
	391	609	5.61	26.8	4	6.66	1.35

Tabela 3.4

Uma conclusão idêntica à anterior pode ser tirada ao comparar a eficiência do método de Murty e do algoritmo Bloco 1 para matrizes P, quando se faz variar a dimensão do conjunto F correspondente à solução do LCP. Na tabela 3.5 apresenta-se o resultado da comparação desses dois métodos para alguns LCPs, que nos permite verificar que o comportamento do algoritmo Bloco 1 para matrizes P é muito semelhante ao do método de Chandrasekaran para matrizes K.

A tabela 3.6 apresenta os resultados da comparação entre o método pivotal simples de Criss-Cross e o algoritmo bloco na resolução de LCPs monótonos não

		MURTY			BLOCO 1		
Problema	VA	NI	NO	T	NI	NO	T
TP1	436	116	13.0	3.41	5	2.73	0.63
	339	241	57.1	13.6	6	10.0	2.29
	242	332	114.4	26.6	6	20.7	4.67
	194	430	117.9	41.5	6	24.6	5.55
	48	660	430.8	95.8	7	44.3	9.95
TP3	720	360	19.9	7.98	5	1.36	0.45
	560	626	101.4	29.2	7	4.25	1.15
	400	748	206.9	56.4	8	8.96	2.29
	240	910	326.7	89.9	7	13.5	3.24
	80	1164	528.9	146.0	6	17.7	4.04
TP7	180	202	39.9	7.98	3	3.59	0.81
	140	324	100.9	19.81	3	6.6	1.44
	100	292	114.9	22.4	3	12.9	2.69
	60	290	153.5	29.8	3	25.7	5.16
	20	240	161.4	30.9	3	46.1	9.03
TP22	900	280	6.03	5.41	2	0.2	0.11
	700	572	35.0	18.5	2	0.49	0.21
	500	804	79.6	35.3	2	0.83	0.3
	300	932	120.4	50.7	2	1.22	0.41
	100	980	154.7	62.9	3	2.03	0.65
TP11	450	192	2.48	1.87	3	0.13	0.08
	350	386	14.9	6.7	3	0.35	0.16
	250	452	28.9	11.32	5	0.82	0.33
	150	470	42.1	16.0	5	1.14	0.44
	50	1401	144.9	53.0	7	2.28	0.84

Tabela 3.5

estritamente monótonos. É fácil de concluir que o número de iterações do método Bloco é muito maior do que o que é vulgar acontecer para os LCPs estritamente monótonos. Neste caso, como há colunas que só podem ser tornadas básicas se outras entretanto forem tornadas não básicas, a aplicação do critério (3.2) não deixa que certos elementos sejam acrescentados ao conjunto F. Esses elementos só entram em F quando finalmente é utilizado o método Criss-Cross. Na tabela 3.6, na coluna correspondente ao número de iterações do método Bloco, aparece a soma de dois inteiros, em que o primeiro é o número de iterações efectuadas pelo método Bloco e o segundo o número de iterações efectuado pelo método Criss- Cross.

		Criss-Cross			Bloco 2 (p = 1)		
Problema	VA	NI	NO	T	NI	NO	T
TP24	410	849	138.8	60.0	7+5	7.2	27.5
TP25	514	843	142.6	64.5	6+5	5.4	28.7

Tabela 3.6

Para avaliar da importância do valor do parâmetro p comparámos o funcionamento do algoritmo Bloco 2 para p = 1 e p = 3. O resultado dessa comparação pode ser visto na tabela 3.7. Como se pode verificar, o algoritmo de um modo geral tem melhor desempenho para p = 3. Isto deve-se ao facto de, para p = 1, o método de Murty ser chamado sempre que o número de inadmissibilidades aumenta. Ora, muitas vezes, esse número aumenta numa iteração para reduzir drasticamente logo na iteração seguinte. Nestes casos, com p = 3, o método de Murty não é utilizado e isso reduz bastante o número total de iterações. É de esperar o mesmo comportamento para valores um pouco maiores para p ($p \leq 10$ digamos).

Finalmente, na tabela 3.8, apresentamos uma comparação entre o método de Keller e o algoritmo Bloco 2. Como a fase de análise simbólica inicial é diferente nos dois métodos, por uma implementação ser orientada por colunas e a outra por linhas,

		p = 1		p = 3	
Problema	VA	NI	T	NI	T
TP1	363	4	2.0	4	2.0
	121	6	4.5	6	4.5
TP4	750	4	2.2	4	2.2
	250	7	4.6	8	4.9
TP8	450	2	2.4	2	2.4
	150	3	4.4	3	4.4
TP23	1125	14	3.5	19	4.0
	375	10	5.2	13	5.8
TP26	814	6	5.6	6	5.6
	270	50	51.0	9	22.5
TP27	1439	22	10.5	7	7.5
	479	8	25.5	8	24.5

Tabela 3.7

optámos por apresentar o tempo total de funcionamento, incluindo o tempo de ordenação e da fase simbólica. Assim

TT = tempo total em segundos de CPU = tempo de ordenação + tempo da fase de análise simbólica + tempo de execução do algoritmo.

Da análise destes resultados pode-se concluir a grande supremacia do método Bloco 2 sobre o método de Keller em todas as situações. Tal como foi referido para o método de Murty, também aqui se verifica que, enquanto que a dimensão do problema e o número de variáveis z_i positivas na solução óptima afectam o funcionamento do método de Keller, tal não tem grande influência no número de iterações do método Bloco 2.

De salientar que mesmo no problema TP23, em que o método Bloco 2 não funciona tão bem como nos outros casos, o seu desempenho é superior ao método de Keller. Este tipo de problema vem referenciado em [57] como um caso em que os métodos iterativos não conseguem obter a solução do LCP. O pior funcionamento do método Bloco 2 fica-se a dever ao facto de o número de inadmissibilidades não decrescer e o método de Murty ser muitas vezes chamado no decorrer do processo.

Problema	N	VA		KELLER	BLOCO 2
TP1	484	242	NI	242	6
			NO	57.7	21.2
			TT	22.2	9.2
			RES	2-11	1-11
TP4	1000	500	NI	550	7
			NO	168.	12.4
			TT	61.8	9.7
			RES	2-12	2-12
TP8	600	300	NI	300	3
			NO	64.4	8.7
			TT	24.3	10.1
			RES	7-12	3-12
TP23	1500	750	NI	829	67
			NO	1492.	105.
			TT	271.1	31.9
			RES	5-12	4-12

Tabela 3.8

O método Bloco tem também um melhor comportamento do que os algoritmos pivotais simples na resolução de LCPs com matrizes triadiagonais . Nos exemplos que apresentamos a seguir verifica-se que, principalmente nos casos em que o número de elementos do conjunto F correspondente à solução não é muito grande, o método bloco funciona com um muito menor número de iterações mas tal não é acompanhado por idêntica diminuição no número de operações e tempo de execução. Tal deve-se ao facto de a implementação que usamos para matrizes tridiagonais ser

particularmente eficaz para modificações de apenas um elemento no conjunto F. Na tabela 3.9 são tratados LCPs com matrizes tridiagonais da classe P e vector q gerado aleatoriamente de modo a ter NEG componentes negativas.

n	NEG		MURTY	BLOCO 1
1000	500	NI	528	3
		NO	0.32	0.23
		T	0.60	0.19
	750	NI	775	3
		NO	0.86	0.44
		T	0.85	0.21
5000	2500	NI	1397	3
		NO	0.61	0.53
		T	4.17	0.71
	3750	NI	4011	3
		NO	4.73	2.43
		T	14.53	1.06

Tabela 3.9

Para terminar esta secção discutimos o comportamento dos métodos Bloco na resolução de dois LCPs referenciados na literatura como exemplos em que o método de Murty tem um comportamento exponencial. O primeiro LCP aparece em [65] e a sua matriz M é triangular superior, com todos os elementos diagonais iguais a 1 e os elementos não nulos iguais a 2. O vector q tem todas as componenetes iguais a -1. Como é demonstrado em [65], o método de Murty resolve este LCP em $2^n - 1$ iterações, com n dimensão da matriz M. Experiência computacional mostrou que o algoritmo Bloco 1 necessita somente de n iterações.

Como exemplo de ilustração considere-se o LCP com n = 4 e M e q dados por:

$$q = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 2 & 2 & 2 \\ & 1 & 2 & 2 \\ & & 1 & 2 \\ & & & 1 \end{bmatrix}$$

A aplicação do método de Murty conduz à seguinte sequência de conjuntos F:

{ 1 }, { 1, 2 }, { 2 }, { 2, 3 }, { 1, 2, 3 }, { 1, 3 }, { 3 }, { 3, 4 }, { 1, 3, 4 }, { 1, 2, 3, 4 }, { 2, 3, 4 }, { 2, 4 }, { 1, 2, 4 }, { 1, 4 }, { 4 }.

Assim o processo requer exactamente as 2^4 -1 iterações previstas.

Resolvendo o mesmo LCP pelo algoritmo Bloco 1 só são necessárias 4 iterações, obtendo-se a seguinte sequência de conjuntos F:

{ 1, 2, 3, 4 }, { 2, 4 }, { 1, 4 }, { 4 }.

Por seu turno Fathi [18] também fornece um outro LCP em que é evidente o comportamento exponencial do método de Murty. As matrizes desses LCPs têm a forma $A = M^T M$ com M a matriz usada por Murty e referida anteriormente. Assim A é a matriz simétrica dada por:

$$i = 1, \ldots n \quad \begin{cases} a_{ii} = 4(i - 1) + 1 \\ a_{ij} = 4(i - 1) + 2, \; j = 1, \ldots, n - 1 \end{cases}$$

Para LCPs com matrizes deste tipo obtêm-se números de iterações idênticos aos obtidos com o caso anterior. Isso é atestado na tabela 3.10 que representa o número de iterações do método de Murty e bloco para três valores de n.

n	MURTY	BLOCO
5	$31 = 2^5 - 1$	5
10	$1023 = 2^{10} - 1$	10
15	$32767 = 2^{15} - 1$	15

Tabela 3.10

CAPÍTULO IV

MÉTODOS PIVOTAIS SIMPLES

PARA O PROBLEMA LINEAR COMPLEMENTAR COM LIMITES

1 INTRODUÇÃO

Dados $-\infty < a_i < b_i < +\infty$, $i = 1, \ldots, n$, um vector $q \in R^n$ e uma matriz $M \in R^{n \times n}$, o Problema Linear Complementar com Limites (BLCP) consiste em encontrar vectores z e w de R^n (ou provar a sua não existência) tais que:

$$\left.\begin{array}{l} w = q + M z \\ a_i \le z_i \le b_i \\ z_i = a_i \Rightarrow w_i \ge 0 \\ z_i = b_i \Rightarrow w_i \le 0 \\ a_i < z_i < b_i \Rightarrow w_i = 0 \end{array}\right\} \quad i = 1, \ldots, n \qquad (4.1)$$

Tal como para o LCP, a existência e unicidade de solução do BLCP depende da classe da matriz M. Se $M \in P$ então o BLCP (4.1) é estritamente monótono e tem solução única para qualquer $q \in R^n$ [2].

O BLCP (4.1) é equivalente ao Problema de Desigualdades Variacionais Lineares (LVI):

Encontrar $\bar{z} \in \Omega$ tal que

$$(z - \bar{z})^T (q + M \bar{z}) \geq 0 \text{ , para todo o } z \in \Omega$$

$$\text{com } \Omega = \{ z \in R^n : a_i \leq z_i \leq b_i \text{ , } i = 1, 2, \text{ , } n \}$$

Quando M é uma matriz simétrica P, então o BLCP e o LVI são equivalentes [10] ao programa quadrático

$$\begin{aligned} &\text{Min } q^T z + \tfrac{1}{2} z^T M z \\ &\text{sujeito a } z \in \Omega \end{aligned} \qquad (4.2)$$

Se o problema (4.2) não for convexo, a resolução de (4.1) fornece um ponto estacionário de (4.2). Inversamente, qualquer ponto estacionário do programa quadrático (4.2) é uma solução do BLCP (4.1).

É de notar que, na definição do conjunto Ω, algumas componentes a_i e b_i podem ser $-\infty$ ou $+\infty$ respectivamente, podendo inclusivamente os limites de algumas variáveis ser $-\infty$ e $+\infty$ simultaneamente.

O BLCP monótono tem um elevado número de aplicações, nomeadamente na resolução de problemas de equações diferenciais às derivadas parciais [26], análise elastoplástica de estruturas [67] [73] e modelos económicos de equilíbrio espacial [74]. Alguns programas convexos não lineares e de desigualdades variacionais podem também ser resolvidos por técnicas sequenciais, envolvendo a resolução de vários BLCPs. Em algumas aplicações a matriz é PSD. Neste caso, o BLCP é monótono, mas não estritamente monótono, não havendo garantia de existência de solução nesse caso.

O BLCP reduz-se ao LCP (2.1) quando $a_i = 0$ e $b_i = +\infty$ para todo $i = 1, \ldots, n$. Por isso, os principais métodos de resolução do BLCP são generalizações dos algoritmos anteriormente descritos para o LCP e que são

convenientemente modificados para lidarem com as restrições $a_i \leq z_i \leq b_i$. Este capítulo debruça-se sobre essas extensões.

2 Solução Básica para o BLCP

Tal como no LCP, os métodos directos para a resolução do BLCP baseiam-se em soluções básicas e operações pivotais com esse tipo de soluções. A existência de dois tipos de limites (inferiores e superiores) implica a necessidade de uma adaptação do conceito de solução básica para o BLCP. Para isso consideremos os conjuntos

$$\begin{aligned} F &= \{ i : z_i \text{ é básica } \} \\ T_1 &= \{ i : w_i \text{ é básica e } z_i = b_i \} \\ T_2 &= \{ i : w_i \text{ é básica e } z_i = a_i \} \end{aligned}$$

que verificam

$$T_1 \cup T_2 = T \quad ; \quad T_1 \cap T_2 = \varnothing \quad ; \quad F \cap T = \varnothing \quad ; \quad F \cup T = \{ 1 , \ldots , n \}$$

Se considerarmos agora o sistema $w = q + Mz$ e efectuarmos uma operação pivotal com pivot M_{FF}, obtém-se o seguinte sistema:

$$\begin{bmatrix} z_F \\ w_{T1} \\ w_{T2} \end{bmatrix} = \begin{bmatrix} \bar{q}_F \\ \bar{q}_{T1} \\ \bar{q}_{T2} \end{bmatrix} + \begin{bmatrix} \bar{M}_{FF} & \bar{M}_{FT1} & \bar{M}_{FT2} \\ \bar{M}_{T1F} & \bar{M}_{T1T1} & \bar{M}_{T1T2} \\ \bar{M}_{T2F} & \bar{M}_{T2T1} & \bar{M}_{T2T2} \end{bmatrix} \begin{bmatrix} w_F \\ z_{T1} \\ z_{T2} \end{bmatrix} \qquad (4.3)$$

em que os valores de $\bar{q}$ e de $\bar{M}$ satisfazem expressões semelhantes às apresentadas em (1.11). A solução básica associada aos conjuntos de índices F , T_1 e T_2 contém as variáveis não básicas

$$z_{T1} = b_{T1} \quad ; \quad z_{T2} = a_{T2} \quad ; \quad w_F = 0$$

e as variáveis básicas

$$\begin{aligned} z_F &= \bar{q}_F + \bar{M}_{FT1}\, b_{T1} + \bar{M}_{FT2}\, a_{T2} = \\ &= - M_{FF}^{-1} (q_F + M_{FT1}\, b_{T1} + M_{FT2}\, a_{T2}) \\ w_T &= \bar{q}_T + \bar{M}_{TT1}\, b_{T1} + \bar{M}_{TT2}\, a_{T2} = \\ &= q_T + M_{TF}\, z_F + M_{TT1}\, b_{T1} + M_{TT2}\, a_{T2} = q_T + M_{T.}\, z \end{aligned}$$

Esta solução é admissível e, por isso, solução do BLCP se

$$a_F \le z_F \le b_F \quad ; \quad w_{T1} \le 0 \quad ; \quad w_{T2} \ge 0 \tag{4.4}$$

Tal como anteriormente, o conjunto das inadmissibilidades H é constituído pelos índices que não satisfazem as relações (4.4). Este conjunto satisfaz $H = \bigcup_{i=1}^{4} H_i$ com

$$\begin{aligned} H_1 &= \{ i \in F : z_i > b_i \} \quad ; \quad H_2 = \{ i \in F : z_i < a_i \} \\ H_3 &= \{ i \in T_1 : w_i > 0 \} \quad ; \quad H_4 = \{ i \in T_2 : w_i < 0 \} \end{aligned} \tag{4.5}$$

É de notar que podemos considerar que $a_i \in \{ 0, -\infty \}$ sem qualquer perda de generalidade. Com efeito, basta efectuar uma mudança linear de variável ($z = z - a'$ com $a'_i = a_i$ se $a_i > -\infty$ e $a'_i = 0$ se $a_i = -\infty$) para que um BLCP com limites inferiores finitos diferentes de zero se transforme num BLCP nestas condições. Tendo em conta essa simplificação do BLCP, qualquer solução básica associada ao quadro (4.3) é dada por:

$$z_F = - M_{FF}^{-1} (q_F + M_{FT1}\, b_{T1}) \; ; \; z_{T1} = b_{T1} \; ; \; z_{T2} = 0 \tag{4.6a}$$

$$w_T = q_T + M_{TF}\, z_F + M_{TT_1}\, b_{T_1} \; ; \; w_F = 0 \qquad (4.6b)$$

Tal como no LCP iremos usar a notação BLCP (b, q, M) para representar um BLCP (4.1) onde os limites inferiores finitos são todos nulos.

Os algoritmos a estudar neste capítulo são normalmente iniciados com uma solução básica que tem associados os seguintes conjuntos de índices

$$F = \{ i : \; a_i = -\infty \text{ e } b_i = +\infty \},$$
$$T_1 = \{ i : \; a_i = -\infty \text{ e } b_i < +\infty \},$$
$$T_2 = \{ i : \; a_i = 0 \}.$$

Os processos efectuam modificações nestes conjuntos, obtendo assim novas soluções básicas. A solução será encontrada quando $H = \bigcup_{i=1}^{4} H_i = \varnothing$ sendo H_i , i = 1, 2, 3, 4 os conjuntos definidos em (4.5).

Nas secções seguintes iremos apresentar alguns dos métodos directos para a resolução do BLCP (4.1). Assim, discutiremos primeiramente duas extensões do método de Murty. O primeiro desses processos tem apenas garantia de convergência para matrizes K enquanto que o segundo é aplicável a qualquer matriz P. A extensão do método de Keller para matrizes simétricas PD ou PSD é também apresentada neste capítulo.

3 Primeira Extensão do Método de Murty

Este algoritmo foi primeiramente apresentado em [40] e, tal como no método de Murty para o LCP, tenta-se remover uma inadmissibilidade por iteração sem

qualquer preocupação quanto à admissibilidade das restantes variáveis. Em cada iteração escolhe-se o primeiro índice de H e tenta-se remover essa inadmissibilidade. Se s é o índice escolhido, então quatro casos podem acontecer que são discutidos a seguir.

(i) $s \in H_1 \Rightarrow s \in F \text{ e } z_s > b_s \quad (b_s < +\infty)$

Efectuando uma operação pivotal com pivot $\bar{m}_{ss}$ a variável z_s torna-se não básica por troca com w_s. Contudo, há neste caso a necessidade da escolha por um dos limites 0 ou b_s. Como b_s está mais próximo do valor de z_s, é natural que a variável se torne não básica com valor b_s. Para que a inadmissibilidade seja removida é necessário que o correspondente valor de w_s seja não positivo. Mas se $z_s = b_s$, então

$$\hat{w}_s = - \frac{\bar{q}_s}{\bar{m}_{ss}} - \sum_{i \in T_1} \frac{\bar{m}_{si}}{\bar{m}_{ss}} b_i + \frac{1}{\bar{m}_{ss}} b_s = \frac{1}{\bar{m}_{ss}} \left(b_s - \bar{q}_s - \sum_{i \in T_1} \bar{m}_{si} b_i \right) =$$

$$= \frac{1}{\bar{m}_{ss}} (b_s - z_s) < 0$$

o que está de acordo com o pretendido.

(ii) $s \in H_2 \Rightarrow s \in F \text{ e } z_s < 0 \quad (a_s = 0)$

Raciocinando de modo semelhante ao caso anterior, z_s passa a não básica com valor zero e w_s ficará básica com valor positivo. Com efeito tem-se

$$\hat{w}_s = - \frac{\bar{q}_s}{\bar{m}_{ss}} - \sum_{i \in T_1} \frac{\bar{m}_{si}}{\bar{m}_{ss}} b_i = - \frac{1}{\bar{m}_{ss}} \left(\bar{q}_s + \sum_{i \in T_1} \bar{m}_{si} b_i \right) =$$

$$= - \frac{z_s}{\bar{m}_{ss}} > 0$$

e a inadmissibilidade é removida.

(iii) $s \in H_3 \Rightarrow s \in T_1$ e $w_s > 0 \quad (b_s < +\infty)$

O valor de w_s é dado pela seguinte expressão:

$$w_s = \bar{q}_s + \sum_{\substack{i \in T_1 \\ i \neq s}} \bar{m}_{si}\, b_i + \bar{m}_{ss}\, b_s.$$

Se a variável z_s tomar o valor zero, então w_s vem dado por

$$\hat{w}_s = w_s - \bar{m}_{ss} b_s.$$

Se este valor continuar a ser positivo a inadmissibilidade é removida sem necessidade de efectuar qualquer operação pivotal. Caso contrário, isto é, se

$$w_s - \bar{m}_{ss} b_s < 0,$$

deve-se efectuar operação pivotal com pivot $\bar{m}_{ss}$ para anular w_s e fazer z_s decrescer. Então tem-se

$$\hat{z}_s = - \frac{\bar{q}_s}{\bar{m}_{ss}} - \sum_{\substack{i \in T_1 \\ i \neq s}} \frac{\bar{m}_{si}}{\bar{m}_{ss}}\, b_i = - \frac{1}{\bar{m}_{ss}} \left(\bar{q}_s + \sum_{\substack{i \in T_1 \\ i \neq s}} \bar{m}_{si}\, b_i \right) =$$

$$= - \frac{1}{\bar{m}_{ss}} (w_s - \bar{m}_{ss} b_s) = b_s - \frac{w_s}{\bar{m}_{ss}}.$$

Donde $0 < \hat{z}_s < b_s$ e a inadmissibilidade é removida.

Se $a_s = -\infty$, então a operação pivotal deve ser realizada em qualquer dos casos ficando $\hat{z}_s = b_s - \frac{w_s}{\bar{m}_{ss}} < b_s$. É de notar que esse valor não é necessariamente positivo mas isso não conduz a qualquer problema por o limite inferior de z_s ser $-\infty$.

(iv) $s \in H_4 \Rightarrow s \in T_2$ e $w_s < 0 \quad (a_s = 0)$

Se a variável z_s tomar o valor b_s ($b_s < +\infty$), então o novo valor de w_s é

$$\hat{w}_s = \bar{q}_s + \sum_{i \in T_1} \bar{m}_{si}\, b_i + \bar{m}_{ss}\, b_s = w_s + \bar{m}_{ss}\, b_s$$

Se este valor continuar a ser negativo, então, a inadmissibilidade é removida sem ser necessário efectuar uma operação pivotal. Caso contrário, isto é, se

$$w_s + \bar{m}_{ss}\, b_s > 0$$

há que efectuar uma operação pivotal com pivot $\bar{m}_{ss}$, após o que se obtém

$$\hat{z}_s = - \frac{\bar{q}_s}{\bar{m}_{ss}} - \sum_{i \in T_1} \frac{\bar{m}_{si}}{\bar{m}_{ss}}\, b_i = - \frac{1}{\bar{m}_{ss}} \left(\bar{q}_s + \sum_{i \in T_1} \bar{m}_{si}\, b_i \right) =$$

$$= - \frac{w_s}{\bar{m}_{ss}} > 0.$$

Como $w_s + \bar{m}_{ss}\, b_s > 0$, então $0 < \hat{z}_s < b_s$ e a inadmissibilidade foi removida.

Se $b_s = +\infty$ deve-se efectuar sempre a operação pivotal e tem-se

$$0 < \hat{z}_s = - \frac{w_s}{\bar{m}_{ss}} < b_s = +\infty \ .$$

Estas considerações mostram que é sempre possível remover a primeira inadmissibilidade associada a uma dada solução básica. À semelhança do método de Murty, podemos desenvolver um processo que em cada iteração procura remover uma inadmissibilidade. Esse algoritmo é apresentado a seguir.

ALGORITMO EXTMURTY 1

PASSO INICIAL: Faça $F = \{ i : a_i = -\infty \text{ e } b_i = +\infty \}$,

$T_1 = \{ i : a_i = -\infty \text{ e } b_i < +\infty \}$, $T_2 = \{ i : a_i = 0 \}$.

PASSO GERAL: Calcule z_F e w_T a partir de (4.6). Defina $H = \bigcup_{i=1}^{4} H_i$,

com H_i (i=1, 2, 3, 4) dados por (4.5).

$$\text{Se } H = \varnothing, \quad \left| \begin{array}{l} z_F = -M_{FF}^{-1} (q_F + M_{FT1} b_{T1}) \\ z_{T1} = b_{T1} \\ z_{T2} = 0 \\ w_F = 0 \\ w_T = q_T + M_{TT1} b_{T1} + M_{TF} z_F \end{array} \right.$$

é solução do BLCP. Caso contrário seja $s = \min \{ i \in H \}$.

Então :

(i) $s \in H_1 \Rightarrow F = F - \{ s \}$ e $T_1 = T_1 \cup \{ s \}$;

(ii) $s \in H_2 \Rightarrow F = F - \{ s \}$ e $T_2 = T_2 \cup \{ s \}$;

(iii) $s \in H_3 \Rightarrow T_1 = T_1 - \{ s \}$

$$a_s = -\infty \Rightarrow F = F \cup \{ s \}$$

$$a_s = 0 \Rightarrow \left| \begin{array}{ll} w_s - \bar{m}_{ss} b_s \geq 0 & \Rightarrow T_2 = T_2 \cup \{ s \} \\ w_s - \bar{m}_{ss} b_s < 0 & \Rightarrow F = F \cup \{ s \} \end{array} \right.$$

(iv) $s \in H_4 \Rightarrow T_2 = T_2 - \{ s \}$

$$b_s = +\infty \Rightarrow F = F \cup \{ s \}$$

$$b_s < +\infty \Rightarrow \left| \begin{array}{ll} w_s + \bar{m}_{ss} b_s \leq 0 & \Rightarrow T_1 = T_1 \cup \{ s \} \\ w_s + \bar{m}_{ss} b_s > 0 & \Rightarrow F = F \cup \{ s \} \end{array} \right.$$

Repita o Passo Geral.

A descrição do algoritmo mostra que em cada iteração é necessário resolver um sistema com matriz M_{FF} para obter $\bar{q}_F$. Além disso, se $s \in H_3$ e $a_s = 0$ ou $s \in H_4$ e $b_s < +\infty$, é ainda necessário conhecer $\bar{m}_{ss} = m_{ss} + M_{sF}\bar{M}_{Fs}$, ou seja, deve-se resolver um sistema com matriz M_{FF} para obter $\bar{M}_{Fs} = -M_{FF}^{-1} M_{Fs}$. Tal como no LCP, é possível aproveitar o facto de se conhecer $\bar{M}_{Fs}$ para actualizar $\bar{q}_F$ sem ser necessário resolver mais nenhum sistema. Isso é mostrado a seguir.

(I) Seja $s \in H_3$. Dois casos podem acontecer.

(i) Se s passa de T_1 a T_2, então

$$\hat{z}_s = 0$$

$$\hat{w}_s = w_s - \bar{m}_{ss} b_s$$

$$\hat{z}_F = z_F - \bar{M}_{Fs} b_s$$

$$\hat{w}_T = w_T - b_s (M_{Ts} + M_{TF} \bar{M}_{Fs})$$

(ii) Se s passa de T_1 a F, tem-se

$$\hat{z}_s = b_s - \frac{w_s}{\bar{m}_{ss}}$$

$$\hat{z}_F = z_F + \bar{M}_{Fs} (\hat{z}_s - b_s)$$

$$\hat{w}_T = w_T - \frac{w_s}{\bar{m}_{ss}} (M_{Ts} + M_{TF} \bar{M}_{Fs})$$

(II) Seja $s \in H_4$. Tal como anteriormente, há dois casos que a seguir se apresentam.

(i) Se s passa de T_2 a T_1, então

$$\hat{z}_s = b_s$$

$$\hat{w}_s = w_s + \bar{m}_{ss} b_s$$

$$\hat{z}_F = z_F + \bar{M}_{FS} b_S$$

$$\hat{w}_T = w_T + b_S (M_{TS} + M_{TF} \bar{M}_{FS})$$

(ii) Se s passa de T_2 a F, vem

$$\hat{z}_S = - \frac{w_S}{\bar{m}_{SS}}$$

$$\hat{z}_F = z_F + \bar{M}_{FS} \hat{z}_S$$

$$\hat{w}_T = w_T - \frac{w_S}{\bar{m}_{SS}} (M_{TS} + M_{TF} \bar{M}_{FS})$$

Como será demonstrado mais adiante, é possível estabelecer a convergência deste algoritmo no caso da matriz do problema ser da classe K. Para uma matriz $M \in P$ que não seja Z, só é possível provar a convergência nos dois casos extremos de serem todos os limites inferiores iguais a $- \infty$ ou de todos os limites superiores iguais a $+ \infty$. Com efeito, encontrámos [37] um exemplo de um ciclo do algoritmo na resolução de um BLCP com todos os limites finitos e $M \in P$. Esse exemplo será apresentado na última secção deste capítulo. Apesar disso, o algoritmo tem um comportamento razoável, mesmo nos casos em que não há garantia de convergência, sendo o exemplo referido o único em que, até agora, se nos deparou um ciclo em dezenas de casos experimentados com matrizes P.

Antes de apresentarmos a convergência do algoritmo, iremos ver que o algoritmo converge num número finito de iterações para qualquer matriz $M \in P$, desde que os limites inferiores sejam todos $- \infty$ ou os limites superiores sejam todos $+ \infty$. Para isso, definam-se os seguintes conjuntos de índices

$$G = \{ i : a_i = - \infty \text{ e } b_i = + \infty \} \quad \text{e} \quad K = \{ 1 , \ldots , n \} - G.$$

Se todos os limites inferiores são $- \infty$. então o algoritmo tem a seguinte forma:

PASSO INICIAL: Faça $F = G$ e $T_1 = K$.

PASSO GERAL: Calcule z_F e w_{T_1} a partir de (4.6) e seja $H = H_1 \cup H_3$, com H_1 e H_3 definidos por (4.5).

$$\text{Se } H = \emptyset, \text{ então } \left| \begin{array}{l} z_F = -M_{FF}^{-1}(q_F + M_{FT1} b_{T1}) \\ z_{T1} = b_{T1} \\ w_F = 0 \\ w_{T1} = q_{T1} + M_{FT1} b_{T1} + M_{TF} z_F \end{array} \right.$$

é solução do BLCP.

Caso contrário, seja $s = \min \{ i : i \in H \}$ e faça

$s \in H_1 \Rightarrow F = F - \{ s \}$ e $T_1 = T_1 \cup \{ s \}$,

$s \in H_3 \Rightarrow T_1 = T_1 - \{ s \}$ e $F = F \cup \{ s \}$.

Repita o Passo Geral.

Considere-se agora o caso de todos os limites superiores serem $+ \infty$. Neste caso o conjunto T_1 é sempre vazio e o algoritmo pode ser simplificado para a seguinte forma:

PASSO INICIAL: Faça $F = G$ e $T_2 = K$.

PASSO GERAL: Calcule z_F e w_{T_2} a partir de (4.6) e seja $H = H_2 \cup H_4$, com H_2 e H_4 definidos por (4.5).

$$\text{Se } H = \emptyset, \text{ então } \left| \begin{array}{l} z_F = -M_{FF}^{-1} q_F \\ z_{T2} = 0 \\ w_F = 0 \\ w_{T2} = q_{T2} + M_{T2F} z_F \end{array} \right.$$

é solução do BLCP.

Caso contrário, seja $s = \min \{ i : i \in H \}$ e faça

$s \in H_2 \Rightarrow F = F - \{ s \}$ e $T_2 = T_2 \cup \{ s \}$,

$s \in H_4 \Rightarrow T_2 = T_2 - \{ s \}$ e $F = F \cup \{ s \}$.

Repita o Passo Geral.

Note-se que se neste último algoritmo $G = \varnothing$, então este processo reduz-se ao método de Murty descrito na segunda secção do capítulo 2. Além disso, é fácil de verificar que, no caso de todos os limites inferiores serem $-\infty$, o algoritmo é equivalente ao outro processo apresentado imediatamente a seguir a ele, se se efectuar a mudança de variável $z_i = b_i - x_i$ para $i \in K$.

Após o passo inicial, a matriz transformada do problema é da forma:

$$\begin{bmatrix} \bar{M}_{GG} & \bar{M}_{GK} \\ \bar{M}_{KG} & \bar{M}_{KK} \end{bmatrix}$$

com $\bar{M}_{KK} = M_{KK} - M_{KG} M_{GG}^{-1} M_{GK} = (M \setminus M_{GG})$ o Complemento de Schur de M_{GG} em M. Como $M \in P$, então o Complemento de Schur também pertence a essa classe [8]. Esse facto permite demonstrar a convergência destes dois algoritmos simplificados aplicando a $\bar{M}_{KK}$ a demonstração apresentada por Murty em [64].

Consideremos agora o caso geral em que há limites superiores e inferiores finitos e infinitos e seja $M \in K = P \cap Z$. Para estabelecer a convergência do algoritmo neste caso, necessitamos dos seguintes conceitos, que são referidos em [70].

Definição 4.1: $\bar{z}$ é o elemento mínimo do conjunto X se $\bar{z} \leq z$ para todo o $z \in X$.

Seja ainda $X = \{ z \in R^n : a_i \leq z_i \leq b_i , i = 1 , \ldots , n \}$ e consideremos os conjuntos $S_i = \{ z \in X : z_i < b_i \Rightarrow w_i \geq 0 \}$, $i = 1 , \ldots , n$ que foram também introduzidos em [70]. Como $M \in K$, então, os n conjuntos S_i são limitados inferiormente se $G = \varnothing$. Por isso, o teorema 3.1 de [70] pode ser aplicado ao BLCP (4.1) dando lugar ao seguinte lema.

Lema 4.1: Se $M \in K$ e $G = \varnothing$, então $S = \bigcap_{i=1}^{n} S_i$ tem um elemento mínimo $\bar{z}$ que é a solução única do BLCP (4.1).

Usando o lema 4.1 é possível enunciar e demonstrar o seguinte resultado.

Lema 4.2: Seja $M \in K$ e $G = \varnothing$. Considerem-se os dois problemas BLCP(b, q, M) e BLCP(b, p, M) tais que $q \leq p$. Se $\bar{z}$ e $\tilde{z}$ são as soluções únicas destes problemas respectivamente, então $\bar{z} \geq \tilde{z}$.

Demonstração: Pelo lema 4.1, $\bar{z}$ é o elemento mínimo do conjunto $S = \bigcap_{i=1}^{n} S_i$ e $\tilde{z}$ é o elemento mínimo do conjunto $R = \bigcap_{i=1}^{n} \{ z \in X : z_i < b_i \Rightarrow (p + Mz)_i \geq 0 \}$.

Mas, $q_i \leq p_i \Rightarrow q_i + \sum_k m_{ik} z_k \leq p_i + \sum_k m_{ik} z_k \Rightarrow w_i = (q + Mz)_i \leq (p + Mz)_i$

Por outro lado, se $w_i \geq 0$, também será $(p + Mz)_i \geq 0$ e, por isso, $S \subset R$. Como $\bar{z}$ é um elemento de S também será elemento de R e $\bar{z} \geq \tilde{z}$, por definição de elemento mínimo.

□

Se aplicarmos o algoritmo EXTMURTY1 ao BLCP (4.1), então podemos considerar em cada uma das suas iterações um quadro da forma (4.3). Se $M \in K$, $M_{FF} \in K$ e $M_{FF}^{-1} \geq 0$ [21]. Então $\bar{M}_{FT} = - M_{FF}^{-1} M_{FT} \geq 0$ e $\bar{M}_{TT} \in K$, pois é o complemento de Schur de M_{FF} em M [8]. Tendo em conta as expressões (4.6) e estes resultados é imediata a seguinte propriedade

Teorema 4.1: Se $M \in K$ então em cada iteração do algoritmo EXTMURTY1 tem-se

(i) $\forall_{r \in F}$ z_r é uma função não decrescente das variáveis z_j , $j \in T$

(ii) $\forall_{s \in T}$ z_s é uma função não crescente das variáveis z_j , $j \in T$-{s}

Baseado nestas propriedades é possível estabelecer a convergência do método EXTMURTY1, isto é, podemos enunciar o seguinte resultado

Teorema 4.2: Se $M \in K$ o método EXTMURTY1 é convergente para a solução única do BLCP. Além disso, $H_2 \neq \varnothing$ em todas as iterações, isto é, nenhuma variável z_i se torna negativa.

Demonstração:

A demonstração deste teorema faz-se por indução sobre a dimensão da matriz M. Considere-se primeiramente o caso de ser $G = \varnothing$.

Seja $n = 1$. O BLCP (4.1) tem a seguinte forma:

$$w_1 = q_1 + m_{11} z_1$$
$$z_1 = a_1 \Rightarrow w_1 \geq 0$$
$$z_1 = b_1 \Rightarrow w_1 \leq 0$$
$$0 < z_1 < b_1 \Rightarrow w_1 = 0$$

Como $a_1 \in \{ 0 , -\infty \}$ devemos considerar os dois casos separadamente.

i) $a_1 = -\infty$

Como $G = \varnothing$, então $b_1 < +\infty$ e o algoritmo começa com $T_1 = \{ 1 \}$, ou seja, com $z_1 = b_1$. Se $w_1 = q_1 + m_{11} b_1 < 0$, a solução está encontrada. Caso contrário faz-se $F = \{ 1 \}$, $z_1 = -\frac{q_1}{m_{11}} < b_1$ e obtém-se a solução. Em qualquer dos casos tem-se $H_2 = \varnothing$.

ii) $a_1 = 0$

Neste caso o algoritmo começa com $T_2 = \{ 1 \}$, ou seja, $z_1 = 0$. Se $w_1 = q_1 \geq 0$ a solução está encontrada . Caso contrário, há que considerar separadamente os casos de ser b_1 finito ou infinito. No caso finito faz-se $F = \{ 1 \}$ se $z_1 = -\frac{q_1}{m_{11}} < b_1$. De outro modo $w_1 = q_1 + m_{11} b_1 < 0$ e faz-se $T_1 = \{ 1 \}$. No caso

de b_1 ser infinito não há qualquer limite superior a considerar e por isso faz-se $F = \{ 1 \}$. Em qualquer caso tem-se $H_2 = \varnothing$.

Considere-se agora, como hipótese de indução, que o algoritmo resolve qualquer BLCP com matriz de classe K e dimensão menor ou igual a n-1 e além disso $H_2 = \varnothing$. Para demonstrar que o algoritmo resolve qualquer BLCP de dimensão n com uma matriz $M \in K$, considera-se a seguinte partição de M e dos vectores q e b:

$$M = \begin{bmatrix} M_{LL} & M_{Ln} \\ M_{nL} & m_{nn} \end{bmatrix} \quad ; \quad q = \begin{bmatrix} q_L \\ q_n \end{bmatrix} \quad ; \quad b = \begin{bmatrix} b_L \\ b_n \end{bmatrix}$$

com $L = \{ 1 , \ldots , n - 1 \}$.

Tal como para n = 1, deve-se considerar separadamente os casos de ser $a_n = 0$ e $a_n = - \infty$.

i) $a_n = - \infty$

Como $G = \varnothing$, então $b_n < + \infty$ e $z_n = b_n$ na solução inicial do BLCP. Como se utiliza a regra de Bland, o algoritmo começa por encontrar a solução única $\bar{\bar{z}}$ do BLCP de dimensão n-1 com matriz M_{LL} e termo independente $q_L + M_{Ln}b_n$. A essa solução vai corresponder para a variável w_n o valor

$$\bar{\bar{w}}_n = q_n + M_{nL}\, \bar{\bar{z}}_L + m_{nn}b_n.$$

Dois casos podem acontecer e são discutidos a seguir.

a) $\bar{\bar{w}}_n \leq 0$. A solução $\bar{z} = (\bar{\bar{z}}_L , b_n)$ foi encontrada.

b) $\bar{\bar{w}}_n > 0 \Rightarrow n \in H_3$

Como $a_n = -\infty$, faz-se $T_1 = T_1 - \{ n \}$ e $F = F \cup \{ n \}$. Assim a variável z_n vai-se tornar básica com valor

$$\bar{\bar{z}}_n = - \frac{1}{m_{nn}} (q_n + M_{nL} \bar{\bar{z}}_L)$$

Como $\bar{\bar{w}}_n > 0$ então $\bar{\bar{z}}_n < b_n$. Seguidamente o algoritmo vai resolver novamente o BLCP(b_L, $q_L + M_{Ln} \bar{\bar{z}}_n$, M_{LL}) e encontra a sua solução única $\tilde{z}_L$. Mas, aplicando o lema 4.2, tem-se $\tilde{z}_L \leq \bar{\bar{z}}_L$, uma vez que $q_L + M_{Ln} b_n \leq q_L + M_{Ln} \bar{\bar{z}}_n$. Além disso, do teorema 4.1, a solução $\tilde{z}_L$ origina para z_n um valor $\tilde{z}_n$ tal que $\tilde{z}_n \leq \bar{\bar{z}}_n < b_n$. Está assim encontrada a solução única do BLCP(b, q, M). Além disso, no decorrer do algoritmo tem-se sempre $H_2 = \varnothing$.

ii) $a_n = 0$

Neste caso o algoritmo começa com uma solução inicial com $z_n = 0$. Pela regra de Bland, o algoritmo resolve primeiro o BLCP(b_L, q_L, M_{LL}). Como esse problema tem dimensão n - 1 então, pela hipótese de indução, encontra a sua solução única $\tilde{z}_L$ num número finito de iterações. A esta solução vai corresponder para w_n o valor

$$\tilde{w}_n = q_n + M_{nL} \tilde{z}_L$$

Se $\tilde{w}_n \geq 0$ a solução $\tilde{z} = (\tilde{z}_L , 0)$ do BLCP foi encontrada. Caso contrário $n \in H_4$. Se $b_n < +\infty$ então deve-se analisar se n deve passar para T_1 ou para F. Para tal calcula-se

$$\tilde{w}_n = q_n + M_{nL} \tilde{z}_L + m_{nn} b_n$$

e podem acontecer dois casos que a seguir se discutem.

a) $\tilde{w}_n \leq 0$

Neste caso faz-se $T_2 = T_2 - \{ n \}$ e $T_1 = T_1 \cup \{ n \}$. Em seguida o algoritmo vai resolver o BLCP(b_L , $q_L + M_{Ln} b_n$, M_{LL}) e encontra a sua solução única $\bar{\bar{z}}_L$. Tal com no caso anterior, por aplicação do lema 4.2 conclui-se ser $\bar{\bar{z}}_L \geq \tilde{z}_L$ e por isso

$$\bar{\bar{w}}_n = q_n + M_{nL} \bar{\bar{z}}_L + m_{nn} b_n \leq \tilde{w}_n \leq 0.$$

Donde $z = (\bar{\bar{z}}_L , b_n)$ é a solução do BLCP(b, q, M)

b) $\tilde{w}_n > 0$

Neste caso faz-se $T_2 = T_2 - \{ n \}$ e $F = F \cup \{ n \}$. Então o novo valor para z_n é

$$\tilde{z}_n = - \frac{1}{m_{nn}} (q_n + M_{nL} \tilde{z}_L) < b_n$$

Seguidamente o algoritmo vai resolver o BLCP(b_L, $q_L + M_{nL} \tilde{z}_L$, M_{LL}) e encontra a sua solução única $\hat{z}_L$. A esta solução vai corresponder para z_n o valor

$$\hat{z}_n = - \frac{1}{m_{nn}} (q_n + M_{nL} \hat{z}_L)$$

e $\hat{z}_n \geq \tilde{z}_n$ pelo teorema 4.1 . Se $\hat{z}_n \leq b_n$, então $z = (\hat{z}_L , \hat{z}_n)$ é a solução do BLCP(b, q, M). Caso contrário $n \in H_1$ e deve-se fazer $F = F - \{ n \}$ e $T_1 = T_1 \cup \{ n \}$. Mais uma vez vai-se resolver um BLCP de dimensão n-1 com termo independente $q_L + M_{Ln} b_n$, sendo o vector $\bar{\bar{z}}_L$ a solução única desse problema. A esta solução vai corresponder para w_n o valor

$$\bar{\bar{w}}_n = q_n + M_{nL} \bar{\bar{z}}_L + m_{nn} b_n$$

Tal como no caso anterior $\bar{\bar{w}}_n < 0$ e por isso foi encontrada a solução $z = (\bar{\bar{z}}_L, b_n)$ do BLCP(b, q, M).

O caso de ser $b_n = + \infty$ é muito semelhante a este com a simplificação de se começar por fazer $T_2 = T_2 - \{ n \}$ e $F = F \cup \{ n \}$ como em b) . Como não é necessário controlar qualquer limite superior para z_n, ($\hat{z}_L$, $\hat{z}_n$) é a solução do BLCP(b, q, M).

Está assim demonstrado que o algoritmo é convergente quando $G = \emptyset$. Para analisar o caso de $G \neq \emptyset$ basta definir o conjunto $\bar{G} = \{ 1, \ldots, n \} - G$ e considerar que o BLCP(b, q, M) é equivalente ao BLCP($b_{\bar{G}}$, p, $\bar{M}_{\bar{G}\bar{G}}$) onde

$$p = q_{\bar{G}} - M_{\bar{G}G} M_{GG}^{-1} q_G \quad \text{e} \quad \bar{M}_{\bar{G}\bar{G}} = M_{\bar{G}\bar{G}} - M_{\bar{G}G} M_{GG}^{-1} M_{G\bar{G}}$$

Como $\bar{M}_{\bar{G}\bar{G}} = (M \setminus M_{GG})$ e $M \in K$ então também $\bar{M}_{\bar{G}\bar{G}}$ é de classe K. Como o algoritmo começa com $F = G$ então pode obter a solução do BLCP neste caso.

□

Provámos assim que se $M \in K$ o algoritmo EXTMURTY1 é convergente e, além disso, há a garantia de ser sempre $H_2 = \emptyset$. Tal facto permite omitir a sua determinação no passo 1 do algoritmo. Por outro lado, como $M_{SF} \leq 0$ e $\bar{M}_{FS} \geq 0$ então

$$\bar{m}_{SS} = m_{SS} + M_{SF}\bar{M}_{FS} \leq m_{SS}$$

Este conhecimento faz com que o cálculo de $\bar{m}_{SS}$ não seja necessário quando $s \in H_3$ ou $s \in H_4$ e b_s finito. Com efeito como

$$w_s - \bar{m}_{SS} b_s \geq w_s - m_{SS} b_s \quad \text{e} \quad w_s + \bar{m}_{SS} b_s \leq w_s + m_{SS} b_s$$

então, não é necessário o cálculo de $\bar{m}_{SS}$ para analisar se um índice pode transitar entre T_1 e T_2. Assim, quando aplicado a matrizes K, o algoritmo passa a ter a seguinte forma mais simplificada.

ALGORITMO EXTMURTY1 (M ∈ K)

PASSO INICIAL: Faça $F = \{ i : a_i = -\infty \text{ e } b_i = +\infty \}$,

$T_1 = \{ i : a_i = -\infty \text{ e } b_i < +\infty \}$,

$T_2 = \{ i : a_i = 0 \}$.

PASSO GERAL: Calcule z_F e w_T a partir de (4.6) e seja $H = H_1 \cup H_2 \cup H_3$,

com H_i (i=1, 3, 4) definido a partir de (4.5).

Se $H = \varnothing$, então
$$\left| \begin{array}{l} z_F = -M_{FF}^{-1}(q_F + M_{FT1} b_{T1}) \\ z_{T1} = b_{T1} \\ z_{T2} = 0 \\ w_F = 0 \\ w_T = q_T + M_{FT1} b_{T1} + M_{TF} z_F \end{array} \right.$$

é solução do BLCP.

Caso contrário, seja $s = \min \{ i \in H \}$. Então

(i) $s \in H_1 \Rightarrow F = F - \{ s \}$ e $T_1 = T_1 \cup \{ s \}$

(ii) $s \in H_3 \Rightarrow T_1 = T_1 - \{ s \}$

$$a_s = -\infty \Rightarrow F = F \cup \{ s \}$$
$$a_s = 0 \Rightarrow \left| \begin{array}{l} w_s - m_{ss} b_s \geq 0 \Rightarrow T_2 = T_2 \cup \{ s \} \\ w_s - m_{ss} b_s < 0 \Rightarrow F = F \cup \{ s \} \end{array} \right.$$

(iii) $s \in H_4 \Rightarrow T_2 = T_2 - \{ s \}$

$$b_s = +\infty \Rightarrow F = F \cup \{ s \}$$
$$b_s < +\infty \Rightarrow \left| \begin{array}{l} w_s + m_{ss} b_s \leq 0 \Rightarrow T_1 = T_1 \cup \{ s \} \\ w_s + m_{ss} b_s > 0 \Rightarrow F = F \cup \{ s \} \end{array} \right.$$

Repita o Passo Geral

4 Segunda Extensão do Método de Murty

Nesta secção iremos apresentar uma segunda extensão do método de Murty [37] para a qual é possível estabelecer convergência desde que a matriz M pertença à classe P. Os passos do processo são apresentados a seguir.

ALGORITMO EXTMURTY 2

PASSO INICIAL: Faça $F = \{ i : a_i = -\infty \text{ e } b_i = +\infty \}$,

$T_1 = \{ i : a_i = -\infty \text{ e } b_i < +\infty \}$,

$T_2 = \{ i : a_i = 0 \}$.

PASSO GERAL: Calcule z_F e w_T a partir de (4.6). Seja $H = \bigcup_{i=1}^{4} H_i$,

com H_i ($i = 1, 2, 3, 4$) definidos por (4.5).

Se $H = \varnothing$, então

$$\begin{aligned} z_F &= -M_{FF}^{-1}(q_F + M_{FT1} b_{T1}) \\ z_{T1} &= b_{T1} \\ z_{T2} &= 0 \\ w_F &= 0 \\ w_T &= q_T + M_{FT1} b_{T1} + M_{TF} z_F \end{aligned}$$

é solução do BLCP. Caso contrário seja $s = \min \{ i \in H \}$.

Então

(i) $s \in H_1 \Rightarrow F = F - \{ s \}$ e $T_1 = T_1 \cup \{ s \}$;

(ii) $s \in H_2 \Rightarrow F = F - \{ s \}$ e $T_2 = T_2 \cup \{ s \}$;

(iii) $s \in H_3 \Rightarrow T_1 = T_1 - \{ s \}$ e $F = F \cup \{ s \}$;

(iv) $s \in H_4 \Rightarrow T_2 = T_2 - \{ s \}$ e $F = F \cup \{ s \}$.

Repita o Passo Geral.

De notar que nos casos (iii) e (iv) não há garantia de se remover a inadmissibilidade. Neste algoritmo o conjunto F é alterado de um elemento em todas as iterações. Assim, ao contrário do algoritmo EXTMURTY1, não são permitidas

passagens directas do limite inferior para o limite superior ou vice-versa. Por outras palavras, é executada uma operação pivotal em todas as iterações.

Para simplificar a demonstração, consideremos que todos os limites inferiores são finitos e iguais a zero. Como vimos anteriormentre, nesse caso o BLCP (4.1) é equivalente ao seguinte LCP:

$$\begin{bmatrix} v_1 \\ \gamma_1 \\ v_2 \\ \gamma_2 \\ \dots \\ v_n \\ \gamma_n \end{bmatrix} = \begin{bmatrix} q_1 \\ b_1 \\ q_2 \\ b_2 \\ \dots \\ q_n \\ b_n \end{bmatrix} + \begin{bmatrix} m_{11} & 1 & m_{12} & 0 & \dots & m_{1n} & 0 \\ -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ m_{21} & 0 & m_{22} & 1 & \dots & m_{2n} & 0 \\ 0 & 0 & -1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ m_{n1} & 0 & m_{n2} & 0 & \dots & m_{nn} & 1 \\ 0 & 0 & 0 & 0 & \dots & -1 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ \lambda_1 \\ z_2 \\ \lambda_2 \\ \dots \\ z_n \\ \lambda_n \end{bmatrix} \qquad (4.7a)$$

$$z_i \ , \ \lambda_i \ , \ \gamma_i \ , \ v_i \geq 0 \quad z_i v_i = \gamma_i \lambda_i = 0 \quad , \quad i = 1 \ , \ \dots \ , n \qquad (4.7b)$$

Se algum $b_i = +\infty$ então a linha e coluna de ordem 2i não existe. Para simplificação da exposição iremos supor que todos os limites superiores são finitos.

Para demonstrar a convergência deste algoritmo para o BLCP com uma matriz P tem de se provar que:

a) O algoritmo de Murty resolve o LCP (4.7)

b) Resolver o LCP (4.7) usando o Método de Murty é equivalente a resolver o BLCP (4.1) com o algoritmo EXTMURTY2.

O LCP (4.7) pode ser escrito na forma habitual:

$$y = p + A\,x \geq 0 \quad , \quad x \geq 0 \quad , \quad x^T y = 0$$

com $A \in R^{2n \times 2n}$, $x \in R^{2n}$, $y \in R^{2n}$ e $p \in R^{2n}$. Além disso para $s = 1 \ , \ 2 \ , \ \dots \ , \ 2n$ tem-se:

$$s = 2r - 1 \quad \Rightarrow \begin{cases} x_s = z_r \\ y_s = v_r \\ p_s = q_r \end{cases} \qquad s = 2r \quad \Rightarrow \begin{cases} x_s = \lambda_r \\ y_s = \gamma_r \\ p_s = b_r \end{cases}$$

Considere-se neste LCP dois conjuntos de índices R e S tais que

$$R = \{ i : x_i \text{ é básica } \} \quad ; \quad S = \{ i : y_i \text{ é básica } \}.$$

Então $R \cap S = \varnothing$ e $R \cup S = \{ 1 , \ldots , 2n \}$. A solução complementar associada a estes conjuntos de índices é:

$$\bar{x}_R = - A_{RR}^{-1} p_R \quad ; \quad \bar{y}_S = p_S + A_{SR} \bar{x}_R \quad ; \quad \bar{x}_S = \bar{y}_R = 0.$$

Para que o algoritmo de Murty funcione é necessário que:

(i) A_{RR} seja não singular

(ii) $\bar{a}_{ss} > 0$, qualquer que seja a inadmissibilidade s.

Seja x_s uma variável básica com s impar. Então $s = 2r - 1$ e a variável z_r é básica. Como $\gamma_r = b_r - z_r$, dois casos podem acontecer, nomeadamente $z_r = b_r$ ou $z_r \neq b_r$. Se $z_r = b_r$ tem-se $\gamma_r = 0$, enquanto que $z_r \neq b_r$ implica $\gamma_r \neq 0$. Como a solução é complementar, então podem acontecer uma das duas seguintes situações:

z_r básica , γ_r não básica , λ_r básica , v_r não básica

ou

z_r básica , γ_r básica , λ_r não básica , v_r não básica

Consideremos agora que a variável x_s é não básica. Então $z_r = 0$ e $\gamma_r = b_r$ (e portanto básica). Portanto só a seguinte situação pode acontecer:

z_r não básica , γ_r básica , λ_r não básica , v_r básica.

Conclui-se assim que quando z_r é não básica, λ_r também o é. Contudo λ_r pode ser ou não básica quando z_r é básica. Podemos então definir os seguintes conjuntos de índices:

$$T = \{ i : z_i \text{ e } \lambda_i \text{ ambas básicas } \} \quad ; \quad U = \{ i : z_i \text{ é básica e } \lambda_i \text{ é não básica } \};$$

$$V = \{ i : z_i \text{ e } \lambda_i \text{ ambas não básicas } \}.$$

É fácil de verificar que os conjuntos R e S podem ser caracterizados à custa dos três conjuntos T , U e V referidos atrás. Com efeito tem-se:

$$R = \{ 2i - 1 : i \in T \} \cup \{ 2i : i \in T \} \cup \{ 2i - 1 : i \in U \}$$

$$S = \{ 2i - 1 : i \in V \} \cup \{ 2i : i \in V \} \cup \{ 2i : i \in U \}$$

Efectuando permutações adequadas de linhas e colunas é possível escrever a matriz A_{RR} na seguinte forma:

$$A_{RR} = \left[\begin{array}{cc|c} M_{TT} & I_T & M_{TU} \\ -I_T & O & O \\ \hline M_{UT} & O & M_{UU} \end{array}\right]$$

sendo I_T a matriz identidade de ordem #T. Para averiguar se esta matriz é não singular calculemos o seu determinante, usando as conhecidas fórmulas de Schur e do quociente [8] aplicadas a uma permutada principal de A_{RR} . Assim tem-se

$$\det (A_{RR}) = \det \left[\begin{array}{cc|c} M_{TT} & I_T & M_{TU} \\ -I_T & O & O \\ \hline M_{UT} & O & M_{UU} \end{array}\right] = \det \left[\begin{array}{cc|c} M_{TT} & M_{TU} & I_T \\ M_{UT} & M_{UU} & O \\ \hline -I_T & O & O \end{array}\right] =$$

$$= \det (M_{T\cup U, T\cup U}) \cdot \det (A_{RR} \mid M_{T\cup U, T\cup U}) =$$

$$= \det (M_{T\cup U, T\cup U}) \cdot \det ((A_{RR} \mid M_{UU}) \mid (M_{T\cup U, T\cup U} \mid M_{UU})) =$$

$$= \det (M_{T\cup U,T\cup U}) \, . \, \det (A_{RR} \mid M_{UU}) / \det (M_{TT} - M_{TU} M_{UU}^{-1} M_{UT})$$

$$= \det (M_{T\cup U,T\cup U}) / \det (M_{TT} - M_{TU} M_{UU}^{-1} M_{UT}),$$

com $M_{T\cup U,T\cup U} = \begin{bmatrix} M_{TT} & M_{TU} \\ M_{UT} & M_{UU} \end{bmatrix}$, uma vez que $\det (A_{RR} \mid M_{UU}) = 1$.

Mas M_{TT} , M_{UU} e $M_{T\cup U,T\cup U}$ são submatrizes principais de M e, por isso, são matrizes P [8]. Por outro lado, $M_{TT} - M_{TU} M_{UU}^{-1} M_{UT}$ é o Complemento de Schur de M_{UU} em $M_{T\cup U,T\cup U}$ e, portanto, também é da classe P [8]. Conclui-se assim que $\det (A_{RR}) > 0$ e a matriz A_{RR} é não singular. Por outro lado, a sua inversa é a matriz:

$$A_{RR}^{-1} = \left[\begin{array}{cc|c} O & - I_T & O \\ I_T & M_{TT} - M_{TU} M_{UU}^{-1} M_{UT} & - M_{TU} M_{UU}^{-1} \\ \hline O & M_{UU}^{-1} M_{UT} & M_{UU}^{-1} \end{array}\right]$$

como facilmente se verifica efectuando o produto $A_{RR} \, A_{RR}^{-1}$. Assim provámos a parte (i).

Seguidamente iremos provar que $\tilde{a}_{ss} > 0$, qualquer que seja o índice s da inadmissibilidade a remover. Dois casos podem acontecer, nomeadamente $s \in R$ ou $s \in S$. Se $s \in R$ então $x_s < 0$. Além disso, x_s é a uma variável z ou a uma variável λ conforme s seja impar ou par. Se $s \in S$ tem-se $y_s < 0$, o que corresponde a uma variável v ou a uma variável γ, conforme s é impar ou par respectivamente. Resumindo ter-se-à uma das quatro situações seguintes:

a) $s \in R$, $s = 2r - 1$, $x_s < 0 \Rightarrow (z_r < 0 \wedge v_r = 0 \wedge \gamma_r > 0 \wedge \lambda_r = 0)$;

b) $s \in R$, $s = 2r$, $x_s < 0 \Rightarrow (z_r > 0 \wedge v_r = 0 \wedge \gamma_r = 0 \wedge \lambda_r < 0)$;

c) $s \in S$, $s = 2r - 1$, $y_s < 0 \Rightarrow (z_r = 0 \wedge v_r < 0 \wedge \gamma_r > 0 \wedge \lambda_r = 0)$;

d) $s \in S$, $s = 2r$, $y_s < 0 \Rightarrow (z_r > 0 \wedge v_r = 0 \wedge \gamma_r < 0 \wedge \lambda_r = 0)$.

No caso a) z_r é básica e λ_r é não básica. Então $r \in U$ e portanto $\bar{a}_{ss}$ é um elemento diagonal da matriz M_{UU}^{-1}. Mas se $M \in P$, então $M_{UU} \in P$ e também $M_{UU}^{-1} \in P$ [8]. Portanto $\bar{a}_{ss} > 0$.

No caso b) as variáveis z_r e λ_r são ambas básicas e, por isso, $r \in T$. Logo $\bar{a}_{ss}$ é um elemento da diagonal de $M_{TT} - M_{TU} M_{UU}^{-1} M_{UT}$. Mas essa matriz é P, pois é o Complemento de Schur de M_{UU} em $M_{T\cup U, T\cup U}$. Portanto $\bar{a}_{ss} > 0$ também neste caso.

No caso c) as variáveis z_r e λ_r são ambas não básicas. Se $s = 2r - 1$, então, quando se efectuar a operação pivotal, λ_r vai-se tornar básica por troca com γ_r. Assim, o índice r vai passar a pertencer ao conjunto U. Se $\bar{U} = U \cup \{ r \}$, então, usando mais uma vez a fórmula de Schur, vem:

$$\det \begin{bmatrix} A_{RR} & A_{RS} \\ A_{SR} & a_{SS} \end{bmatrix} = \det (A_{RR}) \det (a_{SS} - A_{SR} A_{RR}^{-1} A_{RS}) = \det (A_{RR}) \bar{a}_{SS}.$$

Donde

$$\bar{a}_{ss} = \frac{\det \begin{bmatrix} A_{RR} & A_{RS} \\ A_{SR} & a_{SS} \end{bmatrix}}{\det(A_{RR})} = \frac{\det \left[\begin{array}{cc|c} M_{TT} & M_{T\bar{U}} & I_T \\ M_{\bar{U}T} & M_{\bar{U}\bar{U}} & O \\ \hline - I_T & O & O \end{array}\right]}{\det(A_{RR})} =$$

$$= \frac{\det (M_{T\cup\bar{U}, T\cup\bar{U}})}{\det (M_{TT} - M_{T\bar{U}} M_{\bar{U}\bar{U}}^{-1} M_{\bar{U}T}) \det (A_{RR})} > 0.$$

Finalmente no caso d), como $s = 2r$, γ_r é a variável que se vai tornar não básica por troca com λ_r. Além disso a variável z_r era básica e continua sendo básica, pelo que o índice r vai ser transferido do conjunto U para o conjunto T. Assim os dois novos conjuntos de índices $\bar{T}$ e $\bar{U}$ satisfazem

$$\bar{T} = T \cup \{ r \} \quad \bar{U} = U - \{ r \}$$

Então

$$\bar{a}_{ss} = \frac{\det \begin{bmatrix} A_{RR} & A_{RS} \\ A_{SR} & a_{ss} \end{bmatrix}}{\det(A_{RR})} = \frac{\det \left[\begin{array}{cc|c} M_{\bar{T}\bar{T}} & M_{\bar{T}\bar{U}} & I_{\bar{T}} \\ M_{\bar{U}\bar{T}} & M_{\bar{U}\bar{U}} & O \\ \hline - I_{\bar{T}} & O & O \end{array}\right]}{\det(A_{RR})} =$$

$$= \frac{\det(M_{\bar{T}\cup\bar{U},\bar{T}\cup\bar{U}})}{\det(M_{\bar{T}\bar{T}} - M_{\bar{T}\bar{U}} M_{\bar{U}\bar{U}}^{-1} M_{\bar{U}\bar{T}}) \det(A_{RR})} > 0$$

por razões semelhantes ao caso anterior.

Está pois provada a aplicabilidade do método de Murty ao LCP (4.7). Note-se que a matriz A desse LCP não é P e, por isso, é necessário mostrar que o método de Murty conduz à solução num número finito de iterações. Essa prova é feita por indução sobre a ordem da matriz M.

Teorema 4.2 : O método pivotal principal de Murty converge para a solução única do LCP (4.7) num número finito de iterações.

Demonstração: Se a matriz M tem dimensão 1 o LCP (4.7) tem a forma:

$$\begin{bmatrix} v_1 \\ \gamma_1 \end{bmatrix} = \begin{bmatrix} q_1 \\ b_1 \end{bmatrix} + \begin{bmatrix} m_{11} & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ \lambda_1 \end{bmatrix} \qquad (4.8)$$

$$z_1 , \lambda_1 , v_1 , \gamma_1 \geq 0 \quad , \quad z_1 \lambda_1 = v_1 \gamma_1 = 0$$

O algoritmo começa por fazer $z_1 = \lambda_1 = 0$, $v_1 = q_1$, $\gamma_1 = b_1$. Se $q_1 \geq 0$ obteve-se a solução do LCP (4.8) . Caso contrário efectua-se uma operação pivotal com pivot $m_{11} > 0$ e obtém-se

$$\begin{bmatrix} z_1 \\ \gamma_1 \end{bmatrix} = \begin{bmatrix} \frac{-q_1}{m_{11}} \\ b_1 + \frac{q_1}{m_{11}} \end{bmatrix} + \begin{bmatrix} \frac{1}{m_{11}} & \frac{-1}{m_{11}} \\ \frac{-1}{m_{11}} & \frac{1}{m_{11}} \end{bmatrix} \begin{bmatrix} v_1 \\ \lambda_1 \end{bmatrix}$$

Como $q_1 < 0$ e $m_{11} > 0$ então $z_1 > 0$. Se $b_1 + \frac{q_1}{m_{11}} > 0$, então a solução foi encontrada. Caso contrário efectua-se uma operação pivotal com pivot $\frac{1}{m_{11}}$ e tem-se:

$$\begin{bmatrix} z_1 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} b_1 \\ -m_{11}b_1 - q_1 \end{bmatrix} + \begin{bmatrix} 0 & -1 \\ 1 & m_{11} \end{bmatrix} \begin{bmatrix} v_1 \\ \gamma_1 \end{bmatrix}$$

Como $m_{11}b_1 + q_1 < 0$, $\lambda_1 > 0$ e $z_1 = b_1 > 0$, então a solução foi encontrada. Provámos assim que o método de Murty é capaz de resolver o LCP (4.8).

Suponha-se agora, como hipótese de indução, que o método de Murty é capaz de resolver todo o LCP da forma (4.7) de dimensão 2(n - 1), (isto é um LCP dessa forma correspondendo a uma matriz M de dimensão n - 1), e sejam $\hat{z}$, $\hat{v}$, $\hat{\lambda}$, $\hat{\gamma}$ os valores únicos da solução de um tal LCP, em que a matriz M_{n-1} é obtida de uma matriz M de dimensão n suprimindo as suas última linha e coluna.

Na solução do LCP de dimensão 2n há três casos a considerar:

(i) $\hat{z}_n = \hat{\lambda}_n = 0$

(ii) $\hat{z}_n > 0$ e $\hat{\lambda}_n = 0$

(iii) $\hat{z}_n > 0$ e $\hat{\lambda}_n > 0$

É de notar que a hipótese $\hat{z}_n = 0$ e $\hat{\lambda}_n > 0$ é impossível uma vez que

$$\hat{z}_n = 0 \Rightarrow \hat{\gamma}_n = b_n \Rightarrow \hat{\lambda}_n = 0$$

Seguidamente iremos tratar cada um destes casos.

(i) Pela regra de Bland o algoritmo resolve o LCP de ordem 2(**n** - 1) e encontra a sua solução única num número finito de iterações. Como $\hat{z}_n = \hat{\lambda}_n = 0$ basta acrescentar estas duas componentes aos vectores já determinados, uma vez que o cálculo de $\hat{v}_n$ e $\hat{\gamma}_n$ tem que fornecer valores não negativos.

(ii) O algoritmo resolve o LCP de ordem 2(n - 1) e encontra a solução $\bar{z}$, $\bar{v}$, $\bar{\lambda}$, $\bar{\gamma}$. Se os vectores $z = (\bar{z}_1 , \bar{z}_2 , \ldots , \bar{z}_{n-1} , 0)$ e $\lambda = (\bar{\lambda}_1 , \bar{\lambda}_2 , \ldots , \bar{\lambda}_{n-1} , 0)$ fossem solução do problema de dimensão 2n teria que ser $\bar{v}_n \geq 0$ $\bar{\gamma}_n \geq 0$. Mas como $\hat{z}_n > 0$ então $\bar{v}_n < 0$. Portanto efectua-se uma operação pivotal que troca v_n com z_n . Após essa operação há que resolver o LCP de dimensão 2(n - 1) com matriz:

$$B = \begin{bmatrix} (M \mid m_{nn}) & I_{n-1} \\ - I_{n-1} & O \end{bmatrix}$$

Como $B \in P$, pela hipótese de indução, o método de Murty encontra a solução

$$(\bar{\bar{z}}_1 , \bar{\bar{z}}_2 , \ldots , \bar{\bar{z}}_{n-1}) \text{ e } (\bar{\bar{\lambda}}_1 , \bar{\bar{\lambda}}_2 , \ldots , \bar{\bar{\lambda}}_{n-1})$$

num número finito de iterações. Estes valores vão conduzir a $\bar{\bar{z}}_n \geq 0$ e a $\bar{\bar{\gamma}}_n \geq 0$ e, assim, obtém-se a solução

$$z = (\bar{\bar{z}}_1 , \bar{\bar{z}}_2 , \ldots , \bar{\bar{z}}_{n-1} , \bar{\bar{z}}_n) \text{ e } \lambda = (\bar{\bar{\lambda}}_1 , \bar{\bar{\lambda}}_2 , \ldots , \bar{\bar{\lambda}}_{n-1} , 0).$$

(iii) Começa-se, como no caso anterior, por resolver o LCP de ordem 2(n - 1) e obtém-se a solução $(\bar{z}_1 , \bar{z}_2 , \ldots , \bar{z}_{n-1})$, $(\bar{\lambda}_1 , \bar{\lambda}_2 , \ldots , \bar{\lambda}_{n-1})$. Além disso $\bar{v}_n < 0$. Efectuada a operação pivotal obtém-se um novo LCP de dimensão 2(n - 1), que, como no caso anterior, se vai resolver obtendo-se a solução

$$(\bar{\bar{z}}_1 , \bar{\bar{z}}_2 , \ldots , \bar{\bar{z}}_{n-1}) \text{ e } (\bar{\bar{\lambda}}_1 , \bar{\bar{\lambda}}_2 , \ldots , \bar{\bar{\lambda}}_{n-1}).$$

Esta solução deve conduzir a $\bar{\bar{z}}_n > 0$ e $\bar{\bar{\gamma}}_n < 0$. Então, nova operação pivotal deve ser efectuada que troca γ_n com λ_n. Agora tem que se resolver um LCP de ordem 2(n -1) cuja matriz é

$$\left(A \left| \begin{bmatrix} m_{nn} & 1 \\ -1 & 0 \end{bmatrix} \right. \right)$$

Pela fórmula do quociente [13] este Complemento de Schur pode ser escrito na forma

$$\left(A \left| \begin{bmatrix} m_{nn} & 1 \\ -1 & 0 \end{bmatrix} \right. \right) = \left((A \mid m_{nn}) \left| \frac{1}{m_{nn}} \right. \right) = \begin{bmatrix} M_{n-1} & I_{n-1} \\ -I_{n-1} & O \end{bmatrix} \in P$$

Então, o método de Murty encontra a solução deste último LCP que tem obrigatoriamente que conduzir a

$$\hat{z}_n > 0 \quad e \quad \hat{\lambda}_n > 0$$

pois a solução de um BLCP é única.

Está pois demonstrado que o método de Murty é capaz de resolver o LCP (4.7) de dimensão 2n num número finito de iterações.

□

Para estabelecer a convergência do algoritmo EXTMURTY2 há que provar a equivalência entre esse método e o método de Murty aplicado ao LCP (4.7). Para isso começamos por provar que os conjuntos T , U e V do método de Murty satisfazem $T \equiv T_1$, $U \equiv F$ e $V \equiv T_2$, onde F, T_1 e T_2 são os conjuntos de índices introduzidos no algoritmo EXTMURTY2. Com efeito tem-se:

(i) Se $i \in T$, z_i e λ_i são ambas básicas e, como γ_i é não básica será $z_i = b_i$ e, por isso, $i \in T_1$;

(ii) Se $i \in U$, z_i é básica e λ_i é não básica e, por isso, γ_i é básica, o que implica que seja $z_i \neq b_i$ e $i \in F$;

(iii) Se $i \in V$, z_i e λ_i são ambas não básicas o que implica que $z_i = 0$ e $i \in T_2$.

Reciprocamente:

(i) Se $i \in T_1$, $z_i = b_i$ e z_i básica. Como $\gamma_i = b_i - z_i$ então $\gamma_i = 0$ e, por isso, λ_i é básica o que faz com que $i \in T$;

(ii) Se $i \in F$, z_i é básica e diferente de b_i, y_i também é básica e λ_i é não básica, ou seja $i \in U$;

(iii) Se $i \in T_2$, $z_i = 0$, o que implica que $\gamma_i = b_i$. Donde λ_i é não básica e por isso $i \in V$.

Vejamos agora o que acontece em cada iteração de cada um dos métodos, para cada um dos quatro casos de escolha da inadmissibilidade.

a) $s \in R$ e $s = 2r - 1 \Rightarrow x_s < 0 \Rightarrow (z_r < 0 \wedge v_r = 0 \wedge \gamma_r > 0 \wedge \lambda_r = 0)$

Então z_r é básica e λ_r é não básica, ou seja, $r \in U$. A operação pivotal vai tornar z_r não básica mantendo λ_r não básica, ou seja, o índice r vai passar de U para V. Donde

$$U = U - \{ r \} \; ; \; V = V \cup \{ r \}$$

o que, tendo em conta a equivalência entre os conjuntos de índices, é o mesmo que

$$F = F - \{ r \} \; ; \; T_2 = T_2 \cup \{ r \}$$

Isso é exactamente o que acontece no método EXTMURTY2 quando z_r é básica e tem valor negativo.

b) $s \in R$ e $s = 2r \Rightarrow x_s < 0 \Rightarrow (z_r > 0 \wedge v_r = 0 \wedge \gamma_r = 0 \wedge \lambda_r < 0)$

Então z_r e λ_r são ambas básicas, ou seja, $r \in T$. A operação pivotal vai trocar λ_r com γ_r , (ficando $\lambda_r = 0$) e mantém z_r básica. Portanto o índice r vai passar de T para U, ou seja

$$T = T - \{ r \} \ , \ U = U \cup \{ r \}.$$

Então, no método EXTMURTY 2,

$$T_1 = T_1 - \{ r \} \ , \ F = F \cup \{ r \}.$$

Esta situação corresponde a $z_r = b_r$ com $w_r > 0$. Com efeito, como $w_r = v_r - \lambda_r$, então $\lambda_r < 0$ e $v_r = 0$ implica que $w_r > 0$.

$$c)\ s \in S \ \text{ e } \ s = 2r - 1 \Rightarrow y_s < 0 \Rightarrow (z_r = 0 \ \wedge \ v_r < 0 \ \wedge \ \gamma_r = b_r \ \wedge \ \lambda_r = 0)$$

Então, z_r e λ_r são ambas não básicas, ou seja $r \in V$. A operação pivotal vai trocar v_r com z_r e, por isso, r vai passar para U. Então

$$V = V - \{ r \} \ , \ U = U \cup \{ r \}$$

o que em relação ao método EXTMURTY2 significa

$$T_2 = T_2 - \{ r \} \ , \ F = F \cup \{ r \}.$$

Não é difícil de ver que esta situação corresponde a $z_r = 0$ e $w_r < 0$, pois $w_r = v_r - \lambda_r$ com $v_r < 0$ e $\lambda_r = 0$ implica $w_r < 0$. Por isso, mais uma vez, o método de Murty procede de modo equivalente ao algoritmo EXTMURTY2.

$$d)\ s \in S \ \text{ e } \ s = 2r \Rightarrow y_s < 0 \Rightarrow (z_r > b_r \ \wedge \ v_r = 0 \ \wedge \ \gamma_r < 0 \ \wedge \ \lambda_r = 0)$$

Como z_r é básica e λ_r é não básica, r é um elemento de U. A operação pivotal vai anular γ_r tornando λ_r básica, ou seja, r vai passar de U para T. Donde

$$U = U - \{ r \} \ , \ T = T \cup \{ r \}.$$

Em relação ao algoritmo EXTMURTY2 tem-se

$$F = F - \{ r \} \ , \ T_1 = T_1 \cup \{ r \}.$$

Mas tornar $\gamma_r = 0$ significa fazer $z_r = b_r$, conforme a iteração do método EXTMURTY2.

Está assim demonstrada a equivalência entre os dois algoritmos e, portanto, a convergência do algoritmo EXTMURTY2 para qualquer tipo de matriz da classe P.

A demonstração da convergência do algoritmo para o caso da existência de alguns limites inferiores infinitos é feita usando argumentos semelhantes aos utilizados na convergência do algoritmo EXTMURTY1.

5 MÉTODO DE KELLER

Como foi referido no segundo capítulo, o método de Keller [48] foi desenvolvido inicialmente para obter um ponto estacionário de um programa quadrático com relações de desigualdade. Seguidamente iremos mostrar que o algoritmo se simplifica no caso do BLCP. Tal como no caso do LCP, o algoritmo procura remover uma inadmissibilidade por iteração, mas mantendo sempre as variáveis z_i básicas dentro dos seus limites inferiores e superiores. Deste modo, sabe-se que, em todas as iterações, o conjunto das inadmissibilidades H conterá só índices correspondentes a variáveis w básicas, ou seja, será formado pela reunião dos dois conjuntos:

$$H_3 = \{ i \in T_1 : w_i > 0 \} \qquad H_4 = \{ i \in T_2 : w_i < 0 \} \tag{4.9}$$

Em cada iteração escolhe-se a inadmissibilidade s de H tal que

$$| w_s | = \max \{ | w_i | : i \in H \} \tag{4.10}$$

Dois casos podem acontecer e são discutidos a seguir.

i) $s \in T_1 \quad (w_s > 0)$

ii) $s \in T_2 \quad (w_s < 0)$

i) Como $z_s = b_s$, o seu valor só poderá decrescer para 0 ou para um valor positivo menor do que b_s. Se não houvesse qualquer restrição far-se-ia uma operação pivotal principal para tornar z_s básica por troca com a sua complementar que se tornaria nula. Como $w_s > 0$, então o novo valor para z_s satisfaz

$$b_s - \frac{w_s}{\bar{m}_{ss}} < b_s$$

Mas, há que garantir que a variável z_s se mantenha não negativa e, por isso, esta operação só pode ser feita se for $\frac{w_s}{\bar{m}_{ss}} < b_s$. Caso contrário deve-se anular a variável z_s e w_s terá valor igual a $w_s - \bar{m}_{ss}\, b_s > 0$. Além disso tem de se obrigar as restantes variáveis z_i básicas a permanecer dentro dos seus limites inferiores e superiores. Seja λ ($0 \le \lambda \le b_s$) o novo valor da variável z_s. Então, os novos valores dessas variáveis são:

$$\hat{z}_i = z_i + \bar{m}_{is} (\lambda - b_s) , \; i \in F$$

Mas $\lambda - b_s < 0$ e $0 < z_i < b_i$ e, por isso, basta exigir que

$$\lambda \ge b_s - \frac{z_i}{\bar{m}_{is}} \quad \text{para} \quad \bar{m}_{is} > 0 \qquad \text{e} \qquad \lambda \ge b_s - \frac{z_i - b_i}{\bar{m}_{is}} \quad \text{para} \quad \bar{m}_{is} < 0$$

Definindo as quantidades θ_1 , θ_2 e θ_3 do seguinte modo:

$$\theta_1 = \frac{w_s}{\bar{m}_{ss}}$$

$$\theta_2 = \frac{z_r}{\bar{m}_{rs}} = \min \{ \frac{z_i}{\bar{m}_{is}} , \; \bar{m}_{is} > 0 , \; i \in F \}$$

$$\theta_3 = \frac{z_t - b_t}{\bar{m}_{ts}} = \min \{ \frac{z_i - b_i}{\bar{m}_{is}} , \; \bar{m}_{is} < 0 , \; i \in F \}$$

o novo valor da variável z_s vem dado por $\hat{z}_s = b_s - \theta$ com $\theta = \min \{ \theta_1 , \theta_2 , \theta_3 , b_s \}$,

Seguidamente são analisados estes quatro casos

a) $\theta = b_s$

Neste caso faz-se $T_1 = T_1 - \{ s \}$ e $T_2 = T_2 \cup \{ s \}$. A variável complementar w_s continua com valor positivo, pois

$$\hat{w}_s = w_s - \bar{m}_{ss} b_s$$

e $b_s < \theta_1$. Além disso, as restantes variáveis z_i básicas permanecem dentro dos limites.

Os valores das variáveis serão actualizadas do seguinte modo:

$$\hat{z}_s = 0$$
$$\hat{w}_s = w_s - \bar{m}_{ss} b_s$$
$$\hat{z}_i = z_i - \bar{m}_{is} b_s \quad , \quad i \in F$$
$$\hat{w}_T = w_T - (M_{TS} + M_{TF} \bar{M}_{FS}) b_s$$

b) $\theta = \theta_1$

Neste caso faz-se $T_1 = T_1 - \{ s \}$ e $F = F \cup \{ s \}$. Como $\theta_1 < b_s$ a variável z_s ficará com um valor entre os limites e as restantes variáveis z_i mantêm-se básicas. Os valores das variáveis serão actualizados por

$$\hat{z}_s = b_s - \theta_1$$
$$\hat{z}_i = z_i - \bar{m}_{is} \theta_1 \quad , \quad i \in F$$
$$\hat{w}_T = w_T - (M_{TS} + M_{TF} \bar{M}_{FS}) \theta_1$$

c) $\theta = \theta_2$

Então o decréscimo da variável z_s é limitado por z_r que se iria tornar negativa. Portanto, antes de tentar tornar básica a variável z_s, é necessário anular z_r. Assim, faz-se $F = F - \{ r \}$ e $T_2 = T_2 \cup \{ r \}$, o que corresponde a atribuir o valor θ_2 a z_s. Na iteração seguinte só vai ser necessário conhecer os valores das variáveis z_i básicas e da variável w_s, pois vai-se repetir a tentativa de tornar z_s básica. Os valores necessários são dados por

$$\hat{z}_s = b_s - \theta_2$$
$$\hat{w}_s = w_s - \bar{m}_{ss}\,\theta_2$$
$$\hat{z}_i = z_i - \bar{m}_{is}\,\theta_2 \quad , \quad i \in F$$

d) $\theta = \theta_3$

Neste caso o decréscimo da variável z_s é limitado pela variável z_t que iria ultrapassar o limite superior. Então, antes de tentar tornar básica a variável z_s, é necessário atribuir a z_t o valor b_t tornando-a não básica. Assim, faz-se $F = F - \{ t \}$ e $T_1 = T_1 \cup \{ t \}$, o que corresponde a atribuir a z_s o valor θ_3. Tal como no caso anterior, na iteração seguinte só vai ser necessário conhecer os valores das variáveis z_i básicas e da variável w_s, pois vai-se repetir a tentativa de tornar z_s básica. Os valores necessários são dados por

$$\hat{z}_s = b_s - \theta_3$$
$$\hat{w}_s = w_s - \bar{m}_{ss}\,\theta_3$$
$$\hat{z}_i = z_i - \bar{m}_{is}\,\theta_3 \quad , \quad i \in F$$

De notar que, em ambos os casos c) e d), a variável z_s vai ter valor igual a θ no início da iteração seguinte. A determinação do novo valor de θ deve, por isso, ser alterada para $\theta = \min \{ \theta_1 , \theta_2 , \theta_3 , b_s - z_s \}$ e o novo valor da variável z_s será

$\hat{z}_s = z_s - \theta$. A actualização das restantes variáveis deve ser feita tendo em conta que θ é o decréscimo da variável z_s .

ii) Como $z_s = 0$, o seu valor só poderá crescer para b_s ou para outro valor menor que b_s. Se não houvesse qualquer restrição far-se-ia uma operação pivotal principal para tornar z_s básica por troca com a sua complementar, que se tornaria nula. Para tal o novo valor para z_s será

$$- \frac{w_s}{\bar{m}_{ss}} > 0$$

pois $w_s > 0$. Mas, há que garantir que a variável z_s seja inferior ao limite superior b_s e, por isso, esta operação só pode ser feita se $- \frac{w_s}{\bar{m}_{ss}} < b_s$. Caso contrário, deve-se atribuir a z_s o valor do limite superior, ficando w_s com o valor $w_s + \bar{m}_{ss}\, b_s < 0$. Além disso, há que obrigar as restantes variáveis z_i básicas a permanecer dentro dos seus limites inferiores e superiores. Seja λ ($0 \le \lambda \le b_s$) o novo valor da variável z_s. Então:

$$\hat{z}_i = z_i + \bar{m}_{is}\, \lambda \,, \quad i \in F$$

e λ tem de verificar

$$\lambda \le - \frac{z_i}{\bar{m}_{is}} \quad \text{para} \quad \bar{m}_{is} < 0 \qquad \text{e} \qquad \lambda \le - \frac{z_i - b_i}{\bar{m}_{is}} \quad \text{para} \quad \bar{m}_{is} > 0$$

Definindo as quantidades θ_1 , θ_2 e θ_3 do seguinte modo:

$$\theta_1 = - \frac{w_s}{\bar{m}_{ss}}$$

$$\theta_2 = - \frac{z_r}{\bar{m}_{rs}} = \min \{ - \frac{z_i}{\bar{m}_{is}} \,, \quad \bar{m}_{is} < 0 \,, \; i \in F \}$$

$$\theta_3 = - \frac{z_t - b_t}{\bar{m}_{ts}} = \min \{ - \frac{z_i - b_i}{\bar{m}_{is}} \ , \quad \bar{m}_{is} > 0 \ , \ i \in F \}$$

o novo valor da variável z_s será $\hat{z}_s = \theta$ com $\theta = \min \{ \theta_1 , \theta_2 , \theta_3 , b_s \}$. Tal como anteriormente, analisemos separadamente os quatro casos.

a) $\theta = b_s$

Neste caso faz-se $T_2 = T_2 - \{ s \}$ e $T_1 = T_1 \cup \{ s \}$. O valor da variável complementar w_s vem dado por

$$\hat{w}_s = w_s + \bar{m}_{ss} b_s$$

Como $b_s < \theta_1$ então w_s mantém-se negativa. Além disso as restantes variáveis z_i básicas permanecem dentro dos limites. Os valores das variáveis serão actualizados por

$$\hat{z}_s = b_s$$
$$\hat{w}_s = w_s + \bar{m}_{ss} b_s$$
$$\hat{z}_i = z_i + \bar{m}_{is} b_s \quad , \quad i \in F$$
$$\hat{w}_T = w_T + (M_{Ts} + M_{TF} \bar{M}_{FS}) b_s$$

b) $\theta = \theta_1$

Neste caso faz-se $T_2 = T_2 - \{ s \}$ e $F = F \cup \{ s \}$. Com efeito, como $\theta_1 < b_s$, a variável z_s ficará com um valor entre os limites e o mesmo acontece às restantes variáveis z_i básicas. Os valores das variáveis serão dados por:

$$\hat{z}_s = \theta_1$$
$$\hat{z}_i = z_i + \bar{m}_{is} \theta_1 \quad , \quad i \in F$$
$$\hat{w}_T = w_T + (M_{Ts} + M_{TF} \bar{M}_{FS}) \theta_1$$

c) $\theta = \theta_2$

Neste caso o acréscimo da variável z_s é limitado pela variável z_r que ficaria negativa. Então é necessário anular z_r antes de tentar tornar básica a variável z_s. Assim, faz-se $F = F - \{ r \}$ e $T_2 = T_2 \cup \{ r \}$, o que corresponde a atribuir a z_s o valor θ_2. Na iteração seguinte só vai ser necessário conhecer os valores das variáveis z_i básicas e da variável w_s, pois vai-se repetir a tentativa de tornar z_s básica. Os valores necessários são dados por

$$\hat{z}_s = \theta_2$$
$$\hat{w}_s = w_s + \bar{m}_{ss}\, \theta_2$$
$$\hat{z}_i = z_i + \bar{m}_{is}\, \theta_2 \quad , \quad i \in F$$

d) $\theta = \theta_3$

Neste caso o acréscimo da variável z_s é limitado pela variável z_t que irá ultrapassar o limite superior. Então, antes de tentar tornar básica a variável z_s, é necessário atribuir a z_t o valor b_t tornando-a não básica. Faz-se assim $F = F - \{ t \}$ e $T_1 = T_1 \cup \{ t \}$, o que corresponde a atribuir a z_s o valor θ_3. Tal como no caso anterior, na iteração seguinte só vai ser necessário conhecer os valores das variáveis z básicas e da variável w_s, pois vai-se repetir a tentativa de tornar z_s básica. Os valores necessários são dados por

$$\hat{z}_s = \theta_3$$
$$\hat{w}_s = w_s + \bar{m}_{ss}\, \theta_3$$
$$\hat{z}_i = z_i + \bar{m}_{is}\, \theta_3 \quad , \quad i \in F$$

De notar que, em ambos os casos c) e d), no início da iteração seguinte, a variável z_s vai tomar o valor θ. A determinação do novo valor de θ deve, por isso, ser alterada para $\theta = \min \{ \theta_1 , \theta_2 , \theta_3 , b_s - z_s \}$ e o novo valor da variável z_s será

$\hat{z}_s = z_s + \theta$. A actualização das restantes variáveis deve ser feita tendo em conta que θ é o acréscimo da variável z_s .

Se no BLCP houver variáveis cujo limite superior seja $+\infty$ ou cujo limite inferior seja $-\infty$ ou ambos simultaneamente $+$ e $-\infty$, são poucas as alterações a introduzir no algoritmo que se acabou de descrever. O principal cuidado a ter reside na determinação de θ_2 e de θ_3, pois só é necessário analisar se a variável z_i, $i \in F$, permanece dentro dos seus limites se estes forem finitos.

O método de Keller também pode ser utilizado para resolver BLCPs em que a matriz M seja simétrica PSD singular. Neste caso, há que prever a eventualidade de se ter $\bar{m}_{ss} = 0$ Se tal acontecer, virá $\theta_1 = +\infty$ e, mesmo que z_s possa crescer ($s \in T_2$) ou decrescer ($s \in T_1$) sem que isso vá fazer com que as restantes variáveis z básicas ultrapassem os seus limites, a variável z_s não pode ser tornada básica, pois obter-se-ia uma matriz singular. Se, além disso, for $b_s = +\infty$ então o problema não terá solução, pois não é possivel remover a inadmissibilidade de ordem s. A definição dos valores de θ_1 , θ_2 e θ_3 deve ser alterada para comtemplar a possibilidade de ser $\bar{m}_{ss} = 0$ e a existência de limites infinitos.

Para simplificar a escrita das fórmulas considere-se a variável auxiliar ε assim definida:

$$\varepsilon = \begin{cases} +1 & \text{se } s \in T_1 \\ -1 & \text{se } s \in T_2 \end{cases}$$

e os conjuntos de índices A e B :

$$A = \{ i \in F : a_i = 0 \text{ e } \varepsilon \bar{m}_{is} > 0 \} \quad ; \quad B = \{ i \in F : b_i < +\infty \text{ e } \varepsilon \bar{m}_{is} < 0 \}.$$

Então, definam-se os seguintes valores

$$\theta_1 = \begin{cases} \varepsilon \dfrac{w_s}{\bar{m}_{ss}} & \text{se } \bar{m}_{ss} > 0 \\ +\infty & \text{se } \bar{m}_{ss} = 0 \end{cases} \qquad (4.11a)$$

$$\theta_2 = \begin{cases} \dfrac{z_r}{\varepsilon \bar{m}_{rs}} = \min \left\{ \dfrac{z_i}{\varepsilon \bar{m}_{is}} : i \in A \right\} \\ +\infty \quad \text{se} \quad A = \varnothing \end{cases} \qquad (4.11b)$$

$$\theta_3 = \begin{cases} \dfrac{z_t - b_t}{\varepsilon \bar{m}_{ts}} = \min \left\{ \dfrac{z_i - b_i}{\varepsilon \bar{m}_{is}} : i \in B \right\} \\ +\infty \quad \text{se} \quad B = \varnothing \end{cases} \qquad (4.11c)$$

e seja θ como anteriormente.

As fórmulas de actualização também podem ser condensadas usando a variável ε do seguinte modo

$$\hat{z}_s = z_s - \varepsilon\, \theta \qquad (4.12a)$$

$$\hat{w}_s = w_s - \varepsilon\, \bar{m}_{ss}\, \theta \qquad (4.12b)$$

$$\hat{z}_F = z_F - \varepsilon\, \bar{m}_{Fs} \theta \qquad (4.12c)$$

$$\hat{w}_T = w_T - \varepsilon\, (M_{Ts} + M_{TF} \bar{M}_{FS})\, \theta \qquad (4.12d)$$

Se $\theta = \theta_2$ ou $\theta = \theta_3$ não é necessário calcular $\hat{w}_T$. Quando $\theta = \theta_1$ ou $\theta = b_s$ o calculo de $\hat{w}_T$ só pode ser feito a partir de (4.12d) se na iteração anterior não foi $\theta = \theta_2$ ou $\theta = \theta_3$. Caso contrário, deve-se utilizar:

$$\hat{w}_T = q_T + M_{TT_1}\, b_{T_1} + M_{TF}\, \hat{z}_F \qquad (4.12e)$$

O algoritmo de Keller pode então ser resumido na forma seguinte.

MÉTODO DE KELLER PARA M PSD SIMÉTRICA

PASSO INICIAL: Faça $F = \{ i : a_i = -\infty \text{ e } b_i = +\infty \}$,

$T_1 = \{ i : a_i = -\infty \text{ e } b_i < +\infty \}$,

$T_2 = \{ i : a_i = 0 \}$.

Calcule z_F e w_T usando (4.6).

PASSO 1: Defina H_3 e H_4 a partir de (4.9) e determine s por (4.10).

Se $H = \varnothing$, então
$$\left| \begin{array}{l} z_F = -M_{FF}^{-1}(q_F + M_{FT1} b_{T1}) \\ z_{T1} = b_{T1} \\ z_{T2} = 0 \\ w_F = 0 \\ w_T = q_T + M_{FT1} b_{T1} + M_{TF} z_F \end{array} \right.$$

é solução do BLCP. Caso contrário, vá para Passo 2.

PASSO 2: Calcule θ_1 , θ_2 e θ_3 a partir de (4.11) e seja

$\theta = \min \{ b_s - z_s, \theta_1 , \theta_2 , \theta_3 \}$

a) Se $\theta = +\infty$ o BLCP não tem solução,

b) Se $\theta = b_s$ o índice s passa de T_1 para T_2 ou de T_2 para T_1. Calcule z_F e w_T usando (4.12a, b, c) e (4.12d) se é a primeira vez que executa o passo 2 com s ou (4.12e) no caso contrário.

Volte ao Passo 1.

c) Se $\theta = \theta_1$ o índice s passa de T_2 ou de T_1 para F. Calcule z_F e w_T usando (4.12a, c) e (4.12d) se é a primeira vez que executa o passo 2 com s ou (4.12e) no caso contrário.

Volte ao Passo 1.

d) Se $\theta = \theta_2$ faça $F = F - \{ r \}$ e $T_2 = T_2 \cup \{ r \}$. Calcule z_F e w_s usando (4.12a , b , c).

Repita o Passo 2.

e) Se $\theta = \theta_3$ faça $F = F - \{ t \}$ e $T_1 = T_1 \cup \{ t \}$. Calcule z_F e w_s usando (4.12a , b , c).

Repita o Passo 2.

Se a matriz é simétrica P (PD) então o método termina sempre com a solução do BLCP. O algoritmo pode também ser utilizado para matrizes não simétricas P. Contudo a convergência do processo num número finito de iterações ainda não foi estabelecida e será com certeza área de investigação futura.

6 Implementação dos Algoritmos

A implementação destes algoritmos para matrizes esparsas gerais é idêntica à que foi descrita no segundo capítulo para os métodos do LCP. A única alteração a salientar reside no facto de agora ser necessário ter informação respeitante a três conjuntos de índices em lugar de dois. Contudo não vão ser necessários mais vectores do que os já anteriormentre mencionados, pois, como os conjuntos são disjuntos, pode-se guardar a informação respeitante a F nas primeiras #F componentes, a respeitante a T_2 nas $\#T_2$ seguintes e as restantes componentes dirão respeito ao conjunto T_1.

Se a matriz é tridiagonal, então a principal alteração à implementação descrita no capítulo 2 reside na actualização dos valores das variáveis que deve ser feita como se descreve a seguir. Quando um índice transita entre os conjuntos T_1 e T_2 duas situações podem acontecer:

i) $s-1 \notin F$ e $s+1 \notin F$.

Neste caso só é necessário actualizar os valores de w_{s-1} , w_s e w_{s+1}.

ii) $s-1 \in F$ ou $s+1 \in F$ ou ambos.

Então pode acontecer uma de três hipóteses:

a) é necessário actualizar, além de w_s, as componentes de z em todo o bloco que termina em s-1 e a componente de w cujo índice é imediatamente anterior a esse bloco;

b) há que actualizar as componentes de z em todo o bloco que começa em s+1 e a componente de w cujo índice vem imediatamente a seguir ao fim desse bloco;

c) ambas as coisas.

Sempre que um índice transita entre T_1 ou T_2 e F , além de ser necessário actualizar a factorização nos casos que já foram referidos no capítulo 2 é ainda preciso actualizar os valores das variáveis que vão ser afectadas pelo valor da variável de ordem s. Essas variáveis são em menor ou maior número conforme s-1 ou s+1 ou ambos são elementos de F. A situação em que menos variáveis têm que ser alteradas é a que corresponde a que nenhum deles seja elemento de F.

7 Experiência Computacional

Nesta secção é apresentado um estudo computacional comparativo dos métodos descritos neste capítulo em BLCPs com uma matriz simétrica P. As matrizes dos problemas teste são algumas das que já foram descritas no capítulo 2 com a mesma numeração. O termo independente foi gerado de modo idêntico ao descrito no capítulo 2. Neste caso, depois de se gerar aleatoriamente o vector solução z, com o número de componentes positivas pretendido, gera-se também aleatoriamente, o

conjunto T_1 correspondente à solução do BLCP, de tal modo que aos índices de T_1 correspondam componentes positivas do vector z gerado. O vector b é então definido tal que $b_i = z_i$ para $i \in T_1$ e $b_i = z_i + 1$ para $i \notin T_1$. Quanto ao vector w o processo é semelhante ao descrito no capítulo 2, atribuindo a w_i um valor negativo sempre que a sua variável complementar z_i é igual a b_i. Finalmente define-se $q = w - Mz$.

Tal como anteriormente, os resultados foram obtidos no computador CDC CYBER 180-830 da Universidade do Porto.

Os resultados deste estudo computacionàl são apresentados na tabela 4.1. Além dos parâmetros já descritos no capítulo 2, usam-se ainda as seguintes notações

VB = número de variáveis z_i com valor igual ao limite superior na solução óptima.

C = o algoritmo entrou em ciclo.

Da análise da tabela pode-se concluir que o método de Keller tem em geral melhor comportamento para a resolução de BLCPs estritamente monótonos com matrizes esparsas gerais. Para as matrizes tridiagonais o mau funcionamento do método de Keller tem a ver com o critério de escolha da inadmissibilidade, conforme já foi referido no capítulo 2. A primeira extensão do método de Murty é mais eficiente do que a segunda. Com efeito, como seria de esperar esta segunda versão requer mais iterações, uma vez que não permite passagens directas entre T_1 e T_2. Além disso o trabalho por iteração na segunda versão é superior, pois há actualização da decomposição em todas as iterações. No entanto, esta segunda extensão tem a vantagem de ser teoricamente convergente em todos os casos, enquanto que a primeira só é teoricamente convergente para matrizes K.

Problema	N	VA	VB	EXTMURTY1				EXTMURTY2				KELLER			
				NI	NO	T	RES	NI	NO	T	RES	NI	NO	T	RES
TP1	484	340	72	752	79.6	21.8	5-12	1157	253.	61.6	6-12	224	10.4	4.2	9-12
		96	194	1176	294.8	65.7	1-11	1703	862.	195.3	2-11	580	67.4	23.5	2-11
TP5	1500	1050	225	C				1814	240.	104.7	9-13	588	17.8	19.1	1-12
		300	600	C				3799	1350.	433.9	2-12	1655	129.6	90.9	2-12
TP8	600	420	90	332	38.1	12.	2-12	449	61.3	18.5	1-12	191	2.08	2.39	1-12
		120	240	628	82.2	24.4	2-12	903	175.	49.5	2-12	495	15.5	9.4	5-12
TP10	1000	700	150	344	4.9	6.6	1-12	451	15.7	12.1	1-12	344	3.6	7.3	1-12
		200	400	1066	66.4	43.4	3-12	1205	124.	60.9	3-12	1029	32.3	39.1	2-10
TP12	1000	700	150	883	28.9	21.4	1-12	1079	82.5	35.6	1-12	350	5.9	8.2	1-12
		200	400	1795	151.9	75.	2-12	2463	496.	158.3	2-12	1126	55.7	50.8	3-12
TP14	5000	3500	750	2290	1.24	3.62	7-13	3174	1.27	7.6	8-13	1587	0.68	92.5	7-13
		1000	2000	5486	4.7	15.3	2-12	7524	5.29	32.4	2-12	4659	2.8	247.	2-12

Tabela 4.1

Os resultados apresentados na tabela 4.1 indicam a grande dependência desses algoritmos no número de variáveis que muda de subconjunto de índices entre a solução inicial e a solução óptima. Assim, se for possível estabelecer um processo de obter rapidamente uma partição de { 1, ..., n } em { F , T_1 , T_2 } que não difira muito da partição correspondente à solução óptima, então será de esperar que o número de iterações de qualquer um destes métodos para encontrar a solução óptima seja bastante pequeno. O facto das extensões do método de Murty poderem começar em qualquer solução básica torna-os bastante interessantes neste contexto. Contrariamente, o método de Keller necessita de uma solução inicial que satisfaça os limites. Assim, podemos concluir que, apesar do método de Keller ser em geral mais eficiente do que as extensões do método de Murty para a resolução de BLCPs de grandes dimensões, estes últimos processos parecem ser mais interessantes para esse fim, desde que incorporados num processo híbrido que também utilize operações pivotais por bloco. Esse assunto será discutido no próximo capítulo.

Em [41] e [46] apresentamos experiência computacional mais completa que comprova as conclusões tiradas nesta secção.

Como é ilustrado na tabela 4.1 o processo EXTMURTY 1 foi incapaz de resolver o problema TP5, tendo ocorrido um ciclo. Esse ciclo foi devido às cinco primeiras variáveis do problema e é discutido a seguir. Sejam

$$q = \begin{bmatrix} -1 \\ 0 \\ -3 \\ 0 \\ 1 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 2 \end{bmatrix} \qquad a = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$M = \begin{bmatrix} 0.4111478 & & & & \\ 0.3580042 & 1.688328 & & & \\ 0.7129532 & 2.402933 & 3.801952 & & \\ 0.5004849 & 1.042523 & 1.888326 & 1.103488 & \\ -0.5141362 & -1.738911 & -2.541484 & -1.195132 & 1.915638 \end{bmatrix}$$

Os conjuntos F, T_1 e T_2 das iterações do algoritmo EXTMURTY1 têm a seguinte forma

iteração	0	$F = \varnothing$	$T_1 = \varnothing$	$T_2 = \{ 1, 2, 3, 4, 5 \}$
	1	$F = \varnothing$	$T_1 = \{ 1 \}$	$T_2 = \{ 2, 3, 4, 5 \}$
	2	$F = \{ 3 \}$	$T_1 = \{ 1 \}$	$T_2 = \{ 2, 4, 5 \}$
	3	$F = \{ 3 \}$	$T_1 = \{ 1, 5 \}$	$T_2 = \{ 2, 4 \}$
	4	$F = \varnothing$	$T_1 = \{ 1, 3, 5 \}$	$T_2 = \{ 2, 4 \}$
	5	$F = \{ 2 \}$	$T_1 = \{ 1, 3, 5 \}$	$T_2 = \{ 4 \}$
	6	$F = \{ 2 \}$	$T_1 = \{ 1, 3 \}$	$T_2 = \{ 4, 5 \}$
	7	$F = \varnothing$	$T_1 = \{ 1, 3 \}$	$T_2 = \{ 2, 4, 5 \}$
	8	$F = \{ 1 \}$	$T_1 = \{ 3 \}$	$T_2 = \{ 2, 4, 5 \}$
	9	$F = \{ 1, 3 \}$	$T_1 = \varnothing$	$T_2 = \{ 2, 4, 5 \}$
	10	$F = \{ 3 \}$	$T_1 = \{ 1 \}$	$T_2 = \{ 2, 4, 5 \}$

Assim, a partição da iteração 10 já tinha sido encontrada na segunda iteração, o que mostra que o algoritmo entrou em ciclo.

CAPÍTULO V

MÉTODOS PIVOTAIS POR BLOCOS PARA O PROBLEMA LINEAR COMPLEMENTAR COM LIMITES

1 INTRODUÇÃO

No capítulo 3 foram discutidos alguns algoritmos pivotais por blocos para a resolução do LCP. Esses processos admitem alterações de mais de um elemento nos conjuntos de índices associados às variáveis básicas e não básicas. Neste capítulo vamo-nos debruçar sobre este tipo de algoritmos para o BLCP. Tal como foi referido anteriormente, se todos os limites inferiores forem iguais a zero e todos os limites superiores forem finitos, então o BLCP

$$\left.\begin{array}{l} w = q + M z \\ 0 \le z_i \le b_i \\ z_i = 0 \Rightarrow w_i \ge 0 \\ z_i = b_i \Rightarrow w_i \le 0 \\ 0 < z_i < b_i \Rightarrow w_i = 0 \end{array}\right\} \quad i = 1, \ldots, n \qquad (5.1)$$

é equivalente ao LCP

$$\begin{array}{l} t = p + A x \\ x \ge 0 \, , \, t \ge 0 \, , \, x^T t = 0 \end{array} \qquad (5.2)$$

com $A = \begin{bmatrix} M & I \\ -I & O \end{bmatrix}$, $p = \begin{bmatrix} q \\ b \end{bmatrix}$, $x = \begin{bmatrix} z \\ \lambda \end{bmatrix}$, $t = \begin{bmatrix} y \\ \gamma \end{bmatrix}$.

Este tipo de equivalência é explorado no desenvolvimento dos métodos pivotais por blocos para o BLCP. Após a discussão do funcionamento e da convergência de algoritmos de este tipo, apresentaremos alguma experiência computacional que ilustrará a eficiência desses processos para a resolução de BLCPs de grandes dimensões.

2 Método de Pang para $M \in K$

O primeiro dos algoritmos a ser discutido deve-se a Pang [70] e, tal como referimos anteriormente, explora a equivalência entre o BLCP (5.1) e o LCP (5.2). Consideremos então esse último LCP

$$\begin{aligned} & y = q + M z + I \lambda \\ & \gamma = b - I z \\ & z \geq 0 \; ; \; y \geq 0 \; ; \; \lambda \geq 0 \; ; \; \gamma \geq 0 \\ & z^T y = \lambda^T \gamma = 0 \end{aligned} \tag{5.3}$$

e seja $I = \{ i : z_i \text{ é básica } \}$. Da relação de complementaridade tem-se $y_I = 0$. Por outro lado, de $\gamma_I = b_I - z_I$ vem

$$z_I = b_I - \gamma_I \tag{5.4}$$

Substituindo (5.4) em (5.3) e, tendo em conta que $y_I = 0$, obtém-se

$$\begin{aligned} & \lambda_I = - (q_I + M_{II} \, b_I) + M_{II} \, \gamma_I \\ & \lambda_I \geq 0 \; ; \; \gamma_I \geq 0 \; ; \end{aligned} \tag{5.5}$$

Como $M_{II} \in K$, então o algoritmo de Chandrasekaran pode ser usado para a resolução do LCP (5.5). Além disso, é possível provar [70] que a solução $(\gamma_I^*, \lambda_I^*)$ satisfaz a $\gamma_I^* \leq b_I$ e que, portanto, $0 \leq z_I^* \leq b_I$. Agora para $J = \{ 1, \ldots, n \} - I$ tem-se $z_J = 0$ e $\gamma_J = b_J > 0$. Portanto, se $w_J = q_J + M_{JI}\ z_I^* \geq 0$, a solução foi encontrada. Caso contrário, faz-se

$$I = I \cup \{ i \in J : q_i + M_{iI}\ z_I^* < 0 \}$$

e o processo é repetido para este novo conjunto I. Os passos do algoritmo resultante são os seguintes.

Algoritmo Bloco 1 ($M \in K$)

Passo Inicial: $I = \{ i : q_i < 0 \}$

Passo Geral: Use o método de Chandrasekaran para obter a solução $(\lambda_I^*, \gamma_I^*)$ do LCP (5.5).

Calcule $z_I^* = b_I - \gamma_I^*$ e $w_J^* = q_J + M_{JI}\ z_I^*$.

Se $w_J^* \geq 0$ a solução é $z = (z_I^*, 0)$, $w = (0, w_J^*)$ e páre.

Caso contrário faça $I = I \cup \{ i : w_i^* < 0 \}$.

Repita o Passo Geral.

Este algoritmo converge, num número máximo de n iterações, para a solução única do BLCP. Em cada iteração há que resolver um LCP com matriz K usando o algoritmo de Chandrasekaran. Como esse processo é polinomial, então o mesmo acontece com o algoritmo dado.

Tal como é descrito acima, o algoritmo é apenas capaz de lidar com limites inferiores e superiores finitos. A extensão para BLCPs com alguns limites (inferiores

ou superiores) infinitos foi apresentada em [45]. Para entendermos os passos dessa extensão, consideremos a primeira iteração do método anterior. Então

$$I = \{ i : q_i < 0 \}$$

e há que resolver o LCP (5.5) associado a esse conjunto usando o método de Chandrasekaran. Como referimos no capítulo 3, cada iteração do método de Chandrasekaran consiste na modificação dos conjuntos F_1 e T_1 definidos por

$$F_1 = \{ i : \gamma_i \text{ é básica } \} , T_1 = \{ i : \lambda_i \text{ é básica } \}$$

que são associados ao LCP (5.5).

Consideremos agora os conjuntos F, T_1 e T_2 associados a cada solução do BLCP e definidos por

$$F = \{ i : z_i \text{ é básica } \}, T_1 = \{ i : w_i \text{ é básica e } z_i = b_i \}, T_2 = \{ i : w_i \text{ é básica e } z_i = a_i \}$$

Então verificam-se as seguintes equivalências

$$i \in F_1 \Leftrightarrow \gamma_i > 0 \Leftrightarrow z_i < b_i \Leftrightarrow i \in F$$
$$i \in T_1 \Leftrightarrow \gamma_i = 0 \Leftrightarrow z_i = b_i \Leftrightarrow i \in T_1$$
$$i \in J \Leftrightarrow z_i = 0 \Leftrightarrow i \in T_2$$

Além disso, na resolução do LCP (5.5) faz-se, em cada iteração,

$$F_1 = F_1 \cup \{ i \in T_1 : \lambda_i < 0 \} \qquad (5.6)$$

Mas $\lambda_i < 0 \Rightarrow \gamma_i = 0 \Rightarrow z_i = b_i \Rightarrow y_i = 0$. Por outro lado, $y_i = w_i + \lambda_i$ e então

$$\lambda_i < 0 \Rightarrow w_i > 0.$$

Reciprocamente, se $z_i = b_i$ e $w_i > 0$, será $y_i = 0$ e $\lambda_i < 0$. Então (5.6) é equivalente a

$$F = F \cup \{ i \in T_1 : w_i > 0 \}.$$

Tendo em conta as equivalências entre as modificações dos conjuntos F_1, T_1 e J e os respectivos conjuntos F, T_1 e T_2, então, é possível apresentar o método de Pang numa forma equivalente que não só torna a sua implementação mais fácil como permite fazer a sua extensão a limites infinitos. Esse algoritmo pode ser apresentado da seguinte forma

Algoritmo Bloco 1 (M ∈ K)

Passo 0 : Faça $F = \{ i : a_i = -\infty, b_i = +\infty \}$, $T_1 = \{ i : a_i = -\infty, b_i < +\infty \}$, $T_2 = \{ i : a_i = 0 \}$ e $I = F \cup T_1$

Passo 1 : Calcule z_F, w_{T_1}.

Se $w_{T_1} \leq 0$ vá para Passo 2.

Caso contrário faça:

$F = F \cup \{ i \in T_1 : w_i > 0 \}$;

$T_1 = T_1 - \{ i \in T_1 : w_i > 0 \}$.

Repita o Passo1.

Passo 2 : Calcule w_{T_2}.

Se $w_{T_2} \geq 0$ a solução é $z = (z_F, b_{T_1}, 0)$, $w = (0, w_{T_1}, w_{T_2})$ e páre.

Caso contrário faça

$I = I \cup \{ i \in T_2 : w_i < 0 \}$;

$F = \{ i \in I : b_i = +\infty \}$;

$T_1 = \{ i \in I : b_i < +\infty \}$.

Volte a Passo 1.

É de notar que o algoritmo de Chandrasekaran é um caso particular deste processo quando todos os limites superiores são infinitos e que, como dissemos, este algoritmo se reduz ao método de Pang quando todos os limites superiores e inferiores forem finitos.

Seguidamente iremos estabelecer a convergência deste método. Como anteriormente, define-se $G = \{ i : a_i = -\infty \text{ e } b_i = +\infty \}$ e começa-se por mostrar a convergência no caso de $G = \varnothing$. Como $M \in K$ e $G = \varnothing$, são válidos os lemas 4.1 e 4.2. A prova da convergência é dada no teorema que se segue.

Teorema 5.1: Se $M \in K$, então o método Bloco 1 converge para a solução única do BLCP e requer, no máximo, n iterações.

Demonstração: Considere-se primeiro o caso de $G = \varnothing$. No Passo 1 nunca é possível ocorrer um ciclo, pois o número de componentes do conjunto F é crescente e este passo termina sempre (na pior das hipóteses quando $F = I$). Além disso, as variáveis z_j, com $j \in T_1$, que transitam de T_1 para F, vão reduzir o seu valor do limite superior b_j para um valor inferior a b_j. Como foi mostrado no capítulo anterior, as variáveis z_k, $k \in F$, são uma função não decrescente dessas variáveis e, por isso, o valor das variáveis básicas z_j não é incrementado. Por isso, no Passo 1 tem-se sempre $z_j < b_j$ ($j \in F$). Resta mostrar que também se tem $z_j > a_j$ ($j \in F$) no fim do Passo 1. A demonstração pode ser feita por indução sobre o número de iterações k.

Como $G = \varnothing$ então na inicialização do processo ($k = 1$), tem-se $I = \{ i : a_i = -\infty \text{ e } b_i < +\infty \}$. Por isso, para $j \in F \subset I$ $z_j > a_j$, pois $a_j = -\infty$.

Suponhamos, como hipótese de indução, que na iteração k se tem $z_j > a_j$, $j \in F$, no fim do Passo 1. Seja $I = F \cup T_1$, $L = \{ i \in T_2 : w_i < 0 \}$ e $K = I \cup L$. Defina-se o vector p_K tal que

$$p_I = q_I \, , \; p_L = q_L + \varepsilon \, e_L$$

sendo e_L um vector com todas as componentes iguais a 1 e ε um número real positivo tal que

$$q_L + M_{LI} \, z_I + \varepsilon \, e_L \geq 0.$$

Considere-se o BLCP(b_K, p_K, M_{KK}). Então $z_K = (z_I, 0)$ é solução deste BLCP, pois $w_L \geq 0$, $z_I \leq b_I$ e, além disso, $z_F \geq a_F$ ($F \subset I$), pela hipótese de indução. Como $p_K \geq q_K$, então, pelo lema 4.2, a solução, $\tilde{z}_K$, do BLCP(b_K, q_K, M_{KK}), obtida ao fim de k+1 iterações no Passo 1, satisfaz $\tilde{z}_K \geq z_K$. Então $\tilde{z}_F \geq a_F$, pois $F \subset K$.

Então, o algoritmo é convergente num número finito de iterações. Além disso, como o número de elementos do conjunto K é crescente e será no máximo n, o processo termina no máximo em n iterações.

Consideremos agora o caso de ser $G \neq \varnothing$. Tal como anteriormente, seja $\bar{G} = \{ 1, \ldots , n \} - G$ e considere-se o BLCP($b_{\bar{G}}$, $p_{\bar{G}}$, ($M|M_{GG}$)), com $p_{\bar{G}} = q_{\bar{G}} - M_{\bar{G}G} M_{GG}^{-1} q_G$ e ($M|M_{GG}$) o Complemento de Schur de M_{GG} em M. Como ($M|M_{GG}$) é uma matriz K [8], este BLCP pode ser resolvido pelo método Bloco 1. Por outro lado, resolver o BLCP($b_{\bar{G}}$, $p_{\bar{G}}$, ($M|M_{GG}$)) começando com $F = \varnothing$ é equivalente a resolver o BLCP(b, q, M) começando com F = G. Isso mostra que o teorema é também verdadeiro neste caso.

□

A prática mostrou que, sempre que o algoritmo regressa ao Passo 1, vai recomeçar retirando de F todas as variáveis com limite superior finito, para, logo a seguir, as voltar a inserir quase todas em F. A matriz M_{FF} vai assim ter que ser refactorizada pois tinha-se perdido a informação da factorização anterior. É, no entanto, possível fazer uma alteração ao Passo 2 que, de certa maneira, vem

ultrapassar esse inconveniente. Nessa alteração, seja $\bar{F}$ o conjunto dos índices das variáveis z_i básicas obtido no fim do Passo 1. Então o passo 2 passa a ter a seguinte forma:

Passo 2' : (i) Calcule w_{T2}.

Se $w_{T2} \geq 0$ a solução é $z = (z_F, b_{T1}, 0)$, $w = (0, w_{T1}, w_{T2})$ e páre.

Caso contrário faça:

$I = I \cup \{ i \in T_2 : w_i < 0 \}$;

$F = \bar{F} \cup \{ i \in I - \bar{F} : b_i = +\infty \}$;

$T_1 = \{ i \in I - \bar{F} : b_i < +\infty \}$.

(ii) $H = \{ i \in F : z_i > b_i \}$.

Se $H = \varnothing$ vá para Passo 1.

Caso contrário faça:

$F = F - H$; $T_1 = T_1 \cup H$; Repita (ii).

Esta modificação não altera a polinomialidade do algoritmo, pois o número de elementos de H é decrescente em cada iteração. Na pior das hipóteses o processo descrito em (ii) é abandonado ao fim de ser executado um número de vezes inferior ou igual a # F. Na última secção iremos estudar o comportamento destas duas versões do método Bloco para matrizes K.

3 Método Bloco para $M \in P$

No caso de M ser uma matriz P, mas não K, o algoritmo anteriormente descrito não funciona, pois nada garante a admissibilidade das variáveis z_i $i \in F$. No entanto, é possível desenvolver um algoritmo que permita alterações nos conjuntos F,

T_1 e T_2 em mais de um elemento. Para a descrição desse processo considere-se primeiramente o caso de todos os limites serem finitos. Então, como vimos anteriormente, o BLCP é equivalente ao LCP

$$\begin{bmatrix} y \\ \gamma \end{bmatrix} = \begin{bmatrix} q \\ b \end{bmatrix} + \begin{bmatrix} M & I \\ -I & O \end{bmatrix} \begin{bmatrix} z \\ \lambda \end{bmatrix}$$

$$z, \lambda, y, \gamma \geq 0 \, , \, z^T y = \lambda^T \gamma = 0$$

com I a matriz identidade de ordem n. A matriz deste LCP não é simétrica e, além disso, contém zeros na diagonal. Contudo, como $M \in P$, pode-se efectuar uma operação pivotal com pivot M. Assim, obtém-se um LCP equivalente cuja matriz tem os elementos diagonais não nulos. Com efeito tem-se

$$\begin{bmatrix} z \\ \gamma \end{bmatrix} = \begin{bmatrix} -M^{-1}q \\ b + M^{-1}q \end{bmatrix} + \begin{bmatrix} M^{-1} & -M^{-1} \\ -M^{-1} & M^{-1} \end{bmatrix} \begin{bmatrix} y \\ \lambda \end{bmatrix} \qquad (5.7)$$

$$z, \lambda, y, \gamma \geq 0 \, , \, z^T y = \lambda^T \gamma = 0$$

A este LCP pode ser aplicado o método Bloco descrito no capítulo 3, em que os conjuntos das variáveis básicas e não básicas são apresentadas a seguir

$$\bar{F} = \{ i : y_i \text{ ou } \lambda_i \text{ é básica } \}$$

$$\bar{T} = \{ i : z_i \text{ ou } \gamma_i \text{ é básica } \}$$

Então o conjunto das inadmissibilidades $\bar{H}$ será dado pela reunião dos conjuntos $\bar{H}_1$ e $\bar{H}_2$ definidos por

$$\bar{H}_1 = \{ i \in \bar{F} : y_i < 0 \vee \lambda_i < 0 \} \qquad \text{e} \qquad \bar{H}_2 = \{ i \in \bar{T} : z_i < 0 \vee \gamma_i < 0 \}$$

Além disso, cada iteração consiste na modificação dos conjuntos $\bar{F}$ e $\bar{T}$ da seguinte forma

$$\bar{F} = \bar{F} - \bar{H}_1 \cup \bar{H}_2 \qquad \text{e} \qquad \bar{T} = \bar{T} - \bar{H}_2 \cup \bar{H}_1 \qquad (5.8)$$

De notar que o LCP (5.7) tem dimensão 2n e, além disso, implica o cálculo de M^{-1}. Assim torna-se pouco eficiente aplicar o método Bloco directamente ao LCP (5.7). É, contudo, possível desenvolver um método Bloco que procure resolver directamente o BLCP (5.1), explorando convenientemente as equivalências entre os dois problemas. Para isso considere-se o conjunto de inadmissibilidades do BLCP

$$H = \bigcup_{k=1}^{4} H_k$$

com

$$H_1 = \{ i \in F : z_i > b_i \},\ H_2 = \{ i \in F : z_i < a_i \},$$
$$H_3 = \{ i \in T_1 : w_i > 0 \} \text{ e } H_4 = \{ i \in T_2 : w_i < 0 \}. \qquad (5.9)$$

Se $i \in \bar{H}_1$ então $y_i < 0$ ou $\lambda_i < 0$. De notar que as duas condições são incompatíveis. Do mesmo modo, se $i \in \bar{H}_2$ tem-se $z_i < 0$ ou $\gamma_i < 0$ sendo também estas duas condições incompatíveis. Analisemos os quatro casos separadamente:

(i) $i \in \bar{H}_1$ e $y_i < 0$. Então

$y_i < 0 \Rightarrow z_i = 0 \Rightarrow \gamma_i = b_i \Rightarrow \lambda_i = 0$.

Como $y_i = w_i + \lambda_i$, então $w_i < 0$. Portanto $i \in T_2$ e $w_i < 0$, isto é, $i \in H_4$.

Reciprocamente

$i \in H_4 \Rightarrow z_i = 0 \wedge w_i < 0;\ z_i = 0 \Rightarrow \gamma_i = b_i \Rightarrow \lambda_i = 0$.

Como $y_i = w_i + \lambda_i$ vem $y_i < 0$. Logo $i \in \bar{H}_1$ e $y_i < 0$.

(ii) $i \in \bar{H}_1$ e $\lambda_i < 0$. Então

$\lambda_i < 0 \Rightarrow \gamma_i = 0 \Rightarrow z_i = b_i \Rightarrow y_i = 0$.

Como $y_i = w_i + \lambda_i$, então $w_i > 0$. Portanto $i \in T_1$ e $w_i > 0$, isto é, $i \in H_3$.

Reciprocamente

$i \in H_3 \Rightarrow z_i = b_i \wedge w_i > 0;\ z_i = b_i \Rightarrow \gamma_i = 0 \wedge y_i = 0$.

Como $y_i = w_i + \lambda_i$ será $\lambda_i < 0$. Logo $i \in \bar{H}_1$ e $\lambda_i < 0$.

(iii) $i \in \bar{H}_2$ e $z_i < 0$. Então

$z_i < 0 \Rightarrow \gamma_i > 0 \wedge y_i = 0 \Rightarrow \lambda_i = 0 \wedge y_i = 0 \Rightarrow w_i = 0$.

Então $i \in F$ e $z_i < 0$, isto é, $i \in H_2$.

Reciprocamente

$i \in H_2 \Rightarrow z_i < 0 \wedge w_i = 0$; $w_i = 0 \Rightarrow y_i = \lambda_i$.

Mas $z_i < 0 \Rightarrow \gamma_i > 0 \Rightarrow \lambda_i = 0$.

Como $w_i = \lambda_i = 0$ então $y_i = 0$. Logo $i \in \bar{H}_2$ e $z_i < 0$.

(iv) $i \in \bar{H}_2$ e $\gamma_i < 0$. Então

$\gamma_i < 0 \Rightarrow z_i > b_i \Rightarrow y_i = 0$

$\gamma_i < 0 \Rightarrow \lambda_i = 0$.

Portanto $w_i = 0$. Donde $i \in F$ e $z_i > b_i$, isto é, $i \in H_1$.

Reciprocamente

$i \in H_1 \Rightarrow z_i > b_i \wedge w_i = 0$; $z_i > b_i \Rightarrow \gamma_i < 0 \Rightarrow \lambda_i = 0$

e $z_i > b_i \Rightarrow y_i = 0$.

Logo $i \in \bar{H}_2$ e $\gamma_i < 0$.

Tendo em atenção estas equivalências, as modificações (5.8) podem ser escritas em termos dos conjuntos F, T_1 e T_2 do seguinte modo:

$$\begin{aligned} F &= F - (H_1 \cup H_2) \cup (H_3 \cup H_4) \\ T_1 &= T_1 - H_3 \cup H_1 \\ T_2 &= T_2 - H_4 \cup H_2 \end{aligned} \qquad (5.10)$$

com H_i, $i = 1, 2, 3, 4$, os conjuntos definidos por (5.9).

Além disso, deve-se garantir que todas as variáveis z_i sejam inicialmente básicas, isto é, que se tenha inicialmente $F = \{ 1, \ldots, n \}$, $T_1 = T_2 = \varnothing$.

Tal como foi visto no capítulo 3, torna-se necessário combinar esta estratégia pivotal por blocos com um algoritmo pivotal simples de modo a ser possível estabelecer a convergência do processo. O algoritmo pivotal simples a considerar deve ser capaz de lidar com qualquer solução inicial, o que torna o método de Keller inadequado. No capítulo anterior apresentámos uma extensão do método de Murty capaz de resolver qualquer BLCP com uma matriz P e que pode ser iniciado com qualquer solução básica. Se, tal como foi feito para o LCP, combinarmos esse processo com o método Bloco agora apresentado, obtemos um método convergente que deverá possuir todas as vantagens do correspondente método Bloco para o LCP.

Na descrição deste processo considerámos que todos os limites são finitos. O tratamento de limites inferiores ou superiores infinitos é muito simples. Com efeito, neste caso apenas o Passo Inicial deve ser modificado de forma semelhante à apresentada no método anterior. A diferença reside no facto de também deverem ser tornadas básicas todas as variáveis z_i correspondentes a limites superiores finitos. Tendo em conta estas considerações, podemos apresentar os passos do algoritmo Bloco 2 para matrizes P.

Algoritmo Bloco 2 ($M \in P$)

Passo 0: Faça: $k = 1$; $ninf = nmaxpv = n$;

$F = \{ i : a_i = -\infty \text{ e } b_i = +\infty \} \cup \{ i : b_i < +\infty \}$;

$T_1 = \varnothing$; $T_2 = \{ i : a_i = 0 \text{ e } b_i = +\infty \}$

Passo 1: Calcule z_F e w_T

Defina H usando (5.9)

(i) Se $H = \varnothing$ páre. $z = (z_F, b_{T1}, 0)$, $w = (0, w_{T1}, w_{T2})$ é a solução do BLCP .

(ii) Caso contrário,

se #H < ninf, faça ninf = #H, nmaxpv = k + p e vá para Passo 2

se #H ≥ ninf

se k ≤ nmaxpv vá para Passo 2.

se k > nmaxpv vá para Passo 3.

Passo 2: Actualize F, T_1 e T_2 usando (5.10).

Faça k = k + 1 e volte a Passo1.

Passo 3: Defina s = min { i ∈ H }

Se $s \in H_1$ faça $F = F - \{ s \}$ e $T_1 = T_1 \cup \{ s \}$;

Se $s \in H_2$ faça $F = F - \{ s \}$ e $T_2 = T_2 \cup \{ s \}$;

Se $s \in H_3$ faça $F = F \cup \{ s \}$ e $T_1 = T_1 - \{ s \}$;

Se $s \in H_4$ faça $F = F \cup \{ s \}$ e $T_2 = T_2 - \{ s \}$.

Faça k = k + 1 e volte a Passo 1.

Este algoritmo termina num número finito de iterações. Isso é consequência da convergência da extensão do método de Murty já demonstrada no capítulo anterior. O método Bloco só é utilizado enquanto houver decréscimo no número de inadmissibilidades. No caso de haver um acréscimo permite-se ainda que se façam (p - 1) iterações do método Bloco, passando-se então para a extensão do método de Murty (Passo 3) se, entretanto, o número de inadmissibilidades não tiver descido abaixo do valor ninf que existia antes desse número começar a aumentar.

Tal como no método Bloco para o LCP, a escolha do parâmetro p é muito importante. Um valor muito pequeno pode fazer com que se use a extensão do método de Murty cedo de mais e vezes de mais. Um valor muito grande para p pode originar que se executem muitas iterações em bloco sem que se esteja a progredir em relação à solução. Nas nossas experiências usámos valores para p com grandezas semelhantes às utilizadas no método Bloco para o LCP.

4 IMPLEMENTAÇÃO

Os algoritmos Bloco para o BLCP são implementados de modo semelhante ao dos métodos Bloco para LCP. A implementação por colunas, que tinha sido inicialmente proposta, pode aqui ser usada, mas, tal como nos métodos Bloco para o LCP não é muito eficiente, pois não possibilita que a factorização LDL^T da matriz M_{FF} de uma iteração seja actualizada quando vários elementos são alterados no conjunto F. Com efeito, como já foi referido anteriormente, utilizando o esquema por colunas, cada submatriz M_{FF} deve ser completamente factorizada em cada iteração. Principalmente para as matrizes K, há um grande número de componentes do conjunto F que se mantêm de iteração em iteração, justificando que se tente aproveitar o máximo possível da factorização de uma submatriz numa dada iteração para a iteração seguinte. O caso de haver várias variáveis sem restrição alguma também faz com que os conjuntos F de iterações sucessivas tenham muitos elementos em comum. Um tal processo é de difícil implementação mantendo o esquema por colunas e, por isso, usou-se neste caso o esquema por linhas descrito no capítulo 3. Tal como foi aí descrito, é executada inicialmente uma factorização simbólica por colunas que depois é transposta para se obter o esquema por linhas. De igual modo, na fase final, se a extensão do método de Murty é usada, torna-se necessário tornar a

transpor a factorização, para ser possível empregar o algoritmo de Bennett na actualização da factorização quando só um elemento é alterado no conjunto F.

Tal como foi referido no capítulo anterior, a informação de qual é o conjunto F, T_1 ou T_2 a que uma variável pertence é guardada num único vector, não sendo necessário utilizar mais vectores de ponteiros. Quanto aos vectores de elementos reais só é necessário usar mais dois vectores com os valores numéricos dos limites das variáveis. Nos casos de limites infinitos é ainda necessário usar um vector de marcações para se saber rapidamente se uma variável tem limites finitos ou infinitos.

Estes métodos foram também implementados para matrizes tridiagonais seguindo as técnicas já descritas em capítulos anteriores.

5 Experiência Computacional

Nesta secção é apresentada alguma experiência computacional com os dois algoritmos pivotais por blocos (Bloco 1 e Bloco 2) descritos nas secções anteriores.

Na nossa primeira experiência, estudámos a eficiência do método Bloco 1 para matrizes K. As matrizes dos BLCPs utilizados foram geradas como a seguir se descreve.

TP28 e TP29 - Matrizes K geradas completamente ao acaso.

TP30 e TP31 - Matrizes pentadiagonais tais que $m_{ii} = 10$, $m_{ii-1} = -4$, $m_{ii-2} = -1$, $i = 1, \ldots, n$.

TP32 e TP33 - Matrizes K geradas de acordo com [68], tridiagonais por blocos, sendo os blocos diagonais, por sua vez, matrizes

tridiagonais e os blocos na diagonal secundária matrizes diagonais.

TP34 e TP35 - Matrizes K tridiagonais geradas aleatoriamente.

Os limites inferiores finitos foram todos considerados iguais a zero. Na geração dos vectores b e q usámos uma extensão da técnica descrita no capítulo 2. Assim, o vector z é primeiramente construído tal como no processo anterior. Seguidamente escolhem-se as componentes de z que pertencem a T_1 na solução do BLCP e define-se o vector b tal que $b_i = z_i$ para $i \in T_1$ e $b_i = z_i + 1$ para $i \notin T_1$ e $b_i < +\infty$. Finalmente considera-se o vector w tal que $w_i = 0$ para $i \in F$, $w_i < 0$ para $i \in T_1$ e $w_i > 0$ para $i \in T_2$. O vector q obtém-se de $q = w - M z$.

Na tabela 5.1 apresentam-se as principais características das matrizes destes BLCPs. O parâmetro TS representa o tempo de execução, em segundos de CPU, da fase simbólica inicial, isto é, ordenação e factorização simbólica de toda a matriz.

Os resultados apresentados nas tabelas 5.1, 5.2 e 5.3 desta secção foram obtidos no computador CDC - CYBER 180-830 da Universidade do Porto, enquanto que os resultados da tabela 5.4 foram obtidos numa Workstation SUN 3-60 no Departamento de Matemática da Universidade de Coimbra.

As tabelas 5.2 e 5.3 apresentam os resultados das nossas experiências. Para atestar melhor do comportamento do método Bloco, resolvemos compará-lo com o método de Keller descrito no capítulo anterior. Na tabela 5.2 são apresentados resultados com BLCPs em que todos os limites inferiores são finitos e iguais a zero ($G = \varnothing$), enquanto que a tabela 5.3 se refere a BLCPs com limites inferiores finitos e infinitos ($G \neq \varnothing$).

Problem a	n	nonz	TS
TP28	1500	5000	5.06
TP29	2300	7314	6.56
TP30	1500	7494	——
TP31	2000	9994	——
TP32	1280	6208	3.77
TP33	1600	7768	4.82
TP34	1000	2998	——
TP35	5000	14998	——

Tabela 5.1

Além dos parâmetros já usados noutros capítulos, usamos ainda as seguintes notações:

LIF = número de limites inferiores finitos.

LSF = número de limites superiores finitos.

#G = número de variáveis sem qualquer restrição.

$\#T_1$ = número de variáveis iguais ao limite superior na solução do BLCP.

#F = número de variáveis básicas na solução do BLCP.

Além disso, NO representa o número de multiplicações e divisões multiplicado por 10^{-4} e NI o número de vezes que o conjunto F é alterado.

Da análise dos resultados apresentados nas tabelas 5.2 e 5.3 pode-se concluir que o método Bloco é francamente superior ao método de Keller. Das duas versões do método Bloco parece ser a segunda a que tem melhor desempenho, principalmente nos casos em que $G \neq \emptyset$. Com efeito, neste caso, há um grande número de variáveis que permanecem em F (pelo menos todas as de G) o que torna recomendável a

	LSF	#F	$\#T_1$	BLOCO 1 (versão 1)				BLOCO 1 (versão 2)				KELLER			
				NI	NO	T	RES	NI	NO	T	RES	NI	NO	T	RES
TP28	750	300	300	4	0.67	0.56	2-12	2	0.62	0.5	1-12	693	16.4	24.5	2-12
		1250	75	5	12.9	3.77	3-11	4	19.8	5.5	3-11	1371	242.	154.	3-10
TP29	1150	460	460	4	1.04	0.8	2-12	2	0.9	0.75	2-12	1076	36.4	58.2	2-12
		1840	230	5	11.8	4.1	2-11	4	17.7	5.7	2-11	2022	291.	270.	3-10
TP30	750	300	300	5	1.1	0.7	1-12	4	1.2	0.7	1-12	774	30.	29.7	2-12
		1250	75	21	15.1	5.6	6-12	9	10.2	3.6	6-12	1395	108.	172.	1-11
TP31	1000	400	400	6	1.6	1.1	1-12	5	1.8	1.1	1-12	1045	51.4	54.2	2-12
		1600	100	21	20.	7.6	6-12	10	14.	4.9	6-12	1791	167.	262.	1-11
TP32	640	256	256	5	0.95	0.7	3-12	4	1.1	0.8	3-13	671	22.7	23.3	5-13
		1024	64	13	41.6	13.2	6-12	7	31.3	9.9	7-12	1150	169.	116.	5-12
TP33.	800	320	320	6	1.4	1.1	4-13	5	1.6	1.1	4-13	825	32.4	32.9	7-13
		1280	80	11	44.9	14.1	1-11	7	44.7	13.6	1-11	1439	242.	176.	6-12
TP34	500	200	200	5	0.5	0.3	3-13	7	0.5	0.33	4-13	401	0.27	0.31	4-13
		800	50	5	1.5	0.5	8-13	7	1.6	0.54	8-13	851	2.2	1.1	8-13
TP35	2500	1000	1000	4	1.98	1.25	8-13	7	2.4	1.7	8-13	2001	1.27	4.5	8-13
		4000	250	5	7.5	2.4	2-12	8	9.7	3.1	2-12	4251	10.9	13.4	2-12

Tabela 5.2

	#G	LIF	LSF	#F	$\#T_1$	BLOCO 1 (versão 1)				BLOCO 1 (versão 2)				KELLER			
						NI	NO	T	RES	NI	NO	T	RES	NI	NO	T	RES
TP29	575	1150	1725	653	1498	9	5.3	2.62	2-12	4	3.79	1.62	2-12	1073	58.9	65.8	2-8
	1150	575	1150	1224	997	6	5.41	2.79	4-12	3	4.98	1.99	4-12	566	63.4	53.6	5-7
TP31	500	1000	1500	558	1341	9	7.16	2.52	5-12	5	5.39	1.83	5-12	951	50.9	53.6	8-7
	1000	500	1000	1039	921	9	9.55	3.38	1-11	4	6.75	2.08	1-11	497	52.8	42.9	1-7
TP33	400	800	1200	453	1083	9	6.13	2.56	1-13	5	4.66	1.79	1-13	776	34.3	36.	1-8
	800	400	800	848	733	9	12.1	5.14	4-13	5	9.47	3.49	4-13	413	42.6	33.1	5-7
TP35	1250	2500	3750	2156	1441	13	8.53	3.87	2-13	13	8.1	3.63	2-13	1931	2.9	4.52	2-13
	2500	1250	2500	3363	923	11	10.7	3.79	2-13	12	10.8	3.83	2-13	1371	4.22	4.17	2-13

Tabela 5.3

actualização da factorização de M_{FF} já obtida. De notar ainda que, embora os métodos Bloco continuem a ter melhor desempenho para o caso das matrizes tridiagonais, a diferença entre eles e o método de Keller não é, nesse caso, tão significativa. Tal deve-se à implementação própria das matrizes tridiagonais que é adequada às modificações de um só elemento no conjunto F.

Na tabela 5.4 apresentam-se os resultados obtidos na resolução de BLCPs extritamente monótonos com matriz simétrica P usando o método bloco. Como o desempenho é muito semelhante ao obtido com matrizes K, não considerámos a comparação com qualquer outro método. As matrizes dos BLCPs testados são algumas das já usadas em outros capítulos, tendo-se mantido a notação então usada.

				BLOCO 2	
Problema	n	#F	$\#T_1$	NI	T
TP26	1086	255	531	10	6.5
		572	142	11	10.2
TP27	1919	384	302	9	9.2
		497	190	8	9.0
TP1	484	53	273	9	2.8
		115	113	10	2.9
TP4	1000	85	647	12	4.7
		186	129	15	4.2
TP8	600	8	445	5	2.5
		78	72	2	2.1
TP23	1500	150	495	7	3.1
		502	148	4	3.0

Tabela 5.4

Nesta experiência os limites são todos finitos sendo os limites inferiores iguais a zero. Os resultados apresentados indicam um desempenho do método bloco semelhante ao

verificado na resolução de LCPs estritamente monótonos. Assim podemos concluir que este algoritmo é muito eficiente para a resolução de problemas lineares complementares com ou sem limites superiores de grandes dimensões e estrutura esparsa.

CAPÍTULO VI

MÉTODOS DE OPTIMIZAÇÃO NÃO LINEAR

PARA PROBLEMAS LINEARES COMPLEMENTARES

1 INTRODUÇÃO

Neste capítulo descrevem-se métodos de resolução do LCP

$$\begin{aligned} & w = q + M\,z \\ & w \geq 0\ ;\ z \geq 0 \\ & z^T w = 0 \end{aligned} \qquad (6.1)$$

e do BLCP

$$\left.\begin{aligned} & a_i \leq z_i \leq b_i \\ & z_i = a_i \Rightarrow w_i \geq 0 \\ & z_i = b_i \Rightarrow w_i \leq 0 \\ & a_i < z_i < b_i \Rightarrow w_i = 0 \end{aligned}\right\} \quad w = q + M\,z \quad i = 1\ ,\ \ldots\ ,\ n \qquad (6.2)$$

que exploram a sua equivalência com problemas de optimização não linear. Como referimos anteriormente o LCP (6.1) é equivalente ao programa quadrático

$$\min \quad Q(z) = q^T z + \tfrac{1}{2} z^T M z \qquad (6.3)$$

$$z \geq 0$$

desde que a matriz M seja PSD simétrica. Por sua vez, o BLCP (6.2) é equivalente ao programa quadrático

$$\min \quad Q(z) = q^T z + \tfrac{1}{2} z^T M z \qquad (6.4)$$

$$0 \leq z \leq b$$

em iguais circunstâncias.

É também possível [58] estabelecer a equivalência entre o LCP (6.1) e o sistema de equações não lineares

$$H_i(z) = \min(z_i, w_i) \ , \ i = 1, \ldots, n \qquad (6.5)$$

É de notar que as funções H_i não são necessariamente diferenciáveis, o que, como veremos mais adiante, implica um certo cuidado na resolução desse sistema.

Neste capítulo iremos descrever algoritmos para a resolução do LCP (6.1) que exploram a sua equivalência com o PQ (6.3) ou com o sistema (6.5). A maior parte desses métodos podem ser facilmente estendidos para o BLCP (6.2) explorando a sua equivalência com o PQ (6.4). Por isso não nos iremos debruçar demoradamente sobre a resolução do BLCP.

2 MÉTODOS ITERATIVOS BÁSICOS

Como foi visto no capítulo anterior, a utilização de métodos directos na resolução do LCP (6.1) obriga à resolução, em cada iteração, de um sistema de equações lineares, o que acarreta a necessidade de factorizar a sua matriz. Se M é uma matriz esparsa de ordem elevada, então a factorização necessita de um algoritmo de ordenação para controlar o número de enchimentos. Esse número pode ser, em alguns casos, incomportável para a capacidade da máquina que esteja a ser utilizada. Os métodos iterativos são normalmente usados com o objectivo de contornar esse tipo de problemas. Com efeito, de um modo geral, estes métodos limitam-se a executar operações de produto matriz por vector e produto interno de vectores. Não há, assim, quaisquer enchimentos e o espaço de armazenamento é normalmente muito reduzido. À semelhança dos métodos iterativos para a solução de sistemas, estes processos geram uma sucessão de vectores $\{ z^k \}$ teoricamente convergente para a solução do LCP. A convergência só se verifica no limite, pelo que só se consegue obter uma solução aproximada.

Os principais métodos iterativos baseiam-se numa partição (splitting) da matriz M da forma $M = G + H$, com G e H duas matrizes e diferem entre si no modo como essa partição é efectuada. A sucessão $\{ z^k \}$ é obtida resolvendo uma sucessão de LCPs com a matriz G. Assim, se z^k é um vector associado à iteração k, então calcula-se z^{k+1} como sendo a solução do LCP

$$\begin{aligned} & w = (q + H z^k) + G z \\ & w \geq 0 \; ; \; z \geq 0 \\ & z^T w = 0 \end{aligned} \tag{6.6}$$

A matriz G deve ser escolhida de modo a que os LCPs (6.6) tenham sempre solução e sejam fáceis de resolver. Por isso $G \in Q$, com Q a classe de matrizes introduzida no capítulo 1. Se tal acontecer então $M = G + H$ é uma Q-partição de M e é possível mostrar que em certas condições a sucessão $\{ z^k \}$ é convergente e o seu limite é uma solução do LCP [72].

Tal como é referido em [72] existem três processos muito importantes de efectuar a referida partição. Seguidamente iremos descrever essas três formas, onde usámos a igualdade $M = L + D + U$ com L a parte triangular inferior de A, U a triangular superior e D a sua diagonal.

a) $G = D \; ; \; H = L + U$

Este processo é correspondente ao método de Jacobi para a resolução de sistemas de equações lineares e costuma ser designado por Método de Jacobi Projectado. Esta designação deriva do facto de se projectar no primeiro octante o vector obtido segundo as regras do método de Jacobi usual. A resolução do LCP (6.6) fica muito simples, pois trata-se de um problema separado nas variáveis z_i. Assim, dado o vector z^k, o vector z^{k+1} calcula-se com toda a facilidade a partir de:

$$z_i^{k+1} = \max \{ 0 , - (q_i + \sum_{j \neq i} m_{ij} z_j^k) / m_{ii} \} \quad i = 1 , \ldots , n$$

b) $G = D + L \; ; \; H = U$

Este processo é correspondente ao método de Gauss-Seidel para a resolução de sistemas. A resolução do LCP (6.6) é bastante simples pelo facto de se tratar de uma matriz triangular. Também aqui é possível obter uma expressão que dê cada componente do vector z^{k+1} à custa do vector z^k do seguinte modo

$$z_i^{k+1} = \max \{ 0 , - (q_i + \sum_{j<i} m_{ij} z_j^{k+1} + \sum_{j>i} m_{ij} z_j^k) / m_{ii} \} \quad i = 1 , \ldots , n$$

c) $G = L + \frac{1}{\omega} D \; ; \; H = U + (1 - \frac{1}{\omega}) D \quad \text{com } 0 < \omega < 2$

Trata-se do método PSOR (projected successive overrelaxation) que corresponde ao método SOR para resolução de sistemas lineares. Note-se que o método de Gauss-Seidel é o caso particular deste processo em que $\omega = 1$. Embora com uma expressão mais complicada, é ainda possível obter cada componente do vector z^{k+1} à custa do vector z^k da seguinte forma

$$z_i^{k+1} = \max \{ 0 , - (\omega q_i + (\omega - 1) m_{ii} z_i^k + \omega \sum_{j<i} m_{ij} z_j^{k+1} + \omega \sum_{j>i} m_{ij} z_j^k) / m_{ii} \} \quad i = 1 , \ldots , n$$

Apresentámos assim os três métodos iterativos mais conhecidos, nomeadamente os algoritmos de Jacobi, Gauss-Seidel e PSOR. Muitas outras partições podem ser consideradas dando lugar a novos algoritmos. Esses processos podem ser particularmente úteis em LCPs estruturados onde a matriz G é facilmente encontrada a partir desse tipo de estrutura. Contudo, não nos iremos debruçar sobre esse tipo de algoritmos, limitando a nossa análise aos três processos mencionados em cima.

Para estabelecer a convergência dos métodos iterativos algumas considerações devem ser feitas sobre o tipo de partição que conduziu à definição desses processos. Assim, uma partição M = G + H diz-se regular se G - H é positiva definida (PD). É fácil de ver que se M é simétrica e D é uma matriz PD, então as partições definidas em a), b) e c) são regulares [72]. Se a matriz M é simétrica PD o LCP (6.1) é equivalente ao PQ (6.2). A convergência dos métodos iterativos básicos (Jacobi,

Gauss-Seidel e PSOR) é baseada nesse tipo de equivalência e é apresentada no seguinte teorema.

Teorema 6.1: Seja M simétrica e M = G + H uma partição regular de M. Se cada um dos conjuntos de nível $S(\alpha) = \{ z : (z \geq 0 \wedge q^T z + \frac{1}{2} z^T M z \leq \alpha) \}$ com $\alpha \in R$ for limitado, então, para cada vector $z^0 \geq 0$, a sucessão $\{ z^k \}$ obtida por solução dos LCPs (6.6) tem pelo menos um ponto de acumulação que resolve o LCP (6.1)

Este teorema diz que a convergência dos métodos iterativos básicos para uma matriz simétrica M está dependente do facto dos conjuntos de nível $S(\alpha)$ serem limitados. É fácil de ver que tal acontece se $M \in PD$ [59], pelo que esses três processos são convergentes se M é simétrica PD. Estes algoritmos podem ser facilmente estendidos ao BLCP usando uma projecção do tipo

$$z_i^{k+1} > b_i \Rightarrow z_i^{k+1} = b_i$$

Desse modo obtêm-se fórmulas para os métodos de Jacobi, Gauss-Seidel e PSOR semelhantes às apresentadas anteriormente para o LCP, com condições de convergência também semelhantes.

3 Método de Restrições Activas

Tal como no caso anterior, este tipo de algoritmos resolve o LCP (6.1) a partir da determinação de um ponto estacionário para o programa quadrático (6.2). Para isso usam uma estratégia de restrições activas, que consiste em fixar algumas variáveis z_i em zero e minimizar a função objectivo com respeito às outras variáveis

de modo a que só se considerem soluções admissíveis, isto é, $z \geq 0$ em cada iteração. Definam-se os conjuntos de índices F e T como no capítulo anterior, isto é,

$$T = \{ i : z_i = 0 \} , F = \{ i : z_i > 0 \}$$

Então, em cada iteração, há que resolver o programa quadrático

$$\begin{array}{ll} \text{Min} & Q(z) \\ \text{sujeito a} & z_i = 0 , i \in T \\ & z_i > 0 , i \in F \end{array} \qquad (6.7)$$

Seja $g(z) = \nabla Q(z) = q + M z$ o gradiente de Q em z. Como é sabido, z é um ponto estacionário se satisfizer as seguintes condições

$$g_i(z) = 0 \text{ para } i \in F \qquad (6.8a)$$

$$g_i(z) \geq 0 \text{ para } i \in T \qquad (6.8b)$$

Seja $z \geq 0$ um ponto estacionário e escolha-se $s \in T$ tal que $g_s(z) < 0$. Se apenas fizermos incrementar a variável z_s, mantendo todas as outras variáveis sem alteração, obtemos uma direcção p tal que $\nabla Q(z)^T p < 0$. Então a direcção é descendente e existe um número real α positivo tal que $z(\alpha) = z + \alpha p$ satisfaz a condição (6.8b) com $i = s$ e $Q(z(\alpha)) < Q(z)$. Se o ponto que minimiza $Q(z(\alpha))$ é encontrado sem violar nenhuma restrição, então $g_s(z(\alpha)) = 0$ e a condição (6.8a) passa a ser satisfeita para s. Os novos conjuntos F e T são dados por

$$T = T - \{ s \} \ , \ F = F \cup \{ s \}$$

e nova iteração deve ser efectuada. Por outro lado, se o valor de α que minimiza $Q(z(\alpha))$ é tal que $z(\alpha)$ deixa de ser admissível, então alguma variável z_r com $r \in F$ vai-se tornar negativa. Então deve-se tomar o maior valor de α para o qual

$z(\alpha)$ ainda seja admissível. Para esse valor de α a variável z_r anula-se e, por isso, o índice r passa de F para T, isto é,

$$F = F - \{ r \} \ , \ T = T \cup \{ r \}$$

Na nova iteração a variável z_s deve ser novamente incrementada e o processo é repetido.

Existem vários processos para a escolha do índice $s \in T$ [22]. Nesse mesmo artigo é mostrado que a estratégia que consiste em escolher s tal que $-g_s$ seja máximo é bastante competitiva com as outras e tem a vantagem de ser a de mais fácil implementação. Não é difícil mostrar, que seguindo esta estratégia, este algoritmo e o método de Keller são equivalentes desde que se use também um método directo para o cálculo da direcção p. Por essas razões não iremos continuar com a abordagem do método das restrições activas.

4 MÉTODO DE O' LEARY

Como referimos na secção anterior, um método directo é normalmente usado para a resolução do sistema de equações lineares com matriz M_{FF} que tem de ser resolvido para o cálculo da direcção p em cada iteração. Em [68] é proposta a utilização do método de gradientes conjugados para a resolução desse sistema.

O algoritmo resultante lida com dois tipos de iterações que se podem designar por interior e exterior. Na iteração exterior fixam-se certas variáveis em zero, isto é, define-se o conjunto T e na iteração interior resolve-se (às vezes não completamente) o PQ (6.7) assim resultante, por uma técnica de gradientes conjugados em que a direcção descendente p é calculada com recurso a uma matriz de escalonamento E.

Sempre que, na iteração interior, o valor do passo é limitado por alguma ou algumas variáveis se tornarem negativas, o conjunto dos índices das variáveis nulas (conjunto dos índices das restrições activas) é alterado e recomeça-se a resolver o novo subproblema. Os passos do algoritmo são apresentados a seguir

ALGORITMO O' LEARY

PASSO INICIAL :

Escolha $z^o \geq 0$

Faça $I = \{ 1, \ldots, n \}$ e $k = 0$

ITERAÇÃO EXTERIOR :

$k = k + 1$

$z^k = z^{k-1}$

$w^k = q + M z^k$

$I_{k-1} = I$

$I_k = \{ i : z_i^k = 0 \wedge w_i^k > 0 \}$

Se $I_k = I_{k-1}$ então páre. (z^k , w^k) é solução do LCP.

Caso contrário faça $I = I_k$ e execute a Iteração Interior.

ITERAÇÃO INTERIOR :

PASSO 1: $J = \{ 1, \ldots, n \} - I$

(começa-se a resolver $M_{JJ}\, x = - q_J$)

$x^o = z_J^k$

$r^o = - (q_J + M_{JJ}\, x^o)$

$a_{cg} = \| r^o \|_2^2 \; / \; (r^o)^T M_{JJ}\, r^o$

$$a_{mx} = \min_{j \in J} \{ - \frac{x_j^o}{r_j^o}) \quad , r_j^o < 0 \}$$

$a_o = \min (a_{cg} , a_{mx})$

$x^1 = x^o + a_o r^o$

$r^1 = r^o + a_o M_{JJ} r^o$

Faça $z_j^k = x^1$

Se $r^1 = 0$ volte à Iteração Exterior.

Caso contrário determine $K = \{ j : x_j^k = 0 \}$.

Se $K \cup I = \{ 1 , 2 , \ldots , n \}$ volte à Iteração Exterior.

Caso contrário: se $K = \varnothing$ vá para Passo 2;

se $K \neq \varnothing$ faça $J = J - K$

$x^1 = z_J^k$

$r^1 = b_J - x^1$

e vá para Passo 2.

PASSO 2: (escolhe-se a matriz de escalonamento E e continua-se o algoritmo de gradientes conjugados)

$i = 1$ a primeira direcção é dada por $p^1 = E^{-1} r^1$.

PASSO 3:

$$a_{cg} = \frac{(r^i)^T p^i}{(p^i)^T M_{JJ} p^i}$$

$$a_{mx} = \min_{j \in J} \{ \ -\frac{x_J^i}{p_J^i} : p_j^i < 0 \}$$

$a_i = \min \{ a_{cg} , a_{mx} \}$

$x^{i+1} = x^i + a_i p^i$

$r^{i+1} = r^i + a_i M_{JJ} p^i$

Faz $z_J^k = x^{i+1}$

Se $r^{i+1} = 0$ volte à Iteração Exterior.

Caso contrário determine $K = \{ j : x_j^{i+1} = 0 \}$.

Se $K \cup I = \{ 1, 2, \ldots, n \}$ volte à Iteração Exterior.

Se $K \neq \emptyset$ faça $J = J - K$ e repita a Iteração Interior.

Se $K = \emptyset$ procure a nova direcção ortogonal

$$b_i = \frac{(r^{i+1})^T E^{-1} r^{i+1}}{(r^i)^T E^{-1} r^i}$$

$$p^{i+1} = E^{-1} r^{i+1} + b_i p^i$$

$i = i + 1$

e repita o Passo 3.

A escolha da matriz E condiciona de forma decisiva a eficácia do algoritmo. Com efeito, essa matriz deve ser escolhida de forma a que os sistemas $r = E p$ sejam fáceis de resolver e ao mesmo tempo que a rapidez de convergência do método de gradientes conjugados seja acelerada. A prática tem mostrado que não se podem estabelecer regras fixas sobre o processo de determinar uma boa matriz E. Cada caso é um problema novo e um método que dê bons resultados para um caso concreto pode dar resultados desastrosos quando aplicado a outro problema.

De cada vez que a iteração interior é repetida há que obter um escalonamento para a nova matriz M_{JJ} que teria em princípio que ser calculado. Para evitar esse cálculo é sugerido em [68] que se encontre uma matriz E de escalonamento para M e, em cada iteração interior, usar E_{JJ} como matriz de escalonamento para M_{JJ}. Se a matriz de escalonamento E é usada, então é necessário efectuar a primeira iteração interior de modo diferente das seguintes. Como a direcção de busca passa a ser $E^{-1}r$ corre-se o risco de ter $z_s = 0$ e $p_s < 0$ ($r_s > 0$) para algum $s \in J$. Qualquer passo irá violar a restrição s, obtendo-se assim um passo nulo. Para contornar este problema

faz-se, na primeira iteração, $p^o = r^o$ pois assim o algoritmo vai-se deslocar para o interior.

Se não usarmos qualquer tipo de escalonamento (E = I) o algoritmo torna-se bastante mais simples e, nalguns casos, o seu desempenho não é pior do que o conseguido com $E \neq I$. Alguns autores têm também recomendado um escalonamento diagonal, $E = \text{diag} (m_{11} , m_{22} , \ldots , m_{nn})$, que pode ser visto como um processo intermédio entre o uso de um escalonamento e a sua não utilização. As nossas experiências com o algoritmo de O'Leary não foram conclusivas em relação à utilização do precondicionamento. Daí não o utilizarmos na implementação do processo cujo funcionamento prático é discutido na última secção deste capítulo. É ainda de notar que o método de O'Leary é facilmente extensível à resolução do BLCP através da determinação do ponto estacionário do programa quadrático (6.4). Ao algoritmo apresentado só é necessário introduzir alterações no cálculo de a_{mx}, de modo a garantir que o novo vector pertença à região admissível.

5 Métodos de Gradientes Projectados

5.1 Algoritmo de Dembo e Tulowitzki

Do nosso conhecimento, o primeiro deste tipo de algoritmos a ser utilizado para a resolução do programa quadrático com limites foi desenvolvido por Dembo e Tulowitzki [15]. Este processo utiliza direcções de gradiente conjugadas ou projectadas, tentando identificar simultaneamente um grande número de variáveis que terão valores iguais aos limites na solução óptima (restrições activas). Nesta secção iremos descrever o processo para a resolução do LCP, isto é para a determinação de

um ponto estacionário do programa quadrático (6.4). A forma do algoritmo para o BLCP é semelhante e não será apresentada. Para uma descrição do processo para resolução do LCP usamos as seguintes notações:

z^k aproximação do vector z na iteração k;

$Q(z^k) = q^T z^k + \frac{1}{2} (z^k)^T M z^k$ função objectivo;

$w^k = q + M z^k$ gradiente da função objectivo;

$A_k = \{ i : z_i^k = 0 \}$ conjunto das restrições activas;

$B_k = \{ i \in A_k : w_i^k \geq 0 \}$ restrições activas a que corresponde um multiplicador de Lagrange com o sinal correspondente ao óptimo;

$$(y_k^R)_i = \begin{cases} 0 \text{ se } i \in A_k \\ w_i^k \text{ se } i \notin A_k \end{cases}$$ gradiente reduzido;

$$(y_k^P)_i = \begin{cases} 0 \text{ se } i \in B_k \\ w_i^k \text{ se } i \notin B_k \end{cases}$$ gradiente projectado;

$z_i^{\#} = \max \{ z_i , 0 \}$ projecção do vector z na região admissível.

Então os passos do algoritmo são os seguintes:

Algoritmo de Dembo e Tulowitzki

Passo 0: Escolha $z_o \geq 0$; Faça $p_o = 0$; $k = 0$

Passo 1: Se $\| y_k^P \| \approx 0$ termine

Caso contrário determine a direcção p^k através do seguinte processo:

Determine $\eta_k = \min \{ \eta \| y_k^P \| , \| y_k^P \|^2 \}$

$$\text{Se } \| y_k^R \| \le \eta_k \text{ então } \begin{cases} \beta_k = \begin{cases} 0 & \text{se } B_k \neq A_{k-1} \\ \dfrac{(y_k^R)^T y_k^R}{(y_{k-1}^R)^T y_{k-1}^R} & \text{se } B_k = A_{k-1} \end{cases} \\ p_k = - y_k^P + \beta_k p_{k-1} \end{cases}$$

$$\text{Caso contrário } \begin{cases} \beta_k = \begin{cases} 0 & \text{se } A_k \neq A_{k-1} \\ \dfrac{(y_k^R)^T y_k^R}{(y_{k-1}^R)^T y_{k-1}^R} & \text{se } A_k = A_{k-1} \end{cases} \\ p_k = - y_k^R + \beta_k p_{k-1} \end{cases}$$

Passo2: Determine o passo α_k a partir de

$$\alpha_k^* = - \frac{(w^k)^T p^k}{(p^k)^T M p^k}$$

$$\alpha_k = \sigma^{m_k} \alpha_k^*$$

sendo m_k o primeiro inteiro não negativo tal que:

$$Q((z^k + \alpha_k p^k)^{\#}) - Q(z^k) \le \gamma \alpha_k (w^k)^T p^k$$

Passo3: $z^{k+1} = (z^k + \alpha_k p^k)^{\#}$

$k = k + 1$

e volte ao Passo 1.

Neste algoritmo existem parâmetros (σ , γ , η) cujo valor deve ser definido com cuidado, pois pequenas variações do seu valor podem comprometer a rapidez de convergência do algoritmo. Para garantir a convergência do método, os valores de σ e γ devem obedecer a $\sigma \in]\, 0 \,,\, 1 \,[$ e $\gamma \in]\, 0 \,,\, ½ \,[$. Em [15] são recomendadas as escolhas $\sigma = 0.6$, $\gamma = 0.1$ e $\eta = 0.03$. Esses são os valores que usamos na nossa experiência computacional que será apresentada na última secção.

5.2 Algoritmos de Yang e Tolle

Seguidamente iremos discutir dois outros algoritmos de gradientes projectados que, segundo os seus autores [83], têm algumas vantagens sobre os anteriormente descritos. Enquanto que o algoritmo de O'Leary não calcula a função objectivo mas, em contrapartida, só altera o conjunto das restrições activas em um elemento de cada vez, o algoritmo de Dembo e Tulowitzki altera muitos elementos no conjunto das restrições activas por iteração, mas, para a determinação do passo, tem que calcular a função objectivo pelo menos uma vez. Os algoritmos apresentados por Yang e Tolle procuram explorar as vantagens destes dois processos. Assim, não são feitas quaisquer avaliações da função objectivo, mas é possível alterar o conjunto das restrições activas em vários elementos por iteração. Seguidamente apresentamos os dois algoritmos que são propostos por Yang e Tolle. O primeiro é heurístico e possui grande rapidez, enquanto que o outro é mais lento mas é teoricamente convergente. Nesses algoritmos a direcção de busca é primeiramente calculada por técnicas de gradientes conjugados e depois o passo é calculado como a solução óptima do problema sem restrições. Finalmente, o vector assim obtido é projectado na região admissível. Os passos do primeiro algoritmo são os seguintes.

Algoritmo Heurístico de Yang e Tolle

Passo 0: $z^0 = 0 \;;\; k = 0$

Passo 1: $k = k + 1$

$w^k = M z^k + q$

$I = \{ i : z_i^k = 0 \wedge w_i^k \geq 0 \}$

$J = \{ 1, \ldots, n \} - I$

Se $\| w_J^k \| <$ termine: z^k e w^k são a solução.

Caso contrário vá para Passo 2.

Passo 2: (Resolução do sistema $M_{JJ} z_J = - q_J$)

$i = 0 \;;\; x^0 = z_J^k \;;\; p^0 = r^0 = - q_J - M_{JJ} x^0$

Passo3: $acg = \dfrac{(r^i)^T r^i}{(p^i)^T M_{JJ} p^i}$

$x^{i+1} = (x^i + a_{cg} p^i)^{\#}$

se $x^{i+1} \neq x^i + a_{cg} p^i$ faça

$I = I \cup \{ j : j \in J \wedge x_j^{i+1} = 0 \}$

$J = \{ 1, \ldots, n \} - I$

$z_J = x^{i+1}$

e volte a Passo 2.

Caso contrário faça

$r^{i+1} = r^i - a_{cg} M_{JJ} p^i$

e vá para Passo 4.

Passo 4: Se $\| r^{i+1} \| <$ faça $z_J^{k+1} = x^{i+1}$ e volte a Passo 1.

Caso contrário faça

$$b_i = \frac{(r^{i+1})^T r^{i+1}}{(r^i)^T r^i}$$

$$p^{i+1} = r^{i+1} + b_i\, p^i$$

$$i = i + 1$$

e volte a Passo 3.

Este algoritmo tem a vantagem de não calcular a função objectivo pois não há nenhuma busca do passo óptimo em cada iteração. No entanto, a sua convergência não é garantida teoricamente. Os autores apresentam um outro algoritmo, cuja convergência pode ser teoricamente estabelecida e que continua a ter as vantagens apontadas para o algoritmo heurístico, mas com um funcionamento mais lento. Nesse algoritmo é resolvido encadeadamente um conjunto de subproblemas de menor dimensão obtidos igualando a zero um conjunto menor ou maior de variáveis. Os problemas vão sendo gerados e empilhados e a pilha de problemas é resolvida do topo para a base. Cada novo problema tem mais variáveis fixas do que o anterior e, portanto, menor dimensão. Descreve-se seguidamente de modo genérico este algoritmo.

Algoritmo de Yang e Tolle

Passo 0: $z^0 = 0 \;;\; k = 0 \;;\; I_0 = L_0 = \varnothing$

Passo 1: $w^k = q + M z^k$

$K = \{ j : z_j^k = 0 \wedge w_j^k \geq 0 \}$

$I = I_k \cup K \qquad J = \{ 1, \ldots, n \} - I$

Se $\| w_j^k \| < \varepsilon$ então | se $k = 0$ termine $\rightarrow$ a solução é z^k

caso contrário faça $k = k - 1$ e volte a Passo 1.

Caso contrário vá para Passo 2.

Passo 2: $x^0 = z_J^k \;\; ; \;\; p^0 = r^0 = - w_J^k \;\; ; \;\; i = 0.$

Passo 3: $acg = \dfrac{(r^i)^T r^i}{(p^i)^T M_{JJ} p^i}$

$x^{i+1} = (x^i + a_{cg} p^i)^{\#}$

se x^{i+1} é obtido por projecção faça s = índice da primeira componente que é projectada. Faça $z_J^k = x^{i+1}$ e vá para Passo 4.

Caso contrário faça $r^{i+1} = r^i - a_{cg} M_{JJ} p^i$

Se $\| r^{i+1} \| < \varepsilon$ faça $z_J^k = x^{i+1}$ e vá para Passo 2.

Caso contrário faça $b_i = \dfrac{(r^{i+1})^T r^{i+1}}{(r^i)^T r^i}$

$p^{i+1} = r^{i+1} + b_i p^i$

$i = i + 1$

e repita o Passo 3.

Passo 4: $L_{k+1} = L_k \cup \{ s \}$

$I_{k+1} = I_s \cup \{ s \}$ com I_s tal que $I_k \subset I_s \subset I$

$k = k+1$

e volta a Passo 1.

Da maneira como se definem os conjuntos de índices I_k é fácil concluir que se tem sempre $I_k \subset I_{k+1}$. Assim, o conjunto de variáveis fixas em zero no problema de ordem k+1 contém sempre o conjunto de tais variáveis do problema de ordem k. Por isso a solução do problema k+1 é tomada como solução inicial do problema de ordem k.

A escolha do conjunto I_s condiciona a eficiência do algoritmo. Dois casos extremos podem ser considerados nomeadamente $I_s = I$ ou $I_s = L_m$. Fazendo $I_s = I$ obtém-se o maior número possível de variáveis fixas em zero e, consequentemente um problema com a menor dimensão possível para resolver. Com $I_s = L_m$ o conjunto I_{k+1} só terá mais uma componente do que o conjuto I_k e será formado pelos índices das primeiras componentes do vector z que se vão anulando ao fazer as sucessivas projecções. Tal como o algoritmo de Dembo e Tulowitzki, também estes processos podem ser facilmente estendidos à determinação de um ponto estacionário do programa quadrático (6.4) e assim ser utilizados para a solução do BLCP.

5.3 Algoritmo de Moré e Toraldo

Este algoritmo combina técnicas próprias dos métodos de restrições activas com as dos algoritmos de gradientes projectados [63]. O algoritmo de gradientes projectados é utilizado para identificar mais depressa o conjunto das restrições activas.

Seja $\Omega = \{ z \in R^n : z \geq 0 \}$ a região admissível. Se continuarmos a representar por Q a função objectivo ($Q(z) = \frac{1}{2} z^T M x + q^T z$), então o seu gradiente tem a forma $w = q + M z$. Neste método usam-se projecções do vector gradiente na região Ω que se definem do seguinte modo:

$$(\nabla_\Omega Q(z))_i = \begin{cases} w_i & \text{se } z_i > 0 \\ \min \{ w_i ; 0 \} & \text{se } z_i = 0 \end{cases}$$

Da maneira como este vector é definido é imediato que na solução z^* deverá ser $\nabla_\Omega Q(z^*) = 0$.

Cada iteração do algoritmo consiste em escolher um conjunto de restrições activas, que pode ser encarado geometricamente como uma face da região admissível, e, depois explorar cada uma das faces obtidas. Para a obtenção da face utiliza-se uma técnica de gradiente projectado. Para explorar cada face usa-se um método de gradiente conjugado. Além disso, deve-se modificar o processo que está em uso sempre que se não está a obter grande sucesso no decréscimo da função objectivo. Assim, muda-se de face, se se estiver a explorar uma determinada face sem grande sucesso. Por outro lado, se se estiver a utilizar direcções projectadas, então deve-se explorar uma nova face caso não haja grande modificação no valor da função objectivo.

O conjunto de variáveis activas na iteração k é, como habitualmente, definido por $A_k = \{ i : z_i^k = 0 \}$. Tal como foi feito anteriormente, $(x)^\#$ é a projecção do vector x na região admissível. Para a determinação do passo vai-se optimizar uma função $\Phi_k(\alpha)$ definida por $Q((z^k + \alpha\, d^k)^\#)$. O passo de ordem k deste algoritmo tem a forma

Algoritmo de Moré e Toraldo

Passo Geral 1: Faça $y^o = z^k$

Defina $y^{j+1} = (y^j - \alpha_j \nabla_\Omega Q(y^j))^\#$ sendo α_j obtido como se descreve no Passo Geral 2.

$$\text{Se} \begin{cases} Q(y^j) - Q(y^{j+1}) \leq \eta_1 \max \{ Q(y^i) - Q(y^{i+1}) \quad i<j \} \quad \eta_1 > 0 \\ \text{ou} \\ A_{j+1} = A_j \end{cases}$$

Faça $z^k = y^{j+1}$ e vá para Passo Geral 2.

Caso contrário continue no Passo Geral 1.

Passo Geral 2: Defina A_k e $J = \{ 1, \ldots, n \} - A_k$

Resolva $\min F(d_J) = \min \{ \frac{1}{2} d_J^T M_{JJ} d_J + (w_J^k)^T d_J \}$ pelo método dos gradientes conjugados substituindo o critério de paragem por:

$$F(d_J^{j-1}) - F(d_J^j) \leq \eta_2 \max \{ F(d_J^{j-1}) - F(d_J^j), j<i \} \quad \eta_2 > 0$$

Faça $d_J^k = d_J^j$ e $z^{k+1} = (z^k + \alpha_k d^k)^{\#}$

Sendo o valor de $\alpha_k > 0$ obtido de modo a que

$$\Phi_k(\alpha) \leq \Phi_k(0) + \mu \Phi_k'(\alpha) \quad \text{com } \mu \in]0, \frac{1}{2}[$$

Os valores de η_1, η_2 e de μ devem ser cuidadosamente considerados. Os valores de η_1 e η_2 controlam aquilo que se considera um decréscimo pouco aceitável na função objectivo. Sempre que a variação da função objectivo começar a ser pouco considerável o método abandona a estratégia que vinha seguindo. O valor de μ controla o decréscimo aceitável de $Q(z^{k+1})$ em relação a $Q(z^k)$. Se estes valores forem muito grandes corre-se o risco de andar sempre a mudar do Passo 1 para o Passo 2. Pelo contrário, se forem muito pequenos pode-se correr o risco de só sair de um dos passos ao fim de um grande número de iterações.

Embora com um formalismo diferente, este algoritmo é muito semelhante ao método descrito por Dembo e Tulowitzki, pelo que também é fácil estendê-lo à resolução do BLCP.

5.4 Algoritmo de Coleman e Hulbert

Em [6] é apresentado um algoritmo básico de restrições activas que é muito semelhante ao método de Keller. Consideram-se igualmente os conjuntos T das restrições activas e F das variáveis livres, tais que $F = \{ 1, \ldots, n \} - T$. Começando com uma solução admissível z que não seja óptima, procura-se uma direcção p através de

$$M_{FF}\, p = - g_F, \text{ com } g_F = q_F + M_{FF}\, z_F = (\nabla Q (z))_F.$$

e define-se $z (\alpha) = z + \alpha p$, com $\alpha \in [0, 1]$. O valor de α deve ser o maior possível sem violar qualquer restrição. Se $\alpha = 1$, então $z (\alpha)$ é admissível e estacionário, isto é, $g_F = 0$. Se $\alpha < 1$, então alguma ou algumas variáveis z_i, $i \in F$ vão-se anular para o valor máximo de α. Esses índices devem ser retirados de F e acrescentados a T. Sendo $\alpha < 1$ pode não ser $g_F = 0$, deve-se então repetir o processo até que seja $g_F = 0$. Encontrado um ponto estacionário averigua-se se ele é óptimo, isto é, se $g_T \geq 0$. Se não for óptimo considera-se $s \in T$ tal que $-q_s$ seja máximo, introduz-se esse índice em F e recomeça-se o processo.

Tal como no método de Keller, os conjuntos de índices F e T só são alterados em um elemento em cada iteração. Coleman e Hubert [6] apresentam algumas sugestões para melhorar este método que possibilitam transferir vários índices

simultaneamente entre F e T. Tal como em outros métodos já descritos neste capítulo, a ideia fundamental é projectar o ponto z (α) na região admissível, tomando para α o primeiro mínimo de Q (z (α)$^{\#}$). Daí se obtêm processos que combinam uma estratégia de restrições activas com algumas ideias já apresentadas em outros algoritmos, nomeadamente no método de Yang e Tolle. Dada a semelhança com esse processo não nos iremos debruçar mais atentamente sobre essas propostas de melhoria.

6 Método de Newton Global

Tal como referimos anteriormente o LCP (6.1) é equivalente ao sistema de equações não lineares

$$H_i (z) = \min (z_i , w_i) = 0 \quad , \quad i = 1, \ldots , n \qquad (6.9)$$

de tal modo que $\bar{z}$ é solução do LCP se e só se é solução do sistema de equações não lineares.

Nesse sentido não é de estranhar o desenvolvimento de um método de Newton Global para a resolução do LCP através da determinação de uma raiz do sistema (6.9). Para isso é recomendado o método de Newton em que se calcula a direcção de Newton em cada iteração e se usa um critério de Armijo que determina o passo $0 \leq \lambda_k \leq 1$ de modo a tornar o processo globalmente convergente.

Embora o operador H não seja diferenciável, existe derivada direccional H ' (z , d) em cada ponto z e direcção d. Para calcular esta derivada é necessário começar por definir os seguintes conjuntos de índices:

$$A = \{ i : w_i < z_i \}$$

$$B = \{ i : w_i = z_i \}$$
$$C = \{ i : w_i > z_i \}$$

e tem-se [31]

$$H'(z,d) = \begin{cases} (M_{i.})^T d & \text{se } i \in A \\ \min\{ (M_{i.})^T d \; ; \; d_i \} & \text{se } i \in B \\ d_i & \text{se } i \in C \end{cases}$$

O algoritmo de Newton Global aplicado ao LCP (6.1) pode ser sucintamente descrito do seguinte modo.

Dado z^k

Resolver $H(z^k) + H'(z^k, d^k) = 0$ para determinar d^k (6.10)

Calcular $\lambda_k = \mu^{m_k}$ com m_k o primeiro inteiro não negativo m tal que:

$g(z^k) - g(z^k + \mu^m d^k) \geq 2 \sigma \mu^m g(z^k)$ (6.11)

onde $\mu \in]0, 1[$, $\sigma \in]0, \frac{1}{2}[$ e $g(z) = \frac{1}{2} H(z)^T H(z)$.

Fazer $z^{k+1} = z^k + \lambda_k d^k$ e $k = k + 1$.

Repetir até obter convergência

A resolução da equação (6.10) é equivalente [31] à resolução do seguinte BLCP:

$$u = \hat{q} + \hat{M} v$$
$$u^T v = 0 \qquad (6.12)$$
$$v_B \geq 0 \; , \; u_B \geq 0, \; v_A \text{ sem restrição}$$

com $\hat{M} = \begin{bmatrix} M_{AA} & M_{AB} \\ M_{BA} & M_{BB} \end{bmatrix}$ e $\hat{q} = \begin{bmatrix} q_A \\ q_B \end{bmatrix}$.

Como as variáveis v_A são sem restrição terá que ser $u_A = 0$ e, assim, o BLCP (6.12) pode ser reduzido ao seguinte LCP com dimensão #B

$$u_B = (q_B - M_{BA} M_{AA}^{-1} q_A) + (M_{BB} - M_{BA} M_{AA}^{-1} M_{AB}) v_B$$

$$v_B \geq 0 \; ; \; u_B \geq 0 \; ; \; u_B^T v_B = 0 \qquad (6.13)$$

desde que M_{AA} seja não singular. A matriz $M_{BB} - M_{BA} M_{AA}^{-1} M_{AB}$ é o Complemento de Schur de M_{BB} em $\hat{M}$ e portanto pertence à classe P, desde que M seja P.

Uma vez resolvido o LCP (6.13) obtém-se facilmente $v_A = M_{AA}^{-1} (- q_A - M_{AB} v_B)$. De notar que se $B = \varnothing$ a determinação de v_A se reduz à resolução do sistema $q_A + M_{AA} v_A = 0$.

O conjunto B corresponde aos índices das componentes degeneradas do vector z. É possível provar [12] que H é diferenciável se $B = \varnothing$. Além disso se tal acontecer o BLCP reduz-se a um sistema de equações

$$M_{AA} v_A = - q_A$$

Por isso, a estratégia desenvolvida em [31] procura manter $B = \varnothing$ em cada iteração. Como iremos ver mais adiante tal é conseguido em geral.

Continuando a descrição do algoritmo, após a obtenção dos vectores u e v determina-se a direcção d^k a partir de

$$\begin{cases} d_A^k = v_A - z_A^k \\ d_B^k = v_B - z_B^k \\ d_C^k = - z_C^k \end{cases}$$

Se agora definirmos um novo vector $v = (v_A , v_B , 0)$, pode-se escrever abreviadamente

$$d^k = v - z^k$$

Seja $u = q + Mv$ e $w^k = q + Mz^k$. Então, se λ_k é o passo obtido pelo critério de Armijo, tem-se

$$\left.\begin{aligned} w^{k+1} &= (1 - \lambda_k) w^k + \lambda_k u \\ z^{k+1} &= (1 - \lambda_k) z^k + \lambda_k v \end{aligned}\right\} \qquad (6.14)$$

Subtraindo estas duas relações obtém-se

$$z^{k+1} - w^{k+1} = (1 - \lambda_k) (z^k - w^k) + \lambda_k (v - u) \qquad (6.15)$$

Analise-se agora as transformações que podem ocorrer nos conjuntos de índices A, B e C. Se o conjunto B for não vazio, pelo menos um índice abandonará B. Como u e v são não negativos e complementares, a partir da relação (6.15) conclui-se que, como $u_i \neq v_i$ para pelo menos um i de B, então também existirá um i de B tal que $z_i^{k+1} \neq w_i^{k+1}$. Assim, na iteração seguinte, um outro par (u , v) será determinado.

Considere-se agora que $i \in A$. Então $z_i^k - w_i^k > 0$ e $u_i = 0$. Se $v_i \geq 0$ o índice i permanecerá em A. Caso contrário, tudo depende do valor que for atribuido a λ_k. Analisando a relação (6.15) para o caso de i ser um elemento de A conclui-se que se for $\lambda_k < \rho_1$ com

$$\rho_1 = \min \{ \frac{z_i^k - w_i^k}{z_i^k - w_i^k - v_i} \ : \ i \in A \ \wedge \ v_i < 0 \}$$

nenhum elemento deixará o conjunto A. Por outro lado se $i \in C$ será $z_i^k - w_i^k < 0$ e $v_i = 0$. Se $u_i \geq 0$ o índice i permanecerá em C. Caso contrário, se for $\lambda_k < \rho_2$ com

$$\rho_2 = \min \{ \frac{w_i^k - z_i^k}{w_i^k - z_i^k - u_i} \ : \ i \in C \ \wedge \ u_i < 0 \}$$

também nenhum elemento deixará o conjunto C.

Considere-se agora que $B = \varnothing$. As relações (6.14) mostram que o algoritmo vai encontrar um novo ponto no segmento definido por (w^k , z^k) e (u , v). Se for $\lambda_k < \rho_1$ e $\lambda_k < \rho_2$ os conjuntos A e C não serão alterados e, portanto, o par (u , v) será sempre o mesmo. Corre-se assim o risco de obter pontos sempre no mesmo segmento e de nunca se chegar à solução. Um processo de impedir que o método possa falhar é forçar a que seja $\lambda_k > \rho$ com $\rho = \min \{ \rho_1 , \rho_2 \}$. Note-se que fazer $\lambda_k = \rho$ equivale a tirar um só índice de A ou de C que passará para B. É possível mostrar [31] que, se z^k não é solução do LCP (6.1) e é não degenerado, então existe $\delta \in R^+$ tal que para $\lambda \in] \rho , \rho + \delta [$ o vector $z^\lambda = z^k + \lambda d^k$ é não degenerado e obedece ao critério (6.11).

Então, partindo de um z^o não degenerado, é possível construir uma sucessão $\{ z^k \}$ de vectores não degenerados, isto é, tal que $B = \varnothing$ em cada iteração. Assim sendo o algoritmo de Newton Global pode ser descrito do seguinte modo:

Algoritmo de Newton Global

Passo 0: Escolha z^o vector não degenerado.

$k = 0$.

Defina A e C.

Passo 1: Resolva o sistema $q_A + M_{AA} v_A = 0$.

Defina $v = (v_A , 0)$ e $d^k = v - z^k$.

Se v é solução do LCP termine.

Caso contrário vá para Passo 2.

Passo 2: Determine ρ.

Faça $\lambda_k = \rho + \varepsilon$ com $\varepsilon > 0$ tal que

$z^{k+1} = z^k + \lambda_k d^k$ obedece a (6.11).

Faça k = k + 1.

Redifina A e C.

Volte a Passo 1.

A determinação do valor de ε no passo 2 pode ser feita começando por fazer $\lambda_k = \frac{1+\rho}{2}$ (pois deve ser $\rho < \lambda_k < 1$). Se este valor não satisfizer o critério (6.11) tenta-se o ponto médio do intervalo] ρ , λ_k [e continua-se a bissectar o intervalo até que o critério (6.11) seja obedecido. A experiência mostra que, muitas vezes, o valor $\lambda_k = 1$ satisfaz o critério (6.11) e que tomar esse valor, sempre que possível, acelera o processo. Por isso, nas experiências que fizemos alterámos o que é recomendado no artigo e tentámos sempre usar para primeira tentativa o valor $\lambda_k = 1$ e, só no caso de esse valor não obedecer a (6.11), prosseguir como foi explicado acima. Nesse caso, o algoritmo tem convergência superlinear, pois a função H é diferenciável [12].

Se o algoritmo começar com $w^o = q$ e $z^o = 0$, o que é sempre possível se nenhuma componente do vector q for nula, o primeiro conjunto A coincide com o conjunto inicial F do algoritmo pivotal por blocos. Além disso, se $\lambda_k = 1$ em todas as iterações, este algoritmo e o algoritmo bloco pivotal percorrem o mesmo caminho. Portanto o algoritmo pivotal por blocos com $p = \infty$ tem convergência local quadrática. Isso confirma a rapidez do processo para a obtenção da solução do LCP. A utilização da constante p nesse processo aparece assim como uma estratégia de globalização do método pivotal por blocos.

Finalmente, é ainda de acrescentar que nos parece ser possível estender o algoritmo de Newton Global ao BLCP. Isso será, com certeza, assunto de investigação futura.

7 Métodos de Pontos Interiores

Desde que o algoritmo de Karmakar para a solução do Problema Linear simples foi proposto, vários autores têm tentado utilizar algo de semelhante para o LCP. Estes processos são iterativos, sendo desenvolvidos de modo a que os seus iterandos percorram o interior da região admissível. Daí serem denominados por algoritmos de ponto interior. Dum modo geral, estes métodos podem ser encarados como métodos tipo Newton descendentes, em que só são usados vectores com todas as componentes positivas. Na nossa exposição iremos seguir o tratamento apresentado em [50], no qual tem um papel fundamental o PQ equivalente

$$\begin{aligned} \text{Min} \quad & z^T w \\ \text{s. a} \quad & w = q + M z \\ & z \geq 0 \ ; \ w \geq 0 \end{aligned} \tag{6.16}$$

Associado ao PQ (6.16) considera-se ainda um outro problema quadrático em que nenhuma variável se pode anular:

$$\begin{aligned} \text{Min} \quad & z^T w \\ \text{s. a} \quad & w = q + M z \\ & z > 0 \ ; \ w > 0 \end{aligned} \tag{6.17}$$

e a chamada função potencial

$$f(z, w) = (n + \rho) \log (z^T w) - \sum_{i=1}^{n} \log(z_i w_i) - n \log n \qquad (6.18)$$

com $\rho \in R^+$ um parâmetro a escolher. De notar que, enquanto que a parcela $(n + \rho) \log (z^T w)$ de (6.18) está associada à função objectivo de (6.17), $- \sum_{i=1}^{n} \log(z_i w_i)$ funciona como uma barreira que impede que os pontos se aproximem da periferia da região admissível. A função potencial (6.18) pode ainda ser escrita com o aspecto:

$$f(z, w) = (n + \rho) \log (z^T w) + n \log \frac{z^T w / n}{(\prod_{i=1}^{n} z_i w_i)^{1/n}} \qquad (6.19)$$

A segunda parcela de (6.19) é sempre não negativa, pois a média aritmética de números reais positivos é sempre maior do que a sua média geométrica, só se verificando a igualdade no caso de todos os números serem iguais. Assim será

$$f(z, w) \geq \rho \log (z^T w)$$
$$\text{se } w = q + M z \; ; \; w > 0 \; ; \; z > 0 \qquad (6.20)$$

Deste modo qualquer ponto (z, w) que satisfaça (6.20) será solução de (6.16) desde que $f(z, w)$ seja suficientemente pequeno.

A sucessão de pontos $\{ (z^k, w^k) \}$ é gerada com recurso a uma família de direcções de Newton $(\Delta z, \Delta w)$ que são obtidas via solução dos sistemas:

$$\begin{bmatrix} W & Z \\ -M & I \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta w \end{bmatrix} = \begin{bmatrix} \frac{\beta}{n} (z^k)^T w^k \, e - Z w^k \\ O \end{bmatrix} \qquad (6.21)$$

sendo $Z = \text{diag}(z_1^k, \ldots, z_n^k)$, $W = \text{diag}(w_1^k, \ldots, w_n^k)$ e $\beta \in [0, 1]$

Finalmente, há que calcular o passo admissível θ_k para que o novo ponto (z^{k+1} , w^{k+1}) continue a obedecer às condições (6.20). Normalmente determina-se θ_k a partir de

$$\theta_k = \sigma \min \left\{ \min \{ -\frac{z_i^k}{\Delta z_i} : \Delta z_i < 0 \} , \min \{ -\frac{w_i^k}{\Delta w_i} : \Delta w_i < 0 \} \right\} \qquad (6.22)$$

com $\sigma \in] 0 , 1 [$ (normalmente $\sigma = 0.99995$).

O algoritmo UIP pode então ser descrito abreviadamente na seguinte forma:

Algoritmo UIP

Passo 0: $k = 0$; $\varepsilon > 0$; (z^o , w^o) : $z^o > 0$, $w^o > 0$, $w^o = q + M z^o$.

Passo 1: Se $(z^k)^T w^k < \varepsilon$ páre: a solução aproximada foi encontrada
Caso contrário determine $(\Delta z , \Delta w)$ por solução do sistema (6.21).

Passo 2: Escolha θ_k pelo critério (6.22).

Passo3: $(z^{k+1} , w^{k+1}) = (z^k , w^k) + \theta_k (\Delta z , \Delta w)$
Faça k= k + 1 e volte a Passo 1.

Este algoritmo fornece uma solução aproximada $(\bar{z} , \bar{w})$ do problema quadrático (6.18). A partir daí a solução (z*, w*) do LCP (6.1) pode ser obtida usando o critério:

$$z_i^* = 0 \quad \text{se} \quad \bar{z}_i < \delta \quad i = 1 , \ldots , n$$
$$w_i^* = 0 \quad \text{se} \quad \bar{w}_i < \delta \quad i = 1 , \ldots , n$$

em que δ é uma tolerância para zero que deve estar relacionada com o ε escolhido no passo zero do algoritmo e, também, com a precisão da máquina em que se trabalha.

A convergência deste algoritmo é assegurada para LCPs monótonos e estritamente monótonos, desde que o ponto inicial seja admissível [50]. A obtenção do ponto inicial (z^o ,w^o) é um dos factores que mais condiciona a eficiência desse processo. Várias abordagens têm sido propostas [20] em que a obtenção do ponto inicial está dependente da solução de um LCP de dimensão superior a n. Um processo mais eficiente consiste em relaxar a condição $w^o = q + M z^o$ e limitar-se a iniciar com um vector (z^o ,w^o) que tenha todas as componentes simultaneamente positivas. Nesse caso $w^k \neq q + M z^k$ e o algoritmo UIP deve ser ligeiramente alterado para lidar com esse facto. Assim, no passo 1 o critério de paragem deve incluir também a condição $\| w^k - q - M z^k \| < \varepsilon$. Além disso, no sistema (6.21) o termo independente passará a ser

$$\begin{bmatrix} \frac{\beta}{n} (z^k)^T w^k \, e - Z \, w^k \\ \\ - w^k + q + M z^k \end{bmatrix}$$

Recentemente [20] foi desenvolvido um método híbrido que utiliza um algoritmo de ponto interior conjuntamente com o algoritmo bloco pivotal. A ideia é usar o algoritmo de ponto interior para fornecer um bom conjunto inicial F para o algoritmo bloco pivotal. É uma técnica que, embora ainda se encontre a ser desenvolvida, tem conduzido a resultados bastante encorajadores. Sugerimos [20] para um bom estudo computacional da eficiência desse tipo de processos para a resolução de LCPs monótonos.

A extensão dos algoritmos de ponto interior à resolução do BLCP é possível de fazer [50]. Contudo, esse processo ainda precisa de ser refinado, de modo a torná-lo mais eficiente. Esse assunto e o desenvolvimento de um algoritmo híbrido na linha do processo referido atrás merecerão certamente investigação futura.

A implementação dos métodos iterativos é extremamente simples. Como não há necessidade de factorizar qualquer matriz, não é preciso utilizar técnicas de ordenação de matrizes. Os produtos matriz vector a efectuar por iteração, são feitos tirando partido da esparsidade da matriz e do vector, utilizando os procedimentos descritos em [77]. Além disso, como as matrizes são simétricas, é possível, sem grande aumento da complexidade das implementações, guardar exclusivamente um dos triângulos da matriz. É assim possível lidar com problemas de dimensão e/ou densidade superior às que são permitidas pelos métodos directos. Como se verá na secção seguinte, estas vantagens não vão ser extensíveis à rapidez e precisão da solução obtida.

Como referimos anteriormente, em certos casos é conveniente usar uma técnica de precondicionamento. Nesse caso a direcção de busca p é dada pela resolução do sistema $E\,p = r$. Normalmente a matriz E é escolhida de modo a que esses sistemas sejam de fácil solução e que, ao mesmo tempo, não difira muito da matriz M. Um processo que vem sendo sugerido é a utilização da factorização de Cholesky incompleta em que $E = \tilde{L}\,\tilde{L}^{T}$, com $\tilde{L}$ a matriz que se obtém usando as expressões da factorização de Cholesky mas rejeitando os elementos não nulos originados na factorização. Claro que este tipo de precondicionamento implica uma prévia ordenação de matriz M. Em [3] é sugerido que se rejeitem os elementos que sejam, em valor absoluto, inferiores ao produto de um determinado factor ψ pelos elementos da diagonal da linha e da coluna correspondente a esse elemento. Conforme o valor atribuído a ψ assim é maior ou menor a esparsidade da matriz $\tilde{L}$. Outros autores [47] sugerem que se rejeitem os elementos que vão corresponder a posições em que na matriz M havia zeros. Obtém-se assim uma matriz $\tilde{L}$ com o padrão de esparsidade do triângulo inferior da matriz M. Para este tipo de precondicionamento torna-se mais eficiente usar uma técnica de ordenação da matriz

diferente da usual que não entre em consideração com os possíveis enchimentos. Em qualquer dos casos o processo é equivalente a factorizar $M + C = \tilde{L}\,\tilde{L}^T$ com C uma matriz semi-definida positiva simétrica. A matriz C não é explicitamente determinada, sendo só considerada a sua influência na factorização. De facto, de cada vez que se rejeita um elemento em $\tilde{L}$ é criado um elemento em C o que irá alterar a factorização a partir daí.

A implementação do algoritmo Newton Global é semelhante à que foi descrita para o algoritmo bloco pivotal, pelo que não nos iremos alongar em mais considerações sobre o assunto.

No algoritmo UIP é necessário algum cuidado na resolução do sistema (6.20). Efectuando os cálculos, chega-se à conclusão que:

$$\begin{cases} (M + Z^{-1}W)\,\Delta z = \dfrac{\beta}{n}(z^k)^T w^k e - w^k \\ \Delta w = \dfrac{\beta}{n}(z^k)^T w^k Z^{-1} e - w^k - Z^{-1} W \Delta z \end{cases}$$

Assim, em cada iteração é necessário resolver um sistema da forma

$$(M + Z^{-1}W)\,\Delta z = p^k \qquad (6.23)$$

Como as matrizes Z e W são diagonais, as matrizes M e $M + Z^{-1}W$ têm sempre o mesmo padrão de não zeros, só diferindo entre si nos elementos da diagonal. Assim, para resolver eficazmente o sistema (6.23), bastará ordenar e factorizar simbolicamente a matriz M no inicio e, em cada iteração efectuar a factorização numérica de $M + Z^{-1}W$. De notar que se a matriz M for PD ou PSD, $M + Z^{-1}W$ será PD, pelo que a factorização LDL^T ou LU pode sempre ser encontrada.

9 EXPERIÊNCIA COMPUTACIONAL

Alguns dos algoritmos mencionados neste capítulo foram testados na resolução de alguns LCPs estritamente monótonos. As experiências foram mais uma vez efectuadas no computador CDC CYBER 180-830 da Universidade do Porto.

Os LCPs resolvidos nestas experiências foram já descritos nos capítulos 2 e 3 e são apresentados com a mesma notação. Além desses problemas considerámos ainda os LCPs referenciados com as siglas TP36, TP37 e TP38. A matriz de TP36 foi gerada de acordo com [83] e é PD simétrica de dimensão 1600. As matrizes de TP37 e TP38 são pentadiagonais simétricas, construídas de acordo com [57] e de dimensão 100 e 400 respectivamente.

Na tabela 6.1 apresenta-se uma comparação entre os algoritmos de Keller e Bloco já descritos em capítulos anteriores e os algoritmos de O'Leary, Dembo e SOR apresentados neste capítulo. Estes processos foram os escolhidos por os considerarmos mais representativos de cada tipo de algoritmo apresentado. Nessa tabela, além dos parâmetros já descritos nos capítulos anteriores, aparecem ainda as seguintes notações

NEG = número de componentes negativas no vector q

ε = tolerância para o critério de paragem (comparação da norma de r com 0)

PAR = parâmetro η do método de Dembo e Tulowitzki e parâmetro ω do algoritmo SOR projectado

Além disso, nesta tabela e nas seguintes NO indica o número de multiplicações e divisões multiplicado por 10^{-5}.

Finalmente, a indicação NC que aparece nalguns casos significa que o algoritmo não foi capaz de resolver o LCP em menos de 5000 segundos de CPU. É de notar ainda que experimentámos vários valores para os parâmetros η e para ω, sendo os resultados referentes aos melhores desempenhos dos algoritmos.

Da análise da tabela 6.1 chega-se imediatamente à conclusão, já esperada, que a precisão dos métodos iterativos é, de um modo geral, inferior à dos métodos directos. Com efeito, o resíduo para esses métodos é, em geral, bastante superior ao obtido com os métodos directos. Além disso, verifica-se que o algoritmo Bloco é mais eficiente para quase todos os problemas. Os métodos de gradientes e SOR projectados parecem ser influenciados pelo número de componentes negativas de q e pelo número de variáveis z_i básicas na solução do LCP, mas esses factores são menos condicionantes que nos métodos pivotais simples. O número de condição da matriz tem particularmente uma grande influência no desempenho desses métodos. Isso é muito notório nos resultados obtidos com as matrizes pentadiagonais. É de notar que, para essas matrizes o número de condição aumenta com a dimensão da matriz. Os resultados mostram que os algoritmos resolvem os LCPs até n = 100, mas a partir dessa dimensão tornam-se totalmente inoperantes. Assim, podemos concluir que estes algoritmos iterativos não são robustos e são, em geral, bastante inferiores ao método pivotal por blocos. Contudo, a sua utilidade pode ser bastante interessante em alguns problemas bem condicionados e que possuam algum tipo de estrutura. Isto é aliás ilustrado no problema TP8.

A tabela 6.2 apresenta uma comparação entre o método pivotal por blocos e o método Newton Global. Todos os problemas foram gerados de modo a que nenhuma componente do vector q seja nula. Desta forma o algoritmo de Newton Global nunca terá de lidar com soluções degeneradas. Nessas condições, os algoritmos Bloco e Newton Global têm comportamentos bastante semelhantes à excepção do problema

PROBLEMA	N	NEG	VA		BLOCO	KELLER	O'LEARY	DEMBO	SOR
TP1	484	151	242	NI	6	242	1319	279	287
				NO	2.12	5.77	90.8	45.7	19.2
				TT	9.2	22.2	138.2	65.2	40.8
				RES	1-11	2-11	1-6	8-4	1-3
				ε			1-6	1-3	1-4
				PAR				+∞	1.5
TP8	600	303	300	NI	3	300	7	14	614
				NO	0.87	6.44	1.69	2.08	44.3
				TT	10.1	24.3	3.7	4.9	91.6
				RES	3-12	7-12	7-12	3-12	2-7
				ε			1-6	1-6	1-6
				PAR				0.03	1.7
TP4	1000	631	500	NI	7	550	504	305	175
				NO	1.24	16.8	88.9	87.2	20.9
				TT	9.7	61.8	140.8	128.5	43.5
				RES	2-12	2-12	1-6	1-4	1-4
				ε			1-6	1-4	1-4
				PAR				+∞	1.7
TP22	1600	947	800	NI	7	832	319	255	139
				NO	0.62	16.2	70.3	55.2	15.3
				TT	8.1	85.7	125.7	101.2	29.3
				RES	1-12	1-12	6-7	1-4	7-5
				ε			1-6	1-3	1-4
				PAR				+∞	1.3

Tabela 6.1

PROBLEMA	N	NEG	VA		BLOCO	KELLER	O'LEARY	DEMBO	SOR
TP20	1280	326	640	NI	4	640	98	86	85
				NO	1.5	21.8	10.9	11.3	6.53
				TT	7.82	76.2	18.3	24.3	12.1
				RES	1-12	2-12	8-7	9-5	1-9
				ε			1-6	1-3	1-6
				PAR				+∞	1.7
TP21	1600	407	800	NI	4	800	89	88	92
				NO	1.87	28.4	14.3	14.5	8.83
				TT	9.85	111.6	23.9	31.2	16.3
				RES	2-12	2-12	8-7	9-5	1-9
				ε			1-6	1-3	1-6
				PAR				+∞	1.7
TP23	100	23	50	NI	8	50	79	67	1695
				NO	0.03	0.16	2.0	1.51	18.09
				TT	0.14	0.53	1.57	1.32	16.4
				RES	7-13	1-12	6-7	1-5	1-10
				ε			1-6	1-4	1-6
				PAR				+∞	1.7
TP19	1500	357	750	NI	67	829	NC	NC	NC
				NO	10.5	149.2			
				TT	31.9	271.1			
				RES	4-12	5-12			
				ε					
				PAR					

Tabela 6.1 (cont.)

TP38. Para este problema nunca é possível fazer $\lambda_k = 1$, pois este valor nunca obedece à condição (6.11) e vários valores de λ_k têm que ser testados até se conseguir ultrapassar a condição (6.11). Note-se que, apesar de nesse problema o algoritmo Bloco ter um comportamento pior do que o habitual, ele funciona bastante melhor que o algoritmo de Newton Global. Por outro lado, facilmente se conclui ser o método pivotal por blocos bastante mais simples de estender e de se generalizar para o BLCP. Estas considerações levaram à nossa preferência pelo algoritmo pivotal por blocos para a resolução de LCPs e BLCPs estritamente monótonos de grandes dimensões. No entanto, será interessante fazer no futuro um estudo mais aprofundado do método de Newton Global em que outras técnicas de pesquisa unidimensional pudessem ser utilizadas ou pelo menos investigadas.

PROBLEMA	N	NEG	VA	BLOCO		DAMPED-NEWTON	
				NI	TT	NI	TT
TP1	484	268	242	6	9.4	5	8.7
		297	98	5	12.6	5	12.8
TP8	600	314	300	3	10.0	3	10.0
		450	120	3	12.2	3	12.3
TP5	1500	921	750	5	6.8	7	9.6
		1072	300	10	12.3	10	17.1
TP12	1000	478	500	6	1.4	5	2.4
		503	200	5	2.8	5	2.0
TP38	400	145	193	19	1.8	21	3.7
		331	400	223	46.8	696	432.7

Tabela 6.2

CAPÍTULO VII

APLICAÇÃO À PROGRAMAÇÃO QUADRÁTICA ESTRITAMENTE CONVEXA

1 INTRODUÇÃO

O programa quadrático estritamente convexo, na sua forma mais geral, pode ser definido como

$$\begin{array}{ll} \text{Min} & q^T z + \frac{1}{2} z^T M z \\ \text{sujeito a} & A z = b \\ & E z \geq f \\ & z \geq 0 \end{array} \qquad (7.1)$$

onde M é uma matriz simétrica PD de ordem n, $A \in R^{m \times n}$, $E \in R^{k \times n}$, $b \in R^m$ e $f \in R^k$. As condições de optimalidade de (7.1) constituem o seguinte problema linear complementar [12]

$$\begin{bmatrix} y \\ 0 \\ \gamma \end{bmatrix} = \begin{bmatrix} q \\ -b \\ -f \end{bmatrix} + \begin{bmatrix} M & -A^T & -E^T \\ A & O & O \\ E & O & O \end{bmatrix} \begin{bmatrix} z \\ u \\ \lambda \end{bmatrix} \qquad (7.2)$$

$$y, z, \lambda, \gamma \geq 0; \; y^T z = \lambda^T \gamma = 0.$$

Como M $\in$ PD então é não singular e é possível efectuar uma operação pivotal principal com pivot M, obtendo-se o seguinte problema

$$\begin{bmatrix} z \\ 0 \\ \gamma \end{bmatrix} = \begin{bmatrix} -M^{-1}q \\ -b - A M^{-1}q \\ -f - E M^{-1}q \end{bmatrix} + \begin{bmatrix} M^{-1} & M^{-1}A^T & M^{-1}E^T \\ A M^{-1} & A M^{-1}A^T & A M^{-1}E^T \\ E M^{-1} & E M^{-1}A^T & E M^{-1}E^T \end{bmatrix} \begin{bmatrix} y \\ u \\ \lambda \end{bmatrix} \qquad (7.3)$$

$$y, z, \lambda, \gamma \geq 0 \, , \, y^T z = \lambda^T \gamma = 0$$

É de notar que, se A e E têm característica completa, então a matriz deste problema é simétrica PSD com blocos diagonais PD. Além disso, as variáveis u não têm restrição de sinal. Portanto estamos em presença de um BLCP monótono, o qual pode ser resolvido por um dos algoritmos discutidos nos capítulos anteriores. No entanto, a matriz deste LCP é muito densa. Além disso é necessário obter a matriz M^{-1} e calcular os produtos matriciais com essa matriz de modo a conhecer os dados desse problema. Por isso, há que tentar usar algoritmos que trabalhem sobre o LCP (7.2) efectuando operações equivalentes às realizadas pelos algoritmos para LCPs simétricos quando aplicados a (7.3). Assim, a resolução de um programa estritamente convexo a partir da resolução do BLCP (7.2) (equivalentemente (7.3)) só é conveniente de fazer em alguns casos especiais. Como principais exemplos em que esse tipo de redução é interessante citamos os seguintes casos:

(i) M é uma matriz diagonal PD.

(ii) M é uma matriz simétrica PD e existe apenas uma igualdade e as restrições de desigualdade são limites.

Neste capítulo iremos dedicar atenção a estes dois casos, mostrando que os programas quadráticos podem ser resolvidos por adaptação simples dos métodos pivotais principais discutidos nos capítulos anteriores. Em particular, o método pivotal

principal por blocos é recomendado na resolução desses PQs estritamente convexos quando a dimensão é elevada.

2 PROGRAMA QUADRÁTICO SEPARADO ESTRITAMENTE CONVEXO NO SIMPLEX

Este tipo de programa pode ser formulado como

$$\begin{array}{ll} \text{Min} & q^T z + \frac{1}{2} z^T M z \\ \text{sujeito a} & e^T z = b_o \\ & z \geq 0 \end{array} \qquad (7.4)$$

em que b_o é um número real positivo, e é um vector de elementos positivos e $M = \text{diag}(m_{11}, \ldots , m_{nn})$, $m_{ii}, > 0$, $i = 1, \ldots , n$. Sem perda de generalidade pode-se considerar $b_o = 1$ e $e^T = (1, \ldots , 1)$, pois, basta algumas mudanças lineares de variável para reduzir o problema a esse caso.

As condições de optimalidade de (7.4) constituem o seguinte BLCP

$$\begin{aligned} & w = q + M z + \lambda e \\ & 0 = 1 - e^T z \\ & w \geq 0, z \geq 0, w^T z = 0, \lambda \in R. \end{aligned} \qquad (7.5)$$

Dada a estrutura muito especial do problema é possível desenvolver algoritmos bastante eficientes para a resolução deste problema complementar. Seguidamente iremos discutir os algoritmos mais importantes para esse efeito.

Apresentamos também um estudo comparativo da eficiência desses processos para a resolução de alguns programas quadráticos da forma (7.4).

2.1 Método de Pardalos e Kovoor

Este algoritmo foi proposto em [75] e procura resolver o PQ (7.4) a partir de um outro problema equivalente. Considerando a mudança de variável $z_i = \frac{2x_i - q_i}{m_{ii}}$ tem-se

$$f(z) = q^T z + \tfrac{1}{2} z^T M z = \sum_{i=1}^{n} (q_i + \tfrac{1}{2} m_{ii} z_i) z_i =$$

$$= \sum_{i=1}^{n} (q_i + \tfrac{1}{2} m_{ii} \frac{2x_i - q_i}{m_{ii}}) \frac{2x_i - q_i}{m_{ii}} =$$

$$= \sum_{i=1}^{n} \frac{(q_i + 2x_i)(2x_i - q_i)}{2m_{ii}} = \sum_{i=1}^{n} \frac{4x_i^2 - q_i^2}{2m_{ii}}$$

Daqui se conclui que minimizar em z a função f(z) é equivalente a minimizar em x a função $g(x) = x^T C x$ com $C = 2\,M^{-1}$.

Por outro lado, $z_i \geq 0$ implica que $x_i \geq \frac{q_i}{2}$ (i = 1, ... , n). Além disso, de $\sum_{i=1}^{n} z_i = 1$ vem $\sum_{i=1}^{n} \frac{2}{m_{ii}} x_i = 1 + \sum_{i=1}^{n} \frac{q_i}{m_{ii}}$. Estas relações conduzem ao seguinte problema

$$\begin{aligned} &\min g(x) = x^T C x \\ &\text{sujeito a } x \geq a \\ &\qquad c^T x = d \end{aligned} \tag{7.6}$$

onde $a = \frac{1}{2}q$, $d = 1 + q^T M^{-1} e$, $C = 2M^{-1}$ e $c = Ce$. As condições de Kuhn-Tucker de (7.6) podem ser escritas na seguinte forma

$$c^T x = d$$

$$a - x \leq 0$$

$$2Cx + \lambda c - \mu = 0$$

$$\mu^T (a - x) = 0$$

$$\mu \geq 0$$

Explorando estas condições é possível provar [75] que qualquer x pertencente ao conjunto admissível de (7.6) é mínimo de g se existir y tal que

$$i = 1, \ldots, n, \quad x_i(y) = \begin{cases} a_i & \text{se } y < a_i \\ y & \text{se } y \geq a_i \end{cases}$$

Além disso, $y = -\frac{\lambda}{2}$.

Para cada i, $x_i(y)$ é monótona não decrescente linear e diferenciável por pedaços com ponto crítico a_i. Definindo $h(y) = \sum_{i=1}^{n} c_i x_i(y)$, esta função também é monótona não decrescente, linear e diferenciável por pedaços, com pontos críticos $\{ a_1, \ldots, a_n \}$. Para resolver o problema basta encontrar y^* tal que $h(y^*) = d$. O cálculo de y^* pode ser feito por pesquisa binária sobre o conjunto dos pontos críticos. A base do método é escolher um intervalo [Min, Max] que contenha y^* e, por tentativas, ir estreitando esse intervalo. Em cada iteração o conjunto dos pontos críticos passa a ser constituído exclusivamente pelos valores de a_i contidos no

intervalo] Min, Max [. Se $a_i < Min$, então $y^* > a_i$ e $x_i(y^*) = y^*$. Por outro lado, se $a_i > Max$, tem-se $y^* < a_i$ e $x_i(y^*) = a_i$. Deve-se assim considerar separadamente os três conjuntos de índices $A = \{ i : a_i \leq Min \}$, $B = \{ i : a_i \geq Max \}$ e $C = \{ i : Min < a_i < Max \}$. Começando com $Min = - \infty$ e $Max = +\infty$ escolhe-se um qualquer y do conjunto dos pontos críticos. Calcula-se h(y). Se for h(y) > d faz-se Max = y. Se for h(y) < d então Min = y. Claro que se for h(y) = d então $y^* = y$ e o processo termina. À medida que o intervalo] Min, Max [se vai estreitando, vão sair índices do conjunto C para A e para B. Quando o processo termina tem-se, obviamente, $C = \varnothing$.

A escolha de y pode ser feita completamente ao acaso. Em [75] é sugerido que se tome para y a mediana do conjunto dos pontos críticos. A prática mostra que normalmente tal escolha reduz o número de iterações mas aumenta o esforço computacional, pois o cálculo da mediana de um conjunto discreto pressupõe uma ordenação prévia. Assim optámos por escolher y como o primeiro elemento do conjunto não ordenado dos pontos críticos e obtivemos resultados bastante melhores em tempos de execução do que os obtidos com a escolha da mediana.

O algoritmo de Pardalos e Kovoor pode ser sintetizado na seguinte forma

Algoritmo de Pardalos e Kovoor

Passo 0: Seja $C = \{ 1, \ldots , n \}$; $Min = - \infty$; $Max = + \infty$;

$PC = \{ a_i , i = 1, \ldots , n \} \cup \{ - \infty, + \infty \}$;

$SA = SB = 0$.

Passo 1: Se $C = \varnothing$ vá para Passo 2.

Caso contrário escolha $y \in PC$.

$$S = SB + \sum_{\substack{i \in C \\ a_i > y}} c_i a_i + (SA + \sum_{\substack{i \in C \\ y \geq a_i}} c_i \ ) y.$$

Se $S \leq d$ então $Min = y$;

Se $S \geq d$ então $Max = y$.

$$SB = SB + \sum_{\substack{i \in C \\ a_i \geq Max}} c_i a_i \quad ; \quad SA = SA + \sum_{\substack{i \in C \\ a_i \leq Min}} c_i .$$

$C = \{ i \in C : Min < a_i < Max \}$.

$PC = \{ x \in PC : Min < x < Max \}$.

Repita o Passo 1.

Passo 2: $i = 1, \ldots, n$ $\begin{cases} \text{Se } a_i \geq Max \text{ faça } x_i = a_i \\ \text{Se } a_i < Max \text{ faça } x_i = (d - SB) / SA \end{cases}$

Termine.

2.2 Método de Helgarson, Kennington e Lall

Se considerarmos novamente o PQ (7.4) e introduzirmos o multiplicador de Lagrange λ, podemos escrever o PQ na seguinte forma

$$\begin{aligned} \min \quad & q^T z + \tfrac{1}{2} z^T M z + \lambda (e^T z - 1) \\ \text{sujeito a} \quad & z \geq 0 \end{aligned} \qquad (7.7)$$

A solução deste programa é uma função de λ. Como M é diagonal é imediato constatar que essa solução $z(\lambda)$ vem dada por

$$i = 1, \ldots, n, \quad z_i (\lambda) = \begin{cases} -\dfrac{q_i + \lambda}{m_{ii}} & \lambda \leq -q_i \\ 0 & \lambda > -q_i \end{cases}$$

Definindo $r(\lambda) = e^T z(\lambda) - 1$, constata-se que qualquer solução de (7.7) é solução de (7.4) se e só se $r(\lambda) = 0$. O problema reduz-se assim ao cálculo dos zeros da função $r(\lambda)$. Ora esta função é monótona não crescente, linear e derivável por pedaços, sendo os pontos de descontinuidade da derivada os valores $-q_i$, $i = 1, \ldots, n$. Então, para determinar λ^* tal que $r(\lambda^*) = 0$, basta encontrar dois pontos críticos consecutivos em que r tenha sinais contrários. Finalmente, o valor de λ^* é determinado exactamente por interpolação linear, visto a função r ser linear entre dois pontos críticos consecutivos. Tal processo implica que os valores de $-q_i$ ($i = 1, \ldots, n$) estejam ordenados.

Quando se está a resolver um programa quadrático de matriz não diagonal por um algoritmo sequencial acontece, muitas vezes, que os valores de q_i de um problema não são muitos diferentes dos valores obtidos no problema anterior. Assim, é conveniente usar uma técnica de ordenação que tire partido do vector q poder já estar parcialmente ordenado. Como, na iteração inicial, se parte de um vector completamente desordenado, é conveniente usar uma técnica eficiente. Em geral o algoritmo de quick-sort [82] é considerado como o mais rápido. No entanto, tal algoritmo não tira partido do vector poder estar parcialmente ordenado. Assim, na resolução dos problemas seguintes é mais conveniente usar um algoritmo que não efectue trabalho no caso de o vector já estar parcialmente ordenado. Assim, o algoritmo de inserção linear [24, pag 73] é, neste caso, uma boa escolha.

Uma pesquisa binária pode ser usada para encontrar dois valores consecutivos λ_i e λ_{i+1} entre os quais se situa λ^*, isto é tais que $r(\lambda_i) < 0$ e $r(\lambda_{i+1}) > 0$, . Tal como é descrito em [32] é possível tirar partido dos pontos críticos estarem ordenados e do facto da função r ser monótona, para encontrar λ^* por um algoritmo polinomial. Seja $C = \{ c_i : i = 1, \ldots, n \}$ o conjunto dos pontos

críticos ordenados em ordem crescente. Os passos dessa pesquisa binária podem ser resumidos do modo a seguir apresentado.

Algoritmo HEKELA

Passo 0: Faça $k = 1$, $j = n$, $B = 0$ e $L = +\infty$

Passo 1: Se $j - k = 1$ vá para Passo3.

Caso contrário defina $m = [\ ½(\ k + j\ )\ ]$, faça $\lambda = c_m$, calcule $R = \sum_i \max\left\{ -\frac{q_i + \lambda}{m_{ii}}, 0 \right\}$ e vá para Passo 2.

Passo 2: Se $R = 1$ termine com $\lambda^* = c_m$.

Se $R > 1$ faça $k = m$, $L = R$ e volte ao Passo 1.

Se $R < 1$ faça $j = m$, $B = R$ e volte ao Passo 1.

Passo3: Faça $\lambda^* = c_k + \frac{(c_j - c_k)(1 - L)}{B - L}$ e termine.

Este algoritmo é bastante elegante, mas tem a desvantagem de exigir uma ordenação dos pontos críticos. Como iremos ver, este inconveniente vai torná-lo pouco competitivo com outros processos que não necessitam de tal procedimento.

Consideremos o BLCP (7.5). Como a variável λ não tem restrição de sinal, então será sempre básica. Designando por F o conjunto das variáveis z_i básicas, ter-se-à, numa dada iteração

$$\begin{cases} M_{FF}\, z_F + \lambda\, e_F = -\, q_F \\ -\, e_F^T\, z_F = -\, 1 \end{cases}$$

ou seja

$$\begin{cases} \lambda = -\,\dfrac{1 + e_F^T\, M_{FF}^{-1}\, q_F}{e_F^T\, M_{FF}^{-1}\, e_F} \\ z_F = -\, M_{FF}^{-1}\,(\, q_F + \lambda\, e_F\,) \end{cases}$$

Como a matriz M é diagonal estas expressões podem ser escritas abreviadamente como

$$\begin{cases} \lambda = -\,\dfrac{1 + \displaystyle\sum_{i \in F} \dfrac{q_i}{m_{ii}}}{\displaystyle\sum_{i \in F} \dfrac{1}{m_{ii}}} \\ z_i = -\,\dfrac{q_i + \lambda}{m_{ii}} \quad , \quad i = 1, \ldots, n \end{cases} \tag{7.8}$$

Atendendo a (7.5), os valores de w são dados por

$$\begin{cases} w_i = 0 \,, \; i \in F \\ w_i = q_i + \lambda \;\; , \; i \notin F \end{cases} \tag{7.9}$$

O primeiro processo a discutir nesta subsecção foi introduzido em [42] e é exactamente o algoritmo pivotal principal por blocos Bloco 2 com $p = \infty$ que tira partido das fórmulas (7.8) e (7.9). Os passos do processo são apresentados a seguir.

Algoritmo Bloco 1

Passo Inicial: $F = \{ i : q_i < 0 \}$; $T = \{ 1, \ldots, n \} - F$

Passo Geral: Calcule λ usando (7.8)

Defina: $H_1 = \{ i \in F : q_i > -\lambda \}$; $H_2 = \{ i \in T : q_i < -\lambda \}$

Se $H_1 \cup H_2 = \varnothing$ calcule z_F e w_T usando (7.8) e (7.9) e páre.

Caso contrário faça: $F = F - H_1 \cup H_2$; $T = \{ 1, \ldots, n \} - F$.

Repita o Passo Geral.

Como referimos no capítulo 3, este algoritmo é heurístico. No entanto, a experiência computacional, referida na última subsecção desta secção, mostra que o processo é sempre capaz de obter a solução óptima do programa quadrático.

Seguidamente iremos desenvolver um outro algoritmo que mantém a simplicidade do anterior mas é convergente num número finito de iterações. Para isso escrevamos as condições de optimalidade de (7.4) na forma

$$0 = q + M z - I u + \lambda e$$

$$v = 0 + I z$$

$$0 = 1 - e^T z$$

$$u^T v = 0,\ v \geq 0,\ u \geq 0\ ,\ \lambda \text{ e } z \text{ sem restrição}$$

Se resolvermos a primeira equação em ordem a z e substituirmos nas outras duas equações, obtemos

$$\begin{cases} v = - M^{-1} (q - I u + \lambda e) \\ 0 = 1 + e^T M^{-1} (q - I u + \lambda e) \end{cases}$$

Se agora resolvermos a segunda equação em ordem a λ e substituirmos na primeira equação, vem

$$\lambda = \frac{1}{e^T M^{-1} e} \left[-1 - e^T M^{-1} q + e^T M^{-1} u \right]$$

e

$$v = \left[-M^{-1} q + \frac{1 + e^T M^{-1} q}{e^T M^{-1} e} M^{-1} e \right] + \left[M^{-1} - \frac{1}{e^T M^{-1} e} M^{-1} e e^T M^{-1} \right] u$$

Multiplicando por M e fazendo $y = M v$, obtém-se, finalmente,

$$y = \left[-q + \frac{1 + e^T M^{-1} q}{e^T M^{-1} e} e \right] + \left[I - \frac{1}{e^T M^{-1} e} e e^T M^{-1} \right] u$$

ou seja

$$\begin{aligned} & y = p + A u \\ & y \geq 0, \; u \geq 0, \; y^T u = 0 \end{aligned} \qquad (7.10)$$

com

$$p = -q + \frac{1 + e^T M^{-1} q}{e^T M^{-1} e} e, \qquad A = I - \frac{1}{e^T M^{-1} e} e e^T M^{-1}.$$

Mostrámos assim que o BLCP (7.4) é equivalente ao LCP (7.10). Além disso, a matriz A é Z e por isso esse problema pode ser resolvido pelo algoritmo de Chandrasekaran discutido no capítulo 3.

É importante notar que podemos desenvolver este processo sem considerar explicitamente a matriz A. Com efeito, como a matriz M é diagonal, os elementos de A podem ser calculados a partir de:

$$i = 1, \ldots, n \begin{cases} a_{ii} = 1 - \dfrac{1}{m_{ii}} / \gamma_o \\ a_{ij} = -\dfrac{1}{m_{jj}} / \gamma_o \quad j = 1, \ldots, n, \; j \neq i \end{cases}$$

com $\gamma_o = \sum_{k=1}^{n} \dfrac{1}{m_{kk}}$.

Além disso, os elementos do vector p são facilmente obtidos a partir de

$p_i = - q_i + \alpha_o$, $i = 1, \ldots, n$, com $\alpha_o = \dfrac{1+\beta_o}{\gamma_o}$ e $\beta_o = \sum_{k=1}^{n} \dfrac{q_k}{m_{kk}}$.

Na aplicação do método de Chandrasekaran consideram-se os conjuntos de índices F e T tais que $F = \{ i : u_i \text{ é básica} \}$, $T = \{ i : y_i \text{ é básica} \}$. Partindo de $F = \varnothing$, em cada iteração faz-se $F = F \cup \{ i \in T : y_i < 0 \}$. Os valores de y_T e u_F são calculados, como habitualmente, a partir de

$$\begin{cases} u_F = - A_{FF}^{-1} \, p_F \\ y_T = p_T + A_{TF} \, u_F \end{cases}$$

Neste caso, devido à forma especifica da matriz A_{FF} é fácil calcular explicitamente A_{FF}^{-1} não sendo necessário recorrer a qualquer factorização. Com efeito, $A_{FF} = I - ab^T$, com $a = \dfrac{1}{e^T M^{-1} e} e_F = \dfrac{1}{\gamma_o} e_F$ e $b = M_{FF}^{-1} e_F$. Então $A_{FF}^{-1} = I + \dfrac{ab^T}{1 - a^T b}$. Este resultado pode ser facilmente constatado por simples multiplicação. Portanto

$$u_F = -(I + \frac{ab^T}{1 - a^T b}) p_F = -p_F - \frac{b^T p_F}{1 - a^T b} a \qquad (7.11)$$

Por outro lado,

$$a^T b = \frac{1}{\gamma_o} e_F^T M_{FF}^{-1} e_F = \frac{\gamma_F}{\gamma_o}$$

e

$$b^T p_F = (M_{FF}^{-1} e_F)^T (-q_F + \alpha_o e_F) =$$
$$- e_F^T M_{FF}^{-1} q_F + \alpha_o e_F^T M_{FF}^{-1} e_F = -\beta_F + \alpha_o \gamma_F$$

com $\beta_F = \sum_{k \in F} \frac{q_k}{m_{kk}}$ e $\gamma_F = \sum_{k \in F} \frac{1}{m_{kk}}$. Substituindo em (7.11) vem

$$u_F = -p_F - \frac{\alpha_o \gamma_F - \beta_F}{1 - \frac{\gamma_F}{\gamma_o}} \frac{1}{\gamma_o} e_F = q_F - \alpha_o e_F - \frac{\alpha_o \gamma_F - \beta_F}{\gamma_o - \gamma_F} e_F =$$

$$= q_F - \frac{\alpha_o \gamma_o - \beta_F}{\gamma_o - \gamma_F} e_F = q_F - \frac{1 + \beta_o - \beta_F}{\gamma_o - \gamma_F} e_F$$

Finalmente, escrevendo

$$\begin{cases} \beta_T = \beta_o - \beta_F \\ \gamma_T = \gamma_o - \gamma_F \end{cases}$$

tem-se

$$u_F = q_F - \frac{1 + \beta_T}{\gamma_T} e_F$$

$$y_T = -q_T + (\alpha_o - \frac{\beta_F}{\gamma_o} + \frac{1 + \beta_T}{\gamma_T} \frac{\gamma_F}{\gamma_0}) e_T$$

Atendendo a que $\alpha_o = \dfrac{1+\beta_o}{\gamma_o}$, podemos simplificar esta última expressão e obter

$$y_T = -q_T + \frac{1+\beta_T}{\gamma_T}\, e_T$$

Tendo em conta todas estas fórmulas, o algoritmo de Chandrasekaran aplicado ao LCP (7.9) pode ser escrito na seguinte forma:

Algoritmo Bloco 2

Passo Inicial: $F = \varnothing$, $T = \{ 1, \ldots , n \}$,

$$\gamma_F = \beta_F = 0, \qquad \beta_T = \sum_{k=1}^{n} \frac{q_k}{m_{kk}} , \qquad \gamma_T = \sum_{k=1}^{n} \frac{1}{m_{kk}}$$

Passo Geral: $\alpha = \dfrac{1+\beta_T}{\gamma_T}$

$I = \{ i \in T : q_i > \alpha \}$

se $I = \varnothing$ páre: a solução é $u_F = q_F - \alpha\, e_F$, $y_T = - q_T + \alpha\, e_T$

Caso contrário faça $T = T - I$, $F = F \cup I$

$$\beta_F = \beta_F + \sum_{k \in I} \frac{q_k}{m_{kk}} \qquad \beta_T = \beta_T - \sum_{k \in I} \frac{q_k}{m_{kk}}$$

$$\gamma_F = \gamma_F + \sum_{k \in I} \frac{1}{m_{kk}} \qquad \gamma_T = \gamma_T - \sum_{k \in I} \frac{1}{m_{kk}}$$

Repita o Passo Geral.

Uma vez obtido o vector y_T tem-se, finalmente, $z_T = M_{TT}^{-1} y_T$, $z_F = 0$. Este algoritmo é obviamente convergente, pois é exactamente o algoritmo de

Chandrasekaran aplicado a um LCP com matriz Z. Note-se que o valor de α corresponde ao valor de $-\lambda$ do algoritmo Bloco 1. Além disso, o algoritmo Bloco 2 é equivalente ao algoritmo Bloco 1 se inicialmente for $F = \{ 1, \ldots, n \}$. Portanto a convergência do algoritmo Bloco 1 está assegurada se for iniciado com $F = \{ 1, \ldots, n \}$. Contudo a escolha de um conjunto F diferente de $\{ 1, \ldots, n \}$ não assegura teoricamente a convergência do processo.

2.4 Experiência Computacional

Para comparar a eficiência dos algoritmos descritos nesta secção resolvemos alguns problemas da forma (7.4) gerados aleatoriamente do seguinte modo

$q_i \in [-10,10]$ tal que $| q_i |$ é uniformemente distribuído em [0, 10]

$m_{11} = 1$, $m_{22} = \text{cond}$, $m_{ii} \in [1, \text{cond}]$, $i = 3, \ldots, n$, também uniformemente distribuído.

em que cond representa o número de condição da matriz diagonal M. Na tabela 7.1 apresenta-se os resultados obtidos na resolução de um problema com dimensão 10000. Fizemos ensaios com número de condição 1 e 10^5 e concluímos que o número de condição de M não parece influenciar o desempenho de qualquer dos algoritmos testados. Outro factor que também mostrou não ter grande influência nesse desempenho foi o número de componentes negativas do vector q (NEG). Também experimentámos multiplicar todas as componentes do vector q por um factor de escala (ESC) e os algoritmos continuam a mostrar-se insensíveis a esse facto.

Dos resultados obtidos pode-se constatar que o algoritmo HEKELA, é o que normalmente requer um menor número de operações. Contudo o seu tempo de

execução é o maior, devido a utilizar um algoritmo de ordenação. Nestes ensaios utilizou-se o algoritmo quick-sort. Na tabela representa-se por TS o tempo gasto na ordenação, enquanto que T representa o tempo total. Os restantes algoritmos mostram-se bastante semelhantes, embora, de um modo geral, se possa afirmar que, em termos de tempo de execução, o algoritmo Bloco 1 é o mais eficiente. O algoritmo Bloco 2 mostra ser mais eficiente que o algoritmo de Pardalos em todos os problemas testados. De salientar que, nalguns casos, o algoritmo Bloco 2 requer menos tempo que o algoritmo Bloco 1 embora executando mais operações. Tal deve-se ao facto de as comparações necessárias à formação do novo conjunto F terem mais peso no algoritmo Bloco 1.

COND		1				100000			
ESC		1		1000		1		1000	
NEG		2500	7500	2500	7500	2500	7500	2500	7500
HEKELA	NO	2.1	2.2	2.1	2.2	5.4	10.1	2.2	2.3
	TS	1.66	1.65	1.75	1.73	1.66	1.65	1.73	1.73
	T	2.72	2.79	2.86	2.92	3.00	3.38	2.92	2.86
	RES	2-13	8-14	2-11	2-10	4-14	1-13	2-13	1-13
PARDALOS	NI	14	5	13	5	10	8	11	4
	NO	5.0	5.0	5.0	5.0	6.0	6.0	5.0	5.0
	T	1.14	1.06	1.18	1.10	1.43	1.43	1.17	1.10
	RES	1-10	1-9	1-7	7-7	1-12	3-12	4-10	5-10
BLOCO 1	NI	8	8	7	7	3	4	8	8
	NO	0.99	3.0	0.99	3.0	0.84	2.7	0.99	3.0
	T	1.17	1.4	0.67	1.04	0.60	0.90	0.72	1.10
	RES	6-10	5-9	8-6	2-6	3-13	3-12	9-10	1-9
BLOCO 2	NI	9	9	9	8	4	5	9	9
	NO	4.0	4.0	4.0	4.0	3.8	3.7	4.0	4.0
	T	1.03	1.06	1.07	1.06	0.79	0.88	1.07	1.11
	RES	8-8	5-8	8-5	5-5	2-12	3-12	4-9	4-10

Tabela 7.1

3 PROGRAMA QUADRÁTICO SEPARADO ESTRITAMENTE CONVEXO COM RESTRIÇÕES DE MOCHILA

Este tipo de programa pode ser formulado como

$$\begin{array}{ll} \text{Min} & q^T z + \frac{1}{2} z^T M z \\ \text{sujeito a} & e^T z = 1 \\ & 0 \le z \le b \end{array} \qquad (7.12)$$

Iremos supor que $e^T b \ge 1$ para que a região admissível seja não vazia. Seguidamente apresentamos técnicas eficientes para este problema que podem ser vistas como generalizações das apresentadas na secção anterior.

3.1 MÉTODO DE PARDALOS E KOVOOR

Fazendo a mudança de variável já referida anteriormente obtém-se o problema equivalente

$$\begin{array}{ll} \min & g(x) = x^T C x \\ \text{sujeito a} & a \le x \le f \\ & c^T x = d \end{array} \qquad (7.13)$$

com $a = \frac{1}{2} q$, $d = 1 + q^T M^{-1} e$, $C = 2 M^{-1}$, $c = C e$ e $f = \frac{1}{2} (q + M b)$. As condições de Kuhn-Tucker deste PQ podem ser escritas do seguinte modo:

$$c^T x = d$$

$$a - x \le 0$$

$$x - b \le 0$$

$$2Cx + \lambda c + v - \mu = 0$$

$$\mu^T(a - x) + v^T(x - b) = 0$$

$$\mu \geq 0 \ , \ v \geq 0$$

Além disso é possível demonstrar [75] que qualquer x pertencente ao conjunto admissível de (7.13) é mínimo de g se existir y tal que

$$i = 1, \ldots , n, \quad x_i (y) = \begin{cases} a_i & \text{se } y < a_i \\ y & \text{se } a_i \leq y \leq f_i \\ f_i & \text{se } y > f_i \end{cases}$$

e, além disso, $y = -\frac{\lambda}{2}$.

Tal com anteriormente pode-se definir a função monótona não decrescente linear e diferenciável por pedaços $h(y) = \sum_{i=1}^{n} c_i x_i (y)$. Neste caso o conjunto dos pontos críticos é $PC = \{ a_i, i = 1, \ldots , n \} \cup \{ f_i, i = 1, \ldots , n \} \cup \{ -\infty, +\infty \}$. Ainda como na secção anterior, procuramos obter um intervalo] Min, Max [que contenha y* tal que h(y*) = d. Além disso, vamos, por tentativas, estreitando esse intervalo até se obter Min = Max = y*.

Neste caso $a_i >$ Max implica $x_i(y) = a_i$ e $x_i(y) = f_i$ se $f_i <$ Min. Então o conjunto C definido no algoritmo de Pardalos e Kovoor da secção anterior será constituído pelos índices i tal que a_i ou f_i estão compreendidos entre Min e Max. Por outro lado, se $a_i \leq$ Min < Max $\leq f_i$ será sempre $x_i(y) = y$. O algoritmo toma assim o seguinte aspecto:

Algoritmo de Pardalos e Kovoor

Passo 0: Seja $C = \{ 1, \ldots, n \}$; $Min = -\infty$; $Max = +\infty$;

$PC = \{ a_i , i = 1, \ldots, n \} \cup \{ f_i, i = 1, \ldots, n \} \cup \{ -\infty, +\infty \}$;

$SA = SB = 0$.

Passo 1: Se $C = \varnothing$ vá para Passo2.

Caso contrário escolha $y \in PC$.

$$S = SB + \sum_{\substack{i \in C \\ a_i > y}} c_i a_i + \sum_{\substack{i \in C \\ f_i < y}} c_i f_i + \left(SA + \sum_{\substack{i \in C \\ a_i \le y \le f_i}} c_i \right) y.$$

Se $S \le d$ então $Min = y$;

Se $S \ge d$ então $Max = y$.

$$SB = SB + \sum_{\substack{i \in C \\ a_i \ge Max}} c_i a_i + \sum_{\substack{i \in C \\ f_i \le Min}} c_i f_i \; ; \quad SA = SA + \sum_{\substack{i \in C \\ a_i \le Min \wedge b_i \ge Max}} c_i .$$

$C = \{ i \in C : Min < a_i < Max \text{ ou } Min < f_i < Max \}$.

$PC = \{ x \in PC : Min < x < Max \}$.

Repita o Passo 1.

Passo 2:

$$i = 1, \ldots, n \begin{cases} \text{Se } a_i > Max \text{ faça } x_i = a_i \\ \text{Se } b_i < Min \text{ faça } x_i = b_i \\ \text{Se } a_i \le Min \text{ e } bi \ge Max \text{ faça } x_i = (d - SB) / SA \end{cases}$$

Termine.

3.2 Método de Helgarson, Kennington e Lall

Tal como anteriormente introduz-se o multiplicador de Lagrange λ, obtendo-se

$$\begin{aligned} &\min \quad q^T z + \tfrac{1}{2} z^T M z + \lambda (e^T z - 1) \\ &\text{sujeito a} \quad 0 \le z \le b \end{aligned} \tag{7.14}$$

Igualando a zero o gradiente da função $g(z) = q^T z + \frac{1}{2} z^T M z + \lambda (e^T z - 1)$, obtém-se $M z + q + \lambda e = 0$ ou seja $z = - M^{-1}(q + \lambda e)$. Se agora atendermos a que M é diagonal, vem

$$i = 1, \ldots, n, \quad z_i (\lambda) = -\frac{q_i + \lambda}{m_{ii}}$$

Mas, por outro lado, tem que ser $0 \le z_i \le b_i$, ou seja $0 \le -(q_i + \lambda) \le m_{ii} b_i$. Então, para que se verifiquem as condições de optimalidade de (7.14), deve-se definir $z (\lambda)$ do seguinte modo

$$i = 1, \ldots, n, \quad z_i (\lambda) = \begin{cases} 0 & \text{se } -\lambda > q_i + m_{ii} bi \\ -\dfrac{q_i + \lambda}{m_{ii}} & \text{se } q_i \le -\lambda \le q_i + m_{ii} bi \\ b_i & \text{se } -\lambda < q_i \end{cases}$$

Tal como anteriormente define-se $r(\lambda) = e^T z(\lambda) - 1$ e o problema reduz-se ao cálculo dos zeros da função $r(\lambda)$ que é monótona não crescente, linear e derivável por pedaços, com pontos críticos $\{ -q_i, i = 1, \ldots, n \} \cup \{ -q_i - m_{ii} b_i, \ i = 1, \ldots, n \}$. Então, para determinar λ^* tal que $r(\lambda^*) = 0$, basta encontrar dois pontos críticos consecutivos em que r tenha sinais contrários, o que torna necessária a ordenação de 2n valores. Finalmente, o valor de λ^* é determinado exactamente por interpolação linear. Seja $C = \{ c_i : i = 1, \ldots, 2n \}$ o conjunto dos pontos críticos em ordem

crescente. O algoritmo para a determinação de λ^* é muito semelhante ao apresentado na secção anterior e os seus passos são apresentados a seguir.

Algoritmo HEKELA

Passo 0: Faça k = 1, j = 2n, B = 0 e $L = \sum_{i=1}^{n} b_i$

Passo 1: Se j - k = 1 vá para Passo3.

Caso contrário defina m = [½(k + j)], faça $\lambda = c_m$, calcule $R = \sum_i \max \left\{ \min \{ -\frac{q_i + \lambda}{m_{ii}}, b_i \}, 0 \right\}$ e vá para Passo 2.

Passo 2: Se R = 1 termine com $\lambda^* = c_m$.

Se R > 1 faça k = m, L = R e volte ao Passo 1.

Se R < 1 faça j = m, B = R e volte ao Passo 1.

Passo3: Faça $\lambda^* = c_k + \frac{(c_j - c_k)(1 - L)}{B - L}$ e termine.

3.3 MÉTODO PIVOTAL PRINCIPAL POR BLOCOS

As condições de Kuhn-Tucker do PQ (7.12) podem ser escritas na forma do BLCP

$$\begin{bmatrix} y \\ \gamma \\ 0 \end{bmatrix} = \begin{bmatrix} q \\ b \\ -1 \end{bmatrix} + \begin{bmatrix} M & I & -e \\ -I & O & O \\ e^T & O & O \end{bmatrix} \begin{bmatrix} z \\ \lambda \\ u \end{bmatrix}$$

$$y^T z = \gamma^T \lambda = 0 \; ; \; y \geq 0 \; ; \; z \geq 0 \; ; \; \gamma \geq 0 \; ; \; \lambda \geq 0 \; ;$$

Efectuando uma operação pivotal com pivot M obtém-se

$$\begin{bmatrix} z \\ \gamma \\ 0 \end{bmatrix} = \begin{bmatrix} -M^{-1}q \\ b + M^{-1}q \\ -1 - e^T M^{-1}q \end{bmatrix} + \begin{bmatrix} M^{-1} & -M^{-1} & M^{-1}e \\ -M^{-1} & M^{-1} & -M^{-1}e \\ e^TM^{-1} - e^TM^{-1} & e^TM^{-1}e \end{bmatrix} \begin{bmatrix} y \\ \lambda \\ u \end{bmatrix} \qquad (7.15)$$

$$y^Tz = \gamma^T\lambda = 0 \; ; \; y \geq 0 \; ; \; z \geq 0 \; ; \; \gamma \geq 0 \; ; \; \lambda \geq 0 \; ;$$

Este último BLCP tem matriz simétrica singular. Como as primeiras n colunas são simétricas das n seguintes, é imediato constatar que não podem ser simultaneamente básicas as variáveis y_i e λ_i. Por outro lado, como $\gamma_i = b_i - z_i$ e $b_i > 0$, então é impossível que γ_i e z_i sejam simultaneamente não básicas.

Designemos por A a matriz do LCP (7.15). Seja $F = F_1 \cup F_2$ com $F_1 = \{ i : y_i \text{ é básica } \}$ e $F_2 = \{ i : \lambda_i \text{ é básica } \}$ ($F_1 \cap F_2 = \varnothing$). Então A_{FF} tem o aspecto

$$A_{FF} = \begin{bmatrix} M^{-1}_{F_1F_1} & O & M^{-1}_{F_1F_1} e_{F_1} \\ O & M^{-1}_{F_2F_2} & -M^{-1}_{F_2F_2} e_{F_2} \\ e^T_{F_1} M^{-1}_{F_1F_1} & -e^T_{F_2} M^{-1}_{F_2F_2} & e^TM^{-1}e \end{bmatrix}$$

pois a variável u é sempre básica.

Os valores das variáveis básicas y_{F_1}, λ_{F_2} e u são obtidos resolvendo o sistema

$$A_{FF} \begin{bmatrix} y_{F_1} \\ \lambda_{F_2} \\ u \end{bmatrix} = \begin{bmatrix} M^{-1}_{F_1F_1} q_{F_1} \\ -b_{F_2} - M^{-1}_{F_2F_2} q_{F_2} \\ 1 + e^TM^{-1}q \end{bmatrix} \qquad (7.16)$$

A matriz deste sistema pode ser escrita como o produto de duas matrizes triangulares

$$A_{FF} = \begin{bmatrix} M^{-1}_{F_1F_1} & O & O \\ O & M^{-1}_{F_2F_2} & O \\ e^T_{F_1} M^{-1}_{F_1F_1} & -e^T_{F_2} M^{-1}_{F_2F_2} & 1 \end{bmatrix} \begin{bmatrix} I_{F_1} & O & e_{F_1} \\ O & I_{F_2} & -e_{F_2} \\ O & O & \beta \end{bmatrix}$$

com

$$\beta = e^T M^{-1} e - e_{F_1}^T M_{F_1F_1}^{-1} e_{F_1} - e_{F_2}^T M_{F_2F_2}^{-1} e_{F_2} =$$

$$= \sum_{i=1}^{n} \frac{1}{m_{ii}} - \sum_{i \in F_1} \frac{1}{m_{ii}} - \sum_{i \in F_2} \frac{1}{m_{ii}} = \sum_{i \in T} \frac{1}{m_{ii}}$$

e $T_1 = \{ 1, \ldots, n \} - F_1$, $T_2 = \{ 1, \ldots, n \} - F_2$ e $T = T_1 \cap T_2$.

Usando esta factorização, o sistema (7.16) resolve-se sem problemas, obtendo-se

$$\begin{cases} u = \frac{1}{\beta}(1 + e^T M^{-1} q - e_{F_1}^T M_{F_1F_1}^{-1} q_{F_1} - e_{F_2}^T M_{F_2F_2}^{-1} q_{F_2} - e_{F_2}^T b_{F_2}) \\ y_{F_1} = q_{F_1} - e_{F_1} u \\ \lambda_{F_2} = - M_{F_2F_2} b_{F_2} - q_{F_2} + e_{F_2} u \end{cases}$$

ou ainda, simplificando,

$$\begin{cases} u = \frac{1}{\beta}(1 + e_T^T M_{TT}^{-1} q_T - e_{F_2}^T b_{F_2}) \\ y_{F_1} = q_{F_1} - e_{F_1} u \\ \lambda_{F_2} = - M_{F_2F_2} b_{F_2} - q_{F_2} + e_{F_2} u \end{cases} \qquad (7.17)$$

Por outro lado, as variáveis z_{T_1} e γ_{T_2} são dadas por

$$z_{T_1} = -M_{T_1T_1}^{-1} q_{T_1} - M_{T_1T_1}^{-1} \lambda_{T_1} + M_{T_1T_1}^{-1} e_{T_1} u = M_{T_1T_1}^{-1} (-q_{T_1} - \lambda_{T_1} + e_{T_1} u) \qquad (7.18a)$$

$$\gamma_{T_2} = b_{T_2} + M_{T_2T_2}^{-1} q_{T_2} - M_{T_2T_2}^{-1} y_{T_2} + M_{T_2T_2}^{-1} e_{T_2} u = b_{T_2} + M_{T_2T_2}^{-1} (q_{T_2} - y_{T_2} - e_{T_2} u) \qquad (7.18b)$$

Note-se que, enquanto que $F_1 \cap F_2 = \varnothing$, já não é verdade que a intersecção de T_1 com T_2 seja vazia. Assim poderá haver variáveis z_i e γ_i ($i \in T_1 \cap T_2$) simultaneamente básicas.

Tendo isto em consideração pode-se aplicar directamente a este problema o método pivotal principal por blocos descrito no capítulo 3. Os passos do algoritmo resultante são os seguintes

Algoritmo Bloco

Passo inicial: $F_1 = \{ i : -\frac{q_i}{m_{ii}} < 0 \}$, $F_2 = \{ i : b_i + \frac{q_i}{m_{ii}} < 0 \}$.

Passo geral: $F = F_1 \cup F_2$, $T_1 = \{ 1,\ldots, n \} - F_1$, $T_2 = \{ 1,\ldots, n \} - F_2$, $T = T_1 \cap T_2$.

Calcule z, y, λ, γ e u usando (7.17) e (7.18).

Defina $H_1 = \{ i : z_i < 0 \}$, $H_2 = \{ i : y_i < 0 \}$,

$H_3 = \{ i : \gamma_i < 0 \}$, $H_4 = \{ i : \lambda_i < 0 \}$ e $H = \bigcup_{i=1}^{4} H_i$

Se $H = \varnothing$ termine com a solução.

Caso contrário faça $F_1 = F_1 - H_2 \cup H_1$, $F_2 = F_2 - H_4 \cup H_3$

e repita o Passo Geral.

3.4 Experiência Computacional

Para comparar a eficiência dos algoritmos descritos nesta secção geraram-se problemas de modo semelhante ao descrito na secção anterior. Na tabela 7.2 apresentam-se os resultados obtidos na resolução de alguns programas quadráticos separados estritamente convexos com restrições de mochila. Nessa tabela ZP representa o número de variáveis tais que $0 < z_i < b_i$ e ZB o número de variáveis z_i

tais que $z_i = b_i$ na solução óptima. Os resultados da tabela 7.2 já tinham sido apresentados em [43] onde se discutem pela primeira vez estes três algoritmos.

N		1000				5000			
COND		100		100000		100		10000	
ZP		180	58	900	673	1479	507	4752	1479
ZB		4	38	100	14	16	0	248	16
BLOCO	NI	4	2	1	2	3	3	1	2
	NO	1.75	1.06	0.6	0.88	6.7	6.88	3.0	4.8
	T	0.35	0.23	0.12	0.17	1.25	1.28	0.5	0.87
HEKELA	NO	1.15	0.96	3.55	2.77	9.05	5.44	21.0	9.06
	TS	0.27	0.28	0.26	0.26	1.61	1.60	1.62	1.60
	T	0.76	0.72	1.03	1.03	4.76	3.31	6.11	4.74
PARDALOS	NI	11	19	78	78	7	4	56	13
	NO	0.76	0.92	1.06	1.06	3.61	3.46	5.07	4.03
	T	0.21	0.3	0.53	0.53	0.9	0.79	2.68	1.35

Tabela 7.2

Os resultados desta tabela mostram que o algoritmo Bloco é bastante eficiente resolvendo os problemas num número de iterações bastante pequeno, sendo, dum modo geral, o que resolve os problemas em menos tempo. De novo, o algoritmo Hekela é o que tem pior desempenho, em parte devido ao tempo gasto na ordenação dos valores críticos. Tal como se constatou na secção anterior, nenhum dos algoritmos parece ser afectado pelo número de condição da matriz.

4 Programa Quadrático Separado Estritamente Convexo com Restrições de Igualdade e Limites

4.1 Método Pivotal Principal por Blocos

Considere-se finalmente o programa

$$\begin{array}{ll} \text{Min} & q^T z + \frac{1}{2} z^T M z \\ \text{sujeito a} & A z = b \\ & 0 \le z \le f \end{array} \qquad (7.19)$$

com $M \in R^{n \times n}$ uma matriz simétrica PD, $A \in R^{m \times n}$, $b \in R^m$, $f \in R^n$ e $m < n$. Além disso, iremos considerar que a matriz M é diagonal e que o conjunto das restrições é não vazio.

As condições de optimalidade de (7.19) têm a seguinte forma

$$\begin{bmatrix} y \\ \gamma \\ 0 \end{bmatrix} = \begin{bmatrix} q \\ f \\ -b \end{bmatrix} + \begin{bmatrix} M & I & -A^T \\ -I & O & O \\ A & O & O \end{bmatrix} \begin{bmatrix} z \\ \lambda \\ u \end{bmatrix}$$

$$y \ge 0 \; ; \; z \ge 0 \; ; \; \lambda \ge 0 \; ; \; \gamma \ge 0 \; ; \; y^T z = \lambda^T \gamma = 0.$$

Efectuando uma operação pivotal principal com pivot M obtém-se

$$\begin{bmatrix} z \\ \gamma \\ 0 \end{bmatrix} = \begin{bmatrix} -M^{-1}q \\ f + M^{-1}q \\ -b - A M^{-1}q \end{bmatrix} + \begin{bmatrix} M^{-1} & -M^{-1} & M^{-1}A^T \\ -M^{-1} & M^{-1} & -M^{-1}A^T \\ A M^{-1} & -A M^{-1} & A M^{-1}A^T \end{bmatrix} \begin{bmatrix} y \\ \lambda \\ u \end{bmatrix}$$

$$y \ge 0 \; ; \; z \ge 0 \; ; \; \lambda \ge 0 \; ; \; \gamma \ge 0 \; ; \; y^T z = \lambda^T \gamma = 0.$$

Se a matriz A tem característica completa, $C = A M^{-1}A^T$ é não singular. Tal como anteriormente as variáveis u devem ser sempre todas básicas. Então na primeira

iteração deve-se começar por efectuar uma operação pivotal com pivot C para que todas essas variáveis se tornem básicas. Para simplificar as notações define-se $p = - M^{-1}q$ e $r = f + M^{-1}q$. Então:

$$\begin{cases} u = - C^{-1}b \\ z = p + M^{-1}A^{T}u \\ \gamma = r - M^{-1}A^{T}u \end{cases} \tag{7.20}$$

Tal como em algoritmos pivotais principais anteriormente descritos, vamos usar os conjuntos de índices F e T. Neste caso é necessário considerar uma partição do conjunto F, $F = F_1 \cup F_2$, com F_1 e F_2 tais que

$$F_1 = \{ i : y_i \text{ é básica } \} , F_2 = \{ i : \lambda_i \text{ é básica } \}$$

Assim, na primeira iteração, é $F_1 = \{ i : z_i < 0 \}$, $T_1 = \{ 1, \ldots , n \} - F_1$, $F_2 = \{ i : \gamma_i < 0 \wedge i \notin F_1 \}$, $T_2 = \{ 1, \ldots , n \} - F_2$, $F = F_1 \cup F_2$ e $T = T_1 \cap T_2$. Tendo em conta esta partição, a base tem em cada iteração a seguinte forma:

$$B_{FF} = \begin{bmatrix} M_{F_1F_1}^{-1} & O & M_{F_1F_1}^{-1} A_{.F_1}^{T} \\ O & M_{F_2F_2}^{-1} & - M_{F_2F_2}^{-1} A_{.F_2}^{T} \\ A_{.F_1} M_{F_1F_1}^{-1} & -A_{.F_2} M_{F_2F_2}^{-1} & AM^{-1}A^{T} \end{bmatrix}$$

Desde que esta matriz seja não singular ela pode ser escrita como o produto seguinte

$$B_{FF} = \begin{bmatrix} M_{F_1F_1}^{-1} & O & O \\ O & M_{F_2F_2}^{-1} & O \\ A_{.F_1} M_{F_1F_1}^{-1} & -A_{.F_2} M_{F_2F_2}^{-1} & I_m \end{bmatrix} \begin{bmatrix} I_{F_1} & O & A_{.F_1}^{T} \\ O & I_{F_2} & - A_{.F_2}^{T} \\ O & O & E \end{bmatrix} \tag{7.21}$$

com $E = AM^{-1}A^{T} - A_{.F_1} M_{F_1F_1}^{-1} A_{.F_1}^{T} - A_{.F_2} M_{F_2F_2}^{-1} A_{.F_2}^{T}$. Claro que é preciso ter a certeza que esta matriz é não singular. Deve-se começar por averiguar se é possível factorizar E. Se, no decorrer do processo, se encontrar um elemento diagonal nulo isso significa que o elemento correspondente não pode ser inserido em F. Nas iterações seguintes, é

preciso continuar a ter cuidado quando se introduzem elementos em F. O processo é semelhante ao que foi descrito no capítulo 3 para matrizes PSD.

Note-se que este processo será tanto mais eficiente quanto menor for m. Para valores de m muito pequenos a matriz C pode ser tratada como matriz densa. Se o valor de m for muito elevado o processo perde eficácia. Com efeito, mesmo que a matriz A seja esparsa a matriz C terá, normalmente, muitos elementos não nulos, o que pode impedir o seu tratamento como matriz esparsa.

Em cada iteração, os valores das variáveis básicas y_{F1}, λ_{F2} e u são obtidos por resolução do sistema

$$B_{FF}\begin{bmatrix} y_{F1} \\ \lambda_{F2} \\ u \end{bmatrix} = \begin{bmatrix} M_{F1F1}^{-1} q_{F1} \\ - f_{F2} - M_{F2F2}^{-1} q_{F2} \\ b + A\, M^{-1} q \end{bmatrix}$$

Usando a factorização (7.21) a solução deste sistema é dada por

$$\begin{cases} E\, u = b + A_{.T}\, M_{TT}^{-1}\, q_T - A_{.F2} f_{F2} \\ \lambda_{F2} = - q_{F2} - M_{F2F2}\, f_{F2} + A_{.F2}^{T} u \\ y_{F1} = q_{F1} - A_{.F1}^{T}\, u \end{cases} \tag{7.22}$$

Além disso, os valores de z_{T1} e γ_{T2} são calculados a partir de

$$\begin{cases} z_{T1} = M_{T1T1}^{-1} \left(-q_{T1} - \lambda_{T1} + A_{.T1}^{T}\, u \right) \\ \gamma_{T2} = f_{T2} - z_{T2} \end{cases} \tag{7.23}$$

com $z_i = 0$ para $i \in T_2 - T_1$.

Neste caso o método Bloco tem que ser aplicado com muito cuidado, pois há sempre que prever a hipótese de se encontrar uma matriz B_{FF} singular.

Suponhamos que, numa determinada iteração, se tem B_{FF} não singular. Isso implica que é possível realizar a factorização (7.21) e obter a matriz E não singular. Nessa iteração os conjuntos F e T devem ser alterados de acordo com:

$$H_1 = \{ i : z_i < 0 \},\ H_2 = \{ i : y_i < 0 \},\ H_3 = \{ i : \gamma_i < 0 \},\ H_4 = \{ i : \lambda_i < 0 \} \qquad (7.24)$$

$$\bar{F}_1 = F_1 - H_2 \cup H_1\ ,\ \bar{T}_1 = T_1 - H_1 \cup H_2\ ,$$

$$\bar{F}_2 = F_2 - H_4 \cup H_3\ ,\ \bar{T}_2 = T_2 - H_3 \cup H_4\ .$$

Então, na decomposição da nova matriz $B_{\bar{F}\bar{F}}$ obtém-se uma matriz $\bar{E}$ que pode ser relacionada com a matriz E da decomposição de B_{FF}. Com efeito,

$$\bar{E} = A\,M^{-1}A^T - A_{.\bar{F}1}M^{-1}_{\bar{F}1\bar{F}1}A^T_{.\bar{F}1} - A_{.\bar{F}2}M^{-1}_{\bar{F}2\bar{F}2}A^T_{.\bar{F}2} =$$

$$= A\,M^{-1}A^T - [\,A_{.F1}M^{-1}_{F_1F_1}A^T_{.F_1} - A_{.H2}M^{-1}_{H_2H_2}A^T_{.H_2} + A_{.H1}M^{-1}_{H_1H_1}A^T_{.H_1}\,] -$$

$$- [\,A_{.F2}M^{-1}_{F_2F_2}A^T_{.F_2} - A_{.H4}M^{-1}_{H_4H_4}A^T_{.H_4} + A_{.H3}M^{-1}_{H_3H_3}A^T_{.H_3}\,] =$$

$$= E + A_{.H2}M^{-1}_{H_2H_2}A^T_{.H_2} + A_{.H4}M^{-1}_{H_4H_4}A^T_{.H_4} - A_{.H1}M^{-1}_{H_1H_1}A^T_{.H_1} - A_{.H3}M^{-1}_{H_3H_3}A^T_{.H_3}$$

Como todas as parcelas desta expressão são matrizes PSD, concuí-se que há que ter cuidado ao acrescentar elementos a F_1 e a F_2, enquanto que retirar elementos a esses conjuntos não acarreta qualquer problema. Tendo em conta esse facto recomenda-se uma estratégia semelhante à já usada no capítulo 3 para matrizes PSD. Assim, em cada iteração, começa-se por actualizar a matriz E para

$$E + A_{.H2}M^{-1}_{H_2H_2}A^T_{.H_2} + A_{.H4}M^{-1}_{H_4H_4}A^T_{.H_4}$$

que é correspondente a fazer

$$F_1 = F_1 - H_2 \text{ e } F_2 = F_2 - H_4$$

Seguidamente, para cada $i \in H_1 \cup H_3$ faz-se

$$E = E - A_{.i} M_{ii}^{-1} A_{.i}^T$$

Isto equivale a fazer uma correcção de característica um à matriz E. Então, pode-se actualizar a decomposição LDL^T usando o algoritmo de Bennett. Se, ao fazer essa actualização, for detectado algum elemento diagonal nulo, então o índice i não pode ser acrescentado a F. Deve-se repor a factorização anterior e considerar outro índice. Caso contrário faz-se $F = F \cup \{ i \}$ e repete-se o processo até que todos os índices de $H_1 \cup H_3$ tenham sido considerados.

O algoritmo Bloco, aplicado à resolução do problema (7.19), pode ser descrito nos seguintes passos:

Método Bloco

Passo 0: Seja atc um inteiro positivo.

Calcule u, z e γ usando (7.20).

Faça $F_1 = \{ i : z_i < 0 \}$, $F_2 = \{ i : \gamma_i < 0 \}$ e $F = F_1 \cup F_2$

Passo1: Defina $T_1 = \{ 1, \ldots, n \} - F_1$, $T_2 = \{ 1, \ldots, n \} - F_2$.

Calcule y_{F1}, λ_{F2}, u, z_{T1} e γ_{T2} usando (7.22) e (7.23).

Defina H_i, i = 1, 2, 3, 4 usando (7.24) e faça $H = \bigcup_{i=1}^{4} H_i$.

Se $\#H \leq$ atc vá para Passo 3.

Caso contrário faça $F_1 = F_1 - H_2$ e $F_2 = F_2 - H_4$ e vá para Passo 2.

Passo2: $F = F_1 \cup F_2$

Se $H_1 \cup H_3 = \varnothing$ vá para Passo 1.

Caso contrário seja $i = \min \{ i \in H_1 \cup H_3 \}$.

$i \in H_1 \Rightarrow H_1 = H_1 - \{ i \}$; $i \in H_3 \Rightarrow H_3 = H_3 - \{ i \}$.

$\tilde{F} = F \cup \{ i \}$.

Se $B_{\tilde{F}\tilde{F}}$ é não singular faça $F = \tilde{F}$.

Repita o Passo 2.

Passo3: Se $H = \varnothing$ a solução foi encontrada e páre.

Caso contrário faça $s = \min \{ i \in H \}$ e aplique o método Criss-Cross.

4.2 Experiência Computacional

Para avaliar da eficiência deste algoritmo resolvemos alguns problemas da forma (7.19) gerados aleatoriamente. As restrições de igualdade foram geradas de acordo com o recomendado em [28]. Os restantes dados foram gerados de acordo com o já descrito em secções anteriores.

Na tabela 7.3 apresentam-se os resultados obtidos nessa resolução. Nessa tabela n representa a dimensão da matriz M e m o número de restrições de igualdade.

Os resultados computacionais mostram que o algoritmo Bloco continua a ser eficiente, embora o seu desempenho não seja tão bom como nos casos apresentados nas secções anteriores.

m	n	NI	NO	T
24	101	8	2.69	0.65
50	101	12	22.2	4.03
39	159	15	11.9	2.5
79	159	11	68.6	11.6
49	201	8	11.2	2.35
74	302	12	37.7	7.08

Tabela 7.3

5 Programa Quadrático não Separado Estritamente Convexo com uma Restrição e Limites

Considere-se o programa

$$\begin{array}{ll} \text{Min} & q^T z + \frac{1}{2} z^T M z \\ \text{sujeito a} & e^T z = 1 \\ & 0 \leq z \leq b \end{array} \qquad (7.25)$$

com M uma matriz simétrica PD de ordem n, $q \in R^n$, $e^T = (1, \ldots, n)$ e b um vector de R^n cujas componentes podem ser um número real positivo ou $+\infty$.

Se no problema (7.25) a matriz M não for diagonal deixamos de ter um programa separado e outros métodos têm de ser considerados. Nesta secção vamos considerar somente o caso de todas as componentes de b serem iguais a $+\infty$. Tal como foi feito em casos anteriores, é possível desenvolver extensões dos algoritmos para o tratamento do caso dos limites superiores serem finitos. Nesta secção iremos estudar a aplicação de um método de diagonalização e dos algoritmos pivotais principais discutidos nos capítulos 2 e 3.

5.1 Método de Diagonalização SOR

Este método foi proposto em [17] e consiste em gerar uma sucessão de vectores { z^k } convergente para a solução do problema. Partindo de um z^o que pertença ao conjunto admissível de (7.25), cada z^{k+1} é obtido resolvendo o programa separado

$$\begin{array}{ll} \text{Min} & (q^k)^T z + \frac{1}{2} z^T \Delta z \\ \text{sujeito a} & e^T z = 1 \\ & z \geq 0 \end{array} \qquad (7.26)$$

em que $\Delta = \text{diag}\,(m_{11}, \ldots , m_{nn})$ e $q^k = (M - \Delta) z^k + q$.

Seja $\bar{z}^k$ a solução de (7.26). É possível demonstrar [17] que ($\bar{z}^k - z^k$) é uma direcção descendente. Assim determina-se $z^{k+1} = z^k + \rho_k (\bar{z}^k - z^k)$, sendo ρ_k determinado de modo a que z^{k+1} continue a pertencer ao conjunto admissível de (7.25) e, além disso, que a função objectivo de (7.25) tome o menor valor possível na direcção dada por ($\bar{z}^k - z^k$). A terminação do algoritmo é feita tendo em conta o valor de $\| \bar{z}^k - z^k \|$ que deve ser menor que uma dada tolerância.

Para determinar ρ_k começa-se por definir $z (\rho) = z^k + \rho (\bar{z}^k - z^k)$. É fácil ver que, se $\bar{z}^k$ e z^k são tais que $e^T \bar{z}^k = 1$ e $e^T z^k = 1$, também será $e^T z(\rho) = 1$ para qualquer valor de ρ. Assim, o intervalo de variação de ρ vai ser condicionado pela restrição $z (\rho) \geq 0$. Mas,

$$z (\rho) \geq 0 \Rightarrow z_i^k + \rho (\bar{z}_i^k - z_i^k) \geq 0 \, , \, i = 1, \ldots , n.$$

Ora, como $z_i^k \geq 0$, só é preciso ter em atenção a possibilidade de ser $\bar{z}_i^k - z_i^k < 0$. Nesse caso tem que se exigir que seja

$$\rho \leq \frac{z_i^k}{z_i^k - \bar{z}_i^k} \quad , \text{para } i : \bar{z}_i^k - z_i^k < 0.$$

Define-se $\bar{\rho} = \min \{ \frac{z_i^k}{z_i^k - \bar{z}_i^k} \quad , \text{para } i : \bar{z}_i^k - z_i^k < 0 \}$ e finalmente, obtém-se ρ_k como o valor que minimiza $h(\rho)$ em $[0 , \bar{\rho}]$, com $h(\rho) = \frac{1}{2} z(\rho)^T M z(\rho) + q^T z(\rho)$. A função $h(\rho)$ é um polinómio de segundo grau em ρ. Determinando ρ^* tal que $h'(\rho^*) = 0$, o valor de ρ_k é escolhido entre $\{ 0, \rho^*, \bar{\rho} \}$ se $\rho^* \in [0, \bar{\rho}]$ ou entre $\{ 0, \bar{\rho} \}$ caso contrário.

O cálculo de ρ_k é bastante trabalhoso. Seria desejável que se pudesse fazer $\rho_k = 1$, isto é que $z^{k+1} = \bar{z}^k$. Em [17] são referidas algumas condições a que têm que obedecer M e Δ para que tal seja possível. Por exemplo, isso é verdadeiro se $\| \Delta^{-1}(M - \Delta) \| < 1$ ou se M é diagonal dominante e $\Delta = \gamma \, \text{diag}(M)$, com $2\gamma > \lambda_{max}$, e λ_{max} o maior valor próprio de M. Contudo estas condições não são de verificação fácil.

O programa (7.26) pode ser resolvido por qualquer algoritmo apresentado na secção 2 deste capítulo. No final desta secção apresentamos alguma experiência computacional com este algoritmo de diagonalização, em que o programa (7.26) é resolvido pelo método HEKELA (SOR1) ou pivotal por blocos (SOR2).

Se $b \in R^n$ este algoritmo pode ser facilmente estendido bastando alterar a forma de calcular ρ. Com efeito, é agora necessário garantir também que $z(\rho) \leq b$, isto é, que $\rho \leq \frac{b_i - z_i^k}{\bar{z}_i^k - z_i^k}$, para $i : \bar{z}_i^k - z_i^k > 0$. Define-se assim $\bar{\rho} = \min \{ \rho_1, \rho_2 \}$ com $\rho_1 = \min \{ \frac{z_i^k}{z_i^k - \bar{z}_i^k} \quad , \text{para } i : \bar{z}_i^k - z_i^k < 0 \}$ e

$$\rho_2 = \min \{ \frac{b_i - z_i^k}{\bar{z}_i^k - z_i^k} \quad , \text{para } i : \bar{z}_i^k - z_i^k > 0 \}.$$

5.2 Método Paramétrico Pivotal Principal

Neste algoritmo [71] começa-se por introduzir um parâmetro λ e resolve-se o LCP paramétrico

$$w = (q + \lambda e) + M z \qquad (7.27)$$
$$z \geq 0,\ w \geq 0,\ z^T w = 0$$

para valores de λ reais, até encontrar λ^* tal que $e^T z(\lambda^*) = 1$. Então, o vector $z(\lambda^*)$ é a solução única de (7.25). O algoritmo usa sempre pivotagens principais e por isso tem-se em cada iteração um sistema da seguinte forma

$$\begin{bmatrix} z_F \\ w_T \end{bmatrix} = \begin{bmatrix} \bar{q}_F \\ \bar{q}_T \end{bmatrix} + \begin{bmatrix} \bar{e}_F \\ \bar{e}_T \end{bmatrix} \lambda + \begin{bmatrix} \bar{M}_{FF} & \bar{M}_{FT} \\ \bar{M}_{TF} & \bar{M}_{TT} \end{bmatrix} \begin{bmatrix} w_F \\ z_T \end{bmatrix}$$

com $\bar{q}_F$ e $\bar{q}_T$ obtidos como habitualmente e $\bar{e}_F = -M_{FF}^{-1} e_F$, $\bar{e}_T = e_T - M_{TF} M_{FF}^{-1} e_F$.

O algoritmo, é bastante semelhante à aplicação do método de Graves para o LCP (7.27) e tem a seguinte forma:

Método de Pang

Passo 0: Determine k tal que $q_k = \min \{ q_i, i = 1, \ldots, n \}$

Faça $\lambda_b = - q_k$, $F = \{ k \}$ e $T = \{ 1, \ldots, n \} - F$

Passo1: Calcule $\bar{q}$ e $\bar{e}$

Determine k tal que $\dfrac{\bar{q}_k}{\bar{e}_k} = \min \{ \dfrac{\bar{q}_i}{\bar{e}_i}, \bar{e}_i > 0 \}$

Faça $\lambda_a = - \dfrac{\bar{q}_k}{\bar{e}_k}$

$z_F(\lambda) = \bar{q}_F + \lambda\ \bar{e}_F$, $z_T(\lambda) = 0$ é solução de (7.20) para

$\lambda \in [\lambda_a, \lambda_b]$.

Determine $\lambda^* = \dfrac{1 - e_F^T \bar{q}_F}{e_F^T \bar{e}_F}$

Se $\lambda^* \in [\lambda_a, \lambda_b]$ páre: $z(\lambda^*)$ é solução de (7.4).

Caso contrário vá para Passo 2.

Passo 2: Faça $\lambda_b = \lambda_a$.

Se $k \in F$ faça $F = F - \{ k \}$, caso contrário faça $F = F \cup \{ k \}$.

Faça $T = \{ 1, \ldots, n \} - F$.

Volte a Passo 1.

Em cada iteração deste algoritmo resolvem-se dois sistemas com matriz M_{FF}. Por outro lado, o conjunto F é alterado em apenas um elemento por iteração. Então a implementação que foi descrita no capítulo 2 para os métodos pivotais principais simples é adequada para este método.

5.3 Método de Murty

O método de Murty que foi descrito no capítulo 2 também pode ser aplicado a este caso. Para isso escrevemos as condições de optimalidade de (7.25) na seguinte forma

$$\begin{aligned} w &= q + M z + \lambda e \\ 0 &= 1 - e^T z \\ w^T z &= 0 \ , \ w \geq 0 \ , \ z \geq 0 \end{aligned} \qquad (7.28)$$

que pode ser encarado como um BLCP com uma variável sem restrição de sinal, termo independente $p = \begin{bmatrix} q \\ 1 \end{bmatrix}$ e matriz $A = \begin{bmatrix} M & e \\ -e^T & 0 \end{bmatrix}$. Para simplificar definam-se ainda $w_{n+1} = 0$ e $z_{n+1} = \lambda$. Se $F = \{ i : z_i \text{ é básica} \}$, tem-se, em cada iteração,

$$\begin{bmatrix} z_F \\ w_T \end{bmatrix} = \begin{bmatrix} \bar{p}_F \\ \bar{p}_T \end{bmatrix} + \begin{bmatrix} \bar{A}_{FF} & \bar{A}_{FT} \\ \bar{A}_{TF} & \bar{A}_{TT} \end{bmatrix} \begin{bmatrix} w_F \\ z_T \end{bmatrix}$$

Como a variável z_{n+1} não tem restrição de sinal será sempre básica e, por isso, a matriz A_{FF} tem o aspecto

$$A_{FF} = \begin{bmatrix} M_{FF} & e_F \\ -e_F^T & 0 \end{bmatrix}$$

Os valores de $\bar{p}_F = \begin{bmatrix} \bar{q}_F \\ \bar{p}_{n+1} \end{bmatrix}$ obtêm-se por resolução de um sistema com matriz A_{FF}. Para tal comecemos por escrever A_{FF} como o produto de duas matrizes triangulares por blocos:

$$A_{FF} = \begin{bmatrix} M_{FF} & O \\ -e_F^T & 1 \end{bmatrix} \begin{bmatrix} I & M_{FF}^{-1} e_F \\ O & e_F^T M_{FF}^{-1} e_F \end{bmatrix}$$

Deste modo, o sistema $A_{FF}\ \bar{p}_F = \begin{bmatrix} -q_F \\ -1 \end{bmatrix}$ é resolvido do seguinte modo

(i) Resolver o sistema $M_{FF}\, x = -q_F$

(ii) Resolver o sistema $M_{FF}\, \bar{e}_F = e_F$

(iii) Calcular $\bar{p}_{n+1} = (-1 + e_F^T x) / e_F^T \bar{e}_F$

(iv) Calcular $\bar{q}_F = x - \bar{p}_{n+1}\, \bar{e}_F$

Então, em cada iteração, é necessário resolver dois sistemas com matriz M_{FF} e calcular dois produtos internos para obter $\bar{p}_F$. Por seu turno $\bar{p}_T$ obtém-se, como habitualmente, a partir de $\bar{p}_T = p_T + A_{TF}\, \bar{p}_F = q_T + M_{TF}\, \bar{q}_F + e_T\, \bar{p}_{n+1}$.

Como a variável z_{n+1} é sempre básica, o conjunto F deve conter inicialmente o índice (n+1). Mas o elemento diagonal de ordem (n+1) de A é nulo e portanto deve-se fazer F = { r, n+1 }. Assim $A_{FF} = \begin{bmatrix} m_{rr} & 1 \\ -1 & 0 \end{bmatrix}$ que é uma matriz não singular.

Os passos do método de Murty são apresentados a seguir

Método de Murty

Passo Inicial: Escolha r tal que $q_r = \min \{ q_i \}$.

Faça F = { r, n+1 } ; T = { 1, ..., n } - F.

Passo Geral: Calcule $\bar{p}$. Seja $H = \{ i: \bar{p}_i < 0 \text{ e } i \neq n+1 \}$.

Se $H = \varnothing$ a solução é $\begin{cases} z_F = \bar{q}_F & w_F = 0 \\ z_T = 0 & w_T = \bar{q}_T \end{cases}$

Caso contrário seja $r = \min \{ i \in H \}$.

Se $r \in F$ faça $F = F - \{ r \}$, $T = T \cup \{ r \}$.

Caso contrário faça $F = F \cup \{ r \}$, $T = T - \{ r \}$.

Repita o Passo Geral.

A convergência deste tipo de algoritmo não foi ainda apresentada na literatura. Seguidamente apresentamos alguns factos que nos indicam que o método deve ser convergente quando $M \in P$. Em primeiro lugar note-se que, se $M \in P$ então este problema é equivalente ao problema de desigualdades variacionais lineares

Encontrar $\bar{z}$ tal que

$$(q + M \bar{z})^T (z - \bar{z}) \geq 0 \text{ para todo o } z \in K \qquad (7.29)$$

com $K = \{ z \in R^n : z \geq 0, e^T z = 1 \}$

Como K é um conjunto compacto e $M \in P$, então o LVI (7.29) tem solução única para todo o $q \in R^n$ [2] e o mesmo acontece com o LCP (7.28).

Por outro lado, a matriz A goza das seguintes propriedades

(i) $\det (A_{FF}) > 0$ para qualquer $F \subset \{ 1, \ldots , n \}$. Com efeito, pela fórmula de Schur tem-se $\det (A_{FF}) = \det (M_{FF}) \det (e_F^T M_{FF}^{-1} e_F) > 0$.

(ii) Se $\bar{A}$ é uma transformada principal de A obtida por uma operação pivotal com pivot A_{FF}, então:

$$\det (\bar{A}_{JJ}) \geq 0 \text{ para todo o } J \subseteq \{ 1, \ldots , n+1 \}$$

$$\det (\bar{A}_{JJ}) = 0 \Leftrightarrow (F \cup J) - (F \cap J) = \{ n+1 \}$$

pois $\det (\bar{A}_{JJ}) = \dfrac{\det(A_{HH})}{\det(A_{FF})}$ com $H = (F \cup J) - (F \cap J)$

Então, efectuando uma operação pivotal com pivot $\begin{bmatrix} m_{rr} & 1 \\ -1 & 0 \end{bmatrix}$ obtém-se uma matriz em que todos os menores principais são não negativos e o único nulo será $\bar{a}_{nn}$, que nunca será chamado a ser pivot. Portanto as operações pivotais simples são sempre possíveis e a solução é única. Essas duas propriedades levam-nos a concluir que se pode estabelecer a convergência do algoritmo usando uma demonstração semelhante à utilizada na extensão do método de Murty apresentada no capítulo 4. A extensão deste algoritmo ao problema quadrático com limites pode ser desenvolvido de um modo semelhante ao apresentado no capítulo 4.

5.4 MÉTODO DE KELLER

De modo semelhante ao método de Murty também se pode pensar numa extensão do método de Keller adequada à solução do LCP (7.28). Tendo em consideração que a variável z_{n+1} será sempre básica e nunca candidata a sair da base, pode-se escrever o algoritmo na seguinte forma

Método de Keller

Passo 0: Escolha r tal que $q_r = \min \{ q_i \}$.

Faça $F = \{ r, n+1 \}$; $T = \{ 1, \ldots, n \} - F$.

Passo1: Calcule $\bar{p}$. Seja $H = \{ i: \bar{p}_i < 0 \text{ e } i \neq n+1 \}$.

Se $H = \varnothing$ a solução é $\begin{cases} z_F = \bar{q}_F & w_F = 0 \\ z_T = 0 & w_T = \bar{q}_T \end{cases}$

Caso contrário seja r tal que $\bar{p}_r = \min \{ \bar{p}_i , i \in H \}$.

Vá para Passo 2.

Passo2: Calcule $\bar{M}_{FT} = - M_{FF}^{-1} M_{FT}$ e $\bar{m}_{rT} = m_{rT} + M_{rF} \bar{M}_{FT}$

Calcule $\theta_1 = - \dfrac{\bar{q}_r}{\bar{m}_{rT}}$

$$\theta_2 = \begin{cases} - \dfrac{\bar{q}_s}{\bar{m}_{sr}} = \min \{ - \dfrac{\bar{q}_i}{\bar{m}_{ir}} : \bar{m}_{ir} < 0 \text{ e } i \in F - \{ n+1 \} \} \\ + \infty \text{ se } \bar{m}_{ir} \geq 0 \text{ para todo } i \in F - \{ n+1 \} \end{cases}$$

Se $\theta_1 \leq \theta_2$ faça $F = F \cup \{ r \}$, $T = T - \{ r \}$ e vá para Passo 1.

Caso contrário faça $F = F - \{ s \}$, $T = T \cup \{ s \}$, calcule $\bar{q}_F$ e $\bar{q}_r$.

Repita o Passo 2.

Este algoritmo é obviamente convergente quando M é simétrica PD. Além disso, o algoritmo deve ser convergente para M $\in$ P não simétrica. Contudo esse resultado ainda não foi estabelecido.

A extensão para o programa quadrático com limites não é difícil de fazer e é convergente para matrizes simétricas PD.

5.5 Método Pivotal Principal por Blocos

Continuando a seguir o procedimento já descrito para o método de Murty pode-se também reformular o método Bloco 2 descrito no capítulo 3 de modo a torná-lo aplicável ao LCP (7.28). Para tal basta ter em atenção que tem que se ter sempre (n+1) $\in$ F, mas F deve conter sempre mais um elemento que (n + 1). Repare-se que esta última situação corresponderia a ser z = 0, ou seja, $e^T z = 0 \neq 1$. Assim sendo, na primeira iteração será $F = \{ i : q_i < 0 \} \cup \{ n+1 \}$. Caso seja $q \geq 0$, deve-se escolher um r com $q_r > 0$ para iniciar com $F = \{ r, n+1 \}$ ao que corresponderá $z_r = 1$, $z_T = 0$, $\bar{p}_{n+1} = - q_r - m_{Tr}$, $w_T = q_T + M_{Tr} + e_T \bar{p}_{n+1}$. Se for $w_T \geq 0$ o processo termina. Caso contrário prossegue tal como descrito no capítulo 3. O algoritmo Bloco neste caso pode ser descrito nos seguintes passos.

Algoritmo Bloco 2

Passo 0: Seja ip um número inteiro positivo.

Faça k = 1 , ninf = nmaxpv = n , $F = \{ i : q_i < 0 \} \cup \{ n+1 \}$.

Se $q \geq 0$ faça $F = \{ 1, n+1 \}$

Faça $T = \{ 1 , \ldots , n \} - F$.

PASSO 1: Calcule $\bar{q}$ e seja $H = \{ i : \bar{p}_i < 0 \text{ e } i \neq n+1 \}$.

(i) Se $H = \varnothing$, $z = (\bar{q}_F , 0)$, $w = (0 , \bar{q}_T)$ é a solução única do LCP e páre.

(ii) Se $\#H < ninf$, faça $ninf = \#H$, $nmaxpv = k + p$ e vá para Passo 2.

(iii) Se $\#H \geq ninf$ então $\begin{cases} k \leq nmaxpv \Rightarrow \text{vá para Passo 2.} \\ k > nmaxpv \Rightarrow \text{vá para Passo3.} \end{cases}$

PASSO 2: Faça $F = F - (F \cap H) \cup (T \cap H)$ e $T = \{ 1 , \ldots , n \} - F$.

Faça $k = k + 1$ e volte a Passo 1.

PASSO 3: Seja $s = \min \{ i \in H \}$

$\begin{cases} \text{se } s \in F \text{ faça } F = F - \{ s \} \text{ e } T = T \cup \{ s \} \\ \text{se } s \in T \text{ faça } F = F \cup \{ s \} \text{ e } T = T - \{ s \} \end{cases}$

Faça $k = k + 1$ e volte a Passo 1.

5.6 EXPERIÊNCIA COMPUTACIONAL

Apresenta-se nesta subsecção a experiência computacional descrita em [38], em que se compara a eficiência dos algoritmos descritos nesta secção. Os programas quadráticos (7.25) foram obtidos a partir de LCPs já descritos em capítulos anteriores adicionando a restrição de igualdade. Tal como anteriormente, as experiências foram efectuadas usando o computador CDC CYBER 180-230 da Universidade do Porto. A tolerância para o critério de paragem do algoritmo SOR foi de 10^{-4}.

		KELLER	MURTY	BLOCO	PARAM.	SOR1	SOR2
TP1	NI	38	38	5	29	11	11
	NO	3.3	3.9	3.9	3.5	4.7	6.7
	T	5.1	5.2	5.2	6.3	7.2	2.1
TP2	NI	91	91	7	90	242	242
	NO	16.	15.	3.	20.	200.	210.
	T	8.1	7.8	4.8	16.6	236.	58.
TP3	NI	129	131	7	128	340	340
	NO	24.	23.	3.	29.	350.	360.
	T	10.1	9.7	4.3	15.9	526.	94.
TP5	NI	229	229	5	212	209	209
	NO	60.	60.	3.	60.	390.	370.
	T	22.9	21.3	5.5	29.2	988.	87.
TP6	NI	128	128	3	129	33	33
	NO	20.	19.	2.1	21.	24.	18.
	T	5.4	4.9	1.1	5.2	6.3	3.2
TP7	NI	128	128	4	129	43	43
	NO	118.	95.	45.	107.	66.	58.
	T	25.4	21.4	12.2	23.8	14.9	10.9
TP8	NI	368	368	3	369	23	23
	NO	200.	190.	19.	240.	65.	47.
	T	51.4	49.	12.2	58.5	26.4	8.4
TP17	NI	64	64	6	65	10	10
	NO	4.9	4.6	0.7	4.9	5.3	5.6
	T	2.8	2.7	1.5	3.1	5.9	1.2
TP18	NI	120	120	4	121	10	10
	NO	15.	14.	0.9	14.	9.4	9.7
	T	6.4	5.8	1.7	6.8	13.9	2.1
TP19	NI	143	143	4	140	9	9
	NO	23.	21.	1.	22.	10.	9.
	T	10.8	10.1	3.9	11.6	19.4	2.2

Tabela 7.4

Da análise dos resultados apresentados na tabela 7.4 pode-se concluir que os métodos pivotais principais de Murty, Keller e Paramétrico apresentam desempenhos semelhantes. O método Bloco mostra ser bastante eficiente. O método SOR apresenta um mau desempenho, embora a sua eficácia seja bastante melhorada quando se usa o método Bloco para resolver o problema separado. Nesse caso a sua eficiência torna-se mesmo em alguns casos comparável com a do algoritmo pivotal por blocos. Assim, não é de estranhar que recomendemos o método pivotal por blocos para a resolução deste tipo de programas quadráticos.

Apesar do seu pior desempenho, o algoritmo SOR deve ser considerado nos casos em que o número de enchimentos na factorização da matriz M tornem impossível o uso de um método pivotal principal. A eficiência do algoritmo pode ser melhorada se o algoritmo bloco for usado para a resolução dos LCPs separados ou se a ordenação dos valores críticos for feita com certo cuidado. Os resultados que se apresentam foram obtidos usando o algoritmo de inserção linear para a ordenação dos valores críticos. Algum tempo pode ser poupado se for utilizado o algoritmo quick-sort na primeira iteração e a inserção linear nas restantes, tirando assim partido dos valores críticos numa determinada iteração poderem ter a mesma ordem, ou uma ordem muito semelhante, dos valores críticos da iteração anterior.

CAPÍTULO VIII

APLICAÇÃO DA COMPLEMENTARIDADE LINEAR EM ANÁLISE ESTRUTURAL

1 INTRODUÇÃO

Quando se resolve um qualquer problema de análise de estruturas procura-se conhecer as funções tensões e deformações que se desenvolvem na estrutura quando esta está solicitada por forças exteriores. Relacionados com as tensões e deformações são também de considerar os deslocamentos que lhes estão associados.

O estado de tensão de uma estrutura é descrito por intermédio das chamadas variáveis estáticas que se relacionam pelas equações que traduzem o equilíbrio da estrutura. Por outro lado, o estado de deformação é descrito por intermédio das variáveis cinemáticas. Estas variáveis relacionam-se entre si pelas equações que traduzem a compatibilidade da estrutura. Finalmente as tensões e deformações relacionam-se entre si pela chamada equação constitutiva do material.

Fixando um referêncial ortonormado, cada ponto da estrutura pode ser representado por um elemento de R^3. Sendo assim, todas as funções referidas têm como domínio um conjunto $S \subset R^3$. As estruturas em estudo correspondem normalmente a domínios compactos.

Neste capítulo iremos descrever duas aplicações da complementaridade linear em análise estrutural. A primeira consiste na determinação da carga máxima aplicável a uma estrutura em regime elastoplástico. Como se verá mais adiante, o algoritmo permite também o conhecimento das regiões da estrutura plastificadas durante o

processo de aplicação da carga. A segunda aplicação, no domínio da análise elástica, tem como objectivo a determinação da superfície de contacto em meios elásticos lineares sujeitos a pequenas deformações. Apresenta-se um exemplo de aplicação a uma laje enviesada apoiada unilateralmente em dois bordos opostos, em que a aplicação da complementaridade permitiu determinar a extensão dos cantos levantados quando é aplicada uma carga no centro da laje.

2 Determinação da Carga Máxima de uma Estrutura em Regime Elastoplástico

Um dos problemas fundamentais da análise de estruturas, qualquer que seja o material em que seja realizada, consiste em saber qual o maior valor que pode assumir uma força exterior aplicada a um qualquer ponto da estrutura. A força aplicada produz tensões e deformações que satisfazem uma lei com três fases distintas.

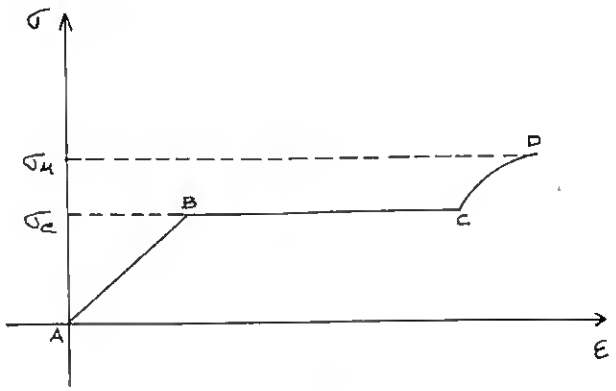


Figura 8.1

Na figura 8.1 representa-se a relação existente entre as tensões (σ) e as deformações (ε) numa estrutura qualquer, para a situação unidimensional. Numa primeira fase a relação é linear (AB). Nesta fase diz-se que se observa um

comportamento elástico linear ou seja, aumentando a tensão, aumenta a deformação de forma proporcional. Durante esta fase, se as tensões se anulam a estrutura voltará ao estado que tinha em A que corresponde a deformações nulas. Numa segunda fase (BC) diz-se que a estrutura segue um regime plástico perfeito. Ao valor σ_c dá-se o nome de limite de elasticidade e é o valor da tensão para o qual ocorre deformação sem haver aumento de tensão. Nesta fase as deformações são permanentes, isto é, a estrutura não volta ao estado inicial mesmo anulando a tensão.

A última fase (CD), acontece antes da rotura, e é caracterizada pelo facto das deformações aumentarem com as tensões de uma forma não linear. A tensão σ_u tem o nome de tensão de rotura. Esta fase não é geralmente considerada nas aplicações, admitindo-se que a estrutura tem um comportamento elástico linear plástico perfeito, ou, mais simplesmente, um comportamento elastoplástico.

A análise elastoplástica de estruturas tem como objectivo último caracterizar os campos de tensões, de deformações e de deslocamentos. Estas funções devem, em qualquer ponto da estrutura, satisfazer às relações fundamentais de equilíbrio, compatibilidade e constitutivas do material. Além disso, como uma estrutura não pode ser considerada isoladamente, há que considerar as condições de fronteira.

O campo das tensões, $\sigma(x)$, e o campo das deformações, $\varepsilon(x)$, são funções contínuas de x, em que x representa as coordenadas de um ponto da estrutura. A resolução analítica do problema é de difícil execução, pelo que se opta por uma solução numérica. Para isso é necessário considerar uma decomposição da estrutura com vista a utilizar o método dos elementos finitos [66]. Neste método, em vez de se estudar a estrutura de forma contínua, estuda-se o comportamento de elementos de pequena dimensão (elementos finitos), interligados entre si em pontos (nós) previamente escolhidos. As equações que vão permitir caracterizar o comportamento

da estrutura são então escritas a partir do comportamento nesses nós. Como o processo de discretização pressupõe uma integração numérica, o comportamento dentro do elemento finito é descrito à custa dos valores das tensões e deformações nos pontos correspondentes à quadratura Gaussiana.

Em cada elemento finito as tensões σ e as forças aplicadas f relacionam-se de forma linear, o mesmo acontecendo às deformações ε e deslocamentos u. Estas relações são dadas por

$$\varepsilon = C\, u \tag{8.1}$$

$$f = C^T \sigma \tag{8.2}$$

em que f representa a força, u o deslocamento e C uma matriz constante cuja definição depende do material estrutural e da geometria do elemento finito a que diz respeito. A relação (8.1) é conhecida por relação de compatibilidade, enquanto que a relação (8.2) tem o nome de relação de equilíbrio.

Na função deformação $\varepsilon(x)$ há que considerar uma componente elástica, $\varepsilon_e(x)$, e uma componente plástica, $\varepsilon_p(x)$, isto é,

$$\varepsilon(x) = \varepsilon_e(x) + \varepsilon_p(x) \tag{8.3}$$

A componente elástica da deformação relaciona-se de forma linear com a tensão através da chamada matriz de elasticidade ou de rigidez do material R, obedecendo à lei de Hooke

$$\sigma(x) = R\, \varepsilon_e (x) \tag{8.4}$$

A matriz de rigidez depende do material que constitui a estrutura. Por isso (8.4) chama-se relação constitutiva do material.

O comportamento plástico da estrutura no caso mais geral de meios pluridimensionais não é fácil de estudar. O valor da tensão de cedência não é constante para todas as direcções e é necessário introduzir o conceito de superfície de cedência. Os pontos sobre esta superfície correspondem às tensões que provocam a entrada na fase plástica em cada direcção. Como é exemplificado na figura 8.2, obtém-se um região convexa de fronteira não linear designada por $\pi(\sigma)$.

De acordo com o sugerido em [61] esta região é aproximada por um poliedro convexo de faces tangentes à superfície de cedência. Na figura 8.2 representa-se um hiperplano de suporte (i) desse poliedro.

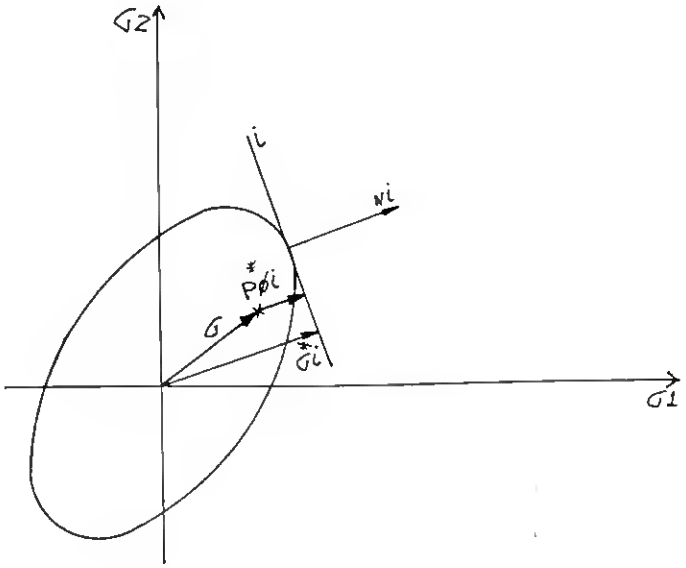


Figura 8.2

Ao hiperplano i chama-se hiperplano de cedência. Relacionadas com este hiperplano podem-se considerar as funções potencial plástico, ϕ_i^* e capacidade plástico σ_i^*. Estas funções estão relacionadas pela expressão

$$N_i^T \sigma + \phi_i^* = \sigma_i^* \tag{8.5}$$

onde $N_i^T\sigma$ traduz o estado de tensão no ponto em estudo, σ_i^* é uma característica do material estrutural que traduz o valor limite do estado de tensão e ϕ_i^* mede a capacidade que o ponto tem para receber acréscimos de tensão. Assim, pode-se exprimir como condição de não plastificação

$$\phi_i^* \geq 0 \tag{8.6}$$

Se for $\phi_i^* = 0$, então atinge-se nesse ponto e direcção N_i a cedência ou plastificação, desenvolvendo-se uma deformação plástica de intensidade ε_i^* com componentes

$$\varepsilon_{pi} = N_i \, . \, \varepsilon_i^* \tag{8.7}$$

em que

$$\varepsilon_i^* \geq 0 \tag{8.8}$$

é um multiplicador associado à deformação plástica. Se $\phi_i^* > 0$, então não há deformação plástica e $\varepsilon_i^* = 0$. Por outro lado, se $\varepsilon_i^* > 0$ então o potencial plástico é nulo. Esta complementaridade entre as variáveis ϕ_i^* e ε_i^* pode ser traduzida pela relação

$$\phi_i^* \, \varepsilon_i^* = 0 \tag{8.9}$$

As condições (8.5) a (8.9) devem ser verificadas em todos os pontos nodais da estrutura considerada. Assim obtêm-se as seguintes relações

$$N^T\sigma + \phi^* = \sigma^* \tag{8.10a}$$

$$\varepsilon_p = N \, \varepsilon^* \tag{8.10b}$$

$$\phi^{*T}\varepsilon^* = 0 \tag{8.10c}$$

$$\phi^* \geq 0 \, , \, \varepsilon^* \geq 0 \tag{8.10d}$$

As equações (8.1) e (8.2) devem ser consideradas em todos os pontos da estrutura. Além disso, para que a estrutura esteja em equilíbrio, tem que se verificar

$$0 = \lambda\, c_o + C^T \sigma \tag{8.11}$$

em que λ é o parâmetro de carga e c_o o vector de carga, correspondente às acções exteriores.

Por outro lado a equação (8.4) deve também ser estendida a toda a estrutura obtendo-se

$$\sigma = K\, \varepsilon_e \tag{8.12}$$

em que as matrizes K e R estão relacionadas de forma linear através dos pesos associados aos pontos de integração da quadratura gaussiana, visto que as funções σ e ε devem ser tais que

$$\sigma^T \varepsilon = \int_V \sigma^T(x)\, \varepsilon(x)\, dV$$

Partindo da relação (8.11) e usando as relações (8.1), (8.3), (8.10b) e (8.12) obtém-se

$$\begin{aligned} 0 = \lambda\, c_o + C^T \sigma &= \lambda\, c_o + C^T (K\, C\, u - K\, \varepsilon_p) = \\ &= \lambda\, c_o + C^T K\, C\, u - C^T K\, N\, \varepsilon^* \end{aligned} \tag{8.13}$$

$$\phi^* = \sigma^* - N^T \sigma \tag{8.14a}$$

$$\phi^* \geq 0\,,\ \varepsilon^* \geq 0 \tag{8.14b}$$

Partindo de (8.14a) e usando (8.1), (8.3), (8.10b) e (8.12) obtém-se

$$\phi^* = \sigma^* - N^T \sigma = \sigma^* - N^T K\, \varepsilon_e = \sigma^* - N^T K\, C\, u + N^T K\, \varepsilon_p =$$

$$= \sigma^* - (C^T K\, N)^T u + N^T K\, N \varepsilon^* \tag{8.15}$$

Como o objectivo é determinar o maior valor possível para o parâmetro de carga λ para o qual a estrutura se mantenha em equilíbrio, somos conduzidos a um problema de optimização em que as restrições são as condições (8.10c), (8.13) , (8.14b) e (8.15). Assim, pretende-se resolver o seguinte problema

Maximize λ

sujeito a
$$0 = c_0 \lambda + C^T K C u - C^T K N \varepsilon^*$$
$$\phi^* = \sigma^* - (C^T K N)^T u + N^T K N \varepsilon^*$$
$$\phi^{*T} \varepsilon^* = 0$$
$$\phi^* \geq 0 , \varepsilon^* \geq 0$$

ou, em forma matricial,

Maximize λ

sujeito a
$$\begin{bmatrix} 0 \\ \phi^* \end{bmatrix} = \begin{bmatrix} 0 \\ \sigma^* \end{bmatrix} + \begin{bmatrix} c_0 \\ 0 \end{bmatrix} \lambda + \begin{bmatrix} C^T K C & -C^T K N \\ -(C^T K N)^T & N^T K N \end{bmatrix} \begin{bmatrix} u \\ \varepsilon^* \end{bmatrix} \qquad (8.16)$$
$$\phi^{*T} \varepsilon^* = 0, \phi^* \geq 0 , \varepsilon^* \geq 0$$

Devido às características fisicas do problema, a matriz K é simétrica positiva definida. Por sua vez, a matrix C é de característica completa, visto que as deformações são independentes. A matriz N também é de característica completa, desde que os hiperplanos i sejam escolhidos de tal modo que as direcções ortogonais sejam linearmente independentes.

O programa (8.16) pode ser considerado um BLCP paramétrico da forma

Maximize λ

$$\text{sujeito a} \quad \begin{bmatrix} y \\ w \end{bmatrix} = \begin{bmatrix} p \\ q \end{bmatrix} + \begin{bmatrix} g \\ h \end{bmatrix} \lambda + \begin{bmatrix} A & B \\ B^T & E \end{bmatrix} \begin{bmatrix} u \\ z \end{bmatrix} \tag{8.17}$$

$$z \geq 0 \, , \, w \geq 0 \, , \, z^T w = 0$$

A matriz $Q = \begin{bmatrix} A & B \\ B^T & E \end{bmatrix}$ pode ser escrita na forma do produto

$$Q = \begin{bmatrix} C & O \\ O & N \end{bmatrix}^T \begin{bmatrix} K & -K \\ -K & K \end{bmatrix} \begin{bmatrix} C & O \\ O & N \end{bmatrix},$$

e portanto é simétrica positiva semi-definida. Além disso $A = C^T K C$ é simétrica positiva definida e por isso não singular. Então em (8.17) é possível efectuar uma operação pivotal com pivot A. Tal operação pivotal torna as variáveis u básicas. Ora estas variáveis (deslocamentos) não têm restrição de sinal e, por isso, as suas variáveis complementares, y, têm que ser nulas. As restrições lineares do programa (8.17), depois de feita esta operação pivotal, ficam com o aspecto

$$\begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} \bar{p} \\ b \end{bmatrix} + \begin{bmatrix} \bar{g} \\ d \end{bmatrix} \lambda + \begin{bmatrix} \bar{A} & -\bar{B} \\ \bar{B}^T & M \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix}$$

com

$$\bar{A} = A^{-1} \, , \; \bar{B} = A^{-1} B \, , \; \bar{p} = A^{-1} p \, , \; \bar{g} = - A^{-1} g \, ,$$

$$M = E - B^T A^{-1} B \, , \, b = q - B^T A^{-1} p \, , \, d = h \, A^{-1} g$$

Como as variáveis y nunca se irão tornar básicas, o programa (8.17) pode ser resolvido através de

$$\begin{aligned} &\text{Maximize} && \lambda \\ &\text{sujeito a} && w = b + \lambda\, d + M\, z \\ & && z \geq 0\,,\ w \geq 0\,,\ \lambda \geq 0\,,\ z^T w = 0 \end{aligned} \tag{8.18}$$

Uma vez obtida uma solução para o programa (8.18), os valores correspondentes para as variáveis u são obtidos resolvendo o sistema

$$A\,u = -\,(\,p + \lambda\, g + B\,z\,) \tag{8.19}$$

O programa (8.18) é um LCP paramétrico com matriz simétrica positiva semi-definida, pois M é o complemento de Schur de A em Q. Por isso o método pivotal principal de Cottle [7] pode ser utilizado para a sua resolução. Este método encontra, em cada iteração, uma solução admissível para valores de λ pertencentes a um determinado intervalo $[\ \underline{\lambda}^k,\ \bar{\lambda}^k\ ]$. Os intervalos encontrados em iterações consecutivas são adjacentes e os valores de λ não decrescentes, ou seja $\underline{\lambda}^{k+1} = \bar{\lambda}^k$ e $\bar{\lambda}^{k+1} \geq \bar{\lambda}^k$. Então ou o valor de λ pode crescer indefinidamente, ou é impossível encontrar um novo intervalo em que exista uma solução admissível. A primeira hipótese é fisicamente impossível e, se essa situação acontecer, isso significa que o problema foi mal formulado. No segundo caso $\bar{\lambda}^k$ corresponde ao maior valor possível para λ.

O algoritmo de Cottle é descrito seguidamente com o mesmo formalismo que foi usado no capítulo 2 para descrever o algoritmo de Graves.

Algoritmo de Cottle

Passo 0: Faça $k = 0$, $F = \varnothing$, $T = \{\ 1, \ldots, n\ \}$

$\bar{b} = b$, $\bar{d} = d$, $\bar{M} = M$ e $\underline{\lambda}^k = 0$.

Passo 1: Se $\bar{d} \geq 0$ o problema é ilimitado e páre.

Caso contrário determine $\bar{\lambda}^k = -\dfrac{\bar{b}_s}{\bar{d}_s} = \min \{ -\dfrac{\bar{b}_i}{\bar{d}_i} : \bar{d}_i < 0 \}$

Para $\lambda \in [\, \underline{\lambda}^k , \bar{\lambda}^k \,]$ a solução é

$$\begin{cases} z_F = \bar{b}_F + \lambda\, \bar{d}_F , \; z_T = 0 \\ w_F = 0 , \; w_T = \bar{b}_T + \lambda\, \bar{d}_T \end{cases}$$

Passo 2: Se $s \in F$ faça $F = F - \{ s \}$, $T = T \cup \{ s \}$,

$\underline{\lambda}^{k+1} = \bar{\lambda}^k$, $k = k+1$ e volte ao Passo 1.

Se $s \in T$ calcule $\bar{m}_{ss}$.

Se $\bar{m}_{ss} > 0$ faça $F = F \cup \{ s \}$, $T = T - \{ s \}$,

$\underline{\lambda}^{k+1} = \bar{\lambda}^k$, $k = k+1$ e volte ao Passo 1.

Se $\bar{m}_{ss} = 0$ vá para Passo 3.

Passo3: Determine $\bar{M}_{FS}$.

Se $\bar{M}_{FS} \geq 0$, $\lambda = \bar{\lambda}^k$ é o valor máximo. Termine.

Caso contrário determine

$$\frac{\bar{b}_r + \bar{\lambda}^k \bar{d}_r}{\bar{m}_{rs}} = \min \{ -\frac{\bar{b}_i + \bar{\lambda}^k \bar{d}_i}{\bar{m}_{is}} : \bar{m}_{is} < 0 \;\; e \; i \in F \}$$

Faça $F = F - \{ r \} \cup \{ s \}$, $T = T - \{ s \} \cup \{ r \}$,

$\underline{\lambda}^{k+1} = \bar{\lambda}^k$, $k = k+1$ e volte ao Passo 1.

Devido ao significado físico das variáveis intervenientes, é necessário impor que as variáveis z_i não possam decrescer com o aumento do parâmetro de carga λ. Esse facto conduziria ao chamado fenómeno de descarregamento da estrutura, que não se deve verificar. Mas teremos garantia de que essas variáveis não decrescem se for $\bar{d}_F \geq 0$ em qualquer iteração. Por isso o algoritmo de Cottle deve ser modificado de modo a garantir que se tenha sempre $\bar{d}_F \geq 0$.

Na primeira iteração tem-se $d_s < 0$ e $m_{ss} > o$. Portanto $\bar{d}_s > 0$ e $F = \{ s \}$. Suponhamos, por hipótese de indução, que $\bar{d}_F \geq 0$ na iteração k. Então o índice s escolhido no passo 1 é um elemento de T pois $\bar{d}_s < 0$.

Dois casos podem acontecer

(i) $\bar{m}_{ss} > 0$

O índice s vai ser incluído no conjunto F. Então

$$\bar{\bar{d}}_s = -\frac{\bar{d}_s}{\bar{m}_{ss}} > 0 \quad e \quad \bar{\bar{d}}_i = \bar{d}_i - \frac{\bar{m}_{is}}{\bar{m}_{ss}} \bar{d}_s \text{ para } i \in F \text{ e } i \neq s$$

Ora nada impede que alguns destes valores se possam tornar negativos. No entanto, se tal acontecer, como $\bar{M}_{FF}$ é definida positiva, é possível mostrar [64] que basta retirar de F os elementos correspondentes às componentes negativas de $\bar{d}$. Mas isso corresponde a anular variáveis z_i que entretanto eram positivas, o que não pode acontecer fisicamente. Então, se se fizer a mudança de variável

$$\Delta z_i = z_i - \bar{z}_i$$

em que $\bar{z}_i$ é o valor positivo da variável z_i antes da operação pivotal ($\bar{z}_i = \bar{b}_i + \bar{\lambda}_i^k \bar{d}_i$), o valor da variável z_i fica fixo em $\bar{z}_i$. Esta mudança de variável corresponde a alterar o termo independente do programa (8.17).

(ii) $\bar{m}_{ss} = 0$

Neste caso a variável z_r deixa de ser básica pelo que é necessário fazer

$$\Delta z_r = z_r - \bar{z}_r$$

e a variável z_s passa a básica na situação descrita no caso (i).

A implementação do algoritmo que acabou de ser descrito pode ser feita de modo muito semelhante ao que já foi referido no capítulo 2. É necessário resolver sistemas com as matrizes A e M. Como ambas são simétricas positivas definidas é possível usar os métodos descritos nesse capítulo. Além disso, é possível pensar em aplicar o método de Cottle directamente ao LCP (8.17) começando com as variáveis u todas na base e seguindo um processo análogo ao que foi descrito no capítulo 2 quando se abordou o problema das variáveis sem restrição de sinal.

Na tabela 8.1 apresentam-se alguns resultados computacionais obtidos num computador CDC CYBER 170 na resolução de alguns problemas teste. Nessa tabela, m representa a dimensão da matriz A e n a dimensão de M. A implementação que foi usada para obter esses resultados está descrita em [67] e não corresponde à que acabámos de referir. Com efeito, não se utilizou o esquema estático que tem sido usado em todas as implementações referidas até agora, mas sim um esquema em que cada nova coluna é acrescentada no fim da matriz a factorizar. Isso acarreta a necessidade de se fazerem reinversões frequentes, pois a esparsidade da factorização deixa de ser assegurada.

m	n	IT	T
27	48	4	3.8
28	72	11	5.7
30	48	14	5.9
36	192	62	62.6
36	128	22	16.6
46	144	35	28.2
76	240	51	78.8
36	480	36	49.8

Tabela 8.1

Quando se formulou o código para a resolução do programa linear complementar paramétrico pensou-se em utilizar um pequeno computador pessoal (286). Optou-se por isso por usar ficheiros para guardar informação intermédia. O acesso a esses ficheiros é sempre demorado, pelo que os tempos de execução apresentados nessa tabela são elevados tendo em consideração o problema que está a ser resolvido.

3 O Problema de Contacto Unilateral em Meios Elásticos

Em certos problemas de análise de estruturas não se conhecem as condições de contorno da estrutura, pois, muitas vezes, as zonas de contacto são desconhecidas.

Tal como no problema anteriormente descrito, o equilíbrio, a compatibilidade, as relações constitutivas do material podem ser traduzidas pelas relações de equilíbrio (8.2) e (8.11), pelas relações de compatibilidade (8.1) e pelas relações constitutivas (8.12).

Por outro lado, é corrente que a mesma estrutura tenha que ser estudada para diferentes solicitações previstas em regulamentos oficiais. Cada solicitação abordada conduz a um novo conjunto de relações, com os mesmos operadores, com excepção do vector correspondente à acção exterior c_0. Daí resulta que os conjuntos das variáveis básicas nas diversas soluções não são muito diferentes.

Neste tipo de problemas pressupõe-se que o meio tem um comportamento elástico linear (não se considera a componente plástica), que as deformações são infinitesimais e que não há atrito nas ligações. Esta última condição leva a que seja possível expressar o contacto em função dos deslocamentos. Mais uma vez é utilizado o método dos elementos finitos para discretizar a estrutura em estudo.

As solicitações exteriores fazem com que se desenvolvam na estrutura tensões que com elas se relacionam através da relação de equilíbrio em cada ponto nodal

$$f = f_0 + C^T\sigma \tag{8.20}$$

em que f é o vector das forças nodais e f_0 o vector da solicitação. A estrutura estará em equilíbrio quando for f = 0 nos pontos nodais interiores ao meio estrutural. Nas zonas de contorno do meio estrutural f é igual às forças exteriores directamente aplicadas que assumem o significado de reacções se existe apoio.

Por sua vez as tensões e deformações estão relacionadas pela lei de Hooke

$$\sigma(x) = R(x)\, \varepsilon(x)$$

em que R(x) é a chamada matriz de rigidez do material calculada no ponto x. Discretizando a estrutura, obtém-se uma relação matricial do mesmo tipo, em cada ponto nodal

$$\sigma = R\, \varepsilon \tag{8.21}$$

Por outro lado as deformações e os deslocamentos também se relacionam através das equações de compatibilidade

$$\varepsilon(x) = D\,u(x)$$

em que D é um operador diferencial de primeira ordem. Discretizando, através do método dos elementos finitos, obtém-se

$$\varepsilon - \varepsilon_o = C\,u \tag{8.22}$$

em que C é uma matriz real que relaciona os deslocamentos com as tensões nos pontos nodais. Esta equação garante que as deformações não provoquem vazios ou sobreposições na estrutura, ou seja, assegura a compatibilidade.

Nas relações (8.20) e (8.22) devem-se ter em atenção as condições fronteira que não digam respeito às zonas de contacto unilateral. Nestas zonas tem que se ter em conta vários condicionalismos que derivam da própria geometria do problema. Assim:

(i) A componente normal dos deslocamentos não pode ser negativa

$$u_n(x) \geq 0 \tag{8.23}$$

uma vez que havendo contacto o deslocamento no outro sentido é impossível;

(ii) A componente normal da tensão não pode ser negativa

$$\sigma_n(x) \geq 0 \tag{8.24}$$

visto que na superfície de contacto não pode haver tracção;

(iii) A tensão só se desenvolve quando há contacto

$$u_n(x)\,\sigma_n(x) = 0 \tag{8.25}$$

visto que um deslocamento nulo significa que há contacto e um deslocamento positivo equivale a ausência de contacto.

Discretizando as relações (8.23), (8.24) e (8.25), obtém-se

$$u_n \geq 0\ ,\ f_n \geq 0\ ,\ u_n^T\ f_n = 0,$$

em que u_n são os deslocamentos nos pontos nodais e f_n são as forças nodais na direcção normal à superfície de contacto (resultantes das tensões $\sigma_n(x)$).

Juntando todas as relações tem-se

$$\varepsilon = C\,u$$

$$0 = C^T\sigma + f_o$$

$$\sigma = R\,\varepsilon$$

$$u_n \geq 0\ ,\ f_n \geq 0\ ,\ u_n^T\ f_n = 0$$

Se se numerarem convenientemente as variáveis pode-se definir o vector genérico dos deslocamentos $u = \begin{bmatrix} u \\ u_n \end{bmatrix}$ e o das forças $f = \begin{bmatrix} 0 \\ f_n \end{bmatrix}$. Além disso, manipulando convenientemente as relações escritas obtém-se

$$\begin{bmatrix} 0 \\ f_n \end{bmatrix} = \begin{bmatrix} f_o \\ f_{on} \end{bmatrix} + E \begin{bmatrix} u \\ u_n \end{bmatrix} \tag{8.26}$$

$$u_n \geq 0\ ,\ f_n \geq 0\ ,\ u_n^T\ f_n = 0$$

com $E = C^T R\,C$.

Obteve-se assim um BLCP com algumas variáveis sem restrição de sinal. Além disso a matriz E é simétrica positiva definida, pois C é de característica completa e, pelo seu significado físico, R é simétrica positiva definida.

O BLCP (8.23) pode pois ser resolvido pelo método bloco descrito no capítulo 3. Neste caso, como já foi referido, há necessidade de resolver BLCPs com a mesma matriz e com termos independentes diferentes. Desses diferentes problemas supõe-se que o conjunto das variáveis positivas na solução não é muito diferente. Assim, uma vez resolvido o primeiro de um grupo de BLCPs, o conjunto F correspondente à solução é utilizado como conjunto inicial para a resolução do BLCP seguinte.

O método foi testado no estudo do comportamento de algumas estruturas elementares. Ensaiou-se uma viga de grande altura apoiada numa superfície plana, através de contacto unilateral, solicitada por 3 forças concentradas de igual valor. Testou-se também uma laje enviesada apoiada em dois bordos opostos unilaterais. A solicitação considerada para este caso foi uma carga concentrada aplicada no centro da laje. A resolução do BLCP permitiu determinar a extensão dos cantos levantados da laje. Em qualquer dos casos o meio estrutural foi discretizado numa malha de elementos finitos isoparamétrica de 8 nós.

Na tabela 8.2 apresentam-se os resultados relativos à resolução dos BLCPs referidos pelo algoritmo Bloco.

N	STOR	TS	#F	IT	NO	T
242	11838	1.50	229	9	23.19	5.13
			230	2	9.60	1.32
			229	2	9.99	1.17
306	10106	2.99	293	7	42.08	9.05
			293	4	23.02	4.55
			294	4	14.74	2.57

Tabela 8.2

Como se pode constatar, além de só ser necessário efectuar uma vez a fase simbólica inicial para cada grupo de BLCPs, a resolução dos BLCPs a seguir ao primeiro é bastante mais rápida do que a deste, pois beneficia de o conjunto F inicial ser bastante próximo do conjunto correspondente à solução.

Como resultado físico, foi possível determinar a extensão dos bordos levantados da laje conforme se mostra na figura 8.3.

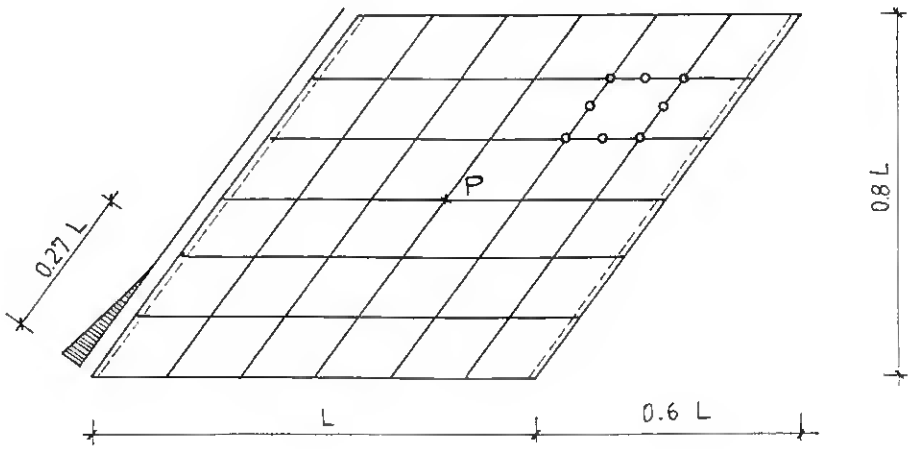


Figura 8.3

CONCLUSÃO

Nesta tese debruçámo-nos sobre os algoritmos pivotais principais para a resolução do Problema Linear Complementar (LCP) Monótono e para algumas das suas generalizações, nomeadamente para o Problema Linear Complementar com Limites (BLCP). Assim começámos por estudar os métodos pivotais principais simples já existentes para o LCP, tendo desenvolvido implementações eficientes para problemas de grandes dimensões e matrizes esparsas. Além disso realizámos vários testes com esses algoritmos, tendo concluído a necessidade do desenvolvimento de algoritmos que efectuassem operações pivotais principais por blocos. Nesse sentido desenvolvemos um algoritmo desse tipo que se mostrou bastante eficiente para a resolução de LCPs de grandes dimensões.

Numa segunda fase do nosso trabalho apresentámos as extensões dos algoritmos pivotais principais simples e por blocos para a resolução do BLCP. Experiências computacionais com os algoritmos existentes permitiram-nos obter conclusões semelhantes às do LCP.

Métodos iterativos de partição e de gradientes projectados têm vindo a ser sugeridos para a resolução de LCPs e BLCPs de grandes dimensões. Nesta tese estudámos e efectuámos algumas experiências com esses processos. Os resultados obtidos não foram muito abonatórios, tendo esses algoritmos mostrado ser pouco competitivos com o método pivotal principal por blocos sugerido nesta tese.

A aplicação dos métodos pivotais principais simples e por blocos em alguns problemas de programação quadrática convexa foi também estudada com algum detalhe. Os resultados obtidos permitiram concluir da grande utilidade das estratégias pivotais principais por blocos neste tipo de problemas. Finalmente apresentámos também duas aplicações da complementaridade no âmbito da análise de estruturas, que mostra o bom funcionamento das metodologias desenvolvidas nesta tese na resolução de problemas reais de grande importância.

Esta tese prova assim que os algoritmos pivotais principais são uma grande metodologia para a resolução de problemas complementares monótonos e suas aplicações. A utilidade da complementaridade na resolução de programas não lineares e de desigualdades variacionais por técnicas sequenciais tem vindo a ser mencionada em alguns trabalhos recentes (ver por exemplo o livro de Cottle e al [12] sobre esse assunto). Pensamos que os algoritmos pivotais principais por blocos poderão ter aí um papel muito importante. A utilidade desse tipo de algoritmos na determinação de pontos estacionários de alguns programas não convexos foi também mencionada recentemente [19]. Parece-nos que essas duas áreas poderão no futuro realçar ainda mais a importância dos processos discutidos nesta tese.

REFERÊNCIAS

[1] **A. A. Ahac, J. J. Buoni e D. D. Olessky,** *Stable LU - factorization of H - matrices,* Linear Algebra and its Applications 99 (1988) 97-110.

[2] **B. H. Ahn,** *Iterative methods for linear complementarity problems with upper-bounds on primary variables,* Mathematical Programming 26 (1983) 295-315.

[3] **M. A. Ajize, A. Jennings,** *A robust incomplete Choleski-gradient algorithm,* International Journal for Numerical Methods in Engineering 20 (1984) 949-966.

[4] **J. M. Bennett**, *Triangular factors of modified matrices,* Numerische Mathematik 7 (1965) 217 - 221.

[5] **R. Chandrasekaran,** *A special case of the complementarity pivot problem,*Opsearch 7 (1970) 263-268.

[6] **T. F. Coleman e L. A. Hulbert,** *A direct active set algorithm for large sparse quadratic programs with bounds,* Mathematical Programming 45 (1989) 373-406.

[7] **R. W. Cottle,** *Monotone solution of the parametric linear complementarity problem*, Mathematical Programming 3 (1972) 210-214.

[8] **R. W. Cottle,** *Manifestations of the Schur complement,* Linear Algebra and its Applications 8 (1974) 189-211.

[9] **R. W. Cottle e G. B. Dantzig,** *Complementary pivot theory of mathematical programming,* em "Mathematics of the Decision Sciences", editado por G. B. Dantzig e A. F. Veinott Jr., American Mathematical Society, Providence (1968) 115-136.

[10] **R. W. Cottle e M. S. Goheen,** *A special class of large quadratic programs,* em "Nonlinear Programming 3", editado por O. L. Mangasarian, R. R. Meyer e S. M. Robinson, Academic Press, New York (1978) 361-390.

[11] **R. W. Cottle, G. H. Golub e R. S. Sacher,** *On the solution of large, structured linear complementarity problems: the block partitioned case,* Applied Mathematics and Optimization 4 (1978) 347-363.

[12] **R. W. Cottle, J. S. Pang e R. E. Stone,** *The Linear Complementarity Problem,* Academic Press, New York, 1992.

[13] **D. E. Crabtree e E. V. Haynsworth,** *An identity for the Schur Complement of a matrix,* Proceedings of American Mathematical Society 22 (1969) 364-366.

[14] **C. W. Cryer**, *The efficient solution of linear complementarity problems for tridiagonal Minkowski matrices,* ACM Transactions on Mathematical Software 9 (1983) 199-214.

[15] **R. S. Dembo e U. Tulowitzski,** *On the minimization of quadratic functions subject to box constraints*, Technical Report, Department of Computer Science, Yale University (1983).

[16] **I. S. Duff, R. G. Grimes e J. G. Lewis,** *Sparse Matrix test problems,* ACM Transactions on Mathematical Software 15 (1989) 1-14.

[17] **J. P. Dussault, J. A. Ferland e B. Lemaire,** *Convex quadratic programming with one constraint and bounded variables,* Mathematical Programming 36 (1986) 90-104.

[18] **Y. Fathi,** *Computational complexity of LCPs associated with positive definite symmetric matrices,* Mathematical Programming 17 (1979) 335-344.

[19] **A. M. Faustino,** *Complementaridade Linear e Aplicação em Optmização Global,* Tese de Doutoramento, Universidade de Coimbra, 1993.

[20] **L. M. M. Fernandes,** *Resolução de Problemas Lineares Complementares Monótonos de Grandes Dimensões,* Tese de Mestrado, Universidade de Lisboa, 1992.

[21] **M. Fiedler e V. Ptâk,** *On matrices with non positive off-diagonal elements and positive principal minors,* Czechoslovak Mathematical Journal 12 (1962) 382- 400.

[22] **R. Fletcher e M. P. Jackson,** *Minimization of a quadratic function subject only to upper and lower bounds,* Journal of Institute Mathematics and Applications 14 (1974) 159-174.

[23] **R. Fletcher e M. J. D. Powell,** *On the modification of LDL^T factorizations,* Mathematics of Computation 28 (1974) 1067-1087.

[24] **A. George e J. W. H. Liu,** *Computer Solution of Large Positive Definite Systems,* Prentice-Hall, Engewood Cliffs, New Jersey, 1981.

[25] **P. E. Gill, W. Murray e M. H. Wright,** *Practical Optimization,* Academic Press, New York, 1981.

[26] **R. Glowinski,** *Finite elements and variational inequalities,* MRC Technical Report 1885, Mathematics Research Center, University of Wisconsin-Madison, 1978.

[27] **G. H. Golub e C. F. Van Loan,** *Unsymmetric positive definite linear systems,* Linear Algebra and its Applications 28 (1979) 85-98.

[28] **N. I. M. Gould,** *An algorithm for large-scale quadratic programming,* a aparecer em IMA Journal of Numerical Analysis.

[29] **R. L. Graves,** *A principal pivoting simplex algorithm for linear and quadratic programming,* Operations Research 15 (1967) 482-494.

[30] **F. G. Gustavson,** *Two fast algorithms for sparse matrices: multiplication and permuted transposition,* ACM Transactions of Mathematical Software 4 (1978) 250-269.

[31] **P. T. Harker e J. S. Pang,** *A damped - Newton method for the linear complementarity problem,* Lecture Notes in Applied Mathematics 26 (1990) 265-284.

[32] **R. Helgason, S. Kennington e H. Lall,** *A polynomial bounded algorithm for a singly constrained quadratic program,* Mathematical Programming 18 (1980) 338-343.

[33] **J. J. Júdice,** *Classes of matrices for the linear complementarity problem,* Linear Algebra and Applications 54 (1984) 122-125.

[34] **J. J. Júdice e G. Mitra,** *Reformulation of mathematical programming problems as linear complementarity problems and investigation of their solution methods,* Journal of Optimization Theory and Applications 57 (1988) 123-149.

[35] **J. J. Júdice e F. M. Pires,** *Direct methods for the solution of linear complementarity problems with symmetric positive semi-definite matrices,* Investigação Operacional 6 (1986) 115-152.

[36] **J. J. Júdice e F. M. Pires,** *Bard-type methods for the linear complementarity problem with symmetric positive definite matrices,* IMA Journal of Mathematics Applied in Business & Industry 2 (1988/89) 51-68.

[37] **J. J. Júdice e F. M. Pires,** *Direct methods for convex quadratic programs subject to box constraints,* Investigação Operacional 9 (1989) 23-56.

[38] **J. J. Júdice e F. M. Pires,** *A comparison between direct and iterative methods for solving large-scale convex quadratic programs on the simplex,* Pesquisa Operacional 9 (1989) 55-78.

[39] **J. J. Júdice e F. M. Pires,** *Tratamento numérico de variáveis sem restrição em programas quadráticos com apenas limites inferiores e superiores,* Actas das XV Jornadas Luso-Espanholas de Matemática 4 (1990) 445-450.

[40] **J. J. Júdice e F. M. Pires,** *A Bard-type method for a generalized linear complementarity problem with a nonsingular M-matrix,* Naval Research Logistics 37 (1990) 279-297.

[41] **J. J. Júdice e F. M. Pires,** *Solution of large-scale strictly convex linear complementarity problems,* Investigação Operacional 11 (1991) 31-51.

[42] **J. J. Júdice e F. M. Pires,** *Solution of large-scale separable strictly convex quadratic programs on the simplex,* Linear Algebra and its Applications 170 (1992) 214-220.

[43] **J. J. Júdice e F. M. Pires,** *Principal pivoting methods for the symmetric monotone linear complementarity problem,* em " Proceedings of the NATO ASI: Computers for solving Linear Algebraic Systems ", editado por E. Spedicato e M. T. Vespucci (1992) 167-185

[44] **J. J. Júdice e F. M. Pires,** *A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems,* a aparecer em Computers and Operations Research.

[45] **J. J. Júdice e F. M. Pires,** *A polynomial method for a generalized linear complementarity problem with a nonsingular M-matrix,* IMA Journal of Mathematics Applied in Business and Industry 4 (1992) 211- 224.

[46] **J. J. Júdice e F. M. Pires,** *A basic- set algorithm for a generalized linear complementarity problem,* Journal of Optimization Theory and Applications 74 (1992) 391-411.

[47] **E. F. Kaasschieter,** *A FORTRAN implementation of the preconditioned method of conjugate gradients,* Report 85-33, Department of Mathematics and Informatics, Delft University of Tecnology (1985).

[48] **E. L. Keller,** *The general quadratic optimization problem,* Mathematical Programming 5 (1973) 311-337.

[49] **E. Klafzky e T. Terlaky,** *Some generalizations of the criss cross method for quadratic programming*, a aparecer em Linear Algebra and its Applications.

[50] **M. Kojima, S. Mizuno e A. Yoshise,** *A polynomial - time algorithm for a class of linear complementarity problems,* Mathematical Programming 44 (1989) 1-26.

[51] **M. Kojima, N. Megiddo, T. Noma e A. Yoshise,** *A unified approach to interior point algorithms for linear complementarity problems: a summary,* Operations Research Letters 10 (1991) 247-254.

[52] **M. Kostreva,** *Block pivot methods for solving the complementarity problem,* Linear Algebra and its Applications 21 (1978) 207-215.

[53] **H. Kuhn e W. Tucker,** *Nonlinear programming,* em "Second Berkeley Symposium in Mathematical Statistics and Probability", editado por J. Neyman, University of California Press, California (1951) 80-90.

[54] **K. H. Law,** *Sparse matrix factor modification in structural reanalysis,* International Journal for Numerical Methods in Engineering 21 (1985) 37-63.

[55] **K. H. Law e S. J. Fenves,** *A node-addition model for symbolic factorization,* ACM Transactions on Mathematical Software 12 (1986) 37-50.

[56] **C. E. Lemke,** *On complementarity pivot theory*, em "Mathematics of the Decision Sciences", editado por by G. B. Dantzig e A. F. Veinott Jr., American Mathematical Society, Providence (1968) 95-114.

[57] **Y. Y. Lin e J. S. Pang,** *Iterative methods for large convex quadratic programs: a survey,* SIAM Journal on Control and Optimization 25 (1987) 383-411.

[58] **O. L. Mangasarian,** *Equivalence of the complementarity problem to a system of nonlinear equations,* SIAM Journal of Applied Mathematics 31 (1976) 89-92.

[59] **O. L. Mangasarian,** *Sparsity-preserving SOR algorithms for separable quadratic and linear programming,* Computers and Operations Research 2 (1984) 105-112.

[60] **Markowitz,** *The elimination form of the inverse and its application to linear programming,* Management Science 3 (1957) 255-269.

[61] **C. E. Massonat e M. A. Save,** *Plastic Analysis and Design,* Blaisdel Press, New York (1965).

[62] **S. R. McCammon,** *On complementary pivoting,* Ph D Thesis, Rensselaer Polytechnic Institute, Troy, New York (1970).

[63] **J. J. Moré e G. Toraldo,** *Algorithms for bound constrained quadratic programming problems,* Numerische Mathematik 55 (1989) 377-400.

[64] **K. G. Murty,** *Note on a Bard-type scheme for solving the complementarity problem,* Opsearch 11 (1974) 123-130.

[65] **K. G. Murty,** *Linear Complementarity, Linear and Nonlinear Programming,* Heldermann Verlag, Berlin (1988).

[66] **A. A. S. Neves,** *Análise de Placas e Cascas em Regime Elastoplástico Aplicando a Metodologia da Programação Matemática,* Tese de Doutoramento, Universidade do Porto (1988).

[67] **A. S. Neves, A. M. Faustino, F. M. Pires e J. J. Júdice,** *Elastoplastic analysis of structures and linear complementarity,* em "O.R. Models on Microcomputers", editado por J. D. Coelho e L. V. Tavares, Elsevier Science Publishers B. V., North-Holland (1986) 217-228.

[68] **D. P. O'Leary,** *A generalized conjugated gradient algorithm for solving a class of quadratic programming problems,* Linear Algebra and its Applications 34 (1980) 371-399.

[69] **D. P. O'Leary,** *Sparse quadratic programming without matrix updating,* Technical Report 1200, Computer Science Departement, University of Maryland (1982).

[70] **J. S. Pang,** *On a class of least-element complementarity problems,* Mathematical Programming 16 (1979) 325-347.

[71] **J. S. Pang,** *A new and efficient algorithm for a class of portfolio selection problems,* Operations Research 28 (1980) 13-27.

[72] **J. S. Pang,** *Methods for quadratic programming: a survey,* Computers and Chemical Engineering 5 (1983) 583-594.

[73] **J. S. Pang, I. Kaneko e W. P. Hallman,** *On the solution of some (parametric) linear complementarity problems with applications to portfolio selection, structural engineering and actuarial graduation,* Mathematical Programming 16 (1979) 325-347.

[74] **J. S. Pang e S. C. Lee,** *A parametric linear complementarity technique for the computation of equilibrium prices in a single commodity spatial model,* Mathematical Programming 20 (1981) 81-102.

[75] **P. M. Pardalos e N. Kovoor,** *An algorithm for a singly constraint class of quadratic programs subject to lower and upper bounds,* Mathematical Programming 46 (1990) 321-328.

[76] **F. M. Pires e A. A. Serra Neves,** *A complementaridade linear na resolução de estruturas com apoios unilaterais,* Investigação Operacional 9 (1989) 89-96.

[77] **S. Pissanetzky,** *Sparse Matrix Technology,* Academic Press, New York (1984).

[78] **B. Ramarao e C. M. Shetty,** *Application of disjunctive programming to the linear complementarity problem,* Naval Research Logistics Quarterly 31 (1984) 589-600.

[79] **R. H. W. Sargent,** *An efficient implementation of the Lemke algorithm and its extension to deal with upper and lower bounds on the variables,* Mathematical Programming Study 7 (1978) 36-54.

[80] **J. A. Tomlin,** *Robust Implementation of Lemke's method for the linear complementarity problem, Mathematical Programming Study 7 (1978) 55-60.*

[81] **B. Wendroff,** *Theoretical Numerical Analysis,* Academic Press, New York (1966).

[82] **N. Wirth,** *Algorihtms + Data Structures = Programs,* Prentice - Hall, Englewood Cliffs, New Jersey (1976).

[83] **E. K. Yang e J. W. Tolle,** *A class of methods for solving large convex quadratic programs subject to box constraints,* Working Paper, Management Sciences Departement, University of Massachussetts at Boston (1985).