VISUALIZAÇÃO GRÁFICA DE ALGORITMOS DE PONTOS INTERIORES

Sérgio Henrique Monteiro da Silva

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Clóvis Caesar Gonzaga, D. Sc.
(presidente)

Prof. Sérgio Granville, Ph. D.

Prof. Paulo Roberto Oliveira, D. Ing.

RIO DE JANEIRO, RJ – BRASIL
NOVEMBRO DE 1991

SILVA, SÉRGIO HENRIQUE MONTEIRO DA

Visualização Gráfica de Algoritmos de Pontos Interiores [Rio de Janeiro] 1991

VIII, 68 p., 29.7 cm, (COPPE/UFRJ, M. Sc., ENGENHARIA DE SISTEMAS E COMPUTAÇÃO, 1991)

TESE – Universidade Federal do Rio de Janeiro, COPPE

1 – Programação Linear 2 – Algoritmos de Trajetória Central

I. COPPE/UFRJ II. Título(Série).

Aos meus pais

Agradecimentos

Agradeço em particular ao professor Clóvis C. Gonzaga pela eficiente orientação, dedicação e incentivo demonstrados ao longo de todo o desenvolvimento do trabalho.

Aos colegas que frequentaram o Laboratório de Sistemas e demais colegas de Otimização pela ajuda e companheirismo durante o desenvolvimento deste trabalho.

Agradeço também aos funcionários técnicos e administrativos do Programa de Engenharia de Sistemas e Computação pela atenção e prestatividade.

Resumo da Tese apresentada à COPPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.)

Visualização Gráfica de Algoritmos de Pontos Interiores

Sérgio Henrique Monteiro da Silva

Novembro de 1991

Orientador: Clóvis Caesar Gonzaga
Programa: Engenharia de Sistemas e Computação

Neste trabalho estudamos a visualização Gráfica dos Algoritmos de Pontos Interiores para Programação Linear. Algoritmos de Pontos Interiores são algoritmos que resolvem o problema de programação linear evoluindo no interior relativo do conjunto viável. Os algoritmos implementados para a visualização gráfica foram : Afim Escala com Passo Unitário e Busca Linear, Karmarkar, Barnes & Jensen, Renegar e Trajetória Central. Os algoritmos foram visualizados para diversos problemas de programação linear gerados graficamente ou pré-definidos. A implementação desses algoritmos deu origem a um programa interativo, que além de visualizar as trajetórias dos referidos algoritmos, permite visualizar as curvas de nível da função barreira.

Palavras-chave: Algoritmos de Pontos Interiores, Programação Linear.

Abstract of Thesis presented to COPPE as partial fulfillment of the requirements for the degree of Master of Science (M. Sc.)

Graphical Visualization of Interior Point Algorithms

Sérgio Henrique Monteiro da Silva

July, 1991

Thesis Supervisor: Clóvis Caesar Gonzaga
Department: Programa de Engenharia de Sistemas e Computação

In this work, we study the graphical visualization of Interior Point Algorithms for Linear Problems. Interior Point Algorithms solve a Linear Problem by evolving in the relative interior of the feasible set. The Algorithms implemented are : Scale Affine with a Unit Step, Scale Affine with Linear Search, Karmarkar, Barnes & Jensen, Renegar and Central Path. The problems used for visualization were either pre-defined or drawn graphically. An interative program resulted from the implementation of these algorithms which visualizes their path and the level curves of the barrier function.

Índice

Lista de Figuras

Capítulo I

Introdução

Nesse trabalho, desejamos visualizar graficamente o comportamento de algoritmos de pontos interiores para problemas lineares. Os métodos de pontos interiores para problemas lineares são uma descoberta recente na área de otimização, os algoritmos resolvem problemas de programação linear evoluindo no interior relativo do conjunto.

A visualização gráfica de métodos de otimização vem sendo cada vez mais utilizada por pesquisadores como uma ferramenta na elaboração de novos algoritmos e para auxiliar o pesquisador a pensar em novas idéias. Os professores a utilizam cada vez mais para que os alunos entendam os métodos de otimização de maneira mais clara. Até mesmo na matemática pura, a visualização gráfica vem sendo cada vez mais aceita como uma ferramenta auxiliar na pesquisa.

Dentre os problemas de otimização, estaremos interessados num problema em particular, o *problema de programação linear.* Esse problema pode ser enunciado da seguinte forma :

$$\begin{array}{lll} \text{(P)} & \text{minimizar} & c^T x \\ & \text{sujeito a} & \bar{A}x = \bar{b} \\ & & x \geq 0 \end{array}$$

onde $c \in \Re^n, \bar{b} \in \Re^m, \bar{A} \in \Re^{m \times n}$ é uma matriz de rank máximo e dimensão $m \times n$, $n > m$.

A região viável do problema (ou seja, a região no espaço $\Re^n$ na qual o conjunto de restrições é satisfeito) será dada por :

$$\bar{S} = \{x \in \Re^n \mid \bar{A}x = \bar{b}, x \geq 0\}.$$

Vamos sempre supor que $\bar{S}$ é limitado e com interior relativo não vazio dado por :

$$\bar{S}^0 = \{x \in \Re^n \mid \bar{A}x = \bar{b}, x > 0\}.$$

O problema de programação linear foi solucionado por Dantzig [4] que desenvolveu o método simplex, que é o algoritmo mais utilizado para a solução deste problema ainda hoje. Apesar da sua eficiência e elegância, o método simplex tem convergência exponencial no pior caso, o que foi demonstrado por Klee e Minty [21].

Em 1978, Khachiyan [19, 20] desenvolveu o método dos elipsóides, que resolveu o problema de programação linear com convergência polinomial. Isto estimulou novos estudos na área, porém as implementações deste método revelaram-se extremamente ineficientes, o que reforçou ainda mais o método simplex.

Somente em 1984, com a publicação do algoritmo de Karmarkar [17] (ver capítulo III), foi que se obteve um algoritmo que resolve o problema de programação linear com complexidade polinomial de maneira eficiente. O limite do número de iterações é $O(nL)$ e do número de operações é de $O(n^{3.5}L)$, onde L é o tamanho da entrada de dados do problema (o total do número de bits utilizado na descrição do problema).

A publicação do algoritmo de Karmarkar não foi revolucionária apenas pelo fato deste apresentar complexidade polinomial, mas também pelo fato de que esse algoritmo evolui pelo interior do conjunto viável, enquanto que o método simplex evolui pela fronteira do conjunto viável (precisamente nos vértices do conjunto viável). A partir de Karmarkar, desenvolveu-se uma famìlia de algoritmos conhecida por *métodos de pontos interiores.* Para um número muito grande de variáveis, implementações dos algoritmos de pontos interiores obtiveram resultados excelentes — ver Adler e Monteiro [1].

A visualização gráfica de algoritmos de pontos interiores é muito interessante, pois como os algoritmos evoluem pelo interior do conjunto, os gráficos resultantes das trajetórias geradas por esses algoritmos ilustram muito bem o que acontece com os diferentes métodos. A figura I.1 ilustra o que foi dito (observe a

evolução da trajetória do algoritmo no interior do conjunto).

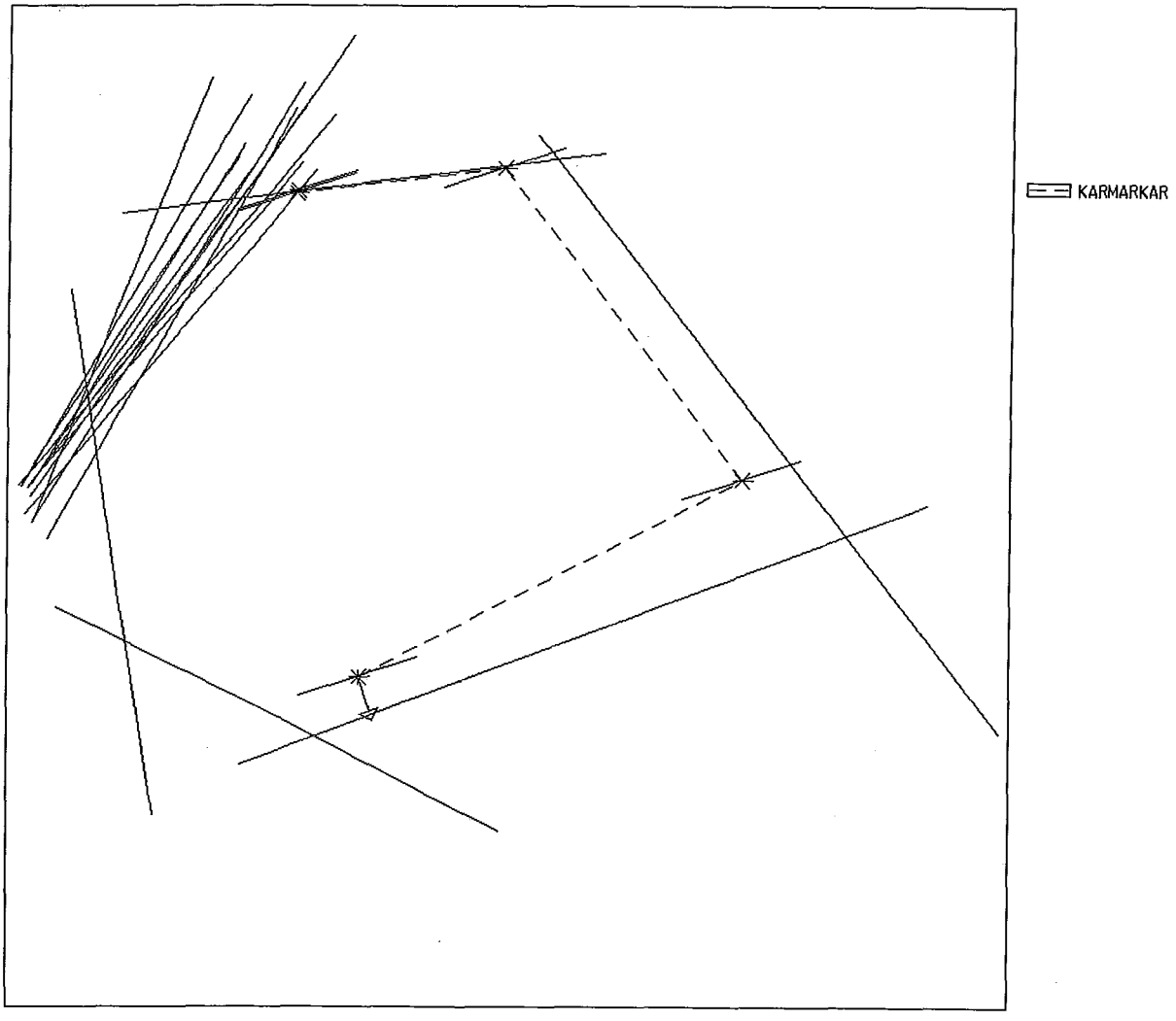


Figura I.1: Algoritmo de Karmarkar

A formulação original do algoritmo de Karmarkar necessitava do chamado simplex unitário, e partia do princípio de que se conhecia o custo de uma solução ótima ou que se existia um método eficiente de se computar limitantes inferiores ("lower bounds") para esse custo. Após vários estudos obtiveram-se algoritmos mais simples como o Afim Escala (ver capítulo II) ou o algoritmo Afim de redução potencial, que são variantes do algoritmo de Karmarkar.

Mais tarde, novos métodos surgiram e ainda mais eficientes que o algoritmo de Karmarkar, os chamados *métodos de trajetória central* — ver capítulos IV, VI, VII.

Atualmente ainda se debate muito sobre as vantagens dos algoritmos de pontos interiores sobre o método simplex, mas sem dúvida, com o renascimento dos algoritmos que trabalham no interior do conjunto, a geometria do comportamento desses algoritmos além de ser interessante e intuitiva, proporciona novas idéias de como se desenvolver algoritmos mais eficientes e que possam ser utilizados para resolver outros problemas como o problema de programação inteira e de programação não-linear.

I.1 Formato dos problemas

Vamos utilizar dois formatos especiais para o problema de programação linear :

- O *formato primal* baseado em restrições de igualdade e em variáveis não-negativas, normalmente conhecido também como formato padrão. O problema (P) definido anteriormente representa um problema nesse formato.

- O *formato dual* que se baseia em restrições de desigualdade, ou seja, um problema está no *formato dual* quando este problema é da forma :

$$\begin{array}{ll} \text{(PD)} & \text{minimizar} \quad c^T x \\ & \text{sujeito a} \quad Ax \leq b \end{array}$$

  onde $c \in \Re^n, x \in \Re^n$, $A \in \Re^{n \times m}$ é uma matriz de rank máximo e de dimensão $m \times n$. A região viável para o problema (PD) será dada por :

$$S = \{x \in \Re^n \mid Ax \leq b\}.$$

  Vamos sempre supor que S é limitado e com interior relativo não vazio dado por :

$$S^0 = \{x \in \Re^n \mid Ax < b\}.$$

Como para a visualização gráfica, o *formato dual* é mais adequado, todos os algoritmos serão dados para problemas nesse formato.

I.2 Definições

Definição I.1 *Dado um problema de programação linear no* formato primal. *Um vetor* $h \in \Re^n$ *é uma direção viável a partir de* $x \in \bar{S}$ *se e somente se existe um* $\alpha > 0$ *tal que* $x + \alpha h \in \bar{S}$.

Definição I.2 *Seja* $M \in \Re^{m \times n}$ *uma matriz de dimensão* $m \times n$, M *qualquer. O seu conjunto imagem é dado por* $\mathcal{R}(M) = \{Mx \mid x \in \Re^n\}$ *e o seu espaço nulo* $\mathcal{N}(M) = \{x \in \Re^n \mid Mx = 0\}$.

Todo vetor $d \in \Re^n$ pode ser unicamente decomposto como $d = d_p + \tilde{d}_p$, onde $d_p \in \mathcal{N}(M)$ e $\tilde{d}_p \in \mathcal{R}(M^T)$, já que $\mathcal{N}(M)$ e $\mathcal{R}(M^T)$ são espaços ortogonalmente complementares. d_p e $\tilde{d}_p$ são as projeções de d no $\mathcal{N}(M)$ e no $\mathcal{R}(M^T)$, respectivamente. A figura I.2 ilustra o vetor custo c e a sua decomposição.

Como o operador de projeção é linear, podemos representá-lo por uma matriz P_M, de modo que $d_p = P_M d$. O complemento ortogonal será dado por $\tilde{d}_p = \tilde{P}_M d$, onde $\tilde{P}_M = I - P_M$. Se M é uma matriz de rank máximo (portanto, inversível) podemos definir explicitamente P_M. Assim, temos que :

Definição I.3 *A matriz de projeção* P_M *no espaço nulo de* M *é dada por :*

$$P_M = I - M^T(MM^T)^{-1}M$$

se M *é uma matriz de rank máximo.*

I.3 Resultado Importante

Lema I.1 *Considere o problema :*

$$\begin{array}{ll} \textit{minimizar} & f(x) \\ \textit{sujeito a} & Ax = b \\ & x \in T \end{array}$$

onde $f : T \longmapsto \Re$ *é diferenciável,* $A \in \Re^{m \times n}$ *é uma matriz de dimensão* $m \times n$, $T \subset \Re^n$ *aberto.*

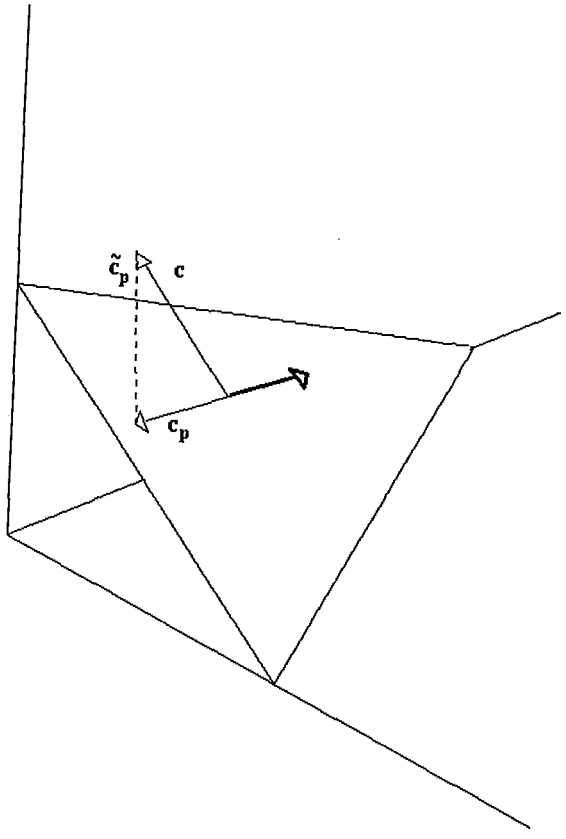


Figura I.2: Decomposição do vetor custo : $c = c_p + \tilde{c}_p$

Uma condição necessária para que $x \in T$ seja solução do problema acima é que :

$$P_A \nabla f(x) = 0, Ax = b$$

onde P_A é a matriz de projeção sobre $\mathcal{N}(A)$ definida como na definição I.3.

I.4 Regras de tradução do formato primal para o formato dual

A partir do problema (PD), podemos adicionar variáveis de folga z obtendo assim restrições de igualdade. O problema resultante pode ser simplificado chegando-se à

formulação primal correspondente ao problema (PD), ou seja :

$$\begin{array}{lll} \text{(PP)} & \text{minimizar} & \bar{c}^T z - c_0 \\ & \text{sujeito a} & \bar{A}z = \bar{b} \\ & & z \geq 0 \end{array}$$

onde $\bar{c}, z \in \Re^m, \bar{b} \in \Re^r, \bar{A} \in \Re^{r \times m}$ é uma matriz $r \times m, m \geq r$. $\bar{A}$ é uma matriz de rank máximo. Essa transformação foi feita em Gonzaga [11], e listamos a seguir os principais resultados.

A variável de folga z é dada por :

$$z = b - Ax$$

e o conjunto de variáveis de folga $\mathcal{F}$ é dado por :

$$\mathcal{F} = \{z \in \Re^m_+ \mid Ax + z = b, \text{ para algum } x \in S\}.$$

Os resultados obtidos são os seguintes :

Lema I.2 *Os espaços nulos e as projeções são relacionados por :*

$$\mathcal{R}(A) = \mathcal{N}(\bar{A}) = \mathcal{N}(P_{A^T}) \perp \mathcal{N}(A^T)$$

$$P_{\bar{A}} + P_{A^T} = I$$

Lema I.3 *Direções viáveis equivalentes h_x e h_z em S e $\mathcal{F}$ respectivamente são relacionadas por :*

$$h_z = -Ah_x$$

$$h_x = -(AA^T)^{-1}A^T h_z$$

Lema I.4 *A direção h_d em S correspondente à projeção de um vetor $d \in \Re^m$ em $\mathcal{N}(\bar{A})$ é dada por :*

$$h_d = -(AA^T)^{-1}A^T d.$$

Em particular, as direções em S correspondentes ao custo projetado $\bar{c}_p$ e o vetor $e = [1 \ldots 1]^T$ projetado e_p no espaço nulo de $\bar{A}$ são dadas por :

$$d_c = (A^T A)^{-1} c \tag{I.1}$$

$$d_e = -(A^T A)^{-1} A^T e. \tag{I.2}$$

Todas as direções utilizadas nos algoritmos são combinações lineares dessas direções.

Lema I.5 *Os resultados dos lemas anteriores continuam idênticos para problemas escalonados na folga z^k, após a seguinte transformação :*

$$A_k \leftarrow Z_k^{-1} A, b_k \leftarrow Z_k^{-1} b,$$

onde $Z_k = diag(z_1^k, \ldots, z_m^k)$.

I.5 Objetivo

O objetivo dessa tese é visualizar graficamente o comportamento dos algoritmos de pontos interiores, analizando inclusive alguns casos patológicos. Procurou-se escolher os algoritmos de pontos interiores mais representativos. Do capítulo II ao capítulo VII, cada algoritmo é analizado separadamente e em ordem cronológica, a fim de se observar também a evolução dos métodos de pontos interiores. O capítulo VIII concerne os aspectos relativos à implementação dos algoritmos.

I.6 Notação

Vamos usar vetores coluna e matriz denotados por letras minúsculas e maiúsculas, respectivamente. Os diferentes vetores serão denotados por índices superiores; os índices inferiores denotarão as componentes do vetor. As diferentes matrizes serão denotadas por índices inferiores. A letra e denotará o vetor dado por : $e = [1 \ldots 1]^T$, com dimensão explicitada no contexto.

Capítulo II

Algoritmo Afim Escala

O algoritmo afim escala para um problema de programação linear dado no *formato primal* foi o primeiro algoritmo de pontos interiores, e foi publicado por Dikin em 1967 [5]. Em 1974, Dikin [6] provou a convergência global do algoritmo afim escala com passo unitário na direção gerada pelo algoritmo. Surpreendentemente, esse algoritmo ficou esquecido até a publicação do algoritmo de Karmarkar [17]; a partir de então, como o algoritmo de Karmarkar provou-se eficiente (ver Capítulo III) vários pesquisadores passaram a reexaminar os métodos que trabalham no interior do conjunto viável (chamados de métodos de pontos interiores). O algoritmo afim escala de Dikin para problemas lineares foi um dos primeiros a serem estudados, devido à sua simplicidade e eficácia. Em 1988, Barnes[3] e Vanderbei, Meketon e Freedman [28] provaram que o algoritmo afim escala converge com a hipótese de não degeneração primal e não degeneração dual. Em 1989, Gonzaga [14] provou que o algoritmo afim escala converge com uma busca na direção gerada pelo algoritmo.

O algoritmo afim escala segue basicamente o método de Cauchy, ou seja, a direção de busca é a direção de máximo declive da função objetivo após uma mudança de escala, seguido de uma busca linear nessa direção.

II.1 O Método de Cauchy

O método de Cauchy pode ser considerado um dos primeiros métodos de otimização para problemas lineares.

Sejam $f : \Re^n \longmapsto \Re$ uma função continuamente diferenciável e $x_k \in \Re^n$ um ponto dado.

Agora, considere o problema de minimização dado a seguir :

$$\begin{array}{ll} \min & f(x) \\ x \in \Re^n & \end{array}$$

Considere a aproximação linear de f baseada na série de Taylor em torno do ponto x_k dada por :

$$f(x_k + h) \approx f(x_k) + g_k^T h$$

onde $g_k = -\nabla f(x_k), h \in \Re^n$.

Uma boa maneira de se reduzir a função objetivo f é tomar uma direção h tal que $g_k^T h$ é grande e negativo, ou seja :

$$g_k^T h < 0$$

Então, o nosso objetivo é escolher h de dentre todos os vetores normalizados de modo a minimizar $g_k^T h$. Dada uma norma $\| \cdot \|$ no $\Re^n$, $h_k \in \Re^n$ é a solução do problema de mínimo dado por :

$$\min\{g_k^T h \mid \| h \| \leq 1\}$$

Tomando-se a norma euclidiana no $\Re^n$, ou seja, $\| h \| = (h^T h)^{1/2}$ a solução do problema será dada pela direção oposta à do gradiente, então :

$$h_k = -\frac{g_k}{\| g_k \|}.$$

Observe que a normalização da direção não é importante em algoritmos que fazem buscas, e toma-se como "direção de máximo declive" simplesmente $h_k = -g_k$.

Como estamos usando uma direção de declive, ou seja, $g_k^T h_k < 0$, para garantir que um algoritmo utilizando essa direção irá convergir, é necessário

que se faça uma busca linear na direção h_k dada acima. Assim temos que, sendo $\bar{\lambda} \in \Re$:

$$\bar{\lambda} = \text{argmin}\{f(x_k + \lambda h_k) \mid \lambda \in \Re_+\}$$

com passo dado por $\bar{\lambda} h_k$. Essa busca pode em geral ser aproximada, necessitando-se de baixa precisão.

II.2 Mudança de Escala

Considere agora o problema (P) (ver capítulo I). Para se aplicar o método de Cauchy é necessário se trabalhar numa bola e para o problema (P) existe o inconviniente de que as variáveis são positivas; e também seria interessante se pudessemos escolher uma região que representasse de maneira mais adequada o formato da região viável. A região mais simples possível é o maior elipsóide simples possível no primeiro ortante. O elipsóide com eixos paralelos aos eixos coordenados (portanto simples; e observe que esse elipsóide não é influenciado pelas restrições $Ax = b$) quando interceptado com a região S é o elipsóide procurado. Esse elipsóide será a região de confiança a ser adotada quando da minimização da função custo.

Sejam um ponto $x^k \in \Re^n$ dentro da região viável S e $h \in \Re^n$; a equação do elipsóide será dada por :

$$h^T D^2 h = 1 \tag{II.1}$$

onde $D = \text{diag}(\frac{1}{x_1^k}, \frac{1}{x_2^k}, \ldots, \frac{1}{x_n^k})$.

O elipsóide está centrado no ponto x^k e tem comprimento de eixo iguais às coordenadas de x^k.

Seja $y \in \Re^n$, com a mudança de escala dada por :

$$y_i = \frac{x_i}{x_i^k}$$

ou equivalentemente :

$$x = X_k y, \text{ onde } X_k = \text{diag}(x_1^k, x_2^k, \ldots, x_n^k) \tag{II.2}$$

o elipsóide se transforma numa bola e leva do ponto x^k para o ponto $e = X_k^{-1} x^k$.

Após a mudança de escala, estaremos trabalhando numa bola e então podemos aplicar o método de Cauchy dado no item anterior. A mudança de escala como definida em (II.2) será utilizada em todos os algoritmos de pontos interiores para problemas dados no formato primal.

II.3 Direção do algoritmo afim escala para o problema (P)

Agora, já temos uma região de confiança onde minimizar a função custo (o elipsóide simples da seção anterior). O resultado dessa minimização é equivalente a uma mudança de escala como da na seção II.2, e em seguida utilizar o método de Cauchy, pois após a mudança de escala, a região de confiança passa a ser uma bola. Aplicando-se o método de Cauchy ao problema (P), obtemos a direção de busca do método afim escala para um problema no *formato primal*, ou seja, a direção de busca do método afim escala $\bar{h}_A$ a partir de um ponto $x^k \in \bar{S}^0$ é dada pela direção oposta ao gradiente da função custo projetado no espaço nulo de A após a mudança de escala como dada na seção II.2, ou seja :

$$\bar{h}_A = -X_k P_{\bar{A}X_k} X_k c \tag{II.3}$$

onde $\bar{h}_A \in \Re^n$ e X_k é a matriz mudança de escala como dada em (II.2).

Observe que a projeção no espaço nulo de $\bar{A}X_k$ é necessária devido ao resultado do Lema I.1.

II.4 Direção do algoritmo afim escala para o problema (PD)

No caso de um problema no *formato dual* a mudança de escala a partir de um ponto $x^k \in S^0$ é dada por :

- Cálculo das Folgas : $z^k \in \Re^m, z^k = b^k - Ax^k$;

- Mudança de Escala :

$$Z_k \in \Re^{m \times m}, Z_k = \text{diag}(z_1^k, z_2^k, \ldots, z_m^k)$$

$$A_k \in \Re^{m \times n}, A_k = Z_k^{-1} A.$$

Utilizando-se das regras de tradução (ver seção I.4 — Lema I.4) e do resultado do item anterior, a direção h_A do algoritmo afim escala para um problema no formato dual é dada por :

$$h_A = -(A_k^T A_k)^{-1} c \tag{II.4}$$

onde $h_A \in \Re^n$.

II.5 Elipses para o problema (PD)

Considere o problema (PD). Seja $x^k \in S^0$ um ponto dado. Os elipsóides utilizados como região de confiança foram definidos a partir do problema no formato primal (ver seção II.2). Assim, é necessário que consideremos o problema (PP) no *formato primal* correspondente ao problema (PD) (ver seção I.4), para obter os elipsóides correspondentes a um problema dado no *formato dual.*

Sejam $z \in \Re^m$ e $x \in S^0$, $z = \bar{b} - \bar{A}x$ e $z^k \in \Re^m$, $z^k = \bar{b} - \bar{A}x^k$.

A equação do elipsóide centrado no ponto z^k para o problema dado acima — ver (II.1), é dada por :

$$\bar{h}^T Z_k^{-2} \bar{h} = 1$$

onde $\bar{h} \in \Re^m$ e $Z_k \in \Re^{m \times m}$, $Z_k = \text{diag}(z_1, z_2, \ldots, z_m)$.

Utilizando-se das regras de tradução (ver seção I.4), obtem-se a equação do elipsóide para o problema no *formato dual.* Então temos que :

$$h^T D h = 1$$

onde $D \in \Re^{n \times n}$, $D = A^T Z_k^{-2} A$ e $h \in \Re^n$.

Para visualização gráfica, consideraremos o problema bi-dimensional no *formato dual*, e pode-se obter explicitamente a equação da elipse, ou seja :

$$h^T D h = 1$$

onde $h \in \Re^2$ e $D \in \Re^{2\times 2}$ e D é dada por :

$$\begin{bmatrix} \sum_{i=1}^{m} \frac{a_{i,1}^2}{(b_i - A_i x^k)^2} & \sum_{i=1}^{m} \frac{a_{i,1}a_{i,2}}{(b_i - A_i x^k)^2} \\ \sum_{i=1}^{m} \frac{a_{i,1}a_{i,2}}{(b_i - A_i x^k)^2} & \sum_{i=1}^{m} \frac{a_{i,2}^2}{(b_i - A_i x^k)^2} \end{bmatrix}.$$

A elipse está centrada no ponto x^k, e com comprimento dos eixos dado pelo inverso da raiz quadrada dos auto-valores da matriz D e com direção dos eixos dada pelos auto-vetores da matriz D.

II.6 Passo Unitário para o problema (PD)

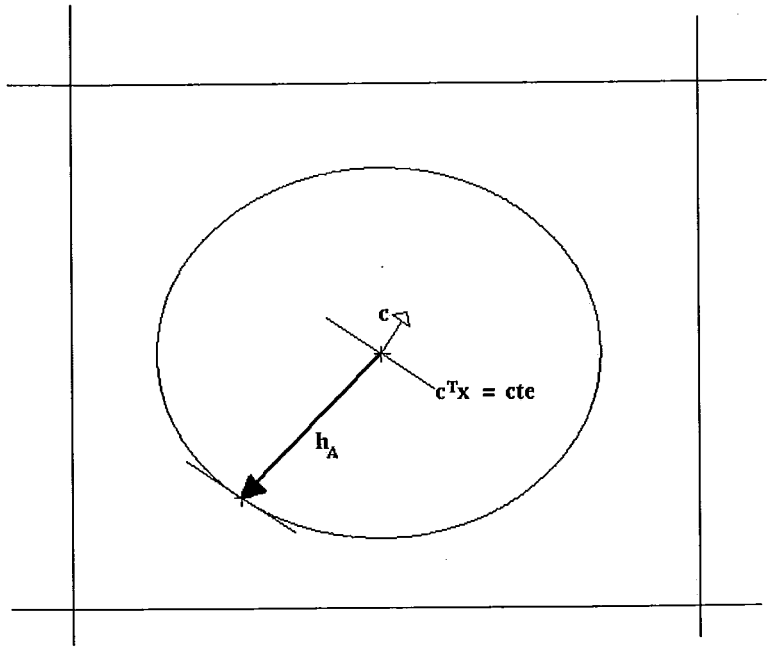


Figura II.1: Passo Unitário na Direção h_A

Seja um ponto $x \in \Re^n$ interior à região viável do problema (PD) e $\alpha \in \Re_+$.

O passo unitário αh_A para o método afim escala será dado pela interseção da elipse centrada no ponto interior x (definida na seção II.5) com a semi-reta definida por $x + \alpha h_A$. Então, o passo αh_A será a solução de :

$$(\alpha h_A)^T D(\alpha h_A) = 1$$

onde $D \in \Re^{2\times 2}$ está definida como na seção II.5 e $h_A \in \Re^n$ está dada como em (II.4). Assim, temos que α será dado por :

$$\alpha = \frac{1}{\sqrt{h_A^T D h_A}} \tag{II.5}$$

II.7 Busca Linear

Seja $x^k \in \Re^n$ um ponto dado e interior à região viável do problema (PD).

Caso não se utilize o passo unitário é necessário que se faça uma busca linear na direção h_A, ou seja, minimizar a função $\theta(\lambda) : \Re \longmapsto \Re$ dada por :

$$\theta(\lambda) = c^T(x + \lambda h_A)$$

para $x + \lambda h_A$ viável.

No caso linear, como a direção h_A é uma direção de declive, ou seja :

$$c^T h_A < 0$$

basta encontrar o maior passo λh_A tal que $x + \lambda h_A$ seja viável, ou seja :

$$\bar{\lambda} = \text{argmax}\{\lambda \mid A(x^k + \lambda h_A) \leq b\}$$

$$\bar{\lambda} = \text{argmax}\{\lambda \mid b - Ax^k - \lambda A h_A \geq 0\}$$

$$\bar{\lambda} = \text{argmax}\{\lambda \mid z^k - \lambda d \geq 0\}$$

onde $d = Ah_A$ e $z^k = b - Ax^k$.

Por hipótese $z^k > 0$.Considere $\lambda \in \Re_+$ tal que $x + \lambda h_A$ seja viável.

Se $d \leq 0$, então $z^k - \lambda d \geq 0$, para todo $\lambda > 0$, e portanto, o conjunto viável é ilimitado o que contraria as hipótese do problema (PD). Logo, existe pelo

menos um $j \in [1, 2, \ldots, m]$ tal que $d_j > 0$. Para todo $j = 1, \ldots, m$ é necessário que $z_k^j - \lambda d_j \geq 0$. Se $d_j \leq 0$ a desigualdade anterior será satisfeita para qualquer $\lambda \geq 0$. Se $d_j > 0$, então $z_j^k \geq \lambda d_j$. Logo :

$$\lambda \leq \frac{z_j^k}{d_j},$$

para todo $j = 1, \ldots, m$ tal que $d_j > 0$.

Assim, como $\bar{\lambda} = \text{argmax}\{\lambda \mid z^k - \lambda d \geq 0\}$, o comprimento do passo $\bar{\lambda}$ será dado por :

$$\bar{\lambda} = \min_{j=1,\ldots,m} \{\frac{z_j^k}{d_j}, d_j > 0\}. \tag{II.6}$$

Observe que com o passo $\bar{\lambda} h_A$ temos que $z_j^k - \lambda d_j = 0$, para pelo menos um $j \in N$ tal que $d_j > 0$, ou seja, com o passo $\bar{\lambda} h_A$ atinge-se a fronteira do conjunto.

A fim de se evitar a fronteira do conjunto (pois estamos trabalhando com pontos interiores), é necessário que se utilize um fator heurístico $\eta \in \Re$ da ordem de 95% (que se revelou eficiente na prática), ou seja, o novo ponto $x^{k+1} \in \Re^n$, será dado por :

$$x^{k+1} = x^k + \eta \bar{\lambda} h_A.$$

II.8 Algoritmo Afim Escala para o problema (PD)

Agora, podemos enunciar o algoritmo Afim Escala.

Algoritmo II.1 Algoritmo Afim Escala : *Dados* $x^0 \in S^0$, *e* $\eta \in \Re, \eta \in (0, 1)$.

$k = 0$;

Repita

Cálculo das Folgas : $z^k = b - Ax^k$;

Mudança de Escala :

$$Z_k = \text{diag}(z_1^k, z_2^k, \ldots, z_m^k)$$

$$A_k = Z_k^{-1} A;$$

Direção:

$$h_A = -(A_k^T A_k)^{-1} c$$

Cálculo do Passo :

Passo Unitário : $\alpha = \frac{1}{\sqrt{h_A^T D h_A}}$ (ver seção II.6)
ou Busca Linear : $\alpha = \bar{\lambda}\eta$ — ver (II.6) ;

Atualização :

$$x^{k+1} = x^k + \alpha h_A;$$

$$k = k + 1;$$

Até CONVERGIR

O parâmetro η é o fator heurístico utilizado pelo algoritmo afim escala para se evitar a fronteira (ver seção II.7). No caso do passo unitário o fator heurístico η não é utilizado.

II.9 Critérios de Parada

Para fins práticos, os critérios de parada utilizados no algoritmo afim escala para um problema no *formato dual* foram :

1. o número de iterações do algoritmo;
2. a norma da direção h_A está próxima de zero;
3. imprecisões numéricas, que fazem com que os pontos gerados pelo algoritmo se aproximem da fronteira ou que não seja possível realizar a busca linear.

II.10 Figuras

Na figura II.2 poderíamos obter uma elipse maior dentro da região viável, mas o cálculo da maior elipse possível dentro da região viável não é simples e é um problema tão difícil como o de programação linear. Portanto, escolheu-se a elipse mais simples possível — ver seção II.5. Isso não é uma restrição muito grande, pois

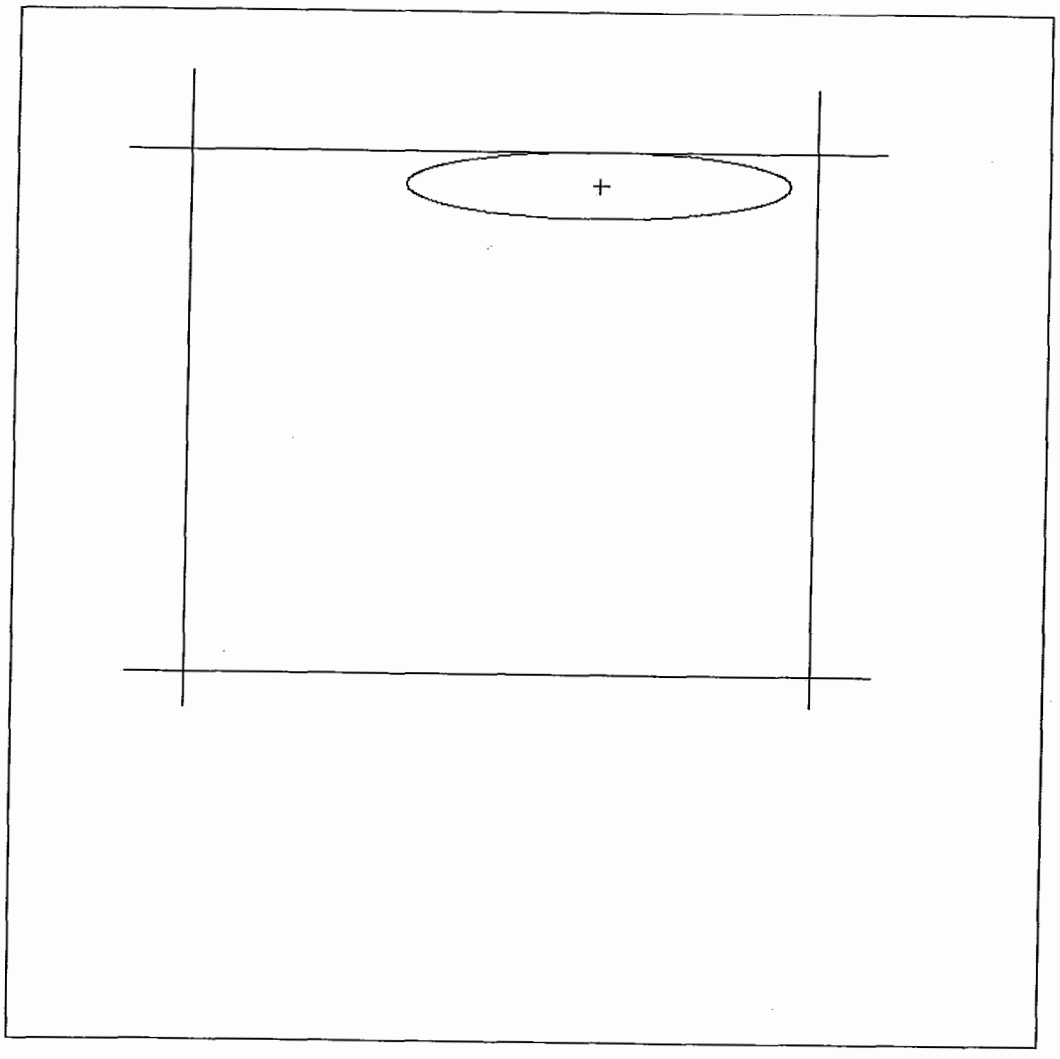


Figura II.2: Elipse mais simples possível dentro da região viável

com a busca linear (ver seção II.7), o que se faz na verdade é aumentar a elipse até chegar "perto" da fronteira (porém a elipse não está mais dentro da região viável).

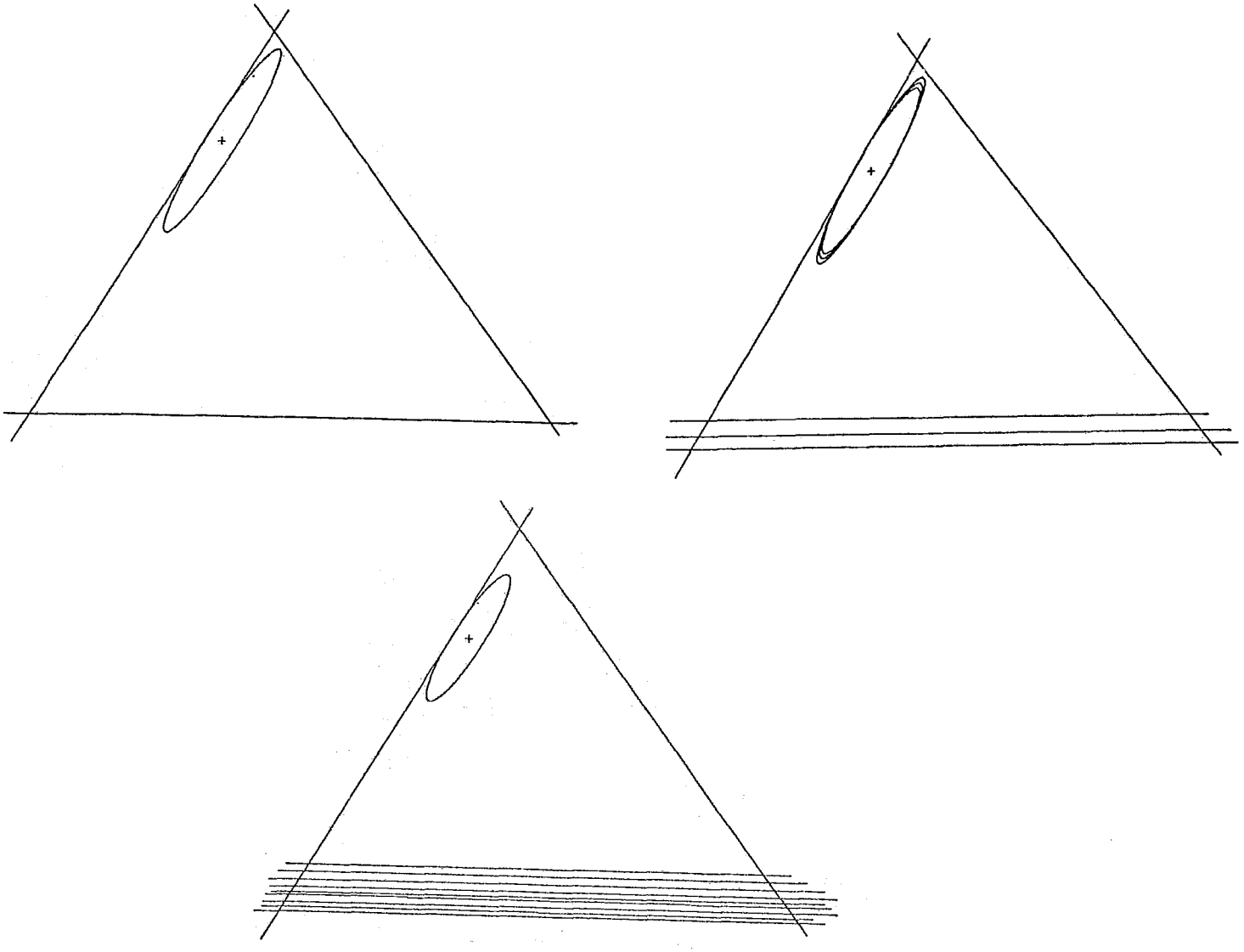


Figura II.3: Variação da elipse com a adição de restrições adicionais

Na figura II.3 mostra-se como a elipse utilizada como região de confiança se comporta com a adição de restrições adicionais. Repare que as elipses diminuem apesar dessas restrições não afetarem a região viável inicial.

A figura II.4 mostra uma iteração do algoritmo Afim Escala com Passo Unitário, observe que o ponto gerado pelo algoritmo pertence à elipse centrada no ponto interior inicial — ver seção II.6.

A figura II.5 mostra iterações do algoritmo Afim Escala com Passo Unitário até um ótimo com as elipses geradas a cada iteração.

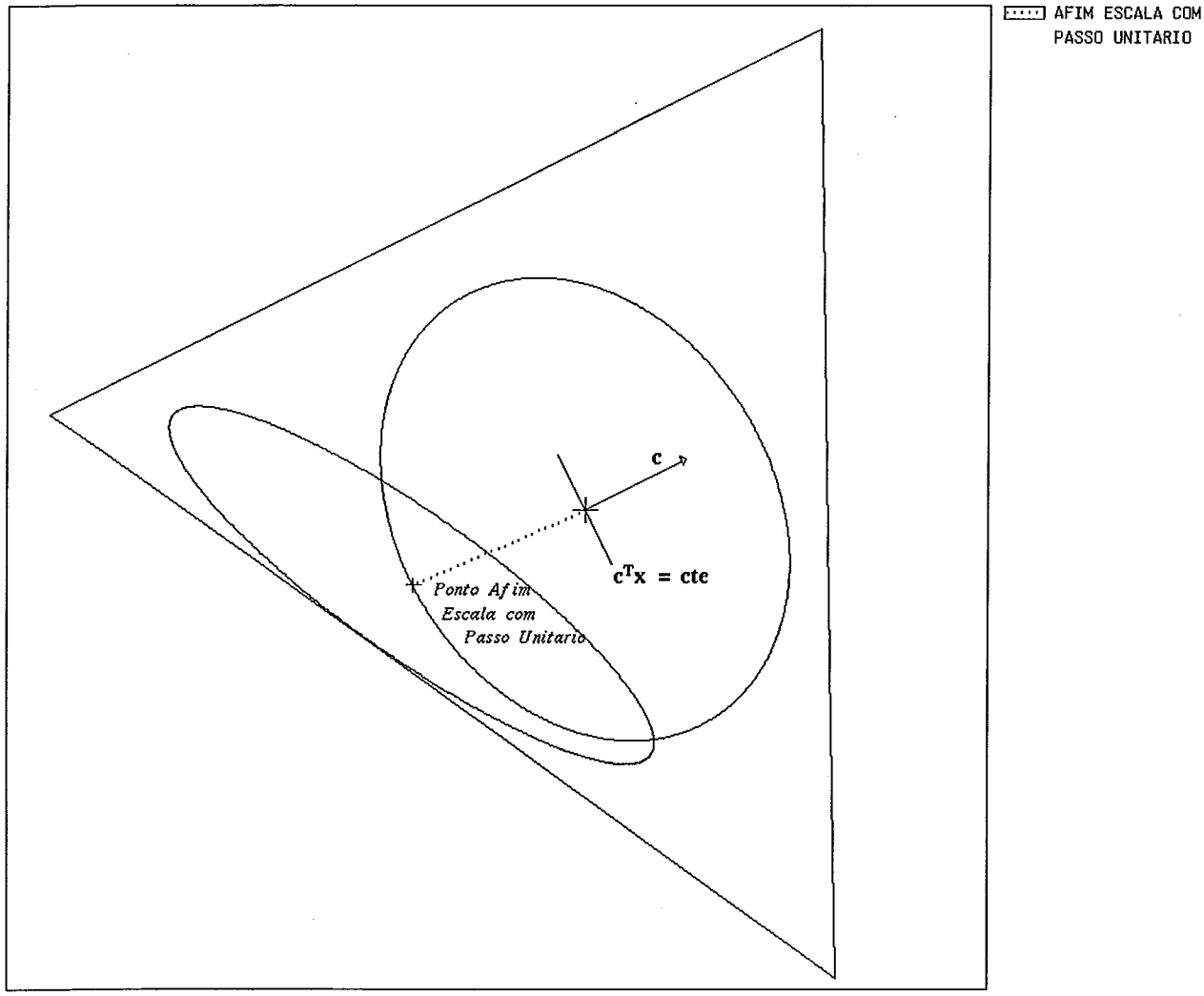


Figura II.4: Uma iteração do Algoritmo Afim Escala com Passo Unitário

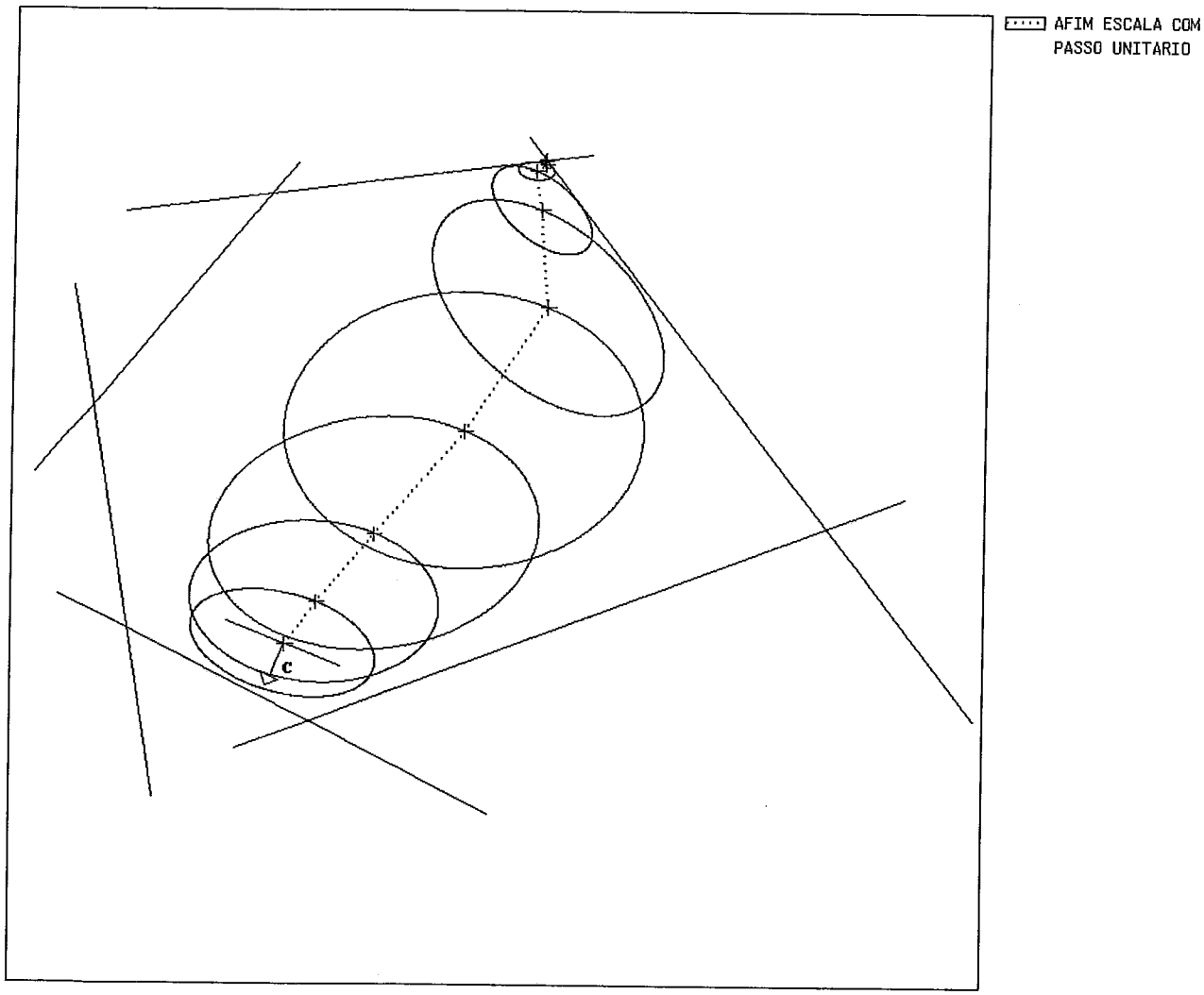
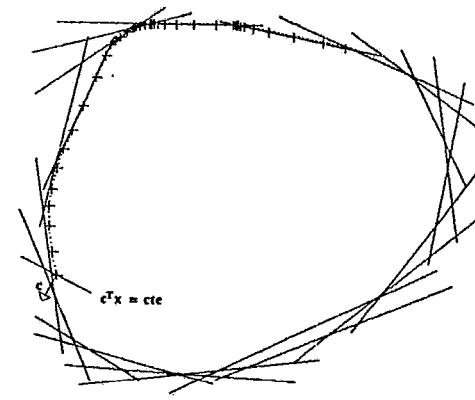


Figura II.5: Iterações do Algoritmo Afim Escala com Passo Unitário até um Ótimo



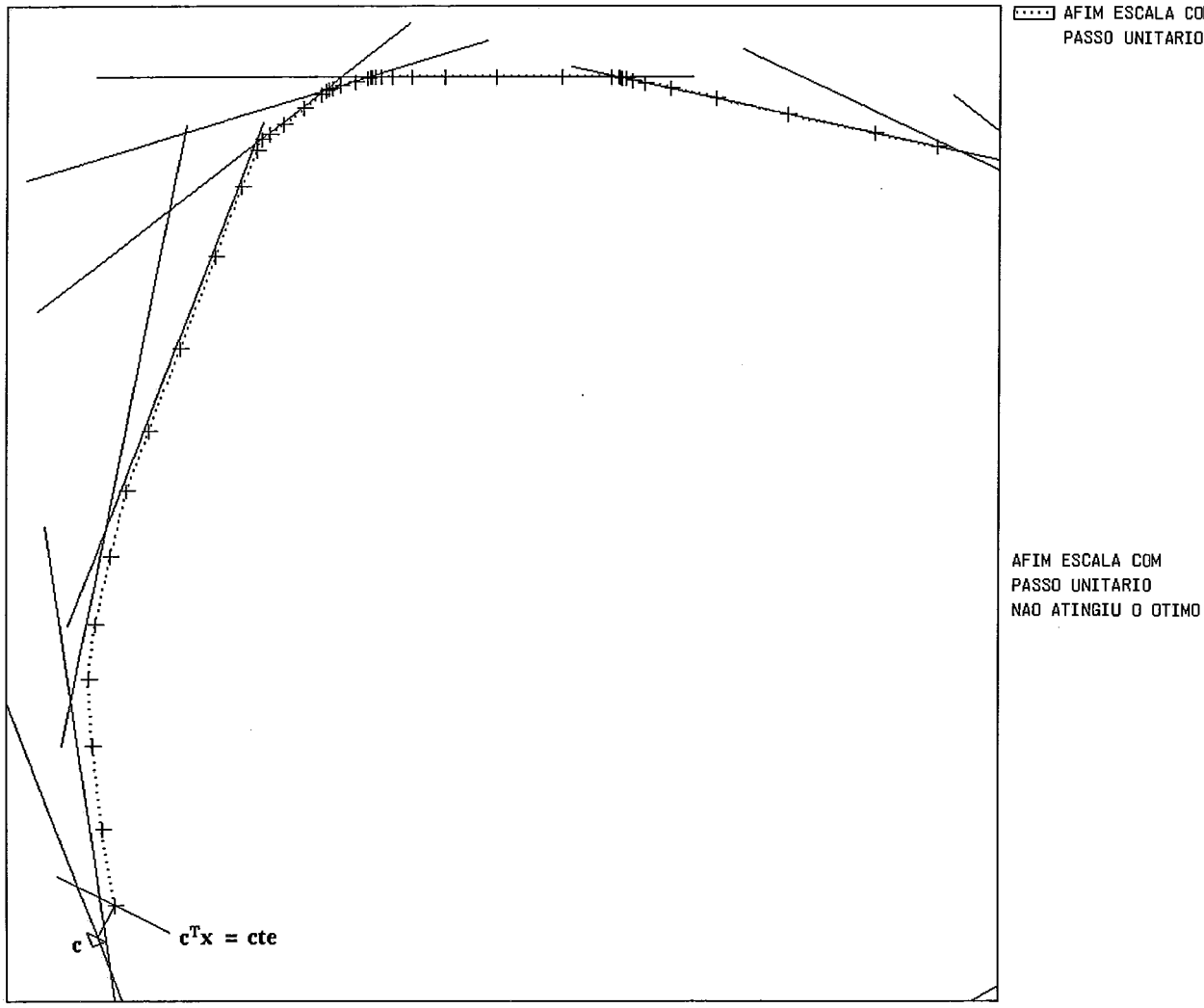


Figura II.6: Iterações do Algoritmo Afim Escala com Passo Unitário quando não atingiu um ótimo

O algoritmo Afim Escala com Passo Unitário não conseguiu atingir um ótimo na figura II.6 devido à imprecisões numéricas e à proximidade à fronteira (observe como o ponto interior está próximo da fronteira).

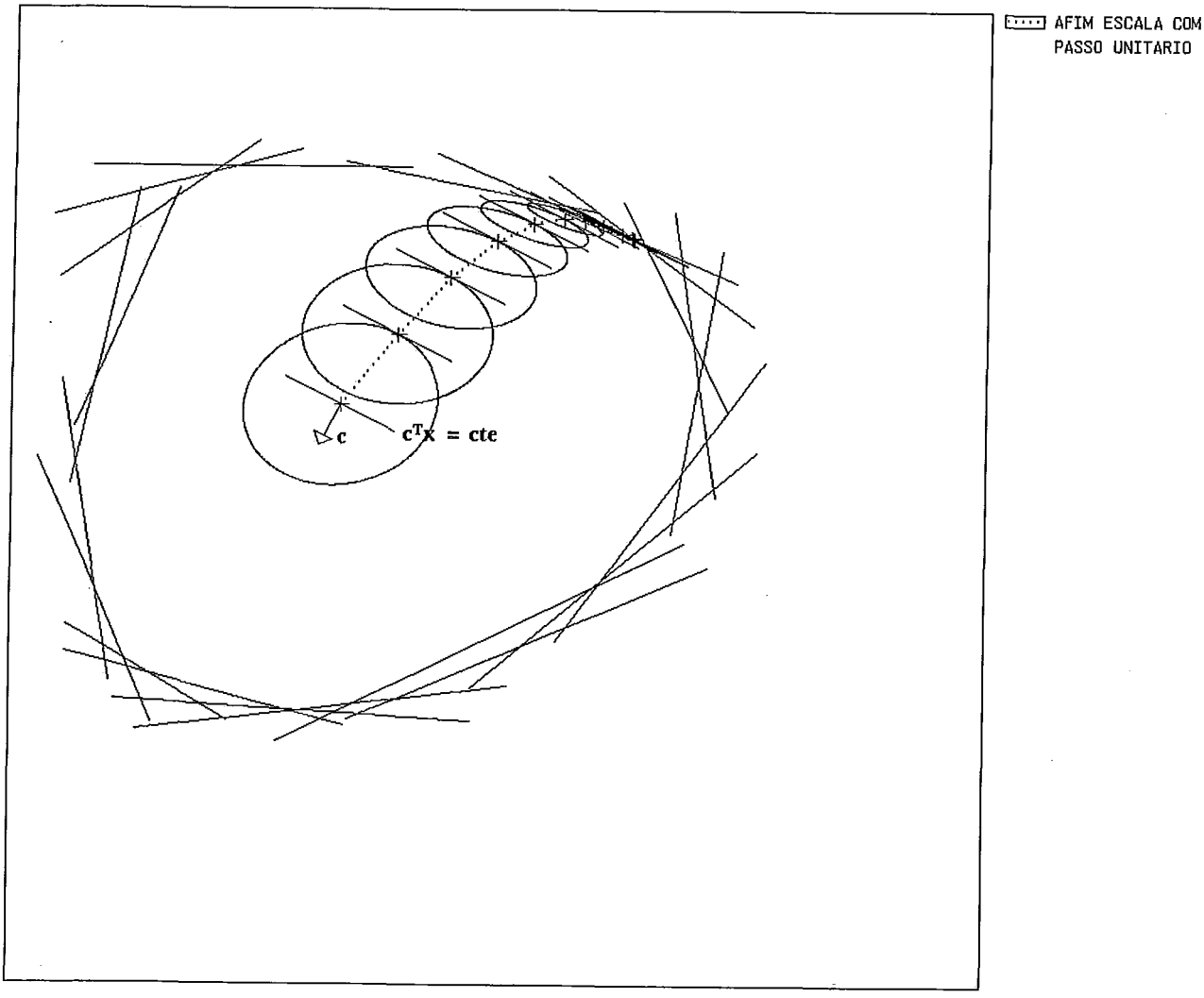


Figura II.7: Iterações do Algoritmo Afim Escala com Passo Unitário até atingir um ótimo

Repare na figura II.7 que se mudarmos o ponto interior inicial para um ponto afastado da fronteira, o algoritmo converge.

A figura II.8 mostra uma iteração do algoritmo Afim Escala com Busca Linear, observe que devido à busca o ponto gerado pelo algoritmo está fora da elipse.

A figura II.9 mostra iterações do algoritmo Afim Escala com Busca

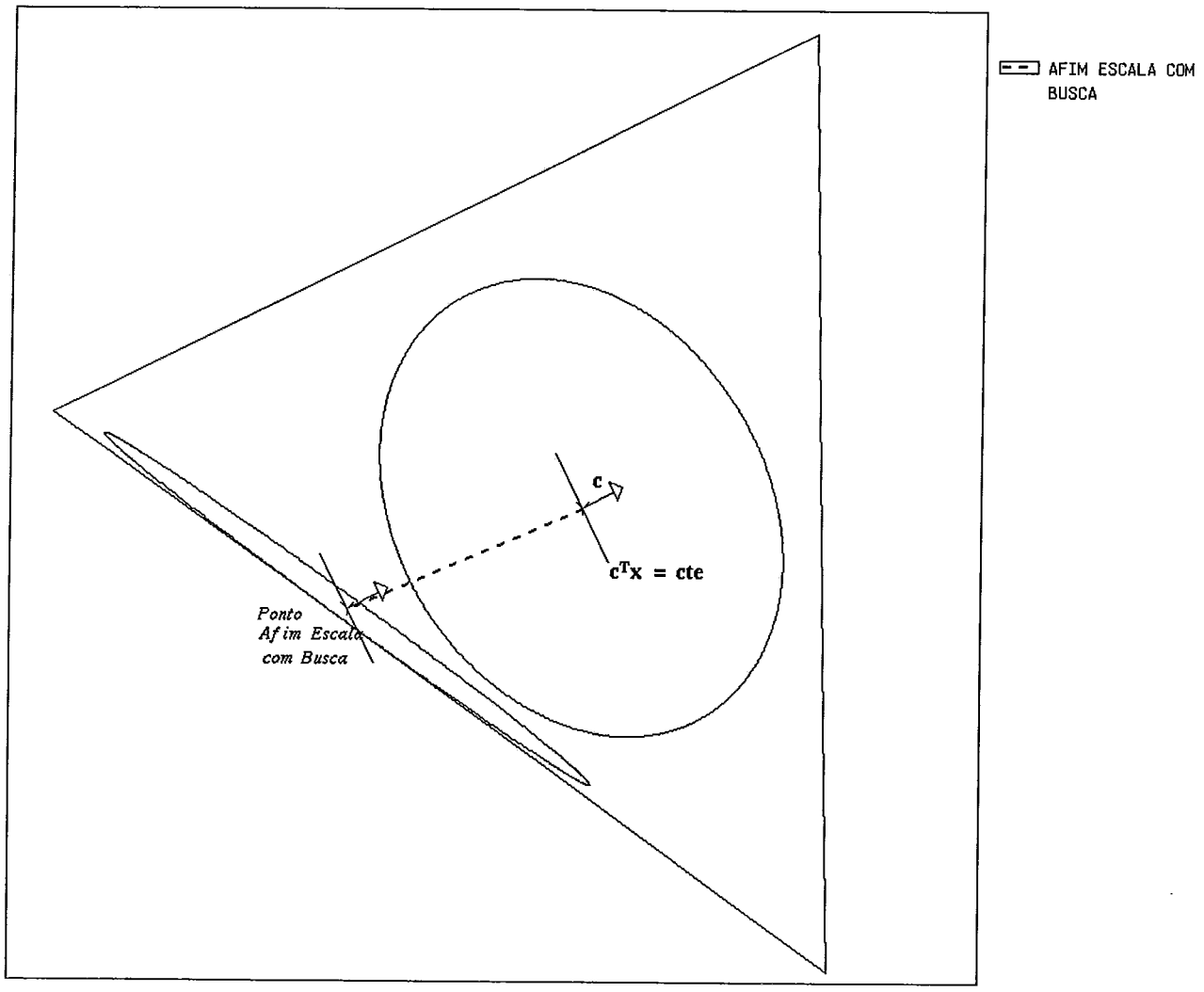


Figura II.8: Uma iteração do Algoritmo Afim Escala com Busca Linear

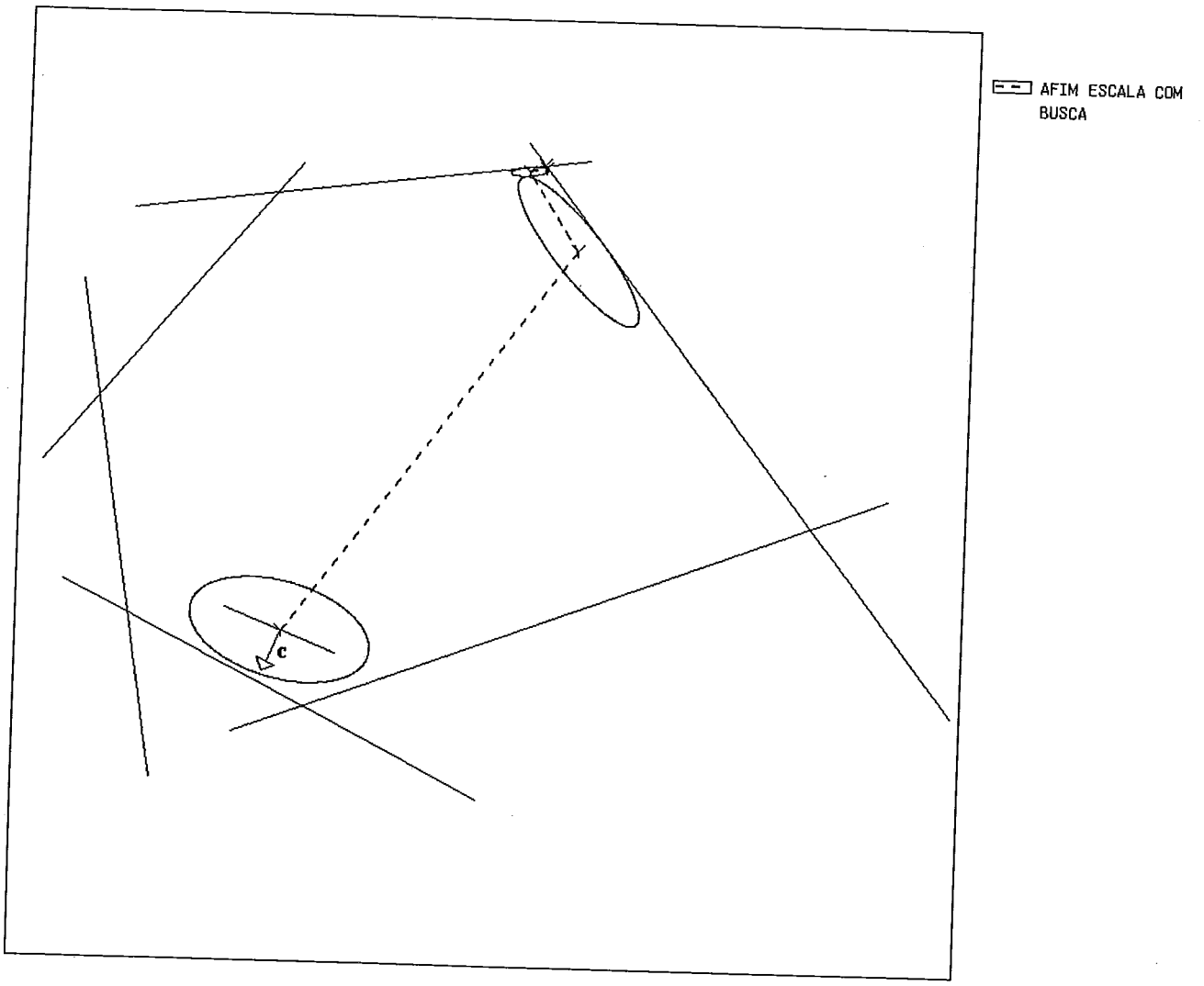


Figura II.9: Iterações do Algoritmo Afim Escala com Busca Linear até um ótimo

Linear até um ótimo, veja como os pontos gerados pelo algoritmo se aproximam da fronteira e que o número de iterações é bem inferior ao do Afim com Passo Unitário.

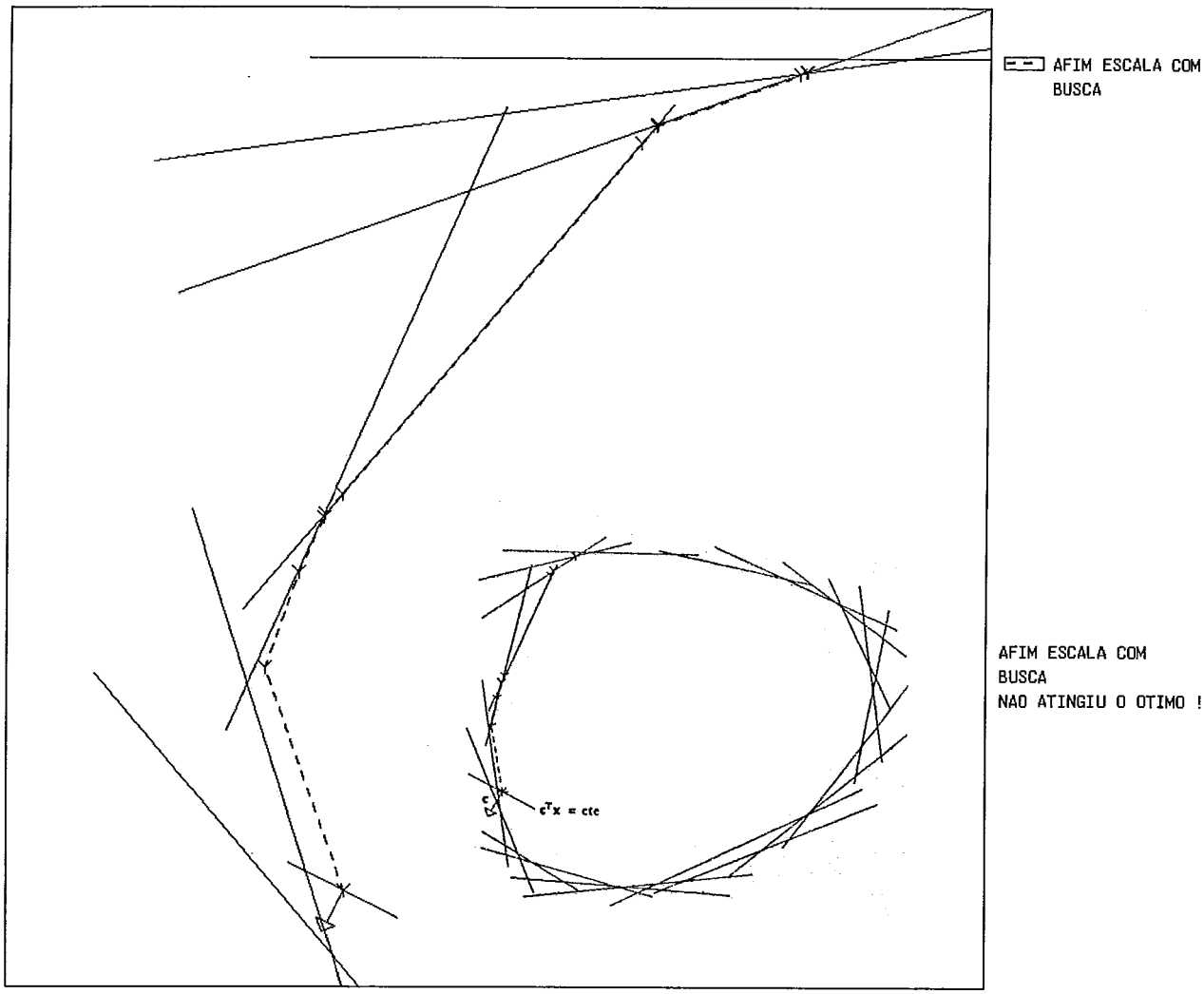


Figura II.10: Iterações do Algoritmo Afim Escala com Busca Linear quando não atingiu um ótimo

Na figura II.10 o algoritmo Afim Escala com Busca não atingiu um ótimo devido à imprecisões numéricas e a proximidade à fronteira (pois a tendência da Busca Linear nesse algoritmo é aproximar os pontos interiores da fronteira da região viável) — o mesmo ocorreu para o Afim Escala com Passo Unitário (ver figura II.6.

Se mudarmos o ponto interior inicial para um ponto afastado da fronteira, o algoritmo Afim Escala com Busca também converge — ver figura II.11.

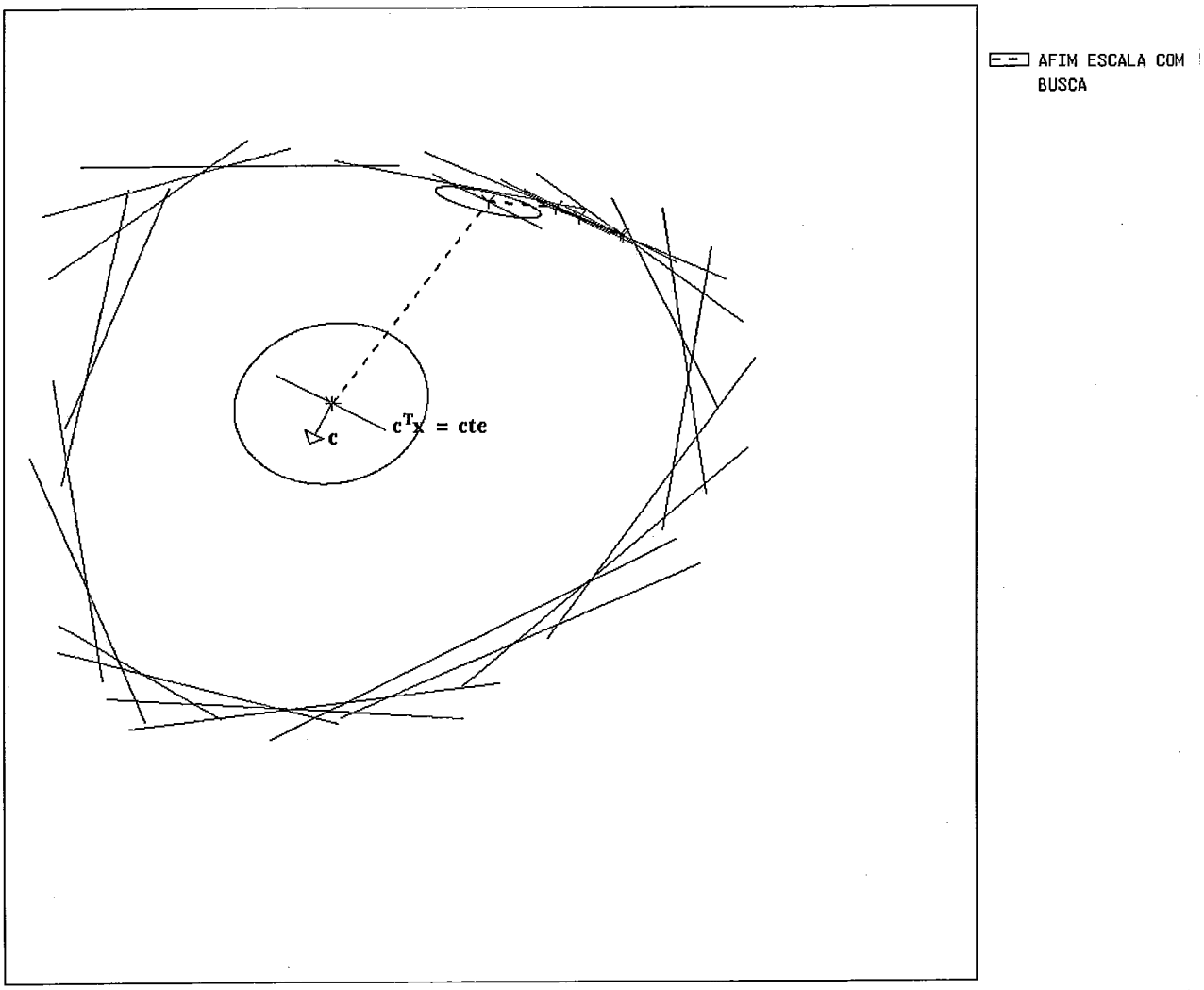


Figura II.11: Iterações do Algoritmo Afim Escala com Busca Linear até um ótimo

Capítulo III

O Algoritmo de Karmarkar

Após a publicação do algoritmo dos elipsóides por Khachiyan [19] em 1978, ficou-se provado que o problema de programação linear pode ser resolvido por um algoritmo polinomial. Apesar do grande interesse despertado por este método em termos de teoria de complexidade, os resultados de implementação revelaram-se ineficientes.

Com a publicação do algoritmo de Karmarkar[17] em 1984, pode-se obter finalmente o resultado que vinha sendo esperado desde que se provou que o método simplex tem convergência exponencial no pior caso, ou seja, obteve-se um algoritmo polinomial que resolve o problema de programação linear e cuja implementação é muito eficiente na prática.

O algoritmo de Karmarkar em sua formulação original apresenta algumas operações que mais tarde se demonstrou serem desnecessárias. Por isso, mesmo, a variante do algoritmo de Karmarkar que apresentaremos a seguir, não segue exatamente o algoritmo original mas é equivalente a este.

III.1 Função Barreira

Antes de explicitarmos o processo de obtenção do algoritmo de Karmarkar é necessário que se analise a função barreira logarítmica. Esta função foi utilizada primeiro por Frisch [8] em 1955 e é definida como se segue :

$$x \in \Re^n, x > 0 \longmapsto \bar{p}(x) = -\sum_{i=1}^{n} \log x_i.$$

Esta função possui uma propriedade especial, a função cresce indefinidamente perto da fronteira do conjunto viável, e pode ser usada como uma penalidade associada aos pontos de fronteira. Isto pode ser observado através do gráfico das curvas de nível da função barreira para um problema de programação linear no *formato dual* :

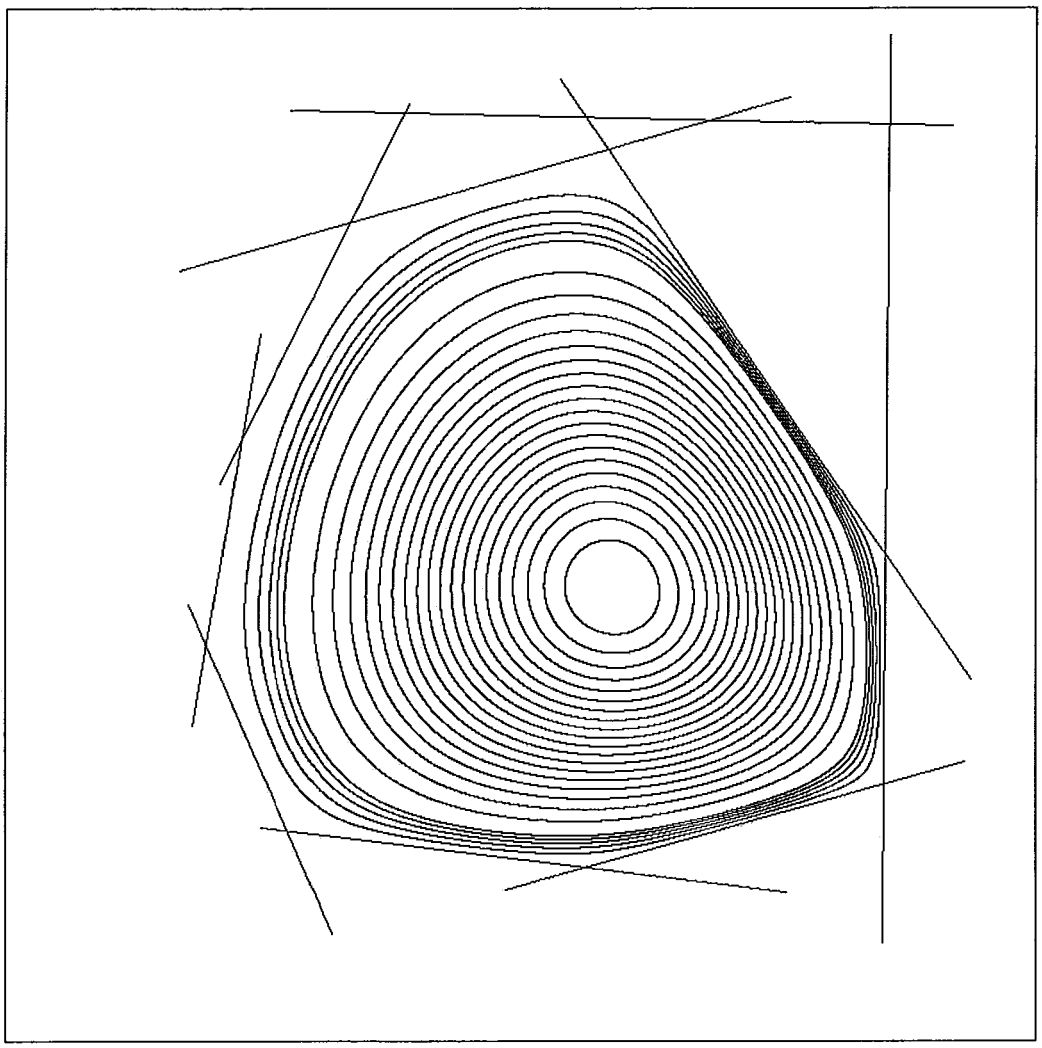


Figura III.1: Politopo $Ax \leq b$ com curvas de nível da função barreira p

- Outras propriedades da função barreira são :

  1. A função barreira é analítica para $x \in \Re^n_{++}$ e suas derivadas são dadas por :

$$\nabla \bar{p}(x) = -x^{-1}$$

e

$$\nabla^2 \bar{p}(x) = X^{-2}$$

onde x^{-1} é o vetor $[x_i^{-1}]_{i=1,\ldots,n}$ e

$$X = \text{ diag } (x_1, \ldots, x_n).$$

A função é estritamente convexa, pois sua matriz hessiana é definida positiva. Vale observar que no ponto $e \in \Re^n$, $e = (1, 1, \ldots, 1)^T$ temos que:

$$\nabla \bar{p}(e) = -e$$

e

$$\nabla^2 \bar{p}(e) = I.$$

2. Considere uma matriz diagonal positiva D. Tem-se que :

$$p(Dx) = p(x) - \sum_{i=1}^{n} \log d_i.$$

Dados dois pontos $x, y > 0$ temos que :

$$p(Dy) - p(Dx) = p(x) - p(y)$$

e portanto mudanças de escala não afetam *variações* de $p(\cdot)$.

A função barreira para um problema de programação linear no *formato dual* é dada por :

$$x \in \Re^n, x \in S \longmapsto p(x) = -\sum_{i=1}^{m} \log(b_i - A_i x)$$

onde A_i é a transposta do vetor linha da matriz A.

É possível combinar-se a função barreira com a função custo, para se obter uma função objetivo para um algoritmo de ponto interior que resolva o problema de programação linear evitando a fronteira e, ao mesmo tempo, reduzindo essa função objetivo. Foi baseado neste fato que Karmarkar definiu a função a ser minimizada no seu algoritmo (a função potencial). Vale ressaltar também que a função barreira já vinha sendo utilizada em programação não-linear, porém ainda não se tinha utilizado esta função em programação linear.

III.2 Formato do problema

O algoritmo de Karmarkar foi desenvolvido para um problema de programação linear num formato especial :

$$\begin{array}{lll} \text{(PR)} & \text{minimizar} & c^T x \\ & \text{sujeito a} & Ax = 0 \\ & & a^T x = 1 \\ & & x \geq 0 \end{array}$$

onde $c, a \in \Re^n$, $a \geq 0$ e $A \in \Re^{m \times n}$ é uma matriz de rank completo, $m > n$. Isto não é uma restrição ao problema geral, pois esse formato pode ser obtido introduzindo-se uma nova variável ao problema no seu formato original.

Além disso, faremos a hipótese de que o problema tem uma solução ótima $\hat{x}$ e que $c^T \hat{x} = 0$.

III.3 Função Potencial

Devido as propriedades da função barreira mencionadas na Seção III.1, Karmarkar definiu a função a ser minimizada pelo seu algoritmo, denominada de *função potencial*, como se segue :

$$x \in \Re^n_{++} \longmapsto \bar{f}_0(x) = q \log(c^T x) + \bar{p}(x)$$

onde $q \geq n$ é uma constante e o sub-índice 0 simboliza o fato de que o custo ótimo é supostamente nulo.

Observe a função $\bar{f}_0(\cdot)$ é *homogênea de grau zero*, ou seja, para qualquer $x \in \Re^n, \alpha > 0$, temos que : $\bar{f}_0(\alpha x) = \bar{f}_0(x)$.

III.4 Modelo do Algoritmo de Karmarkar

Podemos ignorar a restrição $a^T x = 1$, pois se dado $x > 0$ tal que $A^T x = 0$ e $a^T x > 0$ mas $a^T x \neq 1$, o ponto $x/a^T x$ é viável ao problema (PR) (pois $a^T(x/a^T x) = 1$) e o valor da função $\bar{f}_0$ é o mesmo já que $\bar{f}_0$ é homogênea de grau zero — ver seção III.3.

O ponto mencionado no parágrafo anterior é também denominado de *projeção cônica* $K(x)$ de x sobre $\bar{S}$, logo :

$$K(x) = \frac{x}{a^T x}.$$

Agora, podemos enunciar um modelo do algoritmo de Karmarkar aplicado ao problema (P) supondo que (P) está dado no formato do problema (PR).

1. Ignore a restrição $a^T x = 1$;
2. Use o algoritmo Afim Escala para reduzir $\bar{f}_0$;
3. Calcule a projeção cônica do ponto resultante, ou seja :

$$\bar{x} = \frac{x}{a^T x}.$$

III.5 Direção do Algoritmo de Karmarkar para o problema (PR)

Do modelo do algoritmo de Karmakar dado na seção III.4 vamos considerar o problema (PR) e ignorar a restrição $a^T x = 1$ e utilizar a função potencial para obtermos um novo problema equivalente ao original:

$$\begin{array}{lll} \text{(PK)} & \text{minimizar} & \bar{f}_0(x) \\ & \text{sujeito a} & Ax = 0 \\ & & x \geq 0 \end{array}$$

onde $\bar{f}_0$ está dada como na seção III.3 e A está definida como no problema (PR).

Do modelo de algoritmo de Karmarkar dado na seção III.4, sabemos que a direção de busca a ser utilizada para o algoritmo de Karmarkar será a mesma obtida para o algoritmo afim escala aplicado ao problema (PK). Da seção II.3, sabemos que para obter essa direção é necessário aplicar uma mudança de escala (para um problema dado no formato primal) e, em seguida, aplicar o método de Cauchy ao problema (PK).

Então seja $x^k \in \bar{S}^0$ um ponto dado. A partir de x^k podemos definir a mudança de escala como dada na seção II.2, ou seja : no lugar da matriz $\bar{A}$

passaremos a utilizar a matriz $\bar{A}_k$ dada por :

$$\bar{A}_k = \bar{A}X_k$$

onde $\bar{A}_k \in \Re^{m\times n}$ e $X_k = diag(x_1^k, \ldots, x_n^k)$.

A função objetivo para o problema (PK) é dada por $\bar{f}_0$ (supondo que o custo ótimo é nulo). Após a mudança de escala (agora, a região de confiança é uma bola), podemos aplicar o método de Cauchy à $\bar{f}_0$. Assim, temos que a partir do ponto x^k e, após a mudança de escala, a direção de busca para o algoritmo de Karmarkar será dada pela direção oposta ao gradiente da função $\bar{f}^0$ projetado no espaço nulo de $\bar{A}_k$, ou seja :

$$\bar{h}_K = -X_k P_{\bar{A}_k} X_k \nabla \bar{f}_0(x^k)$$

$$\bar{h}_K = -X_k P_{\bar{A}_k}(\frac{n}{c^T x^k} Xc - e) \qquad \text{(III.1)}$$

onde $\bar{h}_K \in \Re^n$.

Observe que ao projetar o gradiente de $\bar{f}_0$ não é necessário projetar o vetor e, uma vez que esse já pertence ao $\mathcal{N}(\bar{A}_k)$. Utilizando esse fato em (III.1) obtemos :

$$\bar{h}_K = -\frac{n}{c^T x^k} X_k P_{\bar{A}_k} X_k c + x^k. \qquad \text{(III.2)}$$

III.6 Função Potencial com Limitante Inferior

A hipótese de que o custo ótimo é nulo pode ser relaxada, para isso, basta que se tenha um parâmetro v que seja um limitante inferior para o custo ótimo $\hat{v}$, supondo conhecido o custo ótimo. Então, considere o problema (P) e vamos supor que conhecemos o valor da solução ótima $\hat{v}$ e que v seja um limitante inferior para esse custo ótimo. Devido as propriedades da função barreira mencionadas na Seção III.1, Karmarkar definiu a função a ser minimizada pelo seu algoritmo, denominada de *função potencial*, como se segue : para $v \leq \hat{v}$ temos que

$$x \in \Re^n_{++} \longmapsto \bar{f}_v(x) = q\log(c^T x - v) + \bar{p}(x)$$

onde o parâmetro v é um limitante inferior para o custo ótimo e $q \geq n$ é uma constante.

Para um problema no *formato dual* utilizando-se as regras de tradução (ver seção I.4), a função potencial será dada por :

$$x \in S \longmapsto f_v(x) = q \log(c^T x - v) + p(x).$$

III.7 Cálculo do Limitante Inferior

Na seção III.6 havíamos mencionado que se podia relaxar a restrição de que o custo ótimo fosse nulo. Para isto, basta ter-se um parâmetro v que seja um limite inferior para o custo ótimo e a função a ser minimizada será dada pela função potencial $\bar{f}_v$. Suponha que o custo ótimo $\hat{v}$ é conhecido porém não necessariamente nulo. Observe que a função $\bar{f}_{\hat{v}}$ como dada na seção III.6 não é homogênea. Observando que $a^T x = 1$ para todo $x \in \bar{S}^0$, a função pode ser reescrita no seguinte formato :

$$\bar{f}_{\hat{v}} = n \log (c - \hat{v}a)^T x + \bar{p}(x)$$

Agora, a função $\bar{f}_{\hat{v}}$ é homogênea de grau zero e coincide com a função original no conjunto viável.

Se $\hat{v}$ é desconhecido, devemos gerar limitantes inferiores $v_k \leq \hat{v}$ e usar a função $\bar{f}_{v_k}$.

O método utilizado para a geração dos limitantes inferiores foi proposto por Todd & Burrell [26]. Descreveremos a seguir uma versão do procedimento de geração de limitantes inferiores enunciada em Gonzaga [11].

Consideraremos o problema (PP) (ver seção I.4) a partir do ponto e, supondo que foi feita uma mudança de escala. Aplicando-se o algoritmo de Karmarkar a este problema, e reescrevendo o problema como se segue

$$\begin{array}{ll} \text{minimizar} & \bar{c}^T z - a^T z\,(c_0 + v) \\ \text{sujeito a} & z \in Q \\ & a^T z = 1 \\ & z \geq 0 \end{array}$$

onde $\bar{c}$, z, c_0 estão definidos como em seção I.4. Q é o subespaço gerado pela interseção do conjunto viável ao problema (PP) com a origem. Um possível valor para a é dado por :

$$a = \frac{e - e_p}{\|e - e_v\|^2}, a \in Q$$

Definindo-se

$$\hat{c} = \bar{c} - (c_0 + v)a$$

Gonzaga mostrou em [11] que $P_Q\hat{c}$ (ou seja, a projeção do vetor custo $\hat{c}$ no espaço Q) é dado por :

$$P_Q\hat{c} = \bar{c}_p + \beta(e - e_p) \tag{III.3}$$

onde

$$\beta = \frac{\bar{c}^T(e - e_p) - c_0 - v}{\|e - e_p\|^2}. \tag{III.4}$$

Demonstrou-se em [10] que se $v = \hat{v}$, então $P_Q\hat{c}$ tem uma componente não-positiva. Logo, o método para gerar o limitante inferior será dado por :

Calcule $P_Q\hat{c}$ como em (III.3). Se $P_Q\hat{c}$ tem uma componente não-positiva, então v é um limitante inferior; senão use o teste da razão para calcular o maior valor de β em (III.3) de modo que $P_Q\hat{c} \geq 0$ e obtenha o valor de v através da expressão (III.4).

III.8 Direção de Karmarkar para o problema no formato dual

Para se obter a direção de busca h_K do algoritmo de Karmarkar para um problema dado no *formato dual*, problema (PD), basta calcular a direção de Karmarkar aplicada ao problema no formato do problema (PR) obtido do problema (PP) dado na seção III.5 e, a partir desta direção, utilizar as regras de tradução — ver seção I.4. Este procedimento foi feito por Gonzaga em [11]. A partir do ponto $x^k \in S^0$ obteve-se a direção h_K dada por :

$$h_K \in \Re^n, h_K = -d_c + \beta d_e,$$

onde

$$\beta = \frac{c^T(x^k - d_e) - v}{\| e + A_k d_e \|^2} \tag{III.5}$$

e

$$e \in \Re^m, e = (1, 1, ..., 1)^T,$$

v é um limitante inferior para o custo ótimo $\hat{v}$, ou seja, $v \leq \hat{v}$, d_c como definida em (I.1) e d_e como em (I.2) com A_k no lugar de A. A_k é a matriz A após a mudança de escala como dada no Lema I.5.

III.9 Cálculo do Limitante Inferior para o problema (PD)

O método de geração de limitantes inferiores para um problema no *formato dual* é o mesmo da seção III.7, só que agora temos que

$$P_Q \hat{c} = -A_k d_c + \beta(e + A_k d_e)$$

e β está definido como em (III.5), d_c e d_e definidos como em (I.1) e (I.2), respectivamente (ver lema I.4), após a mudança de escala dada no lema I.5.

Então, o procedimento de atualização dos limitantes inferiores para o problema (PD) será dado por :

1. se existe uma componente do vetor $P_Q \hat{c}$ que seja não positiva, então mantem-se o atual limitante inferior;

2. caso contrário, obtenha

$$\bar{\beta} = \min\{\beta \mid -A_k d_c + \beta(e + A_k d_e) \geq 0\} \qquad \text{(III.6)}$$

e obtenha o novo limitante inferior de (III.5). Para resolver (III.6), basta aplicar o teste da razão. Assim, temos que :

$$\bar{\beta} = \min_{i=1,\ldots,m} \{\frac{u_i}{v_i}, v_i < 0\}$$

onde $u = -A_k d_c$ e $v = e + A_k d_e$.

III.10 Busca Linear

A busca linear na direção h_K será dada pela minimização unidimensional da função f_v na direção do passo $\bar{\lambda} h_K$. Pode-se utilizar qualquer algoritmo de minimização

unidimensional como Armijo ou Bisseção. Como podemos calcular facilmente o gradiente da função f_v e desejamos reduzir a função com uma boa precisão decidimos utilizar o método da Bisseção. Assim, o comprimento do passo $\bar{\lambda}h_K$ será dado por

$$\bar{\lambda} = \text{argmin}\{ f_v(x + \lambda h_K) \mid x + \lambda h_K \in S^0, \lambda \geq 0\}.$$

III.11 Algoritmo de Karmarkar para o problema (PD)

Agora, podemos enunciar o algoritmo de Karmarkar.

Algoritmo III.1 Algoritmo Karmarkar : *Dados* $x^0 \in S$, $v^0 \leq \hat{v}$ *e* $\epsilon > 0$.

$k = 0$;

Repita

Cálculo das Folgas : $z^k = b - Ax^k$;

Mudança de Escala :

$$Z_k = \text{diag}(z_1^k, z_2^k, \ldots, z_m^k)$$

$$A_k = Z_k^{-1} A;$$

Limitante : Usar o procedimento da seção III.9

Direção:

$$h_K = -d_c + \beta d_e \text{ — ver seção III.8;}$$

Busca :

$$\bar{\lambda} = \text{argmin}\{f_v(x^k + \lambda h_K) \mid x^k + \lambda h_K \in S^0, \lambda \geq 0\};$$

Atualização :

$$x^{k+1} = x^k + \bar{\lambda} h_K;$$

$$k = k + 1;$$

Até que $(c^T x^k - v^k) < \epsilon$

$\hat{v}$ é o valor da solução ótima do problema (PD) e ϵ é a precisão com que se deseja obter o custo ótimo.

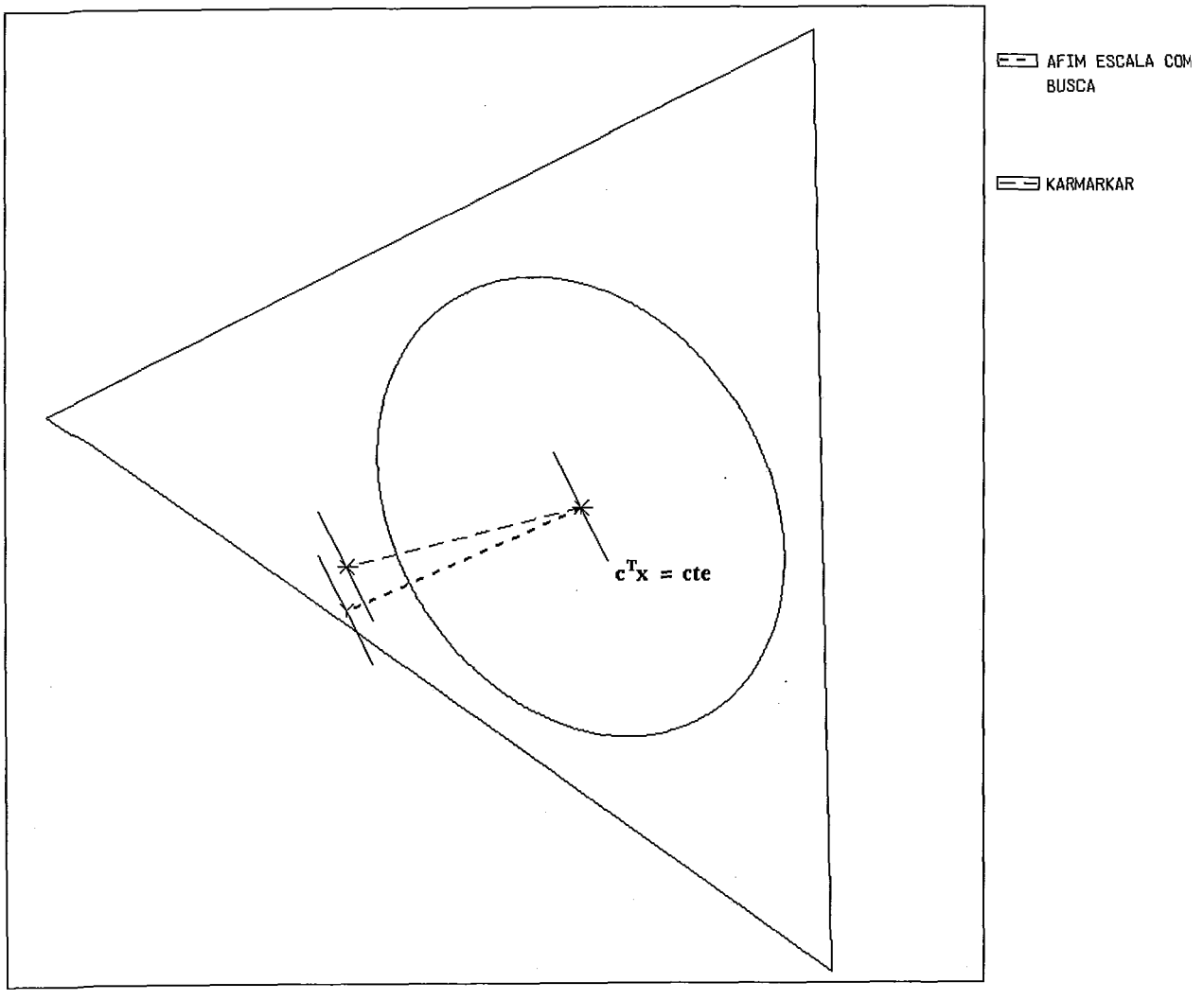


Figura III.2: Uma Iteração dos Algoritmos de Karmarkar e Afim Escala com Busca

III.12 Figuras

A figura III.2 mostra uma iteração dos algoritmos Afim Escala com Busca Linear e Karmarkar, observe que as direções de busca dos algoritmos Afim Escala com Busca Linear e Karmarkar são diferentes a partir de um ponto interior dado.

A figura III.3 mostra iterações do algoritmo de Karmarkar até um ótimo — compare com as figuras II.5 e II.9. Observe que apesar de utilizar a função barreira os pontos gerados pelo algoritmo se aproximam da fronteira.

As figuras III.4 e III.5 mostra como os algoritmos Afim Escala com

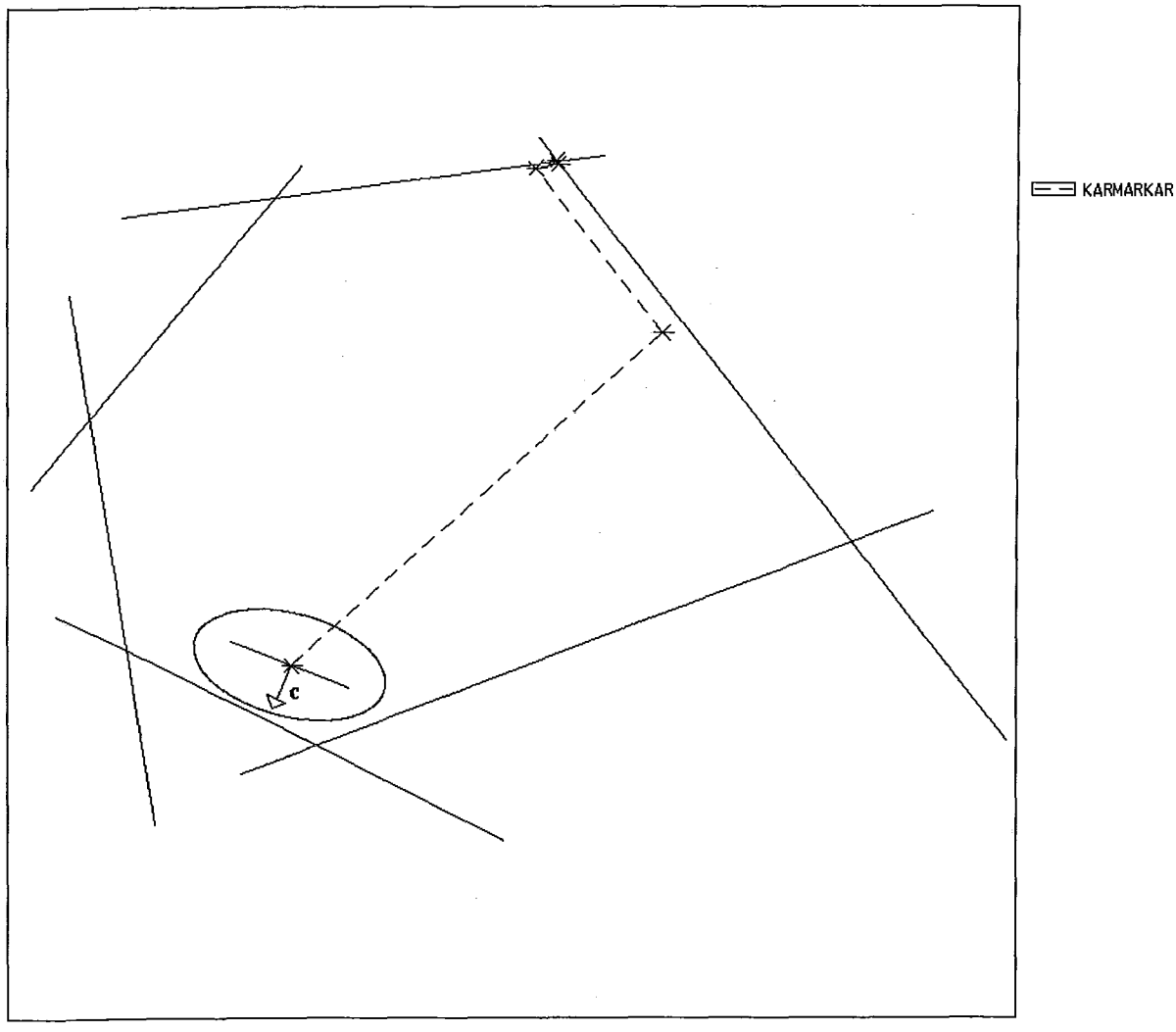


Figura III.3: Iterações do Algoritmo de Karmarkar até um ótimo

Busca Linear e Karmarkar evoluem dentro da região viável. Repare que na figura III.5 o algoritmo de Karmarkar teve um número de iterações bem inferior ao Afim Escala com Busca Linear, pois para o Afim Escala o fato do ponto inicial estar próximo da fronteira faz com que este caminhe próximo da fronteira por causa da Busca Linear, o que já não acontece para o algoritmo de Karmarkar pois esse utiliza a função barreira para evitar a fronteira.

Apesar do algoritmo de Karmarkar utilizar a função barreira, Powell mostrou recentemente no 14^o Simpósio Internacional de Programação Matemática em Amsterdam (Agosto - 1991) que se a região viável se aproxima de um círculo e se o ponto interior inicial estiver "perto" da fronteira, o algoritmo de Karmarkar

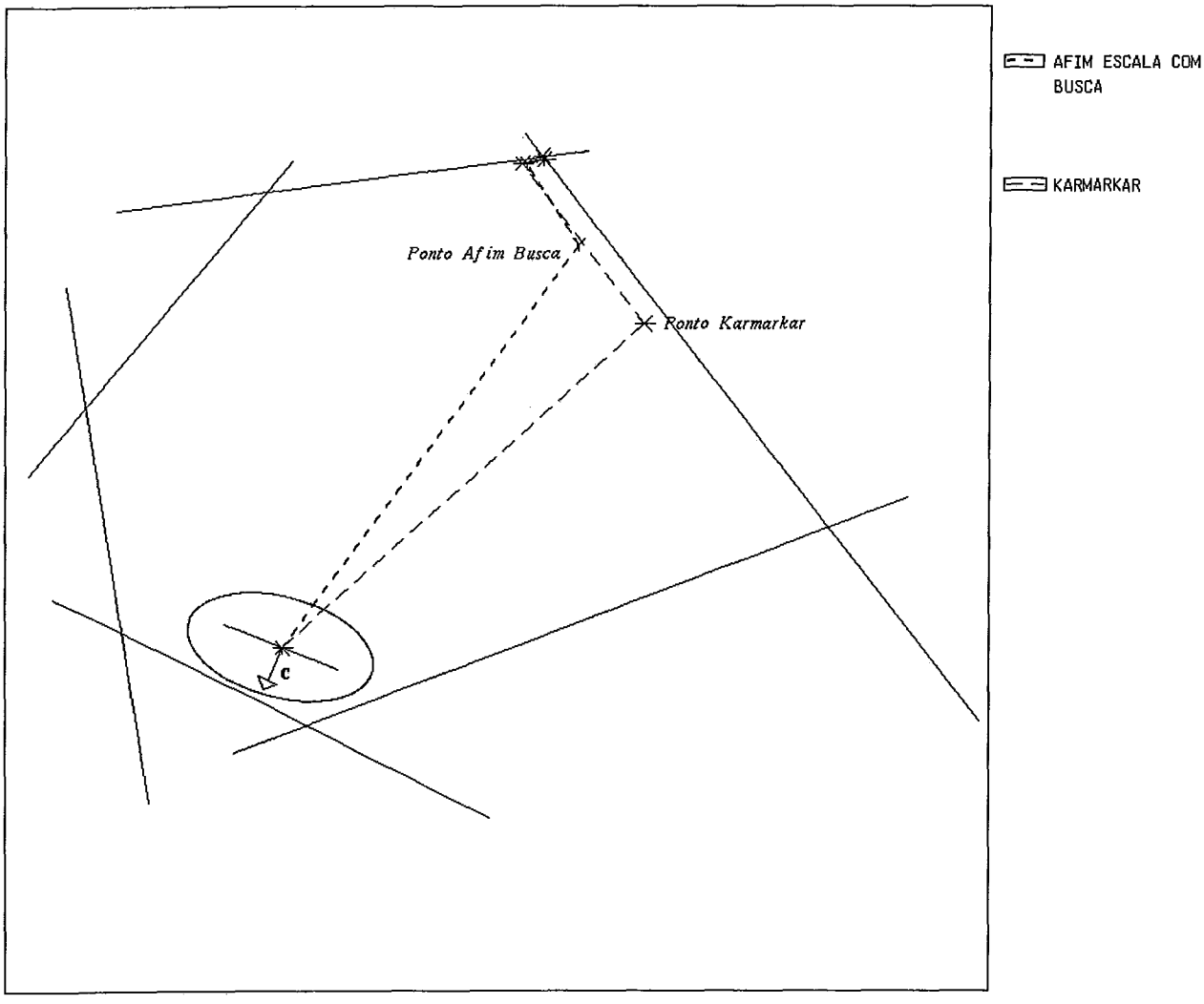


Figura III.4: Iterações dos Algoritmos Afim Escala com Busca e de Karmarkar até um ótimo

caminhará em cada um dos vértices do conjunto viável até um ótimo, ou seja, o algoritmo de Karmarkar é de complexidade $O(nL)$ iterações — ver as figuras III.6 e III.7 para um problema com 58 restrições e as figuras III.8 e III.9 para um problema com 16 restrições.

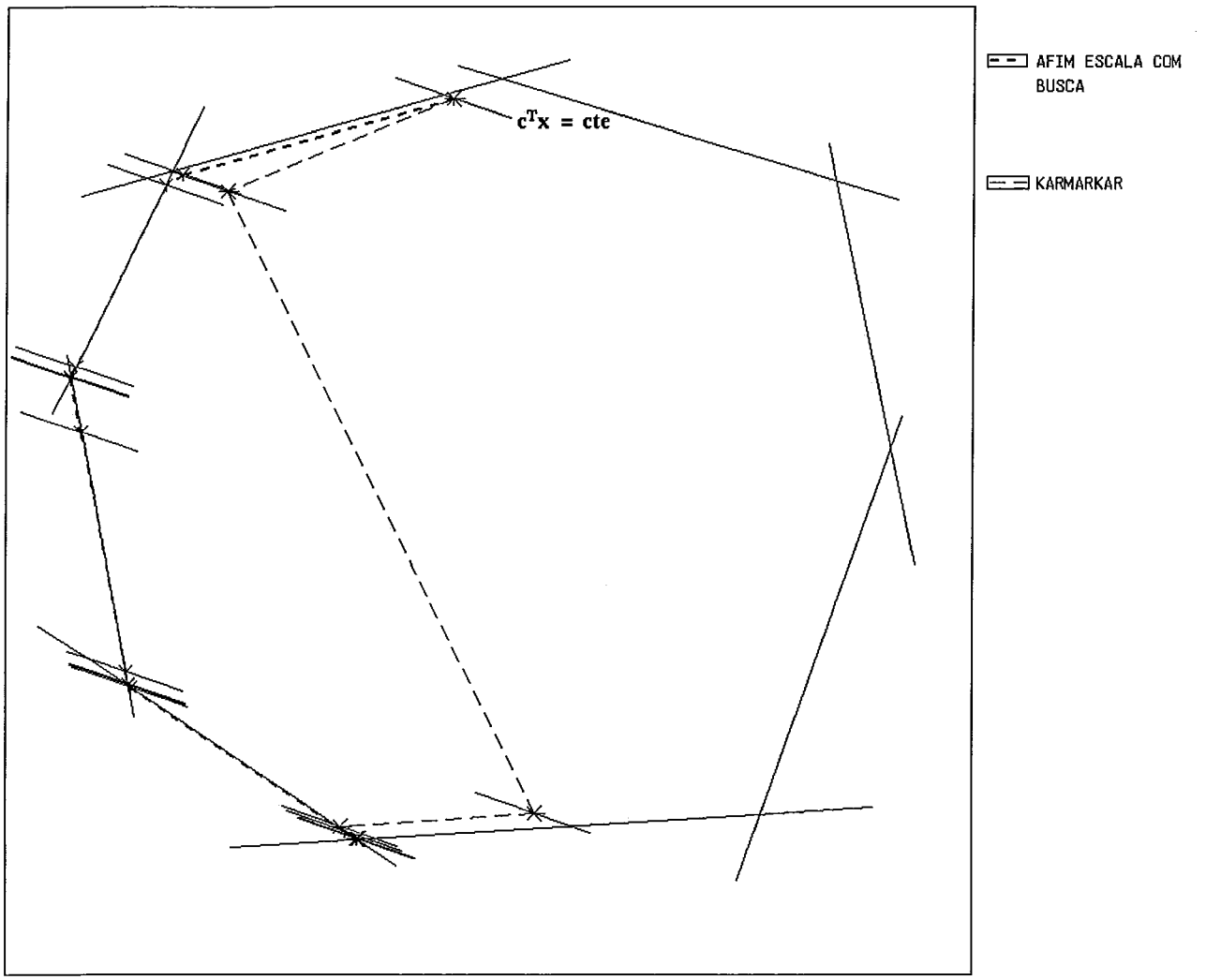


Figura III.5: Iterações dos Algoritmos Afim Escala com Busca e de Karmarkar até um ótimo

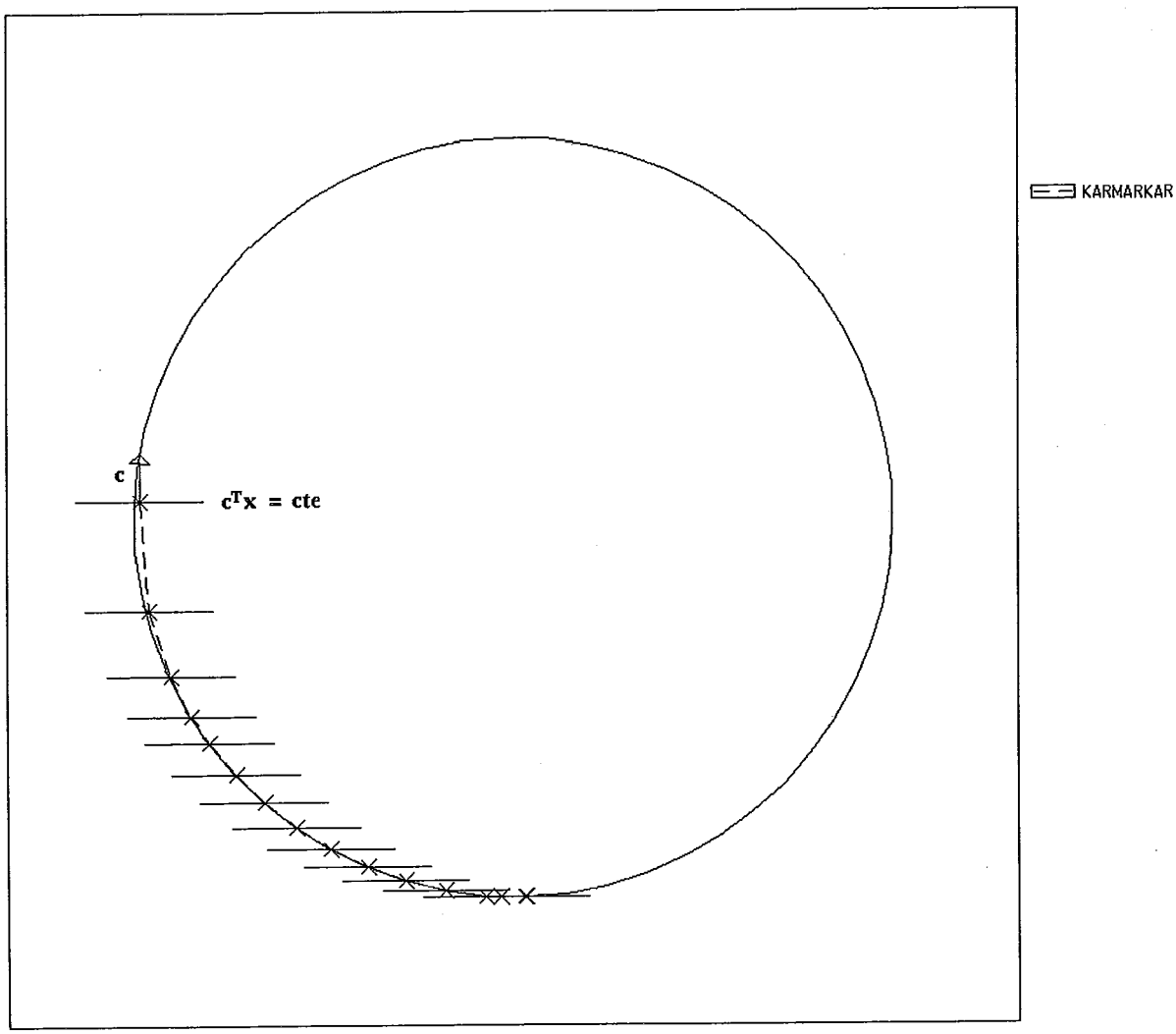


Figura III.6: Iterações do Algoritmo de Karmarkar para um polígono com 58 faces

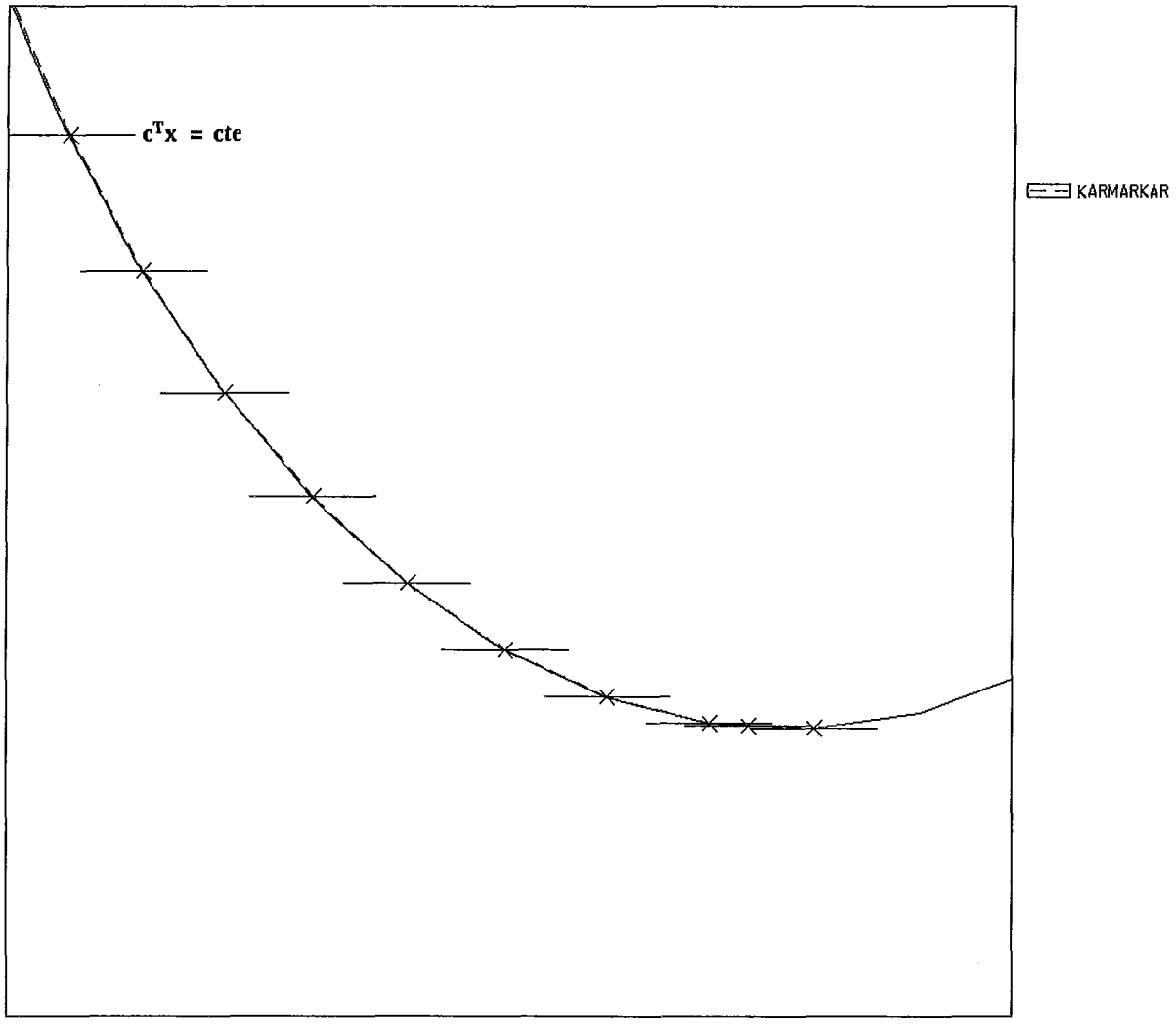


Figura III.7: Ampliação de algumas iterações do Algoritmo de Karmarkar para um polígono com 58 faces

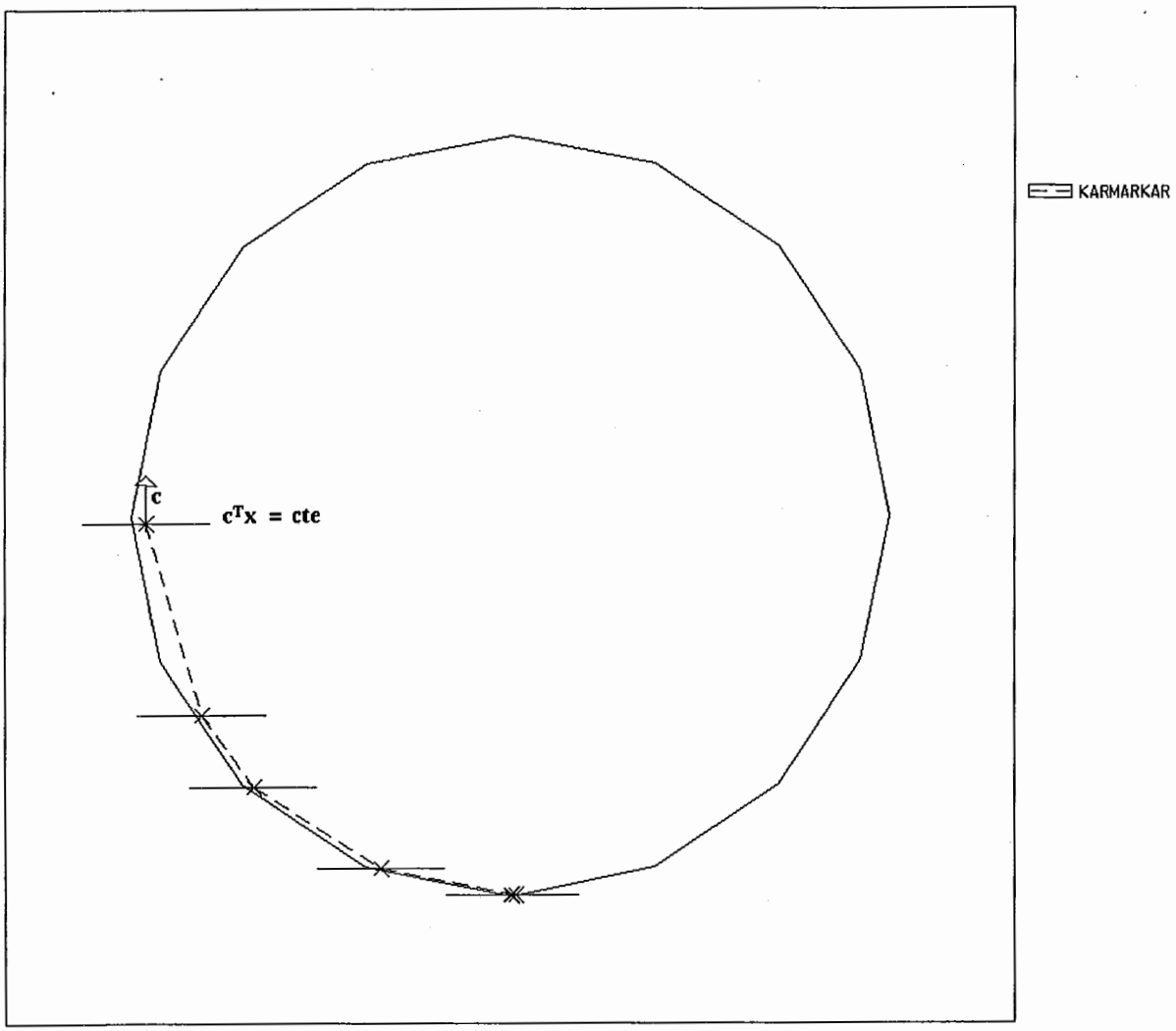


Figura III.8: Iterações do Algoritmo de Karmarkar para um polígono com 16 faces

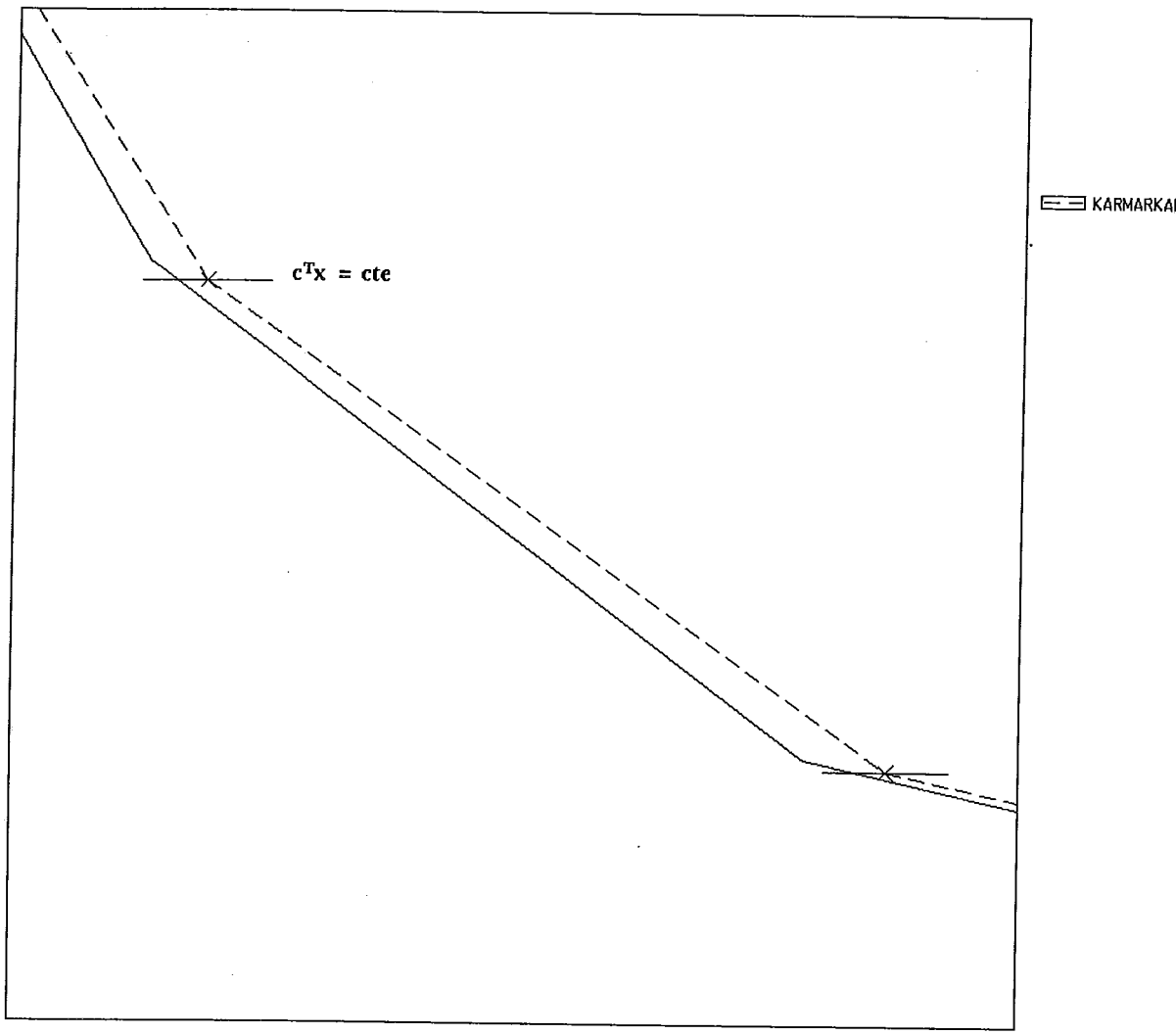


Figura III.9: Ampliação de algumas iterações do Algoritmo de Karmarkar para um polígono com 16 faces

Capítulo IV

Centro Analítico

IV.1 Centralização

Após a publicação [17] e subsequente implementação [18] dos algoritmos de Karmarkar e afim escala de Dikin [5], restava-se ainda a seguinte pergunta : quanto devemos nos afastar da fronteira para obtermos um algoritmo que resolva o problema de programação linear de maneira satisfatória? Enquanto, o algoritmo afim escala se utiliza de um recurso deselegante a fim de evitar a fronteira (deve-se acrescentar ao passo resultante da busca linear um fator heurístico δ — ver seção II.7); o algoritmo de Karmarkar evita a fronteira através do uso da função barreira. Mas, mesmo assim, o algoritmo pode levar a pontos que estejam muito próximos da fronteira, como ilustra a figura IV.1.

Como desejamos trabalhar no interior do conjunto, procurou-se encontrar um método que resolvesse o problema de programação linear mantendo-se o máximo possível afastado da fronteira. A solução encontrada foi através da definição de centro analítico do politopo dada por Sonnevend [25], o único ponto que minimiza a função barreira (veja a figura IV.2).

A utilização deste ponto é muito apropriada, pois uma curva formada pelos centros analíticos de todas as "fatias" de custo constante da região viável de (P) é bem comportada e apresenta propriedades que irão garantir melhoras em relação aos algoritmos citados anteriormente; denominou-se esta curva de *trajetória central.* Ao centro analítico de uma fatia de custo constante da região viável de (P),

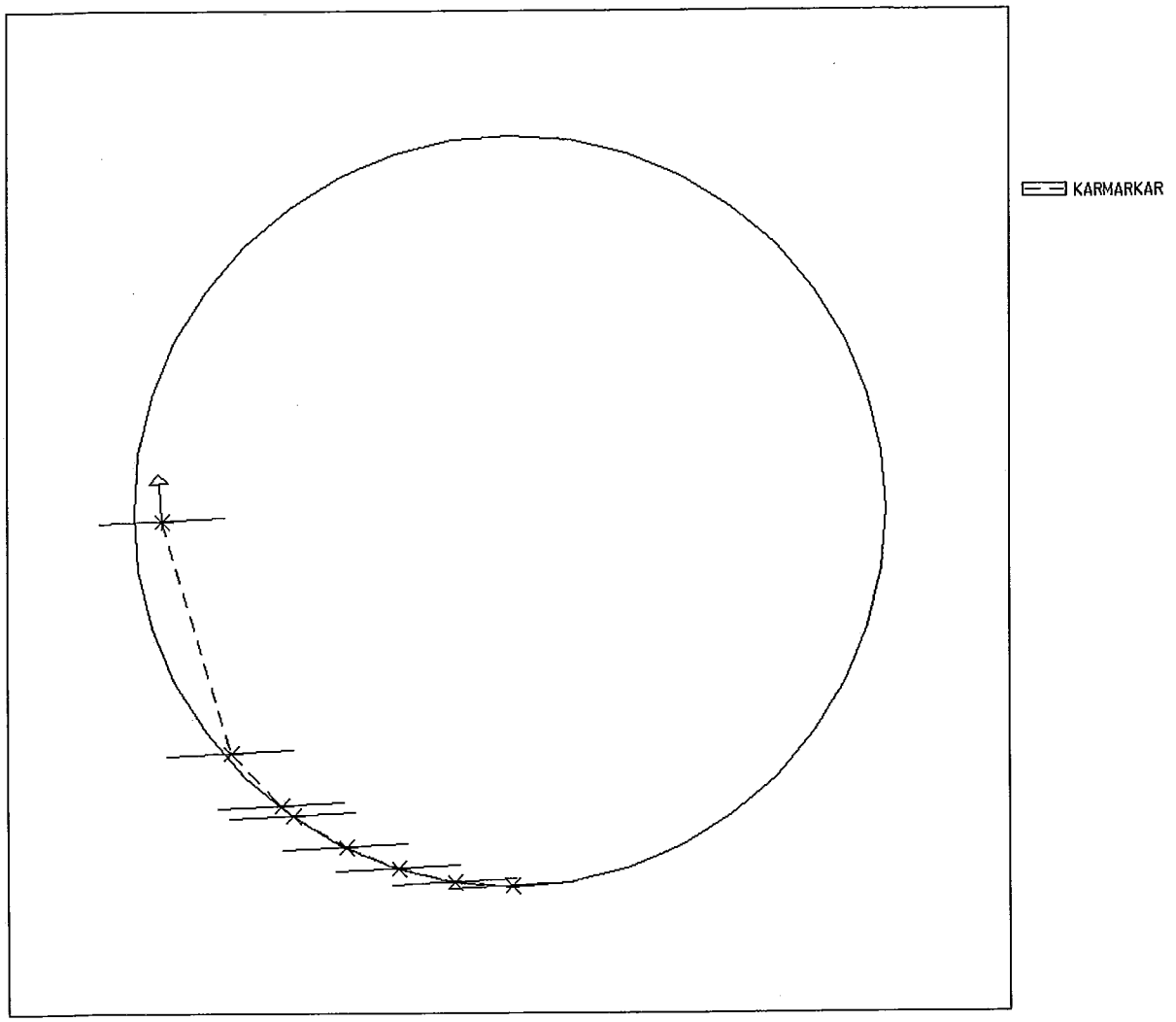


Figura IV.1: Politopo $Ax \leq b$ com Algoritmo de karmarkar

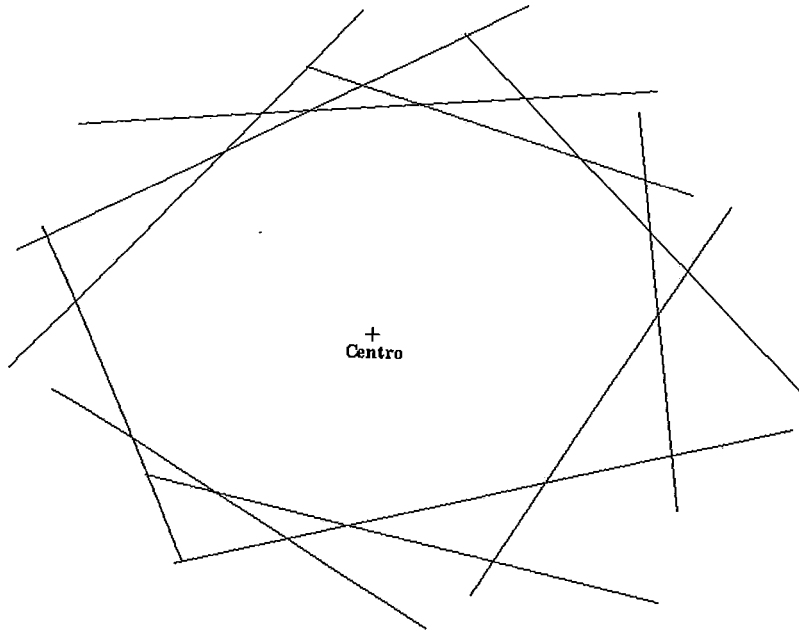


Figura IV.2: Politopo $Ax \leq b$ com Centro Analítico

denominou-se de *ponto central* da região viável de (P), ou seja, formalmente, temos que

Definição IV.1 *Centro Analítico.*

Considere o problema (P). O centro analítico $\tilde{x}$ *de* S *é o único ponto dado por :*

$$\tilde{x} = argmax\{\prod_{i=1}^{n} x_i \mid x \in \bar{S}\} = argmin\{\bar{p}(x) \mid x \in \bar{S}^0\} \qquad \text{(IV.1)}$$

Considere o problema (PD). Tendo em vista a definição da função barreira para um problema na formulação dual, *o centro é definido a partir das folgas :*

$$\tilde{x} = argmin\{\sum_{i=1}^{m} \log(b_i - A_i x) \mid x \in S^0\} \qquad \text{(IV.2)}$$

onde A_i *é a transposta do vetor linha da matriz* A.

Definição IV.2 *Ponto Central.*

Considere o problema (P) e $\bar{S}_K$ *como sendo conjuntos da forma*

$$\bar{S}_K = \{x \in \bar{S} \mid c^T x = K\} \qquad \text{(IV.3)}$$

esses conjuntos são "fatias" de custo constante.

Chamam-se de pontos centrais *do problema aos centros analíticos das "fatias" $\bar{S}_K$ com interior relativo não-vazio, i.e., para todo $K \in \Re$ tal que $\bar{S}_K^0 \neq \emptyset$, defini-se o ponto central :*

$$\bar{x}(K) = argmin\{\bar{p}(x) \mid Ax = b, c^T x = K, x > 0\} \tag{IV.4}$$

Considere o problema (PD). Neste caso a definição é análoga, ou seja, temos que :

$$x(K) = argmin\{p(x) \mid Ax \leq b, c^T x = K\} \tag{IV.5}$$

A figura IV.3 mostra pontos centrais.

IV.1.1 Caracterização de pontos centrais por função barreira

Voltando ao nosso problema inicial, problema (P), temos de levar em consideração que desejamos também reduzir o custo. Nós já temos um bom método de evitar a fronteira que é utilizar pontos centrais. Agora, como devemos proceder para obter uma função que englobe estes dois objetivos? Seguindo a idéia do método de Karmarkar [17], combinamos a função barreira e a função custo para obtermos uma função tradicionalmente conhecida na literatura por *função de penalização interna* :

$$\alpha \in \Re, x \in \Re^n_{++} \longmapsto \bar{f}_\alpha(x) = \alpha c^T x + \bar{p}(x) \tag{IV.6}$$

para um problema no *formato dual*, a definição é dada a partir das folgas :

$$\alpha \in \Re, x \in S^0 \longmapsto f_\alpha(x) = \alpha c^T x + p(x) \tag{IV.7}$$

Esta função foi estudada por Fiacco & McCormick no seu livro [7], e vem sendo utilizada extensivamente em todas as obras de referência de programação não-linear.

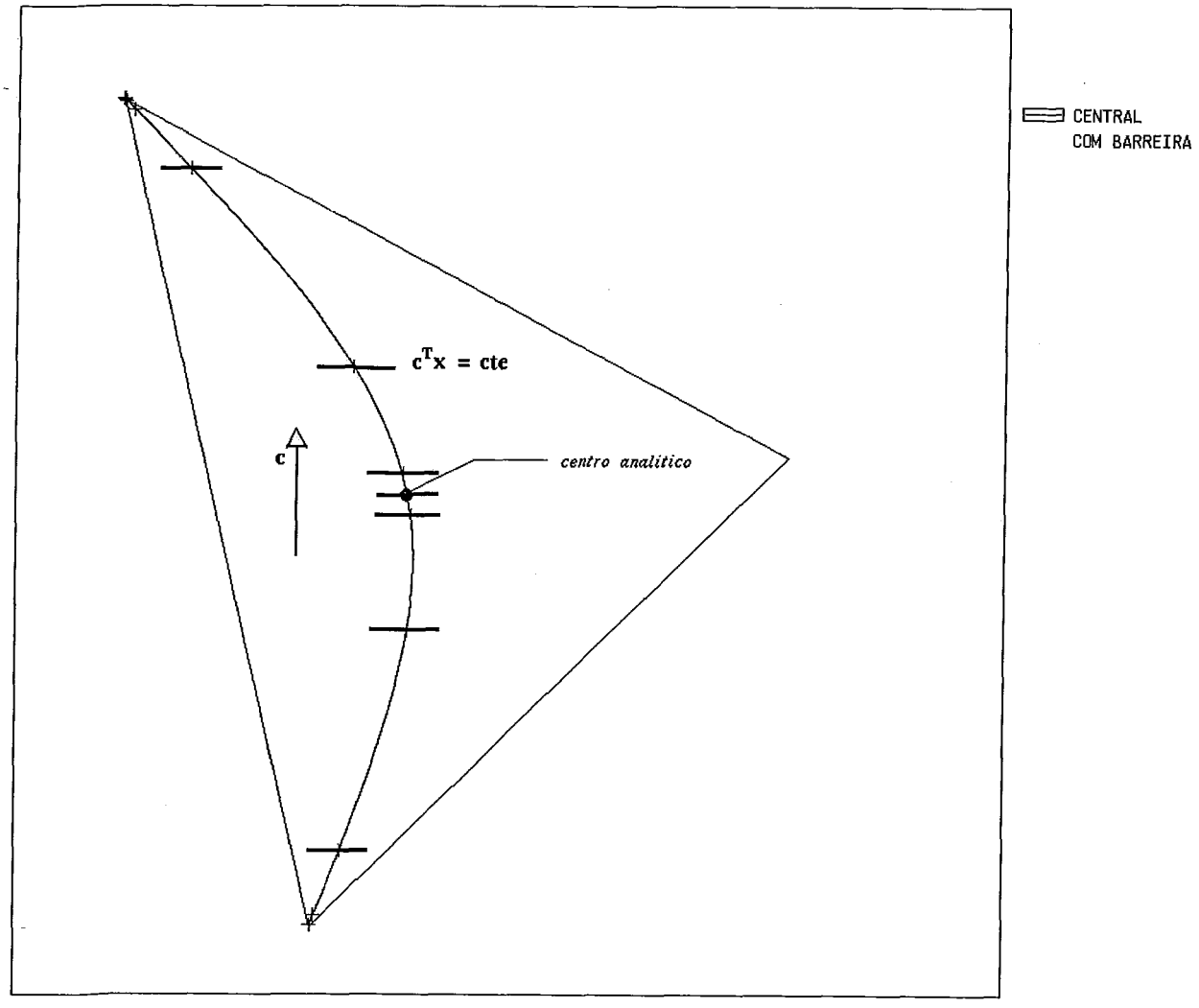


Figura IV.3: Politopo $Ax \leq b$ com Pontos Centrais

Em face do exposto acima, faz-se necessária uma nova definição de ponto central que é completamente equivalente à anterior (IV.4) : a cada valor do parâmetro α associe um ponto central $\bar{x}(\alpha)$ unicamente definido por

$$\bar{x}(\alpha) = \begin{array}{l} \text{argmin} \\ x \in \bar{S}^0 \end{array} \bar{f}_\alpha(x) \tag{IV.8}$$

e no formato dual :

$$x(\alpha) = \begin{array}{l} \text{argmin} \\ x \in S^0 \end{array} f_\alpha(x) \tag{IV.9}$$

A curva $\alpha \in \Re \longmapsto \bar{x}(\alpha)$ é a *trajetória central* para o problema de programação linear (P). Esta curva será estudada mais adiante no capítulo VII.

IV.1.2 Distância ao centro

Para obtermos um método eficiente de centralização (tanto na teoria como na prática) não precisamos calcular exatamente o ponto central, mas encontrar pontos "aproximadamente" centrais que estão a uma certa distância do centro e que satisfaçam a determinadas propriedades que garantem a eficiência do método de centralização. O critério de proximidade a ser adotado será :

Definição IV.3 *Distância ao centro para o problema (P)*

Dado $\alpha > 0, \alpha \in \Re$ *e* $x \in \bar{S}^0$, *a proximidade de* x *em relação a* $\bar{x}(\alpha)$ *é caracterizada por :*

$$\bar{\delta}(x,\alpha) = \| \bar{h}_N(x,\alpha) \|$$

onde $\bar{h}_N$ *é a direção de Newton-Raphson após a mudança de escala, ou seja :*

$$\bar{h}_N(x,\alpha) = -P_{\bar{A}X} X \nabla \bar{f}_\alpha(x)$$

onde X *é a matriz da mudança de escala dada por :* $X = diag(x_1, \ldots, x_n)$.

O ponto x será considerado "aproximadamente" central se dado $\epsilon \in (0, 0.5)$, temos que :

$$\bar{\delta}(x,\alpha) < \epsilon.$$

Observe que agora podemos reescrever (IV.8) como : dado $x^0 \in \bar{S}^0$ e $\epsilon \in (0, 0.5)$, encontre $x \in \bar{S}$ tal que

$$\bar{\delta}(x,\alpha) < \epsilon.$$

Para obtermos a direção de Newton-Raphson h_N para o problema dual, após a mudança de escala, é necessário que a partir de $\bar{h}_N$ utilizemos as regras de tradução dadas no capítulo VIII.

Da definição anterior, sabemos que dados $\alpha \in \Re_+$ e $\bar{x} \in \bar{S}^0$:

$$\bar{h}_N(\bar{x},\alpha) = -P_{\bar{A}X} X \nabla \bar{f}_\alpha(x) = -P_{\bar{A}X}(\alpha \bar{c} - e) = -\alpha \bar{c}_p + \bar{e}_p,$$

onde X é a matriz mudança de escala como dada na definição IV.3, e $\bar{c}_p$ e e_p correspondem respectivamente a $P_{\bar{A}X}Xc$ e $P_{\bar{A}X}e$.

Do lema I.4 temos que :

$$h_N(x,\alpha) = -\alpha d_c + d_e \tag{IV.10}$$

onde $x \in S^0$, d_c e d_e definidos como em (I.1) e (I.2), respectivamente (ver lema I.4), após a mudança de escala dada no lema I.5.

Observe que a partir de (IV.10) podemos obter $\bar{h}_N$ através do resultado do lema I.3. Assim, temos que :

$$\bar{h}_N = -A_k h_N$$

onde a matriz A_k é a matriz A como definida no problema (PD) após a mudança de escala dada no Lema I.5.

Agora, podemos definir proximidade ao centro para um problema dado no *formato dual* :

Definição IV.4 *Distância ao centro para o problema (PD)*

Considere o problema (PD). Dados $\alpha > 0, \alpha \in \Re$ *e* $x \in S^0$, *a proximidade em relação a* $x(\alpha)$ *é caracterizada por :*

$$\delta(x,\alpha) = \| -A_k(-\alpha d_c + d_e) \|$$

onde d_c *e* d_e *definidos como em (I.1) e (I.2), respectivamente (ver lema I.4), com a matriz* A_k *no lugar da matriz A após a mudança de escala dada no lema I.5. Observe que foi necessário utilizar a direção primal correspondente à direção dual de Newton-Raphson* h_N, *pois a definição de proximidade é feita para o problema primal — ver definição IV.3.*

IV.1.3 Método para se determinar o centro da região viável

No capítulo anterior, quando se estudou a função barreira (III.1), mencionou-se o fato de que a variação da função barreira não se altera com a mudança de escala e de

que a matriz hessiana da função barreira é a identidade quando no ponto e. Devido a essas propriedades, a direção de máximo declive no ponto e coincide com a direção de Newton-Raphson. Logo, o algoritmo afim-escala e o método de Newton-Raphson com buscas lineares coincidem para a função barreira — ver Gonzaga [15].

Então, basta aplicar o método de Newton-Raphson com buscas lineares para o problema dado em (IV.8); e se o problema original estiver no formato dual, faz-se o mesmo para o problema dado em (IV.9). O algoritmo resultante é eficiente tanto do ponto de vista teórico como de implementação, a sua complexidade foi estudada por Vaidya [27].

IV.1.4 Direção de Centralização para o problema (PD)

No caso da centralização, $\alpha = 0$, logo temos que a direção de centralização h_C para um problema no formato dual será dada por :

$$h_C = h(x, 0) = d_e \tag{IV.11}$$

onde $h_C \in \Re^n$ e $d_e \in \Re^n$ está dada como em (I.2) — ver lema I.4 — após a mudança de escala dada no lema I.5.

IV.1.5 Busca Linear

A busca linear na direção h_C pode ser feita por qualquer método de minimização unidimensional, como Armijo ou Bisseção. No nosso caso, resolvemos optar pelo método da Bisseção, pois desejamos obter soluções com uma precisão melhor do que a fornecida pelo método de Armijo. Formalmente a busca será dada por :

$$\bar{\lambda} = \text{argmin}\{f_0(x + \lambda h_C) \mid x + \lambda h_C \in S^0, \lambda \geq 0\}, \bar{\lambda} \in \Re$$

IV.1.6 Algoritmo de Centralização

Algoritmo IV.1 Algoritmo Centralização: *Dado* $x^0 \in S^0$.

$k = 0$

Repita

Cálculo das Folgas : $z^k = b - Ax^k$;

Mudança de Escala :

$$Z_k = \text{diag}(z_1^k, z_2^k, \ldots, z_m^k)$$

$$A_k = Z_k^{-1} A;$$

Direção :

$$h_C = d_e = -(A_k^T A_k)^{-1} A_k^T e;$$

Cálculo da distância ao Centro : $\bar{\delta} = \delta(x^k, 0)$ — ver definição IV.4;

Busca : resolver aproximadamente

$$\bar{\lambda} = \text{argmin}\{f_0(x^k + \lambda h_C) \mid x^k + \lambda h_C \in S^0, \lambda \geq 0\}, \bar{\lambda} \in \Re;$$

Novo ponto :

$$x^{k+1} = x^k + \bar{\lambda} h_C$$

Até que $(\bar{\delta} < 0.01)$

IV.2 Curvas de Nível da Função Barreira

A partir da definição de centro analítico, pode-se obter um algoritmo para se visualizar graficamente as curvas de nível para um problema bidimensional.

Considere o problema (PD). As curvas de nível da função barreira p, que denotaremos pelo conjunto $\mathcal{X}$, são definidas por

$$\mathcal{X} = \{x \in S^0 \mid p(x) = K\}, \text{ onde } K \in \Re, \text{ K uma constante.}$$

Agora, definimos a seguinte função :

$$K \in \Re, x \in S^0 \longmapsto f_K(x) = p(x) - K$$

Então, podemos enunciar o conjunto $\mathcal{X}$ das curvas de nível como :

$$\mathcal{X}_K = \{x \in S^0 \mid f_K(x) = 0\}, K \in \Re. \tag{IV.12}$$

O método para se traçar as curvas de nível consiste em obter direções que variem num círculo centrado no ponto analítico (que chamaremos de raias), e encontrar os pontos que satisfaçam (IV.12).

Agora, vamos definir o que a vem ser *raias.*

Definição IV.5 *Seja q o número de raias dado. Vamos chamar de* raias *aos vetores $h \in \Re^2$ tais que :*

$$h_j = (\cos\frac{2\pi}{j}, \sin\frac{2\pi}{j})$$

onde $j \in [1, \ldots, q]$.

Seja $\tilde{x} \in S^0$ o centro analítico do problema (PD). O método para se encontrar as curvas de nível da função barreira p é o seguinte :

Método : Encontrar as curvas de nível da função barreira

Seja $K \in \Re$, K dado e q o número de raias dado.

1. a partir de $\tilde{x}$, obtenha raias $h_j \in \Re^2$ definidas como na definição IV.5, com $j \in [1, \ldots, q]$;
2. para cada $j \in [1, \ldots, m]$ calcule o maior passo $\mu_j h_j$ possível até a fronteira pelo teste da razão;
3. na direção do vetor h_j, encontre $x \in \mathcal{X}_K$ como definido em (IV.12) para cada $j \in [1, \ldots, q]$.

IV.2.1 Busca Linear

Da expressão (IV.12), observe que para cada raia $h_j \in \Re^2$, $j \in [1, \ldots, m]$ temos de encontrar λ_j tal que :

$$f_{K_0}(x + \lambda_j h_j) = 0. \tag{IV.13}$$

com $\lambda_j \in [0, \mu_j]$ e μ_j é o maior passo possível até a fronteira na direção do vetor h_j.

Para se solucionar (IV.13), utilizou-se o método da bisseção.

IV.2.2 Algoritmo para Traçar Curvas de Nível da Função Barreira p

A escolha de um K inicial também é importante, e é muito simples : seja $\epsilon_0 > 0$ dado, tome :

$$K_0 = p(\tilde{x}) + \epsilon_0.$$

Agora, variando-se o parâmetro K, pode se obter pontos das diferentes curvas de nível da função barreira p.

Então, o modelo do algoritmo será :

Algoritmo IV.2 Algoritmo Curvas de Nível da Função Barreira p :
Dados $x^0 \in S^0$, $\epsilon = 0.01$, $\epsilon_0 \in \Re_+$ *e* q *o número de raias.*
Use o algoritmo IV.1 e obtenha $\tilde{x}$ com distância ao centro dada por ϵ;
$k = 0$;
$K_0 = p(\tilde{x}) + \epsilon_0$;
A partir de $\tilde{x}$ obtenha $h_j \in \Re^2$ como dadas na definição IV.5, com $j = [1, \ldots, q]$;
Repita

Busca : resolver o problema de encontrar aproximadamente λ_j

$$f_{K_0}(x + \lambda_j h_j) = 0$$

com $\lambda_j \in [0, \mu_j]$, onde μ_j é o maior passo possível até a fronteira na direção do vetor h_j, para $j = [1, \ldots, q]$;

Traçar uma curva de nível com os pontos obtidos — ver (VIII.1);

Atualização : $K_{k+1} = K_k + \epsilon_0$;

$k = k + 1$;

Até que $k >$ número de curvas desejadas na visualização gráfica

Veja o capítulo VIII, para uma implementação mais eficiente do algoritmo dado acima.

IV.3 Figuras

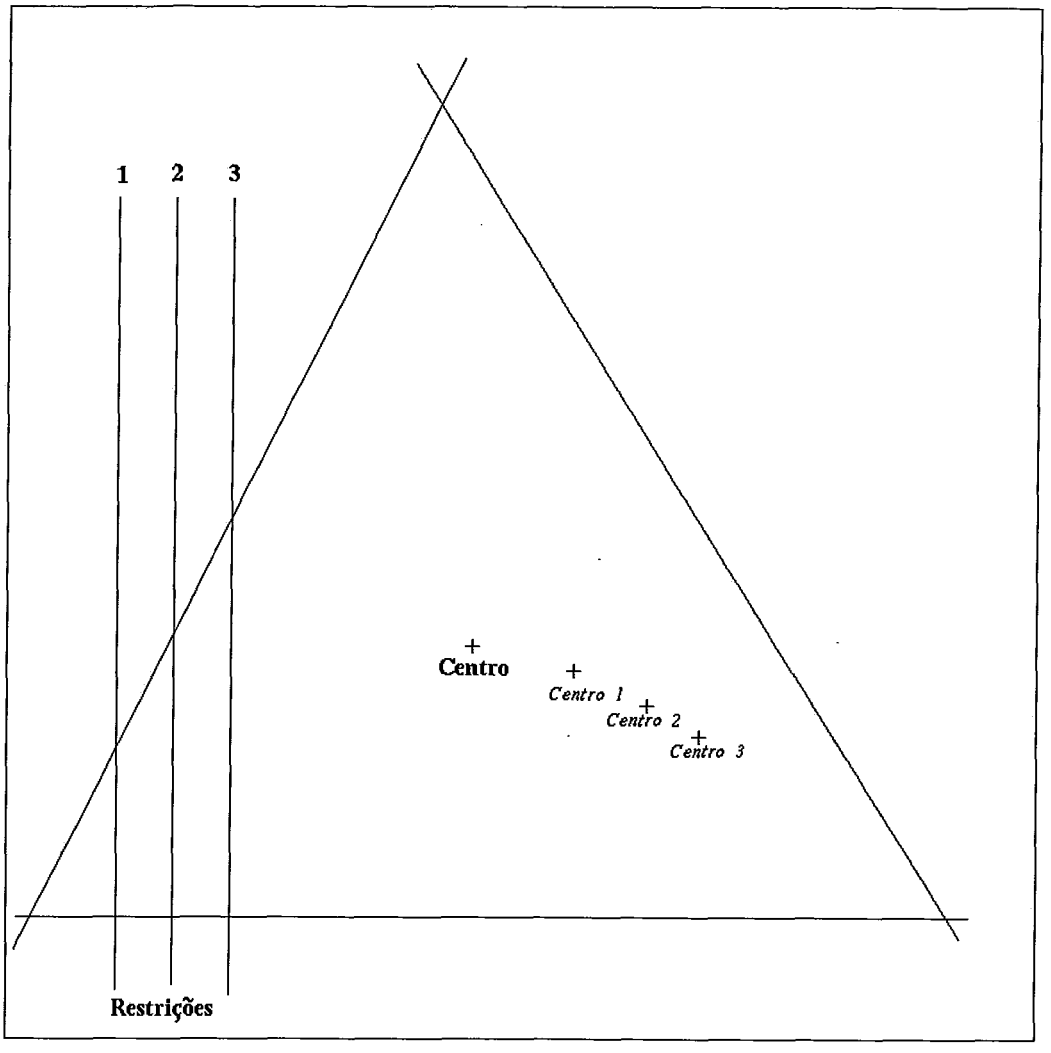


Figura IV.4: Região Viável $Ax \leq b$ com Centro Analítico

A figura IV.4 mostra como o Centro Analítico varia com a adição de restrições ao conjunto viável. Repare que essas restrições não alteram a região viável inicial, mas apesar disso, o centro analítico muda o que não aconteceria se o centro fosse geométrico.

Um resultado interessante é que se utilizarmos o algoritmo Afim Escala com um passo muito pequeno os pontos gerados estarão sobre a trajetória central. Esse resultado foi demonstrado por Monteiro, Adler e Rezende [23] e está representado na figura IV.5.

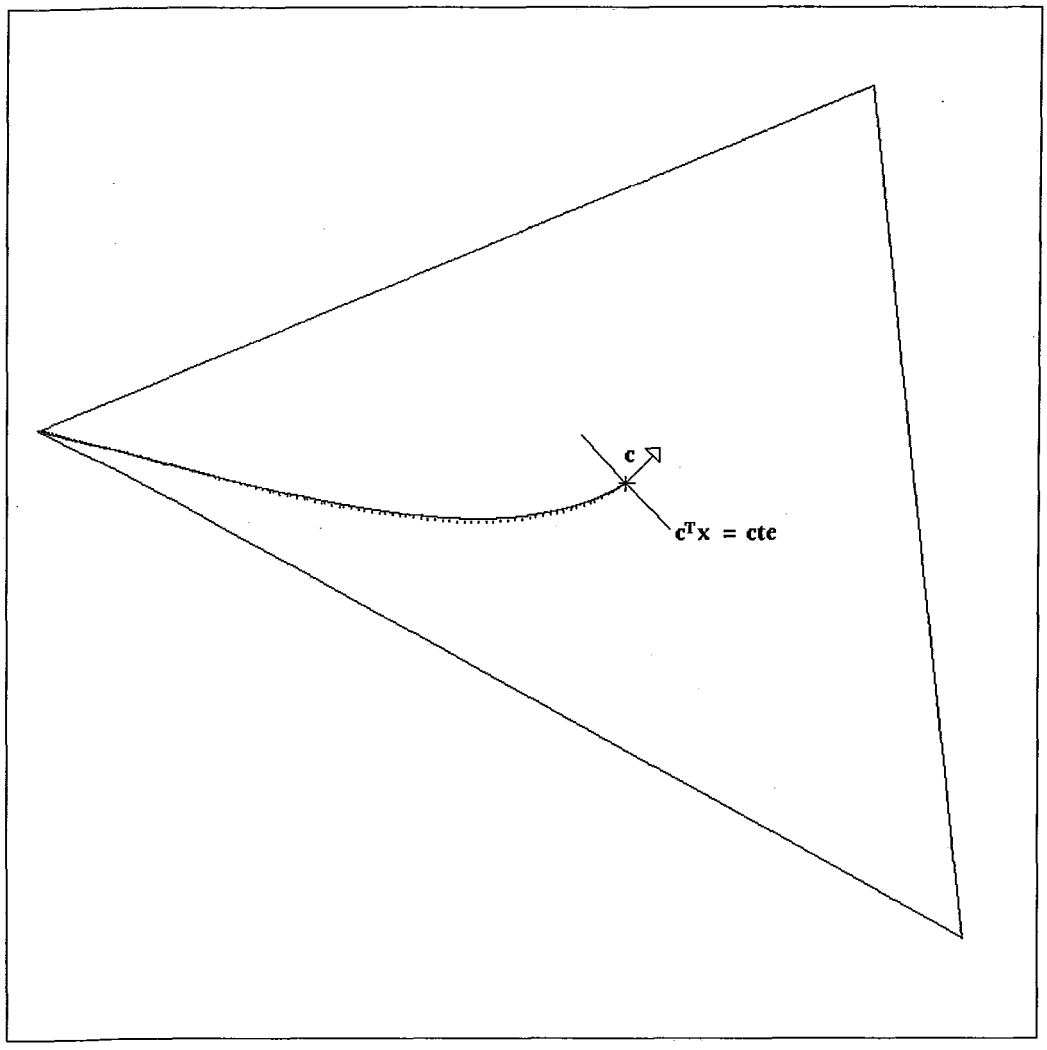


Figura IV.5: Iterações do Algoritmo Afim Escala com Passo Unitário e Trajetória Central

As figuras IV.6, IV.7, IV.8 mostram curvas de nível para diferentes regiões e foram obtidas a partir do algoritmo IV.2 com as modificações dadas na seção VIII.1.

As figuras IV.9, IV.10, IV.11 mostram curvas de nível para diferentes regiões com a Trajetória Central, repare que a trajetória é analítica e não geométrica.

As figuras IV.12 e IV.13 e IV.14 mostram a trajetória central para o seguinte problema :

$$\begin{array}{ll} \text{minimizar} & c^T x \\ \text{sujeito a} & Ax \leq b \end{array}$$

onde $c \in \Re^3, x \in \Re^3$, $A \in \Re^{4\times 3}$ é uma matriz de rank máximo e de dimensão 4×3.

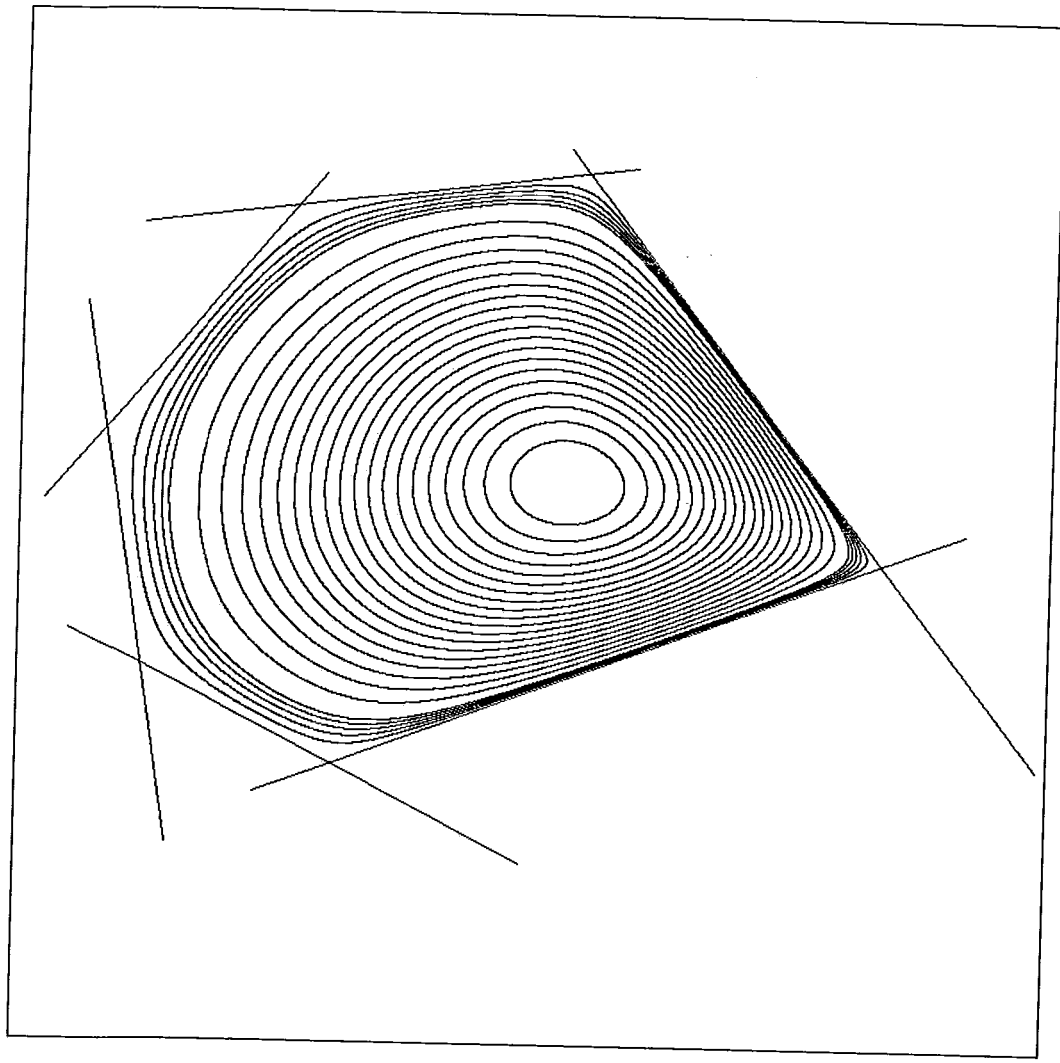


Figura IV.6: Região Viável $Ax \leq b$ com Curvas de Nível

A matriz A é dada por :

$$A = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

e o vetor b por :

$$b = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Para cada uma das figuras, varia-se o vetor de custo c.

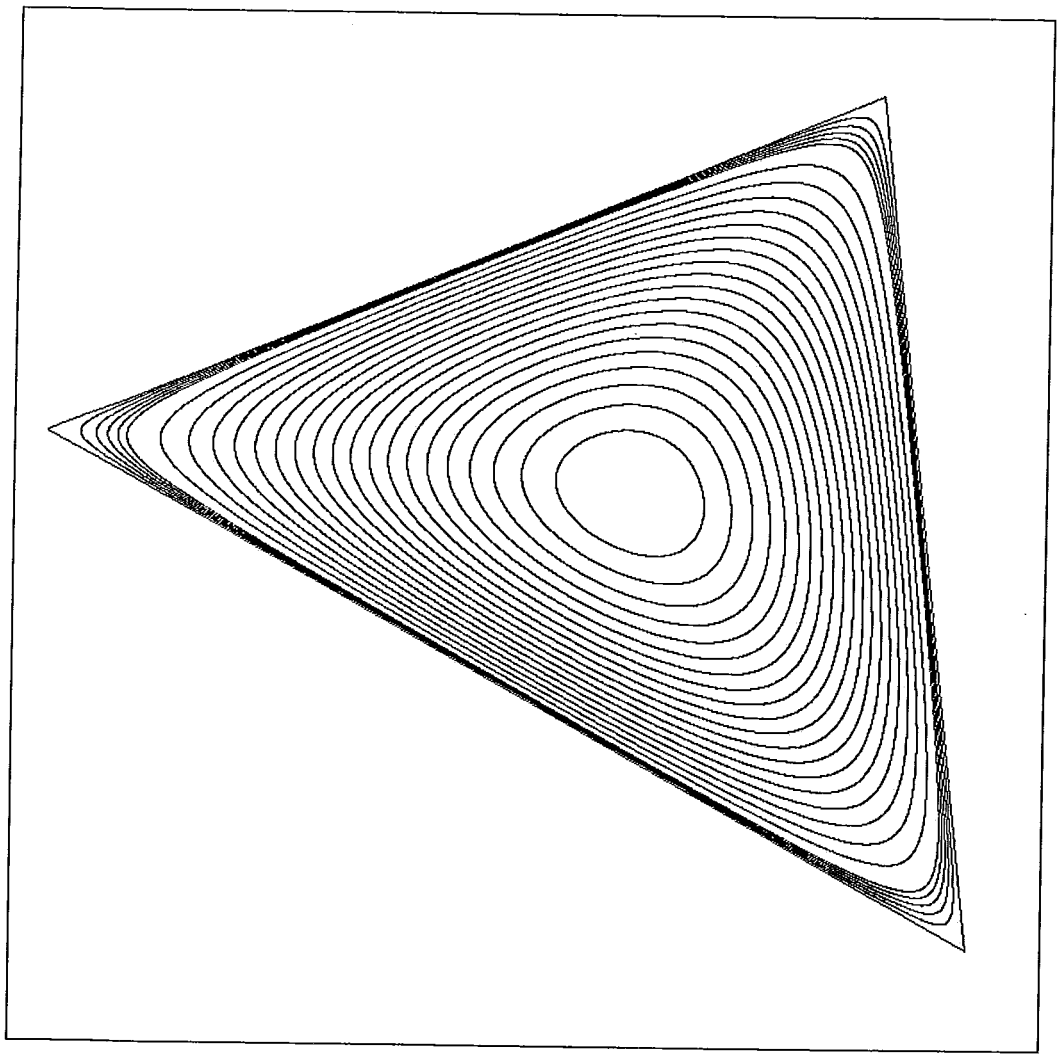


Figura IV.7: Região Viável $Ax \leq b$ com Curvas de Nível

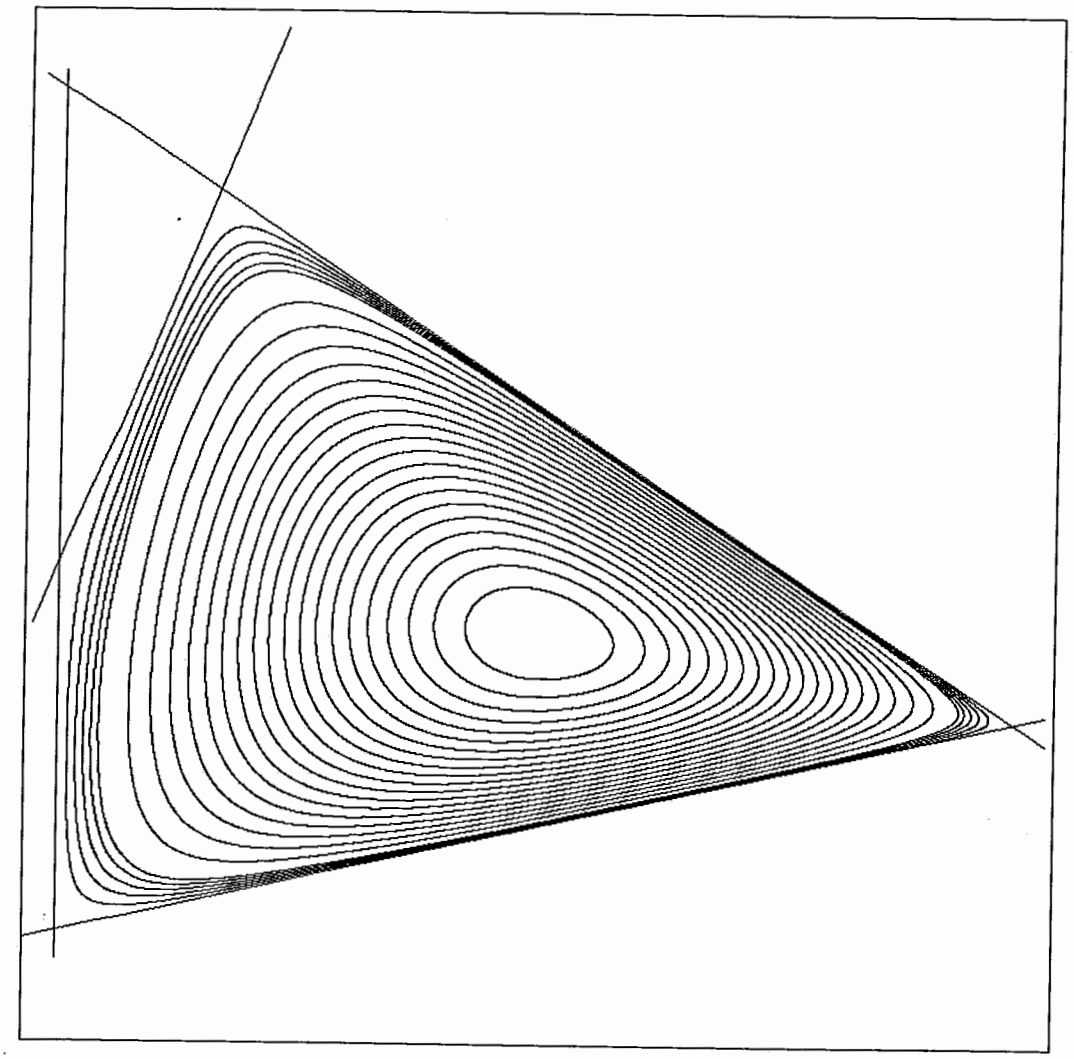


Figura IV.8: Região Viável $Ax \leq b$ com Curvas de Nível

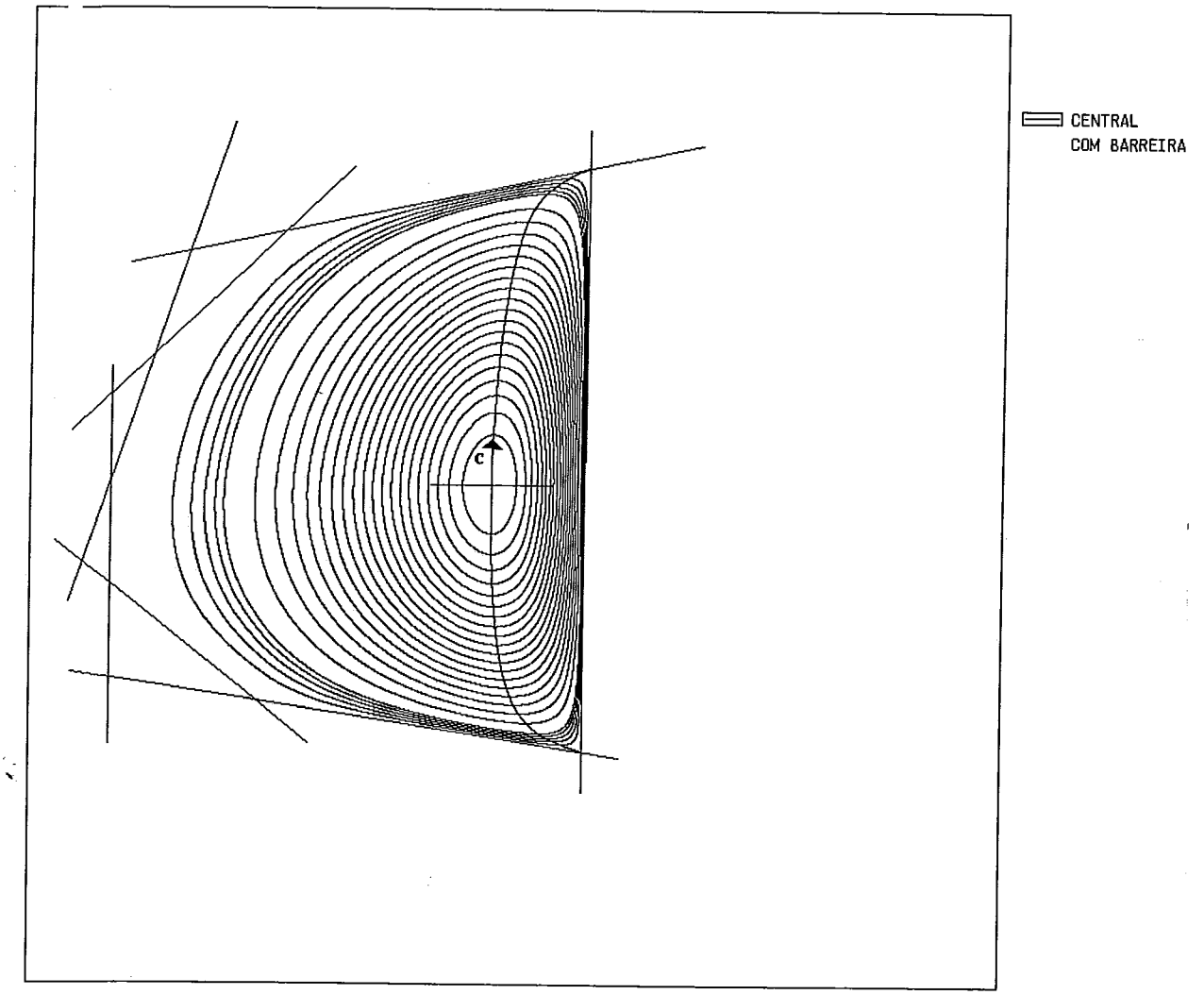


Figura IV.9: Região Viável $Ax \leq b$ com Curvas de Nível e Trajetória Central

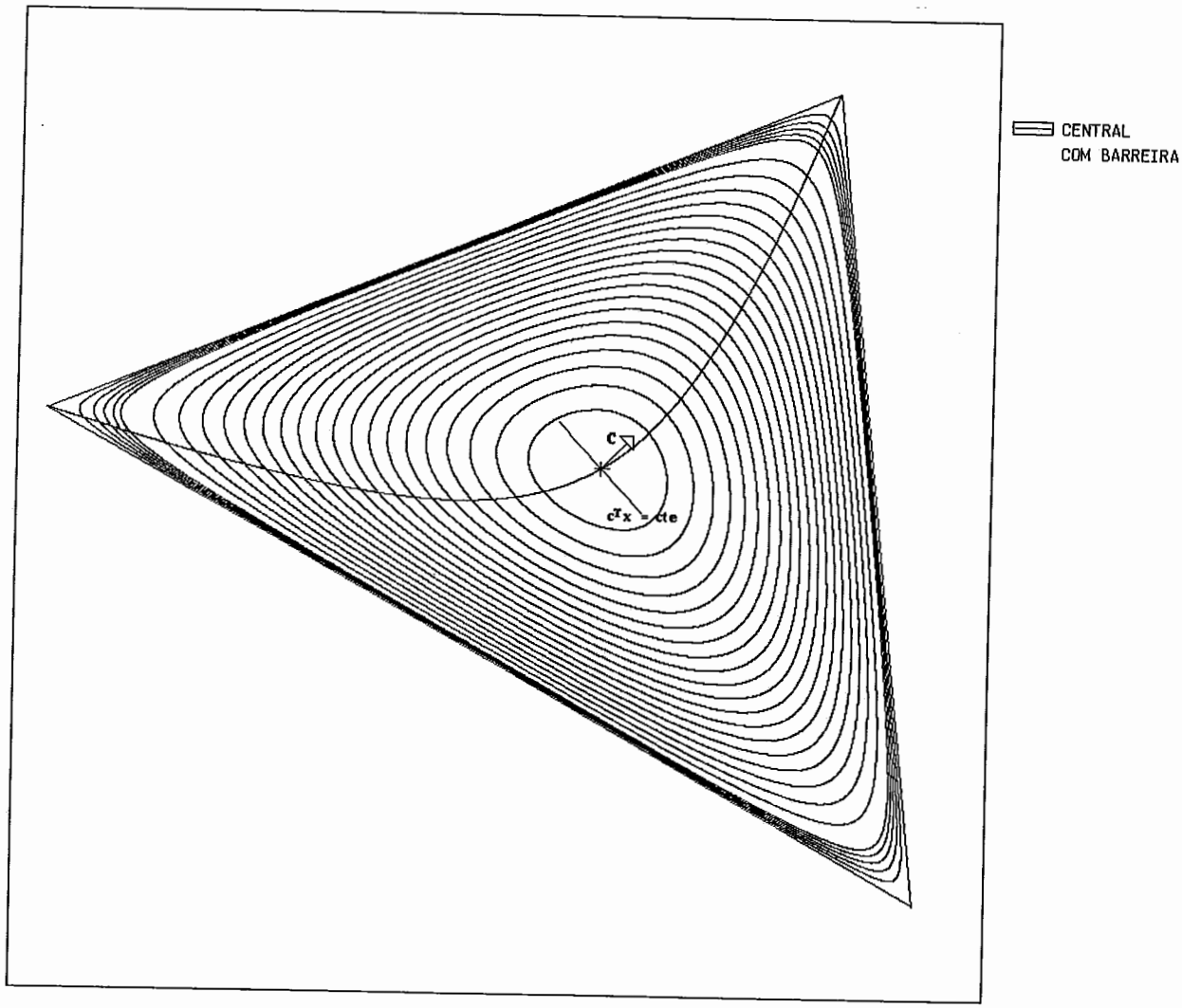


Figura IV.10: Região Viável $Ax \leq b$ com Curvas de Nível e Trajetória Central

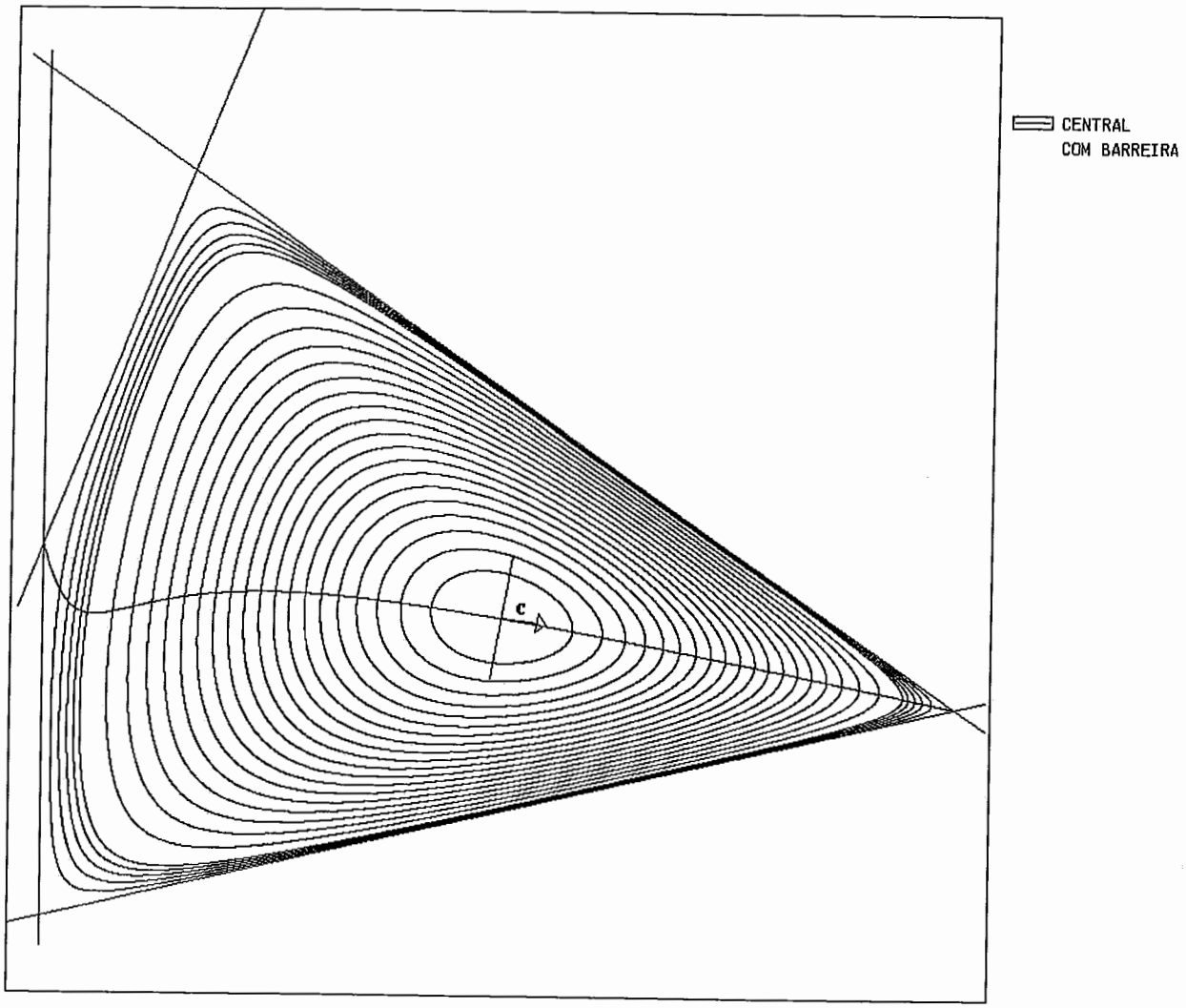


Figura IV.11: Região Viável $Ax \leq b$ com Curvas de Nível e Trajetória Central

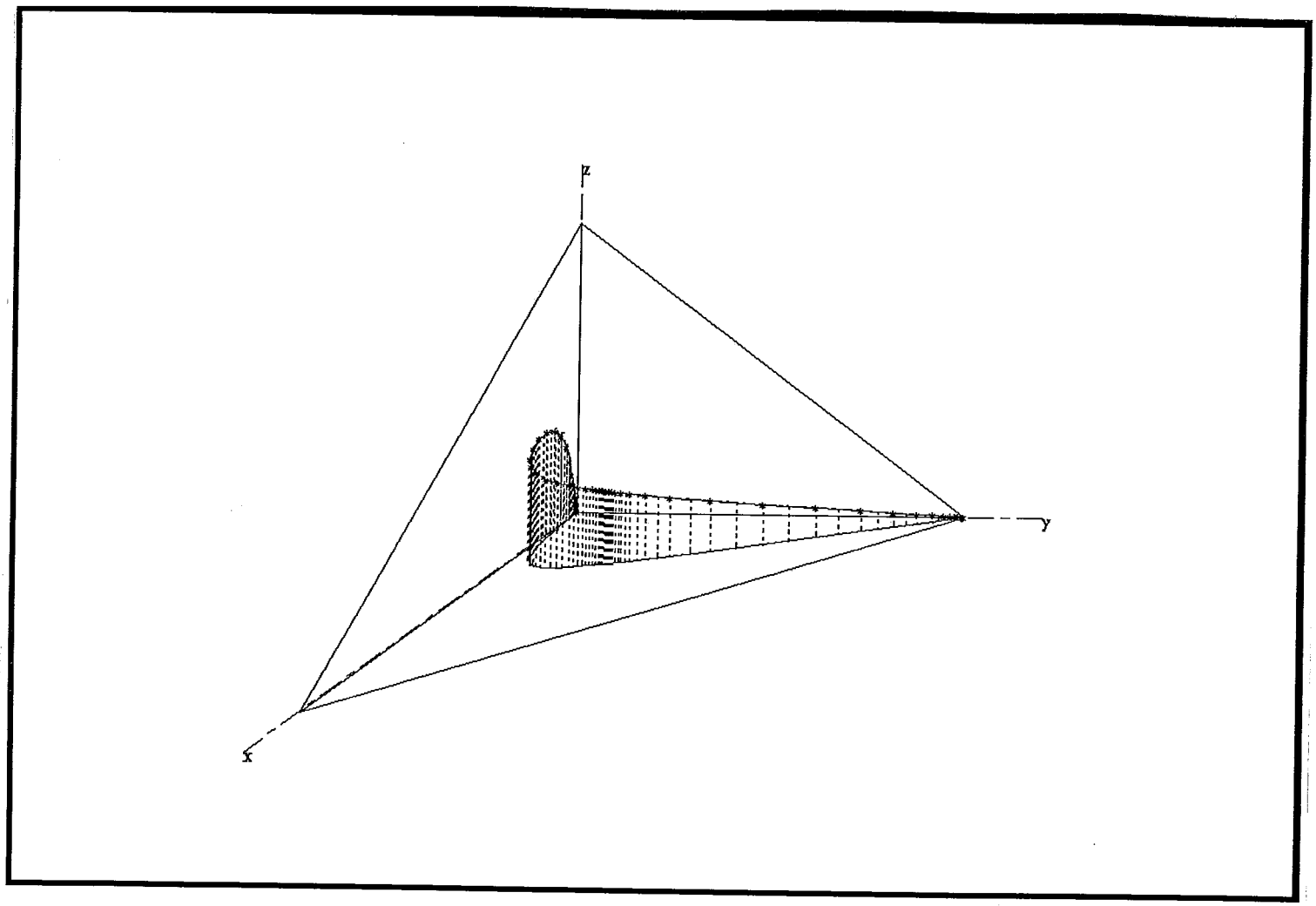


Figura IV.12: Região Viável $Ax \leq b$ com Trajetória Central no espaço $\Re^3$

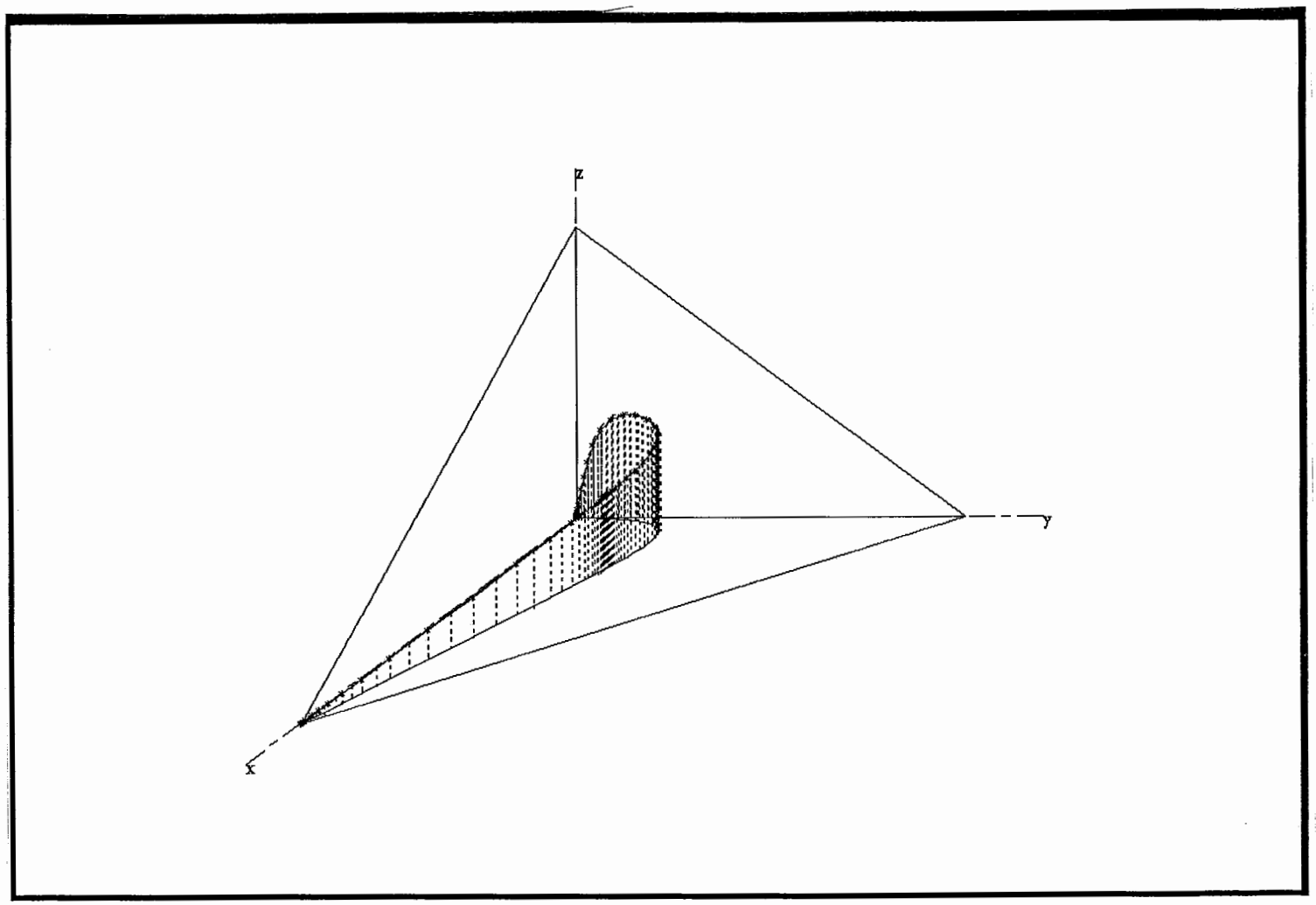


Figura IV.13: Região Viável $Ax \leq b$ com Trajetória Central no espaço $\Re^3$

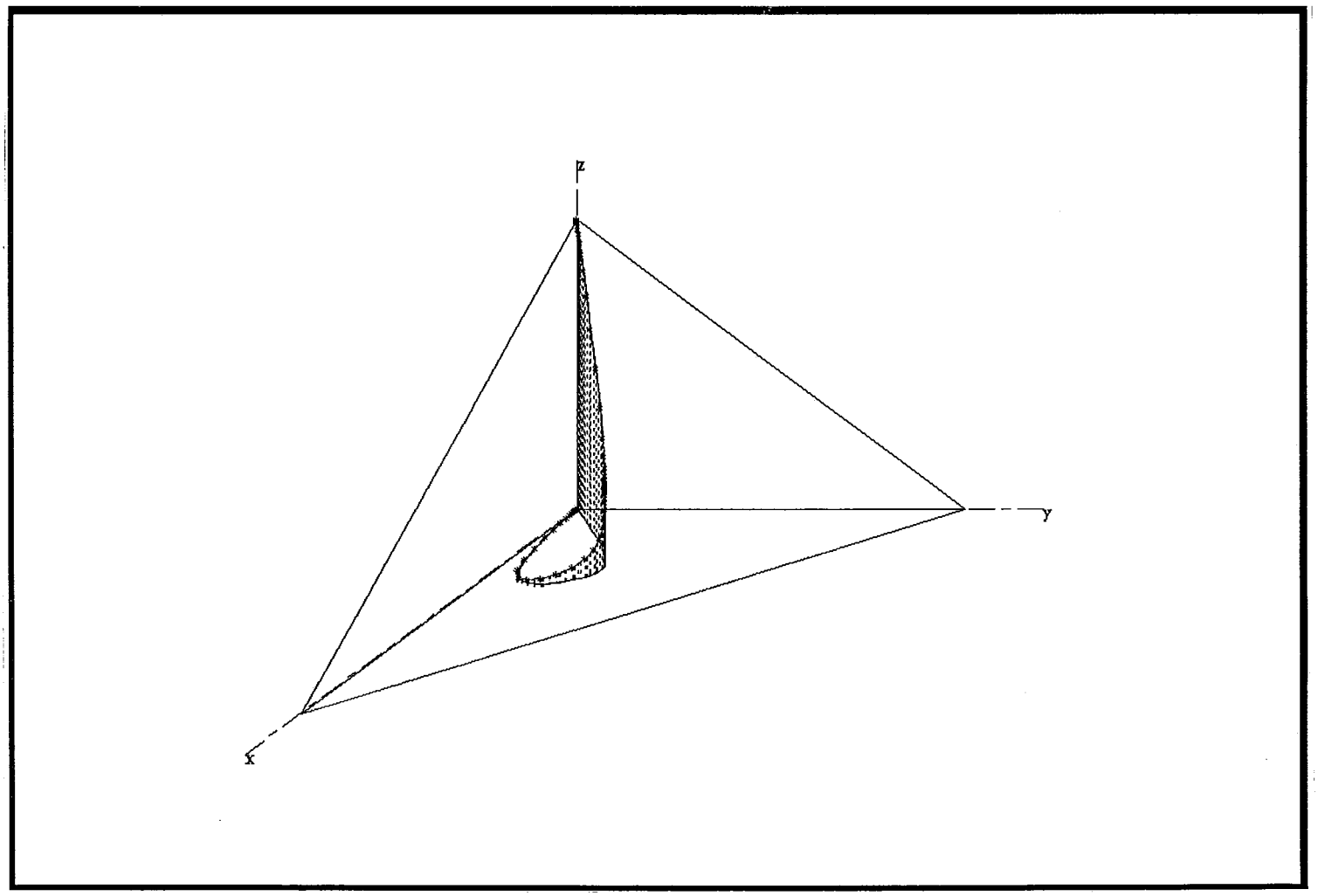


Figura IV.14: Região Viável $Ax \leq b$ com Trajetória Central no espaço $\Re^3$

Capítulo V

Algoritmo de Barnes & Jensen

Após se verificar no capítulo IV que "caminhar" pelos pontos centrais da região viável era uma boa maneira de se evitar a fronteira, surgiu a motivação para se desenvolver algoritmos que caminhem sobre a trajetória central. Esses algoritmos são conhecidos como *algoritmos de trajetória central.*

O primeiro algoritmo de trajetória central foi idealizado por Barnes [2] a partir da seguinte idéia : o algoritmo afim escala (ver capítulo II) aproxima os pontos gerados pelo algoritmo da fronteira da região viável, porém se após uma iteração do algoritmo afim escala mantivermos o *custo constante* e encontrarmos o *ponto central* correspondente a este custo, além de se evitar a fronteira, aumenta-se o elipsóide que será gerado pelo algoritmo afim escala na próxima iteração — ver (II.2). A figura V.1 ilustra este procedimento.

Apesar da simplicidade desse método, resta um problema : como se deve proceder para se obter o ponto central sobre uma "fatia" de custo constante. Vale ressaltar ainda que esta centralização pode ser muito trabalhosa.

Apesar da elegância e simplicadade do método idealizado por Barnes, só após três anos que Barnes com a colaboração de Jensen e Chopra [2] conseguiu mostrar que o algoritmo convergia e produzia bons resultados quando implementado.

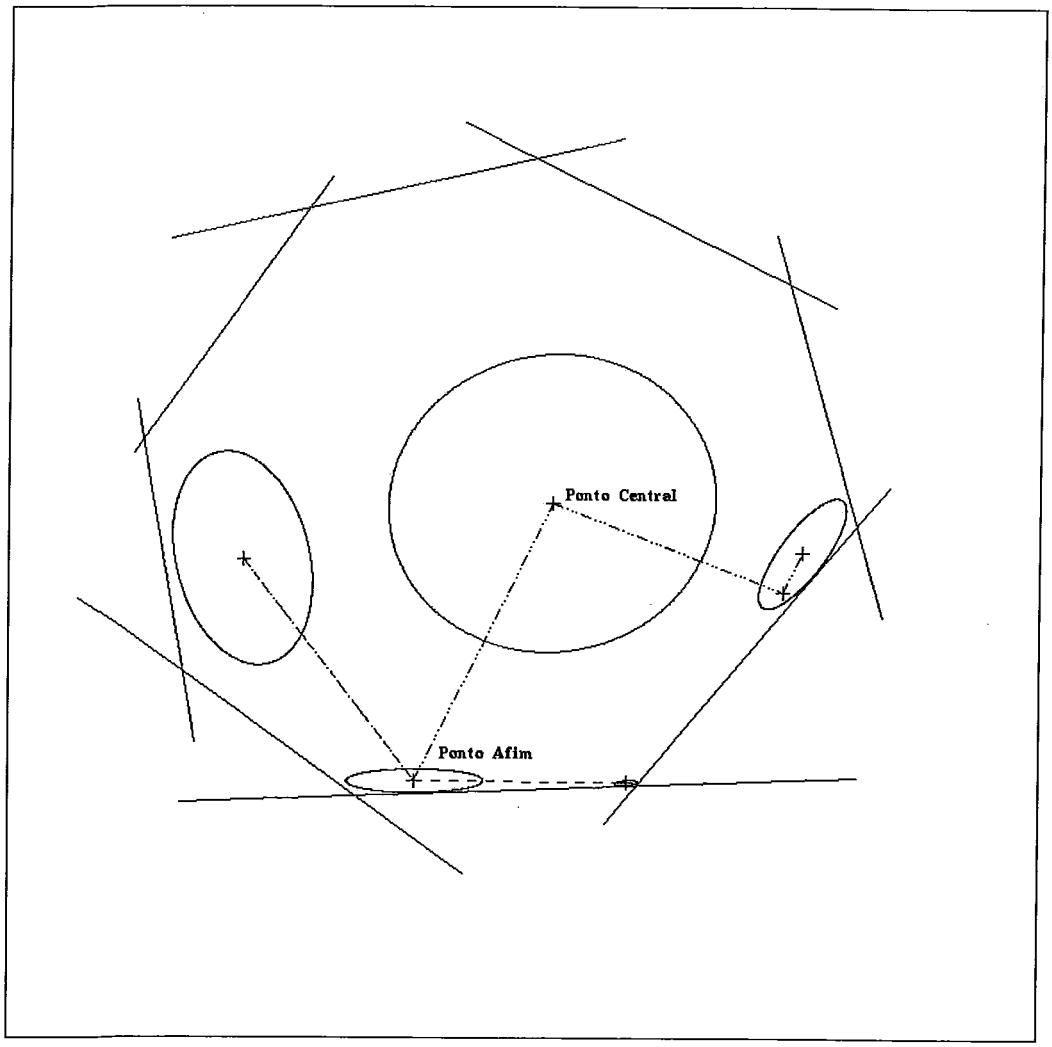


Figura V.1: Uma iteração do algoritmo de Barnes

V.1 Centro sobre uma "fatia" de custo constante

Vamos considerar o problema dual, problema (PD). Nós já sabemos do capítulo IV - seção IV.1.4 que a direção de centralização é dada por :

$$h_C = d_e$$

onde $h_C \in \Re^n$ e $d_e \in \Re^n$ está dada como em (I.2) (ver lema I.4) após a mudança de escala dada no lema I.5.

Agora, devemos encontrar uma direção $h_c \in \Re^n$ que seja a direção

de centralização mas com o custo mantido constante, ou seja, para essa direção devemos ter que :

- Seja $x \in S^0$, o custo é dado por $c^T x$. Se o custo deve permanecer constante após a centralização devemos ter que :

$$c^T(x + \lambda h_c) = c^T x$$

onde λh_c é o passo resultante da busca linear na direção h_c e $\lambda \in \Re, \lambda > 0$. Então, segue-se que :

$$c^T h_c = 0.$$

Assim, temos que a direção de centralização sobre uma "fatia" de custo constante h_c deve ser ortogonal ao custo. h_c é a direção no formato dual correspondente à direção no formato primal obtida pela projeção do vetor e_p sobre o espaço definido por $\bar{c}_p^T h_C = 0$, onde $\bar{c}_p$ e e_p correspondem respectivamente a $P_{\bar{A}X} Xc$ e $P_{\bar{A}X} e$. $\bar{A}$ está definida como no problema (P) e X é a matriz de mudança de escala como dada em (II.2). Gonzaga mostrou em [11] que essa direção de centralização sobre uma "fatia" de custo constante é dada por :

$$h_c = d_e - \frac{c^T d_e}{c^T d_c} d_e \tag{V.1}$$

onde $d_c \in \Re^n$ e $d_e \in \Re^n$ estão dadas como em (I.1) e (I.2), respectivamente (ver lema I.4), após a mudança de escala dada no lema I.5.

V.2 Busca Linear na direção de centralização sobre uma "fatia" de custo constante

Como estamos sobre uma "fatia" de custo constante, podemos ignorar a contribuição do custo na função de penalização interna — ver seção III.1.

A busca linear na direção h_c será dada pela minimização unidimensional da função p na direção do passo $\bar{\lambda} h_c$. Logo, o passo $\bar{\lambda} h_c$ resultante da busca linear será dado por :

$$\bar{\lambda} = \text{argmin}\{p(x + \lambda h_c) \mid x + \lambda h_c \in S^0, \lambda \geq 0\}$$

V.3 Algoritmo de Barnes & Jensen para o problema (PD)

Agora, podemos enunciar o algoritmo de Barnes & Jensen, que é composto de duas partes : uma iteração de centralização sobre uma "fatia" de custo constante seguido de uma do algoritmo afim escala.

Algoritmo V.1 Algoritmo Barnes & Jensen : *Dados* $x^0 \in S^0$, $e\ \epsilon \in (0, 0.5)$ e $\eta \in \Re$, $\eta \in (0,1)$.

$k = 0$;

Repita

$j = 0$;

Repita

Cálculo das Folgas : $z^j = b - Ax^j$;

Mudança de Escala :

$$Z_j = \text{diag}(z_1^j, z_2^j, \ldots, z_m^j)$$

$$A_j = Z_j^{-1} A;$$

Direção:

$$h_c = d_e - \frac{c^T d_e}{c^T d_c} d_e, \text{ onde}$$

$$d_c = (A_j^T A_j)^{-1} c \text{e } d_e = -(A_j^T A_j)^{-1} A_j^T e;$$

Busca :

$$\bar{\lambda} = \text{argmin}\{p(y^j + \lambda h_c) \mid y^j + \lambda h_c \in S^0, \lambda \geq 0\}$$

Atualização :

$$y^{j+1} = y^j + \bar{\lambda} h_c;$$

$$j = j + 1;$$

Até que $\delta(y^j, 0) < \epsilon$

Cálculo das Folgas : $z^j \in \Re^m, z^j = b - Ay^j$;

Mudança de Escala :

$$Z_j = \text{diag}(z_1^j, z_2^j, \ldots, z_m^j)$$

$$A_j = Z_j^{-1} A;$$

Direção:

$$h_A = -(A_j^T A_j)^{-1} c$$

Busca :

$$\bar{\gamma} = \min_{i=1,\ldots,m} \{\frac{z_i^j}{d_i}, d_i > 0\}, \text{ onde } d = Ah_A;$$

Novo Ponto :

$$y^{j+1} = y^j + \bar{\gamma}\eta h_A;$$

$$j = j + 1;$$

Atualização :

$$x^{k+1} = y^j;$$

$$k = k + 1;$$

Até CONVERGIR

O parâmetro η é o fator heurístico utilizado pelo algoritmo afim escala para se evitar a fronteira — ver seção II.7.

Observe que é muito mais eficiente centralizar na primeira iteração, pois se o ponto inicial estiver próximo da fronteira a centralização afastará o ponto da fronteira, melhorando o passo do Afim Escala.

V.4 Critérios de parada do algoritmo

Para fins de implementação, utilizou-se os mesmos critérios de convergência adotados para o algoritmo afim escala (ver seção II.9), acrescido do fato de que devido à imprecisões numéricas pode ser que não seja possível de se encontrar o ponto central sobre uma "fatia" de custo constante com a precisão ϵ requerida.

V.5 Figuras

Observe na figura V.2 como a elipse aumenta devido à centralização com custo constante.

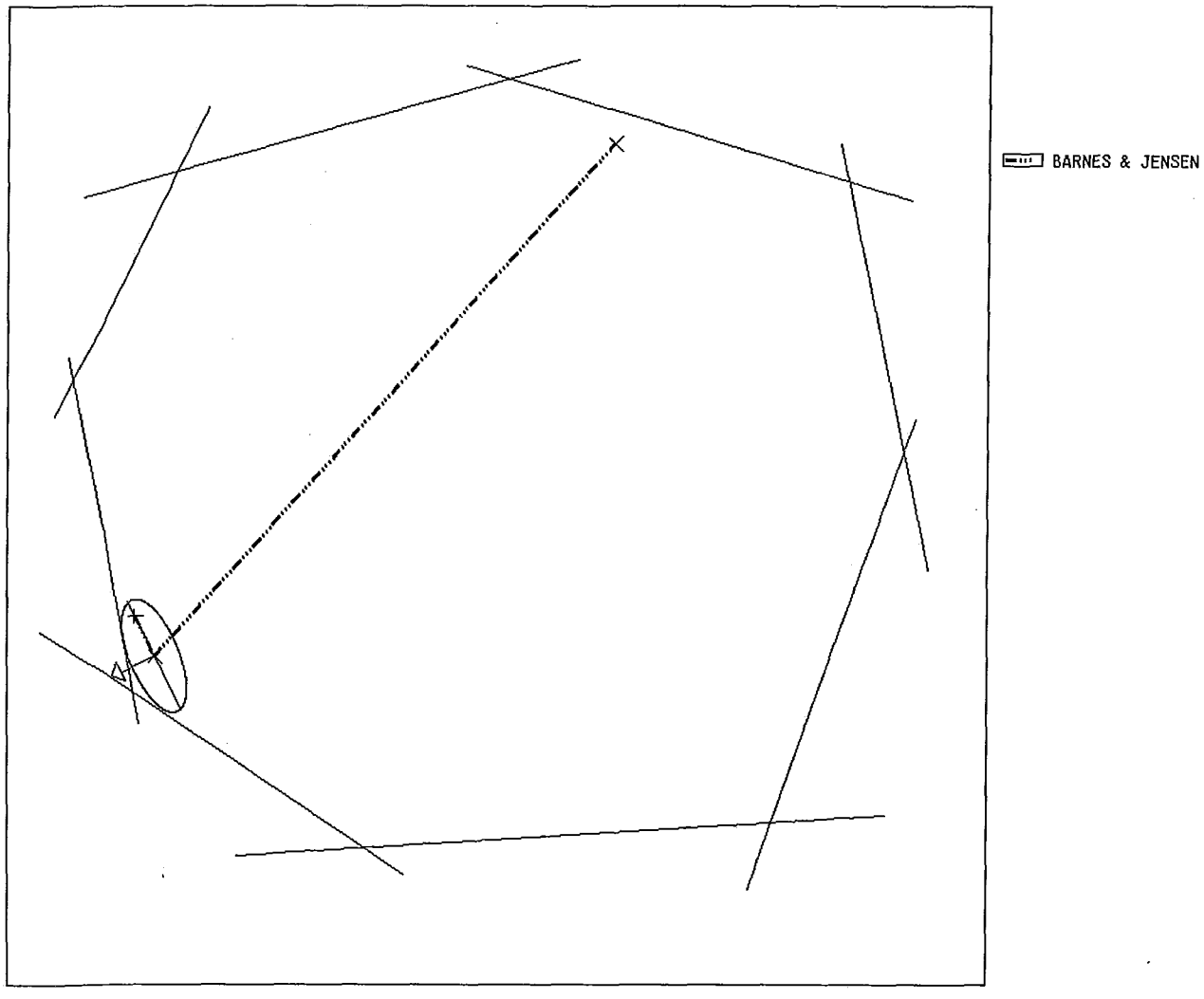


Figura V.2: Uma iteração do algoritmo de Barnes & Jensen

A figura V.3 mostra as iterações do algoritmo de Barnes & Jensen até um ótimo. Repare que a iteração Afim Escala desse algoritmo tende a aproximar os pontos da fronteira; porém com a centralização com custo constante os pontos se afastam da fronteira para o centro do conjunto, o que soluciona um dos principais "problemas" do Afim Escala que é a proximidade à fronteira.

Nas figuras V.4 e V.5 são mostradas as trajetórias dos algoritmos Afim Escala com Passo Unitário e Afim Escala com Busca Linear junto com Barnes & Jensen, respectivamente. Podemos através dessas figuras comparar os algoritmos Afim Escala e Barnes & Jensen. Note como a centralização afasta os pontos da fronteira, ao contrário do Afim Escala. Observe também que o Afim Escala com

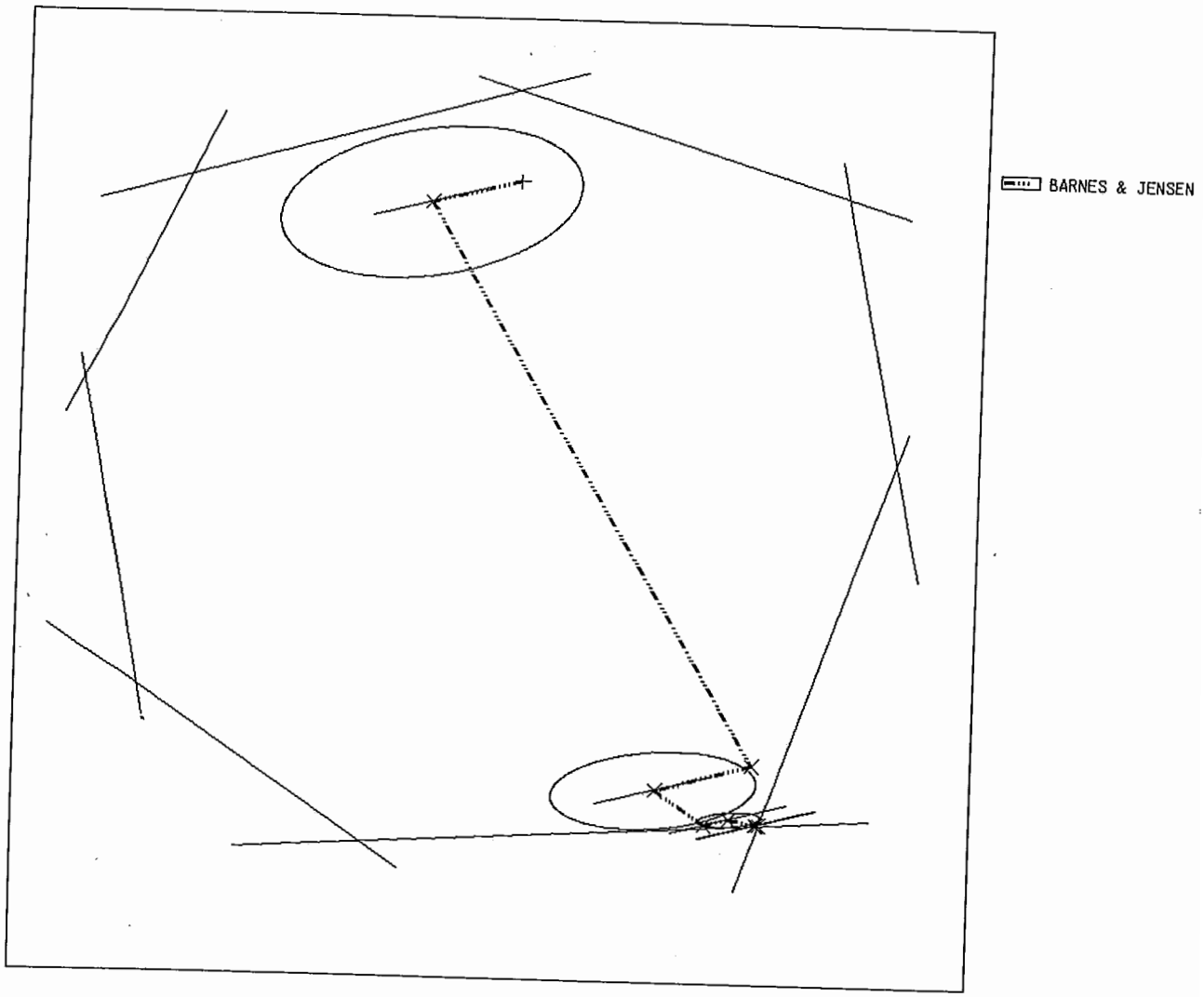


Figura V.3: Iterações do Algoritmo de Barnes & Jensen até um ótimo

Busca atingiu um ótimo em um número menor de iterações.

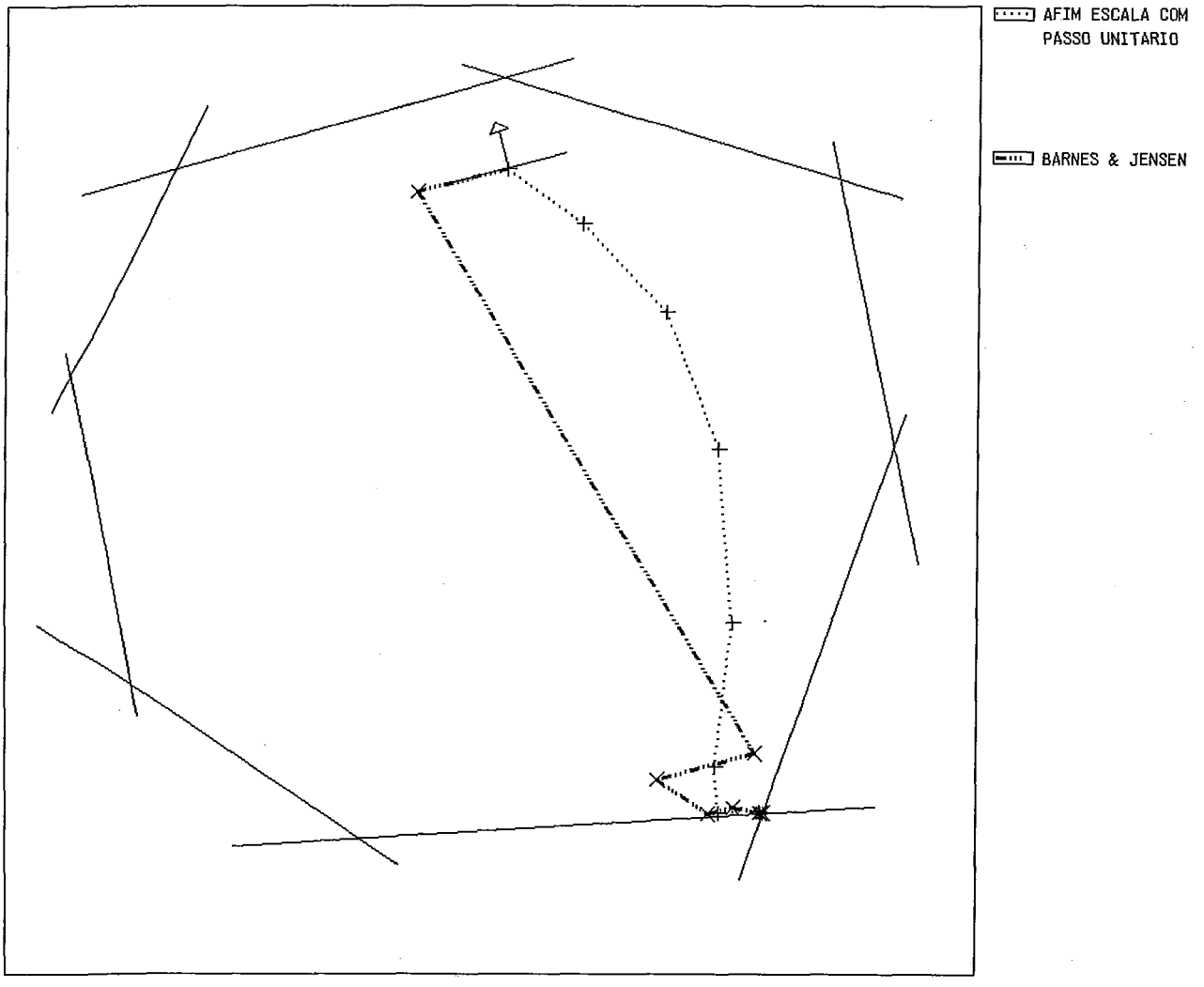


Figura V.4: Iterações dos Algoritmos Afim Escala com Passo Unitário e Barnes & Jensen

Nas figuras V.6 e V.7 são mostradas iterações do Algoritmo de Barnes & Jensen com a trajetória central.

A figura V.8 mostra iterações do algoritmo de Barnes & Jensen para uam região ja mostrada anteriormente (ver figura II.6). Partindo do ponto interior inicial dado na figura, os algoritmos Afim Escala com Passo Unitário e Busca Linear não convergem, mas repare que o algoritmo de Barnes & Jensen atinge um ótimo. Isso se deve a centralização que afasta os pontos gerados pelo algoritmo da fronteira. Devido a esse resultado, na prática para problemas grandes utiliza-se o seguinte

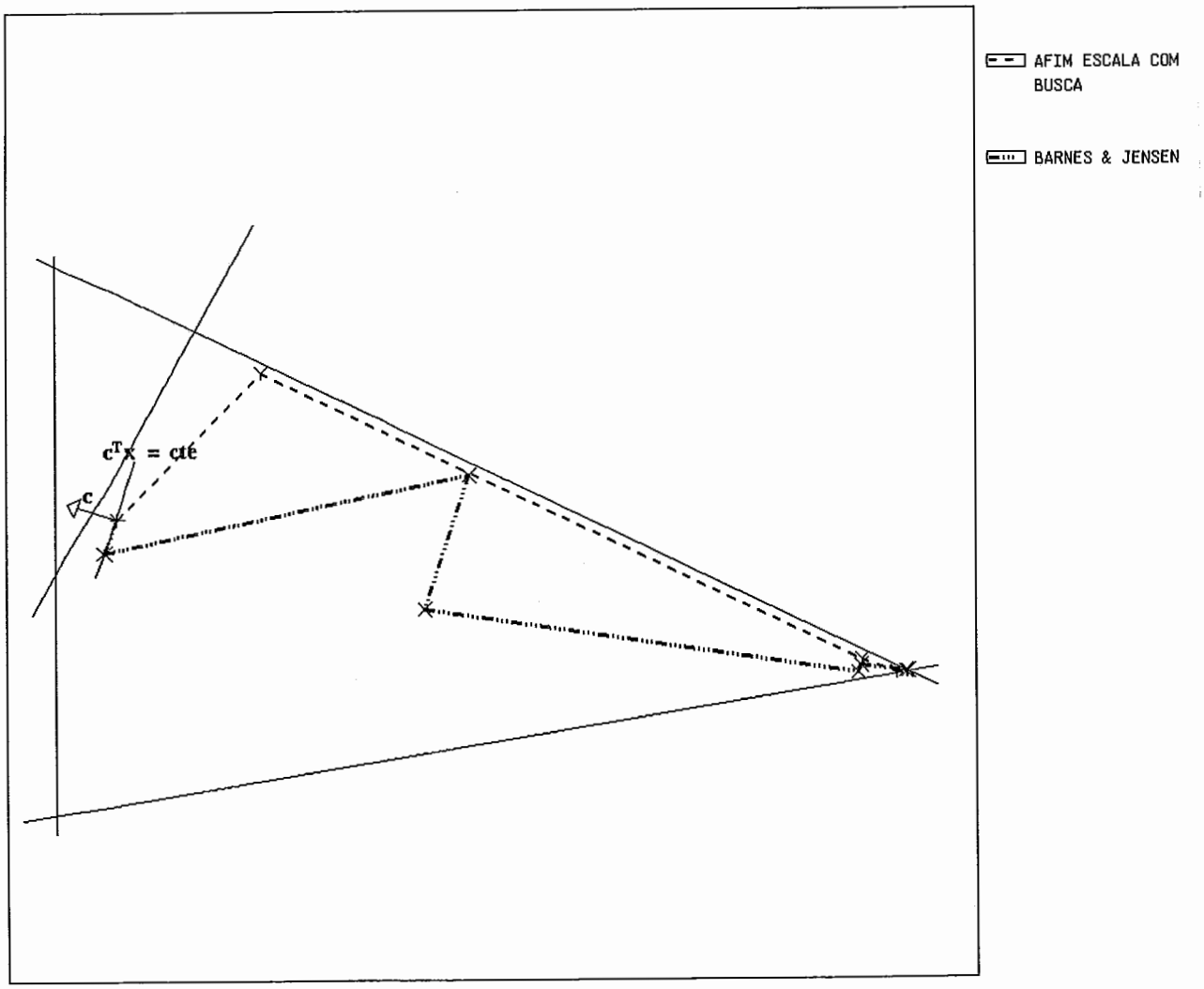


Figura V.5: Iterações dos Algoritmos Afim Escala com Busca e Barnes & Jensen

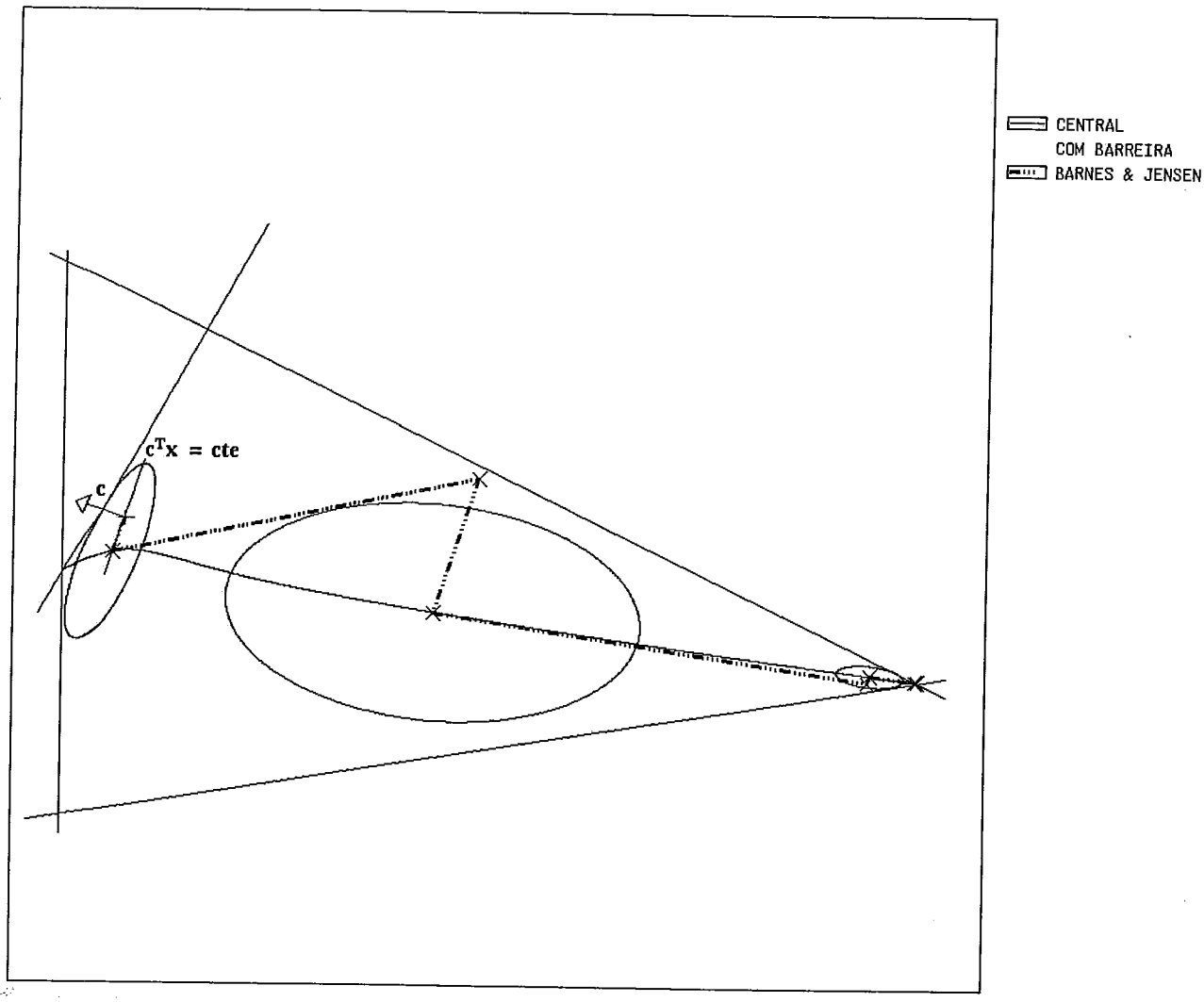


Figura V.6: Iterações do Algoritmo de Barnes & Jensen com a Trajetória Central

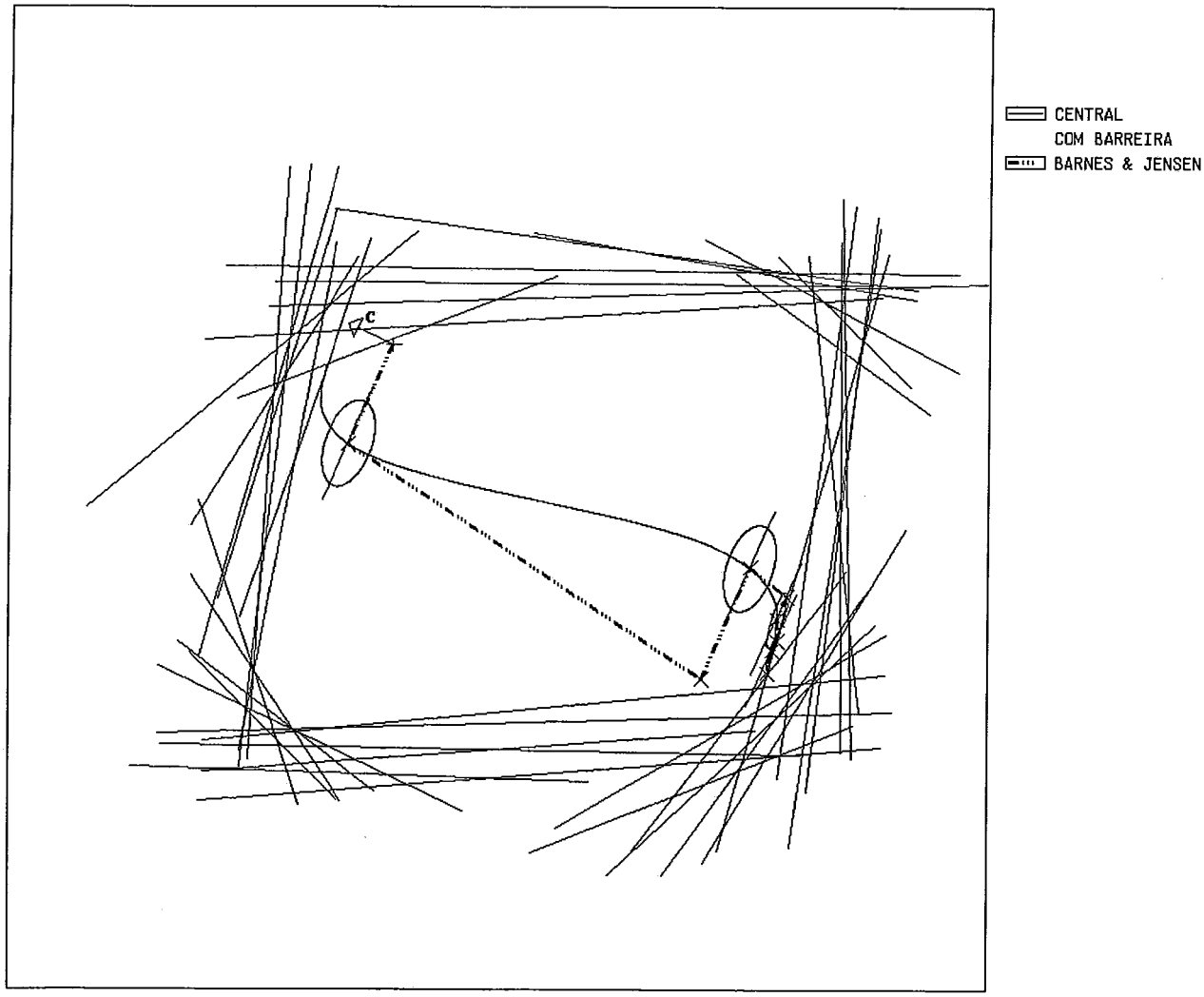


Figura V.7: Iterações do Algoritmo de Barnes & Jensen com a Trajetória Central

algoritmo :

Algoritmo V.2 Modelo de Algoritmo Afim Escala com Centralização com Custo Constante *: Dado um ponto interior viável*

Repita

Utilizar o algoritmo Afim Escala com Busca Linear;

Se o ponto gerado estiver proximo da fronteira Então
Centraliza com Custo constante;

Até CONVERGIR

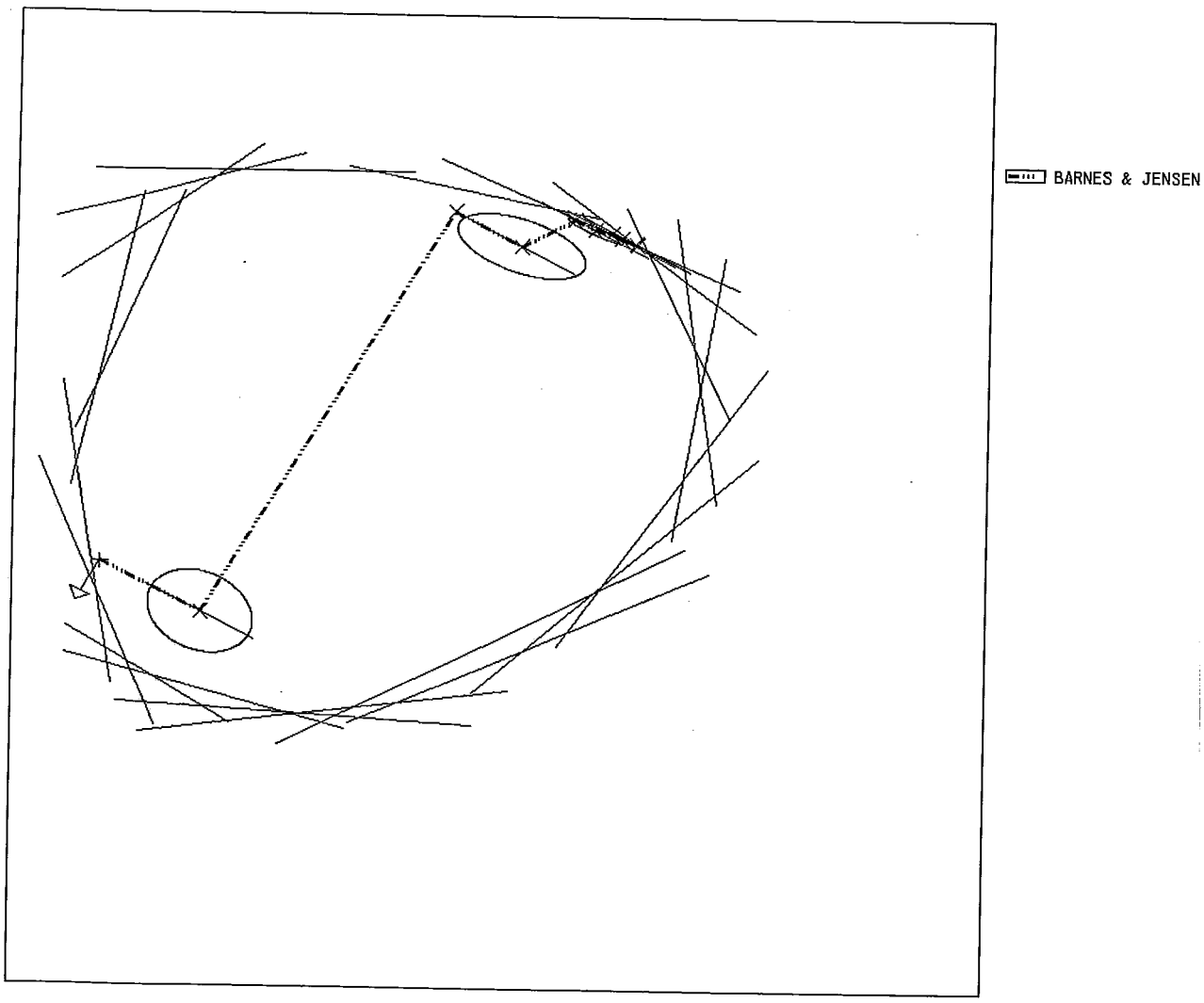


Figura V.8: Iterações do Algoritmo de Barnes & Jensen até um ótimo

Capítulo VI

Método dos Centros

Renegar [24] foi o primeiro a conseguir um algoritmo que resolvesse o problema de programação linear com uma complexidade de $O(\sqrt{n}L)$ iterações, inferior à complexidade do algoritmo de Karmarkar ($O(nL)$ iterações).

A construção feita por Renegar foi a seguinte :

- Considere o problema (PD). Agora, acrescente ao conjunto de restrições deste problema q cópias da restrição $c^T x \leq K$, onde $K \in \Re$ é uma constante e $q \in \Re, q > n$. Agora, o problema será dado por :

$$\begin{array}{lll} \text{(PR)} & \text{minimizar} & c^T x \\ & \text{sujeito a} & Ax \leq b \\ & & c^T x \leq K \end{array}$$

onde A, b, c e x estão definidos como na seção I.1. Para assegurar a existência de pontos viáveis, devemos ter $K > \hat{v}$, onde $\hat{v}$ é o custo de uma solução ótima.

A função de penalização interna associada a este problema será dada por :

$$f_K = -q \log(K - c^T x) - \sum_{i=1}^{m} \log(b - Ax)_i \qquad \text{(VI.1)}$$

Agora, cada iteração do algoritmo da trajetória é uma busca do centro analítico do politopo definido pelo problema (PR), que é um problema já estudado e de geometria simples. A simplicidade deste problema de centralização talvez justifique o fato deste ter sido o primeiro método de trajetória central para programação linear, programação quadrática com restrições e programação convexa.

O método de Renegar é também conhecido como *método de centros*, pois Renegar utilizou a caracterização de pontos centrais usada por Huard [16] no seu método de centros para programação não-linear.

VI.1 Direção de Centralização para o Método de Centros

Podemos caracterizar o ponto central também através de uma nova definição que é equivalente à dada no capítulo IV.

Usando a função f_K definida no item anterior, defina o *ponto central* $x(K) \in \Re^n$ como sendo o ponto que satisfaça :

$$K > \hat{v} \longmapsto x(K) = \text{argmin}\{f_K(x) \mid x \in S^0, c^T x < K\} \qquad \text{(VI.2)}$$

onde $\hat{v}$ é o custo de uma solução ótima para o problema (PD).

Essa é uma boa oportunidade para se verificar porque que o centro é analítico e não geométrico. A inclusão de q cópias da restrição $c^T x \leq K$ aumenta o peso dessa restrição na função f_K, e produz o efeito de afastar o centro da face correspondente a essa restrição; isso pode ser observado nas figuras VI.1 e VI.2.

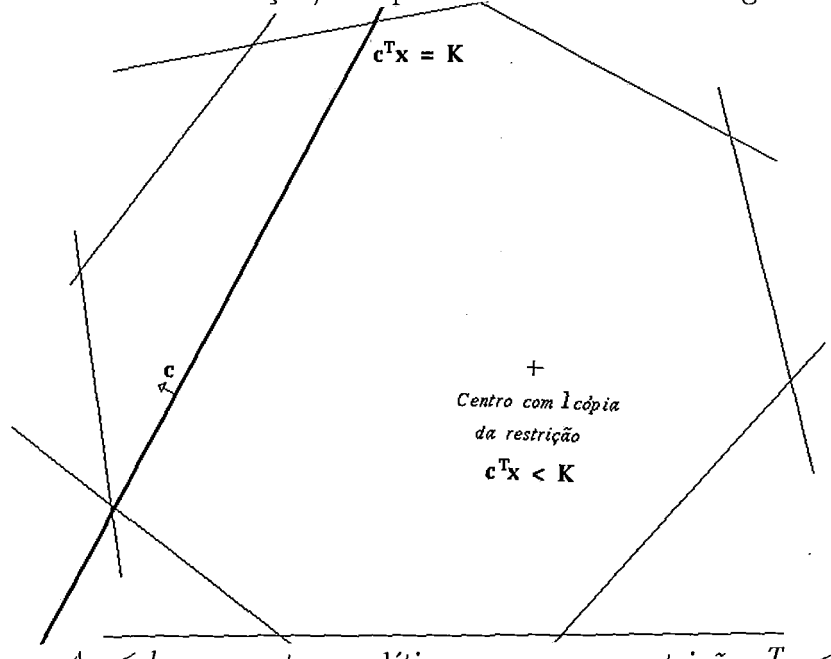


Figura VI.1: Politopo $Ax \leq b$ com centro analítico e com uma restrição $c^T x \leq K$

Renegar descobriu que para $q \geq n$ o centro está "bastante "perto" da solução ótima, ou seja, $c^T x(K) \leq \hat{v} + K/2$.

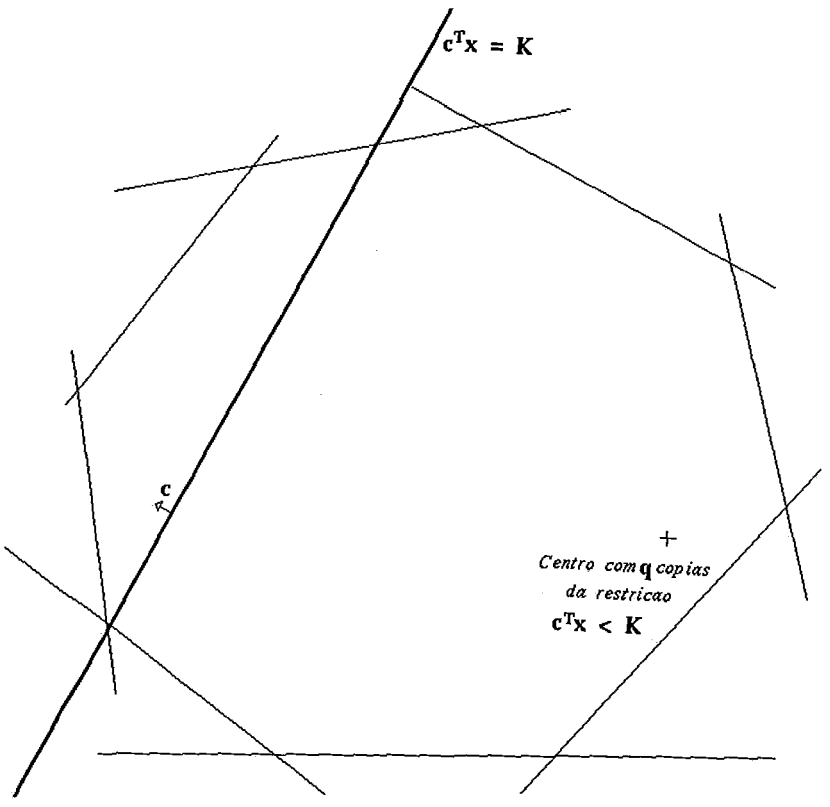


Figura VI.2: Politopo $Ax \leq b$ com centro analítico e com q cópias da restrição $c^T x \leq K$

A nossa caracterização de ponto central está bem definida, pois f_K é estritamente convexa e cresce indefinidamente perto da fronteira do conjunto viável.

Agora, considerando o problema (PR), procedemos da mesma maneira para calcular o centro do conjunto viável que no capítulo IV, só que agora a função de penalização interna será dada por f_K e incorporamos 1 cópia da restrição $c^T x \leq K$ a matriz A do problema original (observe que o peso das q cópias da restrição $c^T x \leq K$ está incorporado ao problema na função f_K).

Assim, reescrevendo o problema (PR), temos que :

$$\begin{array}{ll} \text{minimizar} & c^T x \\ \text{sujeito a} & \bar{A}x \leq \bar{b} \end{array}$$

onde $\bar{A} \in \Re^{(m+1)\times n}, \bar{A} = \begin{bmatrix} A & 0 \\ 0 & c^T \end{bmatrix}$ e $\bar{b} \in \Re^{m+1}, \bar{b} = \begin{bmatrix} b \\ K \end{bmatrix}$.

Aplicando o resultado da seção IV.1.4 temos que a direção de centralização h_R para o método de Renegar será dada por :

$$h_R = \bar{d}_e = -(\bar{A}_k^T \bar{A}_k)^{-1} \bar{A}_k^T e \qquad \text{(VI.3)}$$

onde $h_R \in \Re^n, e \in \Re^m, e = [1 \ldots 1]^T$ e A_k é a matriz A após a mudança de escala dada no Lema I.5.

VI.2 Atualização do parâmetro K

Resta-nos saber como faremos para atualizar o parâmetro K a fim de garantir a convergência do algoritmo.

Renegar demonstrou em [24] que K pode ser especificado como :

$$K_{k+1} = \beta K_k + (1 - \beta) c^T x^k$$

onde o subíndice k denota a k-ésima iteração do algoritmo e $\beta \in (0, 1)$.

O uso de β muito próximo de 1 produz passos curtos, baixa complexidade e pouca eficiência na prática. Valores de β pequenos produzem passos longos e algoritmos eficientes, mas com complexidade de $O(nL)$ iterações — ver Gonzaga [13].

VI.3 Algoritmo de Renegar

Agora, podemos enunciar o algoritmo de Renegar.

Algoritmo VI.1 Algoritmo Renegar : *Dados* $x^0 \in S^0, \epsilon \in (0, 0.5), \beta \in (0, 1), \gamma \in (0, 0.5)$.

$k = 0$;

$K_0 = c^T x^0 + \gamma$;

Repita

$y^0 = x^k$;

$j = 0$

Repita

Cálculo das Folgas : $\bar{z}^j = \bar{b} - \bar{A} y^j$;

Mudança de Escala :

$$\bar{A}_j = \bar{Z}_j^{-1} \bar{A}, \bar{Z}_j = \text{diag}(\bar{z}_1^j, \ldots, \bar{z}_{m+1}^j);$$

Direção:

$$h_R = -(\bar{A}^T\bar{A})^{-1}\bar{A}^T e \text{ (ver seção VI.1);}$$

Busca :

$$\bar{\lambda} = \text{argmin}\{f_{K_k}(y^j + \lambda h_R) \mid y^j + \lambda h_R \in S^0, \lambda \geq 0\};$$

Novo ponto :

$$y^{j+1} = y^j + \bar{\lambda} h_R;$$

$$j = j + 1;$$

Até que $(\delta(y^j, 0) < \epsilon)$

Atualização :

$$x^{k+1} = y^j;$$

$$K_{k+1} = \beta K_k + (1 - \beta) c^T x^k$$

$$k = k + 1;$$

Até CONVERGIR

VI.4 Critérios de parada do algoritmo

Os critérios de parada utilizados para o algoritmo de Renegar, foram:

1. o número de iterações do algoritmo;
2. parar quando se encontrar x tal que $c^T x < 2^{-L}$, na prática em vez de 2^{-L} toma-se um número μ tal que μ seja da ordem de 10^{-8}.

VI.5 Figuras

As figuras VI.3, VI.4 e VI.5 mostram a primeira, segunda e terceira iteração do algoritmo de Renegar, respectivamente, juntamente com a trajetória central e curvas de nível. O mesmo é feito para as figuras VI.6, VI.7 e VI.8 para uma região diferente.

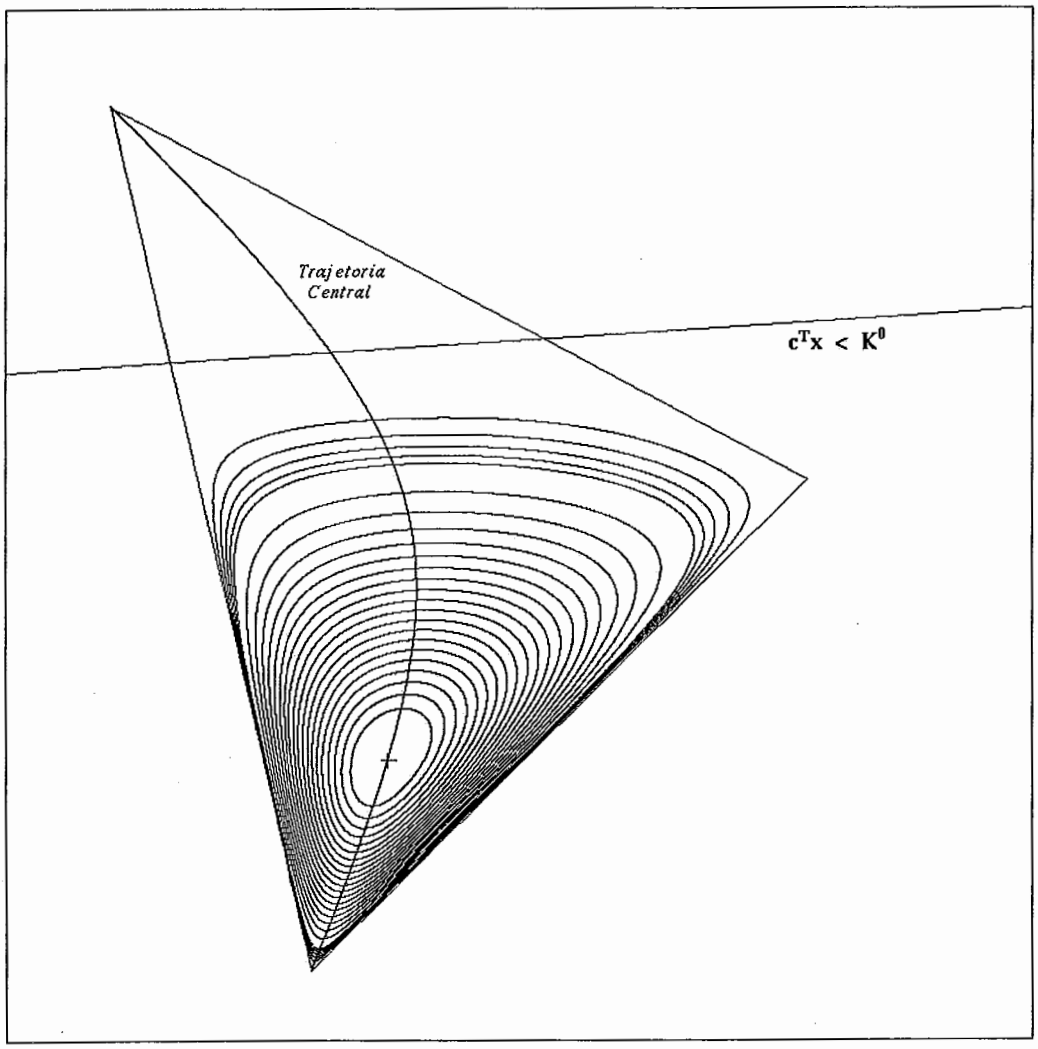


Figura VI.3: Primeira iteração do Algoritmo de Renegar

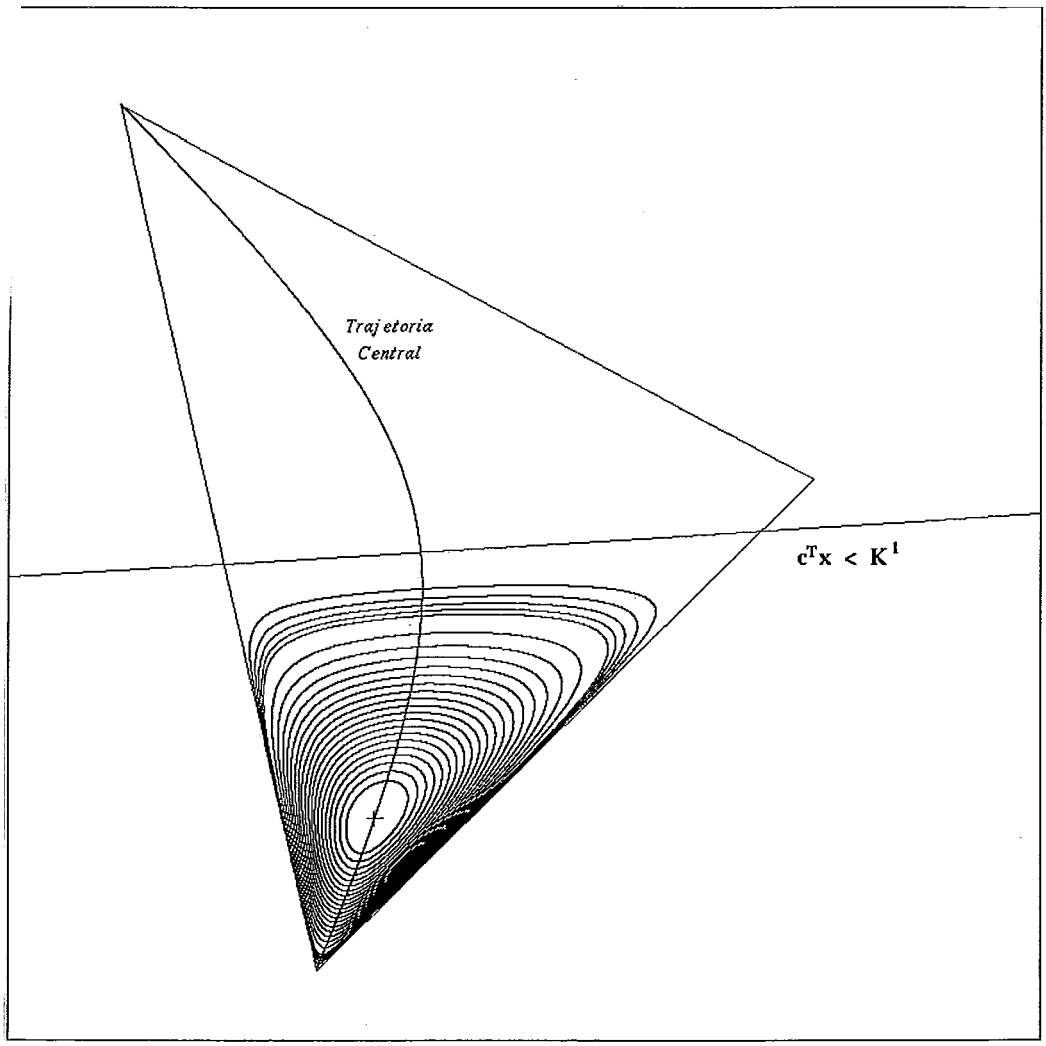


Figura VI.4: Segunda Iteração do Algoritmo de Renegar

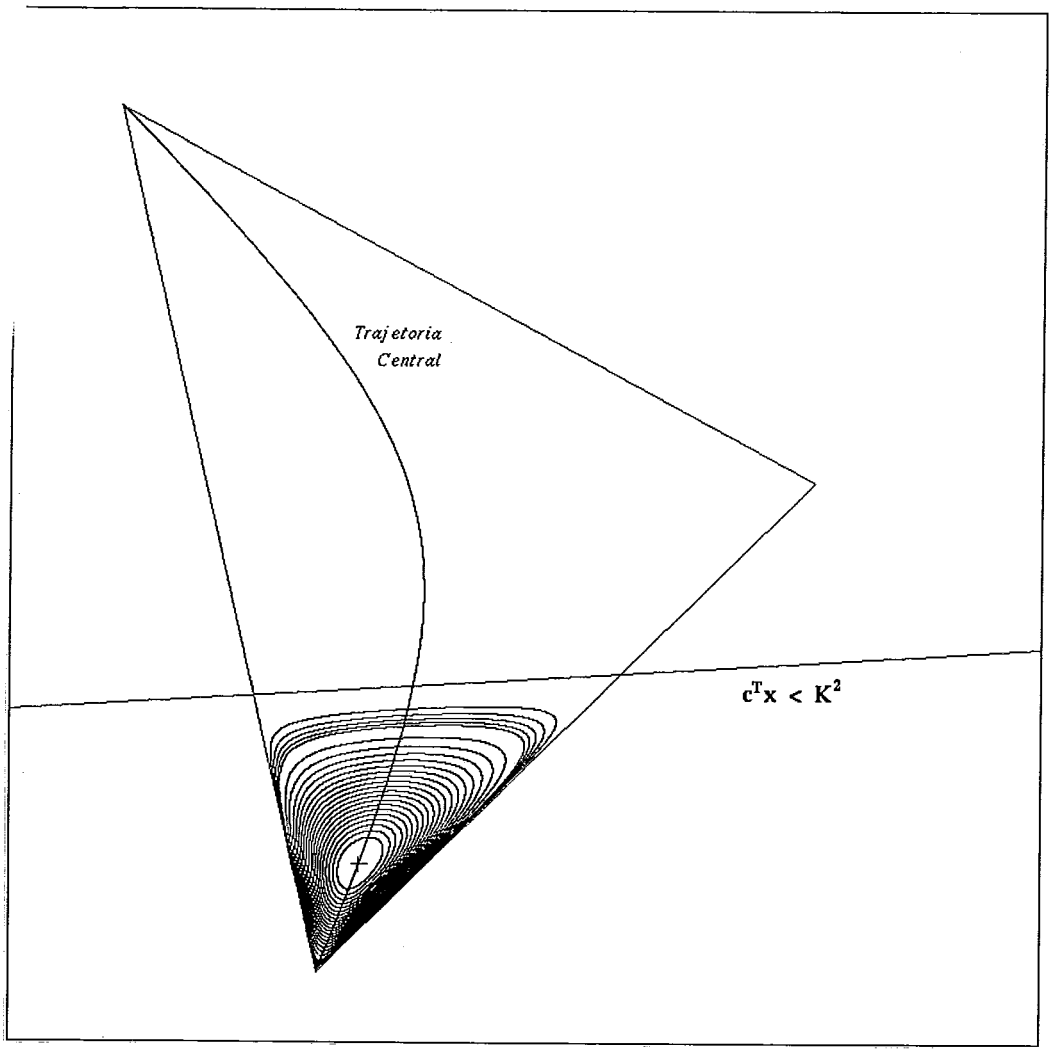


Figura VI.5: Terceira Iteração do Algoritmo de Renegar

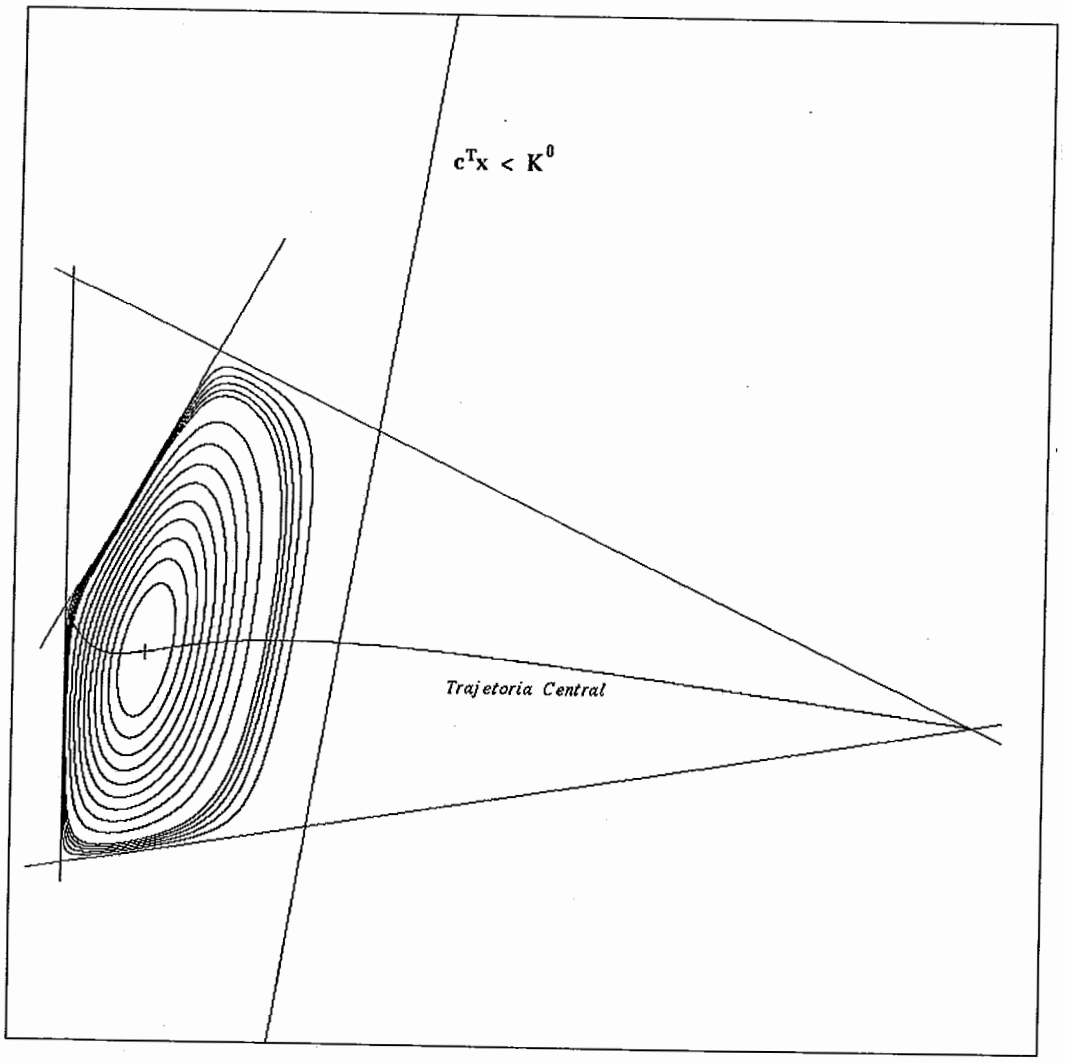


Figura VI.6: Primeira iteração do Algoritmo de Renegar

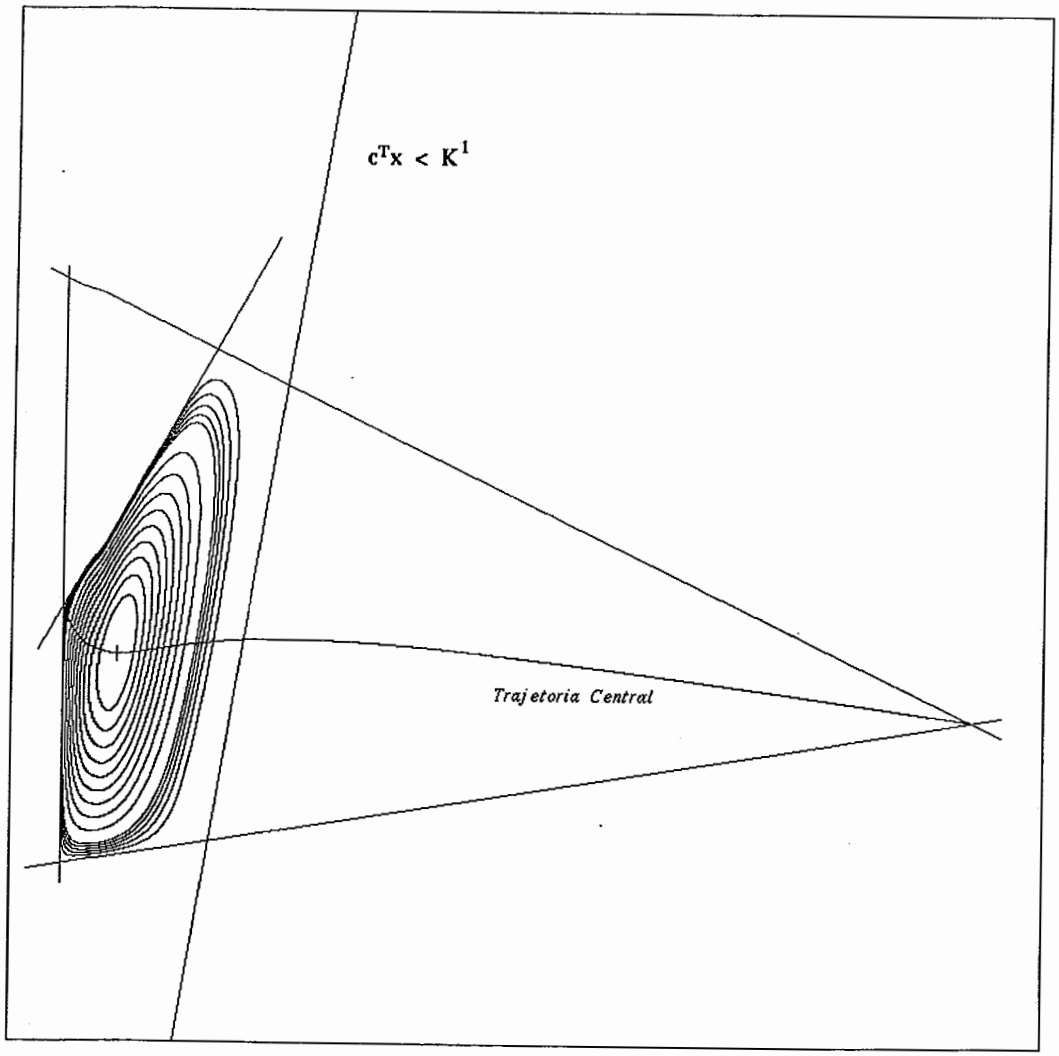


Figura VI.7: Segunda Iteração do Algoritmo de Renegar

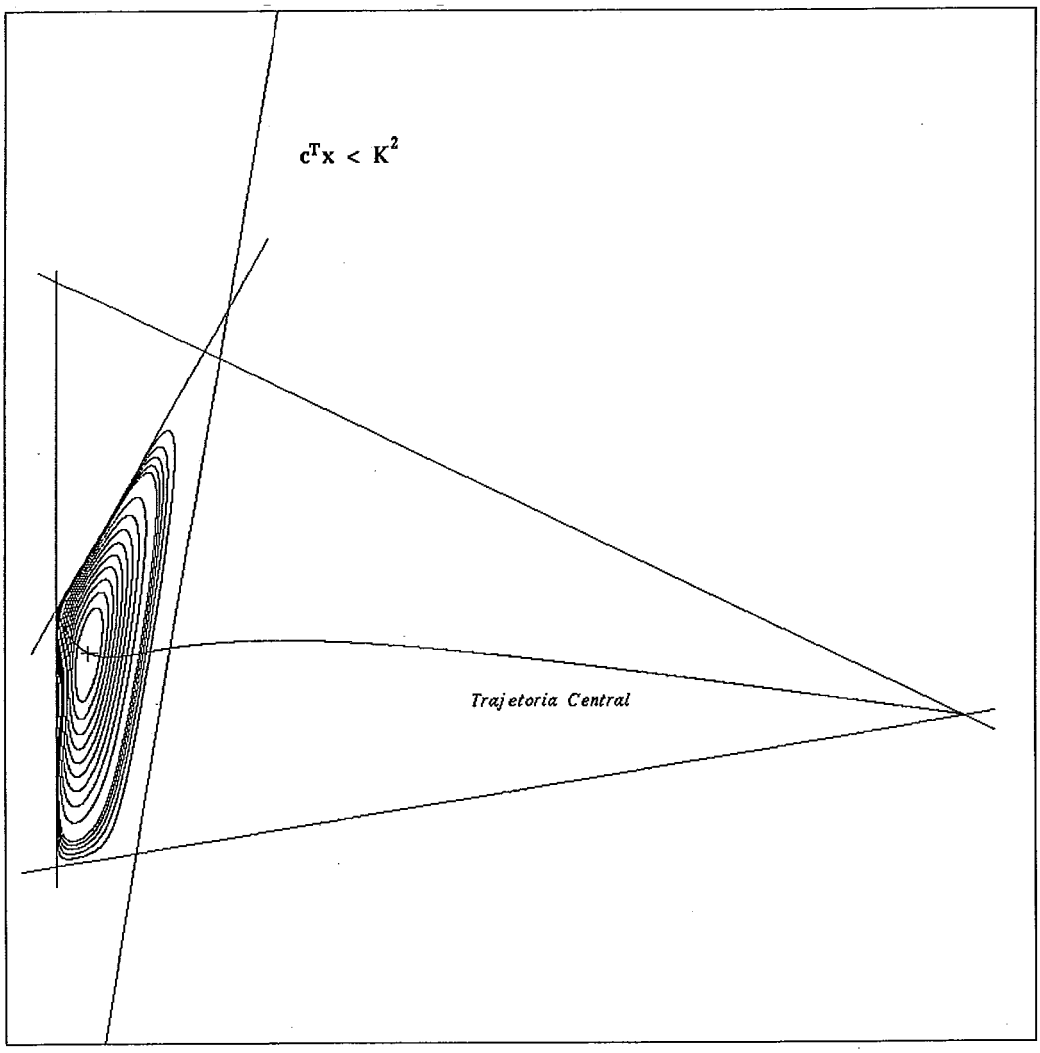


Figura VI.8: Terceira Iteração do Algoritmo de Renegar

Capítulo VII

Trajetória Central

No capítulo IV já havíamos mencionado a utilidade e as propriedades de pontos centrais, e nos capítulos seguintes estudamos dois métodos que "seguiam" a trajetória central, porém sem utilizar a função de penalização interna. Megiddo [22] foi o primeiro a estudar a trajetória central e utilizou a função penalizada, portanto, vamos definir formalmente a trajetória central a partir de (IV.6) :

Definição VII.1 *Trajetória Central*

Considere o problema (P) e a função penalizada $\bar{f}_\alpha$. *A* trajetória central *é dada pela curva* $\alpha \in \Re_+ \longmapsto x(\alpha) \in \bar{S}^0$.

O nosso objetivo é obter um algoritmo que "siga" essa trajetória. Seja $\rho \in (w^-, w^+) \longmapsto x(\rho)$ uma parametrização da trajetória central, com $w^+ > w^-$, w^+ possivelmente infinito. Todos os algoritmos de trajetória seguirão o seguinte modelo :

Algoritmo VII.1 Modelo de Algoritmo de Trajetória Central : *Dados* $x^0 \in S^0$, *e* $\rho_0 \in (w^-, w^+)$, *com* $x^0 = x(\rho_0)$.

$k = 0$;

Repita

Escolha $\rho_{k+1} > \rho_k$;

Chame um algoritmo interno de minimização para encontrar

$$x^{k+1} = x(\rho_{k+1});$$

Até CONVERGIR

No caso da função penalizada, utilizando a parametrização definida na equação IV.9, escolhemos α_{k+1} de modo que :

$$\alpha_{k+1} = (1+\mu)\alpha_k \tag{VII.1}$$

onde μ é uma constante positiva.

Nós já temos um algoritmo para encontrar o ponto "aproximadamente central", que é o algoritmo dado no capítulo IV. Portanto, o nosso algoritmo implementável será dado por :

Algoritmo VII.2 Algoritmo de Trajetória Central : *Dados* $x^0 \in \bar{S}^0$, $\epsilon \in (0, 0.5)$, $\alpha_0 \in (0, 0.5)$ $\bar{\delta}(x_0, \alpha_0) < \epsilon$, $\mu \in \Re_+, L > 0$.

$k = 0$;

Repita

Escolha $\alpha_{k+1} = (1+\mu)\alpha_k$;

Chame um algoritmo interno de centralização para encontrar

$$\{x^{k+1} \in \bar{S}_0 \mid \bar{\delta}(x^{k+1}, \alpha_{k+1}) < \epsilon\}$$

k = k + 1;

Até que $(\alpha_k > 2^L)$

VII.1 Direção de Centralização para o problema (PD)

Recordando do capítulo III, a função de penalização interna para o problema (PD) é dada por :

$$f_\alpha(x) = \alpha c^T x + p(x).$$

Do capítulo VII sabemos que para resolver o problema de centralização, basta aplicarmos o algoritmo de Newton-Raphson à função de penalização, ou seja, a direção de centralização h_T para a *trajetória central* é dada pela direção

de Newton-Raphson para um problema no *formato dual*, após a mudança de escala dada no Lema I.5 — ver (IV.10). Logo :

$$h_T = h_N(x, \alpha) = -\alpha d_c + d_e. \tag{VII.2}$$

onde d_c e d_e estão definidas como em (I.1) e (I.2), respectivamente, após a mudança de escala como dada no Lema I.5.

Observe que a única diferença em relação à *direção de centralização com custo constante* h_C é a presença da contribuição do custo, já que agora $\alpha \neq 0$.

VII.2 Escolha da atualização do parâmetro α

De acordo com a escolha de μ em (VII.1), obtemos diferentes comportamentos para o parâmetro α. Tomando-se $\mu = 0.1/\sqrt{n}$ obtem-se um algoritmo de *trajetória central de passos curtos.* Esse foi o primeiro algoritmo de *trajetória central* para a função penalizada (a descrição de convergência desse algoritmo com uma complexidade de $O(n^3L)$ operações aritméticas é encontrada em Gonzaga [9]). Mais ainda poderíamos aumentar ainda mais o parâmetro α e assim mesmo garantir a convergência do método. Gonzaga mostrou em [12] que tomando-se $\mu > 0.1/\sqrt{n}$, obtem-se um algoritmo de trajetória central para a função penalizada que converge em $O(\mu nL)$ iterações do algoritmo de penalização interna (que é o afim escala aplicado à $\bar{f}_\alpha$). Denominou-se a esse algoritmo de *trajetória central de passos longos.*

VII.3 Algoritmo de Trajetória Central para o problema (PD)

Algoritmo VII.3 Algoritmo de Trajetória Central : *Dados* $x^0 \in S^0$, $\epsilon \in (0, 0.5)$, $\alpha_0 \in (0, 0.5)$, *tal que* $\delta(x_0, \alpha_0) < \epsilon$, $\mu \in \Re_+, L > 0$.

$k = 0$;

Repita

$j = 0$;

$y^0 = x^k$;

Penalidade : $\bar{\alpha} = \gamma\alpha_k$, em geral, utilizamos γ tal que $\gamma = (1 + \mu)$;

Repita

Cálculo das Folgas : $z^j = b - Ay^j$;

Mudança de Escala :

$$Z_k = \text{diag}(z_1^k, z_2^k, \ldots, z_m^k)$$

$$A_k = Z_k^{-1} A;$$

Direção : $h_T = -\alpha d_c + d_e$, onde :

$$d_c = -(A_k^T A_k)^{-1} c$$

e

$$d_e = -(A_k^T A_k)^{-1} A_k^T e;$$

Proximidade : Calcule $\bar{\delta} = \delta(y^j, \bar{\alpha})$ — ver definição IV.4

Busca : resolver aproximadamente

$$\bar{\lambda} = \text{argmin}\{f_\alpha(y^j + \lambda h_T) \mid y^j + \lambda h_T \in S^0, \lambda \geq 0\}, \bar{\lambda} \in \Re;$$

Novo ponto :

$$y^{j+1} = y^j + \bar{\lambda} h_T;$$

$$j = j + 1;$$

Até que $\bar{\delta} < \epsilon$

Atualização :

$$x^{k+1} = y^j;$$

$$\alpha_{k+1} = \bar{\alpha};$$

$$k = k + 1;$$

Até que $(\alpha_k > 2^L)$

Os *Critérios de Parada* são os mesmos utilizados para o método dos centros (ver capí tulo VI - Seção VI.4).

VII.4 Observação

Para a visualização gráfica da trajetória central, deve-se utilizar o algoritmo VII.3 com *passos curtos*. Utilizando-se o algoritmo VII.3 com *passos longos*, é possível de

se visualizar o comportamento do algoritmo a cada iteração, mas os pontos gerados pelo algoritmo não são suficientes para visualizar a trajetória central (consulte o capítulo VIII para maiores detalhes).

VII.5 Figuras

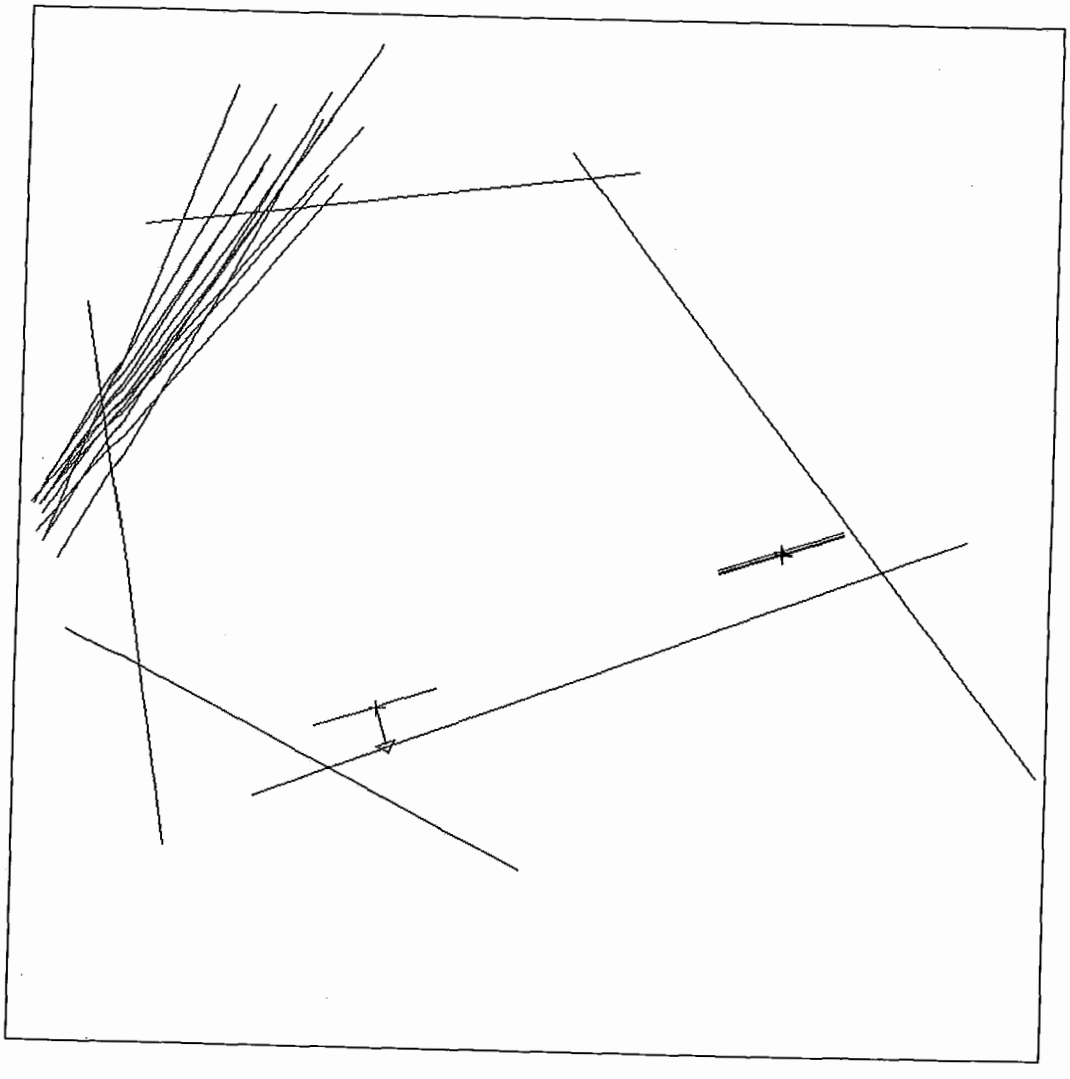


Figura VII.1: Primeira Iteração do algoritmo de Centralização

A figura VII.1 mostra uma iteração do algoritmo de Centralização a partir de um ponto interior inicial. Observe que o algoritmo de Centralização não começa de um ponto inicial, mas do centro analítico do conjunto viável a uma distancia δ — ver definição IV.4.

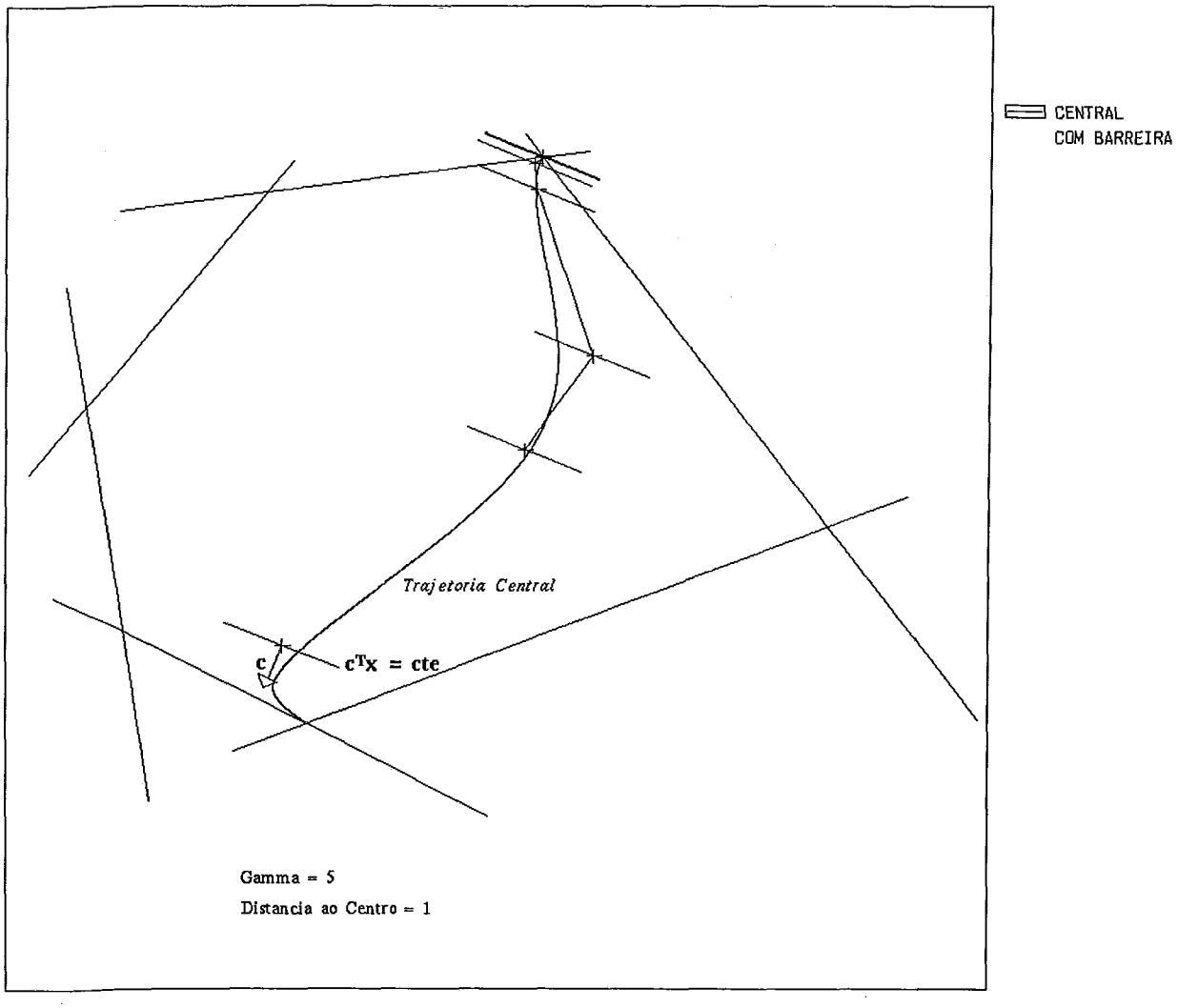


Figura VII.2: Iterações do Algoritmo de Centralização para $\gamma = 5$

As figuras VII.2 e VII.3 mostram iterações do algoritmo de Centralização para diferentes valores de γ. Observe que na figura VII.3 que cada iteração do algoritmo vale por duas iterações da figura VII.2, o que já era esperado pois o valor de atualização do parametro α, γ, foi elevado ao quadrado.

As figuras VII.4 e VII.5 mostram o mesmo que para as figuras VII.2 e VII.3, respectivamente, só que ampliadas com um zoom perto de um ótimo.

As figuras VII.6, VII.7 e VII.8 mostram o que acontece com o algoritmo de Centralização se variarmos a distancia ao centro de um valor com boa precisão ($\delta = 0.01$) até um valor de pouca precisão ($\delta = 5$), ou seja, tomando-se pontos próximos a trajetória central até pontos afastados da trajetória central.

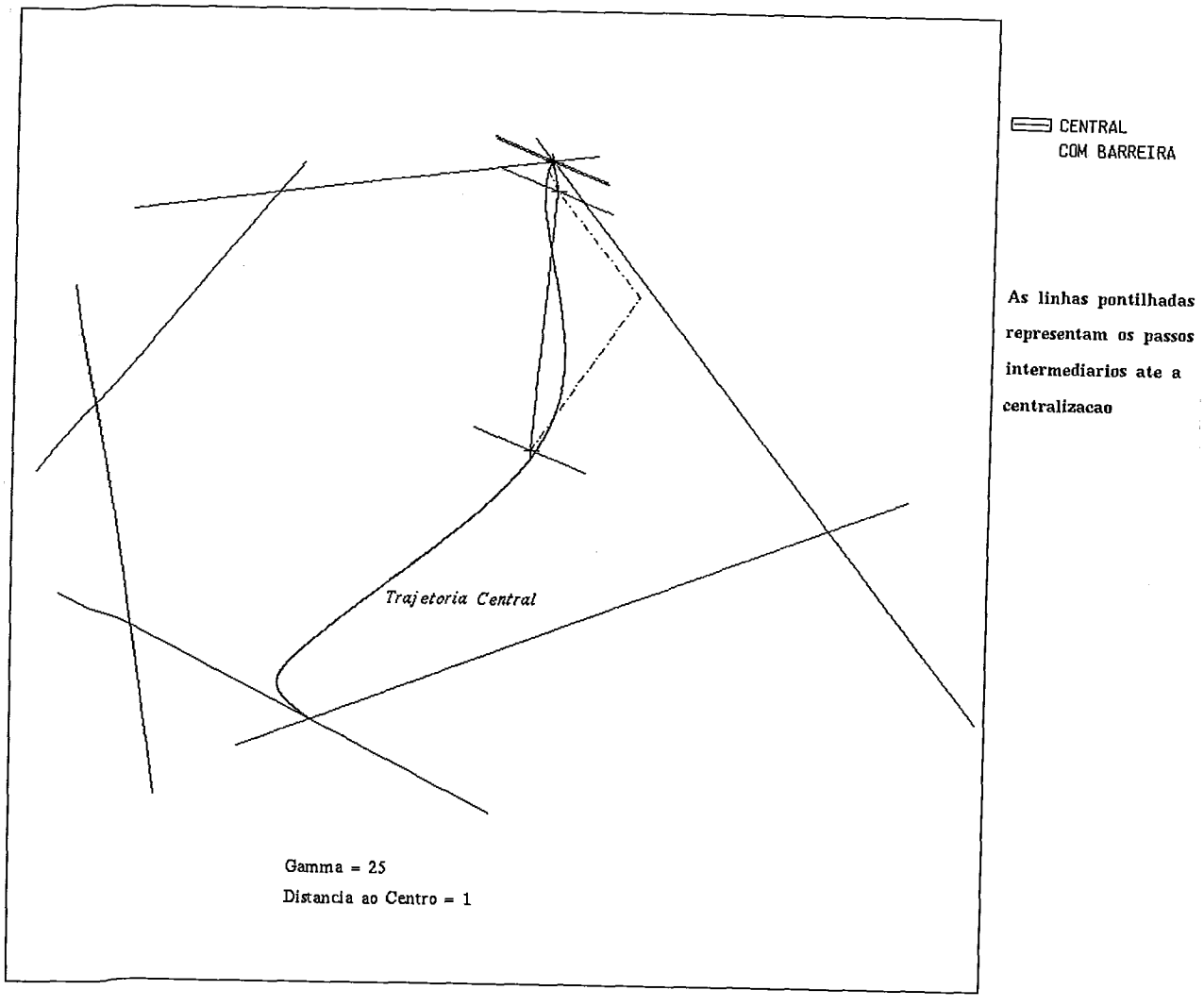


Figura VII.3: Iterações do Algoritmo de Centralização para $\gamma = 25$

As figuras VII.9, VII.10 e VII.11 mostram o mesmo que as figuras VII.6, VII.7 e VII.8, respectivamente, só que agora ampliadas de um zoom e perto do ótimo.

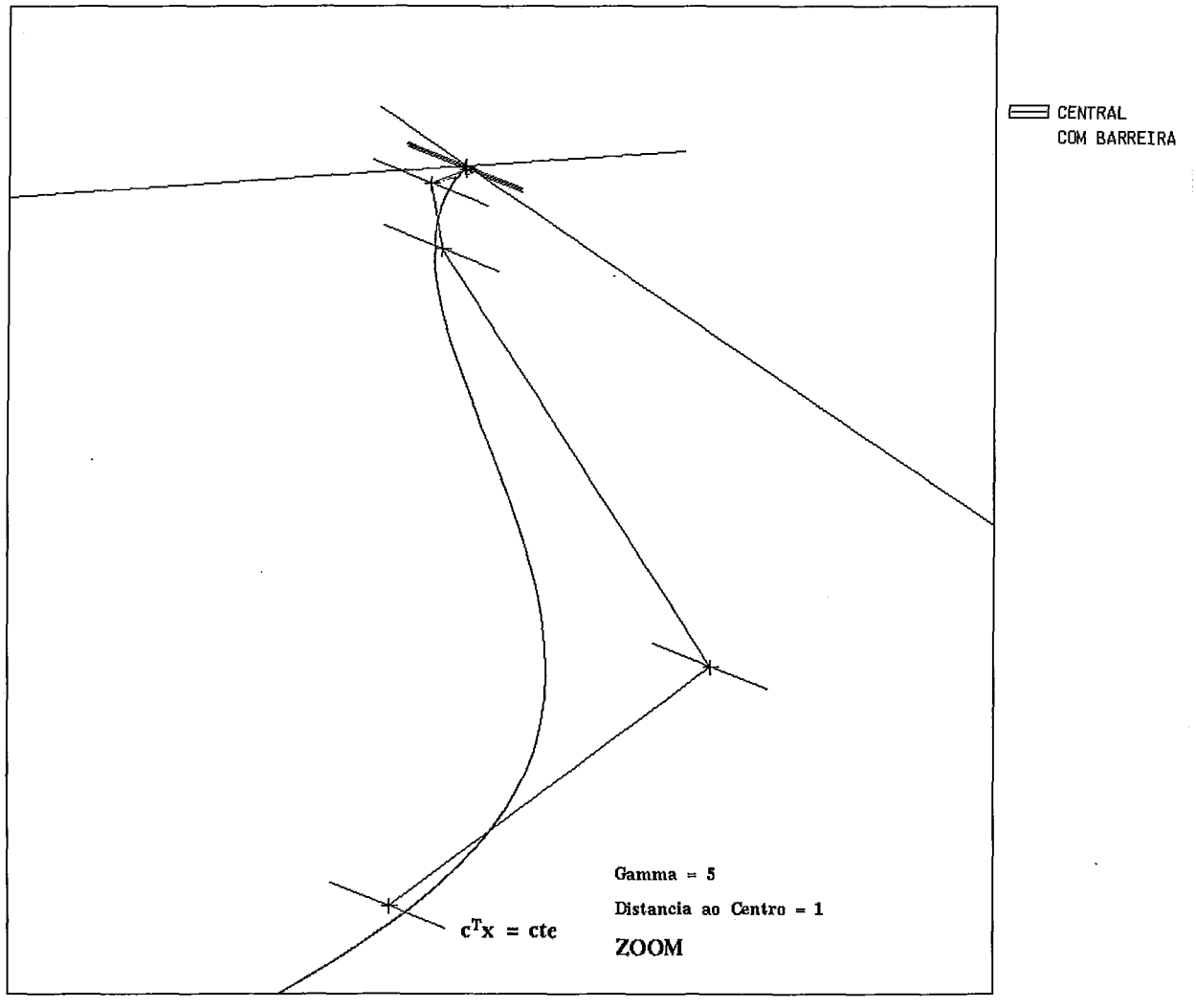


Figura VII.4: Iterações do Algoritmo de Centralização para $\gamma = 5$ com Zoom

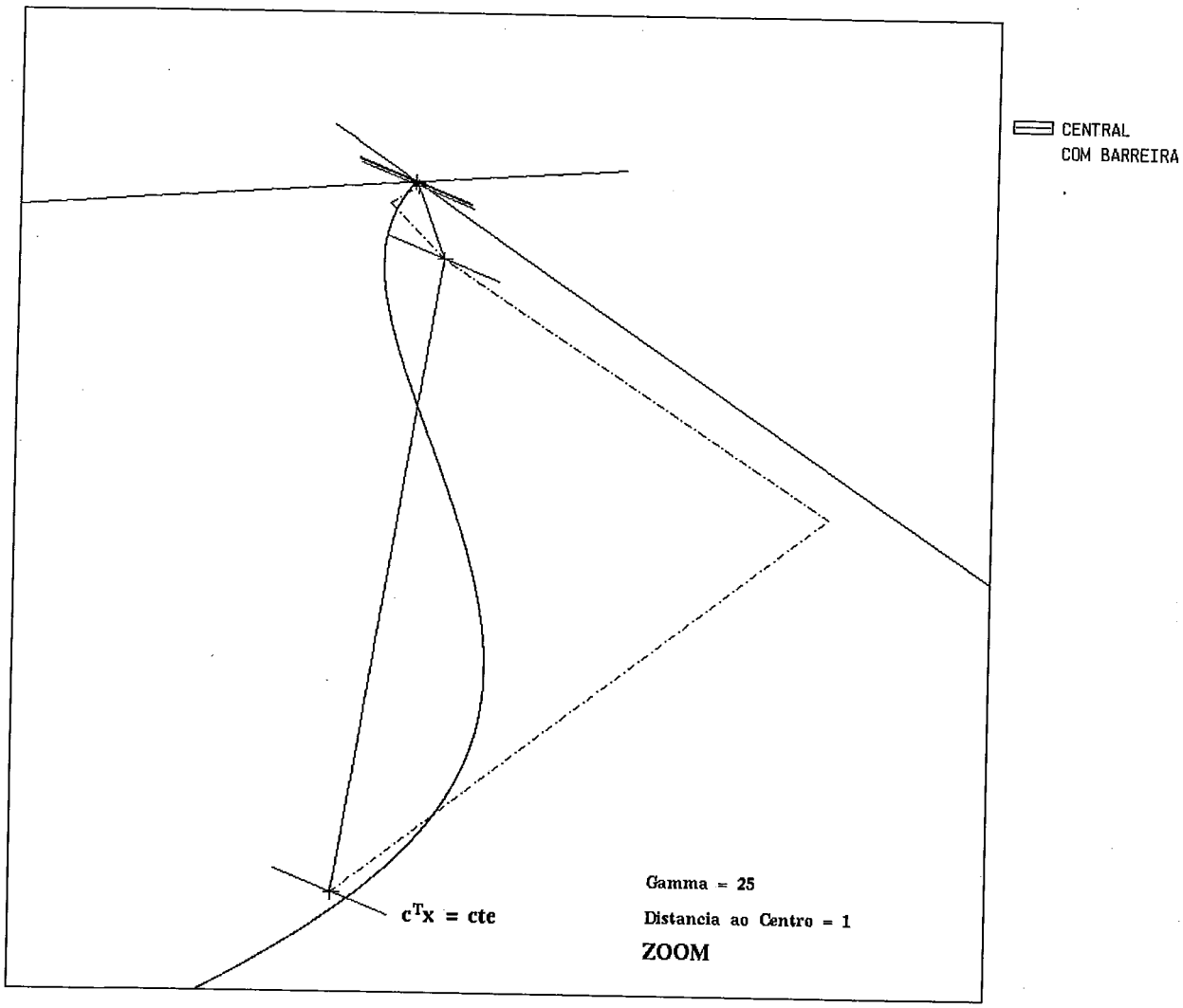


Figura VII.5: Iterações do Algoritmo de Centralização para $\gamma = 25$ com Zoom

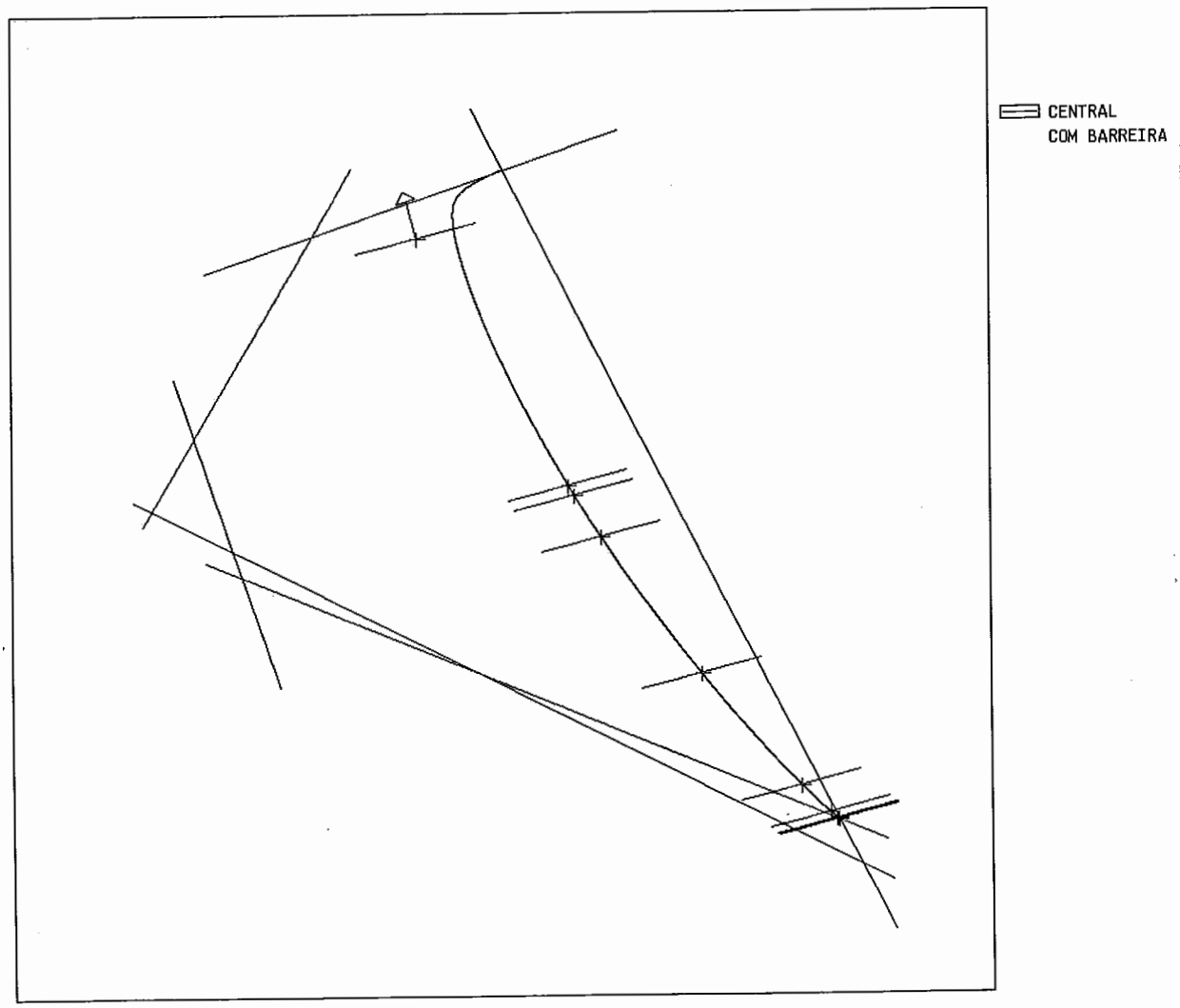


Figura VII.6: Iterações do Algoritmo de Centralização para $\delta = 0.01$

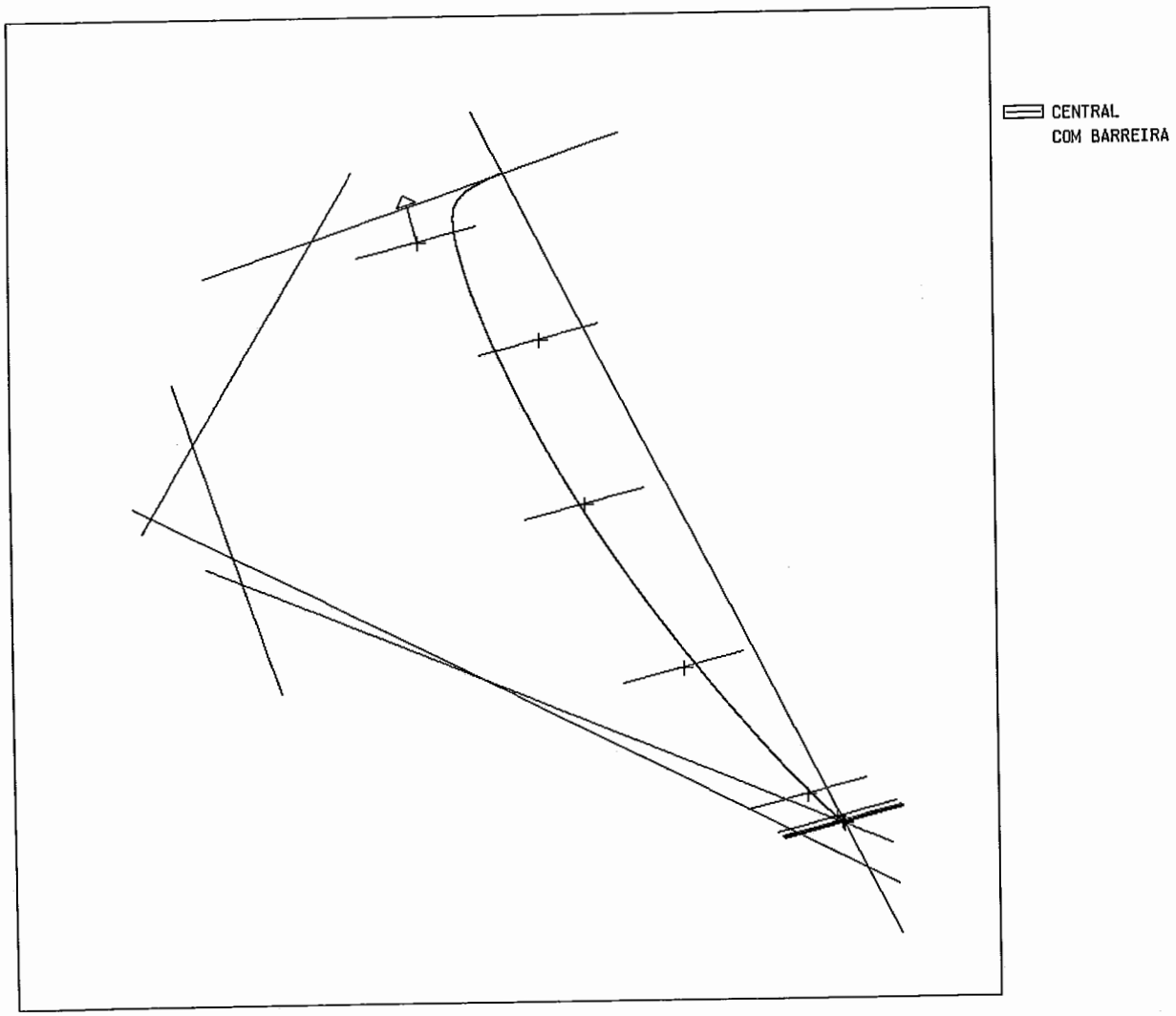


Figura VII.7: Iterações do Algoritmo de Centralização para $\delta = 1$

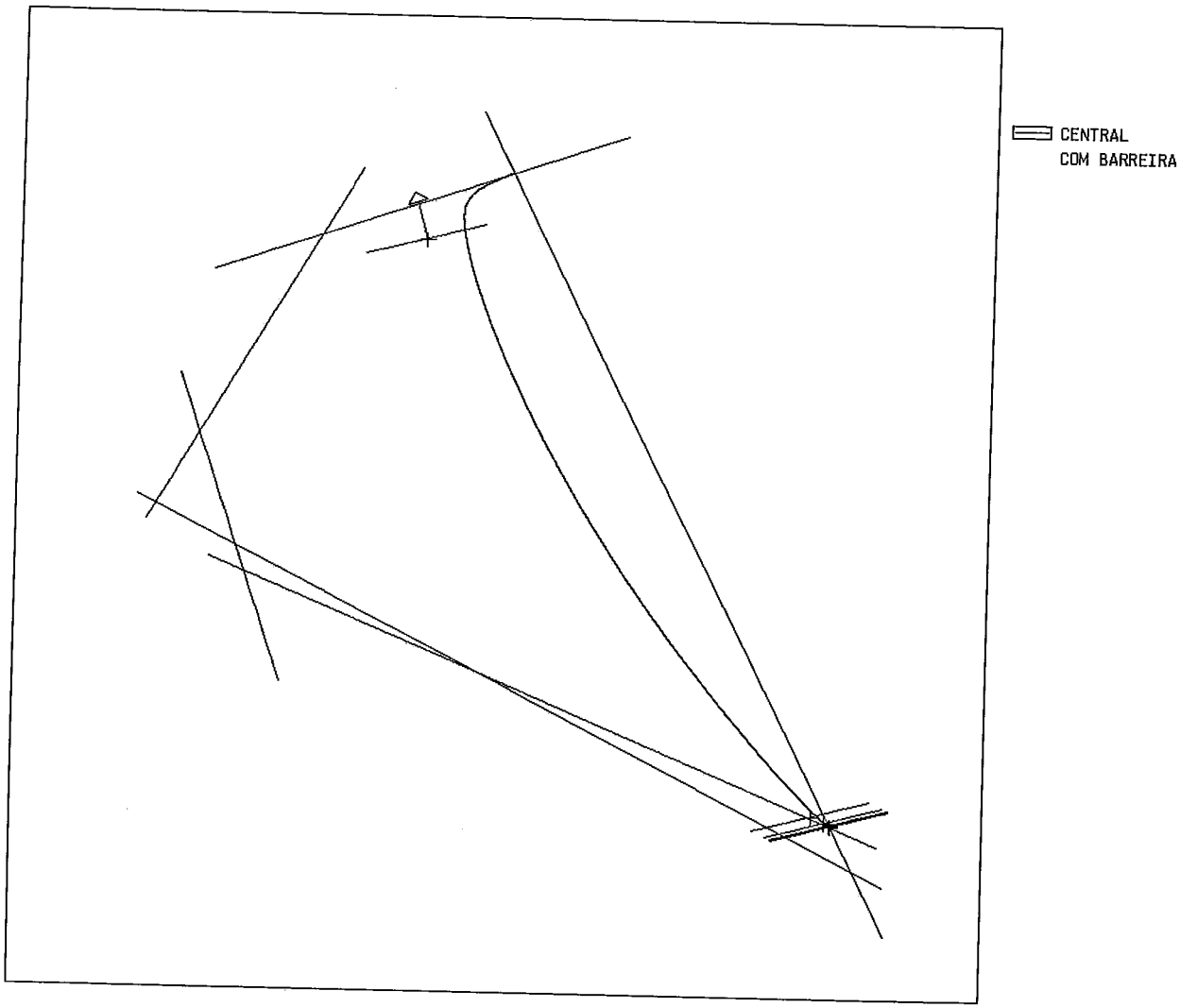


Figura VII.8: Iterações do Algoritmo de Centralização para $\delta = 5$

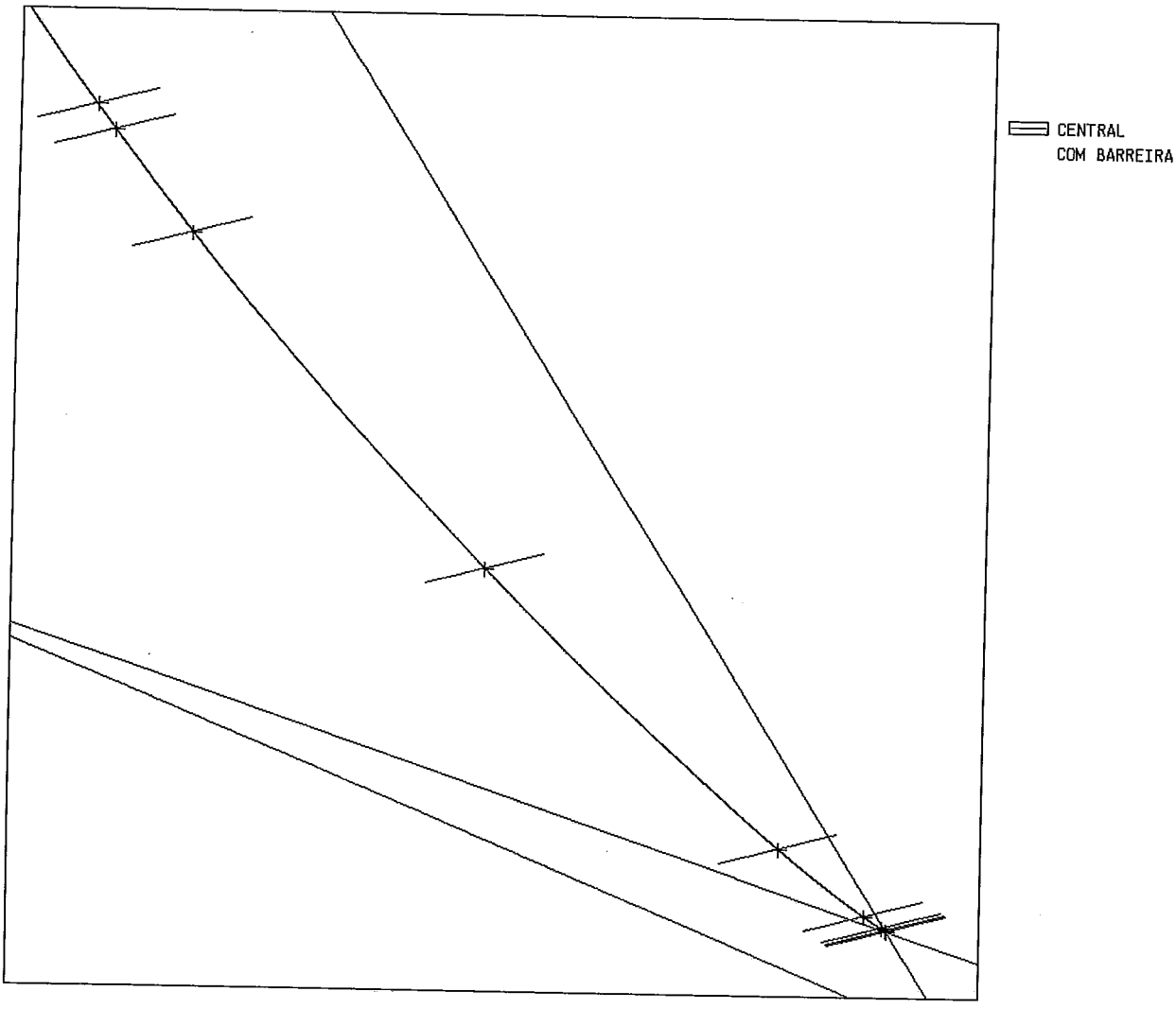


Figura VII.9: Iterações do Algoritmo de Centralização para $\delta = 0.01$ com Zoom

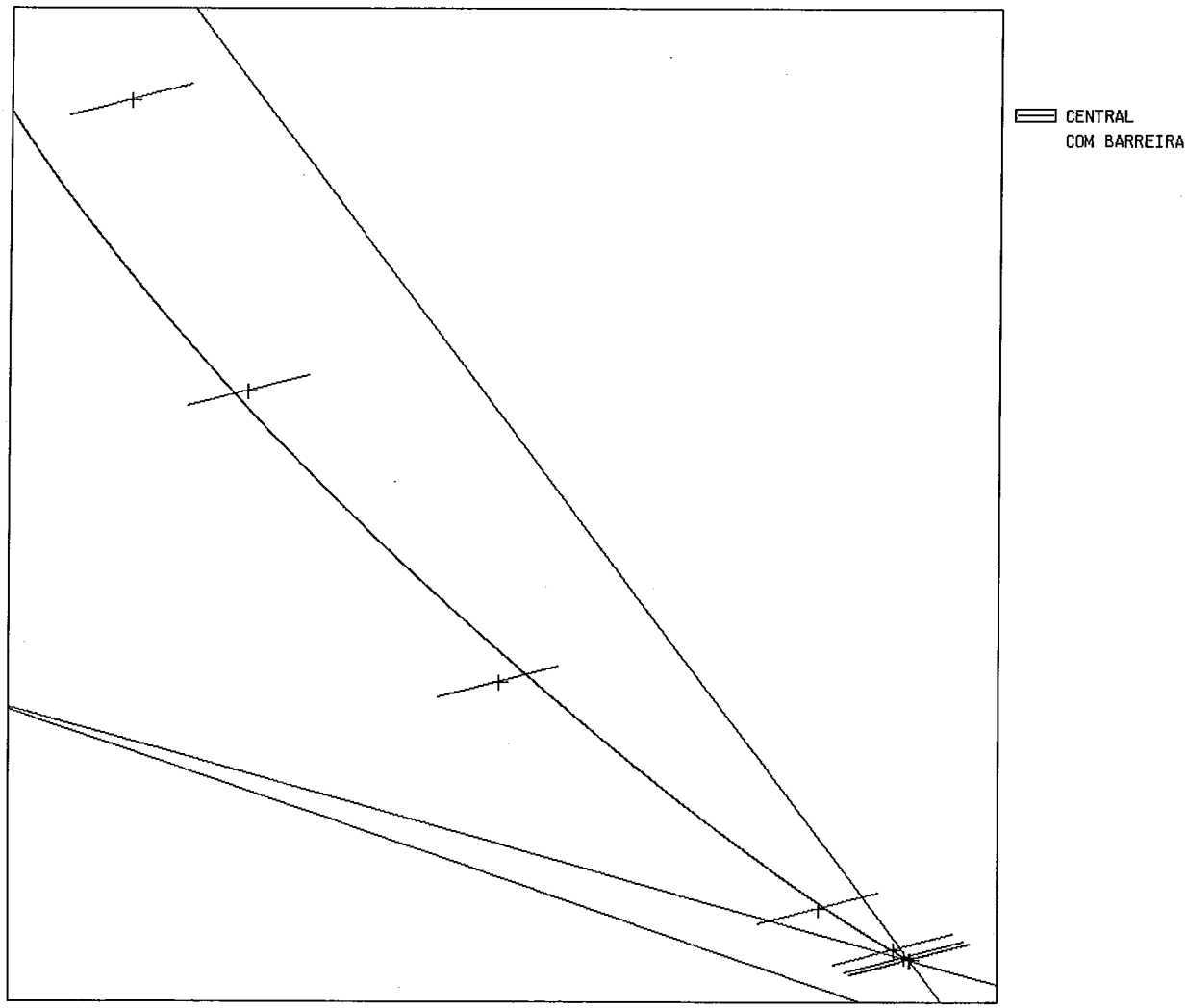


Figura VII.10: Iterações do Algoritmo de Centralização para $\delta = 1$ com Zoom

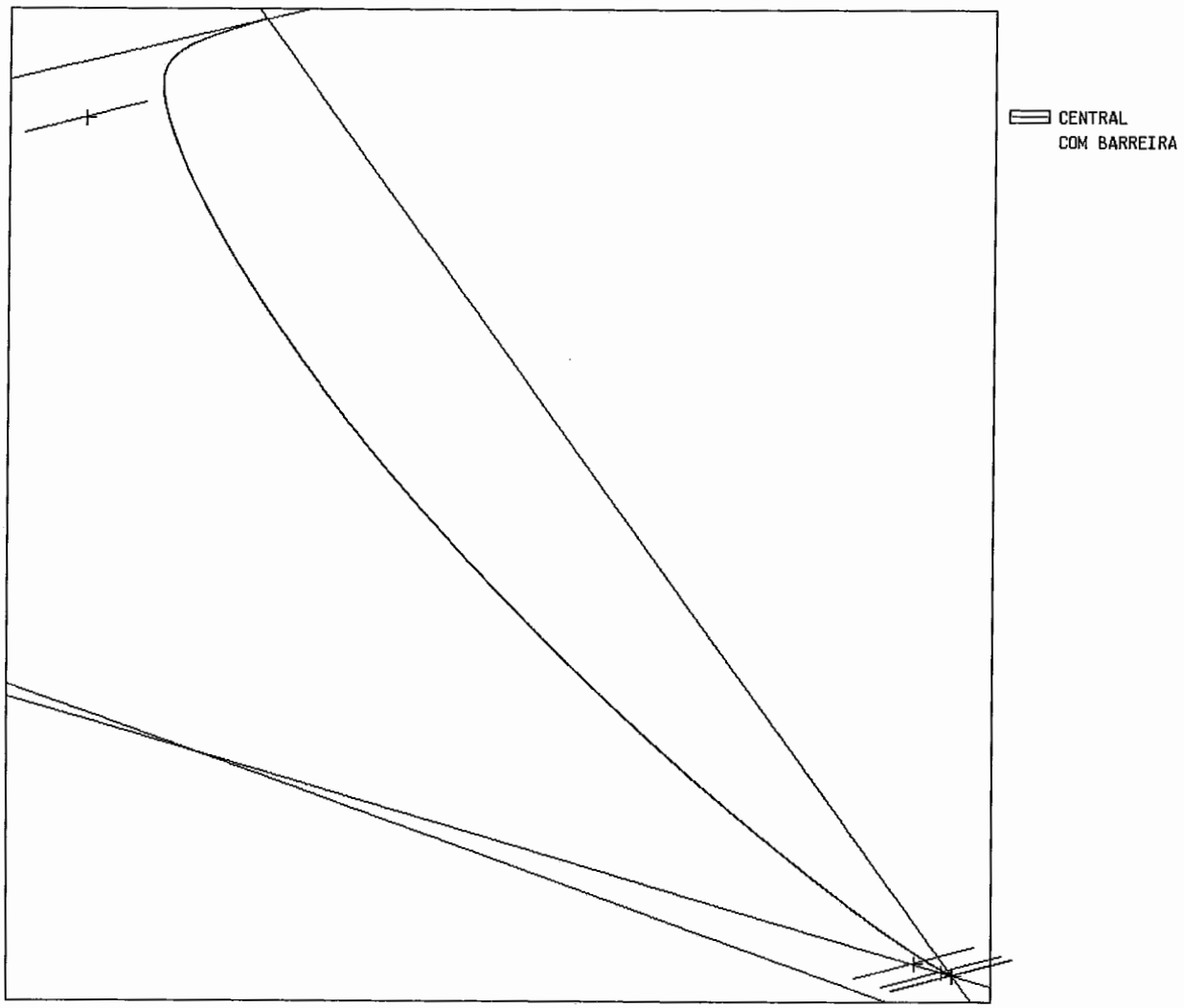


Figura VII.11: Iterações do Algoritmo de Centralização para $\delta = 5$ com Zoom

Capítulo VIII

Implementação

Os algoritmos de pontos interiores são facilmente implementáveis devido à sua simplicidade e eficiência. Entretanto, quando da implementação surgem vários fatores que não são levados em conta quando da análise teórica dos algoritmos. O maior "problema" do ponto de vista de implementação dos algoritmos está na inversão de matrizes, já que todos os algoritmos utilizam uma projeção para obter uma direção ao longo da qual a função objetivo (no caso, a função custo) será minimizada (ver capítulo I — seção I.2).

Para a visualização gráfica, pode-se utilizar qualquer método numérico de inversão de matrizes já que a dimensão dos problemas é pequena (só analisamos problemas no $\Re^2$ e $\Re^3$ e com um número pequeno de restrições - da ordem de 60).

Para se garantir uma melhor precisão numérica dos algoritmos, todos os problemas foram normalizados, tomando-se como norma a norma euclidiana no $\Re^n$.

Todos os aspectos de implementação citados nos parágrafos anteriores são referentes à implementação numérica dos algoritmos de pontos interiores. Nas seções subseqüentes descreveremos a implementação referente à visualização gráfica.

VIII.1 Curvas de Nível da Função Barreira

No capítulo IV - seção IV.2 apresentamos um algoritmo para se visualizar as curvas de nível da função barreira. Do ponto de vista gráfico, para que o algoritmo fique mais eficiente é necessário uma modificação naquele algoritmo. Observe que no algoritmo anterior estávamos procurando pontos $x \in S$ tal que :

$$p(x) = k, \text{ onde } k \text{ é uma constante}$$

mas como $p(x) = \sum_{i=1}^{m} \log[b - Ax]_i$, temos que :

$$p(x) = \log(\prod_{i=1}^{m}[b - Ax]_i)$$

logo :

$$\log(\prod_{i=1}^{m}[b - Ax]_i) = k$$

e então :

$$\prod_{i=1}^{m}[b - Ax]_i = e^k$$

mas como k é constante, e^k também é constante.

Assim, em vez de se utilizar a função barreira logarítmica p, utiliza-se a função multiplicativa $\theta(\cdot)$ dada por :

$$x \in S \longmapsto \theta(x) = \prod_{i=1}^{m}[b - Ax]_i.$$

Para se desenhar as curvas de nível, utilizou-se o método de Bezier.Para tal, é necessário que se escolham pontos de controle. Escolhemos os pontos de controle de modo que a curva obtida fosse contínua de primeira ordem. O procedimento adotado foi o seguinte :

1. a cada quatro pontos da curva p^0, p^1, p^2, p^3, obtenha a reta paralela ao segmento de reta p^0p^2 que passe por p^1; e a reta paralela ao segmento de reta p^1p^3 que passe por p^2;
2. em seguida, obtenha o ponto de interseção entre as duas retas paralelas descritas no item anterior. O ponto resultante é o ponto de controle para a curva de Bezier.

O ponto de controle obtido pelo procedimento acima, irá gerar o segmento de curva entre os pontos p^1 e p^2; o procedimento acima é então repetido para todos os pontos da curva até que a curva fique fechada. O procedimento descrito acima garante a continuidade de primeira ordem, pois os lados dos dois triângulos (gerados por p^1, ponto de controle e p^2) adjacentes ao mesmo ponto estão numa mesma linha.

VIII.2 Curvas de Nível para o método de Renegar

O procedimento a ser adotado é o mesmo que no item anterior só que agora a função $\theta(\cdot)$ é dada por :

$$x \in S^0, c^T x < K \longmapsto \theta(x) = (K - c^T x)^q (\prod_{i=1}^{m+1} [\bar{b} - \bar{A}x]_i)$$

onde K, q estão definidos como em (VI.1) e $\bar{A}$ e $\bar{c}$ estão definidos como no problema (PR) — ver seção VI.1.

VIII.3 Trajetória Central

Para se visualizar a trajetória central, utilizou-se também o método de Bezier. Para se obter os pontos de controle, utilizou-se o fato de que a direção gerada pelo algoritmo afim-escala é tangente à trajetória central; esse fato está demonstrado em [22]. Assim, adotamos o seguinte procedimento para a geração dos pontos de controle :

1. a cada dois pontos $p^0 e p^1$ da curva, obtenha a reta que passe por p^0 na direção gerada pelo algoritmo afim escala a partir de p^0; e a reta que passe por p^1 na direção gerada pelo algoritmo afim escala a partir de p^1;
2. obtenha o ponto de interseção entre as duas retas mencionadas no item anterior. O ponto resultante será o ponto de controle para o "segmento" de curva que passe por p^0 e p^1.

A continuidade de primeira ordem é garantida do mesmo modo que na seção VIII.1, ou seja, os polígonos gerados pelos pontos de controle são triângulos e os polígonos adjacentes gerados possuem um vértice em comum.

VIII.4 Elipses

As elipses foram geradas também através do método de Bezier, com pontos de controle gerados pelo mesmo procedimento adotado para as curvas de nível da função barreira — ver seção VIII.1.

VIII.5 Programa Interativo

O programa interativo que resultou da implementação dos algoritmos de pontos interiores, permite ao usuário visualizar qualquer um dos algoritmos estudados do capítulo II ao capítulo VII. A região viável pode ser pré-definida com os valores da matriz A, dos vetores b e c, e com um ponto inicial viável x^0; pode-se também desenhar a região viável e variar o ponto inicial viável x^0 e o vetor custo c.

Os algoritmos dos capítulos II, III e V podem ser visualizados a cada iteração; enquanto que o algoritmo de trajetória central pode ser visualizado com a trajetória central ou com os passos (curtos ou longos).

Toda a implementação computacional foi feita na linguagem C.

Capítulo IX

Conclusão

Conseguimos implementar todos os algoritmos para problemas com diferentes regiões viáveis e visualizar todos os algoritmos para problemas no $\Re^2$.

Todos os resultados teóricos testados foram os esperados quando da visualização gráfica. Apesar de trabalharmos com problemas de dimensões pequenas, podemos observar utilizando o programa interativo, que a medida que o número de restrições aumenta os algoritmos de trajetória central convergem mais rapidamente para um ótimo.

Para problemas no $\Re^3$ só analisamos o caso de uma região pré-definida bem simples e somente para a trajetória central. Sugerimos que para implementações posteriores sejam analisadas regiões no $\Re^3$ para todos os algoritmos que foram estudadas nessa tese, possivelmente usando técnicas de sombreamento para a trajetória central. Esse seria um estudo muito importante, pois no espaço tridimensional poderia-se visualizar alguns casos patológicos que não ocorrem no $\Re^2$.

Referências Bibliográficas

[1] I. Adler and R. Monteiro. *Limiting Behaviour of the Affine Scaling Continuous Trajectories for Linear Programming Problems.* Manuscript, Dept. of Industrial Engineering and Operations Research, University of California at Berkeley, Berkeley, CA, 1988.

[2] E. Barnes, D. Jensen, and Chopra. *A Polynomial-Time Version of the Affine-Scaling Algorithm.* Manuscript, New York University, New York, NY, 1988.

[3] E. R. Barnes. A variation on Karmarkar's algorithm for solving linear programming problems. *Mathematical Programming*, 36:174–182, 1986.

[4] G. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In Tj. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347, Wiley, N. York, 1951.

[5] I. I. Dikin. Iterative solution of problems of linear and quadratic programming. *Soviet Mathematics Doklady*, 8:674–675, 1967.

[6] I. I. Dikin. On the speed of an iterative process. *Upravlyaemye Sistemi*, 12:54–60, 1974.

[7] A. Fiacco and G. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley and Sons, New York, N.Y., 1968.

[8] K. R. Frisch. *The Logarithmic Potential Method of Convex Programming.* Memorandum, University Institute of Economics, Oslo, Norway, 1955.

[9] C. Gonzaga. An algorithm for solving linear programming problems in $O(n^3L)$ operations. In N. Megiddo, editor, *Progress in Mathematical Pro-*

gramming – Interior Point and Related Methods, chapter 1, Springer Verlag, Berlin, 1989.

[10] C. Gonzaga. Conical projection algorithms for linear programming. *Mathematical Programming*, 43:151–173, 1988.

[11] C. Gonzaga. *Interior Point Algorithms for Linear Programming Problems with Inequality Constraints.* Internal report ES-140/88, Programa de Eng. de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, Brasil, January 1988. To appear in *Mathematical Programming.*

[12] C. Gonzaga. *Large-Steps Path-Following Algorithms for Linear Programming: Barrier Function Method.* Internal report , COPPE – Federal University of Rio de Janeiro, Rio de Janeiro, Brasil, July 1989. To appear in *Siam Journal on Optimization.*

[13] C. Gonzaga. *Path Following Algorithms for Linear Programming.* Manuscript, COPPE – Federal University of Rio de Janeiro, Rio de Janeiro, Brasil, 1990.

[14] C. Gonzaga. Polynomial affine algorithms for linear programming. *Mathematical Programming*, 49:7–21, 1990.

[15] C. Gonzaga. *Search Directions for Interior Linear Programming Methods.* Memorandum UCB/ERL M87/44, Electronics Research Laboratory, University of California, Berkeley, CA, March 1987. To appear in *Algorithmica.*

[16] P. Huard and B. Liêu. La méthode de centres dans un espace topologique. *Numerische Mathematik*, 8:56–67, 1966.

[17] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[18] N. Karmarkar, M. Resende, and K. Ramakrishnan. *An Interior Point Algorithm to Solve Computationally Difficult Set Covering Problems.* Manuscript, AT&T Bell Laboratories, Murray Hill, NJ, 1989.

[19] L. G. Khachiyan. A polynomial algorithm for linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[20] L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computaional Mathematics and Mathematical Physics*, 20:53–72, 1980.

[21] V. Klee and G. Minty. How good is the simplex algorithm? In O. Sisha, editor, *Inequalities III*, Academic Press, New York, NY, 1972.

[22] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming – Interior Point and Related Methods*, chapter 8, Springer Verlag, Berlin, 1989.

[23] R. Monteiro, I. Adler, and M. Resende. A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension. *Mathematics of Operations Research*, 15:191–214, 1990.

[24] J. Renegar. A polynomial-time algorithm based on Newton's method for linear programming. *Mathematical Programming*, 40:59–94, 1988.

[25] G. Sonnevend. An analytical centre for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming. In *Lecture Notes in Control and Information Sciences 84*, pages 866–876, Springer Verlag, New York, NY, 1985.

[26] M. Todd and B. Burrell. An extension of Karmarkar's algorithm for linear programming using dual variables. *Algorithmica*, 1:409–424, 1986.

[27] P. Vaidya. *An Algorithm for Linear Programming which Requires* $O(((m+n)n^2+(m+n)^{1.5}n)L)$ *Arithmetic Operations.* Preprint, AT&T Bell Laboratories, Murray Hill, NJ, 1987.

[28] R. J. Vanderbei, M. J. Meketon, and B. A. Freedman. A modification of Karmarkar's linear programming algorithm. *Algorithmica*, 1:395–407, 1986.