Um Sistema para Animação de Estruturas Articuladas através da técnica *Spacetime Constraints*

por

João Carlos da Silveira Costa Filho

Dissertação apresentada ao
Mestrado em Ciência da Computação
Universidade Federal do Ceará

Fortaleza, Ceará, Brasil 2004.

Orientador: Creto Augusto Vidal, PhD

Agradecimentos

Agradeço em primeiro lugar a Deus por ter me dado forças para superar todos os obstáculos. Aos meus pais, pelo carinho e apoio em todos os momentos da minha vida. Ao professor Creto, pela paciência, apoio e confiança depositada até o último momento no meu trabalho. À minha namorada, pela compreensão e incentivo. A todos os amigos que torceram por mim. E, por fim, a todos que contribuíram de alguma forma para que as idéias se materializassem.

Obrigado.

Sumário

A animação através do computador tem sido explorada nas mais diversas áreas, como, por exemplo, em filmes e jogos de computador. Muitas tentativas foram feitas para tentar solucionar o problema da movimentação realística dos corpos dos personagens através um mínimo de interação por parte do animador. Essa dissertação estuda um modelo de sistema baseado na técnica de otimização de estruturas articuladas por otimização não-linear com restrições, denominada *Spacetime Constraints*.

Summary

Computer animation has been explored in many areas of human knowledge, for instance, movies and computer games. Many experiments were done trying to solve the problem of realistic motion of figures body with the minimum interaction of animator. This work describes a model of a system based in an optimization technique of articulated figures through non-linear optimization with restriction. This technique is called Spacetime Constraints and tries to do animation with a balance between control and automation.

Índice

Agradecimentos ..... **i**
Sumário ..... **ii**
Summary ..... **iii**
Índice ..... **iv**
Lista de Figuras ..... **vii**
Lista de tabelas ..... **xii**
Capítulo 1 Introdução **1**
1.1 Motivação 1
1.2 Objetivos 4
1.3 Estrutura da dissertação 4
Capítulo 2 Técnicas de Animação de Figuras Articuladas **6**
2.1 Introdução 6
2.2 Métodos Baseados em Cinemática 6
2.2.1 Quadros-chaves 6
2.2.2 Captura de movimento 8
2.2.3 Morphing 13
2.2.4 Cinemática Direta 15
2.2.5 Cinemática Inversa 17
2.3 Métodos Baseados em Dinâmica 18
2.3.1 Dinâmica Direta 19
2.3.2 Dinâmica Inversa 21
2.3.3 Controladores 22
2.3.4 Controle por otimização 23
2.4 Comparação entre os métodos de animação 26
2.5 Considerações Finais 26
Capítulo 3 A Técnica de Controle de Movimento de Estruturas Articuladas por otimização não-linear **28**
3.1 Introdução 28
3.2 Descrição da Técnica 29
3.2.1 Modelagem físico-matemática da estrutura articulada 31
3.2.2 Método de otimização 33
3.3 Considerações Finais 39

Capítulo 4 Sistema de Animação Baseado na Técnica de Controle de movimento por otimização com restrições 40
4.1 Introdução 40
4.2 Descrição do Sistema 40
4.2.1 Subsistema de Interfaces Gráficas 42
Descrição da estrutura articulada 43
Descrição do modelo de otimização 44
Ativação do processo de solução 45
Visualização da Animação 46
4.2.2 Subsistema de Processamento Simbólico 47
Processamento simbólico das Restrições Newtonianas 49
Processamento simbólico da Função Objetivo 49
4.2.3 Subsistema de Otimização 50
4.3 Considerações Finais 51
Capítulo 5 Apresentação e Análise dos Resultados 52
5.1 Introdução 52
5.2 Apresentação dos exemplos 52
5.2.1 Minimização da Potência Angular 54
5.2.2 Minimização da Trajetória em relação à Base 66
5.2.3 Minimização da Trajetória em relação ao centro de massa 78
5.3 Análise dos Resultados 90
5.3.1 Observações sobre controle da animação 90
5.3.2 Observações sobre realismo da animação 111
5.3.3 Observações sobre tempo de geração da animação 114
Capítulo 6 Conclusões e Recomendações 116
Anexo A Conceitos básicos 119
Anexo B Equações do movimento 120
B.1 Velocidade 120
B.2 Energia cinética 120
B.3 Inércia rotacional 124
B.4 Dinâmica de Lagrange 126
Anexo C Equações de lagrange na Dinâmica inversa 127
Anexo D Modelagem no *Mathematica* 128
Anexo E Um algoritmo simples para SQP na otimização de figuras articuladas 133
Anexo F Classes Principais do Sistema 135

Anexo G Código de implementação do Sistema 139
Bibliografia.....185

Lista de Figuras

Figura 2.1 - Interpolação com quadros chaves (Discreet, 2004)..... 7
Figura 2.2 - Animação tradicional (Disney, 1937)..... 7
Figura 2.3 - Pacote de animação baseada em Quadros-chaves (Discreet, 2004)..... 8
Figura 2.4 – Animação de figuras articuladas com quadros-chaves..... 8
Figura 2.5 - Sistema de captura de movimento (Meta Motion, 2004)..... 9
Figura 2.6 - Curvas do movimento humano (Popović, 1999)..... 10
Figura 2.7 – Sistema óptico (Mellor, 2004)..... 11
Figura 2.8 – Sistema magnético (Mellor, 2004)..... 12
Figura 2.9 – Sistema mecânico (Mellor, 2004)..... 13
Figura 2.10 - Exemplo de animação com Morphing (Cohen-Or, 1998)..... 14
Figura 2.11 - Corpo humano em um sistema multicamadas (Thalmann et al., 1996b)..... 14
Figura 2.12 - Interpolação na cinemática direta..... 15
Figura 2.13 – Juntas de uma figura articulada..... 16
Figura 2.14 - Cinemática inversa de uma figura articulada..... 17
Figura 2.15 – Dinâmica num corpo articulado..... 20
Figura 2.16 - Software para analise dinâmica com aplicação na medicina (MusculoGraphics, 2002)..... 20
Figura 2.17 - Sistema de simulação de soldados (Digital Biomechanics, 2004)..... 21
Figura 2.18 - Movimento usando a dinâmica inversa..... 22
Figura 2.19 - Exemplo de um sistema controlado por *feedback*..... 23
Figura 2.20 – Espaço de busca de uma partícula..... 25
Figura 2.21 - Balanceamento entre as técnicas de animação..... 26
Figura 3.1 - Animação Luxo Jr. (Pixar 1986)..... 28
Figura 3.2 - Movimento da estrutura..... 29
Figura 3.3 - Figura articulada usada na modelagem..... 30
Figura 3.4 – Discretização e restrições..... 35
Figura 3.5 - Decisões na otimização..... 36
Figura 3.6 - Exemplo de uma função objetivo..... 37
Figura 4.1 – Modularidade do sistema..... 40
Figura 4.2 - Fluxo do sistema..... 41
Figura 4.3 - Diagrama do Sistema..... 41
Figura 4.4 - Janela principal..... 42
Figura 4.5 – Centro de decisões..... 42

Figura 4.6 – *Tab* Descrição da Estrutura Articulada 43
Figura 4.7 – Caixas de diálogos de informações da estrutura 43
Figura 4.8 – *Tab* Descrição do Modelo de Otimização 44
Figura 4.9 - Restrições de pose 44
Figura 4.10 - O critério do objetivo baseado numa curva de trajetória 45
Figura 4.11 - *Tab* Processo de solução 46
Figura 4.12 - *Tab* Visualização da Animação 46
Figura 4.13 - A janela de animação 47
Figura 4.14 - Plotagem do movimento da base 47
Figura 4.15 - Mensagens do Subsistema de Processamento Simbólico 48
Figura 4.16 - Subsistema de Otimização 50
Figura 5.1 – Janelas do Sistema para o Exemplo MPA1a 55
Figura 5.2 – Janelas do Sistema para o Exemplo MPA1b 56
Figura 5.3 – Janelas do Sistema para o Exemplo MPA1c 58
Figura 5.4 – Janelas do Sistema para o Exemplo MPA2a 59
Figura 5.5 – Janelas do Sistema para o Exemplo MPA2b 60
Figura 5.6 – Janelas do Sistema para o Exemplo MPA2c 62
Figura 5.7 – Janelas do Sistema para o Exemplo MPA3a 63
Figura 5.8 – Janelas do Sistema para o Exemplo MPA3b 64
Figura 5.9 – Janelas do Sistema para o Exemplo MPA3c 66
Figura 5.10 – Janelas do Sistema para o Exemplo MTB1a 67
Figura 5.11 – Janelas do Sistema para o Exemplo MTB1b 68
Figura 5.12 – Janelas do Sistema para o Exemplo MTB1c 70
Figura 5.13 – Janelas do Sistema para o Exemplo MTB2a 71
Figura 5.14 – Janelas do Sistema para o Exemplo MTB2b 72
Figura 5.15 – Janelas do Sistema para o Exemplo MTB2c 74
Figura 5.16 – Janelas do Sistema para o Exemplo MTB3a 75
Figura 5.17 – Janelas do Sistema para o Exemplo MTB3b 76
Figura 5.18 – Janelas do Sistema para o Exemplo MTB3c 78
Figura 5.19 – Janelas do Sistema para o Exemplo MTC1a 79
Figura 5.20 – Janelas do Sistema para o Exemplo MTC1b 81
Figura 5.21 – Janelas do Sistema para o Exemplo MTC1c 83
Figura 5.22 – Janelas do Sistema para o Exemplo MTC2a 84
Figura 5.23 – Janelas do Sistema para o Exemplo MTC2c 85
Figura 5.24 – Janelas do Sistema para o Exemplo MTC3a 86

Figura 5.25 – Janelas do Sistema para o Exemplo MTC3b 88
Figura 5.26 – Janelas do Sistema para o Exemplo MTC3c 89
Figura 5.27 – Comparação do resultado para a variável x_0 entre os exemplos MPA1a, MPA1b e MPA1c 96
Figura 5.28 – Comparação do resultado para a variável y_0 entre os exemplos MPA1a, MPA1b e MPA1c 97
Figura 5.29 – Comparação do resultado para a variável q_1 entre os exemplos MPA1a, MPA1b e MPA1c 97
Figura 5.30 – Comparação do resultado para a variável q_2 entre os exemplos MPA1a, MPA1b e MPA1c 98
Figura 5.31 – Comparação do resultado para a variável q_3 entre os exemplos MPA1a, MPA1b e MPA1c 98
Figura 5.32 – Comparação do resultado para a variável x_0 entre os exemplos MTB1a, MTB1b e MTB1c 99
Figura 5.33 – Comparação do resultado para a variável y_0 entre os exemplos MTB1a, MTB1b e MTB1c 99
Figura 5.34 – Comparação do resultado para a variável q_1 entre os exemplos MTB1a, MTB1b e MTB1c 100
Figura 5.35 – Comparação do resultado para a variável q_2 entre os exemplos MTB1a, MTB1b e MTB1c 100
Figura 5.36 – Comparação do resultado para a variável q_3 entre os exemplos MTB1a, MTB1b e MTB1c 101
Figura 5.37 – Comparação do resultado para a variável x_0 entre os exemplos MTC1a, MTC1b e MTC1c 101
Figura 5.38 – Comparação do resultado para a variável y_0 entre os exemplos MTC1a, MTC1b e MTC1c 102
Figura 5.39 – Comparação do resultado para a variável q_1 entre os exemplos MTC1a, MTC1b e MTC1c 102
Figura 5.40 – Comparação do resultado para a variável q_2 entre os exemplos MTC1a, MTC1b e MTC1c 103

Figura 5.41 – Comparação do resultado para a variável q_3 entre os exemplos MTC1a, MTC1b e MTC1c 103
Figura 5.42 – Comparação do resultado para a variável x_0 entre os exemplos MPA1c, MPA2c e MPA3c 104
Figura 5.43 – Comparação do resultado para a variável y_0 entre os exemplos MPA1c, MPA2c e MPA3c 104
Figura 5.44 – Comparação do resultado para a variável q_1 entre os exemplos MPA1c, MPA2c e MPA3c 105
Figura 5.45 – Comparação do resultado para a variável q_2 entre os exemplos MPA1c, MPA2c e MPA3c 105
Figura 5.46 – Comparação do resultado para a variável q_3 entre os exemplos MPA1c, MPA2c e MPA3c 106
Figura 5.47 – Comparação do resultado para a variável x_0 entre os exemplos MTB1c e MTB3c 106
Figura 5.48 – Comparação do resultado para a variável y_0 entre os exemplos MTB1c e MTB3c 107
Figura 5.49 – Comparação do resultado para a variável q_1 entre os exemplos MTB1c e MTB3c 107
Figura 5.50 – Comparação do resultado para a variável q_2 entre os exemplos MTB1c e MTB3c 108
Figura 5.51 – Comparação do resultado para a variável q_3 entre os exemplos MTB1c e MTB3c 108
Figura 5.52 – Comparação do resultado para a variável x_0 entre os exemplos MTC1c e MTC2c 109
Figura 5.53 – Comparação do resultado para a variável y_0 entre os exemplos MTC1c e MTC2c 109
Figura 5.54 – Comparação do resultado para a variável q_1 entre os exemplos MTC1c e MTC2c 110
Figura 5.55 – Comparação do resultado para a variável q_2 entre os exemplos MTC1c e MTC2c 110

Figura 5.56 – Comparação do resultado para a variável q_3 entre os exemplos MTC1c e MTC2c 111
Figura 5.57 – Comparação do resultado para a variável x_0 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c.....112
Figura 5.58 – Comparação do resultado para a variável y_0 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c.....112
Figura 5.59 – Comparação do resultado para a variável q_1 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c.....113
Figura 5.60 – Comparação do resultado para a variável q_2 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c.....113
Figura 5.61 – Comparação do resultado para a variável q_3 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c.....114
Figura 5.62 - Tempo da otimização para o exemplo MPA1 com 8, 16, 32 e 64 discretizações 115

Lista de tabelas

Tabela 2.1 - Comparação entre os métodos de controle por dinâmica. 26
Tabela 3.1 – Topologia e propriedade físicas 32
Tabela 4.1 – Trecho do código das equações de Lagrange através do *mathlink*..... 49
Tabela 5.1 – Descrição da estrutura articulada 53
Tabela 5.2 – Resultado para o Exemplo MPA1a 54
Tabela 5.3 – Resultado para o Exemplo MPA1b 55
Tabela 5.4 – Resultado para o Exemplo MPA1c 57
Tabela 5.5 – Resultado para o Exemplo MPA2a 58
Tabela 5.6 – Resultado para o Exemplo MPA2b 59
Tabela 5.7 – Resultado para o Exemplo MPA2c 61
Tabela 5.8 – Resultado para o Exemplo MPA3a 62
Tabela 5.9 – Resultado para o Exemplo MPA3b 63
Tabela 5.10 – Resultado para o Exemplo MPA3c 65
Tabela 5.11 – Resultado para o Exemplo MTB1a 67
Tabela 5.12 – Resultado para o Exemplo MTB1b 68
Tabela 5.13 – Resultado para o Exemplo MTB1c 69
Tabela 5.14 – Resultado para o Exemplo MTB2a 71
Tabela 5.15 – Resultado para o Exemplo MTB2b 72
Tabela 5.16 – Resultado para o Exemplo MTB2c 73
Tabela 5.17 – Resultado para o Exemplo MTB3a 75
Tabela 5.18 – Resultado para o Exemplo MTB3b 76
Tabela 5.19 – Resultado para o Exemplo MTB3c 77
Tabela 5.20 – Resultado para o Exemplo MTC1a 79
Tabela 5.21 – Resultado para o Exemplo MTC1b 80
Tabela 5.22 – Resultado para o Exemplo MTC1c 81
Tabela 5.23 – Resultado para o Exemplo MTC2a 83
Tabela 5.24 – Resultado para o Exemplo MTC2c 84
Tabela 5.25 – Resultado para o Exemplo MTC3a 86
Tabela 5.26 – Resultado para o Exemplo MTC3b 87
Tabela 5.27 – Resultado para o Exemplo MTC3c 88
Tabela 5.28 – Comparação entre intervalos no tempo igual entre os exemplos MPA1a, MPA1b e MPA1c 90

Tabela 5.29 – Comparação entre intervalos no tempo igual entre os exemplos MPA2a, MPA2b e MPA2c 91
Tabela 5.30 – Comparação entre intervalos no tempo igual entre os exemplos MPA3a, MPA3b e MPA3c 92
Tabela 5.31 – Comparação entre intervalos no tempo igual entre os exemplos MTB1a, MTB1b e MTB1c 93
Tabela 5.32 – Comparação entre intervalos no tempo igual entre os exemplos MTB3a, MTB3b e MTB3c 94
Tabela 5.33 – Comparação entre intervalos no tempo igual entre os exemplos MTC1a, MTC1b e MTC1c 95
Tabela 5.34 – Tempo gasto pela CPU em busca de uma solução ótima 115

Capítulo 1
Introdução

1.1 Motivação

Animação é um problema bastante interessante que ainda está em desenvolvimento na área de computação gráfica, e é considerada hoje no mundo, uma atividade de fundamental importância em diversos campos, tais como: a indústria de jogos e entretenimento, fisiologia dos movimentos, biomecânica, realidade virtual, entre outros.

Criar animações, com ou sem o auxílio do computador, sempre foi um processo que consome muito do tempo do artista. Para que essa tarefa seja realizada com sucesso, é necessário que o animador tenha aptidão natural para vislumbrar como será o produto final e decidir sobre o melhor processo de construção da animação. Com o advento do computador, grande parte dessa tarefa é tratada de forma automática, mas o processo ainda exige um grande esforço de tentativas e ajustes por parte do animador até atingir um resultado satisfatório.

O grande problema da animação tradicional é que ela depende exclusivamente da habilidade do animador em executar animações convincentes (Boulic & Thalmann, 1992). Na animação por computador, por sua vez, a grande dificuldade está em encontrar técnicas que estabeleçam um equilíbrio entre a automação do processo de criação das seqüências e o nível de controle que pode ser exercido pelo animador (Liu & Cohen, 1995).

Antes das animações serem feitas por computador, o animador resolvia o problema através de um processo de tentativas tendo que criar uma grande quantidade de desenhos que, para criar a ilusão de movimento, deveriam ser exibidos numa velocidade de 24 a 30 quadros por segundo. Cada quadro tem que ser desenhado ou pintado individualmente. Grandes projetos de animação usavam uma técnica chamada de quadros-chaves (Welman, 1989; Madhavapeddy, 2004), onde o animador criava a pintura ou desenho de importantes quadros da animação e outros artistas

com menor talento eram contratados para desenhar transições entre os quadros chaves. Esse processo necessitava de um longo tempo até a composição final do trabalho.

Com a introdução do computador como uma ferramenta de auxílio no processo de animação, várias técnicas foram desenvolvidas, inicialmente, com objetivo principal de minimizar o trabalho do animador. Assim, a técnica de quadros-chave foi desenvolvida reproduzindo computacionalmente o processo de animação tradicional. Através dessa técnica o animador, de posse dos modelos dos objetos ou dos personagens que serão animados, tem que localizar e orientar cada um desses objetos para cada um dos quadros principais, que serão desenhados pelo computador, enquanto que os quadros intermediários de transição podem ser gerados por um processo de interpolação. Para animações de estruturas articuladas, as técnicas tradicionais de animação por computador não funcionam corretamente (Welman, 1989). Uma animação bem realizada resulta mais da habilidade e da paciência do animador do que da técnica empregada para calcular os quadros intermediários.

Posteriormente, as técnicas de animação por computador para estruturas articuladas evoluíram não só para minimizar o esforço do animador, mas também para introduzir um maior nível de controle no movimento dessas estruturas. Uma série de técnicas cinemáticas foi introduzida para esse fim (Welman, 1989; Watt, 1992; Parent, 2001). Entre essas técnicas, a técnica de cinemática inversa oferece um controle da hierarquia de juntas através da especificação nos quadros-chaves de um efetor final (vide anexo A). Infelizmente, nesses métodos não se produz animação realística sem necessitar de um grande esforço do animador.

Recentemente, as animações por computador tendem a se tornar cada vez mais baseadas em métodos dinâmicos de simulação (Thalmann, 1996a). A grande vantagem de animar com esses métodos é que eles geram movimentos realísticos. Porém, exigem um maior nível de treinamento por parte do animador, já que ele tem de lidar com aspectos da física do movimento

dos corpos que pretende animar, isto é, massa, momento de inércia, forças de contato, restrições físicas que precisam ser respeitadas, etc. (Multon et al., 1999).

Os métodos de dinâmica direta (Girard et al., 1985; Isaacs & Cohen, 1987; Maciejewski, 1990; Baraff, 1995; Park & Fussell, 1997; Fujimoto, 1998) envolvem a aplicação direta por parte do animador de forças e torques no momento da animação dos objetos. Estender essa técnica para a simulação de estruturas articuladas é um desafio. Em geral, existe uma equação de movimento para cada grau de liberdade do esqueleto, levando a um sistema de equações que normalmente é resolvido por métodos numéricos com um custo computacional bastante considerável. Nos métodos de dinâmica inversa (Isaacs & Cohen, 1987; Badler et al., 1994; Ko, 1996; Multon et al., 1999; Nagano et al., 2000; Pitermann e Munhall, 2001; Sancho-Bru et al., 2001), a trajetória do movimento é primeiramente especificada pelo animador, e o sistema determina as forças e torques necessários à reprodução do movimento especificado. Os métodos de dinâmica inversa são em geral aplicados ao controle de robôs, onde os custos computacionais inerentes ao processo de solução são irrelevantes. Nos métodos de controle da dinâmica por controladores robóticos, as forças e torques aplicados às articulações das estruturas são realizados através da aplicação de atuadores localizados nas juntas, cujas forças variam em função de sinais recebidos de dispositivos controladores. Esses controladores são em geral modelados por um sistema de força-mola que controla a variação das forças e torques em cada instante do tempo da animação com base nos objetivos especificados pelo animador. Apesar dos controladores realizarem movimentos realísticos, o animador tem que especificar um tipo de controlador para cada movimento (Raibert & Hodgins, 1991; Kokkevis et al., 1996; Hodgins et al., 1995; Krabbes & Döschner, 1999), o que resulta em um processo bastante artesanal, exigindo várias tentativas para selecionar um controlador que melhor se adéqüe ao movimento desejado.

O método de otimização, denominado *Spacetime Constraints* introduzido no trabalho de Witkin e Kass (1988), oferece um controle de alto nível, como nos métodos de quadros-chaves e

cinemática inversa, com a geração automática de movimentos. Nesse método assume-se que uma descrição dos caminhos da animação seja feita pelo animador e constrói-se o "melhor" conjunto de forças e torque que move o corpo através desse caminho. O trabalho de Witkin e Kass demonstrou que essa técnica é capaz de sintetizar o movimento de estruturas articuladas simples através da descrição da trajetória e do tempo da animação. O uso das técnicas de otimização por *spacetime constraints* (Cohen, 1992; Liu et al., 1994; Auslander et al., 1995; Park et al., 1997; Brogan et al. 2002) tem-se mostrado bastante promissor na área da animação, assim como capaz de produzir movimentos complexos, coordenados e realísticos (Welman, 1989). Porém, resulta num sistema grande de equações que são resolvidos com um custo computacional elevado, impossibilitando à aplicação em sistemas interativos de tempo real. Vários sistemas (Gleicher, 1997; Lee et al., 1999; Popovi et al., 1999) usaram a técnica de *spacetime constraints* para transformar animações capturadas através da técnica de geração de movimento por dispositivos de captura (Multon et al., 1999; Madhavapeddy, 2004).

1.2 Objetivos

O objetivo desse trabalho é estudar e avaliar a técnica de *spacetime constraints* em relação ao nível de controle e esforço computacional na geração do processo de animação. Para essa finalidade, é proposto um sistema híbrido que integra computação simbólica e otimização numérica. O sistema é capaz criar animações dinâmicas e realísticas de estruturas articuladas através de uma interface gráfica amigável.

1.3 Estrutura da dissertação

A presente dissertação encontra-se organizada em seis capítulos. No Capítulo 2, faz-se uma análise do estado-da-arte das técnicas de animação de estruturas articuladas. No Capítulo 3, descreve-se a modelagem física e matemática necessária à técnica de *spacetime contraints*. No Capítulo 4, descreve-se o desenvolvimento do sistema de animação proposto. No Capítulo 5, mostram-se os resultados de algumas animações geradas pela técnica de *spacetime constraints*.

Finalmente, no Capítulo 6, tecem-se algumas conclusões acerca da técnica estudada e dos resultados analisados.

Capítulo 2
Técnicas de Animação de Figuras Articuladas

2.1 Introdução

Nesse capítulo, são discutidos os termos e técnicas que envolvem a animação de figuras articuladas. As técnicas são divididas em duas categorias: a cinemática, que especifica e estuda o movimento independente das forças inerentes ao movimento; e a dinâmica, que envolve a aplicação explícita das forças e torques variantes com o tempo sobre os objetos articulados.

2.2 Métodos Baseados em Cinemática

O primeiro conjunto de ferramentas desenvolvido para criar animação por computador foi baseado nos métodos cinemáticos (Multon et al., 1999). Nesses métodos, o movimento é controlado localmente através da especificação de coordenadas, ângulos, velocidades ou acelerações que devem ser fornecidos pelo animador. Nessa categoria, estão os métodos de Quadros-chaves, Captura de movimento, Morphing, Cinemática Direta e Cinemática Inversa os quais são descritos a seguir.

2.2.1 Quadros-chaves

A animação por quadros-chaves é uma técnica bastante popular onde a seqüência de cenas estáticas que compõem a animação é dividida em uma sucessão de cenas principais, denominadas quadros-chaves, e seqüências de quadros-intermediários que fazem a transição suave entre os quadros-chaves consecutivos. Cada seqüência de quadros-intermediários é determinada computacionalmente através da interpolação de dois quadros-chaves consecutivos (Figura 2.1).

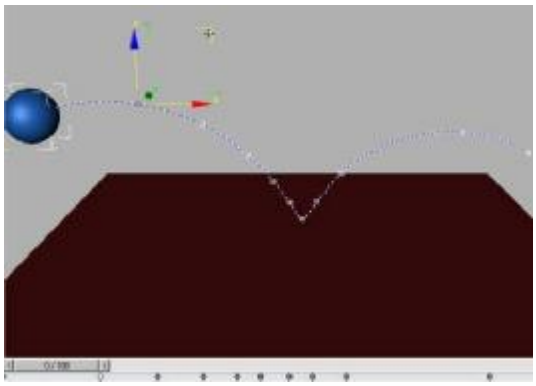


Figura 2.1 - Interpolação com quadros chaves (Discreet, 2004)

Essa técnica de animação por computador foi inspirada em um sistema de desenho animado desenvolvido pela Walt Disney (Figura 2.2), em que o animador desenhava vários quadros-chaves e em seguida tentava gerar quadros intermediários de forma que a seqüência ficasse realística. A desvantagem daquele sistema era sua grande dependência da habilidade dos desenhistas os quais chegavam à seqüência final por um processo de tentativa e erro, sendo constantemente forçados a reajustar os quadros-chaves até atingir o resultado desejável (Madhavapeddy, 2004).

Figura 2.2 - Animação tradicional (Disney, 1937)

A técnica de quadros-chaves, por ser de fácil implementação, passou a ser oferecida de forma generalizada nos softwares de animação (Figura 2.3). No entanto, essa técnica deixa a desejar quando se quer animar uma grande quantidade de objetos ou figuras articuladas, pois exige que o animador lide com um número excessivo de parâmetros em cada quadro da animação. Para minimizar esse problema, os softwares mais modernos utilizam interfaces gráficas que oferecem

um melhor controle dos parâmetros do movimento e dos caminhos da interpolação (Figura 2.1 e Figura 2.4).

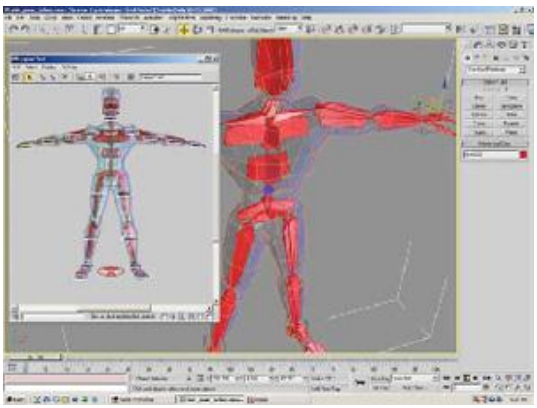


Figura 2.3 - Pacote de animação baseada em Quadros-chaves (Discreet, 2004)

Além de ser utilizada de forma isolada, a técnica de quadros-chaves é utilizada também em associação com outras técnicas da animação. Na técnica de morphing, por exemplo, que envolve a transformação de uma forma em outra, essas duas formas são definidas como quadros-chaves inicial e final; e a animação que mostra a evolução da transformação é obtida através da geração de quadros-intermediários por meio de um processo de interpolação adequado dos dois quadros-chaves. Nas animações baseadas em curvas *splines*, os caminhos da animação são definidos por *splines* e os quadros-chaves são gerados em pontos específicos ao longo do caminho.

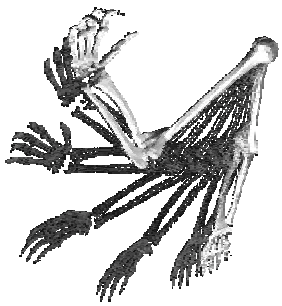


Figura 2.4 – Animação de figuras articuladas com quadros-chaves

2.2.2 Captura de movimento

Essa técnica consiste em capturar os movimentos de um ator real através de meios magnéticos ou ópticos e aplicá-los sobre o modelo computacional do personagem que se deseja

animar (Figura 2.5). Essa técnica tem sido muito utilizada pelas indústrias de filmes e jogos, principalmente, para animação de caracteres 3D. Ela requer o uso de equipamentos especiais caros que, em geral, são fornecidos por estúdios de animação especializados (Madhavapeddy, 2004).

Figura 2.5 - Sistema de captura de movimento (Meta Motion, 2004)

Para certas aplicações que envolvem situações de movimentação – andar, correr, saltar, lutar, dançar, etc. – possíveis de serem executadas por um ator e possíveis de serem capturadas, a técnica de captura de movimentos tem sido uma alternativa vantajosa em comparação com a técnica de animação por quadros-chaves, principalmente quando há necessidade de movimentos mais realistas (Multon et al., 1999). O tempo de produção também é um fator a considerar. Em alguns casos, com a técnica de captura de movimentos, é possível produzir seqüências em um dia que levariam semanas para serem concluídas com a técnica de quadros-chaves (Madhavapeddy, 2004).

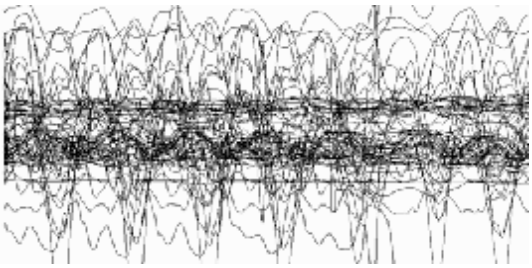


Figura 2.6 - Curvas do movimento humano (Popović, 1999)

Um dos problemas fundamentais da captura de movimentos é a falta de controle sobre os dados gravados (Figura 2.6), não deixando ao animador a possibilidade de manipular as configurações capturadas (Gleicher, 1999). Já que não é possível capturar todas as possibilidades de movimentos, técnicas foram desenvolvidas para parametrizar os movimentos obtidos por captura e efetivamente criar novos movimentos a partir deles (Rose et al., 1996; Bodenheimer et al., 1997; Rose et al. 1998; Tanco et al, 2000).

Os sistemas de captura de movimentos mais comuns estão agrupados em quatro categorias: os sistemas ópticos, os sistemas acústicos, os sistemas magnéticos e os mecânicos (Furniss 2004).

Os sistemas ópticos tornaram-se populares principalmente por três motivos: pela liberdade de movimentos oferecida aos atores devido à inexistência de cabos anexados aos atores, por permitir a captura de movimentos rápidos como, por exemplo, em esportes - devido à taxa alta de captura - e a área de captura é grande o bastante para permitir capturar variações entre movimentos rápidos e lentos como, por exemplo, a transição entre um caminhar e uma corrida (Mellor, 2004). Nesses sistemas, marcadores – pequenas esferas de material refletor – são posicionadas no corpo do ator e rastreadas por câmeras atreladas a um computador que processa as imagens geradas, capturando a posição dos marcadores (Figura 2.7). O sistema de captura necessita de, no mínimo, três câmeras para conseguir determinar as coordenadas de cada marcador (Madhavapeddy, 2004). Essas coordenadas são transferidas, em geral, a um sistema de cinemática inversa responsável pela animação do esqueleto articulado (Madhavapeddy, 2004).

Figura 2.7 – Sistema óptico (Mellor, 2004)

A principal limitação dos sistemas ópticos é a freqüente obstrução dos marcadores, gerando lacunas nos dados processados. A adição de câmeras ao sistema reduz as chances de ocorrer obstrução, mas acarreta uma sobrecarga em conseqüência do aumento da complexidade de rastreamento (Madhavapeddy, 2004). Outra limitação é que a intensidade da luz ambiente pode introduzir erros de captura, assim como, a possibilidade de ocorrer de interferência com outras fontes de luzes, nos dois casos é necessária uma calibração do sistema. (Furniss 2004)

Nos sistemas acústicos, um conjunto de emissores sonoros é fixado ao corpo do ator e receptores de áudio são colocados em pontos específicos do ambiente para rastrear a posição dos transmissores. No processo de rastreamento, cada emissor, seqüencialmente, gera um sinal sonoro que é captado pelos receptores. As distâncias do emissor a cada receptor são calculadas como o produto da velocidade do som pelo tempo decorrido entre o instante da emissão e o da recepção. Em seguida, as coordenadas do emissor são determinadas por triangulação (Silva, 2004).

Os sistemas acústicos não apresentam o problema de oclusão, comum nos sistemas ópticos. No entanto, os cabos de controle ligados aos emissores restringem significativamente os movimentos do ator. A velocidade do som e o número de emissores são fatores limitantes da área de captura (Silva, 2004).

Os sistemas magnéticos utilizam sensores fixados ao corpo do ator – geralmente nas juntas – para medirem o campo magnético de baixa-freqüência gerado por uma fonte transmissora (Figura 2.8). Os sensores e a fonte são conectados por cabo a uma unidade eletrônica de

controle que correlaciona suas localizações no campo. As unidades eletrônicas de controle são conectadas em rede a um computador que processa essas informações para obter posições e rotações no espaço tridimensional. Cinemática inversa é utilizada para calcular os ângulos de várias juntas e os ajustes exigidos pelo fato dos sensores serem posicionados de forma excêntrica com relação aos centros de pivotação das juntas (Madhavapeddy, 2004).

Figura 2.8 – Sistema magnético (Mellor, 2004)

Os sistemas magnéticos também apresentam seus próprios problemas. A área de captura é limitada pela capacidade da fonte transmissora. A filtragem de ruídos nos dados capturados é necessária, o que diminui a velocidade de processamento da animação, gerando atrasos em aplicações de tempo real. O reposicionamento dos sensores e a recalibragem do sistema são freqüentes, em virtude dos sensores saírem de suas posições originais de fixação durante as sessões de captura. Objetos metálicos podem interferir com a área de captura. E, nos casos em que os movimentos de mais de um ator estão sendo capturados simultaneamente, sensores de diferentes atores podem interferir uns com os outros, gerando resultados distorcidos (Silva, 2004).

Figura 2.9 – Sistema mecânico (Mellor, 2004)

Nos sistemas mecânicos, um conjunto de sensores metálicos (como um exoesqueleto) é acoplado às costas do ator (Figura 2.9). Quando o ator movimenta-se, o exoesqueleto acompanha o movimento, ativando cada sensor. Outros dispositivos bastante comuns como luvas de alguns vídeos games e braços mecânicos também podem ser considerados sistemas mecânicos (Furniss, 2004). Nesses dispositivos, não ocorrem interferências de intensidade de fontes luz e de campos magnéticos. Esse sistema apresenta algumas desvantagens: a necessidade de calibração frequentemente do equipamento e a limitação dos movimentos causada pelos equipamentos atrelados ao corpo do ator por essa razão, por exemplo, o salto é um movimento bastante difícil de reproduzir. (Furniss, 2004; Silva, 2004)

2.2.3 Morphing

Morphing consiste em animar a metamorfose de um objeto ou imagem inicial (Figura 2.10) em um outro objeto ou imagem final (Lazarus & Verroust, 1998). O morphing de objetos gráficos 3D inclui a interpolação da forma e dos atributos, como por exemplo, a cor, a textura etc (Figura 2.10). Em movimentos de esqueletos articulados (Figura 2.11), o método do morphing é usado para animar a malha do objeto (Lee & Magnenat-Thalmann, 2001), enquanto

outras técnicas podem ser usadas para controlar o posicionamento e orientação das juntas que realizam o movimento (Thalmann et al., 1996b; Allen et al., 2003).

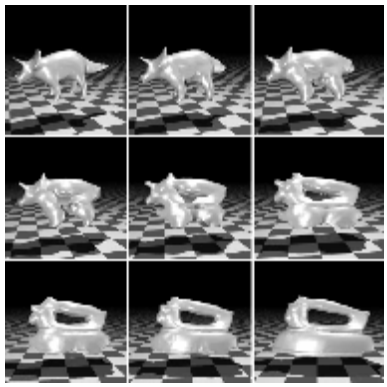


Figura 2.10 - Exemplo de animação com Morphing (Cohen-Or, 1998)

Atualmente, muitos algoritmos foram propostos para calcular a transformação entre dois aspectos do objeto (Lazarus & Verroust, 1998). A técnica é bastante utilizada em vários sistemas de animação comercial e desenho industrial, com aplicações tais como: a animação de animais por reconstrução de malha (Simmons et al., 2002) e animação facial (Blanz et al., 1999). Essa técnica também é utilizada em combinação com outras técnicas de animação como, por exemplo, com a captura de movimentos e com seqüências de vídeo (Ezzat et al. 2002; Chai et al., 2003).

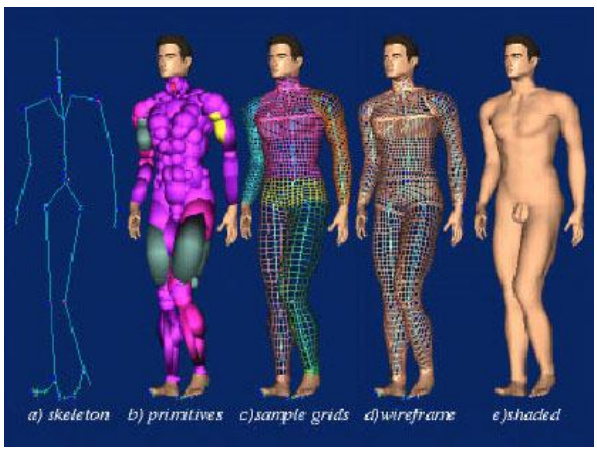


Figura 2.11 - Corpo humano em um sistema multicamadas (Thalmann et al., 1996b)

2.2.4 Cinemática Direta

A cinemática estuda a movimentação dos corpos independente das forças aplicadas que produzem o movimento. Ela respeita a posição, velocidade e aceleração, isto é, toda a geometria e propriedades relacionadas com o tempo do movimento.

Para figuras articuladas, o método de cinemática direta consiste em especificar diretamente o vetor do estado, isto é, todas as variáveis – ângulos e posições – necessárias para definir pose no espaço tridimensional em qualquer instante da animação (Watt, 1992). Essa especificação é normalmente combinada com a técnica de quadros-chaves. No entanto, é complicado encontrar quadros-chaves convenientes e técnicas de interpolação que respeitem as restrições cinemáticas do objeto articulado (Figura 2.12), como por exemplo, a restrição do pé do personagem com o chão. A interpolação para cada um dos graus de liberdade forma uma trajetória que é determinada pelos valores dos quadros-chaves e também pelo tipo de interpolação utilizada. A interpolação linear apesar de sua simplicidade pode gerar movimentos indesejáveis, como por exemplo, transições abruptas, nada realísticas.

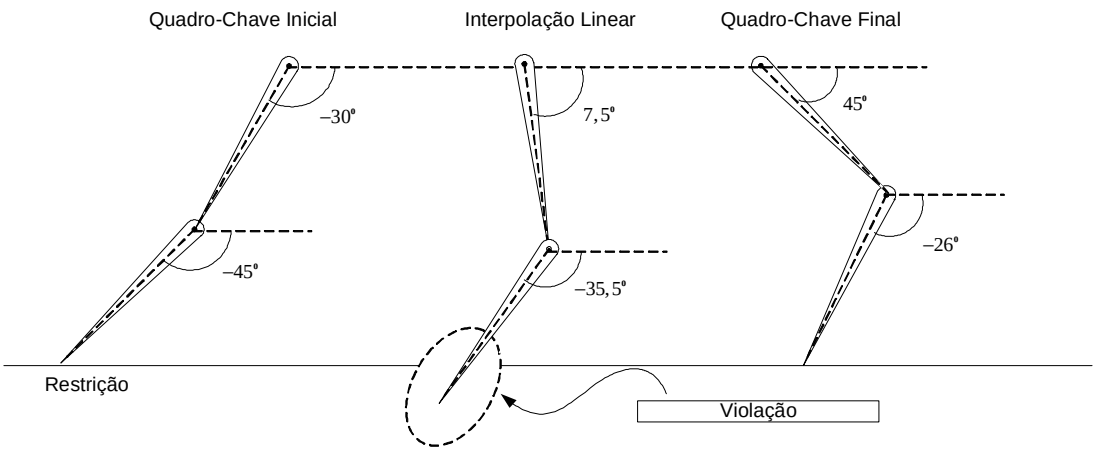


Figura 2.12 - Interpolação na cinemática direta

Apesar dessas dificuldades, a cinemática direta é uma ferramenta indispensável em qualquer pacote comercial de animação por computador, por permitir um controle mais refinado da animação, porém exigindo uma grande habilidade do animador para gerar seqüências realísticas de movimento (Boulic et al, 1992; Multon et al. 1999).

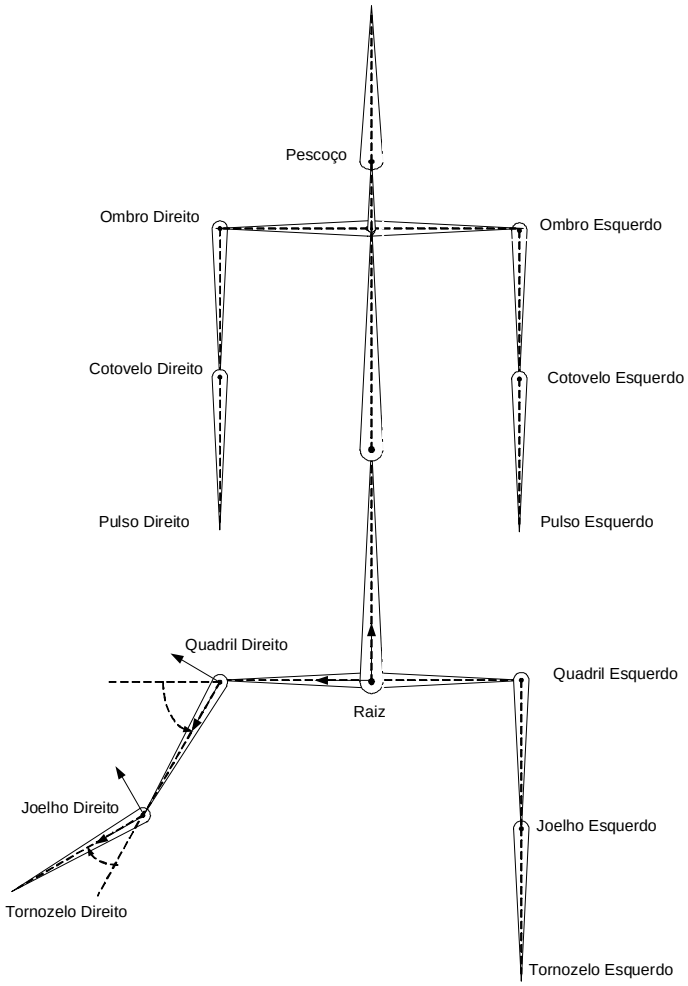


Figura 2.13 – Juntas de uma figura articulada

Na animação, por cinemática direta, da estrutura articulada mostrada na Figura 2.13, para cada quadro-chave o animador especifica a posição e orientação da raiz em um sistema global de coordenadas e em seguida a orientação de todas as juntas em coordenadas locais. Esses valores de posicionamento são suficientes para localizar e respeitar toda a geometria da figura. A pose do caractere em cada quadro-chave é estruturada de forma hierárquica, isto é, o posicionamento de cada uma das juntas é formada pela propagação das transformações ao longo de um caminho da hierarquia. Por exemplo, a posição do tornozelo direito é determinada pelo posicionamento e orientação da raiz, com as posições e orientações locais do quadril direito, joelho direito e tornozelo direito.

2.2.5 Cinemática Inversa

A técnica da cinemática inversa teve sua origem na robótica (Welman, 1989; Watt, 1992; Parent, 2001), onde os movimentos das articulações e braços são calculados a partir da trajetória de um efetor final (Figura 2.14).

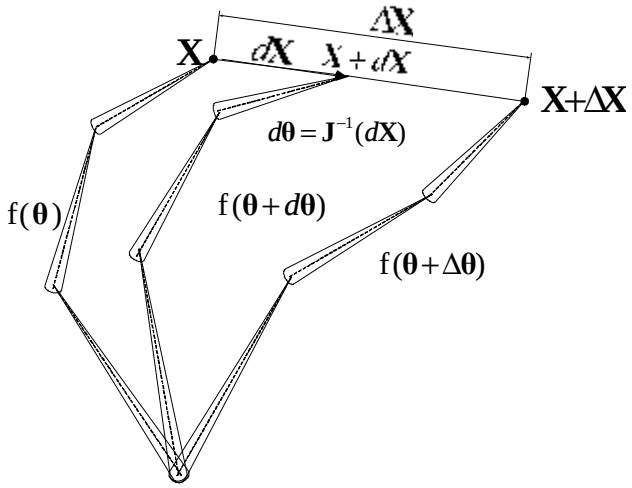


Figura 2.14 - Cinemática inversa de uma figura articulada

Nessa técnica, também conhecida como movimentação direcionada ao objetivo, o animador define a posição do efetor final em cada quadro da animação e o sistema determina as posições e orientações de todas as juntas da hierarquia que resultam na posição especificada do efetor final. Assim, seja $\mathbf{X}$ o vetor posição do efetor final, o vetor $\boldsymbol{\theta}$ das orientações locais de cada uma das juntas é determinado pela equação.

$$\boldsymbol{\theta} = \mathrm{f}^{-1}(\mathbf{X}), \tag{2.1}$$

onde f é a função que calcula o vetor posição do efetor final a partir dos graus de liberdade da estrutura articulada.

Na animação, por cinemática inversa, da figura 2.14, o animador especifica o vetor deslocamento $\Delta\mathbf{X}$ do efetor final em coordenadas globais e o sistema calcula a variação $\Delta\boldsymbol{\theta}$ das orientações das juntas internas. A relação entre $\Delta\mathbf{X}$ e $\Delta\boldsymbol{\theta}$ é determinada pela equação.

$$\Delta\mathbf{X} = \mathbf{J}\Delta\boldsymbol{\theta}, \tag{2.2}$$

Onde o $\mathbf{J}$ é a matriz jacobiana do sistema (Watt, 1992). A matriz $\mathbf{J}$ não pode ser diretamente invertida ao calcular a solução, devido ao fato dos vetores $\mathbf{X}$ e $\boldsymbol{\theta}$ terem dimensões distintas. A solução mais comum para esse problema é representada pela equação.

$$\Delta\boldsymbol{\theta} = \mathbf{J}^{+}\Delta\mathbf{X} + a\,(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\Delta\mathbf{z}, \tag{2.3}$$

Onde o $\mathbf{J}^{+}$ é a matriz pseudo-inversa do jacobiano $\mathbf{J}$, o a é uma constante de penalidade, o $\mathbf{I}$ é a matriz identidade e o $\Delta\mathbf{z}$ é uma restrição a ser minimizada chamada tarefa secundária. Essa tarefa secundária é imposta no espaço nulo de $\Delta\mathbf{X}$. Em geral, $\Delta\mathbf{z}$ é usado para considerar os limites angulares das juntas ou para minimizar algum critério de energia.

A cinemática inversa fornece um nível controle mais elevado sobre a hierarquia das juntas do que o controle por cinemática direta (Welman, 1989), em contrapartida, há um gasto computacional mais elevado.

Infelizmente, os métodos cinemáticos não produzem movimentos convincentes sem um esforço considerável por parte do animador. Os métodos cinemáticos diretos e inversos não produzem movimentos que mantém a integridade dinâmica das leis físicas.

2.3 Métodos Baseados em Dinâmica

Enquanto na cinemática usam-se as informações relativas à orientação, à posição, à velocidade e à aceleração de cada articulação de uma figura articulada, a dinâmica introduz o uso de forças e torques, e massas e momentos de inércia, criando, assim, movimentos realísticos (Park & Fussell, 1997). Os quadros da animação são criados resolvendo as equações da dinâmica. Um exemplo de um sistema que usa dinâmica é o Jack, que foi desenvolvido na universidade da Pensilvânia, para animação de figuras articuladas. Esse sistema incorpora dinâmica direta, cinemática inversa em tempo real para ajuste de posições e foi utilizado na animação de uma figura articulada humana com 88 juntas (Phillips et al., 1988).

Nos métodos dinâmicos o movimento é obtido resolvendo as equações diferenciais da dinâmica e é controlado globalmente, respeitando as integridades físicas do sistema. O animador fornece informações como massa e forças e controlada a animação, impondo um conjunto de restrições ao sistema.

As animações baseadas em dinâmica são classificadas em dois tipos: animações por dinâmica direta e animações por dinâmica inversa que são descritas a seguir.

2.3.1 Dinâmica Direta

A animação por dinâmica direta é um método de simulação onde as equações diferenciais que definem o movimento de um sistema são resolvidas numericamente (Baraff, 1995). Algumas forças como gravidade e colisão entre objetos podem ser tratadas automaticamente pelo sistema. Em estruturas articuladas, a equação de movimento é uma equação diferencial acoplada e não linear envolvendo os n graus de liberdade que definem a configuração da estrutura. Essa equação pode ser obtida por formulações de Euler-Lagrange e D'Alambert, e tem a forma (Maciejewski, 1990; Fujimoto, 1998)

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) + \mathbf{J}(\boldsymbol{\theta})^T \mathbf{f}, \tag{2.4}$$

onde o vetor $\boldsymbol{\theta}$ representa os deslocamentos generalizados das n juntas da estrutura (ver Figura 2.15), o vetor $\dot{\boldsymbol{\theta}}$ representa as velocidades generalizadas, o vetor $\ddot{\boldsymbol{\theta}}$ representa as acelerações generalizadas, $\boldsymbol{\tau}$ é um vetor de torques aplicados às juntas, $\mathbf{H}(\boldsymbol{\theta})$ é a matriz das inércias, $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ é o vetor dos torques centrífugos e de coriolis, $\mathbf{G}(\boldsymbol{\theta})$ representa os torques aplicados à estrutura devido à ação da gravidade, $\mathbf{J}^T(\boldsymbol{\theta})$ é a transposta da matriz jacobiana que relaciona os graus de liberdade descritos no sistema de coordenadas cartesianas ($\mathbf{X}$) e os graus de liberdade $\boldsymbol{\theta}$ e $\mathbf{f}$ representa o vetor das forças externas generalizadas.

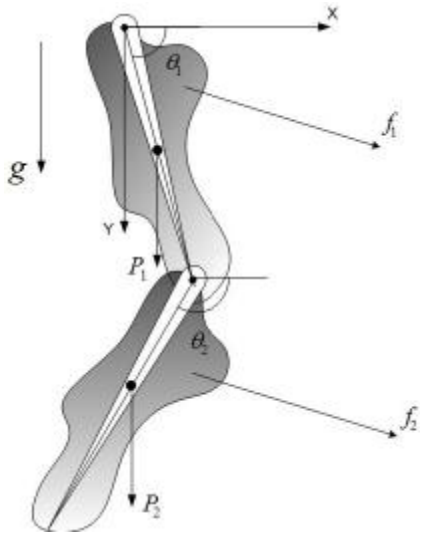


Figura 2.15 – Dinâmica num corpo articulado

O controle sobre a animação necessita da especificação das funções de força e torque aplicados aos segmentos da estrutura, o que é extremamente difícil e não natural para o animador. No entanto, foram utilizadas em algumas aplicações específicas (Girard et al., 1985; Isaacs & Cohen, 1987; Hahn, 1988; Park & Fussell, 1997).

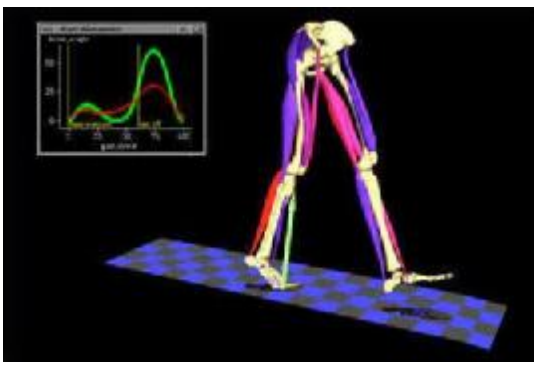


Figura 2.16 - Software para analise dinâmica com aplicação na medicina (MusculoGraphics, 2002)

Alguns sistemas comerciais também usam a dinâmica direta para obter animação realística, como, por exemplo, o sistema para análise dos músculos humanos (Figura 2.16) e o sistema para animação de soldados (Figura 2.17).

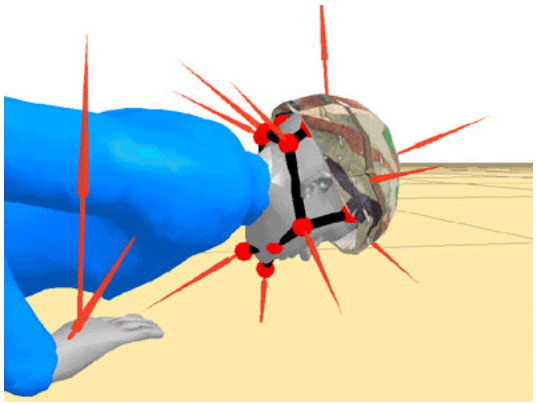


Figura 2.17 - Sistema de simulação de soldados (Digital Biomechanics, 2004)

2.3.2 Dinâmica Inversa

Na animação por dinâmica inversa, por sua vez, a trajetória do movimento é primeiramente especificada pelo animador, e o sistema determina as forças e torques necessários à reprodução do movimento especificado (Isaacs & Cohen, 1987; Ko et al., 1996; Multon et al., 1999) e posteriormente o sistema reproduz o movimento através da simulação com dinâmica direta (Figura 2.18). Devido à complexidade dos métodos numéricos para se calcular a solução, dificilmente esse tipo de método pode ser utilizado em sistemas de animação em tempo real. Com o aumento da capacidade de processamento dos computadores, o uso das técnicas dinâmicas tem-se desenvolvido em diversas áreas, como a biomecânica e a engenharia mecânica (Isaacs & Cohen, 1987; Ko, 1994; Nagano et al., 2000; Pitermann e Munhall, 2001; Sancho-Bru et al., 2001), assim como, na indústria de jogos e entretenimento (Laszlo et al., 2000).

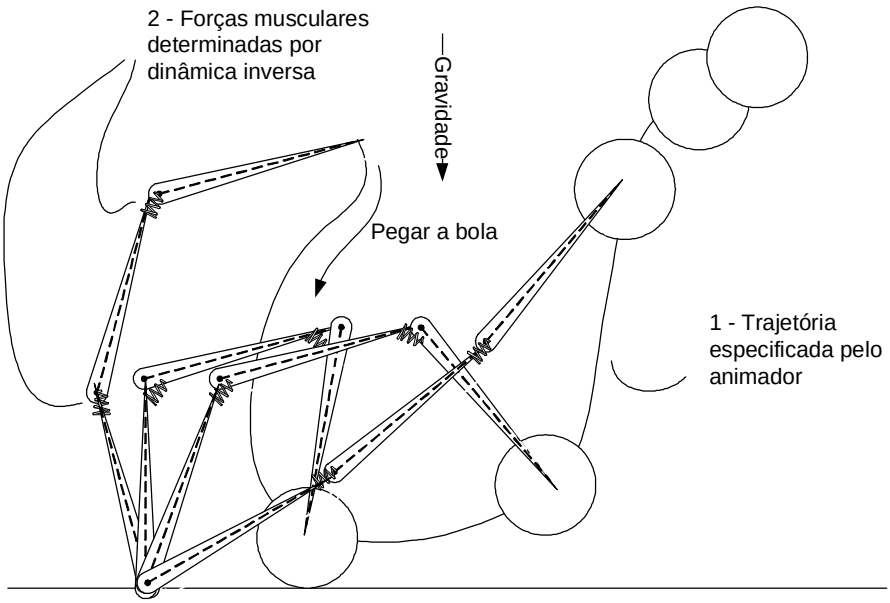


Figura 2.18 - Movimento usando a dinâmica inversa

2.3.3 Controladores

A técnica de animação com uso de controladores consiste em determinar o movimento de estruturas articuladas através da aplicação de atuadores localizados nas juntas, cujas forças variam em função de sinais recebidos de dispositivos controladores (Hodgins et al., 1998). Em comparação com um ser humano, os atuadores seriam os músculos responsáveis pela movimentação de partes do esqueleto e os controladores seriam regiões específicas do cérebro responsáveis pela emissão de impulsos nervosos capazes de contrair ou distender os músculos que atuam nessas partes do esqueleto. O realismo da animação é obtido através da simulação por dinâmica direta.

Os controladores são responsáveis por manter o ciclo da locomoção da figura, controlar a velocidade e as direções do movimento, estabilizar a postura e regular o comportamento das juntas. O uso de controladores é bastante comum na robótica (Krabbes & Döschner, 1999).

Existem duas classes de controladores para a dinâmica, os controladores de laço aberto e os controladores de laço fechado ou *feedback* (Kokkevis et al., 1996). Os controladores de laço aberto geram resposta ao sistema baseando-se apenas no estado desejado do sistema, já os

controladores de *feedback*, consideram também o estado atual do sistema. O mais simples e mais usado controlador de *feedback* (Figura 2.19) é governado por equações derivadas proporcionais (PD)

$$\boldsymbol{\tau} = -k_p(\boldsymbol{\theta} - \boldsymbol{\theta}_d) - k_v(\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{\theta}}_d), \tag{2.5}$$

onde k_p é o ganho proporcional, k_v é o ganho de derivada, $\boldsymbol{\theta}$ é vetor de deslocamento na direção dos graus de liberdade, $\dot{\boldsymbol{\theta}}$ é o vetor velocidade correspondente e $\boldsymbol{\theta}_d$ e $\dot{\boldsymbol{\theta}}_d$ representam os vetores deslocamento e velocidade alvos do sistema.

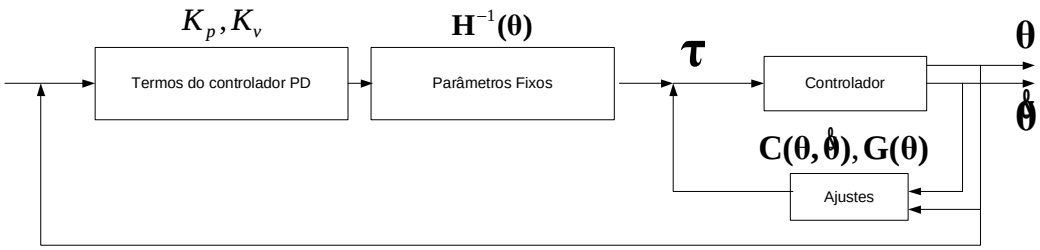


Figura 2.19 - Exemplo de um sistema controlado por *feedback*

Alguns sistemas (Raibert & Hodgins, 1991; Hodgins et. al, 1998) usam os controladores para movimentos cíclicos. O ciclo do movimento é expresso por uma máquina de estados finitos, denominado por grafo de controle de pose, onde os estados estão associados à pose do personagem. A transição entre dois estados é determinada através dos controladores PD.

Infelizmente, nos sistemas de animação conduzidos por controladores, não existe a possibilidade de adaptação, isto é, um controlador usado para realizar um salto, não pode ser usado para movimentos mais simples como uma caminhada. (Kokkevis et al., 1996).

2.3.4 Controle por otimização

A idéia básica dessa técnica de animação, também conhecida por *spacetime constraints* (Witkin & Kass, 1988), é determinar as diversas configurações da estrutura articulada ao longo do tempo de animação através da resolução de um problema de otimização. O espaço de busca nesse problema é composto por um conjunto de variáveis definidas pela discretização em todo

intervalo de tempo da animação das variáveis de posição, velocidade e força, denominada de incertezas (Figura 2.20). Park e Fussell (1997) definem o controle por otimização como um método de resolução das equações da dinâmica através do uso de um otimizador não-linear. Usando-se a notação geral, pode-se construir o problema da seguinte maneira:

Dado $\mathbf{x}$ minimize $E(\mathbf{x})$

$$\min_{\mathbf{x}} E(\mathbf{x}), \textit{sujeito a} \begin{cases} \mathbf{C}_1(\mathbf{x}) = 0 \\ \mathbf{C}_2(\mathbf{x}) \leq 0 \end{cases} \tag{2.6}$$

onde $\mathbf{x}$ representa um ponto no espaço de busca, $E(\mathbf{x})$ é a função objetivo que teve ser minimizada, $\mathbf{C}_1(\mathbf{x})$ é o conjunto de restrições de igualdade e $\mathbf{C}_2(\mathbf{x})$ o conjunto de restrições de desigualdade.

O conjunto de restrições pode ser classificado da seguinte forma:

- restrições nas posições e velocidades iniciais, finais ou intermediárias;
- restrições que limitam as forças musculares ou de contato com o chão;
- restrições das leis físicas entre todos os intervalos consecutivos da discretização.

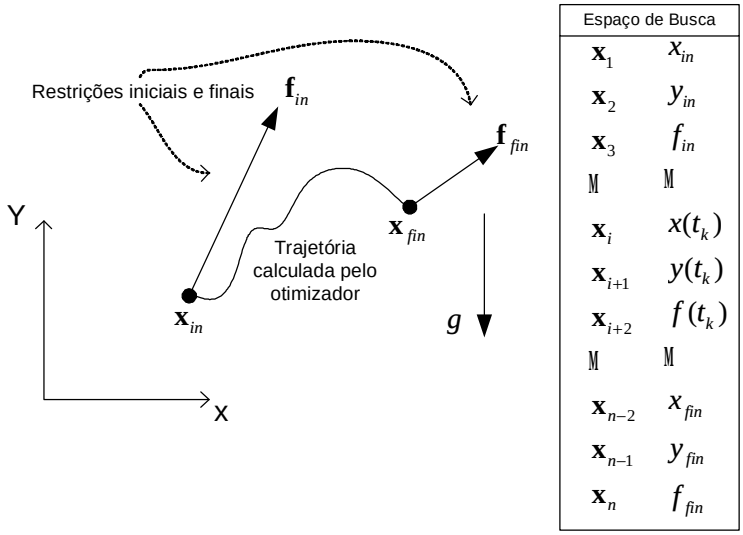


Figura 2.20 – Espaço de busca de uma partícula

Um processo típico de resolução para problemas de otimização não-linear com restrições é denominado de Programação seqüencial quadrática (Fletcher, 1987). Esse processo pode ser aplicado iterativamente até que o animador esteja satisfeito com o resultado.

Embora esse método ofereça um bom controle global da animação, sofre de algumas limitações: 1) a complexidade computacional do problema e o grande número de incertezas limitam o tamanho da animação; 2) não é garantido que o processo numérico irá convergir para uma solução aceitável; e 3) devido às restrições terem que ser definidas anteriormente na modelagem do problema, as colisões e contatos entre objetos não são previstos nessas restrições.

Na literatura, encontram-se alguns aperfeiçoamentos dessa técnica. Em (Liu et al., 1994), os graus de liberdade são expressos através de uma representação hierárquica em *wavelets*, que permite uma redução bastante significativa dos custos computacionais do sistema. Ngo e Marks (1993) propõem o uso de algoritmos genéticos que são capazes de gerar múltiplas trajetórias para o processo de otimização.

2.4 Comparação entre os métodos de animação

Do ponto de vista de um animador é importante que as técnicas de animação lhe ofereçam ao mesmo tempo um bom nível de controle e possibilidades de automação, bem como a capacidade de gerar movimentos realistas. As animações por cinemática são orientadas apenas para o controle do movimento, mas são deficientes tanto nos aspectos de geração de movimentos realísticos quanto nos aspectos de automação de movimentos. As técnicas de animação dinâmicas, por sua vez, oferecem um maior nível de automação de movimentos e realismo, porém, deixam o animador com menor controle sobre a animação (Figura 2.21).

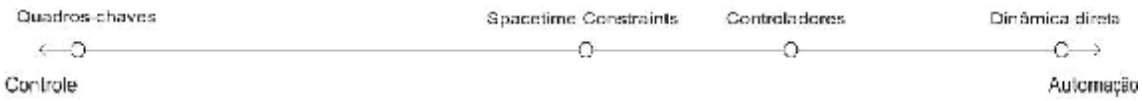


Figura 2.21 - Balanceamento entre as técnicas de animação.

Na Tabela 2.1, são apresentadas comparações sobre os métodos de dinâmica inversa, *spacetime constraints* e controladores, indicando vantagens, desvantagens e as ferramentas necessárias para sua utilização.

Tabela 2.1 - Comparação entre os métodos de controle por dinâmica.

	Controladores	*Spacetime Constraints*	Dinâmica inversa
Vantagens	O controlador considera iterações com o ambiente de simulação.	Controle global do movimento.	Permite controlar a trajetória do movimento.
Desvantagens	Dificuldade na construção dos controladores. Um controlador específico para cada tipo de movimento.	A necessidade de se criar um movimento inicial para entrar no otimizador. Elevado custo computacional.	Não existe planejamento global do movimento. Não pode ser aplicado a movimentos violentos rápidos.
Ferramentas necessárias	Simulação por dinâmica direta. Controladores robóticos.	Programação não-linear.	Métodos numéricos.

2.5 Considerações Finais

Todas as animações descritas têm suas vantagens e desvantagens. Não existe uma abordagem que possa fazer tudo, muito embora ela possa sobrepor em alguns aspectos. Mistura de técnicas tem sido usada mais freqüentemente em sistemas complexos, que dependem de vários fatores

como tempo, custo e viabilidade. Existem outros diferentes tipos de animação para figuras articuladas como animação comportamental, porém estaremos mais interessados em animação que respeitem as informações físicas e geométricas do sistema. Com o *Spacetime Constraints*, existe um melhor balanceamento entre controle e automatização (Liu & Cohen, 1995), porém há um passo com complexidade computacional bastante elevado devido à natureza não-linear do processo de otimização, dificultando a construção de sistemas de tempo real.

Capítulo 3
A Técnica de Controle de Movimento de Estruturas Articuladas por otimização não-linear

3.1 Introdução

Nesse capítulo discute-se a técnica de controle de movimento de figuras articuladas por otimização não-linear, apresentando-se a modelagem física e matemática do problema. Essa técnica oferece um bom nível de balanceamento entre o controle e a automação permitindo a criação de movimentos realísticos, além de, possibilitar adaptação das animações geradas por captura de movimento (Gleicher. 1997; Popović, 1999; Popović & Witkin, 1999). Ela pode ser aplicada, também, em outras áreas como a música (Rhoads, 2002) e a biomecânica (Chang et al., 2001; Brogan et al., 2002).

Figura 3.1 - Animação Luxo Jr. (Pixar 1986)

A exposição no restante desse capítulo é pautada pelo problema de animação da estrutura articulada mostrada na Figura 3.1. Na Seção 3.2, discute-se como, a partir da descrição de uma animação, constrói-se um modelo matemático para solução desse problema que recai em um problema de otimização não-linear com restrições. Na Seção 3.3, mostram-se os ingredientes necessários à formulação e à solução do problema de otimização. Finalmente, na Seção 3.4, tecem-se algumas considerações finais a cerca dessa técnica.

3.2 Descrição da Técnica

Considere-se o problema de animação ilustrado na Figura 3.2, onde a estrutura articulada mostrada deve efetuar um salto a partir de uma configuração inicial e atingir uma configuração final. Essas configurações ou poses são pré-estabelecidas pelo animador que se utiliza de um sistema de coordenadas adequado, denominado sistema de referência global. A trajetória do salto e as configurações definidas pela estrutura articulada ao longo dessa trajetória são determinadas pela técnica de animação adotada.

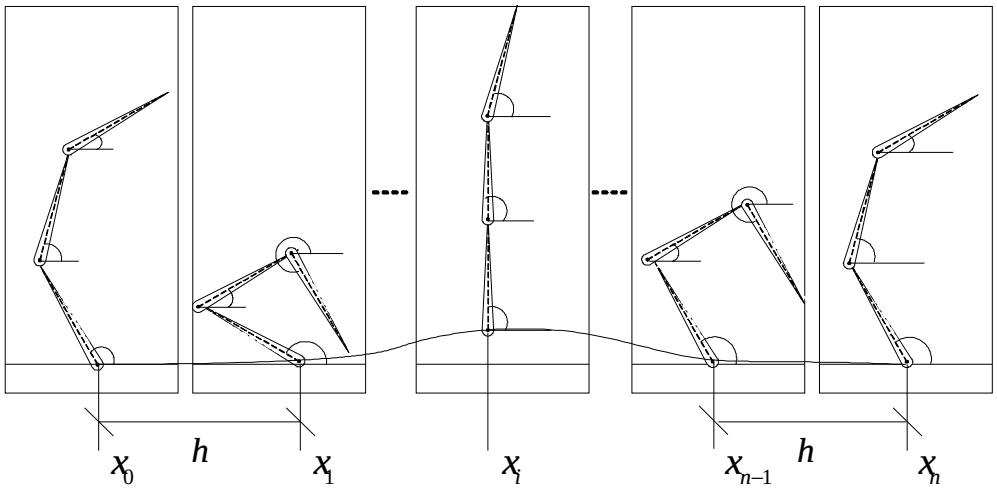


Figura 3.2 - Movimento da estrutura

Na técnica de animação por controle de otimização, o animador define a duração total da movimentação entre as configurações inicial e final, e o número de poses intermediárias que pretende utilizar. Esse número de poses serve como guia para a partição do tempo total da animação em intervalos discretos de tempo. A pose da estrutura articulada em cada ponto da partição é definida especificando-se os valores dos graus de liberdade da estrutura (Figura 3.3). Os valores de posicionamento nos pontos intermediários da partição não são especificados diretamente pelo animador. Para a determinação desses valores, é necessário que o animador especifique alguns objetivos que devem regular o processo da animação. Um objetivo seria, por exemplo, o menor desgaste muscular possível durante o salto. Um outro objetivo possível seria o de que o salto se aproximasse ao máximo de uma trajetória pré-especificada pelo animador.

As leis da física já são implicitamente consideradas por essa técnica. O problema de otimização pode ser descrito usando uma notação convencional (Equação 2.6). Essa notação serve para descrever a matemática do problema: o espaço das soluções formado pelo conjunto das variáveis para os graus de liberdade e os atuadores em todos os instantes do particionamento, um objetivo a ser minimizado e o conjunto das restrições.

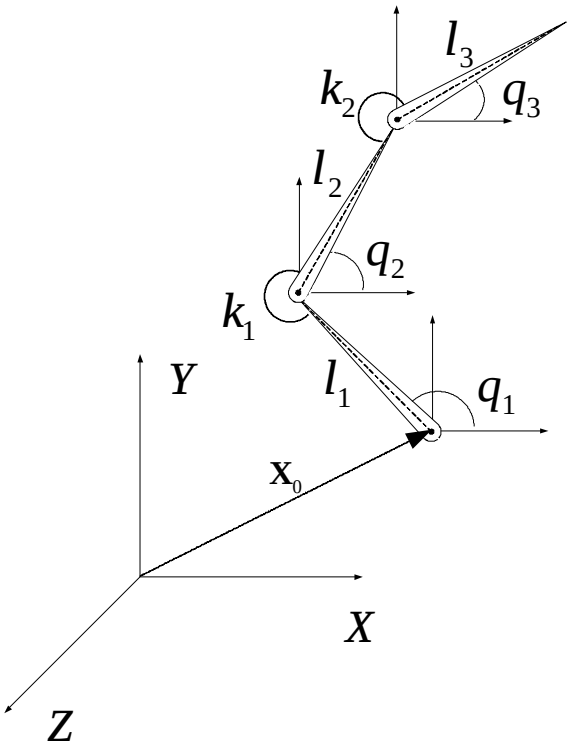


Figura 3.3 - Figura articulada usada na modelagem

Usa-se a técnica de otimização com restrições para minimizar uma função no espaço de todos os possíveis movimentos da figura articulada. A técnica acha o mínimo da função objetivo dentro do espaço de soluções delimitado pelas restrições. Um algoritmo parte de um ponto inicial e procura outros pontos no espaço de busca de forma que a função objetivo retorne valores mais baixos. Se todos os pontos nas direções praticáveis retornam valores mais altos para a função objetivo então esse ponto é o mínimo local no espaço de busca.

Em síntese, para que o problema de animação possa ser modelado para fins de solução pela técnica de controle de otimização, são necessários os seguintes passos: 1) construção do modelo físico-matemático da estrutura articulada e 2) definição do modelo matemático de otimização.

3.2.1 Modelagem físico-matemática da estrutura articulada

Topologia

A topologia da estrutura articulada é definida especificando-se: o número e o tipo de articulação, a geometria de cada uma das ligações, as conectividades entre as articulações, a localização e os tipos de atuadores. A estrutura articulada para o problema de otimização pode ser descrita observando a Figura 3.3. Essa estrutura é dotada de três corpos rígidos articulados e os corpos são orientados usando sistemas de coordenadas locais que podem ser transformados para um único sistema de referência global. A estrutura possui atuadores modelados por molas angulares de rigidez variável *k* acopladas a algumas dessas juntas (Tabela 3.1).

Tabela 3.1 – Topologia e propriedade físicas

Descrição	Exemplo na estrutura
Número e tipo de articulações	Três articulações do tipo uniaxial em relação ao eixo Z local
Geometria de cada uma das ligações	Corpos cilíndricos com tamanho l_1, l_2 e l_3
Conectividades entre as articulações	Corpo 2 atrelado ao corpo 1, na posição (l_1, q_1) local Corpo 3 atrelado ao corpo 2, na posição (l_2, q_2) local
Localização e tipos de atuadores	Atuador angular com rigidez variável k_1 atuante entre os corpos 1 e 2 Atuador angular com rigidez variável k_2 atuante entre os corpos 2 e 3
Massas	m_1, m_2 e m_3 Juntas e molas consideradas sem massa
Momentos de inércia do corpo i em relação ao eixo de rotação passando pela junta i	$m_i\left(\frac{L_i^2}{3}+\frac{R_i^2}{4}\right)$ Para cada um dos três corpos cilíndricos i (Anexo B)
Centros de massa	$\left(\frac{l_i}{2}, q_i\right)$ em relação às coordenadas locais de cada corpo i

Propriedades Físicas

As propriedades físicas da estrutura são definidas por: massa, momento de inércia e centro de massa de cada uma das articulações. (vide Anexo B)

3.2.2 Método de otimização

Descritas a topologia e as informações físicas do corpo articulado, pode-se construir a matemática da otimização ilustrada através da notação usual de um problema de otimização. Considere-se $\mathbf{x}$ o conjunto de variáveis composto por todos os graus de liberdade e todos os atuadores em todos os pontos do particionamento do intervalo de tempo da animação. Assim, o problema de otimização da figura articulada pode ser expresso como,

Encontrar $\mathbf{x}$ tal que

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad Sujeito\,a \begin{cases} \mathbf{C}_{pi}(\mathbf{x}) = 0 \\ \mathbf{C}_{pd}(\mathbf{x}) \leq 0 \\ \mathbf{C}_{mi}(\mathbf{x}) = 0 \\ \mathbf{C}_{md}(\mathbf{x}) \leq 0 \\ \mathbf{C}_{ni}(\mathbf{x}) = 0 \\ \mathbf{C}_{nd}(\mathbf{x}) \leq 0 \end{cases} \tag{3.1}$$

onde $\mathbf{x}$ é um vetor que representa um ponto no espaço do espaço de busca, $f(\mathbf{x})$ é a função objetivo, C_{pi} e C_{pd} são as restrições de pose, C_{mi} e C_{md} são as restrições mecânicas e C_{ni} e C_{nd} são as restrições newtonianas.

No problema do controle por otimização das estruturas articuladas, devido à natureza altamente não-linear das restrições (Witkin & Kass, 1988), existe a necessidade de se utilizar algoritmos de otimização não-linear. Alguns algoritmos capazes de calcular a solução para esse tipo de problema são: programação seqüencial quadrática (SQP - ver Anexo E); alguns membros da família de métodos Quasi-Newton como, por exemplo, Broyden-Fletcher-Goldfarb-Shanno (BFGS); e otimização global por *Simulated Annealing*. Em Witkin e Kass (1988), propõe-se a construção de um algoritmo simples por SQP; em Liu (Liu et al., 1994), calcula-se o problema de *spacetime constraints* com otimizador BFGS; e em Popović (1999), propõe-se um modelo híbrido entre as duas técnicas. Um bom otimizador e uma função objetivo

bem definida são determinantes na qualidade do resultado do problema. Porém, não é possível determinar o melhor algoritmo e a melhor função objetivo (Fletcher, 1987; Goldsmith, 1994).

Espaço das soluções

A primeira informação a se definir na modelagem matemática da otimização é o conjunto de variáveis que irão compor o espaço da solução. São consideradas informações variantes capazes de mudar o estado da solução na busca de atingir o objetivo especificado. Na estrutura articulada considerada, o espaço de soluções é o conjunto de variáveis da topologia em todos os pontos da discretização do tempo: os posicionamentos da base em coordenadas globais, as orientações de cada corpo rígido em coordenadas generalizadas e os valores de rigidez dos atuadores. O espaço das soluções é definido por

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_{p_1} \\ \mathbf{x}_{p_2} \\ \mathbb{M} \\ \mathbf{x}_{p_i} \\ \mathbb{M} \\ \mathbf{x}_{p_{n-1}} \\ \mathbf{x}_{p_n} \end{pmatrix} \qquad \text{onde, } \mathbf{x}_{p_i} = \begin{pmatrix} x(i) \\ y(i) \\ q_1(i) \\ q_2(i) \\ q_3(i) \\ k_1(i) \\ k_2(i) \end{pmatrix} \tag{3.2}$$

representa a pose da estrutura articulada no ponto t_i da partição.

Esse conjunto de variáveis, também chamado de conjunto de incertezas, está relacionado com a natureza física e topológica da estrutura, assim como, com a natureza da solução adotada para realizar a movimentação. O problema de otimização cresce à medida que se utiliza um particionamento mais refinado do intervalo de tempo. O tamanho h dessa discretização, às vezes, é tomado como constante, principalmente quando o calculo das derivadas com relação ao tempo é feito utilizando o método de diferenças finitas (Figura 3.4).

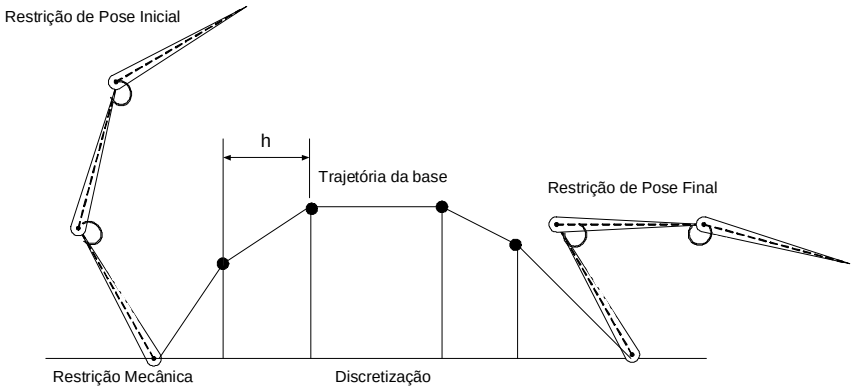


Figura 3.4 – Discretização e restrições

Para percorrer o espaço de busca, é necessário um ponto $\mathbf{x}$ inicial. Alguns otimizadores aceitam valores para o $\mathbf{x}$ inicial fora do espaço restringido pelas restrições. Esses otimizadores usam um método adicional para calcular um $\mathbf{x}$ praticável, projetando o ponto impraticável no espaço das soluções delimitado pelas restrições.

Uma solução explorada na literatura (Cohen, 1992; Liu et al., 1994) é a possibilidade de controlar cada grau de liberdade $\mathbf{x}$ através de coeficientes de funções bases (*hermite*, *B-splines*, *wavelets*). Essas curvas passaram a controlar cada um dos graus de liberdade no tempo. O vetor $\mathbf{x}$ é composto por coeficientes de curvas e pelos coeficientes de rigidez discretos. Internamente, calculam-se as informações pertinentes às restrições com base nas equações da curva do ponto discreto. A vantagem óbvia é a redução no tamanho do problema de otimização.

3.2.2.1 Função objetivo

A função objetivo $f(\mathbf{x})$ é uma função das variáveis do espaço de busca que, em geral, está associada a alguma medida de desempenho da estrutura articulada durante a animação. Essa medida, normalmente, representa um critério energético, e é calculada em cada passo da otimização para os valores correntes de $\mathbf{x}$ (Figura 3.5).

No trabalho de Goldsmith (1994) é feito a comparação entre várias funções objetiva na solução de problemas de animação tanto de corpos simples, quanto de estruturas articuladas. Algumas das funções comparadas naquele trabalho são a aceleração angular da estrutura, a

aceleração tangencial e a energia cinética total dos corpos. A forma geral da função objetivo pode ser expressa por

$$f(\mathbf{x}) = \int_{t_{in}}^{t_{fim}} f(t)dt \tag{3.3}$$

onde $f(t)$ é alguma função de medida de desempenho associado à estrutura articulada no instante de tempo t (por exemplo, a energia cinética).

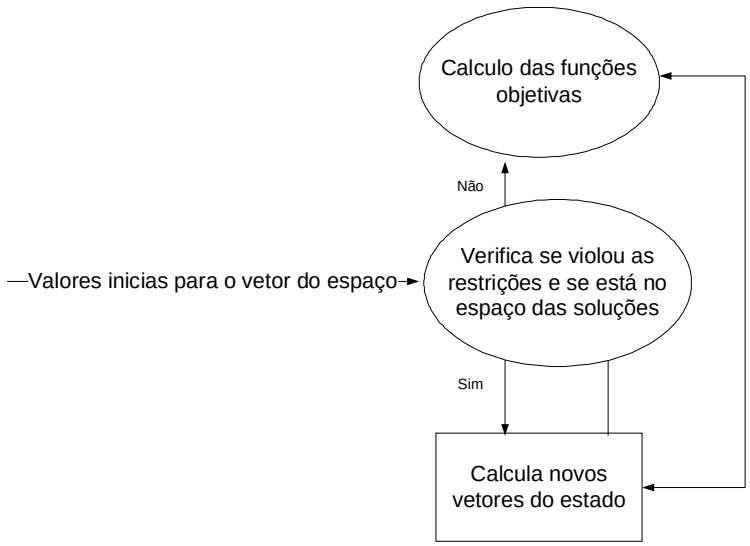


Figura 3.5 - Decisões na otimização

No trabalho de Witkin e Kass (1988), para determinar a função objetivo $f(\mathbf{x})$ utilizam-se funções $f(t)$ que representam as potências dos atuadores angulares dadas por

$$f(t) = \sum_{j=1}^{na} \mathbf{F}_{f_j} . \dot{f}_j \tag{3.4}$$

onde na é o número de atuadores, $\dot{f}_j$ é a velocidade angular relativa dos membros conectados à junta j e F_{f_j} é a força angular. F_{f_j} é dada por

$$F_{f_j} = k_j(f_j - r_j)$$
$$e \tag{3.5}$$
$$f_j = \boldsymbol{\theta}_{j+1} - \boldsymbol{\theta}_j$$

onde k_j é a rigidez do atuador j, r_j é o ângulo de repouso do atuador, em geral, adotado como p.

A Figura 3.6 ilustra um possível exemplo de função objetivo para a animação da estrutura articulada definida na Figura 3.3, em que se quer minimizar a diferença entre a trajetória real da base $\mathbf{x}_0$ da estrutura e uma trajetória pré-especificada pelo animador, expressa pela curva paramétrica $c(t)$. Assim, essa função objetivo é calculada por

$$f(\mathbf{x}) = \frac{\int_{t_{in}}^{t_{fin}} |(\mathbf{x}_o(t) - \mathbf{c}(t))|\, dt}{t_{fin} - t_{in}}. \tag{3.6}$$

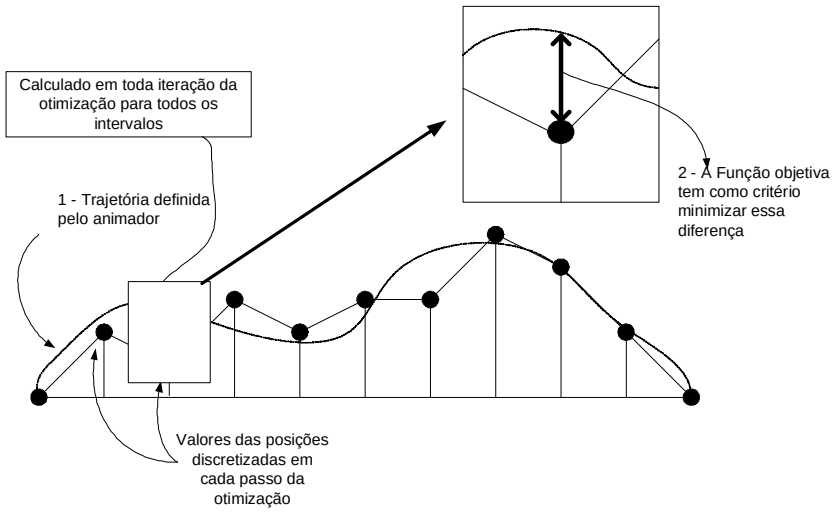


Figura 3.6 - Exemplo de uma função objetivo

3.2.2.2 Restrições

As restrições impostas ao problema de otimização (Equação 3.1) podem ser classificadas em três tipos: restrições de pose, restrições mecânicas e restrições Newtonianas.

Restrições de pose

Nas restrições de igualdade e desigualdade relacionadas à pose C_{pi} e C_{pd}, as restrições definem as posições e orientações da estrutura em alguns instantes de tempo (Figura 3.4). Essas poses têm ser obedecidas pelo otimizador e restringem o espaço de busca. Podem-se definir três tipos de poses para o movimento da estrutura: as inicias, as finais e as intermediarias. Por exemplo, numa situação em que a figura articulada realiza um salto (Figura 3.2), para delimitar o espaço de busca criam-se algumas poses iniciais e finais para fixar o momento de preparação e retorno do salto, e uma pose intermediaria para que a estrutura articulada atinja uma determinada altura. Essas poses terão que ser obedecidas, ou então a solução obtida estará violando o espaço das restrições.

Restrições mecânicas

Nas restrições mecânicas C_{mi} e C_{md}, impõe-se ao sistema restrições com relação ao ambiente de animação. Por exemplo, na animação mostrada na Figura 3.2, os movimentos onde o limite do chão é violado não são considerados. Também são consideradas restrições mecânicas, as restrições que atribuem valores máximos ou mínimos para as rigidezes dos atuadores das juntas. As restrições impostas a essas variáveis podem ser baseadas em informações biomecânicas.

Restrições Newtonianas

As restrições Newtonianas C_{ni} e C_{nd} representam o conjunto de restrições em que as acelerações dos graus de liberdade considerados são proporcionais às forças generalizadas neles exercidas. Em outras palavras, que as equações de movimento são preservadas. Essas equações de movimento dos corpos articulados são obtidas a partir da formulação de Lagrange (vide Anexo B e D) como

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} - F_j = 0, \tag{3.7}$$

onde j indica que existe uma equação de movimento associada a cada grau de liberdade q_j.

As derivadas dos graus de liberdade com relação ao tempo que aparecem na equação 3.7 não podem ser obtidas analiticamente, já que a função que define a evolução de cada grau de liberdade com o tempo corresponde à solução que se deseja alcançar. Essas derivadas são portanto calculadas através de métodos numéricos. No trabalho de Witkin e Kass (1988), as derivadas são aproximadas por diferenças finitas, onde a partição do tempo de animação deve ser feita em intervalos constantes. Em Winzell (1998), explora-se a possibilidade de uma discretização não uniforme do tempo de animação calculando-se as derivadas pelo método de elementos finitos.

3.3 Considerações Finais

Nesse capítulo, foram consideradas as etapas e os métodos necessários para se definir o problema de otimização da estrutura articulada representada pela Figura 3.3 que são indispensáveis à implementação de um sistema de animação apresentado no Capítulo 4.

Capítulo 4
Sistema de Animação Baseado na Técnica de Controle de movimento por otimização com restrições

4.1 Introdução

Nesse capítulo, descreve-se o desenvolvimento do sistema de animação de figuras articuladas através do controle por otimização não-linear com restrições. Esse sistema é capaz de criar, através de um conjunto de interfaces gráficas, animações realísticas com um balanceamento entre controle e automação.

A exposição no restante desse capítulo é regulada pela estrutura articulada descrita na Seção 3.2.1. Na Seção 4.2, mostram-se todos os componentes do sistema. Na Seção 4.2.1, 4.2.2 e 4.2.3, mostram-se os detalhes inerentes à implementação de cada um dos subsistemas. E por último, na Seção 4.3, fazem-se algumas considerações finais sobre o sistema descrito.

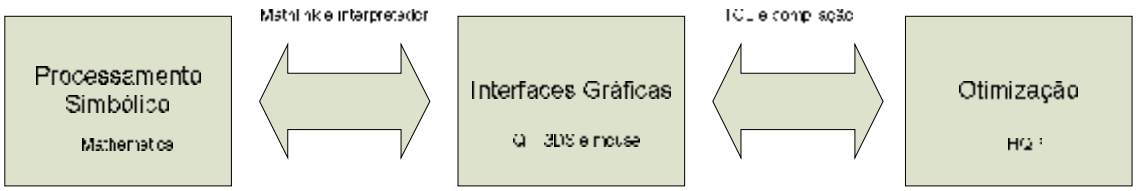


Figura 4.1 – Modularidade do sistema

4.2 Descrição do Sistema

O sistema apresentado nesse trabalho é baseado em (Cohen, 1992) e está estruturado em três subsistemas: interfaces gráficas, processamento simbólico e centro de otimização (Figura 4.1).

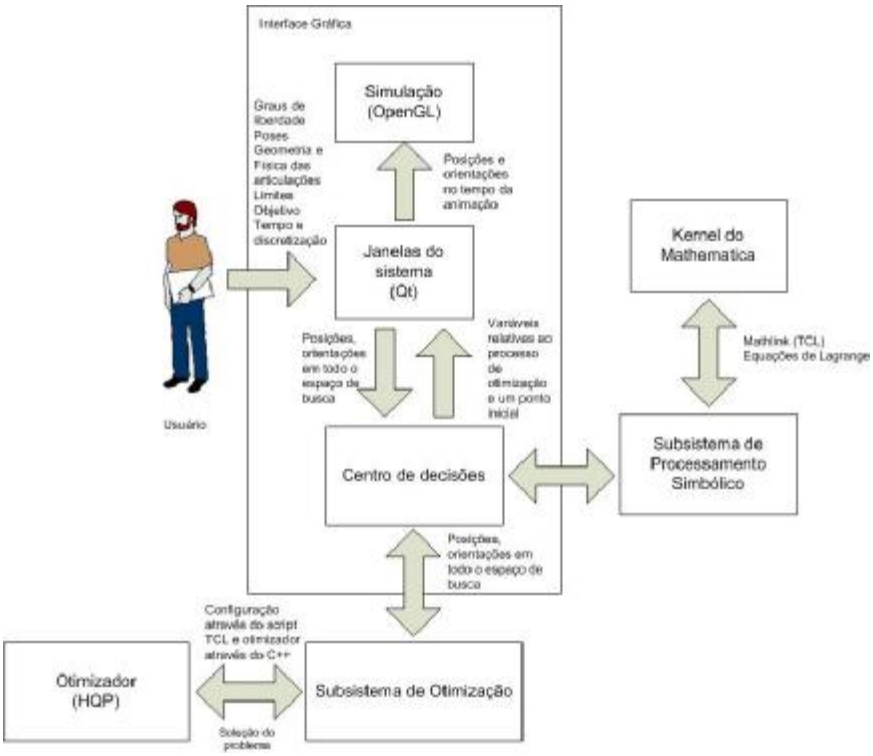


Figura 4.2 - Fluxo do sistema

Grande parte das tecnologias utilizadas para implementação é de código aberto. O sistema é implementado através da linguagem de programação C++ e as animações são geradas através da API *OpenGL*. Os módulos estão interligados e o fluxo de mensagens trocadas entre eles é controlado por um centro de decisões que é parte do subsistema de interfaces gráficas (Figura 4.2).

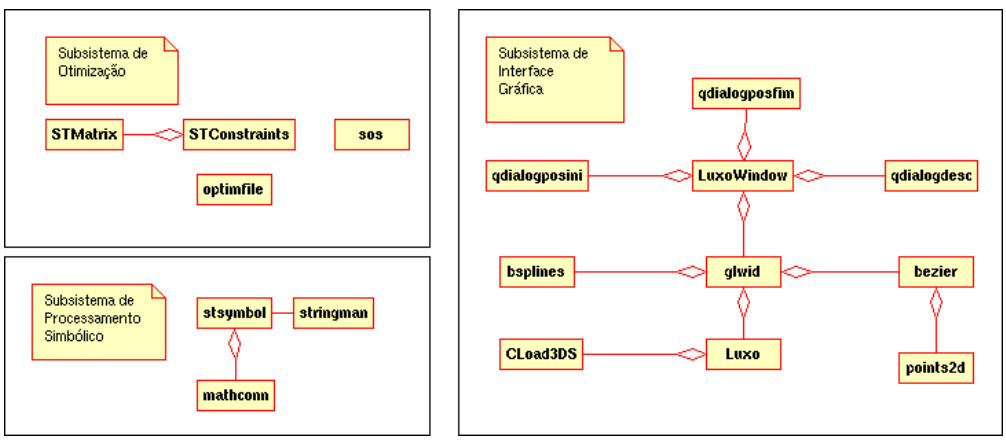


Figura 4.3 - Diagrama do Sistema

4.2.1 Subsistema de Interfaces Gráficas

O subsistema de interfaces gráficas é constituído por todos os elementos que são ativados pelo animador, e seu diagrama de classes está ilustrada na Figura 4.3. As interfaces gráficas do sistema foram desenvolvidas utilizado o *framework* Qt (Trolltech, 2004). A classe *LuxoWindow* trata da janela principal da aplicação (Figura 4.4) e de todos os eventos diretamente gerados pela interação do sistema com o animador.

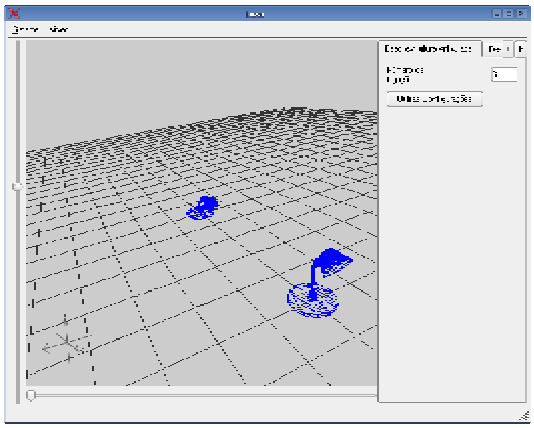


Figura 4.4 - Janela principal

O centro de controle serve de intermediário entre os outros subsistemas e é ativado pelo animador no momento da criação da animação (Figura 4.5).

Figura 4.5 – Centro de decisões

É através desse subsistema que o animador descreve a estrutura articulada, descreve o modelo de otimização e solicita a execução de todo o processo necessário para a geração da animação.

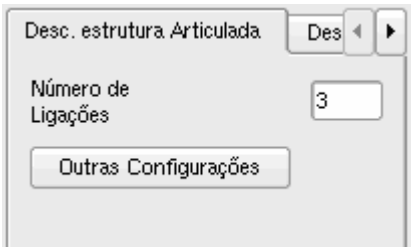


Figura 4.6 – *Tab* Descrição da Estrutura Articulada

Descrição da estrutura articulada

Ao selecionar o *tab* "Descrição da estrutura articulada", o usuário ativa a interface gráfica mostrada na Figura 4.6. Através dessa interface o usuário pode definir a topologia e física da estrutura articulada conforme descrição apresentada na Seção 3.2.1. Na versão atual do sistema, apenas estruturas compostas de ligações cilíndricas, juntas uniaxiais com eixo de rotação paralelo ao eixo Z e atuadores angulares foram implementados. Assim, a interface gráfica para definição da topologia considera apenas o número de ligações, o comprimento de cada ligação, o raio da seção transversal de cada ligação e a massa específica do material componente de cada ligação (Figura 4.7).

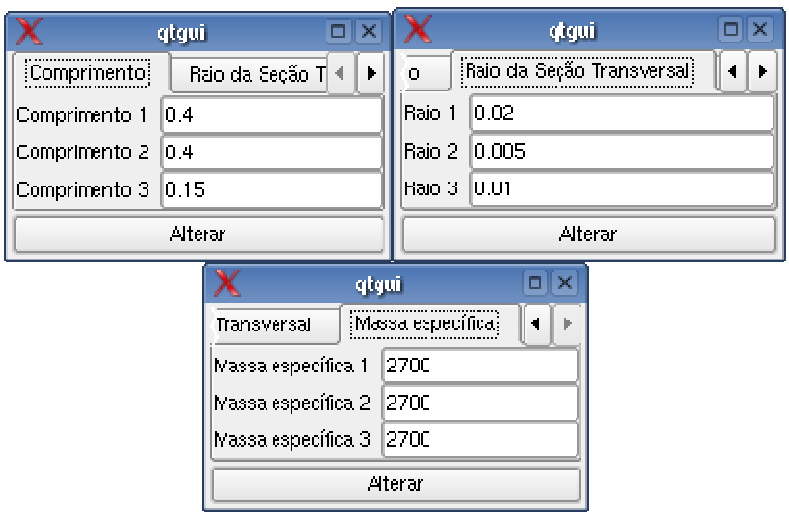


Figura 4.7 – Caixas de diálogos de informações da estrutura

Descrição do modelo de otimização

Ao selecionar o *tab* “Descrição do Modelo de Otimização”, o usuário ativa a interface gráfica mostrada na Figura 4.8. Através dessa interface o usuário pode definir todo o modelo de otimização da Seção 3.2.2, que consiste em: definir a duração da animação, o particionamento do tempo da animação, as restrições de pose e a função objetivo.

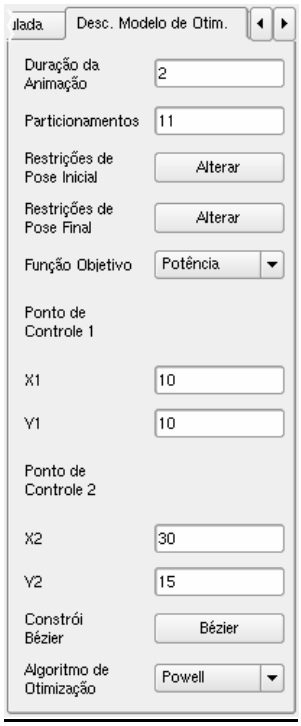


Figura 4.8 – *Tab* Descrição do Modelo de Otimização

Na versão atual do sistema, as restrições de pose consideram apenas as configurações inicial e final da estrutura articulada (Figura 4.9).

Figura 4.9 - Restrições de pose

Apenas duas funções objetivos estão disponíveis para seleção nessa versão: diferença entre uma curva de *Bézier* cúbica (quatro pontos de controle) definindo a trajetória da base $\mathbf{x}_0$ aproximada pelo animador e a trajetória real; e a potência dos atuadores angulares. As interfaces gráficas para a definição das restrições de pose e para a seleção da função objetivo são ilustradas nas Figura 4.10.

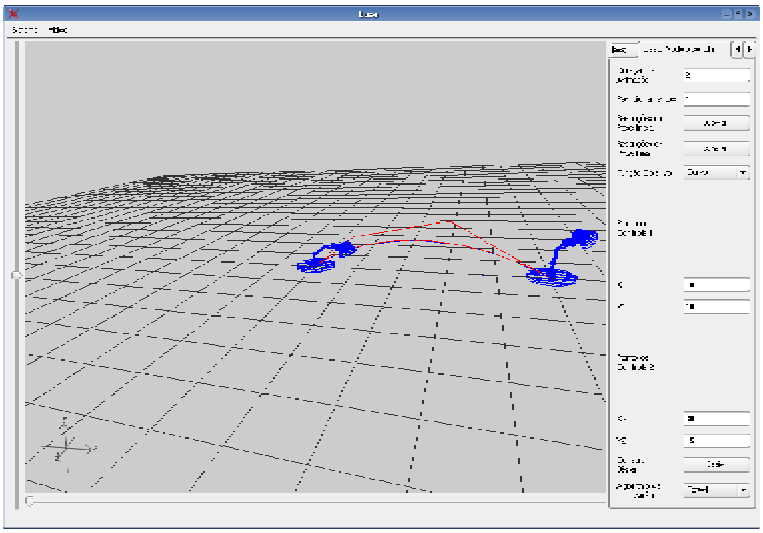


Figura 4.10 - O critério do objetivo baseado numa curva de trajetória

Ativação do processo de solução

Ao selecionar o *tab* "Processo de solução", o usuário ativa o botão da interface gráfica mostrada na Figura 4.11. Através desse elemento de interface o sistema inicia o processo de geração do movimento da estrutura por *spacetime constraints*, passando o controle para o subsistema de otimização. Quando esse processo é concluído, o subsistema de otimização devolve o controle para o subsistema de interface gráfica que permite o usuário visualizar a animação da estrutura articulada.

Figura 4.11 - *Tab* Processo de solução

Visualização da Animação

Ao selecionar o tab "Visualização da animação", o usuário ativa a interface gráfica mostrada na Figura 4.12. Através dessa interface o usuário pode definir controlar a visualização da animação da estrutura articulada .

Figura 4.12 - *Tab* Visualização da Animação

Na versão atual do sistema, o usuário pode visualizar a animação sendo executada de forma contínua ou estudar cada configuração, avançando ou recuando a animação passo a passo. A animação é controlada por um módulo de simulação, que constitui um painel em *OpenGL* e uma barra horizontal (Figura 4.13). Pode-se fazer a manipulação direta da barra, mostrando no tempo discretizado a configuração atual da estrutura através do painel. A estrutura foi modelada graficamente através de uma classe *CLoad3DS* capaz de importar modelos 3D.

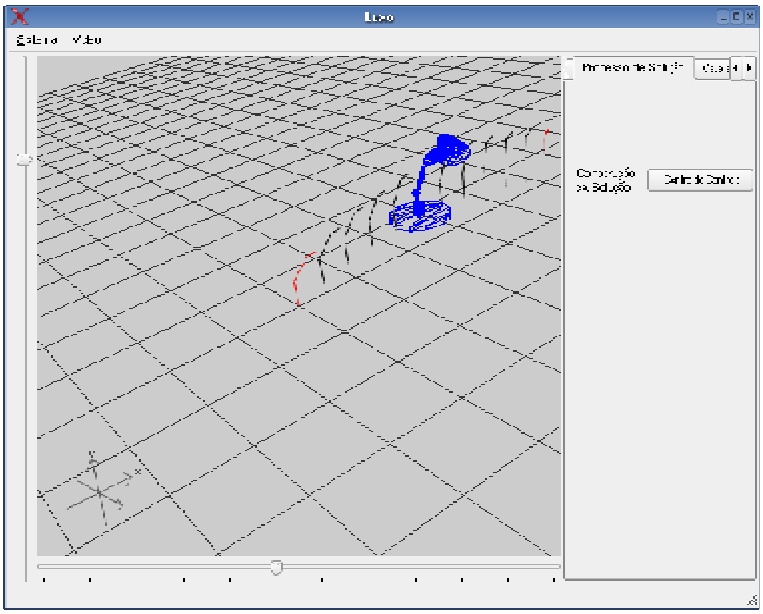


Figura 4.13 - A janela de animação

Uma janela de plotagem pelo *software gnuplot* é utilizado para mostrar a trajetória da animação em relação à base da estrutura articulada após o processo de otimização (Figura 4.14).

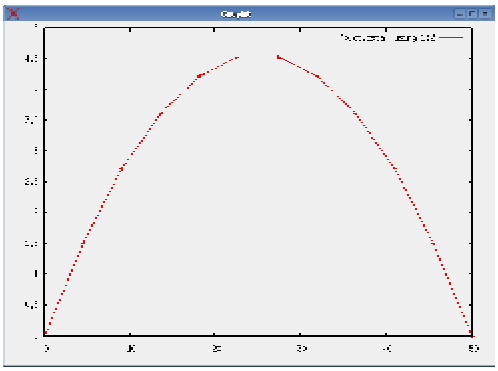


Figura 4.14 - Plotagem do movimento da base

4.2.2 Subsistema de Processamento Simbólico

O subsistema de processamento simbólico constrói todas as equações (função objetivo e restrições Newtonianas) necessárias ao processo de otimização através de um tratamento simbólico, utilizando o *kernel* do software *Mathematica*. No processo de otimização, faz-se necessário modelar toda a dinâmica da estrutura articulada durante a animação através das equações de Lagrange. Essas equações compõem o conjunto de restrições de igualdade

denominadas restrições Newtonianas. Essas equações são resolvidas numericamente em cada passo do processo de otimização A ferramenta *mathlink* foi utilizada para fazer a comunicação entre o subsistema de processamento simbólico e o *kernel* do *Mathematica* (Figura 4.15).

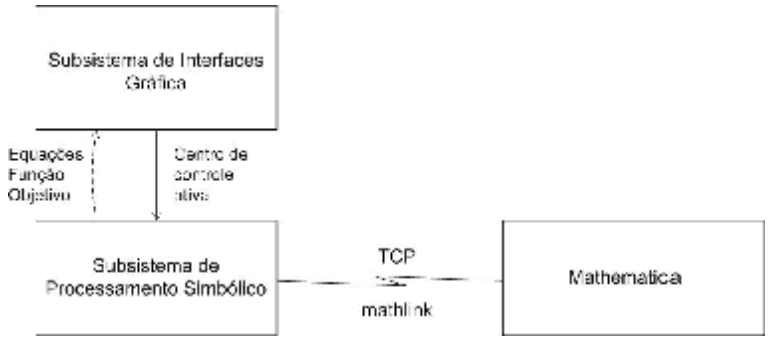


Figura 4.15 - Mensagens do Subsistema de Processamento Simbólico

O *mathlink* consiste de um conjunto de bibliotecas de funções que implementa um protocolo para envio e recebimento de expressões na linguagem do *Mathematica*. Uma das características dessa biblioteca é a independência de plataforma. Além disso, sua conexão com o *kernel* do *Mathematica* pode ser efetuada tanto localmente quanto através de uma rede heterogênea de computadores. O *mathlink* facilita a integração das rotinas realizadas no *Mathematica* incorporando-as à linguagem utilizada na aplicação. A chamada ao *kernel* do *Mathematica* é realizada utilizando um protocolo TCP e as mensagens criadas na linguagem de programação do sistema são empacotadas como *strings* seguindo a sintaxe do *Mathematica* (Tabela 4.1). Por exemplo, o comando *m1->sendmsg ("eq1=Simplify [Tdx0ldt-Tdx0-Fx0];");* envia a *string* ao kernel para o calculo da equação 1 correspondente ao grau de liberdade representado pela coordenada *x* da base da estrutura articulada da Figura 3.3

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{x}_0}\right) - \frac{\partial T}{\partial x_0} = F_{x_0} \qquad 3.8$$

onde T é a energia cinética da estrutura articulada, $\dot{x}_0$ é a velocidade associada ao grau de liberdade x_0 e F_{x_0} é a força generalizada aplicada na direção desse grau de liberdade. O *kernel* do *Mathematica* retornará a expressão simplificada conforme a solicitação "Simplify" embutida

na *string*. Depois de todas as mensagens serem processadas pelo *kernel*, tem um conjunto de equações na sintaxe do *Mathematica*. A classe *stringman* () faz um processamento interno de *strings* adaptando para a linguagem C++ que será processado pelo subsistema de otimização.

Tabela 4.1 – Trecho do código das equações de Lagrange através do *mathlink*

```
/*Lagrange Equation */
ml->sendmsg ("eq1=Simplify[Tdx0ldt - Tdx0 - Fx0 ];");
ml->sendmsg ("eq2=Simplify[Tdy0ldt - Tdy0 - Fy0 ];");
ml->sendmsg ("eq3=Simplify[Tdth1ldt - Tdth1 - Fth1];");
ml->sendmsg ("eq4=Simplify[Tdth2ldt - Tdth2 - Fth2];");
ml->sendmsg ("eq5=Simplify[Tdth3ldt - Tdth3 - Fth3];");
```

Processamento simbólico das Restrições Newtonianas

Para o processamento simbólico das restrições Newtonianas, o subsistema de processamento simbólico, através da classe *mathconn*, abre uma conexão local com o *Mathematica*. A classe *mathconn* também é responsável por todas as mensagens, na sintaxe do *Mathematica*, que serão encaminhadas ao *mathlink* e posteriormente ao *kernel*. A classe *stsymbol* é encarregada de enviar todas as informações necessárias à montagem simbólica das restrições Newtonianas. A Tabela 4.1 ilustra um trecho do método *start ()* da classe *stsymbol* onde essas mensagens são enviadas (o código completo desse método encontra-se no Anexo G).

Processamento simbólico da Função Objetivo

De forma análoga ao processamento simbólico das funções Newtonianas, de acordo com a função objetivo selecionada pelo animador através do subsistema de interface gráfica, podem-se resolver simbolicamente as equações da função objetiva (ver equações 3.3, 3.4, 3.5 e 3.6). A conexão mantida pelo processamento das restrições Newtonianas permanece válida durante o processo simbólico da função objetivo a classe *mathconn* envia as mensagens através do método *start ()* da classe *stsymbol* onde essas mensagens são enviadas.

4.2.3 Subsistema de Otimização

A parte principal da solução do problema de animação pela técnica de *Spacetime constraints* é executada pelo subsistema de otimização. O esquema desse subsistema está apresentado na Figura 4.16. O elemento principal dessa arquitetura é o otimizador representado pela classe *STConstraints*. O otimizador selecionado na implementação desse subsistema é o HQP (Huge Quadratic Programming – otimizador de código fonte aberto – vide (HQP, 2004)). Esse otimizador é bastante robusto e permite o tratamento de problemas com um número grande de variáveis no espaço de busca. Ele resolve o problema de otimização através de uma combinação do algoritmo SQP (*Sequencial Quadratic Programming*) e do método BFGS (Franke, 1998). O calculo dos gradientes e das matrizes Hensianas da função objetivo e das restrições é feito automaticamente através do pacote ADOL-C (2004) de diferenciação automática, tornando a construção da otimização mais simples.

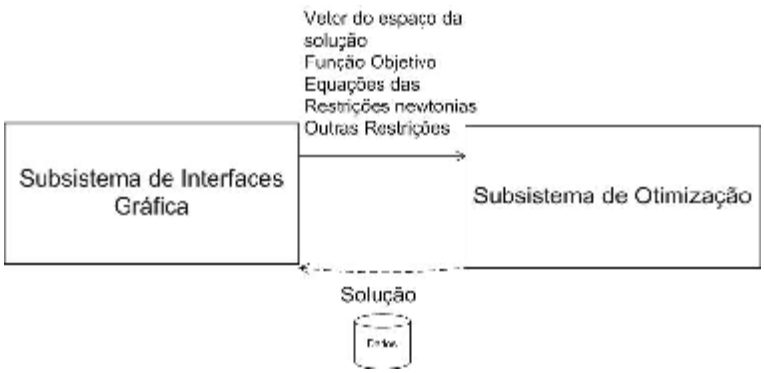


Figura 4.16 - Subsistema de Otimização

O centro de decisões do subsistema de interfaces gráficas repassa ao subsistema de otimização as variáveis que compõem o espaço de busca, a equação da função objetivo e as restrições Newtonianas escritas na sintaxe da linguagem C++. Com essas informações, em C++, sobre a função objetivo e restrições Newtonianas, a classe *optimfile* constrói um arquivo contendo a implementação dos métodos correspondentes que fazem parte da classe *STConstraints* (classe que incorpora o processo de otimização) necessários à funcionalidade do otimizador (Equação 2.7). Para completar a funcionalidade do otimizador a classe *SOS*, inicia o

processo de otimização através da escolha de um algoritmo de SQP (Powel ou Schittkowski) através do subsistema de interface gráfica. O resultado do otimizador é registrado em arquivos que, posteriormente, são lidos pelo subsistema de interfaces gráficas para visualizar a animação.

4.3 Considerações Finais

Nesse capítulo, foram apresentados todos os subsistemas que compõem o sistema de animação por otimização não-linear com restrições. O sistema apresentado permite ao animador criar animações dinâmicas e realísticas de estruturas articuladas através de uma interface gráfica amigável.

Capítulo 5
Apresentação e Análise dos Resultados

5.1 Introdução

Nesse capítulo, descrevem-se os resultados obtidos através do sistema de animação de figuras articuladas. Os resultados obtidos estão agrupados por tipo de função objetivo e número de partições do tempo da animação. Nesse capítulo, foi introduzida mais uma função objetivo (a função de minimização entre a trajetória especificada pelo animador e o centróide do corpo articulado) para dar mais base de comparação entre os resultados. As funções objetivos implementadas no sistema são quadráticas para se adequarem ao algoritmo de otimização utilizado (SQP).

O restante desse capítulo é regulado pela arquitetura desenvolvida no Capítulo 4. Na Seção 5.2, mostram-se todos os resultados obtidos, assim como, suas telas de saída e tabelas de resultados. Nessa seção, os resultados estão distribuídos por: Minimização da Potência Angular (MPA), Minimização da Trajetória em relação à Base (MTB) e Minização da Trajetória em relação ao Centro de massa (MTC). E por último, na Seção 5.3, fazem-se análises sobre os resultados obtidos.

5.2 Apresentação dos exemplos

Nessa seção, são apresentados vários exemplos de animações para avaliar a técnica de *spacetime constraints*, quanto aos critérios de controle da animação, realismo da animação e eficiência. O modelo de estrutura articulada utilizada em todos os exemplos está ilustrado na Figura 3.3 e suas propriedades estão apresentadas na Tabela 5.1.

Tabela 5.1 – Descrição da estrutura articulada

Comprimento da primeira ligação	$0.4\,m$
Comprimento da segunda ligação	$0.4\,m$
Comprimento da terceira ligação	$0.15\,m$
Raio da primeira ligação	$0.02\,m$
Raio da segunda ligação	$0.005\,m$
Raio da terceira ligação	$0.01\,m$
Massa específica da primeira ligação	$2700\,kg/m^3$
Massa específica da segunda ligação	$2700\,kg/m^3$
Massa específica da terceira ligação	$2700\,kg/m^3$

O otimizador utilizado pela técnica de *spacetime constraints* em todos esses exemplos foi o HQP, em que se adotou empiricamente um número máximo de 1000 iterações para a busca do ponto ótimo. A tolerância adotada no estudo da convergência é de 10^{-3} (critério de parada). Quando o otimizador atende algum desses critérios, um resultado qualquer, denominado sub-ótimo, será devolvido ao sistema. Um resultado impraticável será obtido quando o otimizador a partir de um ponto inicial não conseguir iterar no espaço das soluções, violando os limites das restrições. Um resultado ótimo será obtido quando o algoritmo atingir um valor mínimo para a função objetivo, onde qualquer iteração naquele ponto no espaço resultará em soluções menos otimizadas.

Os exemplos analisados foram divididos em três subgrupos de acordo com a função objetivo a ser minimizada: potência angular, diferença entre a posição da base da estrutura e uma trajetória especificada pelo animador por uma curva *Bézier* cúbica e diferença entre a posição do

centro de massa da estrutura e uma trajetória especificada pelo animador por uma curva *Bézier*. Em todos os saltos analisados a estrutura articulada busca vencer uma distância de três metros entre o ponto da base no momento do lançamento e o ponto da base no momento da aterrissagem. O tempo total da animação é de 2 segundos e as poses finais de todos os exemplos são $q_1 = 90^o$, $q_2 = 45^o$, $q_3 = 0^o$. Para todos os exemplos, foram gerados resultados correspondentes a partições do tempo total da animação em 8, 16 e 32 partições.

5.2.1 Minimização da Potência Angular

Em todos os exemplos dessa seção, o ponto inicial do espaço de busca fornecido para o otimizador é obtido incluindo os valores das poses inicial e final e os valores das poses intermediárias definidos por interpolação linear entre as poses inicial e final.

Exemplo MPA1a – Salto com pose inicial $q_1 = 90^o$, $q_2 = 60^o$, $q_3 = 45^o$ e 8 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 0,09116 $(J/s)^2$ (quadrado da potência total do movimento) da função objetivo em 6,87 segundos, num total de 74 passos de otimização. Na Tabela 5.2, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.1, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.2 – Resultado para o Exemplo MPA1a

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	90	60	45	0	0,01959
0,25	0,52577	1,99594	83,3024	67,9187	67,8788	0,00123	6,479
0,5	1,05218	3,42617	76,0223	76,0193	76,0446	689,914	4,4912

0,75	1,55367	4,28493	75,5559	75,5559	75,556	85,1102	0,95084
1	2,05515	4,57246	75,0931	75,0931	75,0929	54,3759	0,29571
1,25	2,55661	4,28875	74,6316	74,6316	74,6415	333,621	0
1,5	3,05807	3,43381	74,1718	74,1716	74,2093	140,594	0,03080
1,75	3,5591	2,00755	73,8313	73,5363	73,4534	18,7888	0,64727
2	4	0	90	45	0	0	0,00001

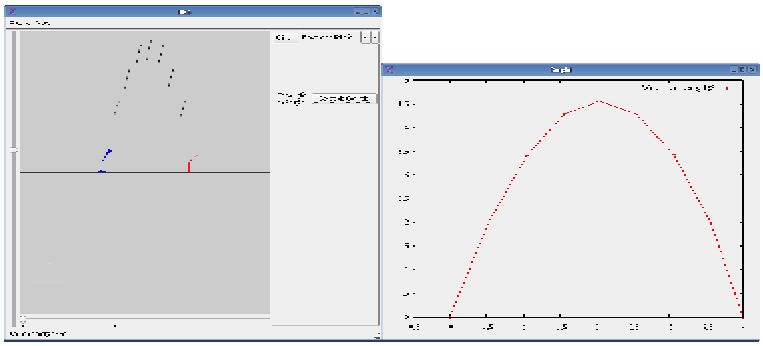


Figura 5.1 – Janelas do Sistema para o Exemplo MPA1a

Exemplo MPA1b – Salto com pose inicial $q_1 = 90^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 16 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 0,00696 $(J/s)^2$ da função objetivo em 94,6 segundos, num total de 179 passos de otimização. Na Tabela 5.3, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.2, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.3 – Resultado para o Exemplo MPA1b

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	90	60	45	0,0001	0,00151
0,125	0,275802	1,0671	83,1683	67,499	57,7817	0,00063	0,00630

0,25	0,550844	1,99489	76,4781	75,918	75,9151	14,0556	6,25393
0,375	0,80296	2,7808	75,8949	75,6123	75,6115	0	0,66209
0,5	1,05508	3,42392	75,3117	75,3079	75,3116	38,5113	2,80741
0,625	1,30677	3,92416	74,8483	74,8475	74,8473	3,72634	1,85293
0,75	1,55844	4,28161	74,3876	74,3855	74,3857	12,4198	7,06366
0,875	1,81004	4,49624	73,9481	73,8974	73,8973	0	6,20198
1	2,06163	4,56808	73,5086	73,4102	73,4085	0	0,43616
1,125	2,31322	4,49712	73,0695	72,9245	72,9242	0	2,8381
1,25	2,5648	4,28337	72,6306	72,4402	72,4445	1,17839	3,76404
1,375	2,81574	3,92665	72,3769	71,7274	71,7831	4,37211	3,38782
1,5	3,05866	3,42513	74,4531	68,0699	68,321	0,25523	0
1,625	3,29682	2,78043	77,8399	62,7706	62,6615	0,06786	1,46232
1,75	3,53195	1,99384	81,9783	56,6115	55,0431	0,02965	0,70065
1,875	3,76645	1,06754	85,8736	49,8846	29,8962	0,00823	0,02100
2	4	0	90	45	0	0	0

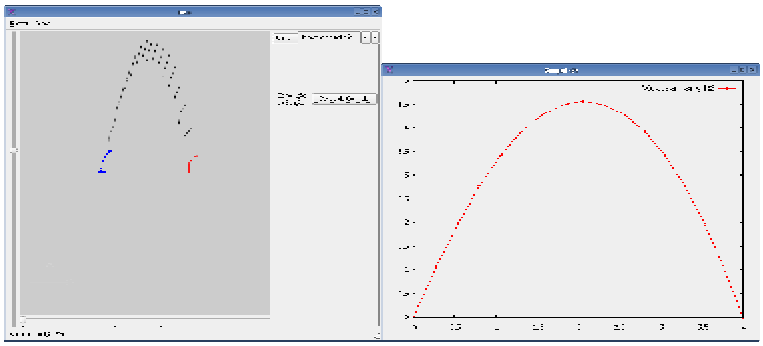


Figura 5.2 – Janelas do Sistema para o Exemplo MPA1b

Exemplo MPA1c – Salto com pose inicial $q_1 = 90^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 32 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 2,25254 $(J/s)^2$ da função objetivo em 1108,47 segundos, num total de 117 passos de otimização. Na Tabela 5.4, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.3, é ilustrado o salto da estrutura

articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.4 – Resultado para o Exemplo MPA1c

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	90	60	45	0	0,00385553
0,0625	0,12979	0,551385	88,9686	61,6121	61,6037	0,0065	387,42
0,125	0,26106	1,06845	87,2697	63,1854	63,0771	0,05077	0
0,1875	0,39147	1,55008	85,8058	64,5127	64,5115	0	46,0159
0,25	0,52189	1,99618	84,3329	65,8523	65,8513	0	23,1532
0,3125	0,65229	2,40674	82,8592	67,2137	67,2766	0	0
0,375	0,78266	2,78176	81,3907	68,6065	68,924	0	0,479966
0,4375	0,91292	3,12115	79,9611	70,0992	71,6718	0	00
0,5	1,04314	3,42499	78,5406	71,6393	74,7099	0	0
0,5625	1,17331	3,69329	77,1306	73,2314	78,0585	0	0
0,625	1,3034	3,92603	75,7335	74,881	81,7436	6,3213	0
0,6875	1,42984	4,12273	75,3165	74,7438	80,595	0	0
0,75	1,55628	4,28375	74,9001	74,6104	79,4642	0	0
0,8125	1,68271	4,40907	74,4845	74,4808	78,3513	151,743	0
0,875	1,80876	4,49863	74,1718	74,1712	76,7645	152,387	0
0,9375	1,93474	4,5525	73,8773	73,8394	75,1334	0	0
1	2,06072	4,57067	73,5845	73,515	73,5356	0	10,1528
1,0625	2,18662	4,55302	73,3374	73,3354	73,4128	66,051	0
1,125	2,31243	4,49965	73,1156	73,1154	73,1946	17,6457	0
1,1875	2,43823	4,41058	72,8948	72,8948	72,9788	731,045	0
1,25	2,56403	4,28582	72,6747	72,6742	72,7665	0	0
1,3125	2,68983	4,12535	72,4548	72,4542	72,5597	0	0
1,375	2,81563	3,92918	72,2351	72,2351	72,3584	220,445	0
1,4375	2,94143	3,69732	72,0162	72,0156	72,1603	0	0
1,5	3,06722	3,42975	71,7975	71,7969	71,9675	0	0
1,5625	3,19301	3,12649	71,5789	71,5788	71,7802	25,2556	0
1,625	3,3188	2,78752	71,361	71,3608	71,5968	369,788	0

1,6875	3,44454	2,41284	71,1552	71,1251	71,3768	0	0
1,75	3,57028	2,00247	70,9496	70,8902	71,1626	83,3159	1,43867
1,8125	3,69252	1,55539	71,7779	69,4088	70,175	8,28306	0
1,875	3,80112	1,06983	76,443	62,3211	57,5539	0,72398	0,66581
1,9375	3,90257	0,55083	82,7674	53,0185	32,1563	0,18665	0,15613
2	4	0	90	45	0	0,00054	0,00033

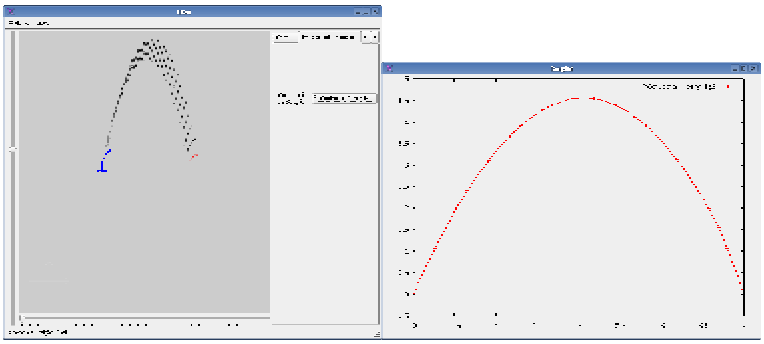


Figura 5.3 – Janelas do Sistema para o Exemplo MPA1c

Exemplo MPA2a – Salto com pose inicial $q_1 = 120^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 8 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 0,0015 $(J/s)^2$ da função objetivo em 5,99 segundos, num total de 66 passos de otimização. Na Tabela 5.5, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.4, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.5 – Resultado para o Exemplo MPA2a

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	120	60	45	0	0,00064

0,25	0,52079	1,97964	110,116	67,5813	65,9869	0	0,10892
0,5	1,04653	3,3947	99,2223	75,9096	75,9237	0	6,27277
0,75	1,57237	4,24707	88,8783	86,8117	104,718	1,27979	0,00507
1	2,07113	4,53751	85,5416	81,7115	87,0795	0,243162	0,02823
1,25	2,55822	4,25755	85,577	74,2916	77,3749	0,05081	0
1,5	3,0389	3,40722	87,3422	64,626	64,5298	0,00883	0,49431
1,75	3,5173	1,98701	89,7126	54,9495	54,3082	0,00365	1,04869
2	4	0	90	45	0	0	0

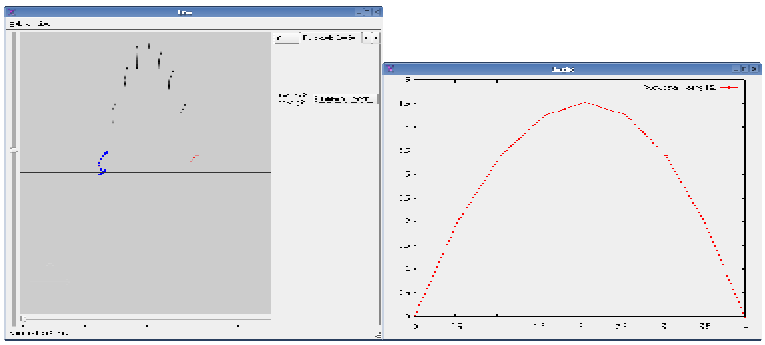


Figura 5.4 – Janelas do Sistema para o Exemplo MPA2a

Exemplo MPA2b – Salto com pose inicial $q_1 = 120^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 16 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 4,19319 $(J/s)^2$ da função objetivo em 57,38 segundos, num total de 79 passos de otimização. Na Tabela 5.6, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.5, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.6 – Resultado para o Exemplo MPA2b

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	120	60	45	0,00157	6,06752

0,125	0,27054	1,05351	111,658	68,0969	53,0596	0	0,00235
0,25	0,54302	1,96893	103,293	77,0887	66,7407	0	0
0,375	0,81662	2,74707	95,0071	87,5643	89,0096	0,84010	0,36888
0,5	1,0739	3,39133	91,3608	91,335	102,462	128,887	0
0,625	1,32292	3,89392	89,8396	89,8395	98,8366	12,6185	0
0,75	1,57196	4,25391	88,3135	88,3175	95,0642	0	0
0,875	1,82099	4,47131	86,7853	86,7853	91,2338	183,573	0
1	2,06999	4,54611	85,2637	85,2494	87,4089	0	0
1,125	2,31897	4,47832	83,7453	83,7449	83,7454	278,081	126,969
1,25	2,56737	4,2678	82,4178	82,4104	82,4113	23,3712	138,792
1,375	2,81496	3,91439	81,3637	81,3637	84,8358	5649,85	0
1,5	3,0625	3,41826	80,3208	80,3208	87,3211	736,44	0
1,625	3,30999	2,77941	79,2861	79,2861	89,8775	203,379	0
1,75	3,55744	1,99784	78,2627	78,2512	92,4736	51,6414	0
1,875	3,80334	1,07336	77,6411	76,3081	92,2188	14,4128	0
2	4	0	90	45	0	0	0

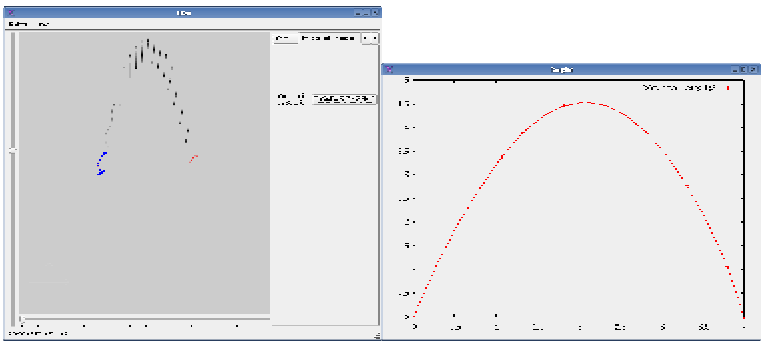


Figura 5.5 – Janelas do Sistema para o Exemplo MPA2b

Exemplo MPA2c – Salto com pose inicial $q_1 = 120^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 32 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 282,855 $(J/s)^2$ da função objetivo em 1848,85 segundos, num total de 316 passos de otimização. Na Tabela 5.7, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.6, é ilustrado o salto da estrutura

articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.7 – Resultado para o Exemplo MPA2c

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	120	60	45	0	2493,26
0,0625	0,16105	0,52298	107,9	76,041	61,0409	0	0,04797
0,125	0,32630	1,0219	95,6195	95,074	95,2008	113,318	48,9491
0,1875	0,45151	1,50518	94,6805	94,6768	94,6768	319,325	3448,28
0,25	0,57586	1,95287	93,981	93,981	93,981	2817,12	2749,98
0,3125	0,70022	2,3649	93,2828	93,2828	93,2828	1549,42	10721,5
0,375	0,82458	2,74127	92,5843	92,5843	92,5843	1875,06	860,79
0,4375	0,94894	3,08198	91,8856	91,8856	91,8856	9609,8	372,404
0,5	1,07331	3,38702	91,1869	91,1869	91,189	7119,69	0,00366
0,5625	1,19768	3,6564	90,4878	90,4878	90,4878	12360,1	5458,61
0,625	1,32204	3,89012	89,7888	89,7888	89,7888	1675,44	156,324
0,6875	1,44641	4,08818	89,0898	89,0898	89,0898	74615,7	13331,2
0,75	1,57077	4,25058	88,3909	88,3909	88,3908	1355,8	104,892
0,8125	1,69513	4,37731	87,6921	87,6921	87,6921	17191	85,0992
0,875	1,81949	4,46839	86,9935	86,9935	86,9935	54663,2	199,133
0,9375	1,94385	4,5238	86,2952	86,2952	86,2952	5612,5	171,14
1	2,0682	4,54354	85,5972	85,5972	85,5972	150,434	974,641
1,0625	2,19255	4,52763	84,8995	84,8995	84,8995	8945,15	329,045
1,125	2,3169	4,47605	84,2024	84,2024	84,2024	9771,94	2506,08
1,1875	2,44124	4,38881	83,5057	83,5057	83,5057	25481,1	53969,7
1,25	2,56557	4,26591	82,8095	82,8095	82,8095	5073,1	8940,55
1,3125	2,68989	4,10734	82,1141	82,1139	82,1138	8,04972	323,402
1,375	2,81421	3,91312	81,4191	81,4191	81,4191	62,4942	8630,8
1,4375	2,93852	3,68322	80,7249	80,7249	80,7249	11734,4	34875,5
1,5	3,06282	3,41767	80,0315	80,0315	80,0315	21401,8	17903,8
1,5625	3,18711	3,11645	79,339	79,3389	79,3389	420,139	38148
1,625	3,31139	2,77956	78,6486	78,6454	78,6454	196,246	12722,3

1,6875	3,43495	2,4067	78,2142	78,2142	81,6619	5614,62	0
1,75	3,55848	1,99815	77,79	77,79	84,7291	7332	0
1,8125	3,68196	1,55393	77,3768	77,369	87,8331	0	0
1,875	3,80543	1,07402	76,9672	76,9639	91,0096	497,395	0
1,9375	3,92781	0,55829	76,846	75,9314	92,2446	85,581	0
2	4	0	90	45	0	0	0,0001

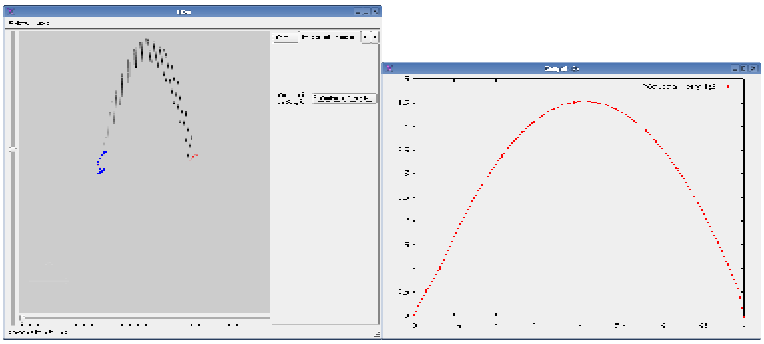


Figura 5.6 – Janelas do Sistema para o Exemplo MPA2c

Exemplo MPA3a – Salto com pose inicial $q_1 = 150^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 8 partições

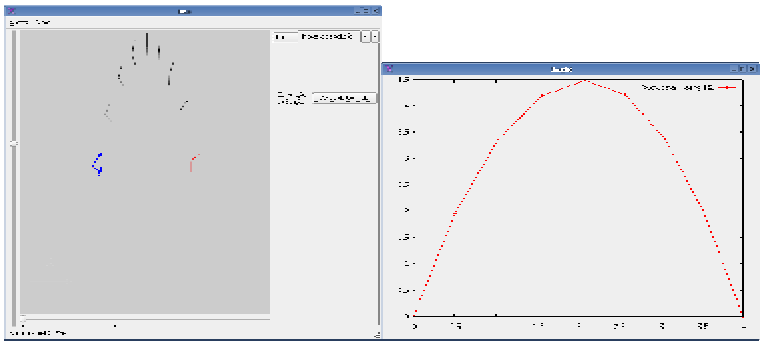
Nesse exemplo, o otimizador atingiu o valor ótimo de 0,00056 $(J/s)^2$ da função objetivo em 8,96 segundos, num total de 111 passos de otimização. Na Tabela 5.8, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.7, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.8 – Resultado para o Exemplo MPA3a

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	150	60	45	0,00013	0
0,25	0,50738	1,96357	136,41	65,8522	61,7676	0	0,02918

0,5	1,02513	3,36342	122,093	72,9429	72,0676	0,00035	0,07551
0,75	1,5514	4,20372	107,308	80,9965	80,4821	0,00049	0,05270
1	2,08289	4,48774	92,2758	89,9513	90,4511	1,50321	0,70459
1,25	2,57574	4,21908	88,1573	83,9587	84,5101	0,28089	0,15620
1,5	3,05543	3,38092	87,6436	72,7382	71,0985	0,04249	0,16951
1,75	3,53043	1,97511	87,9895	57,3762	31,3819	0,01213	0,00501
2	4	0	90	45	0	0,00054	0

Figura 5.7 – Janelas do Sistema para o Exemplo MPA3a



Exemplo MPA3b – Salto com pose inicial $q_1 = 150^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 16 partições

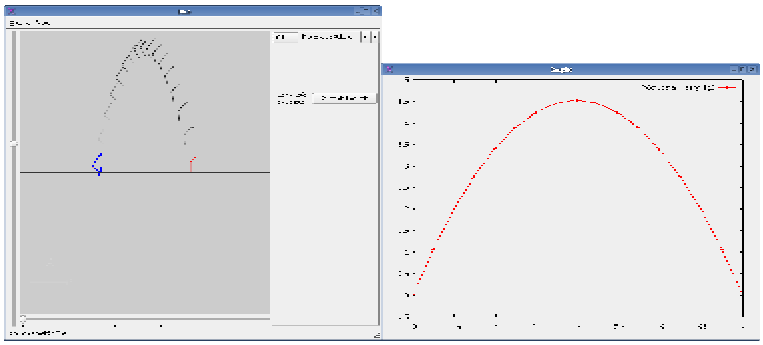
Nesse exemplo, o otimizador atingiu o valor ótimo de $0{,}00001\,(J/s)^2$ da função objetivo em 87,79 segundos, num total de 126 passos de otimização. Na Tabela 5.6, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.5, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.9 – Resultado para o Exemplo MPA3b

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	150	60	45	0,00074	0

0,125	0,24499	1,06776	147,201	58,5166	57,9461	0,00062	1,28761
0,25	0,49139	1,99105	143,45	56,7697	56,0846	0,00059	0
0,375	0,73849	2,7722	139,728	55,0909	54,4236	0,00061	0
0,5	0,98624	3,41124	136,037	53,4798	52,9873	0	0
0,625	1,23463	3,90811	132,351	51,9669	51,7805	0	0,08428
0,75	1,48366	4,26281	128,652	50,5483	50,4936	0	0,24064
0,875	1,73326	4,47539	124,943	49,2291	49,1898	0,00026	0,21183
1	1,98335	4,54594	121,247	47,9995	47,9857	0	0,69782
1,125	2,2339	4,47443	117,548	46,8726	46,854	0,00071	0,68761
1,25	2,48478	4,26098	113,878	45,8173	45,77	0,00019	0,73485
1,375	2,73605	3,90552	110,184	44,8694	44,2894	0,00057	0,35661
1,5	2,98807	3,40802	106,287	44,1375	39,1337	0	0,02908
1,625	3,24071	2,76864	102,249	43,7265	31,4135	0	0,00705
1,75	3,49369	1,98749	98,159	43,699	22,1685	0	0,00297
1,875	3,74686	1,06462	94,057	44,1238	11,5384	0	0,00089
2	4	0	90	45	0	0	0,00015

Figura 5.8 – Janelas do Sistema para o Exemplo MPA3b



Exemplo MPA3c – Salto com pose inicial $q_1 = 150^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 32 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de $0.00378\,(J/s)^2$ da função objetivo em 1077 segundos, num total de 160 passos de otimização. Na Tabela 5.6, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.5, é ilustrado o salto da estrutura

articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.10 – Resultado para o Exemplo MPA3c

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	150	60	45	0	0
0,0625	0,12230	0,551892	148,662	59,1836	51,6311	0	0
0,125	0,24473	1,06817	147,329	58,4972	58,4873	0	151,37
0,1875	0,36775	1,54774	145,494	57,5645	57,5579	0	1,44019
0,25	0,49096	1,99173	143,654	56,6535	56,6302	0	0
0,3125	0,61436	2,40014	141,814	55,7669	55,7545	0	0,43862
0,375	0,73794	2,773	139,971	54,9036	54,9013	0	3,83231
0,4375	0,86170	3,11029	138,126	54,0636	54,0596	0	1,37446
0,5	0,98563	3,41203	136,281	53,2478	53,2465	0	6,8992
0,5625	1,10974	3,67822	134,433	52,4561	52,4444	0	0,19563
0,625	1,23399	3,90887	132,586	51,6894	51,6893	88,3407	
0,6875	1,3584	4,10398	130,737	50,9473	50,9457	0	10,6345
0,75	1,48297	4,26355	128,885	50,2299	50,1802	0	0
0,8125	1,60767	4,3876	127,036	49,538	49,4758	0	0,04796
0,875	1,73251	4,47613	125,187	48,8714	48,8169	0	0
0,9375	1,85747	4,52915	123,341	48,23	48,2195	0	1,48043
1	1,98256	4,54664	121,493	47,6144	47,6087	0	2,08277
1,0625	2,10777	4,52862	119,643	47,0247	47,0032	0	0,01317
1,125	2,23308	4,4751	117,795	46,4601	46,4598	0	39,6078
1,1875	2,35851	4,38606	115,947	45,9218	45,9187	0	2,1917
1,25	2,48402	4,26153	114,098	45,4092	45,4085	0	11,7318
1,3125	2,60963	4,10149	112,25	44,9225	44,9222	0	29,8801
1,375	2,73533	3,90596	110,4	44,4619	44,4558	0	1,94832
1,4375	2,8611	3,67493	108,551	44,0269	44,0241	0	1,03484
1,5	2,98694	3,4084	106,702	43,6178	43,6175	0	27,4838
1,5625	3,11285	3,10639	104,852	43,2346	43,2346	0	183,608
1,625	3,23882	2,76888	103	42,8773	42,8772	0	140,282

1,6875	3,36484	2,39589	101,148	42,5458	42,5453	0	27,5551
1,75	3,49091	1,98741	99,2934	42,2414	42,2121	0	44,5545
1,8125	3,61784	1,54354	97,0909	42,2857	35,679	0	0,10669
1,875	3,74519	1,06434	94,7323	42,7081	25,7396	0	0,02280
1,9375	3,87265	0,54984	92,3385	43,5985	13,632	0	0,00572
2	4	0	90	45	0	0	0

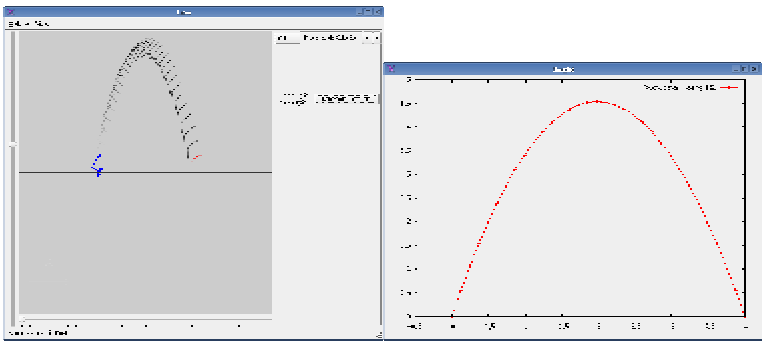


Figura 5.9 – Janelas do Sistema para o Exemplo MPA3c

5.2.2 Minimização da Trajetória em relação à Base

Em todos os exemplos dessa seção, o ponto inicial do espaço de busca fornecido para o otimizador é obtido incluindo os valores das poses inicial e final, e os valores das poses intermediárias. Os valores das poses intermediárias são definidos colocando-se a base da estrutura articulada acompanhando a curva trajetória especificada e as orientações por interpolação linear entre as orientações das poses inicial e final.

Exemplo MTB1a – Salto com pose inicial $q_1 = 90^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 8 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 53,7983 $(m)^2$ (quadrado da soma das distâncias entre a trajetória especificada e a trajetória real) da função objetivo em 5,71 segundos, num total de 58 passos de otimização. Na Tabela 5.11, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.10, é ilustrado o salto da estrutura articulada

indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.11 – Resultado para o Exemplo MTB1a

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	90	60	45	6,90042	6,90034
0,25	0,25830	1,86115	177,8	178,066	423,129	176,661	0,05087
0,5	1,22304	2,54606	74,5196	101,979	69,2029	0	0,13932
0,75	1,68633	3,54666	77,5708	98,5898	98,5848	0	44,1463
1	2,1522	3,97828	79,3383	89,9529	89,9529	0	97,0253
1,25	2,61755	3,83989	81,2229	81,2346	80,8939	119,912	0,45455
1,5	3,0985	3,13347	78,6952	78,7037	78,6601	2,25166	0,15143
1,75	3,57934	1,85633	76,1637	76,1601	76,1603	1567,72	169,58
2	4	0	90	45	0	6,90035	6,90044

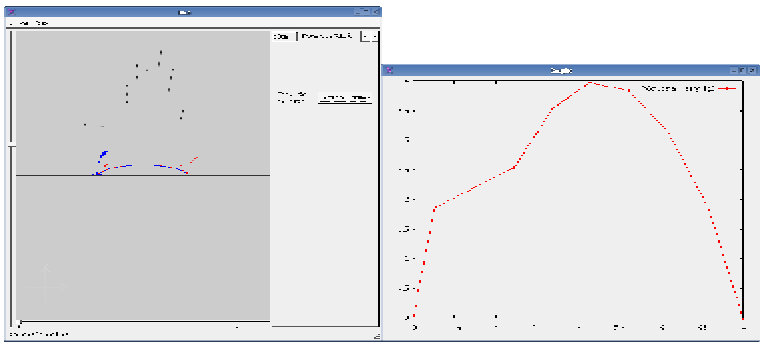


Figura 5.10 – Janelas do Sistema para o Exemplo MTB1a

Exemplo MTB1b – Salto com pose inicial $q_1 = 90^o$, $q_2 = 60^o$, $q_3 = 45^o$ e 16 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 148,444 $(m)^2$ da função objetivo em 47,68 segundos, num total de 58 passos de otimização. Na Tabela 5.11, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.10, é ilustrado o salto da estrutura

articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.12 – Resultado para o Exemplo MTB1b

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	90	60	45	3,52074	3,52075
0,125	0,302991	1,06438	76,0308	76,03	66,7161	21655,7	0,00353
0,25	0,552926	1,99336	76,0813	76,0813	71,2198	5735,74	0
0,375	0,802983	2,77955	76,1011	76,2483	76,0693	0,13533	0,97573
0,5	1,05337	3,42335	75,9502	75,9502	75,9501	5858,5	155,773
0,625	1,30401	3,9244	75,7229	75,7229	75,7229	1464,31	3051,02
0,75	1,55464	4,28264	75,4958	75,4958	75,4959	46188,9	35,6796
0,875	1,80528	4,49808	75,2688	75,2688	75,2688	945,199	140,61
1	2,05592	4,5707	75,0417	75,0417	75,0399	1765,35	0,10704
1,125	2,30655	4,50052	74,8148	74,8148	74,8107	3398,86	0,00283
1,25	2,55718	4,28753	74,5883	74,5883	74,5866	2448,94	0
1,375	2,80781	3,93173	74,3622	74,3622	74,3671	776,017	0,02094
1,5	3,05844	3,43312	74,1365	74,1365	74,1513	2818,64	0
1,625	3,30906	2,7917	73,9105	73,9105	73,9284	3284,51	0
1,75	3,55969	2,00748	73,6849	73,6849	73,7102	3760,09	0,03162
1,875	3,81025	1,08043	73,4773	73,4314	73,431	487,924	685,397
2	4	0	90	45	0	3,52074	3,52078

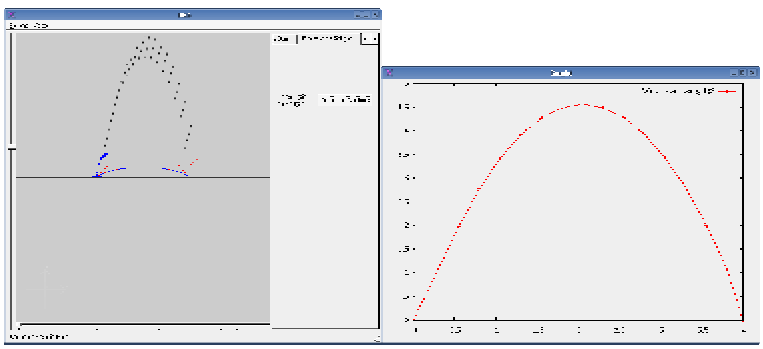


Figura 5.11 – Janelas do Sistema para o Exemplo MTB1b

Exemplo MTB1c – Salto com pose inicial $q_1 = 90^o$, $q_2 = 60^o$, $q_3 = 45^o$ e 32 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 294,612 $(m)^2$ da função objetivo em 973,09 segundos, num total de 115 passos de otimização. Na Tabela 5.13, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.12, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.13 – Resultado para o Exemplo MTB1c

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	90	60	45	0,34273	0,34281
0,0625	0,11765	0,54042	94,4214	94,4211	153,622	30443,9	0
0,125	0,24751	1,05527	92,8277	92,8277	130,148	53332,6	0
0,1875	0,37981	1,53498	90,4533	90,4532	103,155	17213,8	0,16236
0,25	0,51026	1,98109	88,739	88,7387	88,7467	11603,4	213,412
0,3125	0,63747	2,39199	88,1069	88,1069	88,1077	107845	0
0,375	0,76467	2,76721	87,4755	87,4755	87,4742	50439,5	0,25751
0,4375	0,89188	3,10676	86,8446	86,8446	86,8446	14263,1	71,3647
0,5	1,01909	3,41065	86,2138	86,2138	86,2138	1157,52	174,244
0,5625	1,14629	3,67886	85,5829	85,5829	85,5783	245,375	0,0731
0,625	1,27348	3,9114	84,953	84,953	84,953	17946	52,5089
0,6875	1,40067	4,10828	84,3231	84,3231	84,3231	61586,4	58,087
0,75	1,52786	4,26948	83,6935	83,6924	83,6848	0	0,10461
0,8125	1,65504	4,39502	83,0646	83,0646	83,0644	68699,5	21,1216
0,875	1,78222	4,48488	82,4359	82,4359	82,4342	130,546	1,68013
0,9375	1,90939	4,53907	81,8077	81,8077	81,8048	2541,74	0,85813
1	2,03655	4,5576	81,1805	81,1805	81,1803	260,522	29,9732
1,0625	2,16371	4,54045	80,5538	80,5533	80,553	0	17,8136
1,125	2,29086	4,48763	79,9273	79,9273	79,9273	179189	80,732
1,1875	2,418	4,39914	79,3017	79,3017	79,3015	176218	16,2559

1,25	2,54513	4,27497	78,6769	78,6769	78,6768	44184,8	288,8
1,3125	2,67226	4,11514	78,0527	78,0527	78,0527	37216,4	109,981
1,375	2,79937	3,91964	77,4293	77,4293	77,4284	8708,57	3,16613
1,4375	2,92648	3,68846	76,8068	76,8068	76,8068	4625,95	448,994
1,5	3,05358	3,42161	76,1852	76,1852	76,1851	325,629	181,684
1,5625	3,18066	3,11909	75,5637	75,5637	75,5538	86262,2	0,11041
1,625	3,30774	2,78089	74,9447	74,9447	74,9477	44283,4	0,91279
1,6875	3,43479	2,40702	74,3295	74,3293	74,387	154,602	0,22836
1,75	3,5618	1,99745	73,7263	73,7263	74,0058	18498	0,00119
1,8125	3,68879	1,5522	73,128	73,1227	73,6513	5657,05	3,73299
1,875	3,79483	1,06632	78,5718	64,6389	64,6291	0	106,49
1,9375	3,90036	0,547548	83,9894	56,6363	56,636	0,54582	38330,8
2	4	0	90	45	0	0,34273	0,34284

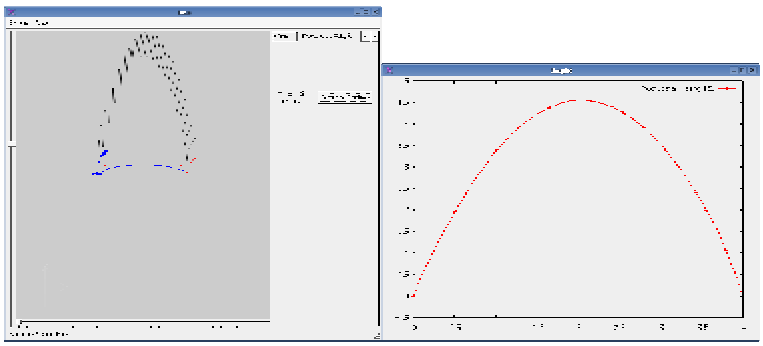


Figura 5.12 – Janelas do Sistema para o Exemplo MTB1c

Exemplo MTB2a – Salto com pose inicial $q_1 = 120^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 8 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 72,5949 $(m)^2$ da função objetivo em 8,18 segundos, num total de 76 passos de otimização. Na Tabela 5.14, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.13, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.14 – Resultado para o Exemplo MTB2a

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	120	60	45	0,16086	0,16084
0,25	0,54975	1,94874	101,593	100,287	128,43	4,23713	0,04104
0,5	1,05857	3,3761	95,2143	95,2142	95,2262	6890,6	24,7733
0,75	1,56025	4,23864	91,3372	91,3372	91,3373	774,387	7,96092
1	2,06196	4,5312	87,4582	87,4565	87,4563	0,61244	14,0968
1,25	2,56361	4,25378	83,5845	83,5845	83,5833	1419,64	5,12744
1,5	3,06504	3,40635	79,7304	79,7278	79,7381	1146,7	19,6391
1,75	3,53288	1,9845	85,1881	62,4444	55,961	0,01638	0,10951
2	4	0	90	45	0	0,16085	0,16085

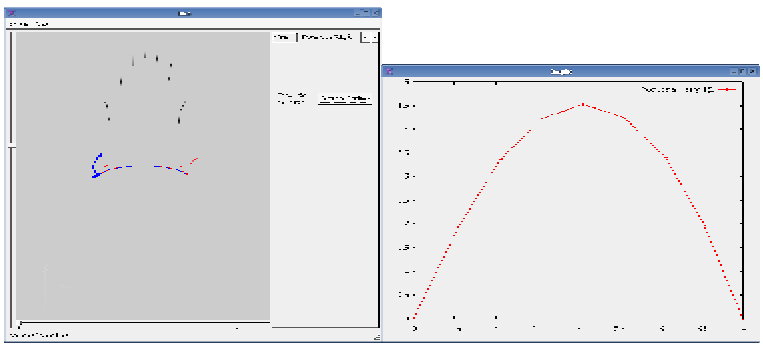


Figura 5.13 – Janelas do Sistema para o Exemplo MTB2a

Exemplo MTB2b – Salto com pose inicial $q_1 = 120^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 16 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 153,009 $(m)^2$ da função objetivo em 95,21 segundos, num total de 90 passos de otimização. Na Tabela 5.15, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.14, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.15 – Resultado para o Exemplo MTB2b

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	120	60	45	17,2059	17,2059
0,125	0,30055	1,02188	103,859	103,852	103,857	4442,93	1285,38
0,25	0,53675	1,96025	108,253	108,253	108,253	1082,8	510,313
0,375	0,77362	2,75727	112,596	112,596	112,6	420,397	21,3133
0,5	1,01127	3,41282	116,876	116,876	116,876	1812,8	792,865
0,625	1,24977	3,9268	121,089	121,089	121,089	2526,91	122,964
0,75	1,48921	4,29906	125,233	125,233	125,235	8060,02	41,8949
0,875	1,72962	4,52951	129,308	129,308	129,308	767,422	868,494
1	1,97102	4,61802	133,316	133,316	133,316	6725,39	734,383
1,125	2,21343	4,56448	137,26	137,26	137,26	3872,1	216,383
1,25	2,45686	4,36882	141,144	141,144	141,148	3926,44	20,1761
1,375	2,70134	4,03092	144,967	144,967	145,157	2075,75	0,27313
1,5	2,94676	3,55074	148,752	148,752	148,767	775,872	3,6462
1,625	3,19316	2,9282	152,503	152,501	152,124	3136,7	13,7187
1,75	3,42431	2,17581	161,561	162,223	46,2971	38,3475	0,05207
1,875	3,70363	1,19251	139,001	175,769	-207,263	0	0
2	4	0	90	45	0	17,2059	17,2059

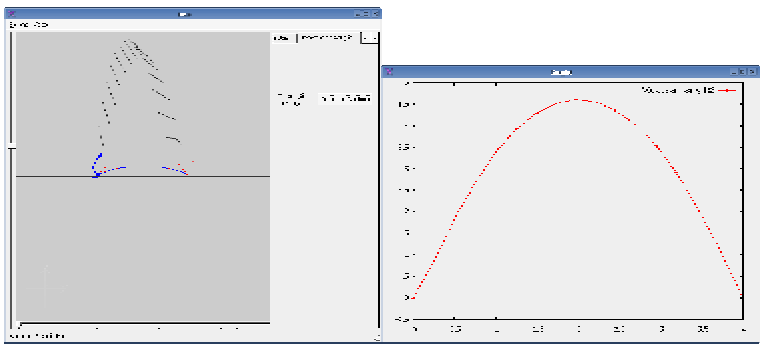


Figura 5.14 – Janelas do Sistema para o Exemplo MTB2b

Exemplo MTB2c – Salto com pose inicial $q_1 = 120^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 32 partições

Nesse exemplo, o otimizador não conseguiu atingir um valor ótimo. O valor sub-ótimo de 308,344 $(m)^2$ da função objetivo foi obtido em 1604,49 segundos, num total de 114 passos de otimização. Na Tabela 5.16, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.15, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.16 – Resultado para o Exemplo MTB2c

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	120	60	45	22,0136	22,0136
0,0625	0,17175	0,49079	103,002	102,987	102,986	978,053	3419,46
0,125	0,27716	1,00513	107,03	107,03	107,115	1549,06	12787,5
0,1875	0,38184	1,48521	111,25	111,25	111,187	1306,69	635,538
0,25	0,48448	1,93192	116,158	114,548	114,143	220,608	2977,34
0,3125	0,59649	2,33399	116,826	116,813	47,0219	7283,6	-0,16917
0,375	0,68053	2,72003	128,214	128,216	92,9745	4024,88	-0,11559
0,4375	0,77954	3,07609	136,829	136,832	136,823	1061,81	801,652
0,5	0,88283	3,39432	143,72	143,72	143,722	20215,9	1833,42
0,5625	0,98880	3,68074	150,764	150,764	155,934	29002,6	0,24077
0,625	1,09766	3,93329	157,758	157,733	165,694	241,688	-0,17999
0,6875	1,20885	4,15423	165,365	165,272	165,275	647,567	1241,55
0,75	1,32583	4,3412	172,67	172,67	172,67	40722,5	1343,88
0,8125	1,44739	4,49221	179,771	179,772	180,899	7723,66	0,964137
0,875	1,57376	4,60803	186,919	186,919	189,358	6290,04	-0,20677
0,9375	1,70499	4,68872	194,429	194,427	194,427	14595,7	2343,74
1	1,84071	4,73232	202,016	200,835	208,226	37,325	-0,20784
1,0625	1,98093	4,73739	209,438	209,439	210,162	5879,03	0,62227

1,125	2,12559	4,70509	217,314	217,33	217,341	1007,61	227,632
1,1875	2,25102	4,62315	218,999	224,65	344,006	-0,20828	-0,20828
1,25	2,34906	4,44438	199,946	199,946	214,774	2167,36	-0,20828
1,3125	2,45719	4,25694	190,746	190,748	190,75	1832,57	3698,72
1,375	2,57009	4,02992	180,884	180,884	180,884	3892,67	985,084
1,4375	2,69081	3,76762	171,199	171,198	171,198	1155,07	1536,35
1,5	2,81879	3,46972	161,295	161,296	161,313	4081,87	72,1482
1,5625	2,95426	3,13888	151,362	151,362	151,363	34070,8	2160,69
1,625	3,09695	2,77516	141,108	141,108	140,238	8385,8	3,21044
1,6875	3,24665	2,38069	130,552	130,552	130,581	39318,7	20784,8
1,75	3,40195	1,9567	119,795	119,796	119,799	4903,8	665,919
1,8125	3,56247	1,50504	108,719	108,72	105,278	19439,1	-0,20414
1,875	3,72836	1,02685	96,8536	96,7043	86,9213	184,775	0,53436
1,9375	3,88783	0,526953	87,3191	84,1909	84,1911	32,9435	1388,31
2	4	0	90	45	0	22,0136	22,0136

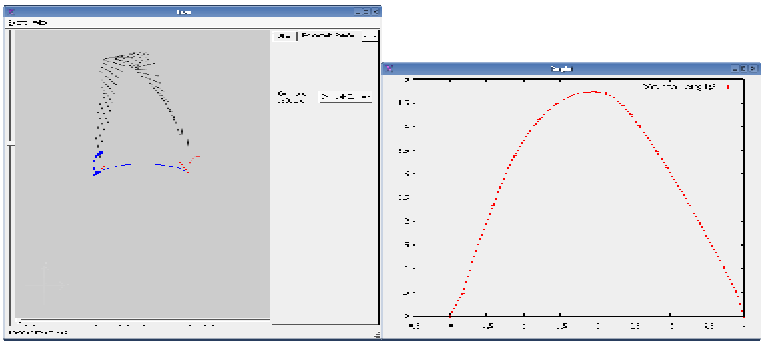


Figura 5.15 – Janelas do Sistema para o Exemplo MTB2c

Exemplo MTB3a – Salto com pose inicial $q_1 = 150^{o}$, $q_2 = 60^{o}$, $q_3 = 45^{o}$ e 8 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 70,273 $(m)^2$ da função objetivo em 4,13 segundos, num total de 34 passos de otimização. Na Tabela 5.17, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.16, é ilustrado o salto da estrutura

articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.17 – Resultado para o Exemplo MTB3a

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	150	60	45	0,30424	0,30424
0,25	0,54933	1,86455	115,363	115,359	115,523	2417,33	21,0326
0,5	1,0494	3,30317	109,5	109,5	109,51	304,869	3,95448
0,75	1,5508	4,1731	103,513	103,58	103,582	0	20,5244
1	2,05295	4,47464	97,5059	97,5059	97,5084	5120,42	6,3928
1,25	2,55575	4,20792	91,4218	91,4218	91,4224	597,225	9,86035
1,5	3,05838	3,37307	85,4041	85,2266	85,2274	13,3841	253,321
1,75	3,53424	1,96839	86,7865	69,1454	69,1222	0,07640	26,9396
2	4	0	90	45	0	0,30424	0,30424

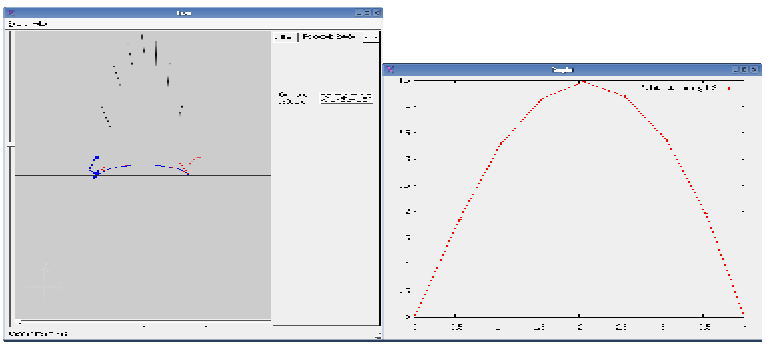


Figura 5.16 – Janelas do Sistema para o Exemplo MTB3a

Exemplo MTB3b – Salto com pose inicial $q_1 = 150^o$, $q_2 = 60^o$, $q_3 = 45^o$ e 16 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 144,622 $(m)^2$ da função objetivo em 78,93 segundos, num total de 74 passos de otimização. Na Tabela 5.18, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.17, é ilustrado o salto da estrutura

articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.18 – Resultado para o Exemplo MTB3b

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	150	60	45	0,83156	0,83156
0,125	0,15025	0,98589	141,701	141,692	141,724	4106,71	583,743
0,25	0,27415	1,92906	158,221	158,221	158,231	4726,14	83,0535
0,375	0,41777	2,7351	174,109	174,109	174,11	29264,8	166,889
0,5	0,58002	3,39971	189,83	189,83	189,829	37798,2	1266,72
0,625	0,76126	3,92188	206,635	206,635	194,952	70745,3	0,00033
0,75	0,95916	4,28921	222,516	222,516	222,516	84134,4	3055,25
0,875	1,17624	4,50441	240,248	240,248	240,234	13572,6	121,055
1	1,40926	4,5567	259,183	259,183	259,182	4168,03	2428,81
1,125	1,64999	4,43712	278,773	278,779	278,782	87,0421	261,163
1,25	1,88397	4,14323	297,787	297,788	297,788	1682,07	2805,58
1,375	2,09948	3,6857	314,603	314,604	291,168	856,509	0,60496
1,5	2,30773	3,10209	324,673	324,739	-4,27887	872,385	0,00157
1,625	2,48038	2,61374	266,741	193,2	193,163	0,08001	673,105
1,75	2,60275	1,7396	178,649	178,624	178,594	1279,76	164,543
1,875	3,2593	0,85630	128,752	128,751	128,757	13369	1958,89
2	4	0	90	45	0	0,83156	0,83156

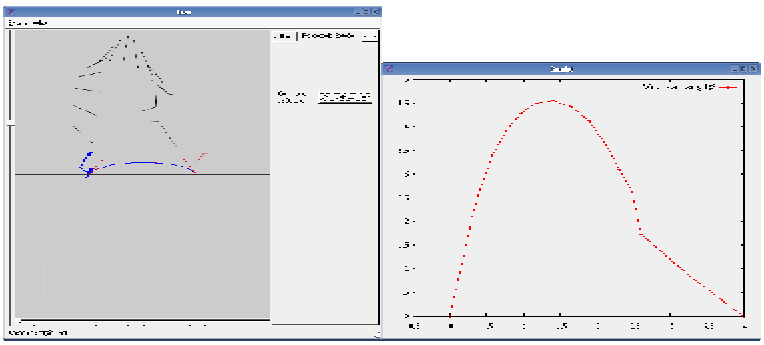


Figura 5.17 – Janelas do Sistema para o Exemplo MTB3b

Exemplo MTB3c – Salto com pose inicial $q_1 = 150^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 32 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de $283{,}031\,(m)^2$ da função objetivo em 1781,64 segundos, num total de 127 passos de otimização. Na Tabela 5.19, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.18, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.19 – Resultado para o Exemplo MTB3c

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	150	60	45	39,5167	39,5167
0,0625	0,16685	0,42781	121,315	121,279	121,497	4325,39	283,805
0,125	0,29157	0,94715	119,825	119,825	119,825	3138,15	1486,98
0,1875	0,41633	1,43097	118,35	118,35	118,35	1822,75	3842,53
0,25	0,54121	1,87924	116,866	116,866	116,866	9851,02	3844,17
0,3125	0,66618	2,29197	115,382	115,382	115,599	9601,75	0,00032
0,375	0,79126	2,66913	113,882	113,883	114,102	23,6964	0,00112
0,4375	0,91645	3,01071	112,365	112,365	112,365	89,2162	12595,6
0,5	1,04173	3,31678	110,85	110,85	110,85	13834,6	4041,66
0,5625	1,16709	3,58732	109,327	109,327	109,329	528,238	29,9534
0,625	1,29252	3,82234	107,804	107,804	107,902	6416,99	0,05121
0,6875	1,41806	4,02181	106,263	106,263	106,278	41712,4	3,21596
0,75	1,54366	4,18577	104,72	104,72	104,72	9798,98	370,979
0,8125	1,66932	4,31421	103,172	103,172	103,172	4249,05	1535,42
0,875	1,79505	4,40714	101,62	101,62	101,62	8801,47	9887,19
0,9375	1,92083	4,46457	100,062	100,062	100,062	12151,1	2015,55
1	2,04666	4,48648	98,5002	98,5002	98,5003	81560,9	375,504
1,0625	2,17253	4,4729	96,9349	96,9349	96,9349	50719,2	2051,62
1,125	2,29844	4,42381	95,3666	95,3666	95,3666	7154,02	2377,67
1,1875	2,42438	4,33923	93,796	93,796	93,796	20396,9	2182,93

1,25	2,55033	4,21915	92,2237	92,2237	92,2237	854,018	9097,85
1,3125	2,67629	4,06358	90,6505	90,6505	90,6505	29,8097	102,099
1,375	2,80226	3,87251	89,0768	89,0767	89,0741	10,0103	1,05858
1,4375	2,92823	3,64594	87,5032	87,5032	87,4951	2774,88	0,853876
1,5	3,05418	3,38388	85,9302	85,9302	85,9112	4385,68	0,307233
1,5625	3,1801	3,08633	84,3617	84,3613	84,3613	7,75306	391,416
1,625	3,306	2,75327	82,7937	82,7937	82,7937	2306,47	15978,5
1,6875	3,43187	2,38472	81,229	81,229	81,229	5111,87	6230,08
1,75	3,55769	1,98066	79,6679	79,6679	79,667	2256,29	27,8757
1,8125	3,68347	1,5411	78,1112	78,1112	78,1112	327,561	1484,79
1,875	3,80918	1,06603	76,5594	76,5593	76,5592	323,084	2742,34
1,9375	3,93482	0,55545	75,0166	75,0078	75,0078	10149,5	28697,8
2	4	0	90	45	0	39,5167	39,5167

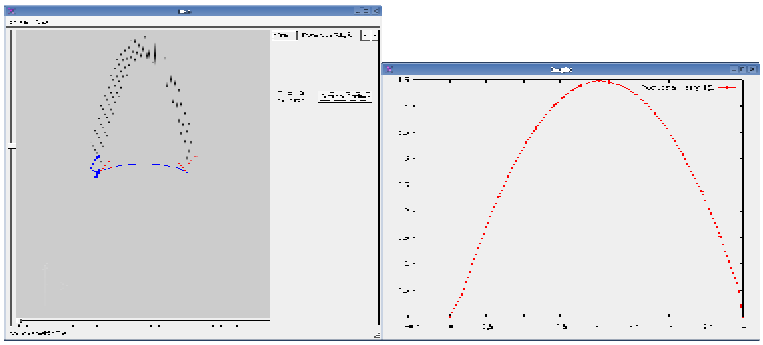


Figura 5.18 – Janelas do Sistema para o Exemplo MTB3c

5.2.3 Minimização da Trajetória em relação ao centro de massa

Em todos os exemplos dessa seção, o ponto inicial do espaço de busca fornecido para o otimizador é obtido incluindo os valores das poses inicial e final, e os valores das poses intermediárias. Os valores das poses intermediárias são definidos colocando-se o centro de massa da estrutura articulada acompanhando a curva trajetória especificada e as orientações por interpolação linear entre as orientações das poses inicial e final.

Exemplo MTC1a – Salto com pose inicial $q_1 = 90^o$, $q_2 = 60^o$, $q_3 = 45^o$ e 8 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 82,0118 $(m)^2$ da função objetivo em 17,12 segundos, num total de 77 passos de otimização. Na Tabela 5.20, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.19, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.20 – Resultado para o Exemplo MTC1a

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	90	60	45	0,58907	0,58906
0,25	0,53140	1,98755	82,6026	82,5976	117,28	490,314	0
0,5	1,04154	3,41824	79,2768	78,0597	82,7137	0,10771	0,03327
0,75	1,54718	4,27869	77,547	77,546	77,5465	231,195	113,684
1	2,04992	4,56734	76,703	76,703	76,7018	81,003	0,46962
1,25	2,55264	4,2848	75,861	75,861	75,8584	101,452	0,26009
1,5	3,05535	3,43107	75,0201	75,0199	75,0059	26,1799	0,004
1,75	3,55799	2,00613	74,1951	74,1661	74,1659	193,079	126,706
2	4	0	90	45	0	0,58907	0,58907

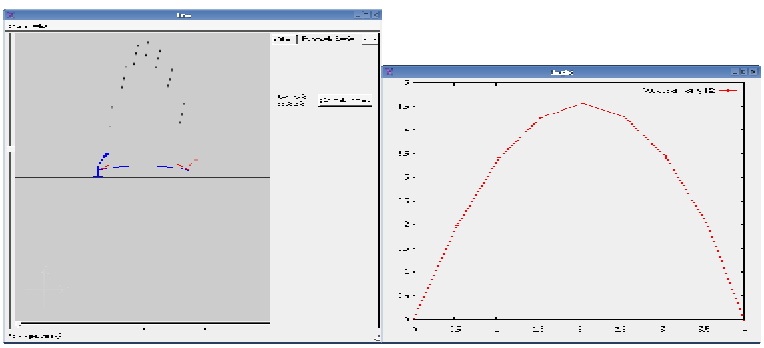


Figura 5.19 – Janelas do Sistema para o Exemplo MTC1a

Exemplo MTC1b – Salto com pose inicial $q_1 = 90^o$, $q_2 = 60^o$, $q_3 = 45^o$ e 16 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 163,848 $(m)^2$ da função objetivo em 50 segundos, num total de 50 passos de otimização. Na Tabela 5.21, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.20, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.21 – Resultado para o Exemplo MTC1b

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	90	60	45	3,00022	3,00021
0,125	0,30129	1,06275	76,6277	76,6275	76,6275	123011	6855,15
0,25	0,55243	1,99186	76,2866	76,2866	76,2866	10333,5	322,074
0,375	0,80357	2,77817	75,946	75,9458	75,9458	0,00165	60,1873
0,5	1,05471	3,42168	75,6053	75,6051	75,6016	1,72642	0,03338
0,625	1,30584	3,92239	75,2653	75,2653	75,2652	452,149	5,39758
0,75	1,55696	4,28029	74,9254	74,9254	74,9254	7781,21	301,274
0,875	1,80808	4,49539	74,5858	74,5858	74,5857	6283,9	11,4676
1	2,0592	4,56768	74,2464	74,2464	74,2464	6729,53	1344,87
1,125	2,31032	4,49718	73,9074	73,9074	73,9074	9427,14	84,7966
1,25	2,56143	4,28386	73,5687	73,5687	73,5687	3595,04	178,317
1,375	2,81254	3,92775	73,2302	73,2302	73,2302	6868,97	45,2374
1,5	3,06365	3,42883	72,8921	72,8921	72,8921	357,83	1257,85
1,625	3,31475	2,7871	72,5543	72,5543	72,5543	10365,6	3604,67
1,75	3,56584	2,00258	72,2182	72,215	72,2157	3685,07	1096,49
1,875	3,78297	1,06664	81,7413	59,2593	59,0359	0,05652	13,7562
2	4	0	90	45	0	3,0002	3,00022

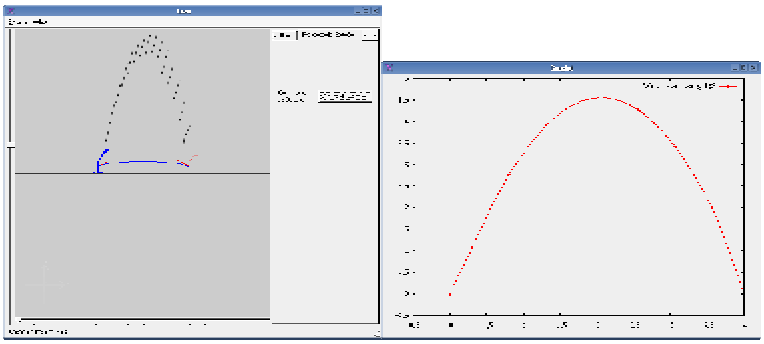


Figura 5.20 – Janelas do Sistema para o Exemplo MTC1b

Exemplo MTC1c – Salto com pose inicial $q_1 = 90^o$, $q_2 = 60^o$, $q_3 = 45^o$ e 32 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 327,476 $(m)^2$ da função objetivo em 722,12 segundos, num total de 68 passos de otimização. Na Tabela 5.22, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.21, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.22 – Resultado para o Exemplo MTC1c

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	90	60	45	0,20361	0,20361
0,0625	0,17573	0,544685	76,7775	76,7539	76,509	2774,55	2,87665
0,125	0,30139	1,06271	76,5937	76,5937	76,5936	78843,7	422,253
0,1875	0,42705	1,54505	76,4028	76,4028	76,4028	12279,2	149,849
0,25	0,55272	1,9917	76,2119	76,2119	76,2119	8683,91	8343,67
0,3125	0,67838	2,40264	76,0211	76,0211	76,0211	16048,6	46,3084
0,375	0,80404	2,77788	75,8305	75,8305	75,8304	232518	46,3614
0,4375	0,92970	3,11743	75,6398	75,6398	75,6398	107414	4413,04
0,5	1,05536	3,42127	75,4493	75,4493	75,4493	30023	30753,8
0,5625	1,18102	3,68941	75,2589	75,2589	75,2589	298518	2371,14
0,625	1,30668	3,92185	75,0685	75,0685	75,0685	8888,62	31,376

0,6875	1,43233	4,1186	74,8782	74,8782	74,8767	9279,36	0,00747
0,75	1,55799	4,27964	74,6881	74,6881	74,6881	136287	2733,17
0,8125	1,68364	4,40498	74,498	74,498	74,498	39987,1	33,2273
0,875	1,80929	4,49462	74,308	74,308	74,308	14746,6	3510,9
0,9375	1,93495	4,54856	74,1181	74,1181	74,1181	64470,8	32,5812
1	2,0606	4,5668	73,9283	73,9283	73,9283	51231,1	45,9312
1,0625	2,18625	4,54934	73,7384	73,7384	73,7356	64964,7	0,03359
1,125	2,31189	4,49618	73,5488	73,5488	73,5458	100687	0
1,1875	2,43754	4,40732	73,3595	73,3595	73,3594	738050	168,912
1,25	2,56319	4,28276	73,17	73,17	73,17	329123	200,131
1,3125	2,68883	4,1225	72,9807	72,9807	72,9807	500886	210,514
1,375	2,81448	3,92653	72,7915	72,7915	72,7914	13289,2	22,8405
1,4375	2,94012	3,69487	72,6023	72,6023	72,6023	74299,1	16006,5
1,5	3,06576	3,42751	72,4133	72,4133	72,4133	119416	2721,69
1,5625	3,1914	3,12445	72,2243	72,2243	72,2243	1756,53	223,829
1,625	3,31704	2,78568	72,0355	72,0355	72,0355	154653	11064,1
1,6875	3,44268	2,41122	71,8467	71,8467	71,8467	132799	33,1146
1,75	3,56831	2,00105	71,6581	71,6581	71,6581	1619,16	125,923
1,8125	3,69395	1,55519	71,4698	71,4693	71,47	67606,5	3144,76
1,875	3,79668	1,06754	77,9565	62,7973	62,7965	0	1488,61
1,9375	3,89879	0,54789	84,3741	54,6414	54,6373	0,268476	2745,01
2	4	0	90	45	0	0,203611	0,20360

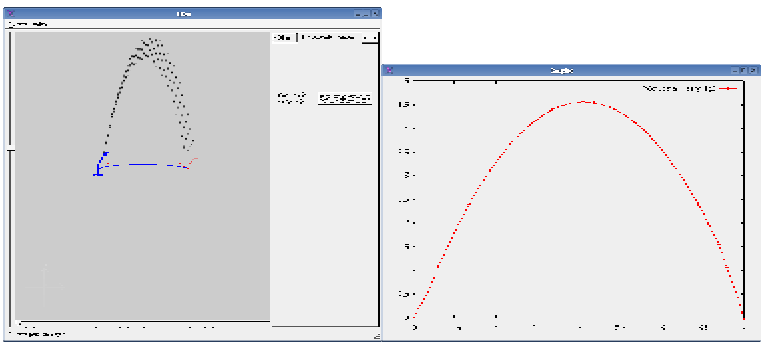


Figura 5.21 – Janelas do Sistema para o Exemplo MTC1c

Exemplo MTC2a – Salto com pose inicial $q_1 = 120^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 8 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 19,2931 $(m)^2$ da função objetivo em 25,52 segundos, num total de 271 passos de otimização. Na Tabela 5.23, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.22, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.23 – Resultado para o Exemplo MTC2a

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	120	60	45	2,5348	2,53488
0,25	0,38213	1,34733	105,633	105,711	160,232	12,2301	0
0,5	0,80240	2,13658	78,5882	121,214	118,389	0,07142	1,35317
0,75	1,28992	2,40757	23,8688	122,748	-287,829	0,75059	0
1	1,63558	2,38014	14,9433	361,885	322,803	0	0
1,25	2,13175	1,55503	83,6455	84,4307	76,4681	44,118	4,3549
1,5	2,77343	1,60672	80,8904	80,3772	80,416	3,50657	2,18539
1,75	3,39527	1,08591	83,5446	67,6614	67,218	0,11689	1,28224
2	4	0	90	45	0	2,53491	2,53486

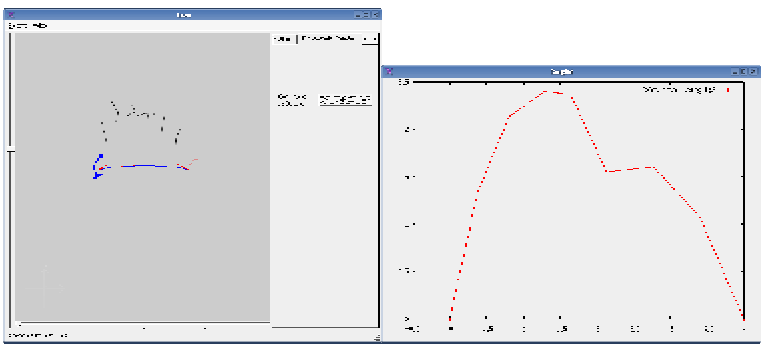


Figura 5.22 – Janelas do Sistema para o Exemplo MTC2a

Exemplo MTC2b – Salto com pose inicial $q_1 = 120^o$, $q_2 = 60^o$, $q_3 = 45^o$ e 16 partições

Nesse exemplo, o otimizador não conseguiu atingir um valor ótimo. O resultado impraticável foi obtido em 238,57 segundos, num total de 118 passos de otimização.

Exemplo MTC2c – Salto com pose inicial $q_1 = 120^o$, $q_2 = 60^o$, $q_3 = 45^o$ e 32 partições

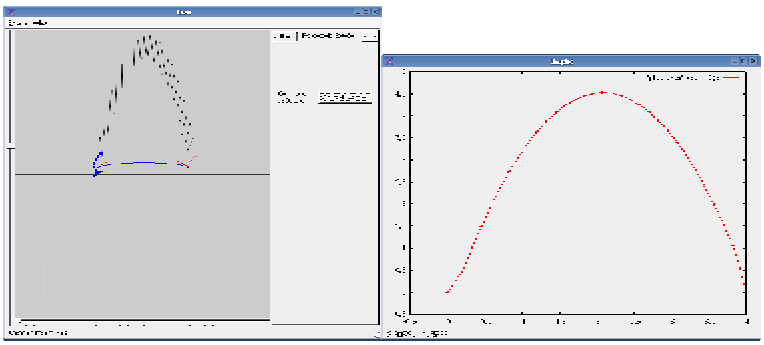
Nesse exemplo, o otimizador atingiu o valor ótimo de 324,134 $(m)^2$ da função objetivo em 954,01 segundos, num total de 88 passos de otimização. Na Tabela 5.24, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.23, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.24 – Resultado para o Exemplo MTC2c

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	120	60	45	4,30317	4,30318
0,0625	0,20055	0,48833	96,5788	96,5736	96,7088	23167,3	133,008
0,125	0,32535	1,00746	95,7664	95,7664	95,7795	13515,9	0,93715
0,1875	0,45013	1,49094	94,9643	94,9643	94,9643	32137,1	1763,19
0,25	0,57491	1,93878	94,1647	94,161	94,165	3,27413	0
0,3125	0,69970	2,35096	93,3608	93,3608	93,3608	11742,5	1297,75
0,375	0,82449	2,7275	92,5584	92,5584	92,5584	320,943	2362,12
0,4375	0,94928	3,06838	91,7557	91,7557	91,7549	467346	46,3477
0,5	1,07408	3,37362	90,9529	90,9529	90,9529	9100,6	225568
0,5625	1,19888	3,64321	90,1499	90,1499	90,1499	486,709	43319,2
0,625	1,32368	3,87715	89,3469	89,3469	89,3469	514,959	27,9415
0,6875	1,44847	4,07544	88,544	88,544	88,544	270688	2265,43

0,75	1,57327	4,23808	87,7412	87,7412	87,7412	32757,5	3128,37
0,8125	1,69806	4,36507	86,9387	86,9387	86,9387	11109,9	1181,32
0,875	1,82285	4,45642	86,1358	86,1358	86,1268	262666	0,00015
0,9375	1,94763	4,51211	85,3348	85,3348	85,3348	152873	94,9749
1	2,07241	4,53216	84,5337	84,5337	84,5336	466961	215,15
1,0625	2,19718	4,51655	83,7329	83,7329	83,7306	336819	1,43324
1,125	2,32194	4,4653	82,9332	82,9332	82,9331	925,455	63,0821
1,1875	2,44669	4,37839	82,1341	82,1341	82,1336	60650,8	9,10106
1,25	2,57143	4,25584	81,3359	81,3359	81,3358	900945	64,479
1,3125	2,69616	4,09763	80,5387	80,5387	80,5386	680,545	199,602
1,375	2,82088	3,90378	79,7424	79,7424	79,7424	360,367	220,28
1,4375	2,94559	3,67427	78,9473	78,9473	78,9465	4537,42	8,76517
1,5	3,07028	3,40911	78,1533	78,1533	78,1526	12100,2	10,047
1,5625	3,19496	3,10829	77,3606	77,3606	77,3596	34801,3	7,54437
1,625	3,31962	2,77183	76,5693	76,5693	76,5693	277520	192,036
1,6875	3,44426	2,39971	75,7793	75,7793	75,778	1907,27	6,19641
1,75	3,56889	1,99194	74,9909	74,9908	74,9908	263,418	157,374
1,8125	3,6935	1,54851	74,2056	74,2056	74,2313	21948	23,5713
1,875	3,81747	1,069	73,6641	73,6634	77,3476	65994,1	0,66196
1,9375	3,90988	0,547018	82,0934	60,309	60,3053	0,36738	3100,27
2	4	0	90	45	0	4,30319	4,30318

Figura 5.23 – Janelas do Sistema para o Exemplo MTC2c



Exemplo MTC3a – Salto com pose inicial $q_1 = 150^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 8 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 79,7566 $(m)^2$ da função objetivo em 4,79 segundos, num total de 35 passos de otimização. Na Tabela 5.25, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.24, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.25 – Resultado para o Exemplo MTC3a

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	150	60	45	3,61718	3,61718
0,25	0,51094	1,94863	136,093	94,4992	122,086	0	0,01438
0,5	1,05532	3,32286	111,206	110,953	110,989	21,2652	29,1265
0,75	1,5549	4,1904	105,08	105,08	105,845	0,371161	0,01842
1	2,05541	4,48949	98,8816	98,8817	98,8822	747,459	53,1461
1,25	2,55645	4,22056	92,6783	92,6811	92,6811	2,46949	29,98
1,5	3,05788	3,38359	86,3974	86,3972	85,6587	152,265	0,01221
1,75	3,55864	1,97865	80,27	80,2619	80,2635	677,887	59,4043
2	4	0	90	45	0	3,61718	3,61718

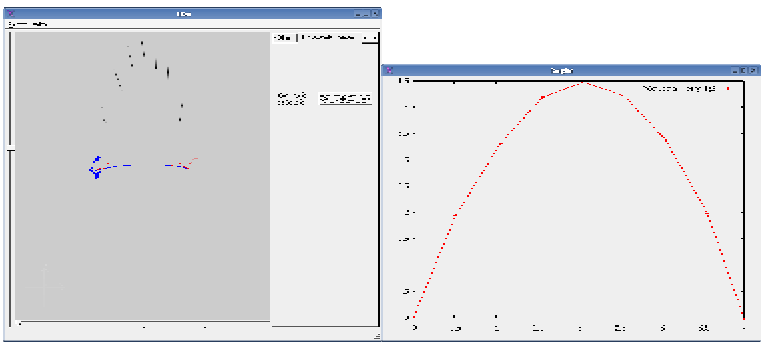


Figura 5.24 – Janelas do Sistema para o Exemplo MTC3a

Exemplo MTC3b – Salto com pose inicial $q_1 = 150^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 16 partições

Nesse exemplo, o otimizador atingiu o valor ótimo de 158,897 $(m)^2$ da função objetivo em 58,1 segundos, num total de 64 passos de otimização. Na Tabela 5.26, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.25, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.26 – Resultado para o Exemplo MTC3b

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	150	60	45	362,445	362,445
0,125	0,29384	0,94244	119,344	119,325	119,389	2136	236,726
0,25	0,54340	1,87489	116,417	116,417	116,417	58864,4	7231,16
0,375	0,79333	2,66512	113,47	113,47	113,47	9034,56	4319,05
0,5	1,04363	3,31316	110,493	110,493	110,493	242164	32300
0,625	1,2943	3,81902	107,479	107,5	107,5	0	3782,92
0,75	1,54521	4,18278	104,459	104,459	104,459	32037,3	2306,75
0,875	1,79641	4,40444	101,408	101,408	101,408	98966,8	2104,16
1	2,04782	4,48404	98,3387	98,3387	98,3388	72471,8	175,832
1,125	2,29937	4,4216	95,258	95,2535	95,2536	0	1543,31
1,25	2,55103	4,21714	92,1701	92,1701	92,2357	80708,4	0,00896
1,375	2,80276	3,87067	89,0695	89,0695	89,0695	28152,3	2278,33
1,5	3,05445	3,38219	85,976	85,976	85,976	33486,1	3456,51
1,625	3,30606	2,75171	82,8893	82,8893	82,8893	34242,2	13968,6
1,75	3,55754	1,9792	79,8144	79,8144	79,8143	711660	228,775
1,875	3,80882	1,06464	76,7568	76,7549	76,7563	12085,6	107,269
2	4	0	90	45	0	362,445	362,445

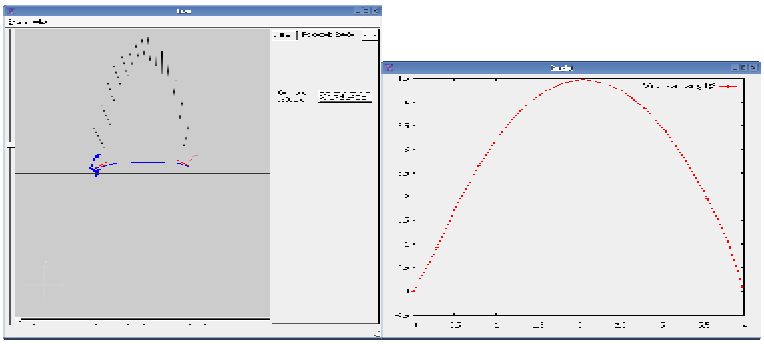


Figura 5.25 – Janelas do Sistema para o Exemplo MTC3b

Exemplo MTC3c – Salto com pose inicial $q_1 = 150^{\circ}$, $q_2 = 60^{\circ}$, $q_3 = 45^{\circ}$ e 32 partições

Nesse exemplo, o otimizador não conseguiu atingir um valor ótimo. O valor sub-ótimo de 328,037 $(m)^2$ da função objetivo foi obtido em 1398,27 segundos, num total de 81 passos de otimização. Na Tabela 5.27, são mostrados os valores das variáveis que definem a pose da estrutura articulada e as variáveis de rigidez para cada instante da partição do tempo de animação. Na Figura 5.26, é ilustrado o salto da estrutura articulada indicando as diversas poses assumidas pela estrutura ao longo da trajetória. Também é ilustrada, na forma de uma curva contínua, a trajetória da base da estrutura articulada.

Tabela 5.27 – Resultado para o Exemplo MTC3c

Tempo (s)	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	0	0	150	60	45	15.9273	15.9273
0,0625	0.18696	0.401054	115.43	115.253	115.257	189.285	14198.1
0,125	0.34573	0.897823	103.879	112.417	111.208	0.80932	5.58166
0,1875	0.58429	1.31857	73.8053	75.4011	75.3857	-0.35762	3853.43
0,25	0.64524	1.78519	85.9284	85.8567	85.8568	62.9097	17216.5
0,3125	0.71653	2.21845	95.8744	95.8739	95.874	18485.9	50200.9
0,375	0.80374	2.61633	102.606	102.605	103.541	14182.8	0.758127
0,4375	0.89884	2.97662	107.555	107.068	107.076	-0.265823	86.594
0,5	0.99758	3.2997	111.564	111.564	111.564	58254.2	8673.82

0,5625	1.10402	3.58411	113.762	113.717	113.718	-0.35629	507.62
0,625	1.21164	3.83124	115.542	115.541	115.542	9033.35	7077.89
0,6875	1.32344	4.04149	116.372	116.371	116.371	14773.7	1771.99
0,75	1.43666	4.21387	116.672	116.672	116.673	71052.6	7772.86
0,8125	1.55542	4.348	115.6	115.575	115.576	51.7139	382.54
0,875	1.67467	4.44562	114.326	114.335	114.338	10282.8	362.336
0,9375	1.79761	4.50657	112.239	111.911	111.863	333.473	5771.26
1	1.92266	4.53042	109.542	109.542	112.043	44183.2	0.719031
1,0625	2.05089	4.51866	106.323	107.924	126.644	-0.13252	-0.21574
1,125	2.17776	4.47165	103.769	103.683	150.081	3.22189	-0.35679
1,1875	2.3246	4.38035	95.5552	95.5528	54.4956	21711.8	-0.35763
1,25	2.42108	4.26569	103.714	104.644	102.702	-0.34979	0.455013
1,3125	2.53284	4.11771	107.349	107.348	106.306	6087.49	3.39742
1,375	2.64834	3.93362	110.115	108.858	108.342	113.089	12.2528
1,4375	2.77735	3.71314	109.613	109.611	105.97	6287.95	0.430785
1,5	2.90706	3.45214	108.513	108.651	91.54	106.276	-0.21764
1,5625	3.04294	3.15671	106.099	108.695	102.239	0.17028	-0.34000
1,625	3.18389	2.82348	102.05	102.05	91.4482	223778	-0.31712
1,6875	3.30554	2.4646	105.853	108.743	225.5	453.304	-0.35230
1,75	3.39464	2.03548	115.283	115.284	139.027	45114.8	0.510063
1,8125	3.53961	1.56999	109.893	109.894	117.235	6300.42	-0.11187
1,875	3.69598	1.08047	101.61	101.604	101.609	1015.49	337.872
1,9375	3.85488	0.550364	93.0299	93.0279	93.0293	55237	16035.8
2	4	0	90	45	0	15.9273	15.9273

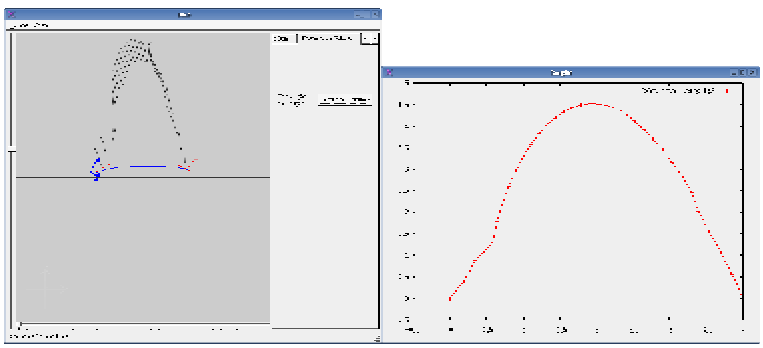


Figura 5.26 – Janelas do Sistema para o Exemplo MTC3c

5.3 Análise dos Resultados

A partir dos resultados na Seção 5.1, podem ser feitas algumas observações a cerca do controle da animação, do realismo obtido na animação e do tempo necessário para gerar a animação.

5.3.1 Observações sobre controle da animação

Nas tabelas 5.28-5.33 são feitas comparações com os resultados obtidos na seção 5.1. Essas comparações são feitas entre os resultados dos exemplos que possuem mesma configuração de pose inicial e mesma função objetivo. Em termos gerais, o aumento da discretização fornece melhores resultados (figuras 5.27-5.41), a um custo computacional significativamente mais elevado como discutido na Seção 5.2.3. Nem todos os resultados dos exemplos descritos na Seção 5.1 foram ótimos ou apresentaram animações realísticas. Por exemplo, no MTC2b, o sistema não conseguiu realizar a otimização, levando a um resultado impraticável que violou as restrições.

Tabela 5.28 – Comparação entre intervalos no tempo igual entre os exemplos MPA1a, MPA1b e MPA1c

Tempo (s)	Exemplo	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	MPA1a	0	0	90	60	45	0	0,01959
	MPA1b	0	0	90	60	45	0,0001	0,00151
	MPA1c	0	0	90	60	45	0	0,00385553
0,25	MPA1a	0,52577	1,99594	83,3024	67,9187	67,8788	0,00123	6,479
	MPA1b	0,550844	1,99489	76,4781	75,918	75,9151	14,0556	6,25393
	MPA1c	0,52189	1,99618	84,3329	65,8523	65,8513	0	23,1532
0,5	MPA1a	1,05218	3,42617	76,0223	76,0193	76,0446	689,914	4,4912
	MPA1b	1,05508	3,42392	75,3117	75,3079	75,3116	38,5113	2,80741
	MPA1c	1,04314	3,42499	78,5406	71,6393	74,7099	0	0
0,75	MPA1a	1,55367	4,28493	75,5559	75,5559	75,556	85,1102	0,95084

	MPA1b	1,55844	4,28161	74,3876	74,3855	74,3857	12,4198	7,06366
	MPA1c	1,55628	4,28375	74,9001	74,6104	79,4642	0	0
1	MPA1a	2,05515	4,57246	75,0931	75,0931	75,0929	54,3759	0,29571
	MPA1b	2,06163	4,56808	73,5086	73,4102	73,4085	0	0,43616
	MPA1c	2,06072	4,57067	73,5845	73,515	73,5356	0	10,1528
1,25	MPA1a	2,55661	4,28875	74,6316	74,6316	74,6415	333,621	0
	MPA1b	2,5648	4,28337	72,6306	72,4402	72,4445	1,17839	3,76404
	MPA1c	2,56403	4,28582	72,6747	72,6742	72,7665	0	0
1,5	MPA1a	3,05807	3,43381	74,1718	74,1716	74,2093	140,594	0,03080
	MPA1b	3,05866	3,42513	74,4531	68,0699	68,321	0,25523	0
	MPA1c	3,06722	3,42975	71,7975	71,7969	71,9675	0	0
1,75	MPA1a	3,5591	2,00755	73,8313	73,5363	73,4534	18,7888	0,64727
	MPA1b	3,53195	1,99384	81,9783	56,6115	55,0431	0,02965	0,70065
	MPA1c	3,57028	2,00247	70,9496	70,8902	71,1626	83,3159	1,43867
2	MPA1a	4	0	90	45	0	0	0,00001
	MPA1b	4	0	90	45	0	0	0
	MPA1c	4	0	90	45	0	0,00054	0,00033

Tabela 5.29 – Comparação entre intervalos no tempo igual entre os exemplos MPA2a, MPA2b e MPA2c

Tempo (s)	Exemplo	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	MPA2a	0	0	120	60	45	0	0,00064
	MPA2b	0	0	120	60	45	0,00157	6,06752
	MPA2c	0	0	120	60	45	0	2493,26
0,25	MPA2a	0,52079	1,97964	110,116	67,5813	65,9869	0	0,10892
	MPA2b	0,54302	1,96893	103,293	77,0887	66,7407	0	0
	MPA2c	0,57586	1,95287	93,981	93,981	93,981	2817,12	2749,98
0,5	MPA2a	1,04653	3,3947	99,2223	75,9096	75,9237	0	6,27277
	MPA2b	1,0739	3,39133	91,3608	91,335	102,462	128,887	0
	MPA2c	1,07331	3,38702	91,1869	91,1869	91,189	7119,69	0,00366
0,75	MPA2a	1,57237	4,24707	88,8783	86,8117	104,718	1,27979	0,00507

	MPA2b	1,57196	4,25391	88,3135	88,3175	95,0642	0	0
	MPA2c	1,57077	4,25058	88,3909	88,3909	88,3908	1355,8	104,892
1	MPA2a	2,07113	4,53751	85,5416	81,7115	87,0795	0,243162	0,02823
	MPA2b	2,06999	4,54611	85,2637	85,2494	87,4089	0	0
	MPA2c	2,0682	4,54354	85,5972	85,5972	85,5972	150,434	974,641
1,25	MPA2a	2,55822	4,25755	85,577	74,2916	77,3749	0,05081	0
	MPA2b	2,56737	4,2678	82,4178	82,4104	82,4113	23,3712	138,792
	MPA2c	2,56557	4,26591	82,8095	82,8095	82,8095	5073,1	8940,55
1,5	MPA2a	3,0389	3,40722	87,3422	64,626	64,5298	0,00883	0,49431
	MPA2b	3,0625	3,41826	80,3208	80,3208	87,3211	736,44	0
	MPA2c	3,06282	3,41767	80,0315	80,0315	80,0315	21401,8	17903,8
1,75	MPA2a	3,5173	1,98701	89,7126	54,9495	54,3082	0,00365	1,04869
	MPA2b	3,55744	1,99784	78,2627	78,2512	92,4736	51,6414	0
	MPA2c	3,55848	1,99815	77,79	77,79	84,7291	7332	0
2	MPA2a	4	0	90	45	0	0	0
	MPA2b	4	0	90	45	0	0	0
	MPA2c	4	0	90	45	0	0	0,0001

Tabela 5.30 – Comparação entre intervalos no tempo igual entre os exemplos MPA3a, MPA3b e MPA3c

Tempo (s)	Exemplo	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	MPA3a	0	0	150	60	45	0,00013	0
	MPA3b	0	0	150	60	45	0,00074	0
	MPA3c	0	0	150	60	45	0	0
0,25	MPA3a	0,50738	1,96357	136,41	65,8522	61,7676	0	0,02918
	MPA3b	0,49139	1,99105	143,45	56,7697	56,0846	0,00059	0
	MPA3c	0,49096	1,99173	143,654	56,6535	56,6302	0	0
0,5	MPA3a	1,02513	3,36342	122,093	72,9429	72,0676	0,00035	0,07551
	MPA3b	0,98624	3,41124	136,037	53,4798	52,9873	0	0
	MPA3c	0,98563	3,41203	136,281	53,2478	53,2465	0	6,8992
0,75	MPA3a	1,5514	4,20372	107,308	80,9965	80,4821	0,00049	0,05270
	MPA3b	1,48366	4,26281	128,652	50,5483	50,4936	0	0,24064

	MPA3c	1,48297	4,26355	128,885	50,2299	50,1802	0	0
1	MPA3a	2,08289	4,48774	92,2758	89,9513	90,4511	1,50321	0,70459
	MPA3b	1,98335	4,54594	121,247	47,9995	47,9857	0	0,69782
	MPA3c	1,98256	4,54664	121,493	47,6144	47,6087	0	2,08277
1,25	MPA3a	2,57574	4,21908	88,1573	83,9587	84,5101	0,28089	0,15620
	MPA3b	2,48478	4,26098	113,878	45,8173	45,77	0,00019	0,73485
	MPA3c	2,48402	4,26153	114,098	45,4092	45,4085	0	11,7318
1,5	MPA3a	3,05543	3,38092	87,6436	72,7382	71,0985	0,04249	0,16951
	MPA3b	2,98807	3,40802	106,287	44,1375	39,1337	0	0,02908
	MPA3c	2,98694	3,4084	106,702	43,6178	43,6175	0	27,4838
1,75	MPA3a	3,53043	1,97511	87,9895	57,3762	31,3819	0,01213	0,00501
	MPA3b	3,49369	1,98749	98,159	43,699	22,1685	0	0,00297
	MPA3c	3,49091	1,98741	99,2934	42,2414	42,2121	0	44,5545
2	MPA3a	4	0	90	45	0	0,00054	0
	MPA3b	4	0	90	45	0	0	0,00015
	MPA3c	4	0	90	45	0	0	0

Tabela 5.31 – Comparação entre intervalos no tempo igual entre os exemplos MTB1a, MTB1b e MTB1c

Tempo (s)	Exemplo	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	MTB1a	0	0	90	60	45	6,90042	6,90034
	MTB1b	0	0	90	60	45	3,52074	3,52075
	MTB1c	0	0	90	60	45	0,34273	0,34281
0,25	MTB1a	0,25830	1,86115	177,8	178,066	423,129	176,661	0,05087
	MTB1b	0,552926	1,99336	76,0813	76,0813	71,2198	5735,74	0
	MTB1c	0,51026	1,98109	88,739	88,7387	88,7467	11603,4	213,412
0,5	MTB1a	1,22304	2,54606	74,5196	101,979	69,2029	0	0,13932
	MTB1b	1,05337	3,42335	75,9502	75,9502	75,9501	5858,5	155,773
	MTB1c	1,01909	3,41065	86,2138	86,2138	86,2138	1157,52	174,244
0,75	MTB1a	1,68633	3,54666	77,5708	98,5898	98,5848	0	44,1463
	MTB1b	1,55464	4,28264	75,4958	75,4958	75,4959	46188,9	35,6796
	MTB1c	1,52786	4,26948	83,6935	83,6924	83,6848	0	0,10461

1	MTB1a	2,1522	3,97828	79,3383	89,9529	89,9529	0	97,0253
	MTB1b	2,05592	4,5707	75,0417	75,0417	75,0399	1765,35	0,10704
	MTB1c	2,03655	4,5576	81,1805	81,1805	81,1803	260,522	29,9732
1,25	MTB1a	2,61755	3,83989	81,2229	81,2346	80,8939	119,912	0,45455
	MTB1b	2,55718	4,28753	74,5883	74,5883	74,5866	2448,94	0
	MTB1c	2,54513	4,27497	78,6769	78,6769	78,6768	44184,8	288,8
1,5	MTB1a	3,0985	3,13347	78,6952	78,7037	78,6601	2,25166	0,15143
	MTB1b	3,05844	3,43312	74,1365	74,1365	74,1513	2818,64	0
	MTB1c	3,05358	3,42161	76,1852	76,1852	76,1851	325,629	181,684
1,75	MTB1a	3,57934	1,85633	76,1637	76,1601	76,1603	1567,72	169,58
	MTB1b	3,55969	2,00748	73,6849	73,6849	73,7102	3760,09	0,03162
	MTB1c	3,5618	1,99745	73,7263	73,7263	74,0058	18498	0,00119
2	MTB1a	4	0	90	45	0	6,90035	6,90044
	MTB1b	4	0	90	45	0	3,52074	3,52078
	MTB1c	4	0	90	45	0	0,34273	0,34284

Tabela 5.32 – Comparação entre intervalos no tempo igual entre os exemplos MTB3a, MTB3b e MTB3c

Tempo (s)	Exemplo	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	MTB3a	0	0	150	60	45	0,30424	0,30424
	MTB3b	0	0	150	60	45	0,83156	0,83156
	MTB3c	0	0	150	60	45	39,5167	39,5167
0,25	MTB3a	0,54933	1,86455	115,363	115,359	115,523	2417,33	21,0326
	MTB3b	0,27415	1,92906	158,221	158,221	158,231	4726,14	83,0535
	MTB3c	0,54121	1,87924	116,866	116,866	116,866	9851,02	3844,17
0,5	MTB3a	1,0494	3,30317	109,5	109,5	109,51	304,869	3,95448
	MTB3b	0,58002	3,39971	189,83	189,83	189,829	37798,2	1266,72
	MTB3c	1,04173	3,31678	110,85	110,85	110,85	13834,6	4041,66
0,75	MTB3a	1,5508	4,1731	103,513	103,58	103,582	0	20,5244
	MTB3b	0,95916	4,28921	222,516	222,516	222,516	84134,4	3055,25
	MTB3c	1,54366	4,18577	104,72	104,72	104,72	9798,98	370,979
1	MTB3a	2,05295	4,47464	97,5059	97,5059	97,5084	5120,42	6,3928

	MTB3b	1,40926	4,5567	259,183	259,183	259,182	4168,03	2428,81
	MTB3c	2,04666	4,48648	98,5002	98,5002	98,5003	81560,9	375,504
1,25	MTB3a	2,55575	4,20792	91,4218	91,4218	91,4224	597,225	9,86035
	MTB3b	1,88397	4,14323	297,787	297,788	297,788	1682,07	2805,58
	MTB3c	2,55033	4,21915	92,2237	92,2237	92,2237	854,018	9097,85
1,5	MTB3a	3,05838	3,37307	85,4041	85,2266	85,2274	13,3841	253,321
	MTB3b	2,30773	3,10209	324,673	324,739	-4,2788	872,385	0,00157
	MTB3c	3,05418	3,38388	85,9302	85,9302	85,9112	4385,68	0,307233
1,75	MTB3a	3,53424	1,96839	86,7865	69,1454	69,1222	0,07640	26,9396
	MTB3b	2,60275	1,7396	178,649	178,624	178,594	1279,76	164,543
	MTB3c	3,55769	1,98066	79,6679	79,6679	79,667	2256,29	27,8757
2	MTB3a	4	0	90	45	0	0,30424	0,30424
	MTB3b	4	0	90	45	0	0,83156	0,83156
	MTB3c	4	0	90	45	0	39,5167	39,5167

Tabela 5.33 – Comparação entre intervalos no tempo igual entre os exemplos MTC1a, MTC1b e MTC1c

Tempo (s)	Exemplo	x_0 (m)	y_0 (m)	q_1 (grau)	q_2 (grau)	q_3 (grau)	k_1 (N.m/rad)	k_2 (N.m/rad)
0	MTC1a	0	0	90	60	45	0,58907	0,58906
	MTC1b	0	0	90	60	45	3,00022	3,00021
	MTC1c	0	0	90	60	45	0,20361	0,20361
0,25	MTC1a	0,53140	1,98755	82,6026	82,5976	117,28	490,314	0
	MTC1b	0,55243	1,99186	76,2866	76,2866	76,2866	10333,5	322,074
	MTC1c	0,55272	1,9917	76,2119	76,2119	76,2119	8683,91	8343,67
0,5	MTC1a	1,04154	3,41824	79,2768	78,0597	82,7137	0,10771	0,03327
	MTC1b	1,05471	3,42168	75,6053	75,6051	75,6016	1,72642	0,03338
	MTC1c	1,05536	3,42127	75,4493	75,4493	75,4493	30023	30753,8
0,75	MTC1a	1,54718	4,27869	77,547	77,546	77,5465	231,195	113,684
	MTC1b	1,55696	4,28029	74,9254	74,9254	74,9254	7781,21	301,274
	MTC1c	1,55799	4,27964	74,6881	74,6881	74,6881	136287	2733,17
1	MTC1a	2,04992	4,56734	76,703	76,703	76,7018	81,003	0,46962
	MTC1b	2,0592	4,56768	74,2464	74,2464	74,2464	6729,53	1344,87

	MTC1c	2,0606	4,5668	73,9283	73,9283	73,9283	51231,1	45,9312
1,25	MTC1a	2,55264	4,2848	75,861	75,861	75,8584	101,452	0,26009
	MTC1b	2,56143	4,28386	73,5687	73,5687	73,5687	3595,04	178,317
	MTC1c	2,56319	4,28276	73,17	73,17	73,17	329123	200,131
1,5	MTC1a	3,05535	3,43107	75,0201	75,0199	75,0059	26,1799	0,004
	MTC1b	3,06365	3,42883	72,8921	72,8921	72,8921	357,83	1257,85
	MTC1c	3,06576	3,42751	72,4133	72,4133	72,4133	119416	2721,69
1,75	MTC1a	3,55799	2,00613	74,1951	74,1661	74,1659	193,079	126,706
	MTC1b	3,56584	2,00258	72,2182	72,215	72,2157	3685,07	1096,49
	MTC1c	3,56831	2,00105	71,6581	71,6581	71,6581	1619,16	125,923
2	MTC1a	4	0	90	45	0	0,58907	0,58907
	MTC1b	4	0	90	45	0	3,0002	3,00022
	MTC1c	4	0	90	45	0	0,203611	0,20360

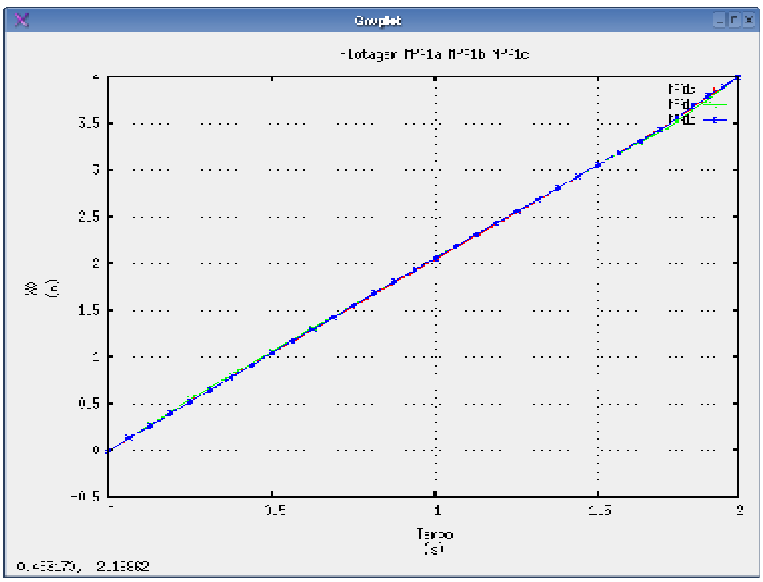


Figura 5.27 – Comparação do resultado para a variável x_0 entre os exemplos MPA1a, MPA1b e MPA1c

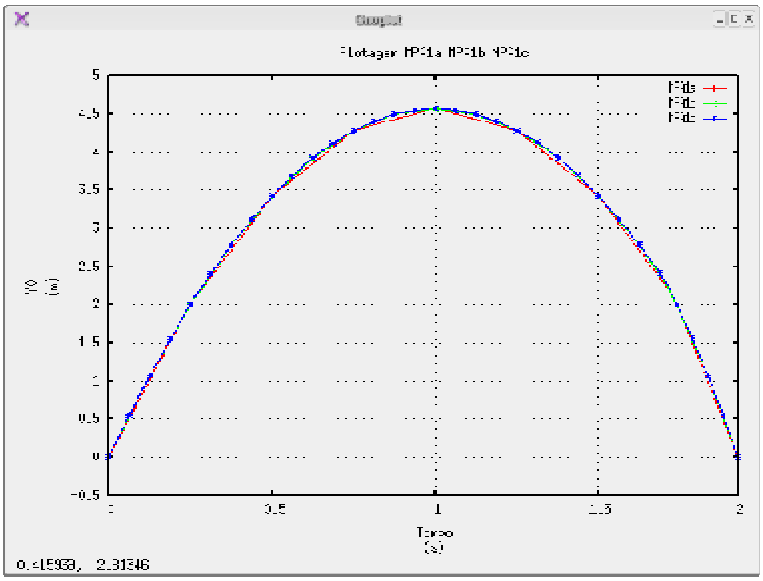


Figura 5.28 – Comparação do resultado para a variável y_0 entre os exemplos MPA1a, MPA1b e MPA1c

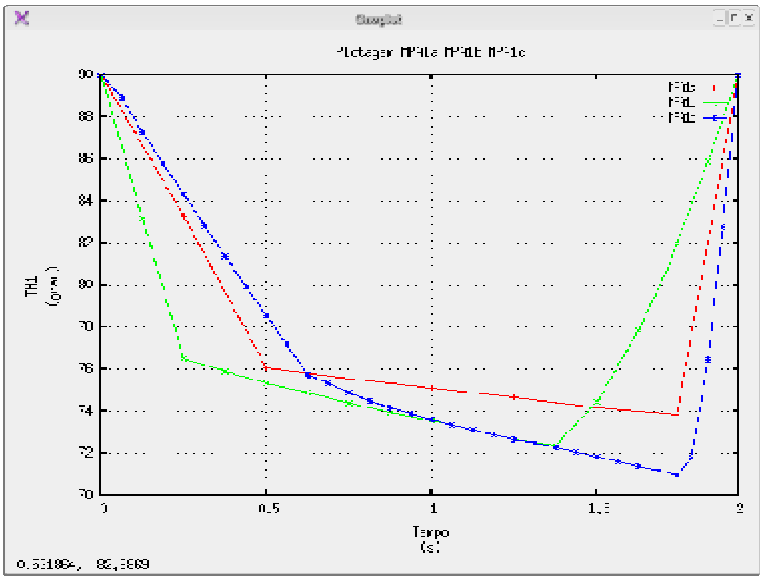


Figura 5.29 – Comparação do resultado para a variável q_1 entre os exemplos MPA1a, MPA1b e MPA1c

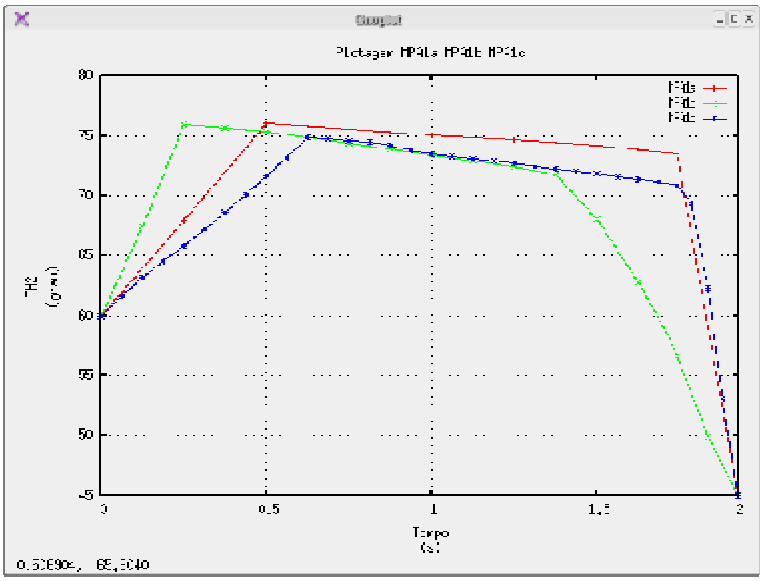


Figura 5.30 – Comparação do resultado para a variável q_2 entre os exemplos MPA1a, MPA1b e MPA1c

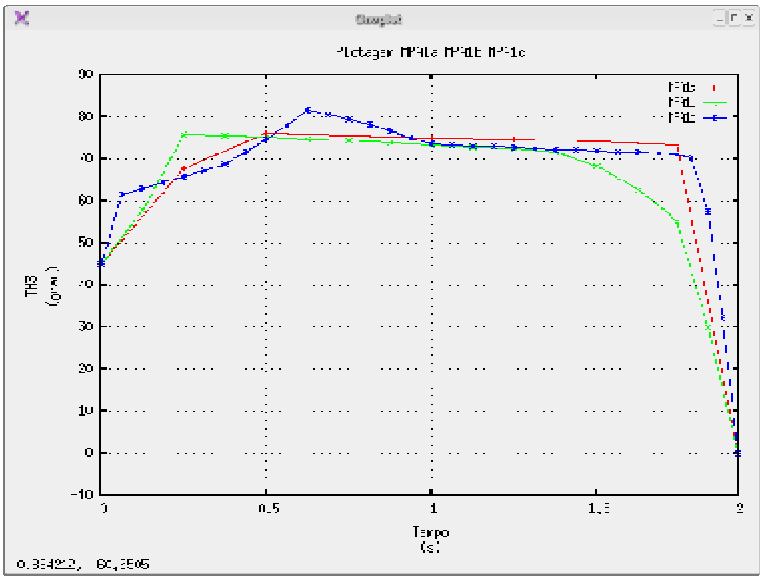


Figura 5.31 – Comparação do resultado para a variável q_3 entre os exemplos MPA1a, MPA1b e MPA1c

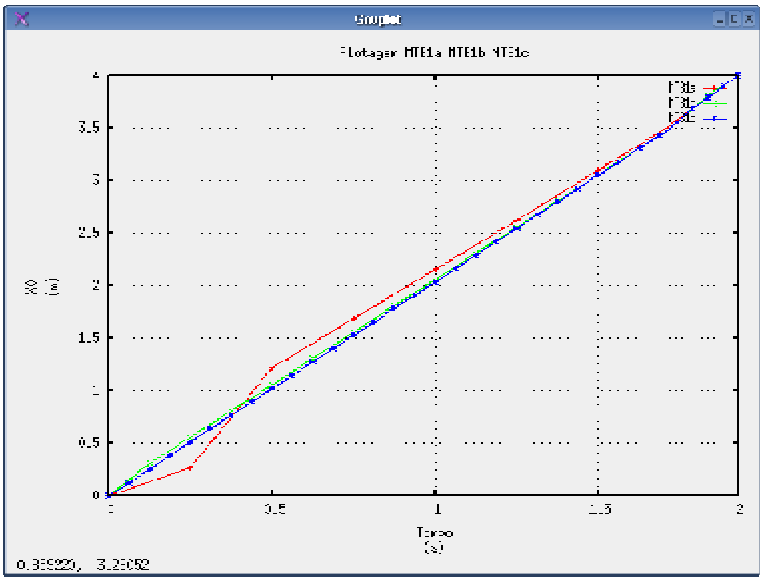


Figura 5.32 – Comparação do resultado para a variável x_0 entre os exemplos MTB1a, MTB1b e MTB1c

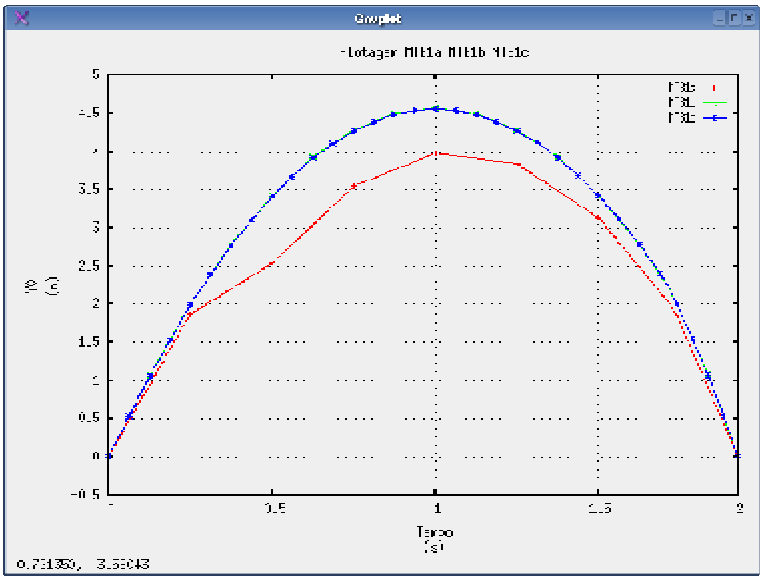


Figura 5.33 – Comparação do resultado para a variável y_0 entre os exemplos MTB1a, MTB1b e MTB1c

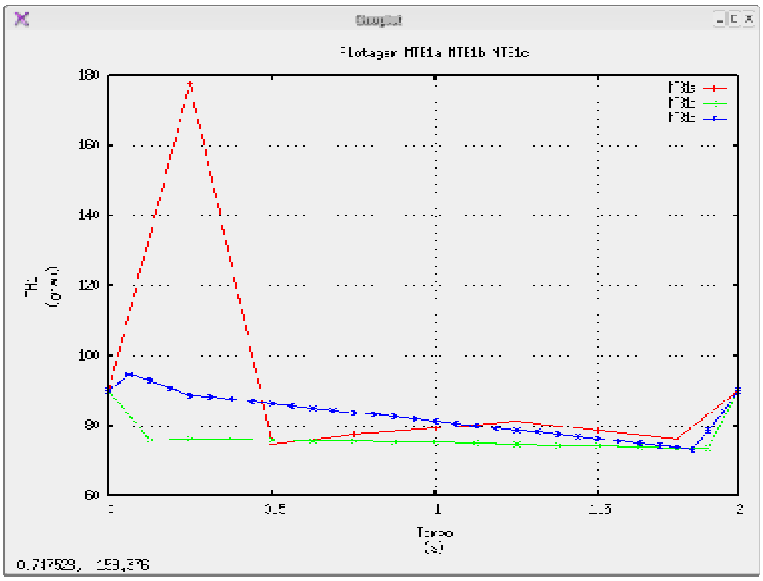


Figura 5.34 – Comparação do resultado para a variável q_1 entre os exemplos MTB1a, MTB1b e MTB1c

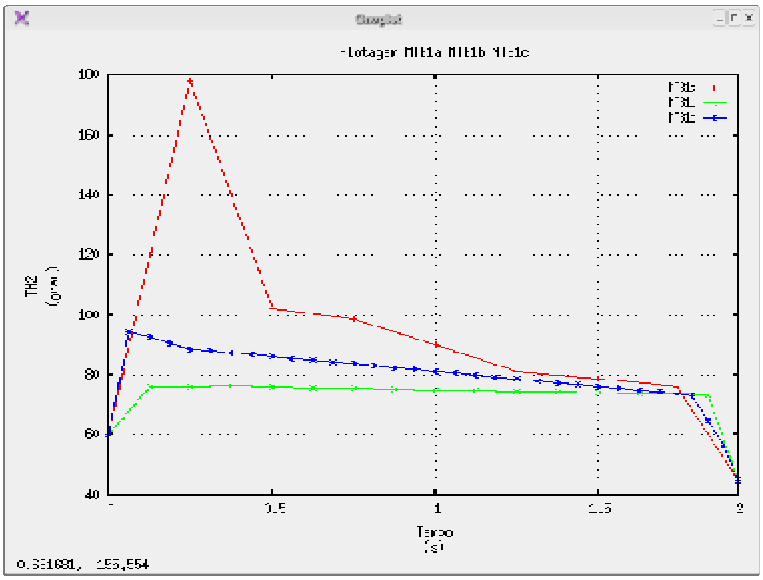


Figura 5.35 – Comparação do resultado para a variável q_2 entre os exemplos MTB1a, MTB1b e MTB1c

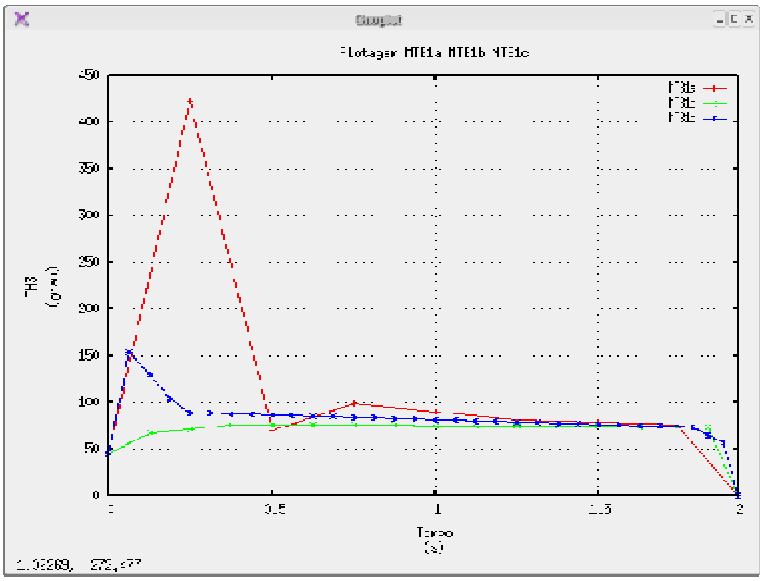


Figura 5.36 – Comparação do resultado para a variável q_3 entre os exemplos MTB1a, MTB1b e MTB1c

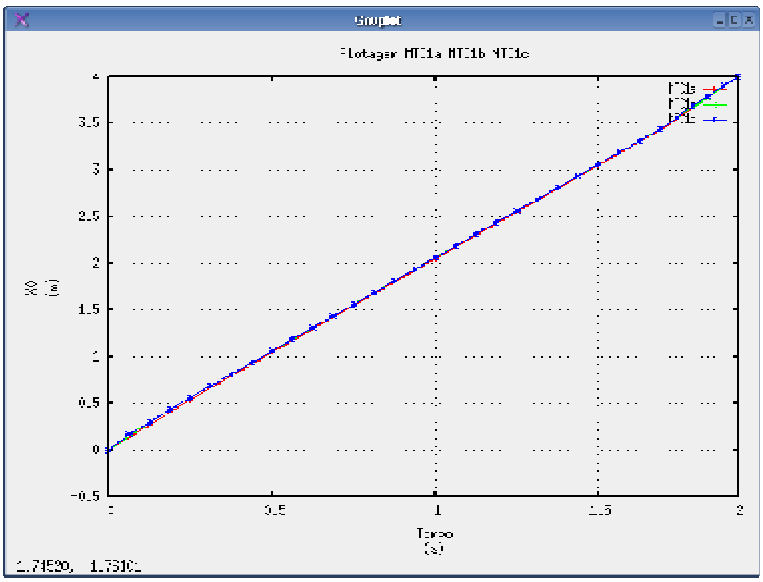


Figura 5.37 – Comparação do resultado para a variável x_0 entre os exemplos MTC1a, MTC1b e MTC1c

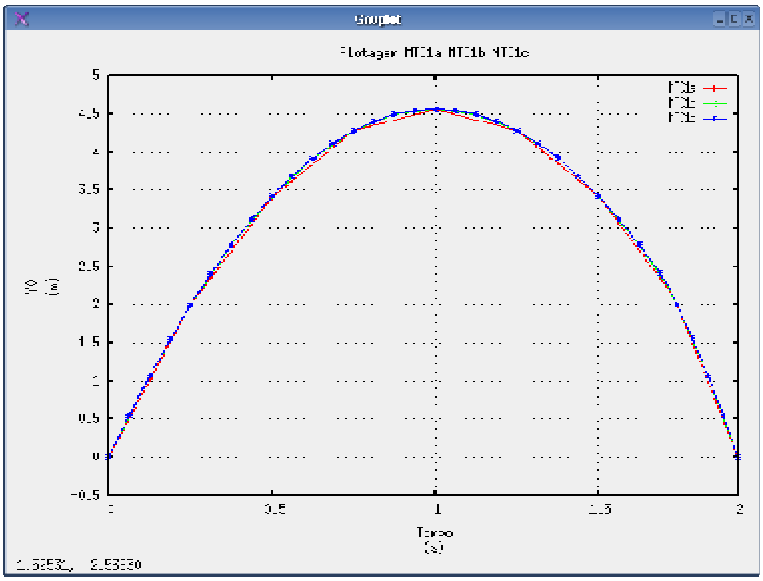


Figura 5.38 – Comparação do resultado para a variável y_0 entre os exemplos MTC1a, MTC1b e MTC1c

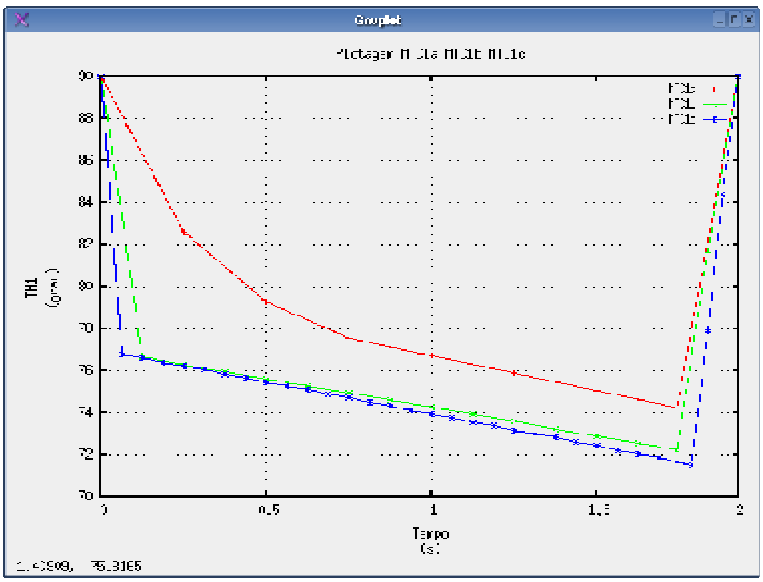


Figura 5.39 – Comparação do resultado para a variável q_1 entre os exemplos MTC1a, MTC1b e MTC1c

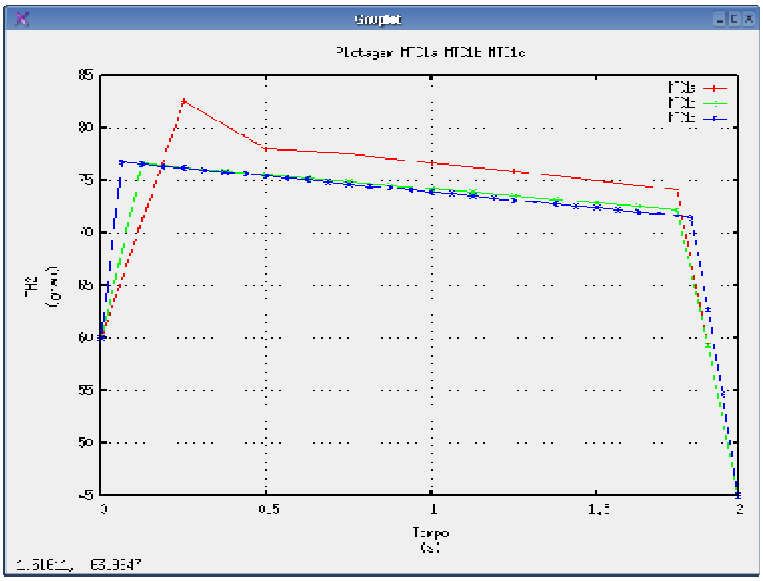


Figura 5.40 – Comparação do resultado para a variável q_2 entre os exemplos MTC1a, MTC1b e MTC1c

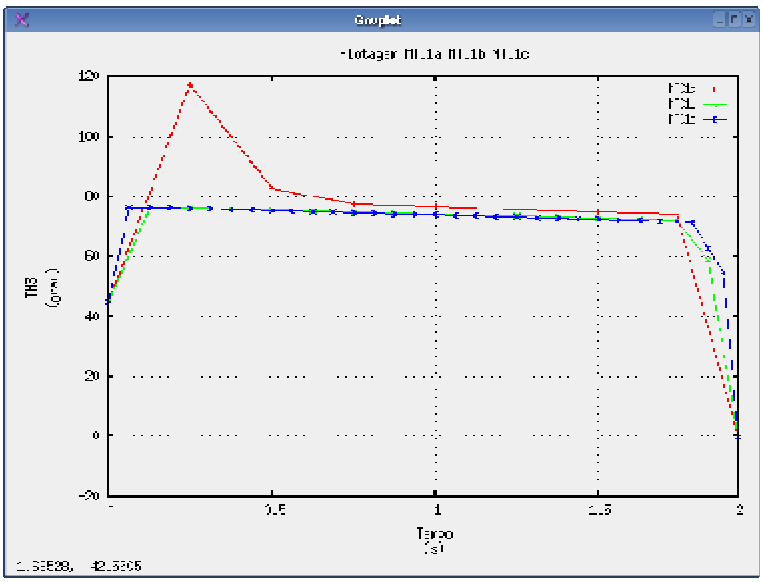


Figura 5.41 – Comparação do resultado para a variável q_3 entre os exemplos MTC1a, MTC1b e MTC1c

A comparação dos exemplos MPA1c, MPA2c e MPA3c (Exemplos com mesma função objetivo, mesma discretização e poses iniciais diferentes) mostrou que houve uma estabilidade na trajetória da base da estrutura articulada, apesar das diferenças do ângulo q_1

$(90^{o},120^{o},150^{o})$ no instante inicial do salto (figuras 5.42 e 5.43). As poses ao longo da trajetória, por sua vez, apresentaram diferenças significativas (figuras 5.44-5.46).

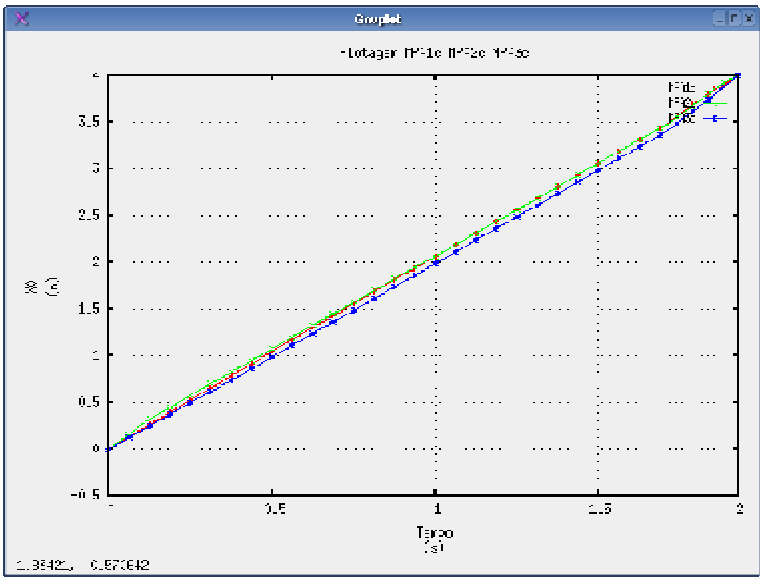


Figura 5.42 – Comparação do resultado para a variável x_0 entre os exemplos MPA1c, MPA2c e MPA3c

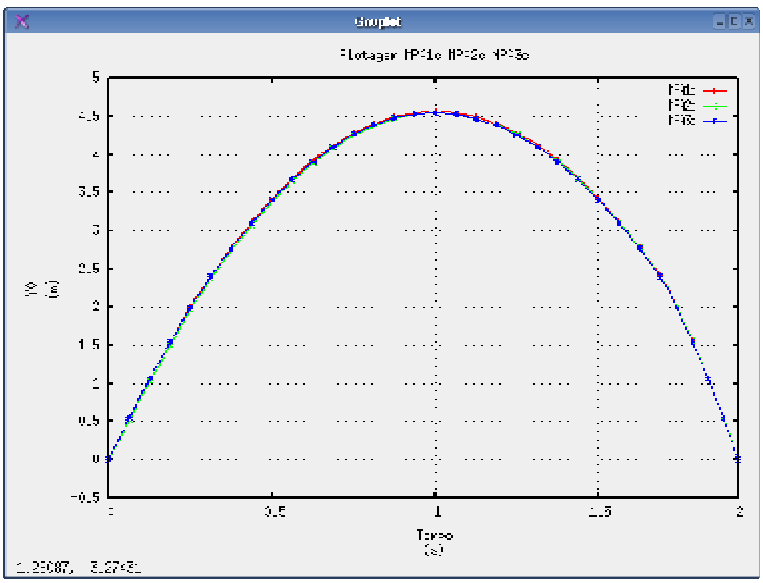


Figura 5.43 – Comparação do resultado para a variável y_0 entre os exemplos MPA1c, MPA2c e MPA3c

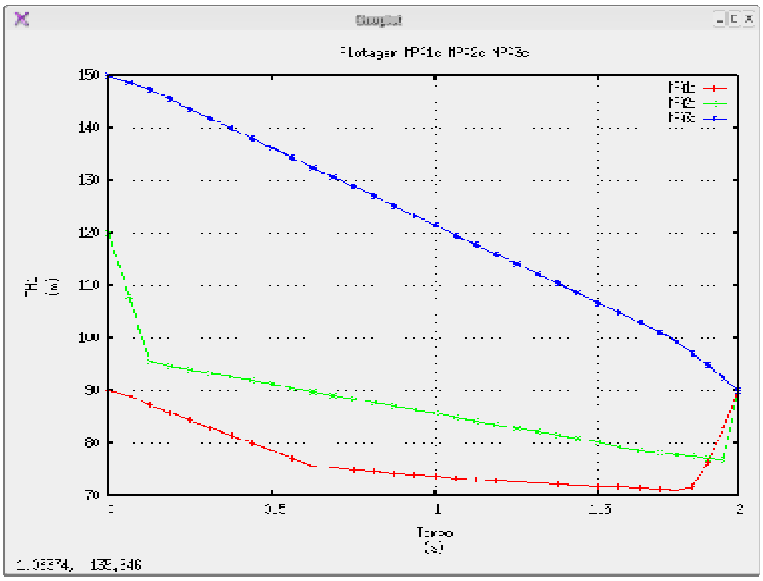


Figura 5.44 – Comparação do resultado para a variável q_1 entre os exemplos MPA1c, MPA2c e MPA3c

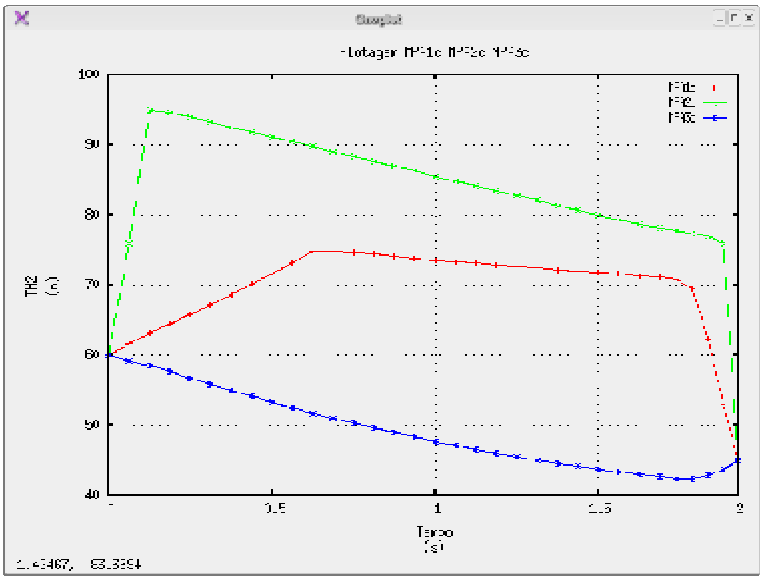


Figura 5.45 – Comparação do resultado para a variável q_2 entre os exemplos MPA1c, MPA2c e MPA3c

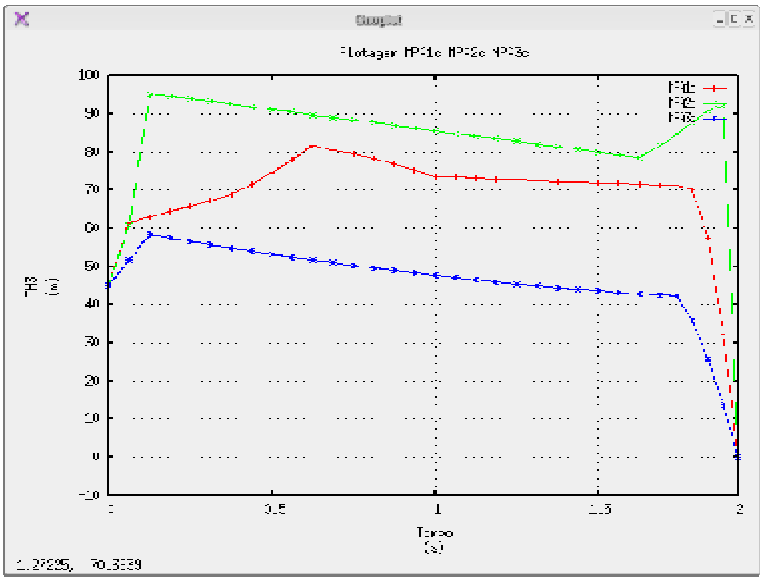


Figura 5.46 – Comparação do resultado para a variável q_3 entre os exemplos MPA1c, MPA2c e MPA3c

Observações semelhantes podem ser feitas comparando-se o exemplo MTB1a com o exemplo MTB1c (figuras 5.47 e 5.48 para as trajetórias da base e figuras 5.49-5.51 para as poses) e comparando-se o exemplo MTC1c e MTC2c (figuras 5.52 e 5.53 para as trajetórias da base e figuras 5.54-5.56 para as poses).

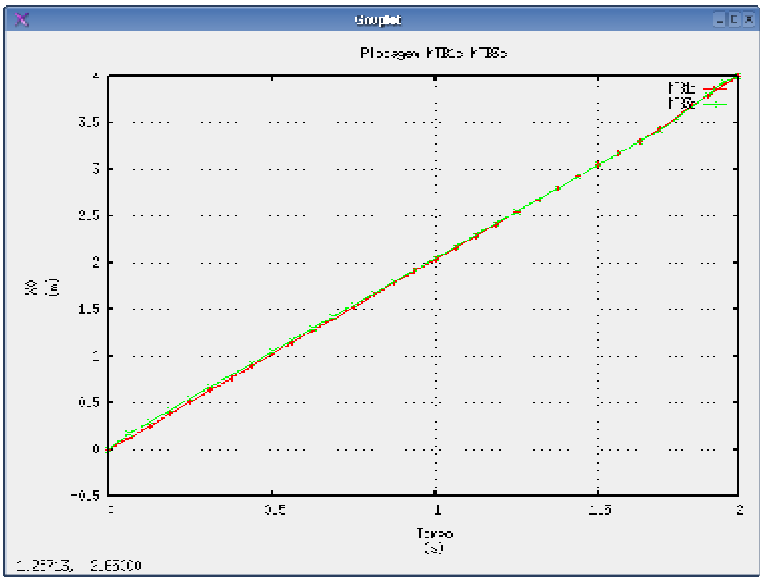


Figura 5.47 – Comparação do resultado para a variável x_0 entre os exemplos MTB1c e MTB3c

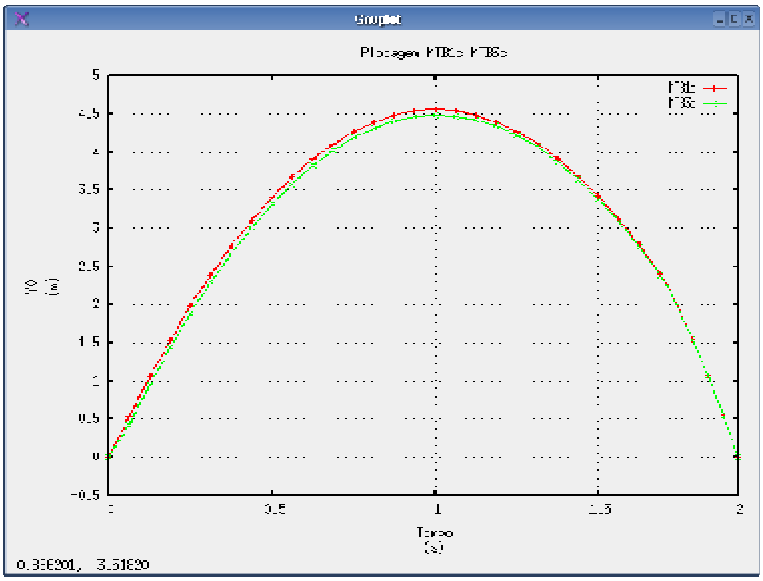


Figura 5.48 – Comparação do resultado para a variável y_0 entre os exemplos MTB1c e MTB3c

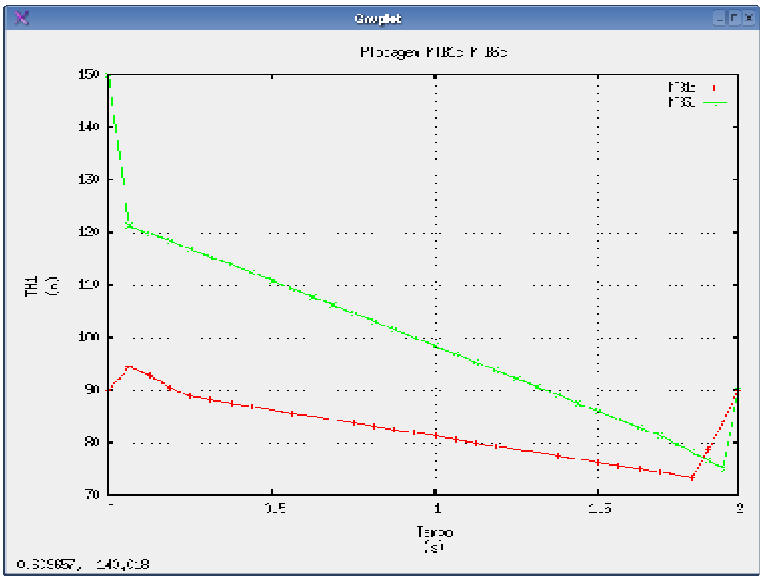


Figura 5.49 – Comparação do resultado para a variável q_1 entre os exemplos MTB1c e MTB3c

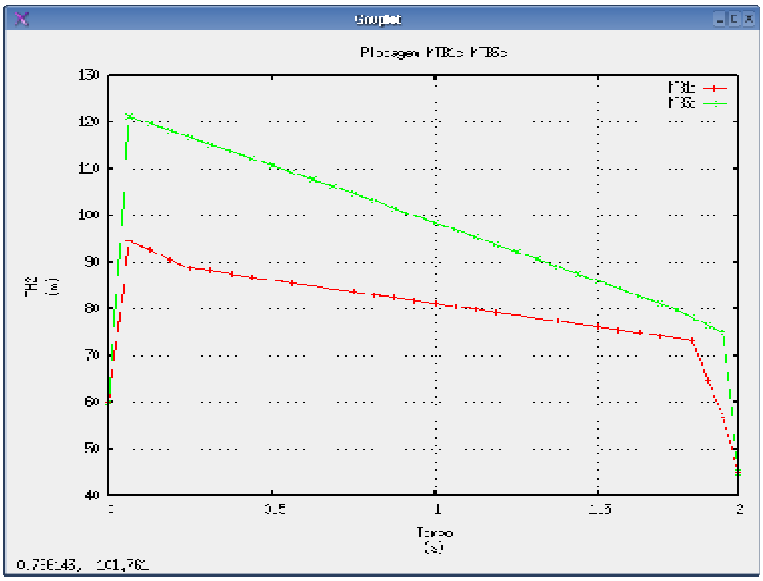


Figura 5.50 – Comparação do resultado para a variável q_2 entre os exemplos MTB1c e MTB3c

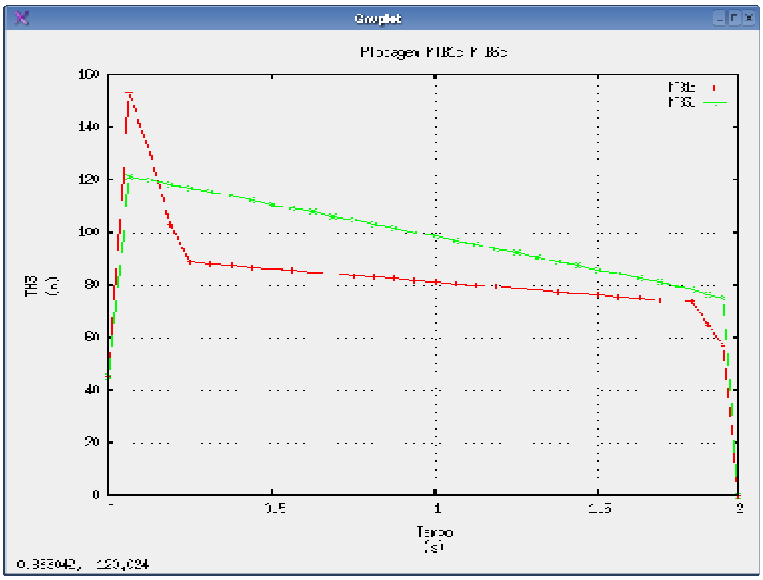


Figura 5.51 – Comparação do resultado para a variável q_3 entre os exemplos MTB1c e MTB3c

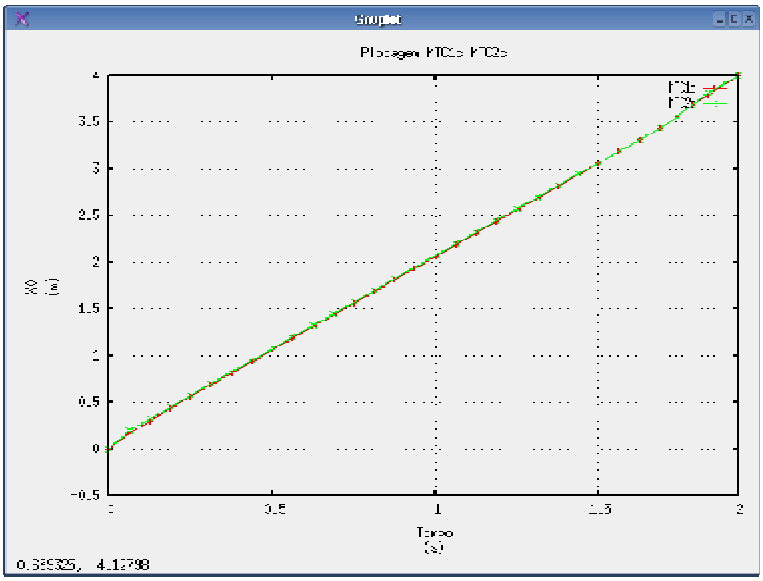


Figura 5.52 – Comparação do resultado para a variável x_0 entre os exemplos MTC1c e MTC2c

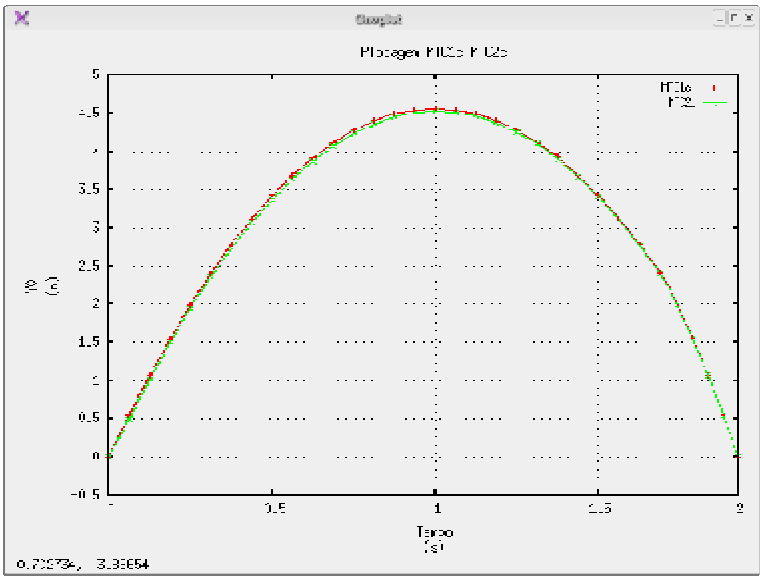


Figura 5.53 – Comparação do resultado para a variável y_0 entre os exemplos MTC1c e MTC2c

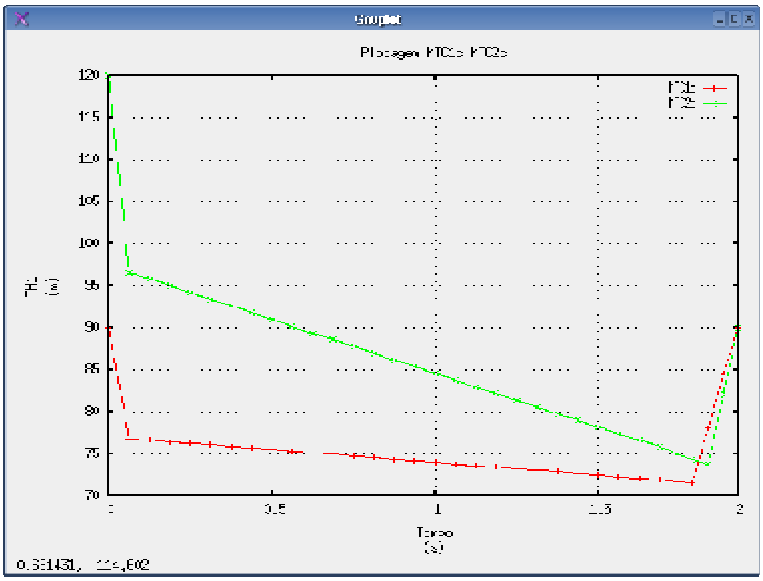


Figura 5.54 – Comparação do resultado para a variável q_1 entre os exemplos MTC1c e MTC2c

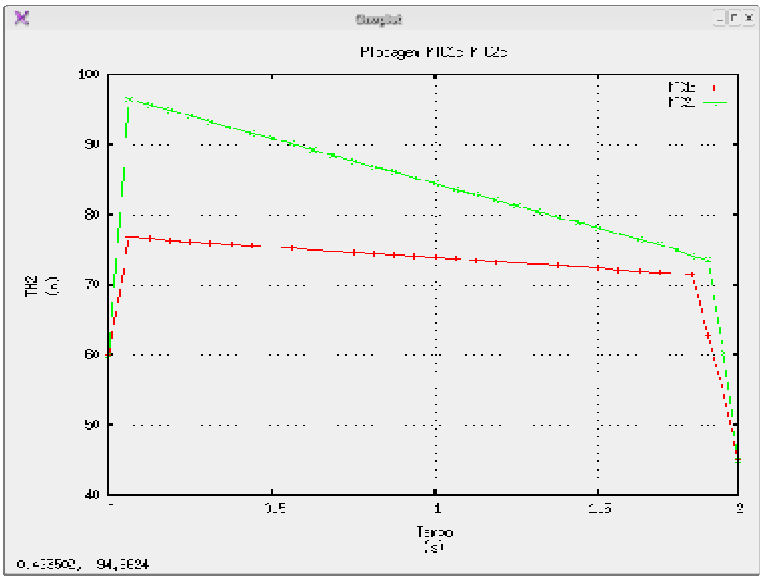


Figura 5.55 – Comparação do resultado para a variável q_2 entre os exemplos MTC1c e MTC2c

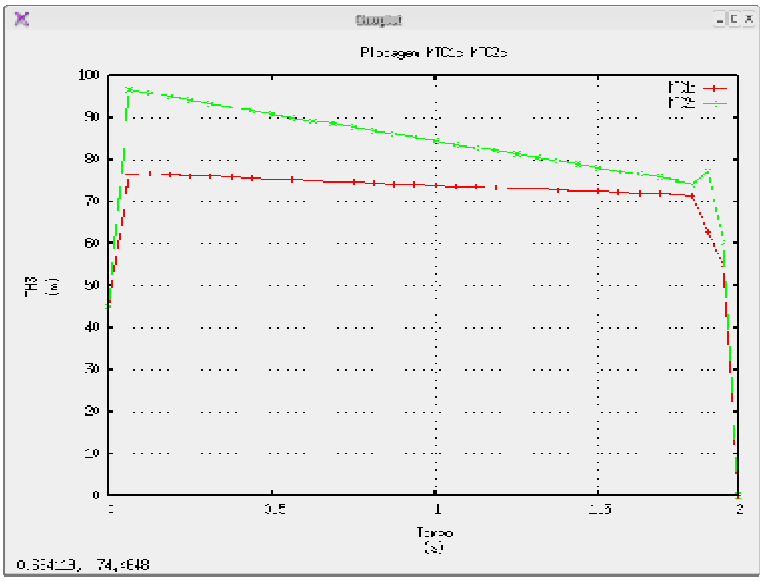


Figura 5.56 – Comparação do resultado para a variável q_3 entre os exemplos MTC1c e MTC2c

5.3.2 Observações sobre realismo da animação

Nos exemplos MTB2c e MTC3c, foram obtidas animações para a estrutura articulada em que a trajetória, apesar de respeitar todas as restrições físicas, não é realística. Nesses exemplos, os resultados obtidos pelo otimizador foram sub-ótimos, isto é, os resultados correspondem a uma solução fornecida pelo otimizador quando o critério de parada é atingido antes de o resultado ótimo ter sido alcançado. Mesmo em alguns casos em que foi obtido o ótimo como, por exemplo, no exemplo MTB1a, a trajetória resultante não foi totalmente realística (Figura 5.10). Porém, com um maior refinamento do tempo a convergência foi atingida e resultados melhores foram obtidos (figuras 5.32-5.36).

Pôde-se observar que a utilização da potência angular como função objetivo gerou animações com transições de pose mais suaves (5.57-5.61) e aparentemente mais realísticas do que as animações obtidas com a utilização das outras funções objetivos. Isso leva a crer que critérios de minimização energético são fisicamente mais plausíveis.

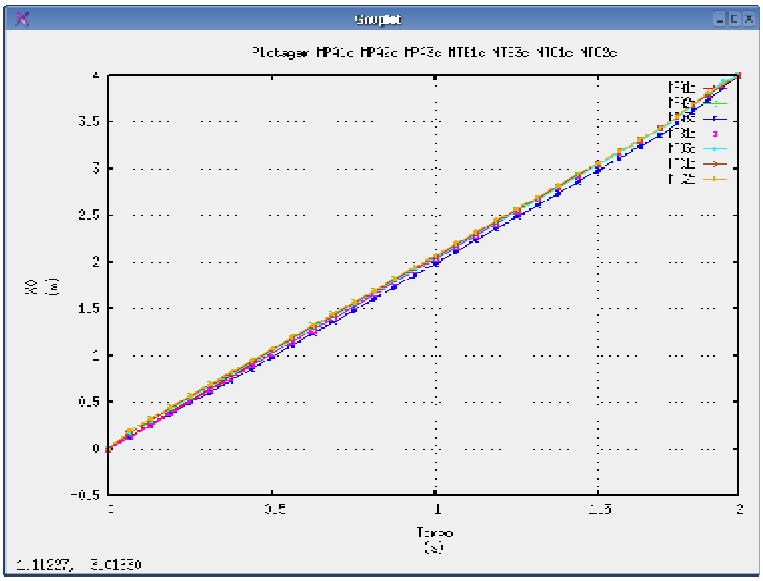


Figura 5.57 – Comparação do resultado para a variável x_0 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c

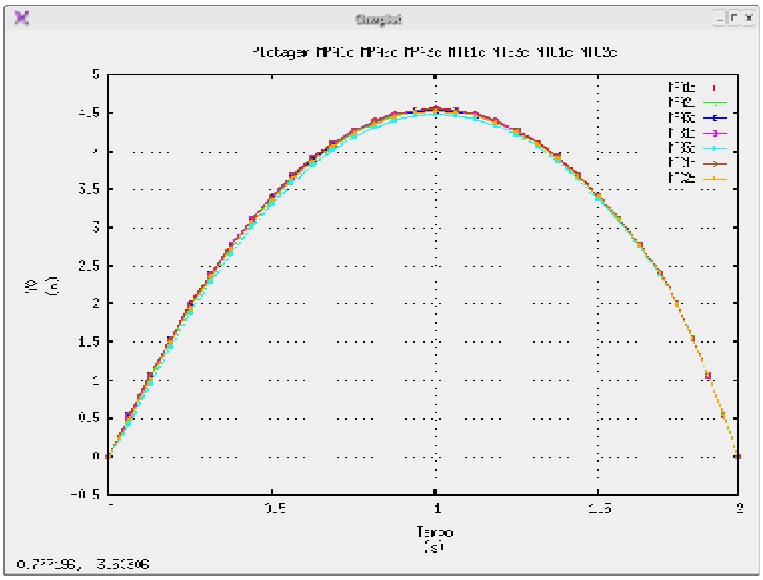


Figura 5.58 – Comparação do resultado para a variável y_0 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c

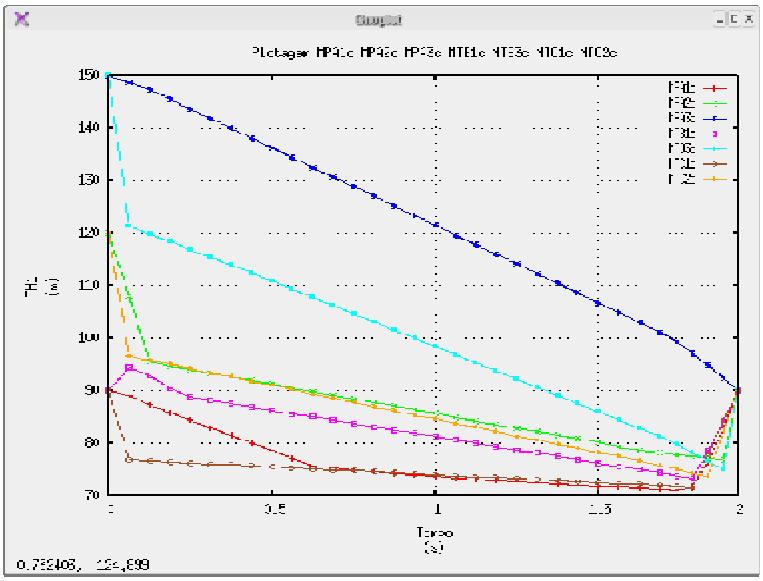


Figura 5.59 – Comparação do resultado para a variável q_1 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c

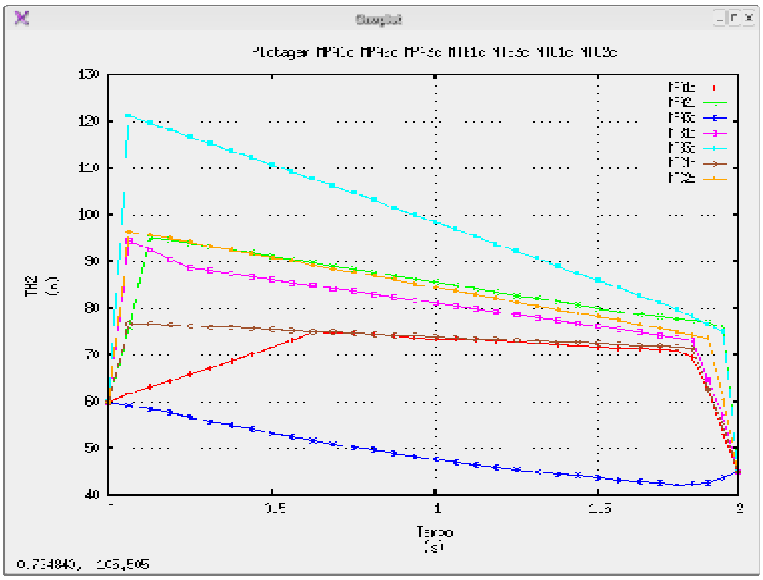


Figura 5.60 – Comparação do resultado para a variável q_2 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c

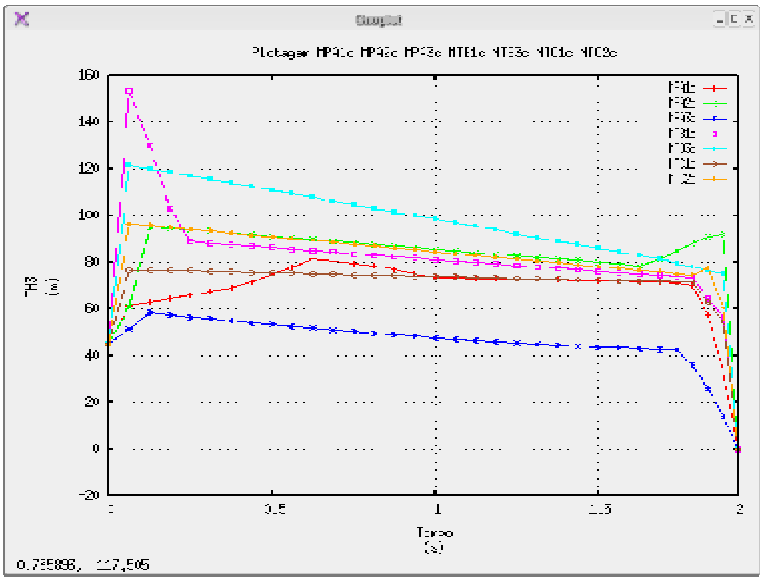


Figura 5.61 – Comparação do resultado para a variável q_3 entre os exemplos MPA1c, MPA2c, MPA3c, MTB1c, MTB3c, MTC1c e MTC2c

5.3.3 Observações sobre tempo de geração da animação

Um outro resultado também já esperado (Cohen, 1992) é em relação ao tempo de CPU. Os resultados mostram que à medida que o número de intervalos do tempo aumenta o tempo de otimização também aumenta (Tabela 5.34). Esse resultado é claramente exponencial e é ilustrado, por exemplo, na Figura 5.62. Esse problema de super-otimização pode ser contornado por um método de multi-fases na otimização, isto é, fazendo uma otimização de um intervalo no tempo suficientemente pequeno e usando esse resultado para outros intervalos subseqüentes até preencher completamente os intervalos do tempo.

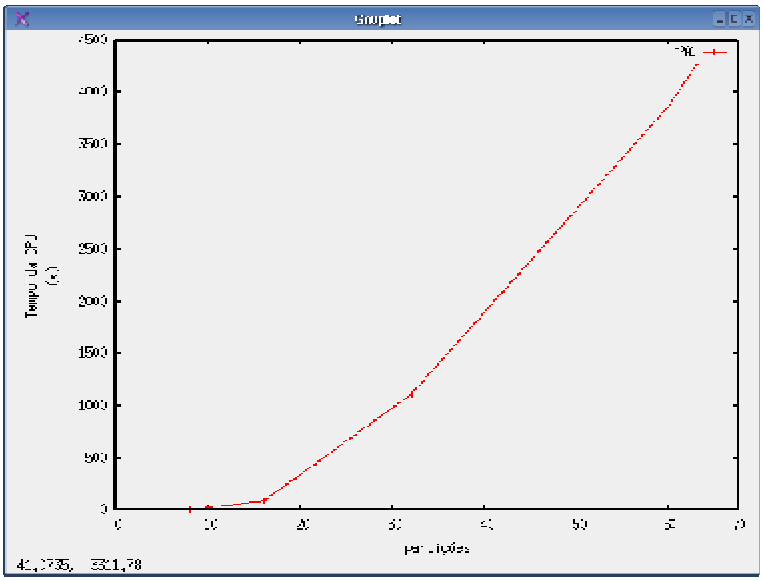


Figura 5.62 - Tempo da otimização para o exemplo MPA1 com 8, 16, 32 e 64 discretizações

Tabela 5.34 – Tempo gasto pela CPU em busca de uma solução ótima

Exemplo	Tempo (s)	Exemplo	Tempo (s)	Exemplo	Tempo (s)
MPA1a	6,87	MTB1a	5,71	MTC1a	17,12
MPA1b	94,6	MTB1b	47,68	MTC1b	50
MPA1c	1108,47	MTB1c	973,09	MTC1c	722,12
MPA2a	5,99	MTB2a	8,18	MTC2a	25,52
MPA2b	57,38	MTB2b	95,21	MTC2c	954,01
MPA2c	1848,85	MTB3a	4,13	MTC3a	4,79
MPA3a	8,96	MTB3b	78,93	MTC3b	58,1
MPA3b	87,79	MTB3c	1781,64		
MPA3c	1077				

Capítulo 6
Conclusões e Recomendações

Nesse trabalho fez-se um estudo e uma avaliação da técnica de *spacetime constraints* em relação ao nível de controle, realismo e esforço computacional necessários ao processo de animação. Um sistema híbrido que integra computação simbólica e otimização numérica foi desenvolvido com essa finalidade: criar animações dinâmicas e realísticas de estruturas articuladas através de uma interface gráfica amigável.

A técnica de *Spacetime Constraints*, quando foi proposta inicialmente, visava à geração de animações compatíveis com as leis da física, o que, na visão dos autores, adicionaria um maior nível de realismo à animação. No entanto, essa técnica exige do animador um conhecimento mínimo sobre o processo de otimização. Esse conhecimento implica em: escolher entre alguns algoritmos de otimização, escolher critérios de paradas e saber selecionar uma boa função objetiva para o movimento. Definir critérios de parada para o otimizador, assim como relacionar a melhor função objetivo é um problema subjetivo e que deve ser determinado pelo animador. O animador deverá, também, ter conhecimento sobre a dinâmica do movimento para poder configurar poses iniciais e finais adequadas. Outro ponto a se considerar é que o processo de otimização não necessariamente converge para a solução ideal em uma primeira tentativa.

Pelos testes apresentados no Capítulo 5, pode-se constatar que a técnica de *Spacetime Constraints* incorporada em um sistema de otimização com interfaces amigáveis permite que o animador avalie diversas animações. Essas animações são soluções de problemas de otimização definidos por diferentes configurações de animação, diferentes funções objetivos, diferentes níveis de discretização e diferentes configurações iniciais, configurados pelo animador. Assim, apesar de o processo numérico de otimização ser bastante lento, ainda é mais vantajoso para o animador especificar todas as variáveis envolvidas nesse processo do que tentar reproduzir uma

animação realística, especificando, ele próprio, todas as poses necessárias ao movimento de uma estrutura articulada complexa.

Na técnica estudada, observou-se que o aumento da discretização fornece melhores resultados a um custo computacional significativamente mais elevado. Observou-se também que nem sempre o sistema conseguiu atingir uma solução ótima, por exemplo, dos 27 exemplos testados, 2 apresentaram resultados viáveis sub-ótimos e 1 não retornou resultado viável.

Para os exemplos de animação testados com a mesma função objetivo e diferentes poses iniciais, observou-se que a trajetória da base da estrutura articulada praticamente se manteve, no entanto, as poses intermediárias variam significativamente e as constantes de rigidez apresentam discrepâncias ainda mais expressivas.

Para níveis de discretização baixos, alguns resultados atingiram o ótimo da função objetivo, porém as poses ao longo da animação se apresentaram insatisfatórias do ponto de vista estético. Com um melhor refinamento, atingiu a um resultado satisfatório.

Das três funções objetivos testadas, a potência angular gerou animações com transições de pose mais suaves e aparentemente mais realísticas do que as animações obtidas com a utilização das outras funções objetivos (trajetória da base e trajetória do centróide).

Observou-se que o tempo de otimização aumenta exponencialmente com o número de partições do tempo da animação (Resultado já relatado na literatura (Cohen, 1992)).

Há necessidade de uma maior investigação para explicar as discrepâncias observadas com relação aos valores das variáveis de rigidez em cada instante da animação para níveis de discretização diferentes. É também importante que uma variedade maior de funções objetivos sejam estudadas e seus efeitos analisados. Em função do elevado custo de otimização associado a um espaço de busca com um grande número de variáveis, é necessário buscar técnicas de redução do tamanho do espaço de busca. Sugere-se também verificar os efeitos de uma solução hierárquica do problema de otimização, iniciando-se com um espaço de busca reduzido e

utilizando-se a solução ótima desse espaço de busca para definir a configuração inicial em um espaço de busca mais refinado.

Anexo A
Conceitos básicos

Graus de liberdade

Chama-se de graus de liberdade, o número de variáveis com posições independentes no tempo. Essas variáveis são necessárias à especificação da pose de uma estrutura articulada e descreve a flexibilidade do movimento.

Efetor final

Chama-se de efetor final, a posição final de uma topologia de ligações de uma estrutura articulada.

Vetor do espaço

Definem-se por vetor do espaço, todas as configurações possíveis de uma estrutura articulada. O vetor do espaço é o conjunto de todos os parâmetros independentes que definem a posição, orientação e rotação das juntas. Numa configuração particular o vetor do estado é descrito

$$q = (q_1, ..., q_N) \qquad \text{A.1}$$

onde N define a dimensão do espaço e é, para alguns casos particulares, equivalente ao número de graus de liberdade de uma figura articulada.

Anexo B

Equações do movimento

B.1 Velocidade

O vetor velocidade $\mathbf{v}$ pra uma partícula g relativo ao sistema de coordenadas globais XYZ

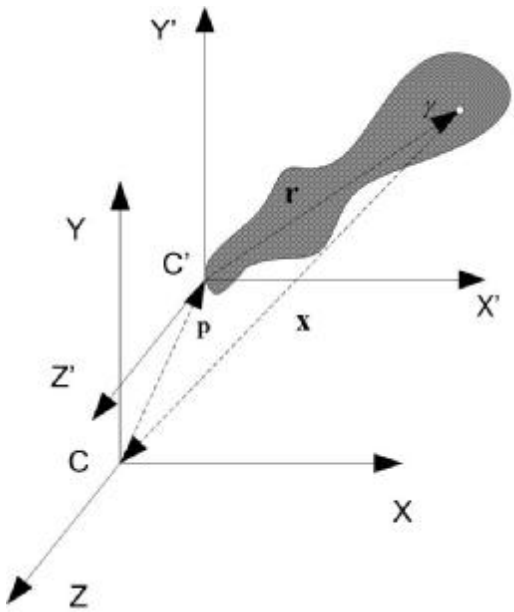


Figura B.1 - Mudança de coordenada para cada partícula

$$\mathbf{v} = \frac{d\mathbf{x}}{dt} = \dot{\mathbf{x}}$$

onde o $\mathbf{x}$ representa o vetor posição da partícula em relação ao sistema de coordenadas globais.

$$\dot{\mathbf{x}} = \frac{d(\mathbf{p}+\mathbf{r})}{dt} = \mathbf{v}_i + \frac{d\mathbf{r}}{dt} = \mathbf{v} + \mathbf{w}\times\mathbf{r}$$

B.2 Energia cinética

A energia cinética para certa partícula g no corpo

$$Tg = \frac{1}{2} dm(\mathbf{v} + \mathbf{w}\times\mathbf{r}_g)\cdot(\mathbf{v} + \mathbf{w}\times\mathbf{r}_g)$$

$$Tg_i = \frac{1}{2}[\mathbf{v}\cdot\mathbf{v} + 2\mathbf{v}\cdot\mathbf{w}\times\mathbf{r}_g + (\mathbf{w}\times\mathbf{r}_g)\cdot(\mathbf{w}\times\mathbf{r}_g)]$$

Para um corpo rígido qualquer a energia cinética

$$T = \frac{1}{2}\left[\int \mathbf{v}\cdot\mathbf{v}dm + 2\int \mathbf{v}\cdot\mathbf{w}\times\mathbf{r}_i dm + \int (\mathbf{w}\times\mathbf{r}_i)\cdot(\mathbf{w}\times\mathbf{r}_i)dm\right]$$

Consideram-se três termos

$$T = [T_1 + T_2 + T_3]$$

$$T_1 = \frac{1}{2}\int \mathbf{v}\cdot\mathbf{v}dm = \frac{1}{2}\mathbf{v}\cdot\mathbf{v}\int dm = \frac{m}{2}\cdot|\mathbf{v}|^2$$

$$T_2 = \int \mathbf{v}\cdot\mathbf{w}\times\mathbf{r}_i dm = \mathbf{v}\cdot\int \mathbf{w}\times\mathbf{r}_i dm$$

Onde

$$\int \mathbf{w}\times\mathbf{r}_i dm = \int \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ w_x & w_y & w_z \\ r_{ix} & r_{iy} & r_{iz} \end{vmatrix} dm = \begin{pmatrix} \int (w_y r_{iz} - w_z r_{iy})\, dm \\ \int (w_z r_{ix} - w_x r_{iz}) dm \\ \int (w_x r_{iy} - w_y r_{ix})\, dm \end{pmatrix}$$

$$T_2 = \mathbf{v}\cdot\int \mathbf{w}\times\mathbf{r}_i dm = v_x\int (w_y r_{iz} - w_z r_{iy})dm + v_y\int (w_z r_{ix} - w_x r_{iz})dm + v_z\int (w_x r_{iy} - w_y r_{ix})dm$$

Considera-se

$$v_x\int (w_y r_{iz} - w_z r_{iy})dm = v_x w_y\int r_{iz}dm - v_x w_z\int r_{iy}dm \qquad \text{(B.2)}$$

$$v_y\int (w_z r_{ix} - w_x r_{iz})dm = v_y w_z\int r_{ix}dm - v_y w_x\int r_{iz}dm \qquad \text{(B.3)}$$

$$v_z\int (w_x r_{iy} - w_y r_{ix})dm = v_z w_x\int r_{iy}dm - v_z w_y\int r_{ix}dm \qquad \text{(B.4)}$$

Visto que

$$\int r_{ix} dm = m \cdot x'_c \tag{B.5}$$

$$\int r_{iy} dm = m \cdot y'_c \tag{B.6}$$

$$\int r_z dm = m \cdot z'_c \tag{B.7}$$

onde (x_c, y_c, z_c) são as coordenadas do centro de massa do corpo rígido.

Substituindo-se as equações B.5, B.6 e B.7 nas equações B.2, B.3 e B.4 e o resultado na equação B.1 obtém

$$T_2 = m[w_x(y'_c v_z - z'_c v_y) + w_y(z'_c v_x - v_z) + w_z(x'_c v_y - y'_c v_x)]$$

$$= m\begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix} \cdot \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x'_c & y'_c & z'_c \\ v_x & v_y & v_z \end{vmatrix} = m\mathbf{w} \cdot \mathbf{r}_c \times \mathbf{v}$$

onde $\mathbf{r}_c$ é o vetor do centro de massa

$$T_3 = \frac{1}{2}\int (\mathbf{w} \times \mathbf{r}_i) \cdot (\mathbf{w} \times \mathbf{r}_i) dm = \frac{1}{2}\int \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ w_x & w_y & w_z \\ r_{ix} & r_{iy} & r_{iz} \end{vmatrix} \cdot \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ w_x & w_y & w_z \\ r_{ix} & r_{iy} & r_{iz} \end{vmatrix} dm$$

$$= \frac{1}{2}\int \begin{pmatrix} (w_y r_{iz} - w_z r_{iy}) \\ (w_z r_{ix} - w_x r_{iz}) \\ (w_x r_{iy} - w_y r_{ix}) \end{pmatrix} \cdot \begin{pmatrix} (w_y r_{iz} - w_z r_{iy}) \\ (w_z r_{ix} - w_x r_{iz}) \\ (w_x r_{iy} - w_y r_{ix}) \end{pmatrix} dm$$

$$= \frac{1}{2}\int [(w_y r_{iz} - w_z r_{iy})^2 + (w_z r_{ix} - w_x r_{iz})^2 + (w_x r_{iy} - w_y r_{ix})^2] dm$$

$$= \frac{1}{2}\int [(w_y^2 r_{iz}^2 - 2w_z w_y r_{iy} r_{iz} + w_z^2 r_{iy}^2) + (w_z^2 r_{ix}^2 - 2w_z w_x r_{ix} r_{iz} + w_x^2 r_{iz}^2) + (w_x^2 r_{iy}^2 - 2w_x w_y r_{iy} r_{ix} + w_y^2 r_{ix}^2)] dm$$

$$= \frac{1}{2}[w_x(w_x\int(r_{iz}^2 + r_{iy}^2)dm - w_y\int r_{ix}r_{iy}dm - w_z\int r_{ix}r_{iz}dm$$

$$+ w_y(w_x(-\int r_{iy}r_{ix}dm) + w_y\int(r_{ix}^2 + r_{iz}^2)dm - w_z\int r_{iy}r_{iz}dm$$

$$+ w_z(w_x(-\int r_{iz}r_{ix}dm) + w_y\int r_{iz}r_{iy}dm - w_z\int(r_{ix}^2 + r_{iy}^2)dm]$$

$$= \frac{1}{2}\begin{pmatrix} w_x & w_y & w_z \end{pmatrix}\begin{bmatrix} \int(r_{iz}^2 + r_{iy}^2)dm & -\int r_{ix}r_{iy}dm & -\int r_{ix}r_{iz}dm \\ -\int r_{iy}r_{ix}dm & \int(r_{ix}^2 + r_{iz}^2) & -\int r_{iy}r_{iz}dm \\ -\int r_{iz}r_{ix}dm & -\int r_{iz}r_{iy}dm & \int(r_{ix}^2 + r_{iy}^2)dm \end{bmatrix}\begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix}$$

$$= \frac{1}{2}\begin{pmatrix} w_x & w_y & w_z \end{pmatrix}\begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}\begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix} = \frac{1}{2}\mathbf{w}^t.\mathbf{I}.\mathbf{w}$$

Então, a energia cinética T do corpo rígido é dada por

$$T = \frac{m}{2}\cdot|\mathbf{v}|^2 + m\mathbf{w}.\mathbf{r}_c\times\mathbf{v} + \frac{1}{2}\mathbf{w}^t.\mathbf{I}.\mathbf{w}$$

$$= \frac{1}{2}|\mathbf{v}|^2 m + \frac{1}{2}|\mathbf{w}|^2\int\mathbf{r}_i^2 dm + \int\mathbf{v}.\mathbf{w}\times\mathbf{r}_i dm$$

$$= \frac{1}{2}|\mathbf{v}|^2 m + \frac{1}{2}|\mathbf{w}|^2\mathbf{I} + \int\mathbf{v}.\mathbf{w}\times\mathbf{r}_i dm \qquad \text{(B.8)}$$

Visto que

$$\int_v r_i dm = r_c m \qquad \text{(B.9)}$$

onde $\mathbf{r}_c$ é o vetor do centro de massa

Visto que

$$\mathbf{w}\times\mathbf{r}_i = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ w_x & w_y & w_z \\ r_{ix} & r_{iy} & r_{iz} \end{vmatrix} = (w_y r_{iz} - w_z r_{iy})\mathbf{i} + (w_z r_{ix} - w_x r_{iz})\mathbf{j} + (w_x r_{iy} - w_y r_{ix})\mathbf{k}$$

$$\mathbf{v}.\mathbf{w}\times\mathbf{r}_i = v_x(w_y r_{iz} - w_z r_{iy}) + v_y(w_z r_{ix} - w_x r_{iz}) + v_z(w_x r_{iy} - w_y r_{ix})$$

$$\int \mathbf{v}.\mathbf{w}\times\mathbf{r}_i = v_x(w_y \int r_{iz}dm - w_z \int r_{iy}dm) + v_y(w_z \int r_{ix}dm - w_x \int r_{iz}dm) + v_z(w_z \int r_{iy}dm - w_y \int r_{ix}dm)$$

Pela equação B.9 temos

$$\int \mathbf{v}.\mathbf{w}\times\mathbf{r}_i = \mathbf{v}.\mathbf{w}\times\mathbf{r}_c m$$

$$T = \frac{1}{2}|\mathbf{v}|^2 m + \frac{1}{2}|\mathbf{w}|^2 \mathbf{I} + \mathbf{v}.\mathbf{w}\times\mathbf{r}_c m \qquad \text{(B.10)}$$

Podemos aplicar a equação B.10 para o caso de corpos rígidos conectados por juntas. A equação da energia cinemática para cada corpo rígido j fica:

$$T_j = \frac{1}{2}|\mathbf{v}_j|^2 m_j + \frac{1}{2}|\mathbf{w}_j|^2 \mathbf{I}_j + \mathbf{v}_j.\mathbf{w}_j\times\mathbf{r}_{cj} m_j$$

$$\begin{aligned} \mathbf{v}_j &= \frac{d\mathbf{p}}{dt}, j = 0 \\ &= \mathbf{v}_{j-1} + \mathbf{w}_{j-1}\times\mathbf{r}_{j-1} \end{aligned}$$

Visto que $\mathbf{r}_j$ é o vetor distância de uma articulação a outra em relação ao sistema de coordenadas local e $\mathbf{p}$ é o vetor distancia que localiza a base da estrutura em relação ao sistema de coordenadas global.

B.3 Inércia rotacional

A inércia rotacional é para cada partícula i é definida no corpo j:

$$\mathbf{I} = \sum r_i^2 d\, m_i$$

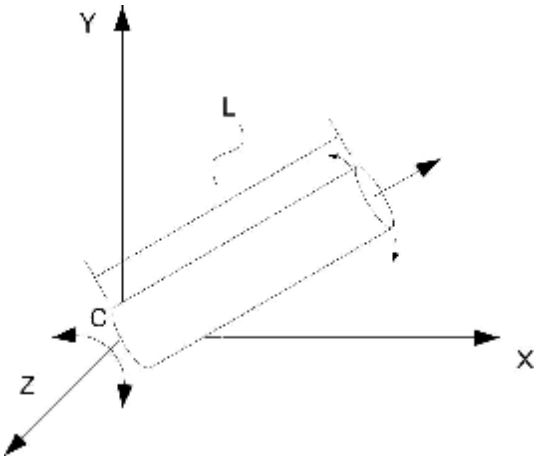


Figura B.2 - Inércia rotacional sobre o eixo Z

Para um corpo cilíndrico oco podemos definir a inércia em relação ao eixo z da seguinte forma:

$$I_z = \int r_m{}^2 dm = \int (x^2 + d^2) dm = \int (x^2 + d^2)\, p dv$$

Transformando para coordenadas polares:

$$p\int_v x^2 (r\, dx\, dr\, dq) + p\int_v r^2 sen^2 q\, (r\, dx\, dr\, dq)$$

$$p\int_{x=0}^{L} x^2 dx \int_{r=0}^{R} r dr \int_{q=0}^{2p} dq + p\int_{x=0}^{L} dx \int_{r=0}^{R} r^3 dr \int_{q=0}^{2p} sen^2 q\, dq$$

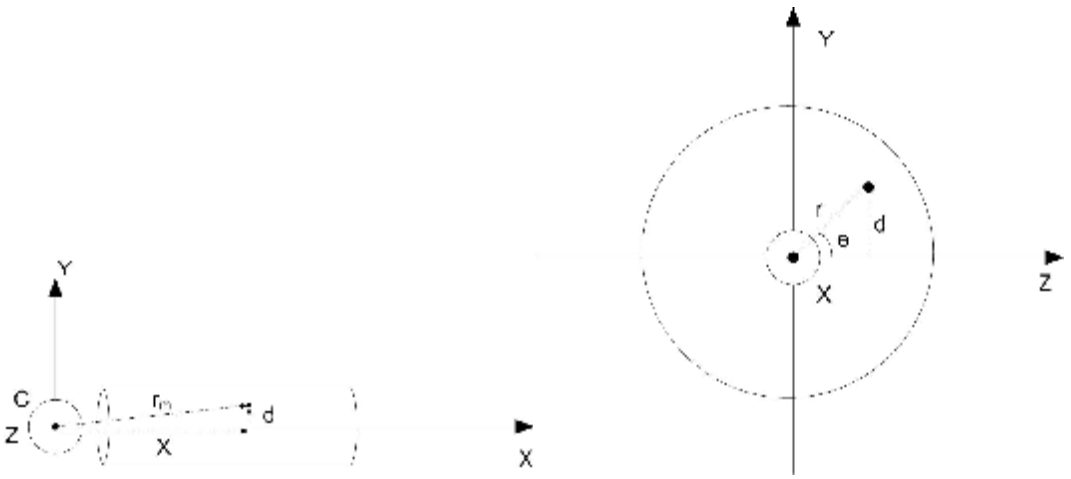


Figura B.3 - Posição de uma partícula no cilindro.

$$p\frac{L^3}{3}\frac{R^2}{2}2p + L\frac{R^4}{4}p = pLR^2p\,\frac{L^2}{3} + pLR^2p\,\frac{R^2}{4} = M\left(\frac{L^2}{3}+\frac{R^2}{4}\right)$$

B.4 Dinâmica de Lagrange

Para desenvolver as equações da física no problema de otimização – no caso representado nas restrições – o sistema de equações foi modelado usando a dinâmica de Langrange.

Na física Lagrangiana, o sistema é definido da seguinte forma:

$$L \equiv T - V \tag{B.11}$$

Onde T é a energia cinética total e V é a energia potencial total. Dado L considera:

$$L' \equiv L + \frac{df(q,t)}{dt} = L + \dot{q}\frac{\partial f}{\partial q} + \frac{\partial f}{\partial t} \tag{B.12}$$

Onde q é a coordenada generalizada e $\dot{q}$ é a primeira derivada. Temos a equação de Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L'}{\partial \dot{q}}\right) - \frac{\partial L'}{\partial q} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q}. \tag{B.13}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}}\right) - \frac{\partial T}{\partial q} = F_q \tag{B.14}$$

Onde F_q representa o vetor das forças generalizadas

Anexo C

Equações de lagrange na Dinâmica inversa

Para a dinâmica Lagrangiana (Wells, 1967), são definidas equações para cada grau de liberdade.

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_r}\right) - \frac{\partial T}{\partial q_r} = F_{q_r} + F'_{q_r}, \tag{C.1}$$

Onde cada q_r representa as coordenadas generalizadas, os F_{q_r} representam a força generalizada atuante para a coordenada q_r e são aplicadas como as molas, a gravidade, etc. O T representa a energia cinética. O F'_{q_r} representam as forças desconhecidas que irão direcionar a solução para a trajetória de cada uma das coordenada independente $q_1, q_2, ..., q_n$ que varia com o tempo de maneira predeterminado. Os q_r representam as restrições do problema e os F'_{q_r} as forças que controlam o movimento. Convenientemente, escolhe-se uma força de controle para cada coordenada independente, isso é para um sistema com n graus de liberdade tem-se n forças $f_1, f_2, ..., f_n$. Tendo T expresso em n coordenadas independentes, n equações de Lagrange são necessárias para representar o corpo articulado. Já que o movimento é conhecido (cada coordenada é uma função do tempo $q_1 = q_1(t)\, etc$), o lado esquerdo das equações do movimento pode ser expresso em termo de t e várias constantes. Para se obter a solução desse sistema de n equações algébricas fica claro a necessidade de especificar condições inicias para que atinja o resultado esperado.

Anexo D

Modelagem no *Mathematica*

Formulação

Velocidades

v0x=D[x0[t],t]

$$x0'[t]$$

v0y=D[y0[t],t]

$$y0'[t]$$

v1x=v0x+l1 θ1'[t] Sin[θ1[t]]

$$x0'[t] + l1\,\text{Sin}[\theta1[t]]\,\theta1'[t]$$

v1y=v0y+l1 θ1'[t] Cos[θ1[t]]

$$y0'[t] + l1\,\text{Cos}[\theta1[t]]\,\theta1'[t]$$

v2x=v1x+l2 θ2'[t] Sin[θ2[t]]

$$x0'[t] + l1\,\text{Sin}[\theta1[t]]\,\theta1'[t] + l2\,\text{Sin}[\theta2[t]]\,\theta2'[t]$$

v2y=v1y+l2 θ2'[t] Cos[θ2[t]]

$$y0'[t] + l1\,\text{Cos}[\theta1[t]]\,\theta1'[t] + l2\,\text{Cos}[\theta2[t]]\,\theta2'[t]$$

Energia Cinética

T1 = 1/2 (m1 (v0x^2 + v0y^2) + m1 l1 θ1'[t] (v0y Cos[θ1[t]] + v0x Sin[θ1[t]]) + θ1'[t]^2 I1)

$$\frac{1}{2}\,(m1\,(x0'[t]^2 + y0'[t]^2) + l1\,m1\,(\text{Sin}[\theta1[t]]\,x0'[t] + \text{Cos}[\theta1[t]]\,y0'[t])\,\theta1'[t] + I1\,\theta1'[t]^2)$$

T2 = 1/2 (m2 (v1x^2 + v1y^2) + m2 l2 θ2'[t] (v1y Cos[θ2[t]] + v1x Sin[θ2[t]]) + θ2'[t]^2 I2)

$$\frac{1}{2}\,(m2\,((y0'[t] + l1\,\text{Cos}[\theta1[t]]\,\theta1'[t])^2 + (x0'[t] + l1\,\text{Sin}[\theta1[t]]\,\theta1'[t])^2) + l2\,m2\,(\text{Cos}[\theta2[t]]\,(y0'[t] + l1\,\text{Cos}[\theta1[t]]\,\theta1'[t]) + \text{Sin}[\theta2[t]]\,(x0'[t] + l1\,\text{Sin}[\theta1[t]]\,\theta1'[t]))\,\theta2'[t] + I2\,\theta2'[t]^2)$$

T3 = 1/2 (m3 (v2x^2 + v2y^2) + m3 l3 θ3'[t] (v2y Cos[θ3[t]] + v2x Sin[θ3[t]]) + θ3'[t]^2 I3)

$$\frac{1}{2}\,(m3\,((y0'[t] + l1\,\text{Cos}[\theta1[t]]\,\theta1'[t] + l2\,\text{Cos}[\theta2[t]]\,\theta2'[t])^2 + (x0'[t] + l1\,\text{Sin}[\theta1[t]]\,\theta1'[t] + l2\,\text{Sin}[\theta2[t]]\,\theta2'[t])^2) + l3\,m3\,(\text{Cos}[\theta3[t]]\,(y0'[t] + l1\,\text{Cos}[\theta1[t]]\,\theta1'[t] + l2\,\text{Cos}[\theta2[t]]\,\theta2'[t]) + \text{Sin}[\theta3[t]]\,(x0'[t] + l1\,\text{Sin}[\theta1[t]]\,\theta1'[t] + l2\,\text{Sin}[\theta2[t]]\,\theta2'[t]))\,\theta3'[t] + I3\,\theta3'[t]^2)$$

T=T1+T2+T3

$$\frac{1}{2}\,(m1\,(x0'[t]^2+y0'[t]^2)+l1\,m1\,(\text{Sin}[\theta1[t]]\,x0'[t]+\text{Cos}[\theta1[t]]\,y0'[t])\,\theta1'[t]+I1\,\theta1'[t]^2)+$$
$$\frac{1}{2}\,(m2\,((y0'[t]+l1\,\text{Cos}[\theta1[t]]\,\theta1'[t])^2+(x0'[t]+l1\,\text{Sin}[\theta1[t]]\,\theta1'[t])^2)+$$
$$l2\,m2\,(\text{Cos}[\theta2[t]]\,(y0'[t]+l1\,\text{Cos}[\theta1[t]]\,\theta1'[t])+\text{Sin}[\theta2[t]]\,(x0'[t]+l1\,\text{Sin}[\theta1[t]]\,\theta1'[t]))$$
$$\theta2'[t]+I2\,\theta2'[t]^2)+\frac{1}{2}\,(m3\,((y0'[t]+l1\,\text{Cos}[\theta1[t]]\,\theta1'[t]+l2\,\text{Cos}[\theta2[t]]\,\theta2'[t])^2+$$
$$(x0'[t]+l1\,\text{Sin}[\theta1[t]]\,\theta1'[t]+l2\,\text{Sin}[\theta2[t]]\,\theta2'[t])^2)+$$
$$l3\,m3\,(\text{Cos}[\theta3[t]]\,(y0'[t]+l1\,\text{Cos}[\theta1[t]]\,\theta1'[t]+l2\,\text{Cos}[\theta2[t]]\,\theta2'[t])+$$
$$\text{Sin}[\theta3[t]]\,(x0'[t]+l1\,\text{Sin}[\theta1[t]]\,\theta1'[t]+l2\,\text{Sin}[\theta2[t]]\,\theta2'[t]))\,\theta3'[t]+I3\,\theta3'[t]^2)$$

Derivadas

Tdx0 = Simplify[D[T,x0[t]]]

$$0$$

Tdx0' = Simplify[D[T,x0'[t]]]

$$(m1+m2+m3)\,x0'[t]+\frac{1}{2}\,(l1\,(m1+2\,(m2+m3))\,\text{Sin}[\theta1[t]]\,\theta1'[t]+l2\,(m2+2\,m3)\,\text{Sin}[\theta2[t]]\,\theta2'[t]+l3\,m3\,\text{Sin}[\theta3[t]]\,\theta3'[t])$$

Tdx0´dt = D[Tdx0',t]

$$(m1+m2+m3)\,x0''[t]+\frac{1}{2}$$
$$(l1\,(m1+2\,(m2+m3))\,\text{Cos}[\theta1[t]]\,\theta1'[t]^2+l2\,(m2+2\,m3)\,\text{Cos}[\theta2[t]]\,\theta2'[t]^2+l3\,m3\,\text{Cos}[\theta3[t]]\,\theta3'[t]^2+$$
$$l1\,(m1+2\,(m2+m3))\,\text{Sin}[\theta1[t]]\,\theta1''[t]+l2\,(m2+2\,m3)\,\text{Sin}[\theta2[t]]\,\theta2''[t]+l3\,m3\,\text{Sin}[\theta3[t]]\,\theta3''[t])$$

Tdy0 = Simplify[D[T,y0[t]]]

$$0$$

Tdy0' = Simplify[D[T,y0'[t]]]

$$(m1+m2+m3)\,y0'[t]+\frac{1}{2}\,(l1\,(m1+2\,(m2+m3))\,\text{Cos}[\theta1[t]]\,\theta1'[t]+l2\,(m2+2\,m3)\,\text{Cos}[\theta2[t]]\,\theta2'[t]+l3\,m3\,\text{Cos}[\theta3[t]]\,\theta3'[t])$$

Tdy0´dt = D[Tdy0',t]

$$(m1+m2+m3)\,y0''[t]+\frac{1}{2}\,(-l1\,(m1+2\,(m2+m3))\,\text{Sin}[\theta1[t]]\,\theta1'[t]^2-$$
$$l2\,(m2+2\,m3)\,\text{Sin}[\theta2[t]]\,\theta2'[t]^2-l3\,m3\,\text{Sin}[\theta3[t]]\,\theta3'[t]^2+$$
$$l1\,(m1+2\,(m2+m3))\,\text{Cos}[\theta1[t]]\,\theta1''[t]+l2\,(m2+2\,m3)\,\text{Cos}[\theta2[t]]\,\theta2''[t]+l3\,m3\,\text{Cos}[\theta3[t]]\,\theta3''[t])$$

Tdθ1 = Simplify[D[T,θ1[t]]]

$$\frac{1}{2}\,l1\,\theta1'[t]\,((m1+2\,(m2+m3))\,\text{Cos}[\theta1[t]]\,x0'[t]-(m1+2\,(m2+m3))\,\text{Sin}[\theta1[t]]\,y0'[t]-$$
$$l2\,m2\,\text{Sin}[\theta1[t]-\theta2[t]]\,\theta2'[t]-2\,l2\,m3\,\text{Sin}[\theta1[t]-\theta2[t]]\,\theta2'[t]-l3\,m3\,\text{Sin}[\theta1[t]-\theta3[t]]\,\theta3'[t])$$

Tdθ1' = Simplify[D[T,θ1'[t]]]

$$\frac{1}{2}\,(l1\,(m1+2\,(m2+m3))\,\text{Sin}[\theta1[t]]\,x0'[t]+l1\,(m1+2\,(m2+m3))\,\text{Cos}[\theta1[t]]\,y0'[t]+$$
$$2\,I1\,\theta1'[t]+2\,l1^2\,m2\,\theta1'[t]+2\,l1^2\,m3\,\theta1'[t]+l1\,l2\,m2\,\text{Cos}[\theta1[t]-\theta2[t]]\,\theta2'[t]+$$
$$2\,l1\,l2\,m3\,\text{Cos}[\theta1[t]-\theta2[t]]\,\theta2'[t]+l1\,l3\,m3\,\text{Cos}[\theta1[t]-\theta3[t]]\,\theta3'[t])$$

Tdθ1´dt = D[Tdθ1',t]

$$\frac{1}{2}\,(l1\,(m1+2\,(m2+m3))\,\text{Cos}[\theta 1[t]]\,x0'[t]\,\theta 1'[t]-l1\,(m1+2\,(m2+m3))\,\text{Sin}[\theta 1[t]]\,y0'[t]\,\theta 1'[t]+$$
$$l1\,l2\,m2\,\text{Sin}[\theta 1[t]-\theta 2[t]]\,\theta 2'[t]\,(-\theta 1'[t]+\theta 2'[t])+2\,l1\,l2\,m3\,\text{Sin}[\theta 1[t]-\theta 2[t]]$$
$$\theta 2'[t]\,(-\theta 1'[t]+\theta 2'[t])+l1\,l3\,m3\,\text{Sin}[\theta 1[t]-\theta 3[t]]\,\theta 3'[t]\,(-\theta 1'[t]+\theta 3'[t])+$$
$$l1\,(m1+2\,(m2+m3))\,\text{Sin}[\theta 1[t]]\,x0''[t]+l1\,(m1+2\,(m2+m3))\,\text{Cos}[\theta 1[t]]\,y0''[t]+$$
$$2\,I1\,\theta 1''[t]+2\,l1^2\,m2\,\theta 1''[t]+2\,l1^2\,m3\,\theta 1''[t]+l1\,l2\,m2\,\text{Cos}[\theta 1[t]-\theta 2[t]]\,\theta 2''[t]+$$
$$2\,l1\,l2\,m3\,\text{Cos}[\theta 1[t]-\theta 2[t]]\,\theta 2''[t]+l1\,l3\,m3\,\text{Cos}[\theta 1[t]-\theta 3[t]]\,\theta 3''[t])$$

Tdθ2 = Simplify[D[T,θ2[t]]]

$$\frac{1}{2}\,l2\,\theta 2'[t]\,((m2+2m3)\,\text{Cos}[\theta 2[t]]\,x0'[t]-(m2+2m3)\,\text{Sin}[\theta 2[t]]\,y0'[t]+$$
$$l1\,m2\,\text{Sin}[\theta 1[t]-\theta 2[t]]\,\theta 1'[t]+2\,l1\,m3\,\text{Sin}[\theta 1[t]-\theta 2[t]]\,\theta 1'[t]-l3\,m3\,\text{Sin}[\theta 2[t]-\theta 3[t]]\,\theta 3'[t])$$

Tdθ2' = Simplify[D[T,θ2'[t]]]

$$\frac{1}{2}$$
$$(l2\,(m2+2m3)\,\text{Sin}[\theta 2[t]]\,x0'[t]+l2\,(m2+2m3)\,\text{Cos}[\theta 2[t]]\,y0'[t]+l1\,l2\,m2\,\text{Cos}[\theta 1[t]-\theta 2[t]]\,\theta 1'[t]+$$
$$2\,l1\,l2\,m3\,\text{Cos}[\theta 1[t]-\theta 2[t]]\,\theta 1'[t]+2\,I2\,\theta 2'[t]+2\,l2^2\,m3\,\theta 2'[t]+l2\,l3\,m3\,\text{Cos}[\theta 2[t]-\theta 3[t]]\,\theta 3'[t])$$

Tdθ2´dt = D[Tdθ2',t]

$$\frac{1}{2}\,(l2\,(m2+2\,m3)\,\text{Cos}[\theta 2[t]]\,x0'[t]\,\theta 2'[t]-l2\,(m2+2\,m3)\,\text{Sin}[\theta 2[t]]\,y0'[t]\,\theta 2'[t]+$$
$$l1\,l2\,m2\,\text{Sin}[\theta 1[t]-\theta 2[t]]\,\theta 1'[t]\,(-\theta 1'[t]+\theta 2'[t])+2\,l1\,l2\,m3\,\text{Sin}[\theta 1[t]-\theta 2[t]]$$
$$\theta 1'[t]\,(-\theta 1'[t]+\theta 2'[t])+l2\,l3\,m3\,\text{Sin}[\theta 2[t]-\theta 3[t]]\,\theta 3'[t]\,(-\theta 2'[t]+\theta 3'[t])+$$
$$l2\,(m2+2\,m3)\,\text{Sin}[\theta 2[t]]\,x0''[t]+l2\,(m2+2\,m3)\,\text{Cos}[\theta 2[t]]\,y0''[t]+$$
$$l1\,l2\,m2\,\text{Cos}[\theta 1[t]-\theta 2[t]]\,\theta 1''[t]+2\,l1\,l2\,m3\,\text{Cos}[\theta 1[t]-\theta 2[t]]\,\theta 1''[t]+$$
$$2\,I2\,\theta 2''[t]+2\,l2^2\,m3\,\theta 2''[t]+l2\,l3\,m3\,\text{Cos}[\theta 2[t]-\theta 3[t]]\,\theta 3''[t])$$

Tdθ3 = Simplify[D[T,θ3[t]]]

$$\frac{1}{2}\,l3\,m3$$
$$(\text{Cos}[\theta 3[t]]\,x0'[t]-\text{Sin}[\theta 3[t]]\,y0'[t]+l1\,\text{Sin}[\theta 1[t]-\theta 3[t]]\,\theta 1'[t]+l2\,\text{Sin}[\theta 2[t]-\theta 3[t]]\,\theta 2'[t])$$
$$\theta 3'[t]$$

Tdθ3' = Simplify[D[T,θ3'[t]]]

$$\frac{1}{2}\,(l3\,m3\,\text{Sin}[\theta 3[t]]\,x0'[t]+l3\,m3\,\text{Cos}[\theta 3[t]]\,y0'[t]+$$
$$l1\,l3\,m3\,\text{Cos}[\theta 1[t]-\theta 3[t]]\,\theta 1'[t]+l2\,l3\,m3\,\text{Cos}[\theta 2[t]-\theta 3[t]]\,\theta 2'[t]+2\,I3\,\theta 3'[t])$$

Tdθ3´dt = D[Tdθ3',t]

$$\frac{1}{2}\,(l3\,m3\,\text{Cos}[\theta 3[t]]\,x0'[t]\,\theta 3'[t]-l3\,m3\,\text{Sin}[\theta 3[t]]\,y0'[t]\,\theta 3'[t]+$$
$$l1\,l3\,m3\,\text{Sin}[\theta 1[t]-\theta 3[t]]\,\theta 1'[t]\,(-\theta 1'[t]+\theta 3'[t])+l2\,l3\,m3\,\text{Sin}[\theta 2[t]-\theta 3[t]]$$
$$\theta 2'[t]\,(-\theta 2'[t]+\theta 3'[t])+l3\,m3\,\text{Sin}[\theta 3[t]]\,x0''[t]+l3\,m3\,\text{Cos}[\theta 3[t]]\,y0''[t]+$$
$$l1\,l3\,m3\,\text{Cos}[\theta 1[t]-\theta 3[t]]\,\theta 1''[t]+l2\,l3\,m3\,\text{Cos}[\theta 2[t]-\theta 3[t]]\,\theta 2''[t]+2\,I3\,\theta 3''[t])$$

Centro de Massa

cy1=y0[t]+l1/2 Sin[θ1[t]]

$$\frac{1}{2}\,l1\,\text{Sin}[\theta 1[t]]+y0[t]$$

cy2=y0[t]+l1 Sin[θ1[t]]+l2/2 Sin[θ2[t]]

$$l1\,\text{Sin}[\theta 1[t]]+\frac{1}{2}\,l2\,\text{Sin}[\theta 2[t]]+y0[t]$$

cy3=y0[t]+l1 Sin[θ1[t]]+l2 Sin[θ2[t]]+l3/2 Sin[θ3[t]]

$$l1\,\text{Sin}[\theta1[t]] + l2\,\text{Sin}[\theta2[t]] + \frac{1}{2}\,l3\,\text{Sin}[\theta3[t]] + y0[t]$$

Força Generalizada

Fx0=0

0

Fy0=-m1 g D[cy1,y0[t]]-m2 g D[cy2,y0[t]]-m3 g D[cy3,y0[t]]

-g m1-g m2-g m3

Fθ1=-m1 g D[cy1,θ1[t]]-m2 g D[cy2,θ1[t]]-m3 g D[cy3,θ1[t]]-k1(θ2[t]-θ1[t])

$$-\frac{1}{2}\,g\,l1\,m1\,\text{Cos}[\theta1[t]] - g\,l1\,m2\,\text{Cos}[\theta1[t]] - g\,l1\,m3\,\text{Cos}[\theta1[t]] - k1\,(\theta1[t] - \theta2[t])$$

Fθ2=-m1 g D[cy1,θ2[t]]-m2 g D[cy2,θ2[t]]-m3 g D[cy3,θ2[t]]+k1(θ2[t]-θ1[t])-k2(θ3[t]-θ2[t])

$$-\frac{1}{2}\,g\,l2\,m2\,\text{Cos}[\theta2[t]] - g\,l2\,m3\,\text{Cos}[\theta2[t]] + k1\,(\theta1[t] - \theta2[t]) - k2\,(\theta2[t] - \theta3[t])$$

Fθ3=-m1 g D[cy1,θ3[t]]-m2 g D[cy2,θ3[t]]-m3 g D[cy3,θ3[t]]+k2(θ3[t]-θ2[t])

$$-\frac{1}{2}\,g\,l3\,m3\,\text{Cos}[\theta3[t]] + k2\,(\theta2[t] - \theta3[t])$$

Equação de Lagrange

eq1=Simplify[Tdx0´dt-Tdx0-Fx0==0]

$$(m1+m2+m3)\,x0''[t] + \frac{1}{2}\,(l1\,(m1+2\,(m2+m3))\,\text{Cos}[\theta1[t]]\,\theta1'[t]^2 + l2\,(m2+2\,m3)\,\text{Cos}[\theta2[t]]\,\theta2'[t]^2 + l3\,m3\,\text{Cos}[\theta3[t]]\,\theta3'[t]^2 + l1\,(m1+2\,(m2+m3))\,\text{Sin}[\theta1[t]]\,\theta1''[t] + l2\,(m2+2\,m3)\,\text{Sin}[\theta2[t]]\,\theta2''[t] + l3\,m3\,\text{Sin}[\theta3[t]]\,\theta3''[t]) == 0$$

eq2=Simplify[Tdy0´dt-Tdy0-Fy0==0]

$$g\,m1 + g\,m2 + g\,m3 + (m1+m2+m3)\,y0''[t] + \frac{1}{2}\,(-l1\,(m1+2\,(m2+m3))\,\text{Sin}[\theta1[t]]\,\theta1'[t]^2 - l2\,(m2+2\,m3)\,\text{Sin}[\theta2[t]]\,\theta2'[t]^2 - l3\,m3\,\text{Sin}[\theta3[t]]\,\theta3'[t]^2 + l1\,(m1+2\,(m2+m3))\,\text{Cos}[\theta1[t]]\,\theta1''[t] + l2\,(m2+2\,m3)\,\text{Cos}[\theta2[t]]\,\theta2''[t] + l3\,m3\,\text{Cos}[\theta3[t]]\,\theta3''[t]) == 0$$

eq3=Simplify[Tdθ1´dt-Tdθ1-Fθ1==0]

$$g\,l1\,m1\,\text{Cos}[\theta1[t]] + 2\,g\,l1\,m2\,\text{Cos}[\theta1[t]] + 2\,g\,l1\,m3\,\text{Cos}[\theta1[t]] + 2\,k1\,\theta1[t] + l1\,l2\,(m2+2\,m3)\,\text{Sin}[\theta1[t]-\theta2[t]]\,\theta2'[t]^2 + l1\,l3\,m3\,\text{Sin}[\theta1[t]-\theta3[t]]\,\theta3'[t]^2 + l1\,m1\,\text{Sin}[\theta1[t]]\,x0''[t] + 2\,l1\,m2\,\text{Sin}[\theta1[t]]\,x0''[t] + 2\,l1\,m3\,\text{Sin}[\theta1[t]]\,x0''[t] + l1\,m1\,\text{Cos}[\theta1[t]]\,y0''[t] + 2\,l1\,m2\,\text{Cos}[\theta1[t]]\,y0''[t] + 2\,l1\,m3\,\text{Cos}[\theta1[t]]\,y0''[t] + 2\,I1\,\theta1''[t] + 2\,l1^2\,m2\,\theta1''[t] + 2\,l1^2\,m3\,\theta1''[t] + l1\,l2\,m2\,\text{Cos}[\theta1[t]-\theta2[t]]\,\theta2''[t] + 2\,l1\,l2\,m3\,\text{Cos}[\theta1[t]-\theta2[t]]\,\theta2''[t] + l1\,l3\,m3\,\text{Cos}[\theta1[t]-\theta3[t]]\,\theta3''[t] == 2\,k1\,\theta2[t]$$

eq4=Simplify[Tdθ2´dt-Tdθ2-Fθ2==0]

$$g\,l2\,m2\,\text{Cos}[\theta2[t]] + 2\,g\,l2\,m3\,\text{Cos}[\theta2[t]] + 2\,(k1+k2)\,\theta2[t] + l2\,l3\,m3\,\text{Sin}[\theta2[t]-\theta3[t]]\,\theta3'[t]^2 + l2\,m2\,\text{Sin}[\theta2[t]]\,x0''[t] + 2\,l2\,m3\,\text{Sin}[\theta2[t]]\,x0''[t] + l2\,m2\,\text{Cos}[\theta2[t]]\,y0''[t] + 2\,l2\,m3\,\text{Cos}[\theta2[t]]\,y0''[t] + l1\,l2\,m2\,\text{Cos}[\theta1[t]-\theta2[t]]\,\theta1''[t] + 2\,l1\,l2\,m3\,\text{Cos}[\theta1[t]-\theta2[t]]\,\theta1''[t] + 2\,I2\,\theta2''[t] + 2\,l2^2\,m3\,\theta2''[t] + l2\,l3\,m3\,\text{Cos}[\theta2[t]-\theta3[t]]\,\theta3''[t] == 2\,k1\,\theta1[t] + 2\,k2\,\theta3[t] + l1\,l2\,(m2+2\,m3)\,\text{Sin}[\theta1[t]-\theta2[t]]\,\theta1'[t]^2$$

```
eq5=Simplify[Tdθ3´dt-Tdθ3-Fθ3==0]
```

$$g\,l3\,m3\,\text{Cos}[\theta3[t]] + 2\,k2\,\theta3[t] + l3\,m3\,\text{Sin}[\theta3[t]]\,x0''[t] + l3\,m3\,\text{Cos}[\theta3[t]]\,y0''[t] + l1\,l3\,m3\,\text{Cos}[\theta1[t] - \theta3[t]]\,\theta1''[t] + l2\,l3\,m3\,\text{Cos}[\theta2[t] - \theta3[t]]\,\theta2''[t] + 2\,I3\,\theta3''[t] == 2\,k2\,\theta2[t] + l1\,l3\,m3\,\text{Sin}[\theta1[t] - \theta3[t]]\,\theta1'[t]^2 + l2\,l3\,m3\,\text{Sin}[\theta2[t] - \theta3[t]]\,\theta2'[t]^2$$

Código

```
ex1=eq1/.{θ1→th1
,θ2→th2,θ3→th3,θ1'→th1',θ2'→th2',θ3'→th3',θ1''→th1'',θ2''
→th2'',θ3''→th3''}
```

$$\frac{1}{2}\,(l1\,(m1 + 2\,(m2 + m3))\,\text{Cos}[th1[t]]\,th1'[t]^2 + l2\,(m2 + 2\,m3)\,\text{Cos}[th2[t]]\,th2'[t]^2 + l3\,m3\,\text{Cos}[th3[t]]\,th3'[t]^2 + l1\,(m1 + 2\,(m2 + m3))\,\text{Sin}[th1[t]]\,th1''[t] + l2\,(m2 + 2\,m3)\,\text{Sin}[th2[t]]\,th2''[t] + l3\,m3\,\text{Sin}[th3[t]]\,th3''[t]) + (m1 + m2 + m3)\,x0''[t] == 0$$

```
CForm[ex1]
(l1*(m1 + 2*(m2 +
m3))*Cos(th1(t))*Power(Derivative(1)(th1)(t),2) +
       l2*(m2 +
2*m3)*Cos(th2(t))*Power(Derivative(1)(th2)(t),2) +
       l3*m3*Cos(th3(t))*Power(Derivative(1)(th3)(t),2) +
       l1*(m1 + 2*(m2 +
m3))*Sin(th1(t))*Derivative(2)(th1)(t) +
       l2*(m2 + 2*m3)*Sin(th2(t))*Derivative(2)(th2)(t) +
       l3*m3*Sin(th3(t))*Derivative(2)(th3)(t))/2. + (m1 +
m2 + m3)*Derivative(2)(x0)(t) == 0
```

```
ex2=eq2/.{θ1→th1
,θ2→th2,θ1'→th1',θ2'→th2',θ1''→th1'',θ2''→th2''}
```

$$g\,m1 + g\,m2 + g\,m3 + (m1 + m2 + m3)\,y0''[t] + \frac{1}{2}\,(-l1\,(m1 + 2\,(m2 + m3))\,\text{Sin}[th1[t]]\,th1'[t]^2 - l2\,(m2 + 2\,m3)\,\text{Sin}[th2[t]]\,th2'[t]^2 - l3\,m3\,\text{Sin}[\theta3[t]]\,\theta3'[t]^2 + l1\,(m1 + 2\,(m2 + m3))\,\text{Cos}[th1[t]]\,th1''[t] + l2\,(m2 + 2\,m3)\,\text{Cos}[th2[t]]\,th2''[t] + l3\,m3\,\text{Cos}[\theta3[t]]\,\theta3''[t]) == 0$$

```
ex3=eq3/.{θ1→th1
,θ2→th2,θ1'→th1',θ2'→th2',θ1''→th1'',θ2''→th2''}
```

$$g\,l1\,m1\,\text{Cos}[th1[t]] + 2\,g\,l1\,m2\,\text{Cos}[th1[t]] + 2\,g\,l1\,m3\,\text{Cos}[th1[t]] + 2\,k1\,th1[t] + l1\,l2\,(m2 + 2\,m3)\,\text{Sin}[th1[t] - th2[t]]\,th2'[t]^2 + l1\,l3\,m3\,\text{Sin}[th1[t] - \theta3[t]]\,\theta3'[t]^2 + 2\,I1\,th1''[t] + 2\,l1^2\,m2\,th1''[t] + 2\,l1^2\,m3\,th1''[t] + l1\,l2\,m2\,\text{Cos}[th1[t] - th2[t]]\,th2''[t] + 2\,l1\,l2\,m3\,\text{Cos}[th1[t] - th2[t]]\,th2''[t] + l1\,m1\,\text{Sin}[th1[t]]\,x0''[t] + 2\,l1\,m2\,\text{Sin}[th1[t]]\,x0''[t] + 2\,l1\,m3\,\text{Sin}[th1[t]]\,x0''[t] + l1\,m1\,\text{Cos}[th1[t]]\,y0''[t] + 2\,l1\,m2\,\text{Cos}[th1[t]]\,y0''[t] + 2\,l1\,m3\,\text{Cos}[th1[t]]\,y0''[t] + l1\,l3\,m3\,\text{Cos}[th1[t] - \theta3[t]]\,\theta3''[t] == 2\,k1\,th2[t]$$

```
ex4=eq4/.{θ1→th1
,θ2→th2,θ1'→th1',θ2'→th2',θ1''→th1'',θ2''→th2''}
```

$$g\,l2\,m2\,\text{Cos}[th2[t]] + 2\,g\,l2\,m3\,\text{Cos}[th2[t]] + 2\,(k1 + k2)\,th2[t] + l2\,l3\,m3\,\text{Sin}[th2[t] - \theta3[t]]\,\theta3'[t]^2 + l1\,l2\,m2\,\text{Cos}[th1[t] - th2[t]]\,th1''[t] + 2\,l1\,l2\,m3\,\text{Cos}[th1[t] - th2[t]]\,th1''[t] + 2\,I2\,th2''[t] + 2\,l2^2\,m3\,th2''[t] + l2\,m2\,\text{Sin}[th2[t]]\,x0''[t] + 2\,l2\,m3\,\text{Sin}[th2[t]]\,x0''[t] + l2\,m2\,\text{Cos}[th2[t]]\,y0''[t] + 2\,l2\,m3\,\text{Cos}[th2[t]]\,y0''[t] + l2\,l3\,m3\,\text{Cos}[th2[t] - \theta3[t]]\,\theta3''[t] == 2\,k1\,th1[t] + 2\,k2\,\theta3[t] + l1\,l2\,(m2 + 2\,m3)\,\text{Sin}[th1[t] - th2[t]]\,th1'[t]^2$$

Anexo E

Um algoritmo simples para SQP na otimização de figuras articuladas

Simbolicamente, o vetor do estado é representado por S, C o vetor das restrições que não podem ser violadas, e R(S) a função objetivo. Deve-se achar S, através de

$J = \frac{\partial C}{\partial S}$ onde J é o jacobiano da função de restrição

$H = \frac{\partial^2 R}{\partial S^2}$ é o hessiano da função objetivo

O processo envolve duas fases. A primeira acha as mudanças locais no vetor do estado que minimiza as funções objetivas sem considerar as violações das restrições. Para minimizar a função objetivo, um conjunto de derivativas é encontrado R'(S)=0. Usa-se a série de Taylor de segunda ordem para aproximar a expansão, aproximando R de S:

$$R'(S) = 0 = \frac{\partial R}{\partial S} + \frac{\partial^2 R}{\partial S^2}(X - S)$$

Fazendo $S_a = X - S$, soluciona

$$-\frac{\partial R}{\partial S} = HS_a$$

Apesar de que R(S) é minimizada em $S + S_a$, o espaço das restrições não pode ser violado.

Tem-se que achar uma segunda mudança no vetor do estado S_b que preserve a minimização de R enquanto eliminam-se as violações $S + S_a$. Um segundo passo é feito projetando o S_a no espaço nulo das restrições jabobianas

$$J(S_a + S_b) + C = 0$$

Nota-se que o vetor das restrições C e o jacobiano J são calculados pelo $S + S_a$. Resolvendo para S_b no

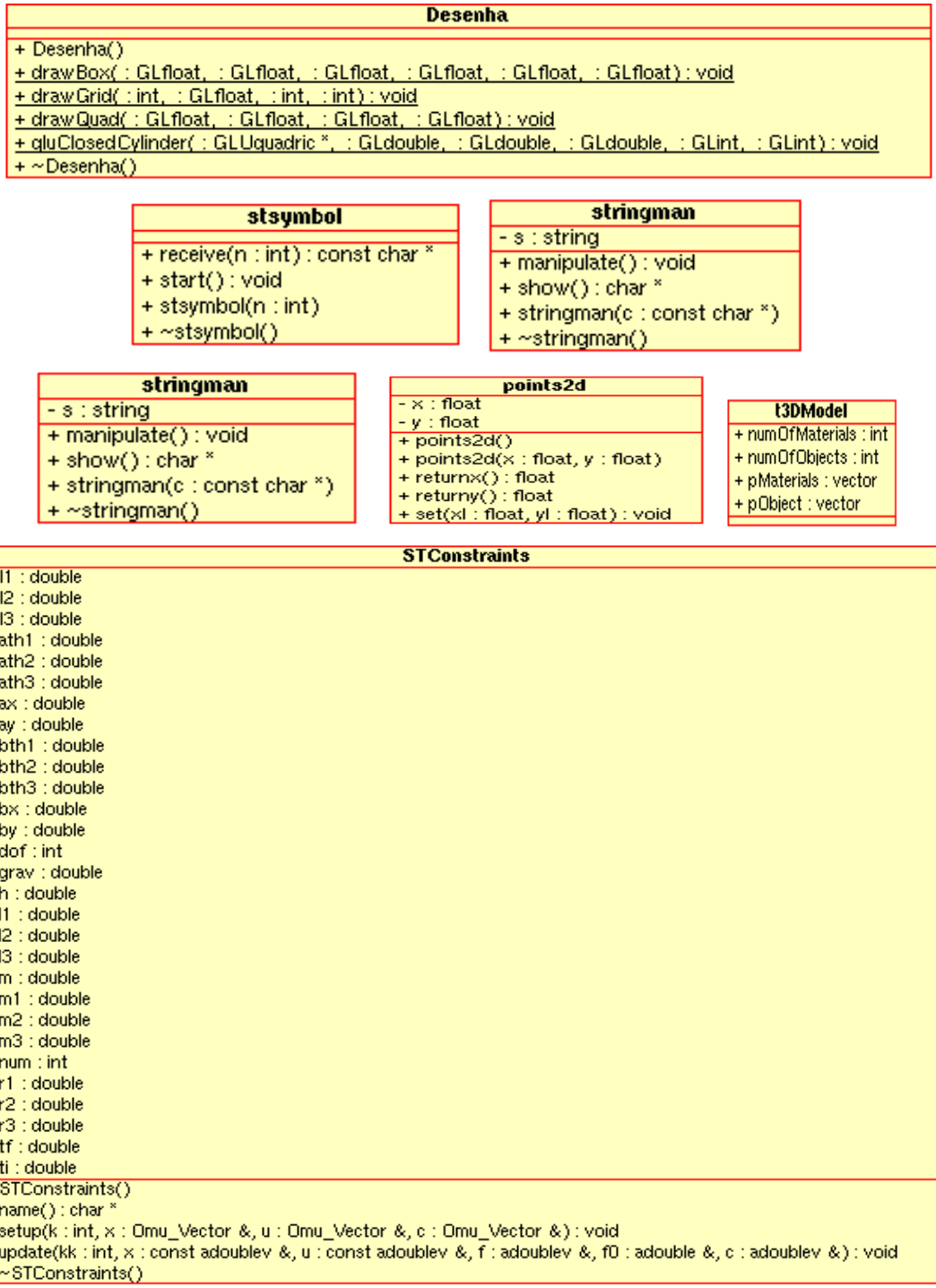
$$-C = J(S_a + S_b)$$

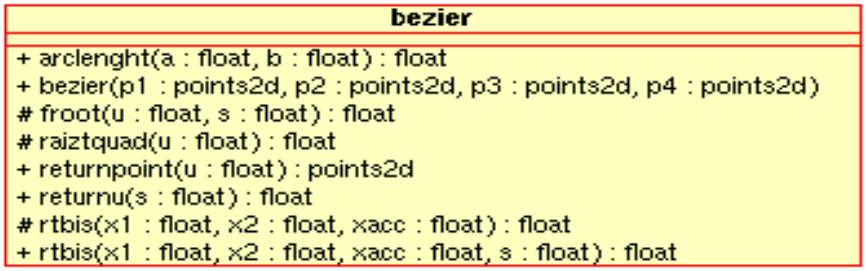
Leva C para zero.

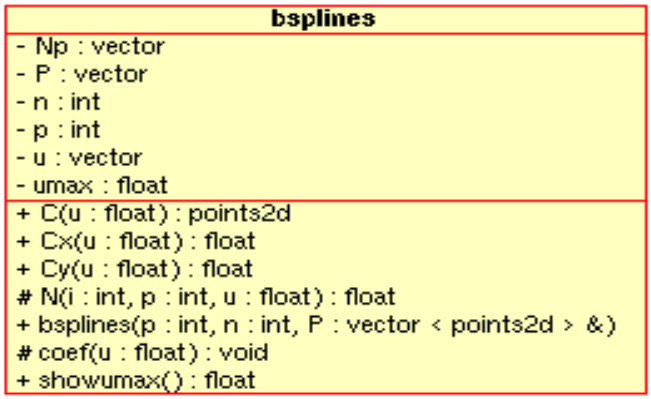
O novo valor do S é incrementado de $(S_a + S_b)$. O Algoritmo calcula o novo vetor do espaço até que uma redução no calculo dos objetivos viole as restrições.

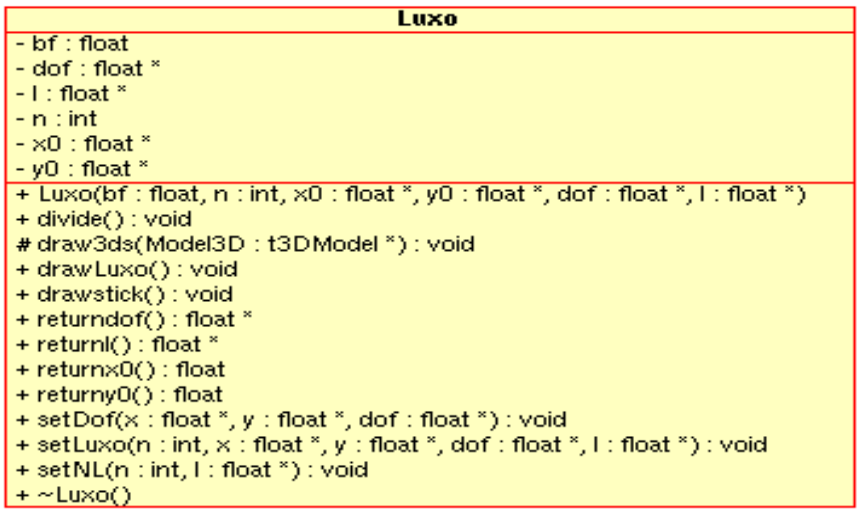
Anexo F

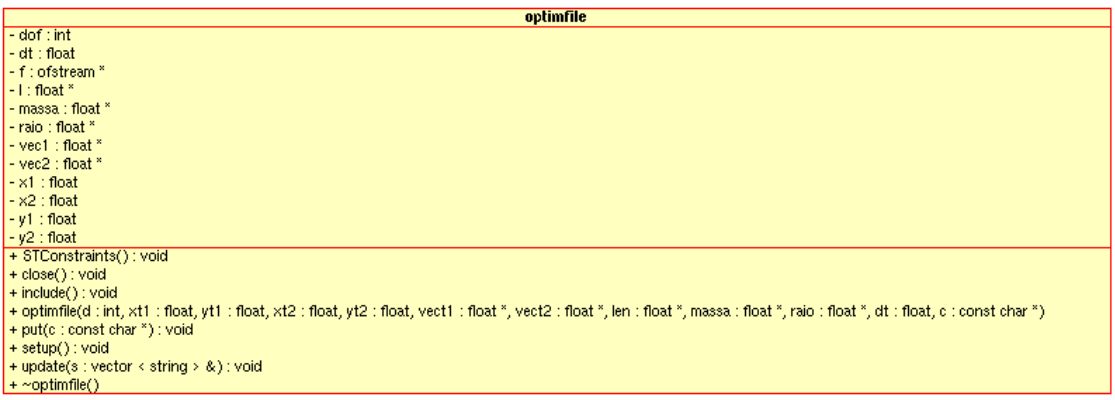
Classes Principais do Sistema

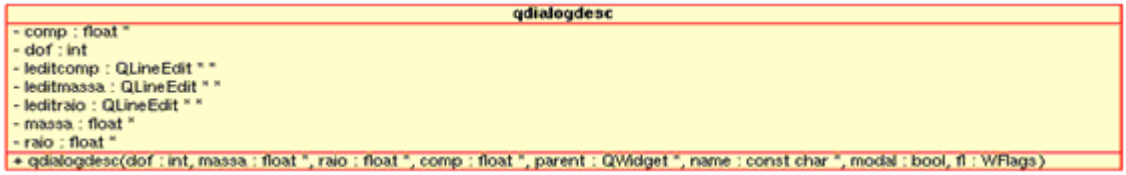


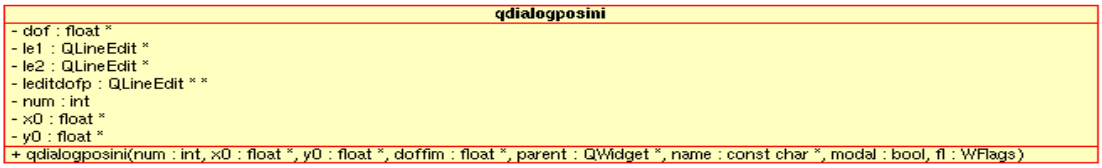

bezier
+ arclenght(a : float, b : float) : float
+ bezier(p1 : points2d, p2 : points2d, p3 : points2d, p4 : points2d)
# froot(u : float, s : float) : float
# raiztquad(u : float) : float
+ returnpoint(u : float) : points2d
+ returnu(s : float) : float
# rtbis(x1 : float, x2 : float, xacc : float) : float
+ rtbis(x1 : float, x2 : float, xacc : float, s : float) : float

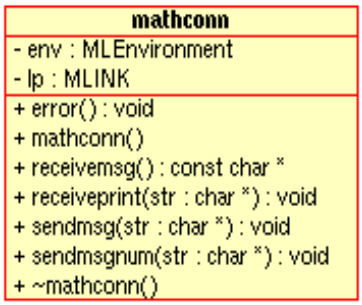

bsplines
- Np : vector
- P : vector
- n : int
- p : int
- u : vector
- umax : float
+ C(u : float) : points2d
+ Cx(u : float) : float
+ Cy(u : float) : float
# N(i : int, p : int, u : float) : float
+ bsplines(p : int, n : int, P : vector < points2d > &)
# coef(u : float) : void
+ showumax() : float

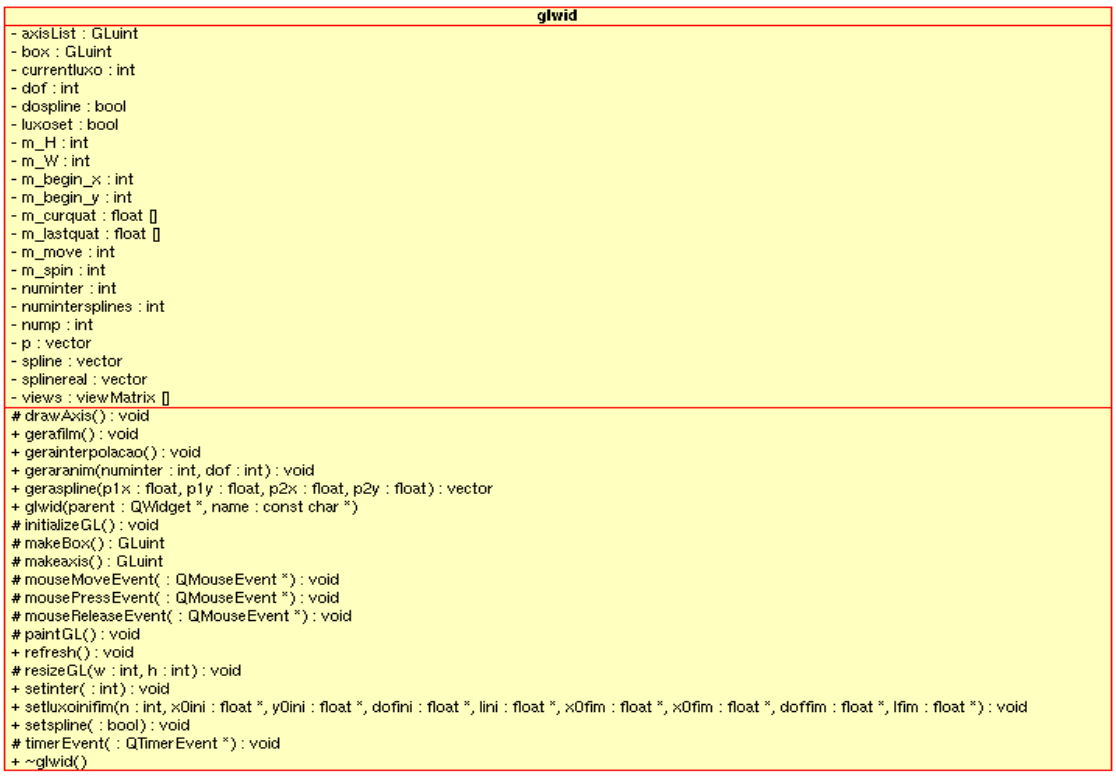

Luxo
- bf : float
- dof : float *
- l : float *
- n : int
- x0 : float *
- y0 : float *
+ Luxo(bf : float, n : int, x0 : float *, y0 : float *, dof : float *, l : float *)
+ divide() : void
# draw3ds(Model3D : t3DModel *) : void
+ drawLuxo() : void
+ drawstick() : void
+ returndof() : float *
+ returnl() : float *
+ returnx0() : float
+ returny0() : float
+ setDof(x : float *, y : float *, dof : float *) : void
+ setLuxo(n : int, x : float *, y : float *, dof : float *, l : float *) : void
+ setNL(n : int, l : float *) : void
+ ~Luxo()

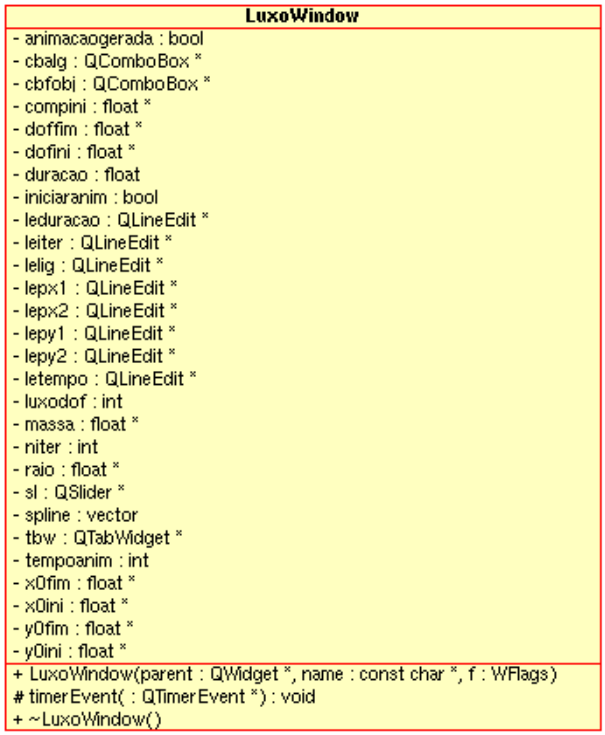

optimfile
- dof : int
- dt : float
- f : ofstream *
- l : float *
- massa : float *
- raio : float *
- vec1 : float *
- vec2 : float *
- x1 : float
- x2 : float
- y1 : float
- y2 : float
+ STConstraints() : void
+ close() : void
+ include() : void
+ optimfile(d : int, xt1 : float, yt1 : float, xt2 : float, yt2 : float, vect1 : float *, vect2 : float *, len : float *, massa : float *, raio : float *, dt : float, c : const char *)
+ put(c : const char *) : void
+ setup() : void
+ update(s : vector < string > &) : void
+ ~optimfile()

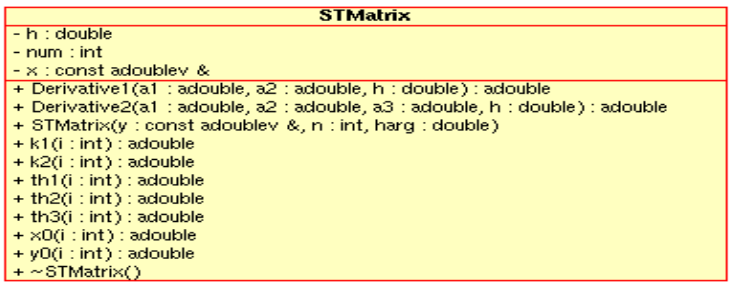

qdialogdesc
- comp : float *
- dof : int
- leditcomp : QLineEdit * *
- leditmassa : QLineEdit * *
- leditraio : QLineEdit * *
- massa : float *
- raio : float *
+ qdialogdesc(dof : int, massa : float *, raio : float *, comp : float *, parent : QWidget *, name : const char *, modal : bool, fl : WFlags)

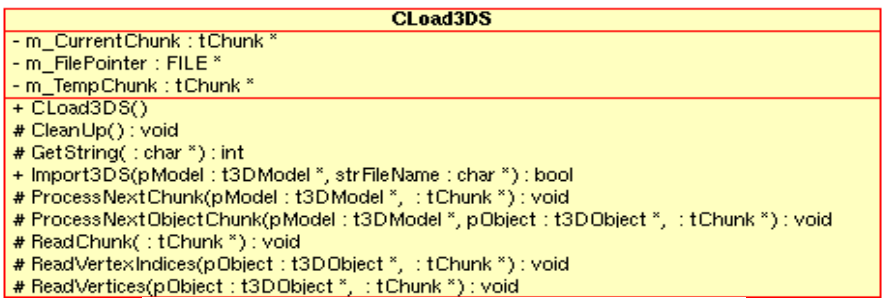

qdialogposini
- dof : float *
- le1 : QLineEdit *
- le2 : QLineEdit *
- leditdofp : QLineEdit * *
- num : int
- x0 : float *
- y0 : float *
+ qdialogposini(num : int, x0 : float *, y0 : float *, doffim : float *, parent : QWidget *, name : const char *, modal : bool, fl : WFlags)

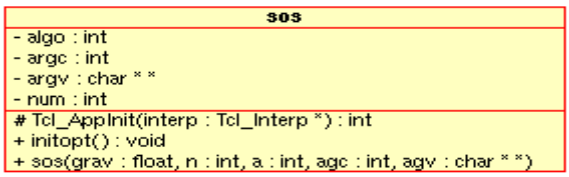

mathconn
- env : MLEnvironment
- lp : MLINK
+ error() : void
+ mathconn()
+ receivemsg() : const char *
+ receiveprint(str : char *) : void
+ sendmsg(str : char *) : void
+ sendmsgnum(str : char *) : void
+ ~mathconn()


glwid
- axisList : GLuint
- box : GLuint
- currentluxo : int
- dof : int
- dospline : bool
- luxoset : bool
- m_H : int
- m_W : int
- m_begin_x : int
- m_begin_y : int
- m_curquat : float []
- m_lastquat : float []
- m_move : int
- m_spin : int
- numinter : int
- numintersplines : int
- nump : int
- p : vector
- spline : vector
- splinereal : vector
- views : viewMatrix []
drawAxis() : void
+ gerafilm() : void
+ gerainterpolacao() : void
+ geraranim(numinter : int, dof : int) : void
+ geraspline(p1x : float, p1y : float, p2x : float, p2y : float) : vector
+ glwid(parent : QWidget *, name : const char *)
# initializeGL() : void
# makeBox() : GLuint
# makeaxis() : GLuint
# mouseMoveEvent(: QMouseEvent *) : void
# mousePressEvent(: QMouseEvent *) : void
# mouseReleaseEvent(: QMouseEvent *) : void
# paintGL() : void
+ refresh() : void
# resizeGL(w : int, h : int) : void
+ setinter(: int) : void
+ setluxoinifim(n : int, x0ini : float *, y0ini : float *, dofini : float *, lini : float *, x0fim : float *, x0fim : float *, doffim : float *, lfim : float *) : void
+ setspline(: bool) : void
# timerEvent(: QTimerEvent *) : void
+ ~glwid()


LuxoWindow
- animacaogerada : bool
- cbalg : QComboBox *
- cbfobj : QComboBox *
- compini : float *
- doffim : float *
- dofini : float *
- duracao : float
- iniciaranim : bool
- leduracao : QLineEdit *
- leiter : QLineEdit *
- lelig : QLineEdit *
- lepx1 : QLineEdit *
- lepx2 : QLineEdit *
- lepy1 : QLineEdit *
- lepy2 : QLineEdit *
- letempo : QLineEdit *
- luxodof : int
- massa : float *
- niter : int
- raio : float *
- sl : QSlider *
- spline : vector
- tbw : QTabWidget *
- tempoanim : int
- x0fim : float *
- x0ini : float *
- y0fim : float *
- y0ini : float *
+ LuxoWindow(parent : QWidget *, name : const char *, f : WFlags)
# timerEvent(: QTimerEvent *) : void
+ ~LuxoWindow()


STMatrix
- h : double
- num : int
- x : const adoublev &
+ Derivative1(a1 : adouble, a2 : adouble, h : double) : adouble
+ Derivative2(a1 : adouble, a2 : adouble, a3 : adouble, h : double) : adouble
+ STMatrix(y : const adoublev &, n : int, harg : double)
+ k1(i : int) : adouble
+ k2(i : int) : adouble
+ th1(i : int) : adouble
+ th2(i : int) : adouble
+ th3(i : int) : adouble
+ x0(i : int) : adouble
+ y0(i : int) : adouble
+ ~STMatrix()


CLoad3DS
- m_CurrentChunk : tChunk *
- m_FilePointer : FILE *
- m_TempChunk : tChunk *
+ CLoad3DS()
# CleanUp() : void
# GetString(: char *) : int
+ Import3DS(pModel : t3DModel *, strFileName : char *) : bool
# ProcessNextChunk(pModel : t3DModel *, : tChunk *) : void
# ProcessNextObjectChunk(pModel : t3DModel *, pObject : t3DObject *, : tChunk *) : void
# ReadChunk(: tChunk *) : void
# ReadVertexIndices(pObject : t3DObject *, : tChunk *) : void
# ReadVertices(pObject : t3DObject *, : tChunk *) : void


sos
- algo : int
- argc : int
- argv : char * *
- num : int
Tcl_AppInit(interp : Tcl_Interp *) : int
+ initopt() : void
+ sos(grav : float, n : int, a : int, agc : int, agv : char * *)

Anexo G

Código de implementação do Sistema

```
#ifndef _3DS_H
#define _3DS_H

#include "3dsothers.h"

//>------ Primary Chunk, at the beginning of each file
#define PRIMARY       0x4D4D

//>------ Main Chunks
#define OBJECTINFO    0x3D3D                // This gives the version
of the mesh and is found right before the material and object
information
#define VERSION       0x0002                // This gives the version
of the .3ds file
#define EDITKEYFRAME  0xB000                // This is the header for
all of the key frame info

//>------ sub defines of OBJECTINFO
#define MATERIAL      0xAFFF                // This stored the texture
info
#define OBJECT        0x4000                // This stores the faces,
vertices, etc...

//>------ sub defines of MATERIAL
#define MATNAME       0xA000                // This holds the material
name
#define MATDIFFUSE    0xA020                // This holds the color of
the object/material
#define MATMAP        0xA200                // This is a header for a
new material
#define MATMAPFILE    0xA300                // This holds the file
name of the texture

#define OBJECT_MESH   0x4100                // This lets us know that
we are reading a new object
```

```
//>------ sub defines of OBJECT_MESH
#define OBJECT_VERTICES     0x4110          // The objects vertices
#define OBJECT_FACES        0x4120          // The objects faces
#define OBJECT_MATERIAL     0x4130          // This is found if the
object has a material, either texture map or color
#define OBJECT_UV           0x4140          // The UV texture
coordinates


// Here is our structure for our 3DS indicies (since .3DS stores 4
unsigned shorts)
struct tIndices {

    unsigned short a, b, c, bVisible;       // This will hold point1,
2, and 3 index's into the vertex array plus a visible flag
};

// This holds the chunk info
struct tChunk
{
    unsigned short int ID;                  // The chunk's ID
    unsigned int length;                    // The length of the chunk
    unsigned int bytesRead;                 // The amount of bytes
read within that chunk
};

// This class handles all of the loading code
class CLoad3DS
{
public:
    CLoad3DS();                             // This inits the data
members

    // This is the function that you call to load the 3DS
    bool Import3DS(t3DModel *pModel, char *strFileName);

private:
    // This reads in a string and saves it in the char array passed in
```

```
    int GetString(char *);

    // This reads the next chunk
    void ReadChunk(tChunk *);

    // This reads the next large chunk
    void ProcessNextChunk(t3DModel *pModel, tChunk *);

    // This reads the object chunks
    void ProcessNextObjectChunk(t3DModel *pModel, t3DObject *pObject,
tChunk *);

    // This reads the objects vertices
    void ReadVertices(t3DObject *pObject, tChunk *);

    // This reads the objects face information
    void ReadVertexIndices(t3DObject *pObject, tChunk *);

    // This frees memory and closes the file
    void CleanUp();

    // The file pointer
    FILE *m_FilePointer;

    // These are used through the loading process to hold the chunk
information
    tChunk *m_CurrentChunk;
    tChunk *m_TempChunk;
};


#endif

#ifndef _3dsothers_h_
#define _3dsothers_h_

#include <vector>
#include <iostream>
```

```
using std::vector;
using std::cout;
using std::endl;

typedef unsigned char BYTE;

 // This is our 3D point class.  This will be used to store the
vertices of our model.
class CVector3
{
public:
    float x, y, z;
};

// This is our 2D point class.  This will be used to store the UV
coordinates.
class CVector2
{
public:
    float x, y;
};

// This is our face structure.  This is is used for indexing into the
vertex
// and texture coordinate arrays.  From this information we know which
vertices
// from our vertex array go to which face, along with the correct
texture coordinates.
struct tFace
{
    int vertIndex[3];           // indicies for the verts that make up
this triangle
    int coordIndex[3];          // indicies for the tex coords to
texture this face
};

// This holds the information for a material.  It may be a texture map
of a color.
```

```
// Some of these are not used, but I left them because you will want
to eventually
// read in the UV tile ratio and the UV tile offset for some models.
struct tMaterialInfo
{
    char  strName[255];         // The texture name
    char  strFile[255];         // The texture file name (If this is
set it's a texture map)
    BYTE  color[3];             // The color of the object (R, G, B)
    int   texureId;             // the texture ID
    float uTile;                // u tiling of texture  (Currently not
used)
    float vTile;                // v tiling of texture  (Currently not
used)
    float uOffset;              // u offset of texture  (Currently not
used)
    float vOffset;              // v offset of texture  (Currently not
used)
} ;

// This holds all the information for our model/scene.
// You should eventually turn into a robust class that
// has loading/drawing/querying functions like:
// LoadModel(...); DrawObject(...); DrawModel(...); DestroyModel(...);
struct t3DObject
{
    int  numOfVerts;            // The number of verts in the model
    int  numOfFaces;            // The number of faces in the model
    int  numTexVertex;          // The number of texture coordinates
    int  materialID;            // The texture ID to use, which is the
index into our texture array
    bool bHasTexture;           // This is TRUE if there is a texture
map for this object
    char strName[255];          // The name of the object
    CVector3  *pVerts;          // The object's vertices
    CVector3  *pNormals;        // The object's normals
    CVector2  *pTexVerts;       // The texture's UV coordinates
    tFace *pFaces;              // The faces information of the object
};
```

```
// This holds our model information.  This should also turn into a
robust class.
// We use STL's (Standard Template Library) vector class to ease our
link list burdens. :)
class t3DModel
{

    public:
        t3DModel::t3DModel(){
        numOfObjects = 0;
        numOfMaterials = 0;
    }
    int numOfObjects;                   // The number of objects in
the model
    int numOfMaterials;                 // The number of materials for
the model
    vector<tMaterialInfo> pMaterials;   // The list of material
information (Textures and colors)
    vector<t3DObject> pObject;          // The object list for our
model
};

#endif
#ifndef DESENHA_H
#define DESENHA_H

#include <math.h>
#include <GL/gl.h>
#include <GL/glu.h>

class Desenha {
public:

    Desenha() {}
    ;
    ~Desenha() {}
    ;
```

```
    static void drawBox(GLfloat, GLfloat, GLfloat, GLfloat, GLfloat,
GLfloat);
    static void drawQuad(GLfloat, GLfloat, GLfloat, GLfloat);
    static void drawGrid(int, GLfloat, int, int);
    static void gluClosedCylinder(GLUquadric*, GLdouble, GLdouble,
GLdouble, GLint, GLint);
};

#endif

#ifndef LUXO_H
#define LUXO_H

#include <math.h>
#include <GL/gl.h>
#include <GL/glu.h>

#include "Desenha.h"
#include "3ds.h"

class Luxo {
public:
    Luxo(float bf,int n,float* x0,float* y0, float* dof, float* l);
    ~Luxo();

    void setLuxo(int n,float* x,float* y, float* dof, float* l);
    void setNL(int n, float* l);
    void setDof(float* x,float* y,float* dof);
    void drawLuxo();
    void drawstick();
    void divide();
    float returnx0 () const {
                    return *x0;
                };
    float returny0 () const {
                    return *y0;
                };
    float * returndof () const {
                      return dof;
```

```
                };
    float * returnl () const {
                    return l;
                }
    private:
    int n;          //numero de graus de liberdade
    float* x0;
    float* y0;
    float* dof;   //vetor dos graus de liberdade
    float* l;     //vetor dos comprimentos dos membros
    float bf;

    // 3DS
    CLoad3DS Loadbase;
    t3DModel *Modelbase;
    CLoad3DS Loadl1;
    t3DModel *Modell1;
    CLoad3DS Loadl2;
    t3DModel *Modell2;
    CLoad3DS Loadl3;
    t3DModel *Modell3;

    void draw3ds (t3DModel *Model3D);



};

#endif

#ifndef BEZIER_H
#define BEZIER_H
#include "points2d.h"
#define JMAX 40

class bezier
{
    public:
        bezier (points2d p1, points2d p2,points2d p3, points2d p4);
```

```
        points2d returnpoint(float u);
        float arclenght (float a, float b);
        float returnu (float s);
        float rtbis(float x1, float x2, float xacc,float s);
    private:
        float raiztquad (float u);
        float rtbis(float x1, float x2, float xacc);
        float froot (float u,float s);
        points2d p1,p2,p3,p4;
};

#endif
#ifndef BSPLINES_H_
#define BSPLINES_H_
#include <math.h>
#include <vector>
#include "points2d.h"

using std::vector;
/*!
*    Classe para manipular Open Uniform B-Splines
*/
class bsplines {
public:
    /*!  Construtor
    *    d d-1 grau do polinomio
    *    n n+1 pontos de controles
    *    p vetor dos pontos de controle
    *    os nós são parametrizados 0<nós<1
    */
    bsplines (int p, int n, vector<points2d>& P);
    float Cx(float u);
    float Cy(float u);
    points2d C(float u);
    float showumax ();

private:
    int p;//Grau é p
    int n;//pontos de controles de 0 a n. Numero de pontos é n+1
```

```
    vector<float> u;
    vector<points2d> P;
    vector <float> Np;
    float umax;
    float N(int i, int p,float u);
    void coef (float u);

};
#endif
#ifndef EXCEPTIONCLASSES_H_
#define EXCEPTIONCLASSES_H_

#include <stdexcept>
using std::exception;
class differentsize : public exception {
public:
    differentsize (int,int) {}
    ;
private:
};

#endif
#if !defined( GLWID_H )
#define GLWID_H
#include <qgl.h>
#include "Luxo.h"

#include "bsplines.h"
#include "bezier.h"
#include "exceptionclasses.h"
#include <stdlib.h>

class glwid : public QGLWidget {
    Q_OBJECT
public:
    glwid( QWidget *parent, const char *name );
    void setluxoinifim(int n,float *x0ini,float *y0ini,float *dofini,
float *lini, float *x0fim,float *x0fim,float *doffim, float *lfim);
    void geraranim(int numinter,int dof);
```

```
    void gerafilm();
    vector <points2d> geraspline(float p1x, float p1y, float p2x,
float p2y);
    void gerainterpolacao();
    void setinter(int);
    void setspline (bool);
    void refresh();
    ~glwid ();

protected:
    void initializeGL();
    void resizeGL( int w, int h );
    void paintGL();
    virtual void mouseMoveEvent ( QMouseEvent * );
    virtual void mousePressEvent ( QMouseEvent * );
    virtual void mouseReleaseEvent ( QMouseEvent * );
    virtual void timerEvent( QTimerEvent * );

private:
    // Matrix armazenadas
    enum Views { CurrentView, AxisView };
    typedef struct viewMatrix {
        GLfloat model[4][4];      // OpenGL model view matrix for the
view
        GLfloat projection[4][4]; // OpenGL projection matrix for the
view
    } viewMatrix;
    viewMatrix  views[2];

    // Alguns objetos
    GLuint makeBox();
    GLuint makeaxis();
    GLuint box,axisList;
    void drawAxis();

    // Luxo
    Luxo *lxini,*lxfim;
    bool luxoset;
    int numinter;
```

```
    int numintersplines;
    Luxo **interlx;
    int currentluxo;
    int dof;

    // Splines
    vector <points2d> spline;
    vector <points2d> splinereal;
    vector <points2d> p;
    int nump;
    bool dospline;

    // Rotation Trackball
    float   m_curquat[4];
    float   m_lastquat[4];
    int     m_spin , m_move;
    int     m_begin_x,m_begin_y;
    int     m_W , m_H ;


public slots:
    void zoom( int );
    void setcurrent(int);
};
#endif // !defined( GLWID_H )
#ifndef LUXOWINDOW_H
#define LUXOWINDOW_H

#include <qmainwindow.h>
#include <qtabwidget.h>
#include <qlineedit.h>
#include <qslider.h>
#include <qcombobox.h>
#include <qcheckbox.h>

#include "glwid.h"

class LuxoWindow : public QMainWindow {
    Q_OBJECT
```

```
public:
    LuxoWindow ( QWidget* parent = 0, const char* name = 0, WFlags f =
WType_TopLevel );
    ~LuxoWindow ();
private:
    glwid * glw;
    QTabWidget *tbw;
    QLineEdit *lelig;
    QLineEdit *letempo;
    QLineEdit *leduracao;
    QLineEdit *leiter;
    QLineEdit *lepx1;
    QLineEdit *lepy1;
    QLineEdit *lepx2;
    QLineEdit *lepy2;
    QComboBox *cbalg;
    QComboBox *cbfobj;
    QCheckBox *cbcm;
    int luxodof;
    int tempoanim;
    int niter;
    float *x0ini,*y0ini;
    float *x0fim,*y0fim;
    float *dofini,*compini;
    float *doffim;
    float *massa;
    float *raio;
    float duracao;
    QSlider *sl;
    bool animacaogerada;
    bool iniciaranim;
    vector <points2d> spline;
public slots:
    void mudaconf();
    void mudapini();
    void mudapfim();
    void centrodecontrole();
    void gerafilm();
    void animar();
```

```
    void animartras();
    void animarfrente();
    void gerasplines();
    void gerarinterpolacao();
    void checkobj (int);
protected:
    virtual void timerEvent( QTimerEvent * );
};
#endif
#ifndef MATHCONN_H
#define MATHCONN_H
#include <mathlink.h>

class mathconn {
public:
    mathconn ();
    ~mathconn ();
    void sendmsg (char *str);
    void sendmsgnum(char *str);
    const char *receivemsg ();
    void receiveprint (char *str);
    void error ();
private:
    MLINK lp;
    MLEnvironment env;

};
#endif
#ifndef OPTIMFILE_H
#define OPTIMFILE_H
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include "points2d.h"

using std::vector;
using std::string;
using std::ofstream;
```

```
using std::ios_base;
using std::cout;
using std::endl;

class optimfile {
public:
    optimfile (int d,float xt1,float yt1,float xt2,float yt2,float
*vect1,float *vect2,float *len,float *massa,float *raio,float dt,int
obj,vector <points2d> spline,int niter,const char *c);
    ~optimfile ();
    void include ();
    void STConstraints();
    void setup();
    void update(vector <string>&s);
    void put (const char *c);
    void close ();

private:
    ofstream *f;
    int niter;
    int dof;
    float x1,x2;
    float y1,y2;
    float *vec1;
    float *vec2;
    float *l;
    float *massa;
    float *raio;
    float dt;
    int obj;
    vector <points2d> spline;
};

#endif
#ifndef POINTS2D_H_
#define POINTS2D_H_

class points2d {
public:
```

```
    points2d ();
    points2d (float x, float y);
    float returnx ();
    float returny ();
    void set
        (float xl, float yl);
private:
    float x;
    float y;
};

#endif
#ifndef POVARQ_H_
#define POVARQ_H_

#include <fstream>

using std::ofstream;
using std::ios_base;
using std::endl;
//! \brief Classe para gerar arquivos povray basado na figura
articulada
class povarq {
private:
    ofstream *f;
    int dof;
    float x0;
    float y0;
    float *dofang;
    float *l;
public:
    /*! Construtor
        *name ponteiro com o nome do arquivo povray
        Incluir a extensão .pov (POVRAY)
        dof número de graus de liberdade
    */

    povarq (const char *name,int dof,float x0,float y0,float *dof,
float *l);
```

```
    ~povarq ();
    void putplane (float r1, float g1, float b1,float r2, float g2,
float b2);
    void putlight (float x, float y, float z,float r, float g, float
b);
    void putcamera (float x, float y, float z, float lax, float lay,
float laz);
    void putbase (float r, float g, float b);
    void putlink (int n,float r, float g, float b);
};

#endif
#ifndef QDIALOGDESC_H
#define QDIALOGDESC_H

#include <qdialog.h>
#include <qlineedit.h>
class qdialogdesc : public QDialog {
    Q_OBJECT
    public:
        qdialogdesc ( int dof,float *massa,float *raio,float
*comp,QWidget* parent = 0, const char* name = 0, bool modal = FALSE,
WFlags fl = 0);
    private:
        QLineEdit **leditcomp;
        QLineEdit **leditmassa;
        QLineEdit **leditraio;
        int dof;
        float *massa;
        float *raio;
        float *comp;
    public slots:
        void accept ();
};

#endif
#ifndef QDIALOGPOSFIM_H
#define QDIALOGPOSFIM_H
#include <qdialog.h>
```

```
#include <qlineedit.h>

class qdialogposfim : public QDialog {
    Q_OBJECT
public:
    qdialogposfim ( int num,float *x0,float *y0,float *doffim,QWidget*
parent = 0, const char* name = 0, bool modal = FALSE, WFlags fl = 0);
private:
    int num;
    float *x0;
    float *y0;
    float *doffim;
    QLineEdit *le1;
    QLineEdit *le2;
    QLineEdit **leditdofp;
public slots:
    void accept ();
};

#endif
#ifndef QDIALOGPOSINI_H
#define QDIALOGPOSINI_H
#include <qdialog.h>
#include <qlineedit.h>

class qdialogposini : public QDialog {
    Q_OBJECT
    public:
        qdialogposini ( int num,float *x0,float *y0,float
*doffim,QWidget* parent = 0, const char* name = 0, bool modal = FALSE,
WFlags fl = 0);
    private:
        int num;
        float *x0;
        float *y0;
        float *dof;
        QLineEdit *le1;
        QLineEdit *le2;
        QLineEdit **leditdofp;
```

```
    public slots:
        void accept ();
};

#endif
#ifndef QUICKARQ_H_
#define QUICKARQ_H_

#include <quicktime.h>
#include <colormodels.h>

//! \brief Classe para gerar arquivos MOV (Quicktime)
class quickarq {
    //! \class quickarq quickarq.h "libutils/quickarq.h"

public:
    /*!
        *name ponteiro para o arquivo de saída
        incluir o a extensão .mov
    */
    quickarq (const char *name, int wi, int hi);
    //! destrutor
    ~quickarq ();
    //! Adiciona o Frame
    void addframe (unsigned char **row_pointers);
private:
    quicktime_t *output;
    int w,h;
};

#endif
#ifndef SOS_H
#define SOS_H

#include <tcl.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <fstream>
```

```
#include <iostream>
#include <stdlib.h>
#include <time.h>


extern "C" int Hqp_Init _ANSI_ARGS_((Tcl_Interp *interp));
extern "C" int Omu_Init _ANSI_ARGS_((Tcl_Interp *interp));
extern "C" int Odc_Init _ANSI_ARGS_((Tcl_Interp *interp));

class sos {
private:
    static int num;
    static int algo;
    int argc;
    char **argv;
    friend int Tcl_AppInit(Tcl_Interp *interp);
public:
    sos (float grav,int n,int a,int agc, char **agv);
    void initopt ();
};

#endif

#ifndef STCONSTRAINTS_H
#define STCONSTRAINTS_H

#include <Omu_Program.h>

class STConstraints: public Omu_Program {
protected:
    double m;
    double grav;
    int num;
    int dof;
    double ti;
    double tf;
    double h;
    double ax;
    double ay;
```

```
    double bx;
    double by;
    double ath1;
    double bth1;
    double ath2;
    double bth2;
    double ath3;
    double bth3;
    double l1;
    double l2;
    double l3;
    double m1;
    double m2;
    double m3;
    double I1;
    double I2;
    double I3;
    double r1;
    double r2;
    double r3;
    inline void setup(int k,
                      Omu_Vector &x, Omu_Vector &u, Omu_Vector &c);
    inline void update(int kk,
                       const adoublev &x, const adoublev &u,
                       adoublev &f, adouble &f0, adoublev &c);
public:
    char *name() {
        return "STConstraints2D";
    }

    STConstraints ();
    ~STConstraints () {}
    ;


};
#endif

#ifndef STMATRIX_H
```

```
#define STMATRIX_H
#include <adouble.h>
class STMatrix {
private:
    const adoublev &x;
    int num;
    double h;
public:
    STMatrix (const adoublev &y,int n,double harg):x(y),num(n),h(harg)
{}
    ;
    ~STMatrix() {}
    ;
    adouble x0 (int i);
    adouble y0 (int i);
    adouble th1 (int i);
    adouble th2 (int i);
    adouble th3 (int i);
    adouble k1 (int i);
    adouble k2 (int i);
    friend adouble Derivative1 (adouble a1,adouble a2,double h);
    friend adouble Derivative2 (adouble a1,adouble a2,adouble a3,
double h);
};
#endif
#ifndef STRINGMAN_H
#define STRINGMAN_H
#include <string>
using std::string;

class stringman {
private:
    string s;
public:
    stringman (const char* c);
    ~stringman ();
    void manipulate ();
    char * show();
};
```

```
#endif
#ifndef STSYMBOL_H_
#define STSYMBOL_H_
#include "mathconn.h"
#include <iostream>

using namespace std;

class stsymbol {
public:
    stsymbol (int n);
    ~stsymbol ();
    void start ();
    const char * receive (int n);
private:
    mathconn *ml;
};


#endif

#ifndef _3DS_H
#define _3DS_H

#include "3dsothers.h"

//>------ Primary Chunk, at the beginning of each file
#define PRIMARY       0x4D4D

//>------ Main Chunks
#define OBJECTINFO    0x3D3D                // This gives the version
of the mesh and is found right before the material and object
information
#define VERSION       0x0002                // This gives the version
of the .3ds file
#define EDITKEYFRAME  0xB000                // This is the header for
all of the key frame info

//>------ sub defines of OBJECTINFO
```

```
#define MATERIAL      0xAFFF                // This stored the texture
info
#define OBJECT        0x4000                // This stores the faces,
vertices, etc...

//>------ sub defines of MATERIAL
#define MATNAME       0xA000                // This holds the material
name
#define MATDIFFUSE    0xA020                // This holds the color of
the object/material
#define MATMAP        0xA200                // This is a header for a
new material
#define MATMAPFILE    0xA300                // This holds the file
name of the texture

#define OBJECT_MESH   0x4100                // This lets us know that
we are reading a new object

//>------ sub defines of OBJECT_MESH
#define OBJECT_VERTICES     0x4110          // The objects vertices
#define OBJECT_FACES        0x4120          // The objects faces
#define OBJECT_MATERIAL     0x4130          // This is found if the
object has a material, either texture map or color
#define OBJECT_UV           0x4140          // The UV texture
coordinates


// Here is our structure for our 3DS indicies (since .3DS stores 4
unsigned shorts)
struct tIndices {

    unsigned short a, b, c, bVisible;       // This will hold point1,
2, and 3 index's into the vertex array plus a visible flag
};

// This holds the chunk info
struct tChunk
{
    unsigned short int ID;                  // The chunk's ID
```

```
    unsigned int length;                    // The length of the chunk
    unsigned int bytesRead;                 // The amount of bytes
read within that chunk
};

// This class handles all of the loading code
class CLoad3DS
{
public:
    CLoad3DS();                             // This inits the data
members

    // This is the function that you call to load the 3DS
    bool Import3DS(t3DModel *pModel, char *strFileName);

private:
    // This reads in a string and saves it in the char array passed in
    int GetString(char *);

    // This reads the next chunk
    void ReadChunk(tChunk *);

    // This reads the next large chunk
    void ProcessNextChunk(t3DModel *pModel, tChunk *);

    // This reads the object chunks
    void ProcessNextObjectChunk(t3DModel *pModel, t3DObject *pObject,
tChunk *);

    // This reads the objects vertices
    void ReadVertices(t3DObject *pObject, tChunk *);

    // This reads the objects face information
    void ReadVertexIndices(t3DObject *pObject, tChunk *);

    // This frees memory and closes the file
    void CleanUp();

    // The file pointer
```

```
    FILE *m_FilePointer;

    // These are used through the loading process to hold the chunk
information
    tChunk *m_CurrentChunk;
    tChunk *m_TempChunk;
};


#endif
#ifndef _3dsothers_h_
#define _3dsothers_h_

#include <vector>
#include <iostream>

using std::vector;
using std::cout;
using std::endl;

typedef unsigned char BYTE;

 // This is our 3D point class.  This will be used to store the
vertices of our model.
class CVector3
{
public:
    float x, y, z;
};

// This is our 2D point class.  This will be used to store the UV
coordinates.
class CVector2
{
public:
    float x, y;
};
```

```
// This is our face structure.  This is is used for indexing into the
vertex
// and texture coordinate arrays.  From this information we know which
vertices
// from our vertex array go to which face, along with the correct
texture coordinates.
struct tFace
{
    int vertIndex[3];           // indicies for the verts that make up
this triangle
    int coordIndex[3];          // indicies for the tex coords to
texture this face
};

// This holds the information for a material.  It may be a texture map
of a color.
// Some of these are not used, but I left them because you will want
to eventually
// read in the UV tile ratio and the UV tile offset for some models.
struct tMaterialInfo
{
    char  strName[255];         // The texture name
    char  strFile[255];         // The texture file name (If this is
set it's a texture map)
    BYTE  color[3];             // The color of the object (R, G, B)
    int   texureId;             // the texture ID
    float uTile;                // u tiling of texture  (Currently not
used)
    float vTile;                // v tiling of texture  (Currently not
used)
    float uOffset;              // u offset of texture  (Currently not
used)
    float vOffset;              // v offset of texture  (Currently not
used)
} ;

// This holds all the information for our model/scene.
// You should eventually turn into a robust class that
// has loading/drawing/querying functions like:
```

```
// LoadModel(...); DrawObject(...); DrawModel(...); DestroyModel(...);
struct t3DObject
{
    int  numOfVerts;            // The number of verts in the model
    int  numOfFaces;            // The number of faces in the model
    int  numTexVertex;          // The number of texture coordinates
    int  materialID;            // The texture ID to use, which is the
index into our texture array
    bool bHasTexture;           // This is TRUE if there is a texture
map for this object
    char strName[255];          // The name of the object
    CVector3  *pVerts;          // The object's vertices
    CVector3  *pNormals;        // The object's normals
    CVector2  *pTexVerts;       // The texture's UV coordinates
    tFace *pFaces;              // The faces information of the object
};

// This holds our model information.  This should also turn into a
robust class.
// We use STL's (Standard Template Library) vector class to ease our
link list burdens. :)
class t3DModel
{

    public:
        t3DModel::t3DModel(){
        numOfObjects = 0;
        numOfMaterials = 0;
    }
    int numOfObjects;                   // The number of objects in
the model
    int numOfMaterials;                 // The number of materials for
the model
    vector<tMaterialInfo> pMaterials;   // The list of material
information (Textures and colors)
    vector<t3DObject> pObject;          // The object list for our
model
};
```

```
#endif
#ifndef DESENHA_H
#define DESENHA_H

#include <math.h>
#include <GL/gl.h>
#include <GL/glu.h>

class Desenha {
public:

    Desenha() {}
    ;
    ~Desenha() {}
    ;
    static void drawBox(GLfloat, GLfloat, GLfloat, GLfloat, GLfloat,
GLfloat);
    static void drawQuad(GLfloat, GLfloat, GLfloat, GLfloat);
    static void drawGrid(int, GLfloat, int, int);
    static void gluClosedCylinder(GLUquadric*, GLdouble, GLdouble,
GLdouble, GLint, GLint);
};

#endif

#ifndef LUXO_H
#define LUXO_H

#include <math.h>
#include <GL/gl.h>
#include <GL/glu.h>

#include "Desenha.h"
#include "3ds.h"

class Luxo {
public:
    Luxo(float bf,int n,float* x0,float* y0, float* dof, float* l);
    ~Luxo();
```

```
void setLuxo(int n,float* x,float* y, float* dof, float* l);
void setNL(int n, float* l);
void setDof(float* x,float* y,float* dof);
void drawLuxo();
void drawstick();
void divide();
float returnx0 () const {
                return *x0;
            };
float returny0 () const {
                return *y0;
            };
float * returndof () const {
                  return dof;
              };
float * returnl () const {
                  return l;
              }
private:
int n;          //numero de graus de liberdade
float* x0;
float* y0;
float* dof;   //vetor dos graus de liberdade
float* l;     //vetor dos comprimentos dos membros
float bf;

// 3DS
CLoad3DS Loadbase;
t3DModel *Modelbase;
CLoad3DS Loadl1;
t3DModel *Modell1;
CLoad3DS Loadl2;
t3DModel *Modell2;
CLoad3DS Loadl3;
t3DModel *Modell3;

void draw3ds (t3DModel *Model3D);
```

```
};

#endif


#ifndef BEZIER_H
#define BEZIER_H
#include "points2d.h"
#define JMAX 40

class bezier
{
    public:
        bezier (points2d p1, points2d p2,points2d p3, points2d p4);
        points2d returnpoint(float u);
        float arclenght (float a, float b);
        float returnu (float s);
        float rtbis(float x1, float x2, float xacc,float s);
    private:
        float raiztquad (float u);
        float rtbis(float x1, float x2, float xacc);
        float froot (float u,float s);
        points2d p1,p2,p3,p4;
};

#endif

#ifndef BSPLINES_H_
#define BSPLINES_H_
#include <math.h>
#include <vector>
#include "points2d.h"

using std::vector;
/*!
*    Classe para manipular Open Uniform B-Splines
*/
```

```
class bsplines {
public:
    /*!  Construtor
    *    d d-1 grau do polinomio
    *    n n+1 pontos de controles
    *    p vetor dos pontos de controle
    *    os nós são parametrizados 0<nós<1
    */
    bsplines (int p, int n, vector<points2d>& P);
    float Cx(float u);
    float Cy(float u);
    points2d C(float u);
    float showumax ();

private:
    int p;//Grau é p
    int n;//pontos de controles de 0 a n. Numero de pontos é n+1
    vector<float> u;
    vector<points2d> P;
    vector <float> Np;
    float umax;
    float N(int i, int p,float u);
    void coef (float u);

};
#endif
#ifndef EXCEPTIONCLASSES_H_
#define EXCEPTIONCLASSES_H_

#include <stdexcept>
using std::exception;
class differentsize : public exception {
public:
    differentsize (int,int) {}
    ;
private:
};

#endif
```

```
#if !defined( GLWID_H )
#define GLWID_H
#include <qgl.h>
#include "Luxo.h"

#include "bsplines.h"
#include "bezier.h"
#include "exceptionclasses.h"
#include <stdlib.h>

class glwid : public QGLWidget {
    Q_OBJECT
public:
    glwid( QWidget *parent, const char *name );
    void setluxoinifim(int n,float *x0ini,float *y0ini,float *dofini,
float *lini, float *x0fim,float *x0fim,float *doffim, float *lfim);
    void geraranim(int numinter,int dof);
    void gerafilm();
    vector <points2d> geraspline(float p1x, float p1y, float p2x,
float p2y);
    void gerainterpolacao();
    void setinter(int);
    void setspline (bool);
    void refresh();
    ~glwid ();

protected:
    void initializeGL();
    void resizeGL( int w, int h );
    void paintGL();
    virtual void mouseMoveEvent ( QMouseEvent * );
    virtual void mousePressEvent ( QMouseEvent * );
    virtual void mouseReleaseEvent ( QMouseEvent * );
    virtual void timerEvent( QTimerEvent * );

private:
    // Matrix armazenadas
    enum Views { CurrentView, AxisView };
    typedef struct viewMatrix {
```

```
        GLfloat model[4][4];      // OpenGL model view matrix for the
view
        GLfloat projection[4][4]; // OpenGL projection matrix for the
view
    } viewMatrix;
    viewMatrix  views[2];

    // Alguns objetos
    GLuint makeBox();
    GLuint makeaxis();
    GLuint box,axisList;
    void drawAxis();

    // Luxo
    Luxo *lxini,*lxfim;
    bool luxoset;
    int numinter;
    int numintersplines;
    Luxo **interlx;
    int currentluxo;
    int dof;

    // Splines
    vector <points2d> spline;
    vector <points2d> splinereal;
    vector <points2d> p;
    int nump;
    bool dospline;

    // Rotation Trackball
    float   m_curquat[4];
    float   m_lastquat[4];
    int     m_spin , m_move;
    int     m_begin_x,m_begin_y;
    int     m_W , m_H ;


public slots:
    void zoom( int );
```

```
    void setcurrent(int);
};
#endif // !defined( GLWID_H )
#ifndef LUXOWINDOW_H
#define LUXOWINDOW_H

#include <qmainwindow.h>
#include <qtabwidget.h>
#include <qlineedit.h>
#include <qslider.h>
#include <qcombobox.h>
#include <qcheckbox.h>

#include "glwid.h"

class LuxoWindow : public QMainWindow {
    Q_OBJECT
public:
    LuxoWindow ( QWidget* parent = 0, const char* name = 0, WFlags f =
WType_TopLevel );
    ~LuxoWindow ();
private:
    glwid * glw;
    QTabWidget *tbw;
    QLineEdit *lelig;
    QLineEdit *letempo;
    QLineEdit *leduracao;
    QLineEdit *leiter;
    QLineEdit *lepx1;
    QLineEdit *lepy1;
    QLineEdit *lepx2;
    QLineEdit *lepy2;
    QComboBox *cbalg;
    QComboBox *cbfobj;
    QCheckBox *cbcm;
    int luxodof;
    int tempoanim;
    int niter;
    float *x0ini,*y0ini;
```

```
    float *x0fim,*y0fim;
    float *dofini,*compini;
    float *doffim;
    float *massa;
    float *raio;
    float duracao;
    QSlider *sl;
    bool animacaogerada;
    bool iniciaranim;
    vector <points2d> spline;
public slots:
    void mudaconf();
    void mudapini();
    void mudapfim();
    void centrodecontrole();
    void gerafilm();
    void animar();
    void animartras();
    void animarfrente();
    void gerasplines();
    void gerarinterpolacao();
    void checkobj (int);
protected:
    virtual void timerEvent( QTimerEvent * );
};
#endif
#ifndef MATHCONN_H
#define MATHCONN_H
#include <mathlink.h>

class mathconn {
public:
    mathconn ();
    ~mathconn ();
    void sendmsg (char *str);
    void sendmsgnum(char *str);
    const char *receivemsg ();
    void receiveprint (char *str);
    void error ();
```

```
private:
    MLINK lp;
    MLEnvironment env;

};
#endif

#ifndef OPTIMFILE_H
#define OPTIMFILE_H
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include "points2d.h"

using std::vector;
using std::string;
using std::ofstream;
using std::ios_base;
using std::cout;
using std::endl;

class optimfile {
public:
    optimfile (int d,float xt1,float yt1,float xt2,float yt2,float
*vect1,float *vect2,float *len,float *massa,float *raio,float dt,int
obj,vector <points2d> spline,int niter,const char *c);
    ~optimfile ();
    void include ();
    void STConstraints();
    void setup();
    void update(vector <string>&s);
    void put (const char *c);
    void close ();

private:
    ofstream *f;
    int niter;
    int dof;
```

```
    float x1,x2;
    float y1,y2;
    float *vec1;
    float *vec2;
    float *l;
    float *massa;
    float *raio;
    float dt;
    int obj;
    vector <points2d> spline;
};

#endif
#ifndef POINTS2D_H_
#define POINTS2D_H_

class points2d {
public:
    points2d ();
    points2d (float x, float y);
    float returnx ();
    float returny ();
    void set
        (float xl, float yl);
private:
    float x;
    float y;
};

#endif
#ifndef POVARQ_H_
#define POVARQ_H_

#include <fstream>

using std::ofstream;
using std::ios_base;
using std::endl;
```

```
//! \brief Classe para gerar arquivos povray basado na figura
articulada
class povarq {
private:
    ofstream *f;
    int dof;
    float x0;
    float y0;
    float *dofang;
    float *l;
public:
    /*! Construtor
        *name ponteiro com o nome do arquivo povray
        Incluir a extensão .pov (POVRAY)
        dof número de graus de liberdade
    */

    povarq (const char *name,int dof,float x0,float y0,float *dof,
float *l);
    ~povarq ();
    void putplane (float r1, float g1, float b1,float r2, float g2,
float b2);
    void putlight (float x, float y, float z,float r, float g, float
b);
    void putcamera (float x, float y, float z, float lax, float lay,
float laz);
    void putbase (float r, float g, float b);
    void putlink (int n,float r, float g, float b);
};

#endif

#ifndef QDIALOGDESC_H
#define QDIALOGDESC_H

#include <qdialog.h>
#include <qlineedit.h>
class qdialogdesc : public QDialog {
    Q_OBJECT
```

```
    public:
        qdialogdesc ( int dof,float *massa,float *raio,float
*comp,QWidget* parent = 0, const char* name = 0, bool modal = FALSE,
WFlags fl = 0);
    private:
        QLineEdit **leditcomp;
        QLineEdit **leditmassa;
        QLineEdit **leditraio;
        int dof;
        float *massa;
        float *raio;
        float *comp;
    public slots:
        void accept ();
};

#endif
#ifndef QDIALOGPOSFIM_H
#define QDIALOGPOSFIM_H
#include <qdialog.h>
#include <qlineedit.h>

class qdialogposfim : public QDialog {
    Q_OBJECT
public:
    qdialogposfim ( int num,float *x0,float *y0,float *doffim,QWidget*
parent = 0, const char* name = 0, bool modal = FALSE, WFlags fl = 0);
private:
    int num;
    float *x0;
    float *y0;
    float *doffim;
    QLineEdit *le1;
    QLineEdit *le2;
    QLineEdit **leditdofp;
public slots:
    void accept ();
};
```

```
#endif

#ifndef QDIALOGPOSINI_H
#define QDIALOGPOSINI_H
#include <qdialog.h>
#include <qlineedit.h>

class qdialogposini : public QDialog {
    Q_OBJECT
    public:
        qdialogposini ( int num,float *x0,float *y0,float
*doffim,QWidget* parent = 0, const char* name = 0, bool modal = FALSE,
WFlags fl = 0);
    private:
        int num;
        float *x0;
        float *y0;
        float *dof;
        QLineEdit *le1;
        QLineEdit *le2;
        QLineEdit **leditdofp;
    public slots:
        void accept ();
};

#endif

#ifndef QUICKARQ_H_
#define QUICKARQ_H_

#include <quicktime.h>
#include <colormodels.h>

//! \brief Classe para gerar arquivos MOV (Quicktime)
class quickarq {
    //! \class quickarq quickarq.h "libutils/quickarq.h"

public:
    /*!
```

```
        *name ponteiro para o arquivo de saída
        incluir o a extensão .mov
    */
    quickarq (const char *name, int wi, int hi);
    //! destrutor
    ~quickarq ();
    //! Adiciona o Frame
    void addframe (unsigned char **row_pointers);
private:
    quicktime_t *output;
    int w,h;
};

#endif

#ifndef SOS_H
#define SOS_H

#include <tcl.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <fstream>
#include <iostream>
#include <stdlib.h>
#include <time.h>


extern "C" int Hqp_Init _ANSI_ARGS_((Tcl_Interp *interp));
extern "C" int Omu_Init _ANSI_ARGS_((Tcl_Interp *interp));
extern "C" int Odc_Init _ANSI_ARGS_((Tcl_Interp *interp));

class sos {
private:
    static int num;
    static int algo;
    int argc;
    char **argv;
    friend int Tcl_AppInit(Tcl_Interp *interp);
```

```
public:
    sos (float grav,int n,int a,int agc, char **agv);
    void initopt ();
};

#endif

#ifndef STCONSTRAINTS_H
#define STCONSTRAINTS_H

#include <Omu_Program.h>

class STConstraints: public Omu_Program {
protected:
    double m;
    double grav;
    int num;
    int dof;
    double ti;
    double tf;
    double h;
    double ax;
    double ay;
    double bx;
    double by;
    double ath1;
    double bth1;
    double ath2;
    double bth2;
    double ath3;
    double bth3;
    double l1;
    double l2;
    double l3;
    double m1;
    double m2;
    double m3;
    double I1;
    double I2;
```

```
    double I3;
    double r1;
    double r2;
    double r3;
    inline void setup(int k,
                      Omu_Vector &x, Omu_Vector &u, Omu_Vector &c);
    inline void update(int kk,
                       const adoublev &x, const adoublev &u,
                       adoublev &f, adouble &f0, adoublev &c);
public:
    char *name() {
        return "STConstraints2D";
    }

    STConstraints ();
    ~STConstraints () {}
    ;


};
#endif

#ifndef STMATRIX_H
#define STMATRIX_H
#include <adouble.h>
class STMatrix {
private:
    const adoublev &x;
    int num;
    double h;
public:
    STMatrix (const adoublev &y,int n,double harg):x(y),num(n),h(harg)
{}
    ;
    ~STMatrix() {}
    ;
    adouble x0 (int i);
    adouble y0 (int i);
    adouble th1 (int i);
```

```
    adouble th2 (int i);
    adouble th3 (int i);
    adouble k1 (int i);
    adouble k2 (int i);
    friend adouble Derivative1 (adouble a1,adouble a2,double h);
    friend adouble Derivative2 (adouble a1,adouble a2,adouble a3,
double h);
};
#endif

#ifndef STRINGMAN_H
#define STRINGMAN_H
#include <string>
using std::string;

class stringman {
private:
    string s;
public:
    stringman (const char* c);
    ~stringman ();
    void manipulate ();
    char * show();
};
#endif

#ifndef STSYMBOL_H_
#define STSYMBOL_H_
#include "mathconn.h"
#include <iostream>

using namespace std;

class stsymbol {
public:
    stsymbol (int n);
    ~stsymbol ();
    void start ();
    const char * receive (int n);
```

```
private:
    mathconn *ml;
};


#endif
```

Bibliografia

ADOL-C (2004). *A Package for Automatic Differentiation of Algorithms Written in C/C++*. [http://www.math.tu-dresden.de/wir/project/adolc/] (Visitado em 2 de setembro de 2004)

Auslander, J., A. Fukunaga, H. Partovi, J. Christensen, L. Hsu, P. Reiss, A. Shuman, J. Marks and J. T. Ngo (1995). *Further Experience with Controller-Based Automatic Motion Synthesis for Articulated Figures.* ACM Transactions on Graphics 14(4): 311 - 336.

Allen, B., B. Curless e Z. Popović. (2003). *The space of human body shapes: reconstruction and parameterization from range scans.* ACM Transactions on Graphics 22(3): 587 - 594.

Baraff, D. (1995). *An Introduction to Physically Based Modeling: Rigid Body Simulation I - Unconstrainned Rigid Body Dynamics.* SIGGRAPH '95 course. [http://www-2.cs.cmu.edu/afs/cs/user/baraff/www/pbm/pbm.html] (Visitado em 2 de setembro de 2004)

Blanz, V. e T. Vetter (1999). *A Morphable Model for the Synthesis of 3D Faces.* Proceedings of the 26th annual conference on Computer graphics and interactive techniques: 187 - 194.

Bodenheimer, B., C. Rose, S. Rosenthal e J. Pella (1997). *The Process of Motion Capture: Dealing with the Data*. Computer Animation and Simulation. Proceedings of the Eurographics Workshop: 3 - 18.

Boulic, R. e D. Thalmann (1992). *Combined Direct and Inverse Kinematic Control for Articulated Figure Motion Editing.* Computer Graphics Forum. 11(4): 189 - 202.

Brogan, D., K. Granata e P. Sheth (2002). *Spacetime constraints for biomechanical movements.* In Proceedings of the IASTED International Conference on Applied Modeling and Simulation (AMS): 67 - 72.

Cohen, M. F. (1992). *Interactive Spacetime Control for Animation.* Proceedings of the 19th annual conference on Computer graphics and interactive techniques: 293 - 302.

Cohen-Or, D., A. Solomovic e D. Levin. (1998). *Three-dimensional distance field metamorphosis. ACM Transactions on Graphics.* 17: 116 - 141.

Chai, J.-x., J. Xiao e J. Hodgins (2003). *Vision-based control of 3D facial animation.* Symposium on Computer Animation: 193 - 206.

Chang, C., D. R. Brown, D. S. Bloswick e S. M. Hsiang (2001). *Biomechanical simulation of manual lifting using spacetime optimization.* Journal of Biomechanics. 34: 527 - 532.

Digital Biomechanics, Boston Dynamics. (2004). [http://www.bdi.com] (Visitado em 2 de setembro de 2004)

Discreet, technical specification (2004). [http://www.discreet.com] (Visitado em 2 de setembro de 2004)

Disney, Snow White And The Seven Dwarfs (1937). [http://www.disney.com] (Visitado em 2 de setembro de 2004)

Evers, T. F. (2000). *Um Estudo Comparativo de Algoritmos para Metamorfose 3D*. Tese Mestrado. Centro de Ciencias Exatas e Tecnologias, Universidade do vale do rio dos sinos.

Ezzat, T., G. Geiger e T. Poggio (2002) *Trainable Videorealistic Speech Animation.* Proceedings of ACM SIGGRAPH 2002: 389-398.

Fletcher, R. (1987). *Practical Methods of Optimization*, John Wiley and Sons.

Franke, R. (1998). *Omuses a tool for the optimization of multistage systems.* Technical report, Dept. of Automation and Systems Engineering, University of Ilmenau.

Fujimoto, Y. (1998). *Study on biped Walking Robot with Environmental Force Interaction.* Phd Thesis, Computing Engineering, Yokohama National University.

Furniss, M. (2004). *Motion Capture, MIT communications Forum.* [http://web.mit.edu/comm-forum/papers/furniss.html] (Visitado em 2 de setembro de 2004)

Girard, M. e A. A. Maciejewski (1985). *Computational modeling for the computer animation of legged figures.* Proceedings of the 12th annual conference on Computer graphics and interactive techniques: 263 - 270.

Gleicher, M. (1997). *Motion Editing with Spacetime Constraints.* Symposium on Interactive 3D Graphics: 139-148.

Gleicher, M. (1999). Animation from Observation: Motion Capture and Motion Editing. ACM SIGGRAPH Computer Graphics. 33(4): 51 - 54.

Goldsmith, J. (1994). *Optimized Computer-Generated Motions for Animation.* Technical Report, California Institute of Technology. [http://caltechcstr.library.caltech.edu/archive/00000229/] (Visitado em 2 de setembro de 2004)

Hahn, J. K. (1988). *Realistic Animation of Rigid Bodies*. Proceedings of the 15th annual conference on Computer graphics and interactive techniques: 299 - 308.

Hodgins, J., W. Wooten, D. Brogan, J. O'Brien (1995). *Animating human athletes.* Proceedings of the 22nd annual conference on Computer graphics and interactive techniques: 71 - 78.

HQP. (2004). *A solver for sparse nonlinear optimization.* [http://hqp.sourceforge.net] (Visitado em 2 de setembro de 2004)

Isaacs, P. e M. Cohen (1987). *Controlling Dynamic Simulation with Kinematic Constraints.* Proceedings of the 14th annual conference on Computer graphics and interactive techniques: 215 - 224.

Ko, H. (1994). *Kinematic and dynamic techniques for analyzing, predicting and animating human locomotion.* PhD thesis, University of Pennsylvannia.

Ko, H. and N. I. Badler (1996). *Animating Human Locomotion with Inverse Dynamics.* IEEE Computer Graphics 16(2): 50-59.

Kokkevis, E., D. Metaxas e N. Badler (1996). *User-controlled physics-based animation for articulated figures.* Proceedings of the Computer Animation: 16 - 26.

Krabbes, M. e C. Döschner (1999). *Modeling and Control of Manipulator Dynamics using Modified Self-Organizing Maps.* Fourth International Workshop Neural Networks in Applications.

Lasseter, J. (1987). *Principles of Traditional Animation Applied to 3D computer animation.* Computer Graphics. 21(4): 35 - 44.

Laszlo, J., M. Panne e E. Fiume (2000). *Interactive Control for Physically-Based Animation.* Proceedings of the 27th annual conference on Computer graphics and interactive techniques: 201 - 208.

Lazarus, F. e A. Verroust (1998). *Three-dimensional metamorphosis: a survey.* The Visual Computer 14(8 - 9): 373 - 389.

Lee, J. e S. Shin (1999). *A Hierarchical Approach to Interactive Motion Editing for Human-like Figures*. Proceedings of the 26th annual conference on Computer graphics and interactive techniques: 39 - 48.

Lee, W. e N. Magnenat-Thalmann (2001). *Virtual Body Morphing*. The Fourteenth Conference on Computer Animation: 158 - 166.

Liu, Z., S. Gortler e M. Cohen (1994). *Hierarchical spacetime control*. Proceedings of the 21st annual conference on Computer graphics and interactive techniques: 35 - 42.

Liu, Z. e M. F. Cohen (1995). *Keyframe Motion Optimization By Relaxing Speed and Timing*. Computer Animation and Simulation '95, 6th Eurographics Workshop: 144 - 153.

Maciejewski, A. A. (1990). *Dealing with ill-Conditioned Equations of Motion for Articulated Figures*. Computer Graphics and Applications. 10 (3): 63 - 71.

Madhavapeddy, N. (2004). *Graphics Research*, The Queen's University of Belfast. [http://3d.recoil.org] (Visitado em 2 de setembro de 2004).

Mellor, D. (2004). *Motion Capture*. [http://www.spgv.com/columns/motioncapture.html] (Visitado em 2 de setembro de 2004)

Meta Motion. (2004). *Gypsy Motion Capture System*. [http://www.metamotion.com] (Visitado em 2 de setembro de 2004)

Multon, F., L. France, M. Cani and G. Debunne (1999). *Computer Animation of Human Walking: a Survey*. The Journal of Visualization and Computer Animation. 10 (1): 39-54.

MusculoGraphics, Inc. Motion Analysis Corporation. (2002). [http://www.musculographics.com] (Visitado em 2 de setembro de 2004)

Nagano, A., K. Gerritsen e S. Fukashiro (2000). *A sensitivity analysis of the calculation of mechanical output through inverse dynamics: a computer simulation study*. Journal of Biomechanics 33: 1313-1318.

Ngo, J. e J. Marks (1993). *Spacetime Constraints Revisited*. Proceedings of the 20th annual conference on Computer graphics and interactive techniques: 343 - 350.

Parent, R. (2001). *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann.

Park, J. e D. Fussell (1997). *Forward dynamics based realistic animation of rigid bodies*. Computers and Graphics. 21 (4): 483 - 496.

Phillips, C. e N. Badler (1988). *Jack: A Toolkit for Manipulating Articulated Figures*. Symposium on User Interface Software and Technology: 221 - 229.

Pitermann, M. e K. Munhall (2001). *An inverse dynamics approach to face animation*. Acoustical Society of America. 110 (3): 1570 - 1580.

Pixar (1986). *Luxo Jr*. [http://www.pixar.com] (Visitado em 2 de setembro de 2004)

Popović, Z. e A. Witkin (1999). *Physically Based Motion Transformation*. Proceedings of the 26th annual conference on Computer graphics and interactive techniques: 11 - 20.

Popović, Z. (1999). *Motion Transformation by Physically based spacetime optimization.* Phd Thesis, School of Computer Science. Pittsburgh, Carnegie Mellon University.

Raibert, M. e J. Hodgins (1991). *Animation of dynamic legged locomotion.* Proceedings of the 18th annual conference on Computer graphics and interactive techniques: 349 - 358.

Rhoads, J. (2002). *Helping artists create digital music videos using spacetime constraints.* Technical Report, The Faculty of the School of Engineering and Applied Science, University of Virginia.

Rose, C., B. Bodenheimer e M. Cohen (1998). *Verbs and Adverbs: Multidimensional Motion Interpolation.* IEEE Computer Graphics and Applications. 18(5): 32 - 41.

Rose, C., B. Guenter, B. Bodenheimer e M. Cohen (1996). *Efficient Generation of Motion Transitions using Spacetime Constraints.* Proceedings of the 23rd annual conference on Computer graphics and interactive techniques: 147 - 154.

Sancho-Bru, J., A. Gonzalez, M. Vergara-Monedero e D. Giurintano (2001). *A 3-D dynamic model of human finger for studying free movements.* Jornal of Biomechanics. 34: 1491 - 1500.

Silva, F. W. d. (2004). *Motion Capture - Introdução à Tecnologia.* Laboratório de Computação Gráfica COPPE/UFRJ. [http://www.visgraf.impa.br/Projects/mcapture/publ/mc-tech/] (Visitado em 2 de setembro de 2004)

Simmons, M., J. Wilhelms e A. Gelder (2002). *Model-based reconstruction for creature animation.* Eurographics symposium on Computer animation: 139 - 146.

Tanco, L. e A. Hilton (2000). *Realistic Synthesis of Novel Human Movements from a Database of Motion Capture Examples.* Proceedings of the Workshop on Human Motion (HUMO'00): 137.

Thalmann, N. M. e D. Thalmann (1996a). *Computer Animation.* ACM Computing Surveys. 28(1): 161 - 163.

Thalmann, D., J. Shen e E. Chauvineau (1996b). *Fast Realistic Human Body Deformations for Animation and VR Applications.* Proceedings of the 1996 Conference on Computer Graphics International: 166.

Trolltech. (2004). *Qt Toolkit.* [http://www.trolltech.com] (Visitado em 2 de setembro de 2004)

Watt, A. e M. Watt (1992). *Advanced Animation and Rendering Techniques: Theory and Practice.* ACM Press.

Welman, C. (1989). *Inverse kinematics and geometric constraints for articulated figure manipulation.* Master thesis, Simon Fraser University.

Wells, D. A. (1967). *Lagrangian Dynamics*. McGraw-Hill book Company.

Winzell, P. (1998). *An Implementation of the Spacetime Constraints Approach to the Synthesis of Realistic Motion*. Master thesis, Linköping Institute of Technology in Sweden.

Witkin, A. e M. Kass (1988). *Spacetime Constraints*. Proceedings of the 15th annual conference on Computer graphics and interactive techniques: 159 - 168.

Wolfram Research (2004). *Mathematica*. [http://www.wolfram.com] (Visitado em 2 de setembro de 2004)