
Universidade Estadual de Campinas
Instituto de Matemática Estatística e Computação Científica
Departamento de Matemática Aplicada

**Descrição de rotações:
prós e contras na teoria e na prática**

Rodrigo Silva Lima*

Mestrado em Matemática Aplicada - Campinas - SP

Orientador: Profa. Dra. Margarida P. Mello

* Este trabalho teve apoio financeiro da CAPES.

Descrição de rotações: prós e contras na teoria e na prática

Este exemplar corresponde redação final da dissertação devidamente corrigida e defendida por **Rodrigo Silva Lima** e aprovada pela comissão julgadora.

Campinas, 23 de Fevereiro de 2007



Profa. Dra. Margarida P. Mello

Orientadora

Banca Examinadora

1. Prof. Dr. Ernesto Julián Goldberg Birgin (DCC/IME/USP)
2. Prof. Dr. José Mario Martínez (DMA/IMECC/Unicamp)
3. Profa. Dra. Margarida P. Mello (DMA/IMECC/Unicamp)

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica, Unicamp, como requisito parcial para obtenção do Título de MESTRE em Matemática Aplicada.

**FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DO IMECC DA UNICAMP**

Bibliotecária: Miriam Cristina Alves – CRB8a / 5094

Lima, Rodrigo Silva

L628d Descrição de rotações: prós e contras na teoria e na prática/Rodrigo Silva Lima -- Campinas, [S.P. :s.n.], 2007.

Orientadora: Margarida Pinheiro Mello

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Grupos de rotações. 2. Dinâmica dos corpos rígidos. 3. Programação não-linear. I. Mello, Margarida Pinheiro. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

(mca/imecc)

Título em inglês: Representation of rotations: advantages and disadvantages in theory and practice

Palavras-chave em inglês (Keywords): 1. Rotations groups. 2. Dynamic of rigid bodies. 3. Nonlinear programming.

Área de concentração: Otimização

Titulação: Mestre em Matemática Aplicada


Banca examinadora: Profª. Dra. Margarida Pinheiro Mello (IMECC/Unicamp)
Prof. Dr. José Mario Martínez (IMECC/Unicamp)
Prof. Dr. Ernesto Julián Goldberg Birgin (IME/USP)

Data da defesa: 23/02/2007

Programa de Pós-Graduação: Mestrado em Matemática Aplicada

Dissertação de Mestrado defendida em 23 de fevereiro de 2007 e aprovada

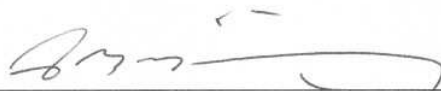
Pela Banca Examinadora composta pelos Profs. Drs.



Prof. (a). Dr (a). MARGARIDA PINHEIRO MELLO



Prof. (a). Dr (a). ERNESTO JULIÁN GOLDBERG BIRGIN



Prof. (a). Dr (a). JOSÉ MÁRIO MARTÍNEZ PÉREZ

Nada será como antes

Eu já estou com o pé nessa estrada
Qualquer dia a gente se vê
Sei que nada será como antes amanhã
Que notícias me dão dos amigos?
Que notícias me dão de você?
Alvorço em meu coração
Amanhã ou depois de amanhã
Existindo na boca da noite um gosto de sol

Num domingo qualquer, qualquer hora
Ventania em qualquer direção
Sei que nada será como antes amanhã
Que notícias me dão dos amigos?
Que notícias me dão de você?
Sei que nada será como está
Amanhã ou depois de amanhã
Existindo na boca da noite um gosto de sol

(Milton Nascimento - Ronaldo Bastos)

Agradecimentos

Agradeço:

A Deus por estar sempre comigo e por me iluminar durante a realização deste trabalho.

Aos meus pais Getúlio e Bernadete, ao meu irmão Rogério e à “quase-irmã” Graça pelo apoio incondicional que me deram ao longo dessa jornada.

À inesquecível professora Margarida Mello, pela amizade e pela excelente orientação desde o meu primeiro trabalho de iniciação científica até agora.

Ao professor José Mario Martínez pelas valiosas dicas durante minhas apresentações nos seminários de otimização.

Ao professor Roberto Andreani por ter me emprestado um computador para que eu pudesse concluir esse trabalho.

Ao químico Leandro Martínez pela ajuda com o software *packmol*.

À Fátima, Tânia, Cidinha e Ednaldo por terem me ajudado sempre que precisei.

Aos meus grandes amigos de Campinas e de Brazópolis, que sempre torceram pelo meu sucesso.

À CAPES pelo apoio financeiro.

Resumo

Robótica, computação gráfica, aeronáutica e biomecânica têm em comum o estudo de movimentos rígidos, aqueles compostos por rotações e translações. Ao contrário das translações, cuja representação matemática não apresenta dificuldades, as rotações trazem consideráveis desafios. Neste trabalho de mestrado estudamos algumas propriedades do grupo de rotações $SO(3)$ bem como três maneiras de se representar uma rotação: via mapa exponencial, via ângulos de Euler e via quatérnions unitários. Como as duas últimas representações são as mais comuns na literatura, fazemos uma comparação do uso de ângulos de Euler e quatérnions unitários na formulação e resolução de dois problemas concretos: o problema do empacotamento de moléculas e o problema da orientação absoluta.

Palavras-chave: descrição de rotações, ângulos de Euler, quatérnions, orientação absoluta, empacotamento de moléculas.

Abstract

Robotics, graphics computation, aeronautics and biomechanics have in common the study of rigid motions, composed by rotations and translations. The translations are mathematically simple, but the treatment of rotations raises some difficulties. In this work we study some properties of rotation group $SO(3)$ and three commonly used methods to describe rotations, namely, the exponential map, Euler angles and unitary quaternions. We employ the latter two in the formulation of two optimization problems: the absolute orientation problem and the molecule packing problem. Several computational experiments are performed in order to establish the relative efficiency of the two representations in these examples. In the first optimization problem there is a tie, whereas in the molecule packing problem the representation using quaternions is slightly superior.

Keywords: representation of rotations, Euler angles, quaternions, absolute orientation, molecule packing.

Sumário

1	Introdução	1
2	Rotações	3
2.1	Rotações no plano	3
2.1.1	Rotações no plano e números complexos	5
2.2	Rotações no espaço	6
2.2.1	Movimentos rígidos	8
2.3	Algumas propriedades do conjunto $SO(3)$	11
3	Descrição de rotações em \mathbb{R}^3	13
3.1	Parametrizações para $SO(3)$	13
3.2	O mapa exponencial	16
3.3	Ângulos de Euler	20
3.3.1	Limitações no uso dos ângulos de Euler	22
3.4	Quatérnions	23
3.4.1	Um pouco de história	23
3.4.2	Álgebra de quatérnions	24
3.4.3	Descrição de rotações via quatérnions	26
3.5	Comparando as representações	29
3.5.1	Fórmulas de conversão	29
3.5.2	Eficiência computacional	31
4	Aplicações	35
4.1	O problema da orientação absoluta	35
4.1.1	Formulação e resolução do problema	36
4.1.2	Experimentos	44
4.2	Problema do empacotamento de moléculas	50
4.2.1	Formulação do problema	50
4.2.2	Representação das rotações envolvidas	52
4.2.3	O programa <i>packmol</i>	55
4.2.4	Experimentos	61
5	Conclusões	78

A Implementação do Algoritmo 4.1

80

Lista de Tabelas

3.1	Seqüências de Euler.	20
3.2	Regras de multiplicação inventadas por Hamilton.	24
3.3	Comparação de uso de memória entre as representações.	32
3.4	Comparação do número de operações realizadas.	33
3.5	Número de operações necessárias para rodar um vetor.	33
4.1	Problema dos vértices da pirâmide.	45
4.2	Problema dos vértices da pirâmide - conjunto V_d contém erros.	46
4.3	Problema dos 20 pontos gerados aleatoriamente.	47
4.4	Problema dos 20 pontos: conjunto V_d contém erros.	48
4.5	Comportamento do pacote ALGENCAN na resolução dos testes.	49
4.6	Principais características de cada formulação.	55
4.7	Desempenho de PB e PG no empacotamento de moléculas de água.	62
4.8	Desempenho de PBQ e PGQ no empacotamento de moléculas de água.	62
4.9	Desempenho de PGQP e PALQ no empacotamento de moléculas de água.	63
4.10	Valor da função objetivo na solução final.	63
4.11	Desempenho de PB e PG no empacotamento de moléculas de uréia.	65
4.12	Desempenho de PBQ e PGQ no empacotamento de moléculas de uréia.	65
4.13	Desempenho de PGQP e PALQ no empacotamento de moléculas de uréia.	66
4.14	Valor da função objetivo na solução final.	66
4.15	Quantidades de moléculas em cada mistura.	67
4.16	Função objetivo na solução final.	67
4.17	Desempenho dos programas no empacotamento de misturas.	68
4.18	Tempos de execução nos problemas das moléculas inventadas.	70
4.19	Descrição dos experimentos com recipientes de formatos diferentes.	71
4.20	Tempos de execução dos testes com recipientes diferentes.	72
4.21	Tempos obtidos nos testes com hessiana verdadeira.	76

Lista de Figuras

2.1	Duas maneiras de caracterizar uma rotação no plano.	4
2.2	Representação de z no plano complexo.	6
2.3	Sistemas inercial (S_1) e rodado (S_2).	7
2.4	Objeto antes e depois de um movimento rígido.	10
3.1	Cartas em uma variedade M arbitrária.	14
3.2	Rotação de \mathbf{u} em torno do eixo gerado por $\boldsymbol{\omega}$	17
3.3	Seqüência ZXZ aplicada a um objeto.	21
3.4	<i>Gimbal Lock</i> : dois eixos de rotação se coincidem.	23
3.5	Correspondência biunívoca entre \mathbb{R}^3 e \mathbb{H}_0	26
3.6	Interpretando geometricamente a ação do operador L	28
4.1	Representação de um ponto com relação aos sistemas S_e e S_d	36
4.2	Pirâmide antes e após a aplicação da transformação T	44
4.3	Empacotamento de moléculas em uma caixa.	51
4.4	Exemplos de arquivos de leitura.	56
4.5	Principais fases do programa <i>packmol</i>	57
4.6	Principais passos das versões PBQ, PGQ, PGQP e PALQ.	60
4.7	Comparando tempos no empacotamento de moléculas de água.	64
4.8	Comparando tempos no empacotamento de moléculas de uréia.	66
4.9	Comparando tempos no empacotamento de misturas.	68
4.10	Moléculas inventadas.	69
4.11	Comparando tempos no empacotamento de moléculas inventadas.	71
4.12	Comparando tempos nos testes com recipientes de formatos diferentes.	72
4.13	Perfis dos programas nos empacotamentos em caixas.	74
4.14	Perfis desconsiderando testes com molécula estrela.	74
4.15	Perfis dos programas que usam a rotina GENCAN.	75
4.16	Estrutura da matriz hessiana nos testes com moléculas de água.	76
4.17	Estrutura da matriz hessiana em testes com moléculas de uréia.	77

Notações utilizadas

$\mathbf{u}^T \in \mathbb{R}^{1 \times n}$ denota o transposto do vetor $\mathbf{u} \in \mathbb{R}^{n \times 1}$.

$\mathbf{u}^T \mathbf{v} = \langle \mathbf{u}, \mathbf{v} \rangle$ é o produto interno euclidiano entre \mathbf{u} e \mathbf{v} .

$\mathbf{u} \times \mathbf{v}$ é o produto vetorial entre \mathbf{u} e \mathbf{v} .

$\|\cdot\|$ representa a norma euclidiana de vetores e matrizes.

$I \in \mathbb{R}^{3 \times 3}$ é a matriz identidade de ordem 3.

$\nabla f \in \mathbb{R}^{n \times 1}$ é o vetor gradiente da função $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

$\nabla^2 f \in \mathbb{R}^{n \times n}$ é a matriz hessiana da função $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

Números em notação científica: $1.2\text{E-}09 = 1,2 \times 10^{-9}$.

$\text{tg}^{-1}(a, b) = \arctan\left(\frac{b}{a}\right)$, $a \neq 0$.

Capítulo 1

Introdução

Robótica, computação gráfica, aeronáutica, biomecânica têm em comum o estudo de movimentos rígidos, aqueles compostos por translações e rotações. Ao contrário das translações, cuja representação matemática não apresenta dificuldades, as rotações trazem consideráveis desafios. Neste trabalho estudamos a teoria por trás de alguns tipos de representações de rotações, resumindo as principais características, vantagens e desvantagens de cada representação. Em particular, estudamos como representar uma rotação através do mapa exponencial, dos ângulos de Euler e dos quatérnions unitários. Fazemos também uma comparação da utilização de quatérnions e ângulos de Euler na modelagem e resolução de dois problemas concretos.

Este trabalho está organizado da seguinte forma:

- **Capítulo 2:** Começamos este capítulo caracterizando as rotações no plano. Mostramos que existem duas maneiras de caracterizar uma rotação em \mathbb{R}^2 e fazemos uma analogia entre rotações no plano e números complexos. Em seguida, passamos ao estudo das rotações no espaço apresentando o grupo de rotações $SO(3)$. Definimos o que é um movimento rígido em \mathbb{R}^3 e provamos algumas propriedades do conjunto $SO(3)$.
- **Capítulo 3:** Neste capítulo introduzimos os conceitos de cartas e parametrizações em uma variedade arbitrária. Utilizando essas definições provamos com detalhes o conhecido fato de que não é possível cobrir o conjunto $SO(3)$ com apenas uma carta. Com esse resultado, concluímos que é preciso restringir o domínio de definição dos ângulos para que a representação de rotações via ângulos de Euler seja considerada uma parametrização para $SO(3)$. Em seguida, mostramos como representar uma rotação no espaço utilizando o mapa exponencial, os ângulos de Euler e os quatérnions unitários e por fim, fazemos uma discussão sobre a eficiência computacional de cada uma dessas representações.
- **Capítulo 4:** Estudamos neste capítulo dois problemas de otimização que envolvem rotações no espaço tridimensional: o problema da orientação absoluta e o problema do empacotamento de moléculas. Em ambos os problemas as rotações foram caracterizadas, inicialmente, via ângulos de Euler, cf. [22, 32]. Utilizamos também quatérnions

para descrever as rotações, obtendo dessa forma uma formulação alternativa para os dois problemas.

Resolvemos o primeiro problema por um método numérico iterativo utilizando-se ambas as formulações, e contrastamos os resultados obtidos com a solução analítica obtida por [13] para o problema formulado com quatérnions. No segundo problema, utilizamos quatérnions para representar as rotações e comparamos a eficiência do método empregado para resolvê-lo nas duas formulações.

O problema de empacotamento consiste em distribuir um conjunto de moléculas em um recipiente de modo que a distância entre pares de átomos de moléculas distintas não seja menor que uma tolerância $\epsilon > 0$. Este problema de viabilidade pode ser modelado como um problema de otimização, minimizando-se uma função objetivo que penaliza a existência de átomos demasiadamente próximos ou fora do recipiente. Uma solução ótima para o problema de otimização com função objetivo nula é então uma solução viável para o problema de empacotamento.

Já o problema da orientação absoluta consiste em recuperar a relação entre dois sistemas 3D de coordenadas S_1 e S_2 a partir das coordenadas cartesianas de um conjunto de n pontos medidas com relação a cada sistema. Ou seja, dados os vetores \mathbf{r}_1^i e \mathbf{r}_2^i com as coordenadas do i -ésimo ponto medidas em S_1 e S_2 , respectivamente, procuramos por uma transformação afim $T : S_1 \rightarrow S_2$ tal que

$$T(\mathbf{r}_1^i) = sR\mathbf{r}_1^i + \mathbf{d}_0 = \mathbf{r}_2^i, \quad \forall \quad i = 1, \dots, n,$$

onde s é um fator de escala, R é uma matriz de rotação e \mathbf{d}_0 é um vetor de deslocamento. Definindo para cada i o vetor resíduo $e_i = \mathbf{r}_2^i - T(\mathbf{r}_1^i)$, construímos um problema de quadrados mínimos não-linear: escolher s , R e \mathbf{d}_0 que minimizam a soma das normas ao quadrado dos resíduos.

- **Capítulo 5:** Neste capítulo final fazemos uma breve discussão dos resultados obtidos ao logo do trabalho e indicamos assuntos que daremos continuidade em pesquisa futura.
- **Apêndice:** No apêndice estão detalhes da implementação do algoritmo utilizado para obter a solução analítica do problema da orientação absoluta.

Capítulo 2

Rotações

As rotações desempenham um papel fundamental em vários ramos das ciências exatas tais como robótica, computação gráfica, química e biomecânica. Essa importância se deve sobretudo ao fato de que a maioria dos movimentos que realizamos ou presenciamos no dia-a-dia podem ser descritos utilizando rotações. Veremos neste capítulo como representar matematicamente uma rotação e as propriedades que esta representação satisfaz. Começaremos estudando rotações no plano e em seguida trataremos de rotações no espaço tridimensional.

2.1 Rotações no plano

Seja XY um sistema de coordenadas em \mathbb{R}^2 formado pelos vetores canônicos $\mathbf{e}_1 = (1, 0)^T$ e $\mathbf{e}_2 = (0, 1)^T$. Considere $\mathbf{v}_1 = (x_1, y_1)^T$ um vetor arbitrário representado no sistema XY , tal que o ângulo entre \mathbf{v}_1 e o vetor \mathbf{e}_1 é α . Sendo $r = \|\mathbf{v}_1\|$, temos que

$$\begin{aligned}x_1 &= r \cos \alpha, \\y_1 &= r \sin \alpha.\end{aligned}\tag{2.1}$$

Rotações ocorrem em dois contextos, ou pontos de vista, ilustrados nas Figuras 2.1 (a) e 2.1 (b). No primeiro caso (Figura 2.1 (a)) mantemos \mathbf{v}_1 fixo e rodamos o sistema XY no sentido anti-horário de um ângulo θ obtendo dessa forma o sistema de coordenadas $X'Y'$ formado pelos vetores $\mathbf{e}'_1, \mathbf{e}'_2$. Assim, o vetor \mathbf{v}_1 tem coordenadas diferentes, o vetor \mathbf{v}_2 , no sistema $X'Y'$, dadas por

$$\begin{aligned}x_2 &= r \cos(\alpha - \theta) = r \cos \alpha \cos \theta + r \sin \alpha \sin \theta, \\y_2 &= r \sin(\alpha - \theta) = r \sin \alpha \cos \theta - r \cos \alpha \sin \theta.\end{aligned}\tag{2.2}$$

Substituindo (2.1) em (2.2), podemos escrever:

$$\begin{aligned}x_2 &= x_1 \cos \theta + y_1 \sin \theta, \\y_2 &= y_1 \cos \theta - x_1 \sin \theta.\end{aligned}\tag{2.3}$$

No caso ilustrado na Figura 2.1 (b) mantemos o sistema XY fixo e rodamos \mathbf{v}_1 no sentido anti-horário de tal forma que o ângulo entre o vetor rodado \mathbf{v}_2 e o vetor \mathbf{v}_1 seja θ . Neste caso, as coordenadas de \mathbf{v}_2 com relação ao sistema XY são dadas por

$$\begin{aligned} x_2 &= r \cos(\alpha + \theta) = r \cos \alpha \cos \theta - r \sin \alpha \sin \theta, \\ y_2 &= r \sin(\alpha + \theta) = r \sin \alpha \cos \theta + r \cos \alpha \sin \theta. \end{aligned} \quad (2.4)$$

e substituindo as equações (2.1) em (2.4) obtemos

$$\begin{aligned} x_2 &= x_1 \cos \theta - y_1 \sin \theta, \\ y_2 &= y_1 \cos \theta + x_1 \sin \theta. \end{aligned} \quad (2.5)$$

De acordo com (2.3) e (2.5) podemos concluir que as duas maneiras de caracterizar a rotação de um vetor no plano são distintas. Porém, o efeito de manter o vetor \mathbf{v}_1 fixo e rodar o sistema XY no sentido anti-horário de um ângulo θ é exatamente o mesmo que manter o sistema XY fixo e rodar \mathbf{v}_1 no sentido horário de um ângulo $-\theta$. De fato, substituindo $-\theta$ nas equações (2.5), chegamos ao mesmo resultado dado em (2.3):

$$\begin{aligned} x_2 &= x_1 \cos(-\theta) - y_1 \sin(-\theta) = x_1 \cos \theta + y_1 \sin \theta, \\ y_2 &= y_1 \cos(-\theta) + x_1 \sin(-\theta) = y_1 \cos \theta - x_1 \sin \theta, \end{aligned} \quad (2.6)$$

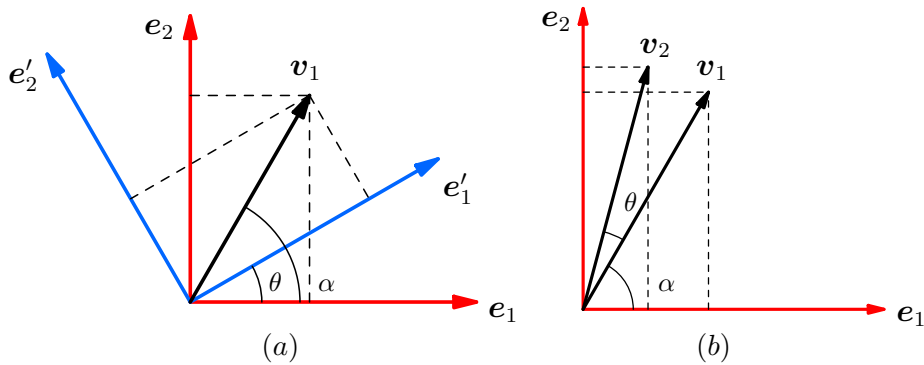


Figura 2.1: Duas maneiras de caracterizar uma rotação no plano.

Em ambas as caracterizações apresentadas anteriormente a relação obtida entre os vetores \mathbf{v}_1 e \mathbf{v}_2 pode ser reescrita em notação matricial. Deste modo, podemos escrever respectivamente (2.3) e (2.5) como

$$\mathbf{v}_2 = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = R_1 \mathbf{v}_1, \quad (2.7)$$

$$\mathbf{v}_2 = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = R_2 \mathbf{v}_1. \quad (2.8)$$

As matrizes R_1 e R_2 que aparecem nas equações (2.7) e (2.8) são denominadas *matrizes de mudança de base*, pois R_1 leva o sistema XY no sistema $X'Y'$ e R_2 faz o contrário, ou seja,

transforma o sistema $X'Y'$ no sistema XY . Como XY e $X'Y'$ são formados por vetores ortonormais, as matrizes R_1 e R_2 são ortogonais. Logo, satisfazem:

$$\begin{aligned} R_1^{-1} &= R_1^T, \\ R_2^{-1} &= R_2^T. \end{aligned} \tag{2.9}$$

Além disso, é fácil verificar que

$$\det(R_1) = \det(R_2) = 1. \tag{2.10}$$

Por satisfazerem as propriedades (2.9) e (2.10), R_1 e R_2 recebem o nome de *matrizes de rotação*. O conjunto de todas as matrizes de rotação de ordem 2 é denotado por $SO(2)$. A sigla SO significa *special orthogonal* e o adjetivo *special* vem do fato do determinante de uma matriz em $SO(2)$ ser igual a 1. Convém lembrar que as matrizes ortogonais de ordem 2 com determinante -1 correspondem a reflexões, movimento inadmissível para as moléculas de um corpo rígido. A seguir, veremos a relação existente entre rotações no plano e números complexos.

2.1.1 Rotações no plano e números complexos

Inicialmente vamos relembrar algumas definições da álgebra de números complexos.

Definição 2.1 *Seja $z = a + ib \in \mathbb{C}$, com $a, b \in \mathbb{R}$, um número complexo arbitrário. A norma e o argumento de z são respectivamente definidos por*

$$\begin{aligned} |z| &= \sqrt{a^2 + b^2}, \\ \theta &= \operatorname{tg}^{-1}(a, b), \quad a \neq 0. \end{aligned} \tag{2.11}$$

De acordo com a Figura 2.2, podemos pensar geometricamente que a norma de z é o comprimento do vetor $(a, b)^T \in \mathbb{R}^2$ e o argumento de z é o ângulo que esse vetor faz com o eixo X . Assim, podemos reescrever z na chamada *forma polar* ou *trigonométrica*

$$z = r(\cos \theta + i \operatorname{sen} \theta), \quad \text{onde } r = |z|. \tag{2.12}$$

Seja $w = x_1 + iy_1 \in \mathbb{C}$ também arbitrário. Lembrando que $i^2 = -1$ e considerando z na forma dada em (2.12), ao efetuar o produto zw da maneira usual obtemos outro número complexo dado por

$$zw = r [(x_1 \cos \theta - y_1 \operatorname{sen} \theta) + i(x_1 \operatorname{sen} \theta + y_1 \cos \theta)]. \tag{2.13}$$

Denotando os números complexos w e zw respectivamente por $(x_1, y_1)^T$ e $(x_2, y_2)^T$, a equação (2.13) reescrita em notação matricial assume a forma

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = r \begin{pmatrix} \cos \theta & -\operatorname{sen} \theta \\ \operatorname{sen} \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}. \tag{2.14}$$

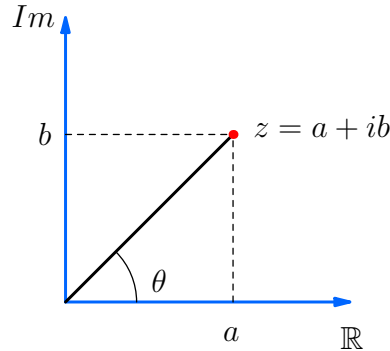


Figura 2.2: Representação de z no plano complexo.

Se em (2.14) fizermos $r = 1$ obteremos uma expressão idêntica à equação (2.8) vista anteriormente. Isto significa que se z for um número complexo de norma unitária ($|z| = 1$), o produto zw é a rotação do número complexo w de um ângulo θ no sentido anti-horário. Parece então razoável concluir que um número complexo unitário pode ser identificado com uma matriz de rotação de ordem 2. De fato, é possível provar que existe uma aplicação bijetora que associa o conjunto de todos os números complexos unitários (denotado por S^1) ao conjunto $SO(2)$ (veja, por exemplo, [6, 21]).

2.2 Rotações no espaço

Para descrever rotações em \mathbb{R}^3 precisamos antes definir o que é um sistema de coordenadas cartesianas orientado segundo a regra da mão direita.

Definição 2.2 *Seja S um sistema de coordenadas cartesianas em \mathbb{R}^3 formado pelos vetores ortonormais $\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$. Dizemos que S é orientado segundo a regra da mão direita se*

$$\begin{aligned}\mathbf{z} &= \mathbf{x} \times \mathbf{y}, \\ \mathbf{x} &= \mathbf{y} \times \mathbf{z}, \\ \mathbf{y} &= \mathbf{z} \times \mathbf{x}.\end{aligned}$$

O exemplo mais comum de um sistema com essa propriedade é aquele formado pelos vetores canônicos de \mathbb{R}^3 : $\mathbf{e}_1 = (1, 0, 0)^T$, $\mathbf{e}_2 = (0, 1, 0)^T$ e $\mathbf{e}_3 = (0, 0, 1)^T$. De agora em diante assumiremos que todos os sistemas de coordenadas em \mathbb{R}^3 são orientados segundo a regra da mão direita.

Veremos agora como caracterizar uma rotação no espaço. Sejam S_1 e S_2 sistemas de coordenadas cartesianas em \mathbb{R}^3 (cujas origens coincidam) formados, respectivamente, pelos conjuntos de vetores ortonormais $\{\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1\}$ e $\{\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_2\}$ como mostra a Figura 2.3. Nosso objetivo é construir uma matriz de mudança de base 3×3 que relaciona os dois sistemas de coordenadas, isto é, que descreva a orientação de um sistema com relação ao outro. Dado $\mathbf{v} \in \mathbb{R}^3$ arbitrário, denotaremos suas coordenadas com relação ao sistema S_1 por $[\mathbf{v}]_{S_1}$. Então,

$$[\mathbf{v}]_{S_1} = (a_1, a_2, a_3)^T \Rightarrow [\mathbf{v}]_{S_2} = a_1\mathbf{x}_2 + a_2\mathbf{y}_2 + a_3\mathbf{z}_2.$$

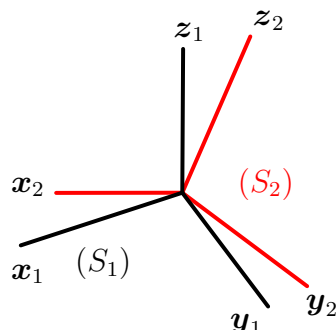


Figura 2.3: Sistemas inercial (S_1) e rodado (S_2) .

Da mesma forma, representando no sistema S_1 cada vetor que forma o sistema S_2 obtemos os seguintes vetores

$$\mathbf{r}_1 = [\mathbf{x}_2]_{S_1}, \quad \mathbf{r}_2 = [\mathbf{y}_2]_{S_1}, \quad \mathbf{r}_3 = [\mathbf{z}_2]_{S_1}. \quad (2.15)$$

Os vetores (2.15) são também ortonormais, isto é

$$\mathbf{r}_i^T \mathbf{r}_j = \begin{cases} 0, & \text{se } i \neq j \\ 1, & \text{se } i = j \end{cases}. \quad (2.16)$$

Com eles podemos definir a seguinte matriz

$$R = \left(\mathbf{r}_1 \mid \mathbf{r}_2 \mid \mathbf{r}_3 \right). \quad (2.17)$$

A matriz (2.17) é denominada *matriz de rotação de ordem 3*. Ela descreve a orientação do sistema S_2 com relação ao sistema S_1 e quando aplicada a um vetor $\mathbf{v} \in \mathbb{R}^3$, tem o efeito de rodar este vetor com relação a S_1 . Esta matriz satisfaz as mesmas propriedades que as matrizes de rotação de ordem 2, ou seja,

$$\begin{aligned} R^{-1} &= R^T, \\ \det(R) &= +1. \end{aligned} \quad (2.18)$$

A primeira propriedade em (2.18) é consequência direta do fato de que as colunas da matriz R são vetores ortonormais, pois, escrevendo (2.16) em termos de R obtemos

$$R^T R = I = R R^T. \quad (2.19)$$

Para verificar a segunda equação em (2.18) basta considerarmos uma das igualdades em (2.19) e aplicarmos algumas propriedades de determinantes. Deste modo:

$$\begin{aligned} \det(R^T R) = \det(I) &\Rightarrow \det(R^T) \det(R) = 1 \\ &\Rightarrow (\det(R))^2 = 1 \\ &\Rightarrow \det(R) = \pm 1. \end{aligned}$$

Escrevendo o determinante como um produto misto dos vetores coluna de R temos

$$\det(R) = \mathbf{r}_1^T(\mathbf{r}_2 \times \mathbf{r}_3).$$

Lembrando que os sistemas de coordenadas são orientados segundo a regra da mão direita e que os vetores $\mathbf{r}_i, i = 1, 2, 3$, são ortonormais, segue que $\mathbf{r}_2 \times \mathbf{r}_3 = \mathbf{r}_1$ e

$$\det(R) = \mathbf{r}_1^T(\mathbf{r}_2 \times \mathbf{r}_3) = \mathbf{r}_1^T \mathbf{r}_1 = +1.$$

Na Seção 2.1 vimos que rotações no plano podem ser pensadas de duas formas distintas. O mesmo acontece com rotações no espaço, ou seja, dado um objeto representado em um sistema de coordenadas tridimensional, podemos pensar que uma rotação aplicada a este objeto é obtida por:

- (a) fixar o objeto e rodar o sistema de coordenadas,

ou

- (b) manter o sistema de coordenadas fixo e rodar o objeto.

É claro que tanto em (a) quanto em (b) as matrizes de rotação serão distintas. Não existe na literatura um padrão a ser seguido quanto a escolha de uma interpretação para uma rotação no espaço. Por exemplo, [18, 21] caracterizam uma rotação de acordo com a maneira (a), enquanto [10, 24, 28] preferem a maneira (b). Por acharmos o caso (b) mais intuitivo, passaremos a segui-lo de agora em diante.

O conjunto de todas as matrizes em $\mathbb{R}^{3 \times 3}$ que satisfazem as propriedades (2.18) é denotado por $SO(3)$. Veremos na Seção 2.3 que este conjunto constitui um *grupo* com relação a operação de multiplicação de matrizes e estudaremos também algumas de suas propriedades topológicas.

2.2.1 Movimentos rígidos

Um movimento rígido de um objeto qualquer é um movimento das partículas que constituem o objeto de tal modo que a distância e a orientação relativa entre quaisquer duas dessas partículas são preservadas durante o movimento. Podemos definir um movimento rígido tanto no plano quanto no espaço. Porém, neste trabalho daremos enfoque ao caso tridimensional. Um movimento rígido em \mathbb{R}^3 pode ser descrito matematicamente por uma aplicação $g : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ (denominada *transformação de corpo rígido*) que satisfaz as seguintes propriedades para todo $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$:

$$\begin{aligned} \|g(\mathbf{u})\| &= \|\mathbf{u}\|, \\ g(\mathbf{u} \times \mathbf{v}) &= g(\mathbf{u}) \times g(\mathbf{v}). \end{aligned} \tag{2.20}$$

A primeira propriedade em (2.20) diz que a aplicação g preserva o comprimento de vetores, enquanto a segunda assegura que g preserva a orientação.

Uma matriz de rotação $R \in SO(3)$ define uma transformação de corpo rígido em \mathbb{R}^3 , isto é, a aplicação

$$\begin{aligned} g: \mathbb{R}^3 &\rightarrow \mathbb{R}^3 \\ \mathbf{v} &\mapsto R\mathbf{v} \end{aligned} \quad (2.21)$$

satisfaz (2.20). Para demonstrarmos essa afirmação, precisamos recordar uma propriedade algébrica do produto vetorial e demonstrar um lema auxiliar.

Dados $\mathbf{u} = (a_1, a_2, a_3)^T$ e $\mathbf{v} = (b_1, b_2, b_3)^T$, o produto vetorial $\mathbf{u} \times \mathbf{v}$ pode ser escrito da seguinte forma

$$\mathbf{u} \times \mathbf{v} = \widehat{\mathbf{u}} \mathbf{v}. \quad (2.22)$$

onde $\widehat{\mathbf{u}} \in \mathbb{R}^{3 \times 3}$ é a matriz anti-simétrica formada com as componentes do vetor \mathbf{u} , dada por

$$\widehat{\mathbf{u}} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}. \quad (2.23)$$

Utilizaremos a identidade (2.22) para demonstrar o seguinte lema:

Lema 2.1 *Dado $R \in SO(3)$ e $\mathbf{u} \in \mathbb{R}^3$ temos que $R\widehat{\mathbf{u}}R^T = \widehat{R\mathbf{u}}$, onde $\widehat{\mathbf{u}}$, $\widehat{R\mathbf{u}} \in \mathbb{R}^{3 \times 3}$ são matrizes anti-simétricas construídas, respectivamente, com as componentes dos vetores \mathbf{u} e $R\mathbf{u}$.*

Demonstração: Sejam $\mathbf{p}_i \in \mathbb{R}^3$, $i = 1, 2, 3$, os vetores coluna de R^T . Pré-multiplicando R^T pela matriz $\widehat{\mathbf{u}}$, as colunas da matriz resultante serão dadas por $\widehat{\mathbf{u}}\mathbf{p}_i = \mathbf{u} \times \mathbf{p}_i$, $i = 1, 2, 3$. Pré-multiplicando a matriz $\widehat{\mathbf{u}}R^T$ por R , obtemos $A = R\widehat{\mathbf{u}}R^T$ cujos elementos são dados por

$$a_{ij} = \mathbf{p}_i^T(\mathbf{u} \times \mathbf{p}_j), \quad i, j = 1, 2, 3.$$

Portanto, para $i = j$ temos que

$$a_{ii} = \mathbf{p}_i^T(\mathbf{u} \times \mathbf{p}_i) = -\mathbf{p}_i^T(\mathbf{p}_i \times \mathbf{u}) = -(\mathbf{p}_i \times \mathbf{p}_i)^T \mathbf{u} = 0$$

e para $i \neq j$

$$\begin{aligned} a_{ij} &= \mathbf{p}_i^T(\mathbf{u} \times \mathbf{p}_j) = -\mathbf{p}_i^T(\mathbf{p}_j \times \mathbf{u}) \\ &= -(\mathbf{p}_i \times \mathbf{p}_j)^T \mathbf{u} \\ &= -(\pm \mathbf{p}_k)^T \mathbf{u}, \end{aligned}$$

onde $k \in \{1, 2, 3\} \setminus \{i, j\}$. Como

$$\widehat{R\mathbf{u}} = \begin{pmatrix} 0 & -\mathbf{p}_3^T \mathbf{u} & \mathbf{p}_2^T \mathbf{u} \\ \mathbf{p}_3^T \mathbf{u} & 0 & -\mathbf{p}_1^T \mathbf{u} \\ -\mathbf{p}_2^T \mathbf{u} & \mathbf{p}_1^T \mathbf{u} & 0 \end{pmatrix},$$

comparando os elementos de $\widehat{R\mathbf{u}}$ com os elementos de A , concluímos que $R\widehat{\mathbf{u}}R^T = \widehat{R\mathbf{u}}$. ■

Provaremos agora, utilizando o resultado do Lema 2.1, que uma matriz de rotação define uma transformação de corpo rígido em \mathbb{R}^3 .

Proposição 2.1 *Uma matriz de rotação $R \in SO(3)$ define uma transformação de corpo rígido em \mathbb{R}^3 .*

Demonstração: Devemos provar que R satisfaz as propriedades (2.20). Com efeito, para todo $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$ temos

$$\|R\mathbf{v}\|^2 = (R\mathbf{v})^T R\mathbf{v} = \mathbf{v}^T R^T R\mathbf{v} = \mathbf{v}^T \mathbf{v} = \|\mathbf{v}\|^2 \Rightarrow \|R\mathbf{v}\| = \|\mathbf{v}\|.$$

Isto prova que R preserva a norma de vetores. Para provar que R preserva o produto vetorial, utilizamos o resultado do Lema 2.1:

$$R(\mathbf{u} \times \mathbf{v}) = R(\widehat{\mathbf{u}\mathbf{v}}) = (R\widehat{\mathbf{u}}R^T)R\mathbf{v} = \widehat{R\mathbf{u}}R\mathbf{v} = R\mathbf{u} \times R\mathbf{v}.$$

Logo, R define um transformação de corpo rígido em \mathbb{R}^3 . ■

Em geral, os movimentos rígidos são descritos por rotações e translações. Tomemos como exemplo a situação ilustrada na Figura 2.4. Considerando o objeto inicialmente em

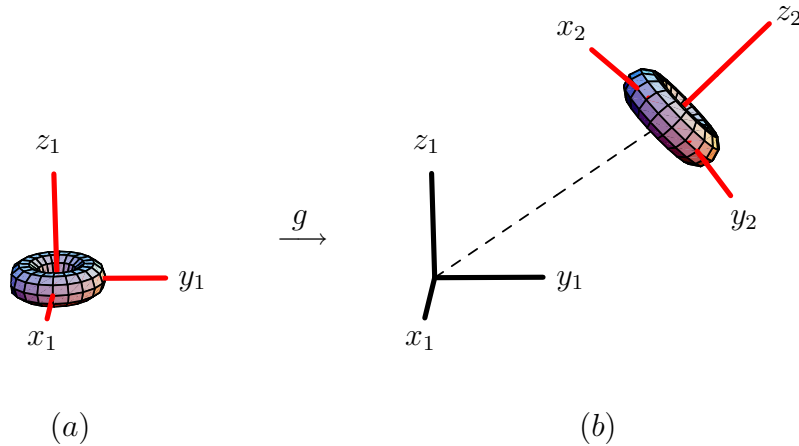


Figura 2.4: Objeto antes e depois de um movimento rígido.

repouso (Figura 2.4 (a)), podemos rodá-lo e deslocá-lo de sua posição original para obter a configuração ilustrada na Figura 2.4 (b). Denotando por \mathbf{p}_1 um ponto qualquer no objeto em repouso e por \mathbf{p}_2 o mesmo ponto no objeto após o movimento de rotação e translação, a relação entre \mathbf{p}_1 e \mathbf{p}_2 é dada por

$$\mathbf{p}_2 = \mathbf{d} + R\mathbf{p}_1,$$

onde $\mathbf{d} \in \mathbb{R}^3$ é um vetor de deslocamento e $R \in SO(3)$. Neste caso, a transformação de corpo rígido g será dada por

$$\begin{aligned} g : \mathbb{R}^3 &\rightarrow \mathbb{R}^3 \\ \mathbf{v} &\mapsto \mathbf{d} + R\mathbf{v}. \end{aligned} \tag{2.24}$$

Dentre os vários contextos onde a aplicação (2.24) aparece, podemos citar [24, 28] onde (2.24) é utilizada para descrever as coordenadas cartesianas da ponta de um braço-robô. No Capítulo 4 abordaremos um problema em que a transformação (2.24) também é utilizada para descrever a posição de átomos de moléculas em um recipiente.

2.3 Algumas propriedades do conjunto $SO(3)$

Definimos nas Seções 2.1 e 2.2 o conjunto $SO(n)$ para $n = 2, 3$. A partir de agora focaremos nossa atenção no conjunto $SO(3)$. Nesta seção recordaremos algumas de suas propriedades: veremos que este conjunto é *compacto* e constitui um grupo sob a operação de multiplicação de matrizes. Estas propriedades são também satisfeitas por $SO(2)$, de forma que as demonstrações que fizermos aqui podem ser facilmente adaptadas para este conjunto.

O conjunto $SO(3)$ é na verdade um subconjunto de $O(3)$. O conjunto $O(3)$, denominado *grupo ortogonal de ordem 3*, contém todas as matrizes ortogonais reais de ordem 3. Veremos na proposição a seguir que $SO(3)$ constitui um grupo com relação a operação de multiplicação de matrizes.

Proposição 2.2 $SO(3)$ é um grupo sob a operação de multiplicação de matrizes.

Demonstração: Basta mostrarmos que $SO(3)$ satisfaz as propriedades de um grupo. Com efeito, dados $R_1, R_2, R_3 \in SO(3)$ arbitrários, temos que:

- $SO(3)$ é fechado com relação a operação de multiplicação. De fato, dado o elemento $R_1 R_2$, temos que

$$\begin{aligned} (R_1 R_2)(R_1 R_2)^T &= R_1 (R_2 R_2^T) R_1^T = R_1 R_1^T = I, \\ \det(R_1 R_2) &= \det(R_1) \det(R_2) = 1. \end{aligned}$$

Logo, $R_1 R_2 \in SO(3)$.

- A matriz $I \in SO(3)$ é o elemento identidade do grupo, pois

$$RI = IR = R.$$

- De (2.18), concluímos que dado $R \in SO(3)$, existe $R^{-1} = R^T$ também pertencente a $SO(3)$. Portanto, todo elemento de $SO(3)$ possui um inverso em $SO(3)$.
- A operação definida sobre $SO(3)$ é associativa pois a multiplicação usual de matrizes é associativa, isto é, $(R_1 R_2) R_3 = R_1 (R_2 R_3)$.

Portanto, $SO(3)$ constitui um grupo sob a operação de multiplicação de matrizes. ■

Como esta operação é não comutativa, dizemos que $SO(3)$ é um grupo *não comutativo* ou *não abeliano*. Outra característica interessante de $SO(3)$ é o fato deste conjunto ser compacto. Para demonstrar essa propriedade utilizaremos o seguinte teorema que está provado em [17]:

Teorema 2.1 *Em um espaço normado de dimensão finita X , qualquer sub-conjunto $M \subset X$ é compacto se, e somente se, M é fechado e limitado.*

Lembrando que $\mathbb{R}^{3 \times 3}$ munido da norma euclidiana de matrizes é um espaço de dimensão finita e $SO(3) \subset \mathbb{R}^{3 \times 3}$, para concluir que $SO(3)$ é compacto basta provar que este conjunto é fechado e limitado em $\mathbb{R}^{3 \times 3}$. Faremos isso na proposição a seguir.

Proposição 2.3 *O conjunto $SO(3)$ é fechado e limitado em $\mathbb{R}^{3 \times 3}$.*

Demonstração: De fato, dado $R \in SO(3)$ arbitrário, suas colunas formam uma base ortonormal de \mathbb{R}^3 e $\det(R) = +1$. Estas condições podem ser expressas como um conjunto de equações não lineares nos elementos de R . O conjunto solução de cada equação pode ser visto como a pré-imagem (ou imagem inversa) de um conjunto discreto ($\{1\}$ ou $\{0\}$) por uma função contínua. Disto, decorre que $SO(3)$ é a intersecção de fechados, sendo portanto um conjunto fechado. Além disso, como R preserva a norma de vetores segue que

$$\|R\| = \sup_{\|\mathbf{x}\|=1} \{\|R\mathbf{x}\|\} = \sup_{\|\mathbf{x}\|=1} \{\|\mathbf{x}\|\} = 1.$$

Logo, $SO(3)$ é um conjunto limitado. ■

Portanto, pelo Teorema 2.1 podemos concluir que $SO(3)$ é compacto. Uma consequência muito importante dessa propriedade é o fato de que se uma função de valor real contínua for definida sobre o conjunto $SO(3)$, ou seja, $f : SO(3) \rightarrow \mathbb{R}$, ela atingirá um valor máximo e um valor mínimo neste conjunto. Assim, se estivermos trabalhando com problemas de otimização onde queremos encontrar $R \in SO(3)$ que minimize o valor de uma função contínua, saberemos pelo menos que existe uma solução para o problema.

Capítulo 3

Descrição de rotações em \mathbb{R}^3

O problema de descrever uma matriz de rotação por meio de *parâmetros* aparece com frequência em várias aplicações. Em robótica [24], [28], por exemplo, surgem situações em que é preciso calcular a posição ($\mathbf{d} \in \mathbb{R}^3$) e a orientação ($R \in SO(3)$) de um braço articulado com relação a um sistema de coordenadas fixo, bem como obter as taxas de variação dos parâmetros relacionados à rotação do braço. Já em computação gráfica, um problema muito interessante é o da interpolação de orientações [3], [25], isto é, conhecidas a posição e a orientação de um objeto em N instantes de tempo, é preciso determinar uma função contínua que interpole suavemente os dados. Em todas essas aplicações uma das dificuldades encontradas é a escolha de uma maneira adequada de representar as rotações envolvidas, ou seja, como escolher convenientemente parâmetros para construir as matrizes de rotação. Neste capítulo, começaremos definindo matematicamente o que é uma *parametrização* para um conjunto arbitrário e em seguida estudaremos as maneiras mais comuns de parametrizar uma matriz de rotação em $SO(3)$.

3.1 Parametrizações para $SO(3)$

Introduziremos inicialmente algumas definições relevantes, cf. [19] e [21].

Definição 3.1 *Seja M um conjunto arbitrário.*

- *Uma carta sobre M é um subconjunto aberto M_i de M juntamente com uma bijeção diferenciável $\varphi_i : M_i \rightarrow \varphi_i(M_i)$.*
- *O inverso φ_i^{-1} da aplicação φ_i é chamado de parametrização ou sistema local de coordenadas de M .*
- *Dois cartas (M_i, φ_i) , (M_j, φ_j) , tais que $M_i \cap M_j \neq \emptyset$, são compatíveis se $\varphi_i(M_i \cap M_j)$*

e $\varphi_j(M_i \cap M_j)$ são subconjuntos abertos de \mathbb{R}^n e as aplicações

$$\varphi_i \circ \varphi_j^{-1}|_{\varphi_j(M_i \cap M_j)} : \varphi_j(M_i \cap M_j) \longrightarrow \varphi_i(M_i \cap M_j)$$

e

$$\varphi_j \circ \varphi_i^{-1}|_{\varphi_i(M_i \cap M_j)} : \varphi_i(M_i \cap M_j) \longrightarrow \varphi_j(M_i \cap M_j)$$

são C^∞ .

- O conjunto M é uma variedade diferenciável n dimensional se admite um atlas, isto é, $M = \cup_{i \in \mathcal{I}} M_i$, e as cartas $\{(M_i, \varphi_i)\}_{i \in \mathcal{I}}$ são compatíveis.

A Figura 3.1 mostra dois subconjuntos abertos M_i e M_j de uma variedade M cuja intersecção é não vazia e as cartas que podemos construir com esses subconjuntos. Como φ_i e φ_j são contínuas e bijetivas, $\varphi_i(M_i)$ e $\varphi_j(M_j)$ são subconjuntos abertos em \mathbb{R}^n .

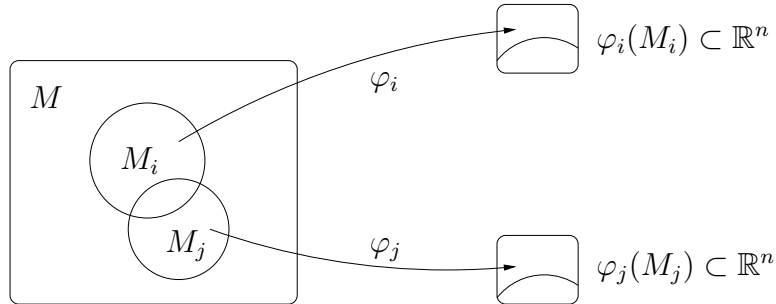


Figura 3.1: Cartas em uma variedade M arbitrária.

É possível provar que o conjunto $SO(3)$ é uma variedade de dimensão 3. Isto significa que se (M_i, φ_i) é uma carta em $SO(3)$, $\varphi_i(M_i) \subset \mathbb{R}^3$. A proposição a seguir traz um resultado muito importante sobre a construção de cartas em $SO(3)$. Ela diz que não é possível cobrir todo o conjunto $SO(3)$ com apenas uma carta.

Proposição 3.1 *Não existe uma carta global $(SO(3), \varphi)$ para $SO(3)$.*

Demonstração: A demonstração é por contradição. Suponhamos que $(SO(3), \varphi)$ seja uma carta global para $SO(3)$. Então, $\varphi(SO(3))$ é um conjunto aberto em \mathbb{R}^3 (pela definição de carta) e também um conjunto fechado (pois $SO(3)$ é fechado). Logo, $\varphi(SO(3)) = \mathbb{R}^3$. Como $SO(3)$ é compacto e φ é contínua, segue que $\varphi(SO(3))$ é compacto. Portanto, \mathbb{R}^3 é compacto e isto não é verdade. Conclusão: não existe uma carta global para $SO(3)$. ■

Mostraremos através do exemplo a seguir que o conjunto $SO(3)$ pode ser coberto por pelo menos 4 cartas.

Exemplo:

Sejam $\mathbf{x} \in \mathbb{R}^3$ não nulo e $\widehat{\mathbf{x}}$ a matriz anti-simétrica formada com as componentes de \mathbf{x} . Definiremos uma aplicação (denominada *mapa de Cayley*) da seguinte forma:

$$\begin{aligned} \text{CAY} : \mathbb{R}^3 &\longrightarrow \text{CAY}(\mathbb{R}^3) \subset SO(3) \\ \mathbf{x} &\mapsto (I + \widehat{\mathbf{x}})(I - \widehat{\mathbf{x}})^{-1} \end{aligned} \quad (3.1)$$

A aplicação (3.1) está bem definida pois a matriz $(I - \widehat{\mathbf{x}})$ é sempre inversível. Com efeito, como seus autovalores são $\{1, 1 - i\|\mathbf{x}\|, 1 + i\|\mathbf{x}\|\}$, temos que $\det(I - \widehat{\mathbf{x}}) \neq 0$. Além disso, pode-se mostrar que CAY é injetivo e $(SO(3), \text{CAY}^{-1})$ é uma carta local de $SO(3)$. Então, para determinar CAY^{-1} , procedemos da seguinte forma:

$$\begin{aligned} \text{CAY}(\mathbf{x}) = R &\iff R(I - \widehat{\mathbf{x}}) = (I + \widehat{\mathbf{x}}) \\ &\iff (R + I)\widehat{\mathbf{x}} = R - I \\ &\iff \widehat{\mathbf{x}} = (R + I)^{-1}(R - I) = \text{CAY}^{-1}(R). \end{aligned} \quad (3.2)$$

Notemos que em (3.2) a matriz $(R + I)$ só admitirá inversa se -1 não for um autovalor de R . Portanto, o mapa de Cayley estabelece um *difeomorfismo* entre \mathbb{R}^3 e $\text{CAY}(\mathbb{R}^3) = SO(3) - \Xi$, onde Ξ é o conjunto de matrizes que têm -1 como autovalor, ou equivalentemente, o conjunto de matrizes que descrevem uma rotação de π em torno de um eixo qualquer. Isto significa que o mapa inverso CAY^{-1} só está bem definido para as rotações por ângulos entre $[0, \pi)$, em torno de qualquer direção. Resta apenas cobrir as rotações por ângulo π , pois sabemos que uma rotação de θ em torno de um vetor \mathbf{x} equivale a uma rotação de $2\pi - \theta$ em torno do vetor $-\mathbf{x}$. Em particular, note que a rotação de π em torno de \mathbf{x} equivale à rotação de π em torno de $-\mathbf{x}$. Então, ao considerar rotações de π , basta considerar as classes de semelhança formadas pelos vetores antípodos.

Queremos agora construir uma parametrização para as matrizes de rotação correspondentes a uma rotação de π em torno de um dado vetor \mathbf{w} não nulo, que podemos supor unitário, sem perda de generalidade. Então \mathbf{w} ou $-\mathbf{w}$ pertencerá a um dos três semi-espacos $H_i = \{\mathbf{x} \in \mathbb{R}^3 \mid x_i > 0\}$, para $i = 1, 2, 3$. Denotaremos por $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ os vetores que formam a base canônica de \mathbb{R}^3 . Considere o caso $\mathbf{w} \in H_1$. Neste caso $\langle \mathbf{w}, \mathbf{e}_1 \rangle = \cos \theta \in [0, 1]$, onde θ é o ângulo entre \mathbf{w} e \mathbf{e}_1 . A matriz que descreve a rotação de π em torno de \mathbf{e}_1 é dada por

$$E_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

Suponha agora que $\cos \theta \in (0, 1)$. Sejam Q a matriz de rotação de π em torno de \mathbf{w} e $R = E_1 Q$. A matriz R corresponde a uma rotação de $\alpha = \|\mathbf{w} + \mathbf{e}_1\|\pi = \sqrt{2(1 + \cos \theta)}\pi$ em torno de $\mathbf{w} + \mathbf{e}_1$. Como $\alpha \in (\sqrt{2}\pi, 2\pi)$, temos que $2\pi - \alpha \in (0, (2 - \sqrt{2})\pi) \subset (0, \pi)$, ou seja, $R \in \text{CAY}(\mathbb{R}^3)$. Portanto, como $E_1^{-1} = E_1^T = E_1$, temos, em qualquer dos casos, que $Q = E_1 R$, onde R é uma matriz em $\text{CAY}(\mathbb{R}^3)$ (lembrando que $I \in \text{CAY}(\mathbb{R}^3)$).

O que foi feito para H_1 pode ser repetido de forma análoga para H_2 e H_3 . Chamando a matriz identidade de E_0 , e de E_2 e E_3 as rotações de π em torno de \mathbf{e}_2 e \mathbf{e}_3 respectivamente, concluímos que $SO(3)$ é coberto pelas parametrizações

$$\begin{aligned}\varphi_i^{-1} : \mathbb{R}^3 &\rightarrow M_i \subset SO(3) \\ \mathbf{x} &\mapsto E_i \text{CAY}(\mathbf{x}), \text{ para } i = 0, 1, 2, 3.\end{aligned}$$

É fácil verificar (M_i, φ_i) definem cartas em $SO(3)$.

As maneiras mais comuns de representar uma matriz de rotação por meio de parâmetros são baseadas no uso do *mapa exponencial*, dos *ângulos de Euler* e dos *quatérnions unitários* (ver, por exemplo, [11, 16, 27, 29, 30, 31]). Todas elas podem ser definidas como parametrizações locais para o conjunto $SO(3)$, ou seja, cobrem apenas um subconjunto de $SO(3)$. Nas próximas seções estudaremos cada uma das representações citadas. Daremos exemplos onde elas são utilizadas e apontaremos suas vantagens e desvantagens tanto na teoria quanto na prática.

3.2 O mapa exponencial

Veremos nessa seção uma maneira de representar o conjunto $SO(3)$ por meio de 4 parâmetros, utilizando para isso o conceito de exponencial de matrizes. Antes de introduzirmos as definições necessárias, vamos demonstrar um teorema muito importante devido a Euler.

Teorema 3.1 *Todo elemento $R \in SO(3)$, $R \neq I$, descreve uma rotação de um ângulo em torno de um eixo fixo.*

Demonstração: Seja $R \neq I$ uma matriz de rotação arbitrária. Primeiramente vejamos que R tem um autovalor igual a 1. Com efeito, os autovalores de R são dados pelas raízes do polinômio característico $p(\lambda) = R - \lambda I$. Como raízes complexas sempre ocorrem em pares conjugados e $p(\lambda)$ tem grau 3, pelo menos um dos autovalores de R é real. Sejam λ o autovalor real e $\boldsymbol{\omega} \in \mathbb{R}^3$ o autovetor associado a λ , que admitiremos ter norma unitária sem perda de generalidade. Assim, temos que

$$\|R\boldsymbol{\omega}\| = \|\lambda\boldsymbol{\omega}\| \implies \|\boldsymbol{\omega}\| = |\lambda|\|\boldsymbol{\omega}\| \implies \lambda = \pm 1.$$

Como o determinante de R é dado pelo produto dos autovalores e, além disso, $\det(R) = 1$, se todas as raízes de $p(\lambda)$ forem reais, elas serão dadas por $\{1, 1, 1\}$ ou $\{1, -1, -1\}$. Por outro lado, se existir uma raiz real e duas complexas, vamos ter $\{1, z, \bar{z}\}$, com $z\bar{z} = |z|^2 = 1$. Portanto, em qualquer dos casos anteriores, pelo menos um autovalor de R é igual a 1.

Consideremos agora o plano P perpendicular ao autovetor $\boldsymbol{\omega}$, ou seja,

$$P = \{\mathbf{y} \in \mathbb{R}^3 \mid \langle \mathbf{y}, \boldsymbol{\omega} \rangle = 0\}.$$

Como R é ortogonal, dado $\mathbf{y} \in P$, o vetor $R\mathbf{y}$ também está em P , pois

$$\langle R\mathbf{y}, \boldsymbol{\omega} \rangle = \langle R\mathbf{y}, R\boldsymbol{\omega} \rangle = \langle \mathbf{y}, \boldsymbol{\omega} \rangle = 0.$$

Então, tomando uma base ortonormal $\{\mathbf{v}_1, \mathbf{v}_2\}$ para P , podemos construir um sistema de coordenadas S para \mathbb{R}^3 formado pelos vetores $\{\boldsymbol{\omega}, \mathbf{v}_1, \mathbf{v}_2\}$. Com relação a esse sistema, a matriz R assume a forma

$$R = \left([\boldsymbol{\omega}]_S \mid [\mathbf{v}_1]_S \mid [\mathbf{v}_2]_S \right) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & a_1 & a_2 \\ 0 & a_3 & a_4 \end{pmatrix}. \quad (3.3)$$

É fácil verificar que a submatriz de R

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$$

pertence a $SO(2)$. Desse modo, podemos concluir que R descreve uma rotação de algum ângulo em torno do eixo gerado pelo vetor $\boldsymbol{\omega}$. A Figura 3.2 mostra o vetor $\boldsymbol{\omega}$ (eixo de rotação) e um vetor \mathbf{u} arbitrário cuja projeção no plano P é dada pelo vetor $\text{Proj}_P \mathbf{u}$. Quando pré-multiplicamos as coordenadas de \mathbf{u} pela matriz (3.3) o vetor $\text{Proj}_P \mathbf{u}$ realiza uma rotação em torno de $\boldsymbol{\omega}$ e a coordenada de \mathbf{u} ao longo do eixo definido por $\boldsymbol{\omega}$ permanece inalterada. ■

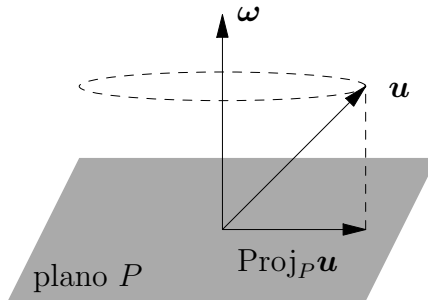


Figura 3.2: Rotação de \mathbf{u} em torno do eixo gerado por $\boldsymbol{\omega}$.

Com base no resultado do Teorema 3.1, dado um vetor unitário $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)^T \in \mathbb{R}^3$ e um número real $\theta \in [0, 2\pi)$, podemos definir uma aplicação que associa o par $(\boldsymbol{\omega}, \theta)$ (que representa o eixo e o ângulo de uma rotação) à uma matriz $R \in SO(3)$. Antes de construí-la vamos precisar da seguinte definição:

Definição 3.2 *Seja $A \in \mathbb{R}^{3 \times 3}$ arbitrária. A exponencial de A é definida por*

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}.$$

Construindo a matriz anti-simétrica $\widehat{\boldsymbol{\omega}}$ com as componentes do vetor $\boldsymbol{\omega}$ como foi feito em (2.23), definimos o mapa exponencial da seguinte forma:

$$\begin{aligned} EXP : S^2 \times [0, 2\pi) &\rightarrow SO(3) \\ (\boldsymbol{\omega}, \theta) &\mapsto e^{\widehat{\boldsymbol{\omega}}\theta} \end{aligned} \quad (3.4)$$

onde $S^2 = \{\boldsymbol{\omega} \in \mathbb{R}^3 \mid \|\boldsymbol{\omega}\| = 1\}$. Para ver que $R = e^{\widehat{\boldsymbol{\omega}}\theta}$ é de fato uma matriz de rotação, basta lembrarmos que a exponencial de uma matriz satisfaz as propriedades usuais de funções exponenciais. Assim, podemos escrever:

$$(e^{\widehat{\boldsymbol{\omega}}\theta})^{-1} = e^{-\widehat{\boldsymbol{\omega}}\theta} = e^{\widehat{\boldsymbol{\omega}}^T\theta} = (e^{\widehat{\boldsymbol{\omega}}\theta})^T.$$

Então $R^{-1} = R^T$ e conseqüentemente $RR^T = I$, implicando que $\det(R) = \pm 1$. Usando o fato de que o determinante de uma matriz e o mapa (3.4) são funções contínuas e que $\det(e^0) = 1$, concluímos que $\det(R) = +1$.

A matriz $R = e^{\widehat{\boldsymbol{\omega}}\theta}$ definida como uma série de potências da matriz anti-simétrica $\widehat{\boldsymbol{\omega}}$ não se presta a cálculos. Podemos entretanto encontrar uma expressão mais simples para R fazendo uso recursivo das identidades

$$\widehat{\boldsymbol{u}}^2 = \boldsymbol{u}\boldsymbol{u}^T - \|\boldsymbol{u}\|^2 I, \quad (3.5)$$

$$\widehat{\boldsymbol{u}}^3 = -\|\boldsymbol{u}\|^2 \widehat{\boldsymbol{u}}. \quad (3.6)$$

Assim, tomando $\boldsymbol{u} = \boldsymbol{\omega}\theta$ em (3.5) e (3.6) e lembrando as expansões do seno e do cosseno em séries de Taylor, podemos escrever:

$$\begin{aligned} e^{\widehat{\boldsymbol{\omega}}\theta} &= I + \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots\right) \widehat{\boldsymbol{\omega}} + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - \dots\right) \widehat{\boldsymbol{\omega}}^2 \\ &= I + \widehat{\boldsymbol{\omega}} \operatorname{sen} \theta + \widehat{\boldsymbol{\omega}}^2 (1 - \cos \theta). \end{aligned} \quad (3.7)$$

A expressão (3.7) é conhecida como *Fórmula de Rodrigues* e nos dá uma maneira eficiente de calcular a matriz $R = e^{\widehat{\boldsymbol{\omega}}\theta}$.

O mapa exponencial definido em (3.4) cobre todo o conjunto $SO(3)$, ou seja, dada $R \in SO(3)$, existe $\boldsymbol{\omega} \in \mathbb{R}^3$ unitário e $\theta \in \mathbb{R}$ tal que $R = e^{\widehat{\boldsymbol{\omega}}\theta}$. Provaremos este fato na proposição a seguir.

Proposição 3.2 *O mapa exponencial (3.4) é sobrejetivo.*

Demonstração: A prova desta proposição é construtiva. Tomando

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad (3.8)$$

e definindo $v_\theta = 1 - \cos \theta$, $c_\theta = \cos \theta$, $s_\theta = \operatorname{sen} \theta$, escrevemos a equação (3.7) como

$$\begin{aligned} e^{\widehat{\boldsymbol{\omega}}\theta} &= \begin{bmatrix} 1 - v_\theta(\omega_2^2 + \omega_3^2) & \omega_1\omega_2v_\theta - \omega_3s_\theta & \omega_1\omega_3v_\theta + \omega_2s_\theta \\ \omega_1\omega_2v_\theta + \omega_3s_\theta & 1 - v_\theta(\omega_1^2 + \omega_3^2) & \omega_2\omega_3v_\theta - \omega_1s_\theta \\ \omega_1\omega_3v_\theta - \omega_2s_\theta & \omega_2\omega_3v_\theta + \omega_1s_\theta & 1 - v_\theta(\omega_1^2 + \omega_2^2) \end{bmatrix} \\ &= \begin{bmatrix} \omega_1^2v_\theta + c_\theta & \omega_1\omega_2v_\theta - \omega_3s_\theta & \omega_1\omega_3v_\theta + \omega_2s_\theta \\ \omega_1\omega_2v_\theta + \omega_3s_\theta & \omega_2^2v_\theta + c_\theta & \omega_2\omega_3v_\theta - \omega_1s_\theta \\ \omega_1\omega_3v_\theta - \omega_2s_\theta & \omega_2\omega_3v_\theta + \omega_1s_\theta & \omega_3^2v_\theta + c_\theta \end{bmatrix}. \end{aligned} \quad (3.9)$$

Igualando as expressões (3.8) e (3.9), obtemos a seguinte relação para o traço da matriz R :

$$\text{tr}(R) = r_{11} + r_{22} + r_{33} = 1 + 2 \cos \theta.$$

Segue então que

$$-1 \leq \text{tr}(R) \leq 3 \quad (3.10)$$

e

$$\theta = \cos^{-1} \left(\frac{\text{tr}(R) - 1}{2} \right), \theta \in [0, \pi]. \quad (3.11)$$

Agora, igualando os outros termos de R e $e^{\widehat{\boldsymbol{\omega}}\theta}$, obtemos

$$\begin{aligned} r_{32} - r_{23} &= 2\omega_1 s_\theta, \\ r_{13} - r_{31} &= 2\omega_2 s_\theta, \\ r_{21} - r_{12} &= 2\omega_3 s_\theta. \end{aligned}$$

Se $\sin \theta \neq 0$, ou seja, $\theta \in (0, \pi)$, podemos tomar

$$\boldsymbol{\omega} = \frac{1}{2s_\theta} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}. \quad (3.12)$$

Os valores de θ e $\boldsymbol{\omega}$ dados, respectivamente, pelas equações (3.11) e (3.12) (denominados *coordenadas exponenciais* de R) não são únicos. Com efeito, observe que se considerarmos um ângulo de $2\pi - \theta$, como $\sin(2\pi - \theta) = -\sin \theta$, de acordo com a equação (3.12) o eixo de rotação será $-\boldsymbol{\omega}$ e uma rotação de $2\pi - \theta$ em torno de $-\boldsymbol{\omega}$ produzirá o mesmo efeito de uma rotação de θ em torno de $\boldsymbol{\omega}$. Agora, se R é tal que $\text{tr}(R) = -1$, então $\theta = \pi$ e $\boldsymbol{\omega}$ pode ser escolhido arbitrariamente. Por outro lado, se $R = I$, então $\text{tr}(R) = 3$ e $\theta = 0$, implicando que $\boldsymbol{\omega}$ também pode assumir qualquer valor. Portanto, o mapa exponencial (3.4) é uma aplicação sobrejetiva. ■

A representação de rotações através do mapa exponencial é também conhecida como *representação eixo-ângulo*. Notemos que para descrever uma rotação por meio dessa representação devemos utilizar quatro parâmetros: um ângulo e três componentes do vetor unitário. Assim, construímos a matriz de rotação correspondente através da fórmula (3.7). Dentre as várias aplicações que utilizam o mapa exponencial para descrever o conjunto $SO(3)$, podemos citar [24] que usa (3.4) para obter as equações cinemáticas que regem o movimento de vários modelos de braços-robô. Já em [30] este mapa é utilizado em um problema de otimização cujo objetivo é encontrar uma matriz $R \in SO(3)$ que minimiza uma função contínua $f : SO(3) \rightarrow \mathbb{R}$.

Veremos na seção a seguir uma outra maneira de representar uma rotação utilizando três parâmetros conhecidos como ângulos de Euler.

3.3 Ângulos de Euler

Leonhard Euler (1707–1783), em conexão com seu trabalho sobre mecânica celeste, afirmou e provou que

Quaisquer dois sistemas ortogonais de coordenadas independentes podem ser relacionados por uma seqüência de rotações (não mais que três) sobre os eixos coordenados, de forma que duas rotações sucessivas não devem ocorrer em torno do mesmo eixo.

Desse modo, o ângulo de uma rotação simples em torno de um eixo coordenado (X , Y ou Z) é chamado de *ângulo de Euler* e uma seqüência de três rotações simples, sendo que duas rotações consecutivas não ocorrem em torno do mesmo eixo é denominada *seqüência de Euler*. Podemos então, obter um total de 12 seqüências diferentes como mostra a Tabela 3.1.

XYX	XZX	YXY
YZY	ZXZ	ZYZ
XYZ	XZY	YZX
YXZ	ZXY	ZYX

Tabela 3.1: Seqüências de Euler.

Para construir as matrizes de rotação correspondentes às seqüências de Euler, necessitamos das matrizes que descrevem uma rotação de um ângulo θ no sentido anti-horário em torno dos eixos coordenados X , Y e Z :

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\text{sen } \theta \\ 0 & \text{sen } \theta & \cos \theta \end{bmatrix}, \quad (3.13)$$

$$R_{y,\theta} = \begin{bmatrix} \cos \theta & 0 & \text{sen } \theta \\ 0 & 1 & 0 \\ -\text{sen } \theta & 0 & \cos \theta \end{bmatrix}, \quad (3.14)$$

$$R_{z,\theta} = \begin{bmatrix} \cos \theta & -\text{sen } \theta & 0 \\ \text{sen } \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3.15)$$

Ilustraremos o uso de uma seqüência de Euler particular no exemplo a seguir.

Exemplo: Considere um objeto em formato de pirâmide cuja base inicialmente repousa sobre o plano XY . Aplicaremos a este objeto a seqüência ZXZ , isto é, primeiramente faremos uma rotação do objeto em torno do eixo Z de um ângulo ψ (Fig. 3.3 (a)). Em seguida, rodamos a pirâmide em torno do eixo X de um ângulo θ (Fig. 3.3 (b)) e por fim, aplicamos a ela uma rotação em torno do eixo Z de um ângulo ϕ (Fig. 3.3 (c)).

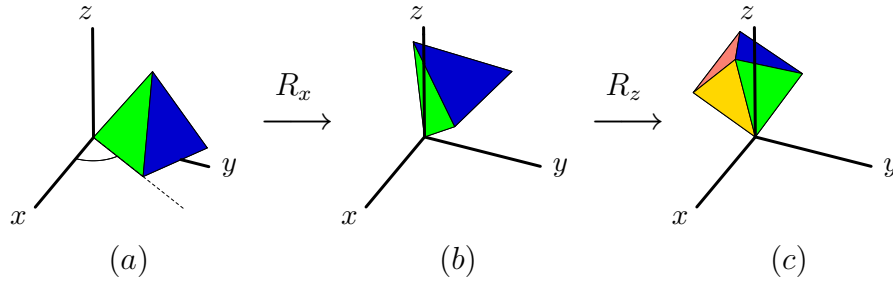


Figura 3.3: Seqüência ZXZ aplicada a um objeto.

A matriz que representa a composição das rotações é dada por

$$A(\psi, \theta, \phi) = R_{z,\phi}R_{x,\theta}R_{z,\psi} = \begin{pmatrix} c_\phi c_\psi - s_\phi c_\theta s_\psi & -c_\phi s_\psi - s_\phi c_\theta c_\psi & s_\phi s_\theta \\ s_\phi c_\psi + c_\phi c_\theta s_\psi & -s_\phi s_\psi + c_\phi c_\theta c_\psi & -c_\phi s_\theta \\ s_\theta s_\psi & s_\theta c_\psi & c_\theta \end{pmatrix} \quad (3.16)$$

onde $c_\phi = \cos \phi$, $s_\phi = \sin \phi$ e assim por diante. Considere a aplicação

$$A : [0, 2\pi) \times [0, \pi] \times [0, 2\pi) \longrightarrow SO(3).$$

Esta aplicação é sobrejetiva pois, dada $R \in SO(3)$ arbitrária:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad (3.17)$$

podemos determinar os valores dos ângulos (ψ, θ, ϕ) da seqüência ZXZ que produzirão R . Assim, se $r_{33} = \pm 1$,

$$\theta = \cos^{-1}(r_{33}) = \begin{cases} 0 \Rightarrow \phi + \psi = \text{tg}^{-1}(r_{22}, r_{21}) \\ \text{ou} \\ \pi \Rightarrow \phi - \psi = \text{tg}^{-1}(r_{11}, r_{12}). \end{cases}$$

Logo, ϕ e ψ não estão univocamente definidos, somente sua soma ou diferença. Agora, se $r_{33} \neq \pm 1$, vamos ter $s_\theta \neq 0$ e

$$\begin{aligned}\theta &= \cos^{-1}(r_{33}) \in (0, \pi), \\ \phi &= \operatorname{tg}^{-1}\left(-\frac{r_{23}}{s_\theta}, \frac{r_{13}}{s_\theta}\right), \\ \psi &= \operatorname{tg}^{-1}\left(\frac{r_{32}}{s_\theta}, \frac{r_{31}}{s_\theta}\right).\end{aligned}\tag{3.18}$$

Apesar de sobrejetiva, a aplicação A não é injetiva pois $A(\phi, 0, -\phi) = I$, para qualquer ϕ . Por essa razão, A não define uma parametrização para $SO(3)$. No entanto, se restringirmos os ângulos aos intervalos

$$0 < \phi < 2\pi, \quad 0 < \theta < \pi, \quad 0 < \psi < 2\pi,$$

a aplicação A torna-se bijetiva e portanto pode ser considerada uma parametrização local para $SO(3)$ ou seja, a imagem de A fica estritamente contida em $SO(3)$.

3.3.1 Limitações no uso dos ângulos de Euler

O exemplo anterior ilustra o fato que a parametrização por ângulos de Euler é necessariamente local, ou seja, não cobre todo o conjunto $SO(3)$. Resumindo, se $A : D \subset \mathbb{R}^3 \rightarrow SO(3)$ é uma parametrização local de $SO(3)$ por uma seqüência de Euler, qualquer tentativa de estender o domínio de definição dos ângulos para obter uma aplicação sobrejetiva fará com que A perca a propriedade da injetividade. Este fato se reflete na prática, principalmente em problemas de mecânica cujo objetivo é controlar a orientação de corpos rígidos via ângulos de Euler. Para entendermos melhor este tópico, consideremos o exemplo a seguir.

Exemplo: A Figura 3.4 (a) mostra um objeto articulado em repouso. A moldura de cor azul está presa a um suporte vermelho por meio de um eixo I e o triângulo está conectado à moldura por um eixo II. Inicialmente os eixos I e II coincidem, respectivamente, com os eixos cartesianos X e Y . O objeto executa rotações da seguinte forma: a moldura azul e o triângulo podem girar livremente em torno dos eixos I e II, respectivamente, e o suporte vermelho roda todo o conjunto em torno do eixo Z (Figura 3.4 (b)). Porém, se aplicarmos à moldura azul uma rotação de 90° em torno do eixo I ela ficará na posição vertical e o eixo II coincidirá com o eixo Z . Mantendo a moldura fixa nesta posição, o triângulo ficará “preso” na posição vertical e rotações do triângulo em torno do eixo II ou rotações do suporte vermelho em torno de Z não alterarão esta configuração (Figura 3.4 (c)).

A matriz que descreve a orientação do objeto articulado é dada por

$$R(\psi, \theta, \phi) = \begin{pmatrix} c_\theta c_\psi & -c_\theta s_\psi & s_\theta \\ c_\phi s_\psi + s_\phi s_\theta c_\psi & c_\phi c_\psi - s_\phi s_\theta s_\psi & -s_\phi c_\theta \\ s_\phi s_\psi - c_\phi s_\theta c_\psi & s_\phi c_\psi + c_\phi s_\theta s_\psi & c_\phi c_\theta \end{pmatrix}, \tag{3.19}$$

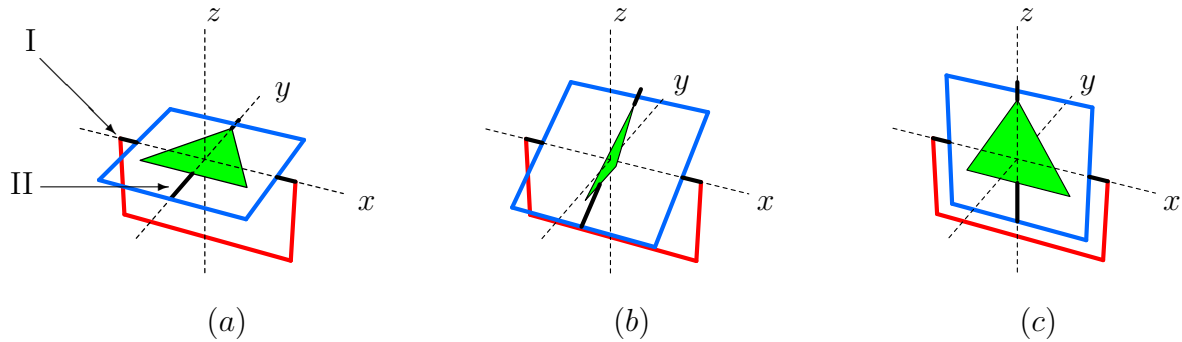


Figura 3.4: *Gimbal Lock*: dois eixos de rotação se coincidem.

onde ϕ é o ângulo da rotação da moldura azul em torno do eixo I, θ é o ângulo da rotação do triângulo em torno do eixo II e ψ é o ângulo da rotação do suporte vermelho em torno do eixo Z. Quando fazemos $\theta = \frac{\pi}{2}$ em (3.19) e deixamos ϕ e ψ livres, obtemos

$$R\left(\psi, \frac{\pi}{2}, \phi\right) = \begin{pmatrix} 0 & 0 & 1 \\ \text{sen}(\psi + \phi) & \text{cos}(\psi + \phi) & 0 \\ -\text{cos}(\psi + \phi) & \text{sen}(\psi + \phi) & 0 \end{pmatrix}. \quad (3.20)$$

A matriz (3.20) descreve a orientação do objeto quando este se encontra na posição ilustrada na Figura 3.4 (c). Notemos que (3.20) só depende da soma dos ângulos ψ e ϕ . Isto significa que para quaisquer ψ, ϕ tais que $\psi + \phi = \text{constante}$, a matriz $R\left(\psi, \frac{\pi}{2}, \phi\right)$ será sempre a mesma.

O problema ilustrado neste exemplo é conhecido em computação gráfica como *Gimbal Lock* e ocorre quando dois eixos de rotação apontam na mesma direção. *Gimbal* é o nome dado em inglês a uma montagem usada para instalar equipamentos que necessitam de uma orientação espacial e, para esse fim, utiliza ângulos de Euler.

Na seção a seguir estudaremos uma outra maneira de representar o conjunto $SO(3)$ utilizando quatérnions unitários. Veremos que é possível construir uma aplicação sobrejetiva que leva o conjunto dos quatérnions unitários ao conjunto $SO(3)$.

3.4 Quatérnions

3.4.1 Um pouco de história

William Rowan Hamilton (1805–1865) inventou os quatérnions em 1843 quando tentou introduzir um *número complexo generalizado*. Considerando números da forma $\mathbf{a} = a_1\mathbf{i} + a_2\mathbf{j} +$

$a_3\mathbf{k}$, com $a_1, a_2, a_3 \in \mathbb{R}$, Hamilton queria encontrar uma regra de multiplicação entre esses números semelhante à regra de multiplicação entre números complexos. Assim, dados \mathbf{a}, \mathbf{b} arbitrários, o produto inventado por Hamilton deveria satisfazer $|\mathbf{ab}| = |\mathbf{a}||\mathbf{b}|$. Porém, essa regra falhava em alguns casos. Por exemplo, se considerarmos $\mathbf{a} = \mathbf{i} + \mathbf{j} + \mathbf{k}$ e $\mathbf{b} = \mathbf{i} + 2\mathbf{j} + 4\mathbf{k}$ vamos ter

$$\begin{aligned} |\mathbf{a}|^2 &= 1^2 + 1^2 + 1^2 = 3, \\ |\mathbf{b}|^2 &= 1^2 + 2^2 + 4^2 = 21, \end{aligned}$$

mas $3 \times 21 = 63 \neq n_1^2 + n_2^2 + n_3^2$, para quaisquer inteiros n_1, n_2, n_3 . Percebendo que seu sistema de números complexos generalizados falhava com relação a esta propriedade, a grande idéia de Hamilton foi considerar o problema em quatro dimensões tomando elementos da forma $\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$, com $q_0, q_1, q_2, q_3 \in \mathbb{R}$, cujas unidades $\mathbf{i}, \mathbf{j}, \mathbf{k}$ satisfazem as seguintes regras de multiplicação:

$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$
$\mathbf{ij} = \mathbf{k} = -\mathbf{ji}$
$\mathbf{jk} = \mathbf{i} = -\mathbf{kj}$
$\mathbf{ki} = \mathbf{j} = -\mathbf{ik}$

Tabela 3.2: Regras de multiplicação inventadas por Hamilton.

Os elementos de quatro componentes foram denominados *quatérnions*. Hamilton dedicou cerca de 22 anos de sua vida ao estudo desses elementos e durante esse período publicou vários trabalhos importantes relacionados a esse assunto. Sua teoria inicialmente não foi bem aceita pela comunidade científica, mas aos poucos foi incorporando adeptos, como os matemáticos Peter Guthrie Tait e Arthur Cayley que também desenvolveram pesquisas nessa área. Em uma de suas principais obras, intitulada *Lectures on Quaternions*, Hamilton mostrou como os quatérnions podem ser aplicados à geometria, trigonometria e também mostrou relações existentes entre esses elementos e logaritmos, séries, equações lineares e quadráticas. Mais detalhes sobre a vida de Hamilton e sua teoria de quatérnions podem ser encontrados em [5, 26].

Estudaremos agora algumas propriedades algébricas dos quatérnions. Começaremos definindo as operações de adição, subtração, multiplicação por um escalar e multiplicação entre quatérnions. Em seguida introduziremos os conceitos de *conjugado* e *norma* de um quatérnion. Por fim, mostraremos como podemos utilizar esses elementos para representar rotações em \mathbb{R}^3 .

3.4.2 Álgebra de quatérnions

De acordo com as idéias de Hamilton, um quatérnion $\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ é constituído de duas partes: uma escalar $q_0 \in \mathbb{R}$ e uma vetorial $q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$. Representando a parte

vetorial de um quatérnion \mathbf{q} por \mathbf{q}_v , podemos escrevê-lo como $\mathbf{q} = (q_0, \mathbf{q}_v)^T = (q_0, q_1, q_2, q_3)^T$. Utilizaremos essa notação de agora em diante para simplificar os cálculos.

As operações de adição/subtração entre quatérnions e multiplicação de um quatérnion por um escalar são análogas às operações entre vetores em \mathbb{R}^4 , ou seja, dados os quatérnions $\mathbf{p} = (p_0, \mathbf{p}_v)^T$, $\mathbf{q} = (q_0, \mathbf{q}_v)^T$ e $\alpha \in \mathbb{R}$, temos que

$$\begin{aligned}\mathbf{p} + \mathbf{q} &= (p_0 + q_0, \mathbf{p}_v + \mathbf{q}_v)^T, \\ \mathbf{p} - \mathbf{q} &= (p_0 - q_0, \mathbf{p}_v - \mathbf{q}_v)^T, \\ \alpha\mathbf{p} &= (\alpha p_0, \alpha\mathbf{p}_v)^T.\end{aligned}\tag{3.21}$$

A operação de multiplicação entre quatérnions é definida segundo as regras da Tabela 3.2. Dessa forma, o produto entre dois quatérnions \mathbf{p} e \mathbf{q} é dado por

$$\mathbf{pq} = (p_0q_0 - \mathbf{p}_v \cdot \mathbf{q}_v, p_0\mathbf{q}_v + q_0\mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v)^T.\tag{3.22}$$

onde $\mathbf{p}_v \cdot \mathbf{q}_v$ e $\mathbf{p}_v \times \mathbf{q}_v$ são, respectivamente, o produto escalar e o produto vetorial entre as partes vetoriais de \mathbf{p} e \mathbf{q} . Observando a expressão (3.22) podemos notar que o produto entre dois quatérnions resulta em outro quatérnion cujas partes escalar e vetorial são dadas, respectivamente, por $p_0q_0 - \mathbf{p}_v \cdot \mathbf{q}_v$ e $p_0\mathbf{q}_v + q_0\mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v$. A operação de multiplicação entre quatérnions é associativa, distributiva e não comutativa. Por essa razão, o conjunto de todos os quatérnions (excluindo o quatérnion nulo), denotado por $\mathbb{H} - \{\mathbf{0}\}$, forma um grupo não comutativo com relação a essa operação.

Assim como um número complexo, um quatérnion também possui um *conjugado* e uma *norma*, como veremos a seguir.

Definição 3.3 *Dado $\mathbf{q} \in \mathbb{H} - \{\mathbf{0}\}$ arbitrário, o conjugado e a norma de \mathbf{q} são, respectivamente, definidos por*

$$\mathbf{q}^* = (q_0, -\mathbf{q}_v)^T,\tag{3.23}$$

$$\|\mathbf{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}.\tag{3.24}$$

Utilizando o produto entre quatérnions (3.22) e a definição anterior, as propriedades a seguir são facilmente verificadas $\forall \mathbf{q}, \mathbf{p} \in \mathbb{H}$

$$\begin{aligned}(\mathbf{q}^*)^* &= \mathbf{q}, \\ (\mathbf{pq})^* &= \mathbf{q}^*\mathbf{p}^*, \\ (\mathbf{p} + \mathbf{q})^* &= \mathbf{p}^* + \mathbf{q}^*, \\ \|\mathbf{q}\| &= \|\mathbf{q}^*\|, \\ \|\mathbf{pq}\| &= \|\mathbf{p}\|\|\mathbf{q}\|.\end{aligned}\tag{3.25}$$

Um estudo detalhado sobre outras propriedades satisfeitas pelo grupo $\mathbb{H} - \{\mathbf{0}\}$ pode ser encontrado em [6, 18, 20].

3.4.3 Descrição de rotações via quatérnions

No Capítulo 2 vimos que um número complexo de norma unitária pode ser identificado com uma matriz de rotação em $SO(2)$. Podemos também estabelecer uma identificação entre um quatérnion e um elemento de $SO(3)$. Para tal, precisamos considerar o conjunto de quatérnions unitários, ou seja, $S^3 = \{\mathbf{q} \in \mathbb{H} \mid \|\mathbf{q}\| = 1\}$. Assim, dado $\mathbf{q} \in S^3$ é possível construir com \mathbf{q} um operador de rotação de tal maneira que o seu efeito sobre um vetor não nulo $\mathbf{v} \in \mathbb{R}^3$ seja rodar \mathbf{v} de uma quantidade $\theta \in \mathbb{R}$ em torno de um eixo especificado. Surge então a seguinte questão:

Como um quatérnion pode operar sobre um vetor \mathbf{v} de \mathbb{R}^3 ?

Para que isto seja possível, definimos um quatérnion \mathbf{v}_a cuja parte vetorial seja dada pelo vetor \mathbf{v} e a parte escalar seja nula, isto é, $\mathbf{v}_a = (0, \mathbf{v})^T$. O conjunto de todos os quatérnions com parte escalar nula é denotado por \mathbb{H}_0 e um elemento qualquer deste conjunto é denominado *quatérnion puro*. A correspondência entre \mathbb{R}^3 e \mathbb{H}_0 está ilustrada na Figura 3.5. Tomando por

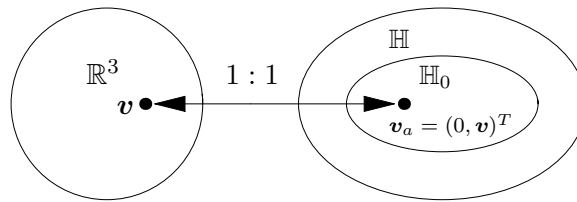


Figura 3.5: Correspondência biunívoca entre \mathbb{R}^3 e \mathbb{H}_0 .

base essa correspondência e utilizando a multiplicação entre quatérnions, podemos definir um operador de rotação da seguinte forma:

Definição 3.4 Dado $\mathbf{q} \in S^3$, o operador de rotação L construído a partir de \mathbf{q} é definido por

$$\begin{aligned} L : \mathbb{H}_0 &\rightarrow \mathbb{H}_0 \\ \mathbf{v}_a &\mapsto \mathbf{w}_a = \mathbf{q}\mathbf{v}_a\mathbf{q}^* \end{aligned} \quad (3.26)$$

onde \mathbf{q}^* é o conjugado de \mathbf{q} .

Desenvolvendo o produto $\mathbf{q}\mathbf{v}_a\mathbf{q}^*$ obtemos $\mathbf{w}_a = (0, \mathbf{w})^T$, onde

$$\mathbf{w} = (2q_0^2 - 1)\mathbf{v} + 2(\mathbf{q}_v \cdot \mathbf{v})\mathbf{q}_v + 2q_0(\mathbf{q}_v \times \mathbf{v}) \quad (3.27)$$

é a parte vetorial de \mathbf{w}_a que fornece as coordenadas do vetor \mathbf{v} após a rotação.

Mostraremos agora que, assim como uma matriz de rotação, L é um operador linear e preserva a norma euclidiana de vetores em \mathbb{R}^3 .

Proposição 3.3 *L é um operador linear e preserva a norma de vetores.*

Demonstração: Sejam $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$ arbitrários, $\alpha \in \mathbb{R}$ e considere os quatérnions puros $\mathbf{u}_a = (0, \mathbf{u})^T$ e $\mathbf{v}_a = (0, \mathbf{v})^T$. Pela distributividade do produto de quatérnions temos que:

$$\begin{aligned} L(\alpha\mathbf{u}_a + \mathbf{v}_a) &= \mathbf{q}(\alpha\mathbf{u}_a + \mathbf{v}_a)\mathbf{q}^* \\ &= (\alpha\mathbf{q}\mathbf{u}_a + \mathbf{q}\mathbf{v}_a)\mathbf{q}^* \\ &= \alpha\mathbf{q}\mathbf{u}_a\mathbf{q}^* + \mathbf{q}\mathbf{v}_a\mathbf{q}^* \\ &= \alpha L(\mathbf{u}_a) + L(\mathbf{v}_a). \end{aligned}$$

Além disso, como $\|\mathbf{q}\| = \|\mathbf{q}^*\| = 1$, segue que

$$\|L(\mathbf{v}_a)\| = \|\mathbf{q}\mathbf{v}_a\mathbf{q}^*\| = \|\mathbf{q}\|\|\mathbf{v}_a\|\|\mathbf{q}^*\| = \|\mathbf{v}_a\| = \|\mathbf{v}\|.$$

Portanto, L é linear e preserva a norma de vetores. ■

O teorema a seguir fornece uma interpretação geométrica para a ação do operador L sobre o quatérnion $\mathbf{v}_a = (0, \mathbf{v})^T$.

Teorema 3.2 *Dado $\mathbf{q} = (q_0, \mathbf{q}_v)^T$ unitário, com $\mathbf{q}_v \neq \mathbf{0}$, a parte vetorial de $\mathbf{w}_a = L(\mathbf{v}_a) = (0, \mathbf{w})^T$ é o resultado da rotação de \mathbf{v} de um ângulo 2θ em torno do eixo gerado por \mathbf{q}_v , onde $\theta = \cos^{-1}(q_0) \in (0, \pi)$.*

Demonstração: Sejam V o subespaço gerado pelo vetor \mathbf{q}_v , V^\perp o complemento ortogonal de V e $\mathbf{v} \in \mathbb{R}^3$. Denotando por $\mathbf{a}_1, \mathbf{a}_2$ as componentes da decomposição ortogonal de \mathbf{v} em V e V^\perp , respectivamente, segue que

$$\mathbf{v} = \mathbf{a}_1 + \mathbf{a}_2 = \alpha\mathbf{q}_v + \mathbf{a}_2, \text{ onde } \alpha \in \mathbb{R}.$$

Dessa forma, o quatérnion puro \mathbf{v}_a pode ser escrito como

$$\mathbf{v}_a = (0, \mathbf{a}_1 + \mathbf{a}_2)^T = (0, \mathbf{a}_1)^T + (0, \mathbf{a}_2)^T = \mathbf{v}_{a_1} + \mathbf{v}_{a_2}.$$

Como L é um operador linear, temos que

$$L(\mathbf{v}_a) = L(\mathbf{v}_{a_1}) + L(\mathbf{v}_{a_2}). \quad (3.28)$$

Por outro lado, é fácil verificar que $L(0, \alpha\mathbf{q}_v) = (0, \alpha\mathbf{q}_v)$, ou seja, o subespaço gerado por $(0, \mathbf{q}_v)$ é invariante sob L . Portanto

$$L(\mathbf{v}_a) = \mathbf{v}_{a_1} + (0, \mathbf{b}_2)^T = (0, \mathbf{a}_1 + \mathbf{b}_2)^T. \quad (3.29)$$

Agora, para determinar a parte vetorial de $L(\mathbf{v}_{a_2})$ utilizamos a fórmula (3.27) e o fato de que \mathbf{q} é unitário. Assim, temos

$$\begin{aligned} \mathbf{b}_2 &= (q_0^2 - \|\mathbf{q}_v\|^2)\mathbf{a}_2 + 2(\mathbf{q}_v \cdot \mathbf{a}_2)\mathbf{q}_v + 2q_0(\mathbf{q}_v \times \mathbf{a}_2) \\ &= (q_0^2 - \|\mathbf{q}_v\|^2)\mathbf{a}_2 + 2q_0(\mathbf{q}_v \times \mathbf{a}_2), \end{aligned} \quad (3.30)$$

pois $\mathbf{q}_v \perp \mathbf{a}_2$. Sendo $\mathbf{u} = \mathbf{q}_v / \|\mathbf{q}_v\| \in \mathbb{R}^3$ um vetor unitário com a mesma direção de \mathbf{q}_v , voltando em (3.30) podemos escrever

$$\mathbf{b}_2 = (q_0^2 - \|\mathbf{q}_v\|^2)\mathbf{a}_2 + 2q_0\|\mathbf{q}_v\|(\mathbf{u} \times \mathbf{a}_2). \quad (3.31)$$

Como $\|\mathbf{q}\|^2 = q_0^2 + \|\mathbf{q}_v\|^2 = 1$, existe um único $\theta \in (0, \pi)$ tal que

$$\begin{aligned} q_0 &= \cos \theta, \\ \|\mathbf{q}_v\| &= \sin \theta. \end{aligned} \quad (3.32)$$

Substituindo (3.32) em (3.31) obtemos

$$\begin{aligned} \mathbf{b}_2 &= (\cos^2 \theta - \sin^2 \theta)\mathbf{a}_2 + (2 \cos \theta \sin \theta)\mathbf{a}_3 \\ &= (\cos 2\theta)\mathbf{a}_2 + (\sin 2\theta)\mathbf{a}_3, \end{aligned} \quad (3.33)$$

onde $\mathbf{a}_3 = \mathbf{u} \times \mathbf{a}_2$. Como

$$\|\mathbf{a}_3\| = \|\mathbf{u} \times \mathbf{a}_2\| = \|\mathbf{u}\|\|\mathbf{a}_2\| \sin\left(\frac{\pi}{2}\right) = \|\mathbf{a}_2\|,$$

concluimos que \mathbf{b}_2 é uma rotação de 2θ do vetor \mathbf{a}_2 em torno do eixo gerado por \mathbf{q}_v . Portanto, o vetor obtido por rodar \mathbf{v} de um ângulo 2θ em torno do eixo gerado por \mathbf{q}_v é dado pela parte vetorial do quatérnio $L(\mathbf{v}_a) = (0, \mathbf{a}_1 + \mathbf{b}_2)^T$. Denotando por $\mathbf{v}_R = \mathbf{a}_1 + \mathbf{b}_2$ o vetor rodado, podemos perceber que após a rotação, a componente \mathbf{a}_1 da decomposição ortogonal de \mathbf{v} ao longo de \mathbf{q}_v não se altera, enquanto que a projeção de \mathbf{v} no plano perpendicular a \mathbf{q}_v sofre uma rotação de 2θ , conforme ilustrado na Figura 3.6. ■

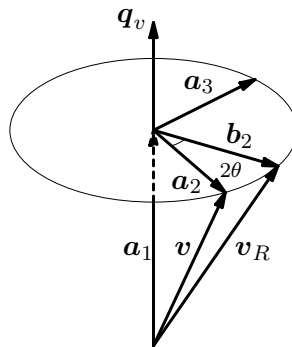


Figura 3.6: Interpretando geometricamente a ação do operador L .

Escrevendo o vetor rodado $\mathbf{v}_R = \mathbf{a}_1 + \mathbf{b}_2$ em notação matricial temos

$$\mathbf{v}_R = Q\mathbf{v}, \quad (3.34)$$

onde $Q(\mathbf{q}) \in SO(3)$ é construída com as componentes do quatérnion \mathbf{q} , isto é,

$$Q(\mathbf{q}) = \begin{pmatrix} 2q_0^2 + 2q_1^2 - 1 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 2q_0^2 + 2q_2^2 - 1 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 2q_0^2 + 2q_3^2 - 1 \end{pmatrix}. \quad (3.35)$$

Podemos então estabelecer uma correspondência entre o conjunto dos quatérnions unitários S^3 e o conjunto $SO(3)$ através da seguinte aplicação:

$$\begin{aligned} \Phi : S^3 &\rightarrow SO(3) \\ \mathbf{q} &\mapsto Q(\mathbf{q}) \end{aligned} \quad (3.36)$$

Pelo Teorema (3.2), como $\theta \in (0, \pi)$, o ângulo da rotação ocorrida satisfaz $0 < 2\theta < 2\pi$. Ora, se a rotação de ϕ em torno de \mathbf{q}_v equivale a uma rotação de $2\pi - \phi$ em torno de $-\mathbf{q}_v$, cada rotação não trivial (i.e., diferente da identidade) pode ser representada por dois quatérnions unitários: $\mathbf{q} = (q_0, \mathbf{q})^T$ e $-\mathbf{q} = (-q_0, -\mathbf{q})^T$. Por outro lado, a rotação identidade corresponde também a dois quatérnions unitários, a saber $(1, 0, 0, 0)^T$ e $(-1, 0, 0, 0)^T$. Assim, cada elemento de $SO(3)$ é imagem pela aplicação (3.36) de exatamente dois elementos de S^3 . Isto significa que Φ é sobrejetiva e *localmente* injetiva. É interessante contrastar esta situação com a que ocorria com a parametrização por ângulos de Euler. A aplicação $A(\psi, 0, \phi)$ definida em (3.16), por exemplo, é tal que $A(\phi, 0, -\phi) = I$ para qualquer ϕ . Portanto é impossível escolher uma vizinhança em torno da origem de modo que A seja injetiva nessa vizinhança. Por outro lado, se restringirmos o domínio para $(0, 2\pi) \times (0, \pi) \times (0, 2\pi)$, a matriz identidade deixa de fazer parte da imagem de A . Com quatérnions, entretanto, embora o mapa Φ definido em (3.36) não seja injetivo, cada matriz em $SO(3)$ é imagem por Φ de dois pontos isolados, o que elimina problemas de singularidade.

3.5 Comparando as representações

Nas seções anteriores vimos como representar uma matriz de rotação por meio do mapa exponencial (representação eixo-ângulo), dos ângulos de Euler e dos quatérnions unitários. Nessa seção obteremos fórmulas de conversão entre as representações estudadas e discutiremos a eficiência computacional de cada uma delas.

3.5.1 Fórmulas de conversão

No desenvolvimento das fórmulas de conversão entre as representações, vamos supor que o ângulo de rotação θ pertence ao intervalo $(0, \pi)$.

- (1) Eixo-ângulo \Leftrightarrow Quatérnions unitários

Se $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)^T$ e θ são, respectivamente, o eixo e o ângulo da rotação, de acordo com o Teorema 3.2 o quatérnio unitário correspondente a essa rotação é dado por

$$\mathbf{q} = \left(\cos \left(\frac{\theta}{2} \right), \boldsymbol{\omega} \operatorname{sen} \left(\frac{\theta}{2} \right) \right)^T. \quad (3.37)$$

Reciprocamente, se $\mathbf{q} = (q_0, q_1, q_2, q_3)^T$ é um quatérnio unitário tal que $|q_0| < 1$, podemos extrair das componentes de \mathbf{q} o ângulo e o eixo da rotação da seguinte forma:

$$\begin{aligned} \theta &= 2 \cos^{-1}(q_0), \\ \boldsymbol{\omega} &= \frac{1}{\sqrt{1 - q_0^2}} (q_1, q_2, q_3)^T. \end{aligned} \quad (3.38)$$

(2) Eixo-ângulo \Leftrightarrow Ângulos de Euler

Dado o par $(\boldsymbol{\omega}, \theta)$, utilizamos a fórmula de Rodrigues (3.7) para construir a matriz (3.9) que corresponde a à rotação de θ em torno de $\boldsymbol{\omega}$. Então, para obtermos os ângulos de Euler referentes a uma seqüência particular, igualamos (3.9) à matriz obtida pela composição das rotações da seqüência e extraímos os ângulos como fizemos no exemplo da Seção 3.3. Reciprocamente, dada uma matriz de rotação R que representa uma seqüência de Euler qualquer, para obtermos o ângulo e o eixo correspondentes a essa rotação, utilizamos, respectivamente, as fórmulas (3.11) e (3.12).

(3) Ângulos de Euler \Leftrightarrow Quatérnions unitários

Neste caso, as fórmulas a serem obtidas dependem da escolha de uma seqüência de Euler particular. Para ilustrar o procedimento utilizaremos a seqüência ZXZ discutida no exemplo da Seção 3.3. Esta seqüência é formada pela composição de três rotações simples: uma rotação de ψ em torno de Z , seguida por uma rotação de θ em torno de X , terminando com uma rotação de ϕ novamente em torno de Z . Os quatérnions unitários que representam cada rotação simples são dados por:

$$\begin{aligned} \mathbf{q}_{z,\psi} &= \left(\cos \left(\frac{\psi}{2} \right), \mathbf{e}_3 \operatorname{sen} \left(\frac{\psi}{2} \right) \right)^T, \\ \mathbf{q}_{x,\theta} &= \left(\cos \left(\frac{\theta}{2} \right), \mathbf{e}_1 \operatorname{sen} \left(\frac{\theta}{2} \right) \right)^T, \\ \mathbf{q}_{z,\phi} &= \left(\cos \left(\frac{\phi}{2} \right), \mathbf{e}_3 \operatorname{sen} \left(\frac{\phi}{2} \right) \right)^T. \end{aligned} \quad (3.39)$$

onde $\mathbf{e}_1 = (1, 0, 0)^T$ e $\mathbf{e}_3 = (0, 0, 1)^T$. Utilizando as regras de multiplicação (3.2) e os quatérnions (3.39), construímos um quatérnio \mathbf{q}_R que representa a seqüência de rotações ZXZ :

$$\mathbf{q}_R = \mathbf{q}_{z,\phi} \mathbf{q}_{x,\theta} \mathbf{q}_{z,\psi} = (q_0, q_1, q_2, q_3)^T, \quad (3.40)$$

onde

$$\begin{aligned}
 q_0 &= \cos\left(\frac{\phi}{2}\right) \cos\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) - \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\phi}{2}\right) \sin\left(\frac{\psi}{2}\right), \\
 q_1 &= \cos\left(\frac{\phi}{2}\right) \cos\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) + \sin\left(\frac{\phi}{2}\right) \sin\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right), \\
 q_2 &= \cos\left(\frac{\phi}{2}\right) \sin\left(\frac{\psi}{2}\right) \sin\left(\frac{\theta}{2}\right) - \cos\left(\frac{\psi}{2}\right) \sin\left(\frac{\phi}{2}\right) \sin\left(\frac{\theta}{2}\right), \\
 q_3 &= \cos\left(\frac{\psi}{2}\right) \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\phi}{2}\right) + \cos\left(\frac{\phi}{2}\right) \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\psi}{2}\right).
 \end{aligned} \tag{3.41}$$

Então, usamos \mathbf{q}_R para definir um operador de rotação cujo efeito sob um vetor de \mathbb{R}^3 seja análogo ao da seqüência de Euler ZXZ . Agora, dado um quatérnio unitário $\mathbf{q} = (q_0, q_1, q_2, q_3)^T$, para determinar os ângulos de Euler que definem a seqüência ZXZ , procedemos da mesma forma que em (3.18).

3.5.2 Eficiência computacional

Depois de estudarmos três diferentes maneiras de representar uma rotação e como obter um conjunto de parâmetros a partir de outro por meio de fórmulas de conversão, nos deparamos com a seguinte questão:

Qual é a melhor maneira de descrever uma rotação?

Infelizmente essa é uma pergunta sem resposta. Tudo dependerá do problema em estudo. Ou seja, o uso de ângulos de Euler, quatérnios unitários ou do mapa exponencial para representar uma matriz de rotação pode ou não ser eficiente em um determinado problema. Existem na literatura, por exemplo, problemas em que as rotações envolvidas são inicialmente representadas de uma maneira, e, em determinados cálculos intermediários, uma fórmula de conversão é utilizada para se obter outra representação para a rotação. Existem também aqueles que usam determinada representação por acharem-na mais intuitiva. Por exemplo, os profissionais de aeronáutica já estão acostumados a descrever a orientação do avião usando três ângulos de Euler: *pitch* (inclinação do bico), *yaw* (esquerda/direita) e *roll* (rotação em torno do eixo longitudinal do avião). Então, o uso continuado de um tipo de representação faz com que ela se torne mais intuitiva, e, portanto, preferida às outras, por um grupo de pessoas.

Faremos agora uma discussão levando em consideração o esforço computacional despendido por cada representação estudada. Analisaremos o uso de memória e as operações necessárias para converter de uma representação para outra e compor rotações. Consideramos para efeito de comparação as operações de adição e/ou subtração (\pm), multiplicação (\times), divisão (\div) e o número de vezes que uma função trigonométrica é avaliada em cada representação (*ntrig*).

A Tabela 3.3 mostra a dimensão dos vetores que devem ser reservados na memória de um computador para armazenar uma matriz de rotação e os parâmetros de cada representação. Na representação por ângulos de Euler os três ângulos devem ser guardados. No caso da representação eixo-ângulo, um vetor de dimensão 4 é suficiente para armazenar todos os parâmetros. Porém, será preciso definir um vetor de dimensão 6 se os valores $\sin \theta$ e $(1 - \cos \theta)$ também forem guardados. Na representação por quatérnions unitários também é preciso um vetor de dimensão 4 para alocar os parâmetros, porém, não há necessidade de se fazer cálculos com funções trigonométricas.

Representação	Dimensão	Comentários
matriz de rotação	9	
ângulos de Euler	3	sem armazenar seno e cosseno dos ângulos
eixo-ângulo	4	sem armazenar $\sin \theta$ e $(1 - \cos \theta)$
eixo-ângulo	6	armazenando $\sin \theta$ e $(1 - \cos \theta)$
quatérnion	4	

Tabela 3.3: Comparação de uso de memória entre as representações.

A Tabela 3.4 traz o número de operações que são necessárias para construir uma matriz de rotação R a partir dos parâmetros e o número de operações realizadas na conversão de uma representação para outra. Devemos notar que a construção de uma matriz de rotação pela fórmula de Rodrigues é a que mais realiza operações, porém, se R for obtida via ângulos de Euler, esta construção efetuará o maior número de avaliações de função trigonométrica desde que os 3 ângulos da seqüência sejam distintos. A obtenção de R via quatérnions unitários não requer nenhum cálculo de função trigonométrica pois todos os cálculos envolvidos são algébricos. Nesta tabela estamos considerando também o esforço computacional nas conversões entre as representações. Notemos que o maior número de multiplicações e avaliações de função trigonométrica acontece na transformação de ângulos de Euler para quatérnion.

A operação de pré-multiplicar um vetor $\mathbf{v} \in \mathbb{R}^3$ por uma matriz de rotação R requer 6 adições e 9 multiplicações. Dependendo do modo como a matriz R for representada, esses valores aumentam significativamente. Se a fórmula de Rodrigues (3.7) for utilizada para computar R , dado o par $(\boldsymbol{\omega}, \theta)$, o produto $R\mathbf{v}$ pode ser calculado da seguinte forma:

$$R\mathbf{v} = \mathbf{v} + \sin \theta (\boldsymbol{\omega} \times \mathbf{v}) + (1 - \cos \theta) \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{v}). \quad (3.42)$$

Em (3.42), o produto vetorial $(\boldsymbol{\omega} \times \mathbf{v})$ é calculado uma única vez e é temporariamente armazenado. Para computá-lo, são necessárias 3 adições e 6 multiplicações. Assim, o cálculo do vetor $R\mathbf{v}$ a partir da expressão (3.42) requer um total de 12 adições e 18 multiplicações. Se ao invés da matriz R utilizarmos o operador de rotação construído com um quatérnion unitário \mathbf{q} , serão necessárias 17 adições e 24 multiplicações para efetuar o produto $\mathbf{q}\mathbf{v}\mathbf{q}^*$.

Tipo de cálculo	\pm	\times	\div	<i>ntrig</i>	Total
R via fórmula de Rodrigues	13	15		2	30
R via ângulos de Euler	4	16		6	26
R via quatérnions unitários	12	12			24
eixo-ângulo \rightarrow quatérnion		4		2	6
quatérnion \rightarrow eixo-ângulo	1	5	1	1	8
eixo-ângulo \rightarrow ângulos de Euler			2	3	5
ângulos de Euler \rightarrow eixo-ângulo	6	2	3	2	13
ângulos de Euler \rightarrow quatérnion	4	19		6	29
quatérnion \rightarrow ângulos de Euler			2	3	5

Tabela 3.4: Comparação do número de operações realizadas.

Por outro lado, o produto $\mathbf{q}\mathbf{v}_q\mathbf{q}^*$ pode ser escrito em notação matricial e a construção da matriz de rotação Q a partir das componentes do quatérnion \mathbf{q} requer 24 operações (12 adições e 12 multiplicações). Como o produto matriz-vetor requer 15 operações (6 adições e 9 multiplicações), teremos um total de 39 operações no cálculo de $Q\mathbf{v}$. Estas informações estão resumidas na Tabela 3.5. Notemos que o produto matriz-vetor na formulação via quatérnions é o que requer o maior número de operações, porém, se a matriz Q for aplicada a um conjunto de n vetores, o número total de operações realizadas será de $24 + 15n$ e nas formulações eixo-ângulo e ângulos de Euler, esses valores serão $30n$ e $26 + 15n$, respectivamente. Essa seção resumiu material que pode ser encontrado em [8, 12].

Representação	\pm	\times	Total	Comentários
matriz de rotação	6	9	15	computando somente $R\mathbf{v}$
eixo-ângulo	12	18	30	efetuando $R\mathbf{v}$ pela expressão (3.42)
quatérnion	17	24	41	usando a fórmula de multiplicação (3.22)
quatérnion	18	21	39	construindo a matriz Q e efetuando $Q\mathbf{v}$

Tabela 3.5: Número de operações necessárias para rodar um vetor.

A representação de uma rotação via mapa exponencial é mais intuitiva que a representação por ângulos de Euler. É muito mais fácil imaginarmos um elemento de $SO(3)$ como uma rotação de uma quantidade θ em torno de um eixo fixo do que pensarmos que este elemento é obtido por uma seqüência de três rotações simples em torno dos eixos coordenados. Porém, em ambas as representações é necessário avaliar funções trigonométricas, e este tipo de

cálculo requer esforço computacional. No entanto, se utilizarmos um quatérnio unitário para descrever uma rotação não precisaremos nos preocupar com as avaliações de funções trigonométricas pois estes cálculos são substituídos por operações com polinômios.

No capítulo a seguir, estudaremos dois problemas de otimização que envolvem rotações. Formularemos cada problema utilizando duas maneiras diferentes de representar as rotações e resolveremos os problemas formulados, comparando os resultados obtidos.

Capítulo 4

Aplicações

Neste capítulo estudamos dois problemas de otimização que envolvem rotações no espaço tridimensional: o problema da orientação absoluta e o problema do empacotamento de moléculas. Em ambos os problemas, as orientações foram caracterizadas inicialmente via ângulos de Euler [22, 32]. Reformulamos o primeiro problema utilizando quatérnions para representar as rotações e mostramos que com essa escolha uma solução analítica pode ser obtida para o problema. Empregamos também um método numérico na resolução deste problema e comparamos os resultados obtidos com a solução analítica. No segundo problema, também utilizamos quatérnions para descrever as rotações envolvidas e comparamos a eficiência do método empregado para resolvê-lo em ambas as formulações.

Todos os problemas são resolvidos pelo pacote de otimização ALGENCAN [1, 2], que emprega um algoritmo de Lagrangiano aumentado, incorporando uma técnica denominada *Newton Truncado* e o conceito de gradiente projetado. A máquina utilizada para rodar os testes numéricos deste capítulo possui dois processadores Opteron 242 com 1.6 MHz, 1 Mb de cache e 1.5 Gb de memória RAM.

4.1 O problema da orientação absoluta

O problema da orientação absoluta, amplamente utilizado em aplicações envolvendo robótica e fotogrametria (ver por exemplo [14] e [32]), consiste em recuperar a relação entre dois sistemas de coordenadas tridimensionais a partir das coordenadas cartesianas de um conjunto de n pontos medidas com relação a cada sistema. Assim, denotando os sistemas de coordenadas por S_e e S_d e representando o vetor de coordenadas do i -ésimo ponto com relação a cada sistema por $\mathbf{r}_e^i = (x_e^i, y_e^i, z_e^i)^T$ e $\mathbf{r}_d^i = (x_d^i, y_d^i, z_d^i)^T$, respectivamente, procuramos uma transformação afim $T : S_e \rightarrow S_d$ tal que

$$T(\mathbf{r}_e^i) = sR\mathbf{r}_e^i + \mathbf{d}_0 = \mathbf{r}_d^i, \quad i = 1, \dots, n, \quad (4.1)$$

onde $0 < s \in \mathbb{R}$ é um fator de escala, $\mathbf{d}_0 \in \mathbb{R}^3$ é um vetor deslocamento e R é uma matriz de rotação. Para determinarmos T completamente, precisamos encontrar s , \mathbf{d}_0 e R que satisfaça (4.1) para todo $i = 1, \dots, n$. Como podemos observar, o sistema de equações (4.1)

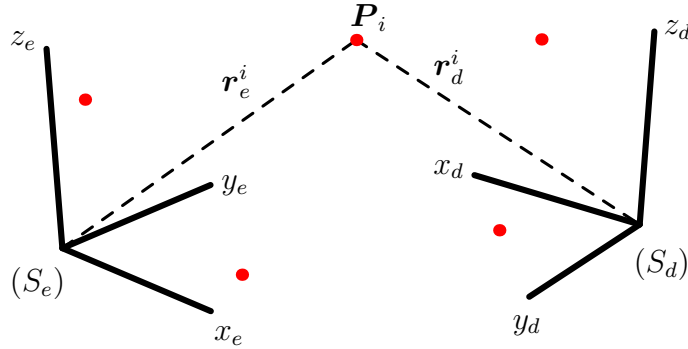


Figura 4.1: Representação de um ponto com relação aos sistemas S_e e S_d .

é não linear e, em geral, uma solução exata para sistemas não lineares é difícil de ser obtida. Por essa razão, a maioria dos métodos existentes para resolver o problema da orientação absoluta são iterativos, isto é, encontram aproximações para s , R e \mathbf{d}_0 . A desvantagem de se empregar um método iterativo para resolver o sistema de equações (4.1) está no fato de que a aproximação obtida para a matriz R pode não ser exatamente ortonormal devido a instabilidades numéricas. Na tentativa de evitar métodos iterativos, Berthold Horn em seu trabalho [13] constrói a partir das equações (4.1) um problema de quadrados mínimos não linear e o resolve analiticamente, determinando expressões para o fator de escala s , o vetor deslocamento \mathbf{d}_0 e a matriz de rotação R . Na formulação proposta por Horn, a matriz de rotação R é construída a partir das componentes de um quatérnio unitário.

A seguir, formularemos o problema de quadrados mínimos não linear e o resolveremos de duas formas:

- a) Analiticamente, seguindo o desenvolvimento de Horn.
- b) Numericamente, utilizando o pacote ALGENCAN.

Para resolver o problema numericamente, empregaremos ângulos de Euler e quatérnios na construção da matriz de rotação R . Por fim, compararemos as soluções numéricas obtidas com a solução analítica do método proposto por Horn.

4.1.1 Formulação e resolução do problema

Dados os conjuntos de vetores $V_e = \{\mathbf{r}_e^1, \dots, \mathbf{r}_e^n\}$ e $V_d = \{\mathbf{r}_d^1, \dots, \mathbf{r}_d^n\}$ medidos com relação aos sistemas de coordenadas S_e e S_d respectivamente, nosso objetivo é encontrar expressões para s , R e \mathbf{d}_0 por meio da resolução de um problema de quadrados mínimos não linear de modo que a transformação T que relaciona S_e e S_d fique completamente determinada. Assim, definindo para cada $i = 1, \dots, n$ um vetor erro $\boldsymbol{\varepsilon}_i$ dado por

$$\boldsymbol{\varepsilon}_i = \mathbf{r}_d^i - T(\mathbf{r}_e^i) = \mathbf{r}_d^i - sR\mathbf{r}_e^i - \mathbf{d}_0, \quad i = 1, \dots, n, \quad (4.2)$$

o problema que devemos resolver consiste em minimizar o somatório do quadrado das normas dos vetores $\boldsymbol{\varepsilon}_i$, ou seja,

$$\min_{s,R,\mathbf{d}_0} \sum_{i=1}^n \|\boldsymbol{\varepsilon}_i\|^2. \quad (4.3)$$

Obtendo a solução analítica

Para resolver o problema (4.3) analiticamente introduziremos uma mudança de variáveis que nos auxiliará nos cálculos que seguem. Começamos definindo os *centróides* dos conjuntos V_e e V_d como sendo os vetores

$$\begin{aligned} \mathbf{c}_e &= \frac{1}{n} \sum_{i=1}^n \mathbf{r}_e^i, \\ \mathbf{c}_d &= \frac{1}{n} \sum_{i=1}^n \mathbf{r}_d^i, \end{aligned} \quad (4.4)$$

e calculamos novos vetores com relação aos respectivos centróides, isto é,

$$\begin{aligned} \hat{\mathbf{r}}_e^i &= \mathbf{r}_e^i - \mathbf{c}_e, \\ \hat{\mathbf{r}}_d^i &= \mathbf{r}_d^i - \mathbf{c}_d, \end{aligned} \quad (4.5)$$

para $i = 1, \dots, n$. Observemos que desta construção segue que

$$\sum_{i=1}^n \hat{\mathbf{r}}_e^i = \mathbf{0} \quad \text{e} \quad \sum_{i=1}^n \hat{\mathbf{r}}_d^i = \mathbf{0}. \quad (4.6)$$

Desta forma, utilizando os vetores (4.5) podemos reescrever cada vetor erro $\boldsymbol{\varepsilon}_i$ como

$$\boldsymbol{\varepsilon}_i = \hat{\mathbf{r}}_d^i - sR\hat{\mathbf{r}}_e^i - \hat{\mathbf{d}}_0, \quad (4.7)$$

onde $\hat{\mathbf{d}}_0$ é dado por

$$\hat{\mathbf{d}}_0 = \mathbf{d}_0 - \mathbf{c}_d + sR\mathbf{c}_e. \quad (4.8)$$

Com essa mudança de variáveis, o problema de minimização (4.3) torna-se

$$\min_{s,R,\hat{\mathbf{d}}_0} \sum_{i=1}^n \|\hat{\mathbf{r}}_d^i - sR\hat{\mathbf{r}}_e^i - \hat{\mathbf{d}}_0\|^2. \quad (4.9)$$

1) Cálculo do vetor deslocamento

Desenvolvendo a função objetivo de (4.9) obtemos

$$\sum_{i=1}^n \|\hat{\mathbf{r}}_d^i - sR\hat{\mathbf{r}}_e^i\|^2 - 2\langle \hat{\mathbf{d}}_0, \sum_{i=1}^n \hat{\mathbf{r}}_d^i \rangle + 2s\langle \hat{\mathbf{d}}_0, \sum_{i=1}^n R\hat{\mathbf{r}}_e^i \rangle + n\|\hat{\mathbf{d}}_0\|^2. \quad (4.10)$$

Substituindo em (4.10) as equações (4.6), esta expressão se reduz a

$$n\|\hat{\mathbf{d}}_0\|^2 + \sum_{i=1}^n \|\hat{\mathbf{r}}_d^i - sR\hat{\mathbf{r}}_e^i\|^2, \quad (4.11)$$

pois os dois termos envolvendo produto interno se anulam. Como apenas o primeiro termo de (4.11) depende de $\hat{\mathbf{d}}_0$, concluímos que esta função é minimizada com relação a este vetor quando $\hat{\mathbf{d}}_0 = 0$. Assim, de (4.8) segue que

$$\mathbf{d}_0 = \mathbf{c}_d - sR\mathbf{c}_e. \quad (4.12)$$

Então, para determinarmos o fator de escala s e a matriz de rotação R devemos agora resolver o problema

$$\min_{s,R} \sum_{i=1}^n \|\hat{\mathbf{r}}_d^i - sR\hat{\mathbf{r}}_e^i\|^2. \quad (4.13)$$

2) Cálculo do fator de escala

Expandindo a função objetivo em (4.13) e usando o fato de que uma matriz de rotação preserva a norma euclidiana de vetores, isto é, $\|R\mathbf{v}\| = \|\mathbf{v}\|$, obtemos

$$\alpha_d - 2sD + s^2\alpha_e, \quad (4.14)$$

onde

$$\alpha_d = \sum_{i=1}^n \|\hat{\mathbf{r}}_d^i\|^2, \quad D = \sum_{i=1}^n \langle \hat{\mathbf{r}}_d^i, R\hat{\mathbf{r}}_e^i \rangle, \quad \text{e} \quad \alpha_e = \sum_{i=1}^n \|\hat{\mathbf{r}}_e^i\|^2. \quad (4.15)$$

Reescrita na forma (4.14) a função objetivo do problema (4.13) torna-se uma quadrática na variável s . Como $\alpha_e > 0$, o valor de s que a minimiza (4.14) é dado por

$$s = \frac{D}{\alpha_e}. \quad (4.16)$$

A fórmula (4.16) é apropriada no caso em que o conjunto de vetores V_e é conhecido com maior precisão que o conjunto V_d . Se a precisão de V_e e V_d é a mesma, substituímos a fórmula do erro (4.7) por

$$\hat{\boldsymbol{\varepsilon}}_i = \frac{1}{\sqrt{s}}(\hat{\mathbf{r}}_d^i - sR\hat{\mathbf{r}}_e^i - \hat{\mathbf{d}}_0), \quad i = 1, \dots, n, \quad (4.17)$$

e, neste caso, resolvemos o problema

$$\min_{s,R,\hat{\mathbf{d}}_0} \|\hat{\boldsymbol{\varepsilon}}_i\|^2. \quad (4.18)$$

Deste modo, procedendo como anteriormente, desenvolvemos a função objetivo de (4.18) para obter $\hat{\mathbf{d}}_0 = 0$ e

$$s = \left[\frac{\alpha_d}{\alpha_e} \right]^{1/2}. \quad (4.19)$$

A vantagem de se utilizar a fórmula (4.19) para o cálculo do fator de escala é que ela não depende da matriz de rotação R .

3) Cálculo da matriz de rotação

Como em (4.14) o único coeficiente que depende da matriz R é D , o valor de R que minimizará a função em (4.13) será aquele que torna o valor de D tão grande quanto possível, ou seja, que maximiza D . Seguindo a idéia proposta em [13], a quantidade D é reescrita de modo que a rotação envolvida seja representada via quatérnions. Para isso, substituímos a matriz R pelo operador de rotação $L(\cdot) = \mathbf{q}(\cdot)\mathbf{q}^*$ construído a partir de um quatérnion unitário \mathbf{q} e para cada par de vetores $\hat{\mathbf{r}}_e^i = (\hat{x}_e^i, \hat{y}_e^i, \hat{z}_e^i)^T$, $\hat{\mathbf{r}}_d^i = (\hat{x}_d^i, \hat{y}_d^i, \hat{z}_d^i)^T$, definimos os quatérnions puros

$$\begin{aligned}\hat{\mathbf{v}}_e^i &= (0, \hat{\mathbf{r}}_e^i)^T = (0, \hat{x}_e^i, \hat{y}_e^i, \hat{z}_e^i)^T, \\ \hat{\mathbf{v}}_d^i &= (0, \hat{\mathbf{r}}_d^i)^T = (0, \hat{x}_d^i, \hat{y}_d^i, \hat{z}_d^i)^T.\end{aligned}\tag{4.20}$$

Como o produto interno entre quatérnions é análogo ao produto interno entre vetores em \mathbb{R}^4 , podemos reescrever D da seguinte forma

$$D = \sum_{i=1}^n \langle \hat{\mathbf{v}}_d^i, (\mathbf{q}\hat{\mathbf{v}}_e^i\mathbf{q}^*) \rangle.\tag{4.21}$$

Portanto, ao invés de encontrar uma matriz de rotação R que maximiza D , nossa tarefa passa a ser a de encontrar um quatérnion unitário \mathbf{q} que torna (4.21) tão grande quanto possível.

Usando uma propriedade da álgebra de quatérnions, escrevemos (4.21) como

$$D = \sum_{i=1}^n \langle (\mathbf{q}\hat{\mathbf{v}}_e^i), (\hat{\mathbf{v}}_d^i\mathbf{q}) \rangle.\tag{4.22}$$

Os produtos entre quatérnions que aparecem em (4.22) podem ser escritos em notação matricial. Ou seja, para $i = 1, \dots, n$ temos:

$$\begin{aligned}\mathbf{q}\hat{\mathbf{v}}_e^i &= M_e^i\mathbf{q}, \\ \hat{\mathbf{v}}_d^i\mathbf{q} &= M_d^i\mathbf{q},\end{aligned}\tag{4.23}$$

onde

$$M_e^i = \begin{pmatrix} 0 & -\hat{x}_e^i & -\hat{y}_e^i & -\hat{z}_e^i \\ \hat{x}_e^i & 0 & \hat{z}_e^i & -\hat{y}_e^i \\ \hat{y}_e^i & -\hat{z}_e^i & 0 & \hat{x}_e^i \\ \hat{z}_e^i & \hat{y}_e^i & -\hat{x}_e^i & 0 \end{pmatrix} \quad \text{e} \quad M_d^i = \begin{pmatrix} 0 & -\hat{x}_d^i & -\hat{y}_d^i & -\hat{z}_d^i \\ \hat{x}_d^i & 0 & -\hat{z}_d^i & \hat{y}_d^i \\ \hat{y}_d^i & \hat{z}_d^i & 0 & -\hat{x}_d^i \\ \hat{z}_d^i & -\hat{y}_d^i & \hat{x}_d^i & 0 \end{pmatrix}.\tag{4.24}$$

Substituindo (4.23) em (4.22) temos que

$$\begin{aligned} D &= \sum_{i=1}^n \langle M_e^i \mathbf{q}, M_d^i \mathbf{q} \rangle = \sum_{i=1}^n \mathbf{q}^T (M_e^i)^T M_d^i \mathbf{q} \\ &= \mathbf{q}^T \left(\sum_{i=1}^n (M_e^i)^T M_d^i \right) \mathbf{q} = \mathbf{q}^T N \mathbf{q}. \end{aligned} \quad (4.25)$$

Como $(M_e^i)^T M_d^i$ é uma matriz simétrica para cada i , a matriz N também é simétrica. Devemos notar que a substituição da matriz R pelo operador de rotação construído com o quatérnio unitário \mathbf{q} permitiu expressar a quantidade D como uma função quadrática em \mathbf{q} . A proposição a seguir garante que existe um quatérnio unitário $\bar{\mathbf{q}}$ que maximiza (4.25) e mostra como calculá-lo.

Proposição 4.1 *O quatérnio $\bar{\mathbf{q}}$ que maximiza (4.25) é o autovetor correspondente ao autovalor mais positivo da matriz N .*

Demonstração: Com efeito, como N é uma matriz simétrica segue que N é ortogonalmente diagonalizável, isto é, existe um conjunto ortonormal de autovetores $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$ com respectivos autovalores reais $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$. Dado um quatérnio unitário \mathbf{q} arbitrário, podemos escrever

$$\mathbf{q} = \sum_{i=1}^4 \beta_i \mathbf{v}_i \quad \text{e} \quad \|\mathbf{q}\|^2 = \sum_{i=1}^4 \beta_i^2 = 1.$$

Supondo que λ_1 é o autovalor mais positivo de N temos que

$$\mathbf{q}^T N \mathbf{q} = \sum_{i=1}^4 \lambda_i \beta_i^2 \leq \lambda_1 \sum_{i=1}^4 \beta_i^2 = \lambda_1.$$

Portanto, para $\bar{\mathbf{q}} = \mathbf{v}_1$ obtemos

$$\bar{\mathbf{q}}^T N \bar{\mathbf{q}} = \mathbf{v}_1^T N \mathbf{v}_1 = \lambda_1 \mathbf{v}_1^T \mathbf{v}_1 = \lambda_1,$$

que completa a demonstração. ■

O resultado da Proposição 4.1 nos diz que o problema de encontrar o quatérnio $\bar{\mathbf{q}}$ que maximiza a quantidade D se reduz ao cálculo dos autovalores e autovetores da matriz N . Com as componentes de $\bar{\mathbf{q}}$, a matriz R pode ser obtida da seguinte forma

$$R = \begin{pmatrix} 2(\bar{q}_0^2 + \bar{q}_1^2) - 1 & 2(\bar{q}_1 \bar{q}_2 - \bar{q}_0 \bar{q}_3) & 2(\bar{q}_1 \bar{q}_3 + \bar{q}_0 \bar{q}_2) \\ 2(\bar{q}_1 \bar{q}_2 + \bar{q}_0 \bar{q}_3) & 2(\bar{q}_0^2 + \bar{q}_2^2) - 1 & 2(\bar{q}_2 \bar{q}_3 - \bar{q}_0 \bar{q}_1) \\ 2(\bar{q}_1 \bar{q}_3 - \bar{q}_0 \bar{q}_2) & 2(\bar{q}_2 \bar{q}_3 + \bar{q}_0 \bar{q}_1) & 2(\bar{q}_0^2 + \bar{q}_3^2) - 1 \end{pmatrix}. \quad (4.26)$$

Por fim, R é utilizada nos cálculos do fator de escala s e do vetor deslocamento \mathbf{d}_0 .

Seguindo os cálculos anteriores, apresentaremos agora um algoritmo que determina a transformação T completamente encontrando valores para o fator de escala s , o vetor deslocamento \mathbf{d}_0 e a matriz de rotação R .

Algoritmo 4.1 *Dado um conjunto de n pontos em \mathbb{R}^3 e as coordenadas cartesianas de cada ponto $\mathbf{r}_e^i, \mathbf{r}_d^i, i = 1, \dots, n$, medidas, respectivamente, com relação a dois sistemas S_e e S_d , os passos para determinar R, \mathbf{d}_0 e s são:*

- (1) *Calcular os centróides $\mathbf{c}_e, \mathbf{c}_d$ pelas equações (4.4) e os novos vetores $\hat{\mathbf{r}}_e^i$ e $\hat{\mathbf{r}}_d^i$ pelas equações (4.5).*
- (2) *Construir a matriz $N = \sum_{i=1}^n (M_e^i)^T M_d^i$ a partir das matrizes (4.24).*
- (3) *Determinar o autovetor $\bar{\mathbf{q}}$ associado ao autovalor mais positivo de N .*
- (4) *Obter a matriz de rotação R a partir das componentes de $\bar{\mathbf{q}}$ de acordo com (4.26).*
- (5) *Calcular \mathbf{d}_0 e s respectivamente pelas equações (4.12) e (4.16) ou (4.12) e (4.19), dependendo da precisão dos conjuntos V_e e V_d .*

No passo (2) do Algoritmo 4.1 o cálculo da matriz N torna-se mais fácil se antes determinarmos os elementos da matriz

$$B = \sum_{i=1}^n \hat{\mathbf{r}}_e^i (\hat{\mathbf{r}}_d^i)^T = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix},$$

onde

$$S_{xx} = \sum_{i=1}^n \hat{x}_e^i \hat{x}_d^i, \quad S_{xy} = \sum_{i=1}^n \hat{x}_e^i \hat{y}_d^i$$

e assim por diante. Deste modo, a matriz N pode ser escrita como função dos elementos de B da seguinte forma

$$N = \begin{pmatrix} (S_{xx} + S_{yy} + S_{zz}) & S_{yz} - S_{zy} & S_{zx} - S_{xz} & S_{xy} - S_{yx} \\ S_{yz} - S_{zy} & (S_{xx} - S_{yy} - S_{zz}) & S_{xy} + S_{yx} & S_{zx} + S_{xz} \\ S_{zx} - S_{xz} & S_{xy} + S_{yx} & (-S_{xx} + S_{yy} - S_{zz}) & S_{yz} + S_{zy} \\ S_{xy} - S_{yx} & S_{zx} + S_{xz} & S_{yz} + S_{zy} & (-S_{xx} - S_{yy} + S_{zz}) \end{pmatrix}$$

e, como N é simétrica, precisamos calcular apenas 10 de seus elementos.

O cálculo dos autovalores e autovetores da matriz N pode ser feito manualmente apesar de seu polinômio característico ter grau 4 (ver [13]). Porém, para determinar o quaternião $\bar{\mathbf{q}}$ no passo (3), utilizamos a rotina *Eigenvectors* do programa *Mathematica*.

No passo (5) o cálculo do fator de escala será feito de acordo com a precisão dos conjuntos de vetores V_e e V_d conforme descrevemos anteriormente. Se em uma determinada situação o conjunto V_d é mais preciso que V_e , para calcularmos o fator de escala devemos considerar a transformação inversa $\bar{T} : S_d \rightarrow S_e$, isto é,

$$\bar{T}(\mathbf{r}_d^i) = \bar{s}\bar{R}\mathbf{r}_d^i + \bar{\mathbf{d}}_0 = \mathbf{r}_e^i, \quad i = 1, \dots, n, \quad (4.27)$$

e, por cálculos semelhantes aos realizados com a transformação T obtemos

$$\bar{s} = \frac{\bar{D}}{\bar{\alpha}_d} \quad \text{onde} \quad \bar{D} = \sum_{i=1}^n \langle \hat{\mathbf{r}}_e^i, \bar{R}\hat{\mathbf{r}}_d^i \rangle \quad \text{e} \quad \bar{\alpha}_d = \alpha_d. \quad (4.28)$$

O Algoritmo 4.1 foi implementado no programa *Mathematica*. Os dados de entrada de nosso código são:

- O número total de pontos considerados ($npontos$).
- As matrizes $rEsq$ e $rDir$ de tamanho $npontos \times 3$ que armazenam, respectivamente, as coordenadas dos pontos com relação aos sistemas de coordenadas S_e e S_d .
- Um inteiro $flag$ cujo valor indicará a fórmula a ser usada para o cálculo do fator de escala: $flag = 0$, se queremos utilizar a fórmula (4.19) e $flag = 1$ para a fórmula (4.16).

Ao final da execução do código, os dados de saída são:

- o vetor deslocamento \mathbf{d}_0 .
- o fator de escala s .
- o quatérnion $\bar{\mathbf{q}}$ e a matriz de rotação R calculada a partir das componentes de $\bar{\mathbf{q}}$.

Os detalhes da implementação do Algoritmo 4.1 estão no apêndice A.

Obtendo a solução numérica

Utilizamos o pacote ALGENCAN para resolver o problema (4.3) numericamente. Formulamos este problema de duas maneiras: empregando ângulos de Euler e quatérnions para descrever a matriz de rotação R . Na formulação via ângulos de Euler, usamos a seqüência ZXZ na construção da matriz R . Isto faz com que o problema tenha 7 variáveis para serem determinadas: os 3 ângulos da seqüência de Euler, as 3 coordenadas do vetor \mathbf{d}_0 e o fator de escala s . Na formulação via quatérnions, a matriz de rotação R é construída com as componentes de um quatérnion não normalizado \mathbf{q} . Por essa razão, devemos incluir ao problema de otimização a restrição não linear de igualdade $\|\mathbf{q}\|^2 = 1$, para assegurar que o quatérnion tenha norma unitária. Assim, na formulação via quatérnions o problema terá 8 variáveis: as 4 componentes do quatérnion \mathbf{q} , o vetor \mathbf{d}_0 e o fator de escala s .

Como a função objetivo do problema que queremos resolver é dada pelo somatório do quadrado das normas dos vetores $\boldsymbol{\varepsilon}_i$, ou seja,

$$f(\mathbf{x}) = \sum_{i=1}^n \|\boldsymbol{\varepsilon}_i\|^2,$$

podemos reescrevê-la da seguinte forma

$$f(\mathbf{x}) = \|\mathbf{F}\|^2, \quad (4.29)$$

com $\mathbf{F} : \mathbb{R}^{n_v} \rightarrow \mathbb{R}^{3n}$ dada por

$$\mathbf{F} = \begin{pmatrix} \boldsymbol{\varepsilon}_1 \\ \boldsymbol{\varepsilon}_2 \\ \vdots \\ \boldsymbol{\varepsilon}_n \end{pmatrix},$$

onde n_v é o número de variáveis do problema (7 ou 8 dependendo da formulação) e n é o número de pontos considerados. Dessa forma, o vetor gradiente e a matriz hessiana de (4.29) são dados por

$$\begin{aligned} \nabla f(\mathbf{x}) &= 2J^T \mathbf{F}, \\ \nabla^2 f(\mathbf{x}) &= 2J^T J + 2 \sum_{i=1}^n \sum_{k=1}^3 \varepsilon_i^k \nabla^2 \varepsilon_i^k, \end{aligned} \quad (4.30)$$

onde $J \in \mathbb{R}^{3n \times n_v}$ é a matriz jacobiana da função vetorial \mathbf{F} , ε_i^k é a k -ésima componente do vetor $\boldsymbol{\varepsilon}_i$ e $\nabla^2 \varepsilon_i^k \in \mathbb{R}^{n_v \times n_v}$ é a matriz hessiana correspondente a ε_i^k .

Em muitos problemas de quadrados mínimos não lineares, podemos desprezar os termos de segunda ordem no cálculo da matriz hessiana e utilizarmos apenas a aproximação

$$\nabla^2 f(\mathbf{x}) \approx 2J^T J.$$

Isto é possível porque na maioria das vezes os resíduos ε_i^k são pequenos ou se comportam como funções lineares. A estratégia de utilizar apenas as derivadas de primeira ordem para calcular a matriz hessiana em um problema de quadrados mínimos não linear é conhecida como *estratégia Gauss-Newton*. Mais detalhes sobre esse assunto podem ser encontrados em [23]. Como o problema que queremos resolver é pequeno e o cálculo das derivadas de primeira e segunda ordem são relativamente simples, além de fornecermos para o programa ALGENCAN o gradiente analítico de (4.29), fornecemos também a expressão verdadeira da matriz hessiana.

4.1.2 Experimentos

Seguindo sugestões do autor do trabalho original [13], inventamos situações em que a transformação T que relaciona os dois sistemas de coordenadas S_e e S_d é conhecida exatamente. Desse modo, criamos um conjunto de vetores V_e medidos com relação a S_e e construímos o conjunto V_d aplicando a transformação (4.1) a cada vetor de V_e . Consideramos também em alguns testes a situação em que um conjunto de vetores é obtido com determinada imprecisão. Isto pode ser feito, por exemplo, perturbando as coordenadas de cada vetor do conjunto por um número real gerado aleatoriamente no intervalo $[0, 1]$.

De posse dos conjuntos de vetores V_e e V_d , resolvemos o problema da orientação absoluta aplicando primeiramente o Algoritmo 4.1 e em seguida o pacote de otimização ALGENCAN. Nas quatro situações propostas a seguir, o sistema de coordenadas S_e é o sistema canônico de \mathbb{R}^3 , isto é, o sistema formado pelos vetores $(1, 0, 0)^T$, $(0, 1, 0)^T$ e $(0, 0, 1)^T$.

Teste 1

Consideremos uma pirâmide cujos vértices com relação ao sistema S_e formam o conjunto

$$V_e = \{(0, 0, 0)^T, (1, 0, 0)^T, (1, 1, 0)^T, (0, 1, 0)^T, (0.5, 0.5, 1)^T\}.$$

Seja $T : S_e \rightarrow S_d$ uma transformação afim tal que

$$s = 2, \quad \mathbf{d}_0 = \begin{bmatrix} 2 \\ 5 \\ 4 \end{bmatrix} \quad \text{e} \quad R = \begin{bmatrix} 0.000000 & -0.500000 & 0.866025 \\ 0.707107 & 0.612372 & 0.353553 \\ -0.707107 & 0.612372 & 0.353553 \end{bmatrix}. \quad (4.31)$$

Para obter o conjunto V_d aplicamos T a cada elemento de V_e . Esta transformação tem o efeito de rodar e deslocar a pirâmide duplicando o seu tamanho original como mostra a Figura 4.2. Nesta figura também estão indicados os sistemas de coordenadas S_e e S_d . Neste exemplo

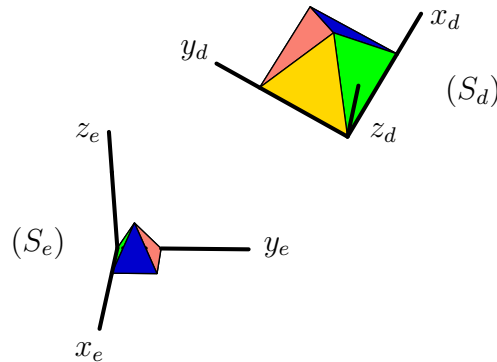


Figura 4.2: Pirâmide antes e após a aplicação da transformação T .

estamos admitindo que os conjuntos V_e e V_d têm a mesma precisão, portanto, podemos aplicar

o Algoritmo 4.1 utilizando a fórmula (4.19) para determinar o fator de escala. A Tabela 4.1 mostra os resultados obtidos (com seis casas decimais de precisão) quando o Algoritmo 4.1 e o pacote ALGENCAN são aplicados na resolução deste problema. Os vetores \mathbf{q} e Θ denotam, respectivamente, o quatérnio e os ângulos de Euler (em radianos) que descrevem a rotação. Notemos que o quatérnio resultante da aplicação do Algoritmo 4.1 difere apenas em sinal do quatérnio encontrado por ALGENCAN. Com efeito, existem apenas dois quatérnios unitários que descrevem a mesma rotação: \mathbf{q} e $-\mathbf{q}$. Estes quatérnios são pontos antípodas na esfera unitária S^3 .

Algoritmo 4.1

$$s = 2.000000$$

$$\mathbf{d}_0 = \begin{bmatrix} 2.000000 \\ 5.000000 \\ 4.000000 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} -0.701057 \\ -0.092296 \\ -0.560986 \\ -0.430459 \end{bmatrix} \quad R = \begin{bmatrix} 0.000000 & -0.500000 & 0.866025 \\ 0.707107 & 0.612372 & 0.353553 \\ -0.707107 & 0.612372 & 0.353553 \end{bmatrix}$$

ALGENCAN: quatérnios

$$s = 1.999999$$

$$\mathbf{d}_0 = \begin{bmatrix} 1.999999 \\ 4.999999 \\ 3.999999 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} 0.701057 \\ 0.092296 \\ 0.560986 \\ 0.430459 \end{bmatrix} \quad R = \begin{bmatrix} -0.000001 & -0.499999 & 0.866026 \\ 0.707106 & 0.612373 & 0.353554 \\ -0.707107 & 0.612373 & 0.353552 \end{bmatrix}$$

ALGENCAN: ângulos de Euler

$$s = 2.000003$$

$$\mathbf{d}_0 = \begin{bmatrix} 1.999998 \\ 4.999993 \\ 3.999999 \end{bmatrix} \quad \Theta = \begin{bmatrix} -1.570792 \\ -0.785396 \\ -1.047193 \end{bmatrix} \quad R = \begin{bmatrix} 0.000000 & -0.500001 & 0.866025 \\ 0.707108 & 0.612372 & 0.353552 \\ -0.707106 & 0.612372 & 0.353557 \end{bmatrix}$$

Tabela 4.1: Problema dos vértices da pirâmide.

Teste 2

Considerando os mesmos dados do teste anterior, simulamos uma situação em que o conjunto V_e é mais preciso que o conjunto V_d , ou seja, adicionamos a cada vetor \mathbf{r}_d^i de V_d um vetor erro $\boldsymbol{\delta}_i$ cujas coordenadas são números reais gerados aleatoriamente no intervalo real $[0, 0.1]$. Dessa forma, para aplicar o Algoritmo 4.1, o fator de escala deve ser calculado pela fórmula (4.16). Os resultados obtidos nesta situação estão indicados na Tabela 4.2. Notemos que, por termos adicionado erros aos vetores do conjunto V_d , os valores encontrados já não são tão precisos quanto os obtidos no Teste 1.

Algoritmo 4.1

$$s = 1.956900$$

$$\mathbf{d}_0 = \begin{bmatrix} 2.05639 \\ 5.08621 \\ 4.04392 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} -0.699871 \\ -0.094465 \\ -0.560617 \\ -0.432395 \end{bmatrix} \quad R = \begin{bmatrix} -0.002513 & -0.499324 & 0.866412 \\ 0.711159 & 0.608222 & 0.352589 \\ -0.703027 & 0.617042 & 0.35357 \end{bmatrix}$$

ALGENCAN: quatérnions

$$s = 1.957031$$

$$\mathbf{d}_0 = \begin{bmatrix} 2.056397 \\ 5.086233 \\ 4.043956 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} 0.699871 \\ 0.094461 \\ 0.560589 \\ 0.432372 \end{bmatrix} \quad R = \begin{bmatrix} -0.002514 & -0.499302 & 0.866366 \\ 0.711119 & 0.608162 & 0.352545 \\ -0.702996 & 0.616989 & 0.353532 \end{bmatrix}$$

ALGENCAN: ângulos de Euler

$$s = 1.956902$$

$$\mathbf{d}_0 = \begin{bmatrix} 2.056390 \\ 5.086213 \\ 4.043922 \end{bmatrix} \quad \boldsymbol{\Theta} = \begin{bmatrix} 1.567260 \\ 3.921236 \\ 2.091131 \end{bmatrix} \quad R = \begin{bmatrix} -0.002514 & -0.499325 & 0.866411 \\ 0.711159 & 0.608221 & 0.35259 \\ -0.703026 & 0.617043 & 0.35357 \end{bmatrix}$$

Tabela 4.2: Problema dos vértices da pirâmide - conjunto V_d contém erros.

Teste 3

Seja V_e um conjunto de 20 pontos em \mathbb{R}^3 cujas coordenadas são geradas aleatoriamente no intervalo real $[0, 1]$. Consideremos uma transformação $T : S_e \rightarrow S_d$ tal que

$$s = 3, \quad \mathbf{d}_0 = \begin{bmatrix} 1 \\ -5 \\ 2 \end{bmatrix} \quad \text{e} \quad R = \begin{bmatrix} 0.612372 & -0.659740 & 0.435596 \\ -0.612372 & -0.047367 & 0.789149 \\ -0.500000 & -0.750000 & -0.433013 \end{bmatrix}. \quad (4.32)$$

O conjunto V_d é obtido como nos testes anteriores, ou seja, aplicando T a cada vetor de V_e . Neste caso estamos supondo que a precisão de ambos os conjuntos é a mesma. Os resultados obtidos para este teste estão indicados na Tabela 4.3.

Algoritmo 4.1

$s = 3.000000$

$$\mathbf{d}_0 = \begin{bmatrix} 1.000000 \\ -5.000000 \\ 2.000000 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} 0.531976 \\ -0.723317 \\ 0.439680 \\ 0.022260 \end{bmatrix} \quad R = \begin{bmatrix} 0.612372 & -0.65974 & 0.435596 \\ -0.612373 & -0.047366 & 0.789149 \\ -0.500000 & -0.750000 & -0.433012 \end{bmatrix}$$

ALGENCAN: quatérnions

$s = 2.999999$

$$\mathbf{d}_0 = \begin{bmatrix} 0.999999 \\ -4.999999 \\ 1.999999 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} -0.531975 \\ 0.723317 \\ -0.439680 \\ -0.022259 \end{bmatrix} \quad R = \begin{bmatrix} 0.612372 & -0.65974 & 0.435596 \\ -0.612373 & -0.047366 & 0.789149 \\ -0.500000 & -0.750000 & -0.433012 \end{bmatrix}$$

ALGENCAN: ângulos de Euler

$s = 3.000000$

$$\mathbf{d}_0 = \begin{bmatrix} 1.000000 \\ -4.999999 \\ 2.000000 \end{bmatrix} \quad \Theta = \begin{bmatrix} 0.785398 \\ -0.523599 \\ 2.094400 \end{bmatrix} \quad R = \begin{bmatrix} 0.612372 & -0.659742 & 0.435592 \\ -0.612372 & -0.047371 & 0.789149 \\ -0.500000 & -0.749998 & -0.433016 \end{bmatrix}$$

Tabela 4.3: Problema dos 20 pontos gerados aleatoriamente.

Teste 4

Consideramos agora o mesmo conjunto de pontos gerados no Teste 3. Porém, adicionamos a cada elemento do conjunto V_d um vetor erro cujas coordenadas são geradas aleatoriamente no intervalo real $[0, 0.1]$. Isso faz com que a precisão dos conjuntos V_e e V_d não seja a mesma e, para aplicar o Algoritmo 4.1 o fator de escala deve ser determinado pela fórmula (4.16). Os resultados obtidos para este teste estão na Tabela 4.4. Observemos que os resultados obtidos já não são tão bons quanto aqueles do Teste 3.

Algoritmo 4.1

$$s = 2.98594$$

$$\mathbf{d}_0 = \begin{bmatrix} 1.035420 \\ -4.937080 \\ 2.028000 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} 0.533106 \\ -0.723434 \\ 0.438306 \\ 0.018134 \end{bmatrix} \quad R = \begin{bmatrix} 0.615118 & -0.653507 & 0.441089 \\ -0.614836 & -0.047371 & 0.787231 \\ -0.493566 & -0.755437 & -0.430939 \end{bmatrix}$$

ALGENCAN: quatérnions

$$s = 2.986141$$

$$\mathbf{d}_0 = \begin{bmatrix} 1.035435 \\ -4.937107 \\ 2.028076 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} -0.533112 \\ 0.723401 \\ -0.438280 \\ -0.018135 \end{bmatrix} \quad R = \begin{bmatrix} 0.615038 & -0.653443 & 0.441067 \\ -0.614769 & -0.047401 & 0.787206 \\ -0.493545 & -0.755412 & -0.430923 \end{bmatrix}$$

ALGENCAN: ângulos de Euler

$$s = 2.985938$$

$$\mathbf{d}_0 = \begin{bmatrix} 1.035419 \\ -4.937081 \\ 2.027998 \end{bmatrix} \quad \Theta = \begin{bmatrix} 0.785168 \\ -0.516184 \\ 2.089209 \end{bmatrix} \quad R = \begin{bmatrix} 0.615118 & -0.653509 & 0.441085 \\ -0.614836 & -0.047376 & 0.787231 \\ -0.493565 & -0.755435 & -0.430943 \end{bmatrix}$$

Tabela 4.4: Problema dos 20 pontos: conjunto V_d contém erros.

Algumas informações sobre o comportamento do pacote ALGENCAN na resolução dos quatro testes anteriores estão indicadas na Tabela 4.5. Nesta tabela *itex* e *itin* são, respectivamente, os números totais de iterações externas e internas realizadas em cada teste; *naf*

é o número de avaliações da função objetivo; $t(s)$ é o tempo de execução em segundos e $f(\mathbf{x}^*)$ é o valor da função objetivo na solução final. Pela tabela podemos notar que o pacote ALGENCAN resolve muito bem todos os testes em um tempo muito pequeno, sendo que em dois teste o tempo de execução é menor que 10^{-6} . Neste pequeno conjunto de testes não conseguimos observar diferenças muito acentuadas entre as duas formulações propostas. Porém constatamos que na formulação via ângulos de Euler, dependendo dos valores iniciais escolhidos para os ângulos, a seqüência de iterandos \mathbf{x}_k gerada por ALGENCAN não converge para um minimizador global do problema. Por essa razão, o método desenvolvido por Berthold Horn é mais conveniente porque o problema de quadrados mínimos pode ser resolvido sem a aplicação de um método numérico. Como vimos, quando quatérnions são utilizados na formulação do problema da orientação absoluta, o quatérnion que representa a rotação ótima é determinado por um cálculo de autovetores.

	Ângulos de Euler				Quatérnions				
	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	$f(\mathbf{x}^*)$	<i>itex</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	$f(\mathbf{x}^*)$
Teste 1	12	21	0.002999	1.82E-09	1	10	24	0.001999	6.91E-14
Teste 2	11	15	0.004000	2.91E-03	4	18	37	0.004000	2.90E-03
Teste 3	14	23	0.004000	7.82E-17	1	12	21	0.000000	1.60E-13
Teste 4	11	18	0.000000	2.74E-02	3	20	31	0.000000	2.74E-02

Tabela 4.5: Comportamento do pacote ALGENCAN na resolução dos testes.

4.2 Problema do empacotamento de moléculas

Pesquisadores da dinâmica molecular procuram compreender o comportamento de uma grande variedade de processos químicos e bioquímicos por meio do estudo das interações entre moléculas no nível atômico. Teorias para o estudo da dinâmica molecular são desenvolvidas e simulações numéricas para a validação dos resultados teóricos obtidos podem ser realizadas graças aos modernos recursos computacionais existentes. No entanto, os *softwares* de simulação necessitam de uma configuração inicial “razoável”, ou seja, uma configuração na qual os átomos de diferentes moléculas não se encontrem demasiadamente próximos. O problema de otimização que será descrito a seguir objetiva exatamente obter este tipo de configuração.

4.2.1 Formulação do problema

Em [22] o problema de encontrar a configuração inicial das moléculas de um sistema é formulado como um problema de empacotamento e resolvido por um algoritmo de otimização. Devemos distribuir uma quantidade $nmol$ de moléculas em uma região finita do espaço de modo que a distância entre pares de átomos de moléculas diferentes não seja menor que uma tolerância $\epsilon > 0$. Para tal, é conveniente tratar cada molécula como um corpo rígido e estabelecer, para cada uma, um sistema de coordenadas com relação ao qual serão expressas as posições dos átomos da molécula. O centro do sistema será posicionado no baricentro da molécula, ou seja, o ponto cujas coordenadas, com relação a qualquer sistema de referência, são as médias aritméticas das coordenadas de seus átomos. Denotando por $\mathbf{A}(i, j)$ as coordenadas do j -ésimo átomo da i -ésima molécula, conhecendo a posição \mathbf{C}_i do baricentro com relação a um sistema referencial fixo S_0 (comum a todas as moléculas) e conhecendo a orientação do sistema S_i (preso à i -ésima molécula) com relação a S_0 , representada pela matriz de rotação R_i , podemos expressar a posição $\mathbf{P}(i, j) = (p_1^{ij}, p_2^{ij}, p_3^{ij})^T$ do j -ésimo átomo da i -ésima molécula com relação ao sistema S_0 da seguinte forma:

$$\mathbf{P}(i, j) = \mathbf{C}_i + R_i \mathbf{A}(i, j). \quad (4.33)$$

Denotando por $natom(i)$ o número de átomos da i -ésima molécula, nosso objetivo é posicionar as moléculas de modo que os átomos de moléculas distintas estejam suficientemente separados, isto é, para $j = 1, \dots, natom(i)$ e $j' = 1, \dots, natom(i')$,

$$\|\mathbf{P}(i, j) - \mathbf{P}(i', j')\|^2 \geq \epsilon, \quad \text{sempre que } i \neq i', \quad i, i' \in \{1, \dots, nmol\}. \quad (4.34)$$

Como as moléculas devem estar confinadas dentro de um recipiente B , devemos pedir também que

$$\mathbf{P}(i, j) \in B \quad \forall \quad i, j. \quad (4.35)$$

As variáveis a serem determinadas em nosso problema de otimização são as coordenadas dos baricentros e os parâmetros que descrevem a rotação de cada molécula. Desse modo, denotando por \mathbf{x} o vetor de variáveis, precisamos encontrar uma configuração \mathbf{x} que satisfaça

(4.34) e (4.35). Seguindo este raciocínio, Martínez et al. [22] constróem para o problema uma função que penaliza as distâncias entre átomos quando estas forem menores que a tolerância ϵ e penaliza os átomos que se encontram fora do recipiente B . Esta função é dada por

$$f(\mathbf{x}) = \sum_{i=1}^{nmol-1} \sum_{i'=i+1}^{nmol} \sum_{j=1}^{natom(i)} \sum_{j'=1}^{natom(i')} (\max\{0, \epsilon - \|\mathbf{P}(i, j) - \mathbf{P}(i', j')\|\}^2)^2 + \sum_{i=1}^{nmol} \sum_{j=1}^{natom(i)} b^{ij}. \quad (4.36)$$

O termo b^{ij} está relacionado com a distância do j -ésimo átomo da i -ésima molécula ao recipiente B em uma norma que depende do formato de B . Por exemplo, se B é uma caixa, isto é,

$$B = \{\mathbf{x} \in \mathbb{R}^3 \mid \boldsymbol{\ell} \leq \mathbf{x} \leq \mathbf{u}\}, \quad \boldsymbol{\ell}, \mathbf{u} \in \mathbb{R}^3,$$

teremos

$$b^{ij} = \sum_{k=1}^3 (\max\{0, p_k^{ij} - u_k, l_k - p_k^{ij}\})^2. \quad (4.37)$$

Este caso de empacotamento está ilustrado na Figura 4.3. Outros formatos de regiões B

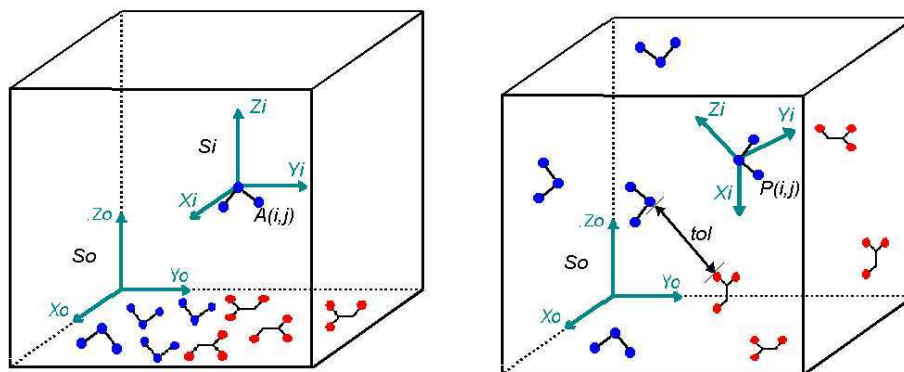


Figura 4.3: Empacotamento de moléculas em uma caixa.

também podem ser considerados. Suponhamos que B seja uma esfera de centro \mathbf{o} e raio r :

$$B = \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x} - \mathbf{o}\|^2 \leq r^2\}, \quad \mathbf{o} = (o_1, o_2, o_3)^T.$$

Neste caso, definimos b^{ij} como

$$b^{ij} = (\max\{0, (p_1^{ij} - o_1)^2 + (p_2^{ij} - o_2)^2 + (p_3^{ij} - o_3)^2 - r^2\})^2. \quad (4.38)$$

Notemos que qualquer que seja o formato do recipiente B onde as moléculas serão alocadas, b^{ij} vale 0 sempre que o vetor $\mathbf{P}(i, j)$ pertencer a B .

A função f definida em (4.36) é uma vez continuamente diferenciável e não negativa. Como o menor valor atingido por f é zero, poderíamos em princípio construir o seguinte problema de minimização:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{sujeita a} \quad & \mathbf{C}_i \in B \quad \forall \quad i = 1, \dots, nmol \end{aligned} \tag{4.39}$$

Notemos que em (4.39) o baricentro de cada molécula é forçado a estar dentro do recipiente B . Esta imposição não impede que, durante a resolução deste problema por um método numérico de otimização, átomos de determinadas moléculas violem as dimensões de B . Porém, como a função f foi definida para penalizar os átomos que excedem as dimensões da região onde são colocados, podemos substituir (4.39) pelo problema irrestrito:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \mathbf{x} \quad & \text{irrestrito} \end{aligned} \tag{4.40}$$

Se existe um vetor \mathbf{x}^* tal que as moléculas encontram-se dentro do recipiente e os átomos de moléculas distintas estão suficientemente separados, então $f(\mathbf{x}^*) = 0$. Como $f \geq 0$, segue que tal vetor seria um minimizador global de (4.40). Se ao resolver (4.40) encontramos \mathbf{x}^* tal que $f(\mathbf{x}^*) = 0$, sabemos que \mathbf{x}^* é um minimizador global e corresponde a um empacotamento satisfatório. No entanto, se a melhor solução encontrada é tal que $f(\mathbf{x}^*) > 0$, não podemos concluir nada. Nesta última hipótese pode ocorrer que o problema tenha um minimizador global com valor de f nulo (que não conseguimos encontrar) ou então que efetivamente não seja possível empacotar satisfatoriamente o conjunto de moléculas considerado no recipiente escolhido.

4.2.2 Representação das rotações envolvidas

A função objetivo do problema de otimização que acabamos de definir depende apenas das coordenadas cartesianas $\mathbf{P}(i, j)$ dos átomos das moléculas. Isto quer dizer que temos liberdade para escolher a maneira de representar a rotação de cada molécula do sistema. Apresentamos a seguir duas maneiras diferentes de representar as rotações das moléculas: via ângulos de Euler e quatérnions. Utilizando cada uma dessas representações, resolvemos o problema de otimização (4.40) e fazemos uma comparação dos resultados numéricos obtidos em cada formulação.

Formulação via ângulos de Euler

Os autores do trabalho original [22] utilizam ângulos de Euler para construir a matriz de rotação R_i em (4.33). Nesta formulação o movimento de rotação de cada molécula é descrito pela composição de rotações consecutivas em torno dos eixos coordenados. Primeiramente a i -ésima molécula é rodada em torno do eixo Z por um ângulo ψ_i , em seguida aplica-se à ela uma rotação de ângulo θ_i em torno do eixo X e por fim rotaciona-se a molécula novamente

em torno do eixo Z por um ângulo ϕ_i . A composição dessas três rotações consecutivas formará a matriz R_i :

$$R_i = \begin{pmatrix} c_{\phi_i}c_{\psi_i} - s_{\phi_i}c_{\theta_i}s_{\psi_i} & -c_{\phi_i}s_{\psi_i} - s_{\phi_i}c_{\theta_i}c_{\psi_i} & s_{\phi_i}s_{\theta_i} \\ s_{\phi_i}c_{\psi_i} + c_{\phi_i}c_{\theta_i}s_{\psi_i} & -s_{\phi_i}s_{\psi_i} + c_{\phi_i}c_{\theta_i}c_{\psi_i} & -c_{\phi_i}s_{\theta_i} \\ s_{\theta_i}s_{\psi_i} & s_{\theta_i}c_{\psi_i} & c_{\theta_i} \end{pmatrix}, \quad (4.41)$$

Construindo R_i desta maneira, o problema (4.40) passa a ter $6 \times nmol$ variáveis (3 coordenadas de baricentro e 3 ângulos por molécula). Notemos que neste caso não é preciso adicionar nenhuma restrição a (4.40) pois $R_i \in SO(3)$. O problema de empacotamento formulado com ângulos de Euler requer muitas avaliações das funções trigonométricas seno e cosseno. Pensando em um modo de evitar o cálculo dessas funções, introduziremos a seguir outra maneira de formular (4.40), onde a matriz de rotação R_i é construída com as componentes de um quatérnion.

Formulação via quatérnions

Na tentativa de evitar cálculos excessivos de funções trigonométricas, substituímos a tripla de ângulos $(\psi_i, \theta_i, \phi_i)$ por um quatérnion normalizado:

$$\mathbf{q}_{ni} = \frac{1}{\|\mathbf{q}_i\|} (q_{i0}, q_{i1}, q_{i2}, q_{i3})^T, \quad \text{onde} \quad \|\mathbf{q}_i\| = \sqrt{q_{i0}^2 + q_{i1}^2 + q_{i2}^2 + q_{i3}^2}.$$

O quatérnion \mathbf{q}_{ni} é utilizado para construir um operador de rotação L que produza o mesmo efeito da seqüência de ângulos de Euler aplicada na formulação anterior. Para tanto, consideramos o vetor $\mathbf{A}(i, j)$ com as coordenadas iniciais do j -ésimo átomo da i -ésima molécula e construímos a partir dele o quatérnion puro $\mathbf{v}_a = (0, \mathbf{A}(i, j))^T$. Então, aplicando o operador L em \mathbf{v}_a obtemos outro quatérnion puro que é dado por

$$\mathbf{w}_a = L(\mathbf{v}_a) = \frac{1}{\|\mathbf{q}_i\|^2} (\mathbf{q}_{ni} \mathbf{v}_a \mathbf{q}_{ni}^*) = (0, \mathbf{A}_R(i, j))^T. \quad (4.42)$$

Lembrando que a parte vetorial de (4.42) fornece as coordenadas do vetor $\mathbf{A}(i, j)$ rodado, utilizando notação matricial, podemos reescrevê-la como $\mathbf{A}_R(i, j) = R_i \mathbf{A}(i, j)$, onde

$$R_i = M_i - I, \quad (4.43)$$

$$M_i = \frac{2}{\|\mathbf{q}_i\|^2} \begin{pmatrix} q_{i0}^2 + q_{i1}^2 & q_{i1}q_{i2} - q_{i0}q_{i3} & q_{i1}q_{i3} + q_{i0}q_{i2} \\ q_{i1}q_{i2} + q_{i0}q_{i3} & q_{i0}^2 + q_{i2}^2 & q_{i2}q_{i3} - q_{i0}q_{i1} \\ q_{i1}q_{i3} - q_{i0}q_{i2} & q_{i2}q_{i3} + q_{i0}q_{i1} & q_{i0}^2 + q_{i3}^2 \end{pmatrix},$$

e I é a matriz identidade de ordem 3. Notemos que a matriz R_i em (4.43) é semelhante à matriz (3.35) definida na Seção 3.4 do Capítulo 3. Esta maneira de construir R_i faz com que o problema de otimização (4.40) também seja irrestrito como no caso da formulação

via ângulos de Euler. Além disso, a função objetivo de (4.40) é mais fácil de ser avaliada pois os cálculos com funções trigonométricas são agora substituídos por cálculos com funções polinomiais, como podemos observar pela estrutura da nova matriz de rotação (4.43).

Se, por outro lado, construirmos R a partir de um quatérnio não normalizado, então temos duas outras possibilidades para reformular o problema de empacotamento:

(a) incorporar em (4.40) as restrições

$$h_i = \|\mathbf{q}_i\|^2 - 1 = 0, \quad i = 1, \dots, nmol,$$

(b) penalizar a função objetivo de (4.40) adicionando a ela o termo

$$\rho \left(\sum_{i=1}^{nmol} (\|\mathbf{q}_i\|^2 - 1)^2 \right),$$

onde $\rho > 0$ é um parâmetro de penalidade previamente estabelecido.

Se a alternativa (a) for escolhida, devemos resolver o seguinte problema:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{sujeita a} \quad & \|\mathbf{q}_i\|^2 - 1 = 0 \quad \forall \quad i = 1, \dots, nmol. \end{aligned} \tag{4.44}$$

Observemos que (4.44) tem tantas restrições não lineares quanto o número de moléculas consideradas. Não podemos portanto aplicar um método de otimização irrestrita para resolvê-lo. Empregaremos um método de Lagrangiano aumentado, que necessitará do cálculo do vetor gradiente da função objetivo e dos vetores gradientes das restrições h_i . Estes últimos são muito simples de serem obtidos, pois as derivadas de h_i com relação às coordenadas do i -ésimo baricentro e i -ésimo quatérnio são dadas, respectivamente, por

$$\begin{aligned} \frac{\partial h_i(\mathbf{x})}{\partial c_{ik}} &= 0, \quad i = 1, \dots, nmol, \quad k = 1, 2, 3, \\ \frac{\partial h_i(\mathbf{x})}{\partial q_{ik}} &= 2q_{ik}, \quad i = 1, \dots, nmol, \quad k = 1, \dots, 4. \end{aligned} \tag{4.45}$$

Por outro lado, se a alternativa (b) for utilizada, deveremos resolver o seguinte problema de otimização irrestrita:

$$\begin{aligned} \min \quad & f(\mathbf{x}) + \rho \left(\sum_{i=1}^{nmol} (\|\mathbf{q}_i\|^2 - 1)^2 \right) \\ \mathbf{x} \quad & \text{irrestrito} \end{aligned} \tag{4.46}$$

Uma vez calculado o gradiente de $f(\mathbf{x})$, o gradiente da função objetivo do problema (4.46) é dado por

$$\nabla f(\mathbf{x}) + 2\rho \left(\sum_{i=1}^{nmol} \nabla h_i(\mathbf{x}) \right),$$

onde os vetores $\nabla h_i(\mathbf{x})$ são obtidos pelos cálculos (4.45).

Em qualquer uma das alternativas apresentadas anteriormente, o problema de otimização a ser resolvido passa a ter $7 \times nmol$ variáveis (3 coordenadas de baricentro e quatro componentes de quatérnion por molécula).

Como dissemos anteriormente, o método numérico de otimização que utilizamos para resolver os problemas de empacotamento necessita do cálculo do vetor gradiente da função $f(\mathbf{x})$. Como esta função não tem uma forma tão simples, poderíamos calcular $\nabla f(\mathbf{x})$ pelo esquema de diferenças finitas centradas. Porém, este procedimento torna-se inviável a medida que a dimensão do problema aumenta, por necessitar de duas avaliações de f no cálculo de cada componente do vetor $\nabla f(\mathbf{x})$. Optamos, portanto, por construir uma rotina para o cálculo do gradiente analítico de $f(\mathbf{x})$.

A Tabela 4.6 resume as principais características do problema do empacotamento de $nmol$ moléculas formulado com ângulos de Euler (1), quatérnions normalizados (2) e quatérnions não normalizados (3 (a) e 3 (b) de acordo com as alternativas (a) e (b), respectivamente).

Prob. Empacotamento	Formulação			
	1	2	3 (a)	3 (b)
número de variáveis	$6 \times nmol$	$7 \times nmol$	$7 \times nmol$	$7 \times nmol$
restrições de norma unitária			$nmol$	
funções trigonométricas	sim			
funções polinomiais		sim	sim	sim

Tabela 4.6: Principais características de cada formulação.

4.2.3 O programa *packmol*

O programa *packmol* é uma interface elaborada em Fortran 77 que resolve o problema do empacotamento de moléculas utilizando ângulos de Euler na descrição das rotações. Para realizar qualquer simulação em *packmol* o usuário deve fornecer ao programa um arquivo de entrada contendo descrições sobre a região do espaço que será destinada ao experimento, os tipos de moléculas que serão alocados nesta região, a quantidade de moléculas por tipo e a distância mínima entre pares de átomos de moléculas diferentes (tolerância ϵ). Para cada tipo de molécula especificado no arquivo de entrada, um arquivo contendo as coordenadas dos átomos da molécula isolada também deve ser fornecido.

A Figura 4.4 mostra um exemplo de dois arquivos que são necessários para um realizar um teste de empacotamento de moléculas de água. O arquivo *interface.inp* especifica que 1019 moléculas de água serão colocadas em uma caixa de coordenadas mínimas $(0, 0, 0)^T$ e máximas $(40, 40, 40)^T$ (medidas em Angstroms¹) de modo que a distância mínima entre pares de átomos de moléculas distintas deve ser de pelo menos 2 Angstroms. As coordenadas dos átomos que compõem uma molécula de água isolada estão no arquivo auxiliar *water.xyz*.

¹ 1 Angstrom é igual a 10^{-10} m.

Arquivo interface.inp			
tolerance	2.0		
filetype	xyz		
output	interface.xyz		
structure	water.xyz		
number	1019		
inside box	0. 0. 0.	40. 40. 40.	
end structure			
Arquivo water.xyz			
3			
water			
H	9.625597	6.787278	12.673000
H	9.625597	8.420323	12.673000
O	10.203012	7.603800	12.673000

Figura 4.4: Exemplos de arquivos de leitura.

A partir da leitura dos arquivos de entrada definidos pelo usuário, *packmol* cria aleatoriamente uma configuração inicial para o sistema de moléculas de tal forma que todos os átomos estejam dentro do recipiente especificado. Esta configuração é armazenada no vetor \mathbf{x}_0 do seguinte modo:

$$\mathbf{x}_0 = (\mathbf{C}_1, \dots, \mathbf{C}_{nmol}, \psi_1, \theta_1, \phi_1, \dots, \psi_{nmol}, \theta_{nmol}, \phi_{nmol})^T, \quad (4.47)$$

onde \mathbf{C}_i é o baricentro e ψ_i, θ_i, ϕ_i são os ângulos de Euler que descrevem a rotação da i -ésima molécula. O vetor (4.47) servirá de ponto inicial para uma rotina de otimização que, por sua vez, tem o papel de tentar encontrar uma solução \mathbf{x}^* para o problema (4.40) por meio da seguinte estratégia: partindo de \mathbf{x}_0 e utilizando uma tolerância $\epsilon_0 > \epsilon$, a rotina realiza dez iterações internas, obtendo um vetor intermediário $\hat{\mathbf{x}}$ ao final da décima iteração. Se $\hat{\mathbf{x}}$ for tal que $f(\hat{\mathbf{x}}) < 10^{-5}$, $\hat{\mathbf{x}}$ será considerado a solução final do problema de empacotamento. Caso contrário, a tolerância ϵ_0 é reduzida e o vetor $\hat{\mathbf{x}}$ é passado como ponto inicial para a próxima chamada da rotina que tentará minimizar a função objetivo com uma tolerância ϵ_1 tal que $\epsilon < \epsilon_1 < \epsilon_0$. Este processo termina quando uma configuração ótima \mathbf{x}^* for encontrada ou quando o número máximo de chamadas da rotina de otimização for alcançado.

Observemos que em cada tentativa de encontrar uma solução para o problema de empacotamento, a rotina de otimização tenta minimizar a função objetivo considerando uma tolerância maior que a especificada no arquivo de entrada. Este procedimento serve para afastar as moléculas que estão muito próximas evitando a formação de aglomerados. O vetor final \mathbf{x}^* obtido é usado pelo programa *packmol* para criar um arquivo de saída em formato de texto contendo as coordenadas ótimas dos átomos das moléculas do sistema. Com este arquivo o usuário pode, com o auxílio de um pacote gráfico, visualizar o resultado de sua simulação (veja por exemplo [15]). Os passos descritos acima podem ser mais bem compreendidos no fluxograma da Figura 4.5.

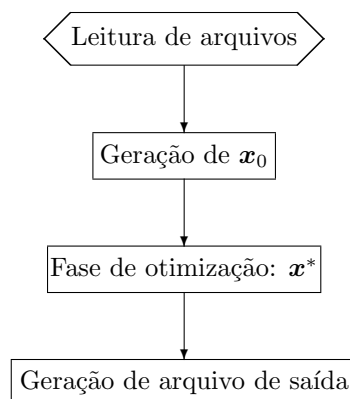


Figura 4.5: Principais fases do programa *packmol*.

Atualmente existem duas versões do programa *packmol*. Ambas têm a estrutura geral do fluxograma da Figura 4.5. Como veremos adiante, as principais diferenças entre estas versões estão nos tipos de arquivos de entrada que podem ser definidos, na maneira como a configuração inicial x_0 é gerada e na fase de otimização para a obtenção da solução final.

Apesar das diferenças, ambas as versões utilizam a mesma estratégia para avaliar a função objetivo (4.36) e o seu vetor gradiente. No cálculo de f ou de ∇f , fixado o j -ésimo átomo da i -ésima molécula, é preciso saber quais átomos das outras moléculas do sistema têm uma distância menor que ϵ do átomo fixado. Num primeiro momento, o programa poderia calcular todas as distâncias

$$\|\mathbf{P}(i, j) - \mathbf{P}(i', j')\| \text{ com } i, j \text{ fixos,}$$

e verificar quais delas são menores que ϵ . Mas esse procedimento requer um tempo considerável, sobretudo quando a quantidade de moléculas é muito grande. A fim de evitar cálculos e comparações excessivas nas avaliações de f e ∇f , *packmol* procede do seguinte modo: dada uma configuração \mathbf{x} , o programa determina uma caixa que contenha todos os átomos do sistema e a divide em cubos de aresta igual a tolerância ϵ . Então, os cubos e os átomos existentes dentro de cada cubo são identificados por índices. Dessa forma, fixado um átomo de uma molécula dentro de um cubo qualquer, para saber quais átomos distam dele um valor menor que ϵ , *packmol* verifica apenas os átomos que estão no cubo fixo e nos 9 cubos adjacentes ao fixo, ou seja, aqueles que compartilham com ele faces, arestas e vértices. Notemos que com essa estratégia não é preciso comparar o átomo fixo com todos os átomos do sistema para saber quais distâncias são menores que a tolerância especificada.

A seguir, apresentamos as duas versões existentes do programa *packmol* e discutimos suas principais diferenças.

Packmol BOX

A versão antiga do programa, que denotaremos por PB, admite apenas recipientes em formato de caixa para alocar as moléculas e resolve o problema de otimização (4.40) por meio da rotina BOX [9]. A partir da leitura dos arquivos de entrada, PB determina aleatoriamente uma configuração inicial \mathbf{x}_0 dentro do recipiente (caixa). Este mesmo recipiente é dividido em cubos que são utilizados nas avaliações de f e ∇f como descrevemos anteriormente. Por fim, a rotina BOX é chamada para tentar encontrar uma configuração ótima \mathbf{x}^* para o sistema de moléculas.

Packmol GENCAN

Nesta versão do programa, que denotaremos por PG, o problema de otimização (4.40) é resolvido pela rotina GENCAN [4]. Em PG o usuário pode definir outros formatos de recipientes para realizar os experimentos, tais como esferas, cilindros e elipsóides. Porém, diferentemente da versão antiga, o programa PG determina uma configuração inicial \mathbf{x}_0 em duas etapas:

- (1) Na primeira etapa, após a leitura dos arquivos de entrada, PG cria uma caixa B_1 de dimensões suficientes para conter o recipiente onde serão alocadas as moléculas. Em seguida, uma configuração inicial \mathbf{z}_0 é gerada aleatoriamente dentro de B_1 . Esta configuração é passada como ponto inicial para a rotina GENCAN, cuja tarefa é minimizar a função

$$\sum_{i=1}^{nmol} \sum_{j=1}^{natom(i)} b^{ij}, \quad (4.48)$$

onde o termo b^{ij} , conforme descrevemos na Seção 4.2.1, mede a distância do j -ésimo átomo da i -ésima molécula ao recipiente do experimento. Como $b^{ij} \geq 0 \forall i, j$, a função (4.48) é não negativa. Logo, o menor valor que ela pode assumir é zero. Portanto, um minimizador global desta função será uma configuração em que todos os átomos do sistema estão confinados dentro do recipiente, porém, a distância entre pares de átomos de moléculas distintas pode ser menor que a tolerância ϵ . Ao resolver este subproblema, a rotina GENCAN encontra uma dessas configurações \mathbf{z}^* que será utilizada por PG para calcular as máximas e mínimas coordenadas dos átomos do sistema. Com estas coordenadas, PG cria uma nova caixa B_2 , menor que a caixa inicial B_1 , mas que ainda contém todo o sistema de moléculas. Por fim, a caixa B_2 é dividida em cubos que serão utilizados nas avaliações de f e ∇f de acordo com a estratégia descrita anteriormente.

- (2) Nesta etapa, o programa usa as informações contidas no vetor \mathbf{z}^* , determinado na etapa anterior, para empacotar um tipo de molécula por vez dentro do recipiente. Isto é, sendo n_t o número de tipos de moléculas presentes no sistema, para cada tipo m de molécula, $m = 1, \dots, n_t$, o programa extrai do vetor \mathbf{z}^* as coordenadas dos baricentros e os ângulos de rotação das moléculas do tipo m e os armazena no vetor $\hat{\mathbf{x}}_m$. Tomando $\hat{\mathbf{x}}_m$ como ponto inicial, a rotina GENCAN é chamada para resolver o problema (4.40),

agora com um número bem menor de variáveis. Ou seja, GENCAN empacota cada tipo de molécula separadamente. As soluções $\hat{\mathbf{x}}_m^*$ obtidas para cada tipo de molécula serão armazenadas no vetor \mathbf{x}_0 , que será o ponto inicial definitivo.

Ao sair da etapa (2), a rotina GENCAN resolverá o problema de empacotar todas as moléculas do sistema adotando \mathbf{x}_0 como ponto inicial. Os autores do trabalho original [22] observaram que o problema de empacotar todas as moléculas de uma vez torna-se mais fácil de ser resolvido se antes cada tipo de molécula for empacotado separadamente. Isso acontece porque o ponto \mathbf{x}_0 , encontrado ao final da etapa (2) está próximo de uma configuração ótima, facilitando a busca por uma solução \mathbf{x}^* .

Novas versões do programa *packmol*

Utilizando quatérnions para descrever as rotações de cada molécula, construímos as seguintes versões para o programa *packmol*:

- (1) Packmol BOX-Quatérnions (PBQ):
As matrizes de rotação são construídas com quatérnions normalizados de acordo com a expressão (4.43).
- (2) Packmol GENCAN-Quatérnions (PGQ):
As matrizes de rotação são construídas da mesma forma que no programa PBQ.
- (3) Packmol GENCAN-Quatérnions-Penalização (PGQP):
Quatérnions não normalizados são utilizados para construir as matrizes de rotação. A condição de norma unitária de cada quatérnion é acrescentada à função objetivo como um termo de penalização.
- (4) Packmol ALGENCAN-Quatérnions (PALQ):
Nesta versão também utilizamos quatérnions não normalizados para construir as matrizes de rotação. Porém, para cada quatérnion, a condição de norma unitária é colocada como uma restrição não linear de igualdade do problema de otimização.

Embora tenhamos obtido quatro novas versões para o programa *packmol*, todas têm a mesma estrutura geral: a partir da leitura dos arquivos de entrada os programas criam uma configuração inicial \mathbf{x}_0 (baricentros + ângulos) para o sistema de moléculas. Este vetor é passado para uma subrotina que converte cada tripla de ângulos $(\psi_i, \theta_i, \phi_i)$ em um quatérnion

unitário $\mathbf{q}_i = (q_{i0}, q_{i1}, q_{i2}, q_{i3})^T$ através das relações

$$\begin{aligned}
 q_{i0} &= \cos\left(\frac{\phi_i}{2}\right) \cos\left(\frac{\psi_i}{2}\right) \cos\left(\frac{\theta_i}{2}\right) - \cos\left(\frac{\theta_i}{2}\right) \sin\left(\frac{\phi_i}{2}\right) \sin\left(\frac{\psi_i}{2}\right), \\
 q_{i1} &= \cos\left(\frac{\phi_i}{2}\right) \cos\left(\frac{\psi_i}{2}\right) \sin\left(\frac{\theta_i}{2}\right) + \sin\left(\frac{\phi_i}{2}\right) \sin\left(\frac{\psi_i}{2}\right) \sin\left(\frac{\theta_i}{2}\right), \\
 q_{i2} &= \cos\left(\frac{\phi_i}{2}\right) \sin\left(\frac{\psi_i}{2}\right) \sin\left(\frac{\theta_i}{2}\right) - \cos\left(\frac{\psi_i}{2}\right) \sin\left(\frac{\phi_i}{2}\right) \sin\left(\frac{\theta_i}{2}\right), \\
 q_{i3} &= \cos\left(\frac{\psi_i}{2}\right) \cos\left(\frac{\theta_i}{2}\right) \sin\left(\frac{\phi_i}{2}\right) + \cos\left(\frac{\phi_i}{2}\right) \cos\left(\frac{\theta_i}{2}\right) \sin\left(\frac{\psi_i}{2}\right),
 \end{aligned} \tag{4.49}$$

onde $i = 1, \dots, nmol$. Os resultados obtidos na conversão são armazenados no vetor $\bar{\mathbf{x}}_0$ que é dado por

$$\bar{\mathbf{x}}_0 = (\mathbf{C}_1, \dots, \mathbf{C}_{nmol}, q_{11}, q_{12}, q_{13}, q_{14}, \dots, q_{nmol1}, q_{nmol2}, q_{nmol3}, q_{nmol4})^T. \tag{4.50}$$

Este vetor é posteriormente passado como ponto inicial para a rotina de otimização que tenta encontrar uma configuração ótima \mathbf{x}^* pela mesma estratégia utilizada no programa original. Por fim, as informações contidas no vetor \mathbf{x}^* são usadas pelo programa principal para a elaboração de um arquivo de saída com as coordenadas ótimas dos átomos das moléculas do sistema. O fluxograma da Figura 4.6 resume as principais passagens comuns as quatro versões construídas.

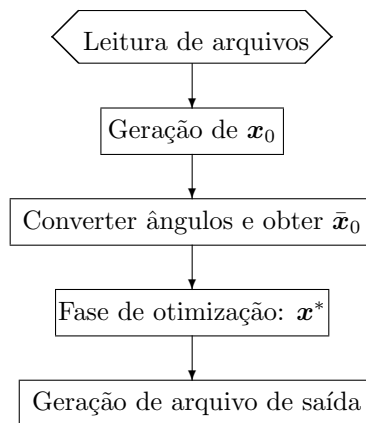


Figura 4.6: Principais passos das versões PBQ, PGQ, PGQP e PALQ.

4.2.4 Experimentos

Nos testes numéricos que vamos apresentar a seguir, a distância mínima entre pares de átomos de moléculas distintas foi tomada igual a 2 Angstroms. Os programas originais PB, PG e as versões construídas PBQ, PGQ, PGQP, e PALQ foram comparados com relação as seguintes características:

- Quantidade total de chamadas da rotina de otimização (nch);
- Número total de iterações internas ($itin$);
- Número total de avaliações da função objetivo (naf);
- Tempo de execução em segundos ($t(s)$).

Como na versão PGQP a função objetivo é penalizada com as restrições de norma unitária de cada quatérnion, o parâmetro de penalidade adotado nos testes foi $\rho = 10^3$. Na versão PALQ, uma vez que os problemas são resolvidos por um método de Lagrangiano aumentado, o número total de iterações externas ($itex$) também foi contabilizado. Devemos lembrar também que as configurações finais \mathbf{x}^* obtidas por cada versão do programa *packmol* devem ser tais que $f(\mathbf{x}^*) < 10^{-5}$, pois assim, os átomos de moléculas distintas ficam toleravelmente separados e dentro do recipiente definido em cada experimento.

Empacotamento de moléculas de água em caixas

Consideramos inicialmente a tarefa de empacotar apenas moléculas de água (H_2O) em recipientes com formato de caixa. Realizamos 10 testes adotando as seguintes quantidades de moléculas: 2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000, 50000 e 100000. O volume da caixa destinada a cada experimento foi calculado proporcionalmente a quantidade de moléculas a serem empacotadas.

Os resultados obtidos pelos programas originais PB, PG estão indicados na Tabela 4.7. A Tabela 4.8 traz os resultados referentes às versões PBQ e PGQ e a Tabela 4.9 mostra os resultados obtidos pelas versões PGQP e PALQ. Nestas três tabelas, as colunas $nmol$ e $nvar$ indicam, respectivamente, a quantidade de moléculas de água empacotadas e o número de variáveis de cada problema. Lembremos que o número de variáveis na formulação via ângulos de Euler é $6 \times nmol$ e na formulação via quatérnions é $7 \times nmol$. O valor da função objetivo na solução final de cada teste está indicado na Tabela 4.10.

Para podermos comparar melhor os tempos neste conjunto de testes construímos os gráficos da Figura 4.7. Nestes gráficos cada segmento vertical mostra os valores mínimo (ponto inferior), máximo (ponto superior) e a média (ponto intermediário) de tempo atingidos por cada programa na resolução dos dez problemas. Notemos que a versão PGQ (Figura 4.7 (b)) é aquela que atinge o menor pico.

<i>nmol</i>	<i>nvar</i>	PB				PG			
		<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>
2500	15000	2	20	29	10.665378	3	21	37	4.276349
5000	30000	3	30	42	42.664513	2	12	23	5.066229
7500	45000	3	30	42	80.699730	2	20	41	9.698524
10000	60000	3	30	42	122.840332	3	21	42	13.552939
12500	75000	3	30	42	130.420181	3	21	45	15.514640
15000	90000	3	30	43	246.700500	2	12	21	11.450259
17500	105000	3	30	42	320.150330	3	21	52	25.853067
20000	120000	4	40	57	252.689575	3	21	45	27.713785
50000	300000	3	30	42	751.538757	3	21	41	69.767387
100000	600000	3	30	43	2229.252200	3	22	47	183.723068

Tabela 4.7: Desempenho de PB e PG no empacotamento de moléculas de água.

<i>nmol</i>	<i>nvar</i>	PBQ				PGQ			
		<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>
2500	17500	3	30	43	11.386269	3	21	44	3.718434
5000	35000	4	39	89	26.377989	2	12	23	4.636294
7500	52500	3	30	45	54.689686	2	22	51	10.723368
10000	70000	3	30	51	56.769367	3	21	57	11.848197
12500	87500	3	30	45	81.032684	2	15	32	9.387572
15000	105000	3	30	42	96.162384	2	14	28	11.106311
17500	122500	3	30	44	117.350159	3	21	45	16.724457
20000	140000	3	30	47	191.933823	2	13	27	13.436956
50000	350000	3	30	44	702.324219	2	13	25	36.803406
100000	700000	5	50	72	3589.327150	3	22	42	148.708389

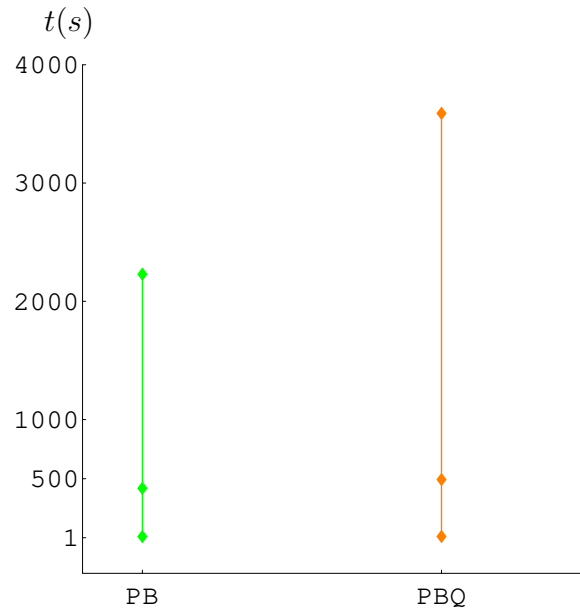
Tabela 4.8: Desempenho de PBQ e PGQ no empacotamento de moléculas de água.

<i>nmol</i>	<i>nvar</i>	PGQP				PALQ				
		<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	<i>nch</i>	<i>itex</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>
2500	17500	2	20	55	5.418175	1	7	64	96	83.580292
5000	35000	2	14	37	7.563849	1	7	67	112	76.124420
7500	52500	2	14	35	9.844502	1	6	60	98	70.564270
10000	70000	2	13	34	12.202144	1	8	77	153	39.056065
12500	87500	3	22	69	25.897062	1	9	86	196	43.189434
15000	105000	3	29	98	64.317223	1	7	67	102	51.446178
17500	122500	3	23	59	34.205799	1	7	67	133	147.255615
20000	140000	2	16	44	30.463368	1	8	74	158	115.666412
50000	350000	3	25	78	286.468445	1	7	69	209	671.303955
100000	700000	3	23	66	450.547516	1	7	64	178	336.712799

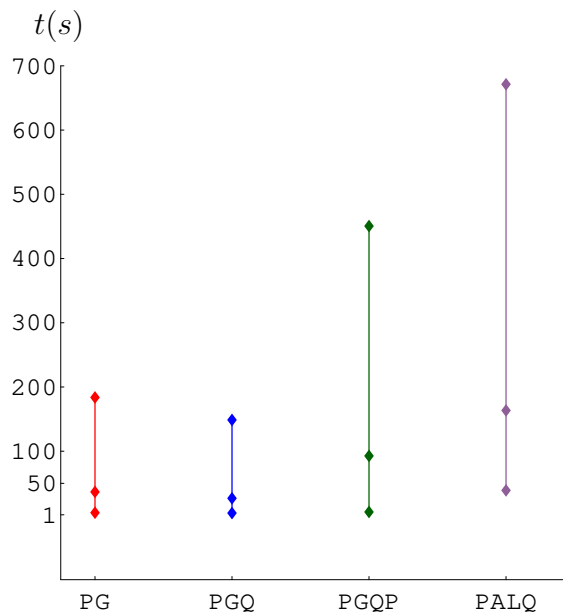
Tabela 4.9: Desempenho de PGQP e PALQ no empacotamento de moléculas de água.

<i>nmol</i>	PB	PG	PBQ	PGQ	PGQP	PALQ
2500	9.68E-06	1.49E-12	1.41E-07	3.79E-07	1.06E-06	0.00E+00
5000	7.23E-08	7.80E-11	0.00E+00	1.26E-06	2.93E-10	0.00E+00
7500	1.31E-07	2.05E-06	8.81E-06	6.44E-11	2.57E-11	0.00E+00
10000	4.22E-09	8.00E-13	4.51E-07	8.89E-07	6.71E-08	0.00E+00
12500	7.21E-10	3.91E-08	3.92E-07	0.00E+00	3.15E-06	0.00E+00
15000	1.20E-07	2.58E-10	2.81E-08	4.52E-09	3.69E-12	0.00E+00
17500	4.15E-09	9.07E-10	1.79E-09	4.98E-06	8.10E-09	0.00E+00
20000	2.47E-13	4.94E-08	4.43E-07	3.11E-09	5.23E-08	0.00E+00
50000	6.04E-08	7.64E-14	1.57E-06	7.40E-08	7.61E-11	0.00E+00
100000	6.62E-08	6.77E-13	6.77E-13	8.22E-07	2.08E-08	0.00E+00

Tabela 4.10: Valor da função objetivo na solução final.



(a)



(b)

Figura 4.7: Comparando tempos no empacotamento de moléculas de água.

Empacotamento de moléculas de uréia em caixas

Consideraremos agora a tarefa de empacotar moléculas de uréia em recipientes com formato de caixa. A uréia é um composto orgânico cristalino e incolor com fórmula dada por $CO(NH_2)_2$. Realizamos ao todo 6 testes, variando a quantidade de moléculas de uréia e calculando o volume das caixas proporcionalmente à cada quantidade. Os resultados obtidos pelos programas estão nas Tabelas 4.11, 4.12, 4.13. O valor da função objetivo na solução final de cada teste está indicado na Tabela 4.14. Como a molécula de uréia é mais alongada que a molécula de água, o empacotamento de grandes quantidades de uréia torna-se mais difícil. Por essa razão, os programas realizam mais iterações e avaliações de função na resolução dos problemas com moléculas de uréia. Conseqüentemente, o tempo de execução de cada problema também aumenta (observe os tempos de execução dos programas no empacotamento de 100000 moléculas de uréia). Os gráficos das Figuras 4.8 (a), 4.8 (b) e 4.8 (c) mostram os valores de tempo mínimo, máximo e a média alcançada por cada programa. Notemos que os menores picos (repare nas escalas verticais) são atingidos pelas versões PG e PGQ.

<i>nmol</i>	<i>nvar</i>	PB				PG			
		<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>
400	2400	5	50	120	7.224901	3	22	46	1.664746
1000	6000	5	45	120	18.062252	3	22	45	3.892407
5000	30000	4	40	61	120.661659	3	23	50	16.977417
10000	60000	5	50	72	417.290558	4	35	69	44.909172
50000	300000	5	50	73	5707.982420	4	33	78	538.587158
100000	600000	6	60	88	26209.459000	8	72	155	3110.556150

Tabela 4.11: Desempenho de PB e PG no empacotamento de moléculas de uréia.

<i>nmol</i>	<i>nvar</i>	PBQ				PGQ			
		<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>
400	2800	6	57	120	7.026930	5	48	97	2.178668
1000	8000	11	108	212	27.797773	3	26	46	3.617449
5000	35000	6	60	81	125.452927	5	45	82	22.056644
10000	70000	13	130	177	570.292297	5	45	94	46.331955
50000	350000	7	70	111	6683.387210	6	52	93	574.947632
100000	700000	7	70	99	24078.197300	9	84	166	3264.764650

Tabela 4.12: Desempenho de PBQ e PGQ no empacotamento de moléculas de uréia.

<i>nmol</i>	<i>nvar</i>	PGQP				PALQ				
		<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	<i>nch</i>	<i>itex</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>
400	2800	11	99	382	9.403569	2	18	174	397	12.303127
1000	7000	6	55	143	8.900645	2	16	154	349	50.199368
5000	35000	11	102	321	107.974579	2	16	156	370	117.993057
10000	70000	11	104	318	267.125397	2	20	200	571	458.745270
50000	350000	13	125	337	2469.871580	2	16	145	462	4496.271480
100000	700000	19	183	538	16109.807600	5	50	500	1672	31190.832000

Tabela 4.13: Desempenho de PGQP e PALQ no empacotamento de moléculas de uréia.

<i>nmol</i>	PB	PG	PBQ	PGQ	PGQP	PALQ
400	0.00E+00	2.26E-06	0.00E+00	1.11E-12	5.21E-06	0.00E+00
1000	2.16E-17	3.26E-07	0.00E+00	8.27E-07	8.47E-06	0.00E+00
5000	8.68E-09	2.83E-06	3.87E-06	2.26E-08	2.16E-07	0.00E+00
10000	1.30E-07	2.63E-06	1.16E-07	3.26E-07	4.94E-11	0.00E+00
50000	1.25E-08	4.12E-09	1.99E-08	9.35E-07	2.16E-07	0.00E+00
100000	8.51E-13	1.15E-06	4.45E-06	0.00E+00	3.80E-07	0.00E+00

Tabela 4.14: Valor da função objetivo na solução final.

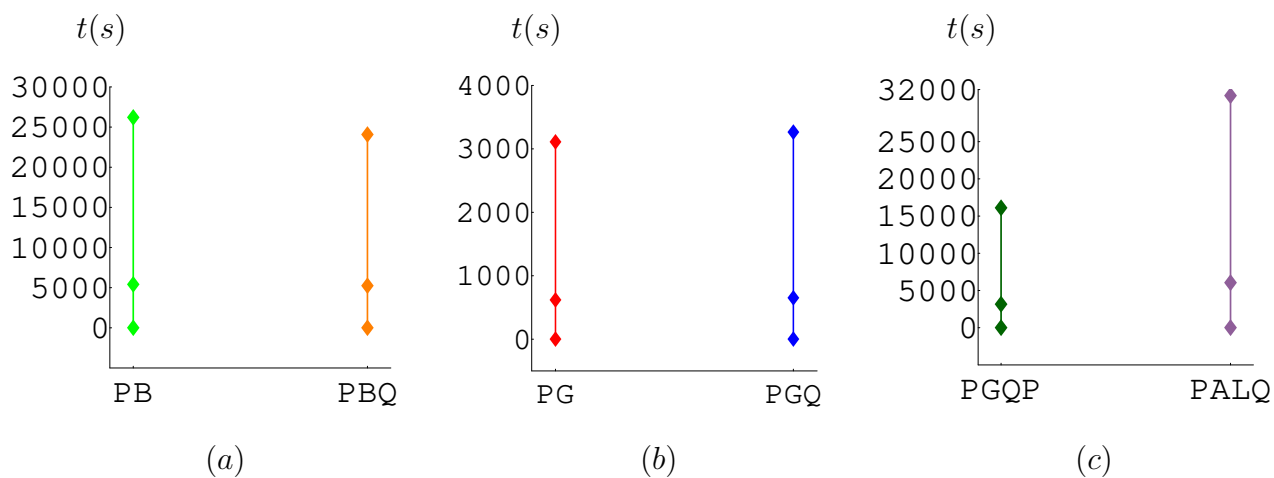


Figura 4.8: Comparando tempos no empacotamento de moléculas de uréia.

Empacotamento de misturas em caixas

Nos testes a seguir, empacotamos misturas em recipientes com formato de caixa. Três tipos de misturas foram consideradas. A primeira mistura é uma solução de água e uréia. A segunda contém água, uréia e duas moléculas de peptídeos que terão seus baricentros fixados em determinadas posições do recipiente. Isto significa que os peptídeos não participarão do processo de empacotamento e funcionarão apenas como obstáculos às moléculas de água e uréia que estarão livres. A terceira mistura é uma solução de água com tetracloreto de carbono (CCl_4). Nesta solução, adicionamos uma molécula de triiodotironina (T_3) que terá o mesmo papel das moléculas de peptídeo da solução anterior, ou seja, funcionará como obstáculo às moléculas livres. As quantidades de moléculas em cada experimento estão indicadas na Tabela 4.15 e os valores da função objetivo nas soluções finais de cada teste estão indicados na Tabela 4.16. Os comportamentos dos programas durante a execução dos experimentos estão resumidos na Tabela 4.17. Nesta tabela, a coluna correspondente às iterações internas do programa PALQ também fornece o número de iterações externas realizadas. Assim, 70–682 significa que PALQ realizou um total de 70 iterações externas e 682 iterações internas. A Figura 4.9 (a) mostra a média e os valores mínimo e máximo de tempo atingidos por cada programa. Na Figura 4.9 (b), a escala no eixo vertical foi reduzida para que os resultados obtidos possam ser mais bem visualizados. Mais uma vez podemos perceber que o programa PGQ é o que atinge o menor pico.

	Mistura 1	Mistura 2	Mistura 3
água	1000	700	1019
uréia	400	130	0
t. carbono	0	0	199
peptídeos	0	2	0
T_3	0	0	1

Tabela 4.15: Quantidades de moléculas em cada mistura.

	Mistura 1	Mistura 2	Mistura 3
PB	2.01E−06	0.00E+00	0.00E+00
PG	8.98E−06	4.81E−07	0.00E+00
PBQ	6.56E−27	0.00E+00	0.00E+00
PGQ	1.15E−06	4.92E−08	0.00E+00
PGQP	4.06E−07	1.66E−08	1.54E−07
PALQ	0.00E+00	0.00E+00	0.00E+00

Tabela 4.16: Função objetivo na solução final.

	Mistura 1				Mistura 2				Mistura 3			
	<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>	<i>nch</i>	<i>itin</i>	<i>naf</i>	<i>t(s)</i>
PB	6	60	93	28.613649	8	77	169	19.215078	5	48	69	15.336668
PG	15	130	315	21.905668	15	125	314	8.486709	8	68	146	4.288347
PBQ	7	64	136	23.701395	7	70	110	13.152000	5	47	87	11.547244
PGQ	12	100	237	11.187298	16	150	293	11.596236	9	70	154	3.560458
PGQP	15	129	378	31.054277	14	122	371	17.848286	9	79	238	5.982089
PALQ	8	70–682	1866	187.463501	9	74–710	1810	93.934715	4	35–339	732	34.552746

Tabela 4.17: Desempenho dos programas no empacotamento de misturas.

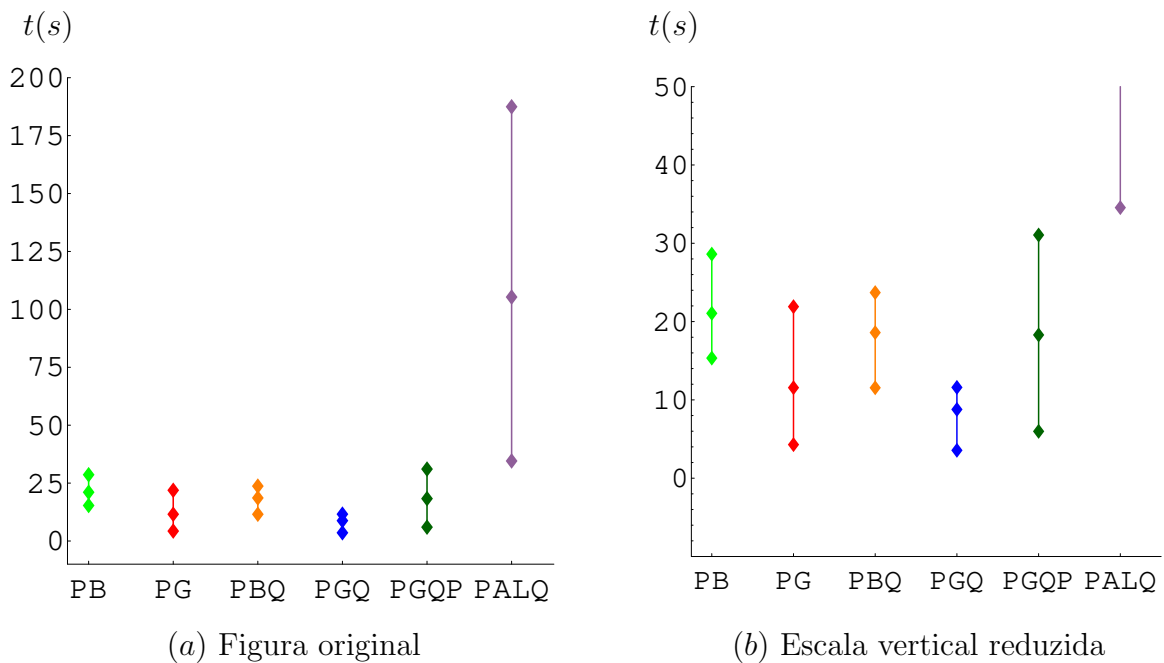


Figura 4.9: Comparando tempos no empacotamento de misturas.

Empacotamento de moléculas inventadas

Com o objetivo de investigar mais detalhadamente o comportamento dos programas, fizemos alguns testes de empacotamento com moléculas fictícias, isto é, inventamos algumas moléculas para serem empacotadas. Neste conjunto de testes, consideramos a molécula *estrela* (Figura 4.10 (a)), a molécula *guarda-chuva* (Figura 4.10 (b)) e a molécula *palito* (Figura 4.10 (c)). Admitindo os recipientes em formato de caixa, realizamos 5 testes com cada molécula inventada. A Tabela 4.2.4 mostra os tempos de execução (em segundos) obtidos por cada programa no empacotamento das moléculas. A coluna *nmol* desta tabela indica a quantidade de moléculas adotadas em cada experimento.

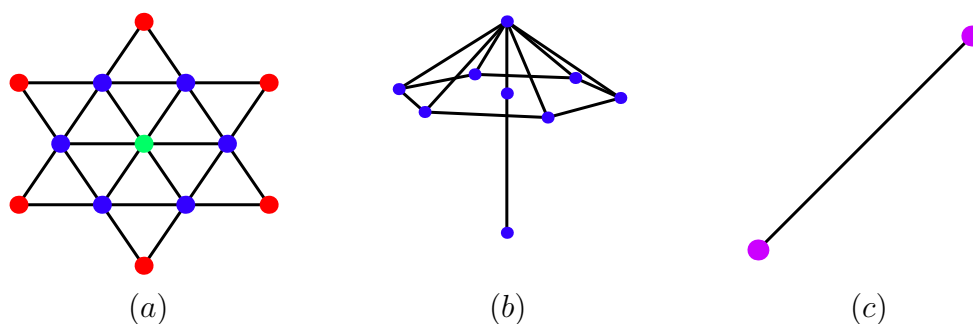


Figura 4.10: Moléculas inventadas.

Os valores máximo, mínimo e as médias de tempo obtidas por cada programa estão na Figura 4.11 (a). Podemos observar que os tempos máximos de execução atingidos pelas versões PB e PBQ estão muito próximos. O mesmo podemos observar com relação às versões PG e PGQ. É interessante observar que neste conjunto de testes PG e PGQ também obtiveram os melhores desempenhos (observe os picos na Figura 4.11 (a)). Por essa razão, na Figura 4.11 (b) comparamos apenas estes dois programas. Observemos que o tempo mínimo e a média são bastante parecidos nas duas versões, mas os valores máximos de tempo são ligeiramente diferentes.

Empacotamentos em recipientes de formatos diferentes

Dando continuidade aos experimentos de empacotamento, realizamos seis testes onde as moléculas são empacotadas em recipientes de formatos diferentes de caixa. A Tabela 4.19 mostra o tipo e a quantidade de moléculas envolvidas e o formato do recipiente adotado em cada experimento. As esferas dos três primeiros testes têm centro no ponto $(0, 0, 0)^T$ e raios respectivamente iguais a 50, 30, 100 Angstroms. Já os cilindros adotados nos testes 4, 5 e 6 têm a base sobre o plano XY e o eixo Z como eixo de simetria. O par (raio da base, altura) medidos em Angstroms dos cilindros são (20, 50), (10, 20) e (20, 60), respectivamente.

Neste conjunto de problemas comparamos apenas os desempenhos dos programas PG, PGQ, PGQP e PALQ pois as demais versões só admitem empacotamentos em recipientes com formato de caixa. Os tempos de execução (em segundos) dos programas estão na

Molécula estrela						
<i>nmol</i>	PB	PBQ	PG	PGQ	PGQP	PALQ
500	28.568655	28.646645	13.059012	18.010260	17.907276	60.235840
1000	61.577637	72.448990	28.394682	34.166805	62.433506	206.625580
5000	444.302429	351.038635	124.579056	244.540833	450.229553	1129.096440
10000	1230.845830	1254.420410	227.158463	831.476624	1482.883540	4156.967770
50000	23418.863300	21774.068400	4082.917240	3394.745850	30313.267600	49090.769500

Molécula guarda-chuva						
<i>nmol</i>	PB	PBQ	PG	PGQ	PGQP	PALQ
500	11.300281	9.043624	2.621601	2.691590	5.497163	18.579174
1000	24.369295	19.297064	5.832112	5.409176	9.239594	22.753540
5000	199.310699	161.895386	33.170956	32.083122	75.191574	235.572189
10000	501.477753	389.408783	87.370712	75.766487	238.669708	1404.280520
50000	8700.850590	7675.796880	892.616333	608.954407	2139.717770	6519.849120

Molécula palito						
<i>nmol</i>	PB	PBQ	PG	PGQ	PGQP	PALQ
500	5.517161	4.137370	1.398786	1.106831	1.270810	24.688246
1000	12.219142	10.106463	2.963549	2.123676	3.576450	62.083561
5000	80.090820	68.809540	18.801142	10.776361	18.299220	633.813660
10000	201.840317	130.669144	35.833550	23.440435	36.255490	1621.146480
50000	1317.118770	1268.357180	278.695648	101.781525	268.05127	9709.453120

Tabela 4.18: Tempos de execução nos problemas das moléculas inventadas.

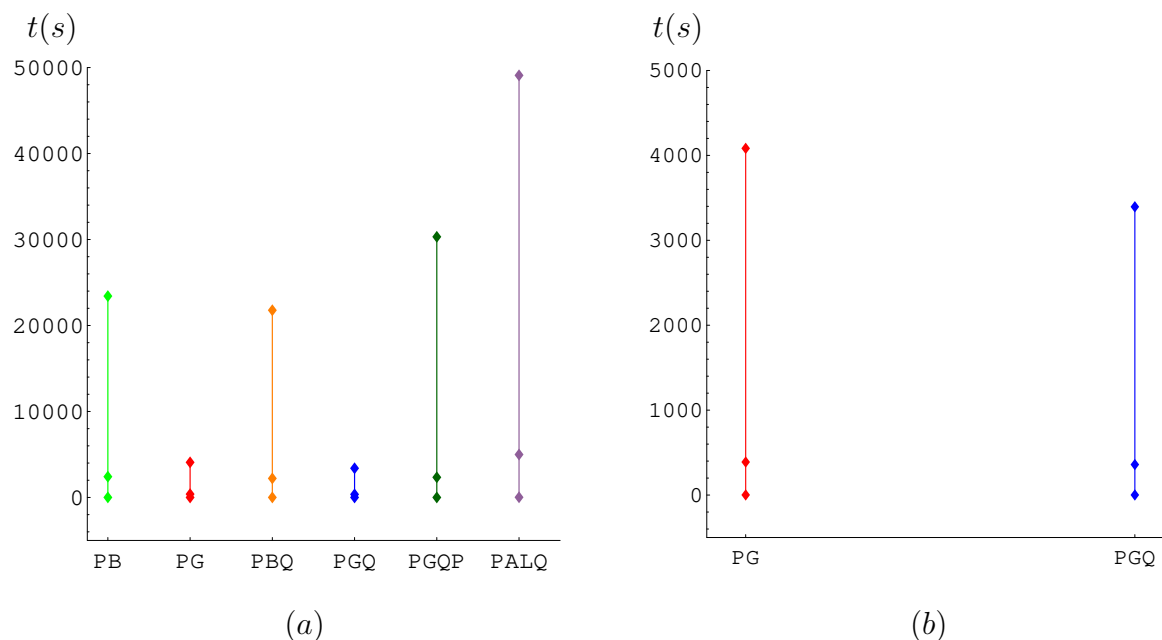


Figura 4.11: Comparando tempos no empacotamento de moléculas inventadas.

Tabela 4.20. Os gráficos das Figuras 4.12 (a) e 4.12 (b) mostram os valores máximos, mínimos e as médias de tempo atingidos por cada programa. Notemos que nestes testes a versão PG é a que resolve os problemas em menor tempo, por essa razão apresenta o menor pico como mostra a Figura 4.12 (a).

	Tipo	Quantidade	Recipiente
Teste 1	água	16500	esfera
Teste 2	estrela	1000	esfera
Teste 3	t. carbono	15000	esfera
Teste 4	uréia	1000	cilindro
Teste 5	g. chuva	200	cilindro
Teste 6	<i>palmitoil</i>	100	cilindro

Tabela 4.19: Descrição dos experimentos com recipientes de formatos diferentes.

Perfis de desempenho dos programas

Elizabeth Dolan e Jorge Moré introduzem em [7] uma nova ferramenta para a comparação do desempenho de n_s programas na resolução de um conjunto de n_p problemas: o gráfico denominado *perfil de desempenho*. Para a construção deste tipo de gráfico, devemos estabelecer um critério de comparação entre os programas. Este critério leva a uma medida, denotada por $m_{s,p}$, que deve ser computada por cada programa p durante a resolução de

	PG	PGQ	PGQP	PALQ
Teste 1	97.306083	472.909546	203.964752	978.405151
Teste 2	22.533407	41.058567	67.344207	136.464523
Teste 3	125.211823	206.05687	7729.311040	522.340637
Teste 4	13.008812	15.2609529	28.593784	397.820862
Teste 5	10.284642	3.98424888	7.632475	49.547096
Teste 6	8.76054573	152.78154	16.529031	36.238266

Tabela 4.20: Tempos de execução dos testes com recipientes diferentes.

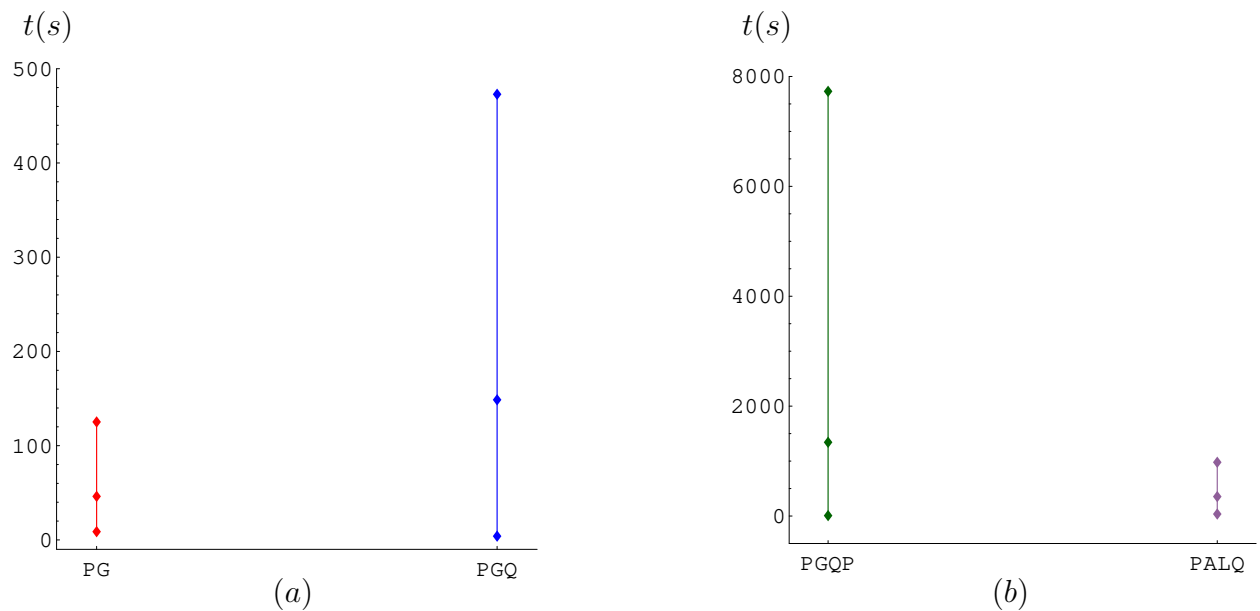


Figura 4.12: Comparando tempos nos testes com recipientes de formatos diferentes.

cada problema s . Por exemplo, $m_{s,p}$ pode ser o tempo gasto pelo programa s para resolver o problema p . Uma vez calculados os valores $m_{s,p}$ para todo s e p , determinamos as razões:

$$r_{s,p} = \begin{cases} \frac{m_{s,p}}{\min\{m_{s,p}, \forall s\}}, & \text{se } s \text{ resolve } p \\ r_M, & \text{caso contrário} \end{cases} \quad (4.51)$$

Em (4.51) temos sempre $r_{s,p} \geq 1$ e r_M deve ser tal que $r_M \geq r_{s,p} \forall s, p$. Assim, a partir dos cálculos das razões (4.51), construímos para cada programa s a função de distribuição acumulada $\rho_s : \mathbb{R} \mapsto [0, 1]$ dada por

$$\rho_s(\tau) = \frac{1}{n_p} \text{card}\{p \mid r_{s,p} \leq \tau\}. \quad (4.52)$$

Esta função é constante por partes, não decrescente e mostra o desempenho do programa s na resolução dos n_p problemas. Se a medida de comparação adotada for o tempo de execução dos programas, o valor $\rho_s(1)$ indica a porcentagem de problemas que o programa s conseguiu resolver em menor tempo. Por exemplo, se $\rho_s(1) = 0.6$, isto significa que o programa s conseguiu resolver 60 % dos problemas em menor tempo. Portanto, quanto maior for $\rho_s(1)$, mais *eficiente* será o programa s . Pela análise do perfil de desempenho de um programa podemos também constatar quão *robusto* o programa é. Existe um valor τ^* tal que $\forall \tau \geq \tau^*$ temos $\rho_s(\tau) = 1$. Desse modo, quanto menor for o valor de τ^* , mais robusto será o programa testado. Então, um programa ideal seria aquele em que $\rho(1) = 1$ e $\tau^* = 1$.

Para construir os perfis de desempenho das versões do programa *packmol*, adotamos como medida de comparação o tempo de execução. A Figura 4.13 mostra os perfis de desempenho dos programas nos 34 testes numéricos realizados com recipientes em forma de caixa. Notemos que neste conjunto de testes o programa PGQ mostrou o melhor desempenho pois resolveu cerca de 65 % dos problemas em menor tempo. Devemos observar também que a curva correspondente à versão PG atinge o nível 1 antes das demais, significando que esta versão mostrou-se mais robusta que as outras. Como era previsto, as versões PGQP, PALQ, PB e PBQ não apresentaram bons perfis de desempenho. Com efeito, os tempos gastos por esses programas para resolver os problemas são consideravelmente maiores que os tempos obtidos pelas versões PG e PGQ.

A Figura 4.14 mostra os perfis de desempenho dos programas quando desconsideramos os 5 experimentos realizados com a molécula fictícia *estrela*. Nota-se que os perfis dos programas PGQP, PALQ, PB e PBQ pouco mudam se os compararmos com os perfis da Figura 4.13. Porém, a versão PGQ, além de continuar sendo a mais eficiente, mostra-se também robusta pelo fato de sua curva atingir o nível 1 antes das curvas das outras versões.

Por fim, considerando os 40 experimentos realizados (empacotamento em caixas, esferas e cilindros), construímos os perfis de desempenho apenas das versões que utilizam a rotina GENCAN. O resultado está ilustrado na Figura 4.15. Pela figura percebemos que PGQ continua sendo a versão mais eficiente, pois resolve a maioria dos problemas em menor tempo. A versão mais robusta neste caso é PG pois sua curva atinge o nível 1 antes das demais versões.

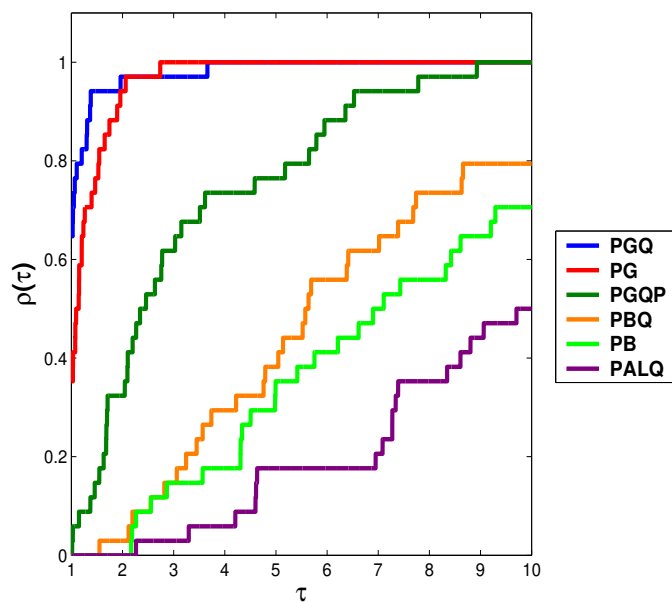


Figura 4.13: Perfis dos programas nos empacotamentos em caixas.

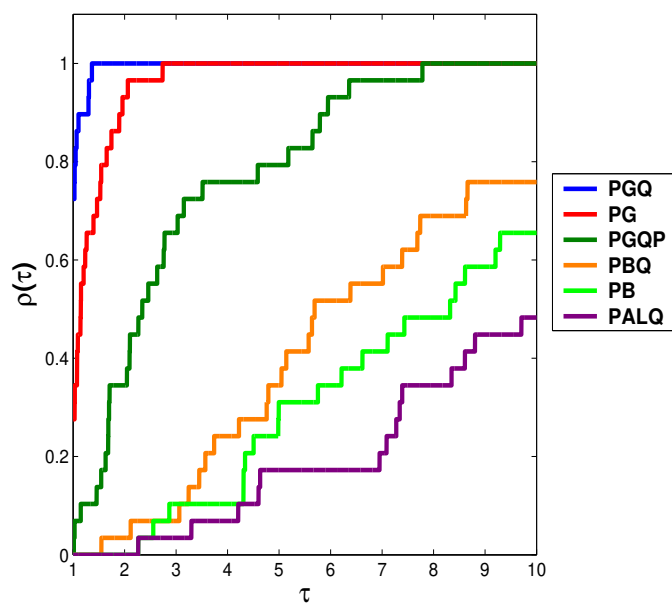


Figura 4.14: Perfis desconsiderando testes com molécula estrela.

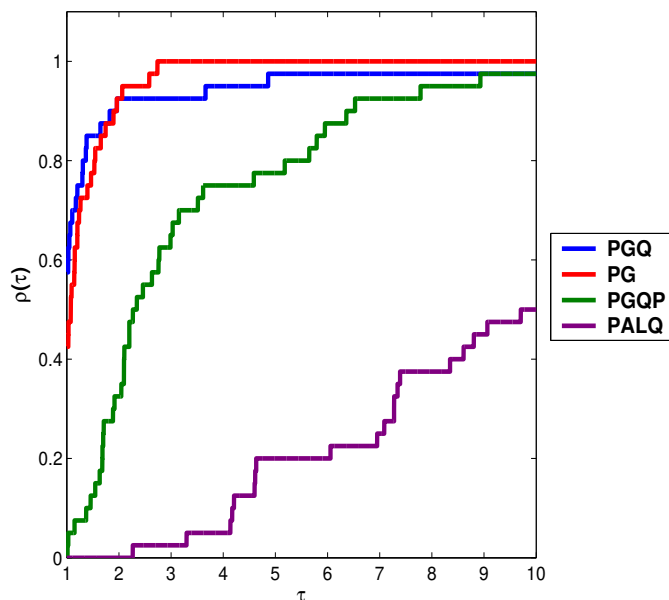


Figura 4.15: Perfis dos programas que usam a rotina GENCAN.

Alguns testes utilizando derivadas de segunda ordem analíticas

Nas versões PG e PGQ do programa *packmol* apenas o gradiente analítico é fornecido para a rotina de otimização GENCAN. Esta, por sua vez, necessita também de informações sobre as derivadas de segunda ordem da função objetivo, que devem ser fornecidas através do cálculo da matriz hessiana de $f(\mathbf{x})$, ou seja, $\nabla^2 f(\mathbf{x})$. Os elementos de $\nabla^2 f(\mathbf{x})$ não são fáceis de serem obtidos, sendo necessária, portanto, uma estratégia de programação muito bem elaborada para o cálculo desses elementos. As versões PG (ângulos de Euler) e PGQ (quatérnions normalizados) ao invés de calcularem a hessiana analítica, obtêm as derivadas de segunda ordem a partir do vetor gradiente. E, durante o processo de otimização, o cálculo de produtos hessiana-vetor é feito da seguinte forma

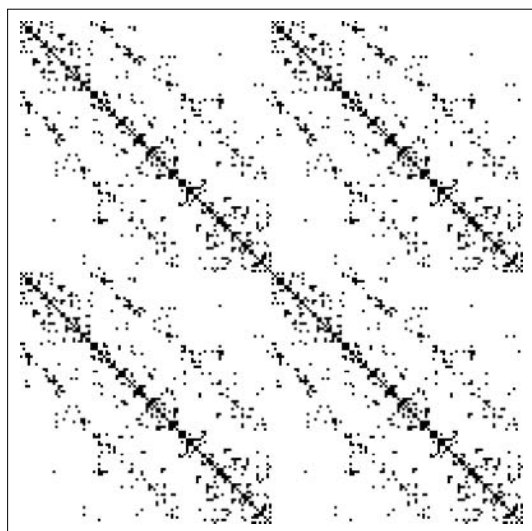
$$\nabla^2 f(\mathbf{x})\mathbf{d} \approx \frac{\nabla f(\mathbf{x}_k + h\mathbf{d}) - \nabla f(\mathbf{x}_k)}{h}, \quad (4.53)$$

onde $h > 0$ é um pequeno incremento.

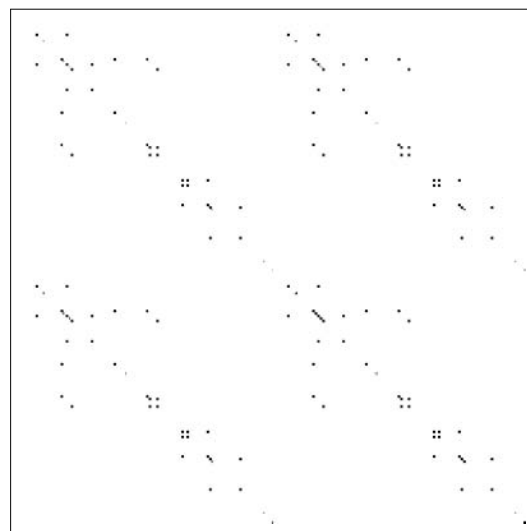
Para finalizar os experimentos de empacotamento de moléculas, resolvemos alguns problemas com as versões PG e PGQ onde a matriz hessiana verdadeira é fornecida para a rotina de otimização. As moléculas adotadas nestes testes são água e uréia e os recipientes têm o formato de caixa. A Tabela 4.21 mostra a quantidade de moléculas empacotadas em cada experimento e os tempos de execução (em segundos) obtidos pelas versões PG e PGQ em cada teste.

<i>nmol</i>	PG		PGQ	
	água	uréia	água	uréia
50	0.299953	0.593909	0.434933	1.216813
100	13.162998	1.105831	12.779055	2.041687
500	5.443171	17.539331	6.491012	19.624015
1000	53.727832	39.908931	18.822135	52.070083

Tabela 4.21: Tempos obtidos nos testes com hessiana verdadeira.

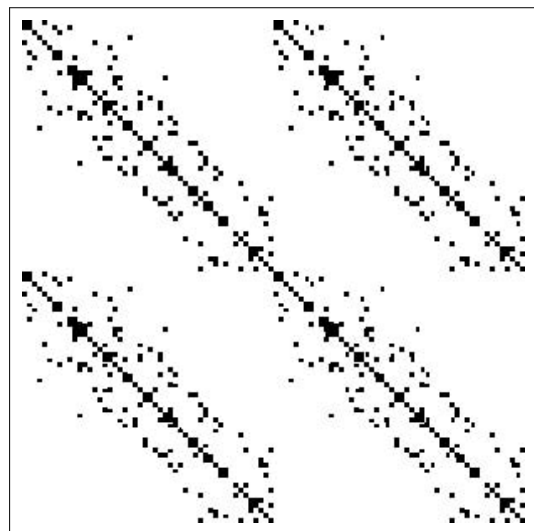


(a) 100 mol. água: 9 chamadas de Gencan

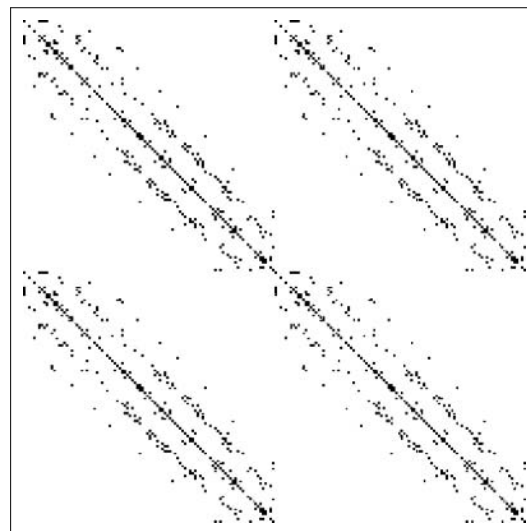


(b) 100 mol. água: 10 chamadas de Gencan

Figura 4.16: Estrutura da matriz hessiana nos testes com moléculas de água.



(a) 50 mol. uréia: 1 chamada de Gencan



(b) 100 mol. uréia: 1 chamada de Gencan

Figura 4.17: Estrutura da matriz hessiana em testes com moléculas de uréia.

O comportamento da matriz hessiana também foi investigado nos experimentos de empacotamento de 50 e 100 moléculas de água e uréia. As Figuras 4.16 (a) e 4.16 (b) mostram, respectivamente, a estrutura da matriz $\nabla^2 f(\mathbf{x})$ no empacotamento de 100 moléculas de água imediatamente após 9 e 10 chamadas da rotina GENCAN. Observemos que em 4.16 (a) a matriz é ainda bem densa (cada ponto preto na figura é um elemento não nulo), mas à medida que outras chamadas de GENCAN são efetuadas a matriz tende a se tornar esparsa como podemos comprovar em 4.16 (b), onde os elementos se concentram ao redor da diagonal principal e de duas sub-diagonais (uma inferior e outra superior). O mesmo comportamento pode ser observado nos testes com 50 e 100 moléculas de uréia logo após a primeira chamada de GENCAN, conforme indicam as Figuras 4.17 (a) e 4.17 (b).

As matrizes ilustradas nas figuras anteriores são do problema formulado com ângulos de Euler. Mas esse comportamento também foi observado no problema formulado com quatérnions. A explicação para o fato de $\nabla^2 f(\mathbf{x})$ se tornar esparsa durante o processo de otimização é a seguinte: as derivadas segundas da função objetivo dependem das coordenadas dos átomos das moléculas. Sendo assim, estas derivadas se tornarão nulas à medida que os átomos se encontrarem dentro do recipiente e toleravelmente separados. Desse modo, em testes de empacotamento com derivadas de segunda ordem analíticas, a matriz hessiana começa densa e vai se tornando esparsa ao longo do processo de otimização.

Capítulo 5

Conclusões

Neste trabalho estudamos um pouco sobre a teoria de rotações. Tínhamos como objetivo reunir algumas das principais características das rotações. Começamos caracterizando uma rotação no plano e em seguida tratamos das rotações no espaço tridimensional. Definimos o que é um movimento rígido em \mathbb{R}^3 e provamos algumas propriedades do conjunto das matrizes de rotação $SO(3)$. Dando continuidade ao estudo teórico, introduzimos também alguns conceitos de geometria diferencial para mostrar que não existe uma carta global para $SO(3)$, ou seja, não é possível cobrir todo o conjunto $SO(3)$ com apenas uma carta. Discutimos quais são as consequências práticas desse resultado em problemas cujo objetivo é controlar a orientação de objetos rígidos.

Elegemos na literatura as três maneiras mais comuns de se representar uma matriz de rotação em $SO(3)$ por meio de parâmetros e fizemos um breve estudo sobre cada uma dessas representações. Primeiramente estudamos o mapa exponencial, onde mostramos que uma rotação pode ser representada pelo par (ω, θ) (eixo + ângulo de rotação). Depois, vimos que qualquer elemento de $SO(3)$ pode ser descrito por três rotações (seqüência de Euler) em torno dos eixos coordenados (X, Y, Z) , sendo que duas rotações sucessivas não devem ocorrer em torno do mesmo eixo. E em seguida, estudamos um pouco da álgebra dos quatérnions e vimos como utilizá-los na representação de uma rotação. Terminamos a parte teórica deste trabalho fazendo uma comparação da eficiência computacional de cada representação estudada. Este estudo nos permitiu concluir que não existe uma melhor maneira de descrever o conjunto $SO(3)$ e a escolha de uma representação particular para este conjunto depende muito do problema em estudo.

Na parte aplicada deste projeto estudamos dois problemas de otimização cujas variáveis a serem determinadas envolvem rotações: o problema da orientação absoluta e o problema do empacotamento de moléculas. Ambos os problemas foram originalmente formulados utilizando-se ângulos de Euler na descrição das rotações. Reformulamos cada problema substituindo os ângulos por quatérnions e os resolvemos para comparar as soluções numéricas de cada formulação.

Resolvemos o problema da orientação absoluta com o pacote de otimização ALGENCAN. Ao comparar as soluções de cada formulação, não conseguimos detectar nenhuma diferença

entre elas. ALGENCAN conseguiu resolver os testes propostos sem nenhuma dificuldade e em um tempo muito pequeno. No problema do empacotamento de moléculas, criamos algumas versões diferentes para o programa *packmol* e observamos que na maioria dos testes de empacotamento o programa reformulado com quatérnions normalizados (versão PGQ) era ligeiramente mais rápido que o programa original (PG). De todas as versões construídas, aquela em que as normas de cada quatérnion são colocadas como restrição do problema de otimização (versão PALQ) foi a mais lenta, não sendo nem um pouco competitiva com a versão original. Além de fazermos tabelas comparativas dos dados obtidos pelas versões de *packmol*, construímos gráficos que medem o perfil de desempenho de cada versão no conjunto de testes resolvidos. Para terminar os experimentos, fizemos alguns testes com as versões PG e PGQ onde a matriz hessiana verdadeira era fornecida para a rotina de otimização. Os resultados não foram animadores porque os tempos de execução dos testes com hessiana verdadeira eram bem maiores do que nos testes em que a hessiana era obtida a partir do gradiente. Uma hipótese para esse fato é que não tomamos o cuidado de elaborar uma rotina de programação que explorasse a estrutura da hessiana, uma vez que esta matriz torna-se esparsa ao longo do processo de otimização.

Como trabalho futuro pretendemos retomar o problema do empacotamento de moléculas, considerando que as moléculas são corpos flexíveis, ou seja, admitem pequenas rotações internas. Investigaremos se esta característica permitirá empacotamentos que não são possíveis quando as moléculas são tratadas como corpos rígidos.

Apêndice A

Implementação do Algoritmo 4.1

Segue abaixo os detalhes da implementação do Algoritmo 4.1 descrito na Seção 4.1 do Capítulo 4. O *software* utilizado para implementação foi o *Mathematica*.

```
(* Cálculo dos centróides *)

vaux = Table[1, {npontos}];
centrEsq = (1/npontos)*(vaux.rEsq);
centrDir = (1/npontos)*(vaux.rDir);

(* Novas coordenadas com relação aos centróides *)

rEsqnew = Table[{0, 0, 0}, {npontos}]; %
rDirnew = Table[{0, 0,0}, {npontos}];

Do[rEsqnew[[i]] = rEsq[[i]] - centrEsq;
   rDirnew[[i]] = rDir[[i]] - centrDir, {i, 1, npontos}];

(*Cálculo da matriz M partir dos novos vetores de coordenadas*)

M = Sum[Outer[Times, rEsqnew[[i]], rDirnew[[i]]], {i, 1,npontos}];

(* Cálculo da matriz final a partir dos elementos de M *)

MatFinal = Outer[Times, {0, 0, 0, 0}, {0, 0, 0, 0}];

MatFinal[[1, 1]] = M[[1, 1]] + M[[2, 2]] + M[[3, 3]]; %
MatFinal[[1, 2]] = MatFinal[[2, 1]] = M[[2, 3]] - M[[3, 2]] ;
MatFinal[[1, 3]] = MatFinal[[3, 1]] = M[[3, 1]] - M[[1, 3]] ;
MatFinal[[1, 4]] = MatFinal[[4, 1]] = M[[1, 2]] - M[[2, 1]] ;
MatFinal[[2, 2]] = M[[1, 1]] - M[[2, 2]] - M[[3, 3]]; %
MatFinal[[2, 3]] = MatFinal[[3, 2]] = M[[1, 2]] + M[[2, 1]] ;
MatFinal[[2, 4]] = MatFinal[[4, 2]] = M[[3, 1]] + M[[1, 3]];
MatFinal[[3, 3]] = -M[[1, 1]] + M[[2, 2]] - M[[3, 3]]; %
MatFinal[[3, 4]] = MatFinal[[4, 3]] = M[[2, 3]] + M[[3, 2]];
MatFinal[[4, 4]] = -M[[1, 1]] - M[[2, 2]] + M[[3, 3]];
```

```

(* Cálculo dos autovalores e autovetores de MatFinal *)

autovalores = Eigenvalues[MatFinal]; %
autovetores = Eigenvectors[MatFinal];

(* Determinando o autovalor mais positivo e o autovetor
correspondente *)

pos = Position[autovalores, Max[autovalores]]; %
quat = autovetores[[pos[[1, 1]]]];

(* Construindo a matriz de rotação a partir do quatérnion *)

R = {{2*(quat[[1]]^2 + quat[[2]]^2), 2*(quat[[2]]*quat[[3]] - quat[[1]]*quat[[4]]),
2*(quat[[2]]*quat[[4]] + quat[[1]]*quat[[3]])}, {2*(quat[[2]]*quat[[3]] +
quat[[1]]*quat[[4]]), 2*(quat[[1]]^2 + quat[[3]]^2), 2*(quat[[3]]*quat[[4]] -
quat[[1]]*quat[[2]])}, {2*(quat[[2]]*quat[[4]] - quat[[1]]*quat[[3]]),
2*(quat[[3]]*quat[[4]] + quat[[1]]*quat[[2]]), 2*(quat[[1]]^2 + quat[[4]]^2)}}
-IdentityMatrix[3];

(* Cálculo do fator de escala *)

If[flag == 0, (* Se flag = 0 faça *)
  snumer = Sum[Dot[rDirnew[[i]], rDirnew[[i]]], i, 1, npontos];
  sdenom = Sum[Dot[rEsqnew[[i]], rEsqnew[[i]]], i, 1, npontos];
  fatoresc = Sqrt[snumer/sdenom],
  (* caso contrário faça *)
  snumer = Sum[Dot[rDirnew[[i]], R.rEsqnew[[i]]], i, 1, npontos];
  sdenom = Sum[Dot[rEsqnew[[i]], rEsqnew[[i]]], i, 1, npontos];
  fatoresc = snumer/sdenom];

(* Cálculo do vetor deslocamento *)

vetdesloc = centrDir - fatoresc*R.centresq;

(* Mostrando resultados na tela *)

Print["Resultados Finais:" ] %
Print["flag = ", flag] %
Print["vetordeslocamento = ", vetdesloc] %
Print["fator de escala = ", fatoresc] %
Print["quatérnion = ", quat] %
Print["matriz de rotação = ", R// MatrixForm]
(*-----*)

```

Referências Bibliográficas

- [1] R. Andreani, E. G. Birgin, J. M. Martínez and M. L. Schuverdt, “On Augmented Lagrangian methods with general lower-level constraints”, Technical Report MCDO-050304, Departamento de Matemática Aplicada, UNICAMP, 2005.
- [2] R. Andreani, E. G. Birgin, J. M. Martínez and M. L. Schuverdt, “Augmented Lagrangian methods under the Constant Positive Linear Dependence constraint qualification”, aceito para publicação em *Mathematical Programming*.
- [3] A. H. Barr, B. Currin, S. Gabriel, J. F. Hughes, “Smooth Interpolation of Orientations with Angular Velocity Constraints using Quaternions”, *Computer Graphics*, vol. 26, no. 2, 1992.
- [4] E. G. Birgin, J. M. Martínez “Large-scale active-set box-constrained optimization method with spectral projected gradients”, *Computational Optimization and Applications*, vol. 23, pp. 101–125, 2002.
- [5] M. J. Crowe, *A History of Vector Analysis: The Evolution of the Idea of a Vectorial System*, New York: Dover Publications, 1985, 288 p.
- [6] M. L. Curtis, *Matrix Groups*, New York: Springer-Verlag, 1984, 228 p.
- [7] E. D. Dolan, J. J. Moré, “Benchmarking optimization software with performance profiles”, *Mathematical Programming*, vol. 91, pp. 201–213, 2002.
- [8] D. Eberly, “Rotation Representations and Performance Issues”, disponível em: <http://www.geometrictools.com/Documentation/RotationIssues.pdf>, 2002.
- [9] A. Friedlander, J. M. Martínez, S. A. Santos, “A new trust region strategy for box-constrained minimization”, *Applied Mathematics and Optimization*, vol. 30, pp. 235–266, 1994.
- [10] J. Gomes, L. Velho, *Fundamentos da Computação Gráfica*, Rio de Janeiro: IMPA, 2003, 624 p.
- [11] F. S. Grassia, “Practical Parametrization of Rotations Using the Exponential Map”, *Journal of Graphics Tools*, vol.3, no. 3, pp. 29–48, 1998.

- [12] A. J. Hanson, *Visualizing Quaternions*, Morgan Kaufmann, 2006, 600 p.
- [13] B. K. P. Horn, “Closed-form solution of absolute orientation using unit quaternions”, *Journal of the Optical Society American*, vol. 4, pp. 629–642, 1987.
- [14] B. K. P. Horn, *Robot Vision*, MIT Press: McGraw-Hill, 1986, 480 p.
- [15] W. Humphrey, A. Dalke, K. Schulten, “VMD - Visual Molecular Dynamics”, *Journal of Molecular Graphics*, vol. 14, pp. 33–38, 1996.
- [16] T. R. Kane, P. W. Likins, D. A. Levinson, *Spacecraft Dynamics*, McGraw-Hill, 1983, 442 p.
- [17] E. Kreyszig, *Introductory Functional Analysis with Applications*, John Wiley & Sons, 1978, 704 p.
- [18] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*, New Jersey: Princeton University Press, 2002, 400 p.
- [19] W. Kühnel, *Differential Geometry: Curves - Surfaces - Manifolds*, AMS, Student Mathematical Library, vol. 16, 2005, 380 p.
- [20] P. Lounesto, *Clifford Algebras and Spinors*, Cambridge University Press, 1997, 352 p.
- [21] J. E. Marsden, T. S. Ratiu, *Introduction to Mechanics and Symmetry: A Basic Exposition of Mechanical Systems*, New York: Springer-Verlag, 1999, 584 p.
- [22] J. M. Martínez, L. Martínez, “Packing optimization for automated generation of complex system’s initial configurations for molecular dynamics and docking”, *Journal of Computational Chemistry*, vol. 24, pp. 819–825, 2003.
- [23] J. Nocedal, S. Wright, *Numerical Optimization*, New York: Springer-Verlag, 1999, 656 p.
- [24] R. M. Murray, Z. Li, S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, 1994, 480 p.
- [25] G. M. Nielson, “Smooth Interpolation of Orientations”, *Models and Techniques in Computer Animation*, Springer, pp. 75–93, 1993.
- [26] R. Penrose, W. Windler, *Spinors and Space-Time: Two-Spinor Calculus and Relativistic Fields*, Cambridge University Press, vol. 1, 1984, 478 p.
- [27] M. D. Shuster, “A Survey of Attitude Representations”, *The Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, 1993.

- [28] M. W. Spong, M. Vidyasagar, *Robot Dynamics and Control*, John Wiley & Sons, 1989, 352 p.
- [29] J. Stuelpnagel, “On the parametrization of the three-dimensional rotation group”, *SIAM Review*, vol. 6, no. 4, pp. 422–430, 1964
- [30] C. J. Taylor, D. J. Kriegman, “Minimization on the Lie Group $SO(3)$ and Related Manifolds”, Technical Report 9405, abril, 1994.
- [31] A. Watt, M. Watt, *Advanced Animation and Rendering Techniques: Theory and Practice*, England: Addison-Wesley, 1992, 472 p.
- [32] P. R. Wolf, B. A. Dewitt, *Elements of Photogrammetry with Applications in GIS*, McGraw-Hill, 2000, 624 p.