Universidade Estadual de Campinas
Instituto de Matemática, Estatística e
Computação Científica - IMECC

Um estudo sobre sistemas de inequações lineares

André Rodrigues Monticeli
andremonticeli@yahoo.com.br
Dissertação de Mestrado

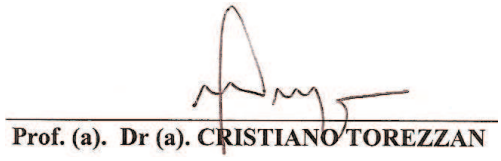
Orientador(a): **Prof. Dr. Cristiano Torezzan**

Março de 2010
Campinas - SP

Um estudo sobre sistemas de inequações lineares

Este exemplar corresponde a redação final da dissertação devidamente corrigida e defendida por **André Rodrigues Monticeli** e aprovada pela comissão julgadora.

Campinas, 25 de março de 2010.



Banca Examinadora:
Prof. Dr. Cristiano Torezzan (FCA - Unicamp)
Prof^a. Dr^a. Sandra A. Santos (IMECC - Unicamp)
Prof. Dr. Paulo A. Valente Ferreira (FEEC - Unicamp)

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica, **UNICAMP**, como requisito parcial para obtenção de Título de **Mestre em Matemática**.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Crisllene Queiroz Custódio – CRB8 / 7966

Monticeli, André Rodrigues
M763e Um estudo sobre sistemas de inequações lineares / André Rodrigues Monticeli -- Campinas, [S.P. : s.n.], 2010.

Orientador : Cristiano Torezzan
Dissertação (Mestrado Profissional) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Inequações lineares. 2. Poliedros. 3. Vértices. 4. Fecho convexo. 5. Pontos extremos. I. Torezzan, Cristiano. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Título em inglês: Studing system of linear inequalities.

Palavras-chave em inglês (Keywords): 1. Linear inequalities. 2. Polyhedron. 3. vertex. 4. Convex hull. 5. Extreme points.

Área de concentração: Matemática

Titulação: Mestre em Matemática

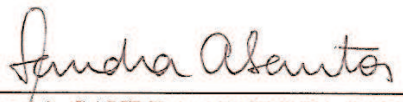
Banca examinadora: Prof. Dr. Cristiano Torezzan (FCA-Unicamp)
Prof. Dra. Sandra Augusta Santos (IMECC-Unicamp)
Prof. Dr. Paulo Augusto Valente Ferreira (FEEC-Unicamp)

Data da defesa: 25/03/2010

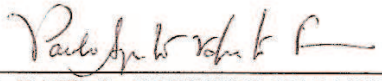
Programa de Pós-Graduação: Mestrado Profissional em Matemática

Dissertação de Mestrado Profissional defendida em 25 de março de 2010 e aprovada pela Banca Examinadora composta pelos Profs. Drs.

Prof. (a). Dr (a). CRISTIANO TOREZZAN



Prof. (a). Dr (a). SANDRA AUGUSTA SANTOS



Prof. (a). Dr (a). PAULO AUGUSTO VALENTE FERREIRA

Há pessoas que desejam saber só por saber, e isso é curiosidade; outras, para alcançarem fama, e isso é vaidade; outras, para enriquecerem com a sua ciência, e isso é um negócio torpe: outras, para serem edificadas, e isso é prudência; outras, para edificarem os outros, e isso é caridade.

São Tomás de Aquino

Agradecimentos

Expresso estima e agradecimentos a todos aqueles que contribuíram com o sucesso deste estudo.

Ao Professor Cristiano Torezzan, orientador deste estudo, por sua dedicação extraordinária, ensinamento, compreensão, amizade e pelo inestimável incentivo recebido.

À Professora Sueli Costa e a toda a Comissão do Programa de Mestrado Profissional em Matemática e aos professores pelo apoio, ensino e por acreditarem em mim.

Aos monitores João Paulo e Agnaldo pelo incentivo, dedicação e amizade.

Aos meus pais Célio e Celeste pelo apoio dado e pela paciência em todas as horas desta caminhada.

Aos meus irmãos, por estarem sempre por perto oferecendo apoio quando precisei.

A todos os meus familiares e amigos que sempre estiveram presentes me aconselhando e incentivando com carinho e dedicação.

À Lilia e Wanessa, por toda ajuda, compreensão e incentivo.

À Aida por ter a palavra certa na hora certa e por sempre acreditar em mim.

A todas as pessoas que, direta ou indiretamente, contribuíram para a execução desse estudo.

Agradeço em especial ao casal Paulo Antônio Arouca e Daizy Aparecida Azevedo Arouca e filhos, por todo apoio que me ofereceram no decorrer desse estudo.

E agradeço, sobretudo, a Deus por ter me dado a oportunidade de realizar esse trabalho.

Resumo

Neste trabalho abordamos o problema de descrever o conjunto solução de um sistema de inequações lineares. Este problema está fortemente relacionado com o problema clássico da enumeração de vértices de um poliedro. Descrevemos o método de Fourier-Motzkin que pode ser utilizado para eliminar variáveis de um sistema de inequações lineares e projetar a região de solução num espaço de dimensão menor. Mostramos como o problema da enumeração de vértices pode ser convertido em um problema de encontrar o fecho convexo do conjunto de pontos dual ao sistema de inequações lineares, uma vez encontrado um ponto interior factível. Alguns algoritmos para o fecho convexo de um conjunto finito de pontos e também para encontrar um ponto interior factível são estudados. Nosso interesse, além de listar os vértices e as faces é também visualizar a região de solução utilizando um programa computacional. Para tanto propomos um método que constrói a lista dos vértices e faces do poliedro definido por um dado sistema de inequações lineares e grava o resultado num arquivo de texto puro com extensão obj, que é compatível com os principais softwares de visualização gráfica 3D. O método foi implementado no Octave e diversos testes foram feitos, analisando o custo computacional e possíveis dificuldades que podem surgir devido a erros numéricos ou falta de memória.

Palavras chave: sistema de inequações lineares; poliedro, vértices, faces, fecho convexo.

Abstract

In this work we approach the problem of describing the solution of a system of linear inequalities. This problem is closely related to the classical problem known as vertex enumeration. We describe the method of Fourier-Motzkin, that can be used to eliminate variables in a system of linear inequalities, projecting its solution in a lower dimensional space. We show how the vertex enumeration problem can be converted into an equivalent problem of finding the convex hull of a set of dual points, once found a feasible interior point. Some algorithms for convex hull and also for finding a feasible interior point are studied. Our interest is not only to store the vertices and faces but also visualize the correspondent polyhedron using a computer graphics software. In this way we propose a method that stores the polyhedron's vertices and faces and output the results into a plain text file with extension obj, which is a geometric definition file format that can be opened with all major 3D graphics software. The method was implemented in Octave and several tests were made, analyzing the computational cost and possible difficulties that may arise due to numerical errors or memory requirements.

Key words: system of linear inequalities, polyhedron, vertices, faces, convex hull.

Lista de Notações

$Ax \leq b :=$ forma matricial de um sistema de inequações lineares.

$A_{m \times n} :=$ matriz de m linhas e n colunas.

$m :=$ número de inequações do sistema $Ax \leq b$.

$n :=$ número de variáveis do sistema de inequações lineares $Ax \leq b$.

$\langle x, y \rangle :=$ produto inteiro.

$x :=$ um vetor coluna pertencente ao $\mathbb{R}^n$.

$H :=$ hiperplano no $\mathbb{R}^n$.

$conv(D) :=$ fecho convexo de um conjunto $D \subset \mathbb{R}^n$.

$P :=$ um poliedro do $\mathbb{R}^n$.

$C :=$ um conjunto convexo do $\mathbb{R}^n$.

$A_= x \leq b_= :=$ sistemas de equações implícitas em $Ax \leq b$.

$A_+ \leq b_+ :=$ sistema de todas as outras inequações em $Ax \leq b$.

$F :=$ face de um poliedro P.

$r_i :=$ raios.

$d :=$ direção do conjunto.

$|G| :=$ cardinalidade do conjunto G.

$I^- :=$ índices das inequações onde os coeficientes de x_1 são negativos.

$I^+ :=$ índices das inequações onde os coeficientes de x_1 são positivos.

$I^0 :=$ índices das inequações onde os coeficientes de x_1 são nulos.

$C \in \mathbb{R}^{m \times n} :=$ matriz real C com m linhas e n colunas.

$v_i :=$ vértices do poliedro P.

$C_{m,n} :=$ combinação de m inequações por n variáveis.

$u_i = (\bar{a}_{i1}, ..., \bar{a}_{in}) :=$ vetor dual do hiperplano $\bar{a}_{i1}x_1 + ... + \bar{a}_{in}x_n$.

$c :=$ um ponto interior do poliedro.

Sumário

Introdução

Em diversos cursos de matemática básica e até universitária, sistemas de equações lineares são bastante estudados e várias publicações a respeito deste assunto são conhecidas. Em contraste disto, os sistemas de inequações lineares são pouco abordados, inclusive em cursos de álgebra linear, sendo que alguns desses cursos nem sequer mencionam tal assunto. Com este trabalho procuramos explorar esta lacuna tendo em vista dois objetivos principais. O primeiro consiste em investigar como o conjunto solução de um sistema de inequações lineares qualquer pode ser obtido. O segundo é elaborar um material introdutório sobre o assunto que seja compatível para uso em aplicações de álgebra linear.

O conjunto solução de um sistema de inequações lineares é denominado poliedro. Quando este conjunto for limitado, diremos que é um politopo. Existem diversos problemas clássicos relacionados com nosso trabalho, tais como o problema da enumeração dos vértices, do fecho convexo, dentre outros. Tais problemas pertencem a uma área da matemática aplicada ou computação conhecida como geometria computacional.

A geometria computacional emergiu da área de desenvolvimento e análise de algoritmos em meados dos anos 70. Seu objetivo é estudar problemas geométricos sob o ponto de vista algorítmico.

A procura por algoritmos para resolver problemas geométricos vem desde a época da antiguidade. São bem conhecidas as construções geométricas de Euclides, que usavam como instrumentos régua e compasso e consistiam de algumas operações que podiam ser realizadas com esses instrumentos. Um problema clássico de construção geométrica através de régua e compasso é o chamado Problema de Apolônio (cerca de 200 a.C.), no qual três circunferências arbitrárias no plano são dadas e pede-se uma quarta circunferência que seja tangente às três circunferências dadas. Euclides apresentou um algoritmo que resolve este problema.

Em geometria computacional o interesse é projetar algoritmos eficientes para resolver problemas geométricos. Muitos desses problemas têm sua origem em outras

áreas, como computação gráfica, robótica, processamento de imagens e até cristalografia.

Muitos problemas que aparecem nestas áreas têm enunciados simples e várias soluções possíveis, desde as mais ingênuas até as mais eficientes. Alguns exemplos desses problemas são:

- Fecho convexo: Dado um conjunto de pontos, determine o menor poliedro convexo contendo todos os pontos.
- Interseção de segmentos: Determinar as intersecções de um dado conjunto de segmentos de retas.
- Programação linear: Minimizar uma função linear tendo como domínio um poliedro.
- Par mais próximo: Dado um conjunto de pontos, determinar o par de pontos com menor distância entre eles.

Neste trabalho estaremos particulamente interessados no problema de representar graficamente o conjunto solução de um sistema de inequações lineares com duas ou três variáveis. Este problema está fortemente relacionado com o problema da enumeração dos vértices, cuja literatura é bastante vasta.

Os algoritmos para este problema dividem-se basicamente em duas classes [1]: grafos transversais e incrementais.

Os algoritmos de *grafos transversais* procuram construir a lista de vértices através de um grafo associado a bases (n facetas que contêm v vértices e cujos hiperplanos são independentes). Dois vértices de P são conectados por uma aresta de P se eles têm bases que diferem em apenas um membro. Alternar de um vértice para um adjacente equivale a uma mudança de base. Esta operação é conhecida como *pivoteamento*. Alguns algoritmos dessa classe são os *gift wrapping* [8], o algoritmo de Seidel [19] e o eficiente algoritmo *reverse search* de Avis e Fukuda [2] que explora uma busca reversa baseado no sistema de troca de base do simplex.

Os algoritmos *incrementais* para o problema da enumeração dos vértices determinam o conjunto de vértices calculando sucessivamente a interseção de uma sequência de semi-espaços. Em geral, inicialmente é construído um subconjunto de n ou $n+1$ semi-espaços e seus vértices. Em seguida, as demais restrições são adicionadas e a lista de vértices é atualizada através da solução de sistemas de equações lineares. Essencialmente cada atualização identifica e remove todos os vértices que não estão contidos no novo semi-espaço.

Com o intuito de tornar este trabalho didático e acessível a estudantes de graduação, vamos optar por uma das classes de algoritmos; caso contrário, poderíamos correr o risco de ter um estudo muito longo ou ainda incompleto. Assim, optamos pelos algoritmos incrementais, visto que esses são essencialmente baseados em conceitos de álgebra linear.

Ao longo do trabalho, vamos descrever e analisar alguns algoritmos para os diversos subproblemas que surgem quando desejamos representar graficamente um poliedro. O trabalho está dividido da seguinte forma:

No Capítulo 1 apresentamos alguns conceitos elementares de matrizes e análise convexa. As seções deste capítulo resumem o conteúdo encontrado em diversas referências: [3, 4, 13, 14, 15, 17].

No Capítulo 2, descrevemos o método de eliminação de Fourier-Motzkin e apresentamos diversos exemplos. Este método, além de possibilitar a eliminação sucessiva de variáveis em um sistema de inequações lineares também nos possibilita encontrar um ponto interior de um poliedro não-vazio, e também verificar se um sistema $Ax \leq b$ é factível. Como veremos, um dos problemas deste método é que o número de inequações cresce exponencialmente mais rápido do que as variáveis são eliminadas.

No Capítulo 3, concentram-se nossas principais contribuições. Apresentamos alguns algoritmos para encontrar o fecho convexo de um conjunto de pontos do $\mathbb{R}^2$ e do $\mathbb{R}^3$ e abordamos o problema da enumeração dos vértices e das faces. No Teorema 3.1 mostramos que este problema pode ser convertido em um problema de encontrar o fecho convexo do conjunto de pontos dual ao sistema de inequações linerares, uma vez encontrado um ponto interior factível. Este resultado é comentado em [10] e também em [16], no entanto sua apresentação formal não foi encontrada nas referências que pesquisamos. Assim, propomos o enunciado e demonstração do Teorema 3.1 como uma maneira de formalizar esta ideia. Além disso, estudamos no Capítulo 3 alguns métodos para encontrar um ponto interior de um poliedro e também para identificar e eliminar restrições redundantes em um sistema de inequações lineares.

Por fim, no Capítulo 4, faremos algumas comparações entre os algoritmos a fim de possibilitar a escolha daqueles que melhor respondem às nossas necessidades. Estes algoritmos são reunidos em uma única rotina (`desenha.m`) que gera um arquivo de texto com extensão `obj` contendo a lista de vértices e faces de um poliedro definido por um sistema de inequações lineares. Realizamos diversos testes comparando a rotina que propomos com o algoritmo `con2vert.m` disponível no site do MatLab para o problema da enumeração de vértices. Conforme mostrado nos Capítulos 3 e 4, a rotina que

propomos permite construir a lista de vértices e faces de um poliedro além de incluir rotinas de pré-processamento que a tornam mais robusta que a função `con2vert.m`.

O trabalho é finalizado com duas aplicações do método de eliminação de Fourier-Motzkin em conjunto com a rotina `desenha.m`. Na aplicação 2, desenhamos uma projeção de um hipercubo do $\mathbb{R}^4$ no $\mathbb{R}^3$ apoiado em um de seus vértices. A mesma ideia pode ser utilizada para visualizar a projeção de poliedros de dimensões maiores no $\mathbb{R}^3$. Os algoritmos que propomos foram implementados em Octave, que possui sintaxe similar à do MatLab e encontram-se disponíveis no anexo deste trabalho.

Capítulo 1

Desigualdades e Conjuntos Convexos

1.1 Introdução

Neste primeiro capítulo, vamos rever alguns conceitos básicos de álgebra linear e análise convexa, que serão utilizados ao longo deste trabalho. Inicialmente, veremos de forma breve alguns conceitos de álgebra linear, como álgebra de vetores e de matrizes. Em seguida iremos definir um conjunto convexo e analisar suas estruturas. Definiremos também o que é um fecho convexo, um poliedro, um cone convexo, além de tornar clara a notação que será utilizada no decorrer do trabalho, para chegarmos ao desenvolvimento do principal objetivo que é a representação gráfica dos poliedros. Os conceitos que são tratados neste capítulo, podem ser encontrados em vários textos como [3, 4, 5, 13, 14, 15, 17].

1.2 Alguns conceitos básicos de álgebra linear

1.2.1 Vetores

Denotamos por $\mathbb{R}^n$ o espaço vetorial de dimensão n sobre $\mathbb{R}$, com as operações usuais de soma e multiplicação. Os elementos $x \in \mathbb{R}^n$ são chamados **vetores** ou **pontos** e serão representados por n-uplas na forma coluna

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

O **produto interno** de dois vetores x e $y \in \mathbb{R}^n$, será denotado por $\langle x, y \rangle$ e definido por

$$\langle x, y \rangle = x_1.y_1 + x_2.y_2 + ... + x_n.y_n = \sum_{j=1}^{n} x_j.y_j.$$

Como é usual, a **norma euclidiana** ou o **comprimento de um vetor** $x \in \mathbb{R}^n$ é definida por:

$$||x|| = \sqrt{\langle x, x \rangle} = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} = \sqrt{\sum_{j=1}^{n} x_j^2}.$$

O produto interno permite, além de mensurar comprimento, introduzir o conceito de ângulo entre vetores.

O **ângulo θ entre dois vetores** x e $y \in \mathbb{R}^n$ satisfaz

$$cos\theta = \frac{\langle x, y \rangle}{||x||.||y||}.$$

Dizemos que x e y são ortogonais se, e somente se, $\langle x, y \rangle = 0$. Se dois vetores não nulos forem ortogonais então o ângulo entre eles será igual a $\frac{\pi}{2}$.

Uma coleção de vetores $x_1, x_2, ..., x_k$ de dimensão n é chamado **linearmente independente** (LI) se, e somente se

$$\sum_{j=1}^{k} \lambda_j x_j = 0 \quad \Rightarrow \quad \lambda_j = 0, \quad \forall j = 1, 2, ..., k.$$

Caso haja pelo menos um escalar $\lambda_j \neq 0$ dizemos que a coleção de vetores é **linearmente dependente** (LD).

Por exemplo, os vetores $a_1 = (1,2)^T$ e $a_2 = (-1,1)^T$ são linearmente independentes pois,

$$\lambda_1(1,2)^T + \lambda_2(-1,1)^T = (0,0)^T$$

$$\Downarrow$$

$$(\lambda_1, 2\lambda_1)^T + (-\lambda_2, \lambda_2)^T = (0,0)^T$$

$$\Downarrow$$

$$(\lambda_1 - \lambda_2, 2\lambda_1 + \lambda_2)^T = (0,0)^T$$

e resolvendo o sistema linear seguinte

$$\begin{cases} \lambda_1 - \lambda_2 &= 0 \\ 2\lambda_1 + \lambda_2 &= 0 \end{cases},$$

obtemos como resultado, $\lambda_1 = 0$ e $\lambda_2 = 0$.

1.2.2 Matrizes

Definição 1.1 *Denominamos **matriz** a um conjunto de números reais, ou a um conjunto de números complexos, dispostos em linhas e colunas, numa certa ordem, e colocados entre colchetes. Assim, uma matriz real, ou uma matriz complexa, que vamos denotar por A, com m linhas e n colunas é representada da forma:*

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

com $a_{ij} \in \mathbb{R}$, ou $a_{ij} \in \mathcal{C}$. Os escalares a_{ij} são denominados elementos da matriz, onde o primeiro índice indica a linha e o segundo índice indica a coluna às quais pertence o elemento. Neste caso, dizemos que a matriz A é de ordem $m \times n$.

Definição 1.2 *Dizemos que uma matriz A de ordem $m \times n$ é **quadrada** se $m = n$, isto é, se possui o mesmo número de linhas e de colunas. Neste caso, dizemos simplesmente que A é uma matriz de ordem n.*

Definição 1.3 *Dizemos que uma matriz A de ordem $m \times n$ é a* ***matriz nula*** *se seus elementos a_{ij} são todos nulos. Neste caso, denotamos $A = 0$. Frequentemente, indicamos $0_{m \times n}$ para denotar uma matriz nula de ordem $m \times n$.*

Definição 1.4 *Sejam $A = [a_{ij}]$ e $B = [b_{ij}]$ duas matrizes de ordem $m \times n$. Definimos a* ***soma*** *das matrizes A e B, que denotamos por $A + B$, como sendo a matriz $C = [c_{ij}]$, de ordem $m \times n$, onde cada elemento é definido da seguinte forma:*

$$c_{ij} = a_{ij} + b_{ij}; \qquad i = 1, ..., m \quad e \quad j = 1, ..., n.$$

Definição 1.5 *Sejam $A = [a_{ij}]$ uma matriz de ordem $m \times n$ e um escalar λ. Definimos a* ***multiplicação da matriz*** *A* ***pelo escalar*** *λ, e denotamos λA, como sendo a matriz $C = [c_{ij}]$, de ordem $m \times n$, onde cada elemento é definido da seguinte forma:*

$$c_{ij} = \lambda a_{ij}; \qquad i = 1, ..., m \quad e \quad j = 1, ..., n.$$

Definição 1.6 *Sejam A uma matriz de ordem $m \times p$ e B uma matriz de ordem $p \times n$. O* ***produto*** *AB, nesta ordem, é a matriz $C = [c_{ij}]$ de ordem $m \times n$ cujos elementos são definidos por:*

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{ik}; \qquad i = 1, ..., m \quad e \quad j = 1, ..., n,$$

isto é, o elemento c_{ij} é o produto da i-ésima linha de A pela j-ésima coluna de B. Assim, podemos definir o produto AB somente quando o número de colunas de A é igual ao número de linhas de B.

Definição 1.7 *Uma matriz quadrada de ordem n é chamada* ***matriz identidade****, e denotada por I, se sua diagonal principal for formada por $1's$ e os demais números por zeros.*

$$I = \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & 1 \end{bmatrix}$$

Definição 1.8 *Uma* ***matriz transposta*** *de uma matriz A, de ordem $m \times n$, é a matriz A^T, de ordem $n \times m$, que se obtém escrevendo ordenadamente as linhas de A como colunas.*

Definição 1.9 *O* ***determinante*** *de uma matriz quadrada A de ordem n, é a soma algébrica dos produtos que se obtém efetuando todas as permutações dos segundos índices do termo principal, fixados os primeiros índices e fazendo-se preceder o produto do sinal $+$ ou $-$, conforme a permutação dos segundos índices, seja de classe par ou de classe ímpar. Segue abaixo a definição:*

$$det(A) = \sum_{p} (-1)^J a_{1j_1} a_{2j_2} ... a_{nj_n}$$

onde $J = J(j_1...j_n)$ é o número de inversões da permutação $(j_1 j_2 ... j_n)$ e p indica que a soma é estendida a todas as $n!$ permutações de $(1, 2, ..., n)$.

Definição 1.10 *Seja A uma matriz quadrada de ordem n. Se B é uma matriz quadrada de ordem n tal que $AB = BA = I$, então B é chamada a* ***matriz inversa*** *de A. A matriz inversa, caso exista, é única e é denotada por A^{-1}. Se A possui inversa, A é chamada* ***não-singular****, caso contrário A será dita* ***singular****.*

Pode-se demonstrar [5] que, uma matriz $A_{n\times n}$ terá inversa se, e somente se, suas linhas forem linearmente independentes ou, equivalentemente, se suas colunas forem linearmente independentes. Também podemos dizer que uma matriz A possui inversa se seu determinante for diferente de zero $(det(A) \neq 0)$.

1.3 Conjuntos convexos

Definição 1.11 *Um conjunto $C \in \mathbb{R}^n$ é dito* ***convexo*** *se, e somente se, dado dois pontos x_1 e $x_2 \in C$, então,*

$$\lambda x_1 + (1 - \lambda) x_2 \in C$$

para todo $\lambda \in [0, 1]$. Notemos que $\lambda x_1 + (1 - \lambda) x_2$ para $\lambda \in [0, 1]$ representa um ponto do segmento de reta que liga x_1 e x_2.

Qualquer ponto da forma $\lambda x_1 + (1-\lambda) x_2$ onde $0 \leq \lambda \leq 1$, é chamado de **combinação convexa** de x_1 e x_2. Se $\lambda \in (0, 1)$, então a combinação convexa é dita **estrita**. Logo a convexidade de C pode ser interpretada geometricamente como se segue. Para cada par de pontos x_1 e $x_2 \in C$, o segmento que os interliga, ou a combinação convexa dos dois pontos deve estar inteiramente contido em C.

A ideia de combinação convexa entre dois pontos pode ser estendida para um conjunto qualquer de pontos da seguinte forma: dados $x_i \in \mathbb{R}^n$, $\lambda_i \in [0, 1], i = 1, ..., p$, tais

que $\sum_{i=1}^{p} \lambda_i = 1$, o ponto $\sum_{i=1}^{p} \lambda_i x_i$ chama-se combinação convexa de pontos $x_i \in \mathbb{R}^n$ com parâmetros $\lambda_i, i = 1, ..., p$.

Na Figura 1.1 vemos um exemplo de um conjunto convexo C_1 e na Figura 1.2 um exemplo de um conjunto não-convexo C_2. Neste último caso, podemos ver que nem todas as combinações convexas de x_1 e x_2 pertencem a C_2.

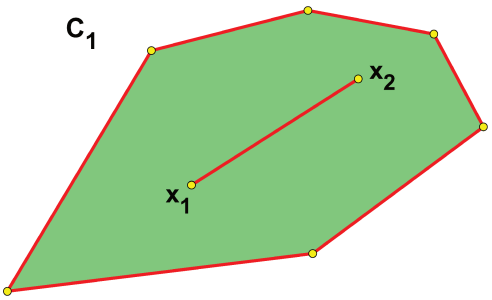


Figura 1.1: Exemplo de um conjunto convexo.

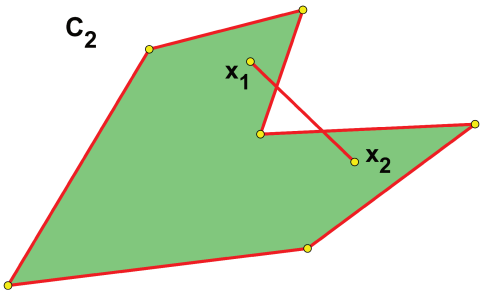


Figura 1.2: Exemplo de um conjunto não-convexo.

1.3.1 Retas, hiperplanos e semi-espaços

Definição 1.12 *Sejam x_1 e $x_2 \in \mathbb{R}^n$. A **reta r** que passa por estes pontos é definida como o conjunto de pontos x tais que*

$$x = x_1 + t(x_2 - x_1), \quad t \in \mathbb{R}.$$

Definição 1.13 *Um **hiperplano** H no $\mathbb{R}^n$ é um conjunto da forma*

$$H = \{x : \langle a, x \rangle = \alpha\}$$

onde $a \in \mathbb{R}^n$ é dito vetor normal ao hiperplano e $\alpha \in \mathbb{R}$ é um escalar.

Teorema 1.1 *Um hiperplano H é um conjunto convexo.*

Demonstração: Seja $x, y \in H$, logo $\langle a, x \rangle = \alpha$ e $\langle a, y \rangle = \alpha$. Queremos mostrar que

$$\lambda x + (1 - \lambda)y \in H, \quad \lambda \in [0, 1].$$

De fato,

$$\langle a, \lambda x + (1-\lambda)y \rangle = \lambda ax + (1-\lambda)ay = \lambda ax + ay - \lambda ay = \lambda\alpha + \alpha - \lambda\alpha = \alpha,$$

o que mostra que H é um conjunto convexo. Podemos notar que a hipótese $\lambda \in [0,1]$ não foi necessária na demonstração do Teorema 1.1, o que significa que a reta $r = x + t(y-x)$ está inteiramente contida em H.

Definição 1.14 *Um hiperplano divide o $\mathbb{R}^n$ em 2 regiões chamadas **semi-espaços**. Um semi-espaço é uma coleção de pontos $\{x : a^T x \leq b\}$; onde a é um vetor não nulo do $\mathbb{R}^n$ e b é um escalar.*

Um semi-espaço também pode ser representado como um conjunto de pontos $\{x : a^T x \geq b\}$. A união dos semi-espaços $\{x : a^T x \geq b\}$ e $\{x : a^T x \leq b\}$ é o próprio $\mathbb{R}^n$.

Referindo a um ponto fixo x_0 em um hiperplano H, definimos o semi-espaço que pode ser representado por $\{x : a^T(x - x_0) \geq 0\}$ ou por $\{x : a^T(x - x_0) \leq 0\}$.

A Figura 1.3 ilustra um hiperplano $a^T x = b$, o respectivo vetor normal a e os semi-espaços $a^T x < b$ e $a^T x > b$.

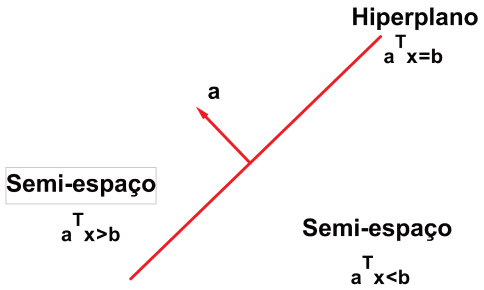


Figura 1.3: Hiperplano e semi-espaços.

1.3.2 Fecho convexo

Definição 1.15 *Seja $D \subset \mathbb{R}^n$ um conjunto qualquer. O **fecho convexo** de D, denotado $conv(D)$, é o menor conjunto convexo em $\mathbb{R}^n$ que contém D (ou, equivalentemente, a interseção de todos os conjuntos convexos em $\mathbb{R}^n$ que contêm D).*

Exemplo 1.1

O polígono P da Figura 1.4 é o fecho convexo do conjunto de pontos $p_1, p_2, ..., p_8$, ou seja, $conv(p_1, p_2, ..., p_8) = P$.

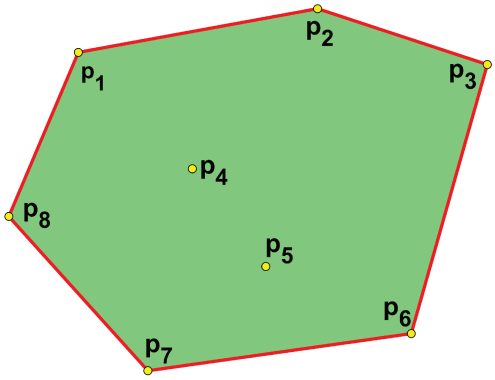


Figura 1.4: Fecho convexo.

Exemplo 1.2

O fecho convexo da superfície esférica é a bola, isto é, para

$$D = \{x \in \mathbb{R}^n : ||x|| = c\},$$

onde $c \in \mathbb{R}$, tem-se

$$conv(D) = \{x \in R^n : ||x|| \leq c\}.$$

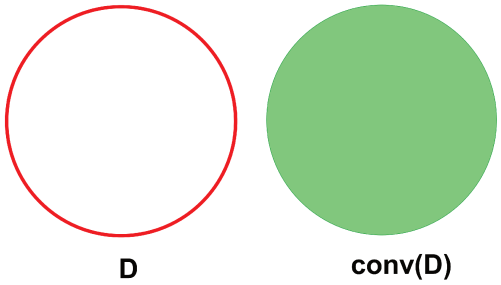


Figura 1.5: Fecho convexo da superfície esférica.

Exemplo 1.3

O fecho convexo de dois pontos x_1 e x_2 é o segmento cujos extremos são x_1 e x_2.

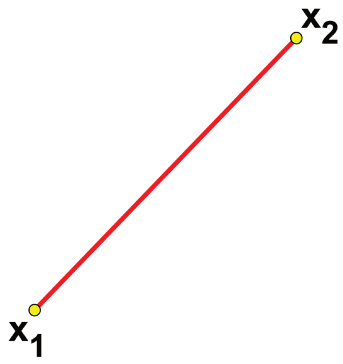


Figura 1.6: Fecho convexo de dois pontos x_1 e x_2.

É possível demonstrar [14] que o fecho convexo de um conjunto qualquer $D \in \mathbb{R}^n$ composto por p pontos é o conjunto de todas as combinações convexas de pontos de D, ou seja,

$$conv(D) = \left\{ y \in \mathbb{R}^n : y = \sum_{i=1}^{p} \lambda_i x^i \right\}$$

onde $x^i \in D$, $\lambda_i \geq 0$ e $\sum_{i=1}^{p} \lambda_i = 1$ para $i = 1, 2, ..., p$ e $p \in N$.

1.3.3 Sistemas de inequações lineares

Quando operamos com números reais, inequações lineares são escritas na forma $f(x) < b$ ou $f(x) \leq b$, onde $f(x)$ é uma função linear em números reais e b é um número real constante. Geralmente elas são escritas da seguinte forma:

$$a_1x_1 + a_2x_2 + ... + a_nx_n < b \quad \text{ou} \quad a_1x_1 + a_2x_2 + ... + a_nx_n \leq b$$

onde $x_1, x_2, ..., x_n$ são chamados de variáveis, $a_1, a_2, ..., a_n$ são os coeficientes lineares da inequação e b é uma constante real.

Exemplos de inequações lineares:

1. $2x + y < 3$
2. $5x_1 - 3x_2 + \dfrac{x_3}{3} \leq 5$
3. $3x_1 - 5x_2 \leq \dfrac{1}{2}$

4. $3x - y > 7$

Um **sistema de inequações lineares** é um conjunto de inequações lineares nas mesmas variáveis:

$$\begin{cases} a_{11}x_1 & + & a_{12}x_2 & + & ... & + & a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & ... & + & a_{2n}x_n & \leq & b_2 \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & ... & + & a_{mn}x_n & \leq & b_m \end{cases}$$

onde $x_1, x_2, ..., x_n$ são as variáveis, $a_{11}, a_{12}, ..., a_{mn}$ são os coeficientes lineares do sistema de inequações e $b_1, b_2, ..., b_m$ são os termos constantes.

Fazendo

$$A = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & ... & a_{mn} \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \text{ e } x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

podemos escrevê-lo na forma matricial,

$$Ax < b \quad ou \quad Ax \leq b$$

onde a matriz $A_{m \times n}$ é denominada matriz dos coeficientes, x é denominado vetor das variáveis, e b é denominado vetor dos termos independentes.

1.3.4 Poliedros

Definição 1.16 *Um **poliedro** P é a interseção de um número finito de semi-espaços $C_i \subset R^n$, ou seja, $P = \bigcap_{i=1}^{p} C_i$.*

De acordo com a Definição 1.14, um semi-espaço pode ser representado por uma inequação do tipo $a_i^T x \leq b$, logo, um poliedro pode ser representado pelo conjunto $\{x : Ax \leq b\}$ onde A é uma matriz $m \times n$, cuja linha i-ésima é o vetor a_i^T e b é um vetor de m componentes, ou seja, um poliedro pode ser representado por um sistema de inequações lineares. Quando o conjunto $\{x : Ax \leq b\}$ for uma região limitada, dizemos que o poliedro correspondente é um **politopo**.

Exemplo 1.4

Vamos considerar o seguinte sistema de inequações lineares:

$$\begin{cases} -x_1 + x_2 & \leq & 1 \\ -2x_1 - x_2 & \leq & 3 \\ -x_2 & \leq & 2 \end{cases} . \tag{1.1}$$

A região de solução de (1.1) está representada na Figura 1.7. Podemos notar que esta região é um poliedro ilimitado.

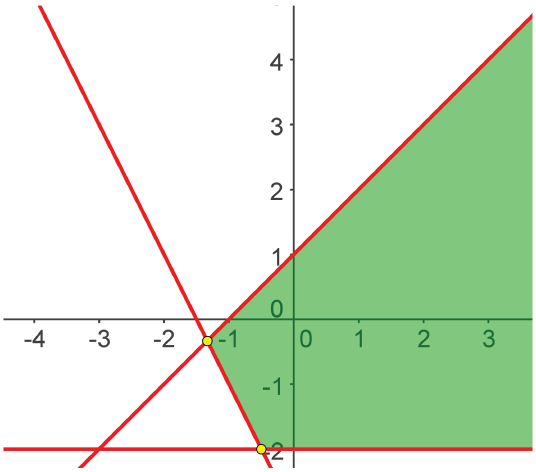


Figura 1.7: Poliedro solução do sistema de inequações lineares (1.1).

Exemplo 1.5

Vamos considerar agora o seguinte sistema

$$\begin{cases} -2x_1 + x_2 & \leq & 4 \\ x_1 + x_2 & \leq & 3 \\ x_1 & \leq & 2 \\ x_1 & \geq & 0 \\ x_2 & \geq & 0 \end{cases} , \tag{1.2}$$

cuja região de solução é o politopo representado na Figura 1.8.

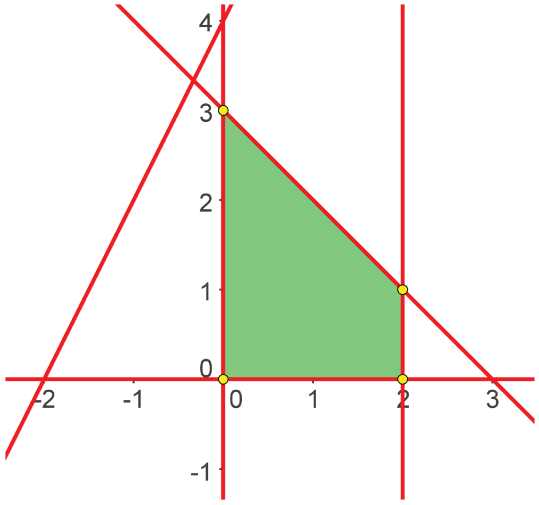


Figura 1.8: Politopo referente ao conjunto solução de (1.2).

Cada inequação de (1.2) determina um semi-espaço e a interseção destes 5 semi-espaços está representada na parte colorida da Figura 1.8. Podemos ver uma diferença entre a primeira inequação e as demais. Se a primeira inequação for ignorada, a região poliedral não é afetada. Quando isso ocorre, dizemos que a inequação é **redundante**, ou seja, pode ser ignorada que não irá alterar a região. Podemos dizer isso de maneira mais formal como segue.

Definição 1.17 *Uma restrição $a_1^T x \leq b_1$ em um sistema de inequações é dita redundante se ela é implicada pelas outras restrições do sistema. Um sistema é dito* ***irredundante*** *quando não possui nenhuma inequação redundante.*

Definição 1.18 *Uma inequação $a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \leq b_i$, sendo i a i-ésima linha do sistema $Ax \leq b$, é chamada* ***igualdade implícita*** *em $Ax \leq b$ se $a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n = b_i$ para todo x que satisfaz $Ax \leq b$.*

Vamos usar a notação:
$A_=x \leq b_=$ para o sistema de equações implícitas em $Ax \leq b$;
$A_+x \leq b_+$ para o sistema de todas as outras inequações em $Ax \leq b$.

É fácil vermos que existe um x no poliedro P que satisfaz $A_=x \leq b_=$ e $A_+x \leq b_+$.

Exemplo 1.6

Consideremos o sistema de inequações lineares

$$\begin{cases} -x+y+z & \leq & 1 \\ x+y+2z & \leq & 1 \\ -y+2z & \leq & 0 \\ z & \leq & 0 \\ -z & \leq & 0 \end{cases}.$$

Vamos escrevê-lo na forma matricial

$$\begin{bmatrix} -1 & 1 & 1 \\ 1 & 1 & 2 \\ 0 & -1 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

que tem a região de solução representada na Figura 1.9.

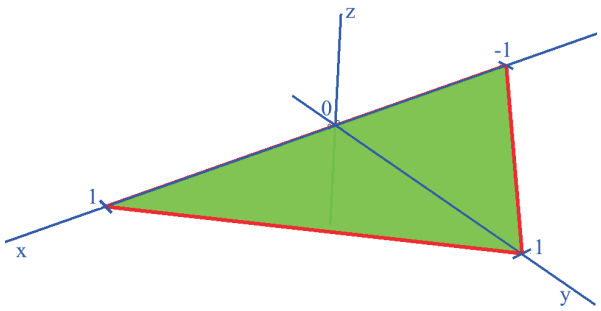


Figura 1.9: Região de solução do Exemplo 1.6.

As igualdades implícitas desse sistema são $z \leq 0$ e $-z \leq 0$, pois para todo ponto que satisfaz todas as inequações do sistema, também satisfaz as igualdades, logo o sistema $A_{=}x \leq b_{=}$ das igualdades implícitas é

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

A **dimensão** de um poliedro P do $\mathbb{R}^n$ é igual a n menos o posto da matriz $A_=$, logo a dimensão do poliedro do Exemplo 1.6 é 2, pois o posto$(A_=) = 1$, portanto, $\dim(P) = 3 - 1 = 2$.

Definição 1.19 *Se c é um vetor diferente de zero, e existe $\delta = \max\{c^T x : Ax \leq b\}$, o hiperplano $\{x : c^T x = \delta\}$ é chamado de* ***hiperplano suporte*** *do poliedro $P = \{x : Ax \leq b\}$.*

Definição 1.20 *Seja P um poliedro. Um subconjunto F de P é chamado de* ***face*** *de P se $F = P$ ou se F é a interseção de P com um hiperplano suporte de P.*

Exemplo 1.7

Seja um poliedro P caracterizado pelo sistema

$$\begin{cases} -x_1 + x_2 & \leq & 1 \\ x_1 + x_2 & \leq & 1 \\ -x_2 & \leq & 0 \end{cases},$$

onde

$$A = \begin{bmatrix} -1 & 1 \\ 1 & 1 \\ 0 & -1 \end{bmatrix} \text{ e } b = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

Tomemos o vetor $c = (0, -1)^T$. Um hiperplano suporte de P referente a este vetor c é

$$\delta = \max\{c^T x : Ax \leq b\} = \max\{\langle (0,-1), (x_1, x_2)\rangle : Ax \leq b\}$$

$$\Downarrow$$

$$\delta = \max\{-x_2 : Ax \leq b\} = 0.$$

Portanto, o hiperplano suporte é a reta $x_2 = 0$, e F é sua face correspondente, como está indicada na Figura 1.10.

Se tomarmos o vetor $c = (1, 0)^T$, teremos como hiperplano suporte a reta $x_1 = 1$. Logo, a face correspondente a este hiperplano seria o vértice v ilustrado na Figura 1.10.

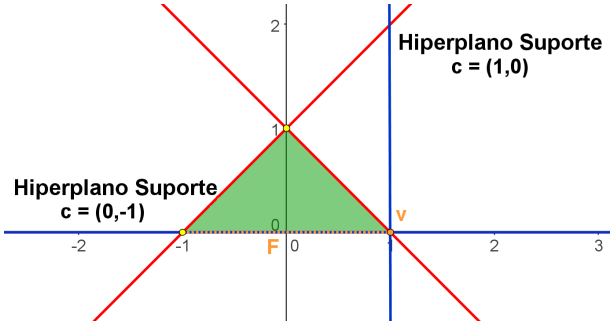


Figura 1.10: Hiperplano suporte e as faces correspondentes do Exemplo 1.7.

Alternativamente, F é uma face de P se, e somente se, F é não-vazio e $F = \{x \in P : A'x = b'\}$ para algum subsistema $A'x \leq b'$ de $Ax \leq b$.

Pode-se demonstrar (ver [18]) que:

- P possui um número finito de faces;
- cada face é um poliedro não-vazio;
- se F é uma face de P e $F' \subseteq F$, então, F' é uma face de P se, e somente se, F' é uma face de F.

Como uma face F é um poliedro, então podemos falar em dimensão da face. Ao longo deste trabalho utilizaremos o termo vértice para denominar as faces de dimensão 0 e arestas para denominar as faces de dimensão 1.

Uma face de maior dimensão que for distinta do próprio poliedro P é denominada **faceta**.

Se o sistema $Ax \leq b$ é irredundante [18], então existe uma correspondência um-a-um das facetas de P com as inequações do sistema.

A dimensão da faceta de P é uma a menos que a dimensão de P.

Consideremos o poliedro P caracterizado pelo sistema de inequações

$$\begin{cases} x & \leq & 1 \\ -x & \leq & -1 \end{cases}.$$

Logo, esse poliedro é o ponto $x = 1$ em $\mathbb{R}$. A matriz

$$A_{=} = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

é de posto 1. Assim, a dimensão desse poliedro é $\dim(P) = n - \text{posto}(A_=) = 1 - 1 = 0$.

Com isso, por convenção, $\dim(\oslash) = -1$.

Um importante exemplo de um politopo no $\mathbb{R}^n$ é o hipercubo.

Definição 1.21 *Um subconjunto P de $\mathbb{R}^n$ é chamado **hipercubo** se for um fecho convexo para todos os 2^n pontos com componentes 0 ou 1. Ele possui exatamente 2^n vértices e $2n$ facetas.*

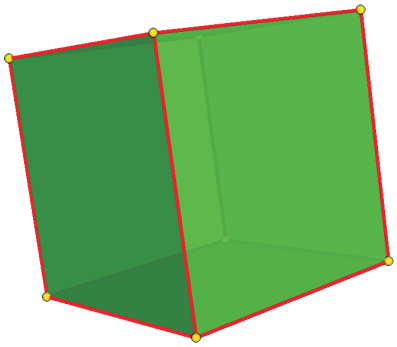


Figura 1.11: Hipercubo no $\mathbb{R}^3$.

Definição 1.22 *Um vetor x_0 pertencente a um conjunto convexo C é um **ponto extremo** de C, se x_0 não pode ser representado como uma combinação convexa estrita de dois vetores distintos de C.*

Um ponto extremo de um politopo é chamado de **vértice**.

Exemplo 1.8

Consideremos o conjunto convexo $C_1 = \{(x_1, x_2) : x_1^2 + x_2^2 \leq 1\}$. O conjunto de pontos extremos de C_1 é $\{(x_1, x_2) : x_1^2 + x_2^2 = 1\}$.

Existem conjuntos convexos que não possuem pontos extremos, como é o caso da reta $\{(x_1, x_2) : x_2 = 0\}$.

Definição 1.23 *Considere um poliedro P definido pelo sistema $Ax \leq b$ e seja y um vetor do $\mathbb{R}^n$. Dizemos que y é um **ponto factível** ou um **ponto viável** se y satisfazer todas as restrições (inequações) do sistema. Se y não satisfaz uma ou mais restrições (inequações), dizemos que y é um **ponto infactível**.*

Definição 1.24 *Seja P um poliedro do $\mathbb{R}^n$. Um ponto $x \in P$ é dito um* ***ponto interior*** *de P se x satisfaz estritamente o sistema de desigualdades, isto é, $Ax < b$.*

No conjunto convexo C representado na Figura 1.12, $x_1 = (1, 1)$ é um vértice (um ponto extremo) de C, enquanto que x_2 é um ponto interior.

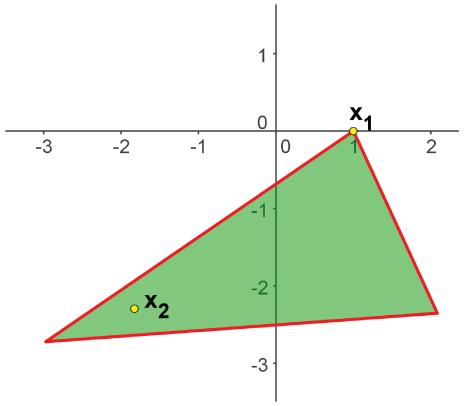


Figura 1.12: Exemplo de um vértice e de um ponto interior.

Definição 1.25 *Um politopo $P \subset \mathbb{R}^n$ é chamado* ***simples*** *se cada vértice está contido em exatamente n facetas.*

Um hipercubo é um exemplo de um politopo simples.

Um exemplo de um politopo não simples é a pirâmide de base hexagonal (Figura 1.13). Neste caso, um vértice está contido em 6 facetas.

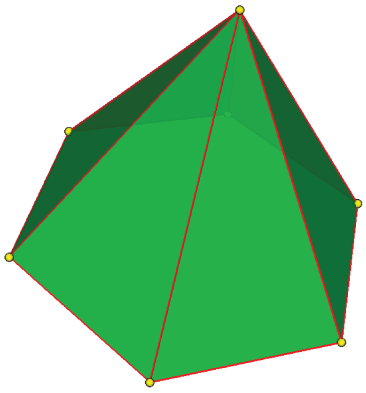


Figura 1.13: Pirâmide no $\mathbb{R}^3$.

Definição 1.26 *Para todo $x_0 \in \mathbb{R}^n$ e $\lambda \in \mathbb{R}_+$, um* ***raio*** *é uma coleção de pontos da forma*

$$r = \{x_0 + \lambda d : \lambda \in \mathbb{R}_+\}$$

onde d é um vetor não nulo do $\mathbb{R}^n$ e x_0 denomina-se ***vértice do raio*** *e d é denominada* ***direção do raio****.*

Definição 1.27 *Dado um conjunto convexo, um vetor não nulo d é chamado uma* ***direção*** *do conjunto, se para cada x_0 do conjunto, o raio $r = \{x_0 + \lambda d : \lambda \geq 0\}$ também pertence ao conjunto.*

É evidente que se o conjunto é limitado, não tem direções.

Definição 1.28 *Uma* ***direção extrema*** *de um conjunto convexo é uma direção do conjunto que não pode ser representada como uma combinação positiva de duas direções distintas do conjunto.*

Dois vetores d_1 e d_2 são ditos distintos, ou não equivalentes, se d_1 não puder ser representado como um múltiplo positivo de d_2. Na Figura 1.14 temos 2 direções extremas d_1 e d_2. Qualquer outra direção do conjunto, como não é múltipla de d_1 e d_2, pode ser representada como $\lambda_1 d_1 + \lambda_2 d_2$ onde $\lambda_1, \lambda_2 > 0$.

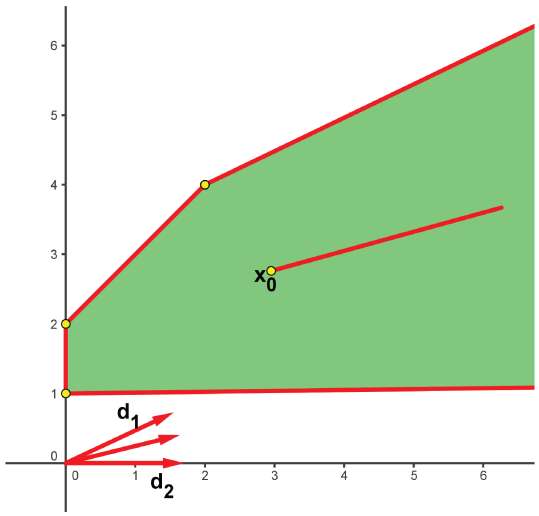


Figura 1.14: Direções de um conjunto convexo.

Qualquer raio que está contido no conjunto convexo, e cuja direção é uma direção extrema, é chamado **raio extremo**.

Vamos considerar agora o caso de uma região poliedral ilimitada. Um exemplo é mostrado na Figura 1.15. Podemos ver que a região tem três pontos extremos, x_1, x_2 e x_3, bem como duas direções extremas, d_1 e d_2. Pode-se demonstrar pelo teorema da representação [3], que em geral, cada ponto da região pode ser representado como uma combinação convexa de pontos extremos, somado a uma combinação linear não-negativa das direções extremas.

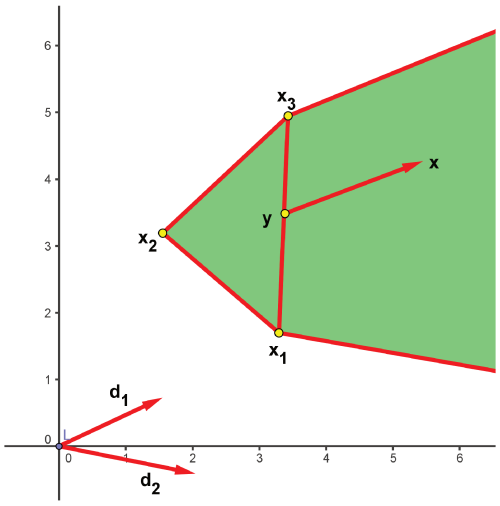


Figura 1.15: Região poliedral ilimitada.

Para ilustrar, consideremos o ponto x na Figura 1.15. O ponto x pode ser representado por y mais um múltiplo positivo da direção extrema d_1.

Notemos que o vetor $(x-y)$ está na direção de d_1. Mas y é uma combinação convexa dos pontos extremos x_1 e x_3, de forma que

$$x = y + \mu d_1 = \lambda x_1 + (1 - \lambda)x_3 + \mu d_1$$

onde $\lambda \in (0, 1)$ e $\mu > 0$.

Neste trabalho estamos interessados na representação gráfica de poliedros definidos por um conjunto de desigualdades $(Ax \leq b)$. Obviamente a projeção gráfica só é possível em dimensões menores ou iguais a 3. Quando a dimensão do poliedro for maior do que 3, podemos projetá-lo num espaço de dimensão menor, até que sua "sombra" possa ser desenhada no $\mathbb{R}^3$. Faremos isso utilizando o método de Fourier-Motzkin que será descrito no Capítulo 2.

O seguinte teorema é importante neste sentido, pois garante que a projeção de um poliedro continua sendo um poliedro.

Teorema 1.2 *Considere a projeção* $\pi : \mathbb{R}^n \to \mathbb{R}^{n-1}$ *dada por*

$$\pi(x_1, x_2, ..., x_n) = (x_2, ..., x_n).$$

Se $P \subset \mathbb{R}^n$ *é um poliedro, então* $\pi(P) \subset \mathbb{R}^{n-1}$ *é um poliedro.*

Demonstração: Suponhamos que P seja o conjunto solução do sistema

$$\begin{cases} a_{11}x_1 & + & a_{12}x_2 & + & ... & + & a_{1n}x_n & \leq & b_1 \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & ... & + & a_{mn}x_n & \leq & b_m \end{cases}.$$

Vamos dividir as m inequações em:

$$G = \{i | a_{i1} > 0\}$$

$$Z = \{i | a_{i1} = 0\}$$

$$L = \{i | a_{i1} < 0\}$$

A inequação i se reduz a

$$x_1 \leq a'_{i2}x_2 + ... + a'_{in}x_n + b'_i$$

se $i \in G$ e a

$$a'_{j2}x_2 + ... + a'_{jn}x_n + b'_j \leq x_1$$

se $j \in L$, onde $a'_{ik} = \dfrac{-a_{ik}}{a_{i1}}$ e $b'_i = \dfrac{b_i}{a_{i1}}$ para $k = 2, ..., n$. Assim, as inequações em G e L resultam na seguinte desigualdade,

$$max(a'_{i2}x_2 + ... + a'_{in}x_n + b'_i | i \in L) \leq x_1 \leq min(a'_{j2}x_2 + ... + a'_{jn}x_n + b'_j | j \in G).$$

Agora $(x_2, ..., x_n) \in \pi(P)$ se, e somente se, existe x_1 tal que $(x_1, ..., x_n) \in P$. Isto acontece se, e somente se, $(x_2, ..., x_n)$ satisfaz as inequações em Z e

$$max(a'_{i2}x_2 + ... + a'_{in}x_n + b'_i | i \in L) \leq min(a'_{j2}x_2 + ... + a'_{jn}x_n + b'_j | j \in G).$$

Essa desigualdade se expande para o sistema de $|L|$ [1] e $|G|$ inequações em $x_2, ..., x_n$ consistindo de

$$a'_{i2}x_2 + ... + a'_{in}x_n + b'_i \leq a'_{j2}x_2 + ... + a'_{jn}x_n + b'_j$$

ou melhor

$$(a'_{i2} - a'_{j2})x_2 + (a'_{in} - a'_{jn})x_n \leq b'_j - b'_i$$

onde $i \in L$ e $j \in G$. Adicionando as inequações em Z (onde x_1 não está presente) nós vemos que $\pi(P)$ é o conjunto de soluções para o sistema de $|L||G| + |Z|$ inequações lineares, isto é, $\pi(P)$ é um poliedro.

Definição 1.29 *Um **cone convexo** C é um conjunto convexo com a propriedade adicional que se $x \in C$, então $\lambda x \in C$, $\forall \lambda \geq 0$.*

Notemos que um cone convexo sempre contém a origem, basta fazer $\lambda = 0$, e também, dado qualquer ponto $x \in C$, o raio $r = \{\lambda x : \lambda \geq 0\} \in C$.

Portanto, um cone convexo é um conjunto convexo que consiste de raios emanados desde a origem. Nas Figuras 1.16 e 1.17 temos exemplos de cones convexos.

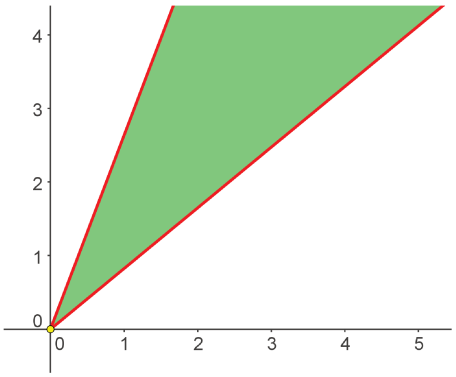


Figura 1.16: Cone convexo no $\mathbb{R}^2$.

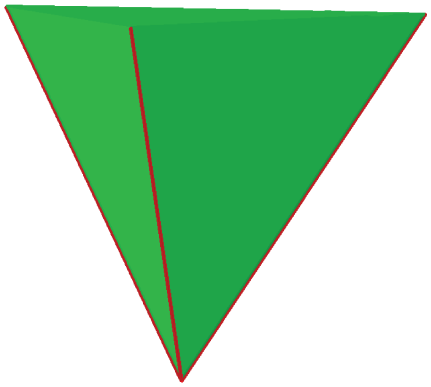


Figura 1.17: Cone convexo no $\mathbb{R}^3$.

Assim, um cone convexo é formado por raios, então ele pode ser inteiramente caracterizado por suas direções. Na verdade, não são necessárias todas as direções, uma direção não extrema pode ser representada como uma combinação positiva de direções extremas.

[1] Se um conjunto A tem n elementos, onde n é um número natural, então diz-se que a cardinalidade de A, denotada por $|A|$ é n.

Dado um conjunto de vetores $a_1, a_2, ..., a_k$, podemos formar o cone convexo C, gerado por estes vetores. Este cone consiste de todas as combinações não-negativas de $a_1, a_2, ..., a_k$, isto é,

$$C = \left\{ \sum_{j=1}^{k} \lambda_j a_j : \quad \lambda_j \geq 0, \quad \forall j = 1, 2, ..., k \right\}.$$

Como um exemplo, consideremos o cone convexo cujas direções extremas são $(1, 1)$ e $(0, 1)$, logo o cone convexo $C = \{(x_1, x_2) : x_1 \geq 0, x_1 \leq x_2\}$. A Figura 1.18 mostra o cone convexo gerado pelos pontos $(1, 1)$ e $(0, 1)$.

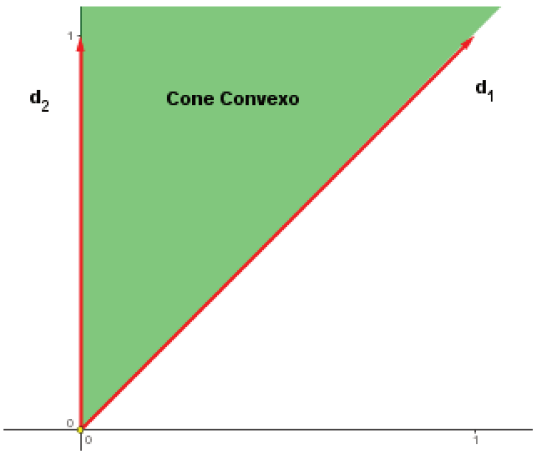


Figura 1.18: Caracterização de cones convexos em termos de direções extremas.

O Teorema de Minkowski-Weyl afirma que cada poliedro é finitamente gerado e cada conjunto finitamente gerado é um poliedro. Mais precisamente, para dois subconjuntos P e Q de $\mathbb{R}^n$, $P + Q$ denota a soma de Minkowski de P e Q:

$$P + Q = \{p + q : p \in P \text{ e } q \in Q\}.$$

Teorema 1.3 *O Teorema de Minkowski-Weyl [12].*

Para um subconjunto P de $\mathbb{R}^n$, as seguintes afirmações são equivalentes:

1. P é um poliedro, isto é, para alguma matriz real (finita) A e algum vetor real b, $P = \{x : Ax \leq b\}$;

2. Existem um número finito de vetores reais $v_1, v_2, ..., v_r$ e raios $r_1, r_2, ..., r_s$ em $\mathbb{R}^n$ tais que $P = conv(v_1, v_2, \ldots, v_r) + (r_1, r_2, \ldots, r_s)$.

Definição 1.30 *O problema da enumeração dos vértices e das faces [12].*

Quando um poliedro P em $\mathbb{R}^n$ tem pelo menos um ponto extremo e possui a dimensão do espaço, as representações (1) e (2) no Teorema de Minkowski-Weyl são únicas de cada inequação e raio r_j.

Sob estas condições de regularidade, as conversões entre as representações dos hiperplanos e as representações dos vértices são problemas fundamentais bem conhecidos. A transformação (1) para (2) é conhecida como a enumeração dos vértices e, a outra, de (2) para (1), é conhecida como a enumeração das faces.

Há problemas fáceis (não-degenerados) e difíceis (degenerados). Por simplicidade, vamos assumir que P é um politopo. A enumeração dos vértices é chamada **não-degenerada** se não houver nenhum ponto $x \in \mathbb{R}^n$ que satisfaça $n+1$ inequações com igualdade, e **degenerada**, caso contrário. A enumeração das faces é chamada não-degenerada se não houver $n+1$ pontos que estão em um mesmo hiperplano, e degenerada, caso contrário.

Neste trabalho, nós iremos tratar dos problemas fáceis (não-degenerados), enumerando os vértices, e em seguida, encontrando as faces. Para isso, no Capítulo 3, apresentaremos alguns algoritmos eficientes seguidos de algumas observações.

Nosso objetivo é enumerar os vértices, encontrar as faces e representar graficamente o politopo, portanto iremos trabalhar em $\mathbb{R}$, $\mathbb{R}^2$ e no $\mathbb{R}^3$. Mas isso não nos impede de termos um conjunto de desigualdades no $\mathbb{R}^n$ com $n > 3$. Caso isso aconteça, vamos utilizar um método para reduzir esse conjunto de desigualdades para $\mathbb{R}^3$ ou $\mathbb{R}^2$ ou $\mathbb{R}$. O método que vamos utilizar para isso é a eliminação de Fourier-Motzkin que será apresentado no capítulo a seguir.

Capítulo 2

Método de Eliminação de Fourier-Motzkin

2.1 Introdução

Será apresentado nesse capítulo um algoritmo matemático para eliminar variáveis de um sistema de inequações lineares, chamado método de eliminação de Fourier-Motzkin. Esse algoritmo nos permite reduzir um sistema do $\mathbb{R}^n$ até o $\mathbb{R}^3$ ou $\mathbb{R}^2$ ou $\mathbb{R}$ e projetar sua região de solução. O algoritmo consiste em criar um outro sistema do mesmo tipo de tal forma que ambos os sistemas tenham as mesmas soluções sobre as demais variáveis. Por esse algoritmo também poderemos decidir se o sistema original tem ou não soluções. Para isso basta eliminar todas as variáveis, obtendo um sistema de desigualdades constantes. Para a apresentação desse algoritmo, iremos analisar três exemplos de forma a construir e formalizar o método. Em seguida, trabalharemos o método por meio de operações com matrizes, o que facilitará sua abordagem computacional.

Antes de formalizar o método de Fourier-Motzkin, vamos estudar alguns exemplos para termos uma ideia intuitiva do seu funcionamento.

Exemplo 2.1

Vamos considerar o seguinte sistema de inequações lineares:

$$\begin{cases} \dfrac{-3x}{4} + y & \leq & \dfrac{5}{4} \\ \dfrac{x}{6} + y & \leq & 7 \end{cases}. \tag{2.1}$$

A região de solução desse sistema é apresentada na Figura 2.1.

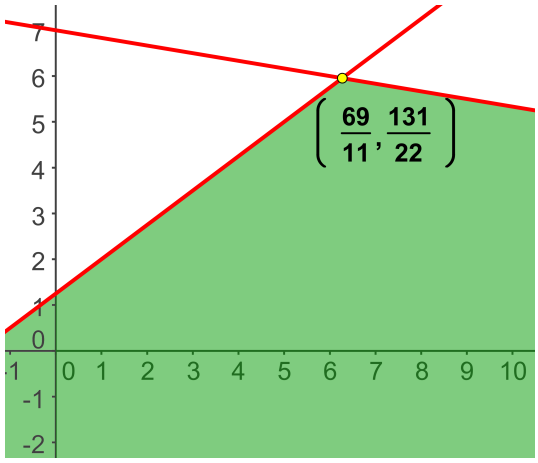


Figura 2.1: Região de solução de 2.1.

Podemos rearranjar o sistema (2.1), deixando os coeficientes de x valendo ± 1, multiplicando ambos os lados de cada inequação por um número real positivo. Esse procedimento não irá alterar a região de solução do sistema. Fazendo isso, obtemos o seguinte resultado:

$$\begin{cases} \left(\dfrac{4}{3}\right). & \dfrac{-3x}{4} + y & \leq & \dfrac{5}{4} & .\left(\dfrac{4}{3}\right) \\ (6). & \dfrac{x}{6} + y & \leq & 7 & .(6) \end{cases} \Longrightarrow \begin{cases} -x + \dfrac{4y}{3} & \leq & \dfrac{5}{3} \\ x + 6y & \leq & 42 \end{cases}.$$

Vamos isolar a incógnita x nas duas inequações, assim teremos:

$$\begin{cases} x & \geq & \dfrac{4y}{3} - \dfrac{5}{3} \\ x & \leq & -6y + 42 \end{cases}.$$

Podemos reescrever esse resultado da seguinte forma:

$$\frac{4y}{3} - \frac{5}{3} \leq x \leq -6y + 42. \tag{2.2}$$

De (2.2) temos que,

$$\frac{4y}{3} - \frac{5}{3} \leq -6y + 42,$$

ou seja,

$$4y - 5 \leq -18y + 126,$$

de onde,

$$22y \leq 131.$$

Portanto os valores de y na solução do sistema de inequações (2.1) deverão satisfazer à seguinte restrição

$$y \leq \frac{131}{22} \cong 5,946.$$

Note que a variável x do sistema (2.1) foi eliminada.

Fazendo $y = \dfrac{131}{22}$ em (2.2), temos:

$$4.\left(\frac{131}{22}\right).\left(\frac{1}{3}\right) - \frac{5}{3} \leq x \leq -6.\left(\frac{131}{22}\right) + 42$$

$$\Downarrow$$

$$\frac{524 - 110}{66} \leq x \leq \frac{-786 + 924}{22}.$$

Logo,

$$y = \frac{131}{22} \quad \Rightarrow \quad \frac{69}{11} \leq x \leq \frac{69}{11}\ .$$

Portanto, quando y assume o máximo valor factível, o valor correspondente de x é $\dfrac{69}{11}$, ou seja, $\left(\dfrac{69}{11}, \dfrac{131}{22}\right)$ é um ponto extremo da região determinada por (2.1).

Podemos observar que o ponto $\left(\dfrac{69}{11}, \dfrac{131}{22}\right)$ é exatamente o ponto extremo do sistema de inequações (2.1) do Exemplo 1.

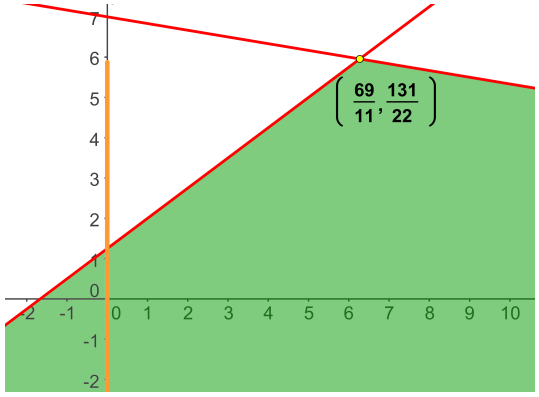


Figura 2.2: Projeção da região de solução do sistema (2.1) sobre o eixo y.

Exemplo 2.2

Vamos agora tomar o sistema de inequações (2.1) e acrescentar duas novas inequações, ficando da seguinte forma:

$$\begin{cases} \dfrac{-3x}{4} + y & \leq & \dfrac{5}{4} \\ \dfrac{x}{6} + y & \leq & 7 \\ \dfrac{7x}{2} - y & \leq & \dfrac{59}{2} \\ -3x - y & \leq & -5 \end{cases} . \tag{2.3}$$

A região de solução de (2.3) está representada na Figura 2.3.

Seguindo os mesmos passos do Exemplo 2.1, vamos multiplicar cada inequação por um número real positivo, deixando o coeficiente de x valendo ± 1. Feito isso, vamos rearranjar o sistema (2.3) colocando as inequações com coeficientes de x negativo primeiro

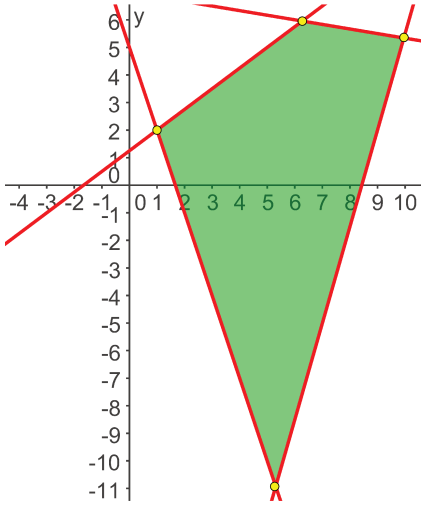


Figura 2.3: Região de solução de (2.3).

e depois as de coeficientes positivos (poderia ser o inverso). Em seguida isolamos x em cada uma das inequações obtendo os resultados:

$$\begin{cases} \frac{-3x}{4} + y \leq \frac{5}{4} & .(\frac{4}{3}) \\ \frac{x}{6} + y \leq 7 & .(6) \\ \frac{7x}{2} - y \leq \frac{59}{2} & .(\frac{2}{7}) \\ -3x - y \leq -5 & .(\frac{1}{3}) \end{cases} \Longrightarrow \begin{cases} -x + \frac{4y}{3} \leq \frac{5}{3} \\ x + 6y \leq 42 \\ x - \frac{2y}{7} \leq \frac{59}{7} \\ -x - \frac{y}{3} \leq \frac{-5}{3} \end{cases} \Longrightarrow \begin{cases} -x + \frac{4y}{3} \leq \frac{5}{3} \\ -x - \frac{y}{3} \leq \frac{-5}{3} \\ x + 6y \leq 42 \\ x - \frac{2y}{7} \leq \frac{59}{7} \end{cases} \Longrightarrow$$

$$\Longrightarrow \begin{cases} x \geq \frac{4y}{3} - \frac{5}{3} \\ x \geq \frac{-y}{3} + \frac{5}{3} \\ x \leq -6y + 42 \\ x \leq \frac{2y}{7} + \frac{59}{7} \end{cases}.$$

Podemos reescrever esse último sistema da seguinte forma:

$$\max\left\{\frac{4y}{3}-\frac{5}{3},\frac{-y}{3}+\frac{5}{3}\right\} \leq x \leq \min\left\{-6y+42,\frac{2y}{7}+\frac{59}{7}\right\}. \tag{2.4}$$

Se desejarmos eliminar a variável x em (2.4), devemos considerar

$$\begin{cases} \frac{4y}{3}-\frac{5}{3} \leq -6y+42 \\ \frac{4y}{3}-\frac{5}{3} \leq \frac{2y}{7}+\frac{59}{7} \\ \frac{-y}{3}+\frac{5}{3} \leq -6y+42 \\ \frac{-y}{3}+\frac{5}{3} \leq \frac{2y}{7}+\frac{59}{7} \end{cases} \Longrightarrow \begin{cases} 22y \leq 131 \\ 22y \leq 212 \\ 17y \leq 121 \\ -13y \leq 142 \end{cases} \Longrightarrow \begin{cases} y \leq \frac{131}{22} \cong 5,955 \\ y \leq \frac{212}{22} \cong 9,636 \\ y \leq \frac{121}{17} \cong 7,118 \\ y \geq \frac{-142}{13} \cong -10,923 \end{cases}.$$

De onde concluímos que os valores de y na solução do sistema de inequação (2.3) devem satisfazer:

$$\frac{-142}{13} \leq y \leq \frac{131}{22}.$$

Se substituirmos os valores extremos de y em (2.4) obtemos, para $y = \frac{-142}{13}$

$$\max\left\{\frac{4}{3}.\left(\frac{-142}{13}\right)-\frac{5}{3},\left(\frac{-1}{3}\right).\left(\frac{-142}{13}\right)+\frac{5}{3}\right\} \leq x \leq$$
$$\min\left\{-6.\left(\frac{-142}{13}\right)+42,\left(\frac{2}{7}\right).\left(\frac{-142}{13}\right)-\left(\frac{59}{7}\right)\right\}.$$

Efetuando as operações temos,

$$\max\left\{\frac{-211}{13},\frac{69}{13}\right\} \leq x \leq \min\left\{\frac{1398}{13},\frac{69}{13}\right\}.$$

Logo,

$$\frac{69}{13} \leq x \leq \frac{69}{13}.$$

Portanto, para $y = \frac{-142}{13}$, temos, $x = \frac{69}{13}$.

Agora vamos substituir $y = \frac{131}{22}$ em (2.4).

$$\max\left\{\frac{4}{3}.\left(\frac{131}{22}\right)-\frac{5}{3},\left(\frac{-1}{3}\right).\left(\frac{131}{22}\right)+\frac{5}{3}\right\}\leq x\leq$$
$$\min\left\{-6.\left(\frac{131}{22}\right)+42,\left(\frac{2}{7}\right).\left(\frac{131}{22}\right)-\left(\frac{59}{7}\right)\right\}.$$

Efetuando as operações temos,

$$\max\left\{\frac{69}{11},\frac{-7}{22}\right\}\leq x\leq\min\left\{\frac{69}{11},\frac{780}{77}\right\}.$$

Logo,

$$\frac{69}{11}\leq x\leq\frac{69}{11}.$$

Portanto, para $y=\dfrac{131}{22}$, temos, $x=\dfrac{69}{11}$ e, consequentemente, dois vértices da região de solução de (2.3), como vemos na Figura 2.4.

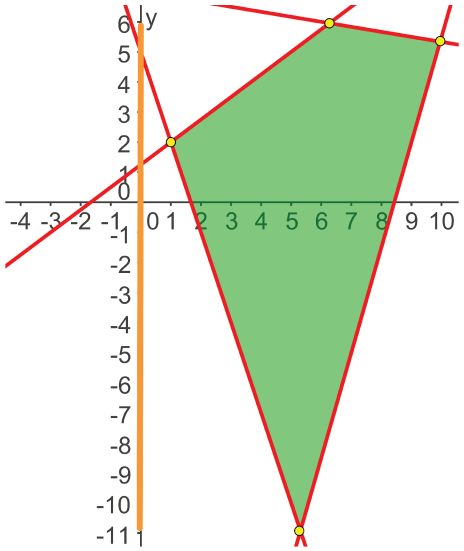


Figura 2.4: Projeção da região de solução do Exemplo 2.2 sobre o eixo y.

Exemplo 2.3

Nos Exemplos 2.1 e 2.2 todas as inequações envolviam as duas variáveis. Vamos acrescentar ao sistema (2.3) três inequações, sendo duas dessas envolvendo apenas a variável y. Assim, consideremos o seguinte sistema de inequações lineares:

$$\begin{cases} \dfrac{-3x}{4} + y & \leq & \dfrac{5}{4} \\ \dfrac{x}{6} + y & \leq & 7 \\ \dfrac{7x}{2} - y & \leq & \dfrac{59}{2} \\ -3x - y & \leq & -5 \\ -y & \leq & 3 \\ \dfrac{3x}{4} + y & \leq & \dfrac{35}{4} \\ -y & \leq & 5 \end{cases} . \tag{2.5}$$

Seguindo os mesmos passos dos exemplos anteriores, vamos multiplicar cada inequação por um número real positivo, deixando o coeficiente de x valendo 0 ou ± 1, depois rearranjamos o sistema (2.5) colocando as inequações com coeficientes de x negativo primeiro, depois as de coeficientes positivos (poderia ser o inverso) e em seguida as de coeficientes nulos. Depois disso, isolamos a variável x nas inequações do sistemas e obtemos o seguinte resultado:

$$\begin{cases} x & \geq & \dfrac{4y}{3} - \dfrac{5}{3} \\ x & \geq & \dfrac{-y}{3} + \dfrac{5}{3} \\ x & \leq & -6y + 42 \\ x & \leq & \dfrac{2y}{7} + \dfrac{59}{7} \\ x & \leq & \dfrac{-4y}{3} + \dfrac{35}{3} \\ y & \geq & -3 \\ y & \geq & -5 \end{cases} . \tag{2.6}$$

Podemos notar que a inequação $-y \leq 5$ é redundante, ou seja, pode ser eliminada do sistema de inequações (2.5) que não vai alterar sua região de solução, conforme podemos observar na Figura 2.5.

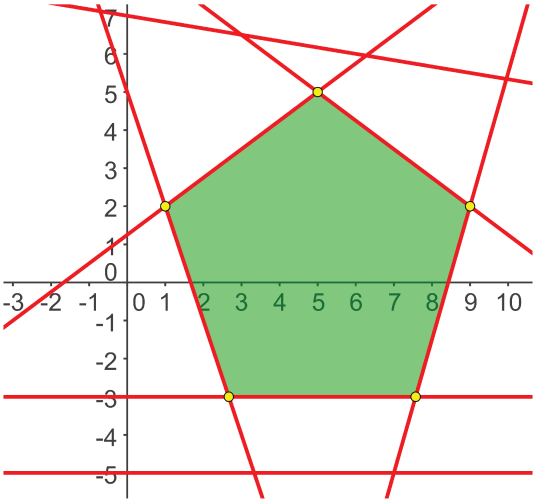


Figura 2.5: Região de solução de (2.5).

Vamos reescrever (2.6) da seguinte maneira:

$$\max\left\{\frac{4y}{3}-\frac{5}{3}, \frac{-y}{3}+\frac{5}{3}\right\} \leq x \leq \min\left\{-6y+42, \frac{2y}{7}+\frac{59}{7}, \frac{-4y}{3}+\frac{35}{3}\right\} \tag{2.7}$$

Logo,

$$\begin{cases} \frac{4y}{3} - \frac{5}{3} \leq -6y + 42 \\ \frac{4y}{3} - \frac{5}{3} \leq \frac{2y}{7} + \frac{59}{7} \\ \frac{4y}{3} - \frac{5}{3} \leq \frac{-4y}{3} + \frac{35}{3} \\ \frac{-y}{3} + \frac{5}{3} \leq -6y + 42 \\ \frac{-y}{3} + \frac{5}{3} \leq \frac{2y}{7} - \frac{59}{7} \\ \frac{-y}{3} + \frac{5}{3} \leq \frac{-4y}{3} + \frac{35}{3} \\ y \geq -3 \end{cases} \Longrightarrow \begin{cases} y \leq \frac{131}{22} \\ y \leq \frac{212}{22} \\ y \leq 5 \\ y \leq \frac{121}{17} \\ y \geq \frac{-142}{13} \\ y \leq 6 \\ y \geq -3 \end{cases}.$$

Portanto, os valores de y para a solução do sistema de inequações (2.5) deve satisfazer à seguinte restrição:

$$-3 \leq y \leq 5. \tag{2.8}$$

Fazendo $y = 5$ em (2.7) temos:

$$\max\left\{\frac{4.(5)}{3} - \frac{5}{3}, \frac{-5}{3} + \frac{5}{3}\right\} \leq x \leq \min\left\{-6.(5) + 42, \frac{2.(5)}{7} + \frac{59}{7}, \frac{-4.(5)}{3} + \frac{35}{3}\right\}.$$

Logo, $\max\{5, 0\} \leq x \leq \min\left\{12, \frac{69}{7}, 5\right\}$, ou seja, $x = 5$.

Para $y = -3$,

$\max\left\{\frac{-17}{3}, \frac{8}{3}\right\} \leq x \leq \min\left\{60, \frac{53}{7}, \frac{47}{3}\right\}$, ou seja, $\frac{8}{3} \leq x \leq \frac{53}{7}$.

Portanto, obtemos três vértices pertencentes a região de solução do sistema (2.5), $(5, 5)$, $\left(\frac{8}{3}, -3\right)$ e $\left(\frac{53}{7}, -3\right)$.

Na Figura 2.6 podemos ver à projeção da região de solução no eixo y.

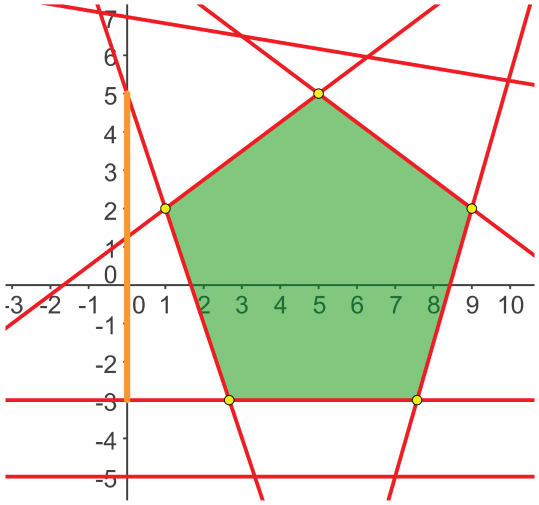


Figura 2.6: Projeção das regiões de solução do Exemplo 2.3 sobre o eixo y.

Vejamos algumas observações a respeito desses três exemplos resolvidos:

1. Nos três exemplos apresentados eliminamos a variável x do sistema de inequações, obtendo um novo sistema com uma variável a menos. Geometricamente isso equivale a projetar a região de solução do sistema original num espaço de dimensão menor. Neste caso, de $\mathbb{R}^2$ para $\mathbb{R}$.

2. No Exemplo 2.3, conseguimos identificar uma equação redundante.

3. Também foi possível encontrar alguns pontos extremos do conjunto solução, o que pode ser útil para encontrar um ponto interior, como veremos no Capítulo 3.

O que fizemos nesses três exemplos particulares foi a aplicação do método de eliminação de Fourier-Motzkin, que pode ser generalizado para o $\mathbb{R}^n$, como veremos na próxima seção.

2.2 Método de eliminação de Fourier-Motzkin

A eliminação de Fourier-Motzkin é um método que nos permite eliminar variáveis de um sistema de inequações, reduzindo-o a um sistema em espaço de dimensão menor.

Seja o sistema de inequações lineares,

$$Ax \leq b$$

onde $A \in \mathbb{R}^{m\times n}$ e $b \in \mathbb{R}^n$.

Vamos escrevê-lo da seguinte forma:

$$\left\{\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & ... & + & a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & ... & + & a_{2n}x_n & \leq & b_2 \\ \vdots & & \vdots & & \vdots & & \ddots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & ... & + & a_{mn}x_n & \leq & b_m \end{array}\right. . \tag{2.9}$$

Digamos que queremos eliminar x_1 do sistema (2.9).

Para cada i, onde $a_{i1} \neq 0$, podemos multiplicar cada uma das inequações por $\frac{1}{|a_{i1}|}$, deixando como valor do coeficiente de x_1, 0 ou ± 1.

Temos, então

$$\left\{\begin{array}{ccccccccc} -x_1 & + & a'_{i2}x_2 & + & ... & + & a'_{in}x_n & \leq & b'_i \quad (i \in I^-) \\ x_1 & + & a'_{i2}x_2 & + & ... & + & a'_{in}x_n & \leq & b'_i \quad (i \in I^+) \\ & + & a'_{i2}x_2 & + & ... & + & a'_{in}x_n & \leq & b'_i \quad (i \in I^0) \end{array}\right. . \tag{2.10}$$

Onde $I^- = (i : a_{i1} < 0)$, ou seja, as inequações onde os coeficientes de x_1 são negativos, $I^+ = (i : a_{i1} > 0)$, ou seja, as inequações onde os coeficientes de x_1 são positivos e $I^0 = (i : a_{i1} = 0)$, ou seja, as inequações onde os coeficientes de x_1 são nulos, $a'_{ij} = \frac{a_{ij}}{|a_{i1}|}$ e $b'_i = \frac{b_i}{|a_{i1}|}$.

Assim, o conjunto de índices da linha $I = \{1, 2, ..., m\}$ é particionado nos subconjuntos I^-, I^+ e I^0, sendo que algum destes pode ser vazio. Daqui resulta que $x_1, x_2, x_3, ..., x_n$ é uma solução do sistema original (2.9) se, e somente se, $x_1, x_2, x_3, ..., x_n$ satisfazem as seguintes restrições:

$$\max\{(a'_{k2}x_2 + ... + a'_{kn}x_n) - b'_k\} \leq x_1 \leq \min\{b'_i - (a'_{i2}x_2 + ... + a'_{in}x_n)\} \tag{2.11}$$

onde $i \in I^+, k \in I^-$ e

$$a'_{i2}x_2 + a'_{i3}x_3 + ... + a'_{in}x_n \leq b'_i \tag{2.12}$$

para o coeficiente de x_1 valendo zero, com $i \in I^0$.

A inequação (2.11) diz que x_1 encontra-se em um certo intervalo, que é determinado por $x_2, x_3, ..., x_n$. Por essa inequação podemos escrever:

$$\max\{(a'_{k2}x_2 + ... + a'_{kn}x_n) - b'_k\} \leq \min\{b'_i - (a'_{i2}x_2 + ... + a'_{in}x_n)\}, \tag{2.13}$$

que podemos resolver combinando cada inequação do primeiro conjunto de restrições com cada inequação do segundo conjunto de restrições.

Juntando os resultados obtidos resolvendo (2.13) com (2.12), obtemos uma projeção P no espaço das variáveis $x_2, x_3, ..., x_n$.

Pode-se então proceder da mesma forma e eliminar x_2, x_3, etc.

Observações [9]:

- Se I^- ou I^+ for vazio, o primeiro conjunto de restrições de (2.11) desaparece e o sistema será inconsistente ou terá uma região ilimitada, pois a variável x_1 não terá limite superior ($I^- = \oslash$) ou não terá limite inferior ($I^+ = \oslash$).

  Exemplo: Vamos considerar um sistema cujo conjunto I^+ é vazio:

$$\begin{cases} -x+y \leq 1 \\ -x-y \leq 5 \\ y \leq -1 \end{cases} \Rightarrow \begin{cases} x \geq y-1 \\ x \geq -y-5 \\ y \leq -1 \end{cases}$$

  Assim, $x \geq \max(y-1, -y-5)$ e $y \leq -1$, ou seja, x não possui limite superior, logo a região de solução é ilimitada, conforme podemos observar na Figura 2.7.

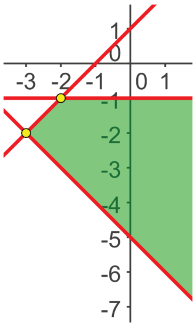


Figura 2.7: Região de solução do sistema.

- Se o conjunto I^0 for vazio e apenas um dos conjuntos I^- ou I^+ for não vazio, não podemos eliminar a primeira variável. Uma alternativa é reordenar o sistema e iniciar a eliminação por outra variável.

  Exemplo: Considere o seguinte sistema

$$\begin{cases} -x+y \leq 8 \\ -x-y \leq 2 \end{cases} \Rightarrow \begin{cases} x \geq y-8 \\ x \geq -y-2 \end{cases}$$

$x \geq \max\{y-8, -y-2\}$. Podemos observar que não há limites para a variável y, portanto, se eliminarmos a variável x, a projeção da região de solução do sistema será todo o eixo y, conforme a Figura 2.8.

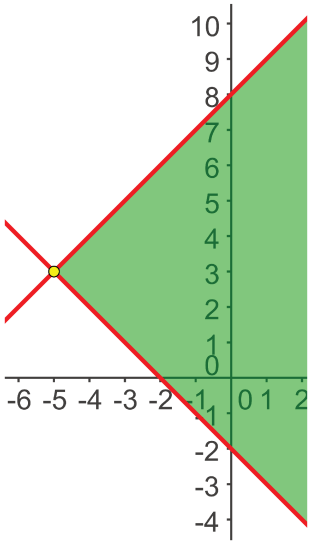


Figura 2.8: Região de solução do sistema.

- O método de eliminação de Fourier-Motzkin nos permite concluir se o sistema é inconsistente ou não. Considere o seguinte sistema de inequações:

$$\begin{cases} x+y \leq 8 \\ x-y \leq 2 \\ -2x+y \leq -15 \end{cases} \Rightarrow \begin{cases} x \leq -y+8 \\ x \leq y+2 \\ x \geq \dfrac{y}{2}+\dfrac{15}{2} \end{cases}$$

$$\max\left\{\frac{y}{2}+\frac{15}{2}\right\} \leq x \leq \min\{-y+8, y+2\}.$$

Eliminando x da restrição acima, obtemos $11 \leq y \leq \dfrac{1}{3}$, que é inconsistente. Logo, o sistema não admite solução.

- Um problema desse método é que o número de restrições pode crescer exponencialmente mais rápido do que as variáveis são eliminadas, o que pode tornar cara sua aplicação em sistemas grandes.

Na próxima seção vamos apresentar uma versão matricial do método de Fourier-Motzkin com o intuito de facilitar sua abordagem computacional.

2.2.1 Eliminação de Fourier-Motzkin através de operações com matrizes

Podemos utilizar a notação matricial para apresentar o método de Fourier-Motzkin de outro ponto de vista [9]. A seguir descrevemos, de forma resumida, esta abordagem.

Considere o sistema linear (2.9) e sua matriz de coeficientes $A \in \mathbb{R}^{m \times n}$. Seja A' a matriz dos coeficientes de um novo sistema linear (2.10), ainda com todas as variáveis $x_1, x_2, x_3, ..., x_n$, mas cujos coeficientes de x_1 são ± 1 ou 0. O novo sistema de inequações lineares pode ser representado por

$$A'x \leq b'.$$

Vamos escrevê-lo da seguinte forma:

$$\begin{bmatrix} a'_{11} & a'_{12} & ... & a'_{1n} \\ a'_{21} & a'_{22} & ... & a'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a'_{m1} & a'_{m2} & ... & a'_{mn} \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_m \end{bmatrix} .$$

Vamos criar uma matriz positiva $C \in R^{m' \times m}$, onde $m' = |I^+| \times |I^-| + |I^0|$, ou seja, o número de linhas da matriz C é o produto entre o número de coeficientes $a'_{i1} > 0$ e $a'_{i1} < 0$ mais o número de coeficientes $a'_{i1} = 0$, e o número de colunas é igual ao número de linhas da matriz A'.

A matriz C será uma matriz binária (seus elementos serão 0 ou 1) de forma a representar as combinações entre o conjunto das inequações que satisfazem $a'_{i1} < 0$ com aquelas que satisfazem $a'_{i1} > 0$. Vamos supor que as linhas $a'_1, a'_2, \ldots, a'_p$ são aquelas onde $a'_{i1} < 0$, as linhas $a'_{p+1}, a'_{p+1}, \ldots, a'_{p+r}$ são aquelas onde $a'_{i1} > 0$ e $a'_{p+r+1}, \cdots, a'_m$ as linhas que satisfazem $a'_{i1} = 0$. Neste caso a matriz C terá a forma:

$$C = \left[\begin{array}{cccc|cccc|cccc}
1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
1 & 0 & \dots & 0 & 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 \\
1 & 0 & \dots & 0 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 \\
\hline
\vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & 0 & \dots & 0 \\
\hline
0 & 0 & \dots & 1 & 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & \dots & 1 & 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & \dots & 1 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 \\
\hline
\vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & 0 \\
\hline
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 \\
\vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \dots & \ddots & 0 \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1
\end{array}\right].$$

Com a matriz C construída, agora para eliminarmos a primeira variável basta fazermos as seguintes operações com as matrizes:

$$C.A' \qquad e \qquad C.b'.$$

Assim, construímos a seguinte desigualdade:

$$\begin{bmatrix} 0 & a''_{12} & ... & a''_{1n} \\ 0 & a''_{22} & ... & a''_{2n} \\ \vdots & \vdots & ... & \vdots \\ 0 & a''_{m'2} & ... & a''_{m'n} \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b''_1 \\ b''_2 \\ \vdots \\ b''_{m'} \end{bmatrix}$$

onde a''_{ni} são os resultados da operação $C.A'$ e b''_n são os resultados da operação $C.b'$.

A variável x_1 pode ser eliminada deletando a primeira coluna da matriz $C.A'$.

Podemos então proceder da mesma forma e eliminar x_2, x_3, etc.

O processo ficará ilustrado e mais compreensível nos exemplos a seguir.

Exemplo 2.4

Vamos considerar o sistema de inequações lineares (2.5), processado de modo que os coeficientes da variável x sejam todos $1, -1$ ou 0:

$$\begin{cases} -x + \frac{4y}{3} & \leq & \frac{5}{3} \\ -x - \frac{y}{3} & \leq & -\frac{5}{3} \\ x + 6y & \leq & 42 \\ x - \frac{2y}{7} & \leq & \frac{59}{7} \\ x + \frac{4y}{3} & \leq & \frac{35}{3} \\ -y & \leq & 3 \\ -y & \leq & 5 \end{cases}.$$

Escrevendo a matriz A, o vetor b e construindo a matriz C, temos

$$A = \begin{bmatrix} -1 & \frac{4}{3} \\ -1 & \frac{-1}{3} \\ 1 & 6 \\ 1 & \frac{-2}{7} \\ 1 & \frac{4}{3} \\ 0 & -1 \\ 0 & -5 \end{bmatrix}, b = \begin{bmatrix} \frac{5}{3} \\ \frac{-5}{3} \\ 42 \\ \frac{59}{7} \\ \frac{35}{3} \\ 3 \\ 5 \end{bmatrix}.$$

A matriz C terá, 7 colunas e $(2 \times 3) + 2 = 8$ linhas. Logo,

$$C = \left[\begin{array}{cc|ccc|cc} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

Fazendo as operações com matrizes, eliminamos x e temos as seguintes restrições:

$$C.A = \begin{bmatrix} 0 & \frac{22}{3} \\ 0 & \frac{22}{21} \\ 0 & \frac{8}{3} \\ 0 & \frac{17}{3} \\ 0 & \frac{-13}{21} \\ 0 & 1 \\ 0 & -1 \\ 0 & -5 \end{bmatrix}, C.b = \begin{bmatrix} \frac{131}{3} \\ \frac{212}{21} \\ \frac{40}{3} \\ \frac{121}{3} \\ \frac{142}{21} \\ 10 \\ 3 \\ 5 \end{bmatrix} \Rightarrow \begin{cases} \frac{22y}{3} & \leq & \frac{131}{3} \\ \frac{22y}{21} & \leq & \frac{212}{21} \\ \frac{8y}{3} & \leq & \frac{40}{3} \\ \frac{17y}{3} & \leq & \frac{121}{3} \\ \frac{-13y}{21} & \leq & \frac{142}{21} \\ y & \leq & 10 \\ -y & \leq & 3 \\ -5y & \leq & 5 \end{cases}.$$

Resolvendo uma a uma encontramos o mesmo resultado obtido em (2.8):

$$-3 \leq y \leq 5$$

Exemplo 2.5

Vejamos agora um exemplo de um sistema de inequações inconsistente. Consideremos o sistema

$$\begin{cases} -x+y & \leq & -1 \\ 7x-y & \leq & -4 \\ x-y & \leq & -3 \\ -x+y & \leq & -3 \end{cases}. \tag{2.14}$$

Multiplicando cada inequação por um número real positivo e rearranjando as linhas temos

$$A = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ 1 & \frac{-1}{7} \\ 1 & -1 \end{bmatrix}, b = \begin{bmatrix} -1 \\ -3 \\ \frac{-4}{7} \\ -3 \end{bmatrix}.$$

A matriz C terá, 4 colunas e $(2 \times 2) + 0 = 4$ linhas. Logo,

$$C = \left[\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right].$$

Fazendo as operações com matrizes, eliminamos x e temos as seguintes restrições:

$$C.A = \begin{bmatrix} 0 & \frac{6}{7} \\ 0 & 0 \\ 0 & \frac{6}{7} \\ 0 & 0 \end{bmatrix}, C.b = \begin{bmatrix} \frac{-11}{7} \\ -4 \\ \frac{-25}{7} \\ -6 \end{bmatrix} \Rightarrow \begin{cases} \frac{6y}{7} & \leq & \frac{-11}{7} \\ 0 & \leq & -4 \\ \frac{6y}{7} & \leq & \frac{-25}{7} \\ 0 & \leq & -6 \end{cases}.$$

Note que neste sistema apareceram inconsistências como $0 \leq -4$ e $0 \leq -6$, o que nos permite concluir que o sistema (2.14) é inconsistente ou infactível.

Exemplo 2.6

Até agora mostramos exemplos no $\mathbb{R}^2$, vamos resolver um sistema de inequações lineares no $\mathbb{R}^3$ e fazer sua interpretação geométrica.

Consideremos o seguinte sistema de inequações lineares:

$$\begin{cases} z & \leq & 4 \\ -z & \leq & -2 \\ 4x+z & \leq & 10 \\ -4x-z & \leq & 0 \\ -2y+z & \leq & 0 \\ 2y-z & \leq & 4 \end{cases}. \tag{2.15}$$

A região de solução do sistema de inequações (2.15) é mostrada na Figura 2.9 abaixo:

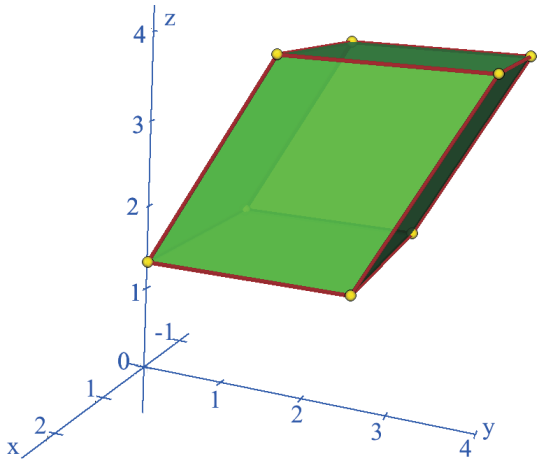


Figura 2.9: Região de solução de (2.15).

Vamos eliminar a variável z, projetando no plano xy a região de solução de (2.15). Para isso, vamos rearranjá-lo da seguinte forma:

$$\begin{cases} -z & \leq & -2 \\ -z-4x & \leq & 0 \\ -z+2y & \leq & 4 \\ z & \leq & 4 \\ z+4x & \leq & 10 \\ z-2y & \leq & 0 \end{cases}.$$

Agora, como os coeficientes de z já são ± 1, escreveremos as matrizes A e b, determinamos C e fazemos as operações das matrizes. Assim temos,

$$A = \begin{bmatrix} -1 & 0 & 0 \\ -1 & -4 & 0 \\ -1 & 0 & 2 \\ 1 & 0 & 0 \\ 1 & 4 & 0 \\ 1 & 0 & -2 \end{bmatrix}, b = \begin{bmatrix} -2 \\ 0 \\ 4 \\ 4 \\ 10 \\ 0 \end{bmatrix}.$$

A matriz C terá, 6 colunas e $(3 \times 3) + 0 = 9$ linhas. Logo,

$$C = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right].$$

Fazendo as operações com matrizes temos:

$$C.A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & -2 \\ 0 & -4 & 0 \\ 0 & 0 & 0 \\ 0 & -4 & -2 \\ 0 & 0 & 2 \\ 0 & 4 & 2 \\ 0 & 0 & 0 \end{bmatrix}, C.b = \begin{bmatrix} 2 \\ 8 \\ -2 \\ 4 \\ 10 \\ 0 \\ 8 \\ 14 \\ 4 \end{bmatrix}.$$

Portanto, eliminamos z e obtemos o seguinte sistema com as variáveis x e y,

$$\begin{cases} 0 & \leq & 2 \\ 4x & \leq & 8 \\ -2y & \leq & -2 \\ -4x & \leq & 4 \\ 0 & \leq & 10 \\ -4x-2y & \leq & 0 \\ 2y & \leq & 8 \\ 4x+2y & \leq & 14 \\ 0 & \leq & 4 \end{cases} . \tag{2.16}$$

Assim, projetamos a região de solução do sistema de inequações (2.15) no plano xy conforme as Figuras 2.10 e 2.11.

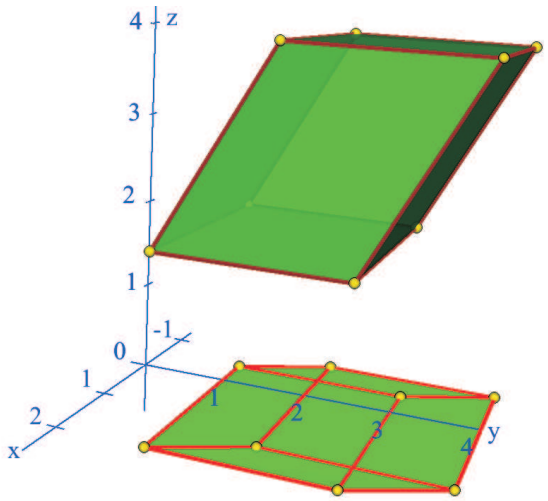


Figura 2.10: Projeção da região de solução de (2.15) no plano xy no $\mathbb{R}^3$.

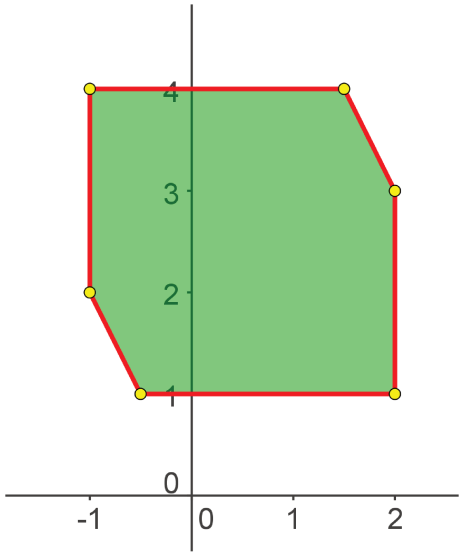


Figura 2.11: Projeção da região de solução de (2.15) no plano xy no $\mathbb{R}^2$.

Agora, no sistema (2.16), procederemos da mesma maneira para eliminar x, e assim projetaremos a região de solução do sistema original (2.15) no eixo y. Então, multiplicando por um número real positivo, obtemos as matrizes A' e b' e definimos C'. Depois fazendo as operações de matrizes obtemos os seguintes resultados:

$$A' = \begin{bmatrix} -1 & \frac{-1}{2} \\ -1 & 0 \\ 1 & 0 \\ 1 & \frac{1}{2} \\ 0 & -2 \\ 0 & 2 \end{bmatrix}, b' = \begin{bmatrix} 0 \\ 1 \\ 2 \\ \frac{7}{2} \\ -2 \\ 8 \end{bmatrix} e, C' = \left[\begin{array}{cc|cc|cc} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

Logo,

$$C'.A' = \begin{bmatrix} 0 & \frac{-1}{2} \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{2} \\ 0 & -2 \\ 0 & 2 \end{bmatrix}, C'.b' = \begin{bmatrix} 2 \\ \frac{7}{2} \\ 3 \\ \frac{9}{2} \\ -2 \\ 8 \end{bmatrix}.$$

Assim temos as seguintes restrições em y.

$$\begin{cases} \frac{-y}{2} & \leq & 2 \\ 0 & \leq & \frac{7}{2} \\ 0 & \leq & 3 \\ \frac{y}{2} & \leq & \frac{9}{2} \\ -3y & \leq & -2 \\ 2y & \leq & 8 \end{cases}.$$

Resolvendo cada uma das inequações acima, obtemos:

$$1 \leq y \leq 4.$$

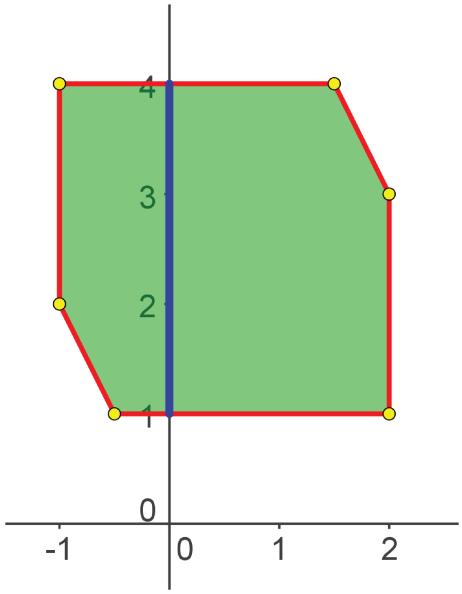


Figura 2.12: Projeção da região de solução de (2.15) no eixo y.

Nos Capítulos 3 e 4 vamos utilizar uma implementação computacional do método de Fourier-Motzkin na linguagem do software Octave, que é compatível com o MatLab. O código utilizado foi implementado por Sebastian Siegel e está disponível no site oficial do MatLab.

A implementação de Sebastian Siegel utiliza, além do método de Fourier-Motzkin, uma rotina para eliminação de inequações redundantes. Como veremos no Capítulo 4 a rotina *reduce_rows* de Sebastian Siegel não elimina todas as redundâncias. Em uma nova implementação, propomos uma maneira de eliminar as redundâncias, que será apresentada no próximo capítulo.

Reduzindo um sistema de inequações lineares do $\mathbb{R}^n$ para o $\mathbb{R}^3$, $\mathbb{R}^2$ ou até para o $\mathbb{R}$, podemos representar graficamente a região de solução, ou seja, o politopo do sistema original. Para isso são necessários alguns algoritmos para encontrar os vértices e as faces do politopo. Esses algoritmos bem como suas implementações e a representação gráfica do politopo em um arquivo `obj` é o que veremos no próximo capítulo.

Capítulo 3

Representação Gráfica de um Sistema de Inequações Lineares

3.1 Introdução

Neste capítulo vamos estudar alguns métodos para representação gráfica de poliedros, que conforme vimos no Capítulo 1, é o conjunto solução de um sistema de inequações lineares $Ax \leq b$. Como estamos interessados efetivamente em desenhar o poliedro, vamos nos restringir às representações em dimensões menores ou iguais a três. Caso o sistema $Ax \leq b$ possua mais de três variáveis, podemos utilizar o método de Fourier-Motzkin para obter projeções deste poliedro no $\mathbb{R}^3$ ou $\mathbb{R}^2$.

De maneira geral, um poliedro pode ser descrito de duas formas: pela interseção de semi-espaços ($Ax \leq b$) e como o fecho convexo de seus n pontos extremos (ou vértices) [1]. Existem três problemas computacionais clássicos relacionados com as descrições de um poliedro:

- o problema da enumeração dos vértices, que consiste em determinar os vértices a partir de uma representação por semi-espaços;
- o problema da enumeração das faces, que consiste em encontrar os semi-espaços a partir dos vértices;
- o problema da verificação de um poliedro, dadas uma descrição de vértices e uma descrição de semi-espaços, decidir se ambas representações definem o mesmo poliedro.

Neste trabalho, iremos tratar do primeiro problema: obter todos os vértices de um poliedro, a partir de sua representação por semi-espaços, ou seja, a partir de um sistema de inequações lineares, $Ax \leq b$.

O problema da enumeração dos vértices tem aplicações em várias áreas e uma relação bastante próxima com programação linear. Em virtude disso, muitos algoritmos para o problema da enumeração exploram a relação entre os vértices de P e bases do método simplex [11, 2].

Os algoritmos para resolver esse problema podem ser divididos em duas classes [1]: algoritmos de *grafos transversais* e algoritmos *incrementais*.

Os algoritmos de *grafos transversais* procuram construir a lista de vértices através de um grafo associado a bases (n facetas que contém v vértices e cujos hiperplanos são independentes). Dois vértices de P são conectados por uma aresta de P se eles têm bases que diferem em apenas um membro. Alternar de um vértice para um adjacente equivale a uma mudança de base. Esta operação é conhecida como *pivoteamento*. Alguns algoritmos dessa classe são os *gift wrapping* [8], o algoritmo de Seidel [19] e o eficiente algoritmo *reverse search* de Avis e Fukuda [2], que explora uma busca reversa, baseado no sistema de troca de base do simplex.

Os algoritmos *incrementais* para o problema da enumeração dos vértices determinam o conjunto de vértices calculando sucessivamente a interseção de uma sequência de semi-espaços. Em geral, inicialmente é construído um subconjunto de n ou $n+1$ semi-espaços e seus vértices. Em seguida, as demais restrições são adicionadas e a lista de vértices atualizada através da solução de sistemas de equações lineares. Essencialmente cada atualização identifica e remove todos os vértices que não estão contidos no novo semi-espaço.

No intuito de tornar o texto didático e eventualmente servir como motivação para aplicações de conceitos de álgebra linear, optamos por concentrar os estudos nos algoritmos incrementais, cujos fundamentos matemáticos estão baseados na resolução de sistemas lineares e na determinação do fecho convexo. Salientamos que uma comparação entre algoritmos incrementais e algoritmos de grafos transversais pode constituir um interessante problema de pesquisa, porém foge do escopo deste trabalho.

A dificuldade na abordagem do problema através da solução de sistemas de equações lineares está diretamente relacionada com o número de vértices, a dimensão do poliedro P e o número de semi-espaços.

Em [11], encontramos o seguinte resultado sobre o número de vértices $v(P)$

$$v(P) \leq v_{max} = \begin{pmatrix} m + \left\lfloor \frac{1}{2}n \right\rfloor \\ m \end{pmatrix} + \begin{pmatrix} m + \left\lceil \frac{1}{2}n \right\rceil - 1 \\ m \end{pmatrix}$$

onde m é o número de inequações e n o número de variáveis.

Se o poliedro for não-vazio, simples, e o sistema $Ax \leq b$, não possuir redundâncias, em [11] temos,

$$v(P) \geq v_{min} = m(n-1) + 2.$$

Portanto, o número de vértices de um poliedro representado por m inequações e n variáveis satisfaz

$$m(n-1)+2 \quad \leq \quad v(P) \quad \leq \quad \begin{pmatrix} m + \left\lfloor \frac{1}{2}n \right\rfloor \\ m \end{pmatrix} + \begin{pmatrix} m + \left\lceil \frac{1}{2}n \right\rceil - 1 \\ m \end{pmatrix}.$$

Como v_{min} é quadrático em m e n é razoável esperarmos obter um algoritmo que encontra os vértices de P em tempo polinomial. Na seção 3.3 mostramos que este problema está bastante relacionado com o problema de encontrar o fecho convexo do poliedro dual, que no $\mathbb{R}^2$ e no $\mathbb{R}^3$ tem complexidade[1] de ordem $O(m \log m)$.

Nosso objetivo é, além de listar o conjunto de vértices, determinar todas as faces do poliedro. Mais especificamente, estaremos interessados em determinar quais vértices pertencem a cada uma das faces de P.

Para visualizar graficamente o poliedro P, vamos utilizar o formato de arquivo `obj`. Um arquivo `obj` é um documento de texto puro, inicialmente desenvolvido pela Wavefront Technologies[2] que tem sido amplamente adotado por vários outros softwares de visualização gráfica 3D, como o JavaView, 3D Studio Max, GLC_Player dentre outros.

Um arquivo `obj` contém as coordenadas dos vértices de um objeto, além das faces que formam cada polígono. Opcionalmente podemos associar a cada vértice e face, sua textura e o vetor normal.

[1] A complexidade de um algoritmo é uma medida de sua eficiência computacional em relação ao tamanho da variável de entrada. Por exemplo, se um algoritmo tem ordem $O(n^2)$ significa que existe um escalar c tal que o número total de operações realizadas para terminar o processo é menor do que cn^2.

[2] A Wavefront Technologies é uma empresa de desenvolvimento de softwares 3D de modelagem, animação e efeitos especiais.

Para visualizar o poliedro gerado no formato `obj`, vamos utilizar o software JavaView, um software gratuito de visualização de geometria 3D, que pode ser obtido diretamente em seu site oficial, http://www.javaview.de/.

O arquivo `obj` será construído por uma rotina implementada no software Octave[3], que encontra os vértices e as faces do poliedro a partir de sua representação por semi-espaços e gera o arquivo `obj` pronto para ser visualizado.

A rotina que propomos para encontrar os vértices e as faces é similar à rotina `con2vert.m` implementada por Michael Kleder e disponível no site oficial do MatLab. No entanto, a função `con2vert.m` lista apenas os vértices e não permite a determinação das faces. Nossa proposta, além de incluir esta opção, possui também algumas rotinas simples de pré-processamento que permitem resolver casos para os quais a função do `con2vert.m` falha, como por exemplo poliedros ilimitados e sistemas de inequações com certos tipos de redundâncias.

Este capítulo está organizado da seguinte maneira, na seção 3.2 apresentamos uma revisão dos principais algoritmos para determinação do fecho convexo de um conjunto de pontos no $\mathbb{R}^2$ e no $\mathbb{R}^3$. Na seção 3.3 mostramos que o problema de enumeração dos vértices é equivalente ao problema de determinar o fecho convexo do poliedro dual e com base nisso deduzimos um algoritmo. O problema de encontrar um ponto no interior do poliedro é estudado na seção 3.4. Na seção 3.5 representamos graficamente alguns sistemas de inequações lineares. Na seção 3.6 propomos uma rotina para eliminação de restrições redundantes num sistema de inequações lineares baseada no fecho convexo do dual.

3.2 Algoritmo para o fecho convexo

Vamos estudar o problema de computar o fecho convexo de um conjunto finito P de n pontos. Para este problema existem vários algoritmos. O próprio Octave ou o MatLab já possuem implementados em sua instalação uma função para abordar tal problema. Essa função é chamada $k =$`convhull`(x, y) se for no $\mathbb{R}^2$ e $k =$`convhulln`(P) para o $\mathbb{R}^n$. Elas retornam os índices dos pontos que formam o fecho convexo de P.

Suponha que V seja uma lista contendo n vértices de um politopo; o fecho convexo de V é o próprio V, a menos de ordem. A seguir, vamos apresentar alguns algoritmos para encontrar o fecho convexo de um conjunto de pontos no plano.

[3] Octave é um software livre, que utiliza a mesma sintaxe de programação do MatLab

3.2.1 Fecho convexo de um conjunto de pontos no $\mathbb{R}^2$

Inicialmente vamos considerar um problema simplificado. Suponha que o conjunto V contenha n vértices de um polígono no plano e desejamos apenas determinar em que ordem estão ligados. Podemos fazer isto fixando um vértice do conjunto e calculando os ângulos formado pelos vetores definidos por V.Vamos detalhar este processo.

Primeiramente, ordenamos o conjunto de vértices V pela primeira variável (a variável x), e fixamos os dois primeiros vértices do conjunto já ordenado v_1 e v_2. Em seguida, encontramos todos os vetores formados com relação ao vértice v_1, como podemos ver no exemplo da Figura 3.1.

Seja $W = \{w_1, w_2, ..., w_{n-1}\}$, onde $w_j = v_j - v_1$, $j = 1, 2, ..., n-1$

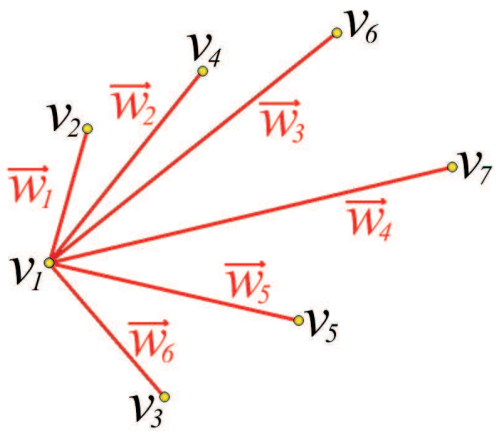


Figura 3.1: Conjunto W de vetores.

Após formarmos o conjunto W de vetores, calculamos o ângulo θ que cada vetor de W forma com $w_1 = v_2 - v_1$. Este ângulo pode ser calculado da seguinte forma:

$$\arccos \theta_i = \frac{w_1.w_i}{|w_1|.|w_i|}$$

onde $i = 2, 3, ..., n-1$.

Assim, colocando os valores dos ângulos em ordem crescente, obtemos a sequência dos vértices do polígono. No exemplo da Figura 3.1, a sequência seria

$$V = \{v_1, v_2, v_4, v_6, v_7, v_5, v_3\},$$

como podemos observar no polígono da Figura 3.2.

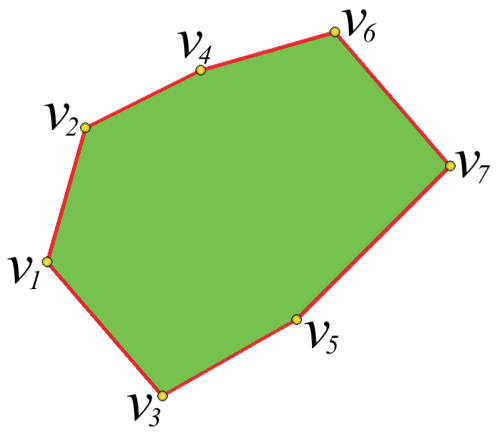


Figura 3.2: Exemplo de fecho convexo.

O algoritmo a seguir resume este processo.

Algoritmo 1: Ângulos entre vetores

Entrada: O conjunto V de vértices de um polígono no $\mathbb{R}^2$.
Saída: Uma lista ordenada do fecho convexo de V.

início
  Ordene o conjunto V de vértices pela coordenada x.
  Fixe os dois primeiros vértices, v_1 e v_2 e calcule o vetor $w_1 = v_2 - v_1$.
  para $i = 3$ *até número de linhas de* V **faça**
    Encontre todos os $w = v_i - v_1$ vetores.
    Fixe o primeiro vetor w_1 e calcule o valor do ângulo formado entre cada vetor com relação ao w_1.
  fim
  Ordene os ângulos e veja qual vértice gerou os ângulos, formando a sequência do fecho convexo.
    $F = \{f_1, f_2, ..., f_k\}$

fim

A complexidade do Algoritmo 1 é dominada pela ordenação dos vértices, que é $O(n \log n)$. Entretanto, para um número muito grande de vértices ($n > 2000$) o cálculo dos ângulos θ pode tornar o algoritmo muito lento.

Vamos considerar agora o caso geral de encontrar o fecho convexo de um conjunto qualquer de pontos no $\mathbb{R}^2$. A ideia que iremos descrever pode ser encontrada em [10]. O algoritmo encontra o fecho convexo em duas etapas. A primeira para a parte superior do polígono e a segunda para a parte inferior.

Iniciamos ordenando os vértices pela coordenada x, obtendo uma sequência de pontos $v_1, v_2, ..., v_k$. Em seguida fixamos os vértices, v_1 e v_2 e os colocamos na *lista superior*, sendo v_1 o primeiro ponto. Aqui, vamos listar os vértices no sentido horário, formando a parte superior do polígono da esquerda para a direita.

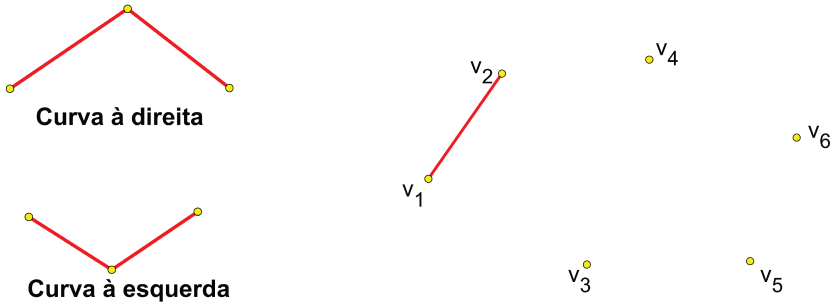


Figura 3.3: Fixamos v_1 e v_2.

Em seguida incluímos o próximo ponto v_i tal que $i = 3, 4, ..., k$ na lista superior. Ligamos os três últimos pontos da lista superior. Se fizer uma curva à direita, v_i permanece na lista e passamos para o próximo ponto. Caso contrário, se a curva for para a esquerda, retiramos da lista superior o ponto do meio, dentre os três últimos pontos da lista. Na Figura 3.4 ilustramos esse processo.

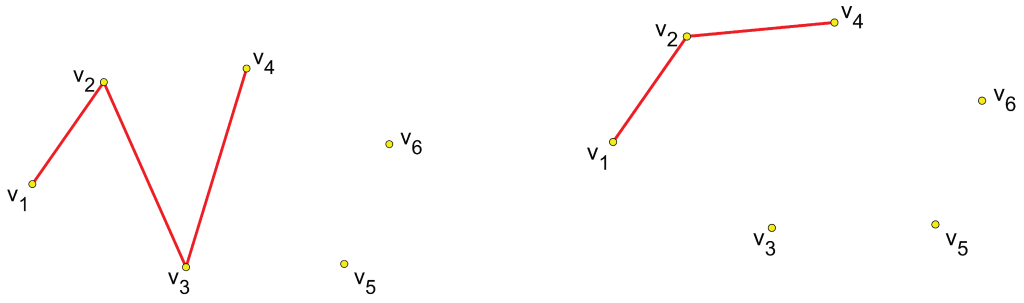


Figura 3.4: Conectamos v_1, v_2 e v_3, em seguida v_2, v_3 e v_4. Como esses últimos três não formaram uma curva à direita, retiramos da lista o ponto do meio, ou seja, v_3.

Seguimos esse processo até v_k, formando toda a *lista superior*, conforme podemos ver na Figura 3.5.

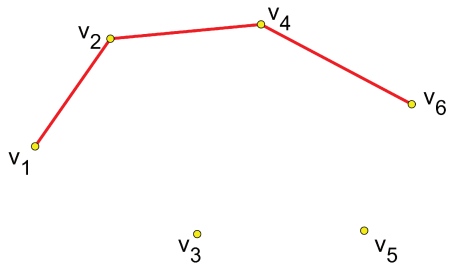


Figura 3.5: Lista superior.

Agora, vamos formar a *lista inferior*, sendo v_k o primeiro elemento e v_{k-1} o segundo. Essa lista nos dará os vértices no sentido anti-horário, formando a parte inferior do polígono. Para tanto, vamos seguir um procedimento análogo. Se v_i tal que $i = k-2, k-3, ..., 2, 1$ fizer uma curva à esquerda com os últimos dois pontos da *lista inferior*, este entra na lista, caso contrário, retiramos da *lista inferior* o segundo ponto dos três que estão sendo analisados. Podemos observar que, quando conectamos v_5, v_4 e v_3 formamos uma curva à direita, então retiramos da lista v_4 e conectamos v_5 em v_3, conforme ilustrado na Figura 3.6.

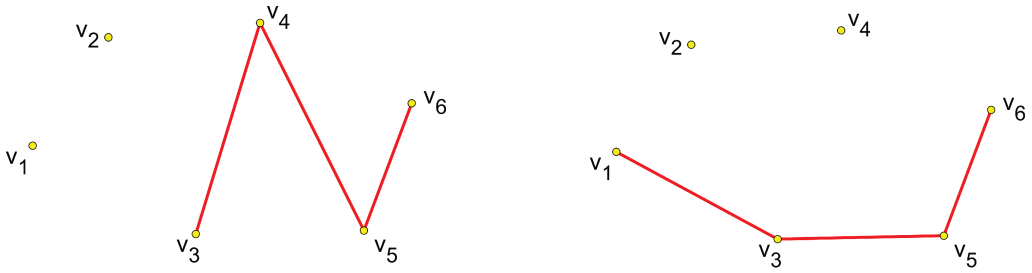


Figura 3.6: Conectamos v_6, v_5 e v_4, em seguida v_5, v_4 e v_3. Como esses últimos três não formam uma curva à esquerda, retiramos o ponto do meio, ou seja, v_4.

Retiramos da *lista inferior* o primeiro e o último vértice e a unimos com a *lista superior*, formando a lista dos vértices do polígono.

Esse método é resumido no pseudocódigo apresentado no Algoritmo 2.

Algoritmo 2: Algoritmo do fecho convexo no plano

Entrada: Um conjunto V de vértices do politopo.
Saída: Uma lista de sequência dos vértices que formam o politopo.

início

Ordene o conjunto de vértices pela coordenada x, resultando na sequência $v_1, v_2, ..., v_n$.

Fixe os vértices v_1 e v_2 em uma $L_{superior}$, tendo v_1 como o primeiro ponto da lista.

para *i = 3 até n* **faça**
Acrescente v_i em $L_{superior}$.

Enquanto $L_{superior}$ contém mais de dois pontos **e** os três últimos pontos de $L_{superior}$ formam uma curva à direita, retorne. Se não formar, retire da $L_{superior}$ o ponto do meio entre os três últimos da lista.
fim

Coloque os vértices v_n e v_{n-1} em uma lista $L_{inferior}$ sendo v_n o primeiro da lista.

para *i = n-2 até 1* **faça**
Acrescente v_i em $L_{inferior}$.

Enquanto $L_{inferior}$ contém mais de dois pontos **e** os três últimos pontos de $L_{inferior}$ formam uma curva à esquerda, retorne. Se não formar, retire da $L_{inferior}$ o ponto do meio entre os três últimos da lista.
fim

Remova da lista $L_{inferior}$ o primeiro e o último vértice para evitar duplicação de vértices.

Adicione $L_{inferior}$ em $L_{superior}$, resultando na lista F.

$F = \{f_1, f_2, ..., f_k\}$

fim

A complexidade deste algoritmo também é de ordem $O(n \log n)$ onde n é o número de vértices do polígono [10].

Dada a importância do problema de encontrar o fecho convexo, constantemente surgem algoritmos que melhoram os existentes. Luigi Giaccari, propôs em agosto de 2009, um algoritmo mais eficiente do que o `convhull(x,y)` (nativo do MatLab) para encontrar o fecho convexo de um conjunto de pontos no $\mathbb{R}^2$. O código `ConvHull2D(x,y)` desta implementação está disponível no site oficial do MatLab.

3.2.2 Algoritmo para o fecho convexo de um conjunto de pontos no $\mathbb{R}^3$

Seja P um conjunto de n pontos no $\mathbb{R}^3$. Podemos encontrar o fecho convexo de P, $conv(P)$, procedendo da seguinte forma.

Escolhemos 3 pontos quaisquer de P. Em seguida, ligamos estes três pontos em um quarto ponto qualquer de P, conforme a Figura 3.8.

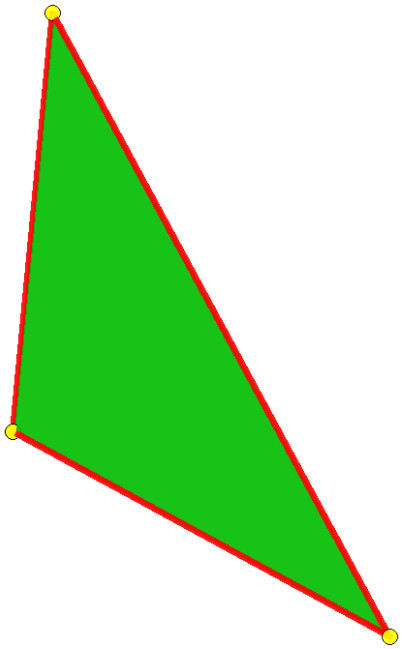


Figura 3.7: Três pontos do $\mathbb{R}^3$

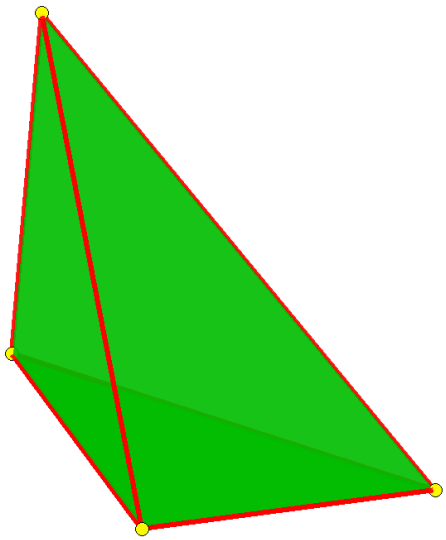


Figura 3.8: Adicionando um quarto ponto e ligando os pontos

Assim, vamos conectando os pontos restantes $p_5, ..., p_n$ um a um aos pontos extremos do poliedro (Figura 3.9). Se um ponto estiver no interior do poliedro, este é eliminado da lista, passando ao ponto seguinte (Figura 3.10).

Procedemos desta maneira até percorrer todos os pontos de P encontrando seu fecho convexo. Esta ideia pode ser encontrada descrita em detalhes no Capítulo 11 de [10].

Até aqui, vimos como encontrar o fecho convexo de um conjunto de pontos no $\mathbb{R}^2$ e no $\mathbb{R}^3$. Na próxima seção passamos a estudar o problema de encontrar o fecho convexo do conjunto solução de um sistema de inequações lineares.

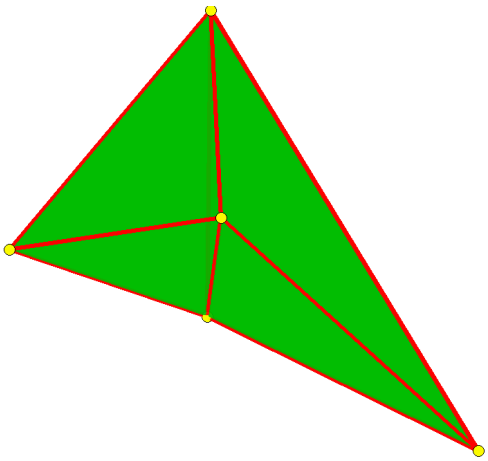


Figura 3.9: Conectando um quinto ponto ao poliedro já formado.

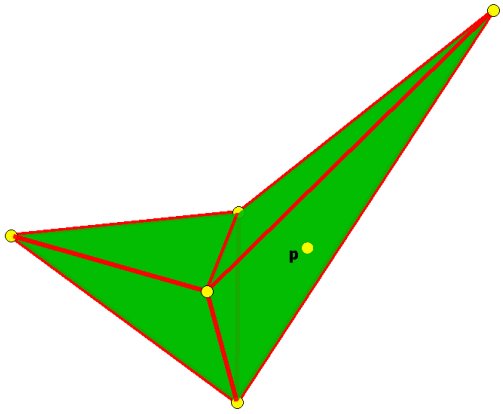


Figura 3.10: Eliminando um ponto p que está no interior do fecho convexo.

3.3 O problema da enumeração dos vértices e das faces

A interseção de semi-espaços do $\mathbb{R}^n$ pode ser representada como um sistema de inequações $\{x \in \mathbb{R}^n : Ax \leq b\}$, onde A é a matriz dos coeficientes, x é o vetor das variáveis e b é o vetor dos termos independentes. Como visto no Capítulo 1, a interseção de semi-espaços é um poliedro. Se o conjunto solução do sistema de inequações for limitado, então o poliedro correspondente a esse conjunto possui um número finito de pontos extremos (ou vértices) [1].

O problema da enumeração dos vértices consiste em determinar o conjunto de pontos extremos de um poliedro a partir de sua representação por semi-espaços, isto é, dado um sistema de inequações lineares $Ax \leq b$, determinar todos os vértices e faces do poliedro definido por este sistema.

Como mencionamos na introdução deste Capítulo, os algoritmos para resolver esse problema podem ser divididos em duas classes [1]: algoritmos de *grafos transversais* e algoritmos *incrementais*.

A seguir, apresentamos alguns algoritmos incrementais para este problema. O primeiro algoritmo apresentado é bastante ingênuo, e faz uso apenas da resolução de sistemas de equações lineares. No entanto ele será útil para adquirirmos sensibilidade no estudo do problema.

3.3.1 Algoritmo da combinação

Podemos encontrar os vértices do poliedro definido por $Ax \leq b$, resolvendo todos os sistemas de equações lineares $n \times n$ formados pelas linhas de A, ou seja, computar todas as interseções de n hiperplanos definidos pelas desigualdades em $Ax \leq b$.

Seja $P = \{p_1, p_2, ..., p_k\}$ o conjunto das soluções de todas as intersecções dos n hiperplanos definidos por $Ax \leq b$. Para que $p_i \in P$ seja um vértice, é necessário que

$$Ap_i - b \leq 0$$

Após testarmos todos os pontos $p_i \in P$ e selecionar aqueles que são factíveis, obtemos o conjunto de vértices $V = \{v_1, v_2, ..., v_n\}$.

Se nos restringirmos a esta abordagem, não há muito a ser feito além de resolver de forma eficiente os sistemas de equações lineares. O método pode ser implementado de maneira a percorrer as linhas de A de cima para baixo de forma que, cada sistema difere do anterior pela troca de algumas linhas. Isso pode ser explorado para obtermos uma fatoração LU eficiente da matriz A e, posteriormente, a resolução de um sistema triangular equivalente.

O problema desse algoritmo é que resolvemos todos os sistemas gerados pelas combinações das inequações, encontrando muitos pontos infactíveis, que serão descartados por não pertencerem ao poliedro. No Exemplo 3.1 podemos entender melhor esse problema.

Exemplo 3.1

Consideremos o seguinte sistema de inequações lineares:

$$\begin{bmatrix} 0.7071 & 0.4082 & 0.5774 \\ -0.7071 & -0.4082 & -0.5774 \\ -0.7071 & 0.4082 & 0.5774 \\ 0.7071 & -0.4082 & -0.5774 \\ 0 & -0.8165 & 0.5774 \\ 0 & 0.8165 & -0.5774 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Utilizando o algoritmo da `combinação`, devemos resolver $C_{6,3} = 20$ sistemas de equações lineares, dos quais apenas 14 geram soluções factíveis que correspondem aos 14 vértices do poliedro.

$$V = \begin{bmatrix} -1.4142 & -0.8165 & -0.5774 \\ -1.4142 & -0.8165 & 1.1547 \\ -1.4142 & 0.8165 & -1.1547 \\ -1.4142 & 0.8165 & 0.5774 \\ 0 & -1.6330 & -1.1547 \\ 0 & -1.6330 & 0.5774 \\ 0 & 0 & -1.7321 \\ 0 & 0 & 1.7321 \\ 0 & 1.6330 & -0.5774 \\ 0 & 1.6330 & 1.1547 \\ 1.4142 & -0.8165 & -0.5774 \\ 1.4142 & -0.8165 & 1.1547 \\ 1.4142 & 0.8165 & -1.1547 \\ 1.4142 & 0.8165 & 0.5774 \end{bmatrix}.$$

Então, as soluções de 6 sistemas de equações lineares foram descartadas. Essa proporção aumenta conforme aumenta o número de inequações. Por exemplo, vamos considerar um sistema com 83 inequações no $\mathbb{R}^3$, cujo poliedro está representado na Figura 3.11. Para encontrar os vértices desse poliedro utilizando o algoritmo da `combinação` devemos resolver $C_{83,3} = 91.881$ sistemas de equações lineares, dos quais 91.859 sistemas resultam em pontos infactíveis e somente 22 formaram os vértices, o que corresponde a $0,024\%$ do total.

Podemos contornar esse problema se conseguirmos determinar a priori quais inequações formam sistemas de equações lineares que resultam nos vértices, deixando de resolver aquelas cuja solução seria um ponto infactível. No exemplo, em vez de resolver 91.881 sistemas, só resolveríamos os 22 sistemas que nos interessam, tornando o processo mais eficiente. Isso é possível, como veremos na seção 3.3.2.

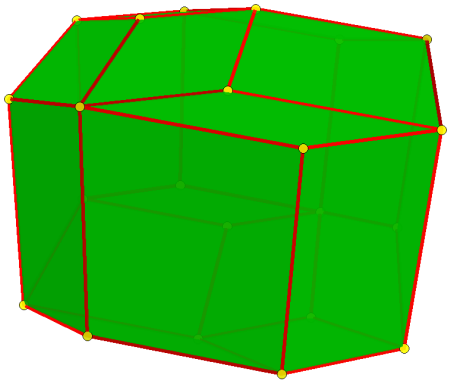


Figura 3.11: Projeção no $\mathbb{R}^3$ de um hipercubo do $\mathbb{R}^5$.

Lema 3.1 *A complexidade do algoritmo da combinação no $\mathbb{R}^n$ é de $O(m^n)$, onde m é o número de inequações do sistema $Ax \leq b$ e n é o número de variáveis.*

Demonstração: Fixada a dimensão n, o número de operações gasto para resolver cada sistema é fixo. Assim a complexidade do algoritmo é dominada pelo número de sistemas de equações lineares que deverão ser resolvidos, que é dado por

$$C_{m,n} = \frac{m!}{n!(m-n)!} = \frac{m.(m-1)...(m-n-1)(m-n)!}{n!(m-n)!} = \frac{m.(m-1)...(m-n-1)}{n!}$$

Portanto, a complexidade do algoritmo é de ordem $O(m^n)$, para n fixo.

3.3.2 Encontrando os vértices através do fecho dual

Nesta seção vamos mostrar como o número de sistemas de equações lineares necessários para a obtenção dos vértices do poliedro pode ser reduzido. Para tanto vamos introduzir a ideia de poliedro dual.

Definição 3.1 *Consideremos o seguinte sistema de inequações lineares*

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \leq b_2 \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \ddots \quad\quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \leq b_m \end{cases}, \quad com\ b_i > 0, \forall i = 1, ..., m.$$

Dividindo cada inequação por b_i, obtemos o ***sistema normalizado***

$$\begin{cases} \bar{a}_{11}x_1 + \bar{a}_{12}x_2 + ... + \bar{a}_{1n}x_n \leq 1 \\ \bar{a}_{21}x_1 + \bar{a}_{22}x_2 + ... + \bar{a}_{2n}x_n \leq 1 \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \ddots \quad\quad \vdots \\ \bar{a}_{m1}x_1 + \bar{a}_{m2}x_2 + ... + \bar{a}_{mn}x_n \leq 1 \end{cases}.$$

Cada inequação i do sistema normalizado está associada a um vetor normal $u_i = (\bar{a}_{i1}, \bar{a}_{i2}, ..., \bar{a}_{in})$. O vetor u_i é dito ***dual*** *do hiperplano $\bar{a}_{i1}x_1 + ... + \bar{a}_{in}x_n = 1$.*

Assim, associado ao sistema normalizado, que chamaremos de **primal**, temos um conjunto de m pontos duais.

A ideia que vamos apresentar a seguir consiste em transformar o problema de encontrar os vértices de $Ax \leq b$, em um outro problema mais simples, encontrar o fecho convexo do dual, que é dado por um conjunto finito de pontos. Essa ideia está comentada em algumas referências, por exemplo em [10] e [16], sem muitos detalhes. Na sequência propomos um teorema que garante a validade do método.

Inicialmente notemos que a exigência $b_i > 0$ implica que a origem é um ponto interior do poliedro. Para que possamos aplicar o método para um poliedro qualquer, será necessária uma translação desse poliedro, conforme descrevemos a seguir.

Seja P um poliedro $Ax \leq b$ não-vazio qualquer e x_0 um ponto interior a P, isto é, $Ax_0 < b$.

Vamos transladar o poliedro, de forma que a origem coincide com x_0.

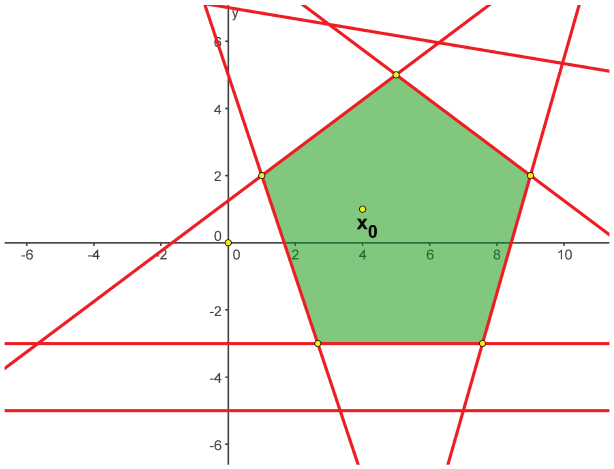


Figura 3.12: Poliedro $Ax \leq b$.

Para transladar, definimos um vetor

$$w = b - Ax_0 > 0.$$

Então temos,

$$Ax - Ax_0 \leq b - Ax_0$$

$$\Downarrow$$

$$A(x - x_0) \leq w$$

$$\Downarrow$$

$$Ay \leq w$$

onde $y = x - x_0$.

O poliedro transladado P' tem $w > 0$, ou, o que é equivalente, a origem é ponto interior de P', conforme ilustrado na Figura 3.13.

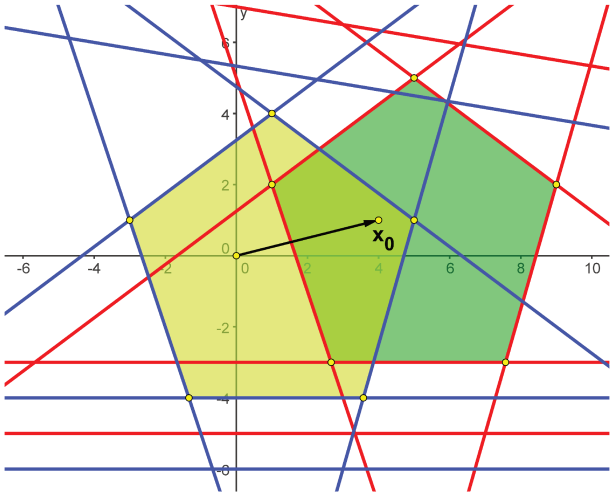


Figura 3.13: Poliedro P e sua translação P'.

Vamos normalizar o sistema $Ay \leq w$, dividindo cada linha por w_i, obtendo $\bar{A}y \leq 1$, de onde podemos extrair o conjunto dual, que está representado pelos vetores na Figura 3.14.

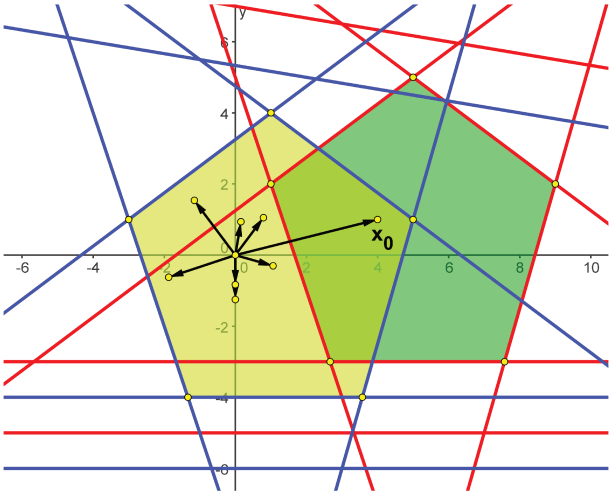


Figura 3.14: Poliedro transladado P' e os vetores que formam o dual.

Encontrando o fecho do dual (Figura 3.15), podemos determinar quais as inequações de $Ax \leq b$ irão resultar nos vértices de P, de acordo com o Teorema 3.1.

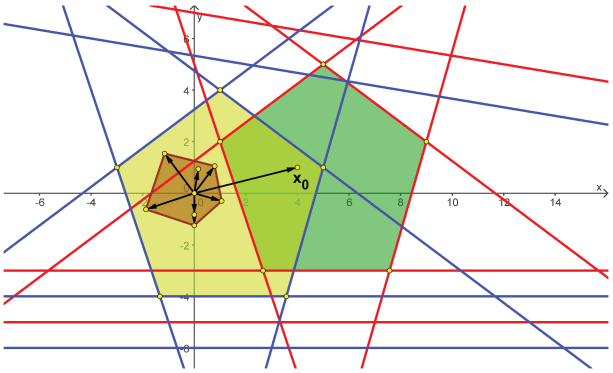


Figura 3.15: Fecho do dual.

Teorema 3.1 *Seja $Ax \leq 1$ um sistema de inequações lineares normalizado e $\bar{A}$ uma submatriz de ordem $n \times n$ de A, cujas linhas são representadas por $(\bar{A}_1, \bar{A}_2, ..., \bar{A}_n)$. Então o vetor c, solução do sistema $\bar{A}x = 1$ é um vértice do sistema primal se, e somente se, os pontos $(\bar{A}_1, \bar{A}_2, ..., \bar{A}_n)$ estiverem na mesma face do fecho convexo do dual de $Ax \leq 1$.*

Demonstração: Vamos supor que c seja um vértice do poliedro primal normalizado, que satisfaz $\bar{A}c = 1$.

Suponhamos, por absurdo, que as linhas de $\bar{A}$ não estão numa mesma face do dual. Isto equivale dizer que existe um vetor dual A_j, tal que $A_j = \alpha_1\bar{A}_1 + ... + \alpha_n\bar{A}_n$, com $\alpha_i \geq 0$ e $\sum_{i=1}^{n} \alpha_i > 1$.

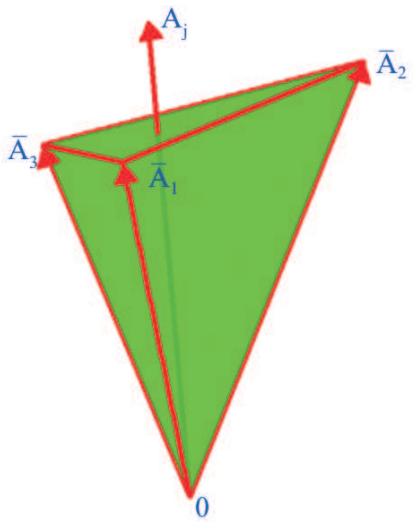


Figura 3.16: Linhas $\bar{A}$ e A_j.

Como c é um vértice de $Ax \leq 1$, temos que $A_jc \leq 1$. Porém,

$$\begin{aligned} A_jc &= (\alpha_1\bar{A}_1 + ... + \alpha_n\bar{A}_n)c \\ &= \alpha_1\bar{A}_1c + ... + \alpha_n\bar{A}_nc \\ &= \alpha_1 + ... + \alpha_n > 1, \end{aligned}$$

o que é uma contradição. Logo A_j não pode existir, portanto os vetores $(\bar{A}_1, \bar{A}_2, ..., \bar{A}_n)$ pertencem à mesma face do dual.

Sejam agora $(\bar{A}_1, \bar{A}_2, ..., \bar{A}_n)$ n pontos que estão na mesma face do dual de $Ax \leq 1$.

A matriz

$$\bar{A} = \begin{bmatrix} \bar{A}_1 \\ \vdots \\ \bar{A}_n \end{bmatrix}$$

é não singular, pois $\bar{A}_1, \bar{A}_2, ..., \bar{A}_n$ são LI.

Seja c a solução de $\bar{A}x = 1$. Devemos mostrar que c satisfaz $Ax \leq 1$, ou seja, c é um ponto factível para o sistema primal normalizado $Ax \leq 1$.

Do fato de $(\bar{A}_1, \bar{A}_2, ..., \bar{A}_n)$ estarem na mesma face do dual, temos que todos os pontos do dual P' devem pertencer ao semi-espaço definido por $c^T x \leq 1$ (não poderia ser o semi-espaço $c^T x \geq 1$, pois o zero é sempre ponto interior do dual).

Como os pontos do dual são iguais aos vetores-linha de A, temos que

$$A_j^T c \leq 1, \ \forall \ j = 1, ..., m.$$

Com base neste teorema, podemos encontrar o conjunto de vértices de um poliedro qualquer. Se a origem for um ponto interior do poliedro, então $b > 0$, e o sistema pode ser normalizado diretamente. Após encontrarmos o fecho convexo do dual, saberemos quais os sistemas de equações lineares devem ser resolvidos para obtenção dos vértices.

No caso de $b_i < 0$ a origem não é ponto interior e necessitamos de transladar o poliedro para depois normalizá-lo. Neste caso temos um problema adicional de encontrar um ponto interior a $Ax \leq b$ que será discutido na seção 3.4. Após encontrado os vértices do sistema transladado, basta desfazer o deslocamento e obter os vértices do poliedro original.

3.4 Encontrando um ponto interior de $Ax \leq b$

Um ponto c é dito ponto interior de um sistema $Ax \leq b$ com m inequações se $a_i^T c < b_i$ para todo i $(i = 1, 2, ..., m)$.

O problema de encontrar um ponto interior a um poliedro tem aplicações em muitas áreas, sobretudo em programação linear e, em geral, é considerado como um problema difícil. Existem vários trabalhos dedicados exclusivamente a esta questão, por exemplo [6] e [7].

Nesta seção, vamos apresentar alguns métodos para encontrar um ponto interior de um poliedro, que satisfazem nossas necessidades (poliedros em dimensões menores que 3).

A primeira tentativa para obter um ponto interior consiste em observar que se $b > 0$ então $x_0 = 0$ é ponto interior do poliedro $Ax \leq b$, pois $Ax_0 = 0 < b$.

Uma segunda tentativa pode ser feita procurando um vetor x_0 que resolve o problema de mínimos quadrados $\min ||b - Ax||$. Sabemos que, neste caso, $x_0 = (A^T A)^{-1} A^T b = A \setminus b$. Se $Ax_0 < b$ então o ponto interior está encontrado.

Um caso complicado ocorre quando nenhuma das duas primeiras tentativas tem sucesso. Para este problema vamos propor dois outros métodos que resolvem um problema para um poliedro qualquer.

3.4.1 Encontrando um ponto interior através do método da eliminação de Fourier-Motzkin

Para encontrarmos um ponto interior c de um poliedro $Ax \leq b$, podemos utilizar o método de eliminação de Fourier-Motzkin visto no Capítulo 2. Escolhemos um eixo para projetar a solução, encontrando seus pontos extremos, depois pegamos o ponto médio e calculamos as coordenadas correspondentes a este ponto escolhido. Caso não haja limite superior, escolhemos como o ponto pertencente à projeção o ponto resultante do limite inferior + k, sendo k um escalar. Se não existir um limite inferior, o ponto será limite superior - k. Caso não exista limite inferior e nem superior, escolhemos o zero. No Exemplo 3.2 ilustramos esse processo.

Exemplo 3.2

Considere o sistema

$$\begin{cases} -36x - 20y - 14z & \leq & 43.4971 \\ -15x + 31y + 10z & \leq & 35.8608 \\ -10x + 7y + 17z & \leq & 20.9284 \\ -4x + 38y - 23z & \leq & 44.5982 \\ 9x + 33y + 2z & \leq & 34.2637 \\ 34x - 14y - 16z & \leq & 40.0999 \end{cases}.$$

Eliminando as variáveis y e z, e projetando a região de solução no eixo x, podemos escolher o ponto $x = 0.5001$. Calculando o valor correspondente para as variáveis y e z temos o seguinte ponto interior:

$$pint = [0.5001, -2.4445, 1.6137]^T$$

que está representado na Figura 3.17. Esse método é bem robusto, porém bastante lento para um sistema com um número grande de inequações. Além disso, como a projeção utilizando a eliminação de Fourier-Motzkin pode aumentar exponencialmente o número de restrições, podem surgir problemas de demanda excessiva de memória.

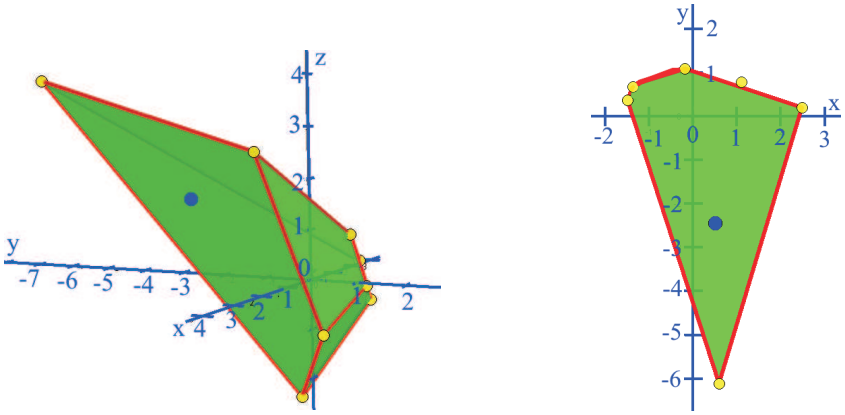


Figura 3.17: Ponto interior do poliedro referente ao Exemplo 3.2 e sua projeção no plano.

3.4.2 Encontrando um ponto interior utilizando programação linear

Podemos encontrar um ponto interior do poliedro P caracterizado pelo sistema de inequações lineares

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

resolvendo o seguinte problema de programação linear:

$$minimizar(x_{n+1} + x_{n+2})$$

$$\text{sujeito a} \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & -b_1 & 1 \\ a_{21} & a_{22} & \dots & a_{2n} & -b_2 & 1 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & -b_m & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

E, $x_{n+1} > 0$ e $x_{n+2} > 0$. Logo temos,

$$a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n - b_j x_{n+1} + x_{n+2} \leq 0 \ \ ; \ j = 1, 2, \dots, n.$$

Que podemos escrever

$$a_{j1}\frac{x_1}{x_{n+1}} + a_{j2}\frac{x_2}{x_{n+1}} + ... + a_{jn}\frac{x_n}{x_{n+1}} - b_j \leq \frac{-x_{n+2}}{x_{n+1}} < 0;$$

Ou seja, $c = \left(\frac{x_1}{x_{n+1}}, \frac{x_2}{x_{n+1}}, ..., \frac{x_n}{x_{n+1}}\right)$ é um ponto interior ao poliedro P.

Na rotina `pontointerior.m` (Apêndice A) apresentamos uma implementação das ideias descritas nesta seção.

Agora que podemos encontrar o ponto interior do poliedro $Ax \leq b$, podemos utilizar o fecho dual para listar as inequações que formam os sistemas que resultam nos vértices e resolvê-los.

3.5 Representação geométrica

Nesta seção vamos apresentar alguns exemplos tanto no $\mathbb{R}^2$ quanto no $\mathbb{R}^3$ do funcionamento geral do método para encontrar os vértices de um sistema de inequações lineares pelo dual. Vamos mostrar também como geramos o arquivo `obj` e incluir alguns desenhos do JavaView.

Exemplo 3.3

Vamos iniciar representando graficamente o poliedro de um sistema de inequações lineares do $\mathbb{R}^2$.

Então, seja o sistema

$$\begin{bmatrix} -37 & 29 \\ -25 & -16 \\ -19 & 6 \\ -14 & -5 \\ 14 & -20 \\ 20 & 40 \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 47.0106 \\ 29.6816 \\ 19.9249 \\ 14.8661 \\ 24.4131 \\ 44.7214 \end{bmatrix} . \tag{3.1}$$

Precisamos encontrar seus vértices e em seguida seu fecho, para assim gerar o desenho do poliedro.

Para encontrar seus vértices, vamos utilizar o algoritmo `vertdual.m`. Além dos vértices, queremos saber quais inequações geraram cada vértice, para em seguida encontrar o fecho convexo e assim poder gerar a representação geométrica do poliedro.

Usando as funções implementadas no Octave, chegamos no seguinte resultado

$$V = \begin{bmatrix} -1.0549 & -0.0196 & 4.0000 & 5.0000 \\ -0.9035 & -0.4433 & 2.0000 & 5.0000 \\ -0.8990 & 0.4741 & 1.0000 & 4.0000 \\ -0.2833 & 1.2597 & 1.0000 & 3.0000 \\ -0.2804 & -1.4169 & 2.0000 & 6.0000 \\ 1.9489 & 0.1436 & 3.0000 & 6.0000 \end{bmatrix}$$

Nas duas primeiras colunas, temos os vértices do poliedro, e nas duas últimas, os índices das inequações que geraram cada vértice. Por exemplo, o vértice (-1.0549 , -0.0196) veio da solução do sistema de equações formado pelas inequações 4 e 5 do sistema $Ax \leq b$.

Agora, determinando o fecho convexo desse conjunto de vértices, obtemos o desenho do poliedro.

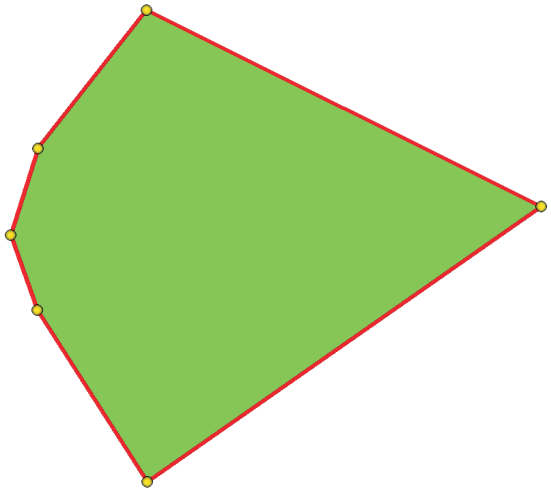


Figura 3.18: Representação geométrica do politopo do sistema (3.1).

Exemplo 3.4

Consideremos o seguinte sistema de inequações do $\mathbb{R}^3$:

$$\begin{bmatrix} -35 & -27 & 5 \\ -35 & 27 & -26 \\ -32 & -24 & -38 \\ -16 & 37 & 19 \\ -15 & 33 & 15 \\ -11 & -11 & 33 \\ -6 & 18 & 36 \\ 10 & -35 & 22 \\ 17 & -40 & -16 \\ 29 & 1 & -26 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq \begin{bmatrix} 44.4860 \\ 51.2835 \\ 55.1725 \\ 44.5646 \\ 39.2301 \\ 36.4829 \\ 40.6940 \\ 42.5323 \\ 46.3141 \\ 38.9615 \end{bmatrix} . \tag{3.2}$$

Usando os mesmo algoritmos do Exemplo 3.3, obtemos os seguintes resultados:

$$V = \begin{bmatrix} -1.4817 & 0.3432 & 0.3786 & 1.0000 & 2.0000 & 7.0000 \\ -1.3514 & 0.2403 & 0.7352 & 1.0000 & 7.0000 & 8.0000 \\ -1.3048 & 0.2526 & 0.7548 & 6.0000 & 7.0000 & 8.0000 \\ -1.2656 & -0.0703 & -0.3417 & 1.0000 & 2.0000 & 3.0000 \\ -1.2146 & 0.2727 & 0.7916 & 6.0000 & 8.0000 & 10.0000 \\ -0.4529 & -0.9419 & 0.6406 & 1.0000 & 8.0000 & 9.0000 \\ -0.4068 & -1.1869 & -0.3597 & 1.0000 & 3.0000 & 5.0000 \\ -0.2784 & -1.2834 & 0.0181 & 1.0000 & 5.0000 & 9.0000 \\ -0.0751 & 0.2890 & -1.5712 & 2.0000 & 3.0000 & 4.0000 \\ 0.2136 & -0.5544 & -1.2816 & 3.0000 & 4.0000 & 5.0000 \\ 0.5427 & 1.8098 & -0.8236 & 2.0000 & 4.0000 & 7.0000 \\ 0.8324 & -0.1367 & 1.3375 & 8.0000 & 9.0000 & 10.0000 \\ 1.1377 & 1.7792 & -0.1611 & 4.0000 & 6.0000 & 7.0000 \\ 1.6090 & -0.5835 & 0.2736 & 4.0000 & 5.0000 & 9.0000 \\ 1.8140 & 1.6861 & 0.5897 & 4.0000 & 6.0000 & 10.0000 \\ 2.5577 & 0.3759 & 1.3687 & 4.0000 & 9.0000 & 10.0000 \end{bmatrix} .$$

Nas três primeiras colunas estão listadas as coordenadas dos vértices e, nas três últimas, os índices das inequações que formaram os sistemas de equações que geraram cada vértice.

Exemplo 3.5

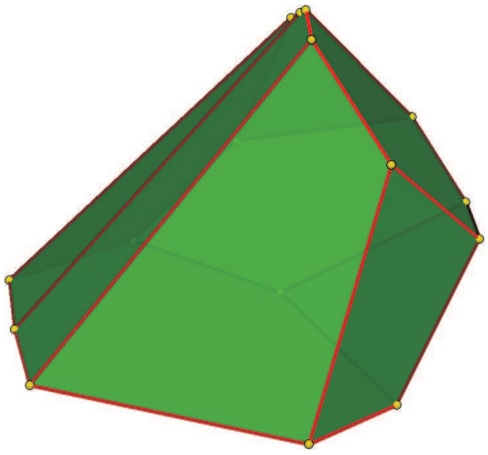


Figura 3.19: Representação geométrica do poliedro do sistema (3.2).

Vejamos como podemos representar a projeção de um sistema de inequações do $\mathbb{R}^4$ no $\mathbb{R}^3$ e no $\mathbb{R}^2$.

Seja o seguinte sistema de inequações:

$$\begin{bmatrix} -35 & -1 & 18 & 5 \\ -25 & -29 & 21 & 6 \\ -25 & 39 & 40 & -21 \\ -21 & -38 & 16 & -5 \\ -20 & 33 & -23 & -18 \\ -17 & -3 & 3 & 33 \\ 3 & -13 & 0 & -4 \\ 13 & 14 & 22 & 16 \\ 20 & -33 & -8 & 17 \\ 25 & -35 & -8 & 36 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leq \begin{bmatrix} 39.6863 \\ 44.0795 \\ 64.7070 \\ 46.5403 \\ 48.3942 \\ 37.3631 \\ 13.9284 \\ 33.2415 \\ 42.9185 \\ 56.6569 \end{bmatrix}. \tag{3.3}$$

Precisamos reduzir esse sistema para o $\mathbb{R}^3$. Para isso aplicamos o `fourmotz.m`, ou seja, a eliminação de Fourier-Motzkin. Assim, eliminando a variável x_4, vamos projetar o poliedro do sistema (3.3) nos eixos x_1, x_2 e x_3. O novo sistema reescrito nestas três variáveis é:

$$\begin{bmatrix} -0.0140 & -0.0840 & 1.4342 \\ -3.0235 & -9.5412 & 2.7294 \\ -3.5056 & -8.5722 & 2.9778 \\ -3.4167 & -8.0833 & 3.5000 \\ -3.3875 & -6.7250 & 4.5750 \\ -4.7152 & -7.6909 & 3.2909 \\ -8.3667 & -12.4333 & 6.7000 \\ -11.2000 & -7.8000 & 6.8000 \\ -5.2778 & -3.0000 & 2.2222 \\ -5.3571 & -2.9762 & 5.4048 \\ -6.2500 & -3.4500 & 3.6000 \\ -8.1111 & 1.6333 & 2.3222 \\ -8.1905 & 1.6571 & 5.5048 \\ -1.7056 & 1.7662 & 1.9957 \\ -1.6263 & 1.7424 & -1.1869 \\ -0.4960 & 0.8849 & 1.6825 \\ -0.4167 & 0.8611 & -1.5000 \\ -0.3780 & 2.7321 & 3.2798 \\ -0.2986 & 2.7083 & 0.0972 \\ 0.2348 & -3.3409 & 0.0909 \\ 1.4444 & -4.2222 & -0.2222 \\ 1.9265 & -5.1912 & -0.4706 \\ 0.0654 & -0.1078 & -1.7484 \\ 1.5625 & -2.3750 & 1.3750 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 5.6059 \\ 11.8327 \\ 10.8819 \\ 10.8287 \\ 11.3857 \\ 10.4403 \\ 16.6546 \\ 17.2453 \\ 10.0351 \\ 10.4279 \\ 11.4194 \\ 10.6258 \\ 11.0185 \\ 4.2135 \\ 3.8208 \\ 4.6551 \\ 4.2624 \\ 5.1589 \\ 4.7662 \\ 4.6143 \\ 5.0559 \\ 6.0067 \\ 5.2132 \\ 5.5597 \end{bmatrix}.$$

Utilizando o algoritmo para encontrar os vértices, temos

$$V = \begin{bmatrix}
-2.0027 & -0.8954 & -1.7896 & 0 & 1.0000 & 4.0000 & 8.0000 & 15.0000 \\
-1.8117 & -1.5117 & -2.9562 & 0 & 8.0000 & 12.0000 & 15.0000 & 23.0000 \\
-1.7372 & -1.5342 & -2.9521 & 23.0000 & 12.0000 & 13.0000 & 0 & 0 \\
-1.5859 & -0.8204 & -1.0171 & 8.0000 & 1.0000 & 2.0000 & 0 & 0 \\
-1.3286 & -0.2230 & 0.0919 & 1.0000 & 4.0000 & 3.0000 & 0 & 0 \\
-1.0235 & 1.2976 & 0.0882 & 0 & 3.0000 & 4.0000 & 14.0000 & 15.0000 \\
-0.8852 & -0.6593 & -2.9742 & 23.0000 & 19.0000 & 15.0000 & 0 & 0 \\
-0.6642 & 0.9081 & 0.7400 & 3.0000 & 14.0000 & 20.0000 & 0 & 0 \\
-0.6549 & -0.2971 & 1.1167 & 0 & 1.0000 & 2.0000 & 3.0000 & 6.0000 \\
-0.6384 & -1.1826 & -0.5062 & 12.0000 & 8.0000 & 2.0000 & 0 & 0 \\
-0.6093 & 1.6892 & 0.0956 & 0 & 14.0000 & 15.0000 & 20.0000 & 21.0000 \\
-0.2816 & -0.0794 & 1.6066 & 6.0000 & 3.0000 & 20.0000 & 0 & 0 \\
0.3873 & -0.5028 & 2.0364 & 0 & 2.0000 & 6.0000 & 11.0000 & 20.0000 \\
0.4126 & -1.0923 & 0.9740 & 0 & 2.0000 & 10.0000 & 12.0000 & 13.0000 \\
0.7025 & -0.9182 & 1.6592 & 0 & 2.0000 & 10.0000 & 11.0000 & 16.0000 \\
0.7301 & -1.0181 & 1.4553 & 10.0000 & 16.0000 & 13.0000 & 0 & 0 \\
0.7461 & -0.6002 & 2.1589 & 11.0000 & 20.0000 & 16.0000 & 0 & 0 \\
1.2696 & 1.9738 & -2.0611 & 19.0000 & 21.0000 & 15.0000 & 0 & 0 \\
5.4103 & 2.4616 & -2.9313 & 19.0000 & 23.0000 & 21.0000 & 0 & 0 \\
7.1701 & 2.5402 & 0.2832 & 16.0000 & 20.0000 & 21.0000 & 0 & 0 \\
10.6005 & 3.0281 & -2.7722 & 0 & 13.0000 & 16.0000 & 21.0000 & 23.0000
\end{bmatrix}$$

que resulta na projeção vista na Figura 3.20. As cinco últimas colunas da matriz V indicam os índices das inequações que formaram os sistemas para gerar cada vértice. O índice 0 deve ser ignorado.

Eliminando x_3 e utilizando os mesmos algoritmos projetamos no plano o poliedro correspondente ao sistema (3.3).

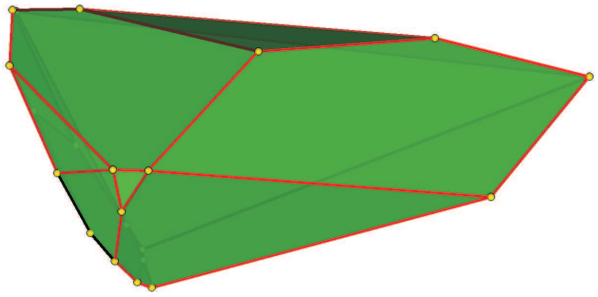


Figura 3.20: Representação geométrica da projeção do poliedro do sistema (3.3 nas variáveis x_1, x_2 e x_3).

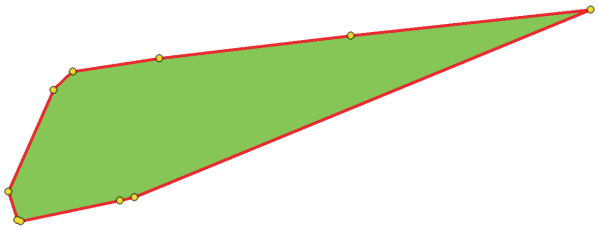


Figura 3.21: Representação geométrica da projeção do poliedro do sistema (3.3) nas variáveis x_1 e x_2.

3.6 Eliminação de inequações redundantes utilizando o fecho do dual

Vamos mostrar como o método descrito na seção 3.3.2 pode ser utilizado para eliminar inequações redundantes num sistema $Ax \leq b$. Salientamos que existem muitos métodos para fazer isto, referências para o assunto podem ser encontradas em [12].

Podemos identificar as inequações redundantes num sistema $Ax \leq b$ da seguinte maneira: reduzimos este sistema ao sistema normalizado $\bar{A}y \leq 1$ e encontramos o $conv(\bar{A})$, ou seja, o fecho do dual. Somente as restrições que formam o fecho do dual devem ser consideradas. Assim, temos todas as inequações relevantes do sistema $Ax \leq b$ e basta reescrever o sistema original, eliminando as inequações redundantes.

Vimos alguns algoritmos que nos permitem encontrar os vértices de um poliedro, enumerar suas facetas e como representá-lo graficamente. Também podemos ver como eliminar as inequações redundantes com eficiência e como encontrar o ponto interior de um poliedro.

Neste momento, fazemos alguns questionamentos: qual dos algoritmos é o mais eficiente? Qual o ideal em deteminados tipos de sistemas? Qual a diferença, vantagens e desvantagens entre um e outro?

Para respondermos esses questionamentos procuramos fazer algumas experiências computacionais. Essas experiências estão registradas no próximo capítulo.

Capítulo 4

Experiências Computacionais

4.1 Introdução

Neste capítulo iremos investigar os algoritmos apresentados neste trabalho com o intuito de criar uma rotina que permita determinar o conjunto de vértices e faces de um poliedro definido por um sistema de inequações lineares $Ax \leq b$. Esta rotina irá produzir um arquivo de texto puro em formato `obj` que pode ser visualizado em diversos softwares de geometria 3D. Nos exemplos que vamos apresentar, utilizamos o programa JavaView para visualizar as figuras.

Essa rotina incluirá algoritmos para encontrar um ponto interior, identificar e eliminar inequações redundantes, listar o conjunto de vértices e faces e determinar o seu fecho convexo. Neste trabalho, já apresentamos vários algoritmos que executam essas tarefas. Para a rotina, precisamos escolher os algoritmos que melhor atendem às nossas necessidades, visando alcançarmos o principal objetivo, que é desenhar o poliedro. Enfatizamos que vamos nos concentrar em poliedros de dimensão menor ou igual a 3, caso o sistema de inequações lineares possua mais de 3 variáveis, vamos utilizar o método de Fourier-Motzkin para obter uma projeção deste poliedro no $\mathbb{R}^2$ ou no $\mathbb{R}^3$.

A escolha dos algoritmos será baseada nas análises feita através de algumas experiências computacionais realizadas no Octave e no MatLab R2009b, em um computador Intel Celeron, 2.13 GHz, 2MB de memória RAM, Windows XP. Para cada experiência, vamos considerar a média entre 10 testes realizados com sistemas de inequações lineares construídos de forma pseudo-aleatória.

Ao gerar um sistema de inequações lineares de forma aleatória, existe uma grande possibilidade de produzirmos sistemas infactíveis. Para contornar isto, resolvemos criar duas classes de problemas: uma com hipercubos rotacionados centrados na origem e outra com hiperplanos tangentes à hiperesfera de raio igual a 1 centrada na origem (que eventualmente pode ter o centro deslocado). Algoritmos para construir essas duas classes de problemas são apresentados no Apêndice F.

4.2 Experiência 1

Nesta experiência vamos investigar o desempenho dos algoritmos para encontrar um ponto interior de um poliedro qualquer apresentados na seção 3.4 do Capítulo 3. Caso $b > 0$, a origem é ponto interior. Se existe $b_i < 0$, resolvemos o problema de mínimos quadrados $\min ||b - Ax||$, cuja solução no Octave ou MatLab pode ser obtida fazendo $c = A \backslash b$. Se $Ac < b$, então c é um ponto interior. Essas duas estratégias para encontrar um ponto interior são bastante eficientes e serão priorizadas.

Caso c não seja um ponto interior, precisamos escolher entre o algoritmo utilizando o método de eliminação de Fourier-Motzkin ou o algoritmo da solução do problema de programação linear, descritos na seção 3.4. Para fazer isto, vamos comparar o esforço computacional (medido através do tempo de processamento) requerido para encontrar um ponto interior em cada um destes dois algoritmos.

Para esta experiência vamos gerar sistemas de inequações lineares pseudo-aleatórias cuja matriz A tem dimensão $m \times 3$. Cada sistema corresponde a m planos tangentes à esfera unitária do $\mathbb{R}^3$. Para cada valor de m, o tempo de execução de cada rotina, apresentado na Tabela 4.1, refere-se à média entre 10 experiências aleatórias realizadas. Nessas experiências foram comparados apenas o método de Fourier-Motzkin e o método usando a programação linear.

Com base nos tempos apresentados na Tabela 4.1, podemos concluir que, para encontrar um ponto interior do conjunto solução de um sistema de inequações $Ax \leq b$ com m inequações e 3 variáveis, o método de eliminação de Fourier-Motzkin pode ser mais eficiente para casos onde temos um número pequeno de inequações ($m < 20$). Para casos maiores ($m > 100$), além de tornar-se bem mais lento, em alguns casos o uso excessivo de memória pode ser restritivo. Em contra-partida, com este método é possível identificar se o sistema $Ax \leq b$ possui um conjunto solução ilimitado ou até se é incompatível.

m	Eliminação de Fourier-Motzkin	Programação Linear
	Tempo (segundos)	Tempo (segundos)
5	0.0023	0.0124
10	0.0035	0.0127
20	0.0055	0.0223
40	0.0982	0.0228
50	0.2580	0.0236
100	6.7746	0.0284
200	Excesso de memória	0.0363
500	Excesso de memória	0.0929
1000	Excesso de memória	0.2506
2000	Excesso de memória	0.7809

Tabela 4.1: Tempo gasto para encontrar um ponto interior a um poliedro no $\mathbb{R}^3$.

Como estamos interessados em encontrar, de forma eficiente, um ponto interior ao poliedro, o algoritmo que melhor responde às nossas necessidade é aquele que utiliza a solução de um problema de programação linear descrito na seção 3.4.2, pois este mostrou-se eficiente mesmo quando cresce o número de restrições.

Portanto, na rotina final, utilizaremos a seguinte sequência de passos para encontrar um ponto interior ao poliedro:

1. Se $b > 0$, então a origem é um ponto interior;

2. Se existe $b_i < 0$, fazemos $c = (A^T A)^{-1} A^T b$ (em MatLab ou Octave, $c = A \setminus b$).Se $Ac < b$ então c é um ponto interior.

3. Caso as duas primeiras estratégias falhem, utilizaremos o método descrito na seção 3.4.2 que usa programação linear.

4.3 Experiência 2

Nesta experiência vamos considerar o problema de eliminar inequações redundantes em um sistema $Ax \leq b$. Essencialmente estaremos interessados em obter uma rotina que utilize os conceitos envolvidos neste trabalho e que tenha desempenho igual ou melhor do que a rotina `reduce_rows` implementada por Sebastian Siegel e disponível no site oficial do MatLab.

A rotina `reduce_rows` não elimina todas as redundâncias, como no caso do Exemplo 4.1.

Exemplo 4.1

Seja o seguinte sistema de inequações,

$$\begin{bmatrix} -1.1547 & -0.8165 & -1.4142 \\ -0.8165 & 0.2887 & 0.5000 \\ 0 & -0.8660 & 0.5000 \\ 0 & 0 & 0 \\ 0 & 0.8660 & -0.5000 \\ 0.8165 & -0.2887 & -0.5000 \\ 1.1547 & 0.8165 & 1.4142 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 2.8284 \\ 1.0000 \\ 1.0000 \\ 2.8284 \\ 1.0000 \\ 1.0000 \\ 2.8284 \end{bmatrix}.$$

Podemos notar que a quarta linha do sistema é redundante. Aplicando o método `reduce_rows`, o sistema não é alterado.

Conhecendo um ponto interior ao poliedro definido por um sistema de inequações lineares $Ax \leq b$, podemos utilizar o teorema do fecho do dual (Teorema 3.1), para identificar e eliminar redundâncias no sistema $Ax \leq b$. Esse algoritmo é o que denominamos `eliminaredu.m`, visto no capítulo anterior.

Para comparar esses dois métodos realizamos alguns testes gerando sistemas de inequações aleatórios $Ax \leq b$ e posteriormente considerando para os testes a matriz $\bar{A}^T = [A^T A^T 0]$ e $\bar{b}^T = [b^T b^T 0]$. Conforme ilustrado no Exemplo 4.1 o algoritmo `reduce_rows` não eliminou a linha nula em nenhum caso. Nosso método `eliminaredu.m` mostrou-se robusto e eficiente nos testes realizados. Além de eliminar todas as redundâncias, o tempo total gasto neste processo foi considerado aceitável. Para $m = 1000$ a média foi de 0.013 segundos, para $m = 10000$ a média foi de 0.1622 segundos.

Com base nesta experiência, vamos escolher o algoritmo `eliminaredu.m` quando for necessário identificar e eliminar redundâncias em um sistema de inequações lineares.

4.4 Experiência 3

Vamos tratar agora do problema central deste trabalho: escolher um algoritmo para listar de forma eficiente o conjunto de vértices e faces de um poliedro e possibilitar sua visualização gráfica. Os algoritmos que iremos comparar são `con2vert.m` e `vertdual.m`, que foram apresentados no Capítulo 3.

Como já dissemos, o algoritmo `con2vert.m` está disponível no site oficial do MatLab e utiliza algumas ideias semelhantes às que implementamos no `vertdual.m`. A maior

crítica ao `con2vert.m` consiste no fato de que este método produz apenas uma lista de vértices e não permite identificar as faces do poliedro. Além disso, este método não possui nenhuma rotina de pré-processamento e pode falhar em casos em que o sistema tenha restrições redundantes.

O método que propomos resolve estes problemas e lista o conjunto de vértices e faces do poliedro. Além disso, quando utilizado na rotina `desenha.m` nosso método permite que os resultados sejam gravados em um arquivo `obj` e possibilita a imediata visualização do poliedro em um software gráfico.

Utilizaremos nesta experiência sistemas de inequações lineares no $\mathbb{R}^2$ e sistemas de inequações lineares no $\mathbb{R}^3$, correspondentes a planos tangentes a uma esfera de raio igual a um, gerados aleatoriamente, não tendo a origem como ponto interior.

Para cada valor de m, foram feitos 10 testes aleatórios e considerado como resultado o tempo médio gasto por cada um dos métodos para resolver os mesmos problemas. Os resultados estão apresentados nas Tabelas 4.2 e 4.3.

m	número de vértices	`con2vert.m`	`vertdual.m`
		Tempo (segundos)	Tempo (segundos)
10	10	0,0025	0,0028
20	20	0,0035	0,0047
40	39	0,0055	0,0070
100	96	0,0086	0,0162
200	182	0,0162	0,0363
300	267	0,0215	0,1203
400	345	0,0279	0,4141
600	481	0,0390	1,3847

Tabela 4.2: Tempo gasto para encontrar o conjunto de vértices de um poliedro no $\mathbb{R}^2$.

Como podemos observar, o algoritmo `vertdual.m` é mais eficiente se o sistema de inequações $Ax \leq b$ possuir um número pequeno de restrições ($m < 30$ no $\mathbb{R}^2$ e $m < 20$ no $\mathbb{R}^3$). Para problemas maiores o algoritmo `con2vert.m` mostrou-se mais eficiente, porém devido ao fato de não possuir nenhuma rotina de pré-processamento, este método falhou em vários casos. Além disso, como já observamos, `con2vert.m` não permite descrever as faces do poliedro.

O desempenho um pouco mais lento observado no algoritmo `vertdual.m` é justamente devido a rotina interna que gera a lista de vértices que pertencem à uma mesma face. Para este procedimento é necessário verificar se um determinado vértice que está sendo adicionado à lista não é igual a algum ponto desta lista.

m	número de vértices	`con2vert.m`	`vertdual.m`
		Tempo (segundos)	Tempo (segundos)
10	16	0,0052	0,0031
20	36	0,0053	0,0067
40	76	0,0094	0,0121
100	196	0,0199	0,0325
200	393	0,0387	0,1156
300	592	0,0549	0,4559
400	786	0,0751	1,0310
600	1181	0,1100	3,3356
1000	1965	0,1841	15,010

Tabela 4.3: Tempo gasto para encontrar o conjunto de vértices de um poliedro no $\mathbb{R}^3$.

Como estamos interessados em, além de encontrar os vértices, determinar as faces do poliedro, o algoritmo `vertdual.m` é o mais indicado. Este algoritmo, além de possuir algumas rotinas de pré-processamento que o tornam mais robusto, permite identificar todos os vértices pertencentes a uma mesma face, o que nos possibilita desenhar cada face como um polígono qualquer e não somente de forma triangularizada.

Para finalizar, criamos uma rotina denominada `desenha.m` que reúne todas as sub-rotinas que permitem construir o arquivo `obj` que contém a lista de vértices e faces de um poliedro (em dimensão 2 ou 3) definido por um sistema de inequações lineares $Ax \leq b$. Os parâmetros de entrada da rotina `desenha.m` são apenas a matriz A e o vetor b, conforme podemos ver na implementação que sugerimos a seguir.

```
function [figura] = desenha(A,b)
[m,n]=size(A);
%Encontra um ponto interior ao poliedro
pint = pontointerior(A,b);
%Elimina redundâncias
[A, b] = eliminaredu(A,b,m,n,pint);
%Encontra o conjunto de vértices e as faces
[ V ] = vertdual(A,b,pint);
%Encontra o fecho e grava o arquivo figura.obj
grava_arquivo;
[ F ] = fecho(V,n);
%Abre a figura.obj no JavaView
!figura.obj
```

```
end
```

4.5 Aplicações

Para ilustrar o funcionamento da rotina `desenha.m` juntamente com o método de Fourier-Motzkin, vamos apresentar duas aplicações. Na primeira vamos projetar o cubo tridimensional centrado na origem de aresta igual a dois, no plano ortogonal ao vetor $(1,1,1)$ que passa pela origem.

Na segunda aplicação fazemos um procedimento análogo projetando o hipercubo do $\mathbb{R}^4$ centrado na origem de aresta igual a dois no sub-espaço ortogonal ao vetor $(1,1,1,1)$.

4.5.1 Aplicação 1

Seja $C_3 = \{(x_1, x_2, x_3) \in \mathbb{R}^3; |x_i| \leq 1\}$ o cubo tridimensional centrado na origem de aresta igual a dois. Este poliedro é definido pelo seguinte sistema de inequações lineares:

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} . \tag{4.1}$$

Para projetá-lo no subspaço ortogonal ao vetor $(1,1,1)$ utilizando o método de Fourier-Motzkin vamos efetuar uma rotação que leva o vértice $(1,1,1)$ no vetor $(0,0,\sqrt{3})$ e em seguida eliminar a variável x_3 do poliedro rotacionado.

O sistema que define o poliedro rotacionado é dado por:

$$\begin{bmatrix} 0.7071 & 0.4082 & 0.5774 \\ -0.7071 & -0.4082 & -0.5774 \\ -0.7071 & 0.4082 & 0.5774 \\ 0.7071 & -0.4082 & -0.5774 \\ 0 & -0.8165 & 0.5774 \\ 0 & 0.8165 & -0.5774 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} .$$

Utilizando o método de Fourier-Motzkin para eliminar a variável x_3 obtemos o sistema

$$\begin{bmatrix} -1.2247 & -2.1213 \\ -2.4495 & 0 \\ -1.2247 & 2.1213 \\ 1.2247 & -2.1213 \\ 2.4495 & 0 \\ 1.2247 & 2.1213 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 3.4641 \\ 3.4641 \\ 3.4641 \\ 3.4641 \\ 3.4641 \\ 3.4641 \end{bmatrix}.$$

Utilizando os dados do último sistema como parâmetros de entrada para a rotina `desenha.m`, obtemos a Figura 4.1, que corresponde a uma rotação do hexágono resultante da projeção do cubo C_3 apoiado no vértice $(1, 1, 1)$.

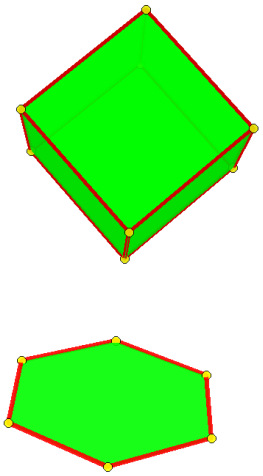


Figura 4.1: Projeção do cubo C_3 apoiado em um vértice.

4.5.2 Aplicação 2

Vamos proceder de maneira análoga a Aplicação 1 e projetar o hipercubo do $C_4 \subset \mathbb{R}^4$ no sub-espaço ortogonal ao vetor $(1, 1, 1, 1)$.

Seja $C_4 = \{(x_1, x_2, x_3, x_4) \in \mathbb{R}^4; |x_i| \leq 1\}$ o hipercubo de dimensão quatro centrado na origem de aresta igual a dois. Este poliedro é definido pelo seguinte sistema de inequações lineares:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Para projetá-lo no subspaço ortogonal ao vetor $(1, 1, 1, 1)$ utilizando o método de Fourier-Motzkin vamos efetuar uma rotação que leva o vértice $(1, 1, 1, 1)$ no vetor $(0, 0, 0, 2)$ e em seguida eliminar a variável x_4 do poliedro rotacionado.

O sistema que define o poliedro rotacionado é dado por:

$$\begin{bmatrix} 0.7071 & 0.4082 & 0.2887 & 0.5000 \\ -0.7071 & -0.4082 & -0.2887 & -0.5000 \\ -0.7071 & 0.4082 & 0.2887 & 0.5000 \\ 0.7071 & -0.4082 & -0.2887 & -0.5000 \\ 0 & -0.8165 & 0.2887 & 0.5000 \\ 0 & 0.8165 & -0.2887 & -0.5000 \\ 0 & 0 & -0.8660 & 0.5000 \\ 0 & 0 & 0.8660 & -0.5000 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Após a eliminação da variável x_4, utilizando o método de Fourier-Motzkin obtemos o sistema

$$\begin{bmatrix} -1.4142 & -2.4495 & 0 \\ -1.4142 & -0.8165 & -2.3094 \\ -2.8284 & 0 & 0 \\ -1.4142 & 0.8165 & 2.3094 \\ -1.4142 & 2.4495 & 0 \\ 0 & -1.6330 & 2.3094 \\ 0 & 1.6330 & -2.3094 \\ 1.4142 & -2.4495 & 0 \\ 1.4142 & -0.8165 & -2.3094 \\ 2.8284 & 0 & 0 \\ 1.4142 & 0.8165 & 2.3094 \\ 1.4142 & 2.4495 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \end{bmatrix},$$

cujo poliedro correspondente está representado na Figura 4.2 obtida através da rotina `desenha.m`.

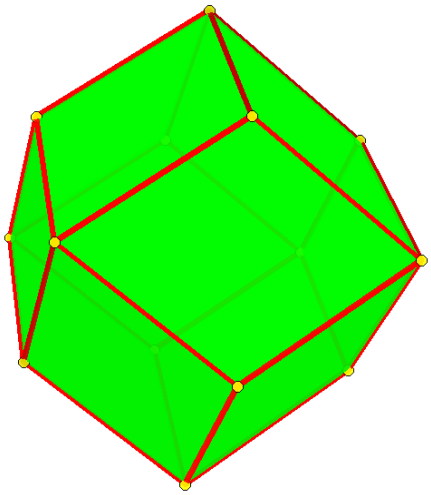


Figura 4.2: Projeção do hipercubo do $\mathbb{R}^4$ no subespaço ortogonal ao vetor $(1, 1, 1, 1)$.

Utilizando a rotina `desenha.m` podemos desenhar o conjunto solução de qualquer sistemas de inequações lineares com até três variáveis que seja não-vazio e limitado. Caso o poliedro seja ilimitado, o algoritmo `vertdual.m` não consegue listar os vértices do poliedro, pois este algoritmo utiliza o fecho do dual, que neste caso, não existe.

Considerações finais e perspectivas futuras

O foco central deste trabalho foi o estudo sobre sistemas de inequações lineares. Apresentamos o método de eliminação de Fourier-Motzkin, que permite projetar a região de solução de um sistema $Ax \leq b$ em um espaço de dimensão menor, através da eliminação de variáveis. Grande parte da investigação que procedemos concentrou-se no problema de desenhar a região de solução de um sistema de inequações lineares. Propomos uma rotina que possibilita desenhar esta região e visualizá-la em um software de geometria 3D através de um arquivo de descrição geométrica com extensão `obj`.

Durante o estudo, apresentamos alguns algoritmos para encontrar o fecho convexo de um conjunto de pontos no plano e no espaço tridimensional. Em seguida, provamos um importante teorema que nos possibilita transformar o problema de encontrar os vértices de um sistema de inequações lineares em um outro problema mais simples que é encontrar o fecho convexo do dual, que é dado por um conjunto finito de pontos. O grande problema na utilização desse teorema é encontrar um ponto interior de forma eficiente. Apresentamos algumas estratégias para contornar esse problema, utilizando um problema de programação linear e o método de eliminação de Fourier-Motzkin. No entanto, este problema é bastante complexo, sobretudo quando a dimensão e o número de restrições do sistema aumentam. Um estudo mais específico deste problema pode tornar o algoritmo mais eficiente.

Na parte experimental, comparamos a rotina que propusemos com a função `con2vert.m`, que está disponível no site do MatLab para o problema da enumeração de vértices. Apesar da função `con2vert.m` ser um pouco mais rápida quando o número de restrições cresce, destacamos algumas vantagens na utilização do algoritmo `vertdual.m` que propusemos. A principal delas é o fato de que este lista os vértices e também informa as restrições que geraram os vértices, o que facilita encontrarmos as faces do poliedro, enquanto que `con2vert.m` só lista os vértices. Além disso, nossa rotina inclui alguns

procedimentos de pré-processamento que permitem resolver casos onde o `con2vert.m` falha.

Ao longo do trabalho nos deparamos com diversas perguntas que ficaram em aberto e podem ser retomadas em estudos futuros. Por exemplo, além de melhorarmos o algoritmo `vertdual.m` como já foi mencionado, podemos melhorar o método para encontrar o fecho convexo, tornando-o mais eficiente. Também poderemos estudar como calcular o volume e a área de superfície dos politopos, como encontrar um ponto interior mais central e ainda comparar nosso algoritmo com algoritmos de grafos transversais, como o método de Avis & Fukuda [2].

Referências Bibliográficas

[1] Avis, D., & Bremner, D. How good are convex hull algorithms? *Computational Geometry: Theory and Applications 7* (1995), 265–301.

[2] Avis, D., & Fukuda, K. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. In *SCG '91: Proceedings of the seventh annual symposium on Computational geometry* (New York, NY, USA, 1991), ACM, pp. 98–104.

[3] Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. *Linear programming and network flows (2nd ed.)*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[4] Berkovitz, L. D. *Convexity and optimization in* $\mathbb{R}^n$. Pure and Applied Mathematics (New York). Wiley-Interscience [John Wiley & Sons], New York, 2002.

[5] Boldrini, J. L., Costa, S. I. R., Figueiredo, V. L., & Wetzler, H. G. *Álgebra Linear*, 3 ed. Harbra Ltda., 1980.

[6] Cartis, C., & M.Gould, N. I. Finding a point in the relative interior of a polyhedron. In *Council for the Central Laboratory of the Research Councils* (2006), CCLRC Rutherford Appleton Laboratory.

[7] Cartis, C., & Nicholas I. M., G. Finding a point in the relative interior of a polyhedron, with applications to compressed sensing. In *SPARS'09 - Signal Processing with Adaptive Sparse Structured Representations* (Saint Malo France, 2009), Rémi Gribonval, Ed., Inria Rennes - Bretagne Atlantique.

[8] Chand, D. R., & Kapur, S. S. An algorithm for convex polytopes. *J. ACM 17*, 1 (1970), 78–86.

[9] Dahl, G. Combinatorial properties of Fourier-Motzkin elimination. *Electron. J. Linear Algebra 16* (2007), 334–346 (electronic).

[10] de Berg, M., van Kreveld, M., Overmars, M., & Schwarzkopf, O. *Computational Geometry: Algorithms and Applications*, third ed. Springer-Verlag, 2008.

[11] Dyer, M. E. The complexity of vertex enumeration methods. *MATHEMATICS OF OPERATIONS RESEARCH 8*, 3 (August 1983), 381–402.

[12] Fukuda, K. Frequently asked questions in polyhedral computation. World Wide Web electronic publication, June 2004 (Consultada em Janeiro de 2010). http://www.ifor.math.ethz.ch/ fukuda/polyfaq/polyfaq.html.

[13] Gallier, J. Notes on Convex Sets, Polytopes, Polyhedra Combinatorial Topology, Voronoi Diagrams and Delaunay Triangulations. Research Report RR-6379, INRIA, 2007.

[14] Izmailov, A., & Solodov, M. *Otimização - Condições de Otimalidade, Elementos de Análise Convexa e de Dualidade*, pimeira ed., vol. 1. IMPA, 2005.

[15] Lauritzen, N. Lectures on convex sets. Notas de aula, Aarhus University: http://home.imf.au.dk/niels/lecconset.pdf, Março 2009.

[16] Preparata, F. P., & Shamos, M. I. *Computational geometry: an introduction.* Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[17] Pulino, P. *Álgebra Linear e suas Aplicações: Notas de aula.* Disponível em: http://www.ime.unicamp.br/ pulino/ALESA/Texto/, janeiro 2010.

[18] Schrijver, A. *Theory of linear and integer programming.* Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons Ltd., Chichester, 1986. A Wiley-Interscience Publication.

[19] Seidel, R. *Output-size sensitive algorithms for constructive problems in computational geometry.* PhD thesis, Cornell University, Ithaca, NY, USA, 1987.

Apêndice A

Implementação para encontrar um ponto interior

```
function [ pint ] = pontointerior(A,b)
[m,n]=size(A);

if min(b) > 10^(-15)
  pint = zeros(n,1);
else
  pintcand=A\b;
  if abs(max(A*(pintcand)-b)) > 10^(-15)
    pint = pintcand;
  else
    pint=pontointeriorPL(A,b);
  end
end

function [pint]=pontointeriorPL(A,b);
%eliminar os zeros
[m,n]=size(A);
An  = [A -b ones(m,1)];
An = [An;zeros(2,n+2)];
An(m+1,n+1)=-1;
```

```
An(m+2,n+2)=-1;
bn = [zeros(m,1);10^(-10);10^(-10)];
custo = [zeros(n,1);1;1];
size(An);
size(bn);
size(custo);
options = optimset('LargeScale','off','Simplex','on');
xx =  linprog(custo,An,bn,[],[],[],[],[],options);
pint = (1/(xx(n+1)))*xx;
pint([n+1,n+2])=[];
```

Apêndice B

Implementação do algoritmo para encontrar um ponto interior pelo método de eliminação de Fourier-Motzkin

```
function [ pint] = pontointeriorFM(A,b)
[m,n] = size(A);
if n == 2
  [Ax, bx] = procFM(A,b,[1]) ;
  ind_pos = find(Ax > 0);
  ind_neg = find(Ax < 0);
  ind_zero = find(Ax == 0);
  if min(bx(ind_zero)) < 0
    error('Sistema Inconsistente')
  end
  if isempty(ind_pos)
    disp('Poliedro ilimitado');
    xneg = bx(ind_neg)./Ax(ind_neg);
    x = max(xneg) + 3;
  end
  if isempty(ind_neg)
```

```
  disp('Poliedro ilimitado');
  xpos = bx(ind_pos)./Ax(ind_pos);
  x = min(xpos) - 3;
end
if isempty(ind_pos) & isempty(ind_neg);
  x = 0;
end
if ~isempty(ind_pos) & ~isempty(ind_neg);
  xpos = bx(ind_pos)./Ax(ind_pos);
  xneg = bx(ind_neg)./Ax(ind_neg);
  x = (max(xneg)+min(xpos))/2;
  if max(xneg) > min(xpos);
    error('Sistema Inconsistente');
  end
end
ind_pos = find(A(:,2) > 0);
ind_neg = find(A(:,2) < 0);
ind_zero = find(A(:,2) == 0);
if isempty(ind_pos)
  disp('Poliedro ilimitado');
  yneg = (b(ind_neg) - x*A(ind_neg,1))./A(ind_neg,2);
  y = max(yneg) + 3;
end
if isempty(ind_neg)
  disp('Poliedro ilimitado');
  ypos = (b(ind_pos) - x*A(ind_pos,1))./A(ind_pos,2);
  y = min(ypos) - 3;
end
if isempty(ind_pos) & isempty(ind_neg)
  y = 0;
end
if ~isempty(ind_pos) & ~isempty(ind_neg)
  ypos = (b(ind_pos) - x*A(ind_pos,1))./A(ind_pos,2);
  yneg = (b(ind_neg) - x*A(ind_neg,1))./A(ind_neg,2);
  y = (max(yneg)+min(ypos))/2;
```

```
    if max(yneg) > min(ypos)
      error('Sistema Inconsistente');
    end
  end
  pint = [x y]';
end

if n == 3
  [A1,b1] = procFM(A,b,[1 2]);
  [Ax, bx] = procFM(A1,b1,[1]);
  ind_pos = find(Ax > 0);
  ind_neg = find(Ax < 0);
  ind_zero = find(Ax == 0);
  if min(bx(ind_zero)) < 0
    error('Sistema Inconsistente');
  end
  if isempty(ind_pos)
    disp('Poliedro ilimitado');
    xneg = bx(ind_neg)./Ax(ind_neg);
    x = max(xneg) + 3;
  end
  if isempty(ind_neg)
    disp('Poliedro ilimitado');
    xpos = bx(ind_pos)./Ax(ind_pos);
    x = min(xpos) - 3;
  end
  if isempty(ind_pos) & isempty(ind_neg);
    x = 0;
  end
  if ~isempty(ind_pos) & ~isempty(ind_neg)
    xpos = bx(ind_pos)./Ax(ind_pos);
    xneg = bx(ind_neg)./Ax(ind_neg);
    x = (max(xneg)+min(xpos))/2;
    if max(xneg) > min(xpos)
      error('Sistema Inconsistente');
```

```
  end
end
ind_pos = find(A1(:,2) > 0);
ind_neg = find(A1(:,2) < 0);
ind_zero = find(A1(:,2) == 0);
if isempty(ind_pos);
  disp('Poliedro ilimitado');
  yneg = (b1(ind_neg) - x*A1(ind_neg,1))./A1(ind_neg,2);
  y = max(yneg) + 3;
end
if isempty(ind_neg)
  disp('Poliedro ilimitado');
  ypos = (b1(ind_pos) - x*A1(ind_pos,1))./A1(ind_pos,2);
  y = min(ypos) - 3;
end
if isempty(ind_pos) & isempty(ind_neg)
  y = 0;
end
if ~isempty(ind_pos) & ~isempty(ind_neg)
  ypos = (b1(ind_pos) - x*A1(ind_pos,1))./A1(ind_pos,2);
  yneg = (b1(ind_neg) - x*A1(ind_neg,1))./A1(ind_neg,2);
  y = (max(yneg)+min(ypos))/2;
  if max(yneg) > min(ypos)
    error('Sistema Inconsistente')
  end
end
ind_pos = find(A(:,3) > 0);
ind_neg = find(A(:,3) < 0);
ind_zero = find(A(:,3) == 0);
if isempty(ind_pos)
  disp('Poliedro ilimitado');
  zneg = (b(ind_neg) - x*A(ind_neg,1)- y*A(ind_neg,2))./A(ind_neg,3);
  z = max(zneg) + 3;
end
if isempty(ind_neg)
```

```
    disp('Poliedro ilimitado');
    zpos = (b(ind_pos) - x*A(ind_pos,1) - y*A(ind_pos,2))./A(ind_pos,3);
    z = min(zpos) - 3;
  end
  if isempty(ind_pos) & isempty(ind_neg)
    z = 0;
  end
  if ~isempty(ind_pos) & ~isempty(ind_neg)
    zpos = (b(ind_pos) - x*A(ind_pos,1) - y*A(ind_pos,2))./A(ind_pos,3);
    zneg = (b(ind_neg) - x*A(ind_neg,1)- y*A(ind_neg,2))./A(ind_neg,3);
    z = (max(zneg)+min(zpos))/2;
    if max(zneg) > min(zpos)
      error('Sistema Inconsistente');
    end
  end
  pint = [x y z]';
end
```

Apêndice C

Implementação para eliminar as redundâncias

```
function [A, b] = eliminaredu(A,b,m,n,pint)
if nargin < 3
  [m,n] = size(A);
end
if n == 2
  indzzero = find(A(:,1)==0&A(:,2)==0);
  if ~isempty(indzzero)
    if b(indzzero)>=0
      A(indzzero,:)=[];
      b(indzzero)=[];
    else
      error('Sistema infactivel');
    end
  end
elseif n ==3
  indzzero = find(A(:,1)==0&A(:,2)==0&A(:,3)==0);
  if ~isempty(indzzero)
    if b(indzzero)>=0
      A(indzzero,:)=[];
      b(indzzero)=[];
```

```
    else
      error('Sistema infactivel');
    end
  end
end
if nargin < 5
  pint = pontointerior(A,b);
end
%Transformando o sistema primal em um sistema dual
w = b - A*pint;
NA = A ./ repmat(w,[1 n]);
k = convhulln(NA)';
if n == 2
  I = unique([k(1,:) k(2,:)]);
end
if n == 3
  I = unique([k(1,:) k(2,:) k(3,:)]);
end
A = A(I,:);
b = b(I);
```

Apêndice D

Implementação do vertdual.m

```
function [ V ] = vertdual(A1,b1,pint)
[m1,n1]=size(A1);
eps = 10^(-10);
if n1 == 2
  indzzero = find(A1(:,1)==0&A1(:,2)==0);
  if ~isempty(indzzero)
    if b1(indzzero)>=0
      A1(indzzero,:)=[];
      b1(indzzero)=[];
    else
      error('Sistema infactivel');
    end
  end
elseif n1 ==3
  indzzero = find(A1(:,1)==0&A1(:,2)==0&A1(:,3)==0);
  if ~isempty(indzzero)
    if b1(indzzero)>=0
      A1(indzzero,:)=[];
      b1(indzzero)=[];
    else
      error('Sistema infactivel');
    end
```

```
  end
end
if nargin < 3
  pint = pontointerior(A1,b1);
end
w = b1 - A1*pint;
NA = A1 ./ repmat(w,[1 n1]);
k = convhulln(NA);
[mk,nk]=size(k);
V =zeros(mk,2*n1);
for i = 1:mk
  A = [A1(k(i,:),:)];
  b = [b1(k(i,:))];
  vertice = A\b;
  if n1 == 2
    V(i,:) = [vertice' k(i,1) k(i,2)];
  end
  if n1 == 3
    V(i,:) = [vertice' k(i,1) k(i,2) k(i,3)];
  end
end
V = sortrows(V);
[Vi Vj Vk]=unique(num2str(V(:,[1:3]),6),'rows');
kmax = max(Vk);
Vf = zeros(kmax,n1+m1);
for i = 1:kmax
  r=V(Vk==i,[n1+1:2*n1]);
  indices = unique(r(:))';
  t = V(Vk==i,[1:n1]);
  t = t(1,:);
  Vf(i,:) = [t,indices,zeros(1,m1-length(indices))];
end
V = sortrows(Vf);
eliminadas = 0;
[mv,nv]=size(V);
```

```
for j = 3:nv-eliminadas;
  ind = j - eliminadas;
  if all(V(:,ind)==0)
    V(:,ind) = [];
    eliminadas = eliminadas + 1;
  end
end
```

Apêndice E

Implementação do Algoritmo para o Fecho (ângulos entre vetores)

```
function [ F ] = fecho(V,dim) % V é o conjunto de vértices
clc
[mV,nV] = size(V)
fid = fopen('figura.obj', 'a');
%#para o R2
if dim == 2
  F = zeros(1,mV);
  Angulos = zeros(mV-3,2);
  W1 = [V(2,[1:2]) - V(1,[1:2])];
  NorW1 = norm(W1);
  for i = 3:mV
    W = [V(i,[1:2]) - V(1,[1:2])];
    A = acos(dot(W1,W)/( NorW1* norm(W)));
    Angulos(i-2,:) = [A,i];
  end
  Angulos = sortrows(Angulos);
  F(1) = 1;
  F(2) = 2;
  F([3:mV]) = Angulos(:,2);
  F = [F(1) F(2) F([3:mV])];
```

```
  fprintf(fid,'f ');
    for t = 1:length(F)
      fprintf(fid,'%d ',F(t));
    end
  fprintf(fid,'\n ');
end

%#para o R3
if dim == 3
  Vindice = V(:,[4:nV]);
  Imax = max(max(Vindice));
  for k = 1:Imax
    [i,j]=find(Vindice==k);
    if length(i) > 2
      Vertices = sortrows([V(i,[1:3]) i]);
      [m,n] = size(Vertices);
      F = zeros(1,m);
      Angulos = zeros(m-3,2);
      I = [Vertices(:,4)]';
      W1 = [V(I(1,2),[1:3]) - V(I(1,1),[1:3])];
      NorW1 = norm(W1);
      for y = 3:m
        W = [V(I(1,y),[1:3]) - V(I(1,1),[1:3])];
        NorW = norm(W);
        A = acos(dot(W1,W)/( NorW1* NorW));
        Angulos(y-2,:) = [A,I(1,y)];
      end
      Angulos = sortrows(Angulos);
      F(1) = I(1,1);
      F(2) = I(1,2);
      F([3:m]) = Angulos(:,2);
      F = [F(1) F(2) F([3:m])];
      fprintf(fid,'f ');
      for t = 1:length(F)
        fprintf(fid,'%d ',F(t));
```

```
      end
      fprintf(fid,'\n ');
    end
  end
end
fclose(fid);
%endfunction
```

Apêndice F

Implementação das classes de exemplos

Hipercubo rotacionado

```
function [A,b] = hipercubo(n,g)
A = zeros(2*n,n);
b = ones(2*n,1);
for i=1:n
  A(2*i-1,i) = 1;
  A(2*i,i) = -1;
end
%g == 0 Hipercubo sem rotação
if g==1 %Giro de 45ž
  for i=1:n-1
    roda = eye(n);
    c = 1/sqrt(i+1);
    s = sqrt(i)/sqrt(i+1);
    roda(i,i)=c;
    roda(i+1,i+1)=c;
    roda(i,i+1)=-s;
    roda(i+1,i)=s;
    A = (roda*A')';
  end
```

```
end
if g==2  %Giro aleatório
  for i=1:n
    for j = i+1:n
      roda = eye(n);
      ang = 2*pi*rand*(-1)^(round(rand*10));
      c = cos(ang);
      s = sin(ang);
      roda(i,i)=c;
      roda(j,j)=c;
      roda(i,j)=s;
      roda(j,i)=-s;
      A = (roda*A')';
    end
  end
end
```

Hiperplanos Tangentes a uma Hiperesfera

```
function [ A, b ] = gera (m,n)
%A = round(d*rand(m,n)+1);
A = zeros(m,n);
b = zeros(m,1);
for i = 1:m
  for j = 1:n
    A(i,j)=round(rand*(-1)^(round(rand*10))*40);
  end
end
A = sortrows(A);
for i=1:m
  b(i)=norm(A(i,:));
end
```