MÉTODOS DE PROGRAMAÇÃO GEOMÉTRICA

Edmundo Sérgio Spoto

Orientador
Prof. Dr. Marcos N. Arenales

Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Computação.

São Carlos
Maio/1988

A G R A D E C I M E N T O S

Agradeço ao Prof. Dr. Marcos Arenales, pela sua total dedicação a esse trabalho, e pela sua amizade.

Agradeço aos meus colegas do ICMSC, aos meus professores e a todos os funcionários, que acima de tudo foram amigos .

A todos colegas do Departamento de Informática da UEM, do CTC e da PPG pelo total apoio.

Agradeço aos meus pais, irmãos e a minha esposa pelo carinho e incentivo.

Agradeço ao CNPq, CAPES e FAPESP pelo apoio financeiro.

Agradeço a todas amizades que se firmaram durante o periodo de pós-graduação e aqueles que, direta ou indiretamente, colaboraram para a realização deste trabalho.

E a Deus, por tudo .

ÍNDICE

CAPÍTULO IV

Melhoria da Solução Antes da Condensação

CAPÍTULO V

Experiências Computacionais

CAPÍTULO VI

APÊNDICE A

APÊNDICE B

Programas e Rotinas

APÊNDICE C

Protensão em Vigas de Concreto Modelado por Programação Geométrica.

RESUMO

Este trabalho tem a finalidade de apresentar a teoria e uma classe de métodos da Programação Geométrica, que têm sido consideradas de grande utilidade para solução de problemas de Engenharia.

A classe de métodos estudados é baseada num conceito de condensação, que visa aproximar um programa posinomial a um programa linear. Este último pode ser resolvido usando técnicas da Programação Linear.

A idéia de projeção de Censor-Lent foi usada, em combinação com os métodos estudados, numa tentativa de acelerá-los.

Finalmente apresentamos as experiências numéricas e conclusões.

A B S T R A C T

This work presents a theory and a class of Geometric Programming Methods, which have been considered of great interest in solving Engineering problems.

The studied methods are based on condensation concept, which approaches a "posynomial" program to linear program. The later one can be solved by Linear Programming methods.

In order to acelerate the methods, the Censor-Lent projection idea was used.

Numerical experiences and conclusions are presented.

CAPÍTULO I

Programação Geométrica (P.G.) :

1.1 - Introdução :

A Programação Geométrica surgiu na década de 60 como uma importante ferramenta a ser utilizada na resolução de problemas de programação não-linear. Teve seu início em 1961, quando Clarence Zener, diretor de Ciências na Westinghouse Corporation, descobriu que muitos problemas de projetos de engenharia eram formados por uma simples soma de componentes de custos e poderiam ser determinados quase por inspeção, usando as condições disponíveis. Richard Duffin, professor de matemática, havia desenvolvido a teoria da dualidade, com aplicação voltada para problemas de programação não-linear. Ao entrar em contato com o trabalho desenvolvido pelo Professor Zener, oportunidade que teve durante uma visita a Westinghouse Corporation, empenhou-se em entender o método matemático de Zener, aplicando suas teorias recentemente desenvolvidas. O nome 'Programação Geométrica', teve sua origem através do desenvolvimento matemático elaborado por Duffin, que consistia na utilização da desigualdade entre a média aritmética-geométrica.

A técnica da Programação Geométrica, para problemas irrestritos, garante uma solução fácil, toda vez que o grau de dificuldade, assim chamado por Duffin e Zener, for zero; onde:

Grau de dificuldade = Número de termos - (No. de variáveis+1)

Nossa motivação inicial para estudos de PG decorreu de contatos com pesquisadores da área de Engenharia Civil. No Apendice C escrevemos uma aplicação que não temos vistos na literatura.

A fim de entendermos os princípios básicos desses desenvolvimentos, considere o seguinte problema de projeto de engenharia altamente simplificado.

Exemplo 1.1 - [13].

Deseja-se construir uma caçamba cilíndrica para ser usada em transportes de matéria-prima em um processo de fabricação. A caçamba é cilíndrica e deverá carregar $500\pi\ m^3$ de matéria-prima. O custo de fabricação da caçamba será 2 UM/m^2 (UM : Unidade Monetária) de material utilizado (área de superficie da caçamba). O objetivo é determinar a altura h e o raio r, da caçamba circular, satisfazendo assim as especificações do projeto e minimizar os custos de construção.

Esse é um problema de resolução imediata, pois o volume será $\pi r^2 h$, e o custo da construção da caçamba (considerando apenas o material) será $2(2\pi rh + \pi r^2)$ UM. Como a especificação do volume é de 500π, logo :

$$500\pi = \pi r^2 h$$

ou

$$h = \frac{500}{r^2}$$

Para minimizar o custo, teremos um problema que é :

$$\text{minimizar} \quad 4\,\Pi\, r\left[\frac{500}{r^2}\right] + 2\,\Pi\, r^2 \qquad (1.1)$$

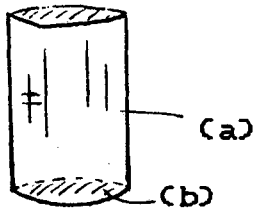
fazendo $c_1 = 2000\,\Pi$ e $c_2 = 2\,\Pi$, a equação (1.1) pode ser escrita por :

$$\text{minimizar} \quad g = \underbrace{c_1 r^{-1}}_{a} + \underbrace{c_2 r^2}_{b} \qquad (1.2)$$

onde o grau de dificuldade é zero (2-(1+1)=0). Note que a construção da caçamba é composta pela soma de duas parcelas de custos :

(a) Custo da borda lateral da estrutura e

(b) Custo da superfície inferior da estrutura.



A solução desse problema é facilmente encontrada através do cálculo diferencial:

$$\frac{\partial g}{\partial r} = -c_1 r^{-2} + 2\,c_2 r = 0$$

resultando

$$r^* = \left[\frac{c_1}{2c_2}\right]^{1/3} \quad e \quad h^* = 500\left[\frac{2c_2}{c_1}\right]^{2/3} \qquad (1.3)$$

logo o custo ótimo para a construção será :

$$g^* = c_1\left[\frac{c_1}{2c_2}\right]^{-1/3} + c_2\left[\frac{c_1}{2c_2}\right]^{2/3}$$

ou

$$g^* = \underbrace{(2)^{1/3}\,[c_1^{2/3}c_2^{1/3}]}_{(a)} + \underbrace{(2)^{-2/3}\,[c_1^{2/3}\,c_2^{1/3}]}_{(b)}$$

Observe que a razão entre as parcelas de g^* independe de c_1 e c_2. Esse fato ocorre devido a $c_1^{2/3}c_2^{1/3}$ ser comum aos dois termos (a) e (b). Portanto se ocorrer uma mudança nos valores de c_1 e c_2, a contribuição relativa das parcelas (a) e (b) será a mesma. Analisando algebricamente as componentes (a) e (b), notamos então que a parcela (a) é sempre duas vezes maior que (b). Na otimalidade, a parcela de contribuição da componente (b) é 1/3 e a parcela de (a) é 2/3. Uma observação importante é que se outra distribuição nas parcelas de (a) e (b) forem tomadas, o custo será superior.

Podemos usar essa relação da seguinte maneira :

Seja δ_i a proporção com que a parcela i contribui com a função objetivo ($\delta_1 = \frac{a}{g}$ e $\delta_2 = \frac{b}{g}$).

$$\delta_1 + \delta_2 = 1$$

Se além disso

$$(-1)\delta_1 + (2)\delta_2 = 0$$

(onde -1 é o expoente de r na primeira parcela e 2 é o expoente de r na segunda parcela).

Então

$$\delta_1 = 2/3 \quad e \quad \delta_2 = 1/3$$

Assim

$$\frac{2}{3} g^* = c_1 r^{-1} \quad e \quad \frac{1}{3} g^* = c_2 r^2$$

logo,

$$\frac{3}{2} c_1 r^{-1} = 3 c_2 r^2$$

ou

$$r^* = \left[\frac{c_1}{2c_2}\right]^{1/3}$$

este resultado está de acordo com os obtidos inicialmente e abre um caminho novo para resoluções de problemas de programação matemática. Antes havia uma necessidade de primeiro pesquisar o vetor ótimo depois determinar o custo ótimo resultante, já na PG pesquisa-se primeiro a distribuição de parcelas do custo ótimo e então determina-se os valores das variáveis na solução. Este método pode muitas vezes ser usado na solução de problemas de projetos de engenharia.

Posteriormente, novas técnicas foram desenvolvidas.

Elmor Peterson, um aluno de Duffin, colaborou com a extenção do método, incluindo restrições com desigualdades. Em 1967, Duffin, Peterson e Zener publicaram o primeiro texto, agora um clássico, de Programação Geométrica [13]. Este livro contém todos os desenvolvimentos teóricos, até a data, com alguns exemplos ilustrativos.

Os trabalhos de Duffin, Peterson e Zener foram restritos a problemas com coeficientes positivos. Embora esta classe abranja um grande número de problemas na engenharia, a teoria da Programação Geométrica foi generalizada para problemas com coeficientes negativos por Passy e Wilde.

Para formalizar as idéias iniciais de Zener, faremos uma descrição matemática [13,23].

Considere o seguinte problema (primal):

$$\begin{aligned} &\textbf{minimizar} && g_o(x) \\ &\text{sujeito a} && x_j > 0 \qquad j = 1,2,\ldots,N \end{aligned} \tag{1.4}$$

onde a função $g_o(x)$ pode ser escrita na forma $g_o = \mu_1+\mu_2+..+\mu_{T_o}$ sendo que

$$\mu_i = \theta_i x_1^{\varphi_{i1}} \ldots x_n^{\varphi_{in}} = \theta_i \prod_{j=1}^{N} x_j^{\varphi_{ij}} \tag{1.5}$$

os coeficientes θ_i são positivos, os expoentes φ_{ij} são constantes reais, e as variáveis x_j são positivas. Essa função μ_i é chamada de posinomial.

Desigualdade entre as médias aritmética e geométrica

Dado T_o termos $U_1, U_2, \ldots, U_{T_o}$, a *média aritmética é* :

$$\frac{1}{T_o}(U_1+U_2+\ldots+U_{T_o}) \qquad (1.6)$$

e sua média geométrica :

$$(U_1\ U_2 \ldots U_{T_o})^{1/T_o} \qquad (1.7)$$

A média aritmética de um conjunto de números positivos sempre será maior ou igual a sua média geométrica; isto é, (1.6) $\geq$ (1.7). Em particular para $T_o = 2$, temos

$$\frac{1}{2}(U_1 + U_2) \geq (U_1 U_2)^{1/2} \qquad (1.8)$$

onde U_1 e U_2 são números positivos. Para provar isto, faremos:

$$(U_1 - U_2)^2 \geq 0$$

ou

$$U_1^2 - 2\,U_1 U_2 + U_2^2 \geq 0$$

adicionando $4\,U_1\,U_2$ em ambos os lados temos

$$(U_1 + U_2)^2 \geq 4\,U_1 U_2$$

logo

$$\frac{1}{2} U_1 + \frac{1}{2} U_2 \geq (U_1 U_2)^{1/2}$$

note que a igualdade ocorre se e somente se $U_1 = U_2$. Generalizando para quatro números positivos, temos :

$$\frac{1}{4} U_1 + \frac{1}{4} U_2 + \frac{1}{4} U_3 + \frac{1}{4} U_4 \geq \left(\frac{U_1+U_2}{2}\right)^{1/2} \left(\frac{U_3+U_4}{2}\right)^{1/2}$$

aplicando (1.8) duas vezes, temos

$$\frac{1}{4} U_1 + \frac{1}{4} U_2 + \frac{1}{4} U_3 + \frac{1}{4} U_4 \geq U_1^{1/4} U_2^{1/4} U_3^{1/4} U_4^{1/4} \qquad (1.9)$$

onde a igualdade é satisfeita se e somente se $U_1 = U_2 = U_3 = U_4$. Podemos ainda, obter uma desigualdade entre as médias ponderadas, fazendo $U_2 = U_3 = U_4$, escrevendo (1.9) como :

$$\frac{1}{4} U_1 + \frac{3}{4} U_2 \geq U_1^{1/4} U_2^{3/4}$$

onde as parcelas de contribuição do primeiro e do segundo termo são $\frac{1}{4}$ e $\frac{3}{4}$ respectivamente.

Como vemos, podemos formar uma variedade de desigualdades geométricas. Considerando diferentes pesos w_i, obtemos diferentes desigualdades. Uma forma geral entretanto é :

$$w_1 U_1 + w_2 U_2 + \ldots + w_{To} U_{To} \geq U_1^{w_1} U_2^{w_2} \ldots U_{To}^{w_{To}} \qquad (1.10)$$

onde U_i , $w_i > 0$ para todo i, e $\sum_{i=1}^{To} w_i = 1$.

Fazendo $\mu_i = w_i U_i$ (i = 1,...,To), então de (1.10) temos :

$$g_o(x) = \mu_1 + \ldots + \mu_{To} \geq \left(\frac{u_1}{w_1}\right)^{w_1} \ldots \left(\frac{u_{To}}{w_{To}}\right)^{w_{To}} = V(w,x) \qquad (1.11)$$

Os lados esquerdo $g(x)$ e direito $V(w,x)$ de (1.11) são denominados função primal e função pré-dual respectimamente. A função pré-dual $V(w,x)$ é uma função produto dos termos μ_i elevado a potência w_i. Substituindo $\mu_i(x)$, dado por (1.5), em (1.11) a função pré-dual se escreve por :

$$V(w,x) = \left(\frac{\theta_1}{w_1}\right)^{w_1}\left(\frac{\theta_2}{w_2}\right)^{w_2} \ldots \left(\frac{\theta_{To}}{w_{To}}\right)^{w_{To}} x_1^{d_1} x_2^{d_2} \ldots x_N^{d_N} \qquad (1.12)$$

onde d_j são combinações lineares

$$d_j = \sum_{i=1}^{To} w_i \varphi_{ij} \qquad j = 1,2,\ldots,N \qquad (1.13)$$

dos expoentes φ_{ij}. Podemos escolher os pesos w_i, tal que todos expoentes d_j sejam zero, então a função pré-dual $V(w,x)$ não mais dependerá das variáveis x_j, e assim teremos a função dual:

$$v(w) = \left(\frac{\theta_1}{w_1}\right)^{w_1} \left(\frac{\theta_2}{w_2}\right)^{w_2} \ldots \left(\frac{\theta_{To}}{w_{To}}\right)^{w_{To}} = \prod_{i=1}^{To} \left(\frac{\theta_i}{w_i}\right)^{w_i} \qquad (1.14)$$

o vetor peso w deve satisfazer as condições de normalidade e de ortogonalidade, respectivamente se $\sum_{i=1}^{To} w_i = 1$ e $d_j = 0$ (j=1,...,N).

Assim,

$$g_o(x) \geq v(w)$$

$$\text{sujeito a} \quad \sum_{i=1}^{To} w_i \varphi_{ij} = 0 \qquad j = 1,2, \ldots ,N$$

Note que o problema primal deseja minimizar $g_o(x)$ e para cada w tal que $\sum_{i=1}^{To} w_i \varphi_{ij} = 0$, o valor da função $v(w)$ é um limitante inferior a $g_o(x)$.

O problema dual consiste em determinar o maior limitante inferior a $g_o(x)$:

1.1.1 - Problema Dual

$$\max \quad v(w)$$

$$\text{suj. a} \quad \sum_{i=1}^{To} w_i \varphi_{ij} = 0 \qquad j = 1,\ldots,N \qquad (1.15)$$

$$\sum_{i=1}^{To} w_i = 1$$

$$w_i \geq 0$$

Note que se o grau de dificuldade for zero, então o sistema (1.15) é quadrado. Se as equações forem linearmente independentes então o problema dual tem uma única solução.

Teorema 1.1 :

O máximo da função dual é igual ao mínimo da função primal.

Prova : Suponha que o ponto $x^o = (x_1^o, x_2^o, \ldots, x_N^o)$ seja o ponto mínimo de $g(x)$ onde $x_i^o > 0$ para todo i. A derivada de $g(x)$ em relação a cada variável x_j, no ponto x^o é zero; isto é,

obteremos n equações do tipo :

$$0 = x_j^o \frac{\partial g(x^o)}{\partial x_j} = \sum_{i=o}^{To} x_j^o \frac{\partial \mu_i(x^o)}{\partial x_j} = \sum_{i=o}^{To} \mu_i(x^o)\ \varphi_{ij}$$

Dividindo estas equações por $g(x^o)$, teremos :

$$\bar{\omega}_i = \mu_i(x^o) / g(x^o) > 0 \qquad i = 1,2,\ldots,To \qquad (1.16)$$

obtendo assim

$$0 = \sum_{i=1}^{To} \bar{\omega}_i\ \varphi_{ij} \qquad j = 1,2,\ldots,N \qquad (1.17)$$

de onde, $\bar{\omega}$ satisfaz a condição $d_j = 0$ (j=1,2,..,N). Assumindo agora os termos em (1.16) onde $\Sigma\ \mu_i = g$, teremos que $\bar{\omega}_i$ satisfaz também as condições $\sum_{i=1}^{To} \bar{\omega}_i = 1$.

Mostraremos agora que $v(\bar{\omega}) = g(x^o)$. De (1.16) $\mu_i(x) = \bar{\omega}_i\ g(x^o)$ para todo i ou ainda

$$\theta_i x_1^{o\varphi_{i1}} \ldots x_N^{o\varphi_{iN}} = \bar{\omega}_i\ g(x^o) \quad \text{para todo } i$$

Assim, para qualquer $\bar{\omega}_i$ tal que :

$$(\theta_i)^{\bar{\omega}_i} \prod_{j=1}^{N} (x_j^o)^{\varphi_{ij}\bar{\omega}_i} = (\bar{\omega}_i\ g(x^o))^{\bar{\omega}_i}$$

ou

$$(\theta_i / \bar{\omega}_i)^{\bar{\omega}_i} \prod_{j=1}^{N} (x_j^o)^{\bar{\omega}_i \varphi_{ij}} = g(x^o)^{\bar{\omega}_i}$$

portanto, multiplicando, desta maneira, todos To termos da equação juntas, teremos :

$$\prod_{i=1}^{To} (\theta_i/\bar{\omega}_i)^{\bar{\omega}_i} \prod_{j=1}^{N} (x_j^o)^{\sum_{i=1}^{To} \bar{\omega}_i \varphi_{ij}} = g(x^o)^{\sum_{i=1}^{To} \bar{\omega}_i} = g(x^o)$$

mas de (1.17), temos :

$$\prod_{i=1}^{To} (\theta_i/\bar{\omega}_i)^{\bar{\omega}_i} = g(x^o)$$

portanto

$$v(\bar{\omega}) = g(x^o) \qquad (1.18)$$

- Determinação do ponto de mínimo :

Pelo fato que $g(x^o)\bar{\omega}_i = \mu_i(x^o)$ $(i=1,2,..,To)$ e do teorema anterior (1.18), temos :

$$v(\bar{\omega})\ \bar{\omega}_i = \mu_i(x^o) \qquad i=1,2,..,To \qquad (1.19)$$

isto é, a solução ótima do problema primal é determinada resolvendo um sistema de equações lineares, tomando se o logaritmo de (1.19). Para exemplificar considere o seguinte problema didático que tem sido utilizado nas literaturas [13,14,23]. O problema ilustra que esse sistema de equações pode ser transformado em um simples sistema de equações lineares.

Exemplo 1.2 :

Suponha que 400 m^3 de cascalho devam ser transportados de uma margem a outra, em um rio de grande largura. Para isso usaremos uma caixa aberta de comprimento x_1, largura x_2 e altura x_3. As laterais maiores e a base da caixa custarão 10 UM/m^2 e o restante da caixa (x_2x_3) custa 20 UM/m^2. A única despesa de transporte da caixa será de 0,1 UM para cada viagem completada. Calcular o preço mínimo para transportar os 400m^3 de cascalho, e quais são as dimensões ótima da caixa, que satisfaçam o problema.

Solução :

É fácil de ver que o custo total em UM é

$$\min \quad g(x) = \frac{400 * 0.1}{x_1 x_2 x_3} + 2*20\,(x_2x_3) + 2 * 10(x_1x_3) + 10(x_1x_2)$$

ou

$$g(x) = \frac{40}{x_1x_2x_3} + 40\,x_2x_3 + 20\,x_1x_3 + 10\,x_1x_2 \qquad (1.20)$$

observe que o grau de dificuldade desse problema é zero.

A função dual de (1.20) é :

$$v(w) = \left[\frac{40}{w_1}\right]^{w_1} \left[\frac{40}{w_2}\right]^{w_2} \left[\frac{20}{w_3}\right]^{w_3} \left[\frac{10}{w_4}\right]^{w_4} \qquad (1.21)$$

onde os pesos satisfazem as equações (veja (1.15)) :

$$\begin{aligned} d_1 &= -w_1 + w_3 + w_4 = 0 \\ d_2 &= -w_1 + w_2 + w_4 = 0 \\ d_3 &= -w_1 + w_2 + w_3 = 0 \end{aligned} \qquad (1.22)$$

e

$$w_1 + w_2 + w_3 + w_4 = 1 \qquad (1.23)$$

As quatro equações acima (1.22) e (1.23), tem uma única solução w' dada por

$$w_1' = \frac{2}{5}\ ,\ w_2' = \frac{1}{5}\ ,\ w_3' = \frac{1}{5}\ ,\ w_4' = \frac{1}{5} \qquad (1.24)$$

substituindo esses valores em (1.21) obtemos o custo mínimo

$$v(w') = \left(\frac{40}{2/5}\right)^{2/5} \left(\frac{20}{1/5}\right)^{1/5} \left(\frac{20}{1/5}\right)^{1/5} \left(\frac{10}{1/5}\right)^{1/5} = 100 \text{ UM}$$

devemos escrever a proporção com que cada parcela que a função objetivo contribui para o custo mínimo (veja (1.19)) :

$$\begin{aligned} \left(\frac{2}{5}\right) 100 &= 40\, x_1^{-1} x_2^{-1} x_3^{-1} \\ \left(\frac{1}{5}\right) 100 &= 40\, x_2\, x_3 \\ \left(\frac{1}{5}\right) 100 &= 20\, x_1\, x_3 \\ \left(\frac{1}{5}\right) 100 &= 10\, x_1\, x_2 \end{aligned} \qquad (1.25)$$

Tomando o logaritmo em cada equação de (1.25), e fazendo $z_1 = \log x_1$, $z_2 = \log x_2$ e $z_3 = \log x_3$, veremos que (z_1', z_2', z_3') satisfaz o sistema linear.

$$
\begin{aligned}
0 &= -z_1 - z_2 - z_3 \\
-\log 2 &= \quad z_2 + z_3 \\
0 &= z_1 \quad + z_3 \\
\log 2 &= z_1 + z_2
\end{aligned}
$$

Esse sistema tem solução única $z_1' = \log 2$, $z_2' = 0$ e $z_3' = -\log 2$. Portanto a solução ótima será $x_1' = 2$, $x_2' = 1$ e $x_3' = \frac{1}{2}$. Outros problemas são apresentados [13,14,23], dando uma ampla visão do método.

A Programação Geométrica que vamos estudar é uma classe particular da programação não-linear (PNL), onde a função objetivo e as restrições são definidas por funções do tipo :

$$g(x) = \sum_{i=1}^{T} \theta_i \prod_{j=1}^{N} x_j^{\varphi_{ij}} \qquad (1.26)$$

onde T é o número de termos e N o número de variáveis da função $g(x)$. Os coeficientes θ_i e os expoentes φ_{ij} são números reais, e as variáveis $x_1, x_2, \ldots, x_N$ são positivas. Embora a PG seja um subconjunto da programação matemática, ela abrange uma gama enorme dos modelos não lineares estudados.

Dizemos que $g(x)$ é uma função posinomial, quando todos os θ_i's forem positivos, caso contrário, dizemos que $g(x)$ é uma função signomial ou geométrica generalizada.

O termo $\mu_i(x)$ é chamado de monômio :

$$\mu_i(x) = \theta_i \prod_{j=1}^{N} x_j^{\varphi_{ij}} \qquad (1.27)$$

Observe que a PG em seu início, trabalhava apenas com problemas irrestritos [10,13,23], Peterson colaborou com o seu

desenvolvimento introduzindo problemas com restrições, onde mais tarde outros pesquisadores ampliaram essas técnicas para problemas signomiais.

Um problema de programação geométrica é definido por [13] :

PG :

$$\begin{aligned} \min \quad & g_0'(x') \\ \text{suj. a} \quad & g_k'(x') \leq 1 \qquad k = 1,2,..,K \\ & x_j' > 0 \qquad\quad j = 1,2,..,N \end{aligned} \tag{1.28}$$

onde $g_k'(x) = \sum_{i=1}^{Tk} \theta_{ki} \prod_{j=1}^{N} x_j^{\varphi_{kij}}$ para $k = 0,1,..,K$ e Tk é o número de monômios de g_k , $x' = (x_1, x_2, .., x_N)$.

Equivalentemente, colocando $g_0'(x') \leq x_0$, e supondo $g_k'(x') > 0$ para todo x' factível podemos escrever (1.28) da seguinte forma :

PG :

$$\begin{aligned} \min \quad & x_0 \\ \text{suj. a} \quad & g_k(x) \leq 1 \qquad k = 0,1,..,K \\ & 0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned} \tag{1.29}$$

considerando aqui $x = (x_0, x_1, x_2, \ldots, x_N)$, $g_0(x) = x_0^{-1} g_0'(x')$ e $g_k(x) = g_k'(x')$ e os limites $\underline{x}_j$ e $\bar{x}_j$ podem ser artificiais. Em geral, estes limites surgem, naturalmente, do problema físico. Esta forma será adotada no desenvolvimento dos métodos.

1.2 - Programação Geométrica Posinomial (PGP)

Como diz o próprio nome, estaremos trabalhando apenas com funções posinomiais, isto é, os coeficientes θ_i da função

(1.26) serão todos positivos, não havendo, aqui, nenhum caso com θ_i negativo.

O estudo da programação geométrica, separando os casos PGP e PGG (Programação Geométrica Generalizada ou Signomial), para efeito de teoria e métodos, é muito conveniente, pois o PGP possui propriedades desejáveis; por exemplo, embora em geral, um PGP seja não convexo ($g(x) = x^{1/2}$), pode desfrutar das propriedades de problemas convexos, pois podemos transformá-los num problema convexo mediante a transformação de variáveis, como :

$$x_j = e^{y_j}$$

Desta forma o problema (1.28) pode ser escrito por :

$$\begin{aligned} \text{min} \quad & \sum_{i=1}^{T_0} \theta_{oi}\, e^{\sum_{j=1}^{N} \varphi_{oij} y_j} \\ \text{suj a} \quad & \sum_{i=1}^{T_k} \theta_{ki}\, e^{\sum_{j=1}^{N} \varphi_{kij} y_j} \leq 1 \quad k = 1,2,..,K \end{aligned} \tag{1.30}$$

que é um problema convexo, pois tanto a função objetivo, como as restrições são combinações positivas de funções convexas. Desta forma, um mínimo local é global e isso não pode ser encontrado no caso signomial, o que veremos mais a seguir.

1.2.1 - Problema Primal :

Com o intuito de simplificar as expressões do problema dual, adotaremos a seguinte forma para o problema Primal (1.28).

$$\begin{aligned} \min \quad & g_o(x) \\ \text{suj. a} \quad & \sigma_k(1 - g_k(x)) \geq 0 \qquad k = 1,2,..,K \\ & x_j > 0 \qquad\qquad\qquad j = 1,2,..,N \end{aligned} \tag{1.31}$$

onde σ_k é uma variável de sinal que vale ± 1. Note que (1.31) é mais geral que (1.28), admitindo-se também restrições do tipo $g_k(x) \geq 1$.

Para efeito de desenvolvimento teórico, a restrição $x_j > 0$, será mantida.

1.2.2 - Problema Dual :

Como no caso irrestrito, os pesos também serão obtidos através dos termos da função objetivo,

$$\delta_{oi} = \mu_{oi}(x^o) / g_o(x^o) \qquad i = 1,2,..,To, \tag{1.32}$$

já os pesos para os termos das restrições serão simplesmente os próprios valores dos termos, isto é :

$$\delta_{ki} = \mu_{ki} = \theta_{ki} \prod_{j=1}^{N} x_j^{\varphi_{kij}} \qquad \begin{cases} k = 1,2,..,K \\ i = 1,2,..,Tk \end{cases} \tag{1.33}$$

onde Tk é o número de termos da késima restrição. Seja $m = \sum_{k=o}^{K} Tk$ o número total de termos.

Note que, se uma restrição de (1.31) for satisfeita com igualdade, então os termos correspondentes devem somar uma unidade.

Faremos agora uma mudança de variável com o intuito de facilitar a construção do problema dual.

Sejam $e^{z_j} = x_j$ $(j=0,1,..,ND$ onde $x_o = g_o(x)$.

Tomando o logaritmo natural, o problema original se escreve equivalentemente por :

$$
\begin{aligned}
\min \quad & z_o \\
\text{suj.a} \quad & -z_o + \sum_{j=1}^{N} \varphi_{oij}\, z_j = \ln(\delta_{oi}/\theta_{oi}) \qquad i = 1,2,..,T_o \\
& \sigma_k \sum_{j=1}^{N} \varphi_{kij}\, z_j = \sigma_k \ln(\delta_{ki}/\theta_{ki}) \qquad \begin{matrix} k = 1,2,..,K \\ i = 1,2,..,Tk \end{matrix} \\
& \sum_{i=1}^{To} \delta_{oi} = 1 \qquad (1.34) \\
& \sigma_k(1 - \sum_{i=1}^{Tk} \delta_{ki}) \geq 0 \\
& \delta_{ki} > 0 \qquad k = 1,2,..,K \ \text{ e } \ i = 1,2,..,Tk
\end{aligned}
$$

A função Lagrangeana para o problema acima é :

$$
\begin{aligned}
L(z,\delta,\lambda,w) = {} & z_o - \lambda_o \Big(1 - \sum_{i=1}^{To} \delta_{oi} \Big) \\
& - \sum_{i=1}^{To} w_{oi} \Big[\ln\big(\delta_{oi} / \theta_{oi} \big) + z_o - \sum_{j=1}^{N} \theta_{oij}\, z_j \Big] \\
& - \sum_{k=1}^{K} \sum_{i=1}^{Tk} w_{ki}\, \sigma_k \Big[\ln\big(\delta_{ki} / \theta_{ki} \big) - \sum_{j=1}^{N} \varphi_{kij} z_j \Big] \\
& - \sum_{k=1}^{K} \lambda_k\, \sigma_k \Big(1 - \sum_{i=1}^{Tk} \delta_{ki} \Big) \qquad (1.35)
\end{aligned}
$$

O problema dual é formulado por [17] :

$$
\max_{\substack{\lambda \in \mathbb{R}^{K+1} \\ w \in \mathbb{R}^{M}}} \Big\{ \min_{\substack{z \in \mathbb{R}^{N+1} \\ \delta \in \mathbb{R}^{M}}} L(z,\delta,\lambda,w) \Big\} \qquad (1.36)
$$

com $\lambda_k \geq 0$.

Para expressarmos a função dual :

$$V(\lambda, w) = \min_{z,\delta} \; L\,(z, \delta, \lambda, w) \qquad (1.37)$$

em função das variáveis duais, usaremos o seguinte teorema, que escreve, no ponto de mínimo, parcialmente as variáveis primais em função das variáveis duais :

Teorema 1.2 :

Dados os multiplicadores Lagrangeanos ótimos λ e w em (1.35), a minimização de (1.37) fornece :

(a) $\lambda_o = 1$;

(b) $\delta_{oi} = w_{oi} \qquad i = 1,2,\ldots,To$

(c) $\delta_{ki} = w_{ki} \Big/ \sum_{i=1}^{T_k} w_{ki}$ se $\lambda_k \neq 0 \quad k = 1,2,\ldots,K$

Prova :

A minimização em (1.37) vem diretamente do cálculo, fazendo-se $\nabla_{(z,\delta)} L = 0$. Para mostrar (a) e (b), fazemos :

$$\frac{\partial L}{\partial z_o} = 1 - \sum_{i=1}^{To} w_{oi} = 0$$

$$\frac{\partial L}{\partial \delta_{oi}} = \lambda_o - \frac{w_{oi}}{\delta_{oi}} = 0 \qquad i = 1,2,..,To \qquad (1.38)$$

de onde temos :

$$w_{oi} = \lambda_o \, \delta_{oi} \qquad i = 1,2,..,To \qquad (1.39)$$

somando sobre i, usando (1.32) segue :

$$1 = \sum_{i=1}^{To} w_{oi} = \lambda_o \sum_{i=1}^{To} \delta_{oi} = \lambda_o$$

assim, $\lambda_o = 1$ e (b) é obtido substituindo $\lambda_o = 1$ em (1.39). A seguir considere a derivada da função Lagrangeana em relação a δ_{ki} ($k \neq 0$).

$$\frac{\partial L}{\partial \delta_{ki}} = - \frac{w_{ki} \sigma_k}{\delta_{ki}} + \lambda_k \sigma_k = 0$$

ou

$$w_{ki} = \lambda_k \delta_{ki} \qquad (1.40)$$

somando sobre i obtemos $\sum_{i=1}^{Tk} w_{ki} = \lambda_k$ desde que : $\lambda_k (1 - \sum_{i=1}^{T_k} \delta_{ki}) = 0$ e se $\lambda_k \neq 0$. Assim, de (1.40) temos :

$$\delta_{ki} = w_{ki} / \lambda_k \, , \qquad \text{se} \quad \lambda_k \neq 0$$

c.q.d.

onde $w_{ki} \geq 0$.

Os pesos w_{ki} são determinados através da derivada da função Lagrangeana em relação a z_j ($j \neq 0$) :

$$\frac{\partial L}{\partial z_j} = \sum_{k=0}^{K} \sum_{i=1}^{Tk} w_{ki} \sigma_k \varphi_{kij} = 0 \qquad \text{com} \quad \sigma_o = 1$$

Note que as variáveis são os multiplicadores Lagrangeanos w_{ki}, e os multiplicadores $w_{oi} = \delta_{oi}$ estão sujeitos à $\Sigma \delta_{oi} = 1$.

Teorema 1.3 :

A função objetivo do problema dual associado a (1.31) é

$$v(w) = \prod_{k=0}^{K} \prod_{i=1}^{Tk} \left(\theta_{ki} \, w_{ki}^{-1} \sum_{j=1}^{Tk} w_{kj} \right)^{\sigma_k w_{ki}} \qquad (1.41)$$

Prova :

Considere a função Lagrangeana (1.35), e usando o teorema 1.2 temos :

$$L(z, \delta, \lambda, w) = \sum_{k=0}^{K} \sum_{i=1}^{Tk} \sigma_k \, w_{ki} \ln \left(\theta_{ki} w_{ki}^{-1} \sum_{j=1}^{Tk} w_{kj} \right)$$
$$+ (z_o - 1)\left(1 - \sum_{i=1}^{To} w_{oi}\right)$$
$$+ \sum_{j=1}^{N} z_j \sum_{k=0}^{K} \sum_{i=1}^{Tk} \sigma_k \, \varphi_{kij} w_{ki} - \sum_{k=1}^{K} \lambda_k \sigma_k \left(1 - \sum_{i=1}^{Tk} w_{ki} / \lambda_k\right)$$

e aplicando $\dfrac{\partial L}{\partial z_j} = 0$, segue :

$$h(w) = \sum_{k=0}^{K} \sum_{i=1}^{Tk} \sigma_k \, w_{ki} \ln\left(\theta_{ki} w_{ki}^{-1} \sum_{j=1}^{Tk} w_{kj} \right)$$
$$= \ln \left(\prod_{k=0}^{K} \prod_{i=1}^{Tk} \theta_{ki} w_{ki}^{-1} \sum_{j=1}^{Tk} w_{kj} \right)^{\sigma_k w_{ki}}$$

Como a função exponencial é crescente, a maximização de $h(w)$ não se altera aplicando a função exponencial. Assim o problema dual pode ser formado por

PGD :

$$\max \; v(w) = \exp [\, h(w) \,] = \prod_{k=0}^{K} \prod_{i=1}^{Tk} \left(\theta_{ki} w_{ki}^{-1} \sum_{j=1}^{Tk} w_{kj}\right)^{\sigma_k w_{ki}} \qquad (1.42)$$

$$\text{suj.a :} \sum_{i=1}^{To} w_{oi} = 1 \qquad \text{(normalidade)} \qquad (1.43)$$

$$\sum_{k=0}^{K} \sum_{i=1}^{Tk} \sigma_k \, \varphi_{kij} \, w_{ki} = 0 \quad j = 1, 2, .., N \text{ (ortog.)} \qquad (1.44)$$

$$w_{ki} \geq 0 \quad k = 0, 1, .., K \quad i = 1, 2, .., Tk \quad \text{(não negativ.)} \qquad (1.45)$$

onde

$$\lambda_k = \sum_{i=1}^{Tk} w_{ki} \qquad (1.46)$$

e w_{ki} são as variáveis duais.

Dembo [10] descreve algumas tentativas de resolver o PGD. A primeira [13] foi de eliminar (N+1) variáveis básicas (o problema tem N+1 restrições das quais se escrevem N+1 variáveis (chamadas básicas) em função das restantes), criando-se um problema reduzido, Programação Geométrica Dual Reduzida (PGDR). Entretanto, Dembo sugere que seria melhor não fazer esta redução e trabalhar com a função-objetivo separável, que pode ser obtida, explicitando a restrição :

$$\lambda_k = \sum_{i=1}^{Tk} w_{ki}$$

tomando o logaritmo de (1.42).

Entretanto, tanto o problema PGD como os demais equivalentes têm um número de características que não permitem a aplicação direta dos Softwares de Programação Não-Linear para determinar uma solução numérica.

A maior dificuldade é a não-diferenciabilidade da função objetivo, em relação às variáveis duais, nos pontos onde algumas variáveis se anulam. Alguns artifícios foram empregados para contornar a não-diferenciabilidade [10].

a) A primeira idéia foi trabalhar com limites artificiais : $w_{ki} \geq \varepsilon > 0$ (ε pequeno). Mas surgiram outros

problemas, como :

i) Hessiana mal condicionada nos pontos próximos da solução;

ii) O algoritmo toma um caminho em zig-zag, até concluir que $w_{ki} = \varepsilon$.

b) Outro caminho foi aproximar a função objetivo (após ter tomado o logaritmo) por uma função quadrática, para os pontos próximos de zero ($0 \leq w_{ki} \leq \varepsilon$). Isto é :

$$w_{ki} \log (\theta_{ki} / w_{ki}) \cong \alpha w_{ki}^2 + \beta w_{ki} \qquad 0 \leq w_{ki} \leq \varepsilon$$

onde

$$\alpha = -1/\varepsilon,$$

$$\beta = \log (\theta_{ki} / \varepsilon) + 1.$$

A vantagem dessa aproximação é o fato da função objetivo obtida, ser contínua e diferenciavel e α e β são escolhidos de tal forma que tanto os valores das derivadas, como os valores das funções $w_{ki} \log (\theta_{ki} / w_{ki})$ e $\alpha w_{ki}^2 + \beta w_{ki}$, sejam iguais para $w_{ki} = \varepsilon$. Os valores das duas funções também são iguais para $w_{ki} = 0$, porém seus gradientes diferem neste ponto. Quando w_{ki} tende a zero, o gradiente de $w_{ki} \log (\theta_{ki} / w_{ki})$ tende a infinito. Já o gradiente de $\alpha w_{ki}^2 + \beta w_{ki}$ é igual a β em $w_{ki} = 0$.

A escolha de ε é mais fácil agora. Deve ser feito um balanço a fim de se evitar valores muito pequenos ou grandes demais para se contornar novos problemas como :

- Matriz Hessiana mal condicionada, para ε pequeno,

- O gradiente da aproximação no ponto $w_{ki} = 0$ é muito diferente do gradiente do objetivo verdadeiro, para ε grande.

Dembo [10], diz que em vários testes realizados com o método, a escolha de $\varepsilon = 10^{-5}$ parece ser razoável.

1.2.3 - Teorema da Dualidade :

Este teorema relaciona o problema primal com o problema Dual [17]. Sejam x^* e w^* soluções ótimas primal e dual, respectivamente, então :

$$g_o(x^*) = v(w^*)$$

e

$$w^*_{ki} = \begin{cases} \mu_{ki}(x^*)/g_o(x^*) & i = 1,2,..,To \\ \lambda^*_k \ \mu_{ki}(x^*) & i = 1,2,..,Tk \\ & k = 1,2,..,K \end{cases} \qquad (1.47)$$

onde $\lambda^*_k = \sum_{i=1}^{Tk} w^*_{ki}$ e $\mu_{ki}(x)$ é um monômio de $g_k(x)$.

A solução primal pode ser obtida de (1.47), tomando-se o logaritmo e obtendo-se um sistema linear de equações em x^*. Novas dificuldades podem surgir caso o sistema seja indeterminado [13], necessitando de novos artifícios para contornar a dificuldade.

A atração inicial pelo problema dual devido às restrições lineares (mostrado nas literaturas, com forte tendência a métodos duais) desapareceram, devido a estas dificuldades. Métodos que tratam diretamente com o primal, usando o conceito de condensação, tiveram melhor sorte. Neste sentido, ficamos mais motivados a estudar tais métodos.

1.3 - Programação Geométrica Generalizada (PGG)

Ao problema (1.29) chamamos de problema de Programação Geométrica Generalizada (ou Signomial) quando pelo menos uma função g_k for do tipo signomial; isto é, se algum dos termos μ_i (i = 0,1,..,T) tiver coeficiente negativo. Neste caso, o problema (1.29) pode ser escrito da seguinte forma :

PGG :

$$\begin{aligned} &\min && x_o \\ &\text{suj. a} && p_k(x) - q_k(x) \leq 1 \qquad k = 0,1,..,K \\ &&& \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned} \tag{1.48}$$

onde $x = (x_o, x_1, .., x_N)^T$. Os monômios negativos estão embutidos em $q_k(x)$, apenas colocando o sinal em evidência. Assim sendo, $p_k(x)$ e $q_k(x)$ são posinômios que definimos como :

$$p_k(x) = \sum_{i=1}^{Tk} u_{ki} = \sum_{i=1}^{Tk} c_{ki} \prod_{j=1}^{N} x_j^{a_{kij}} \tag{1.49}$$

$$q_k(x) = \sum_{i=1}^{Lk} v_{ki} = \sum_{i=1}^{Lk} d_{ki} \prod_{j=1}^{N} x_j^{b_{kij}} \tag{1.50}$$

Os expoentes a_{kij} e b_{kij} são números reais e os coeficientes c_{ki} e d_{ki} são positivos. Por conveniência, podemos reescrever o problema PGG, considerando a forma equivalente para a k-ésima restrição :

$$\frac{p_k(x)}{1 + q_k(x)} \leq 1 \tag{1.51}$$

Note-se que $1 + q_k(x)$ é um posinômio. Assim, o problema pode ser escrito na forma :

PGG :

$$\begin{aligned} &\min \quad x_o \\ &\text{suj. a} \quad \frac{p_k(x)}{1 + q_k(x)} \leq 1 \qquad k = 0,1,..,K \qquad (1.52) \\ &\qquad 0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned}$$

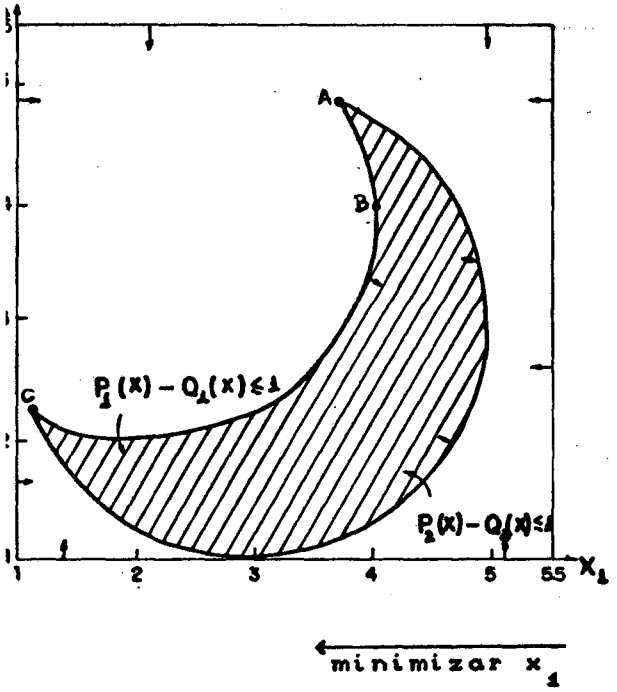
Exemplo 1.4 -

Vejamos agora um exemplo de [2], para ilustrar um problema de Programação Geométrica Generalizada. Considere o seguinte problema :

PGG :

$$\begin{aligned} &\min \quad x_1 \\ &\text{suj. a} \quad p_1(x) - q_1(x) = x_1/4 + x_2/2 - x_1^2/16 - x_2^2/16 \leq 1 \qquad (1.53) \\ &\qquad p_2(x) - q_2(x) = x_1^2/(6x_2) + x_2/6 + 7/(3x_2) - x_1/x_2 \leq 1 \end{aligned}$$

o esboço desse problema é mostrado na fig. 1.1.



Solução de Kuhn-Tucker

A (3.823, 4.823) min. Local

B (4.0, 4.0) Pto. de Inflexão.

C (1.178, 2.178) min. global

fig. 1.1.- Esboço das restrições do exemplo 1.4.

Este exemplo ilustra bem as situações que podem ocorrer num problema PGG : mínimo local (A), mínimo Global (C) e ponto de inflexão (B).

Nos capítulos seguintes retomaremos este exemplo para ilustrar a convergência de métodos numéricos.

CAPÍTULO II

Condensação

2.1 - Condensação de Posinômios

A condensação de um Posinômio consiste em obter um "*monômio tangente*" em torno de um ponto dado, chamado ponto de condensação; ou seja, obter um monômio que coincide com o posinômio, tanto no valor como no gradiente, no ponto de condensação.

Considere-se um posinômio :

$$g(x) = \sum_{i=1}^{T} \mu_i(x) = \sum_{i=1}^{T} \theta_i \prod_{j=1}^{N} x_j^{\varphi_{ij}} \qquad (2.1)$$

. Desigualdade entre as médias aritméticas-geométricas :

A conhecida desigualdade, relacionando as médias aritméticas-geométricas, vista no capítulo anterior :

$$\sum_{i=1}^{T} \delta_i w_i \geq \prod_{i=1}^{T} (w_i)^{\delta_i}$$

onde $\sum_{i=1}^{T} \delta_i = 1$, $w_i, \delta_i \geq 0$, é usada para justificar alguns métodos de programação geométrica.

Para nosso estudo, uma forma mais conveniente pode ser obtida definindo-se :

$$\mu_i = \delta_i w_i$$

então,

$$\sum_{i=1}^{T} \mu_i \geq \prod_{i=1}^{T} \left(\frac{\mu_i}{\delta_i}\right)^{\delta_i}$$

Assim,

$$g(x) = \sum_{i=1}^{T} \mu_i(x) \geq \prod_{i=1}^{T} \left(\frac{\mu_i}{\delta_i}\right)^{\delta_i} = \hat{c} \prod_{j=1}^{N} x_j^{\hat{a}_j} = \hat{g}(x) \qquad (2.2)$$

onde

$$\hat{c} = \prod_{i=1}^{T} \left(\frac{\theta_i}{\delta_i}\right)^{\delta_i} \qquad (2.3)$$

e

$$\hat{a}_j = \sum_{i=1}^{T} \delta_i \varphi_{ij} \qquad (2.4)$$

Como podemos ver, o último termo em (2.2) é um monômio, a saber, o posinômio $g(x)$ passa a ser limitado superiormente por um monômio para todo $x > 0$.

O elemento δ_i será usualmente definido como a parcela com que cada termo do posinômio contribui com a função, ou seja, dado um ponto $\hat{x} > 0$, então :

$$\delta_i = \delta_i(\hat{x}) = \frac{\mu_i(\hat{x})}{g_i(\hat{x})} \qquad i = 1,2,..,T \qquad (2.5)$$

Observe que $\delta_i > 0$ e $\sum_{i=1}^{T} \delta_i = 1$. Esse monômio :

$$\hat{g}(x) = \hat{c} \prod_{j=1}^{N} x_j^{\hat{a}_j} \qquad (2.6)$$

limitante de $g(x)$ é denominado "condensação" de $g(x)$ no ponto $\hat{x}$.

Resumindo, $g(x)$ calculado de (2.3) a (2.6) satisfaz :

$$g(x) \geq \hat{g}(x) \qquad (2.7)$$

para qualquer $x > 0$ e $\hat{x} > 0$.

Essa condensação satisfaz também as seguintes propriedades :

i) Substituindo (2.5) em

$$\hat{g}(x) = \prod_{i=1}^{T} \left(\frac{\mu_i(x)}{\delta_i}\right)^{\delta_i}$$

temos :

$$\hat{g}(x) = \prod_{i=1}^{T} \left(\frac{\mu_i(x)}{\mu_i(\hat{x})}\, g(\hat{x})\right)^{\delta_i}$$

Para $x = \hat{x}$ obtemos :

$$\hat{g}(x) = \prod_{i=1}^{T} (g(\hat{x}))^{\delta_i} = (g(\hat{x}))^{\sum_{i=1}^{T} \delta_i} = g(\hat{x})$$

portanto

$$\hat{g}(x) = g(\hat{x}) \quad \text{para} \quad x = \hat{x} \qquad (2.8)$$

ii) Derivando $g(x)$ e $\hat{g}(x)$ em relação a um certo x_k obteremos :

$$\frac{\partial\, g(x)}{\partial\, x_k} = \frac{\sum_{i=1}^{T} \theta_i \varphi_{ik} \prod_{j=1}^{N} x_j^{\varphi_{ij}}}{x_k} = \frac{\sum_{i=1}^{T} \varphi_{ik} \mu_i(x)}{x_k} \qquad (2.9)$$

fazendo-se $x = \hat{x}$, de (2.5) e (2.9) segue que :

$$\frac{\partial\, g(\hat{x})}{\partial\, x_k} = \frac{\sum_{i=1}^{T} \varphi_{ik}\ \mu_i(\hat{x})}{\hat{x}_k} = \frac{g(\hat{x}) \sum_{i=1}^{T} \varphi_{ik}\ \delta_i}{\hat{x}_k}$$

e de (2.4) :

$$\frac{\partial\, g(\hat{x})}{\partial\, x_k} = \frac{g(\hat{x})\ \hat{a}_k}{\hat{x}_k}$$

Por outro lado, derivando $\hat{g}(x)$ em relação a x_k temos :

$$\frac{\partial\, \hat{g}(x)}{\partial\, x_k} = \frac{\hat{a}_k\ \hat{c} \prod_{j=1}^{N} x_j^{\hat{a}_j}}{x_k} = \frac{\hat{a}_k\ \hat{g}(x)}{x_k} \qquad (2.10)$$

De (2.8) segue que :

$$\frac{\partial\, \hat{g}(\hat{x})}{\partial\, x_k} = \frac{\hat{a}_k\ g(\hat{x})}{\hat{x}_k}$$

logo,

$$\nabla\, g(\hat{x}) = \nabla\, \hat{g}\,(\hat{x}) \qquad (2.11)$$

Ver figura a seguir :

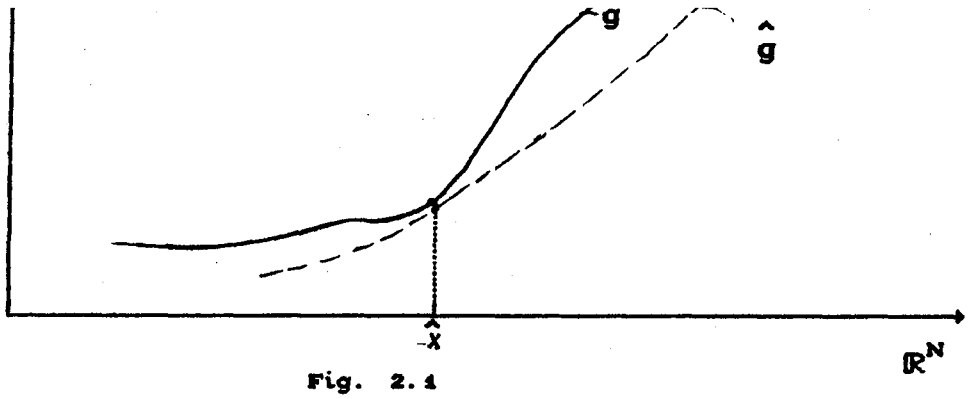


Fig. 2.1

2.2 - Problemas Condensados :

2.2.1 - Problemas Posinomiais Condensados

Vamos considerar, inicialmente, um problema de Programação Geométrica Posinomial.

PGP :

$$\begin{aligned} \min \quad & x_o \\ \text{suj.a} \quad & g_k(x) \leq 1 \qquad k = 0,1,..,K \\ & 0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned} \tag{2.12}$$

onde $g_k(x)$ é uma função posinomial para $(k = 0,1,..,K)$.

Dado um ponto $\hat{x} > 0$ qualquer, condensamos cada restrição $g_k(x)$ $(k=0,1,..,K)$ neste ponto, conforme as equações apresentadas de (2.3) a (2.6), obtemos :

PGP($\hat{x}$) :

$$\begin{aligned} \min \quad & x_o \\ \text{suj.a} \quad & \hat{g}_k(x) \leq 1 \qquad k = 0,1,..,K \\ & 0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned} \tag{2.13}$$

onde $\hat{g}_k(x)$ $(k = 0,1,..,K)$ são restrições do tipo monomiais.

Tomando a função logaritmo natural, podemos transformar equivalentemente $PGP(\hat{x})$ em um programa linear :

$PL(\hat{x})$:

$$\begin{aligned} \min \quad & z_o \\ \text{suj. a} \quad & \sum_{j=o}^{N} \hat{a}_{kj}\, z_j \leq -\ln \hat{c}_k \qquad k = 0,1,..,K \qquad (2.14) \\ & \underline{z}_j \leq z_j \leq \bar{z}_j \qquad\qquad j = 0,1,..,N \end{aligned}$$

onde

$$z_j = \ln x_j \qquad j = 0,1,..,N$$

e lembrando que $\hat{g}_k(x) = \hat{c}_k \prod_{j=1}^{N} x_j^{\hat{a}_{kj}}$ em (2.13).

Quando condensamos um posinômio em um ponto $\hat{x} > 0$ qualquer, na verdade estamos aumentando a região de factibilidade em relação à região de factibilidade do PGP, assim, temos uma propriedade muito importante : é que a região de factibilidade do problema (2.12) está contida na região de factibilidade do problema (2.13), isto é, dado um ponto x^F factível para PGP, ou seja :

$$g_k(x^F) \leq 1 \qquad k = 0,1,..,K$$

de (2.7) segue :

$$\hat{g}_k(x^F) \leq g_k(x^F) \leq 1 \qquad k = 0,1,..,K \qquad (2.15)$$

De onde x^F é factível para $PGP(\hat{x})$; mas nem todo x^F factível ao $PGP(\hat{x})$ será factível ao PGP. A figura 2.2, ilustra

essa propriedade :

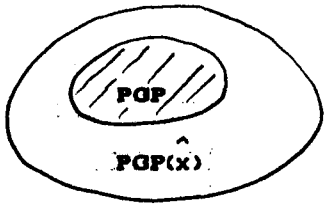


fig. 2.2. : A região de factibilidade do PGP está contida na região de factibilidade do $PGP(\hat{x})$.

Desta forma, se x^* resolve $PGP(\hat{x})$, e se for factível em PGP, então será ótimo também ao PGP. Devido à propriedade de programas posinomiais serem equivalentes a programas convexos, o ótimo será global.

No capítulo seguinte, veremos como gerar uma sequência de soluções, usando (2.14), que convirja para a solução de (2.12).

2.2.2 - Problemas Signomiais Condensados (PGG)

Consideramos, agora, um problema de Programação Geométrica Generalizada (PGG), já partindo da forma equivalente dada em (1.52) :

PGG :

$$\begin{aligned} \min \quad & x_o \\ \text{suj.a} \quad & \frac{p_k(x)}{1 + q_k(x)} \leq 1 \qquad k = 0,1,..,K \qquad (2.16) \\ & 0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned}$$

onde $p_k(x)$ e $1 + q_k(x)$ são posinômios já vistos anteriormente.

Em princípio, condensamos o posinômio $1 + q_k(x)$, em um ponto $\hat{x} > 0$ escolhido inicialmente (factível), e obtemos um monômio condensado que denominamos de $\hat{q}_k(x)$, tornando o problema na forma :

$PGG(\hat{x})$:

$$\begin{aligned} \min \quad & x_o \\ \text{suj.a} \quad & \frac{p_k(x)}{\hat{q}_k(x)} \leq 1 \qquad k = 0,1,..,K \qquad (2.17) \\ & 0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned}$$

O problema $PGG(\hat{x})$ apresenta algumas propriedades importantes, como :

i) O $PGG(\hat{x})$ é um problema do tipo PGP, visto que as restrições $p_k(x)/\hat{q}_k(x)$ resultam em restrições posinomiais (como $g_k(x)$). Isso é fácil de ver, pois um posinômio dividido por um monômio qualquer resultará em um posinômio.

ii) Ao contrário do que ocorreu na condensação feita no PGP, qualquer ponto x^F, que satisfaça as restrições do $PGG(\hat{x})$, ou seja :

$$\frac{p_k(x^F)}{\hat{q}_k(x^F)} \leq 1 \qquad k = 0,1,..,K$$

de (2.7) :

$$1 + q_k(x^F) \geq \hat{q}_k(x^F)$$

segue que :

$$\frac{p_k(x^F)}{1 + q_k(x^F)} \leq \frac{p_k(x^F)}{\hat{q}_k(x^F)} \leq 1 \qquad k = 0,1,..,K \qquad (2.18)$$

a desigualdade (2.18) implica que o conjunto de soluções factíveis de $PGG(\hat{x})$ está contido no conjunto das soluções factíveis do PGG e, portanto, a solução ótima no $PGG(\hat{x})$ será um ponto factível para o PGG, mas não necessariamente será o ótimo final. A figura 2.3. ilustra essa propriedade.

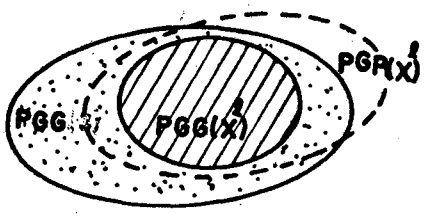


Fig. 2.3

O problema $PGG(\hat{x})$, que é agora um problema de Programação Geométrica Posinomial, pode ser novamente condensado, como na secção 2.1. Utilizaremos essa estratégia no desenvolvimento de métodos, que veremos no capítulo III.

Com as propriedades i) e ii) temos a possibilidade de construir os métodos para o PGG, recorrendo à solução do PGP.

Exemplo 2.1 -

Reconsideremos o exemplo 1.4 do capítulo anterior, para ilustrar os problemas condensados . Dado o seguinte PGG :

PGG :

$$\begin{aligned} \min \quad & x_1 \\ \text{suj.a} \quad & x_1/4 + x_2/2 - x_1^2/16 - x_2^2/16 \le 1 \\ & x_1^2/(6x_2) + x_2/6 + 7/(3x_2) - x_1/x_2 \le 1 \\ & 1 \le x_j \le 5.5 \qquad j = 1,2 \end{aligned} \tag{2.19}$$

Note que a função objetivo é linear e o problema está no formato padrão de PGG . O ponto factível assumido foi :

$$\hat{x}_1 = 4.0 \quad e \quad \hat{x}_2 = 4.5$$

Inicialmente descrevemos as restrições (2.19) na forma (2.16);

$$\begin{aligned} \frac{p_1(x)}{1 + q_1(x)} &= \frac{x_1/4 + x_2/2}{1 + x_1^2/16 + x_2^2/16} \le 1 \\ \frac{p_2(x)}{1 + q_2(x)} &- \frac{x_1^2/(6x_2) + x_2/6 + 7/(3x_2)}{1 + x_1/x_2} \le 1 \end{aligned} \tag{2.20}$$

Considere agora, a condensação dos posinômios $1 + q_1(x)$ e $1 + q_2(x)$ no ponto $\hat{x} = (\hat{x}_1, \hat{x}_2)$

$$(1\hat{+}q_1)(x) = \left(\frac{1}{\delta_{11}}\right)^{\delta_{11}} \left(\frac{x_1^2/16}{\delta_{12}}\right)^{\delta_{12}} \left(\frac{x_2^2/16}{\delta_{13}}\right)^{\delta_{13}} \tag{2.21}$$

$$(1\hat{+}q_2)(x) = \left(\frac{1}{\delta_{21}}\right)^{\delta_{21}} \left(\frac{x_1/x_2}{\delta_{22}}\right)^{\delta_{22}} \tag{2.22}$$

onde os pesos δ_{ki} são calculados em $\hat{x}$ conforme (2.5). Por exemplo :

$$\delta_{11} = \frac{1}{1+q_1(\hat{x})} = \frac{1}{3.26} = 0.306$$

$$\delta_{12} = \frac{x_1^2/16}{1+q_2(\hat{x})} = \frac{1}{3.26} = 0.306 \qquad (2.23)$$

substituindo $\hat{x} = (4.0, 4.5)$ temos :

$$(1+\hat{q}_1)(x) = 0.435\, x_1^{0.612}\, x_2^{0.775}$$

$$(1+\hat{q}_2)(x) = 1.997\, x_1^{0.471}\, x_2^{-0.471} \qquad (2.24)$$

Assim $PGG(\hat{x})$ será o seguinte problema :

$$\begin{aligned}
\min \quad & x_1 \\
\text{suj.a} \quad & g_1(x) = 0.574\, x_1^{0.388} x_2^{-0.775} + 1.148\, x_1^{-0.612} x_2^{0.225} \leq 1 \\
& g_2(x) = 0.083\, x_1^{1.529} x_2^{-0.529} + 0.083\, x_1^{-0.471} x_2^{1.471} + \\
& \quad + 1.169\, x_1^{-0.471} x_2^{-0.529} \leq 1 \qquad (2.25) \\
& 1.0 \leq x_1 \leq 5.5 \\
& 1.0 \leq x_2 \leq 5.5
\end{aligned}$$

Observe que este é um problema de Programação Geométrica Posinomial, o qual pode ser condensado conforme a secção 2.1.

Condensamos as restrições (2.25) no ponto $\hat{x}$, e obtemos:

$$\hat{g}_1(x) = 1.720\, x_1^{-0.29786} x_2^{-0.083} \leq 1$$

$$\hat{g}_2(x) = 0.516\, x_1^{0.166}\, x_2^{0.277} \leq 1 \qquad (2.26)$$

usando a transformação dada em (2.14), alcançamos o programa linear :

$PL(x_0)$:

$$\begin{aligned} \text{min} \quad & z_1 \\ \text{suj.a} \quad & h_1(z) = 0.305\, z_1 - 0.082\, z_2 \leq -0.542 \\ & h_2(z) = 0.166\, z_1 + 0.277\, z_2 \leq 0.661 \\ & 0 \leq z_j \leq \ln 5.5 \qquad j = 1,2 \end{aligned} \tag{2.27}$$

A solução para o programa linear $PL(x_0)$ é $z_1^1 = 1.350$; $z_2^1 = 1.579$ e assim de (2.14) temos que $x_1^1 = 3.858$ e $x_2^1 = 4.852$.

Observe que esta solução não satisfaz a segunda restrição do $PGG(\hat{x})$:

$$g_2(x^1) > 1. + \varepsilon$$

Se a solução ótima do PL (2.27) for factível ao (2.25) então esta também será ótima para o problema (2.25), conforme seção 2.1. Caso contrário (no ex.2.1, vemos que o ponto x^1 violou a restrição $g_2(x) \leq 1$), a solução obtida é usada para condensar as restrições violadas e então formar um novo PL, e assim sucessivamente. No capítulo III, formalizaremos um algoritmo que usa esta estratégia e estudaremos sua convergência.

Note que a solução do PL pode ser factível ao PGG e infactível ao $PGG(\hat{x})$ (Veja figura 2.3). Este fato é explorado em [2] para acelerar o algoritmo.

CAPÍTULO III

Métodos

3.1 - Introdução

Dembo [10] descreve, que desde o início, PG foi desacreditada pelos pesquisadores de programação matemática que, a consideravam como uma curiosidade altamente especializada (um caso particular) e também como uma área sem interesse de pesquisa. Uma das razões para isso foi o desenvolvimento histórico dos procedimentos computacionais e a teoria da dualidade em PG que, na maior parte, não seguia o estado da arte aceito em PNL. A outra foi o desconhecimento, por parte dos programadores matemáticos, sobre a enorme aplicabilidade da PG em problemas da engenharia. No entanto, devido a essa aplicabilidade, a comunidade de engenharia aceitou-a entusiasticamente.

Foram então, os engenheiros que mais trabalharam no desenvolvimento dos softwares de PG o que, num certo grau, piorou a imagem nos círculos de programação matemática, por negligenciarem o desenvolvimento dos softwares da PNL. Como exemplo disto, Dembo comenta que os programas para o problema dual (que é de programação não-linear, com restrições lineares) não possuem qualquer implementação das técnicas mais recentes, numericamente estáveis, como as discutidas em [16]. Inclusive, a decomposição de matrizes é completamente desconhecida nos meios de P.G.

Por outro lado os pesquisadores de PG não se influenciaram com o folclore da programação matemática e fizeram implementações e testes do algoritmo de cortes de Kelley para

programas convexos.

O método de Kelley era visto como, na melhor das hipóteses, geometricamente convergente, numéricamente instável e não era recomendado para problemas convexos. Mesmo existindo argumentação teórica contrária definitiva, não há evidências computacionais para mostrar que o algoritmo de Kelley tem de fato, um desempenho pior ou melhor do que os outros algoritmos para programação convexa.

Rijckaert e Martens [22] confirmavam que um dos pacotes mais eficientes (em termos de tempo de CPU) e robustos atualmente disponível para PG é uma simples aplicação do algoritmo de Kelley. Esse algoritmo tem obtido excelentes resultados, e já existem, pelo menos, quatro pacotes conhecidos e publicados [10,12].

3.2 - Métodos de Cortes

Vários métodos foram propostos para resolver os problemas de programação geométrica. Beck e Ecker [6] sugerem um método com base na solução numérica da programação geométrica dual. Dinkel e outros [12], comentam que recentemente alguns pesquisadores de PG desenvolveram métodos com base na resolução da PG primal.

Geralmente a estratégia dos métodos de cortes consistem em determinar um conjunto que envolva a região de factibilidade do problema original, que seja mais fácil de se trabalhar. A partir desse conjunto, podemos obter uma solução que forneça um limite inferior (problema de minimização) à função objetivo do problema original (já que o elenco de soluções possíveis agora será maior).

Se a solução do problema for factível ao problema

original, então ela será ótima. Caso contrário, devemos determinar um novo conjunto envolvente contido no anterior, que exclua a solução anterior. Repete-se o processo, sempre eliminando as soluções obtidas anteriormente, até encontrarmos uma solução ótima.

Os métodos mais conhecidos utilizam poliedros, determinados por restrições lineares para os conjuntos envolventes e são denominados métodos de plano de cortes.

Esses métodos são aplicados a problemas do tipo:

$$\min \; c^T x \qquad (3.1)$$
$$x \in S$$

onde $S = \{x \in \mathbb{R}^N / g_k(x) \leq 0,\ k = 1,2,\ldots,K,\ g_k$ são funções convexas $\}$.

Podemos descrever um algoritmo geral, para esses métodos, como:

Passo 0 : Inicialização:

Determine um poliedro envolvente Po, faça ℓ=0.

Passo 1 :

Seja x^ℓ a solução do Programa Linear (PL)

$$\min \; c^T x \qquad (3.2)$$
$$x \in P_\ell$$

Se $x^\ell \in S$ então x^ℓ resolve (3.1), e é a solução ótima. Fim. Caso contrário, vá para o passo 2.

Passo 2 :

Determine a_ℓ, b_ℓ tal que :

i) $a_\ell^T x \leq b_\ell \quad \forall x \in S$.

ii) $a_\ell^T x^\ell > b_\ell$

onde ($a_\ell \in \mathbb{R}^N$, $b_\ell \in \mathbb{R}$).

Faça,

$$P_{\ell+1} = P_\ell \cap \{ x / a_\ell^T x \leq b_\ell \},$$

$$\ell \longleftarrow \ell + 1$$

Retorne ao passo 1.

Os algoritmos diferem basicamente nos cortes a serem usados (a_ℓ, b_ℓ), e quanto mais profundo for o corte, melhor será a convergência do método [18].

O algoritmo de planos de cortes de Kelley [12], utiliza no passo 2, o hiperplano :

$$g_i(x^\ell) + \nabla g_i(x^\ell)(x - x^\ell) \leq 0 \qquad (3.3)$$

onde i representa o índice de uma restrição violada em x^ℓ. Podemos ter várias estratégias de inclusão da restrição (3.3), por exemplo :

a) Apenas a restrição mais violada;

b) Todas restrições violadas.

(naturalmente vários cortes podem ser incluídos).

Uma outra idéia para conseguir cortes "mais profundos" é tentar determinar um hiperplano suporte [18], pagando o preço de uma busca unidimensional.

Dividimos o passo 2 em duas etapas :

a) Determine $\hat{x}^\ell$ na fronteira de S como uma combinação convexa entre x^ℓ (obtido no passo 1) e um ponto factível (conhecido a priori). (Isto corresponde a uma busca

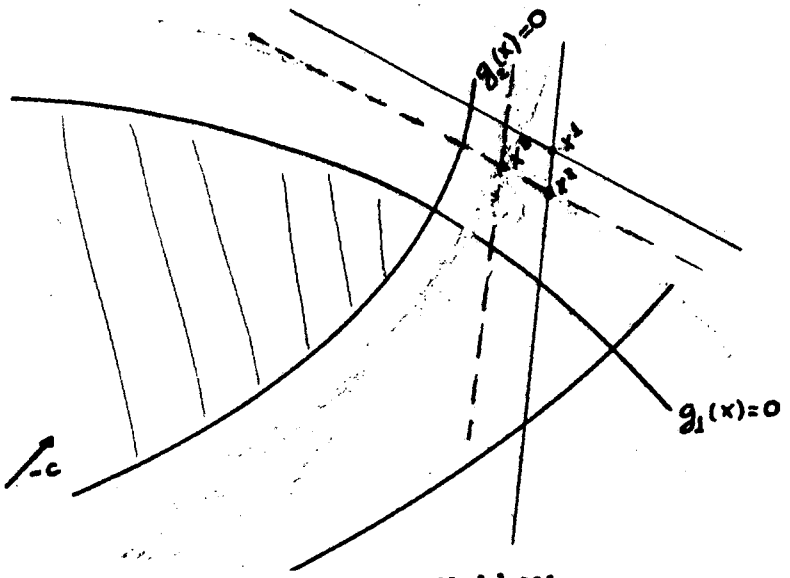
unidimensional).

b) Defina o novo corte por : (3.4)

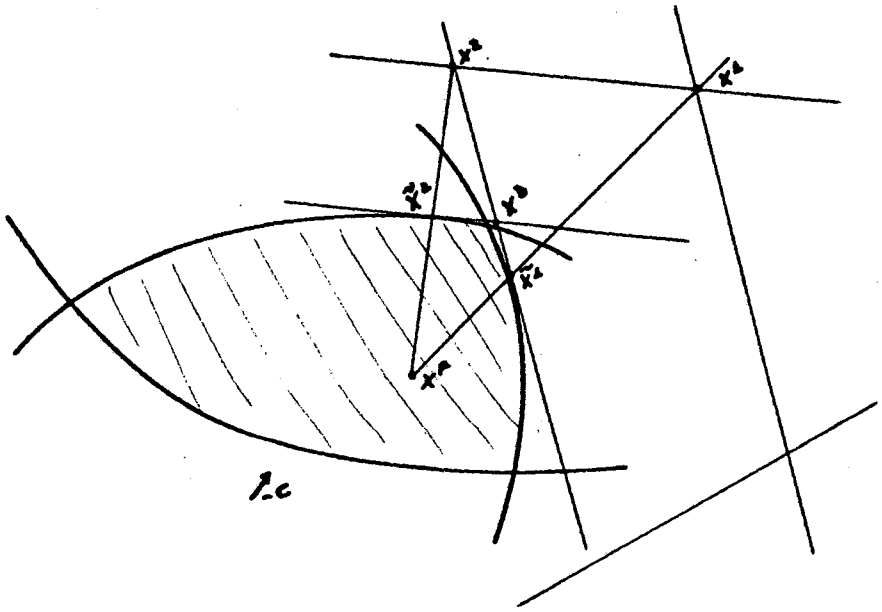
$$g_i(\hat{x}^\ell) + \nabla g_i(\hat{x}^\ell)(x - \hat{x}^\ell) \leq 0$$

onde i é um índice tal que $g_i(\hat{x}^\ell) = 0$.

Podemos ilustrar graficamente estes métodos :



(a) Cortes de Kelley



(b) Hiperplano Suporte.

Fig. 3.1

A convergência destes métodos é estabelecida pelo seguinte teorema [18] :

Teorema 3.1 :

Suponha que $g_k(x)$ $k = 1,2,..,K$ sejam funções convexas, continuamente diferenciáveis. Então qualquer ponto limite da sequência $\{x^\ell\}$ das soluções geradas pelos algoritmos de plano de cortes é solução de (3.1).

3.3 - Um Método de Cortes para Programação Geométrica Posinomial

Nesta seção adotaremos a seguinte forma para um PGP :

PGP :

$$\begin{aligned} &\min && x_o \\ &\text{suj. a} && g_k(x) \le 1 \qquad k = 0,1,..,K \\ & && 0 < \underline{x}_j \le x_j \le \bar{x}_j \quad j = 0,1,..,N \end{aligned} \tag{3.5}$$

onde $g_k(x)$ são posinômios. O conjunto de soluções factíveis denotamos por $\mathbb{S}$.

A construção do conjunto envolvente e dos cortes é baseada na condensação de posinômios (Cap. II - 2.2) :

$$g_k(x) = \sum_{i=1}^{Tk} \theta_{ki} \prod_{j=1}^{N} x_j^{\varphi_{kij}} \ge \hat{g}_k(x) = \hat{c}_k \prod_{j=1}^{N} x_j^{\hat{a}_{kj}} \tag{3.6}$$

onde $\hat{c}_k$ e $\hat{a}_{kj}$ são dados por (2.3) e (2.4) respectivamente.

Assim, podemos escolher $\hat{x} > 0$ qualquer, e construirmos P^o :

$$P^o = \{ x / \hat{g}_k(x) \le 1,\ k = 0,1,..,K \ \text{ e } \ \underline{x}_j \le x_j \le \bar{x}_j \quad j=0,1,..,N \}.$$

onde $\hat{g}_k(x)$ é a condensação de $g_k(x)$ em $\hat{x}$. Das propriedades estudadas no capítulo anterior, segue que $P^o \supseteq \mathbb{S}$.

Embora o problema condensado em $\hat{x}$:

PC(0) :

$$\min \quad x_0$$
$$x \in P^0$$

seja não-linear, podemos transformá-lo num problema linear, mediante a seguinte mudança de variáveis :

$$z_j = \ln x_j \tag{3.7}$$

como já foi visto em (2.14), PC(0) é então equivalente ao problema de programação linear :

PL(0) :

$$\begin{aligned} \min \quad & z_0 \\ \text{suj. a} \quad & \sum_{j=0}^{N} \hat{a}_{kj} z_j \leq -\ln \hat{c}_k \qquad k = 0,1,..,K \qquad (3.8) \\ & \underline{z}_j \leq z_j \leq \bar{z}_j \qquad j = 0,1,..,N \end{aligned}$$

onde $\hat{c}_k$ e $\hat{a}_{kj}$ são obtidos por (2.3) e (2.4) e

$$\underline{z}_j = \ln \underline{x}_j \,, \quad \bar{z}_j = \ln \bar{x}_j$$

o qual pode ser resolvido eficientemente por métodos de programação linear.

Seja x^0 uma solução de PC(0) (o que pode ser obtido fazendo $x_j^0 = e^{z_j^0}$, $j = 0,1,..,N$, onde z^0 resolve PL(0)).

Se $x^0 \in S$, então x^0 resolve (3.5), uma vez que x^0 minimiza x_0 sobre um conjunto mais amplo de soluções. Caso $x^0 \notin S$, digamos

$$g_r(x^0) > 1 + \varepsilon \qquad (3.9)$$

com ε positivo e pequeno. Acrescentamos à PC(0) a seguinte restrição :

$$g_r^0(x) \leq 1 \qquad (3.10)$$

onde g_r^0 é a condensação de g_r em x^0. O índice r em (3.9) é geralmente tomado por :

$$g_r(x^0) = \max_k \{ g_k(x^0) > 1; \quad k = 0,1,..,K \} \qquad (3.11)$$

ou seja, r é a restrição mais violada. Chamamos de PC(1) ao novo problema obtido pela inclusão da restrição (3.10).

De (2.8) segue que :

$$g_k^0(x^0) = g_k(x^0) > 1$$

conforme foi mostrado no capítulo anterior. Assim, x^0 não satisfaz (3.10) e portanto é infactível para PC(1). Isto é, a restrição (3.10) "corta" x^0 do novo conjunto envolvente.

Agora, resolvemos PC(1), obtemos x^1, e repetimos o mesmo procedimento percorrido com x^0. Geramos assim, a sequência $x^0, x^1, x^2, ..$, cuja convergência veremos com maior detalhe após a seção seguinte.

Alguns elementos sobre a resolução de PL(i) :

Escrevemos PL(i) na forma matricial :

$$\begin{aligned} \min \quad & c^T z \\ \text{suj. a} \quad & A^{(i)^T} z \leq b^{(i)} \\ & \underline{z} \leq z \leq \bar{z} \end{aligned} \tag{3.12}$$

com $A^{(i)^T} \in R^{MxN}$. A linha ℓ de $A^{(i)^T}$ é denotada por $a_\ell^{(i)^T}$.

O problema dual de (3.12) pode ser escrito na forma :

$$\begin{aligned} \max \quad & b^{(i)^T} u + \sum_{i=o}^{N} f_i(v_i) \\ \text{suj. a} \quad & A^{(i)} u + v = c \\ & u \leq 0 \end{aligned} \tag{3.13}$$

onde

$$f_i(v_i) = \begin{cases} \bar{z}_i v_i & \text{se } v_i \leq 0 \\ \underline{z}_i v_i & \text{se } v_i \geq 0 \end{cases}$$

A função objetivo de (3.13) a ser maximizada é côncava linear por partes e o ponto crítico de f_i (onde muda a inclinação) é zero. Um método do tipo primal-simplex linear por partes, pode ser aplicado a (3.13) sem maiores dificuldades (Melhores detalhes sobre o algoritmo quando existem limites superiores e inferiores, inclusive nas restrições, podem ser obtidos em [1,4]).

O problema (3.13) é resolvido e o vetor multiplicador $z^{(i)}$ (variáveis duais de (3.13)) é a solução de (3.12).

Com relação a solução inicial de PL(i) notamos o seguinte :

a) Para $i = 0$, fazemos $u = 0$ (variáveis não-básicas), $v = c$ (variáveis básicas), e o vetor multiplicador (que é a

solução primal infactível até a otimalidade de (3.13) ser obtida), associado à base $B = I$ é :

$$z_i = \bar{z}_i \quad \text{se} \quad v_i = c_i < 0$$
$$z_i = \underline{z}_i \quad \text{se} \quad v_i = c_i \geq 0$$

b) Para $i > 0$, a solução dual de PL(i-1) é factível para (3.13) (com $u_M = 0$), pois $a_j^{(i)} = a_j^{(i-1)}$ $j = 0,1,..,M-1$, porém não é ótima, uma vez que :

$$a_M^{(i)^T} \; z^{(i-1)} > b_M^{(i)} \qquad (3.14)$$

onde $a_M^{(i)}$ é a M-ésima coluna de $A^{(i)}$, que foi incluida após a resolução de PL(i-1), $z^{(i-1)}$ é a solução de PL(i-1), e $b_M^{(i)}$ o termo independente da restrição adicionada.

A desigualdade em (3.14) vem do fato de que $z^{(i-1)}$ não satisfaz a restrição acrescentada à PL(i-1). Assim sendo, o custo relativo, associado à base ótima do PL(i-1), que fornece $z^{(i-1)}$ é negativo. Como $u_M \leq 0$, fazemos então u_M negativo (entrar na base), aumentando assim, a função objetivo dual.

Desta forma o problema da fase I (que ocupa boa parte das iterações na resolução de um programa linear) é sempre eliminado, e a solução do problema PL(i-1) pode ser facilmente aproveitada para PL(i), sendo um bom ponto de partida.

.Critérios de Parada :

À medida em que novas restrições (cortes) são introduzidas, temos uma representação cada vez melhor do problema

original, pois soluções antes infactíveis à esse problema são então eliminadas (em especial a solução ótima do problema condensado anterior). Com isto a sequência gerada tende à factibilidade devido ao teorema 3.1 e a relação entre condensação e plano de corte de Kelley (veja secão 3.4). No instante em que tivermos uma solução aceitável interrompemos o processo; ou seja, para um ε positivo e pequeno, escolhido a priori, paramos quando

$$g_k(x^{\ell}) \leq 1 + \varepsilon \qquad k = 0,1,..,K$$

e adotamos x^{ℓ} como a solução de PC(ℓ).

Algoritmo :

Aqui não estaremos considerando a escolha da restrição violada a ser condensada. Deixaremos esses detalhes para o estudo computacional (Capítulo V).

Passo 0 : Inicialização :

Escolha um ponto inicial $x^0 > 0$ e condense todas restrições $g_k(x)$ no ponto x^0, para $k = 0,1,..,K$ conforme (3.6). Faça $\ell = 0$ (aqui forneceremos o valor para ε).

Passo 1 : Solução do Problema Condensado :

Resolva :

PC(ℓ) :

$$\min \quad x_0$$
$$x \in P^{\ell}$$

onde $P^{\ell} = \{x \,/\, g_k^{\ell}(x) \leq 1,\ k = 0,1,..,K \ \text{ e } \ \underline{x} \leq x \leq \bar{x}\}$

(Para resolver PC(ℓ) faça a transformação de PC(ℓ) para PL(ℓ) utilizando a mudança de variáveis dada em (3.7),

obtendo assim (3.8). O método dual-simplex é recomendado).

Seja $x^{\ell+1}$ uma solução de PC(ℓ).

Passo 2 : Teste de Otimalidade :

Se $\{ g_k(x^{\ell+1}) \leq 1 + \varepsilon \qquad k = 0,1,..,K \}$

então

$x^{\ell+1}$ é ótimo para PGP (3.5). Fim.

senão

escolha uma restrição $r \nearrow g_r(x^{\ell+1}) > 1 + \varepsilon$.

Condense esta restrição em $x^{\ell+1}$ e obtenha o monômio condensado $g_r^{\ell+1}(x) \leq 1$.

Redefina $P^{\ell+1} = P^{\ell} \cap \{g^{\ell+1}(x) \leq 1\}$.

Faça $\ell \longleftarrow \ell+1$ e repita o passo 1.

3.4 - Relação entre Condensação e Plano de Corte

O algoritmo de Kelley lineariza um problema convexo não-linear, tomando-se o plano tangente (desenvolvimento de Taylor 1ª ordem) das restrições (a função objetivo é considerada linear); enquanto que os algoritmos de condensação lineariza um PGP, tomando-se o logaritmo do monômio condensado (desigualdade aritmética-geométrica). Mostraremos, a seguir, que as estratégias de linearizar usando o plano tangente e a condensação são equivalentes.

1) Inicialmente expressaremos a restrição linear dada pelo corte de Kelley.

Considere uma restrição $f(x) \leq 0$.

O corte de Kelley é definido por :

$$f(\hat{x}) + \nabla f(\hat{x}) (x - \hat{x}) \leq 0 \qquad (3.15)$$

Considerando uma restrição violada do problema principal com a mudança de variável $x_j = e^{z_j}$, temos :

$$g(x) = h(z) = \sum_{i=1}^{T} \theta_i \, e^{\sum_{j=0}^{N} \varphi_{ij} z_j} = \sum_{i=1}^{T} v_i(z) \leq 1 \qquad (3.16)$$

onde $v_i(z) = \theta_i e^{\sum_{j=0}^{N} \varphi_{ij} z_j}$, ou equivalentemente,

$$\ln [h(z)] \leq 0 \qquad (3.17)$$

Aplicando (3.15) em (3.17) segue :

$$\ln[h(\hat{z})] + \nabla\{\ln[h(\hat{z})]\}(z - \hat{z}) \leq 0 \qquad (3.18)$$

onde :

$$\frac{\partial h(z)}{\partial z_k} = \sum_{i=1}^{T} \theta_i \varphi_{ik} e^{\sum_{j=0}^{N} \varphi_{ij} z_j} = \sum_{i=1}^{T} v_i(z) \varphi_{ik}$$

Assim,

$$\nabla h(z) = (v_1 \ldots v_T) A$$

onde

$$A = \begin{bmatrix} \varphi_{10} & \varphi_{11} & \cdots & \varphi_{1N} \\ \varphi_{20} & \varphi_{21} & & \varphi_{2N} \\ \cdot & \cdot & \cdot & \cdot \\ \varphi_{T0} & \varphi_{T1} & \cdots & \varphi_{TN} \end{bmatrix}$$

agora,

$$\frac{\partial}{\partial z_k}\{\ln[h(z)]\} = \frac{1}{h(z)} \quad \frac{\partial h}{\partial z_k}$$

portanto

$$\nabla\{\ln[h(z)]\} = \left(\frac{v_1}{h(z)}, \frac{v_2}{h(z)}, \ldots, \frac{v_T}{h(z)}\right) A = \delta A \qquad (3.19)$$

onde

$$\delta = (\delta_1 \ldots \delta_T) \quad e \quad \delta_i = v_i / h(z) \qquad (3.20)$$

Substituindo (3.19) em (3.18) temos então, a seguinte relação :

$$\{\ln[h(\hat{z})] - \delta A \hat{z}\} + \delta A z \leq 0 \qquad (3.21)$$

2) A restrição monomial obtida pela condensação de $g(x)$ em um ponto $\hat{x}$, é definido por :

$$\hat{g}(x) = \hat{c} \prod_j x_j^{\hat{a}_j} = \prod_i \left(\frac{\theta_i}{\delta_i}\right)^{\delta_i} \prod_j x_j^{\sum_i \delta_i \varphi_{ij}} \leq 1 \qquad (3.22)$$

Tomando a função logaritmo em (3.22) temos :

$$\ln[\hat{g}(x)] = \sum_i \delta_i \ln\left(\frac{\theta_i}{\delta}\right) + \sum_j \ln x_j^{\sum_i \delta_i \varphi_{ij}} =$$

$$= \sum_i \delta_i \ln\left(\frac{\theta_i}{\delta_i}\right) + \sum_i \sum_j \delta_i \varphi_{ij} \ln x_j \leq 0 \qquad (3.23)$$

Da equação (2.5), é dado que $u_i(\hat{x}) = v_i(\hat{z})$ e $g(\hat{x}) = h(\hat{z})$, temos que $\delta_i = v_i(\hat{z})/h(\hat{z})$; e usando $z_j = \ln x_j$, segue

$$\ln[\hat{g}(x)] = \sum_i \delta_i \ln \left[\frac{\theta_i}{\frac{v_i(\hat{z})}{h(\hat{z})}} \right] + \sum_i \sum_j \delta_i \varphi_{ij} z_j =$$

$$= \sum_i \delta_i \ln \left[\frac{\theta_i \, h(\hat{z})}{\theta_i e^{\sum_j \varphi_{ij} \hat{z}_j}} \right] + \delta A z =$$

$$= \left\{ \sum_i \delta_i \ln [h(\hat{z}) + \sum_i \delta_i \ln e^{-\sum_j \varphi_{ij} \hat{z}_j} \right\} + \delta A z =$$

$$= \{ \ln [h(\hat{z})] - \delta A \hat{z} \} + \delta A z \leq 0 \qquad (3.24)$$

Notamos que a restrição (3.24) gerada pela condensação é a mesma que (3.21) gerada pelo corte de Kelley aplicado a (3.17). Com isto a convergência do algoritmo de condensação é garantida pelo teorema 3.1.

3.5 - Método de Cortes para Programação Geométrica Generalizada.

Nesta seção queremos apenas ressaltar o uso da condensação para transformar o PGG em um PGP, e a partir dessa transformação todo o processo visto na seção anterior será repetido.

Considere um problema de Programação Geométrica Generalizada :

PGG :

$$\begin{aligned} \min \quad & x_o \\ \text{suj.a} \quad & \frac{p_k(x)}{1 + q_k(x)} \leq 1 \qquad k = 0,1,..,K \qquad (3.25) \\ & 0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned}$$

Conforme já foi visto no capítulo anterior, a condensação do PGG (que consiste em condensar $1 + q_k(x)$) em um ponto x^0 factível, escolhido inicialmente, faz com que a região de factibilidade do PGG(x^ℓ) (2.17), esteja contida na região de factibilidade do PGG.

O programa posinomial obtido (PGG(x^0)) pode então ser resolvido pelo método descrito na seção anterior.

Obtemos assim, um ponto factível ao PGG que novamente será usado para condensar (3.25), e assim por diante. Segundo [2,10] a sequência gerada desta forma converge para uma solução que satisfaz as condições necessárias de Kuhn-Tucker. O critério de parada a ser usado agora é :

$$\| x^{\ell+1} - x^{\ell} \| < \varepsilon \qquad (3.26)$$

onde x^ℓ é a solução obtida na iteração ℓ, e ε é um número positivo e pequeno.

Agora, ao contrário de PGP onde garantimos o ótimo global, a não-convexidade de PGG pode levar a ótimos locais, característica de problemas não-convexos.

Algoritmo :

(fase 1)

Passo 0 : Inicialização :

Determine um ponto inicial x^0 factível ao PGG .

PGG :

$$\min \quad x_o$$

$$\text{suj.a} \quad \frac{p_k(x)}{1 + q_k(x)} \leq 1 \qquad k = 0,1,..,K$$

$$0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N$$

faça $\ell = 0$

(fase 2)

Passo 1 : Condensação do PGG :

Condense $1 + q_k(x)$ $k=0,1,..,K$ em x^{ℓ} (conforme a secção (2.2)) e obtenha $PGG(x^{\ell})$ (veja (2.17)).

Passo 2 : Resolução do $PGG(x^{\ell})$:

Resolva o programa posinomial $PGG(x^{\ell})$, obtido no passo anterior, utilizando o algoritmo dado na seção 3.3.

Seja $x^{\ell+1}$ a solução ótima de $PGG(x^{\ell})$.

Passo 3 : Teste de otimalidade :

Se $\| x^{\ell+1} - x^{\ell} \| \leq \varepsilon$

então

$x^{\ell+1}$ é uma solução de Kuhn-Tucker. Fim.

senão

Faça $\ell \longleftarrow \ell + 1$ e repita o passo 1.

A solução de $PGG(x^{\ell})$ é factível para PGG (não necessariamente ótima). Em [2], é sugerido uma modificação no passo 2, com o objetivo de acelerar a convergência do método.

Uma solução obtida na resolução de $PGG(x^{\ell})$ pode ser

infactível a este (quando for factível, será ótimo de PGG(x^{ℓ})), porém pode ser factível a PGG. Esta solução é então usada como $x^{\ell+1}$.

Fase 1 :

Esta fase pode ser usada (caso não haja disponível um ponto factível) com o intuito de encontrarmos um ponto inicial factível para ser usado na fase 2 (que é o algoritmo de resolução do problema PGG).

Dado um problema PGG, (2.16), a fase 1 consiste em resolver o seguinte PGG.

PGG(w) :

$$\begin{aligned} \min \quad & \prod_{k=0}^{K} w_k \\ \text{suj. a} \quad & \frac{p_k(x)}{1 + q_k(x)} \leq w_k \qquad k = 0,1,..,K \qquad (3.27) \\ & w_k \geq 1 \qquad k = 0,1,..,K \\ & 0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned}$$

Se (w^*, x^*) resolve PGG(w) e $\prod_{k=0}^{K} w_k^* = 1$, então x^* é um ponto factível para PGG e pode ser usado na fase 2.

Algoritmo :

Passo 1 :

Seja x^0 qualquer ponto que satisfaça :

$$\underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N$$

Defina

$$w_k^o = \text{Max} \left\{ \frac{p_k(x^o)}{1 + q_k(x^o)} , 1 \right\} \quad k = 0,1,..,K$$

o ponto (w^o, x^o) onde $w^o = (w_o^o, w_1^o, \ldots, w_K^o)$ é assim uma solução factível para o PGG(w).

Passo 2 :

a) Se $w_k^o = 1$ para todos k, então x^o é factível ao PGG (2.16). Fim da fase 1.

b) Se existir algum k, tal que, $w_k^o > 1$, então resolvemos o PGG(w) usando a fase 2 uma única vez; com um ponto inicial (w^o, x^o).

Passo 3 :

Examine a solução ótima (w^*, x^*) de PGG(w) :

a) Se $w_k^* = 1$ para todo $k = 1,2,..,K$ então x^* é um ponto factível para PGG (2.16). Fim da fase 1.

b) Caso exista algum k , tal que $w_k^* > 1$, então a fase 1 falhou em encontrar uma solução factível.

Esta sugestão para fase 1 é dada em [2]. Uma outra possibilidade seria resolver o seguinte probrema :

$$\begin{aligned} &\min \quad w \\ &\text{suj. a} \quad \frac{p_k(x)}{1 + q_k(x)} \leq w \quad k = 0,1,..,K \qquad (3.28) \\ &\qquad w \geq 1 \\ &\qquad 0 < \underline{x}_j \leq x_j \leq \bar{x}_j \qquad j = 0,1,..,N \end{aligned}$$

As conclusões obtidas com a solução de (3.27) podem ser aplicadas a (3.28). Há uma aparente vantagem de (3.28) sobre (3.27), pois este utiliza apenas uma variável artificial, enquanto que (3.27) utiliza K + 1 variáveis artificiais. Além

disso, a função objetivo já é linear.

Exemplo 3.1 -

Retomemos o mesmo exemplo visto nos capítulos anteriores, ilustrando agora os algoritmos e resultados obtidos.

$$\begin{aligned} \min \quad & x_1 \\ \text{suj. a} \quad & \frac{x_1/4 + x_2/2}{1 + x_1^2/16 + x_2^2/16} \leq 1 \qquad (3.29) \\ & \frac{x_1^2/(6x_2) + x_2/6 + 7/(3x_2)}{1 + x_1/x_2} \leq 1 \\ & 1.0 \leq x_1 \leq 5.5 \\ & 1.0 \leq x_2 \leq 5.5 \end{aligned}$$

Note que o algoritmo de PGG tem 2 fases, onde a primeira fase (Fase 1) tem como objetivo encontrar um ponto de partida.

Consideramos, entretanto, o ponto de partida dado por :

$$x^0 = (4.0, 4.5)^T \quad \text{(factível)}$$

No exemplo 2.1, fizemos os primeiros passos de cada algoritmo.

O problema (2.20) (PGG) foi condensado obtendo-se (2.25)(PGP). Para resolver este último, novamente condensamos (agora poderia ser qualquer $x>0$) as restrições em $(4.0, 4.5)^T$ e obtivemos (2.27)(PL), cuja solução $z^T = (1.35, 1.58)$. Com a mudança de variável $x_j = e^{z_j}$, tivemos a primeira aproximação no PGP : $x^T = (3.858, 4.852)$.

Este ponto é infactível ao $PGG(x^0)$ e portanto gerará na nova restrição ao PL, que fornecerá uma nova solução, e assim por diante até determinarmos a solução ótima de $PGG(x^0)$, que denotamos por x^1. Em geral denotaremos por $x^{\ell+1}$ a solução de $PGG(x^{\ell})$.

Faremos a seguir duas tabelas mostrando a convergência deste problema. As tabelas são formadas por 5 colunas que serão :

a) Localização, da solução atual, na figura 3.2.

b) Número da iteração do PGG (fase 2), ITG.

c) Número de cortes (número de iterações no $PGG(x^{\ell})$.

d) Solução do $PGG(x^{\ell})$ (ponto aproximado).

e) Comentários.

Local. fig.3.2	fase 2 ITG	Numeros de cortes	Solução do $PGG(x^{\ell})$	Comentários
1	0	---	(4.0, 4.5)	Ponto inicial fact.
2	1	1	(3.872, 4.786)	
3	2	0	(3.824, 4.824)	
3	3	0	(3.823, 4.823)	Ponto ótimo local(A)

Tab. 3.1 - Convergência para um mínimo local, com o ponto inicial adotado, $x^0 = (4.0, 4.5)$.

Pudemos observar que a solução na iteração 2 de $PGG(x^0)$ já é factível a PGG. Em [2], Avriel, Dembo e Passy sugerem que o primeiro ponto factível a PGG encontrado ao se resolver $PGG(x^{\ell})$ seja usado como $x^{\ell+1}$, com o intuito de acelerar a sua convergência.

Note que o ponto obtido na iteração 2 e 3 do PGG são praticamente os mesmos (pois este é o critério de parada). A

convergência se deu após 3 iterações no PGG a qual é demonstrada na tabela 3.1 onde denominamos por ITG (iteração do PGG).

Outro uso da Fase 1 -

Tentaremos agora encontrar um outro ponto inicial, utilizando a fase 1, com o objetivo de melhorar a solução ótima anterior, eliminando tal solução ótima, supostamente local, ou região de factibilidade. Para isto será usada a restrição $x_1 \leq 3.5$, visto que desejamos minimizar x_1.

Se a fase 1 for concluida com melhor sucesso, o ponto obtido é factível, portanto na fase 2 esse ponto será usado como ponto inicial.

De (3.27) e (3.29) temos :

$$\begin{aligned} &\min && w_1 \; w_2 \\ &\text{suj. a} && \frac{x_1/4 + x_2/2}{1 + x_1^2/16 + x_2^2/16} \leq w_1 \\ & && \frac{x_1^2/(6x_2) + x_2/6 + 7/(3x_2)}{1 + x_1/x_2} \leq w_2 \\ & && 1.0 \leq x_1 \leq 3.5 \\ & && 1.0 \leq x_2 \leq 5.5 \\ & && 1.0 \leq w_1 \leq 2.5 \\ & && 1.0 \leq w_2 \leq 2.5 \end{aligned} \qquad (3.30)$$

Iniciando a fase 1 com o ponto $x^0 = (3.5, 4.823)$ o qual viola a restrição $p_1(x) - q_1(x) \leq 1$. Note que o ponto inicial será portanto : $x_1 = 3.5$, $x_2 = 4.823$, $w_1 = 2.5$, $w_2 = 1.0$ que satisfaz agora as restrições (3.30) (o ponto é representado por 4

na figura 3.2). A fase 1 tem como solução, dada em [2], o ponto factível $x = (2.33, 1.38)$ (ponto 5 na fig. 3.2).

O ponto de mínimo global $x^* = (1.178, 2.178)$, usamos o algoritmo fase 2, e a solução inicial obtida na sub-seção 'outro uso da fase 1', é mostrada na tabela 3.2.

Local. fig. 3.2	Fase 2 ITG	Numeros de cortes	Soluções do PGG(x^l)	Comentários :
5	0	——	(2.330, 1.384)	Factível.
6	1	1	(1.574, 1.961)	
7	2	2	(1.261, 2.125)	
8	3	0	(1.178, 2.176)	
8	4	0	(1.178, 2.178)	Ótimo Global

Tab. 3.2 - Convergência para o mínimo global.

É bom lembrar, que a convergência para uma solução ótima global do PGG, depende muito da escolha do ponto inicial, por isso devemos fazer algumas tentativas de melhoria do ponto ótimo (quando não sabemos se a solução obtida é a ótima global), utilisando outros pontos iniciais (com o uso da fase 1).

Na figura 3.2, podemos ver o caminho dos pontos obtidos pelos algoritmos, dado em [2].

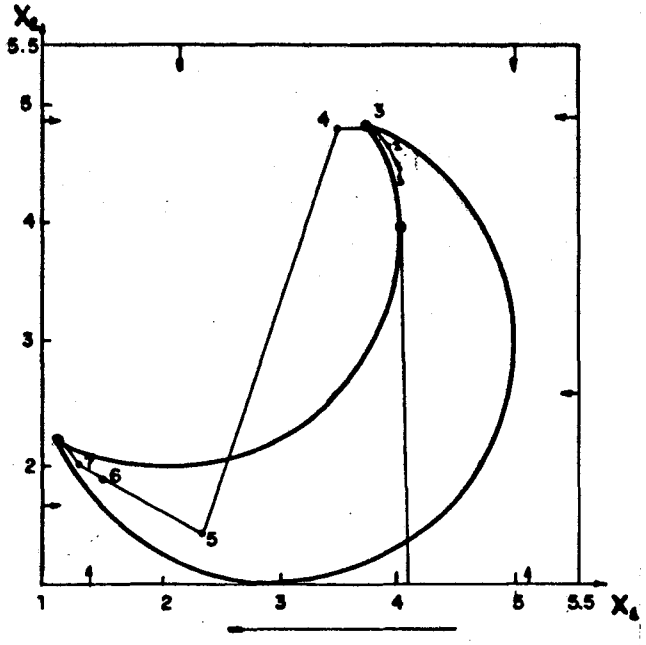


Fig. 3.2

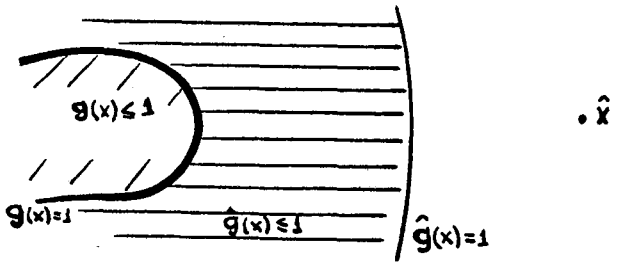
CAPÍTULO IV

MELHORIA DA SOLUÇÃO ANTES DA CONDENSAÇÃO

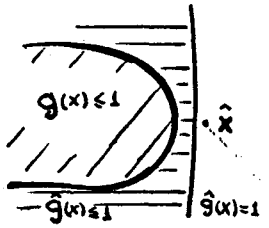
4.1 - Introdução -

A condensação, estudada no capítulo II, é uma ferramenta muito importante para os problemas posinomiais, visto que ela transforma o PGP em um programa monomial aproximado, mais simples de se trabalhar.

Suas propriedades, mostram que a restrição monomial $(\hat{g}(x) \leq 1$, onde $\hat{g}(x)$ é a condensação da função posinomial $g(x)$ no ponto $\hat{x}$), representa melhor a restrição original $g(x) \leq 1$ se $g(\hat{x}) \cong 1$ (pelo menos nas vizinhanças de $\hat{x}$), como indicam as figuras 4.1 (a) e (b) :



(a) Condensação em um ponto mais distante a $g(x)=1$.



(b) Condensação em um ponto próximo a $g(x)=1$.

fig. 4.1

Neste sentido, procuramos investir em uma aproximação do

ponto $\hat{x}$ à região de factibilidade, antes de executar a condensação, na tentativa de obtermos uma região melhor representada e como consequência um processo de resolução mais acelerado.

A seguir, veremos uma descrição matemática dessa nossa tentativa.

4.2 - Melhorando a solução

Seja $g(x) \leq 1$ uma restrição posinomial, onde :

$$g(x) = \sum_{i=1}^{T} \theta_i \prod_{j=0}^{N} x_j^{\varphi_{ij}} \leq 1 \qquad (4.1)$$

e x^o uma solução infactível ao PGP (fornecida pelo programa monomial a cada iteração de métodos de cortes - seção 3, cap.III), tal que :

$$g(x^o) > 1$$

(por simplicidade de notação abandonamos aqui o índice da restrição violada).

Procuramos encontrar uma solução $x/g(x) = 1$, antes de procedermos à condensação. Restringimos a procura na curva parametrizada em γ :

$$x_j(\gamma) = x_j^o \, e^{-\gamma a_j^o g^o} \qquad j = 0,1,..,N \qquad (4.2)$$

onde a_j^o é o expoente de x_j no monômio $g^o(x)$, obtido na

condensação de $g(x)$ em x^0 (veja (2.4)) e g^0 é :

$$g^0 = g(x^0) = g^0(x^0) \tag{4.3}$$

conforme (2.8). Nosso objetivo é determinar γ tal que :

$$g(x(\gamma)) = 1 \tag{4.4}$$

A solução de (4.4) em geral não é trivial. Usamos a aproximação monomial de $g(x)$ em (4.4) :

$$g^0(x(\gamma)) = 1 \tag{4.5}$$

e definimos $x^1 = x(\gamma^0)$, onde γ^0 resolve (4.5). Sendo assim

$$g^0(x^1) = c^0 \prod_j (x_j^1)^{a_j^0} = 1 \tag{4.6}$$

Substituindo (4.2) em (4.6) temos que :

$$g^0(x^1) = c^0 \prod_j (x_j^0 \, e^{-\gamma a_j^0 g^0})^{a_j^0} =$$

$$= c^0 \prod_j (x_j^0)^{a_j^0} \; e^{-\gamma^0 g^0 \sum_j (a_j^0)^2}$$

logo

$$g^0(x^1) = g^0 \, e^{-\gamma^0 g^0 \|a^0\|^2} = 1 \tag{4.7}$$

Aplicando logaritmo natural em (4.7), obtemos :

$$\ln g^0 - \gamma^0 g^0 \| a^0 \|^2 = 0$$

logo

$$\gamma^o = \frac{\ln g^o}{g^o \|a^o\|^2} \qquad (4.8)$$

Substituindo (4.8) em (4.2), e fazendo $x_j^1 = x_j(\gamma^o)$ para $j = 0,1,..,N$, temos :

$$x_j^1 = x_j^o \, e^{\frac{-\, g^o \, a_j^o \ln g^o}{g^o \| a^o \|^2}} = x_j^o \, e^{\ln [g^o]^{\frac{-\, a_j^o}{\|a^o\|^2}}}$$

obtendo assim :

$$x_j^1 = x_j^o \, [g^o]^{\frac{-\, a_j^o}{\|a^o\|^2}} \qquad j = 0,1,..,N \qquad (4.9)$$

Podemos repetir o processo, usando agora x_j^1 no lugar de x_j^o e obter uma nova solução x_j^2 e assim por diante.

$$x^{k+1} = x^k \, [g^k]^{\frac{-\, a^k}{\|a^k\|^2}} \qquad j = 0,1,..,N \qquad (4.10)$$

onde $g^k = g(x^k)$ e a^k é dado por (2.4) no ponto x^k.

4.3 - Equivalência com Censor-Lent

Teorema 4.1 -

O método definido por (4.10) é equivalente ao método de Censor-Lent (apêndice A) aplicado a $\ln [h(z)]$, onde $h(z)=g(x)$ com $x_j = e^{z_j}$.

Prova :

Considere a mudança de variáveis :

$$x_j = e^{z_j},$$

então

$$h(z) = \sum_i \theta_i \prod_j e^{\varphi_{ij} z_j} = \sum_i \theta_i e^{\sum_j \varphi_{ij} z_j} \qquad (4.11)$$

Uma iteração de Censor-Lent com $\alpha = 1$ em $\ln [h(z)]$ é :

$$z^{k+1} = z^k - \frac{\ln [h(z^k)]}{\| \nabla \{\ln [h(z^k)]\} \|^2} \nabla \{ \ln [h(z^k)] \} \qquad (4.12)$$

Cálculo de $\nabla\{\ln[h(z^k)]\}$:

$$\frac{\partial}{\partial z_l} \{ \ln [h(z)] \} = \frac{1}{h(z)} \frac{\partial h(z)}{\partial z_l} \qquad (4.13)$$

e

$$\frac{\partial h(z)}{\partial z_l} = \sum_i \theta_i \varphi_{il} e^{\sum_j \varphi_{ij} z_j} = \sum_i \varphi_{il} \mu_i(x) \qquad (4.14)$$

onde

$$\mu_i(x) = \theta_i \prod_j x_j^{\varphi_{ij}} \qquad \text{com} \quad x_j = e^{z_j}$$

Substituindo (4.14) em (4.13), e aplicando em $z = z^k$ com $g(x^k) = h(z^k)$ temos :

$$\frac{\partial}{\partial z_l} \{ \ln [h(z)] \} \Big|_{z=z^k} = \sum_i \frac{\mu_i(x^k)}{g(x^k)} \varphi_{il} = \sum_i \delta_i \varphi_{il} = a_l^k$$

conforme (2.4).

Logo,

$$\nabla \{ \ln [h(z)] \} = a^k \qquad (4.15)$$

Substituindo (4.15) em (4.12) segue :

$$z_j^{k+1} = z_j^k - \frac{\ln (h(z^k))}{\| a^k \|^2} a_j^k \quad j = 0,1,..,N \qquad (4.16)$$

Aplicando a função exponencial em (4.16) temos que :

$$e^{z_j^{k+1}} = e^{z_j^k} e^{[\ln h(z^k)] \frac{-a_j^k}{\| a^k \|^2}} \quad j = 0,1,..,N$$

e substituindo acima

$$h(z^k) = g(x^k) \qquad e \qquad e^{z_j} = x_j$$

segue (4.10).

4.4 - Uso da "Projeção" na Programação Geométrica Posinomial

Chamaremos (4.10) de "Projeção" do ponto x^k na restrição $g_k(x)=1$, levando em conta a equivalência com Censor-Lent.

Existem várias maneiras de utilizarmos a projeção. Neste trabalho foi utilizada projeção na restrição mais violada, e projeção em todas as restrições violadas. No capítulo seguinte apresentaremos resultados numéricos destas implementações.

Para efeito de exposição apenas, escolhemos uma restrição $g_r(x)$ qualquer tal que :

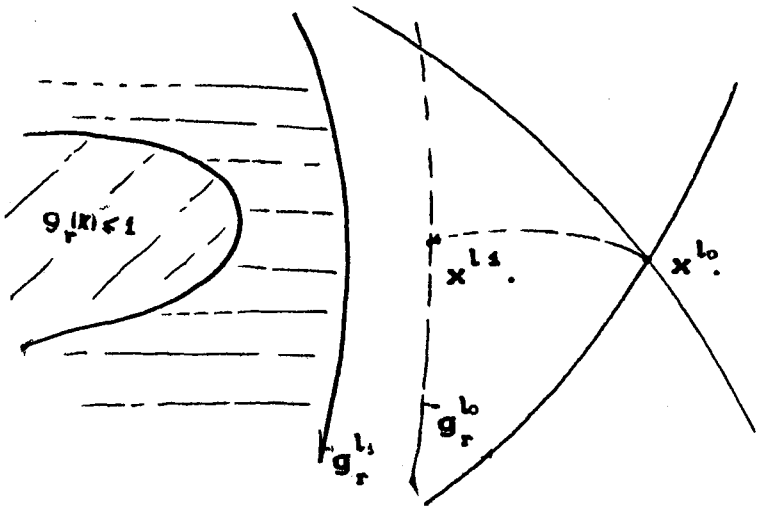
$$g_r(x) > 1 + \varepsilon \qquad (4.17)$$

onde r é a restrição mais violada (se existir). Projetamos o ponto x^l, obtido no PL na iteração l, na restrição (4.17), usando o seguinte procedimento :

i) Calcula-se a^l e g_r^l onde a^l (vetor de expoentes do monômio condensado em x^{lo}) e $g_r^{lo} = g_r(x^{lo})$;

ii)Determina-se x^{l1}, usando a equação (4.9).

Após a projeção ter sido efetuada, onde obtemos o ponto x^{l1}, podemos condensar $g_r(x)$ (ou outra restrição qualquer), no ponto x^{l1} obtido. Essa estratégia pode ser vista geometricamente, através da figura 4.2 :



g_r^l : é a condensação de $g_r(x)$ no ponto x^l.

g_r^l : é a condensação da $g_r(x)$ no ponto x^{l1}.

fig. 4.2

De (4.5) segue que x^{l1} satisfaz :

$$g_r^{lo}(x^{l1}) = 1$$

Algoritmo :

Passo único :

Seja x^{lo} a solução fornecida pelo programa monomial na iteração ℓ.

Se existir algum r tal que $g_r(x^l) > 1 + \varepsilon$ para $0 \leq r \leq K$

Então determinar :

i) a_j^{lo} através das equação (2.4),

ii) $g_r^{lo} = g_r(x^{lo})$

e faça :

a) $x_j^{li} = x_j^{lo} \left[g_r^{lo} \right]^{- \frac{a_j^{lo}}{\|a^{lo}\|^2}}$ $\quad j = 0,1,..,N$

b) Se $x_j^{li} < \underline{x}_j$

então

$x_j^{li} = \underline{x}_j$

senão

Se $x_j^{li} > \bar{x}_j$

então

$x_j^{li} = \bar{x}_j$

Senão FIM (neste caso a restrição não está violada).

Outra idéia é projetar só nas restrições muito violadas, isto é, a partir de um valor mínimo de projeção (VMP): Neste caso, basta substituirmos no algoritmo acima a seguinte restrição :

$$g_r(x^l) > VMP \quad (\text{onde } VMP > > 1 + \varepsilon)$$

no lugar da $g_r(x^l) > 1 + \varepsilon$.

CAPÍTULO V

EXPERIÊNCIAS COMPUTACIONAIS

5.1 - Introdução

Faremos a seguir uma apresentação dos resultados obtidos sobre alguns exemplos extraidos dos artigos de estudo. A elaboração dos programas está descrita no apêndice B e as suas listagens foram colocadas em anexo.

Os programas foram desenvolvidos utilizando a linguagem FORTRAN 77 e executados em micro computadores da linha PCxt (16 bits), pertencentes ao Laboratório do ICMSC.

Na seção 5.3 apresentaremos problemas Posinomiais e na seção 5.4 problemas Signomiais.

5.2 - Tipos de Cortes :

Para efeito de desenvolvimento de métodos foram utilizados dois tipos diferentes de cortes, aplicado no PGP :

a) Condensar a restrição mais violada;

b) Condensar todas as restrições violadas.

As escolhas desses cortes foram baseadas em algumas sugestões dadas pelas referências [12,18]. Fizemos, porém, uma nova tentativa de melhorar esses cortes, com o uso da projeção (capítulo IV), que resultou em quatro tipos de estratégias a serem utilizadas num problema posinomial (ver apêndice B, seção 3), visto que a condensação ocorre nas restrições posinomiais.

Essas estratégias diferenciam os quatro programas que

usamos para efeito de experiências computacionais :

i)Condensar a restrição mais violada, sem projeção;

ii)Condensar a restrição mais violada, com projeção;

iii)Condensar todas as restrições violadas, sem projeção;

iv)Condensar todas as restrições violadas, com projeção.

Os exemplos que adotamos para comparações foram usados na literatura para o mesmo fim.

5.3 - Experiências numéricas com problemas Posinomiais :

Para cada exemplo apresentado a seguir, faremos uma comparação entre os métodos desenvolvidos neste trabalho. Construiremos uma tabela mostrando para cada iteração de PGP, o número total de iterações, o número de restrições e a solução ótima do PL. O resultado final mostrará o total de iterações que as rotinas do PL e PGP sofreram.

Exemplo 5.1 :

Considere o seguinte problema posinomial :

$$\begin{aligned} \min \quad & x_0 \\ \text{suj. a} \quad & x_0^{-1} x_1^{-1} x_2^{-1} x_3^{-1} \leq 1 \\ & 2x_1 + x_2 + 3x_3 \leq 1 \\ & x_1 + 3x_2 + 2x_3 \leq 1 \\ & x_1 + x_2 + x_3 \leq 1 \\ & 0.1 \leq x_j \leq 1000,0 \quad j = 0,1,2,3 \end{aligned}$$

ponto inicial adotado $x^0 = (1.5,\ 0.25,\ 0.2,\ 0.167)$

Resultados obtidos :

i) Condensar a restrição mais violada :

a) Sem Projeção (SP) :

Programa :PCOMACP.FOR
TOLERANCIA PGP = .00050

EXEMPLO Numero : 5.1

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
1	4	4	196.56840	
			.27953	
			.18199	
			.10000	
2	2	5	199.91960	
			.23947	
			.17704	
			.11799	
3	2	6	201.57000	
			.22203	
			.17466	
			.12792	
4	2	7	202.30330	
			.22878	
			.17712	
			.12199	
5	2	8	202.40240	
			.21648	
			.17416	
			.13104	
6	3	9	202.58820	
			.21377	
			.17324	
			.13329	
7	2	10	202.72940	
			.21754	
			.17389	
			.13040	
7	17	10		202.72940
				.21754
				.17389
				.13040

Tabela 5.1 - A : Sem projeção.

b) Com Projeção (CP) :

Programa :PCOMACP.FOR
TOLERANCIA PGP = .000500

EXEMPLO Numero : 5.1

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
1	4	4	196.56840	
			.27953	
			.18199	
			.10000	
2	2	5	200.19280	
			.23649	
			.17664	
			.11957	
3	2	6	201.71580	
			.22065	
			.17447	
			.12878	
4	2	7	202.26330	
			.22692	
			.17662	
			.12336	
5	2	8	202.45520	
			.21547	
			.17396	
			.13177	
6	3	9	202.64550	
			.21306	
			.17316	
			.13376	
7	2	10	202.73050	
			.21655	
			.17376	
			.13109	
7	17	10		202.73050
				.21655
				.17376
				.13109

Tabela 5.1 - B : Com projeção.

As tabelas desta seção mostram para cada iteração k (1ª coluna) do PGP, o número de iterações no PL (2ª coluna); o número de restrições do PL (3ª coluna) e a solução ótima do PL. As últimas linhas mostram os totais de iterações e a solução do PGP.

Representaremos no gráfico 5.1 o erro de cada iteração :

$$ERR = \| x^{*} - x^{i} \|_{\infty} ,$$

a fim de melhor visualizarmos a convergência de cada método.

ITP	ERR SP (A)	ERR CP (B)
1	6.161	6.1621
2	2.8098	2.5377
3	1.1514	1.0147
4	0.4261	0.4672
5	0.327	0.2753
6	0.1412	0.085
7	0.0	0.0

Tabela 5.1

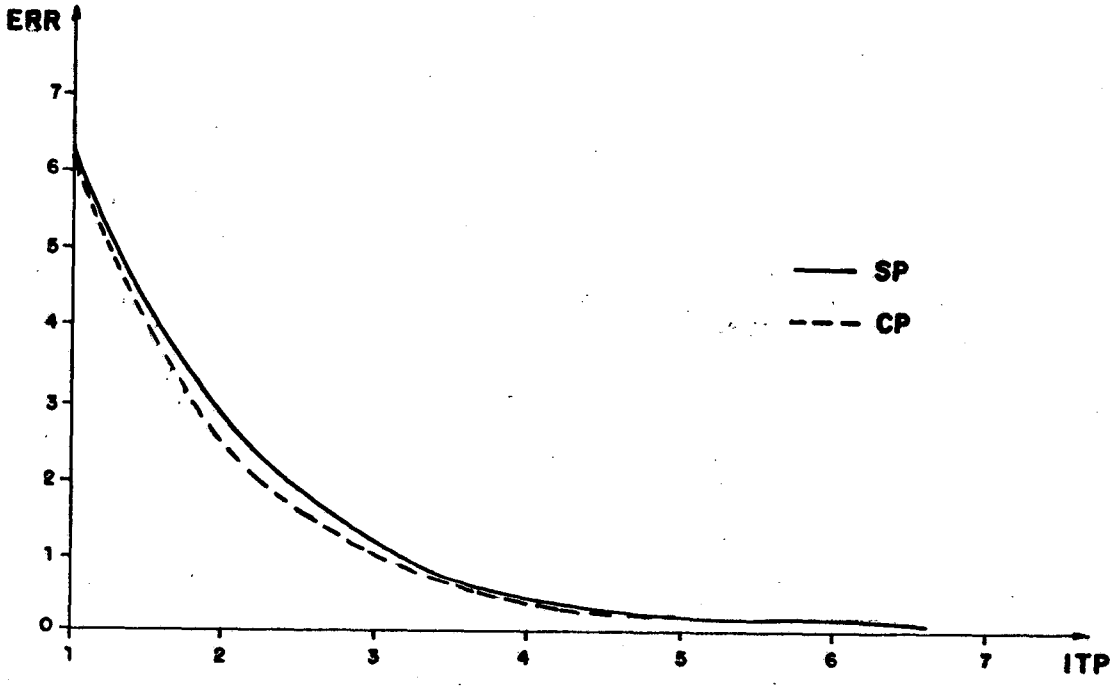


Gráfico 5.1 : $\| x^* - x^{ITP} \|_\infty$ x ITP

ii)Condensar todas as restrições violadas :

a) Sem Projeção (TSP) :

Programa :PCOTOSP.FOR
TOLERANCIA PGP = .00050

EXEMPLO Numero : 5.1

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
1	4	4	196.56840	
			.27953	
			.18199	
			.10000	
2	2	6	199.99930	
			.23860	
			.17692	
			.11045	
3	2	8	201.71970	
			.22061	
			.17446	
			.12880	
4	4	10	202.56730	
			.21299	
			.17327	
			.13377	
5	2	11	202.70920	
			.21650	
			.17378	
			.13112	
5	14	11		202.70920
				.21650
				.17378
				.13112

Tabela 5.2 - A : Sem projeção.

b) Com Projeção (TCP) :

Programa :PCOTOCP.FOR
TOLERANCIA PGP = .00050

EXEMPLO Numero : 5.1

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
1	4	4	196.56840	
			.27953	
			.18199	
			.10000	
2	3	6	200.19280	
			.23649	
			.17664	
			.11957	
3	3	8	201.81430	
			.22070	
			.17433	
			.12879	
4	3	10	202.56540	
			.21301	
			.17327	
			.13375	
5	2	11	202.70900	
			.21654	
			.17379	
			.13109	
5	15	11		202.70900
				.21654
				.17379
				.13109

Tabela 5.2 - B :Com projeção.

O gráfico 5.2 faz uma comparação entre o processo com projeção e sem projeção, mostrando um diagrama de ERRO x ITP.

ITP	ERR TSP (A)	ERR TCP (B)
1	6.1406	6.1406
2	2.7099	2.5162
3	0.9895	0.8947
4	0.1419	0.1436
5	0.0	0.0

Tabela 5.2

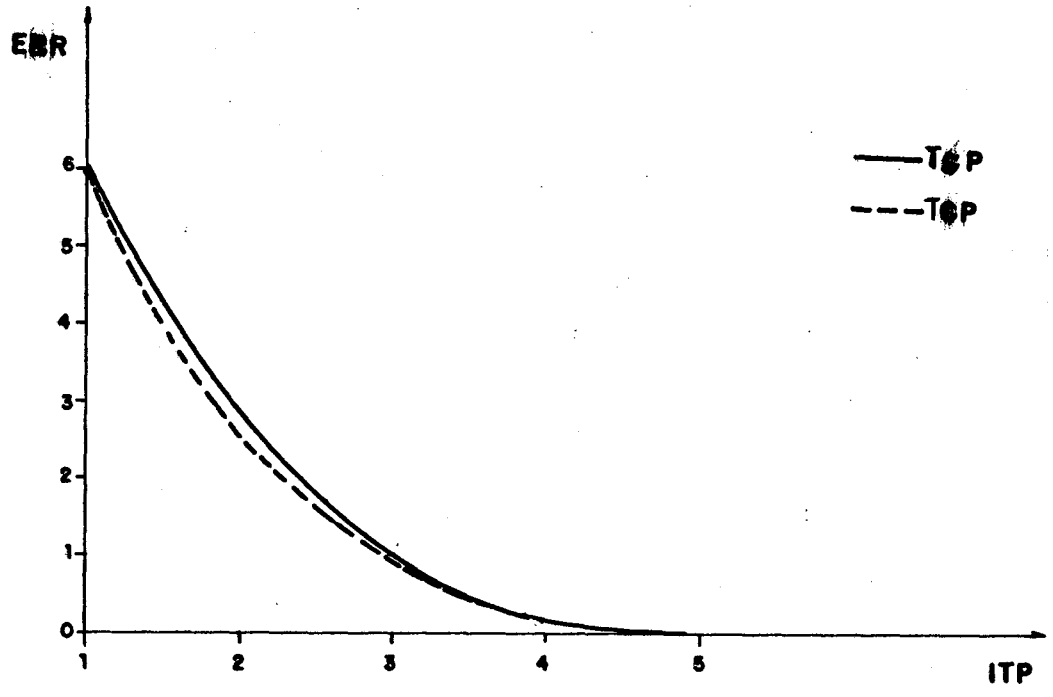


Gráfico 5.2 : $\|x^* - x^{ITP}\|_\infty$ x ITP

Analisando os gráficos 5.1 e 5.2, podemos notar que a influência da projeção foi desprezível. Isto se deve ao fato de que já na primeira solução obtida pelo PL violou muito pouco as restrições.

Como há um custo de projetar e a projeção se mostra ineficaz para restrições pouco violadas, adotaremos então a seguinte estratégia :

- Seja VMP (valor mínimo para a projeção, naturalmente maior do que 1 + ε), e assim acionamos a rotina de projeção apenas

quando :

$$g_k(x^{ITP}) > VMP \ .$$

Exemplo 5.2 -

Seja o seguinte problema posinomial :

$$\min \quad x_0$$

$$\text{suj. a} \quad 20x_0^{-1}x_1 + 10x_0^{-1}x_2 + 30x_0^{-1}x_3 + 15x_0^{-1}x_4 + 1500x_0^{-1}x_2^{-1}x_3^{-1}x_4^{-1} \leq 1$$

$$\frac{1}{6}x_1 + \frac{1}{12}x_2 + \frac{1}{12}x_3 + \frac{1}{60}x_4 \leq 1$$

$$\frac{2}{37}x_1 + \frac{6}{37}x_2 + \frac{8}{37}x_3 + \frac{4}{37}x_4 \leq 1$$

$$\frac{4}{45}x_1 + \frac{3}{45}x_2 + \frac{7}{45}x_3 + \frac{14}{45}x_4 \leq 1$$

$$\frac{3}{45}x_1 + \frac{3}{45}x_2 + \frac{3}{45}x_3 + \frac{3}{45}x_4 \leq 1$$

$$0.1 \leq x_j \leq 1000.0 \qquad j = 0,1,2,3,4$$

Ponto inicial $x^0 = (20.0,\ 20.0,\ 20.0,\ 20.0,\ 20.0)$.

i) Condensar a restrição mais violada :

a) Sem projeção :

Programa :PCOMASP.FOR
TOLERANCIA PGP = .00050

EXEMPLO Numero : 5.2

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
1	2	5	7.50129	
			.10000	
			.10000	
			.10000	
			.10000	
2	5	6	65.57029	
			2.18129	
			195.77940	
			.10000	
			.53562	
3	3	7	86.70947	
			10.10795	
			5.94562	
			.10000	
			2.87851	
4	3	8	138.41960	
			1.14257	
			5.99017	
			.87335	
			1.81295	
5	3	9	191.48140	
			5.01404	
			.72616	
			4.31439	
			.84019	
6	2	10	199.53050	
			3.81408	
			3.33810	
			1.21356	
			1.23536	
7	3	11	223.61180	
			2.40613	
			5.97579	
			.78682	
			1.54474	
8	2	12	234.06810	
			2.13993	
			3.29361	
			1.33326	
			1.39420	
9	2	13	239.19850	
			2.02415	
			2.48271	
			1.71228	
			1.32801	
10	3	14	244.25980	
			2.81331	
			3.40021	
			.72931	
			1.73147	
11	2	15	247.24700	
			3.09137	
			2.48072	
			1.48484	
			1.26380	
12	2	16	256.91530	
			3.16909	
			3.18530	
			.93443	
			1.23977	

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
13	2	17	262.39240	
			3.51357	
			2.42815	
			1.76209	
			.89079	
14	2	18	262.64500	
			2.81719	
			2.67233	
			1.42206	
			1.13241	
15	2	19	262.78440	
			2.49700	
			2.81592	
			1.26492	
			1.29106	
16	2	20	265.85450	
			2.14104	
			2.68286	
			1.42909	
			1.36564	
17	2	21	266.09230	
			2.59925	
			2.85171	
			1.22527	
			1.27228	
18	2	22	267.01090	
			2.48585	
			2.58188	
			1.45781	
			1.23579	
19	2	23	267.07110	
			2.73890	
			2.67567	
			1.33861	
			1.19610	
20	2	24	267.50210	
			2.52453	
			2.72536	
			1.33697	
			1.24407	
21	2	25	267.59340	
			2.60458	
			2.63163	
			1.39912	
			1.20842	
22	2	26	267.75550	
			2.53125	
			2.70906	
			1.33793	
			1.24499	
23	2	27	267.75600	
			2.64365	
			2.73522	
			1.30254	
			1.22274	
24	2	28	267.85060	
			2.62397	
			2.68246	
			1.35039	
			1.21547	
24	56	28		267.85060
				2.62397
				2.68246
				1.35039
				1.21547

Tabela 5.3 - A : Sem projeçao.

Tabela 5.3 - A : Sem projeção.

b) Com projeção : VMP = 1.2

TOLERANCIA PGP = .000500
VMP = 1.2
EXEMPLO Numero : 5.2

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
1	2	5	7.50129	
			.10000	
			.10000	
			.10000	
			.10000	
2	5	6	101.20970	
			3.31521	
			62.86142	
			.10000	
			2.38797	
3	2	7	197.69000	
			4.42382	
			3.75253	
			1.07472	
			1.22711	
4	2	8	222.92030	
			2.45559	
			4.05121	
			1.09473	
			1.41511	
5	2	9	235.30910	
			3.15228	
			2.62592	
			1.53185	
			1.22235	
6	3	10	247.94440	
			3.28573	
			3.67868	
			.46539	
			1.75136	
7	2	11	249.84080	
			3.48529	
			2.82825	
			1.37930	
			.95810	
8	2	12	256.21810	
			2.25102	
			3.04279	
			1.08241	
			1.48814	
9	3	13	257.89430	
			2.41533	
			2.74876	
			1.63625	
			1.14279	
10	2	14	263.20000	
			2.86753	
			2.95481	
			1.18173	
			1.19442	
11	2	15	264.09550	
			2.72023	
			2.67782	
			1.36756	
			1.18568	
12	2	16	264.27580	
			2.27979	
			2.77432	
			1.33439	
			1.34925	
13	2	17	266.32590	
			2.60549	
			2.70105	
			1.35939	
			1.22366	
14	2	18	266.68070	
			2.71764	
			2.78203	
			1.24804	
			1.23460	
15	2	19	267.33950	
			2.50362	
			2.55269	
			1.48019	
			1.22617	
16	2	20	267.49980	
			2.47729	
			2.71761	
			1.33870	
			1.25998	
17	4	21	267.71220	
			2.52168	
			2.84079	
			1.23881	
			1.26836	
18	2	22	267.71740	
			2.60910	
			2.73005	
			1.31130	
			1.22967	
19	2	23	267.72010	
			2.65573	
			2.67420	
			1.35064	
			1.21003	
20	2	24	267.90270	
			2.73728	
			2.72790	
			1.29711	
			1.20017	
21	3	25	267.92660	
			2.58682	
			2.74318	
			1.30406	
			1.23650	
21	50	25		267.92660
				2.58682
				2.74318
				1.30406
				1.23650

Tabela 5.3 - B : Com projeçao VMP=1.2 .

c) Com projeção : VMP = 1.4

Programa :PCOMACP.FOR
TOLERANCIA PGP = .000500
VMP = 1.4
EXEMPLO Numero : 5.2

ITP *	ITL *	N.CO *	SOL. OTIMA PL *	SOL. OTIMA PGP *
1	2	5	7.50129	
			.10000	
			.10000	
			.10000	
			.10000	
2	5	6	101.20970	
			3.31521	
			62.86142	
			.10000	
			2.38797	
3	2	7	197.69000	
			4.42382	
			3.75253	
			1.07472	
			1.22711	
4	2	8	223.64330	
			2.41692	
			4.05958	
			1.09528	
			1.42056	
5	2	9	236.16660	
			3.08693	
			2.63950	
			1.52966	
			1.22921	
6	3	10	249.14390	
			3.09552	
			3.68611	
			.46920	
			1.81246	
7	2	11	250.07600	
			3.33537	
			2.87154	
			1.31169	
			1.01652	
8	3	12	257.47960	
			2.26916	
			2.76834	
			1.61503	
			1.19740	
9	2	13	257.66540	
			2.16711	
			2.97275	
			1.20883	
			1.43425	
10	2	14	263.48920	
			2.76729	
			2.96366	
			1.18175	
			1.22231	
11	2	15	264.91260	
			2.69415	
			2.67890	
			1.37213	
			1.18946	
12	2	16	265.46830	
			2.37110	
			2.75204	
			1.33842	
			1.30765	
13	2	17	266.73360	
			2.63177	
			2.69217	
			1.36589	
			1.21030	
14	2	18	267.03990	
			2.75489	
			2.77169	
			1.25462	
			1.21840	
15	2	19	267.57500	
			2.58032	
			2.60590	
			1.41930	
			1.21672	
16	2	20	267.59340	
			2.54846	
			2.74024	
			1.31141	
			1.24656	
17	3	21	267.76890	
			2.62461	
			2.79869	
			1.25751	
			1.23700	
18	2	22	267.79410	
			2.65929	
			2.73089	
			1.30462	
			1.21802	
18	42	22		267.79410
				2.65929
				2.73089
				1.30462
				1.21802

Tabela 5.3 - C : Com projeçao VMP = 1.4.

d) Erro x ITP

Na tabela 5.3 D, apresentamos o erro de cada iteração das tabelas 5.3 A,B e C.

$$\|x^{*} - x^{ITP}\|_{\infty} = ERR$$

ITP	SP	CP VMP=1.2	CP VMP=1.4
1	260,35	260,42	260,29
2	202,27	166,42	166,58
3	181,14	70,24	70,10
4	129,48	45,64	44,15
5	76,87	32,62	31,63
6	68,32	19,98	18,65
7	44,24	18,09	17,72
8	33,78	11,71	10,91
9	28,65	10,03	10,18
10	28,60	4,78	4,03
11	20,60	3,83	2,88
12	10,94	3,65	2,33
13	5,46	1,61	1,06
14	5,20	1,25	0,75
15	5,07	0,59	0,22
16	2,00	0,43	0,20
17	1,76	0,21	0,07
18	0,84	0,21	0,0
19	0,78	0,21	--
20	0,35	0,15	--
21	0,26	0,0	--
22	0,10	--	--
23	0,09	--	--
24	0,0	--	--

Tabela 5.3 - D : ERR x ITP

A seguir faremos uma representação gráfica do erro apresentado pela tabela 5.3 - D (ERR x ITP).

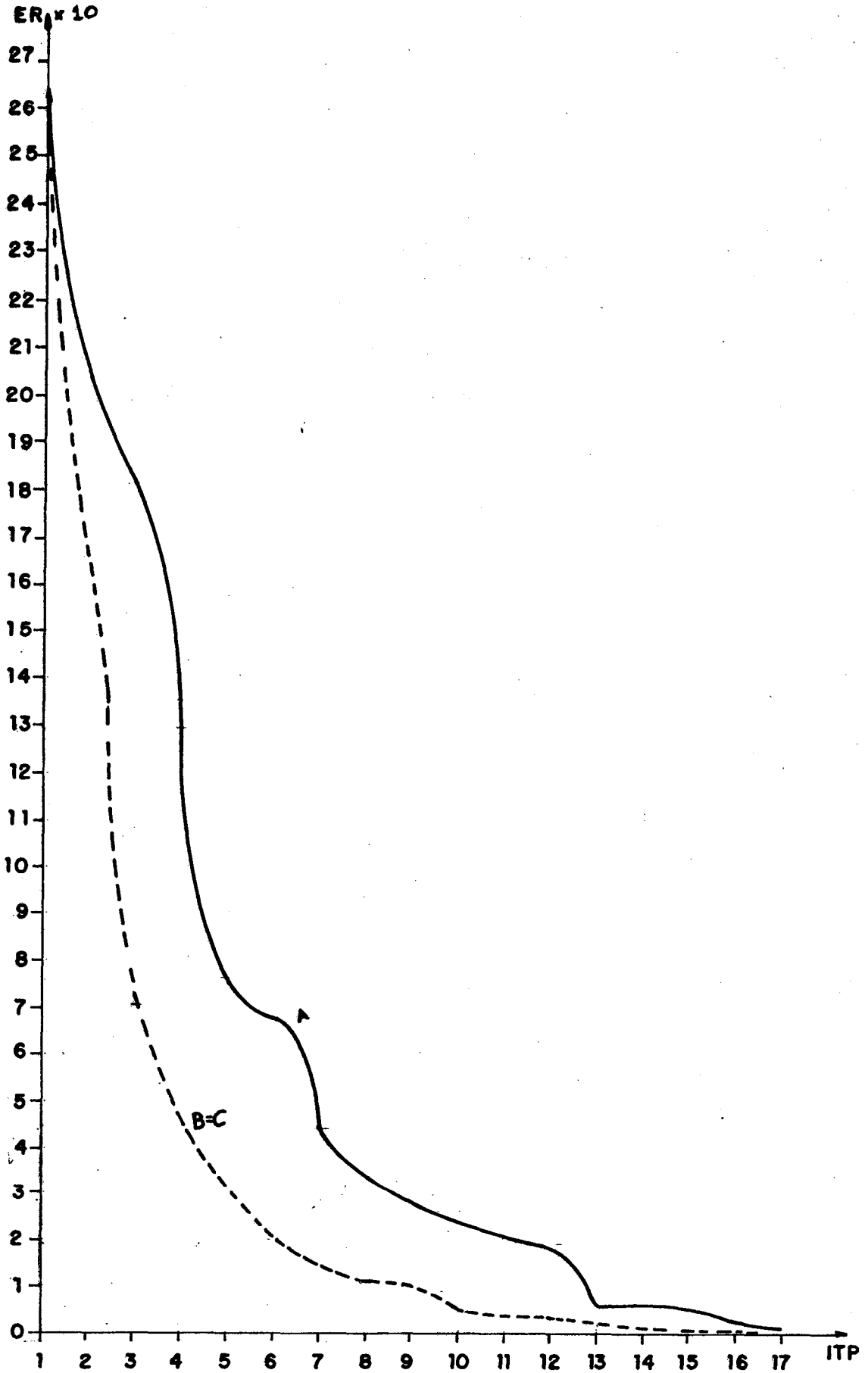


Gráfico 5.3 : Representação de ERR x ITP (tabela 5.3 - D).

ii) Condensar todas as restrições violadas :

a) Sem projeção :

Programa :PCOTOSP.FOR
TOLERANCIA PGP = .00050

EXEMPLO Numero : 5.2

ITP *	ITL *	N.CO *	SOL. OTIMA PL *	SOL. OTIMA PGP *
1	2	5	7.50129	
			.10000	
			.10000	
			.10000	
			.10000	
2	5	6	65.57829	
			2.18129	
			195.77940	
			.10000	
			.53562	
3	4	11	86.70947	
			10.10793	
			5.94563	
			.10000	
			2.87851	
4	4	16	138.41970	
			1.14257	
			5.99017	
			.87335	
			1.81295	
5	4	19	191.48130	
			5.01403	
			.72616	
			4.31439	
			.84019	
6	3	23	199.53050	
			3.81407	
			3.33811	
			1.21356	
			1.23536	
7	6	27	237.35210	
			2.06478	
			2.74658	
			1.56582	
			1.35130	
8	6	30	261.15770	
			2.71523	
			2.77381	
			1.29324	
			1.23875	
9	4	33	266.97860	
			3.58902	
			2.12816	
			1.88422	
			.93852	
10	3	36	267.00730	
			2.86714	
			2.40663	
			1.60769	
			1.10631	
11	3	39	267.02330	
			2.52741	
			2.57871	
			1.47059	
			1.21338	
12	5	42	267.57770	
			2.65917	
			2.65195	
			1.37565	
			1.20079	
13	4	45	267.65910	
			2.41413	
			2.73402	
			1.33896	
			1.27508	
14	4	48	267.79100	
			2.57464	
			2.71187	
			1.33298	
			1.23236	
14	57	48		267.79100
				2.57464
				2.71187
				1.33298
				1.23236

Tabela 5.4 - A : Sem projeção.

b) Com projeção : VMP=1.2

Programa :PCOTOCP.FOR
TOLERANCIA PGP = .00050
VMP = 1.2
EXEMPLO Numero : 5.2

ITP *	ITL *	N.CO *	SOL. OTIMA PL *	SOL. OTIMA PGP *
1	2	5	7.50129	
			.10000	
			.10000	
			.10000	
			.10000	
2	5	6	101.20970	
			3.31521	
			62.86142	
			.10000	
			2.38797	
12	50	38		267.89660
				2.72517
				2.72021
				1.30367
				1.20212

Tabela 5.4 - B : Com projeção VMP = 1.2

c) Com projeção : VMP=1.4

VMP = 1.4

ITP *	ITL *	N.CO *	SOL. OTIMA PL *	SOL. OTIMA PGP *
1	2	5	7.50129	
			.10000	
			.10000	
			.10000	
			.10000	
2	5	6	101.20970	
			3.31521	
			62.86142	
			.10000	
			2.38797	
3	3	11	197.69000	
			4.42382	
			3.75253	
			1.07472	
			1.22711	
12	46	38		267.83450
				2.60212
				2.71634
				1.32469
				1.22728

Tabela 5.4 - C :Com projeção VMP = 1.4 .

d) ERR x ITP

Faremos a seguir a tabela de erro para cada resultado visto em (ii) :

$$\| x^{*} - x^{ITP} \|_{\infty} = \text{ERR x ITP}$$

ITP	SP	CP > 1.2	CP > 1.4
1	260,29	260,40	260,99
2	202,21	166,16	166,69
3	181,08	70,16	70,14
4	129,97	.	.
5	76,91	.	.
6	68,26	.	.
7	30,44	.	.
8	6,69	.	.
9	0,81	.	.
10	0,81	.	.
11	0,78	.	.
12	0,21	0,0	0,0
13	0,16		
14	0,00		

Tabela 5.4 - D : ERR x ITP.

Podemos notar dois fatos neste exemplo :

a) A projeção tem um excelente desempenho nas primeiras iterações, quando a violação nas restrições é bem acentuada. Isto fica claro quando o valor VMP passa de 1,2 a 1,4 (ver tabela 5.3 - B e C).

A lição do exemplo anterior, o qual sugeria que projetar com pouca violação não trazia benefícios, fica claro agora. Valores superiores a 1,4 foram testados, porém sem melhoria. Para valores muito grandes tenderia ao caso sem projeção. Surge a questão : qual VMP escolher ? Provavelmente este valor dependerá da característica do problema em questão.

b) Quando comparamos as tabelas de erros dos casos (i) e

(ii) (ou diretamente no gráfico 5.2), até a iteração 6 não houve diferença substancial entre condensar na mais violada ou em todas violadas, embora neste último a 6ª iteração apresente 23 restrições no PL, enquanto que a primeira tinha apenas 10.

Apresentaremos a seguir dois exemplos. O primeiro, é um caso onde as restrições são monomiais e o segundo exemplo trata de um problema onde todas restrições são não-lineares e obtiveram resultados diferentes dos demais.

Exemplo 5.3 :

Considere o seguinte problema posinomial :

$$\begin{aligned} \min \quad & x_0 \\ \text{suj. a} \quad & 2x_0^{-1} x_1 x_2^{1/2} + 5x_0^{-1} x_2 x_3^{-1} x_4^2 + 3x_0^{-1} x_2^{-1} x_3^2 \leq 1 \\ & 5.94\, x_1 x_2^{1.475} x_3 x_4^{-1.95} \leq 1 \\ & 2.0\, x_1^{-1} x_2^{-1} x_3^{-1} \leq 1 \\ & 0.001 \leq x_j \leq 1000.0 \qquad j = 0,1,2,3,4 \end{aligned}$$

Ponto inicial $x^0 = (1.0,\ 1.0,\ 1.0,\ 1.0,\ 1.0)$

i) Condensar a restrição mais violada :

Mostraremos apenas este caso (i) por achar irrelevante o caso (ii).

a) Sem projeção :

Programa :PCOMACP.FOR
TOLERANCIA PGP = .000500

EXERCICIO Numero : 5.3

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
1	5	3	1.90821	
			2.00000	
			.00100	
			1000.00000	
			.66133	
2	3	4	10.11743	
			1000.00000	
			.01059	
			.18894	
			1.17497	
3	2	5	19.08244	
			24.20592	
			.10238	
			.80700	
			2.04217	
4	2	6	22.87761	
			25.55379	
			.15224	
			.51411	
			2.24936	
5	2	7	24.35209	
			11.38697	
			.21028	
			.83525	
			2.43350	
6	2	8	26.63377	
			11.63476	
			.25609	
			.67126	
			2.55315	
7	2	9	26.77489	
			8.22355	
			.28024	
			.86784	
			2.60983	
8	2	10	26.97661	
			5.02364	
			.31852	
			1.24992	
			2.69250	
9	2	11	27.15366	
			7.26136	
			.27381	
			1.00590	
			2.59512	
10	2	12	27.26147	
			8.07683	
			.24985	
			.88190	
			2.53786	
11	3	13	27.47856	
			12.62937	
			.21984	
			.72034	
			2.45999	
12	2	14	27.49850	
			10.23610	
			.24362	
			.80202	
			2.52230	
13	2	15	27.51122	
			8.95160	
			.26012	
			.85891	
			2.56289	
14	2	16	27.51993	
			9.88496	
			.24217	
			.83548	
			2.51864	
15	2	17	27.52301	
			9.18118	
			.25233	
			.86330	
			2.54398	
16	2	18	27.52904	
			7.94713	
			.27343	
			.92038	
			2.59425	
17	2	19	27.52938	
			8.56333	
			.26116	
			.89431	
			2.56537	
18	2	20	27.52958	
			8.93786	
			.25437	
			.87970	
			2.54897	
18	41	20		27.52958
				8.93786
				.25437
				.87970
				2.54897

Tab. 5.5 - A : Sem projeção.

Tabela 5.5 - A : Sem projeção .

b) Com projeção :

VMP = 1.2

IIP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
1	5	3	1.90821	
			2.00000	
			.00100	
			1000.00000	
			.66133	
2	3	4	10.25173	
			1000.00000	
			.01090	
			.16341	
			1.16351	
3	2	5	21.58987	
			12.28971	
			.15757	
			1.03281	
			2.26829	
4	2	8	23.80044	
			22.90169	
			.17059	
			.51192	
			2.31261	
5	2	7	25.32514	
			10.83301	
			.23230	
			.79475	
			2.49324	
6	2	8	26.61104	
			5.96236	
			.29717	
			1.12878	
			2.64738	
7	2	9	27.08727	
			8.89490	
			.29932	
			.96909	
			2.65205	
	2	10	27.16574	
			8.85358	
			.24260	
			.93117	
			2.51972	
8	2	11	27.34146	
			8.73602	
			.25763	
			.88864	
			2.55688	
10	2	12	27.47922	
			10.42028	
			.22673	
			.84652	
			2.47855	
11	2	13	27.50474	
			9.91179	
			.23973	
			.84170	
			2.51243	
12	2	14	27.51286	
			8.35067	
			.24286	
			.86071	
			2.52039	
13	2	15	27.52008	
			8.31993	
			.24701	
			.86875	
			2.53082	
14	2	16	27.53023	
			8.89667	
			.25179	
			.89281	
			2.54266	
14	32	16		27.53023
				8.89667
				.25179
				.89281
				2.54266

VMP = 1.4

IIP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
1	5	3	1.90821	
			2.00000	
			.00100	
			1000.00000	
			.66133	
2	3	4	10.25173	
			1000.00000	
			.01090	
			.16341	
			1.16351	
3	2	5	21.58987	
			12.28971	
			.15757	
			1.03281	
			2.26829	
4	2	8	23.80044	
			22.90169	
			.17059	
			.51192	
			2.31261	
5	2	7	25.26362	
			11.15531	
			.22951	
			.78118	
			2.48591	
6	2	8	26.51176	
			6.23718	
			.29170	
			1.09929	
			2.63542	
7	2	9	27.05858	
			7.11452	
			.29853	
			.94802	
			2.64600	
8	2	10	27.15638	
			8.99211	
			.24306	
			.81509	
			2.52088	
8	2	11	27.35500	
			8.96770	
			.25635	
			.87001	
			2.55379	
10	2	12	27.48129	
			10.47751	
			.22813	
			.83673	
			2.48227	
11	2	13	27.49487	
			9.28547	
			.24478	
			.88000	
			2.52518	
12	3	14	27.51892	
			7.71337	
			.27318	
			.94923	
			2.59361	
13	2	15	27.52013	
			8.48315	
			.26009	
			.90645	
			2.56283	
14	2	16	27.52089	
			9.00915	
			.25215	
			.88040	
			2.54355	
14	33	16		27.52089
				9.00915
				.25215
				.88040
				2.54355

VMP = 1.2 **VMP = 1.4**

Tabelas 5.6 - A e B : Com projeção.

Este exemplo é um caso em que apenas uma restrição é violada desde a iteração ITP = 1 até a última. Com isto a condensação em todas violadas geraria a mesma sequência.

Exemplo 5.4 :

Considere o seguinte problema posinomial :

$$\min \quad x_o$$

$$\text{suj. a} \quad g_k(x) = \sum_{i=o}^{Tk} \theta_{ki} \prod_{j=o}^{N} x_j^{\varphi_{kij}} \leq 1 \qquad k = 0,1,2,3$$

$$0.1 \leq x_o \leq 10.0$$

$$0.1 \leq x_j \leq 100.0 \qquad j = 1,2,..,12$$

onde

$T_o = 1 \quad T_1 = 4 \quad T_2 = 12 \quad T_3 = 14$

Tabela dos valores de θ_{ki} :

i	θ_{oi}	θ_{1i}	θ_{2i}	θ_{3i}
1	100000	0.05367379	10^{-6}	10^{-6}
2	--	0.02186375	10^{-5}	10^{-5}
3		0.09773359	10^{-6}	10^{-6}
4		0.06694080	10^{-10}	10^{-9}
5		--	10^{-8}	10^{-9}
6			10^{-2}	10^{-3}
7			10^{-4}	10^{-3}
8			0.10898645	0.10898645
9			1.61080E-4	1.6108052E-05
10			1.00 E-23	1.0 E-23
11			1.93045E-6	1.9304541E-08
12			0.001	0.00001
13			-- --	1.1184059E-04
14				0.0001

- Valores de φ_{kij} :

j	φ_{oij}	j	φ_{oij}
1	-0.001331720	7	-0.008092
2	-0.002270927	8	-0.005
3	-0.002485460	9	-0.000909
4	-4.67	10	-0.00088
5	-4.671973	11	-0.00119
6	-0.008140		

k	i	j	φ_{kij}	k	i	j	φ_{kij}
1	1	1	1.0	2	11	5	1.0
1	2	2	1.0	2	11	12	-2.0
1	3	3	1.0	2	12	10	1.0
1	4	4	1.0	2	12	12	-1.0
1	4	5	1.0	3	1	1	1.0
2	1	1	1.0	3	2	2	1.0
2	2	2	1.0	3	3	3	1.0
2	3	3	1.0	3	4	4	1.0
2	4	4	1.0	3	5	5	1.0
2	4	12	1.0	3	6	7	1.0
2	5	5	1.0	3	7	8	1.0
2	5	12	-1.0	3	8	4	1.0
2	6	6	1.0	3	8	5	1.0
2	6	12	-1.0	3	9	2	1.0
2	7	7	1.0	3	9	5	1.0
2	7	12	1.0	3	10	2	1.0
2	8	4	1.0	3	10	4	1.0
2	8	5	1.0	3	10	5	1.0
2	9	2	1.0	3	11	2	1.0
2	9	5	1.0	3	11	4	-1.0
2	9	12	-1.0	3	11	5	1.0
2	10	2	1.0	3	12	9	1.0
2	10	4	1.0	3	13	1	1.0
2	10	5	1.0	3	13	9	1.0
2	11	2	1.0	3	14	11	1.0
2	11	4	-1.0	-	-	-	-

i) Condensar a restrição mais violada :

a) Sem Projeção :

Programa :PCONASP.FOR
TOLERANCIA PGP = .00050

EXEMPLO Numero : 5.4

ITP	ITL	N.CO	SOL. OTIMA PL Xo	SOL. OTIMA PGP
1	5	4	2.89874	
2	2	5	2.90383	
3	3	6	2.90588	
4	2	7	2.91450	
5	3	8	2.93764	
6	2	9	2.93920	
7	2	10	2.93986	
8	2	11	2.96488	
9	2	12	2.96644	
10	3	13	2.99720	
11	3	14	2.99996	
12	2	15	3.00027	
13	2	16	3.00366	
14	5	17	3.07871	
15	2	18	3.07997	
16	2	19	3.09598	
17	2	20	3.09600	
18	2	21	3.09604	
19	3	22	3.09803	
20	8	23	3.13053	
21	2	24	3.13096	
22	2	25	3.13115	
23	2	26	3.13128	
24	3	27	3.13165	
25	4	28	3.13320	
26	2	29	3.13327	
27	2	30	3.13343	
28	2	31	3.13355	
29	2	32	3.13356	
30	6	33	3.13841	
31	3	34	3.13845	
32	5	35	3.13916	
33	5	36	3.13966	
34	2	37	3.13967	
35	4	38	3.14016	
36	2	39	3.14032	
37	3	40	3.15062	
38	3	41	3.15200	
39	2	42	3.15477	
40	2	43	3.15479	
41	2	44	3.15610	
42	12	45	3.16066	
43	3	46	3.16187	
44	2	47	3.16188	
45	2	48	3.16284	
46	2	49	3.16663	
47	2	50	3.16701	
48	2	51	3.16745	
49	3	52	3.16866	
50	2	53	3.16874	
51	3	54	3.16881	
52	5	55	3.17040	
53	3	56	3.17111	
54	3	57	3.17147	
55	2	58	3.17148	
56	2	59	3.17206	

ITP	ITL	NRPL	SOL.OTIMA PGP
56	165	59	3.17206
			1.12450
			1.23005
			3.09296
			.51729
			17.44209
			1.44298
			4.00913
			2.75807
			5.81009
			1.87757
			5.20111
			8.32296

Tabela 5.7 - A : Sem projecao.

b) Com projeção :

VMP = 1.2

Programa :PCOMACP.FOR
VMP = 1.2
TOLERANCIA PGP = .000500

EXEMPLO Numero : 5.4

ITP	ITL	N.CO	SOL. OTIMA PL Xo	SOL. OTIMA PGP
1	5	4	2.89874	
2	2	5	2.90722	
3	3	6	2.91347	
4	2	7	2.91885	
5	2	8	2.94285	
6	2	9	2.94485	
7	2	10	2.96007	
8	2	11	2.96894	
9	3	12	2.98109	
10	2	13	2.98285	
11	2	14	2.98353	
12	2	15	2.99640	
13	2	16	3.00026	
14	2	17	3.00361	
15	3	18	3.08181	
16	3	19	3.08350	
17	6	20	3.09064	
18	4	21	3.11048	
19	3	22	3.11092	
20	3	23	3.11103	
21	6	24	3.12056	
22	2	25	3.12099	
23	2	26	3.12099	
24	10	27	3.13059	
25	5	28	3.13353	
26	4	29	3.13713	
27	2	30	3.13733	
28	2	31	3.13741	
29	7	32	3.13848	
30	3	33	3.13865	
31	2	34	3.13888	
32	4	35	3.14087	
33	2	36	3.14089	
34	9	37	3.14206	
35	2	38	3.14297	
36	2	39	3.14299	
37	3	40	3.14630	
38	2	41	3.14632	
39	3	42	3.15706	
40	2	43	3.15708	
41	2	44	3.15815	
42	2	45	3.15854	
43	2	46	3.15856	
44	11	47	3.16101	
45	2	48	3.16125	
46	2	49	3.16129	
47	7	50	3.16270	
48	2	51	3.16270	
49	4	52	3.16651	
50	3	53	3.16694	
51	3	54	3.16935	
52	2	55	3.17007	
53	2	56	3.17265	
54	2	57	3.17265	
55	3	58	3.17338	
56	3	59	3.17407	
57	6	60	3.17490	
58	3	61	3.17649	
59	3	62	3.17808	
60	2	63	3.17863	
60	195	63		3.17863
				1.12310
				1.32311
				3.07616
				.22036
				41.39471
				2.43116
				2.71354
				1.75930
				1.42818
				2.55744
				7.67564
				12.19078

Tabela 5.7 - B : Com projeçao VMP = 1.2

VMP = 1.4

Programa :PCOMACP.FOR
TOLERANCIA PGP = .000500
VMP = 1.4
EXEMPLO Numero : 5.4

ITP	ITL	N.CO	SOL. OTIMA PL Xo	SOL. OTIMA PGP
1	5	4	2.89874	
2	2	5	2.90722	
3	3	6	2.91347	
4	2	7	2.91885	
5	2	8	2.94437	
6	2	9	2.94634	
7	2	10	2.96929	
8	2	11	2.97018	
9	3	12	2.98136	
10	2	13	2.98326	
11	2	14	2.99739	
12	2	15	2.99797	
13	2	16	3.00176	
14	2	17	3.00513	
15	3	18	3.08035	
16	4	19	3.09436	
17	2	20	3.09456	
18	6	21	3.11689	
19	3	22	3.11771	
20	2	23	3.11773	
21	2	24	3.11776	
22	2	25	3.11825	
23	2	26	3.11830	
24	7	27	3.13776	
25	4	28	3.13862	
26	5	29	3.13875	
27	2	30	3.13875	
28	4	31	3.13984	
29	3	32	3.14004	
30	3	33	3.14011	
31	2	34	3.14017	
32	2	35	3.14020	
33	3	36	3.14138	
34	2	37	3.14153	
35	2	38	3.14408	
36	2	39	3.14471	
37	11	40	3.14672	
38	4	41	3.14995	
39	2	42	3.15004	
40	2	43	3.15015	
41	8	44	3.15473	
42	6	45	3.15645	
43	2	46	3.15660	
44	2	47	3.15660	
45	2	48	3.15662	
46	3	49	3.15750	
47	9	50	3.16286	
48	3	51	3.16608	
49	3	52	3.16690	
50	3	53	3.16835	
51	2	54	3.16880	
52	7	55	3.16945	
53	3	56	3.17123	
54	2	57	3.17175	
55	2	58	3.17183	
56	2	59	3.17206	
57	3	60	3.17206	
58	2	61	3.17207	
58	183	61		3.17207
				1.08363
				1.34741
				3.08589
				.46132
				19.78500
				1.07606
				4.68186
				2.87715
				5.07811
				2.05219
				5.42653
				5.20968

Tabela 5.7 - C : Com projeçao VMP = 1.4

ii) Condensar todas as restrições violadas :

a) Sem Projeção :

Programa :PCOTOSP.FOR
TOLERANCIA PGP = .00050

EXEMPLO Numero : 5.4

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
			Xo	
1	5	4	2.89874	
2	5	7	2.90608	
3	5	10	2.93095	
4	7	13	2.96269	
5	5	16	2.90267	
6	6	19	3.05497	
7	11	22	3.08136	
8	5	25	3.09325	
9	6	28	3.09705	
10	6	31	3.10379	
11	9	34	3.11417	
12	11	37	3.12592	
13	7	40	3.12089	
14	11	43	3.14332	
15	6	46	3.14766	
16	6	49	3.16203	
17	4	52	3.16337	
18	4	55	3.16517	
19	7	58	3.16609	
20	11	61	3.17004	
21	4	64	3.17049	
22	3	65	3.17106	
23	3	66	3.17140	
23	147	66		3.17140
				1.12989
				1.32894
				3.06498
				.89860
				10.15743
				1.21130
				4.88488
				2.85450
				3.14621
				1.81729
				9.30608
				5.26806

Tabela 5.8 - A : Sem projeçao .

b) Com projeção :

Programa :PCOTOCP.FOR
TOLERANCIA PGP = .00050
VMP = 1.4
EXEMPLO Numero : 5.4

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
			Xo	
1	5	4	2.89874	
2	5	7	2.91292	
3	4	10	2.94786	
4	4	13	2.97352	
5	7	16	3.00285	
6	5	19	3.04361	
7	10	22	3.08699	
8	9	25	3.09697	
9	11	28	3.10940	
10	13	31	3.12014	
11	8	34	3.13118	
12	5	37	3.13502	
13	9	40	3.15637	
14	21	43	3.16274	
15	9	46	3.16485	
16	6	49	3.16649	
17	9	52	3.16762	
18	11	55	3.16989	
19	4	58	3.17014	
20	3	59	3.17137	
21	7	61	3.17215	
22	5	64	3.17294	
22	170	64		3.17294
				1.17246
				1.36169
				3.03586
				.43481
				20.99300
				1.81483
				3.02721
				2.15454
				2.87607
				2.13880
				7.47516
				0.86513

Tabela 5.8 - B : Com projeçao VMP = 1.4

VMP = 3.0
EXEMPLO Numero : 5.4

ITP	ITL	N.CO	SOL. OTIMA PL	SOL. OTIMA PGP
			Xo	
1	5	4	2.89874	
2	5	7	2.90808	
3	4	10	2.94208	
4	3	13	2.97038	
5	7	16	3.03633	
6	8	19	3.06430	
7	6	22	3.07030	
8	11	25	3.09711	
9	4	28	3.10161	
10	15	31	3.11990	
11	9	34	3.13174	
12	11	37	3.13808	
13	9	40	3.14772	
14	15	43	3.15390	
15	14	46	3.16121	
16	5	49	3.16438	
17	4	52	3.16725	
18	10	55	3.16943	
19	16	58	3.17028	
20	9	60	3.17179	
21	3	61	3.17239	
21	173	61		3.17239
				1.16383
				1.38148
				3.03116
				.85319
				10.70424
				.91573
				4.90923
				2.54525
				6.10253
				1.33590
				5.26997
				4.98384

Tabela 5.8 - C : Com projeçao VMP = 3.0

Neste exemplo não houve grandes relações, e como já havíamos previsto anteriormente a projeção não trouxe grandes melhorias. Foram observadas situações onde a projeção na restrição mais violada fazia com que outras se tornassem a mais violada, como ilustrada na figura 5.3 .

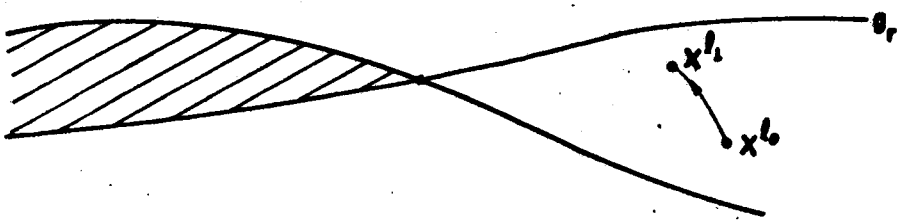


Figura 5.3

Outra observação que podemos fazer é que o VMP agora deve ser bem maior que o exemplo anterior. A projeção começa a surtir efeito quando VMP ≥ 2.05 (valores de VMP variando de 0.01 a partir de 2.0 foram testados), enquanto que no exemplo anterior 1.4 já fazia diferença. Estes valores, naturalmente como já foi observado, devem depender do problema em questão.

5.4 - Experiências numéricas com problemas Signomiais :

Voltaremos agora ao exemplo dado nos capítulos anteriores, e mostraremos os resultados computacionais obtidos, iniciando de vários pontos factíveis. Faremos uma comparação entre o número de vezes que foram acionados os programas (rotinas) do PGP e PL (denotamos por ITP e ITL). Os símbolos utilizados são :

ITG : Iteração do programa PGG,

ITP : Número total de iterações do PGP na iteração ITG,

ITL : Número total de iterações do PL na iteração ITG,

ITTL : Total final de iterações do PL.

Exemplo 5.5 :

Consideremos o mesmo problema de Programação Geométrica do capítulo II.

$$\min \quad x_0$$

$$\text{suj. a} \quad x_1/4 + x_2/2 - x_1^2/16 - x_2^2/16 \leq 1$$

$$x_1^2/(6x_2) + x_2/6 + 7/(3x_2) - x_1/x_2 \leq 1$$

$$1.0 \leq x_1 \leq 5.5$$

$$1.0 \leq x_2 \leq 5.5$$

A) Ponto inicial adotado $x^0 = (4.0,\ 4.5)$

i) Condensar a restrição mais violada :

Ponto inicial x = (4.0,4.5)

TOLER.PGG = .00050

Programa PGGPRCM.FOR

ALFA = .0000

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	3	3.85775 4.85210		
0	2	2	3.87223 4.78570		
0	3	2	3.88394 4.78189		
0	3			3.88394 4.78189	
1	1	3	3.82436 4.82409		
1	1			3.82436 4.82409	
2	1	3	3.82288 4.82287		
2	1			3.82288 4.82287	
3	1	3	3.82288 4.82287		
3	1			3.82288 4.82287	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PGG
3	6	16		3.82288 4.82287

Tab. 5.9 - A : Sem Projeção.

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	3	3.85775 4.85210		
0	2	2	3.87254 4.78427		
0	3	2	3.88423 4.78023		
0	3			3.88423 4.78023	
1	1	3	3.82441 4.82413		
1	1			3.82441 4.82413	
2	1	3	3.82288 4.82287		
2	1			3.82288 4.82287	
3	1	3	3.82288 4.82287		
3	1			3.82288 4.82287	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PG
3	8	16		3.8228 4.8228

Tab. 5.9 - B : Com Projeção.

ii) Condensar todas as restrições violadas :

Nome do programa : PGGPCT.FOR

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	3	3.85775 4.85210		
0	2	3	3.88339 4.78207		
0	2			3.88339 4.78207	
1	1	3	3.82434 4.82407		
1	1			3.82434 4.82407	
2	1	3	3.82288 4.82288		
2	1			3.82288 4.82288	
3	1	3	3.82288 4.82287		
3	1			3.82288 4.82287	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PGG
3	5	15		3.82288 4.82287

Tab. 5.10 - A : Sem Projeção.

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PG
0	1	3	3.85775 4.85210		
0	2	3	3.88379 4.78038		
0	2			3.88379 4.78038	
1	1	3	3.82439 4.82412		
1	1			3.82439 4.82412	
2	1	3	3.82288 4.82287		
2	1			3.82288 4.82287	
3	1	3	3.82288 4.82287		
3	1			3.82288 4.82287	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PG
3	5	15		3.8228 4.8228

Tab. 5.10 - B : Com Projeção.

B) Fase 1 : Esta fase, dada em (3.27), utiliza o mesmo programa PGG mas o teste de parada é **w = 1.**

$$\min \quad w$$

$$\text{suj. a} \quad p_1(x) - q_1(x) \leq w$$

$$p_2(x) - q_2(x) \leq w$$

$$1.0 \leq w \leq 2.5$$

$$1.0 \leq x_1 \leq 3.5$$

$$1.0 \leq x_2 \leq 5.5$$

Neste exemplo, o limite superior de x_1 foi alterado no sentido de excluir a solução local (3.823, 4.823). Assim, se for

possível encontrar $w^* = 1$, significa que temos uma solução factível melhor que a anterior. Esta será usada então, como ponto inicial (Fase II). O limitante superior de w é encontrado substituindo o ponto $\bar{x}_j$ para $j = 1,2$.

FASE 1 (teste de parada w=1.0)

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	2	1.00000		
			1.21453		
			1.00000		
0	2	4	1.01146		
			3.50000		
			1.77708		
0	3	2	1.08150		
			2.15255		
			1.73647		
0	4	2	1.09402		
			2.90759		
			2.30822		
0	5	2	1.09686		
			2.42003		
			2.13261		
0	6	2	1.10280		
			1.65165		
			1.80877		
0	7	2	1.10860		
			1.73075		
			1.89540		
0	8	2	1.11707		
			1.92877		
			1.89236		
0	9	2	1.11753		
			1.78617		
			1.87557		
0	10	2	1.11779		
			1.71056		
			1.86618		
0	10			1.11779	
				1.71056	
				1.86618	
1	1	2	1.00000		
			2.80729		
			1.00000		
1	2	2	1.00000		
			2.06371		
			1.27401		
1	3	2	1.00000		
			2.59053		
			1.43252		
1	4	3	1.00000		
			3.50000		
			1.68817		
1	5	2	1.00000		
			2.92069		
			1.57692		
1	6	2	1.00000		
			2.89387		
			1.59612		

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
1	7	2	1.00000		
			2.87731		
			1.59009		
1	7			1.00000	
				2.87731	
				1.59009	

Tabela 5.11 - A : Fase 1.

Fase 2 : O ponto inicial será agora o ponto obtido na fase 1.

$x^* = (2.87731, 1.59009)$

i) Condensar a restrição mais violada :

Programa = PGGPRCM.FOR

Ponto Inicial Xo=(2.87731,1.59009)

TOLER.PGG = .00500

ALFA = .0000

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	1	1.00000		
			1.00000		
0	2	3	1.51618		
			1.63655		
0	3	2	1.72966		
			1.73059		
0	4	2	1.75550		
			1.59613		
0	5	2	1.77991		
			1.60930		
0	6	2	1.78103		
			1.60581		
0	6			1.78103	
				1.60581	
1	1	3	1.15914		
			2.12261		
1	2	2	1.38907		
			2.06197		
1	3	2	1.39285		
			1.95635		
1	4	2	1.42653		
			1.95999		
1	5	2	1.42695		
			1.95776		
1	5			1.42695	
				1.95776	
2	1	3	1.17951		
			2.16509		
2	2	2	1.22919		
			2.14628		
2	3	2	1.23563		
			2.11799		
2	4	2	1.23816		
			2.11762		
2	4			1.23816	
				2.11762	
3	1	3	1.17750		
			2.17623		
3	2	2	1.18178		
			2.17436		
3	2			1.18178	
				2.17436	
4	1	3	1.17713		
			2.17712		
4	1			1.17713	
				2.17712	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PGG
4	18	40		1.17713
				2.17712

Tab. 5.11 - B: (SP)

ALFA = 1.0000

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PG
0	1	1	1.00000		
			1.00000		
0	2	3	1.67174		
			1.70576		
0	3	2	1.74078		
			1.58484		
0	4	2	1.78124		
			1.60598		
0	4			1.78124	
				1.60598	
1	1	3	1.15910		
			2.12257		
1	2	2	1.40896		
			2.05727		
1	3	2	1.41452		
			1.95687		
1	4	2	1.42741		
			1.95781		
1	4			1.42741	
				1.95781	
2	1	3	1.17951		
			2.16507		
2	2	2	1.22969		
			2.14608		
2	3	2	1.23591		
			2.11774		
2	3			1.23591	
				2.11774	
3	1	3	1.17747		
			2.17626		
3	2	2	1.18158		
			2.17446		
3	2			1.18158	
				2.17446	
4	1	3	1.17713		
			2.17712		
4	1			1.17713	
				2.17712	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PG
4	14	32		1.1771
				2.1771

Tab. 5.11 - C :(CP)

ii) Condensar todas as restrições violadas :

Programa = PGGPCT.FOR

ALFA = 0.00

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	1	1.00000		
			1.00000		
0	2	3	1.51618		
			1.63655		
0	3	3	1.75420		
			1.60259		
0	4	3	1.78100		
			1.60586		
0	4			1.78100	
				1.60586	
1	1	3	1.15914		
			2.12261		
1	2	3	1.39205		
			1.97749		
1	3	3	1.42661		
			1.95821		
1	3			1.42661	
				1.95821	
2	1	3	1.17951		
			2.16513		
2	2	3	1.23506		
			2.11998		
2	3	2	1.23745		
			2.11972		
2	3			1.23745	
				2.11972	
3	1	3	1.17749		
			2.17628		
3	2	2	1.18159		
			2.17448		
3	2			1.18159	
				2.17448	
4	1	3	1.17713		
			2.17712		
4	1			1.17713	
				2.17712	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PGG
4	13	35		1.17713
				2.17712

Tab. 5.12 - A :(SP)

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	1	1.00000		
			1.00000		
0	2	3	1.67174		
			1.70576		
0	3	3	1.77503		
			1.60275		
0	4	2	1.78128		
			1.60600		
0	4			1.78128	
				1.60600	
1	1	3	1.15909		
			2.12256		
1	2	3	1.41395		
			1.96691		
1	3	3	1.42735		
			1.95780		
1	3			1.42735	
				1.95780	
2	1	3	1.17951		
			2.16507		
2	2	3	1.23564		
			2.11889		
2	2			1.23564	
				2.11889	
3	1	3	1.17747		
			2.17628		
3	2	2	1.18149		
			2.17452		
3	2			1.18149	
				2.17452	
4	1	3	1.17713		
			2.17712		
4	1			1.17713	
				2.17712	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PG
4	12	32		1.1771
				2.1771

Tab. 5.12 - B : (CP)

Assim, a Fase 1 pode ser usada para obtenção de um ponto factível, como também, conforme o exemplo acima, para "escapar" do ótimo local.

C) Ponto inicial : $x^0 = (4.8, 3.9)$

i) Condensar a restrição mais violada :

PONTO INICIAL NO PGG
X(1) = 4.6000000
X(2) = 3.8000000

TOLER.PGG = .00050

Nome do programa : PGGPRCM.FOR

ALFA = .0000

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	2	3.52001		
			1.00000		
0	2	2	3.67974		
			1.54433		
0	3	2	3.83269		
			2.30230		
0	4	2	3.91274		
			2.81739		
0	5	2	3.95390		
			3.12146		
0	6	2	3.95436		
			2.98101		
0	6			3.95436	
				2.98101	
1	1	2	2.31901		
			1.00000		
1	2	2	2.49839		
			1.18632		
1	3	2	2.75797		
			1.17143		
1	4	2	2.78115		
			1.22493		
1	5	2	2.78965		
			1.22604		
1	5			2.78965	
				1.22604	
2	1	2	1.00000		
			1.17700		
2	2	4	1.62625		
			1.64346		
2	3	2	1.69316		
			1.55805		
2	4	2	1.81092		
			1.86074		
2	5	2	1.91459		
			1.83754		
2	6	2	1.93688		
			1.84878		
2	6			1.93688	
				1.84878	
3	1	3	1.12094		
			2.08149		
3	2	2	1.41198		
			2.03705		
3	3	2	1.41558		
			1.91620		
3	4	2	1.46108		
			1.92549		
3	5	2	1.46168		
			1.92243		
3	5			1.46168	
				1.92243	
4	1	3	1.17939		
			2.16161		
4	2	2	1.24222		
			2.13658		
4	3	2	1.24929		
			2.10404		
4	4	2	1.25305		
			2.10361		
4	4			1.25305	
				2.10361	
5	1	3	1.17767		
			2.17577		
5	2	2	1.18406		
			2.17300		
5	3	2	1.18568		
			2.16854		
5	3			1.18568	
				2.16854	
6	1	3	1.17713		
			2.17711		
6	1			1.17713	
				2.17711	
7	1	3	1.17712		
			2.17713		
7	1			1.17712	
				2.17713	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PGG
7	31	89		1.17712
				2.17713

Tab. 5.13 - A :(SP)

ALFA = 1.0000

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	2	3.52001		
			1.00000		
0	2	2	3.66634		
			1.49012		
0	3	2	3.80758		
			2.15764		
0	4	2	3.88761		
			2.64515		
0	5	2	3.93686		
			2.99220		
0	6	2	3.95034		
			2.80684		
0	6			3.95034	
				2.80684	
1	1	2	2.17352		
			1.00000		
1	2	2	2.36453		
			1.18436		
1	3	2	2.48298		
			1.16921		
1	4	2	2.48940		
			1.17894		
1	4			2.48940	
				1.17894	
2	1	2	1.00000		
			1.74808		
2	2	4	1.83865		
			1.98686		
2	3	2	1.85392		
			1.71202		
2	4	2	1.86768		
			1.71610		
2	4			1.86768	
				1.71610	
3	1	3	1.13705		
			2.09663		
3	2	2	1.40046		
			2.05017		
3	3	2	1.40750		
			1.95291		
3	4	2	1.41942		
			1.95451		
3	4			1.41942	
				1.95451	
4	1	3	1.17949		
			2.16536		
4	2	2	1.22854		
			2.14662		
4	3	2	1.23467		
			2.11896		
4	3			1.23467	
				2.11896	
5	1	3	1.17745		
			2.17629		
5	2	2	1.18140		
			2.17456		
5	2			1.18140	
				2.17456	
6	1	3	1.17713		
			2.17712		
6	1			1.17713	
				2.17712	
7	1	3	1.17713		
			2.17712		
7	1			1.17713	
				2.17712	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PGG
7	25	57		1.17713
				2.17712

Tab. 5.13 - B : (CP)

ii) Condensar todas restrições violadas :

Nome do programa : PGGPCT.FOR

ALFA = 0.0000

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	2	3.52001		
			1.00000		
0	2	3	3.75068		
			1.86178		
0	3	2	3.86991		
			2.52950		
0	4	2	3.93189		
			2.95540		
0	5	2	3.94984		
			2.76782		
0	6	2	3.95327		
			2.86887		
0	6			3.95327	
				2.86887	
1	1	2	2.22305		
			1.00000		
1	2	3	2.56449		
			1.15337		
1	3	2	2.58270		
			1.18860		
1	3			2.58270	
				1.18860	
2	1	2	1.00000		
			1.57101		
2	2	4	1.69435		
			1.68771		
2	3	3	1.87792		
			1.69434		
2	4	3	1.89293		
			1.69586		
2	4			1.89293	
				1.69586	
3	1	3	1.13128		
			2.09069		
3	2	3	1.39380		
			1.96399		
3	3	3	1.43235		
			1.94404		
3	3			1.43235	
				1.94404	
4	1	3	1.17950		
			2.16423		
4	2	3	1.23860		
			2.11653		
4	3	2	1.24126		
			2.11626		
4	3			1.24126	
				2.11626	
5	1	3	1.17754		
			2.17618		
5	2	3	1.18335		
			2.17091		
5	2			1.18335	
				2.17091	
6	1	3	1.17713		
			2.17712		
6	1			1.17713	
				2.17712	
7	1	3	1.17712		
			2.17712		
7	1			1.17712	
				2.17712	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.PGG
7	23	61		1.17712
				2.17712

ALFA = 1.0000

ITG	ITP	ITL	SOL.FIN.PL	SOL.FIN.PGP	SOL.FIN.PGG
0	1	2	3.52001		
			1.00000		
0	2	3	3.78394		
			2.03000		
0	3	2	3.87274		
			2.54767		
0	4	2	3.92799		
			2.92679		
0	5	2	3.95134		
			2.72355		
0	6	2	3.95336		
			2.82067		
0	6			3.95336	
				2.82067	
1	1	2	2.18549		
			1.00000		
1	2	3	2.48962		
			1.17083		
1	3	3	2.50978		
			1.18032		
1	3			2.50978	
				1.18032	
2	1	2	1.00000		
			1.71021		
2	2	4	1.85766		
			1.74160		
2	3	3	1.87332		
			1.71154		
2	3			1.87332	
				1.71154	
3	1	3	1.13581		
			2.09533		
3	2	3	1.40972		
			1.96115		
3	3	3	1.42244		
			1.95207		
3	3			1.42244	
				1.95207	
4	1	3	1.17950		
			2.16510		
4	2	3	1.23549		
			2.11902		
4	2			1.23549	
				2.11902	
5	1	3	1.17746		
			2.17629		
5	2	2	1.18147		
			2.17453		
5	2			1.18147	
				2.17453	
6	1	3	1.17713		
			2.17712		
6	1			1.17713	
				2.17712	
7	1	3	1.17712		
			2.17712		
7	1			1.17712	
				2.17712	

ITG	ITTP	ITTL	*** RESULTADOS FINAIS ***	SOL.
7	21	56		1.17
				2.17

Tab. 5.14 - A : (SP) 5.14 - B : (CP)

Note que a projeção diminui um pouco o número de iterações do PGP e PL nos casos (i) e (ii) deste exemplo. Isto decorre das observações já feitas sobre a influência da projeção no PGP, isto é, se existem grandes violações, então a projeção tem um bom desempenho, caso contrário, pode até piorar a convergência.

CAPÍTULO VI

Conclusões e Perspectivas Futuras :

Neste trabalho revimos a teoria da Programação Geométrica e exploramos uma classe de métodos baseados no conceito de condensação (cortes). A idéia de Censor-Lent (projeção), para determinar soluções factíveis num convexo, foi usada para tentar acelerar a convergência dos métodos. A formula de "projeção" usada, explora a particularidade do problema em sua forma original. Mostramos, entretanto, que tal formula é equivalente à de Censor-Lent quando aplicada num problema transformado.

Os métodos foram implementados no computador, usando a linguagem FORTRAN 77, adotando-se as estratégias recomendadas na literatura : condensação apenas na restrição mais violada e em todas as restrições violadas. Para cada uma destas estratégias introduzimos também a projeção (capítulo IV).

O uso da projeção, ao contrário do que imaginávamos no início, nem sempre obteve sucesso. Organizamos os exemplos numéricos, neste trabalho, no sentido de induzir a razão deste fato. No exemplo 5.1 o efeito de se projetar foi desprezível. Neste exemplo as violações eram de pequenas proporções. Assim introduzimos um valor mínimo para projeção (VMP), fazendo com que a projeção somente fosse usada caso a violação fosse superior a VMP. O exemplo 5.2 mostra um caso com grandes violações e o efeito benéfico da projeção.

Outra observação, que coincide com a de [12], foi o melhor desempenho para a estratégia condensando todas as restrições violadas. Entretanto pudemos observar que esta

estratégia é muito superior quando estamos próximos da solução. As primeiras iterações usando a mais violada ou em todas violadas tiveram desempenhos (a medida de desempenho usada aqui foi $\|x^i - x^*\|_\infty$) semelhantes. Do ponto de vista computacional, a estratégia (ii) todas restrições violadas foi inferior, pelo menos inicialmente, pois condensou mais restrições e fez mais iterações nos programas lineares.

Combinando estas observações, feitas nos dois últimos parágrafos, parece-nos que uma estratégia ideal se delineia : "usar projeção até que o ponto satisfaça o critério do valor mínimo para projeção (VMP); e a partir daí, condensar em todas as restrições violadas ". Não apresentamos neste trabalho esta estratégia, a qual deixaremos para o passo seguinte quando pretendemos fazer um estudo mais profundo das observações aqui descritas.

A "faixa" para projeção, ditada por VMP, deve depender das características do problema em questão. Neste trabalho apenas mostramos a importância de VMP sem explorar qual o valor ideal para esta variável.

Apresentamos no capítulo III uma nova formulação para a fase 1, que apresenta uma vantagem sobre a formulação dada por Avriel e outros [2], uma vez que usa apenas uma variável artificial. O exemplo 5.5 - B, mostra um uso alternativo para a fase 1, além daquele de procurar um ponto factível para o PGG; ou seja, o de tentar obter o ótimo global, uma vez que a Programação Geométrica Generalizada admite ótimos locais, ao contrário da Programação Geométrica Posinomial.

APENDICE A

Método das Projeções Cíclicas dos Subgradientes (PCS) :

1 - Introdução :

O método das projeções cíclicas do subgradiente foi proposto por Censor & Lent [8], no intuito de resolver problemas de factibilidade convexa, tornando-o, assim, uma excelente ferramenta, graças às várias características existentes.

Determinar x tal que $g_i(x) \leq 0$, $i \in M = \{ 1,2,..,m \}$, onde g_i é função convexa.

Este método é iterativo e projeta nas restrições violadas, uma de cada vez, de ordem cíclica, não havendo assim nenhum esforço computacional para selecionar a função a ser usada na próxima iteração.

2 - Método PCS :

Inicialização : $x_o \in Q$ é arbitrário.

Definição : Dado um conjunto $Q \subseteq \mathbb{R}^n$ e um ponto $x \in \mathbb{R}^n$, $P_Q(x)$ denotará a Projeção Ortogonal de x sobre Q, isto é, o ponto tal que :

$$\| x - P_Q(x) \| = \inf_{z \in Q} \| x - z \|$$

Onde $\|.\|$ é a norma Euclidiana em $\mathbb{R}^n$. Se Q é fechado e não vazio, então $P_Q(x)$ sempre existe e se Q é convexo, então $P_Q(x)$ é unicamente determinado por x.

Passo único :

$$x^{i+1} = P_{\Omega}(\tilde{x}^{i+1}), \quad \text{com :}$$

$$\tilde{x}^{i+1} = x^{i} - \alpha_{i} \frac{g_{ki}^{+}(x^{i}) \; t_{k}^{i}}{\| t_{k}^{i} \|^{2}}$$

onde $ki = i \pmod n + 1$, $g_{k}^{+}(x) = \max \{ 0, g_{k}(x) \}$ e

$t_{i} \in \partial g_{ki}^{+}(x^{i}) = \{$ conjunto dos subgradientes de g_{ki}^{+} em $x^{i} \}$.

$\{\alpha_{i}\}_{i=o}^{\infty}$ é uma sequência de parâmetros de relaxação confinada ao intervalo $\varepsilon_{1} \leq \alpha_{i} \leq 2 - \varepsilon_{2}$ com $\varepsilon_{1}, \varepsilon_{2} > 0$, pequenos.

Teorema : Se

(i) As $g_{k}(x)$ são funções contínuas e convexas em $\mathbb{R}^{n}$ para todo $k \in M$;

(ii) $\exists$ um ponto factível;

(iii) Para algum $\hat{x}$ existe uma constante $T = T(\hat{x})$ tal que $\| t \| \leq T$ factível para todo $t \in \partial g_{k}^{+}(x)$; para todo $k \in M$ e para todo x com $\|x - \hat{x}\| \leq \| x^{o} - \hat{x} \|$.

Então, a sequência $\{x^{i}\}$ gerada pelo método PCS converge para uma solução do problema de factibilidade convexa, isto é :

$$x^{i} \longrightarrow x^{*} \in S$$

onde S é o conjunto das soluções factíveis $\in \mathbb{R}^{n}$.

A prova consiste nos seguintes passos :

1) $\{ x^{i} \}$ é uma sequência Fejer-Monotônica$^{(*)}$ com respeito a S,

2) $\lim g_{ki}^{+}(x^{i}) = 0$,

3) $\| x^{i+1} - x^{i} \| \xrightarrow[i \to \infty]{} 0$,

4) Para todo $k \in M$ fixo, $\lim_{i \to \infty} g_k^+(x^i) = 0$,

5) $\lim_{i \to \infty} x^i = x^* \in S$,

(*) Uma sequência $\{x^i\}_{i=0}^{\infty}$ é chamada Fejer-Monotônica com respeito a um conjunto fechado $S \subseteq \mathbb{R}^n$ se :

$$\| x^{i+1} - x \| \leq \| {}^i x - x \|; \ \forall\, i \geq 0, \quad \forall\, x \in S.$$

APENDICE B

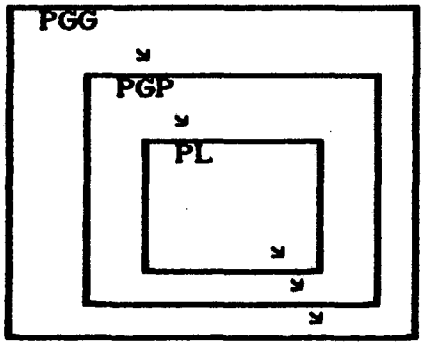
PROGRAMAS E ROTINAS

1. Introdução :

Os métodos vistos nos capítulos III e IV desse trabalho e seus respectivos algoritmos, foram interligados em um só programa. Estes programas foram desenvolvidos na linguagem FORTRAN 77 e foram executados em micro computadores PCxt (16 bits).

Os programas estão assim estruturados :

PGG



Os métodos de cortes e projeção fazem parte do programa PGP, por isso foram feitos vários testes utilizando apenas o programa PGP, com diferentes estratégias (visto no capítulo V).

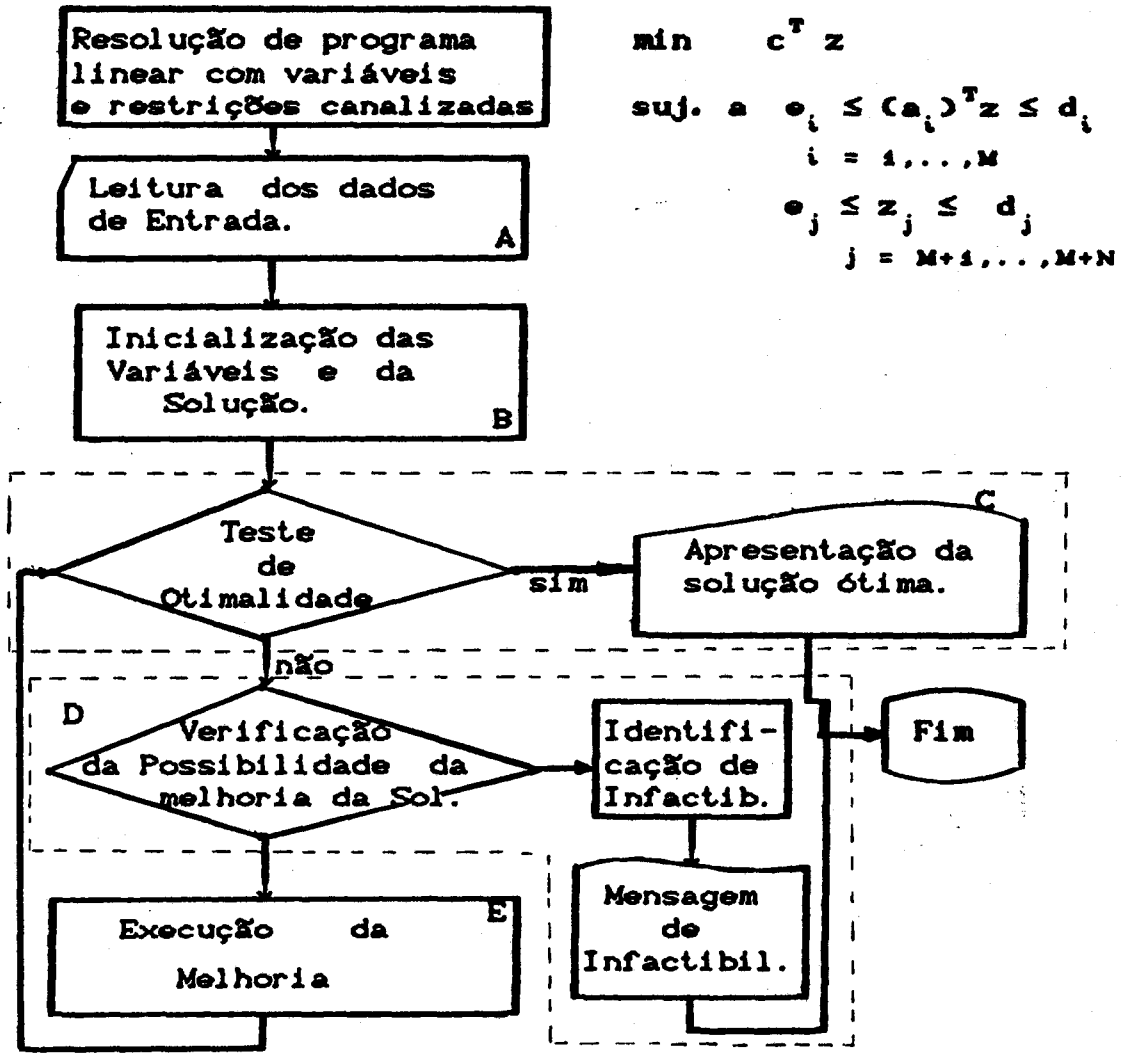
Achamos conveniente, separarmos os 3 programas, para um estudo mais detalhado. Os programas finais, foram colocados em apéndice como garantia de não se perderem.

2 - Programa que resolve um problema linear (PL) :

Este programa utiliza o método Dual Simplex. A entrada de dados é feita automaticamente, para o nosso caso, mas caso exista um problema linear a ser executado, colocaremos a subrotina que lê os dados iniciais.

Faremos a seguir um esquema do funcionamento do programa principal do PL .

Fluxo de informação :



$$\min \quad c^T z$$

$$\text{suj. a} \quad e_i \leq (a_i)^T z \leq d_i \qquad i = 1,\ldots,M$$

$$e_j \leq z_j \leq d_j \qquad j = M+1,\ldots,M+N$$

Fig. A - Fluxo de Informação de um PL.

O programa principal controla 5 subrotinas que completam o programa PL. Essas subrotinas serão mostradas mais a seguir, e cada subrotina está representada por uma letra de (A,B,C,D,E),

ilustrada na figura A.

Fluxo de Informação das subrotinas utilizadas :

A) Subrotina LEITUR :

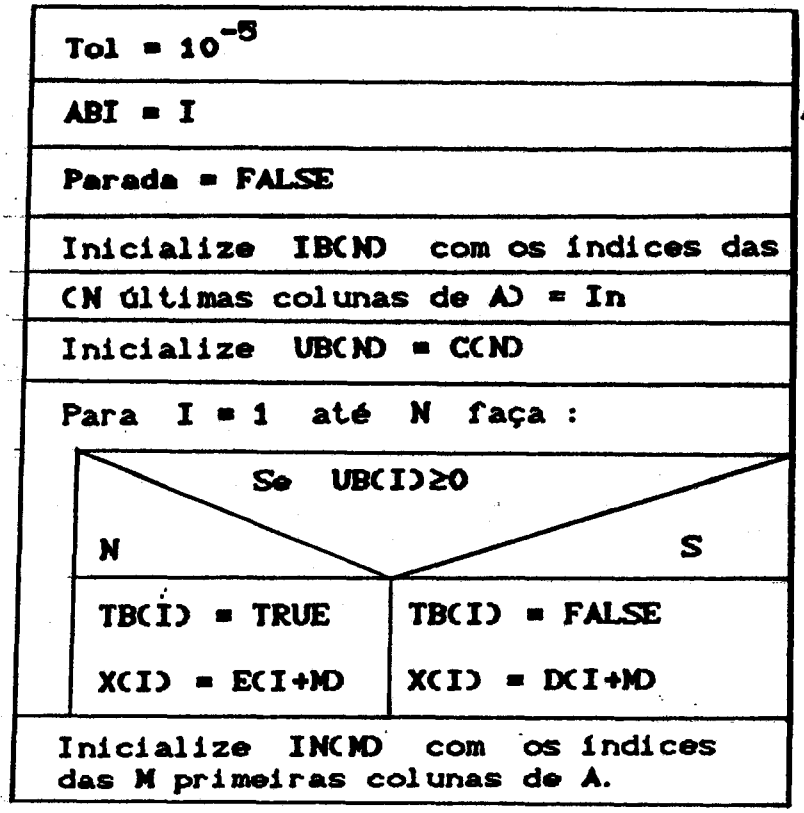
Esta subrotina só será utilizada em casos de problemas específicos de programação linear. No nosso caso, essa subrotina não foi utilizada, pois a leitura pode ser entendida como uma transferência de variáveis diretamente do problema original.

Leia M,N
Leia C(N)
Leia A por colunas
Leia D(M+N)
Leia E(M+N)

onde M é o número de restrições e N é o número de variáveis. D(M+N) e E(M+N) são os vetores dos limitantes superiores e inferiores, respectivamente, das restrições do problema.

B) Subrotina INICIA :

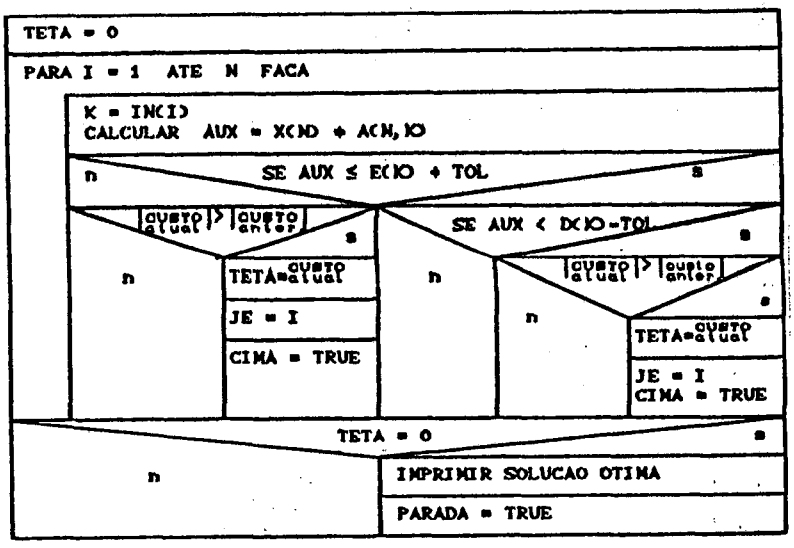
Neste procedimento definimos os valores iniciais das variáveis e limites de tolerância do problema.



onde IB(N) são os índices básicos e UB(N) são os índices não básicos.

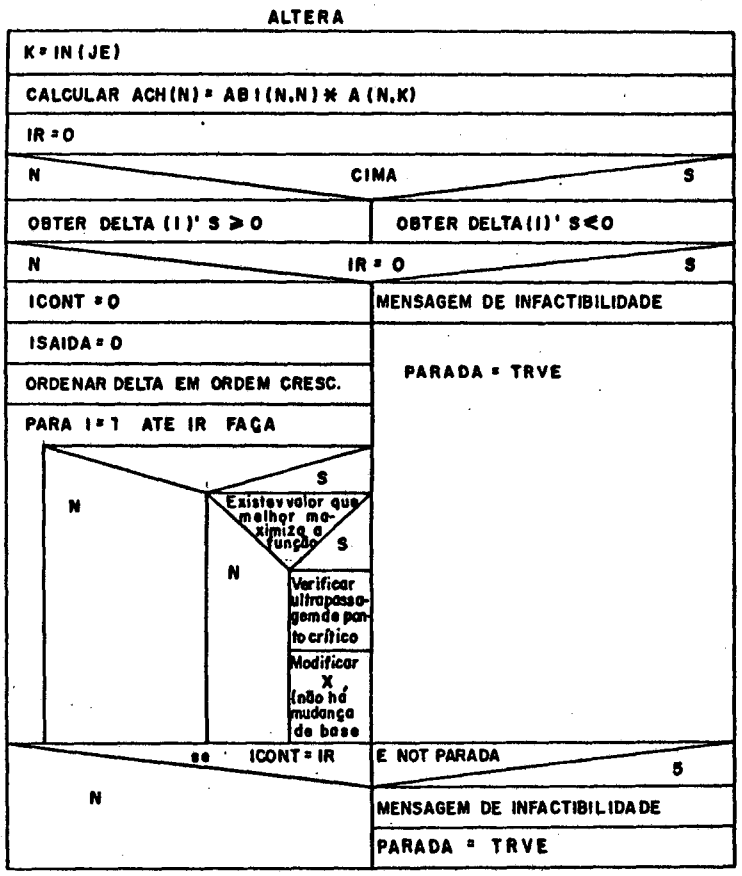
C) Subrotina TESOTI :

Verifica se a Solução é ótima. Se for, Imprime a solução ótima e a parada torna verdadeira (TRUE). Caso contrário, continua parada igual a falso(FALSE).



D) Subrotina Altera

Promove a alteração das restrições que entram e saem da base. Neste procedimento é feita a identificação de infactibilidade, caso seja infactível será emitido uma mensagem.



E) Subrotina ATUALI :

Faz a atualização das variáveis que entraram na base, dos índices básicos e não-básicos.

TB(K2) = CIMA
INC(JE) = IB(K2)
ATUALIZAR UB(N)
ATUALIZAR ABI(N,N)
ATUALIZAR X(N) (Com mudança de base)

Para utilizarmos esses procedimentos no PGG (e PGP) foi necessário acrescentarmos uma variável lógica e fazermos algumas adaptações como :

variável lógica : FACTVL → {Se for factivel → TRUE / Caso contrário → FALSE

Modificamos as mensagens da solução ótima, da entrada de dados, e não foi necessário usarmos z como variável (aproveitamos a própria variável x), etc.

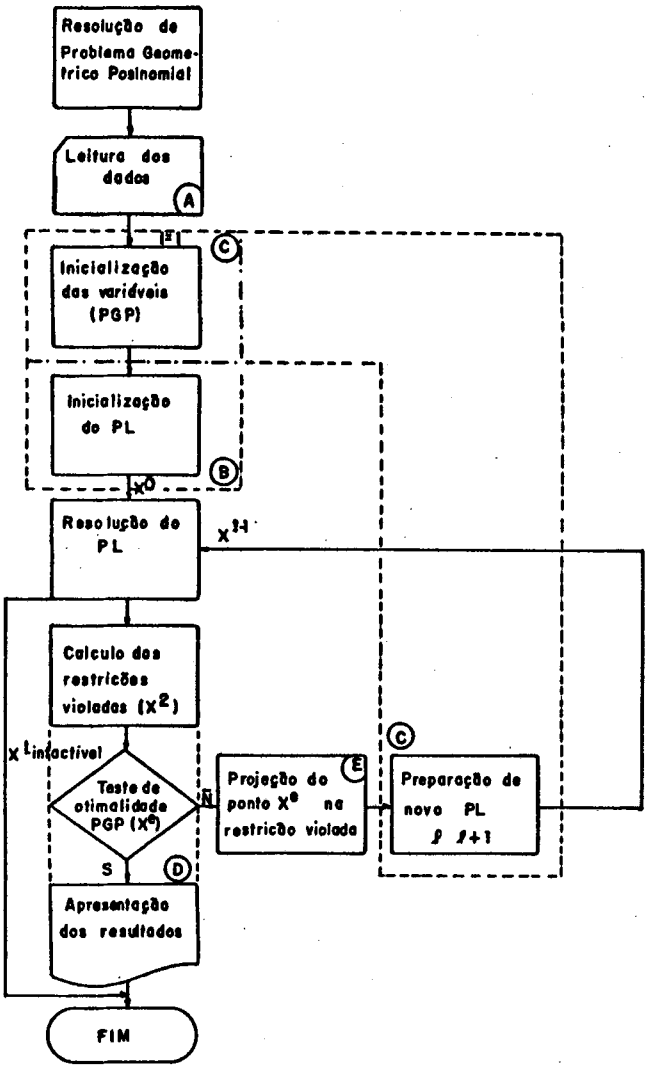
O programa PL, passará a ser representado agora por um simples procedimento de execução para o programa PGP, que veremos a seguir.

3 - Programa que resolve um PGP :

O programa PGP.FOR é o bloco mais importante de toda nossa programação, a saber, é nele que aplicaremos todas as nossas estratégias de execução.

A princípio, faremos um esquema geral do funcionamento do programa, onde apresentaremos as idéias dessas estratégias. A seguir daremos as rotinas pertencentes a esta fase. Foram feitos quatro programas diferentes nesta fase, onde comentaremos sobre eles no final desta seção.

Fluxo de Informação (PGP) :



$$\min \quad x_0$$

$$\text{suj.a} \quad g_k(x) \leq 1 \quad k=0,..,K$$

$$0 < \underline{x}_j \leq x_j \leq \bar{x}_j \quad j=0,..,N$$

Note que, existem duas maneiras de encerrarmos esse procedimento :

i) Ocorrência de infactibilidade no programa PL e

ii) Solução ótima no PGP (ponto factível em PGP).

As letras colocadas no diagrama, mostram as fases de utilização das subrotinas pertencentes a este programa, as quais apresentaremos a seguir.

A) Rotina LERPGP :

A função desta rotina é ler os dados de entrada do problema PGP a ser resolvido.

LER KZAO, NZAO
LER O NÚMERO DE MONÔMIOS (Tk) EM CADA RESTRIÇÃO $g_k(x)$
LER OS COEFICIENTES θ_{ki} POR LINHA (g_k)
LER OS EXPOENTES φ_{kij} POR LINHA
LER OS LIMITANTES INFERIORES ($\underline{x}_j$)
LER OS LIMITANTES SUPERIORES ($\bar{x}_j$)

Quando estivermos resolvendo um problema de PGG, esta rotina não será necessária. Os dados para o PGP serão automaticamente gerados pelo programa PGG.

B) Rotinas INIPGP :

Esta rotina complementa a leitura dos dados de entrada, inicializa as variáveis, lineariza o PGP e inicializa o PL.

INTER=0 E TOL = 10^{-5}
N = NZAO + 1
M = KZAO + 1
K = KZAO + 1
CCND = (1;0;..;0)
CONDENSAR E LINEARIZAR AS RESTRIÇÕES PGP(CNDLIN)
CONVERTER OS LIMITANTES INFERIORES E SUPERIORES PARA O PL.
INICIALIZAR O PL
OTIMO = FALSE
FACTVL = TRUE

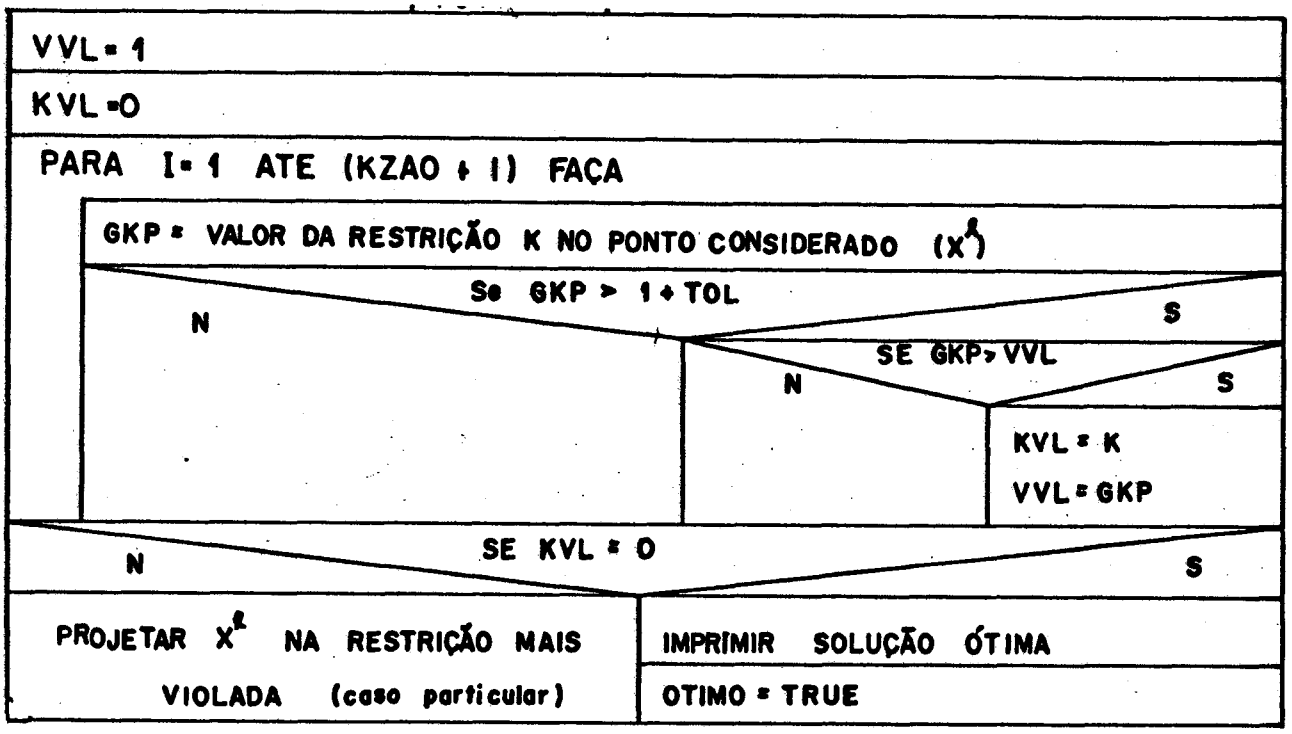
C) Rotina CNDLIN :

Esta rotina condensa e lineariza uma restrição g(x) qualquer.

GK = 0 (Variável que representará o valor da restrição $g_k(x)$
CALCULAR O VALOR DA RESTRICAO NO PONTO CONDENSADO
CALCULAR OS EPSP(I) (Onde EPSPi representará a variável δi)
CALCULAR A COLUNA KX DE ACN,MD (Para o PL)
CALCULAR OS LIMITANTES SUPERIORES E INFERIORES NO PL.

D) Rotina TOTPGP :

Calcula o valor de todas restrições $g_k(x)$ no ponto x^l e faz o teste de otimalidade. Se for ótimo ela emite uma relação do resultado.



Esta rotina foi planejada para resolver a restrição mais violada, como um caso particular.

E) Rotina PROJX :

A função desta rotina é de executar os passos descritos

no capítulo IV. Esta rotina é considerada opcional em nossas execuções, portanto será utilizada uma variável denominada por α (alfa), que assumirá os valores de $0 \leq \alpha \leq 1.0$. Com isso, os problemas poderão ser executados com ou sem projeção.

PROJET = TRUE
CALCULAR $\hat{a}$
CALCULAR $\| \hat{a} \|^2$
CALCULAR x_j^{li} (Ponto x projetado)
PROJET = FALSE

Foram criados quatro tipos diferentes de programas, onde cada tipo representa uma estratégia de como resolvemos um problema de PGP. O aspecto que consideramos aqui, foi em relação à escolha da restrição a ser condensada e projetada. Esses tipos são:

i) Condensar a restrição mais violada, sem projeção;

ii)Projetar na restrição mais violada e condensar no ponto encontrado;

iii)Condensar todas restrições violadas sem projeção;

iv)Projetar e condensar em todas restrições violadas.

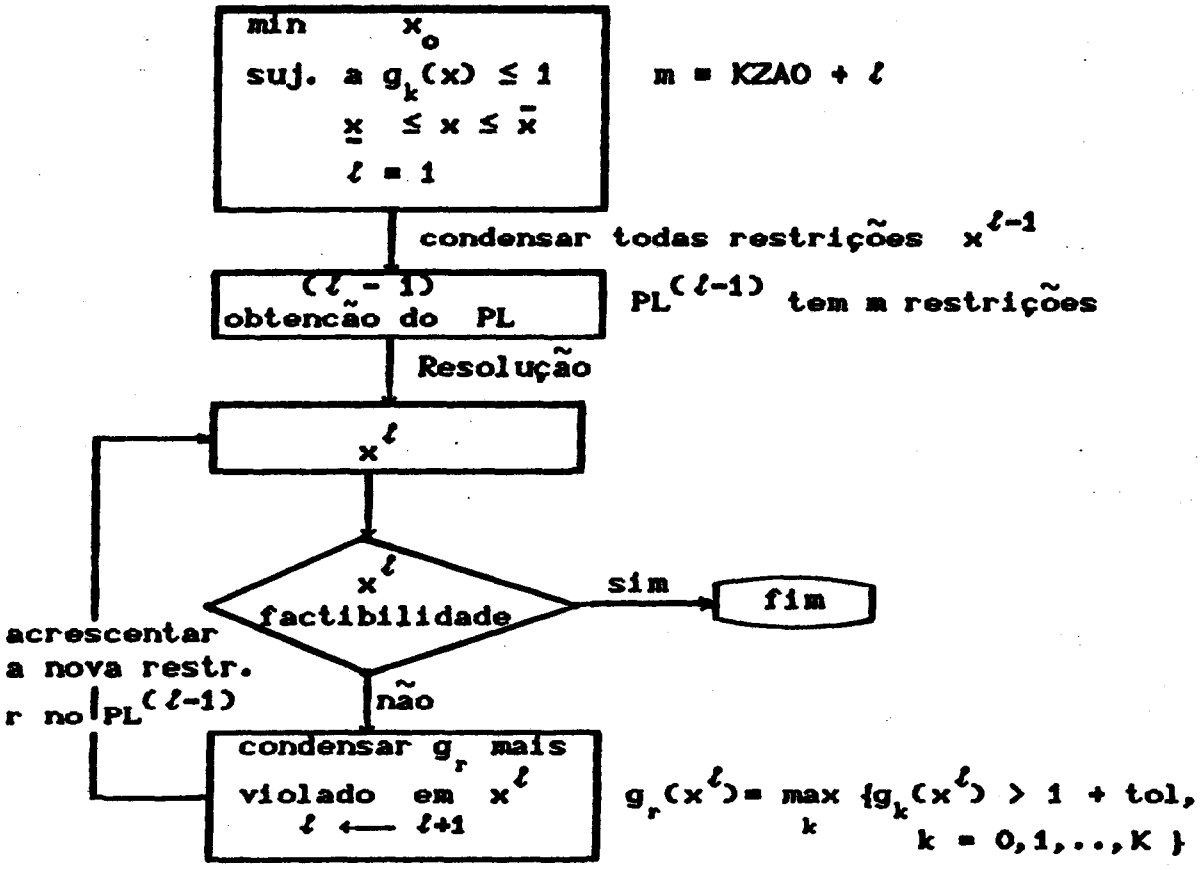
Nos casos (i) e (iii) significa que a rotina "projx" não fará parte do programa, e nos casos (ii) e (iv), a rotina "projx" será uma opção de executar ou não a projeção, fazendo assim ALFA=(1 ou 0) respectivamente (o valor de α pode ser assumido $0 \leq \alpha \leq 1.0$, mas para experiências computacionais foi usado valor 0 ou 1), para $\alpha = 0$ a rotina será executada sem efeito algum,

isto é, $x_j^{l1} = x_j^{lo}$, implicando assim em cálculos desnecessários. Por isso fizemos os programas sem a projeção.

i) Programa que condensa a restrição mais violada, sem a projeção.

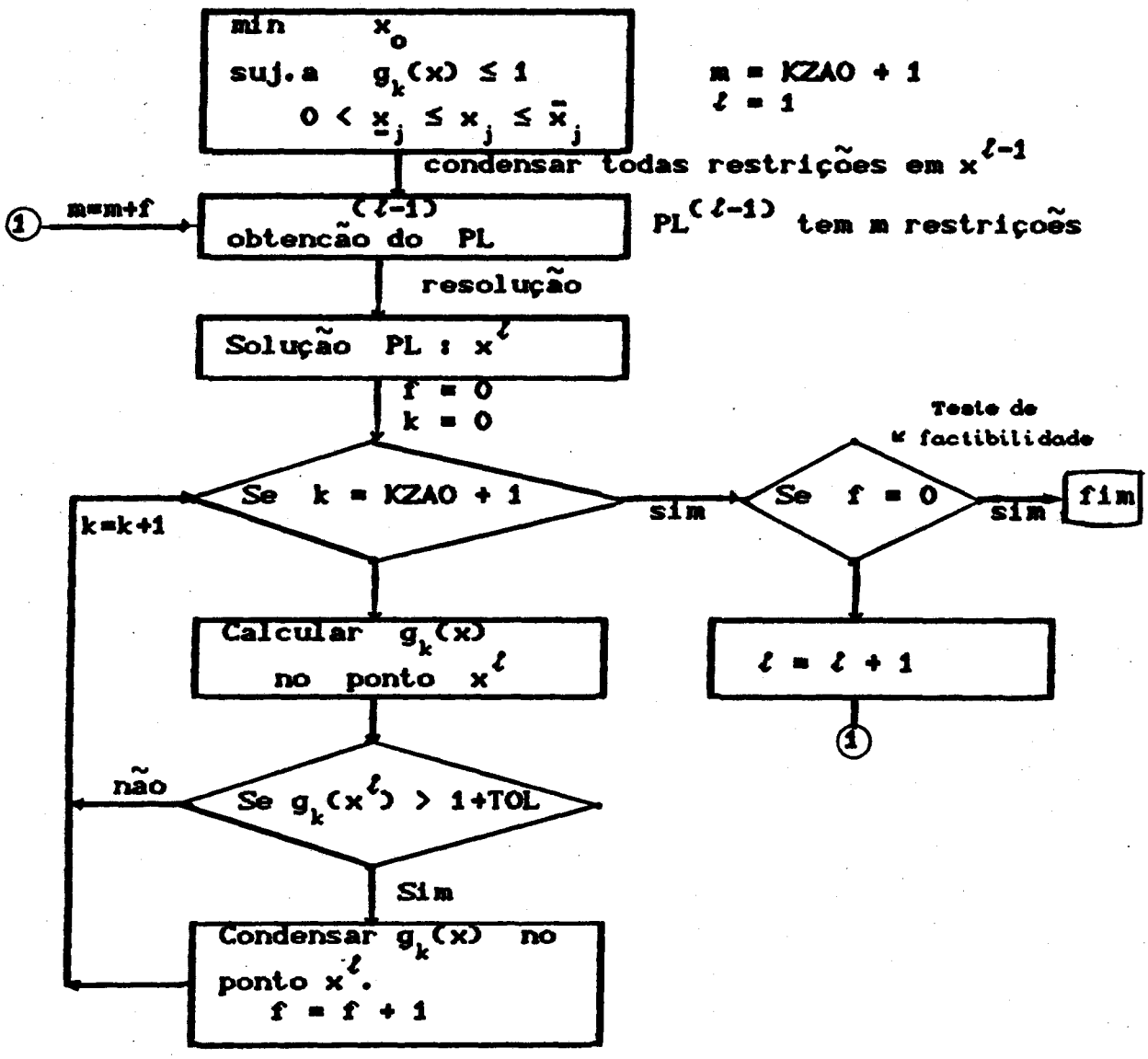
É o mesmo algoritmo dado no método de corte por condensação, onde a restrição a ser condensada será a mais violada.

Fluxo de Informação :



ii) Programa que projeta na restrição mais violada e depois condensa a restrição mais violada no ponto obtido.

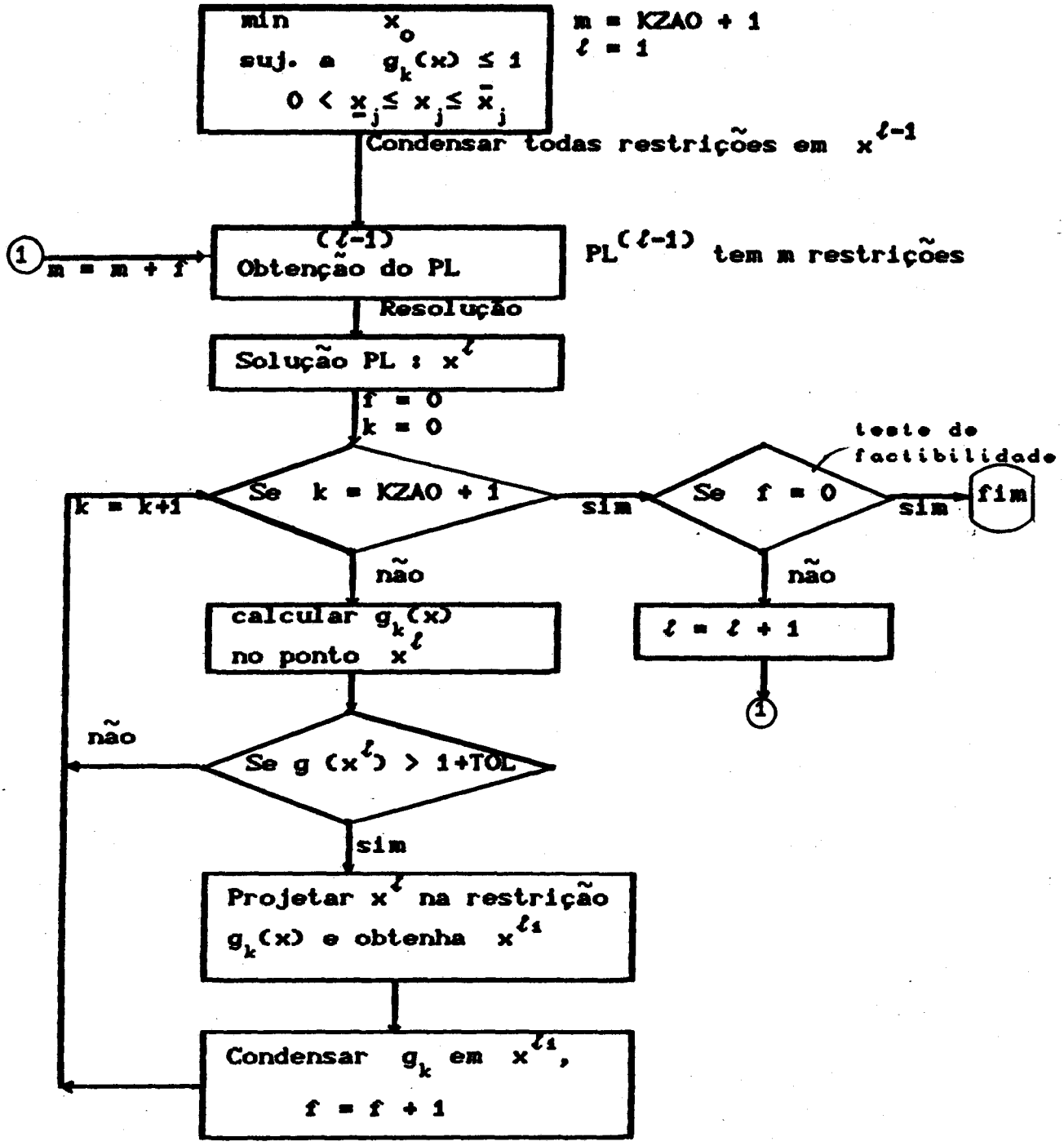
Seja $x^{\ell o}$ a solução do PL na iteração ℓ do PGP, encontramos $x^{\ell 1}$ através da projeção de $x^{\ell o}$ na restrição mais violada $g_r(x)$. Condensamos a seguir a restrição $g_r(x)$ no ponto $x^{\ell 1}$, obtendo uma nova restrição que será acrescentada ao PL, e assim sucessivamente até encontrarmos uma solução ótima a PGP.



iv) Projetar e condensar em todas restrições violadas :

Este procedimento é semelhante ao anterior acrescido da projeção antes da condensação. Como já vimos, a variável ALFA pode assumir valores de 0 até 1.0, mas para efeito de

experiências feitas no capítulo V, assumimos valores 1.0 ou 0 para o caso de projeção e não projeção respectivamente (no caso α=0, a rotina PROJX não será usada).

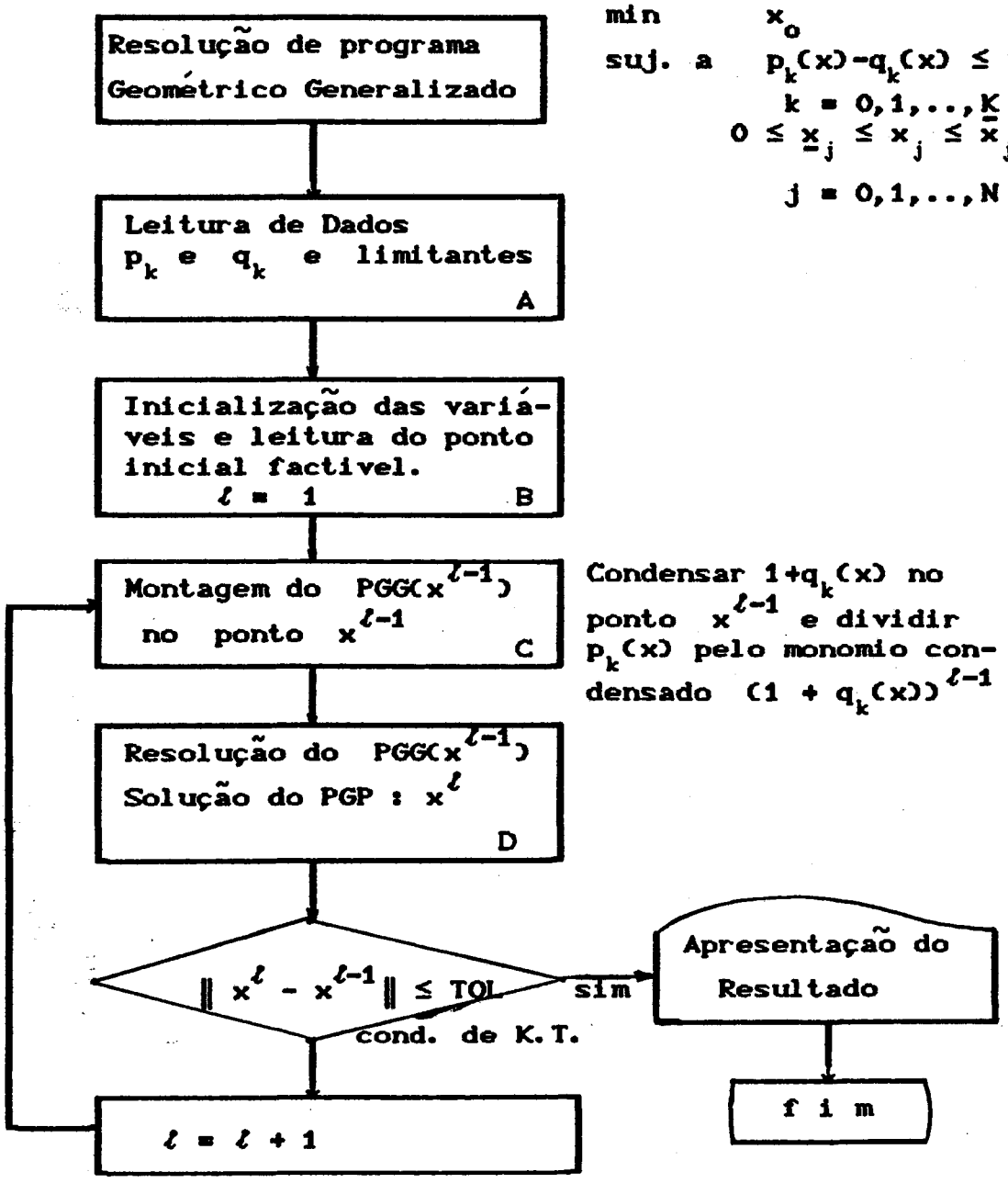


Esses quatro casos foram comparados com alguns exemplos e os resultados obtidos encontram-se no capítulo V.

4. Programa que resolve um PGG (PGG.FOR)

Este programa é usado apenas para os casos em que o problema é do tipo Signomial ou Generalizado. Ele envolve os programas PGP e PL, visto nas seções 3 e 2 respectivamente. Faremos agora um esquema do funcionamento deste programa. As letras de A até D representam as rotinas específicas deste programa.

Fluxo de Informação :



As rotinas que compõem o programa PGG.FOR, são as mesmas dos programas PL e PGP, mais as quatro rotinas apresentadas a seguir :

A) Rotina LERPGG :

Lê os dados de entrada do PGG (1.48)

LER KZAO, NZAO
LER O NÚMERO DE MONÔMIOS DE p_k EM CADA RESTRIÇÃO.
LER O NÚMERO DE MONÔMIOS DE q_k EM CADA RESTRIÇÃO
LER OS COEFICIENTES c_{ki} POR LINHA (p_k)
LER OS COEFICIENTES d_{ki} POR LINHA (q_k)
LER OS EXPOENTES a_{kij} POR LINHA (p_k)
LER OS EXPOENTES b_{kij} POR LINHA (q_k)
LER OS LIMITANTES INFERIORES ($\underline{x}_j$)
LER OS LIMITANTES SUPERIORES ($\bar{x}_j$)

A) Rotina INIPGG

Inicia as variáveis para o PGG (leitura do ponto inicial factível). Essa rotina chama a rotina LERPGG.

ITG = 0 (0)
TOLG = 10^{-4} (ϵ)
LER OS DADOS DE ENTRADA
N = NZAO + 1
M = KZAO + 1
LER PONTO INICIAL PARA CONDENSAÇÃO
LER O VALOR DE ALFA (Caso haja a utilização da projeção PGP)

C) Rotina CONDV :

Esta rotina condensa o posinômio $1 + q_k(x)$ no ponto x^ℓ e faz a divisão do posinômio $p_k(x)$ pelo monômio gerado $q_k^\ell(x)$, gerando assim o problema de PGP.

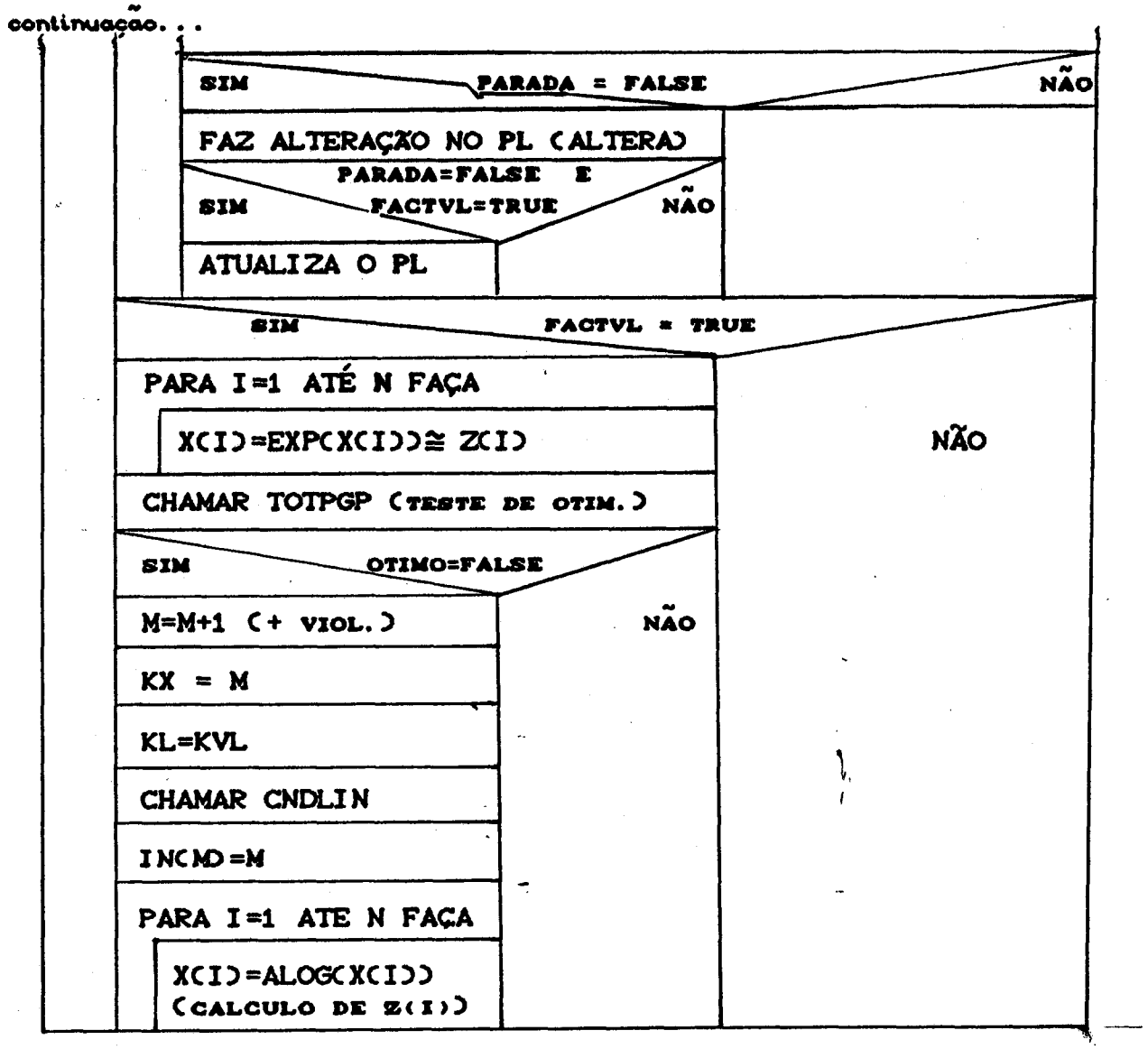
A montagem do PGP, feita nesta rotina, transfere os valores dos expoentes e coeficientes a serem utilizados no PGP.

QK = 0.0 (Usada para calcular o valor da restrição q_k no ponto x^ℓ)
CALCULAR OS VALORES DAS RESTRIÇÕES $1 + q_k(x)$ NO PONTO x^ℓ
CALCULAR OS ESPQ(k,i+1) (δ_{ki})
CALCULAR O MONÔMIO q_k^ℓ
CALCULAR O NOVO PGP RESULTANTE DA DIVISAO DE p_k / q_k^ℓ
MONTAGEM DE UM PGP (Transferência de dados para o PGP)

D) Rotina RESPGP :

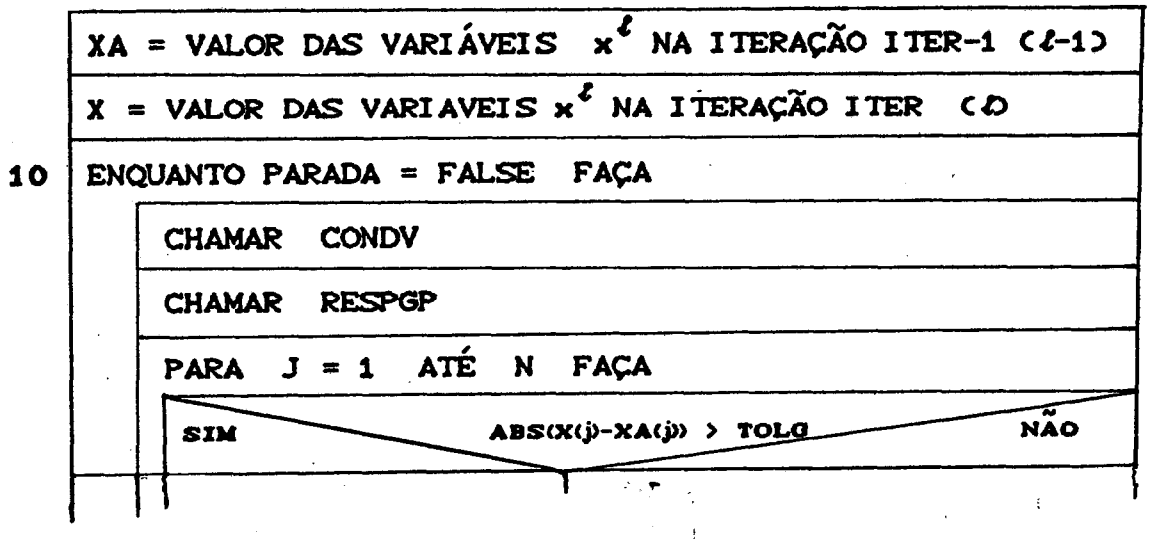
Resolve o PGP mostrado na rotina anterior. Esta rotina é praticamente o programa principal do PGP.

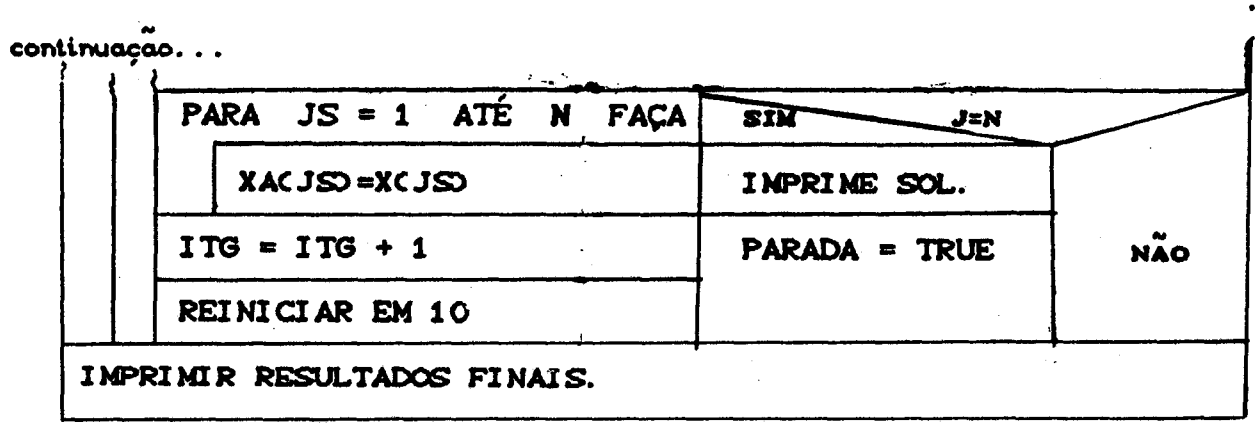
INICIA PGP	
ENQUANTO (OTIMO = FALSE E FACTVL = TRUE) FAÇA	
ITER = ITER + 1	
PARADA = FALSE	
SOJE = TRUE	
ENQUANTO PARADA = FALSE E FACTVL = TRUE FAÇA	
SIM — SOJE = TRUE E ITER ≠ 1	NÃO
JE = M	FAZER O TESTE DE OTIMALI-
SOJE = FALSE	DADE DO PL.



Para encerrarmos esse apêndice, faremos o fluxo de informação para o teste de otimalidade do PGG, existente no programa principal.

pp) Teste de parada do PGG.





Observe que a parada só será efetuada caso $|x_j - xA_j| < TOL$ para todos j. Outra maneira de parar, pode ser feita, levando em consideração o valor da Iteração (ITG), fazendo um teste antes de reiniciar o programa principal.(Ex.: Se ITG=100 então PARADA = TRUE, senão reiniciar em 10).

A estrutura utilizada neste trabalho, pode ser alterada em alguns casos, no sentido de adaptarmos o sistema geral para o uso em alguma área de pesquisa que envolva problemas deste tipo.

APÊNDICE C

PROTENSÃO, EM VIGAS DE CONCRETO, MODELADA POR PROGRAMAÇÃO GEOMÉTRICA

1. Introdução :

Problemas de Programação Geométrica surgem nas mais variadas áreas das ciências, como em Química, Física, Engenharias, entre outras. Ecker [14,15] descreve uma vasta bibliografia de aplicações dentro da Engenharia. Apresentaremos aqui uma aplicação na Engenharia Civil que não temos visto relacionada na literatura de Programação Geométrica. Trata-se de protensão em vigas de concreto.

A finalidade da protensão é criar esforços que se opõem àqueles provocados pelas ações externas (vento, carregamento, e outros), e também pelo peso próprio da estrutura a ser protendida.

A título de ilustração sobre Protensão, podemos comparar uma viga protendida com uma pilha de livros que queremos levantar. Para isso devemos submeter uma força de compressão longitudinal de maneira que ao levantarmos a pilha de livros, certas forças externas não desfaça a pilha.

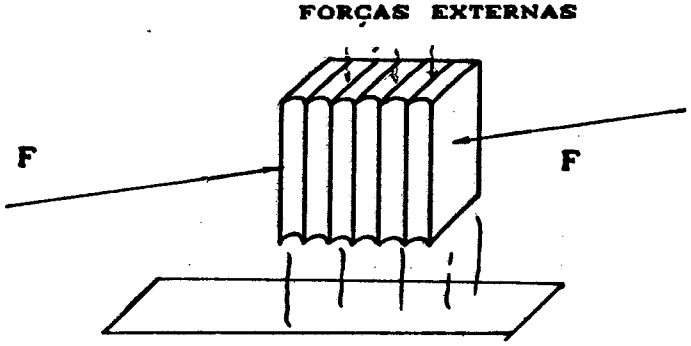


Fig. A

A força F deverá ser pelo menos suficiente para erguermos a pilha de livros sem desfazê-la.

Já em uma viga, a protensão é feita através de cabos apropriados que são colocados em seu interior (passando-os por bainhas já contidas na viga, introduzidas no momento de sua fabricação). Este cabo será preso em uma das extremidades da viga e esticado pela outra extremidade (por um macaco apropriado).

Após esticarmos o cabo, prendemos também essa extremidade, forçando o cabo a comprimir a viga longitudinalmente. A essa estratégia dá se o nome de protensão. O cabo, naturalmente, produzirá forças verticais, ao longo da viga, em função de seu traçado. Estas forças deverão vencer cargas externas.

Na medida que se faz necessário aumentar a protensão, os custos elevam-se rapidamente. Surge então, um problema de otimização : Dadas as características da viga e o tipo do concreto, como traçar os cabos, de modo que a viga suporte as cargas, com o mínimo possível de protensão. Maiores detalhes podem ser obtidos em [9].

2. Modelo Matemático :

O modelo a ser apresentado nesta seção foi inicialmente tratado em [9]. A figura B, ilustra nossas notações, mostrando um traçado aproximado de um cabo em uma viga.

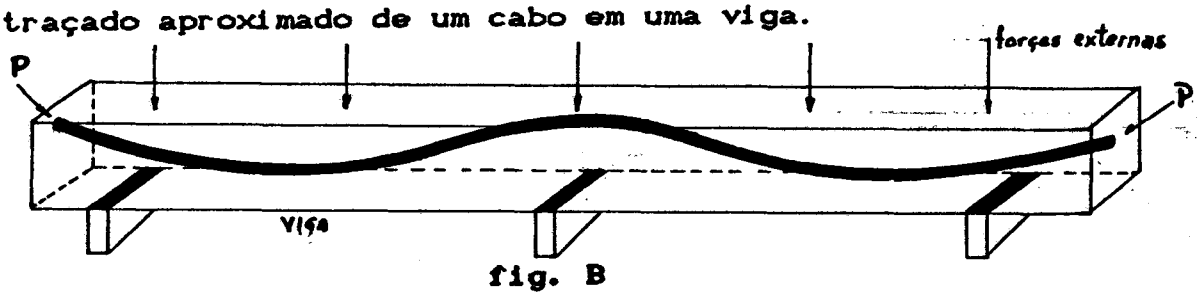


fig. B

A vantagem da protensão é possibilitar o uso de vigas com grandes vãos, e esbeltez numa proporção de 1/15 a 1/35, relação que é dada entre altura/vão, facilitando assim o seu uso (da viga) na Engenharia.

Geralmente o traçado dos cabos são curvos porém com uma curva bem suave, facilitando uma aproximação do traçado linear por partes. Isso é feito dividindo a viga em trechos iguais, e a partir dai estudaremos cada etapa destes trechos, supondo o traçado linear.

2.1 - Fuso limite :

Atendendo às condições fornecidas no ítem 4.1 da Norma Brasileira (NB - 116), para qualquer seção genérica, é necessário verificar quatro situações limites de tensões normais, para as fibras mais afastadas do baricentro da seção.

Assim, para cada seção (por simplicidade de notação supriremos o índice da seção) temos :

a) fibra superior :

. à compressão

(i) $$\sigma = \frac{-P}{Av} - \frac{M_{MAX}}{Ws} - \frac{M_P}{Ws} \geq \sigma_{cadm} \qquad (C.1)$$

. à tração

(ii) $$\sigma = -\frac{P}{Av} - \frac{M_{MAX}}{Ws} - \frac{M_P}{Ws} \leq \sigma_{tadm} \qquad (C.2)$$

b) fibra inferior :

. à compressão

(iii) $$\sigma = -\frac{P}{Av} - \frac{M_{MIN}}{Wi} - \frac{M_P}{Wi} \geq \sigma_{cadm} \qquad (C.3)$$

. à tração

(iv) $$\sigma = -\frac{P}{Av} - \frac{M_{MAX}}{Wi} - \frac{M_P}{Wi} \leq \sigma_{tadm} \qquad (C.4)$$

onde :

P - Força de Protensão;

σ - Tensão normal;

σ_{cadm} - Tensão admissível de compressão no concreto;

σ_{tadm} - Tensão admissível de tração no concreto;

Av - Área da seção transversal geométrica da viga;

MMIN - Momento mínimo;

MMAX - Momento máximo;

Mp - Momento hiperestático de protensão;

Ws - Módulo de resistência de uma seção referido à fibra pertencente à borda superior;

Wi - Módulo de resistência de uma seção referido à fibra pertencente à borda inferior;

E no ítem 5.3.2 da NB - 116 são dados os valores exigidos às tensões normais admissíveis, os quais são :

Tensão admissível à compressão :

$$\sigma \geq -\frac{f_{ck}}{2} \qquad (C.5)$$

onde f_{ck} representa a resistência à compressão do concreto.

Tensão admissível à tração :

$$\sigma_{tadm} \leq \begin{cases} 0 \text{ (protensão completa)} \\ 2\,\sigma_t \text{ (protensão limitada)} \end{cases} \qquad (8.6)$$

Podemos fazer uma representação gráfica das inequações (C.1) a (C.4) dadas anteriormente, no plano P x Mp, para cada

seção :

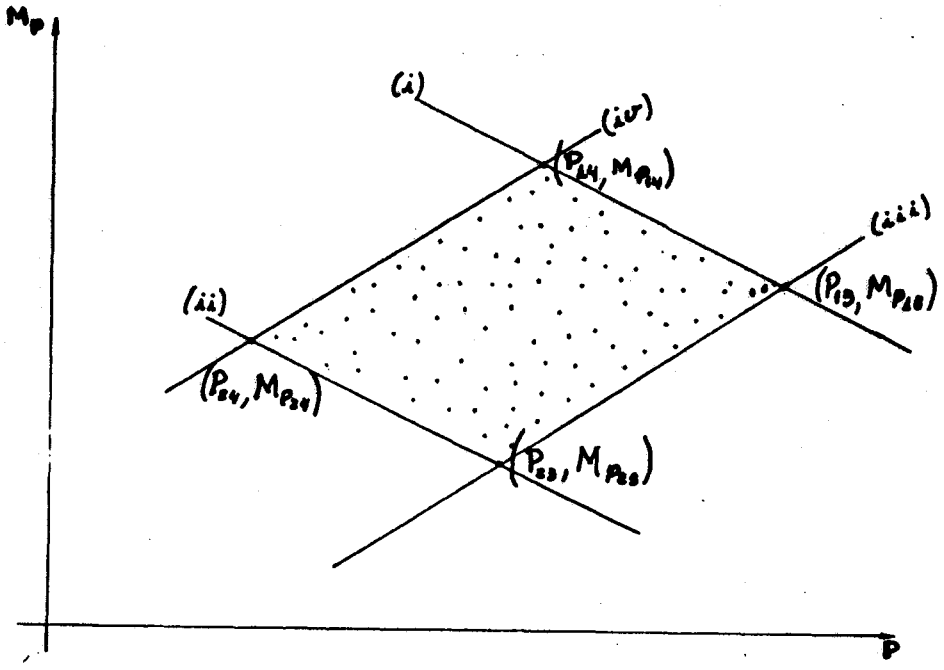


Fig. C : Fuso limite para uma função genérica .

O fuso limite nada mais é do que a região de factibilidade resultante pelas inequações (C.1) a (C.4) em uma seção.

Para ocorrer a protensão em uma certa seção, o fuso limite deve ser não vazio e P_{13} deverá ser maior do que zero. Portanto iremos supor que todas possibilidades de existência da protensão foram verificadas e passaremos assim à próxima fase.

2.2 - Obtenção do momento devido a protensão :

Inicialmente, veremos a aproximação de um cabo curvo por

uma poligonal, conforme ilustra a figura D.

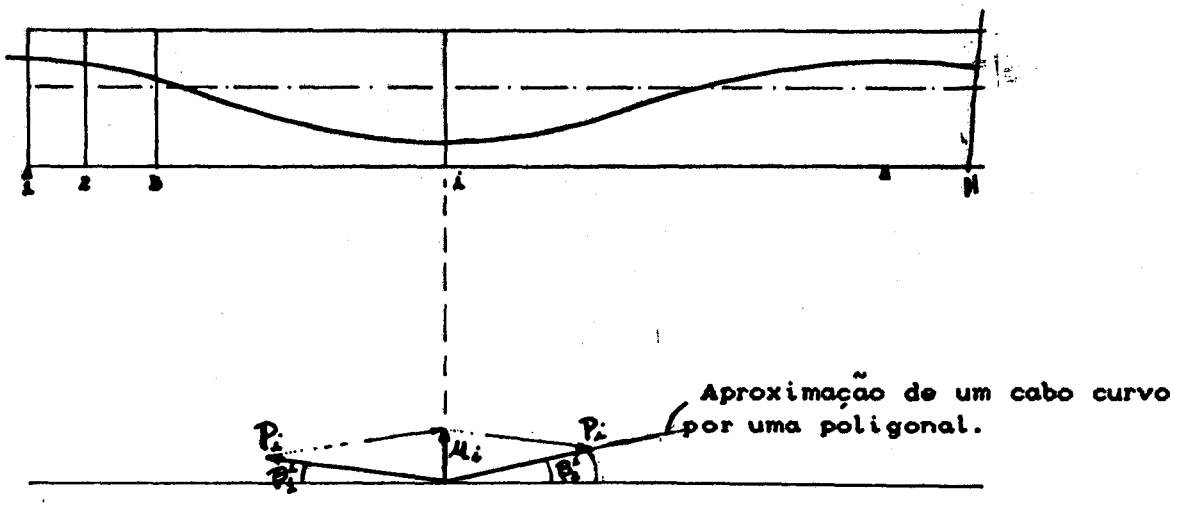


Fig. D .

onde α e β são ângulos de desvio de um cabo de protensão, u_i ação provocada por mudança de direção ou ancoragem de um cabo e n é o número de seções de estudo de uma viga.

A força u_i representa o esforço solicitante que um cabo protendido provoca na viga devido a mudança de direção "α_i", e essa ação será considerada sempre na direção transversal ao eixo da viga e seu valor será :

$$u_i = P_i \text{ sen } \beta_1^i + P_i \text{ sen } \beta_2^i \qquad (C.7)$$

Devido β^i ser bem pequeno, desprezamos os termos de ordem superior a 2 e usamos a aproximação :

$$\text{sen } \beta \cong \beta,$$

e substituindo em (8.7), com $\alpha_i = \beta_1^i + \beta_2^i$, temos :

$$u_i = \alpha_i * P_i \qquad (C.8)$$

onde P_i é a força de protensão na seção i .

O vetor de momento de protensão Mp, que representa os momentos obtidos ao longo da viga nos pontos estabelecidos, devido a protensão, pode ser escrito por :

$$Mp = MI * u \qquad (C.9)$$

onde a matriz MI é obtida aplicando-se ações unitárias, uma de cada vez nos pontos escolhidos, calculando-se a seguir os esforços produzidos em todos os outros pontos. Esse processo nada mais é do que a obtenção das "linhas de influência" para esses pontos. Ou seja, o elemento MI_{ki} representa o momento atuante na seção k quando aplicado uma força unitária na seção i. A essa matriz dá-se o nome de Matriz de Influência.

Substituindo (C.8) em (C.9) podemos relacionar o momento de protensão com a forma geométrica do cabo e a força de protensão. Desde que para os nós extremos temos :

$$(Mp)_1 = u_1 = y_1 P_1 \ , \ (Mp)_n = u_n = y_n P_n ,$$

segue que :

$$(Mp)_k = \sum_{i=1}^{N} MI_{ki} u_i = y_1 P_1 + \sum_{i=2}^{N-1} MI_{ki} \alpha_i P_i + y_n P_n \qquad (C.10)$$

Os ângulos α_i entre dois trechos lineares adjacentes pode ser escrito em função do traçado do cabo :

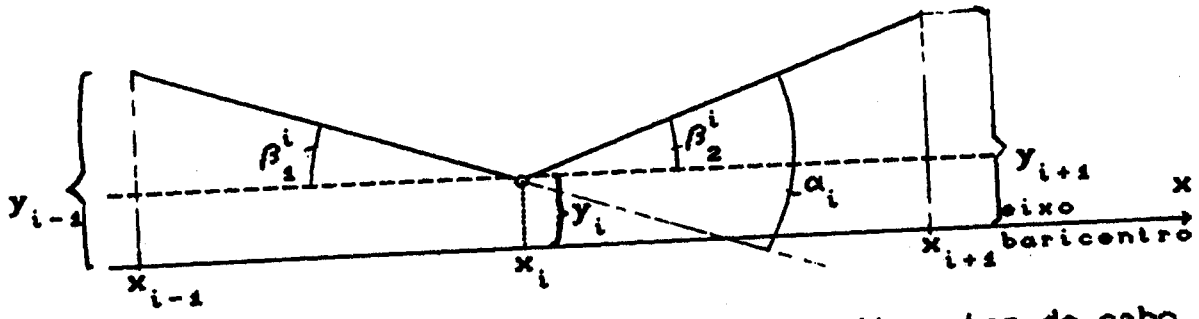


Fig. E : Representação de duas seções adjacentes do cabo linearizado

temos :

$$tg\,(\Pi - \beta_1^i) = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \qquad (tg\,\beta_1^i = -\frac{y_i - y_{i-1}}{x_i - x_{i-1}})$$

$$tg\,(\beta_2^i) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

como α e β são pequenos, novamente podemos fazer a aproximação

$tg\,\alpha = \alpha$,

$$\alpha_i = \beta_1^i + \beta_2^i = \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \right)$$

ou

$$\alpha_i = \frac{1}{(x_i - x_{i-1})}\, y_{i-1} - \frac{(x_{i+1} - y_i)}{(x_{i+1} - x_i)(x_i - x_{i-1})}\, y_i + \frac{1}{(x_{i+1} - x_i)}\, y_{i+1} \qquad (C.11)$$

Substituindo (C.11) em (C.10), obteremos o momento de protensão na seção k em função das forças de protensão ao longo da viga e do traçado do cabo :

$$(Mp)_k = P_1 y_1 + \sum_{i=2}^{n-1} \frac{M_{ki}}{(x_i - x_{i-1})} y_{i-1} P_i +$$
$$- \sum_{i=2}^{n-1} \frac{M_{ki} (x_{i+1} - x_{i-1})}{(x_{i+1} - x_i)(x_i - x_{i-1})} y_i P_i$$
$$+ \sum_{i=2}^{n-1} \frac{M_{ki}}{(x - x)} y_{i+1} P_i + P_n y_n \qquad (C.12)$$

Podemos representar $(Mp)_k$ por :

$$(Mp)_k = y^T A^k P \qquad (C.13)$$

onde

$$y^T = (y_i \ldots y_n), \quad P^T = (P_1, \ldots, P_n)$$

e A^k é uma matriz tridiagonal, com

$$a^k_{i,i-1} = \frac{M_{ki}}{(x_i - x_{i-1})}$$
$$a^k_{i-1,i} = \frac{M_{ki}}{(x_{i+1} - x_i)}$$

$$A^k = \begin{bmatrix} 1 & \frac{M_{k2}}{(x_2 - x_1)} & 0 & 0 & \ldots & 0 \\ 0 & \frac{M_{k2}(x - x)}{(x_2 - x_2)(x_2 - x_1)} & \frac{M_{k3}}{(x_3 - x_2)} & 0 & \ldots & 0 \\ 0 & \frac{M_{k2}}{(x_3 - x_2)} & \frac{M_{k2}(x_2 - x_1)}{(x_3 - x_2)(x_2 - x_1)} & \frac{M_{k4}}{(x_4 - x_3)} & \ldots & 0 \\ 0 & 0 & & & & \\ \cdot & \cdot & & & & \\ 0 & 0 & & & 0 \ldots & \frac{M_{kn}}{(x_n - x_{n-1})} \ 1 \end{bmatrix}$$

Agora, retornando às inequações (C.1) - (C.4), podemos escrevê-las em função da protensão (P) do traçado do cabo (y) :

Eq. C.1 -

$$\frac{1}{Av} P_k - \frac{1}{Ws} y^T A^k P \leq - \sigma_{cadm} - \frac{Mmax}{Ws} \qquad (C.14)$$

Eq. C.2 -

$$\frac{1}{Av} P_k + \frac{1}{Ws} y^T A^k P \geq - \sigma_{tadm} - \frac{Mmin}{Ws} \qquad (C.15)$$

Eq. C.3 -

$$\frac{1}{Av} P_k + \frac{1}{Wi} y^T A^k P \leq - \sigma_{cadm} - \frac{Mmin}{Wi} \qquad (C.16)$$

Eq. C.4 -

$$\frac{1}{Av} P_k + \frac{1}{Wi} y^T A^k P \geq - \sigma_{tadm} - \frac{Mmax}{Wi} \qquad (C.17)$$

para $k = 1,2,..,n$.

O problema de otimização consiste em determinar P e y que satisfaça as 4n restrições acima e

$$\text{minimize} \quad \sum_{j=1}^{n} P_j \qquad j = 1,2,..,n$$

onde n representa o número de seções inicialmente adotados na viga estudada.

Este problema, assim como outros dentro da Engenharia Civil, nos motivou ao estudo da Programação Geométrica. A estrutura particular das restrições permite consideráveis simplificações nos cálculos. A aplicação direta dos programas gerais, que foram desenvolvidos, é desnecessariamente dispendiosa, tanto na armazenagem dos coeficientes, como na avaliação e condensação das funções restrições. Era nossa intenção apresentar aqui um estudo de caso para este problema, porém devido às dificuldades de geração dos coeficientes para um problema real, aliado às perspectivas que surgiram (apresentadas nos capítulos anteriores), deixamos este estudo para uma etapa seguinte.

BIBLIOGRAFIA

[1] - ARENALES, M. N. - Programação Linear : Novos Métodos e Alguns Problemas Particulares; Tese de Doutorado, IEC - UNICAMP, (1984).

[2] - AVRIEL, M.; DEMBO, R. and PASSY, U. - "Solution of Generalized Geometric Programs ", International Journal of Numerical Methods in Engineering, vol. 9, 149-168,(1975).

[3] - AVRIEL, M. and WILLIAMS, A.C. - "Complementary Geometric Programming", Siam J. Appl. Math., vol. 19, no 1, 125-141, (1970).

[4] - BAZARAA, M. S. and JARVIS, J. J. - Linear Programming and Network Flows; John Wiley & Sons, (1979).

[5] - BEIGHTLER, C. S. and PHILLIPS, D. T. - Applied Geometric Programming, John Willey & Sons, (1976).

[6] - BECK, P. A. and ECKER, J. G. - "A Modified Concave Simplex Algorithm for Geometric Programming", Journal of Optimiz. Theory and Applications, (1978).

[7] - BRADLEY, J. - " Computational Aspects of Geometric Programming - 2. Polynomial Programming ", Engineering Optimization, vol. 3, 121-145, (1978).

[8] - CENSOR, Y. and LENT, A. - A Cyclic Subgradient Projections Method for the Convex Feasibility Problem, Technical Report, University of Haifa, Israel, (1980).

[9] - CEOTTO, L. H. - Contribuição à Otimização do Traçado de Cabos em Vigas Contínuas Protendidas; Dissertação de Mestrado , apresentada na Escola de Engenharia de São Carlos, Departamento de Estruturas, (1985).

[10] - DEMBO, R. S. - "Current State of the Art of Algorithms and Computer Software for Geometric Programming ", Journal of Optimization Theory and Applications, vol.26, no 2, 149-183, (1978).

[11] - DEMBO, R. S. and RIJCKAERT, M. J. - " Computational Aspects of Geometric Programming - 4. Computational Experiments in Geometric Programming ", Engineering Optimization, vol. 3, 161-173, (1978).

[12] - DINKEL, J. J.; ELLIOTT, W. H. and KOCHENBERGER, G. A. - Computational Aspects of Cutting-Plane Algoritms for Geometric Programming Problems ", Mathematical Programming, vol. 13, 200-220, (1977).

[13] - DUFFIN, R. J.; PETERSON, E. L. and ZENER, C. - Geometric Programming - Theory and Applications, John Wiley & Sons, (1967).

[14] - ECKER, J. G. - "Geometric Programming Methods, Computations and Applications", Siam Review, 22,no 3, 338-361,(1980).

[15] - ECKER, J. C.; GOCHET, W. and SMEERS, Y. - " Computational Aspects of Geometric Programming. 3 - Some Primal and Dual Algorithms for Posynomial and Signomial Geometric Programs", Engineering Optimization, vol. 3, 147-160, (1978).

[16] - GILL, P. E. and MURRAY, W. - Numerical Methods in Constrained Optimization, Academic Press New York, (1979).

[17] - LASDON, L.S. - Optimization Theory for Large Systems, The Macmillan Company, (1970).

[18] - LUENBERGER, D. G. - Linear and Nonlinear Programming, Addison-Wesley, 2ª edition, (1984).

[19] - MORRIS, A. J. - " Structural Optimization by Geometric Programming ", Int. J. Solids Struct., vol. 8, 847-864, (1972).

[20] - MORRIS, A. J. - " On Condensed Geometric Programming in Structural Optimization ", Comp. in Applied Mechanics and Engineering, 15, 139-148, (1978).

[21] - MOSES, F. and SUSUMU, O. - " Minimum Weight Design of Structures with Application to Elastic Grillages", International Journal for Numerical Methods in Engineering,

vol. I, 311-331, (1969).

[22] - RIJCKAERT, M. J. and MARTENS, X. M. - "A Comparison of Generalized Geometric Programming Algorithms", Katholieke Universiteit te Leuven, Report No. CE-RM-7503, (1975).

[23] - SPOSITO, V. A. - Linear and Nonlinear Programming, The Iowa State University Press/Ames, (1975).

[24] - TEMPLEMAN, A. B. - " Computational Aspects of Geometric Programming. 1 - Introduction and Basic Notation ", Eng. Optimization, vol. 3, 115-120, (1978).

ANEXOS

```
*                DISSERTACAO DE MESTRADO EM COMPUTACAO - USP-SC
*
*                          PROGRAMA GERAL - PL.FOR
*
*  APLICACAO : RESOLUCAO DE PROBLEMA LINEAR COM RESTRICOES E
*              VARIAVEIS CANALIZADAS
*
*              MINIMIZAR  CX
*              SUJEITO A  D<=AX<=E
*                         D1<=X<=E1
*              APRESENTANDO O SEGUINTE DIMENSIONAMENTO   :
*              C(N),X(N),D(M),A(M,N),E(M),D1(N),E1(N)
*
*  PROJETO : PROGRAMACAO GEOMETRICA POSINOMIAL
*
*
*  INSTITUICAO : INSTITUTO DE CIENCIAS MATEMATICAS DE SAO.CARLOS-USP
*
*  ORIENTADOR : PROF. DR. MARCOS NEREU ARENALES
*
*
*****************************************************************************

                        DESCRICAO DAS VARIAVEIS

V01) A(M,N) : MATRIZ TRANSPOSTA DO SISTEMA DE RESTRICOES

V02) ABI(M,M) : INVERSA DA BASE AB (A=(AB,AN))

V03) ACH(M) : PRODUTO ACH(I)=ABI(I,J)*A(J,K) I,J,K=1,...,M

V04) C(N) : VETOR CUSTO

V05) CIMA : INDICA SE A VIOLACAO OCORREU NA DESIGUALDADE D OU D1<= (FALSE)
            OU NA DESIGUALDADE <=E OU E1 (TRUE)

V06) D(M+N) : VETOR DOS LIMITES INFERIORES DAS RESTRICOES E DAS VARIAVEIS

V07) DELTA(N) : VETOR DOS VALORES POSSIVEIS PARA A VARIAVEL QUE ENTRA

V08) E(M+N) : IDEM A D(M+N) PARA LIMITES SUPERIORES

V09) IB(M) : VETOR DOS INDICES DAS VARIAVEIS BASICAS

V10) IBDEL(M) : VETOR DOS INDICES DE IB(M)  A QUE SE REFERE O DELTA

V11) ICONT : CONTADOR PARA O NUMERO DE PASSAGENS DOS PONTOS CRITICOS

V12) IK(I) : VETOR DOS INDICES DE DELTA ORDENADO CRESCENTEMENTE

V13) IN(N) : VETOR DOS INDICES DAS VARIAVEIS NAO-BASICAS

V14) IR : QUANTIDADE DE DELTA'S

V15) ISAIDA : FLAG PARA SAIDA DO MODULO

V17) K1 : INDICE AUXILIAR DE IBDEL NA FASE DE ESCOLHA DO DELTA

V18) K2 : IDEM A K1 PARA IB(M)

V19) K3 : IDEM A K1 PARA E(M+N) E D(M+N)

V20) M : NUMERO DE RESTRICOES

V21) N: NUMERO DE VARIAVEIS

V22) PARADA : CONTROLA O TERMINO DO PROGRAMA

V23) TB(I) : INDICA SE A VARIAVEL APONTADA POR IB(I) E IGUAL A E(IB(I)) (TRUE
             OU SE E IGUAL A D(IB(I)) (FALSE)

V24) TETA : CUSTO DA CONDICAO DE OTIMALIDADE MAIS VIOLADA

V25) TOL : TOLERANCIA NAS APROXIMACOES

V26) UB(M) : VETOR DA SOLUCAO DUAL

V27) X(N) : VETOR DA SOLUCAO PRIMAL

                        DESCRICAO DAS ROTINAS

R01) ALTERA
       FUNCAO : VERIFICA SE O PRIMAL E FACTIVEL E ALTERA A SOLUCAO
       ENTRADA : JE,M,N,CIMA,TOL,PARADA,TETA,K2,IN(M),ACH(M),ABI(M,M),
                 A(M,M),UB(M),TB(M),D(M+M),E(M+M),X(M)
       MODIFICACOES : PARADA,TETA,K2,ACH(M),TB(M),X(M)

R02) ATUALI
       FUNCAO : ATUALIZA ÁS VARIAVEIS E A SOLUCAO
       ENTRADA : CIMA,JE,TETA,N,K2,TB(M),IN(M),UB(M),IB(M),ACH(M),ABI(M,M),
                 X(M)
       MODIFICACOES : TB(M),IN(M),IB(M),UB(M),ABI(M,M),X(M)

R03) INICIA
       FUNCAO : INICIALIZA AS VARIAVEIS E A SOLUCAO
       ENTRADA : M,N,TOL,PARADA,C(N),D(M+N),E(M+N),ABI(M,M),IB(M),UB(M),TB(M)
                 X(M),IN(M)
       MODIFICACOES : TOL,PARADA,ABI(M,M),IB(M),UB(M),TB(M),X(M),IN(M)

R04) LEITUR
       FUNCAO : LE OS DADOS DE ENTRADA
       ENTRADA : M,N,C(N),A(M,M),D(M+N),E(M+N)
       MODIFICACOES : MM,N,A(M,M),D(M+N),E(M+N)

R05) TESOTI
       FUNCAO : FAZ O TESTE DE OTIMALIDADE E IMPRIME A SOLUCAO OTIMA
       ENTRADA : TETA,M,N,TOL,JE,CIMA,PARADA,IN(M),A(M,M),X(M),E(M+M),
                 D(M+M)
       MODIFICACOES : TETA,JE,CIMA,PARADA

                        OUTRAS INFORMACOES
```

```
C  ARQUIVO DE ENTRADA : T1PLP.DAT
C
C  FORMATO DE ENTRADA  DOS DADOS :
C       M,N
C       C(N)
C       MATRIZ A POR COLUNAS, UMA COLUNA EM CADA LINHA
C       D(M)
C       D1(M)
C       E(M)
C       E1(M)
C
C  ARQUIVO DE SAIDA : T1PLP.OUT
C
C  LINGUAGEM DE CODIFICACAO : FORTRAN 77
C
C  EQUIPAMENTO : PDP 11/45      SISTEMA : RSX-11M
C
C  OBSERVACAO :
C
C     COM RELACAO A LOGICA DAS ROTINAS, EXISTEM DOCUMENTACAO ANEXA
C
C
C     PROGRAMA PRINCIPAL DO PL.
C
      LOGICAL TB(25),CIMA,PARADA
      COMMON TB,CIMA,PARADA
      COMMON M,N,IB(25),IN(35),JE,K2
      COMMON C(25),A(25,60),D(85),E(85),ABI(25,25),X(25),UB(25)
      COMMON ACH(25),TOL,TETA
      CALL LEITUR
      CALL INICIA
   10 IF (.NOT.PARADA) THEN
        CALL TESOTI
        IF (.NOT.PARADA) THEN
          CALL ALTERA
        ENDIF
        IF (.NOT.PARADA) THEN
          CALL ATUALI
        ENDIF
        GOTO 10
      ENDIF
      stop
      END
C
```

```
C     SUBROTINA DE LEITURA DOS DADOS DE ENTRADA
C
      SUBROUTINE LEITUR
      LOGICAL TB(25),CIMA,PARADA
      COMMON TB,CIMA,PARADA
      COMMON M,N,IB(25),IN(35),JE,K2
      COMMON C(25),A(25,60),D(85),E(85),ABI(25,25),X(25),UB(25)
      COMMON ACH(25),TOL,TETA
      open(5,file='dados')
      open(6,file='saida',status='new')
      READ(5,*)M,N
      READ(5,*)(C(I),I=1,N)
      DO 10 I=1,M
        READ(5,*)(A(I,J),J=1,M)
   10 CONTINUE
      K=M+1
      L=M+N
      READ(5,*)(D(I),I=1,M)
      READ(5,*)(D(I),I=K,L)
      READ(5,*)(E(I),I=1,M)
      READ(5,*)(E(I),I=K,L)
      RETURN
      END
C
```

```
      SUBROUTINE INICIA

      PARAMETER (MV=50,MM=90,NC=90,MT=115)
      LOGICAL TB(60),CIMA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z0/TB
      COMMON /Z1/CIMA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NBMP(MM)
      COMMON /Z3/KX,KR,KVL,ITP,IB(MV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(MV)
      COMMON /Z6/A(MV,MC)
      COMMON /Z7/D(MT)
      COMMON /Z8/E(MT)
      COMMON /Z9/ABI(MV,MV)
      COMMON /ZA/X(MV)
      COMMON /ZB/UB(MV)
      COMMON /ZC/ACH(MV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/FI(20,60,MV)
      COMMON /ZG/XLI(MV),XLS(MV)
      COMMON /ZH/VVL
      COMMON /ZI/TTOL,IFIL
      COMMON /ZJ/IAE
      TTOL=0.5E-05
      ITL=0
      DO 10 I=1,M
        DO 20 J=1,M
          ABI(I,J)=0.0
   20   CONTINUE
   10 CONTINUE
      PARADA=.FALSE.
      DO 30 I=1,M
        IB(I)=I
        UB(I)=C(I)
        IF (UB(I).GE.0.0) THEN
          TB(I)=.FALSE.
          X(I)=D(I)
        ELSE
          TB(I)=.TRUE.
          X(I)=E(I)
        ENDIF
        ABI(I,I)=1.0
   30 CONTINUE
      DO 40 I=1,N
        IN(I)=I+M
   40 CONTINUE
      RETURN
      END
```


A

```
      SUBROUTINE ALTERA

      PARAMETER (NV=50,NH=90,NC=90,NT=115)
      LOGICAL TB(60),CIMA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB
      COMMON /Z1/CIMA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRMP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/F1(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL
      COMMON /ZI/TTOL,IFIL
      COMMON /ZJ/IAE
      DIMENSION DELTA(NV),IBDEL(NV),IK(NV)
      K=IN(JE)
      IF (K.GT.N) THEN
        DO 10 I=1,N
          ACH(I)=0.0
          DO 20 J=1,N
            ACH(I)=ABI(I,J)*A(J,K-N)+ACH(I)
   20     CONTINUE
   10   CONTINUE
      ELSE
        DO 30 I=1,N
          ACH(I)=ABI(I,K)
   30   CONTINUE
      ENDIF
      IR=0
      IF (CIMA) THEN
        DO 40 I=1,N
          Z=ACH(I)
          IF (ABS(Z).GT.TTOL) THEN
            AUX=UB(I)/Z
            IF (AUX.LT.(-TTOL)) THEN
              IR=IR+1
              DELTA(IR)=-AUX
              IBDEL(IR)=I
            ELSE
              IF (ABS(AUX).LE.TTOL) THEN
                IF ((TB(I).AND.(Z.GT.0.0)).OR.
     *              ((.NOT.TB(I)).AND.(Z.LT.0.0))) THEN
                  IR=IR+1
                  DELTA(IR)=AUX
                  IBDEL(IR)=I
                ENDIF
              ENDIF
            ENDIF
          ENDIF
   40   CONTINUE
      ELSE
        DO 50 I=1,N
          Z=ACH(I)
          IF (ABS(Z).GT.TTOL) THEN
            AUX=UB(I)/Z
            IF (AUX.GT.TTOL) THEN
              IR=IR+1
              DELTA(IR)=AUX
              IBDEL(IR)=I
            ELSE
              IF (ABS(AUX).LE.TTOL) THEN
                IF (((.NOT.TB(I)).AND.(Z.GT.0.0))
     *              .OR.(TB(I).AND.(Z.LT.0.0))) THEN
                  IR=IR+1
                  DELTA(IR)=AUX
                  IBDEL(IR)=I
                ENDIF
              ENDIF
            ENDIF
          ENDIF
   50   CONTINUE
      ENDIF
      IF (IR.EQ.0) THEN
        WRITE(6,60)
   60   FORMAT(15X,'>>>  PROBLEMA INFACTIVEL <<<')
        PARADA=.TRUE.
        FACTVL=.FALSE.
      ELSE
        ICONT=0
        ISAIDA=0
        IF (IR.EQ.1) THEN
          IF (DELTA(1).LE.TTOL) THEN
            K2=IBDEL(1)
            ISAIDA=1
          ELSE
            IK(1)=1
          ENDIF
        ELSE
          AUX=1.0E+05
          DO 70 I=1,IR
            DO 80 J=1,IR
              IF ((AUX.GT.DELTA(J)).AND.(DELTA(J).NE.(-1.0))) THEN
                AUX=DELTA(J)
                L=J
              ENDIF
   80       CONTINUE
            IK(I)=L
            DELTA(L)=-1.0
            AUX=1.0E+05
   70     CONTINUE
        ENDIF
        DO 90 I=1,IR
          IF (ISAIDA.EQ.0) THEN
            K1=IK(I)
            K2=IBDEL(K1)
            K3=IB(K2)
            AUX=(E(K3)-D(K3))*(ABS(ACH(K2)))
            IF ((TETA-AUX).LT.(-TTOL)) THEN
              ISAIDA=1
            ELSE
              ICONT=ICONT+1
              TETA=TETA-AUX
              IF (TB(K2)) THEN
                AUX=D(K3)-E(K3)
              ELSE
                AUX=E(K3)-D(K3)
              ENDIF
              TB(K2)=.NOT.TB(K2)
              DO 100 J=1,N
                X(J)=ABI(K2,J)*AUX+X(J)
  100         CONTINUE
            ENDIF
          ENDIF
   90   CONTINUE
      ENDIF
      IF (((ICONT.EQ.IR).AND.(.NOT.PARADA)) THEN
        WRITE(6,60)
        PARADA=.TRUE.
        FACTVL=.FALSE.
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE TESOTI

      PARAMETER (NV=50,NH=90,NC=90,NT=115)
      LOGICAL TB(60),CIMA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB
      COMMON /Z1/CIMA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRMP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/F1(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL
      COMMON /ZI/TTOL,IFIL
      COMMON /ZJ/IAE
      TETA=0.0
      DO 10 I=1,N
        K=IN(I)
        IF (IN(I).GT.N) THEN
          AUX=0.0
          DO 20 J=1,N
            AUX=A(J,K-N)*X(J)+AUX
   20     CONTINUE
        ELSE
          AUX=X(K)
        ENDIF
        IF (AUX.LE.(E(K)+TTOL)) THEN
          IF (AUX.LT.(D(K)-TTOL)) THEN
            IF ((D(K)-AUX).GT.(TETA+TTOL)) THEN
              TETA=D(K)-AUX
              JE=I
              CIMA=.FALSE.
            ENDIF
          ENDIF
        ELSE
          IF ((AUX-E(K)).GT.(TETA+TTOL)) THEN
            TETA=AUX-E(K)
            JE=I
            CIMA=.TRUE.
          ENDIF
        ENDIF
   10 CONTINUE
      IF (TETA.EQ.0.0) THEN
        PARADA=.TRUE.
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE ATUALI

      PARAMETER (NV=50,NH=90,NC=90,NT=115)
      LOGICAL TB(60),CIMA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB
      COMMON /Z1/CIMA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRMP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/F1(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL
      COMMON /ZI/TTOL,IFIL
      COMMON /ZJ/IAE
      TB(K2)=CIMA
      L=IN(JE)
      IN(JE)=IB(K2)
      IB(K2)=L
      AUX=ACH(K2)
      UB(K2)=UB(K2)/AUX
      IF (CIMA) THEN
        U=-TETA
      ELSE
        U=TETA
      ENDIF
      DO 10 I=1,N
        ABI(K2,I)=ABI(K2,I)/AUX
        X(I)=ABI(K2,I)*U+X(I)
   10 CONTINUE
      DO 20 I=1,N
        IF (I.NE.K2) THEN
          AUX=-ACH(I)
          UB(I)=UB(K2)*AUX+UB(I)
          DO 30 J=1,N
            ABI(I,J)=ABI(K2,J)*AUX+ABI(I,J)
   30     CONTINUE
        ENDIF
   20 CONTINUE
      RETURN
      END
```

```
C  ***************************************************************************
C  *                                                                         *
C  *                  BOLSA DE MESTRADO COMPUTACAO  -  FAPESP                *
C  *                                                                         *
C  *                      PROGRAMA NUMERO 1                                  *
C  *                                                                         *
C  *  APLICACAO : RESOLUCAO DE PROBLEMA GEOMETRICO POSINOMIAL ACELERADO      *
C  *                                                                         *
C  *              MINIMIZAR  X(0)                                            *
C  *              SUJEITO A  GX(K)<=1  K=0.....KZAO                          *
C  *                     0<XLI(J)<=X(J)<=XLS(J)   J=0.....NZAO               *
C  *                                                                         *
C  *  PROJETO : PROGRAMACAO GEOMETRICA POSINOMIAL                            *
C  *                                                                         *
C  *  PROCESSO : 85/0830-1                                                   *
C  *                                                                         *
C  *  INSTITUICAO : INSTITUTO DE CIENCIAS MATEMATICAS DE SAO CARLOS-USP      *
C  *                                                                         *
C  *  ORIENTADOR : PROF. DR.  MARCOS NEREU ARENALES                          *
C  *                                                                         *
C  *  BOLSISTA : EDMUNDO SERGIO SPOTO                                        *
C  *                                                                         *
C  ***************************************************************************
C
C
C                        DESCRICAO DAS VARIAVEIS
C
C
C  V01) EPSP(I) : EXPOENTE DA DESIGUALDADE ARITMETICA-GEOMETRICA
C
C  V02) FACTVL : INDICA SE O PGP E FACTIVEL (TRUE)
C
C  V03) FI(K,I,J) : EXPOENTE DA VARIAVEL X(J) DO MONOMIO I DA RESTRICAO K
C
C  V04) GK : ACUMULADOR PARA O VALOR DA RESTRICAO EM QUESTAO
C
C  V05) GKP : IDEM A GK
C
C  V06) ITER : CONTADOR DE ITERACOES
C
C  V07) KR : COLUNA DA RESTRICAO MAIS VIOLADA
C
C  V08) KVL : INDICE DA RESTRICAO MAIS VIOLADA
C
C  V09) KX : COLUNA DA NOVA RESTRICAO INCORPORADA
C
C  V10) KZAO : NUMERO DE RESTRICOES - 1
C
C  V11)  L : INDICE DAS VARIAVEIS PARA IMPRESSAO DA SOLUCAO OTIMA
C
C  V12) NRMP(K) : NUMERO DE MONOMIOS DA RESTRICAO K
C
C  V13) NZAO : NUMERO DE VARIAVEIS -1
C
C  V14) OTIMO : INDICA SE O PGP TEM SOLUCAO OTIMA (TRUE)
C
C  V15) SOJE : INDICA SE DEVEMOS ALTERAR A VARIAVEL JE DIRETAMENTE, OU
C              SE DEVEMOS CHAMAR A SUBROTINA TESOTI
C
C  V16) TET(K,I) : COEFICIENTE DO MONOMIO I NA RESTRICAO K
C
C  V17) VVL : VALOR DA RESTRICAO MAIS VIOLADA
C
C  V18) XLI(J) : LIMITE INFERIOR PARA X(J)
C
C  V19) XLS(J) : LIMITE SUPERIOR PARA X(J)
C
C
C                        DESCRICAO DAS ROTINAS
C
C
C  R01) CNDLIN
C         FUNCAO : CONDENSA E LINEARIZA UMA RESTRICAO
C         ENTRADA : NZAO,NRMP(K),TET(K,I),FI(K,I,J),A(J,K),D(Y),E(Y),X(J)
C         MODIFICACOES : A(J,K),D(Y),E(Y)
C
C  R02) INIPGP
C         FUNCAO : LE OS DADOS DE ENTRADA, INICIALIZA AS VARIAVEIS,
C                  LINEARIZA O PGP E INICIALIZA O PL
C         ENTRADA:TOL,N,M,OTIMO,FACTVL,C(N),X(N)(PONTO INICIAL),D(Y),E(Y)
C                 E OS DAS ROTINAS LERPGP, CNDLIN E INICIA
C         MODIFICACOES : TOL,N,M,OTIMO,FACTVL,C(N),X(N),D(Y),E(Y), E AS
C                        MODIFICADAS PELAS ROTINAS LERPGP,CNDLIN E INICIA
C
C  R03) LERPGP
C         FUNCAO : LE OS DADOS DE ENTRADA DO PGP
C         ENTRADA : KZAO,NZAO,NRMP(K),TET(K,I),FI(K,I,J),XLI(J),XLS(J)
C         MODIFICACOES : KZAO,NZAO,NRMP(K),TET(K,I),FI(K,I,J),XLI(J),XLS(J)
C
C  R04) TOTPGP
C         FUNCAO : FAZ O TESTE DE OTIMALIDADE DO PGP
C         ENTRADA : KZAO,NZAO,TOL,KVL,OTIMO,NRMP(K),TET(K,I),FI(K,I,J),X(J)
C         MODIFICACOES : KVL,OTIMO
C
C
C                        OUTRAS INFORMACOES
C
C
C  ARQUIVO DE ENTRADA : PGPDAD
C
C  FORMATO DE ENTRADA DOS DADOS :
C       KZAO,NZAO
C       NRMP(K)
C       CONJUNTO DE COEFICIENTES DA RESTRICAO, UMA RESTRICAO POR LINHA
C       CONJUNTO DE EXPOENTES DO MONOMIO, UM MONOMIO POR LINHA
C       XLI(J)
C       XLS(J)
C
C  ARQUIVO DE SAIDA : PGPSAI
C
C  LINGUAGEM DE CODIFICACAO : FORTRAN 77
C
C  EQUIPAMENTO : XT-2002
C
```

```
C  OBSERVACOES :
C
C    PARA A RESOLUCAO DO PL FORAM UTILIZADAS ROTINAS DO PROGRAMA NUMERO 1.
C    SAO ELAS INICIA, TESOTI, ALTERA E ATUALI, COM AS SEGUINTES MODIFICACOES
C       1) TOL E PARADA NAO SAO INICIALIZADOS EM INICIA;
C
C       2) TESOTI NAO IMPRIME A SOLUCAO OTIMA DO PL;
C
C       3) EM ALTERA, APOS MENSAGEM DE INFACTIBILIDADE E FEITO FACTVL=FALSE.
C
C    QUANDO UMA NOVA RESTRICAO E INTRODUZIDA EM PL, CERTAMENTE, ELA ENTRARA NA
C BASE. PORTANTO, NA PRIMEIRA ITERACAO PARA RESOLVER O NOVO PL, NAO E  NECES-
C SARIO O TESTE DE OTIMALIDADE. BASTA MODIFICARMOS APENAS A VARIAVEL JE. ISSO
C E CONTROLADO PELA VARIAVEL LOGICA SOJE.
C
C    PARA A VARIAVEL Z(J), DA TEORIA,  E UTILIZADO O PROPRIO VETOR X(J), COM A
C MODIFICACAO, NOS INSTANTES CONVENIENTES, DADA POR Z(J)=LN(X(J)).
C
C    A DOCUMENTACAO DAS VARIAVEIS E ROTINAS DO PROGRAMA NUMERO 1, AQUI UTILIZA-
C DAS, NAO E FEITA AQUI.
C
C    COM RELACAO A LOGICA DAS ROTINAS, EXISTE DOCUMENTACAO ANEXA.
C
```

```
C          NOME : PCONASP.FOR
C
C
C  CONDENSA NA RESTRICAO MAIS VIOLADA, SEM PROJECAO...
C
      PARAMETER (NV=50,NH=90,NC=90,NT=115)
      LOGICAL TB(60),CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRNP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/FI(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL
      COMMON /ZI/TTOL,IFIL
      COMMON /ZJ/IAE
      DIMENSION Z(NV)
      OPEN(5,FILE='PGPDA1',STATUS='OLD',ACCESS='SEQUENTIAL')
      OPEN(6,FILE='PCONASP.SAI',STATUS='NEW',ACCESS='SEQUENTIAL',FORM='F
     &ORMATTED')
      OPEN(7,FILE='PGPDA2',STATUS='OLD',ACCESS='SEQUENTIAL')
      OPEN(8,FILE='PGPDE1',STATUS='OLD',ACCESS='SEQUENTIAL')
      OPEN(9,FILE='PGPDE2',STATUS='OLD',ACCESS='SEQUENTIAL')
      OPEN(10,FILE='PGPTAB',STATUS='NEW',FORM='FORMATTED')
      CALL INIPGP
      IFIL=0
      WRITE(6,5)TOL,IAE
      WRITE(10,5)TOL,IAE
    5 FORMAT(20X,'Programa :PCONASP.FOR'/20X,'TOLERANCIA PGP = ',F8.5//,
     *20X,'EXERCICIO Numero :',I2//,2X,' ITP  ITL  N.CO  SOL. OTIMA P
     *L  SOL. OTIMA PGP '/)
   10 IF (.NOT.OTIMO.AND.FACTVL) THEN
        ITP=ITP+1
        ITL=0
        PARADA=.FALSE.
        SOJE=.TRUE.
   20   IF ((.NOT.PARADA).AND.FACTVL) THEN
          ITL=ITL+1
          IFIL=IFIL+1
          IF (SOJE.AND.(ITP.NE.1)) THEN
            JE=N
            CINA=.TRUE.
            AUX=0.
            DO 25 J=1,N
              AUX=AUX+A(J,M)*X(J)
   25       CONTINUE
            TETA=AUX-E(M+N)
            SOJE=.FALSE.
          ELSE
            CALL TESOTI
          ENDIF
          IF (.NOT.PARADA) THEN
            CALL ALTERA
          ENDIF
          IF ((.NOT.PARADA).AND.FACTVL) THEN
            CALL ATUALI
          ENDIF
          GOTO 20
        ENDIF
        IF (FACTVL) THEN
          DO 30 J=1,N
            Z(J)=X(J)
            X(J)=EXP(X(J))
   30     CONTINUE
        WRITE(6,45)ITP,ITL,N,X(1)
        WRITE(10,45)ITP,ITL,N,X(1)
   45   FORMAT(3X,I3,3X,I3,3X,I4,3X,F11.5)
        DO 47 J=2,N
          WRITE(10,48)X(J)
   47   CONTINUE
   48   FORMAT(22X,F11.5)
          CALL TOTPGP
          IF (.NOT.OTIMO) THEN
            M=M+1
            KX=M
            KR=KVL
            CALL CNDLIN
            IN(M)=M+N
            DO 100 J=1,N
              X(J)=Z(J)
  100       CONTINUE
          ENDIF
        ENDIF
        GOTO 10
      ENDIF
      STOP
      END
```

```
C
      SUBROUTINE INIPGP
C
      PARAMETER (NV=50,NH=90,NC=90,NT=115)
      LOGICAL TB(60),CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRNP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/FI(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL
      COMMON /ZI/TTOL,IFIL
      COMMON /ZJ/IAE
      COMMON /ZL/ATIL(NV)
      ITP=0
      TOL=.5E-03
      CALL LERPGP
      CINA=.TRUE.
      M=KZAO+1
      C(1)=1.0
      DO 10 I=2,M
        C(I)=0.0
   10 CONTINUE
      READ(IAE,*)(X(I),I=1,M)
      KAUX=KZAO+1

C     Condensa todas restricoes inicialmente, no ponto inicial . .

      DO 20 K=1,KAUX
        KX=K
        KR=K
        CALL CNDLIN
   20 CONTINUE
      N=NZAO+1
      DO 30 J=1,N
        D(J)=ALOG(XLI(J))
        E(J)=ALOG(XLS(J))
   30 CONTINUE
      CALL INICIA
      OTIMO=.FALSE.
      FACTVL=.TRUE.
      RETURN
      END
```

```
      SUBROUTINE LERPGP

      PARAMETER (NV=50,NH=90,NC=90,NT=115)
      LOGICAL TB(60),CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRNP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/FI(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL
      COMMON /ZI/TTOL,IFIL
      COMMON /ZJ/IAE
      COMMON /ZL/ATIL(NV)

C     IAE  E O INDICE DO ARQUIVO DE ENTRADA. . .

      WRITE(*,*)'FORNECER O INDICE DO ARQUIVO DE ENTRADA (IAE)'
      WRITE(*,*)'5  =  PGPDA1'
      WRITE(*,*)'7  =  PGPDA2'
      WRITE(*,*)'8  =  PGPDE1'
      WRITE(*,*)'9  =  PGPDE2'
      READ(*,*)IAE
      READ(IAE,*)KZAO,NZAO
      KAUX=KZAO+1
      N=NZAO+1
      READ(IAE,*)(NRNP(K),K=1,KAUX)
      DO 15 K=1,KAUX
        IAUX=NRNP(K)
        DO 10 I=1,IAUX
          READ(IAE,*)TET(K,I)
   10   CONTINUE
   15 CONTINUE
```

```
C     Verifica se a restricao tem mais do que 7 variaveis. . .
      IF (N.GT.7) THEN
        JXT1=7
        READ(IAE,*)(FI(1,1,J),J=1,JXT1)
        READ(IAE,*)(FI(1,1,J),J=8,N)
        DO 30 K=2,KAUX
          IAUX=NRNP(K)
          DO 20 I=1,IAUX
            READ(IAE,*)(FI(K,I,J),J=1,N)
   20     CONTINUE
   30   CONTINUE
        ELSE
        DO 35 K=1,KAUX
          IAUX=NRNP(K)
          DO 34 I=1,IAUX
            READ(IAE,*)(FI(K,I,J),J=1,N)
   34     CONTINUE
   35   CONTINUE
      ENDIF
      READ(IAE,*)(XLI(J),J=1,N)
      READ(IAE,*)(XLS(J),J=1,N)
      WRITE(6,36)
   36 FORMAT(40X,'PCONASP.FOR'///,30X,'Este programa resolve um PGP con
     &densando a restricao'/30X,'mais violada no ponto obtido no PL.'/)
      RETURN
      END
```

```
      SUBROUTINE CNDN.IN

      PARAMETER (NV=50,NH=90,NC=90,NT=115)
      LOGICAL TB(60),CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/N,M,JE,K2,KZAO,NZAO,NRNP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/FI(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL
      COMMON /ZI/TTOL,IFIL
      COMMON /ZJ/IAE
      DIMENSION EPSP(NH)
      IAUX=NRNP(KR)
      GK=0.0
      DO 20 I=1,IAUX
        AUX=TET(KR,I)
        DO 10 J=1,N
          AUX1=X(J)**(FI(KR,I,J))
          AUX=AUX*AUX1
   10   CONTINUE
        EPSP(I)=AUX
        GK=GK+AUX
   20 CONTINUE
      DO 30 I=1,IAUX
        EPSP(I)=EPSP(I)/GK
   30 CONTINUE
      DO 50 J=1,N
        AUX=0.0
        DO 40 I=1,IAUX
          AUX=AUX+FI(KR,I,J)*EPSP(I)
   40   CONTINUE
        A(J,KK)=AUX
   50 CONTINUE
      AUX=1.0
      DO 60 I=1,IAUX
        IF (EPSP(I).LT.(1.E-07)) THEN
          AUX2=EPSP(I)**EPSP(I)
          AUX1=(TET(KR,I)**EPSP(I))/AUX2
          AUX=AUX*AUX1
        ELSE
          AUX=AUX*((TET(KR,I)/EPSP(I))**EPSP(I))
        ENDIF
   60 CONTINUE
      E(KK+N)=-ALOG(AUX)
      D(KK+N)=-1.0E+04
   70 RETURN
      END

C
```

```
      SUBROUTINE TOTPGP
C
      PARAMETER (NV=50,NH=90,NC=90,NT=115)
      LOGICAL TB(60),CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/N,M,JE,K2,KZAO,NZAO,NRNP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/FI(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL
      COMMON /ZI/TTOL,IFIL
      COMMON /ZJ/IAE
      KAUX=KZAO+1
      VVL=1.0
      KVL=0
      DO 30 K=1,KAUX
        GKP=0.0
        IAUX=NRNP(K)
        DO 20 I=1,IAUX
          AUX=TET(K,I)
          DO 10 J=1,N
            AUX=AUX*X(J)**FI(K,I,J)
   10     CONTINUE
          GKP=GKP+AUX
   20   CONTINUE
        IF (GKP.GT.1+TOL) THEN
          IF (GKP.GT.VVL) THEN
            KVL=K
            VVL=GKP
          ENDIF
        ENDIF
   30 CONTINUE
      IF (KVL.EQ.0) THEN
        OTIMO=.TRUE.
        WRITE(6,36)ITP,IFIL,N,X(1)
        WRITE(10,36)ITP,IFIL,N,X(1)
   36   FORMAT(2X,I3,2X,I3,2X,I3,20X,F11.5)
        DO 37 J=2,N
          WRITE(6,38)X(J)
          WRITE(10,38)X(J)
   37   CONTINUE
   38   FORMAT(38X,F11.5)
      ENDIF
      RETURN
      END
C
```

— Rotinas do PL —

```
C          Nome do Programa : PPRCONA.FOR

C     Condensa depois de projetar na restricao mais violada. . .

      PARAMETER (NV=50,NH=100,NC=90,NT=115)
      LOGICAL TB(30),PROJET,CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB,PROJET
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/N,M,JE,K2,KZAO,NZAO,NRNP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/FI(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL,ALF,VMP
      COMMON /ZI/TTOL,ITL
      COMMON /ZJ/IAE,IFIL
      COMMON /ZL/ATIL(NV)
      DIMENSION Z(NV)
      OPEN(5,FILE='PGPDA1',STATUS='OLD')
      OPEN(6,FILE='PPRCONA.SAI',STATUS='NEW',FORM='FORMATTED')
      OPEN(7,FILE='PGPDA2',STATUS='OLD')
      OPEN(8,FILE='PGPDE1',STATUS='OLD')
      OPEN(9,FILE='PGPDE2',STATUS='OLD')
      OPEN(10,FILE='PGPTAB',STATUS='NEW',FORM='FORMATTED')
      CALL IN!PGP
      IF (ALF.EQ.0.) THEN
        PROJET=.FALSE.
      ENDIF
      IFIL=0
      WRITE(6,5)TOL,IAE
      WRITE(10,5)TOL,IAE
    5 FORMAT(20X,'Programa :PCONACP.FOR'/20X,'TOLERANCIA PGP = ',F9.6//,
     *20X,'EXERCICIO Numero :',I2//,2X,' ITP * ITL * N.CO * SOL. OTIMA P
     *L * SOL. OTIMA PGP *'/)
   10 IF (.NOT.OTIMO.AND.FACTVL) THEN
        ITP=ITP+1
        ITL=0
        PARADA=.FALSE.
        SOJE=.TRUE.
   20   IF ((.NOT.PARADA).AND.FACTVL) THEN
          ITL=ITL+1
          IFIL=IFIL+1
          IF (SOJE.AND.(ITP.NE.1)) THEN
            JE=M
            CINA=.TRUE.
            AUX=0.
            DO 25 J=1,N
              AUX=AUX+A(J,M)*X(J)
   25       CONTINUE
            TETA=AUX-E(M+M)
            SOJE=.FALSE.
          ELSE
            CALL TESOT!
          ENDIF
          IF (.NOT.PARADA) THEN
            CALL ALTERA
          ENDIF
          IF ((.NOT.PARADA).AND.FACTVL) THEN
            CALL ATUAL!
          ENDIF
          GOTO 20
        ENDIF
        IF (FACTVL) THEN
          DO 30 J=1,N
            Z(J)=X(J)
            X(J)=EXP(X(J))
   30     CONTINUE
        WRITE(6,45)ITP,ITL,N,X(1)
        WRITE(10,45)ITP,ITL,N,X(1)
   45   FORMAT(3X,I3,3X,I3,3X,I4,3X,F11.5)
        DO 47 J=2,N
          WRITE(10,48)X(J)
   47   CONTINUE
   48   FORMAT(22X,F11.5)
          CALL TOTPGP
C         Condensa na restricao mais violada no ponto projetado nesta
C         nesta restricao. . .
          IF (.NOT.OTIMO) THEN
            M=M+1
            KX=M
            KR=KVL
            CALL CNDL!N
            IN(M)=M+N
            DO 100 J=1,N
              X(J)=Z(J)
  100       CONTINUE
          ENDIF
        ENDIF
        GOTO 10
      ENDIF
      STOP
      END
```

rotinas do PGP (ja vistas anteriores)

```
      SUBROUTINE PROJX

      PARAMETER (NV=50,NH=100,NC=90,NT=115)
      LOGICAL TB(30),PROJET,CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB,PROJET
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRHP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/FI(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL,ALF,VMP
      COMMON /ZI/TTOL,ITL
      COMMON /ZJ/IAE,IFIL
      COMMON /ZL/ATIL(NV)
      ATINOD=0.
      DO 10 J=1,N
        ATINOD=ATINOD+ATIL(J)*ATIL(J)
   10 CONTINUE
      DO 20 J=1,N
        EXPTIL=-ALFA(ATIL(J)/ATINOD)
        X(J)=X(J)*VVL**EXPTIL
   20 CONTINUE

C     Terminado a projecao na restricao mais violada, projetamos
C     agora sobre a regiao definida pelos limitantes inferiores e
C     superiores .

        DO 30 J=1,N
          ATIL(J)=0.0
          IF (X(J).LT.XLI(J)) THEN
            X(J)=XLI(J)
          ELSE
            IF (X(J).GT.XLS(J)) THEN
              X(J)=XLS(J)
            ENDIF
          ENDIF
   30   CONTINUE

C       Fim da projecao. . .

      RETURN
      END
```

```
      SUBROUTINE TOTPGP

      PARAMETER (NV=50,NH=100,NC=90,NT=115)
      LOGICAL TB(30),PROJET,CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB,PROJET
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRHP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(20,60)
      COMMON /ZF/FI(20,60,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL,ALF,VMP
      COMMON /ZI/TTOL,ITL
      COMMON /ZJ/IAE,IFIL
      COMMON /ZL/ATIL(NV)
      DIMENSION EPSAUX(60),EPSP(60)
      KAUX=KZAO+1
      VVL=1.0
      KVL=0
      DO 30 K=1,KAUX
        GKP=0.0
        IAUX=NRHP(K)
        DO 20 I=1,IAUX
          AUX=TET(K,I)
          DO 10 J=1,N
            AUX1=X(J)**FI(K,I,J)
            AUX=AUX*AUX1
   10     CONTINUE

C         Valor do monomio U(i) armazenado em epsaux(i)

          EPSAUX(I)=AUX
          GKP=GKP+AUX
   20   CONTINUE

C       Projetando apenas na restricao mais violada. . .

        IF (GKP.GT.1+TOL) THEN
          IF (GKP.GT.VVL) THEN

C       Calculando epsp(i) = U(i)/gkp . . .

            DO 25 I=1,IAUX
              EPSP(I)=EPSAUX(I)/GKP
   25       CONTINUE
            KVL=K
            VVL=GKP
          ENDIF
        ENDIF
   30 CONTINUE

C       Teste de otimalidade do PGP . . .

      IF (KVL.EQ.0) THEN
        OTIMO=.TRUE.
        WRITE(6,36)ITP,IFIL,M,X(1)
        WRITE(10,36)ITP,IFIL,M,X(1)
   36   FORMAT(3X,I3,3X,I3,3X,I3,20X,F11.5)
        DO 37 J=2,N
          WRITE(6,38)X(J)
          WRITE(10,38)X(J)
   37   CONTINUE
   38   FORMAT(38X,F11.5)
        ELSE
```

```
C       Caso PGP nao seja otimo entao projetamos a rest. mais violada.

        IF (PROJET.AND.(VVL.GT.VMP)) THEN

C       Calculando atil(j). . .

          IAUX=NRHP(KVL)
          DO 110 J=1,N
            AUX=0.0
            DO 100 I=1,IAUX
              AUX=AUX+FI(KVL,I,J)*EPSP(I)
  100       CONTINUE
            ATIL(J)=AUX
  110     CONTINUE
          CALL PROJX

C       Fim do if da projecao . . .

        ENDIF

C     Fim do if de otimalidade. . .

      ENDIF
      RETURN
      END
```

rotinas do PL (ja foram vistas)

```
C           Nome do programa : PPRTOFI.FOR

C     programa geometrico posinomial

C     Projeta em todas restricoes violadas a partir do ponto obtido
C     no PL e depois condensa as restricoes no ponto projetado.

C
C           PROGRAMA PRINCIPAL DO PPRTOFI.FOR
C
      PARAMETER (NV=25,NH=50,NC=90,NT=115)
      LOGICAL TB(30),PROJET,CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /ZO/TB,PROJET
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRHP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(10,40)
      COMMON /ZF/FI(10,40,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL,ALF,VMP
      COMMON /ZI/TTOL
      COMMON /ZJ/IAE,IFIL
      COMMON /ZL/ATIL(NV)
      DIMENSION Z(NV)
      OPEN(5,FILE='PGPDA1',STATUS='OLD')
      OPEN(6,FILE='PPRTOFI.SA1',STATUS='NEW',FORM='FORMATTED')
      OPEN(7,FILE='PGPDA2',STATUS='OLD')
      OPEN(8,FILE='PGPDE1',STATUS='OLD')
      OPEN(9,FILE='PGPDE2',STATUS='OLD')
      OPEN(10,FILE='PGPTAB',STATUS='NEW',FORM='FORMATTED')
      CALL INIPGP
      WRITE(10,5)TOL,IAE
      WRITE(6,5)TOL,IAE
    5 FORMAT(20X,'Programa :PCOTOCP.FOR'/20X,'TOLERANCIA PGP = ',F8.5//,
     *20X,'EXERCICIO Numero :',I2//,2X,' ITP # ITL # N.CO # SOL. OTIMA P
     *AL # SOL. OTIMA PGP #'/)
   10 IF (.NOT.OTIMO.AND.FACTVL) THEN
        ITP=ITP+1
        ITL=0
        PARADA=.FALSE.
        SOJE=.TRUE.
   20   IF ((.NOT.PARADA).AND.FACTVL) THEN
          ITL=ITL+1
          IFIL=IFIL+1
          IF (SOJE.AND.(ITP.NE.1)) THEN
            JE=N
            CINA=.TRUE.
            AUX=0.
            DO 25 J=1,N
              AUX=AUX+A(J,N)*X(J)
   25       CONTINUE
            TETA=AUX-E(N+N)
            SOJE=.FALSE.
          ELSE
            CALL TESOTI
          ENDIF
          IF (.NOT.PARADA) THEN
            CALL ALTERA
          ENDIF
          IF ((.NOT.PARADA).AND.FACTVL) THEN
            CALL ATUALI
          ENDIF
          GOTO 20
        ENDIF
        IF (FACTVL) THEN
          DO 30 J=1,N
            Z(J)=X(J)
            X(J)=EXP(X(J))
   30     CONTINUE
          WRITE(10,45)ITP,ITL,M,X(1)
          WRITE(6,45)ITP,ITL,M,X(1)
   45     FORMAT(3X,I3,3X,I3,3X,I4,3X,F11.5)
          DO 47 J=2,N
            WRITE(10,48)X(J)
   47     CONTINUE
   48     FORMAT(22X,F11.5)
          CALL TOTPGP
          IF (.NOT.OTIMO) THEN
            DO 100 J=1,N
              X(J)=Z(J)
  100       CONTINUE
          ENDIF
        ENDIF
        GOTO 10
      ENDIF
      STOP
      END
```

```
C
C       ROTINAS DO PGP. . .
C
```

```
      SUBROUTINE CNDLIN

      PARAMETER (NV=25,NH=50,NC=90,NT=115)
      LOGICAL TB(30),PROJET,CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z0/TB,PROJET
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/N,M,JE,K2,KZAO,NZAO,NRMP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(10,40)
      COMMON /ZF/FI(10,40,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL,ALF,VMP
      COMMON /ZI/TTOL
      COMMON /ZJ/IAE,IFIL
      COMMON /ZL/ATIL(NV)
      COMMON /ZM/EPSP(40)
      IAUX=NRMP(KR)
c     DO 5 J=1,N
c       WRITE(6,8)'X(',J,')  = ',X(J)
c 5   CONTINUE
      IF ((PROJET).OR.(ITP.EQ.0)) THEN
        GK=0.0
        DO 20 I=1,IAUX
          AUX=TET(KR,I)
          DO 10 J=1,N
            AUX1=X(J)**(FI(KR,I,J))
            AUX=AUX*AUX1
   10     CONTINUE
          EPSP(I)=AUX
          GK=GK+AUX
   20   CONTINUE
c       WRITE(6,9)'VALOR DE KR ',KR,'VALOR DE GK = ',GK
        DO 30 I=1,IAUX
          EPSP(I)=EPSP(I)/GK
   30   CONTINUE
        DO 50 J=1,N
          AUX=0.0
          DO 40 I=1,IAUX
            AUX=AUX+FI(KR,I,J)*EPSP(I)
   40     CONTINUE
          A(J,KX)=AUX
   50   CONTINUE
      ELSE

c       Caso nao houve projecao . . .

        DO 52 I=1,IAUX
          EPSP(I)=EPSP(I)/VVL
   52   CONTINUE
        DO 55 J=1,N
          AUX=0.
          DO 53 I=1,IAUX
            AUX=AUX+FI(KR,I,J)*EPSP(I)
   53     CONTINUE
          A(J,KX)=AUX
   55   CONTINUE
      ENDIF
      AUX=1.0
      DO 60 I=1,IAUX
        IF (EPSP(I).LT.(1.E-07)) THEN
          AUX2=EPSP(I)**EPSP(I)
          AUX1=(TET(KR,I)**EPSP(I))/AUX2
          AUX=AUX*AUX1
        ELSE
          AUX=AUX*((TET(KR,I)/EPSP(I))**EPSP(I))
        ENDIF
   60 CONTINUE
      E(KX+N)=-ALOG(AUX)
      D(KX+N)=-1.0E+04
      RETURN
      END
```

```
      SUBROUTINE PROJX

      PARAMETER (NV=25,NH=50,NC=90,NT=115)
      LOGICAL TB(30),PROJET,CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z0/TB,PROJET
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRMP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(10,40)
      COMMON /ZF/FI(10,40,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL,ALF,VMP
      COMMON /ZI/TTOL
      COMMON /ZJ/IAE,IFIL
      COMMON /ZL/ATIL(NV)
      ATINOD=0.0

c     Calculo da norma de atil(j) (atinod) . . .

      DO 10 J=1,N
        ATINOD=ATINOD+ATIL(J)*ATIL(J)
   10 CONTINUE
      DO 20 J=1,N
        EXPTIL=-ALFA(ATIL(J)/ATINOD)
        X(J)=X(J)*VVL**EXPTIL
   20 CONTINUE

c       terminado a projecao (apenas uma iteracao) sobre a
c       restricao k, projetamos agora sobre a regiao definida
c       pelos limitantes inferiores e superiores ( P/Q )

      DO 25 J = 1,N
        IF ( X(J) .LT. XLI(J) ) THEN
          X(J) = XLI(J)
        ELSE
          IF ( X(J) .GT. XLS(J) ) THEN
            X(J) = XLS(J)
          ENDIF
        ENDIF
   25 CONTINUE

c       fim da projecao

      RETURN
      END
```

```
      SUBROUTINE TSTPC

      PARAMETER (NV=25,NH=50,NC=90,NT=115)
      LOGICAL TB(30),PROJET,CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z0/TB,PROJET
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE
      COMMON /Z2/M,N,JE,K2,KZAO,NZAO,NRMP(NH)
      COMMON /Z3/KX,KR,KVL,ITP,IB(NV)
      COMMON /Z4/IN(NC)
      COMMON /Z5/C(NV)
      COMMON /Z6/A(NV,NC)
      COMMON /Z7/D(NT)
      COMMON /Z8/E(NT)
      COMMON /Z9/ABI(NV,NV)
      COMMON /ZA/X(NV)
      COMMON /ZB/UB(NV)
      COMMON /ZC/ACH(NV)
      COMMON /ZD/TOL,TETA
      COMMON /ZE/TET(10,40)
      COMMON /ZF/FI(10,40,NV)
      COMMON /ZG/XLI(NV),XLS(NV)
      COMMON /ZH/VVL,ALF,VMP
      COMMON /ZI/TTOL
      COMMON /ZJ/IAE,IFIL
      COMMON /ZL/ATIL(NV)
      COMMON /ZM/EPSP(40)
      DIMENSION XA(25)
      OTIMO =.TRUE.
      KAUX=KZAO+1
      VVL=1.0
      DO 5 J=1,N
        XA(J)=X(J)
    5 CONTINUE
      IFKP=0
      DO 30 K=1,KAUX
        GKP=0.0
        IAUX=NRMP(K)
        DO 20 I=1,IAUX
          AUX=TET(K,I)
          DO 10 J=1,N
            AUX=AUX*X(J)**FI(K,I,J)
   10     CONTINUE
c
c         valor do consumo u(i) armazenado em epsp(i)
c
          EPSP(I) = AUX
          GKP=GKP+AUX
   20   CONTINUE

c       Projetando em todas restricoes violadas . . .

        IF (GKP.GT.(1.0+TOL)) THEN
          OTIMO = .FALSE.
          VVL=GKP

c         Verifica se projeta a restricao Gk no ponto x . . .

          IF ((ALF.EQ.1.).AND.(VVL.GT.VMP)) THEN
            PROJET=.TRUE.

c         calculando epsp(i) = u(i)/gkp

            DO 22 I = 1,IAUX
              EPSP(I) = EPSP(I)/VVL
   22       CONTINUE
c
c         calculando atil(j)
c
            DO 24 J = 1,N
              AUX = 0.
              DO 23 I = 1,IAUX
                AUX = AUX + FI(K,I,J)*EPSP(I)
   23         CONTINUE
              ATIL(J)=AUX
   24       CONTINUE
            CALL PROJX
          ELSE
            PROJET=.FALSE.
          ENDIF
c
c         Condensa a restricao violada no ponto projetado
c
          M=M+1
          IFKP=IFKP+1
          KR=K
          KX=M
          CALL CNDLIN
          IN(M)=M+N
          DO 28 J=1,N
            X(J)=XA(J)
   28     CONTINUE
        ENDIF
   30 CONTINUE
      IF (OTIMO) THEN
        WRITE(10,32)ITP,IFIL,N,X(1)
        WRITE(6,32)ITP,IFIL,N,X(1)
   32   FORMAT(3X,I3,3X,I3,3X,I3,20X,F11.5)
        DO 33 J=2,N
         WRITE(6,34)X(J)
         WRITE(10,34)X(J)
   33   CONTINUE
   34   FORMAT(38X,F11.5)
      ENDIF
      RETURN
      END

C
C         ROTINAS DO PL ...
C
```

```
C  *                  BOLSA DE MESTRADO EM COMPUTACAO  -  FAPESP
C  *
C  *                     PROGRAMA NUMERO 2
C  *
C  *  MINIMIZAR  .....
C  *
C  *              MINIMIZAR  X(0)
C  *              SUJEITO A  Pk(X)-Qk(X)<=0  k=0,....,NZAO ( a )
C  *                         0<XLI(J)<=X(J)<=XLS(J)   J=0,....,NZAO
C  *
C  *         ONDE  Pk(X) E Qk(X) SAO POSINOMIOS.
C  *
C  *  PROJETO : PROGRAMACAO GEOMETRICA GENERALIZADA
C  *
C  *  PROCESSO : 85/0630-1
C  *
C  *  INSTITUICAO : INSTITUTO DE CIENCIAS MATEMATICAS DE SAO CARLOS-USP
C  *
C  *  ORIENTADOR : PROF. DR.  MARCOS NEREU ARENALES
C  *
C  *  BOLSISTA : EDMUNDO SERGIO SPOTO
C  *
C  ***************************************************************

C                       DESCRICAO DAS VARIAVEIS
C
C  V01) AP(k,i,j) : EXPOENTE DA VARIAVEL X(j) DO MONOMIO i NA RESTRICAO k DO
C                   POSINOMIO Pk
C
C  V02) BQ(k,i,j) : EXPOENTE DA VARIAVEL X(j) DO MONOMIO i NA RESTRICAO k DO
C                   POSINOMIO Qk
C
C  V03) CP(k,i) : COEFICIENTE DO MONOMIO i NA RESTRICAO k DO POSINOMIO Pk
C
C  V04) BQ(k,i) : COEFICIENTE DO MONOMIO i NA RESTRICAO k DO POSINOMIO Qk
C
C  V05) AUXF(k,j) : AUXILIAR DE OPERACAO PARA A CONDENSACAO DE 1+Qk
C
C  V06) EPSQ(k,i+1) : EXPOENTE DA DESIGUALDADE ARITMETICA-GEOMETRICA
C
C  V07) ITO : CONTADOR DE ITERACOES DO PGG
C
C  V08) FIQ(k,j) : AUXILIAR PARA OPERACAO NA MONTAGEM DO FI PARA O POP,USADO
C                  NA SUBROTINA CONDV
C
C  V09) NRMQ(k) : NUMERO DE MONOMIOS DA RESTRICAO k DO POSINOMIO Qk
C
C  V10) TOLG : TOLERANCIA PARA VERIFICAR SE SATISFAZ KUHN-TUCKER
C
C                       DESCRICAO DAS ROTINAS
C
C  R01) CONDV
C         FUNCAO : CONDENSA O POSINOMIO 1+Pk(X) E FAZ A DIVISAO DO POSINOMIO
C                  Pk(X) PELO MONOMIO GERADO Q(X,XI), SEM COMO MONTA O POP.
C         ENTRADA : AP(k,i,j), BQ(k,i,j),CP(k,i),DQ(k,i),NRMQ(k),NRMP(k),XO(j)
C
C                   MODIFICACOES : AUXF(k,j),EPSQ(k,i+1),FIQ(k,i,j),FIO(k,j),ITETS,ITO
C
C  R02) INIPGG
C         FUNCAO : LE OS DADOS DE ENTRADA, INICIALIZA AS VARIAVEIS PARA O PGG
C         ENTRADA:ITO,TOLG,N,M,XO(j)(PONTO INICIAL) E OS DAS ROTINAS LERPGG
C         MODIFICACOES : ITO,TOLG,N,M,X(j) E AS MODIFICADAS PELAS ROTINAS
C                        LERPGG
C
C  R03) LERPGG
C         FUNCAO : LE OS DADOS DE ENTRADA DO PGG
C         ENTRADA : KZAO,NZAO,NRMP(k),NRMQ(k),CP(k,i),DQ(k,i),AP(k,i,j),
C                   BQ(k,i,j),XLI(j),XLS(j)
C         MODIFICACOES : KZAO,NZAO,NRMP(k),NRMQ(k),CP(k,i),DQ(k,i),AP(k,i,j),
C                        BQ(k,i,j),XLI(j),XLS(j),IAUXP,IAUXQ,KAUX,JAUX
C
C  R04) RESPGP
C         FUNCAO : RESOLVE O POP QUE FOI MONTADO NA RESTRICAO CONDV
C         ENTRADA : KZAO,NZAO,NRMP(k),TET(k,i),FI(k,i,j),XLI(j),XLS(j),X(j)
C                   E OS DAS ROTINAS INIPOP,TESOTI, ALTERA, ATUALI, TOIPOP
C                   E CONDLIN
C         MODIFICACOES : IIER,PARADA,SOJE,X(j),M,KK,KR, E  AS  MODIFICACOES
C                        PELAS ROTINAS INIPOP,TESOTI,ALTERA,ATUALI,TOIPOP E
C                        CONDLIN
C
C                       OUTRAS INFORMACOES
C
C  ARQUIVO DE ENTRADA : PGGDAD
C
C  FORMATO DE ENTRADA DOS DADOS :
C       KZAO,NZAO
C       NRMP(k)
C       NRMQ(k)
C     CONJUNTO DE COEFICIENTES DAS FUNCOES Pk E Qk INTERCALADOS POR LINHAS
C     SENDO UMA RESTRICAO POR LINHA NA ORDEM:
C       CP(k,i)
C       DQ(k,i)
C     CONJUNTO DE EXPOENTES DOS MONOMIOS DA FUNCAO Pk UM MONOMIO POR LINHA
C       AP(k,i,j)
C     CONJUNTO DE EXPOENTES DOS MONOMIOS DA FUNCAO Qk UM MONOMIO POR LINHA
C       BQ(k,i,j)
C       XLI(j)
C       XLS(j)
C
C  ARQUIVO DE SAIDA : PGGSAI
C
C  LINGUAGEM DE CODIFICACAO : FORTRAN 77
C
C  EQUIPAMENTO : XT2002
C
C  OBSERVACOES :
C
C     PARA A RESOLUCAO DO PGP FORAM UTILIZADAS ROTINAS DO PROGRAMA NUMERO 1.
C     SAO ELAS:INIPOP,CONDLIN,LERPGP,TOIPGP,INICIA, TESOTI, ALTERA E ATUALI,
C              COM AS SEGUINTES MODIFICACOES:
C        1) FOI ALTERADO LERPGP PARA LERPGG, ONDE OS DADOS ENTRAM NO FORMATO
C           PROPRIO PARA O PGG.
C
C        2) INIPGP APENAS INICIALIZA AS VARIAVEIS PARA O USO DO  PGP  E  NAO
C           CHAMA MAIS A ROTINA LERPGP, POIS OS DADOS QUE ERAM LIDOS NO  PGP
C           ENTRAM PELA ROTINA CONDV.
C
C        3) O PROGRAMA PRINCIPAL DO PGP FOI TRANSFORMADO EM SUBROTINA DO PGG.
C
C        4) A SUBROTINA PROJEX NAO FOI INSERIDA DEFINITIVAMENTE NESTE PROGRA-
C           MA POR ESTAR SENDO TESTADA AINDA NO PGP.
C
C           APENAS LEMBRANDO QUE O PGP E RESOLVIDO INDEPENDENTEMENTE  DO  PGG.
C     E APOS A OBTENCAO DO XI(j)(PONTO OTIMO) NO PGP, RETORNAMOS AO PGG ONDE FA-
C     REMOS A VERIFICACAO SE SATISFAZ K.T. E CASO O PONTO AINDA NAO FOR OTIMO NO
C     POU RETORNAMOS AO POP ONDE TODAS AS VARIAVEIS DE CONTROLE DO PGP SE INICI-
C     ALIZARAM NOVAMENTE E O PONTO INICIAL PARA O PGP SERA O  PONTO  ENCONTRADO
C     NO POP NA ITERACAO ANTERIOR DO PGG.
C
C     A DOCUMENTACAO DAS VARIAVEIS E ROTINAS DO PROGRAMA NUMERO 1, AQUI UTILIZA-
C     DAS, NAO E FEITA NESTE.
C
C       COM RELACAO A LOGICA DAS ROTINAS, FORAM EMITIDOS DOCUMENTACAO ANEXA AO
C     RELATORIO ANTERIOR.
```

```
C     SUBROUTINE INIPGG
C
      PARAMETER (NP=90,MP=115)
      COMMON /Z1/M,N,JE,K2,KZAO,NZAO,NRMP(50)
      COMMON /Z2/KX,KR,KVL,ITP,IB(25)
      COMMON /ZN/IN(NP)
      COMMON /Z3/C(25)
      COMMON /ZA/A(25,NP)
      COMMON /ZD/D(NP)
      COMMON /ZE/E(NP)
      COMMON /ZR/ABI(25,25)
      COMMON /ZX/X(25)
      COMMON /ZU/UB(25)
      COMMON /Z/ACH(25)
      COMMON /ZS/TOL,TETA
      COMMON /ZT/TET(15,40)
      COMMON /ZF/FI(15,40,25)
      COMMON /ZLS/XLI(25),XLS(25)
      COMMON /ZV/VVL,ALF
      COMMON /T/NRMQ(50),CP(15,40),DQ(15,40),AP(15,40,25),BQ(15,40,25)
      COMMON /U/TOLG,EPSP(40)
      COMMON /V/XO(25)
      COMMON /TT/ITTP,ITTL,ITC
      ITC=0
      ITTP=0
      ITTL=0
      CALL LERPGG
      WRITE(*,*)'FORNECER O PONTO INICIAL'
      WRITE(6,*)'PONTO INICIAL NO PGG'
      DO 10 I=1,N
        WRITE(*,*)'X(',I,') ='
        READ(*,*)X(I)
        WRITE(6,*)'X(',I,') =',X(I)
 10   CONTINUE
      WRITE(*,*)'FORNECER O VALOR DA TOLERANCIA NO PGG (TOLG=)'
      READ(*,*)TOLG
      DO 20 I=1,N
        XO(I)=X(I)
 20   CONTINUE
      RETURN
      END
C
C     SUBROUTINE LERPGG
C
      PARAMETER (NP=90,MP=115)
      COMMON /Z1/M,N,JE,K2,KZAO,NZAO,NRMP(50)
      COMMON /Z2/KX,KR,KVL,ITP,IB(25)
      COMMON /ZN/IN(NP)
      COMMON /Z3/C(25)
      COMMON /ZA/A(25,NP)
      COMMON /ZD/D(NP)
      COMMON /ZE/E(NP)
      COMMON /ZR/ABI(25,25)
      COMMON /ZX/X(25)
      COMMON /ZU/UB(25)
      COMMON /Z/ACH(25)
      COMMON /ZS/TOL,TETA
      COMMON /ZT/TET(15,40)
      COMMON /ZF/FI(15,40,25)
      COMMON /ZLS/XLI(25),XLS(25)
      COMMON /ZV/VVL,ALF
      COMMON /T/NRMQ(50),CP(15,40),DQ(15,40),AP(15,40,25),BQ(15,40,
      COMMON /V/XO(25)
      READ(5,*)KZAO,NZAO
      N=KZAO+1
      M=NZAO+1
      READ(5,*)(NRMP(K),K=1,N)
      READ(5,*)(NRMQ(K),K=1,N)
      DO 10 K=1,N
        IAUXP=NRMP(K)
        IAUXQ=NRMQ(K)
        IF (IAUXP.NE.0.) THEN
          READ(5,*)(CP(K,I),I=1,IAUXP)
        ENDIF
        IF (IAUXQ.NE.0.) THEN
          READ(5,*)(DQ(K,I),I=1,IAUXQ)
        ENDIF
 10   CONTINUE
      DO 40 K=1,N
        IAUXP=NRMP(K)
        IAUXQ=NRMQ(K)
        IF (IAUXP.NE.0.) THEN
          DO 20 I=1,IAUXP
            READ(5,*)(AP(K,I,J),J=1,N)
 20       CONTINUE
        ENDIF
        IF (IAUXQ.NE.0.) THEN
          DO 30 I=1,IAUXQ
            READ(5,*)(BQ(K,I,J),J=1,N)
 30       CONTINUE
        ENDIF
 40   CONTINUE
      READ(5,*)(XLI(J),J=1,N)
      READ(5,*)(XLS(J),J=1,N)
      WRITE(*,50)
 50   FORMAT(2X,'FORNECER O VALOR DE ALFA PARA A PROJECAO (0. OU 1.
      READ(*,*)ALF
      WRITE(6,60)ALF
 60   FORMAT(10X,'O VALOR DE ALFA = ',F8.3)
      RETURN
      END
```

```
C     SUBROUTINE CONDV
C
      PARAMETER (NP=90,MP=115)
      LOGICAL TB(90),CIMA,PARADA,OTIMO,FACTVL,SOJE,PROJET
      COMMON /ZO/TB
      COMMON /Z1/CIMA,PARADA,OTIMO,FACTVL,SOJE,PROJET
      COMMON /Z1/M,N,JE,K2,KZAO,NZAO,NRMP(50)
      COMMON /Z2/KX,KR,KVL,ITP,IB(25)
      COMMON /ZN/IN(NP)
      COMMON /Z3/C(25)
      COMMON /ZA/A(25,MP)
      COMMON /ZD/D(NP)
      COMMON /ZE/E(NP)
      COMMON /ZR/ABI(25,25)
      COMMON /ZX/X(25)
      COMMON /ZU/UB(25)
      COMMON /Z/ACH(25)
      COMMON /ZS/TOL,TETA
      COMMON /ZT/TET(15,40)
      COMMON /ZF/FI(15,40,25)
      COMMON /ZLS/XLI(25),XLS(25)
      COMMON /ZV/VVL,ALF
      COMMON /T/NRMQ(50),CP(15,40),DQ(15,40),AP(15,40,25),BQ(15,40,25)
      COMMON /V/XO(25)
      DIMENSION EPSQ(15,26),FIQ(15,25),AUXF(15,25)
      KAUX=KZAO+1
```

```
      DO 60 K=1,KAUX
        QK=0.0
        IAUXQ=NRNQ(K)
        IF (IAUXQ.EQ.0.) THEN
          AUX=1.0
          DO 5 J=1,N
            FIQ(K,J)=0.
    5     CONTINUE
        ELSE
          EPSQ(K,1)=1.0
          IAUXP=NRMP(K)
          DO 20 I=1,IAUXQ
            AUX=DQ(K,I)
            DO 10 J=1,N
              FIQ(K,J)=0.0
              AUX=AUX*XO(J)**BQ(K,I,J)
   10       CONTINUE
            IQ=I+1
            EPSQ(K,IQ)=AUX
            QK=QK+AUX
   20     CONTINUE
          QK=QK+1
          EPSQ(K,1)=EPSQ(K,1)/QK
          DO 30 I=1,IAUXQ
            IQ=I+1
            EPSQ(K,IQ)=EPSQ(K,IQ)/QK
   30     CONTINUE
          AUX=1.0
          AUX=(AUX/EPSQ(K,1))**EPSQ(K,1)
          DO 40 I=1,IAUXQ
            IQ=I+1
            AUX=AUX*((DQ(K,I)/EPSQ(K,IQ))**EPSQ(K,IQ))
            DO 35 J=1,N
              AUXF(K,J)=BQ(K,I,J)*EPSQ(K,IQ)
              FIQ(K,J)=FIQ(K,J)+AUXF(K,J)
   35       CONTINUE
   40     CONTINUE
        ENDIF
        DO 50 I=1,IAUXP
          TET(K,I)=CP(K,I)/AUX
          DO 45 J=1,N
            FI(K,I,J)=AP(K,I,J)-FIQ(K,J)
   45     CONTINUE
   50   CONTINUE
   60 CONTINUE
      DO 80 K=1,KAUX
        IAUXP=NRMP(K)
        WRITE(8,8)'VALORES DE TET PARA RESTRICAO ',K
        WRITE(8,8)(TET(K,I),I=1,IAUXP)
        WRITE(8,8)'VALORES DE FI'
      DO 70 I=1,IAUXP
        WRITE(8,8)(FI(K,I,J),J=1,N)
   70 CONTINUE
   80 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RESPGP
C
      PARAMETER (NP=90,MP=115)
      LOGICAL TB(30),CIMA,PARADA,OTIMO,FACTVL,SOJE,PROJET
      COMMON /Z0/TB
      COMMON /Z1/CIMA,PARADA,OTIMO,FACTVL,SOJE,PROJET
      COMMON /Z1/M,N,JE,K2,KZAO,NZAO,NRMP(50)
      COMMON /Z2/KX,KR,KVL,ITP,IB(25)
      COMMON /ZN/IN(NP)
      COMMON /Z3/C(25)
      COMMON /ZA/A(25,NP)
      COMMON /ZD/D(MP)
      COMMON /ZE/E(MP)
      COMMON /ZR/AB(25,25)
      COMMON /ZX/X(25)
      COMMON /ZU/UB(25)
      COMMON /Z/ACH(25)
      COMMON /ZS/TOL,TETA
      COMMON /ZT/TET(15,40)
      COMMON /ZF/F1(15,40,25)
      COMMON /ZLS/XLI(25),XLS(25)
      COMMON /ZV/VVL,ALF
      COMMON /Z9/ATIL(25)
      COMMON /Z7/TTOL
      COMMON /TT/ITTP,ITTL,ITG
      DIMENSION Z(25)
      CALL INIPGP
   10 IF (.NOT.OTIMO.AND.FACTVL) THEN
        ITP=ITP+1
        ITTP=ITTP+1
        ITL=0
        WRITE(*,*)'ITERACAO PGP = ',ITP
        WRITE(6,*)'ITERACAO PGP = ',ITP
        WRITE(6,15)M
   15   FORMAT(20X,'NUMERO DE RESTRICOES NO PL =',I3)
        PARADA=.FALSE.
        SOJE=.TRUE.
   20   IF ((.NOT.PARADA).AND.FACTVL) THEN
          ITL=ITL+1
          ITTL=ITTL+1
          IF (SOJE.AND.(ITP.NE.1)) THEN
            JE=M
            CIMA=.TRUE.
            AUX=0.
            DO 25 J=1,N
              AUX=AUX+A(J,M)*X(J)
   25       CONTINUE
            TETA=AUX-E(M+N)
            SOJE=.FALSE.
          ELSE
            CALL TESOTI
          ENDIF
          IF (.NOT.PARADA) THEN
            CALL ALTERA
          ENDIF
          IF ((.NOT.PARADA).AND.FACTVL) THEN
            CALL ATUALI
          ENDIF
          GOTO 20
```

```
        ENDIF
        IF (FACTVL) THEN
          DO 30 J=1,N
            Z(J)=X(J)
            X(J)=EXP(X(J))
   30     CONTINUE
          WRITE(6,35)ITL
          WRITE(7,36)ITG,ITP,ITL,X(1)
   35     FORMAT(20X,'TOTAL DE ITERACOES PL =',I4)
   36     FORMAT(2X,I3,2(3X,I3),3X,F11.5)
          DO 37 J=2,N
          WRITE(7,38)X(J)
   37     CONTINUE
   38     FORMAT(20X,F11.5)
          WRITE(6,40)
   40     FORMAT(20X,'SOLUCAO OTIMA DO PL')
          WRITE(6,50)
   50     FORMAT(20X,'====================',/)
          WRITE(6,60)
   60     FORMAT(20X,'++++++++++++++++++++++++++++++')
          WRITE(6,70)
   70     FORMAT(20X,'+',27X,'+')
          DO 80 J=1,N
            L=J-1
            WRITE(6,90)L,X(J)
   80     CONTINUE
   90     FORMAT(20X,'+  X(',I2,') =',E15.6,3X,'+')
          WRITE(6,70)
          WRITE(6,60)
          CALL TOTPGP
          IF (.NOT.OTIMO) THEN
            DO 100 J=1,N
              X(J)=Z(J)
  100       CONTINUE
          ENDIF
        ENDIF
        GOTO 10
      ENDIF
      RETURN
      END

C
C     ROTINAS DO PGP QUE CONDENSA E PROJETA EM TODAS RESTRICOES
C     VIOLADAS...
C
```

```
      nome do programa : PGGPRCM.FOR
        projeto e condensa na mais violada para alf=1.
        condensa na mais violada,s/proj para alf=0.


      PARAMETER (NP=90,MP=115)
      LOGICAL TB(30),CIMA,PARADA,OTIMO,FACTVL,SOJE,PROJET
      COMMON /Z0/TB
      COMMON /Z1/CIMA,PARADA,OTIMO,FACTVL,SOJE,PROJET
      COMMON /Z1/M,N,JE,K2,KZAO,NZAO,NRMP(50)
      COMMON /Z2/KX,KR,KVL,ITP,IB(25)
      COMMON /ZN/IN(NP)
      COMMON /Z3/C(25)
      COMMON /ZA/A(25,NP)
      COMMON /ZD/D(MP)
      COMMON /ZE/E(MP)
      COMMON /ZR/AB(25,25)
      COMMON /ZX/X(25)
      COMMON /ZU/UB(25)
      COMMON /Z/ACH(25)
      COMMON /ZS/TOL,TETA
      COMMON /ZT/TET(15,40)
      COMMON /ZF/F1(15,40,25)
      COMMON /ZLS/XLI(25),XLS(25)
      COMMON /ZV/VVL,ALF
      COMMON /Z9/ATIL(25)
      COMMON /Z7/TTOL
      COMMON /T/NRMQ(50),CP(15,40),DQ(15,40),AP(15,40,25),BQ(15,40,25)
      COMMON /U/ITG,TOLG,EPSP(40)
      COMMON /V/XO(25)
      COMMON /TT/ITTP,ITTL
      OPEN(5,FILE='PGGDAD',STATUS='OLD')
      OPEN(6,FILE='PGGSAI',STATUS='NEW')
      OPEN(7,FILE='PGGTAB',STATUS='NEW')
      CALL INIPGG
      WRITE(7,3)ALF,TOLG
    3 FORMAT(20X,'ALFA = ',F7.4//,20X,'TOLER.PGG =',F7.5/)
      IF (ALF.NE.0.) THEN
        PROJET=.TRUE.
       ELSE
        PROJET=.FALSE.
      ENDIF
      WRITE(7,5)
    5 FORMAT(2X,'ITG * ITP * ITL * SOL.FIN.PL * SOL.FIN.PGP * SOL.FIN.PG
     *G *'//)
   10 CALL CONDV
      CALL RESPGP
      DO 30 J=1,N
        IF (ABS(X(J)-XO(J)).GT.TOLG) THEN
          DO 20 JS=1,N
            XO(JS)=X(JS)
   20     CONTINUE
          ITG=ITG+1
          GOTO 10
        ENDIF
   30 CONTINUE
      WRITE(6,40)
   40 FORMAT(20X,'SOLUCAO DO PGG QUE VERIFICA AS CONDICOES DE K.T.')
      WRITE(6,50)
   50 FORMAT(20X,'======== == === === ========= == ========== == ====='/)
      WRITE(6,60)
   60 FORMAT(20X,'++++++++++++++++++++++++++++++')
```

```
      WRITE(6,70)
 70   FORMAT(20X,'+',27X,'+')
      WRITE(7,73)
 73   FORMAT(1X/,2X,'ITG * ITTP* ITTL *   *** RESULTADOS FINAIS ***
     *OL.PGG'/)
      WRITE(7,75)ITG,ITTP,ITTL,X(1)
 75   FORMAT(2X,I3,2(3X,I3),3IX,F11.5)
      DO 77 J=2,N
        WRITE(7,78)X(J)
 77   CONTINUE
 78   FORMAT(48X,F11.5)
      DO 80 J=1,N
        L=J-1
        WRITE(6,90)L,X(J)
 80   CONTINUE
 90   FORMAT(20X,'+  X(',I2,') =',E15.6,3X,'+')
      WRITE(6,70)
      WRITE(6,60)
      WRITE(6,100)ITG
100   FORMAT(20X,'NUMERO DE ITERACOES NO PGG = ',I6)
      STOP
      END
C
C
C           ROTINAS DO PGG EM GERAL...
C
C           ROTINAS DO PGP QUE CONDENSA E PROJETA NA RESTRICAO
C           MAIS VIOLADA...
C
```

```
C
C              nome do programa : PGGPCT.FOR
C                projeta e condensa as todas restricoes violadas alf=1.
C                condensa todas restricoes violadas sem projecao alf=0.
C
      PARAMETER (NP=90,MP=115)
      LOGICAL TB(30),CINA,PARADA,OTIMO,FACTVL,SOJE,PROJET
      COMMON /ZO/TB
      COMMON /Z1/CINA,PARADA,OTIMO,FACTVL,SOJE,PROJET
      COMMON /Z1/M,N,JE,K2,KZAO,NZAO,NRHP(50)
      COMMON /Z2/EX,KR,KVL,ITP,IB(25)
      COMMON /ZH/IH(MP)
      COMMON /Z3/C(25)
      COMMON /ZA/A(25,MP)
      COMMON /ZD/D(MP)
      COMMON /ZE/E(MP)
      COMMON /ZL/AI(25,25)
      COMMON /ZX/X(25)
      COMMON /ZH/H(25)
      COMMON /Z/ACK(25)
      COMMON /ZS/TOL,TETA
      COMMON /ZT/TET(15,40)
      COMMON /ZF/FI(15,40,25)
      COMMON /ZLS/XLI(25),XLS(25)
      COMMON /ZV/VV,.ALF
      COMMON /Z3/ATIL(25)
      COMMON /Z7/TTOL
      COMMON /T/RHSQ(50),CP(15,40),DQ(15,40),AP(15,40,25),BQ(15,40,25)
      COMMON /U/TOL2,EPSP(40)
      COMMON /V/XO(25)
      COMMON /TT/ITTP,ITTL,ITG
      OPEN(5,FILE='PGGDAD',STATUS='OLD')
      OPEN(6,FILE='PGGSAI',STATUS='NEW')
      OPEN(7,FILE='PGGTAB',STATUS='NEW')
      CALL INIPGG
      IF (ALF.EQ.0.) THEN
        PROJET=.FALSE.
       ELSE
        PROJET=.TRUE.
      ENDIF
      WRITE(7,3)
   3  FORMAT(30X,'Nome do programa : PGGPCT.FOR'/)
      WRITE(6,4)
   4  FORMAT(20X,'Este programa resolve um pgg, projetando e condensando
     &',20x,'todas violadas, mas partindo sempre no ponto do PL.'/)
      WRITE(7,5)
   5  FORMAT(2X,'ITG * ITP * ITL * SOL.FIN.PL * SOL.FIN.PGP * SOL.FIN.PG
     *G *'//)
  10  CALL CONDV
      CALL RESPGP
      DO 30 J=1,N
        IF (ABS(X(J)-XO(J)).GT.TOLG) THEN
          DO 20 JS=1,N
            XO(JS)=X(JS)
  20      CONTINUE
          ITG=ITG+1
          GOTO 10
        ENDIF
  30  CONTINUE
      WRITE(6,40)
  40  FORMAT(20X,'SOLUCAO DO PGG QUE VERIFICA AS CONDICOES DE K.T.')
      WRITE(6,50)
  50  FORMAT(20X,'--------- -- --- --- --------- -- ---------- -- -----'/)
      WRITE(6,60)
  60  FORMAT(20X,'+++++++++++++++++++++++++++++++')
      WRITE(6,70)
  70  FORMAT(20X,'+',27X,'+')
      WRITE(7,73)
  73  FORMAT(1X/,2X,'ITG * ITTP* ITTL *   *** RESULTADOS FINAIS ***    S
     *OL.PGG'/)
      WRITE(7,75)ITG,ITTP,ITTL,X(1)
  75  FORMAT(2X,I3,2(3X,I3),3IX,F11.5)
      DO 77 J=2,N
        WRITE(7,78)X(J)
  77  CONTINUE
  78  FORMAT(48X,F11.5)
      DO 80 J=1,N
        L=J-1
        WRITE(6,90)L,X(J)
  80  CONTINUE
  90  FORMAT(20X,'+  X(',I2,') =',E15.6,3X,'+')
      WRITE(6,70)
      WRITE(6,60)
      WRITE(6,100)ITG
 100  FORMAT(20X,'NUMERO DE ITERACOES NO PGG = ',I6)
      STOP
      END
C
C
```

```
C
C       ROTINAS EM COMUM DO PGG...  e  PGP, PL
C
```