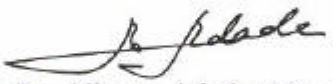
Tese apresentada à Pró-Reitoria de Pós-Graduação e Pesquisa do Instituto Tecnológico de Aeronáutica, como parte dos requisitos para obtenção do título de Mestre em Ciências no Curso de Engenharia Aeronáutica e Mecânica, Área de Mecatrônica e Dinâmica de Sistemas Aeroespaciais

**Glauco Oaklonde de Campos Pacheco**

# ANÁLISE COMPUTACIONAL DE SISTEMAS MULTICORPOS

Tese aprovada em sua versão final pelos abaixo assinados:



Prof. Dr. Alberto Adade Filho
Orientador

Prof. Dr. Homero Santiago Maciel
Pró-Reitor de Pós-Graduação e Pesquisa

Campo Montenegro
São José dos Campos, SP - Brasil
2006

**Dados Internacionais de Catalogação-na-Publicação (CIP)**

**Divisão Biblioteca Central do ITA/CTA**

Pacheco, Glauco Oaklonde de Campos

Análise Computacional de Sistemas Multicorpos / Glauco Oaklonde de Campos Pacheco. São José dos Campos, 2006.

210f.

Tese de Mestrado – Curso de Engenharia Aeronáutica e Mecânica. Área de Mecatrônica e Dinâmica de Sistemas Aeroespaciais – Instituto Tecnológico de Aeronáutica, 2006. Orientador: Prof. Dr. Alberto Adade Filho. .

1. Dinâmica. 2. Computacional. 3. Multicorpos. I. Centro Técnico Aeroespacial. Instituto Tecnológico de Aeronáutica. Divisão de Engenharia Mecânica-Aeronáutica. II. Título.

## REFERÊNCIA BIBLIOGRÁFICA

PACHECO, Glauco Oaklonde de Campos. **Análise Computacional de Sistemas Multicorpos**. 2006. 210f. Tese de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

## CESSÃO DE DIREITOS

NOME DO AUTOR: Glauco Oaklonde de Campos Pacheco

TITULO DO TRABALHO: Análise Computacional de Sistemas Multicorpos.

TIPO DO TRABALHO/ANO: Tese / 2006



---

Glauco Oaklonde de Campos Pacheco

Rua Gavião Peixoto, 343, 1902

CEP 24230-093 – Niterói-RJ

# ANÁLISE COMPUTACIONAL DE SISTEMAS MULTICORPOS

**Glauco Oaklonde de Campos Pacheco**

Composição da Banca Examinadora:

| | | | | |
|---|---|---|---|---|
| Prof. Dr. | Luiz Carlos Sandoval Góes | Presidente | - | ITA |
| Prof. Dr. | Alberto Adade Filho | Orientador | - | ITA |
| Prof. Dr. | Douglas Eduardo Zampieri | Membro Externo | - | UNICAMP |
| Prof. Dr. | João Carlos Menezes | Membro | - | ITA |
| Prof. Dr. | Alfredo Rocha de Faria | Membro | - | ITA |

**ITA**

Este texto foi gerado com o auxílio da classe ITA, versão 2.1, desenvolvida por Fábio Fagundes Silveira, Benedito C. O. Maciel e Giovani Volnei Meinertz.

# Agradecimentos

A Deus, por tudo...

A minha mãe e a meu irmão pelo amor e incentivo, sempre presentes,

Ao meu orientador Adade,

Ao ITA,

A CAPES,

E aos membros da Banca Examinadora.

*"Ignorance is the night of mind,*

*but a night without moon and stars..."*

— Confucius

# Resumo

Neste texto será apresentada a teoria básica da dinâmica computacional, juntamente com alguns métodos de solução para as equações diferenciais-algébricas de movimento. Neste contexto, uma melhoria de um método de ortogonalização existente para solução numérica das equações de movimento será apresentada, caracterizando a contribuição científica deste texto. Do ponto de vista aplicado, foi desenvolvido um programa computacional visando a simulação computacional de sistemas multicorpos. O método implementado neste programa permite a simulação de sistemas rígidos holonômicos arbitrários, conectados por juntas de vários tipos, onde esses sistemas rígidos podem conter cadeias fechadas. As únicas exceções referem-se a situações de contato e impacto, não incorporadas no programa desenvolvido. A simulação de sistemas flexíveis não foi implementada. Contudo, como o programa permite a criação de conjuntos mola-amortecedor lineares e angulares, a representação de barras flexíveis por meio de parâmetros concentrados é simples de ser introduzida. O programa desenvolvido permite ainda ao usuário a inserção de funções MATLAB, abrindo caminho para a implementação de controladores para os sistemas multicorpos. O programa elaborado foi validado utilizando-se alguns exemplos teste, onde cada exemplo concentra-se num aspecto distinto, como complexidade computacional do método de solução, estudo de cadeias fechadas, bem como a análise de sistemas tridimensionais, onde os resultados obtidos com o programa desenvolvido foram

comparados, em termos de acurácia e eficiência, com aqueles fornecidos pelo programa comercial ADAMS. Foi realizada ainda uma aplicação do programa a um robô com três graus de liberdade, controlado pela técnica de torque computado, visando ilustrar a possibilidade de integração do programa desenvolvido com o Matlab. Por fim, cabe ressaltar que o intuito do programa implementado é que este sirva como mais uma ferramenta disponível para a análise computacional na área de robótica do Instituto. Contudo, devido à generalidade do programa, capaz de simular sistemas multicorpos além daqueles comumente encontrados no domínio da robótica, é esperado que este sirva para as inúmeras áreas onde a análise computacional de sistemas multicorpos se faz necessária.

# Abstract

*In this text the basic theory of computational dynamics will be presented, together with some methods for the solution of the differential-algebraic equations of motion. In this context, an improvement of an existing method based on ortogonalization for the numerical solution of the equations of motion will be presented, characterizing the scientific contribution of this text. On the applied point of view, a computational program was developed aiming at the computational simulation of multibody systems. The method implemented in this program allows the simulation of arbitrary holonomic multibody systems, connected with many types of joints, where those multibody systems can contain closed chains. The only exceptions are the situations of contact and impact, not incorporated in the developed program. The simulation of flexible systems was not implemented. However, as the program allows the creation of linear and angular sets of spring-damper elements, the representation of flexible bars by means of lumped parameters is straightforward. The developed program still allows to the user the insertion of functions written in MATLAB, opening way for the implementation of controllers for the multibody systems. The program implemented was validated using some examples, where each example concentrates on a distinct aspect, as computational complexity of the solution method, study of closed chains and the analysis of three-dimensional systems, where the results obtained with the developed program were compared, in terms of accuracy and efficiency, with those supplied*

*by the comercially available program ADAMS. An application of the program to a robot with three degrees of freedom was carried through. This robot was controlled using the technique of computed torque, illustrating the possibility of integration of the program developed with MATLAB. Finally, it fits to stand out that the intention of the implemented program is to serve as an available tool for the computational analysis in the area of robotics of the Institute. However, due to the generality of the program, capable to simulate multibody systems beyond those normally found in the domain of robotics, it is expected that it'll serve for many areas where the computational analysis of multibody systems is necessary.*

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas e Siglas

ADAMS Automatic Dynamic Analysis of Mechanical Systems

# Lista de Símbolos

| | |
|---|---|
| $\mathcal{R}$ | referencial considerado fixo no caso cinemático e inercial global no caso dinâmico |
| $\mathbf{p}^{\mathrm{P/O}}$ | vetor posição do ponto $P$ com respeito ao ponto $O$ |
| ${}^{\mathrm{R}}\mathbf{v}^{\mathrm{P}}$ | vetor velocidade do ponto $P$ com respeito ao referencial $\mathcal{R}$ |
| ${}^{\mathrm{R}}\mathbf{a}^{\mathrm{P}}$ | vetor aceleração do ponto $P$ com respeito ao referencial $\mathcal{R}$ |
| $\mathbf{u}_{\mathrm{i}}$ | vetor $\mathbf{u}$ escrito na base ortonormal do sistema coordenado do corpo $i$ |
| $A^{\mathrm{i}}_{\mathrm{R}}$ | matriz de rotação mapeando o referencial do corpo $i$ no referencial $\mathcal{R}$ |
| ${}^{\mathrm{R}}\boldsymbol{\omega}^{\mathrm{i}}$ | vetor velocidade angular do corpo $i$ com respeito ao referencial $\mathcal{R}$ |
| ${}^{\mathrm{R}}\boldsymbol{\alpha}^{\mathrm{i}}$ | vetor aceleração angular do corpo $i$ com respeito ao referencial $\mathcal{R}$ |
| $\mathbf{e}$ | vetor dos parâmetros de Euler |
| $\mathbf{G}_i$ | matriz que relaciona o vetor velocidade angular com os parâmetros de Euler |
| $\delta$ | prefixo indicando variação |

# 1 Introdução

## 1.1 Considerações Gerais

Este texto estuda a dinâmica computacional de sistemas mecânicos multicorpos rígidos. A dinâmica computacional de sistemas multicorpos consiste no estudo de técnicas que permitam a eficiente implementação computacional dos princípios da mecânica clássica. Como a mecânica clássica repousa sobre uma sólida fundamentação teórica há mais de um século, não existem mais questões abertas neste sentido, restando à dinâmica computacional de sistemas multicorpos buscar descrever as já conhecidas leis da mecânica clássica numa forma matemática conveniente à implementação nos computadores digitais, onde essa conveniência se traduz num compromisso entre a simplificação na obtenção computacional das equações de movimento e sua subseqüente solução numérica. Como a forma das equações de movimento depende diretamente das coordenadas utilizadas na descrição do movimento dos sistemas mecânicos multicorpos, os métodos computacionais distinguem-se de acordo com o critério utilizado na escolha dessas coordenadas. A forma mais empregada na escolha dessas coordenadas, visando a implementação de programas computacionais com a maior aplicabilidade possível, consiste na utilização de um conjunto semelhante de coordenadas para cada um dos corpos do sistema, o que torna o método computacional baseado nesta escolha independente da topologia do sistema, estando as

juntas mecânicas representadas através de equações algébricas não-lineares relacionando essas coordenadas. Esta forma de descrição foi a escolhida neste texto. Esta descrição do movimento representa matematicamente as equações dinâmicas através de um sistema de equações diferenciais-algébricas. Como conjuntos de equações diferenciais-algébricas são de difícil solução, possuindo ainda, do ponto de vista teórico, um certo grau de imaturidade em relação à teoria das equações diferenciais ordinárias, alguns métodos numéricos desenvolvidos para análise dos sistemas multicorpos baseiam-se na representação do sistema mecânico em função somente de equações diferenciais, onde as equações algébricas são derivadas duas vezes em relação ao tempo, sendo escritas portanto em função das acelerações. Neste caso, procuram-se métodos numéricos eficientes para solução das equações resultantes. Uma classe importante desses métodos baseia-se em processos de ortogonalização.

O imenso desenvolvimento dos computadores nas últimas décadas propiciou um avanço tecnológico sem precedentes para nossa sociedade. De fato, hoje em dia lidamos naturalmente com viagens espaciais tripuladas e não-tripuladas, onde robôs altamente sofisticados pousam em planetas distantes; comercializamos bens de consumo manufaturados de forma automatizada dentro de modernas fábricas e, principalmente, como consumidores, demandamos cada vez mais novas tecnologias nos produtos que adquirimos, reforçando essa tendência de desenvolvimento tecnológico. Contudo, devemos notar que embora exista uma grande diversidade de dispositivos destinados a facilitar nossas vidas, grande parte destes deve realizar algum tipo de movimento em sua tarefa, como máquinas em geral, mecanismos, robôs, aviões etc, sendo denominados genericamente por sistemas mecânicos multicorpos. Assim, o estudo do movimento ocupa posição fundamental no projeto desses sistemas. Antes dos computadores entrarem em cena auxiliando

o projeto de sistemas multicorpos, o projeto desses sistemas era conduzido utilizando-se métodos gráficos, pouco precisos, forçando o analista à concepção de sistemas que fossem o mais simples possível. Sistemas mais complexos dependiam fortemente da construção de protótipos, o que encarecia enormemente o projeto de tais sistemas.

Inicialmente o computador era utilizado para resolver numericamente um modelo matemático fornecido pelo usuário. Dessa forma, determinava-se manualmente um modelo matemático representativo do sistema e submetia-se este modelo à solução numérica. Com o grande desenvolvimento dos computadores, modelos cada vez mais complexos tornavam-se passíveis de solução numérica. Entretanto, formular manualmente o modelo matemático, caracterizado pelas equações de movimento no caso dos sistemas mecânicos, era uma árdua tarefa altamente suscetível a erros para sistemas mais complexos, sendo também esses modelos fortemente dependentes do sistema em questão, visto que qualquer modificação do sistema mecânico geralmente implica em grandes modificações nas equações de movimento, devido as não-linearidades presentes, inerentes à maioria dos sistemas mecânicos. Do exposto, observa-se que a formulação manual das equações de movimento dificultava enormemente a otimização desses sistemas, onde várias configurações devem ser testadas e comparadas. A solução encontrada para suplantar essa dificuldade residia na formulação automática das equações de movimento, cuja pesquisa iniciou-se a partir da decáda de 70, principalmente na Alemanha e Estados Unidos da América.

Modernamente, no âmbito da dinâmica dos sistemas multicorpos, o computador é utilizado não só para formular e resolver, de forma automática, as equações de movimento desses sistemas. De fato, todo o avanço computacional das últimas décadas permitiu a solução computacional de sofisticados modelos matemáticos em estações de trabalho, disponíveis a preços acessíveis, o que influenciou o projeto de engenharia como um todo.

Atualmente, o projeto em engenharia acopla fenômenos de várias áreas dentro de um contexto multidisciplinar, onde características mecânicas como tensões, deformações, modos naturais de vibração são analisadas, de forma integrada, ao projeto de controladores, análises térmicas, aeroelásticas, aerodinâmicas etc. Essa tendência multidisciplinar visa, do ponto de vista da indústria, à minimização dos gastos e também à facilitação na incorporação de novas tecnologias aos artigos produzidos. Isto reflete uma nova tendência de mercado, onde tecnologia é um produto que ao ser incorporado agrega valor. Logo, facilitar a incorporação de novas tecnologias torna-se crucial e, como exemplo, a indústria automobilística reflete muito bem esta tendência, visto que a discrepância tecnológica entre os carros de corrida e os veículos de passeio é muito menor atualmente do que a poucas décadas atrás. Outro exemplo importante é o caso da indústria aeronáutica, que empenha em novos projetos de aviões grande parte do capital disponível, não deixando espaço para falhas, já que o fracasso de um projeto desse porte pode comprometer seriamente o futuro da empresa.

Neste contexto, a análise computacional desempenha um papel primordial viabilizando o projeto de sistemas de grande porte a um menor custo, reduzindo a necessidade de construção de seguidos protótipos, uma atividade de capital intensivo, propiciando, através de simulações computacionais, um maior conhecimento do sistema antes que seja construído qualquer protótipo. Contudo, cabe notar que a análise computacional em engenharia e a construção de protótipos não são atividades concorrentes mas sim complementares que, conjuntamente, auxiliam na árdua (e maravilhosa) tarefa do projeto em engenharia.

## 1.2 Revisão Bibliográfica

Os métodos computacionais empregados na formulação automática das equações de movimento distinguem-se pelas coordenadas utilizadas na descrição da configuração espacial dos sistemas mecânicos (SHABANA, 2001), permitindo, dessa forma, a classificação desses métodos a partir do critério de escolha das coordenadas utilizadas. Como as duas principais formas de caracterizar a configuração espacial de um sistema multicorpos são as descrições relativa ou absoluta, os métodos computacionais normalmente enquadram-se numa dessas descrições.

Na descrição relativa, as coordenadas são escolhidas de acordo com o movimento relativo entre os corpos, sendo esse movimento aquele possibilitado pelas juntas que os conectam. Dessa forma, um corpo é descrito em relação a outro. Logo, para obtermos as propriedades cinemáticas de um dado corpo, devemos conhecê-la para o corpo ao qual exista uma descrição relativa, bem como devemos saber transportar a propriedade para o corpo desejado. Assim, como devemos obter as quantidades desejadas de forma recursiva, denomina-se os métodos computacionais baseados na descrição relativa de Métodos Recursivos. Originalmente desenvolvidos para analisar mecanismos de cadeia aberta, esses métodos recursivos evoluíram e atualmente existem formulações (BAE; HAUG, 1987), (BAE; HAUG, 1988a), (BAE; HAUG, 1988b) aplicáveis a uma grande variedade de sistemas multicorpos. Contudo, a vasta maioria dos códigos computacionais genéricos emprega a descrição absoluta (SHABANA, 2001), onde todos os corpos são descritos em relação a um único referencial atribuindo-se um conjunto semelhante de coordenadas para cada um dos corpos do sistema, sendo este referencial considerado como fixo no caso cinemático e inercial no caso dinâmico. Isso ocorre visto que, ao contrário do que acontece com o uso da descrição relativa, é mais fácil formular as equações de movimento fazendo uso da

descrição absoluta. Entretanto, essa maior facilidade tem um custo, já que as equações de movimento, obtidas a partir do emprego da descrição absoluta, possuem grande dimensionalidade. Como o número de graus de liberdade de um sistema multicorpos é independente das coordenadas escolhidas na sua descrição, com o uso da descrição absoluta formulam-se as equações de movimento em função de um conjunto de coordenadas que, geralmente, não é independente, sendo essa dependência caracterizada por equações de restrição, representando as juntas mecânicas.

Assim, conceitualmente, os métodos recursivos não fazem suposição quanto ao movimento dos corpos; este movimento é caracterizado nesses métodos pelas juntas, responsáveis pela introduçao dos movimentos relativos. No caso dos métodos computacionais baseados na descrição absoluta, supõe-se antecipadamente os corpos como não vinculados, estando as juntas representadas pelos movimentos que estas impedem, por meio das equações de restrição. Os métodos recursivos foram desenvolvidos na década de 70 pensando-se unicamente na eficiência, visto que o intuito era permitir a simulação de robôs de cadeia aberta com os recursos computacionais disponíveis na época. Assim, inicialmente, desenvolveu-se esses métodos somente para sistemas de cadeias abertas. Dessa forma, a evolução dos métodos recursivos ocorre da especificidade para a generalização, onde o método de formulação das equações de movimento é aprimorado, utilizando-se basicamente Teoria de Grafos (HWANG; HAUG, 1989), no intuito de explorar a eficiência desses métodos em sistemas mecânicos mais gerais. No caso da descrição absoluta, ocorreu o contrário. A idéia primordial dos métodos baseados na descrição absoluta estava na generalização, ao custo de equações de maior complexidade. Neste caso, a evolução ocorrida não reside na forma como as equações são formuladas mas sim como estas são resolvidas. É neste contexto de aprimoramento dos métodos de solução das equações de

movimento, obtidas a partir da descrição absoluta, que está a contribuição científica deste trabalho.

A grande dificuldade encontrada na solução das equações de movimento formuladas com o uso das coordenadas absolutas está no fato dessas coordenadas existirem, em geral, em número muito superior ao número de graus de liberdade do sistema, significando na prática que essas coordenadas quase sempre não são independentes. Assim, de forma natural, esses sistemas multicorpos são representados por modelos matemáticos constituídos de equações diferenciais-algébricas. Entretanto, como veremos, é possível escrever as equações de movimento envolvendo somente as acelerações e os multiplicadores de Lagrange, representando as juntas mecânicas. Embora essa formulação básica seja atraente à primeira vista, ela não é factível do ponto de vista numérico, visto que as equações de restrição são representadas na sua forma derivada, satisfazendo equações de restrição envolvendo as acelerações. Assim, devido aos erros numéricos acumulados durante o processo de integração das equações de movimento, ocorre o não cumprimento das equações de restrição na sua forma natural, não derivada, envolvendo somente as coordenadas. Isto acarreta a deteriorização do resultado obtido, a ponto de inviabilizá-lo(NIKRAVESH, 1988).

Uma das formas de solucionar o problema descrito acima é utilizar métodos de estabilização de vínculos. O mais conhecido dentre esses métodos é o chamado Método de Estabilização de Baumgarte (BAUMGARTE, 1972), baseado na Teoria de Controle. Embora seja, como veremos posteriormente neste texto, simples de ser implementado, o principal problema deste método está na escolha apropriada de seus dois parâmetros. Uma escolha mal sucedida pode resultar em erros substanciais na solução (SHABANA, 2001). Uma melhoria do método de Baumgarte foi proposta por Pan Zhenkuan, (ZHENKUAN, 1996), onde é apresentada uma forma para obter-se, automaticamente, os parâmetros do método

proposto por Baumgarte. Contudo, neste caso, esses parâmetros tornam-se funções do passo. Na prática, a escolha automática dos parâmetros do método de Baumgarte através do proposto por Zhenkuan força o integrador a escolher um passo muito pequeno. Como esta escolha automática também torna o sistema de equações singular conforme o passo tende a zero, deve-se limitar esses parâmetros a um valor máximo. Uma característica do método de Baumgarte reside no fato da estabilização ser inserida diretamente nas equações de movimento. Outra alternativa que se mostrará muito atraente é utilizar métodos de estabilização separadamente das equações de movimento. Nesses métodos a integração é conduzida normalmente, sendo as restrições verificadas periodicamente. Quando detecta-se uma violação dessas restrições, corrige-se os valores tanto para posição quanto para a velocidade. A grande vantagem desses métodos encontra-se na possibilidade de satisfazer tolerâncias pré-estabelecidas, já que esses métodos são geralmente iterativos, o que não ocorre com o método de Baumgarte. Um desses métodos, e o que será utilizado no programa desenvolvido, é a estabilização baseada na projeção (BLAJER, 1995), onde obtém-se um incremento de correção tanto para posição quanto para velocidade. O método é dito de projeção já que, como veremos posteriormente, essas correções são projetadas no espaço ortogonal à hipersuperfície definida pelas equações de restrição, onde ocorrem os movimentos dependentes. Um outro método de estabilização nessa mesma classe foi proposto por Yu e Chen (YU; CHEN, 2000). Contudo, neste método não há projeção, estando a solução determinada através da utilização da inversa generalizada de Moore-Penrose. A principal vantagem dos métodos de estabilização está na eliminação da necessidade de resolvermos um sistema de equações diferenciais-algébricas, sem que seja necessário, durante o processo de solução das equações de movimento, resolvermos um sistema algébrico não linear, o que torna o método de solução mais eficiente.

Uma outra alternativa visando tornar a solução numérica das equações de movimento mais eficiente, obtidas com o uso de formulações baseadas na descrição absoluta, é empregando o chamado Método de Partição das Coordenadas, proposto por Wehage (WEHAGE, 1980), que consiste na separação das coodenadas em dependentes e independentes. Dessa forma, podemos desacoplar a solução das equações diferencias da solução das equações algébricas não lineares, isto é, podemos expressar as equações de movimento em função do conjunto das coordenadas independentes, em igual número ao número de graus de liberdade do sistema para sistemas holonômicos, representando matematicamente o sistema por um conjunto de equações diferenciais ordinárias, de mais simples solução. Após cada passo da integração dessas equações diferenciais, utilizamos as relações algébricas não-lineraes (equações de restrição), relacionando as coordenadas dependentes e as independentes, para podermos obter os valores das coordenadas dependentes. As derivadas das coordenadas dependentes podem ser obtidas em função das independentes através da solução de um sistema linear. Um dos problemas associados à utilização do método de partição de coordenadas está no fato do número de operações matriciais ser substancialmente maior do que o que ocorre com os outros métodos de solução (SHABANA, 2001). Cabe citar também que, quando empregados métodos implícitos de solução de equações diferenciais ordinárias, é difícil obter-se o jacobiano das equações de movimento (JALóN; BAYO, 1988). Assim, embora o método de partição de coordenadas elimine a introdução artificial de rigidez nas equações de movimento, característica comum de sistemas diferenciais-algébricos, se essa rigidez for natural do sistema multicorpos, será contra producente a aplicação do método de partição de coordenadas.

Uma melhoria que pode ser obtida com o método de partição de coodenadas consiste na ortogonalização, como apresentado por Wojciech Blajer (BLAJER, 1995), onde

é utilizado o método de ortogonalização de Gram-Schmidt, ajustado ao formalismo da geometria Riemaniana. Nesta abordagem, considera-se um sistema representado por $n$ coordenadas e $m$ equações de restrição como um ponto no espaço n-dimensional, sujeito a mover-se na m-variedade, formada pelas equações de restrição. Define-se então um produto interno e uma norma associada a este produto interno, visando gerar-se uma base ortonormal[1] do espaço tangente, caracterizado pelo espaço nulo do jacobiano das equações de restrição, onde ocorrem os movimentos independentes. Assim, ao escrevermos as equações de movimento em função das chamadas velocidades tangentes, obtemos como matriz dos coeficientes do sistema linear a ser resolvido para as derivadas temporais das velocidades tangentes a matriz identidade. Embora este método pareça ser mais eficiente, ainda devemos determinar as coordenadas em função das velocidades tangentes, como também devemos gerar $n - m$ vetores independentes entre si formando uma base para o espaço nulo, para podermos ortogonalizá-los, o que é um grande incoveniente.

## 1.3 Objetivos

A contribuição teórica deste trabalho reside na melhoria de um método de ortogonalização, originalmente proposto por Blajer (BLAJER, 1995). No método proposto, o produto interno definido será diferente e a ortogonalização ocorrerá no espaço linha do jacobiano das equações algébricas não-lineares de restrição, onde ocorrem os movimentos dependentes. Isto eliminará a necessidade de termos que gerar, a cada passo no processo de integração, uma base do espaço nulo do jacobiano das equações de restrição, o que é computacionalmente dispendioso, além de dar origem a matrizes densas durante o processo de solução das equações de movimento, o que é um incoveniente do método proposto por

[1] Ortonormal em relação ao produto interno definido

Blajer. Embora o método apresentado neste texto seja, do ponto de vista numérico, superior àquele originalmente desenvolvido por Blajer, do ponto de vista prático, métodos de solução baseados em ortogonalização ficam restritos a sistemas específicos, visto que para obter as acelerações dos sistemas multicorpos, é mais eficiente resolvermos diretamente o sistema linear representativo das equações de movimento. Assim, o método implementado computacionalmente utiliza uma decomposição $LDL^T$ da matriz simétrica dos coeficientes do sistema linear a ser resolvido para obtermos as acelerações e os multiplicadores de Lagrange. Será utilizado conjuntamente um método de estabilização de vínculos baseado na projeção, sendo ainda desenvolvido um programa computacional para Windows que implementará o procedimento proposto e permitirá sua comparação, em termos de eficiência, com o programa comercial ADAMS.

# 2 Análise de Multicorpos - Aspectos Básicos

Neste capítulo serão tratados os aspectos computacionais básicos da cinemática e dinâmica de sistemas multicorpos. Contudo, a intenção principal não é deduzir e apresentar fórmulas já conhecidas, mas sim mostrar que, apesar de não introduzir nada de novo do ponto de vista teórico, a análise computacional cinemática e dinâmica de sistemas multicorpos requer uma nova abordagem, onde o objetivo principal é facilitar a implementação computacional. Neste contexto, surgem as importantíssimas equações de restrição, desempenhando papel primordial na formulação das equações cinemáticas e dinâmicas de movimento. Outras questões importantes como, por exemplo, a obtenção computacional do número de graus de liberdade de um dado sistema também aparecem inseridas neste contexto, o que nos força a expor essas questões básicas acerca dos sistemas multicorpos no início do texto, na forma de um capítulo.

## 2.1 Análise Cinemática

### 2.1.1 Parâmetros de Euler

Neste texto, serão utilizados como parâmetros de rotação os parâmetros de Euler. Sua escolha deve-se ao fato desses parâmetros não estarem associados a singularidades, bem como de terem seus valores limitados, o que computacionalmente são grandes vantagens (BARUH, 1999). Contudo, antes de introduzirmos os parâmetros de Euler, será deduzida uma fórmula para a matriz de rotação que permitirá uma maior compreensão na definição desses parâmetros.

Como, pelo Teorema de Euler, qualquer orientação de um corpo rígido pode ser obtida a partir de uma única rotação em torno de um eixo que passe pelo corpo[1] (BARUH, 1999), considerando o vetor unitário $\mathbf{v}$ na direção desse eixo, temos que a matriz de rotação é a representação matricial da transformação linear $\mathbf{A} : R^3 \rightarrow R^3$, que associa a cada vetor no $R^3$ um outro vetor, resultado de uma dada rotação $\theta$ em torno do vetor $\mathbf{v}$. Assim, claramente temos que $\mathbf{A}$ leva qualquer vetor paralelo a $\mathbf{v}$ nele mesmo. Em particular $\mathbf{Av} = \mathbf{v}$. Temos também que $\mathbf{A}$ não modifica o módulo do vetor, alterando somente sua orientação. Considerando o vetor $\mathbf{u}$, perpendicular a $\mathbf{v}$ e sendo $\mathbf{A}$ a matriz de rotação de $90^o$, em torno de $\mathbf{v}$, temos que $\mathbf{Au} = \mathbf{v} \times \mathbf{u}$. Como era de se esperar, o resultado foi um vetor ainda ortogonal a $\mathbf{v}$, visto que $\mathbf{u}$ era ortogonal a $\mathbf{v}$ e girou em torno deste vetor. Logo, os vetores $\mathbf{v}$, $\mathbf{v} \times \mathbf{u}$ e $\mathbf{u}$ são mutuamente ortogonais. Dessa forma, caso tenhamos não mais uma rotação de $90^o$, mas uma rotação genérica $\theta$, podemos utilizar estes vetores para escrevermos o vetor $\mathbf{Au}$, resultado dessa rotação de $\theta$ em torno do vetor $\mathbf{v}$. Como o vetor $\mathbf{Au}$ faz um ângulo $\theta$ e $(\frac{\pi}{2} - \theta)$ com os vetores $\mathbf{u}$ e $\mathbf{v} \times \mathbf{u}$, respectivamente,

[1] Aqui não há distinção entre corpo rígido e referencial. Assim, associaremos sistemas coordenados a corpos rígidos livremente.

permanecendo ortogonal ao vetor $\mathbf{v}$, podemos escrever

$$\begin{aligned}\mathbf{Au} &= \mathbf{Au}\cdot\frac{\mathbf{u}}{|\mathbf{u}|}\frac{\mathbf{u}}{|\mathbf{u}|}+\mathbf{Au}\cdot\frac{\mathbf{v}\times\mathbf{u}}{|\mathbf{v}\times\mathbf{u}|}\frac{\mathbf{v}\times\mathbf{u}}{|\mathbf{v}\times\mathbf{u}|}+\mathbf{Au}\cdot\mathbf{v}\,\mathbf{v}\\ &= \cos\theta\,\mathbf{u}+\cos(\frac{\pi}{2}-\theta)\,\mathbf{v}\times\mathbf{u}\\ &= \cos\theta\,\mathbf{u}+\operatorname{sen}\theta\,\mathbf{v}\times\mathbf{u}. \qquad (2.1)\end{aligned}$$

Caso $\mathbf{u}$ seja um vetor de direção arbitrária, podemos decompô-lo na soma de dois vetores, sendo um paralelo e outro ortogonal ao vetor $\mathbf{v}$. Assim, temos que $\mathbf{u}$ pode ser escrito como $\mathbf{u}=\mathbf{u}_{//}+\mathbf{u}_{\perp}$ onde $\mathbf{u}_{//}$ e $\mathbf{u}_{\perp}$ são, respectivamente, paralelo e ortogonal a $\mathbf{v}$. Esses vetores podem ser escritos da forma $\mathbf{u}_{//}=\mathbf{u}\cdot\mathbf{v}\,\mathbf{v}$ e $\mathbf{u}_{\perp}=(\mathbf{v}\times\mathbf{u})\times\mathbf{v}$. Assim, temos que

$$\begin{aligned}\mathbf{Au} &= \mathbf{A}(\mathbf{u}_{//}+\mathbf{u}_{\perp})\\ &= \mathbf{A}(\mathbf{u}\cdot\mathbf{v}\,\mathbf{v}+(\mathbf{v}\times\mathbf{u})\times\mathbf{v})\\ &= \mathbf{u}\cdot\mathbf{v}\mathbf{A}\mathbf{v}+\mathbf{A}((\mathbf{v}\times\mathbf{u})\times\mathbf{v})\\ &= \mathbf{u}\cdot\mathbf{v}\,\mathbf{v}+\cos\theta\,(\mathbf{v}\times\mathbf{u})\times\mathbf{v}+\operatorname{sen}\theta\,\mathbf{v}\times((\mathbf{v}\times\mathbf{u})\times\mathbf{v}), \qquad (2.2)\end{aligned}$$

onde foi aplicada a Eq. (2.1) ao termo $\mathbf{A}((\mathbf{v}\times\mathbf{u})\times\mathbf{v})$.

O resultado obtido na Eq. (2.2) pode ser simplificado se notarmos que

$$\begin{aligned}\mathbf{v}\times((\mathbf{v}\times\mathbf{u})\times\mathbf{v}) &= (\mathbf{v}\cdot\mathbf{v})\mathbf{v}\times\mathbf{u}-\mathbf{v}\cdot(\mathbf{v}\times\mathbf{u})\mathbf{v}\\ &= \mathbf{v}\times\mathbf{u}. \end{aligned} \tag{2.3}$$

Utilizando o resultado obtido na Eq. (2.3), a Eq. (2.2) pode ser escrita da forma

$$\mathbf{A}\mathbf{u} = \mathbf{u}\cdot\mathbf{v}\,\mathbf{v} + \cos\theta\,(\mathbf{v}\times\mathbf{u})\times\mathbf{v} + \operatorname{sen}\theta\,\mathbf{v}\times\mathbf{u}. \tag{2.4}$$

Utilizando as relações trigonométricas $\operatorname{sen}\theta = 2\operatorname{sen}\frac{\theta}{2}\cos\frac{\theta}{2}$ e $\cos\theta = 1-2\operatorname{sen}^2\frac{\theta}{2}$ na Eq. (2.4), obtemos

$$\begin{aligned}\mathbf{A}\mathbf{u} &= \mathbf{u}\cdot\mathbf{v}\,\mathbf{v} + \cos\theta\,(\mathbf{v}\times\mathbf{u})\times\mathbf{v} + \operatorname{sen}\theta\,\mathbf{v}\times\mathbf{u}\\ &= \mathbf{u}_{//} + (1-2\operatorname{sen}^2\frac{\theta}{2})\,(\mathbf{v}\times\mathbf{u})\times\mathbf{v} + 2\operatorname{sen}\frac{\theta}{2}\cos\frac{\theta}{2}\,\mathbf{v}\times\mathbf{u}\\ &= \mathbf{u}_{//} + \mathbf{u}_{\perp} + 2\operatorname{sen}\frac{\theta}{2}\cos\frac{\theta}{2}\,\mathbf{v}\times\mathbf{u} - 2\operatorname{sen}^2\frac{\theta}{2}\,(\mathbf{v}\times\mathbf{u})\times\mathbf{v}\\ &= \mathbf{u} + 2\operatorname{sen}\frac{\theta}{2}\cos\frac{\theta}{2}\,\mathbf{v}\times\mathbf{u} + 2\operatorname{sen}^2\frac{\theta}{2}\,\mathbf{v}\times(\mathbf{v}\times\mathbf{u}). \end{aligned} \tag{2.5}$$

Observando que o produto vetorial entre dois vetores $\mathbf{v} = [v_1\ \ v_2\ \ v_3]^T$ e $\mathbf{u} = [u_1\ \ u_2\ \ u_3]^T$ pode ser escrito na forma matricial $\tilde{\mathbf{v}}\mathbf{u}$, onde $\tilde{\mathbf{v}}$ é uma matriz anti-simétrica associada ao vetor $\mathbf{v}$, da forma

$$\tilde{\mathbf{v}} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{2.6}$$

podemos escrever a Eq. (2.5) como abaixo

$$\begin{aligned} \mathbf{A}\mathbf{u} &= \mathbf{u} + 2\text{sen}\,\frac{\theta}{2}\cos\frac{\theta}{2}\,\mathbf{v}\times\mathbf{u} + 2\text{sen}^{\,2}\frac{\theta}{2}\,\mathbf{v}\times(\mathbf{v}\times\mathbf{u}) \\ &= \mathbf{I}\,\mathbf{u} + 2\text{sen}\,\frac{\theta}{2}\cos\frac{\theta}{2}\,\tilde{\mathbf{v}}\mathbf{u} + 2\text{sen}^{\,2}\frac{\theta}{2}\,\tilde{\mathbf{v}}\tilde{\mathbf{v}}\mathbf{u} \\ &= \mathbf{I}\,\mathbf{u} + 2\,\tilde{\mathbf{v}}\text{sen}\,\frac{\theta}{2}\left[\mathbf{u}\cos\frac{\theta}{2} + \tilde{\mathbf{v}}\mathbf{u}\text{sen}\,\frac{\theta}{2}\right] \\ &= \left[\mathbf{I} + 2\,\tilde{\mathbf{v}}\text{sen}\,\frac{\theta}{2}\left[\mathbf{I}\cos\frac{\theta}{2} + \tilde{\mathbf{v}}\text{sen}\,\frac{\theta}{2}\right]\right]\mathbf{u}. \end{aligned} \tag{2.7}$$

Na Eq. (2.7), identificamos a matriz de rotação, reescrita abaixo

$$\mathbf{A} = \mathbf{I} + 2\,\tilde{\mathbf{v}}\text{sen}\,\frac{\theta}{2}\left[\mathbf{I}\cos\frac{\theta}{2} + \tilde{\mathbf{v}}\text{sen}\,\frac{\theta}{2}\right]. \tag{2.8}$$

Observando-se a Eq. (2.8), apresentada acima, vemos que os termos variáveis desta equação são $\tilde{\mathbf{v}}\text{sen}\,\frac{\theta}{2}$ e $\cos\frac{\theta}{2}$. Como $\tilde{\mathbf{v}}$ é dado pela Eq. (2.6), observa-se que esses termos variáveis são da forma $\cos\frac{\theta}{2}$, $v_1\text{sen}\,\frac{\theta}{2}$, $v_2\text{sen}\,\frac{\theta}{2}$ e $v_3\text{sen}\,\frac{\theta}{2}$.

Definindo então os seguintes parâmetros

$$e_0 = \cos\frac{\theta}{2} \quad e_1 = v_1\text{sen}\,\frac{\theta}{2} \quad e_2 = v_2\text{sen}\,\frac{\theta}{2} \quad e_3 = v_3\text{sen}\,\frac{\theta}{2}, \tag{2.9}$$

temos que a matriz de rotação dada na Eq. (2.8) escrita em função desses parâmetros fica

da forma

$$A = \begin{bmatrix} 1-2e_2^2-2e_3^2 & 2(e_1e_2-e_0e_3) & 2(e_1e_3+e_0e_2) \\ 2(e_1e_2+e_0e_3) & 1-2e_1^2-2e_3^2 & 2(e_2e_3-e_0e_1) \\ 2(e_1e_3-e_0e_2) & 2(e_2e_3+e_0e_1) & 1-2e_1^2-2e_2^2 \end{bmatrix}. \tag{2.10}$$

Os parâmetros definidos na Eq. (2.9) são denominados de Parâmetros de Euler. A partir de sua definição, vê-se que esses parâmetros não são independentes, já que

$$\mathbf{e}^T\mathbf{e} = 1, \tag{2.11}$$

onde

$$\mathbf{e} = \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix}. \tag{2.12}$$

Antes de darmos esta seção por encerrada, cabe notar que os parâmetros de Euler estão inseridos num contexto mais amplo, onde as rotações são representadas utilizando-se os quaternions (os parâmetros de Euler são, de fato, um exemplo de quaternions). Esta abordagem permite a algebrização das rotações, permitindo que propriedades geométricas sejam interpretadas de forma algébrica.

### 2.1.2 Coordenadas Generalizadas

Nesta dissertação será utilizada a descrição absoluta para descrever a configuração espacial dos sistemas multicorpos. Isto significa que um conjunto semelhante de coordenadas será atribuído a cada um dos corpos, sendo este conjunto caracterizado por três coordenadas representando a translação do centro de massa do corpo, conjuntamente com os Parâmetros de Euler do corpo, definindo sua orientação, introduzidos na seção anterior.

Nesta seção serão apresentadas as equações cinemáticas básicas, relacionando posições, velocidades e acelerações de pontos em referenciais (corpos) distintos. A notação utilizada é semelhante àquela apresentada em (KANE; LEVINSON, 1985; TENENBAUM, 1997). Estas equações, encontradas em livros elementares de mecânica, são apresentadas abaixo (ver Fig. (2.1)).

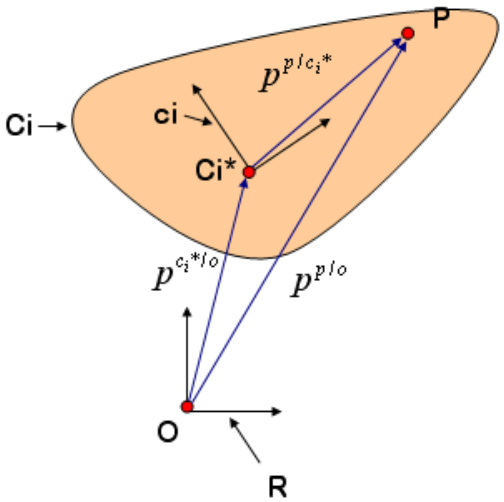



FIGURA 2.1 – Corpo rígido.

$$\mathbf{p}^{P/O} = \mathbf{p}^{P/Ci^*} + \mathbf{p}^{Ci^*/O}, \tag{2.13}$$

$${}^{R}\mathbf{v}^{P} = {}^{Ci}\mathbf{v}^{P} + {}^{R}\mathbf{v}^{Ci^*} + {}^{R}\boldsymbol{\omega}^{Ci} \times \mathbf{p}^{P/Ci^*}, \tag{2.14}$$

$${}^{R}\mathbf{a}^{P} = {}^{Ci}\mathbf{a}^{P} + {}^{R}\mathbf{a}^{Ci^*} + {}^{R}\boldsymbol{\omega}^{Ci} \times ({}^{R}\boldsymbol{\omega}^{Ci} \times \mathbf{p}^{P/Ci^*}) + {}^{R}\boldsymbol{\alpha}^{Ci} \times \mathbf{p}^{P/Ci^*} + 2{}^{R}\boldsymbol{\omega}^{Ci} \times {}^{Ci}\mathbf{v}^{P}, \tag{2.15}$$

onde ${}^{R}\boldsymbol{\omega}^{Ci}$ é o vetor velocidade angular do corpo, ${}^{R}\boldsymbol{\alpha}^{Ci}$ é o vetor aceleração angular do corpo, $R$ é o referencial global, cujo sistema coordenado possui origem no ponto $O$, e $Ci^*$ é o centro de massa do corpo $Ci$, origem do sistema coordenado $ci$, afixado ao corpo.

Considerando-se o ponto $P$ como fixo no corpo, as equações anteriores para velocidade e aceleração, Eqs. (2.14, 2.15), simplificam-se, ficando da forma

$${}^{R}\mathbf{v}^{P} = {}^{R}\mathbf{v}^{Ci^*} + {}^{R}\boldsymbol{\omega}^{C} \times \mathbf{p}^{P/Ci^*} \tag{2.16}$$

$${}^{R}\mathbf{a}^{P} = {}^{R}\mathbf{a}^{Ci^*} + {}^{R}\boldsymbol{\omega}^{Ci} \times ({}^{R}\boldsymbol{\omega}^{Ci} \times \mathbf{p}^{P/Ci^*}) + {}^{R}\boldsymbol{\alpha}^{Ci} \times \mathbf{p}^{P/Ci^*} \tag{2.17}$$

Embora as equações para posição, velocidade e aceleração mostradas já apresentem uma notação carregada, ainda falta explicitar em que base estes vetores estão escritos. Como corpos rígidos podem ser considerados referenciais, associaremos, a cada corpo $C_i$ do sistema multicorpos, um sistema coordenado $ci$, com origem no centro de massa e uma base ortonormal, e todo vetor $\mathbf{u}$ representado nesse sistema será escrito da forma $\mathbf{u}_{ci}$[2]. No sistema coordenado associado ao referencial global este vetor será escrito como $\mathbf{u}_{R}$. Dessa forma, as Eqs. (2.13, 2.16, 2.17) podem ser escritas como abaixo (ver Fig. (2.1))

[2]Cabe notar que um referencial pode ter mais de um sistema coordenado associado. Contudo, neste caso os sistemas coordenados serão identificados por nomes distintos, o que evitará qualquer dubiedade na notação utilizada.

$$\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/O}} = \mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci^*}} + \mathbf{p}_{\mathrm{ci}}^{\mathrm{Ci^*/O}}, \tag{2.18}$$

$${}^{\mathrm{R}}\mathbf{v}_{\mathrm{ci}}^{\mathrm{P}} = {}^{\mathrm{R}}\mathbf{v}_{\mathrm{ci}}^{\mathrm{Ci^*}} + {}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{ci}}^{\mathrm{Ci}} \times \mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci^*}}, \tag{2.19}$$

$${}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} = {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{Ci^*}} + {}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{R}}^{\mathrm{Ci}} \times ({}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{R}}^{\mathrm{Ci}} \times \mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci^*}}) + {}^{\mathrm{R}}\boldsymbol{\alpha}_{\mathrm{R}}^{\mathrm{Ci}} \times \mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci^*}}, \tag{2.20}$$

onde as Eqs. (2.18, 2.19) estão escritas no sistema *ci* do corpo e a Eq. (2.20) está escrita no sistema coordenado associado ao referencial global.

Embora as Eqs. (2.18, 2.19, 2.20) estejam, cada uma delas, escritas com todos os seus vetores num mesmo sistema coordenado, isto não é necessário. De fato, a principal função da notação empregada é permitir que possamos escrever as equações vetoriais com seus termos representados em diferentes sistemas coordenados. Neste caso, faz-se uso das matrizes de rotação. À matriz de rotação relacionando os sistemas coordenados $T$ e $S$ será atribuída a seguinte notação: ${}^{\mathrm{S}}\mathbf{A}^{\mathrm{T}}$, onde essa matriz mapeia vetores escritos no sistema coordenado $T$ ao sistema coordenado $S$. À primeira vista, a notação utilizada pode parecer demasiadamente carregada. Contudo, conforme formos avançando no texto, veremos que a notação adotada facilitará sobremaneira a compreensão das equações dinâmicas de movimento, evitando ambigüidades em sua interpretação.

As equações dinâmicas, obtidas com o uso de métodos baseados nas coordenadas absolutas, devem ser escritas em função dessas coordenadas. Assim, definindo-se, para cada corpo $C_i$ do sistema multicorpos, o vetor de coordenadas generalizadas $\mathbf{q}_i$, representativo das coordenadas absolutas, como mostrado abaixo

$$\mathbf{q}_i^T = \left[ (\mathbf{p}_{\mathrm{R}}^{\mathrm{Ci^*/O}})^T \quad \mathbf{e}^T \right], \tag{2.21}$$

é desejável que possamos descrever a cinemática dos corpos diretamente em função de $\mathbf{q}_i$. Dessa forma, devemos escrever as equações para velocidade e aceleração diretamente em função desse vetor e suas derivadas temporais. Essas equações, considerando-se o ponto $P$ fixo ao corpo, são da forma

$$\begin{aligned}
{}^{\mathrm{R}}\mathbf{v}_{\mathrm{R}}^{\mathrm{P}} &= {}^{\mathrm{R}}\mathbf{v}_{\mathrm{R}}^{\mathrm{Ci}^*} + {}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{R}}^{\mathrm{Ci}} \times \mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci}^*} \\
&= {}^{\mathrm{R}}\mathbf{v}_{\mathrm{R}}^{\mathrm{Ci}^*} - \mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci}^*} \times {}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{R}}^{\mathrm{Ci}} \\
&= {}^{\mathrm{R}}\mathbf{v}_{\mathrm{R}}^{\mathrm{Ci}^*} - \tilde{\mathbf{p}}_{\mathrm{R}}^{\mathrm{P/Ci}^*}\, {}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{R}}^{\mathrm{Ci}} \\
&= {}^{\mathrm{R}}\mathbf{v}_{\mathrm{R}}^{\mathrm{Ci}^*} - \tilde{\mathbf{p}}_{\mathrm{R}}^{\mathrm{P/Ci}^*}\, \mathbf{G}_{\mathrm{R}}^{\mathrm{Ci}}\, \dot{\mathbf{e}} \qquad (2.22) \\
&= [\mathbf{I}_{3\mathrm{x}3} \quad - \tilde{\mathbf{p}}_{\mathrm{R}}^{\mathrm{P/Ci}^*}\, \mathbf{G}_{\mathrm{R}}^{\mathrm{Ci}}]\, \dot{\mathbf{q}}_i, \qquad (2.23)
\end{aligned}$$

para velocidade, onde $\mathbf{I}_{3\mathrm{x}3}$ é a matriz unitária $3 \times 3$, $\tilde{\mathbf{p}}_{\mathrm{R}}^{\mathrm{P/Ci}^*}$ é a matriz anti-simétrica associada ao vetor $\mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci}^*}$ (ver Eq. (2.6)), e $\mathbf{G}_{\mathrm{R}}^{\mathrm{Ci}}$ é dada abaixo

$$\mathbf{G}_{\mathrm{R}}^{\mathrm{Ci}} = 2 \begin{bmatrix} -e_1 & e_0 & -e_3 & e_2 \\ -e_2 & e_3 & e_0 & -e_1 \\ -e_3 & -e_2 & e_1 & e_0 \end{bmatrix}. \qquad (2.24)$$

Analogamente a equação para aceleração é dada por

$$
\begin{aligned}
{}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} &= {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{Ci}*} + {}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{R}}^{\mathrm{Ci}} \times ({}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{R}}^{\mathrm{Ci}} \times \mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci}*}) + {}^{\mathrm{R}}\boldsymbol{\alpha}_{\mathrm{R}}^{\mathrm{Ci}} \times \mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci}*} \\
&= {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{Ci}*} + {}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{R}}^{\mathrm{Ci}}\, {}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{R}}^{\mathrm{Ci}}\, \mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci}*} - \tilde{\mathbf{p}}_{\mathrm{R}}^{\mathrm{P/Ci}*}\, {}^{\mathrm{R}}\boldsymbol{\alpha}_{\mathrm{R}}^{\mathrm{Ci}} \\
&= {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{Ci}*} + ({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{R}}^{\mathrm{Ci}})^2\, \mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci}*} - \tilde{\mathbf{p}}_{\mathrm{R}}^{\mathrm{P/Ci}*}\, \mathbf{G}_{\mathrm{R}}^{\mathrm{Ci}}\, \ddot{\mathbf{e}} \qquad (2.25) \\
&= [\mathbf{I}_{3\mathrm{x}3} \quad - \tilde{\mathbf{p}}_{\mathrm{R}}^{\mathrm{P/Ci}*}\, \mathbf{G}_{\mathrm{R}}^{\mathrm{Ci}}]\, \ddot{\mathbf{q}}_i + ({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{R}}^{\mathrm{Ci}})^2\, \mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci}*}. \qquad (2.26)
\end{aligned}
$$

Por fim, vamos escrever as Eqs. (2.22, 2.25) considerando o vetor $\mathbf{p}^{\mathrm{P/Ci}*}$ descrito no sistema do corpo $C_i$. Observando-se que, em termos de matrizes anti-simétricas, $\tilde{\mathbf{p}}_{\mathrm{R}}^{\mathrm{P/Ci}*} = {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, \tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci}*}\, ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}})^T$, temos que

$$
\begin{aligned}
{}^{\mathrm{R}}\mathbf{v}_{\mathrm{R}}^{\mathrm{P}} &= {}^{\mathrm{R}}\mathbf{v}_{\mathrm{R}}^{\mathrm{Ci}*} - {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, \tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci}*}\, ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}})^T\, \mathbf{G}_{\mathrm{R}}^{\mathrm{Ci}}\, \dot{\mathbf{e}} \\
&= {}^{\mathrm{R}}\mathbf{v}_{\mathrm{R}}^{\mathrm{Ci}*} - {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, \tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci}*}\, \mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}}\, \dot{\mathbf{e}}, \qquad (2.27)
\end{aligned}
$$

para velocidade, onde $\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}} = ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}})^T\, \mathbf{G}_{\mathrm{R}}^{\mathrm{Ci}}$. A partir da Eq. (2.27) podemos escrever

$$
\delta\mathbf{p}_{\mathrm{R}}^{\mathrm{P/O}} = \delta\mathbf{p}_{\mathrm{R}}^{\mathrm{Ci}*/\mathrm{O}} - {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, \tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci}*}\, \mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}}\, \delta\mathbf{e}., \qquad (2.28)
$$

onde $\delta\mathbf{p}_{\mathrm{R}}^{\mathrm{P/O}}$ é o deslocamento virtual do vetor $\delta\mathbf{p}_{\mathrm{R}}^{\mathrm{P/O}}$.

Para aceleração a equação fica da forma

$$
{}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} = {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{Ci}*} + {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, ({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2\, \mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci}*} - {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, \tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci}*}\, \mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}}\, \ddot{\mathbf{e}}. \qquad (2.29)
$$

As Eqs. (2.28, 2.29) serão utilizadas na dedução da força de inércia generalizada, obtida mais adiante, onde é melhor ter o vetor $\mathbf{p}^{P/Ci^*}$ descrito no sistema coordenado do corpo.

### 2.1.3 Equações de Restrição

Embora o uso das coordenadas absolutas facilite a descrição cinemática dos sistemas multicorpos, simplificando a implementação computacional dos métodos baseados nesta descrição, as coordenadas absolutas aumentam a dimensionalidade do sistema, isto é, existe, em geral, um número de coordenadas muito superior ao número de graus de liberdade. Isso ocorre devido às juntas, que diminuem o número de graus de liberdade do sistema. Dessa forma, é preciso representar essas juntas, sendo essa representação caracterizada pelas equações de restrição. Neste ponto podemos diferenciar conceitualmente o método baseado nas coordenadas absolutas do método recursivo. Enquanto o primeiro permite, como veremos adiante nesta seção, a obtenção das equações de restrição de forma sistemática, considerando os corpos como não vinculados e representando as juntas pelos movimentos que estas impossibilitam, o método recursivo representa as juntas pela mobilidade que estas acrescentam. Contudo, enquanto no método recursivo os sistemas de cadeia aberta não necessitam que suas juntas sejam representadas através de equações de restrição, estas são necessárias em sistemas de cadeia fechada. Todavia, neste caso, a geração dessas equações de restrição é mais difícil de ser sistematizada. Assim, conceitualmente, esses métodos são opostos no sentido que enquanto num deles (coordenadas absolutas) supõe-se o sistema como livre e adiciona-se restrições ao movimento, no outro (método recursivo) não se faz nenhuma suposição *a priori* sobre o movimento, sendo a mobilidade introduzida com as juntas.

#### 2.1.3.1 Equações representando Juntas

Os sistemas multicorpos podem ter os seus corpos conectados por uma grande variedade de juntas, como juntas esféricas, cilíndricas, prismáticas etc. Entretanto, essas juntas são representadas por um conjunto de equações de restrição básicas, que impedem rotações ou translações relativas. Dessa forma, antes de definirmos as juntas mais comuns, vamos apresentar três dessas equações básicas. Inicialmente será introduzida a equação de restrição que impossibilita a translação relativa entre corpos.

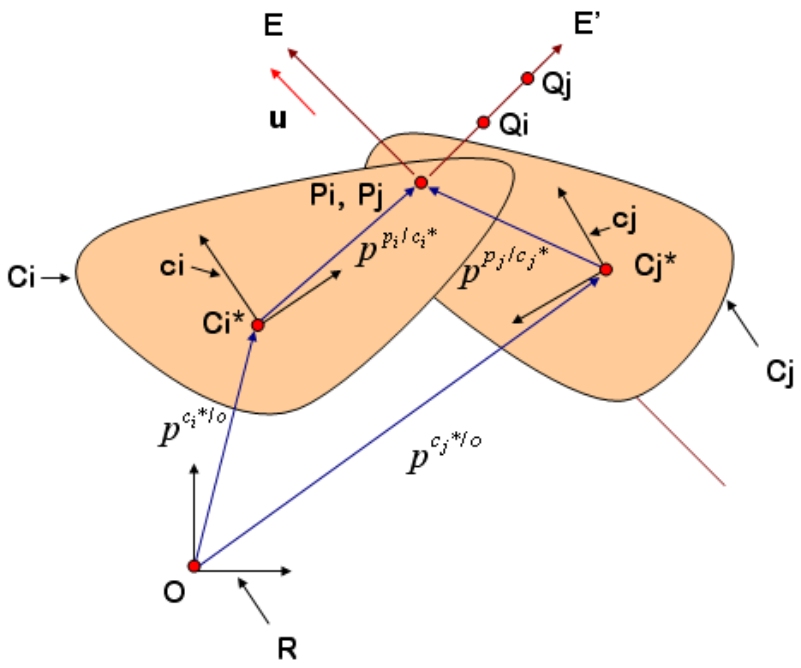



FIGURA 2.2 – Restrição entre dois corpos.

Sejam $C_i$ e $C_j$ dois corpos rígidos (ver Fig. (2.2)) que não possuem translação relativa entre si. Neste caso, podemos modelar esta restrição definindo dois pontos $P_i$ e $P_j$, nos corpos $C_i$ e $C_j$ respectivamente, assegurando que eles sempre permaneçam em contato.

Procedendo dessa forma, podemos representar esta restrição como abaixo

$$\mathbf{\Phi}_{3t}(\mathbf{q}_i, \mathbf{q}_j) = \mathbf{p}_{\mathrm{R}}^{\mathrm{Ci^*/O}} + {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, \mathbf{p}_{\mathrm{ci}}^{\mathrm{Pi/Ci^*}} - \mathbf{p}_{\mathrm{R}}^{\mathrm{Cj^*/O}} - {}^{\mathrm{R}}\mathbf{A}^{\mathrm{cj}}\, \mathbf{p}_{\mathrm{cj}}^{\mathrm{Pj/Cj^*}} = \mathbf{0}, \tag{2.30}$$

onde $\mathbf{q}_i$ e $\mathbf{q}_j$ são os vetores das coordenadas generalizadas dos corpos $C_i$ e $C_j$ (ver Eq. (2.21)), e o subíndice $3t$ indica que esta restrição elimina três translações relativas. Na formulação computacional da Eq. (2.30), os vetores $\mathbf{p}_{\mathrm{ci}}^{\mathrm{Pi/Ci^*}}$ e $\mathbf{p}_{\mathrm{cj}}^{\mathrm{Pj/Cj^*}}$ são constantes, devido à hipótese de rigidez dos corpos. Satisfazendo-se essa equação, garante-se que os pontos $P_i$ e $P_j$ coincidirão durante o movimento, eliminando a translação relativa entre os corpos $C_i$ e $C_j$.

A segunda restrição a ser apresentada impossibilita a rotação relativa em torno de um dado eixo. Considerando-se ainda os corpos $C_i$ e $C_j$ definidos anteriormente (ver Fig. (2.2)), vamos restringir a rotação relativa em torno do eixo $E$. Para tal, basta obter dois vetores $\mathbf{v}^{\mathrm{i}}$ e $\mathbf{v}^{\mathrm{j}}$ (não mostrados na figura mencionada), ortogonais ao eixo $E$ e definidos nos corpos $C_i$ e $C_j$, respectivamente, tais que, inicialmente, $\mathbf{v}^{\mathrm{i}} \cdot \mathbf{v}^{\mathrm{j}} = 0$. Assim, para garantir que esses corpos não possuam rotação relativa em torno do eixo citado, basta assegurar que

$$\mathbf{\Phi}_{1r}(\mathbf{q}_i, \mathbf{q}_j) = ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, \mathbf{v}_{\mathrm{ci}}^{\mathrm{i}})^T\, {}^{\mathrm{R}}\mathbf{A}^{\mathrm{cj}}\, \mathbf{v}_{\mathrm{cj}}^{\mathrm{j}} = 0, \tag{2.31}$$

durante todo o movimento dos corpos em questão, onde os vetores $\mathbf{v}_{\mathrm{ci}}^{\mathrm{i}}$ e $\mathbf{v}_{\mathrm{cj}}^{\mathrm{j}}$ são constantes. Novamente, note que o subíndice $1r$, utilizado na equação indica que esta restrição impede uma rotação relativa.

A terceira e última equação básica a ser mostrada não permite a translação relativa ao longo de um dado eixo. Mais uma vez, considerando os corpos $C_i$ e $C_j$ (ver Fig. (2.2)), vamos obter uma equação que impossibilite a translação relativa entre os corpos na direção

TABELA 2.1 – Definição das juntas mecânicas

| Junta | $\mathbf{\Phi}_{1t}$ | $\mathbf{\Phi}_{3t}$ | $\mathbf{\Phi}_{1r}$ |
|---|---|---|---|
| Esférica | - | 1 | - |
| Revoluta | - | 1 | 2 |
| Cilindrica | 2 | - | 2 |
| Prismática | 2 | - | 3 |
| Fixa | - | 1 | 3 |
| Plana | 1 | - | 2 |

do eixo $E$. Para tal, basta definirmos um eixo perpendicular ao eixo $E$, e dois pontos sobre este eixo. Na Fig. (2.2), este eixo ortogonal é denominado $E'$ e os pontos $Q_i$, $Q_j$, definidos respectivamente nos corpos $C_i$ e $C_j$, estão situados sobre este eixo. Sendo $\mathbf{u}$ um vetor unitário na direção do eixo $E$, podemos escrever esta equação de restrição como abaixo

$$\mathbf{\Phi}_{1t}(\mathbf{q}_i, \mathbf{q}_j) = ({}^{R}\mathbf{A}^{ci}\,\mathbf{u}_{ci})^T(\mathbf{p}_{R}^{Ci^*/O} + {}^{R}\mathbf{A}^{ci}\,\mathbf{p}_{ci}^{Qi/Ci^*} - \mathbf{p}_{R}^{Cj^*/O} - {}^{R}\mathbf{A}^{cj}\,\mathbf{p}_{cj}^{Qj/Cj^*}) = 0, \tag{2.32}$$

onde os vetores $\mathbf{u}_{ci}$, $\mathbf{p}_{ci}^{Qi/Ci^*}$ e $\mathbf{p}_{cj}^{Qj/Cj^*}$ são definidos inicialmente e permanecem constantes durante o movimento dos corpos. Mais uma vez, utilizamos um subíndice (1t) para indicar que a restrição impede uma translação relativa.

Tendo definido as equações básicas de restrição, vamos formular as equações para as juntas mecânicas. Essas juntas serão representadas matematicamente por vetores, cujos componentes serão as equações básicas apresentadas anteriormente. A Tab. (2.1) ilustra a definição dessas juntas.

Assim, por exemplo, uma junta revoluta é representada matematicamente da forma

$$\mathbf{\Phi}_{\text{rev}} = \begin{bmatrix} \mathbf{\Phi}_{\text{3t}} \\ \mathbf{\Phi}_{\mathbf{1}_{\text{1r}}} \\ \mathbf{\Phi}_{\mathbf{2}_{\text{1r}}} \end{bmatrix}, \tag{2.33}$$

onde $\mathbf{\Phi}_{\text{rev}}$ indica a equação representativa de uma junta revoluta, sendo esta equação constituída de três equações básicas a saber: $\mathbf{\Phi}_{\text{3t}}$ impedindo três translações translativas e as duas equações $\mathbf{\Phi}_{\mathbf{1}_{\text{1r}}}$ e $\mathbf{\Phi}_{\mathbf{1}_{\text{1r}}}$, impedindo, cada uma delas, uma rotação relativa. Observe que neste caso diferenciamos estas duas últimas equações de restrição por números nos subíndices.

Cabe notar que as equações básicas, e por conseguinte as juntas mecânicas a partir delas obtidas, possuem a mesma estrutura, independentemente de quais corpos do sistema estejam conectados pela junta, sistematizando, dessa forma, as equações para as juntas mecânicas. Note que esta sistematização não significa que seja fácil obter-se essas equações. A simplificação na obtenção das equações de restrição só é atingida através do uso de um mesmo conjunto de coordenadas para cada um dos corpos, isto é, utilizando-se as coordenadas absolutas. Na próxima seção será discutida a análise cinemática de sistemas multicorpos, onde várias operações envolvendo as equações de restrição representativas das juntas serão efetuadas. Visando à simplificação da análise a ser realizada, vamos definir essas operações já na atual seção, para as equações básicas anteriormente apresentadas. Observando-se que

$$ {}^{\mathrm{R}}\dot{\mathbf{A}}^{\mathrm{ci}}\mathbf{w} = \mathbf{G}_{\mathrm{R}}^{\mathrm{ci}}\,\breve{\mathbf{w}}\,\dot{\mathbf{e}}_{\mathrm{i}} \tag{2.34} $$

$$ {}^{\mathrm{R}}\ddot{\mathbf{A}}^{\mathrm{ci}}\,\mathbf{w} = \mathbf{G}_{\mathrm{R}}^{\mathrm{ci}}\,\breve{\mathbf{w}}\,\ddot{\mathbf{e}}_{\mathrm{i}} + \dot{\mathbf{G}}_{\mathrm{R}}^{\mathrm{ci}}\,\breve{\mathbf{w}}\,\dot{\mathbf{e}}_{\mathrm{i}} \tag{2.35} $$

onde ${}^{\mathrm{R}}\dot{\mathbf{A}}^{\mathrm{ci}}$ é a matriz de rotação já apresentada e $\mathbf{w}$ é um vetor tridimensional arbitrário, podemos escrever a derivada temporal das equações básicas de restrição, Eqs. (2.30,2.31,2.32), da forma

$$ \dot{\mathbf{\Phi}}_{3\mathrm{t}} = \begin{bmatrix} \mathbf{I}_{3\mathrm{x}3} & \mathbf{G}_{\mathrm{R}}^{\mathrm{ci}}\,\breve{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{Pi/Ci*}} & -\mathbf{I}_{3\mathrm{x}3} & -\mathbf{G}_{\mathrm{R}}^{\mathrm{cj}}\,\breve{\mathbf{p}}_{\mathrm{cj}}^{\mathrm{Pj/Cj*}} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_i \\ \dot{\mathbf{q}}_j \end{bmatrix} \tag{2.36} $$

$$ \dot{\mathbf{\Phi}}_{1\mathrm{r}} = \begin{bmatrix} \mathbf{0}_{3\mathrm{x}3} & \left({}^{\mathrm{R}}\mathbf{A}^{\mathrm{cj}}\,\mathbf{v}_{\mathrm{cj}}\right)^{T}\mathbf{G}_{\mathrm{R}}^{\mathrm{ci}}\,\breve{\mathbf{v}}_{\mathrm{ci}} & \mathbf{0}_{3\mathrm{x}3} & \left({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\mathbf{v}_{\mathrm{ci}}\right)^{T}\mathbf{G}_{\mathrm{R}}^{\mathrm{cj}}\,\breve{\mathbf{v}}_{\mathrm{cj}} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_i \\ \dot{\mathbf{q}}_j \end{bmatrix} \tag{2.37} $$

$$ \dot{\mathbf{\Phi}}_{1\mathrm{t}} = \begin{bmatrix} \left({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\mathbf{u}_{\mathrm{ci}}\right)^{T} & \left\{\left({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\mathbf{u}_{\mathrm{ci}}\right)^{T}\mathbf{G}_{\mathrm{R}}^{\mathrm{ci}}\,\breve{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{Qi/Ci*}} + (\mathbf{r})^{T}\mathbf{G}_{\mathrm{R}}^{\mathrm{ci}}\,\breve{\mathbf{u}}_{\mathrm{ci}}\right\} & -\left({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\mathbf{u}_{\mathrm{ci}}\right)^{T} & \left({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\mathbf{u}_{\mathrm{ci}}\right)^{T}\mathbf{G}_{\mathrm{R}}^{\mathrm{ci}}\,\breve{\mathbf{p}}_{\mathrm{cj}}^{\mathrm{Qj/Cj*}} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_i \\ \dot{\mathbf{q}}_j \end{bmatrix} \tag{2.38} $$

onde

$$ \mathbf{r} = \mathbf{p}_{\mathrm{R}}^{\mathrm{Ci*/O}} + {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\mathbf{p}_{\mathrm{ci}}^{\mathrm{Qi/Ci*}} - \mathbf{p}_{\mathrm{R}}^{\mathrm{Cj*/O}} - {}^{\mathrm{R}}\mathbf{A}^{\mathrm{cj}}\,\mathbf{p}_{\mathrm{cj}}^{\mathrm{Qj/Cj*}}, \tag{2.39} $$

e a notação $\breve{}$, utilizada nas Eqs. (2.36, 2.37, 2.38) para os vetores $\mathbf{p}_{\mathrm{ci}}^{\mathrm{Pi/Ci*}}$, $\mathbf{p}_{\mathrm{cj}}^{\mathrm{Pj/Cj*}}$, $\mathbf{v}_{\mathrm{ci}}$ e $\mathbf{v}_{\mathrm{cj}}$, pode ser empregada a qualquer vetor tridimensional. Sendo $\mathbf{w}$ um vetor tridimensional genérico, temos que $\breve{\mathbf{w}}$ é da forma

$$ \breve{\mathbf{w}} = \begin{bmatrix} 0 & -\mathbf{w}^{T} \\ \mathbf{w} & -\tilde{\mathbf{w}} \end{bmatrix}, \tag{2.40} $$

uma matriz de dimensão $4 \times 4$, onde $\tilde{\mathbf{w}}$ é a matriz anti-simétrica associada ao vetor $\mathbf{w}$ (ver Eq. (2.6)).

As equações para a derivada temporal das equações de restrição básicas também poderiam ser obtidas utilizando-se a regra da cadeia da seguinte forma

$$\dot{\mathbf{\Phi}}_{\text{-}} = (\mathbf{\Phi}_{\text{-}})_{\mathbf{q}}\dot{\mathbf{q}} + \frac{\partial \mathbf{\Phi}_{\text{-}}}{\partial t} = (\mathbf{\Phi}_{\text{-}})_{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}_{\text{t}}. \tag{2.41}$$

Na equação apresentada acima podemos substituir o índice − pelo correspondente a qualquer uma das três equações básicas introduzidas. O termo $(\mathbf{\Phi}_{\text{-}})_{\mathbf{q}}$ refere-se ao jacobiano da equação de restrição e o termo $\mathbf{C}_{\text{t}}$ é identicamente nulo[3]. Os jacobianos para as três equações básicas de restrição podem ser obtidos de forma imediata a partir das Eqs. (2.36, 2.37, 2.38).

A derivada temporal segunda das equações básicas de restrição pode ser escrita da forma

$$\ddot{\mathbf{\Phi}}_{\text{-}} = (\mathbf{\Phi}_{\text{-}})_{\mathbf{q}}\ddot{\mathbf{q}} - \mathbf{Q}_{\text{d-}}. \tag{2.42}$$

Como os jacobianos já foram apresentados, ver Eqs. (2.36, 2.37,- 2.38), basta mostrar os vetores $\mathbf{Q}_{\text{d-}}$ para as equações básicas. Estes vetores são mostrados abaixo

$$\mathbf{Q}_{\text{d3t}} = -\dot{\mathbf{G}}_{\text{R}}^{\text{ci}}\,\breve{\mathbf{p}}_{\text{ci}}^{\text{Pi/Ci*}}\,\dot{\mathbf{e}}_i + \dot{\mathbf{G}}_{\text{R}}^{\text{cj}}\,\breve{\mathbf{p}}_{\text{cj}}^{\text{Pj/Cj*}}\,\dot{\mathbf{e}}_j \tag{2.43}$$

$$\mathbf{Q}_{\text{d1r}} = -({}^{\text{R}}\mathbf{A}^{\text{cj}}\,\mathbf{v}_{\text{j}})^T\,\dot{\mathbf{G}}_{\text{R}}^{\text{ci}}\,\breve{\mathbf{p}}_{\text{ci}}^{\text{Pi/Ci*}}\,\dot{\mathbf{e}}_i - ({}^{\text{R}}\mathbf{A}^{\text{ci}}\,\mathbf{v}_{\text{i}})^T\,\dot{\mathbf{G}}_{\text{R}}^{\text{cj}}\,\breve{\mathbf{p}}_{\text{cj}}^{\text{Pj/Cj*}}\,\dot{\mathbf{e}}_j - 2(\mathbf{G}_{\text{R}}^{\text{ci}}\,\breve{\mathbf{p}}_{\text{ci}}^{\text{Pi/Ci*}}\,\dot{\mathbf{e}}_i)^T\,\mathbf{G}_{\text{R}}^{\text{cj}}\,\breve{\mathbf{p}}_{\text{cj}}^{\text{Pj/Cj*}}\,\dot{\mathbf{e}}_j \tag{2.44}$$

$$\mathbf{Q}_{\text{d1t}} = -2({}^{\text{R}}\dot{\mathbf{A}}^{\text{ci}}\,\mathbf{u}_{\text{i}})^T\,\dot{\mathbf{r}} - ({}^{\text{R}}\mathbf{A}^{\text{ci}}\,\mathbf{u}_{\text{i}})^T\,\dot{\mathbf{G}}_{\text{R}}^{\text{ci}}\,\breve{\mathbf{p}}_{\text{ci}}^{\text{Pi/Ci*}}\,\dot{\mathbf{e}}_i + ({}^{\text{R}}\mathbf{A}^{\text{ci}}\,\mathbf{u}_{\text{i}})^T\,\dot{\mathbf{G}}_{\text{R}}^{\text{cj}}\,\breve{\mathbf{p}}_{\text{cj}}^{\text{Pj/Cj*}}\,\dot{\mathbf{e}}_j - \mathbf{r}^T\,\dot{\mathbf{G}}_{\text{R}}^{\text{cj}}\,\breve{\mathbf{u}}_{\text{i}}\,\dot{\mathbf{e}}_i \tag{2.45}$$

[3]Note que as equações de restrição definidas caracterizam-se como esclereonômicas. Juntas reonômicas serão obtidas com a introdução de movimentos prescritos, a serem apresentados posteriormente no texto.

#### 2.1.3.2 Equações representando Movimentos Prescritos

Além das juntas mecânicas, os movimentos prescritos também são modelados por equações de restrição. Esses movimentos prescritos podem caracterizar rotações ou translações. Embora essas rotações e/ou translações possam envolver diretamente os corpos e suas coordenadas, através da especificação do movimento de pontos do corpo, caso comum na robótica (SHABANA, 2001), onde, por exemplo, especifica-se diretamente um movimento para a garra de um robô do tipo *pick and place* e determina-se os *inputs* nas juntas visando obter-se o resultado desejado, o mais comum para um mecanismo é introduzir esses movimentos prescritos nas juntas, especificando-se os movimentos relativos habilitados pela junta como uma função do tempo, como, por exemplo, a translação para uma junta prismática ou a rotação no caso de uma junta revoluta. Antes de definirmos as equações de movimento prescrito para as juntas, vamos apresentá-las quando estas especificam diretamente o movimento de um ponto do corpo como uma função do tempo. Seja então um dado corpo $C_i$ e um ponto arbitrário $P_i$, fixo no corpo. Prescreve-se o movimento do ponto $P_i$ como uma função do tempo da seguinte forma

$$\mathbf{p}_{\mathrm{R}}^{\mathrm{Ci^*/O}} + {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\mathbf{p}_{\mathrm{ci}}^{\mathrm{Pi/Ci^*}} - \mathbf{f}(t) = 0, \tag{2.46}$$

onde $\mathbf{f}(t) \in R^3$ é uma função vetorial do tempo.

No caso da especificação dos movimentos relativos habilitados pela junta, utilizam-se as equações de restrição básicas apresentadas anteriormente. Contudo, neste caso, ao invés de igualarmos essas equações a zero, fazemos com que elas assumam os valores especificados pelo usuário. Dessa forma, utilizando-se a Eq. (2.31), podemos caracterizar

uma rotação relativa prescrita como abaixo

$$\begin{aligned} \mathbf{\Phi}_{\text{1rp}} &= f(t) \\ \mathbf{v}_{\text{R}}^{\text{i}} \cdot \mathbf{v}_{\text{R}}^{\text{j}} &= \cos(\theta(t)), \end{aligned} \tag{2.47}$$

onde $\theta(t)$ descreve o ângulo entre os vetores $\mathbf{v}_{\text{R}}^{\text{i}}$ e $\mathbf{v}_{\text{R}}^{\text{j}}$ em função do tempo. Embora a Eq. (2.47) esteja correta no sentido de possuir como solução vetores com o ângulo especificado entre si, a equação mostrada também possui outras soluções decorrentes do fato da função cosseno ser par, ou seja, $\cos(\theta) = \cos(-\theta)$. Visando à eliminação desta ambigüidade, deve-se definir três vetores a saber: $\mathbf{v}_{\text{R}}^{\text{1i}}$, $\mathbf{v}_{\text{R}}^{\text{2i}}$, perpendiculares e unitários, definidos no corpo $i$ e o vetor também unitário $\mathbf{v}_{\text{R}}^{\text{j}}$, definido no corpo $j$, de forma que $\mathbf{v}_{\text{R}}^{\text{1i}}$ possua um ângulo $\theta$ com o vetor $\mathbf{v}_{\text{R}}^{\text{j}}$ (ver Fig(2.3)). Assim, a equação especificando a rotação relativa permitida por uma junta fica da seguinte forma

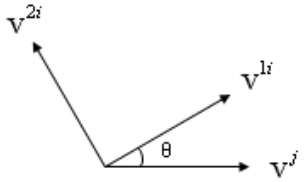



FIGURA 2.3 – Eliminando ambigüidades na rotação relativa.

$$\begin{aligned} \mathbf{v}_{\text{R}}^{\text{1i}} \cdot \mathbf{v}_{\text{R}}^{\text{j}} + \mathbf{v}_{\text{R}}^{\text{2i}} \cdot \mathbf{v}_{\text{R}}^{\text{j}} + \text{sen}\,(\theta(t)) - \cos(\theta(t)) &= 0 \\ (\mathbf{v}_{\text{R}}^{\text{1i}} + \mathbf{v}_{\text{R}}^{\text{2i}}) \cdot \mathbf{v}_{\text{R}}^{\text{j}} + \text{sen}\,(\theta(t)) - \cos(\theta(t)) &= 0. \end{aligned} \tag{2.48}$$

Para a translação relativa entre corpos, basta utilizar a Eq. (2.32), incluindo um termo representando a função temporal responsável pelo fornecimento da distância desejada entre os pontos.

### 2.1.4 Análise Cinemática de Sistemas Multicorpos

A principal razão de termos introduzido as equações de restrição é facilitar a formulação computacional das equações cinemáticas e dinâmicas. Considerando um dado sistema multicorpos com $n$ corpos e $l$ juntas, temos que serão utilizadas $7n$ coordenadas para descrever a configuração espacial do sistema[4]. Supondo que as juntas e movimentos prescritos sejam representados por $m$ equações de restrição independentes, podemos definir o vetor restrição do sistema, $\mathbf{\Phi}(\mathbf{q}, t)$[5] da seguinte forma

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_1 \\ \vdots \\ \mathbf{\Phi}_m \end{bmatrix}, \tag{2.49}$$

onde $\mathbf{q}^T = [\mathbf{q}_1^T \cdots \mathbf{q}_n^T]$ é o vetor de coordenadas de todo sistema e cada uma das equações $\mathbf{\Phi}_i$, $i = 1 \ldots m$ depende de $\mathbf{q}$ e do tempo.

A partir do jacobiano para as juntas[6], podemos obter o jacobiano para o vetor restrição

[4] Note que os parâmetros de Euler estão sendo empregados.

[5] O vetor restrição depende do tempo em virtude dos movimentos prescritos, definidos com o auxílio de funções temporais.

[6] O jacobiano para os movimentos prescritos definidos na seção anterior não foi apresentado. Contudo, este pode ser obtido diretamente das equações obtidas para as equações de restrição.

como mostrado abaixo

$$\mathbf{\Phi}_{\mathrm{q}} = \begin{bmatrix} \frac{\partial \mathbf{\Phi}_1}{\partial q_1} & \cdots & \frac{\partial \mathbf{\Phi}_1}{\partial q_{7n}} \\ \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{\Phi}_{\mathrm{m}}}{\partial q_1} & \cdots & \frac{\partial \mathbf{\Phi}_{\mathrm{m}}}{\partial q_{7n}} \end{bmatrix}. \tag{2.50}$$

O jacobiano do vetor restrição, dado pela Eq. (2.50), permite a obtenção do número de graus de liberdade do sistema. De fato, o número de graus de liberdade é obtido a partir do jacobiano das equações das juntas e dos parâmetros de Euler, sem que as equações de movimento prescrito, caso existam, estejam presentes. Assumindo que temos $r$ equações independentes de restrição devido as juntas e aos parâmetros de rotação e considerando um deslocamento virtual nas coordenadas, podemos escrever

$$\mathbf{\Phi} = \mathbf{0}_{\mathrm{rx1}} \Rightarrow \mathbf{\Phi}_{\mathrm{q}} \delta \mathbf{q} = \mathbf{0}_{\mathrm{rx1}}, \tag{2.51}$$

onde $\mathbf{0}_{\mathrm{rx1}}$ é o vetor zero com $r$ componentes.

Observando-se a equação acima vê-se que, caso o posto do jacobiano seja igual ao número de coordenadas utilizadas na descrição do sistema, então a *única* solução do sistema será $\delta \mathbf{q} = \mathbf{0}_{\mathrm{rx1}}$, onde, nesta situação peculiar, $r = 7n$, e não há mobilidade, sendo o sistema multicorpos uma estrutura. Caso o posto do jacobiano seja inferior ao número de coordenadas, então o espaço nulo, denominado $\mathcal{N}$, do jacobiano terá dimensão $7n - r > 0$[7]. Temos que $\forall \mathbf{u} \in \mathcal{N} \Rightarrow \mathbf{\Phi}_{\mathrm{q}} \mathbf{u} = \mathbf{0}_{\mathrm{rx1}}$. Assim, a solução do sistema apresentado na Eq. (2.51) pode ser escrita na forma (STRANG, 1988)

$$\delta \mathbf{q} = \sum_{i=1}^{7n-r} \delta q_i^* \mathbf{t}_i, \tag{2.52}$$

[7]Note que $\mathcal{N} \subset R^{7n}$, embora $dim(\mathcal{N}) = 7n - r$

onde $\forall i \in \{1, \cdots, 7n-r\} \;\; \exists j \in \{1, \cdots, 7n\} \,|\, \delta q_i^* = \delta q_j$ e os vetores $\mathbf{t}_i$, $i = 1, \cdots, 7n - r$ formam uma base do espaço nulo $\mathcal{N}$. Dessa forma, podemos escrever a solução da Eq. (2.51) em função dos deslocamentos $\delta q_i^*$. Em (STRANG, 1988), os $\delta q^*$'s são chamados de *variáveis livres*, sendo os outros deslocamentos restantes denominados de *variáveis básicas*. O termo *livre* adotado na referência citada indica que essas variáveis podem ser escolhidas arbitrariamente, ou melhor, que quaisquer que sejam seus valores, a solução dada na Eq. (2.52) resolve o sistema apresentado na Eq. (2.51), ou seja, o vetor $\delta\mathbf{q}$ obtido será ortogonal ao espaço linha do jacobiano. No contexto de sistemas mecânicos essas variáveis (deslocamentos virtuais) livres são os deslocamentos virtuais dos graus de liberdade do sistema multicorpos. Do exposto, temos que o número de graus de liberdade é igual à dimensão do espaço nulo $\mathcal{N}$.

Para ilustrar as afirmações realizadas no parágrafo anterior vamos apresentar um exemplo que permitirá a visualização geométrica dos espaços linha e coluna do jacobiano, bem como da superfície definida pelas equações de restrição. Seja a esfera $P$, conectada ao ponto $O$ através de um fio leve de comprimento $r$. Supondo que o ponto $O$ esteja fixo num referencial inercial e que a esfera seja de pequenas dimensões, podemos adotar o modelo de partícula[8] para a esfera (ver Fig.(2.4).

O sistema apresentado na Fig. (2.4) possui dois graus de liberdade, estando a partícula $P$ restringida a mover-se numa esfera de raio igual a $r$. Embora fosse mais fácil descrevermos a configuração espacial da partícula utilizando as coordenadas $\theta$ e $\phi$, mostradas na figura, vamos utilizar as coordenadas cartesianas da partícula $P$ para descrevermos a configuração desta. Assim, existirá uma equação de restrição relacionando essas co-

[8]Cabe ressaltar que os métodos apresentados neste texto permitem a simulação de corpos rígidos, sem adotar o modelo de partícula. Neste exemplo este modelo é utilizado, intencionalmente, visto que a simplicidade do modelo de partícula permite que concentremos nossa atenção em objetivos específicos.

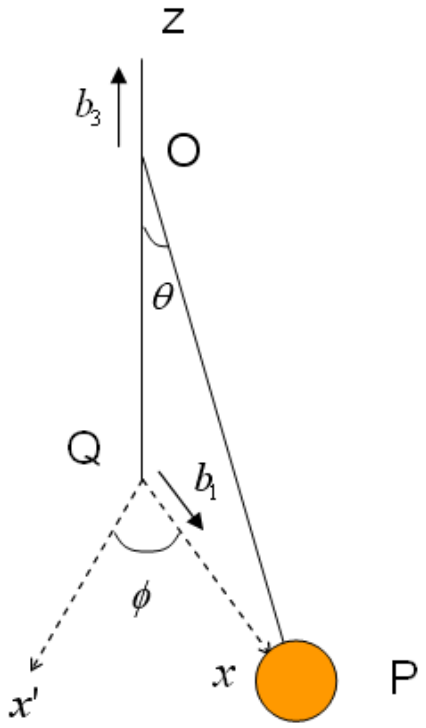



FIGURA 2.4 – Sistema com dois graus de liberdade.

ordenadas, visto que estamos utilizando um conjunto dependente de coordenadas para descrevermos a configuração espacial da partícula. Esta restrição é dada abaixo

$$x^2 + y^2 + z^2 = r^2, \tag{2.53}$$

sendo a equação da superfície de uma esfera de centro no ponto $O$ e raio $r$. O jacobiano da Eq. (2.53) é dado abaixo

$$\Phi_{xyz}(x, y, z) = [2x \quad 2y \quad 2z]. \tag{2.54}$$

Nota-se que, dado $x$, $y$ e $z$, temos que o vetor $[2x \quad 2y \quad 2z]$ é perpendicular ao plano tangente à superfície da esfera, definida pela Eq. (2.53), no ponto ($x$, $y$, $z$) (ver Fig. (2.5)), onde este vetor está mostrado em vermelho. Cabe notar que o plano tangente é o espaço

nulo[9] do jacobiano, e os vetores em amarelo são uma base deste espaço.

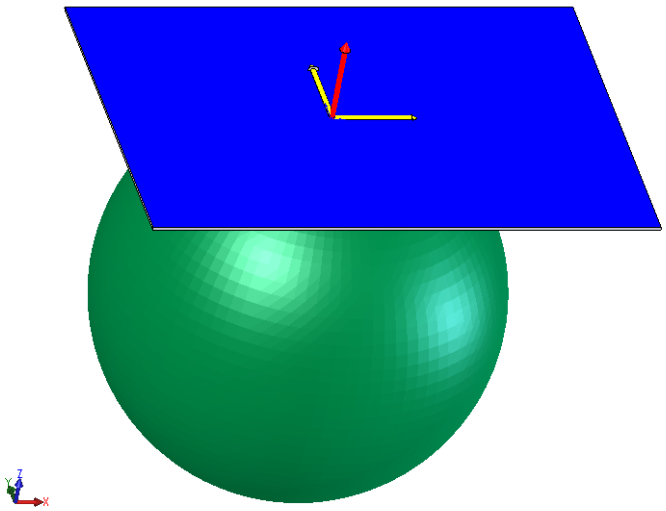


FIGURA 2.5 – Visualização Geométrica

Embora esta seção aborde a cinemática de multicorpos, podemos fazer uma observação referente à dinâmica do exemplo apresentado anteriormente. No exemplo em estudo, o vetor perpendicular ao plano tangente, que forma a base do espaço linha do jacobiano, indica a direção da força responsável à manutenção da restrição entre as coordenadas, apresentada na Eq. (2.53). De fato, esta força deve ser ortogonal ao plano tangente, podendo ser escrita da forma $\mathbf{F} = \Phi_{xyz}^T \lambda$, confirmando que a força exercida pelo fio é na direção do fio. No caso geral, de $n$ dimensões, essas forças de vínculo deverão estar no espaço linha do jacobiano, sendo por conseguinte perpendiculares ao espaço nulo[10], e os movimentos livres, referentes aos graus de liberdade, ocorrerão no espaço nulo. Retornaremos a este exemplo mais adiante, no próximo capítulo, quando formos tratar da estabilização de vínculos na análise dinâmica.

Caso o número de equações independentes de movimento prescrito seja igual ao número

[9]Observe que estamos considerando, de forma implícita, a existência de um sistema coordenado com origem no plano tangente, visando podermos considerar tal plano um espaço vetorial.

[10]Note que o espaço nulo é o complemento ortogonal do espaço linha.

de graus de liberdade do sistema multicorpos, o problema será cinemático, significando que somente equações cinemáticas, não envolvendo forças externas nem de inércia, serão necessárias para conhecermos o movimento completo do sistema. Caso contrário, teremos a situação mais geral, contemplando a dinâmica, que será estudada posteriormente ainda neste capítulo.

Supondo que tenhamos o caso cinemático, vamos obter as equações necessárias à análise do movimento. Neste caso, o jacobiano das equações de restrição, envolvendo agora tanto as juntas e os parâmetros de Euler quanto os movimentos prescritos será uma matriz quadrada. Assim, assumindo como conhecidas a posição, velocidade e aceleração num dado tempo $t$, para obtermos esses valores para um tempo $t + \Delta t$ devemos realizar três análises, a saber: a de posição, a de velocidade e, finalmente, a de aceleração. Para análise de posição devemos resolver um sistema de equações algébricas não-lineares, apresentado na Eq. (2.49), utilizando o método de Newton-Raphson. Para análise de velocidade e aceleração, basta derivar a Eq. (2.49) em relação ao tempo uma e duas vezes, respectivamente. Procedendo essa diferenciação, obtemos

$$(\mathbf{\Phi})_{\mathbf{q}}\dot{\mathbf{q}} = -\frac{\partial \mathbf{\Phi}}{\partial t}, \tag{2.55}$$

para velocidade e

$$(\mathbf{\Phi})_{\mathbf{q}}\ddot{\mathbf{q}} = -(\mathbf{\Phi}_{\mathrm{q}}\dot{\mathbf{q}})_{\mathbf{q}}\dot{\mathbf{q}} - 2\left(\frac{\partial \mathbf{\Phi}}{\partial t}\right)_{\mathbf{q}}\dot{\mathbf{q}} - \frac{\partial^2 \mathbf{\Phi}}{\partial t^2}, \tag{2.56}$$

para aceleração. É importante notar que, no caso das juntas mecânicas, o termo do lado direito das Eqs. (2.55, 2.56) são, respectivamente, iguais a zero e ao vetor $\mathbf{Q}_{\mathrm{d}}$ (ver Eqs. (2.43, 2.44, 2.45)). No caso de movimentos prescritos existirão termos adicionais

referentes à explícita dependência temporal dessas equações.

Outra observação importante, do ponto de vista computacional, é que a melhor forma de resolvermos as análises de posição, velocidade e aceleraçao é utilizando-se a fatoração **LU** no jacobiano, visto que, dessa forma, quando a iteração Newton-Raphson tiver satisfeito a tolerância requisitada, as análises de velocidade e aceleração serão realizadas por mera substituição, eliminando a necessidade de fatorações, o que aumentará a eficiência da solução numérica.

## 2.2 Análise Dinâmica

Nesta seção, abordaremos os aspectos elementares da dinâmica computacional de sistemas multicorpos. O intuito aqui é introduzir o conhecimento básico necessário no próximo capítulo, onde serão estudados métodos específicos visando à solução numérica das equações dinâmicas do movimento.

### 2.2.1 Análise de Forças

A principal distinção entre as análises cinemática e dinâmica reside na análise de forças. As forças que agem sobre um sistema podem ser classificadas em forças de inércia, externas e de vínculo[11].

[11] As forças de vínculo serão abordadas na próxima seção, tratando das equações de movimento

#### 2.2.1.1 Força Generalizada de Inércia

A força generalizada de inércia será obtida com o uso das Eqs. (2.28, 2.29), apresentadas anteriormente. Observe-se que

$$\int_V \rho \mathbf{p}_{\rm ci}^{\rm P/Ci^*}\, dV = 0, \tag{2.57}$$

$$\int_V \rho \tilde{\mathbf{p}}_{\rm ci}^{\rm P/Ci^*}\, dV = 0, \tag{2.58}$$

$$\int_V \rho (\tilde{\mathbf{p}}_{\rm ci}^{\rm P/Ci^*})^T\, \tilde{\mathbf{p}}_{\rm ci}^{\rm P/Ci^*}\, dV = \mathbf{I}_{\rm ci}^{\rm Ci/Ci^*}, \tag{2.59}$$

onde $\mathbf{I}_{\rm ci}^{\rm Ci/Ci^*}$ é o tensor de inércia do corpo $Ci$ com respeito ao seu centro de massa $Ci^*$, escrito no referencial do corpo. Para simplificar a notação, escreveremos este tensor da forma $\mathbf{I}^*$. Cabe notar que, neste caso, a matriz representativa do tensor de inércia é constante.

A força de inércia generalizada é dada abaixo (ver Fig. (2.6))

$$\delta \mathbf{W}^{\rm Ci} = \int_V \rho\, {}^{\rm R}\mathbf{a}_{\rm R}^{\rm P} \cdot \delta \mathbf{p}_{\rm R}^{\rm P/O}\, dV, \tag{2.60}$$

onde os termos $\delta \mathbf{p}_{\rm R}^{\rm P/O}$ e ${}^{\rm R}\mathbf{a}_{\rm R}^{\rm P}$ são dados pelas Eqs. (2.28, 2.29) respectivamente. Vamos obter a força de inércia por partes, da forma mostrada abaixo.

$$\delta \mathbf{W}^{\rm Ci} = \mathbf{W}_{\rm I}^{\rm Ci} + \mathbf{W}_{\rm II}^{\rm Ci} + \mathbf{W}_{\rm III}^{\rm Ci}, \tag{2.61}$$

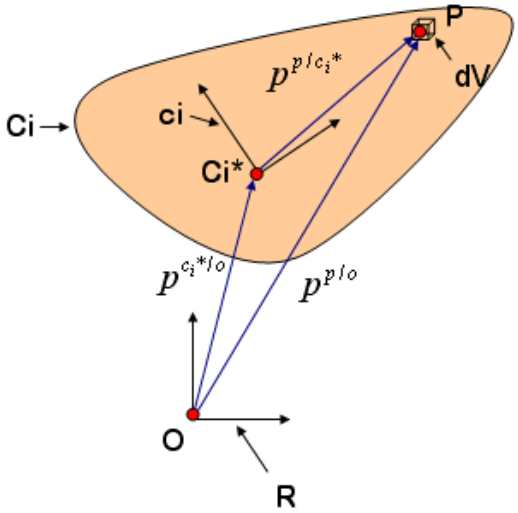



FIGURA 2.6 – Obtendo a força generalizada de inércia.

onde

$$\mathbf{W}_{\mathrm{I}}^{\mathrm{Ci}} = \int_V \rho\, {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} \cdot \delta \mathbf{p}_{\mathrm{R}}^{\mathrm{P/O}}\, dV, \tag{2.62}$$

$$\mathbf{W}_{\mathrm{II}}^{\mathrm{Ci}} = \int_V \rho({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2 \mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}) \cdot \delta \mathbf{p}_{\mathrm{R}}^{\mathrm{P/O}}\, dV, \tag{2.63}$$

$$\mathbf{W}_{\mathrm{III}}^{\mathrm{Ci}} = -\int_V \rho({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, \tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\, \mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}} \ddot{\mathbf{e}}) \cdot \delta \mathbf{p}_{\mathrm{R}}^{\mathrm{P/O}}\, dV. \tag{2.64}$$

A contribuição do termo $\mathbf{W}_{\mathrm{I}}^{\mathrm{Ci}}$ na força generalizada de inércia é da forma

$$\begin{aligned}
\mathbf{W}_{\mathrm{I}}^{\mathrm{Ci}} &= \int_V \rho\, {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} \cdot \delta \mathbf{p}_{\mathrm{R}}^{\mathrm{P/O}}\, dV \\
&= \int_V \rho\, {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} \cdot \delta \mathbf{p}_{\mathrm{R}}^{\mathrm{Ci*/O}}\, dV - \int_V \rho\, {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} \cdot ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\, \tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\, \mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}} \delta \mathbf{e})\, dV \\
&= {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} \cdot \delta \mathbf{p}_{\mathrm{R}}^{\mathrm{Ci*/O}} \int_V \rho\, dV - {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} \cdot ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}} \int_V \rho \tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\, dV\, \mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}} \delta \mathbf{e}) \\
&= m\, {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} \cdot \delta \mathbf{p}_{\mathrm{R}}^{\mathrm{Ci*/O}} + \mathbf{0} \\
&= m\, {}^{\mathrm{R}}\mathbf{a}_{\mathrm{R}}^{\mathrm{P}} \cdot \delta \mathbf{p}_{\mathrm{R}}^{\mathrm{Ci*/O}}.
\end{aligned} \tag{2.65}$$

O termo $\mathbf{W}_{\mathrm{II}}^{\mathrm{Ci}}$ contribuirá com a força generalizada de inércia, fornecendo termos

quadráticos na velocidade, da forma apresentada abaixo.

$$
\begin{aligned}
\mathbf{W}_{\mathrm{II}}^{\mathrm{Ci}} &= \int_V \rho({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}) \cdot \delta\mathbf{p}_{\mathrm{R}}^{\mathrm{P/O}}\, dV \\
&= \int_V \rho({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}) \cdot \delta\mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci*}}\, dV - \int_V \rho({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}) \cdot ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\,\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}}\delta\mathbf{e})\, dV \\
&= -({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2 \int_V \rho\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\, dV) \cdot \delta\mathbf{p}_{\mathrm{R}}^{\mathrm{P/Ci*}} + \int_V \rho({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}) \cdot ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\,\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}}\delta\mathbf{e})\, dV \\
&= -\int_V \rho({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}) \cdot ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\,\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}}\delta\mathbf{e})\, dV \\
&= -\int_V \rho({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\,\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}}\delta\mathbf{e}) \cdot ({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}})\, dV \\
&= -\int_V \rho(\delta\mathbf{e}^T\,(\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}})^T\,(\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}})^T({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}})^T{}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}}({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\, dV \\
&= -\int_V \rho(\delta\mathbf{e}^T\,(\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}})^T\,(\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}})^T({}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}})^2\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\, dV \\
&= -\int_V \rho(\delta\mathbf{e}^T\,(\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}})^T\,(\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}})^T\,{}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}}\,{}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}}\,\mathbf{p}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\, dV \\
&= -\int_V \rho(\delta\mathbf{e}^T\,(\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}})^T\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\,{}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}}\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\,{}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{ci}}^{\mathrm{Ci}}\, dV \qquad (2.66) \\
&= -\int_V \rho(\delta\mathbf{e}^T\,(\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}})^T\,{}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}}\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\,{}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{ci}}^{\mathrm{Ci}}\, dV \qquad (2.67) \\
&= -\delta\mathbf{e}^T\,(\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}})^T\,{}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}} \int_V \rho\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\, dV\,{}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{ci}}^{\mathrm{Ci}} \\
&= \delta\mathbf{e}^T\,(\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}})^T\,{}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}} \int_V \rho(\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}})^T\,\tilde{\mathbf{p}}_{\mathrm{ci}}^{\mathrm{P/Ci*}}\, dV\,{}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{ci}}^{\mathrm{Ci}} \\
&= \delta\mathbf{e}^T\,(\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}})^T\,{}^{\mathrm{R}}\tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}}\,\mathbf{I}^*\,{}^{\mathrm{R}}\boldsymbol{\omega}_{\mathrm{ci}}^{\mathrm{Ci}}. \qquad (2.68)
\end{aligned}
$$

O termo $\mathbf{W}_{\mathrm{II}}^{\mathrm{Ci}}$, mostrado acima, foi obtido utilizando-se algumas simplificações dignas de nota. A primeira delas foi introduzida para obtermos a Eq. (2.66), onde foi utilizado o fato de $\tilde{\mathbf{a}}\mathbf{b} = -\tilde{\mathbf{b}}\mathbf{a}$ e $(\tilde{\mathbf{a}})^T = -\tilde{\mathbf{a}}$. A segunda e última simplificação, utilizada na obtenção da Eq. (2.67) é mais sútil, sendo obtida a partir da consideração que $\tilde{\mathbf{a}}\tilde{\mathbf{b}} = \tilde{\mathbf{b}}\tilde{\mathbf{a}} + \mathbf{b}(\mathbf{a})^T - \mathbf{a}(\mathbf{b})^T$.

O termo $\mathbf{W}_{\text{III}}^{\text{Ci}}$ contribuirá para a força generalizada de inércia da seguinte forma

$$
\begin{aligned}
\mathbf{W}_{\text{III}}^{\text{Ci}} &= -\int_V \rho({}^{\text{R}}\mathbf{A}^{\text{ci}}\,\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\ddot{\mathbf{e}})\cdot\delta\mathbf{p}_{\text{R}}^{\text{P/O}}\,dV \\
&= -\int_V \rho({}^{\text{R}}\mathbf{A}^{\text{ci}}\,\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\ddot{\mathbf{e}})\cdot\delta\mathbf{p}_{\text{R}}^{\text{P/Ci*}}\,dV + \int_V \rho({}^{\text{R}}\mathbf{A}^{\text{ci}}\,\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\ddot{\mathbf{e}})\cdot({}^{\text{R}}\mathbf{A}^{\text{ci}}\,\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\delta\mathbf{e})\,dV \\
&= -({}^{\text{R}}\mathbf{A}^{\text{ci}}\int_V \rho\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,dV\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\ddot{\mathbf{e}})\cdot\delta\mathbf{p}_{\text{R}}^{\text{P/Ci*}}\int_V \rho({}^{\text{R}}\mathbf{A}^{\text{ci}}\,\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\ddot{\mathbf{e}})\cdot({}^{\text{R}}\mathbf{A}^{\text{ci}}\,\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\delta\mathbf{e})\,dV \\
&= \int_V \rho\ddot{\mathbf{e}}^T\,(\mathbf{G}_{\text{ci}}^{\text{Ci}})^T\,(\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}})^T({}^{\text{R}}\mathbf{A}^{\text{ci}})^T{}^{\text{R}}\mathbf{A}^{\text{ci}}\,\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\delta\mathbf{e}\,dV \\
&= \int_V \rho\ddot{\mathbf{e}}^T\,(\mathbf{G}_{\text{ci}}^{\text{Ci}})^T\,(\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}})^T\,\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\delta\mathbf{e}\,dV \\
&= \ddot{\mathbf{e}}^T\,(\mathbf{G}_{\text{ci}}^{\text{Ci}})^T\int_V \rho(\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}})^T\,\tilde{\mathbf{p}}_{\text{ci}}^{\text{P/Ci*}}\,dV\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\delta\mathbf{e} \\
&= \ddot{\mathbf{e}}^T\,(\mathbf{G}_{\text{ci}}^{\text{Ci}})^T\,\mathbf{I}^*\,\mathbf{G}_{\text{ci}}^{\text{Ci}}\delta\mathbf{e}.
\end{aligned}
\tag{2.69}
$$

A partir dos termos $\delta\mathbf{W}_{\text{I}}^{\text{Ci}}$, $\delta\mathbf{W}_{\text{II}}^{\text{Ci}}$ e $\delta\mathbf{W}_{\text{III}}^{\text{Ci}}$, dados pelas Eqs. (2.65, 2.68 e 2.69) respectivamente, podemos escrever a força generalizada de inércia como abaixo.

$$
\begin{aligned}
\delta\mathbf{W}^{\text{Ci}} &= \delta\mathbf{W}_{\text{I}}^{\text{Ci}} + \delta\mathbf{W}_{\text{II}}^{\text{Ci}} + \delta\mathbf{W}_{\text{III}}^{\text{Ci}} \\
&= \delta\mathbf{p}_{\text{R}}^{\text{Ci*/O}}m^{\text{R}}\mathbf{a}_{\text{R}}^{\text{Ci*}} + \delta\mathbf{e}^T(\mathbf{G}_{\text{ci}}^{\text{Ci}})^T\mathbf{I}^*\mathbf{G}_{\text{ci}}^{\text{Ci}}\ddot{\mathbf{e}} + \delta\mathbf{e}^T(\mathbf{G}_{\text{ci}}^{\text{Ci}})^{T\,\text{R}}\tilde{\boldsymbol{\omega}}_{\text{ci}}^{\text{Ci}}\mathbf{I}^{*\,\text{R}}\boldsymbol{\omega}_{\text{ci}}^{\text{Ci}} \\
&= \delta\mathbf{q}_i^T\begin{bmatrix} m\mathbf{I}_{3\text{x}3} & \mathbf{0} \\ \mathbf{0} & (\mathbf{G}_{\text{ci}}^{\text{Ci}})^T\,\mathbf{I}^*\,\mathbf{G}_{\text{ci}}^{\text{Ci}} \end{bmatrix}\ddot{\mathbf{q}} + \delta\mathbf{q}_i^T\begin{bmatrix} \mathbf{0} \\ (\mathbf{G}_{\text{ci}}^{\text{Ci}})^{T\,\text{R}}\tilde{\boldsymbol{\omega}}_{\text{ci}}^{\text{Ci}}\mathbf{I}^{*\,\text{R}}\boldsymbol{\omega}_{\text{ci}}^{\text{Ci}} \end{bmatrix} \\
&= \delta\mathbf{q}_i^T\mathbf{M}^{\text{i}}\ddot{\mathbf{q}}_i + \delta\mathbf{q}_i^T\mathbf{Q}_{\text{v}}^{\text{i}},
\end{aligned}
\tag{2.70}
$$

onde

$$
\mathbf{M}^{\text{i}} = \begin{bmatrix} m\mathbf{I}_{3\text{x}3} & \mathbf{0} \\ \mathbf{0} & (\mathbf{G}_{\text{ci}}^{\text{Ci}})^T\,\mathbf{I}^*\,\mathbf{G}_{\text{ci}}^{\text{Ci}} \end{bmatrix}, \tag{2.71}
$$

é a matriz de massa do corpo $C_i$ e

$$\mathbf{Q}_{\mathrm{v}}^{\mathrm{i}} = \begin{bmatrix} \mathbf{0} \\ (\mathbf{G}_{\mathrm{ci}}^{\mathrm{Ci}})^{T\mathrm{R}} \tilde{\boldsymbol{\omega}}_{\mathrm{ci}}^{\mathrm{Ci}} \mathbf{I}^{*\mathrm{R}} \boldsymbol{\omega}_{\mathrm{ci}}^{\mathrm{Ci}} \end{bmatrix}, \tag{2.72}$$

é o vetor que representa os termos da força de inércia quadráticos na velocidade. Observe que, como estamos utilizando o centro de massa como origem do sistema coordenado afixado ao corpo, não há acoplamento entre a translação e a rotação, isto é, o corpo pode somente transladar sem que haja qualquer força de inércia agindo no sentido de provocar alguma rotação no corpo.

#### 2.2.1.2 Forças Generalizadas Externas

As forças generalizadas externas representam molas, amortecedores, atuadores, forças de atrito, etc. As fórmulas tanto para o conjunto mola-amortecedor quanto para modelagem de atrito não serão apresentadas. As fórmulas referentes ao conjunto mola-amortecedor não serão apresentadas devido ao fato dessas fórmulas estarem amplamente apresentadas nos textos de mecânica computacional (SHABANA, 2001), (NIKRAVESH, 1988), (JALóN; BAYO, 1988). No caso do atrito, essas fórmulas não serão apresentadas já que o atrito é um fenômeno complicado, tomando, na análise computacional, um grau ainda maior de complexidade, já que, neste caso, entram em cena algoritmos de computação gráfica muito mais sofisticados, visando o tratamento matemático adequado deste fenômeno.

Supondo que um atuador exerça uma força $\mathbf{F}$ no ponto $P$ do corpo e utilizando a Eq. (2.28), podemos representar a força generalizada associada à força $\mathbf{F}$ da seguinte

forma

$$\begin{aligned}
\delta \mathbf{W}^{\mathrm{Ci}}_{\mathrm{Fext}} &= \mathbf{F} \cdot \delta \mathbf{p}^{\mathrm{P/O}}_{\mathrm{R}} \\
&= \mathbf{F}^T \delta \mathbf{p}^{\mathrm{P/O}}_{\mathrm{R}} \\
&= \begin{bmatrix} \mathbf{F}^T & -\mathbf{F}^T {}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}} \tilde{\mathbf{p}}^{\mathrm{P/Ci*}}_{\mathrm{ci}} \mathbf{G}^{\mathrm{Ci}}_{\mathrm{ci}} \end{bmatrix} \delta \mathbf{q}_i \\
&= \delta \mathbf{q}_i^T \begin{bmatrix} \mathbf{F} \\ -({}^{\mathrm{R}}\mathbf{A}^{\mathrm{ci}} \tilde{\mathbf{p}}^{\mathrm{P/Ci*}}_{\mathrm{ci}} \mathbf{G}^{\mathrm{Ci}}_{\mathrm{ci}})^T \mathbf{F} \end{bmatrix}.
\end{aligned} \tag{2.73}$$

Observando-se a Eq. (2.73), podemos representar a força generalizada externa devido à gravidade da seguinte forma

$$\delta \mathbf{W}^{\mathrm{Ci}}_{\mathrm{g}} = \delta \mathbf{q}_i^T \begin{bmatrix} \mathbf{F} \\ \mathbf{0}_{3\mathrm{x}1} \end{bmatrix}, \tag{2.74}$$

onde $\mathbf{F} = mg\mathbf{n}$, sendo $m$ massa do corpo, $g$ o módulo da aceleração da gravidade e $\mathbf{n}$ o vetor unitário no sentido para o qual a força gravitacional atua.

### 2.2.2 Equações de Movimento

Tendo obtido as expressões para as forças generalizadas de inércia e externas, podemos formular as equações de movimento do sistema multicorpos. Utilizaremos o princípio do trabalho virtual na dinâmica para obtermos essas equações. Este princípio estabelece que

$$\delta \mathbf{W}^{\mathrm{Ci}} = \delta \mathbf{W}^{\mathrm{Ci}}_{\mathrm{Fext}} + \delta \mathbf{W}^{\mathrm{Ci}}_{\mathrm{Rest}}, \tag{2.75}$$

onde $\delta\mathbf{W}^{\mathrm{Ci}}$, $\delta\mathbf{W}^{\mathrm{Ci}}_{\mathrm{Fext}}$ e $\delta\mathbf{W}^{\mathrm{Ci}}_{\mathrm{Rest}}$ são respectivamente os trabalhos virtuais das forças de inércia, externas e de restrição, para o corpo $C_i$ do sistema. Representaremos o trabalho virtual das forças de vínculo e externas da seguinte forma

$$\delta\mathbf{W}^{\mathrm{Ci}}_{\mathrm{Rest}} = \delta\mathbf{q}_i^T\mathbf{Q}^{\mathrm{i}}_{\mathrm{Rest}} \tag{2.76}$$

$$\delta\mathbf{W}^{\mathrm{Ci}}_{\mathrm{Fext}} = \delta\mathbf{q}_i^T\mathbf{Q}^{\mathrm{i}}_{\mathrm{Fext}}, \tag{2.77}$$

onde $\mathbf{Q}^{\mathrm{i}}_{\mathrm{Rest}}$ e $\mathbf{Q}^{\mathrm{i}}_{\mathrm{Fext}}$ são, respectivamente, a força generalizada de restrição e a força generalizada externa para o corpo $C_i$.

Substituindo as expressões para o trabalho virtual das forças de inércia, externas e de restrição, Eqs. (2.70, 2.77 e 2.76) respectivamente, e aplicando o princípio do trabalho virtual, obtemos

$$\mathbf{M}^{\mathrm{i}}\ddot{\mathbf{q}}_i = \mathbf{Q}^{\mathrm{i}}_{\mathrm{Fext}} + \mathbf{Q}^{\mathrm{i}}_{\mathrm{Rest}} - \mathbf{Q}^{\mathrm{i}}_{\mathrm{v}}. \tag{2.78}$$

Note-se que, embora os deslocamentos virtuais $\delta\mathbf{q}_i$ sejam dependentes, pudemos igualar a zero o termo que multiplica esses deslocamentos, obtendo a equação mostrada acima, visto que essas restrições estão consideradas explicitamente, através das forças de vínculo. Caso tenhamos $n$ corpos rígidos, as equações de movimento podem ser escritas da forma

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{Q}_{\mathrm{Fext}} + \mathbf{Q}_{\mathrm{Rest}} - \mathbf{Q}_{\mathrm{v}}, \tag{2.79}$$

onde

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}^1 & & \\ & \ddots & \\ & & \mathbf{M}^n \end{bmatrix} \tag{2.80}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}^1 \\ \vdots \\ \mathbf{Q}^n \end{bmatrix}. \tag{2.81}$$

A Eq. (2.81) representa como as forças generalizadas externas, de vínculo e dos termos de inércia quadráticos na velocidade são formadas, justificando a omissão do subscrito na equação citada. A força generalizada de vínculo, $\mathbf{Q}_{\text{Rest}}$, pode ser escrita em função dos multiplicadores de Lagrange, ficando da forma (SHABANA, 2001)

$$\mathbf{Q}_{\text{Rest}} = -(\mathbf{\Phi}_{\text{q}})^T \boldsymbol{\lambda}, \tag{2.82}$$

onde $\boldsymbol{\lambda} = [\lambda_1 \cdots \lambda_m]^T$ são os $m$ multiplicadores de Lagrange, sendo $m$ o número de equações independentes de restrição. Utilizando a Eq. (2.82) podemos escrever a Eq. (2.79) como abaixo

$$\mathbf{M}\ddot{\mathbf{q}} + (\mathbf{\Phi}_{\text{q}})^T \boldsymbol{\lambda} = \mathbf{Q}_{\text{Fext}} - \mathbf{Q}_{\text{v}}. \tag{2.83}$$

Note-se que a Eq. (2.83) possui $n+m$ incógnitas para $n$ equações de movimento. Para possibilitarmos sua solução, devemos introduzir $m$ equações. Adicionando a Eq. (2.56)

obtemos $n+m$ equações. Essas equações podem ser representadas na forma

$$\begin{bmatrix} \mathbf{M} & (\mathbf{\Phi}_{\mathrm{q}})^T \\ \mathbf{\Phi}_{\mathrm{q}} & \mathbf{0}_{\mathrm{mxm}} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{\mathrm{Ext}} - \mathbf{Q}_{\mathrm{v}} \\ \mathbf{Q}_{\mathrm{d}} \end{bmatrix}, \tag{2.84}$$

onde

$$\mathbf{Q}_{\mathrm{d}} = -(\mathbf{\Phi}_{\mathrm{q}}\dot{\mathbf{q}})_{\mathbf{q}}\dot{\mathbf{q}} - 2\left(\frac{\partial \mathbf{\Phi}}{\partial t}\right)_{\mathbf{q}} \dot{\mathbf{q}} - \frac{\partial^2 \mathbf{\Phi}}{\partial t^2}. \tag{2.85}$$

Uma outra forma conveniente de expressarmos as equações de movimento é descartando as forças de vínculo. Neste caso, as equações de movimento ficam da seguinte forma

$$\delta\mathbf{q}^T \left(\mathbf{M}\ddot{\mathbf{q}} - \mathbf{Q}_{\mathrm{Fext}} - \mathbf{Q}_{\mathrm{Rest}} + \mathbf{Q}_{\mathrm{v}}\right) = \mathbf{0}. \tag{2.86}$$

A Eq. (2.86) será utilizada quando apresentarmos o método de partição de coordenadas, assunto do próximo capítulo. Cabe observar que esta equação poderia ser obtida a partir da soma dos trabalhos virtuais para todos os corpos do sistema e da aplicação do terceiro princípio de Newton. Somando-se esses trabalhos virtuais da forma mostrada abaixo

$$\sum_{i=1}^{7n} \delta\mathbf{W}^{\mathrm{Ci}} = \sum_{i=1}^{7n} \delta\mathbf{W}^{\mathrm{Ci}}_{\mathrm{Fext}} + \sum_{i=1}^{7n} \delta\mathbf{W}^{\mathrm{Ci}}_{\mathrm{Rest}}, \tag{2.87}$$

e, aplicando o terceiro princípio de Newton, temos que $\sum_{i=1}^{7n} \delta\mathbf{W}^{\mathrm{Ci}}_{\mathrm{Rest}} = 0$. Assim, obtemos a representação dada pela Eq. (2.86).

# 3 Métodos Numéricos

Na introdução observamos que os métodos computacionais baseados na descrição absoluta evoluíram, basicamente, no sentido da melhora de eficiência da solução das equações de movimento, sem que a forma de obter-se essas equações fosse alterada de maneira significativa, visto que a principal vantagem desses métodos reside exatamente na facilidade de obtermos tais equações. Neste capítulo vamos apresentar algumas formas para tornarmos mais eficiente a solução das equações de movimento. Inicialmente discutiremos o chamado método rígido, consistindo na representação das equações de restrição na sua forma não derivada, dando origem a um sistema diferencial-algébrico. Após introduzirmos o método rígido vamos mostrar o método de estabilização mais conhecido e justificaremos sua importância, bem como apresentaremos um método de estabilização mais sofisticado, baseado na projeção, visando à estabilização dos vínculos discutindo, em seguida, o método de partição de coordenadas. Posteriormente, apresentaremos um método de ortogonalização aplicado ao método de partição de coordenadas, discutindo suas vantagens e desvantagens. No próximo capítulo, apresentaremos uma alternativa ao método de ortogonalização introduzido e mostraremos a solução implementada no programa desenvolvido.

## 3.1 Método Rígido

Quando obtivemos a equação de movimento, Eq. (2.83), no capítulo anterior, ressaltamos a insuficiência no número de equações em relação ao número de variáveis. Naquele caso, adicionamos as equações de restrição para acelerações, isto é, $\ddot{\mathbf{\Phi}}(\mathbf{q}, t) = \mathbf{0}$, visando possibilitar a solução das equações de movimento para as acelerações (e multiplicadores de Lagrange) a partir da solução de um sistema linear. Contudo, existem outras possibilidades que tornam a solução das equações de movimento possível. Uma delas é acrescentar às equações de movimento, obtidas com o uso da Eq. (2.83), as equações de restrição na sua forma não derivada, ou seja, como um conjunto de equações algébricas não-lineares. Embora tenhamos, neste caso, um sistema diferencial-algébrico, de mais difícil solução, garantimos o cumprimento dessas equações de restrição diretamente, já que, como veremos na próxima seção, a Eq. (2.84), utilizando a forma derivada das equações de restrição, não assegura o cumprimento dos vínculos, comprometendo a solução numérica, devido à erros numéricos intrínsecos ao processo de solução, requerendo algoritmos para estabilização dos vínculos. Ao método que adiciona as equações de restrição sem derivá-las, nas equações de movimento, obtidas com o uso da Eq. (2.83), denominaremos de método rígido. O termo rígido, escolhido para denominar o método descrito acima, deve-se ao fato de ocorrer a introdução artificial de rigidez[1] nas equações de movimento. Basicamente, dizer que uma dada equação diferencial(-algébrica) é rigida significa afirmar que seus autovalores apresentam uma grande dispersão. O sistema diferencial-algébrico obtido nesta seção é da forma

$$\begin{cases} \mathbf{M}\ddot{\mathbf{q}} + (\mathbf{\Phi}_{\mathrm{q}})^T \boldsymbol{\lambda} = \mathbf{Q}_{\mathrm{Fext}} - \mathbf{Q}_{\mathrm{v}} \\ \mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0} \end{cases} \tag{3.1}$$

[1] *Stiffness*, em inglês

Fazendo $\mathbf{p} = \dot{\mathbf{q}}$, podemos escrever

$$\begin{cases} \dot{\mathbf{q}} = \mathbf{p} \\ \mathbf{M}\dot{\mathbf{p}} = -(\mathbf{\Phi}_{\mathrm{q}})^T\boldsymbol{\lambda} + \mathbf{Q}_{\mathrm{Fext}} - \mathbf{Q}_{\mathrm{v}} \\ \mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0} \end{cases}, \tag{3.2}$$

transformando as equações diferenciais num sistema de primeira ordem. Observando que, para um sistema de equações diferenciais de primeira ordem do tipo

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}), \tag{3.3}$$

a fórmula genérica para o método de Gear (SHAMPINE, 1994) é da forma

$$\mathbf{y}^{i+1} = \sum_{j=0}^{k} a_j \mathbf{y}_{i-j} + hb\mathbf{f}(t_{i+1}, \mathbf{y}_{i+1}), \tag{3.4}$$

podemos aplicar esta fórmula ao sistema da Eq. (3.2), obtendo

$$\begin{cases} \mathbf{q}^{i+1} = \sum_{j=0}^{k} a_j \mathbf{q}_{i-j} + hb\mathbf{p}_{i+1} \\ \mathbf{M}\mathbf{p}^{i+1} = \sum_{j=0}^{k} a_j \mathbf{p}_{i-j} + hb\left(-(\mathbf{\Phi}_{\mathrm{q}})^T\boldsymbol{\lambda} + \mathbf{Q}_{\mathrm{Fext}} - \mathbf{Q}_{\mathrm{v}}\right) \\ \mathbf{\Phi}(\mathbf{q}, t) = 0 \end{cases} \tag{3.5}$$

Definindo o vetor $\mathbf{z}^T = [\mathbf{q}^T \quad \mathbf{p}^T \quad \boldsymbol{\lambda}^T]$, podemos escrever o conjunto de equações algébricas não lineares, dado pela Eq. (3.5) da seguinte maneira

$$\varphi(\mathbf{z}) = \begin{cases} \mathbf{q} = hb\mathbf{p} + \mathbf{d_1} \\ \mathbf{M}\mathbf{p} = hb\left(-(\mathbf{\Phi}_{\mathrm{q}})^T\boldsymbol{\lambda} + \mathbf{Q}_{\mathrm{Fext}} - \mathbf{Q}_{\mathrm{v}}\right) + \mathbf{d_2} = hb\,\mathbf{s}(\mathbf{q}, \mathbf{p}, \boldsymbol{\lambda}) + \mathbf{d_2} \\ \mathbf{\Phi}(\mathbf{q}, t) = 0 \end{cases}, \tag{3.6}$$

onde $\mathbf{d_1}$ e $\mathbf{d_2}$ são vetores que englobam valores previamente calculados, já conhecidos e $\mathbf{s}(\mathbf{q},\mathbf{p},\boldsymbol{\lambda}) = \left(-(\boldsymbol{\Phi}_{\mathrm{q}})^T\boldsymbol{\lambda} + \mathbf{Q}_{\mathrm{Fext}} - \mathbf{Q}_{\mathrm{v}}\right)$. A Eq. (3.6) deve ser resolvida utilizando-se o algoritmo de Newton-Raphson, da forma mostrada abaixo

$$\begin{array}{c} \varphi_{\mathrm{z}}\Delta\mathbf{z} = -\varphi(\mathbf{z}) \\ \mathbf{z} = \mathbf{z} + \Delta\mathbf{z} \end{array}, \tag{3.7}$$

onde $\varphi_{\mathrm{z}}$, o jacobiano das equações algébricas, é da forma

$$\varphi_{\mathrm{z}} = \begin{bmatrix} \mathbf{I} & hb\mathbf{I} & \mathbf{0} \\ \mathbf{M_q p} + hb\mathbf{s_q} & \mathbf{M} + hb\mathbf{s_p} & -hb\boldsymbol{\Phi}_{\mathrm{q}} \\ \boldsymbol{\Phi}_{\mathrm{q}} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{3.8}$$

Observando-se o jacobiano dado acima, verifica-se que esta matriz ficará cada vez mais mal condicionada conforme formos reduzindo o passo. De fato, pode-se demonstrar que o número de condição para a matriz apresentada acima é de ordem $O(1/h^3)$, onde $h$ é o passo (JALóN; BAYO, 1988). O programa comercial ADAMS implementa, em sua formulação padrão, um método semelhante ao método rígido apresentado, onde as equações de restrição aparecem na sua forma não derivada e as equações diferencias de movimento são obtidas utilizando-se as equações de Lagrange (NEGRUT; HARRIS, 2001). Devido às dificuldades encontradas na solução de sistemas diferenciais-algébricos, o ADAMS possibilita a estabilização do índice dessas equações, significando, na prática, que as equações de restrição serão representadas na forma derivada $\dot{\Phi} = \mathbf{0}$ ou $\ddot{\Phi} = \mathbf{0}$. Neste último caso, obtemos as equações dadas na Eq. (2.84). No caso do ADAMS, essa estabilização do índice eleva a acurácia da solução. Contudo, o custo computacional aumenta até 100% (RAMPALLI, 2000). A estabilização do índice dessas equações diferencias-algébricas requer,

como veremos a seguir, a estabilização de vínculos, assunto da próxima seção.

## 3.2 Método de Estabilização de Vínculos

Quando empregamos a Eq. (2.84) para representarmos matematicamente um sistema multicorpos, estamos assegurando o cumprimento das equações de restrição na sua forma derivada, não garantindo, explicitamente, sua satisfação na forma não derivada. Isso se deve ao fato da equação diferencial $\ddot{\mathbf{\Phi}}(\mathbf{q},t) = \mathbf{\Phi}_{\mathrm{q}}\ddot{\mathbf{q}} - \mathbf{Q}_{\mathrm{d}} = \mathbf{0}$ permitir, como solução, um crescimento exponencial, caracterizando uma situação de instabilidade. No método de estabilização proposto por Baumgarte (BAUMGARTE, 1972), substitui-se a equação $\ddot{\mathbf{\Phi}}(\mathbf{q},t) = \mathbf{0}$ por $\ddot{\mathbf{\Phi}} + 2\alpha\dot{\mathbf{\Phi}} + \beta^2\mathbf{\Phi} = \mathbf{0}$, com $\alpha, \beta \in R$ e $\alpha > 0, \beta \neq 0$. Esta equação pode ser escrita da seguinte forma

$$\mathbf{\Phi}_{\mathrm{q}}\ddot{\mathbf{q}} = \mathbf{Q}_{\mathrm{d}} - 2\alpha(\mathbf{\Phi}_{\mathrm{q}}\dot{\mathbf{q}} - \frac{\partial\mathbf{\Phi}}{\partial t}) - \beta^2\mathbf{\Phi}. \tag{3.9}$$

Embora o método de Baumgarte seja utilizado de forma bem sucedida, inicialmente não havia uma maneira sistemática de selecionar os parâmetros $\alpha$ e $\beta$ (SHABANA, 2001). A experiência (NIKRAVESH, 1988) confirmava que a escolha com $1 < \alpha, \beta < 10$ era apropriada na maioria dos casos. Agora, aplicaremos a Eq. (2.84) para obtermos o movimento do pêndulo esférico, apresentado no capítulo anterior. A equação de movimento obtida é da forma

$$\begin{bmatrix} m & 0 & 0 & 2x \\ 0 & m & 0 & 2y \\ 0 & 0 & m & 2z \\ 2x & 2y & 2z & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \\ -2\dot{x}^2 - 2\dot{y}^2 - 2\dot{z}^2 \end{bmatrix}. \tag{3.10}$$

Eliminando o multiplicador de Lagrange $\lambda$ da equação acima, podemos escrever

$$\begin{cases} x\ddot{x} + y\ddot{y} + z\ddot{z} = -\dot{x}^2 - \dot{y}^2 - \dot{z}^2 \\ \ddot{x} - \frac{x}{z}\ddot{z} = \frac{x}{z}g \\ \ddot{y} - \frac{y}{z}\ddot{z} = \frac{y}{z}g \end{cases} \tag{3.11}$$

Resolveremos o sistema de equações diferenciais de segunda ordem apresentado na Eq. (3.11), mostrada acima, utilizando o programa Maple, inicialmente sem realizar qualquer estabilização de vínculos. O arquivo Maple desenvolvido para esse fim é denominado *SemEstabilização.mws* [2]. Para facilitar a visualização dos dados, vamos fornecer condições iniciais tais que a coordenada $z$ permaneça constante durante a simulação. Para obter essas condições, basta fazer $\ddot{z} = \dot{z} = 0$ nas equações de movimento, Eq. (3.11), obtendo

$$\begin{cases} x\ddot{x} + y\ddot{y} = -\dot{x}^2 - \dot{y}^2 \\ \ddot{x} = \frac{x}{z}g \\ \ddot{y} = \frac{y}{z}g \end{cases} \tag{3.12}$$

A partir da Eq. (3.12), mostrada acima, podemos obter o módulo do vetor velocidade inicial da forma

$$|\mathbf{V_o}| = \sqrt{-\frac{x^2 + y^2}{z}g}. \tag{3.13}$$

Cabe ressaltar que, como $z$ será constante durante todo o movimento, não haverá componente da velocidade $\mathbf{V_o}$ na direção $z$, como também temos que as coordenadas $x$ e $y$ vão estar sobre uma circunferência. Desta forma, o vetor $\mathbf{V_o}$ deve ser tangente a esta

[2] As rotinas implementadas no programa Maple estão disponíveis no Apêndice A

circunferência. Dando as seguintes condições iniciais

$$\begin{array}{c} x(0) = \frac{r}{\sqrt{2}} \quad y(0) = 0 \quad z(0) = -\frac{r}{\sqrt{2}} \\ \dot{x}(0) = 0 \quad \dot{y}(0) = V \quad \dot{z}(0) = 0 \end{array} , \tag{3.14}$$

obtemos para a coordenada $z$ e sua derivada temporal $\dot{z}$, como também para o termo $\sqrt{x^2+y^2+z^2}$ os resultados[3] ilustrados na Fig (3.1), nas letras **(a)**, **(b)** e **(e)**, respectivamente.

Observando-se os resultados obtidos, verifica-se que esses valores estão em total desacordo ao esperado. Primeiro, $z(t)$ deveria ser constante e $\dot{z}$ nula, o que não acontece. Segundo, a quantidade $\sqrt{x^2+y^2+z^2}$ deveria ser constante e igual a $r$ durante todo o movimento do sistema. Da Fig (3.1), letra **(e)**, verifica-se claramente a violação dos vínculos (comprimento do fio), já que pode-se observar a discrepância entre o valor do raio, igual a $5m$ (em verde no gráfico) e aquele obtido com a expressão, teoricamente equivalente, $\sqrt{x^2+y^2+z^2}$ (em vermelho). Agora, vamos utilizar a estabilização de Baumgarte, escolhendo para $\alpha$ e $\beta$ o mesmo valor constante $\alpha = \beta = 10$. O arquivo Maple implementando a estabilização chama-se *ComEstabilizacao.mws*. Neste caso, as equações de movimento ficam da seguinte forma

$$\begin{cases} x\ddot{x} + y\ddot{y} + z\ddot{z} = -\dot{x}^2 - \dot{y}^2 - \dot{z}^2 - 2\alpha(x\dot{x} + y\dot{y} + z\dot{z}) - \frac{\beta^2}{2}(x^2+y^2+z^2-r^2) \\ \ddot{x} - \frac{x}{z}\ddot{z} = \frac{x}{z}g \\ \ddot{y} - \frac{y}{z}\ddot{z} = \frac{y}{z}g \end{cases} \tag{3.15}$$

Note que as condições iniciais, anteriormente obtidas visando à constância de $z(t)$ ao longo do movimento do pêndulo, continuam válidas. Essas condições *não* poderiam ser

[3]Para resolver numericamente as equações de movimento, foram atribuídos os seguintes valores: $m = 1Kg$, $r = 5m$ e $g = 9.81\frac{m}{s^2}$.

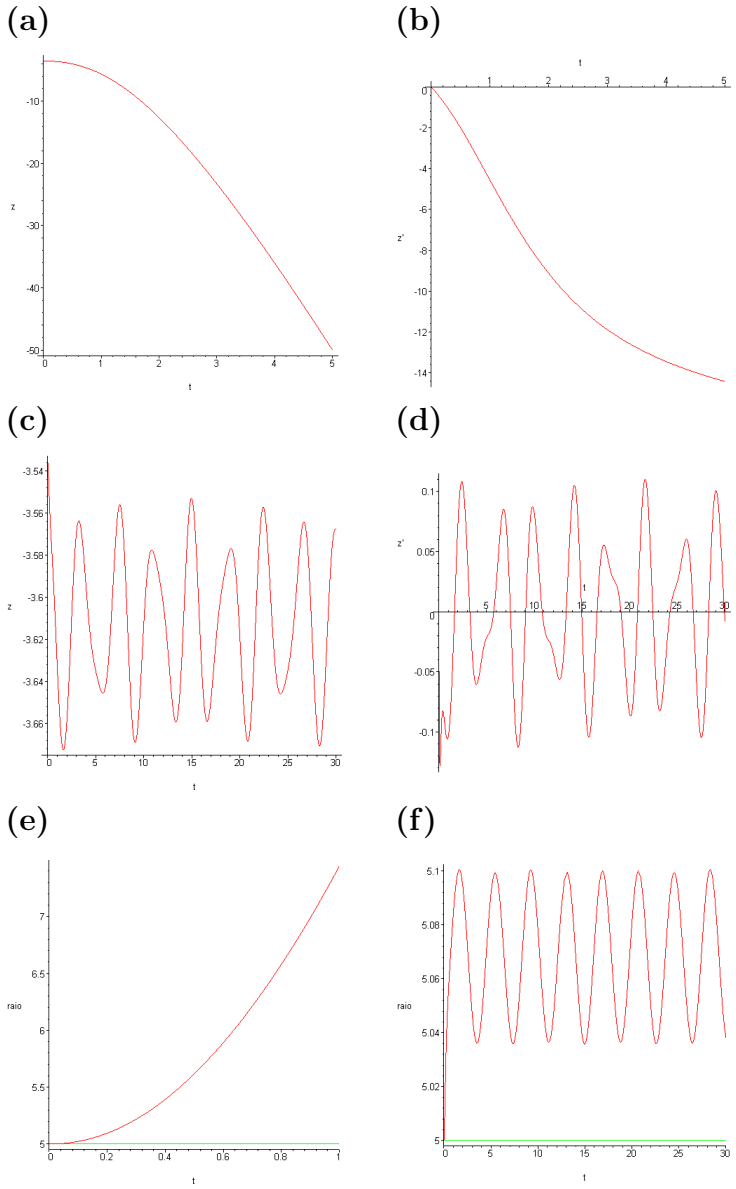



FIGURA 3.1 – Gráficos para $z$, $\dot{z}$ e $\sqrt{x^2+y^2+z^2}$ com e sem estabilização

obtidas a partir do sistema mostrado na Eq. (3.15), já que este sistema possui um termo a mais, devido à estabilização de vínculos. Aplicando, neste caso, as condições iniciais dadas na Eq. (3.14), obtemos para $z(t)$, sua derivada $\dot{z}$ e para o termo $\sqrt{x^2+y^2+z^2}$ os resultados mostrados na Fig. (3.1) letras **(c)**, **(d)** e **(f)**, respectivamente. Observando-se os resultados obtidos, vê-se que, no caso onde foi empregada a estabilização de vínculos de Baumgarte, obtiveram-se resultados muito melhores que aqueles obtidos sem qualquer estabilização. Constata-se, no caso do emprego da estabilização de Baumgarte, a existência

de uma forma de controle, onde procura-se manter os erros envolvidos dentro de uma faixa de tolerância. De fato, o método de Baumgarte é inspirado na teoria de controle, aplicando-se retroação de posição e velocidade à equação (instável) $\ddot{\mathbf{\Phi}} = \mathbf{0}$, através dos termos $\alpha$ e $\beta$, visando estabilizá-la. Um procedimento buscando sistematizar a escolha dos parâmetros $\alpha$ e $\beta$ do método de Baumgarte foi proposta em (ZHENKUAN, 1996), onde os parâmetros $\alpha$ e $\beta$ tornam-se funções do passo tomado na integração numérica das equações de movimento. Este procedimento é baseado na expansão na série de Taylor da função $\mathbf{\Phi}(\mathbf{q}(t), t) = \mathbf{0}$. Observando-se que, supondo conhecidos $t^i$ e $\mathbf{q}^i$, podemos obter o valor das equações de restrição para um passo $h$, onde $t^{i+1} - t^i = h$ da seguinte forma

$$\mathbf{\Phi}^{\mathrm{i+1}} = \mathbf{\Phi}^{\mathrm{i}} + h\dot{\mathbf{\Phi}}^{\mathrm{i}} + \frac{h^2}{2}\ddot{\mathbf{\Phi}}^{\mathrm{i}}. \tag{3.16}$$

O objetivo na Eq. (3.16) é termos $\mathbf{\Phi}^{\mathrm{i+1}} = \mathbf{0}$. Assim, realizando operações simples, obtemos para $\alpha$ e $\beta$ os seguintes valores

$$\alpha = \frac{1}{h} \tag{3.17}$$

$$\beta = \frac{\sqrt{2}}{h}. \tag{3.18}$$

O grande problema do procedimento proposto por (ZHENKUAN, 1996) consiste no fato de, geralmente, não termos acesso ao passo escolhido por programas de solução de equações diferenciais, visto que esses programas são como caixas-pretas. Como neste texto vamos implementar os algoritmos numéricos, esta não é uma dificuldade em potencial. Para ilustrarmos a melhora obtida com este procedimento, vamos resolver a Eq. (3.15) utilizando-o. Para integração numérica das equações de movimento, faremos uso da fórmula de Merson, apresentada em livros que tratam da solução numérica de equações diferencias ordinárias,

como (SHAMPINE, 1994). O programa Maple implementando esse procedimento, denominado *ComEstabilização02.mws*, também é apresentado no Apêndice A. Os resultados obtidos para $z$, sua derivada $\dot{z}$, o termo $\sqrt{x^2+y^2+z^2}$ bem como para o passo estão ilustrados na Fig. (3.2) nas letras **(a)**, **(b)**, **(c)** e **(d)**, respectivamente.

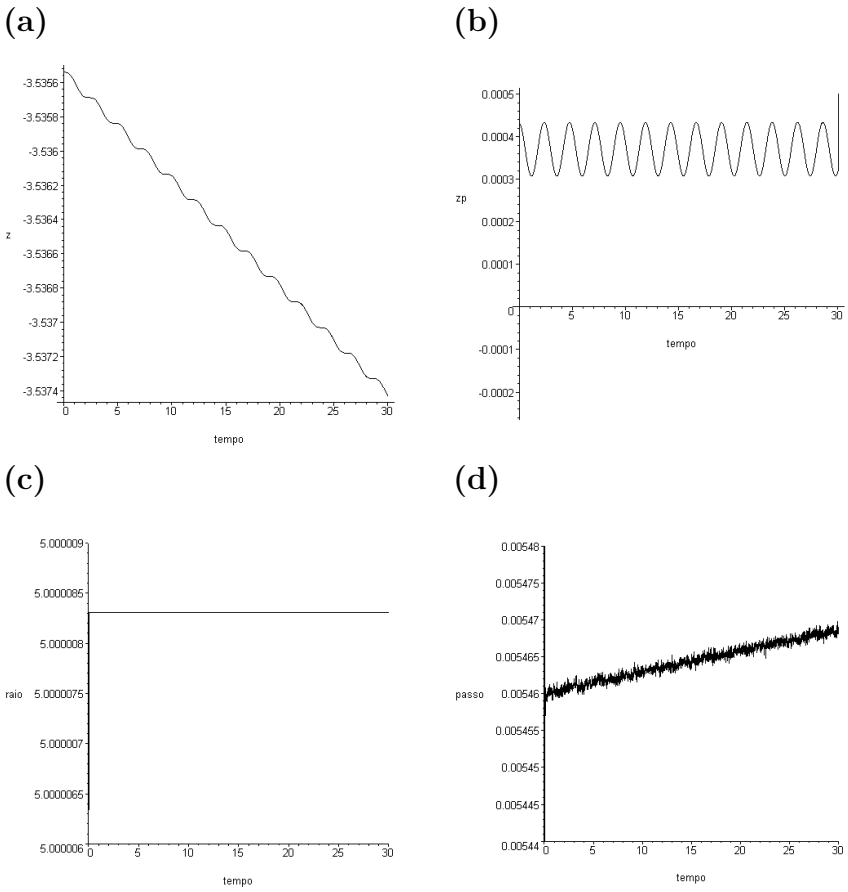



FIGURA 3.2 – Gráficos para $z$, $\dot{z}$ e $\sqrt{x^2+y^2+z^2}$ com estabilização e escolha automática de $\alpha$ e $\beta$.

Como podemos observar da Fig. (3.2), os resultados obtidos foram muito melhores neste caso, em comparação com a escolha de $\alpha$ e $\beta$ constantes. A tolerância escolhida para seleção do passo foi de $tol = 10^{-3}$. Caso quiséssemos escolher uma tolerância menor, deveríamos utilizar uma fórmula de passo múltiplo, visto que essas fórmulas geralmente permitem o cumprimento de uma tolerância mais restritiva, em relação aos métodos de passo único, para um mesmo tamanho de passo. Embora a seleção automática pro-

posta por Zhenkuan tenha melhorado a acurácia da solução, é importante observarmos a influência provocada pela introdução de termos dependentes do passo nas equações de movimento, visto que o passo utilizado durante a solução numérica das equações de movimento foi muito pequeno. De fato, a introdução de termos explicitamente dependentes do passo nas equações de movimento *não* é recomendável, visto que toda a análise matemática realizada nos métodos numéricos de solução de equações diferenciais visando sua validação e, principalmente, objetivando obter estimativas de erro que permitam a escolha automática do passo, assume uma forma para esses métodos que não é mais válida, devido à dependência direta do passo de alguns termos dessas equações. Assim, a aplicabilidade do método de Zhenkuan fica, na prática, restrita a escolha de passo fixo, o que é inaceitável para um método numérico geral, que deve possuir ampla aplicabilidade, permitindo ainda uma maior flexibilidade, como a escolha automática do passo. Outro problema consiste no possível aumento descontrolado dos termos de controle conforme o passo tende a zero. Dessa forma, torna-se necessária a introdução de um método de estabilização de vínculos que seja eficiente e com ampla aplicabilidade, sem introduzir restrições de qualquer tipo. Uma solução satisfatória é utilizar métodos baseados na projeção. O método de projeção apresentado resumidamente abaixo é utilizado no programa computacional desenvolvido, e está apresentado em (BLAJER, 1997). Supondo que após avançarmos a integração das equações de movimento sem qualquer estabilização as restrições estejam fora de uma tolerância especificada, isto é, $|\mathbf{\Phi}| > tol$, podemos corrigir as posições iterativamente, resolvendo-se o seguinte sistema linear

$$\begin{bmatrix} \mathbf{M} & \left(\mathbf{\Phi}_{\mathrm{q}}\right)^T \\ \mathbf{\Phi}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{\Delta}_x \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{\Phi} \end{bmatrix} \tag{3.19}$$

$$\mathbf{q} = \mathbf{q} + \mathbf{\Delta}_x.$$

Note que o sistema linear apresentado acima, procura a solução de Newton-Raphson do problema $\mathbf{\Phi} = \mathbf{0}$ projetada no espaço linha do jacobiano das equações de restrição. Esta projeção permite manter inalterados os valores referentes aos movimentos independentes, já que a projeção efetuada é ortogonal ao espaço nulo[4]. Para velocidade basta resolver diretamente (sem iterações) o seguinte sistema linear

$$\begin{bmatrix} \mathbf{M} & \left(\mathbf{\Phi}_{\mathrm{q}}\right)^T \\ \mathbf{\Phi}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{\Delta}_{\dot{x}} \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\dot{\mathbf{\Phi}} \end{bmatrix} \tag{3.20}$$

$$\dot{\mathbf{q}} = \dot{\mathbf{q}} + \mathbf{\Delta}_{\dot{x}}.$$

O método de projeção apresentado acima está implementado no arquivo *ComProjecao.mws*, também listado no Apêndice A. Os resultados obtidos com este método para o problema do pêndulo esférico estão mostrados abaixo (ver Fig. (3.2))

Observando-se os resultados obtidos, verifica-se que, embora do ponto de vista da acurácia da solução os métodos de projeção e de Baumgarte com seleção automática de passo proposta por Zhenkuan assemelhem-se, o passo utilizado no método de projeção é aproximadamente 25 vezes maior[5], confirmando o método de projeção como a melhor escolha para um programa genérico de simulação de sistemas multicorpos.

[4]Observe que, supondo $\mathbf{x}_{\mathrm{s}}$ uma solução para as equações de restrição, então para todo vetor $\mathbf{v}_{\mathrm{n}}$ pertencente ao espaço nulo, o vetor $\mathbf{x}_{\mathrm{s}} + \mathbf{v}_{\mathrm{n}}$ também será uma solução. Neste sentido, a projeção efetuada na Eq. (3.19) fornece uma solução de norma mínima.

[5]Note que o método utilizado é de passo variável. A invariância do passo apresentada na Fig (3.3) decorre da equação diferencial com a condição inicial dada e da simetria do problema.

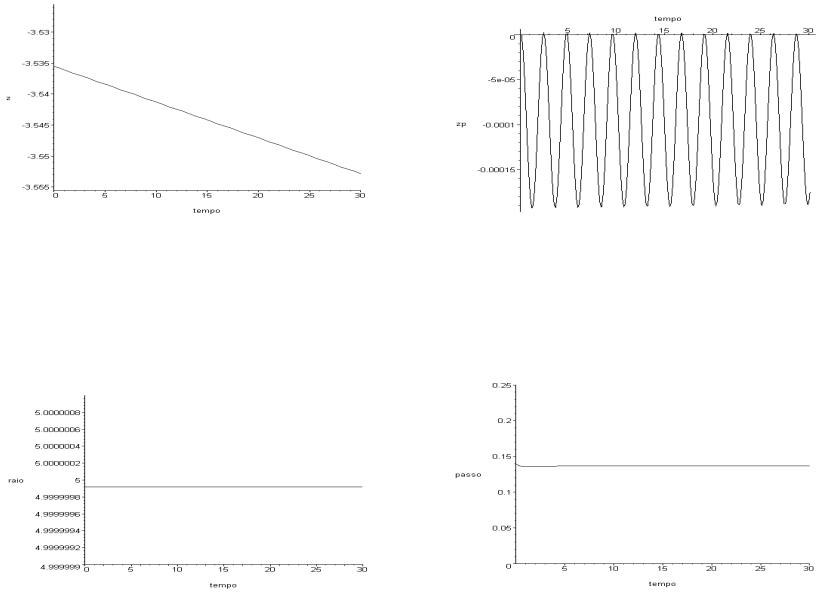



FIGURA 3.3 – Gráficos para $z$, $\dot{z}$ e $\sqrt{x^2+y^2+z^2}$ com projeção.

## 3.3 Método de Partição de Coordenadas

A solução utilizada até agora para resolvermos a Eq. (2.83) baseava-se na introdução de equações extras. Uma alternativa reside em projetarmos as equações de movimento para um espaço de dimensão inferior, onde as coordenadas passariam a ser independentes, após a projeção. Embora existam, como veremos posteriormente, infinitas projeções possíveis, nesta seção vamos nos ater a projeções que relacionem as coordenadas dependentes com um conjunto reduzido *dessas mesmas coordenadas*. Esta é a base para o chamado Método de Partição de Coordenadas proposto por (WEHAGE, 1980), onde as coordenadas são classificadas em dependentes e independentes. Esta classificação é obtida considerando-se um deslocamento virtual nas coordenadas para o vetor de restrições. Este procedimento foi adotado no capítulo anterior, resultando na Eq. (2.51), reescrita abaixo

$$\mathbf{\Phi} = \mathbf{0} \Rightarrow \mathbf{\Phi}_{\mathrm{q}} \delta\mathbf{q} = \mathbf{0}, \tag{3.21}$$

onde, neste caso, o jacobiano das equações de restrição possui o número de linhas independentes inferior ao número de coordenadas. Supondo que o jacobiano possua $l$ linhas linearmente independentes[6] e $n$ colunas, podemos particionar o jacobiano da forma mostrada abaixo

$$\mathbf{\Phi}_{\mathrm{q}} = \begin{bmatrix} \mathbf{\Phi}_{\mathrm{qd}} & \mathbf{\Phi}_{\mathrm{qi}} \end{bmatrix}, \tag{3.22}$$

onde $\mathbf{\Phi}_{\mathrm{qd}}$ e $\mathbf{\Phi}_{\mathrm{qi}}$ são matrizes de dimensão $l \times l$ e $l \times (n-l)$, respectivamente. Sendo o posto do jacobiano igual a $l$, podemos efetuar a partição mostrada na Eq. (3.22) de forma que $\mathbf{\Phi}_{\mathrm{qd}}$ seja inversível. Procedendo desta maneira, podemos escrever a Eq. (3.21) como abaixo

$$\mathbf{\Phi}_{\mathrm{q}} \delta\mathbf{q} = \mathbf{\Phi}_{\mathrm{qd}} \delta\mathbf{q_d} + \mathbf{\Phi}_{\mathrm{qi}} \delta\mathbf{q_i} = \mathbf{0} \Rightarrow \delta\mathbf{q_d} = -\left(\mathbf{\Phi}_{\mathrm{qd}}\right)^{-1} \mathbf{\Phi}_{\mathrm{qi}} \delta\mathbf{q_i}. \tag{3.23}$$

Rearranjando o vetor de coordenadas da forma $\mathbf{q} = [\mathbf{q_d} \;\; \mathbf{q_i}]^T$, o vetor do deslocamento virtual de todas as coordenadas pode ser escrito em função dos deslocamentos virtuais independentes da seguinte maneira

$$\delta\mathbf{q} = \begin{bmatrix} -\left(\mathbf{\Phi}_{\mathrm{qd}}\right)^{-1} \mathbf{\Phi}_{\mathrm{qi}} \\ \mathbf{I} \end{bmatrix} \delta\mathbf{q_i} = \mathbf{B}_{\mathrm{p}} \delta\mathbf{q_i}. \tag{3.24}$$

Observando-se a Eq. (3.24), podemos identificar as colunas da matriz $\mathbf{B}_{\mathrm{p}}$ como sendo

[6] Neste caso, essas equações de restrição englobam as equações de movimento prescrito.

os vetores $\mathbf{t}_i$ da Eq. (2.52). Outra característica importante da matriz $\mathbf{B}_{\text{p}}$ é apresentada abaixo

$$\left(\mathbf{B}_{\text{p}}\right)^T (\mathbf{\Phi})^T = \left[-(\mathbf{\Phi}_{\text{qi}})^T((\mathbf{\Phi}_{\text{qd}})^T)^{-1}) \;\; \mathbf{I}\right] \begin{bmatrix} (\mathbf{\Phi}_{\text{qd}})^T \\ \\ (\mathbf{\Phi}_{\text{qi}})^T \end{bmatrix} = \mathbf{0}. \tag{3.25}$$

Utilizando o resultado da Eq. (3.25), podemos multiplicar a Eq. (2.83) pela transposta da matriz $\mathbf{B}_{\text{p}}$, obtendo

$$(\mathbf{B}_{\text{p}})^T \mathbf{M}\ddot{\mathbf{q}} = (\mathbf{B}_{\text{p}})^T(\mathbf{Q}_{\text{Fext}} - \mathbf{Q}_{\text{v}}). \tag{3.26}$$

Na equação mostrada acima temos $n-l$ equações de movimento independentes, embora essas equações estejam escritas em função de todas as acelerações. Para contornar este problema, podemos realizar a partição de coordenadas na Eq. (2.56), obtendo

$$\ddot{\mathbf{q}}_{\mathbf{d}} = -(\mathbf{\Phi}_{\text{qd}})^{-1}((\mathbf{\Phi}_{\text{q}}\dot{\mathbf{q}})_{\mathbf{q}}\dot{\mathbf{q}} - 2\left(\frac{\partial \mathbf{\Phi}}{\partial t}\right)_{\mathbf{q}} \dot{\mathbf{q}} - \frac{\partial^2 \mathbf{\Phi}}{\partial t^2}) - (\mathbf{\Phi}_{\text{qd}})^{-1}\mathbf{\Phi}_{\text{qi}}\ddot{\mathbf{q}}_{\mathbf{i}}. \tag{3.27}$$

Empregando a Eq. (3.27), podemos escrever o vetor $\ddot{\mathbf{q}}$ da forma

$$\ddot{\mathbf{q}} = \mathbf{B}_{\text{p}}\ddot{\mathbf{q}}_{\mathbf{i}} + \mathbf{\Gamma}, \tag{3.28}$$

onde

$$\mathbf{\Gamma} = \begin{bmatrix} -(\mathbf{\Phi}_{\text{qd}})^{-1}((\mathbf{\Phi}_{\text{q}}\dot{\mathbf{q}})_{\mathbf{q}}\dot{\mathbf{q}} - 2\left(\frac{\partial \mathbf{\Phi}}{\partial t}\right)_{\mathbf{q}} \dot{\mathbf{q}} - \frac{\partial^2 \mathbf{\Phi}}{\partial t^2}) \\ \\ \mathbf{0} \end{bmatrix}. \tag{3.29}$$

Aplicando a Eq. (3.28) na Eq. (3.26), podemos escrever

$$(\mathbf{B}_{\mathrm{p}})^T\mathbf{M}\mathbf{B}_{\mathrm{p}}\ddot{\mathbf{q}}_{\mathbf{i}} = (\mathbf{B}_{\mathrm{p}})^T(\mathbf{Q}_{\mathrm{Fext}} - \mathbf{Q}_{\mathrm{v}}) - (\mathbf{B}_{\mathrm{p}})^T\mathbf{M}\boldsymbol{\Gamma}. \tag{3.30}$$

A Eq. (3.30) representa as equações de movimento projetadas num espaço de dimensão $n - l$. Essas equações sintetizam o método de partição de coordenadas onde, ao particionar-se o jacobiano e, consequentemente, o vetor das coordenadas, em dependentes e independentes, pode-se escrever as equações de movimento em função das acelerações independentes. Deste modo, pode-se integrar numericamente essas equações, obtendo-se o vetor das coordenadas independentes e sua derivada temporal. A obtenção das coordenadas dependentes está condicionada à solução das equações de restrição[7]. Para obtermos a derivada temporal das coordenadas dependentes basta resolver um sistema linear.

Embora o método de partição de coordenadas possua um belo aspecto teórico, contribuindo para um maior compreensão da dinâmica de sistemas multicorpos, do ponto de vista numérico a aplicação deste método é restrita. Isto se deve ao maior número de operações matriciais necessárias limitando, na prática, a aplicação deste método a sistemas peculiares.

## 3.4 Método Baseado em Ortogonalização - Parte I

Observando-se a Eq. (3.25), vê-se que esta equação indica que as colunas da matriz $\mathbf{B}_{\mathrm{p}}$ formam uma base para o espaço nulo do jacobiano das equações de restrição. Daí surge a possibilidade de termos infinitas escolhas para o conjunto de coordenadas independentes,

[7]Note que neste caso o jacobiano das equações de restrição necessário ao método de Newton-Raphson **não** conterá as derivadas em relação às coordenadas independentes, garantindo que, caso exista solução desse problema, esta será única.

visto que existem infinitas bases para o espaço nulo. Nesta seção vamos escolher uma base ortogonal de acordo com um produto interno definido. Toda a formulação matemática deste método pode ser encontrada em (BLAJER, 1995). Em resumo, este método ortogonaliza *as colunas*, utilizando o processo de ortogonalização de Gram-Schimidt, da matriz $\mathbf{B}_{\text{p}}$ de acordo com o seguinte produto interno

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{M} \mathbf{v}, \tag{3.31}$$

e a norma

$$\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}, \tag{3.32}$$

naturalmente associada a este produto[8]. Sendo $\mathbf{B}_{\text{p}}^*$ a matriz ortogonalizada, temos que

$$(\mathbf{B}_{\text{p}}^*)^T \mathbf{M} \mathbf{B}_{\text{p}}^* = \mathbf{I}. \tag{3.33}$$

Visando utilizar o resultado obtido acima na Eq. (3.26)[9], devemos relacionar a derivada temporal segunda do vetor de coordenadas com um conjunto de coordenadas independentes. Procedendo desta maneira podemos fazer a seguinte definição

$$\dot{\mathbf{q}} = \mathbf{B}_{\text{p}}^* \mathbf{v}, \tag{3.34}$$

[8] As propriedades básicas necessárias para uma função ser considerada um produto interno é que esta seja simétrica, bilinear, positiva-definida. É também possível provar matematicamente que sempre existe uma norma tal como dada na Eq. (3.32) associada a um produto interno. Para mais informações a este respeito o leitor deve procurar um livro de Análise Matemática.

[9] Note que a matriz $\mathbf{B}_{\text{p}}$ após o processo de ortogonalização continua tendo suas colunas como uma base do espaço nulo.

onde $\mathbf{v}$ são denominadas de velocidades tangentes[10]. Derivando a Eq. (3.34) em relação ao tempo, podemos escrever

$$\ddot{\mathbf{q}} = \mathbf{B}_{\mathrm{p}}^{*}\dot{\mathbf{v}} + \dot{\mathbf{B}}_{\mathrm{p}}^{*}\mathbf{v}. \tag{3.35}$$

Multiplicando a Eq. (2.83) pela transposta da matriz $\mathbf{B}_{\mathrm{p}}^{*}$ e utilizando a Eq. (3.35), podemos escrever a equação de movimento em função das velocidades tangentes da seguinte forma

$$\dot{\mathbf{v}} = (\mathbf{B}_{\mathrm{p}}^{*})^{T}(\mathbf{Q}_{\mathrm{Fext}} - \mathbf{Q}_{\mathrm{v}}) - (\mathbf{B}_{\mathrm{p}}^{*})^{T}\mathbf{M}^{T}\dot{\mathbf{B}}_{\mathrm{p}}^{*}\mathbf{v}. \tag{3.36}$$

Utilizando a Eq. (3.33), podemos escrever

$$2(\mathbf{B}_{\mathrm{p}}^{*})^{T}\mathbf{M}\dot{\mathbf{B}}_{\mathrm{p}}^{*} + (\mathbf{B}_{\mathrm{p}}^{*})^{T}\dot{\mathbf{M}}\mathbf{B}_{\mathrm{p}}^{*} = \mathbf{0} \Rightarrow (\mathbf{B}_{\mathrm{p}}^{*})^{T}\mathbf{M}\dot{\mathbf{B}}_{\mathrm{p}}^{*} = -\frac{1}{2}(\mathbf{B}_{\mathrm{p}}^{*})^{T}\dot{\mathbf{M}}\mathbf{B}_{\mathrm{p}}^{*}. \tag{3.37}$$

Inserindo o resultado obtido na Eq. (3.37) nas equações de movimento, Eq. (3.36), obtemos

$$\dot{\mathbf{v}} = (\mathbf{B}_{\mathrm{p}}^{*})^{T}(\mathbf{Q}_{\mathrm{Fext}} - \mathbf{Q}_{\mathrm{v}}) + \frac{1}{2}(\mathbf{B}_{\mathrm{p}}^{*})^{T}\dot{\mathbf{M}}^{T}\mathbf{B}_{\mathrm{p}}^{*}\mathbf{v}, \tag{3.38}$$

que é o principal resultado apresentado no artigo de Blajer. Note que, embora a equação de movimento já esteja resolvida no sentido de não necessitarmos da solução de um sistema linear, este método apresenta, do ponto visto computacional, algumas desvantagens. Em primeiro lugar, devemos gerar uma matriz cujas colunas formem uma base para o espaço

[10]Observe que o espaço nulo forma um hiperplano tangente à hiperfície $\mathbf{\Phi} = \mathbf{0}$ no ponto em questão.

nulo, sendo esta matriz, em geral, densa. Após obtermos esta matriz, devemos ortogonalizar suas colunas. Todo este processo não compensa o fato de termos as equações de movimento já resolvidas, como na Eq. (3.38).

# 4 Método Proposto

No capítulo anterior foram apresentadas algumas maneiras de tornarmos mais eficiente a solução numérica das equações de movimento. Este capítulo é uma seqüência natural do anterior, onde será apresentada uma melhoria no método de ortogonalização, introduzido no final do capítulo anterior, bem como será mostrada a solução implementada no programa computacional desenvolvido, apresentado de forma mais detalhada no sexto capítulo.

## 4.1 Método Baseado em Ortogonalização - Parte II

Os dois métodos apresentados neste capítulo possuem um desenvolvimento inicial semelhante, cujo ponto de partida é a Eq. (2.84), reescrita abaixo

$$\begin{bmatrix} \mathbf{M} & (\mathbf{\Phi}_{\mathrm{q}})^T \\ \mathbf{\Phi}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{\mathrm{Ext}} - \mathbf{Q}_{\mathrm{v}} \\ \mathbf{Q}_{\mathrm{d}} \end{bmatrix}. \tag{4.1}$$

O procedimento adotado é semelhante ao descrito em (TASORA, 2000). Contudo, no artigo citado, é utilizado uma decomposição $LDL^T$ da matriz dos coeficientes. Embora uma decomposição semelhante tenha sido implementada no programa desenvolvido, a principal importância deste procedimento reside no fato deste permitir uma melhoria

do método de ortogonalização apresentado anteriormente. Este procedimento, resumidamente descrito abaixo, permitirá transformarmos a matriz de massa $\mathbf{M}$, apresentada na Eq. (4.1), numa matriz constante e diagonal. Para tal, escreveremos as equações de rotação diretamente em função das acelerações angulares dos corpos, onde esses vetores serão escritos no sistema coordenado dos corpos. Observando-se que para um dado corpo rígido $i$ temos a útil relação

$$\frac{1}{4}(\mathbf{G}_{\mathrm{ci}}^{\mathrm{ci}})^T\mathbf{G}_{\mathrm{ci}}^{\mathrm{ci}} + \mathbf{e}_{\mathrm{i}}(\mathbf{e}_{\mathrm{i}})^T = \mathbf{I}, \tag{4.2}$$

onde $\mathbf{I}$ é a matriz identidade de ordem 4, vamos definir, para *todos* os $n$ corpos de um dado sistema multicorpos, as matrizes $\mathbf{T}_{\mathrm{q}}$, $\mathbf{U}_{\mathrm{q}}$ e $\mathbf{S}_{\mathrm{q}}$ da seguinte forma

$$\mathbf{T}_{\mathrm{q}} = \begin{bmatrix} \mathbf{I}_{3\mathrm{x}3} & & & & \\ & \frac{1}{4}(\mathbf{G}_{\mathrm{c1}}^{\mathrm{c1}})^T & & & \\ & & \ddots & & \\ & & & \mathbf{I}_{3\mathrm{x}3} & \\ & & & & \frac{1}{4}(\mathbf{G}_{\mathrm{cn}}^{\mathrm{cn}})^T \end{bmatrix}, \tag{4.3}$$

$$\mathbf{U}_{\mathrm{q}} = \begin{bmatrix} \mathbf{I}_{3\mathrm{x}3} & & & & \\ & \frac{1}{4}(\mathbf{G}_{\mathrm{c1}}^{\mathrm{c1}})^T\mathbf{G}_{\mathrm{c1}}^{\mathrm{c1}} & & & \\ & & \ddots & & \\ & & & \mathbf{I}_{3\mathrm{x}3} & \\ & & & & \frac{1}{4}(\mathbf{G}_{\mathrm{cn}}^{\mathrm{cn}})^T\mathbf{G}_{\mathrm{cn}}^{\mathrm{cn}} \end{bmatrix}, \tag{4.4}$$

$$\mathbf{S}_{\mathrm{q}} = \begin{bmatrix} \mathbf{0} & & & & \\ & \mathbf{e}_{\mathrm{1}}(\mathbf{e}_{\mathrm{1}})^T & & & \\ & & \ddots & & \\ & & & \mathbf{0} & \\ & & & & \mathbf{e}_{\mathrm{n}}(\mathbf{e}_{\mathrm{n}})^T \end{bmatrix}, \tag{4.5}$$

onde claramente temos que

$$\mathbf{U}_{\mathrm{q}} + \mathbf{S}_{\mathrm{q}} = \mathbf{I}. \tag{4.6}$$

Utilizando a Eq. (4.6), podemos escrever as equações de movimento dadas na Eq. (4.1) da seguinte forma

$$\begin{bmatrix} \mathbf{M} & (\mathbf{\Phi}_{\mathrm{q}})^T \\ \mathbf{\Phi}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{\mathrm{q}} + \mathbf{S}_{\mathrm{q}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{\mathrm{Ext}} - \mathbf{Q}_{\mathrm{v}} \\ \mathbf{Q}_{\mathrm{d}} \end{bmatrix} \tag{4.7}$$

$$\Rightarrow \begin{bmatrix} \mathbf{M}\mathbf{U}_{\mathrm{q}} & (\mathbf{\Phi}_{\mathrm{q}})^T \\ \mathbf{\Phi}_{\mathrm{q}}\mathbf{U}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{\mathrm{Ext}} - \mathbf{Q}_{\mathrm{v}} - \mathbf{M}\mathbf{S}_{\mathrm{q}}\ddot{\mathbf{q}} \\ \mathbf{Q}_{\mathrm{d}} - \mathbf{\Phi}_{\mathrm{q}}\mathbf{S}_{\mathrm{q}}\ddot{\mathbf{q}} \end{bmatrix} \tag{4.8}$$

$$\Rightarrow \begin{bmatrix} \mathbf{M}\mathbf{U}_{\mathrm{q}} & (\mathbf{\Phi}_{\mathrm{q}})^T \\ \mathbf{\Phi}_{\mathrm{q}}\mathbf{U}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{\mathrm{Ext}} - \mathbf{Q}_{\mathrm{v}} \\ \mathbf{Q}_{\mathrm{d}} - \mathbf{\Phi}_{\mathrm{q}}\boldsymbol{\gamma}_{\mathrm{q}} \end{bmatrix}. \tag{4.9}$$

Na Eq. (4.9) foi realizada a definição $\boldsymbol{\gamma}_{\mathrm{q}} = \mathbf{S}_{\mathrm{q}}\ddot{\mathbf{q}}$, onde $\boldsymbol{\gamma}_{\mathrm{q}}$ pode ser escrito da forma

$$\boldsymbol{\gamma}_{\mathrm{q}} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{e}_{1}(\dot{\mathbf{e}}_{1})^{T}\dot{\mathbf{e}}_{1} \\ \vdots \\ \mathbf{0} \\ -\mathbf{e}_{\mathrm{n}}(\dot{\mathbf{e}}_{\mathrm{n}})^{T}\dot{\mathbf{e}}_{\mathrm{n}} \end{bmatrix}, \tag{4.10}$$

e, ainda na Eq. (4.9) foi utilizada a seguinte simplificação

$$\mathbf{M}\boldsymbol{\gamma}_{\mathrm{q}} = \mathbf{0}, \tag{4.11}$$

já que este produto matricial envolve termos do tipo $\mathbf{G}_{\mathrm{ci}}^{\mathrm{ci}}\mathbf{e}_{\mathrm{i}}$[1] para os $n$ corpos do sistema multicorpos. Notando na Eq. (4.9) que a matriz $\mathbf{U}_{\mathrm{q}}$ multiplica o vetor $\ddot{\mathbf{q}}$ e que a aceleração angular de um dado corpo rígido $i$ pode ser escrita em função da derivada temporal segunda dos parâmetros de Euler desse mesmo corpo da forma

$$\boldsymbol{\alpha}_{\mathrm{ci}}^{\mathrm{ci}} = \mathbf{G}_{\mathrm{ci}}^{\mathrm{ci}}\ddot{\mathbf{e}}_{\mathrm{i}}, \tag{4.12}$$

podemos escrever o termo $\mathbf{U}_{\mathrm{q}}\ddot{\mathbf{q}}$, da seguinte maneira

$$\mathbf{U}_{\mathrm{q}}\ddot{\mathbf{q}} = \mathbf{T}_{\mathrm{q}}\ddot{\mathbf{q}}_{\boldsymbol{\alpha}}, \tag{4.13}$$

onde

[1]Note que a multiplicação $\mathbf{G}_{\mathrm{ci}}^{\mathrm{ci}}\mathbf{e}_{\mathrm{i}}$ é igual ao vetor zero (NIKRAVESH, 1988).

$$\ddot{\mathbf{q}}_{\boldsymbol{\alpha}} = \begin{bmatrix} \ddot{\mathbf{p}}_{\mathrm{R}}^{\mathrm{C1*/O}} \\ \boldsymbol{\alpha}_{\mathrm{ci}}^{\mathrm{C1}} \\ \vdots \\ \ddot{\mathbf{p}}_{\mathrm{R}}^{\mathrm{Cn*/O}} \\ \boldsymbol{\alpha}_{\mathrm{cn}}^{\mathrm{Cn}} \end{bmatrix}. \tag{4.14}$$

A equação de movimento apresentada na Eq. (4.9) pode então ser escrita em função do vetor $\ddot{\mathbf{q}}_{\boldsymbol{\alpha}}$ da seguinte forma

$$\begin{bmatrix} \mathbf{M}\mathbf{T}_{\mathrm{q}} & (\boldsymbol{\Phi}_{\mathrm{q}})^T \\ \boldsymbol{\Phi}_{\mathrm{q}}\mathbf{T}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_{\boldsymbol{\alpha}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{\mathrm{Ext}} - \mathbf{Q}_{\mathrm{v}} \\ \mathbf{Q}_{\mathrm{d}} - \boldsymbol{\Phi}_{\mathrm{q}}\boldsymbol{\gamma}_{\mathrm{q}} \end{bmatrix}. \tag{4.15}$$

Multiplicando-se a Eq. (4.15), mostrada acima, pela matriz

$$\begin{bmatrix} (\mathbf{T}_{\mathrm{q}})^T & \\ & \mathbf{I} \end{bmatrix}, \tag{4.16}$$

obtemos

$$\begin{bmatrix} (\mathbf{T}_{\mathrm{q}})^T\mathbf{M}\mathbf{T}_{\mathrm{q}} & (\mathbf{T}_{\mathrm{q}})^T(\boldsymbol{\Phi}_{\mathrm{q}})^T \\ \boldsymbol{\Phi}_{\mathrm{q}}\mathbf{T}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_{\boldsymbol{\alpha}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} (\mathbf{T}_{\mathrm{q}})^T(\mathbf{Q}_{\mathrm{Ext}} - \mathbf{Q}_{\mathrm{v}}) \\ \mathbf{Q}_{\mathrm{d}} - \boldsymbol{\Phi}_{\mathrm{q}}\boldsymbol{\gamma}_{\mathrm{q}} \end{bmatrix}. \tag{4.17}$$

Utilizando a seguinte relação

$$\mathbf{G}_{\mathrm{ci}}^{\mathrm{ci}}(\mathbf{G}_{\mathrm{ci}}^{\mathrm{ci}})^T = \mathbf{I}, \tag{4.18}$$

onde, neste caso, a matriz $\mathbf{I}$ é de ordem 3, temos que o termo $(\mathbf{T}_{\mathrm{q}})^T\mathbf{M}\mathbf{T}_{\mathrm{q}}$, apresentado

na Eq. (4.17), é igual a

$$\overline{\mathbf{M}} = (\mathbf{T}_{\mathrm{q}})^T \mathbf{M} \mathbf{T}_{\mathrm{q}} = \begin{bmatrix} m_1\mathbf{I} & & & & \\ & \mathbf{I}_1^* & & & \\ & & \ddots & & \\ & & & m_n\mathbf{I} & \\ & & & & \mathbf{I}_{\mathrm{n}}^* \end{bmatrix}, \tag{4.19}$$

onde $m_i$ é a massa e $\mathbf{I}_{\mathrm{i}}^*$ é o tensor de inércia do corpo $i$, podendo ser representado por uma matriz diagonal[2]. O produto $\mathbf{\Phi}_{\mathrm{q}}\mathbf{T}_{\mathrm{q}}$ será escrito como $\boldsymbol{\varphi}_{\mathrm{q}}$. Cabe observar que neste produto, as equações de restrição devido aos parâmetros de Euler serão iguais a zero[3]. Isto ocorre devido à projeção das derivadas temporais segunda dos parâmetros de Euler dos corpos nos componentes do vetor aceleração angular desses corpos. Assim sendo, podemos eliminar diretamente essas equações de restrição na formação do jacobiano. Logo as equações de movimento podem ser escritas da seguinte forma

$$\begin{bmatrix} \overline{\mathbf{M}} & (\boldsymbol{\varphi}_{\mathrm{q}})^T \\ \boldsymbol{\varphi}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_{\boldsymbol{\alpha}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{Q}} \\ \overline{\mathbf{Q}}_{\mathrm{d}} \end{bmatrix}, \tag{4.20}$$

onde

$$\overline{\mathbf{Q}} = (\mathbf{T}_{\mathrm{q}})^T (\mathbf{Q}_{\mathrm{Ext}} - \mathbf{Q}_{\mathrm{v}}), \tag{4.21}$$

e

[2]Note que é sempre possível escrever o tensor de inércia como uma matriz diagonal escolhendo as direções principais de inércia.

[3]O vetor $\mathbf{Q}_{\mathrm{d}} - \mathbf{\Phi}_{\mathrm{q}}\boldsymbol{\gamma}_{\mathrm{q}}$ simplificará de acordo, dando origem a equações identicamente nulas.

$$\overline{\mathbf{Q}}_{\mathrm{d}} = \mathbf{Q}_{\mathrm{d}} - \boldsymbol{\Phi}_{\mathrm{q}}\boldsymbol{\gamma}_{\mathrm{q}}. \tag{4.22}$$

Note que a simplificação nas equações de movimento, obtida acima, se deve ao fato de termos escrito as equações de movimento referentes à rotação dos corpos num sistema coordenado fixo nesses corpos, tornando a matriz de inércia constante. As equações de translação continuam sendo escritas num sistema coordenado fixo no referencial inercial[4].

A semelhança com o procedimento descrito em (TASORA, 2000), termina neste ponto. No artigo citado, é utilizada uma decomposição diretamente na matriz dos coeficientes, mostrada acima na Eq. (4.20)[5]. Multiplicando a Eq. (4.20) pela matriz

$$\begin{bmatrix} (\overline{\mathbf{M}})^{-1} & \mathbf{0} \\ \boldsymbol{\varphi}_{\mathrm{q}}(\overline{\mathbf{M}})^{-1} & -\mathbf{I} \end{bmatrix}, \tag{4.23}$$

obtemos

$$\begin{bmatrix} \mathbf{I} & (\overline{\mathbf{M}})^{-1}(\boldsymbol{\varphi}_{\mathrm{q}})^{T} \\ \mathbf{0} & \boldsymbol{\varphi}_{\mathrm{q}}(\overline{\mathbf{M}})^{-1}(\boldsymbol{\varphi}_{\mathrm{q}})^{T} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_{\boldsymbol{\alpha}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} (\overline{\mathbf{M}})^{-1}\overline{\mathbf{Q}} \\ \boldsymbol{\varphi}_{\mathrm{q}}(\overline{\mathbf{M}})^{-1}\overline{\mathbf{Q}} - \overline{\mathbf{Q}}_{\mathrm{d}} \end{bmatrix}. \tag{4.24}$$

A Eq. (4.24) é a base para o método de ortogonalização. Observando o termo

$$\boldsymbol{\varphi}_{\mathrm{q}}(\overline{\mathbf{M}})^{-1}(\boldsymbol{\varphi}_{\mathrm{q}})^{T}, \tag{4.25}$$

[4]Não devemos confundir sistema coordenado com referencial. As equações de movimento claramente são escritas em relação a um referencial considerado inercial. Contudo, podemos representar essas equações em relação a um sistema coordenado arbitrário.

[5]Como dito no início deste capítulo, uma decomposição semelhante foi implementada no programa desenvolvido. Esta implementação será apresentada no próximo capítulo, tratando da solução numérica das equações de movimento.

vemos que este guarda uma semelhança com o termo $(\mathbf{B}_{\mathrm{p}}^{*})^{T}\mathbf{M}\mathbf{B}_{\mathrm{p}}^{*}$, encontrado no método apresentado no capítulo anterior e desenvolvido por (BLAJER, 1995). Contudo, naquele caso, ortogonalizamos as colunas da matriz $\mathbf{B}_{\mathrm{p}}$ e neste caso ortogonalizaremos as linhas da matriz $\boldsymbol{\varphi}_{\mathrm{q}}$ (note a ordem das transpostas nas respectivas equações), utilizando o seguinte produto interno

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^{T}(\overline{\mathbf{M}})^{-1}\mathbf{v}, \tag{4.26}$$

e a seguinte norma associada a este produto

$$\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}. \tag{4.27}$$

Denotando a matriz $\boldsymbol{\varphi}_{\mathrm{q}}^{*}$ como sendo a versão ortogonalizada da matriz $\boldsymbol{\varphi}_{\mathrm{q}}$, temos que

$$\boldsymbol{\varphi}_{\mathrm{q}}^{*}(\overline{\mathbf{M}})^{-1}(\boldsymbol{\varphi}_{\mathrm{q}}^{*})^{T} = \mathbf{I}. \tag{4.28}$$

As vantagens de utilizarmos o método proposto de ortogonalização está na eliminação da necessidade de gerarmos uma base do espaço nulo para então ortogonalizá-la. Observe que, embora existam outras maneiras de gerarmos uma base deste espaço, além daquela apresentada na Eq. (3.24), onde foi utilizada a partição de coordenadas, todas incorrem num elevado custo computacional. Outra vantagem do método proposto neste capítulo reside no fato de, realizada a ortogonalização, as equações já estão resolvidas diretamente para as coordenadas. Contudo, a principal vantagem está no fato do método proposto no texto ocasionar um número muito inferior de preenchimentos, permitindo o uso de métodos para matrizes esparsas, proporcionando um considerável aumento de eficiência.

De fato, verifica-se que a matriz $\mathbf{B}_{\mathrm{p}}$ é geralmente densa (SHABANA, 2001).

## 4.2 Método Implementado

Embora os métodos de ortogonalização sejam atraentes, contribuindo para um aprofundamento do hábito de interpretar geometricamente os espaços onde, localmente, podemos analisar o comportamento dos sistemas multicorpos, esses métodos não são tão eficientes do ponto de vista computacional, estando restritos a aplicações específicas. Visando implementar um método genérico de solução das equações de movimento que apresente uma boa eficiência para uma ampla classe de sistemas multicorpos é preferível, do ponto de vista numérico, resolvermos o sistema dado na Eq. (4.20). Note que um procedimento alternativo à solução direta das equações anteriormente citadas, seria resolvermos o seguinte sistema linear

$$[\boldsymbol{\varphi}_{\mathrm{q}}(\overline{\mathbf{M}})^{-1}(\boldsymbol{\varphi}_{\mathrm{q}})^{T}]\boldsymbol{\lambda} = \boldsymbol{\varphi}_{\mathrm{q}}(\overline{\mathbf{M}})^{-1}\overline{\mathbf{Q}} - \overline{\mathbf{Q}}_{\mathrm{d}}, \tag{4.29}$$

onde obtemos o vetor dos multiplicadores de Lagrange, $\boldsymbol{\lambda}$, e substituimos diretamente este vetor na equação mostrada abaixo para obtermos as acelerações

$$\ddot{\mathbf{q}}_{\boldsymbol{\alpha}} = (\overline{\mathbf{M}})^{-1}(\overline{\mathbf{Q}} - (\boldsymbol{\varphi}_{\mathrm{q}})^{T}\boldsymbol{\lambda}). \tag{4.30}$$

Uma característica importantíssima da matriz dos coeficientes da Eq. (4.29) é o fato desta matriz ser positiva definida, viabilizando implementações específicas para métodos de solução direta[6], visando aumento da eficiência computacional. Contudo, esse procedi-

[6]No próximo capítulo mostraremos porque métodos iterativos para solução do sistema linear mostrado na Eq. (4.29) não são recomendados.

mento alternativo também não é concorrente da solução direta da Eq. (4.20). Isto se deve ao fato de, neste procedimento alternativo, termos que acessar os elementos da matriz esparsa (jacobiano) por linhas e colunas. Isto forçou a utilização de uma estrutura de dados para a matriz esparsa baseada em endereçamento indireto, o que não é muito eficiente para computadores escalares (DUFF; ERISMAN; REID, 1989). Além disso, no procedimento alternativo também temos que realizar um maior número de operações matriciais, bem como a estrutura final do sistema a ser resolvido não facilita a implementação de métodos implícitos de solução de equações diferenciais ordinárias, dificultando ainda a implementação de métodos tolerantes a equações redundantes, já que a presença dessas equações faz com que a matriz $\boldsymbol{\varphi}_{\mathrm{q}}(\overline{\mathbf{M}})^{-1}(\boldsymbol{\varphi}_{\mathrm{q}})^{T}$ deixe de ser positiva definida. Embora este procedimento tenha sido inicialmente implementado no programa desenvolvido, este apresentou uma performance bem inferior à solução direta da Eq. (4.20) (no mínimo 50% mais lento). Esta solução direta é baseada, à semelhança da decomposição utilizada em (TASORA, 2000), numa decomposição $LDL^{T}$ da matriz dos coeficientes. Contudo, no nosso caso apresentaremos a implementação do procedimento de solução, ao passo que no artigo citado são apresentados somente fragmentos em C++ de códigos não otimizados, como ressaltado pelo autor do artigo, para fatoração $LDL^{T}$ das equações de movimento. No próximo capítulo, tratando especificamente da solução numérica das equações de movimento, abordaremos a questão da estrutura de dados para matrizes esparsas de forma mais aprofundada, bem como discutiremos a solução numérica das equações diferenciais de movimento, tema principal daquele capítulo.

Por fim, devemos notar algumas implicações em utilizarmos as acelerações angulares para escrevermos as equações de movimento. A primeira delas é que devemos relacionar o vetor $\ddot{\mathbf{q}}$ com o vetor $\ddot{\mathbf{q}}_{\boldsymbol{\alpha}}$. Fazendo uso da Eq. (4.2) e da relação $\mathbf{e}^{T}\ddot{\mathbf{e}} = -\dot{\mathbf{e}}^{T}\dot{\mathbf{e}}$, podemos

obter para cada corpo $i$ do sistema multicorpos o vetor $\ddot{\mathbf{e}}_i$ em função do vetor $\boldsymbol{\alpha}_{ci}^{\text{ci}}$ da seguinte forma

$$\ddot{\mathbf{e}}_i = \frac{1}{4}(\mathbf{G}_{\text{ci}}^{\text{ci}})^T \boldsymbol{\alpha}_{\text{ci}}^{\text{ci}} + (\dot{\mathbf{e}}_i^T \dot{\mathbf{e}}_i)\mathbf{e}_i. \tag{4.31}$$

A Eq. (4.31) permite escrevermos o vetor $\ddot{\mathbf{q}}$ em função do vetor $\ddot{\mathbf{q}}_{\boldsymbol{\alpha}}$.

Outra implicação importante se refere ao método de estabilização de vínculo baseado na projeção, apresentado no capítulo anterior. Embora a matriz dos coeficientes seja função das coordenadas e do tempo, esta varia muito pouco após um passo do integrador numérico, permitindo a utilização de uma única decomposição $LDL^T$ para obtermos as acelerações e realizarmos a projeção, se necessária. Entretanto, os vetores $\boldsymbol{\Delta}_x$ e $\boldsymbol{\Delta}_{\dot{x}}$, presentes nas Eqs. (3.19, 3.20), respectivamente, devem ser adaptados de acordo com o método apresentado neste capítulo, de forma a possibilitar que a fatoração realizada para as equações de movimento seja aplicável à projeção.

Aplicando às Eqs. (3.19, 3.20) o procedimento apresentado neste capítulo, Eqs. (4.1-4.20), obtemos

$$\begin{bmatrix} \overline{\mathbf{M}} & (\boldsymbol{\varphi}_{\text{q}})^T \\ \boldsymbol{\varphi}_{\text{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}_{x\boldsymbol{\alpha}} \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\boldsymbol{\Phi} - \boldsymbol{\Phi}_{\text{q}} \mathbf{S}_{\text{q}} \boldsymbol{\Delta}_x \end{bmatrix}, \tag{4.32}$$

para posição e

$$\begin{bmatrix} \overline{\mathbf{M}} & (\boldsymbol{\varphi}_{\mathrm{q}})^T \\ \boldsymbol{\varphi}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}_{\dot{x}\boldsymbol{\alpha}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\dot{\boldsymbol{\Phi}} - \boldsymbol{\Phi}_{\mathrm{q}}\mathbf{S}_{\mathrm{q}}\boldsymbol{\Delta}_{\dot{x}} \end{bmatrix}, \tag{4.33}$$

para velocidade, onde

$$\mathbf{U}_{\mathrm{q}}\boldsymbol{\Delta}_{x} = \mathbf{T}_{\mathrm{q}}\boldsymbol{\Delta}_{x\boldsymbol{\alpha}}, \tag{4.34}$$

e

$$\mathbf{U}_{\mathrm{q}}\boldsymbol{\Delta}_{\dot{x}} = \mathbf{T}_{\mathrm{q}}\boldsymbol{\Delta}_{\dot{x}\boldsymbol{\alpha}}. \tag{4.35}$$

A principal questão sobre a adaptação do método de projeção ao procedimento proposto por Tasora reside no fato de como obter os vetores $\boldsymbol{\Delta}_{x}$ e $\boldsymbol{\Delta}_{\dot{x}}$ em função dos vetores $\boldsymbol{\Delta}_{x\boldsymbol{\alpha}}$ e $\boldsymbol{\Delta}_{\dot{x}\boldsymbol{\alpha}}$, respectivamente, como também na forma para simplificar os termos $\mathbf{S}_{\mathrm{q}}\boldsymbol{\Delta}_{x}$, $\mathbf{S}_{\mathrm{q}}\boldsymbol{\Delta}_{\dot{x}\boldsymbol{\alpha}}$. Observando-se que

$$\boldsymbol{\Delta}_{x} = \begin{bmatrix} \boldsymbol{\Delta}\mathbf{p}^{\mathrm{c1/c1^*}} \\ \boldsymbol{\Delta}\mathbf{e}_{1} \\ \vdots \\ \boldsymbol{\Delta}\mathbf{p}^{\mathrm{cn/cn^*}} \\ \boldsymbol{\Delta}\mathbf{e}_{\mathrm{n}} \end{bmatrix}, \tag{4.36}$$

vamos considerar que as correções $\boldsymbol{\Delta}\mathbf{e}_{\mathrm{i}}$, para cada um dos corpos $i$ do sistema em questão,

deve satisfazer a condição de módulo unitário, isto é

$$\begin{aligned}(\mathbf{e}_{\mathrm{i}} + \boldsymbol{\Delta}\mathbf{e}_{\mathrm{i}})^T (\mathbf{e}_{\mathrm{i}} + \boldsymbol{\Delta}\mathbf{e}_{\mathrm{i}}) &= 1 \\ (\mathbf{e}_{\mathrm{i}})^T \boldsymbol{\Delta}\mathbf{e}_{\mathrm{i}} &= \frac{1 - (\mathbf{e}_{\mathrm{i}})^T(\mathbf{e}_{\mathrm{i}})}{2},\end{aligned} \tag{4.37}$$

onde o termo $(\boldsymbol{\Delta}\mathbf{e}_{\mathrm{i}})^T(\boldsymbol{\Delta}\mathbf{e}_{\mathrm{i}})$ foi desconsiderado face aos demais. Analogamente, para projeção das velocidades, notando que

$$\boldsymbol{\Delta}_{\dot{x}} = \begin{bmatrix} \boldsymbol{\Delta}\dot{\mathbf{p}}^{\mathrm{c1/c1^*}} \\ \boldsymbol{\Delta}\dot{\mathbf{e}}_{1} \\ \vdots \\ \boldsymbol{\Delta}\dot{\mathbf{p}}^{\mathrm{cn/cn^*}} \\ \boldsymbol{\Delta}\dot{\mathbf{e}}_{\mathrm{n}} \end{bmatrix}, \tag{4.38}$$

vamos considerar que as correções $\boldsymbol{\Delta}\dot{\mathbf{e}}_{\mathrm{i}}$, para cada um dos corpos $i$ do sistema em questão, deve satisfazer a condição de módulo constante do vetor, ou seja

$$\begin{aligned}(\mathbf{e}_{\mathrm{i}})^T (\dot{\mathbf{e}}_{\mathrm{i}} + \boldsymbol{\Delta}\dot{\mathbf{e}}_{\mathrm{i}}) &= 0 \\ (\mathbf{e}_{\mathrm{i}})^T \boldsymbol{\Delta}\dot{\mathbf{e}}_{\mathrm{i}} &= -(\mathbf{e}_{\mathrm{i}})^T(\dot{\mathbf{e}}_{\mathrm{i}}).\end{aligned} \tag{4.39}$$

As Eqs. (4.37, 4.39) permitem a obtenção direta dos termos $\boldsymbol{\Phi}_{\mathrm{q}}\mathbf{S}_{\mathrm{q}}\boldsymbol{\Delta}_{x}$, $\boldsymbol{\Phi}_{\mathrm{q}}\mathbf{S}_{\mathrm{q}}\boldsymbol{\Delta}_{\dot{x}}$ (ver Eqs. (4.32, 4.33)). Em relação à obtenção dos vetores de correção em função dos vetores projetados, temos que para cada corpo $i$ do sistema,

$$\mathbf{\Delta p}^{\mathrm{ci/ci^*}} = \mathbf{\Delta p}_{\boldsymbol{\alpha}}^{\mathrm{ci/ci^*}} \tag{4.40}$$

$$\mathbf{\Delta e}_{\mathrm{i}} = \frac{1}{4}(\mathbf{G}_{\mathrm{ci}}^{\mathrm{ci}})^T \mathbf{\Delta e}_{i\boldsymbol{\alpha}} - \frac{1-(\mathbf{e}_{\mathrm{i}})^T(\mathbf{e}_{\mathrm{i}})}{2}\mathbf{e}_{\mathrm{i}}, \tag{4.41}$$

para correção da posição e

$$\mathbf{\Delta \dot{p}}^{\mathrm{ci/ci^*}} = \mathbf{\Delta \dot{p}}_{\boldsymbol{\alpha}}^{\mathrm{ci/ci^*}} \tag{4.42}$$

$$\mathbf{\Delta \dot{e}}_{\mathrm{i}} = \frac{1}{4}(\mathbf{G}_{\mathrm{ci}}^{\mathrm{ci}})^T \mathbf{\Delta \dot{e}}_{i\boldsymbol{\alpha}} + (\mathbf{e}_{\mathrm{i}})^T(\dot{\mathbf{e}}_{\mathrm{i}})\mathbf{e}_{\mathrm{i}}, \tag{4.43}$$

para correção da velocidade, onde os termos nos quais aparecem o subscrito $\boldsymbol{\alpha}$ pertencem aos vetores $\mathbf{\Delta}_{x\boldsymbol{\alpha}}$ ou $\mathbf{\Delta}_{\dot{x}\boldsymbol{\alpha}}$.

# 5 Solução Numérica

Neste capítulo discutiremos a solução numérica das equações de movimento. Esta discussão abordará não somente o importante tema da integração numérica das equações diferenciais de movimento; trataremos, também, da solução numérica de sistemas lineares esparsos. Inicialmente, discutiremos as formas para se armazenar uma matriz esparsa apresentando, em seguida, a questão da solução de sistemas lineares esparsos. Por fim, trataremos da solução numérica de equações diferenciais ordinárias, onde será discutido o método preditor-corretor implementado no programa desenvolvido. No Anexo A é apresentada a questão da solução numérica das equações de movimento de um ponto de vista mais aplicado, voltado à implementação computacional propriamente dita.

## 5.1 Matrizes Esparsas - Parte I

Sistemas lineares esparsos surgem com freqüência em problemas de engenharia, como na discretização de equações diferenciais parciais, onde essas matrizes geralmente apresentam um padrão bem definido. No caso da simulação computacional de sistemas multicorpos, a primeira matriz esparsa a qual temos contato é com o jacobiano das equações de restrição. Nesta seção, abordaremos o problema da representação dessas matrizes esparsas, bem como discutiremos operações básicas com essas matrizes. Essas funções básicas

dão suporte a algoritmos mais sofisticados, como os de ortogonalização e de solução de sistemas lineares.

Uma das principais questões relativas às matrizes esparsas é a forma de sua representação computacional. Uma das técnicas comumente empregadas é a representação através de listas encadeadas (DUFF; ERISMAN; REID, 1989), permitindo a economia de memória. Isto possibilita a solução de sistemas com milhares ou até mesmo milhões de equações. Por isso, as estruturas de dados para matrizes esparsas priorizam fundamentalmente a economia de memória, explorando o padrão de preenchimento dessas matrizes. Entretanto, no caso da dinâmica de sistemas multicorpos rígidos, o uso de métodos esparsos se faz necessário quase que exclusivamente por questões de eficiência[1]. Por exemplo, um sistema multicorpos constituído por um pêndulo com cem corpos conectados por juntas de revolução possui um jacobiano de dimensão $500 \times 700$, ocupando na representação densa em ponto flutuante com precisão dupla, 2.8MB. Para efeito de comparação, o programa utilizado pelo autor para visualização de arquivos com extensão .pdf ocupa, somente para inicializar, cerca de 30MB. Do exposto, uma alternativa seria gerar a matriz completa e matrizes de índices, indicando os elementos não nulos na matriz completa. Contudo, este método apresenta a desvantagem de utilizar endereçamento indireto o que pode ser um problema, visto que esses métodos podem ser menos eficientes quando executados em máquinas escalares (DUFF; ERISMAN; REID, 1989). Para contornar esse possível problema, devemos utilizar uma armazenagem em vetores compactos, guardando somente os elementos não-nulos. No programa desenvolvido os dois métodos foram implementados e forneceram resultados semelhantes em termos de eficiência somente para sistemas simples com reduzido número de corpos. Note que a estrutura de dados escolhida influ-

[1] Cabe ressaltar também que, no caso da dinâmica de multicorpos, o jacobiano das equações de restrição *não* possui um padrão pré-determinado.

encia diretamente a performance de certas operações. De fato escolhe-se uma estrutura de dados visando as operações necessárias a serem implementadas. Por exemplo, a forma completa foi implementada no método de solução apresentado no capítulo anterior para solução da Eq. (4.29), sendo a forma compacta utilizada na decomposição direta da matriz dos coeficientes, Eq. (4.20). O uso da forma completa permite um maior número de operações, como a multiplicação da matriz por vetor (busca nas linhas) e multiplicação da transposta da matriz por vetor (busca nas colunas), sem perda de eficiência, o que não ocorre com a forma compacta. Contudo, os algoritmos implementados para a forma compacta são mais sofisticados do que aqueles implementados na forma completa, visto que, na forma compacta, os algoritmos implementados tiram todo o proveito da estrutura da matriz dos coeficientes. Cabe notar que, embora o jacobiano das equações de restrição não apresente uma forma bem definida, inviabilizando implementações específicas para o algoritmo de ortogonalização, o mesmo não ocorre com a matriz da Eq. (4.20). Esta característica contribui para a grande eficiência obtida com o método implementado a partir da forma compacta.

## 5.2 Matrizes Esparsas - Parte II

Nesta seção trataremos da solução de sistemas lineares esparsos. Neste contexto, o capítulo anterior introduziu duas formas para resolvermos o sistema linear fornecendo as acelerações e os multiplicadores de Lagrange. No primeiro deles, foi utilizado um método de ortogonalização, onde se obteve uma decomposição $QR$[2] da matriz. Embora este método forneça bons resultados, infelizmente, em termos de eficiência, este não se

[2]Observe que a matriz Q *não* é gerada. Seu efeito é obtido alterando-se o vetor do lado direito. No caso do método apresentado, este vetor é dado pela Eq. (4.22).

compara com o método direto, também apresentado no capítulo anterior em duas versões (ver Eqs. (4.29, 4.20). A principal razão desta ineficiência está no fato do número de preenchimentos ser superior no caso da ortogonalização. Este preenchimento é maior visto que, durante todo o processo de ortogonalização, operamos com todos os componentes dos vetores ao passo que, no método direto, emprega-se o algoritmo de Gauss onde, a cada iteração, opera-se com os vetores de uma sub-matriz de menor dimensão. Nas simulações efetuadas, o tempo necessário à solução das equações de movimento fazendo uso do método de ortogonalização é, no mínimo, 20% superior ao tempo obtido com o emprego da solução direta.

Outra técnica largamente utilizada na solução de sistemas lineares esparsos são os métodos iterativos (SAAD, 2000). Poderíamos utilizar, por exemplo, o método *SOR*[3] no sistema da Eq. (4.29), já que a matriz dos coeficientes desta equação é simétrica e positiva definida, assegurando a convergência do método[4]. Contudo, o emprego desses métodos não é recomendável em substituição ao método direto utilizado na solução da Eq. (4.29), pois a matriz dos coeficientes desta equação possui um elevado número de condição[5], retardando a convergência desses métodos iterativos.

O método direto empregado no programa desenvolvido para a solução do sistema linear mostrado na Eq. (4.29) é uma especialização do tradicional método direto de Gauss, adaptado ao caso de matrizes simétricas, positiva-definidas, denominado Método de Grau Mínimo (DUFF; ERISMAN; REID, 1989). Neste método os pivos são escolhidos na diago-

[3]Sigla para *Successive Over Relaxation*.

[4]Em (SAAD, 2000) há o teorema (sem prova) da convergência do método *SOR* no caso de matrizes simétricas, positiva-definidas (Teorema 4.6).

[5]Note que o número de condição da matriz da respectiva equação é aumentado devido ao fato da matriz dos coeficientes ser obtida em função de outras matrizes. Esta é a razão de, no caso geral, não multiplicarmos o sistema linear $Ax = b$ pela transposta da matriz $A$, obtendo uma matriz de coeficientes simétrica.

nal[6], visando preservar a simetria, onde o pivo escolhido é aquele que se situa na linha com o menor número de entradas. Com este critério é possível escolher uma ordenação inicial para a matriz dos coeficientes, visando a minimização dos preenchimentos. Dessa forma, não é necessária a troca de linhas durante as eliminações a cada passo do integrador, tornando o método mais eficiente. Outra característica importante do método direto implementado é o fato deste ser tolerante a equações redundantes. Esta característica do método foi implementada eliminando-se as equações de restrição redundantes, sendo esta redundância detectada através de um método de ortogonalização. Esta característica é muito importante, visto que essas equações redundantes ocorrem com certa freqüência. Utilizar um método de ortogonalização na detecção de equações redundantes é uma vantagem, visto que este método é o mais robusto na detecção dessas restrições (JALóN; BAYO, 1988). No capítulo anterior, foi dito que a decomposição $LDL^T$ da matriz dos coeficientes da Eq. (4.20) havia sido implementada. Embora esta decomposição seja obtida com o método direto de Gauss, a fatoração é realizada de forma diferente, onde é empregada a fatoração de Doolittle (DUFF; ERISMAN; REID, 1989), que melhor se adequa a estrutura de dados implementada. Neste caso, a ordenação ótima das linhas é obtida considerando-se que, sendo $n$ o número de corpos, após $6n$ passos na fatoração, a matriz dos coeficientes será da forma

$$\begin{bmatrix} \overline{\mathbf{M}} & (\varphi_{\mathrm{q}})^T \\ \mathbf{0} & (\varphi_{\mathrm{q}})(\overline{\mathbf{M}})^{-1}(\varphi_{\mathrm{q}})^T \end{bmatrix}. \tag{5.1}$$

Note que os termos da matriz $(\varphi_{\mathrm{q}})(\overline{\mathbf{M}})^{-1}(\varphi_{\mathrm{q}})^T$ e as inserções devido ao processo de fatoração constituem os preenchimentos da matriz original. Como a matriz $(\varphi_{\mathrm{q}})(\overline{\mathbf{M}})^{-1}(\varphi_{\mathrm{q}})^T$

[6]O fato da matriz ser simétrica e positiva definida garante que os pivos da diagonal produzem uma fatoração estável.

é a que aparece na Eq. (4.29), o mesmo algoritmo para ordenação das linhas utilizado naquele caso pode ser utilizado também neste. Contudo, como não geramos a matriz $(\varphi_q)(\overline{\mathbf{M}})^{-1}(\varphi_q)^T$ diretamente, já que os elementos inseridos são aqueles provenientes da fatoração, este método torna-se mais eficiente. Logo, antes de dar início à integração numérica das equações de movimento, onde a Eq. (4.20) deverá ser resolvida sucessivas vezes, é realizada uma extensa análise na estrutura da matriz dos coeficientes dessa equação.

Por fim, cabe ressaltar também que, no caso do programa comercial Adams, o *solver* linear tolerante a equações dependentes é o *Harwell*, cerca de 30% mais lento que o *solver* linear padrão, o *Calahan* (RAMPALLI, 2000).

## 5.3 Equações Diferenciais - Solução Numérica

A solução numérica das equações diferencias ordinárias representando as equações de movimento é, como não podia deixar de ser, o tema principal deste capítulo. Embora existam inúmeras técnicas para integração numérica dessas equações, somente alguns poucos métodos são largamente utilizados (SHAMPINE, 1994). Os métodos numéricos classificam-se entre explícitos e implícitos. Os métodos explícitos, como o próprio nome sugere, constituem fórmulas explícitas, onde todos os termos envolvidos são conhecidos, podendo ser resolvidos diretamente, enquanto nos métodos implícitos surgem termos, em geral não lineares, dependendo diretamente da solução. Assim, esta é obtida geralmente por meio de métodos iterativos.

O método numérico empregado no programa desenvolvido é do tipo preditor-corretor, onde é utilizada uma fórmula explícita, visando aproximar a solução, sendo esta corrigida

por um método implícito formando assim um par de fórmulas. O par implementado possui como preditor um método da família de Adams-Bashforth e o corretor da família de Adams-Moulton. Supondo que tenhamos a equação diferencial $\mathbf{y}' = \mathbf{F}$ e que tenhamos também as derivadas em $\nu$ pontos previamente calculados, a fórmula de Adams-Bashforth de ordem $\nu$ é da forma (SHAMPINE, 1994)

$$\mathbf{y}_{\mathrm{j+1}} = \mathbf{y}_{\mathrm{j}} + h_j \sum_{k=1}^{\nu} \beta^{\mathrm{B}}_{\nu,k} \mathbf{F}_{\mathrm{j+1-k}}, \tag{5.2}$$

onde

$$\beta^{\mathrm{B}}_{\nu,k} = \frac{1}{h_j} \int_{t_j}^{t_j+h_j} \prod_{m=1 \quad m\neq k}^{\nu} \left( \frac{x - t_{j+1-m}}{t_{j+1-k} - t_{j+1-m}} \right) dx. \tag{5.3}$$

A fórmula de Adams-Moulton de ordem $\nu$ é da forma

$$\mathbf{y}_{\mathrm{j+1}} = \mathbf{y}_{\mathrm{j}} + h_j \sum_{k=0}^{\nu-1} \beta^{\mathrm{M}}_{\nu,k} \mathbf{F}_{\mathrm{j+1-k}}, \tag{5.4}$$

onde

$$\beta^{\mathrm{M}}_{\nu,k} = \frac{1}{h_j} \int_{t_j}^{t_j+h_j} \prod_{m=0 \quad m\neq k}^{\nu-1} \left( \frac{x - t_{j+1-m}}{t_{j+1-k} - t_{j+1-m}} \right) dx. \tag{5.5}$$

Observando-se as Eqs. (5.3, 5.5) constata-se o motivo desses métodos terem sido originalmente empregados procurando-se minimizar a mudança do passo, visto que o cômputo dessas integrais era muito dispendioso. Atualmente, o cômputo dessas integrais não representa maiores problemas do ponto de vista de performance computacional. Entretanto, deve-se ainda evitar o cômputo dessas integrais, visto que, conforme o passo torna-se

pequeno, surgem problemas numéricos relacionados à erros de representação em ponto flutuante[7]. Quando se utiliza um passo fixo, os valores dos $\beta$'s dos dois métodos tornam-se constantes. Assim, a primeira solução visando permitir mudanças do passo utilizando essas constantes seria interpolar os valores já obtidos para cada novo passo. O problema deste método é termos que interpolar vetores relativamente grandes. A solução encontrada, implementada no programa desenvolvido, baseia-se na interpolação, permitindo o uso de uma malha com espaçamento variável. Contudo, neste caso, eliminamos a necessidade de trabalharmos com os vetores diretamente. Supondo ainda que tenhamos $\nu$ pontos previamente calculados, para dar um passo $h_j$ que permita utilizar os $\beta$'s constantes, temos que obter valores para os $\mathbf{F}$'s igualmente espaçados de $h_j$ entre si. Admitindo que tenhamos interpolado os $\mathbf{F}$'s, obtendo $\mathbf{F}^*$'s monoespaçados, poderíamos aplicá-los nas fórmulas para os métodos de Adams fazendo uso dos $\beta$'s constantes. Como cada um dos $\mathbf{F}^*$, anteriormente calculados, podem ser escritos em função dos $\mathbf{F}$'s utilizando-se interpolação[8] da forma $\mathbf{F}^*_{\mathrm{j+1-k}} = \sum_{p=1}^{\nu} \alpha_p \mathbf{F}_{\mathrm{j+1-p}}$, as fórmulas de Adams podem ser escritas diretamente em função dos $\mathbf{F}$'s, bastando obtermos os multiplicadores adequados, que dependem do espaçamento da malha, bem como dos $\beta$'s, constantes. Note que as fórmulas escolhidas para o preditor e corretor são de ordem 10 e 11, respectivamente. Para alterar o passo, devemos estimar o erro. Supondo que $\mathbf{y}_P$ e $\mathbf{y}_C$ sejam os valores obtidos com o preditor e o corretor respectivamente, temos que a estivativa do erro é da forma

$$est = MultTol\sqrt{\|\mathbf{y}_C - \mathbf{y}_P\|}, \tag{5.6}$$

onde o escalar $MultTol$, depende da ordem dos métodos. A partir da estimativa do erro,

[7]Uma versão inicial do método de solução implementava o cômputo dessas integrais a cada passo. Todavia, constatou-se o surgimento destes problemas conforme diminuia-se o passo.

[8]No programa desenvolvido é utilizado o polinômio interpolador de Lagrange.

podemos obter o novo passo a partir da fórmula mostrada abaixo

$$h_{j+1} = 0.9 \left| \frac{tol}{est} \right|^{\frac{1}{12}} h_j, \tag{5.7}$$

onde *tol* é a tolerância especificada pelo usuário e *est* é a estimativa do erro, mostrada na Eq. (5.6). Note que, caso $est > tol$, isto é, o passo deve ser rejeitado, o novo passo a ser tentado é, também, obtido com a Eq. (5.7).

No programa desenvolvido, o arquivo *Solver Dinâmico.f90* implementa as rotinas do *solver* dinâmico. Este arquivo possui, aproximadamente, 5000 linhas de programação. A função principal do *solver* dinâmico é denominada *Solver_Dinamico_AdamsEsp*. Esta função chama a rotina *IntegradorMerson* para preencher o *array* de Adams, que possui os termos anteriormente calculados. Esta rotina utiliza a fórmula de Merson, constituindo num dos métodos de Runge-Kutta (SHAMPINE, 1994). Como a fórmula de Merson possui ordem 4, é utilizada uma tolerância mais restritiva neste método, visando preencher o *array* de Adams com termos suficientes acurados, prevenindo a redução da ordem do método preditor-corretor.

Por fim, vamos apresentar uma tabela comparativa de alguns dos programas comerciais para análise de sistemas multicorpos. A Tab. (5.1) compara alguns desses programas. Na tabela apresentada, temos o nome do programa, seu país de origem, sua formulação, sua descrição, podendo ser absoluta ou relativa e os campos modelo e solução. O campo modelo refere-se ao modelo matemático representando as equações de movimento. Neste caso, EDA indica equação diferencial algébrica e EDO equação diferencial ordinária. Em todos os casos onde temos a situação EDO/EDA, significa que para sistemas de cadeias abertas teremos as equações de movimento representadas por um conjunto de equações

TABELA 5.1 – Comparação de alguns programas comerciais

| Nome Programa | Origem | Formulação | Descrição | Modelo | Solução |
|---|---|---|---|---|---|
| ADAMS | EUA | Lagrange | Absoluta | EDA | N |
| Autolev | EUA | Kane | Relativa | EDO/EDA | N/S |
| Recurdyn | Alemanha | Newton-Euler | Relativa | EDO/EDA | N |
| Simpack | Alemanha | Newton-Euler | Relativa | EDO/EDA | N |

diferenciais ordinárias, sendo os sistemas com cadeias fechadas representados matematicamente por um conjunto de equações diferenciais algébricas. O campo solução indica se o programa utiliza algoritmos numéricos ou simbólicos na formulação das equações de movimento. Na solução numérica as equações de movimento são formuladas a cada passo durante o processo de solução. Em contrapartida, na solução simbólica, o programa oferece ao usuário a formulação simbólica das equações de movimento, onde as equações de movimento são descritas, em geral, utilizando-se alguma linguagem de programação.

# 6 Implementação Computacional

Este capítulo destina-se a apresentar o programa desenvolvido. O intuito aqui não é listar o código computacional do programa, visto que este possui aproximadamente 37000 linhas de programação, o que tomaria cerca de 350 páginas em formato A4 na versão impressa. Entretanto, apresentaremos fragmentos de código computacional, importantes à compreensão da questão da interface gráfica, de forma a proporcionar uma visão mais detalhada do programa desenvolvido, visto que a grande deferença de programas baseados em eventos daqueles não possuidores de inteface gráfica é mais conceitual do que especificamente técnica.

## 6.1 Programa Computacional - Parte I

Nesta seção introduziremos os conceitos existentes por trás da implementação propriamente dita. Basicamente, um sistema multicorpos constitui-se num conjunto de corpos, juntas, sensores e atuadores. O programa desenvolvido permite a criação de corpos rígidos no formato de cones, esferas, paralelepípedos e, por fim, cilindros. Estes corpos por sua vez podem estar conectados por juntas revolutas, cilindricas, esféricas ou prismáticas.

O desenvolvimento de um programa de computador é muito semelhante ao projeto em engenharia. Primeiro, é necessário saber qual é o objetivo a ser alcançado com o projeto

ou, na linguagem de programação, quais são os requisitos do sistema, sintetizando *o que* este deverá ser capaz de fazer. Com os requisitos determinados, procura-se desenvolver sistemas que atendam a esses requisitos. Neste ponto, esses sistemas são representados não pelo código computacional, mas geralmente de uma forma gráfica, utilizando-se, por exemplo, dos diagramas da UML[1].

A linguagem UML, à semelhança das plantas de um prédio na construção civil, permite que se tenha um bom conhecimento do sistema sem que este seja implementado. Nesta analogia, os vários diagramas fornecidos pela UML podem ser comparados às plantas das partes hidráulica, elétrica, estrutural, dos dutos de refrigeração de um edifício, etc.... No caso da descrição de um sistema computacional, esses diagramas podem fornecer tanto uma visão estática quanto dinâmica do sistema a ser implementado. Embora cada um desses diagramas forneça uma visão alternativa do sistema, baseada numa abordagem específica deste, esses diagramas em conjunto possibilitam que se tenha uma visão integrada do sistema, contribuíndo com o difícil aspecto de ter de se gerenciar a complexidade de um sistema de grande porte, facilitando à implementação desses sistemas. Um ponto comum na maioria desses diagramas é a presença de objetos. Esses objetos provém do conceito de orientação à objetos (COAD; YOURDON, 1990), (COAD; YOURDON, 1992) e (COAD; YOURDON, 1993). Para se compreender a necessidade (e utilidade) da orientação a objetos, deve-se observar que, ao tentarmos implementar um sistema numa linguagem do programação, devemos fazer uso das possíveis definições de variáveis e das possíveis funções aplicáveis a essas variáveis visando mapearmos o domínio onde o sistema a ser desenvolvido está com aquele fornecido pela linguagem em uso. Assim, no caso do Fortran 77, uma linguagem arcaica porém ainda em uso hoje em dia, graças à enorme

[1]Unified Modelling Language. Mais informações podem ser encontradas em (BOOCH JAMES RUMBAUGH, 1998)

quantidade de código disponível nesta linguagem, deveríamos descrever um simulador da dinâmica de multicorpos utilizando-se números inteiros, em ponto flutuante, caracteres e arrays dessas variáveis. Note que juntas, corpos, atuadores, etc, deveríam ser mapeados (descritos) utilizando-se essas variáveis diretamente, o que é um grande incoveniente. De fato, o grande transtorno na abordagem de programação tradicional, antes da orientação a objetos, está no fato do mapeamento do mundo real, onde os sistemas geralmente residem, com aquele fornecido por tipos básicos de variáveis ser muito artificial. Com o uso de orientação à objetos podemos definir tipos de variáveis que sejam representações diretas dos entes encontrados no mundo real. Assim, podemos definir, por exemplo, a classe corpo, sendo cada objeto (variável) desta classe um corpo propriamente dito. Claro está que os membros desses objetos serão variáveis básicas ou outros objetos. Embora, em última instância, do ponto de vista de tipos de variáveis, os objetos serão definidos utilizando-se os tipos básicos disponíveis na linguagem, a orientação à objetos permite que esses dados sejam abstraídos do usuário[2]. De fato, numa implementação correta da orientação à objetos o usuário que utiliza um dado objeto não modifica os dados (estado) de um dado objeto diretamente.

A forma de interação entre o usuário e um dado objeto é definida pela interface, constituída por um conjunto de funções bem definidas. Assim, por exemplo, numa variável (objeto) do tipo corpo, é melhor para o usuário especificar a massa através de uma dada função do que definir um valor de uma variável. Para ser mais específico, caso o usuário tenha alocado memória para uma variável do tipo corpo, denominada de *varCorpo*, o usuário especificaria a massa por uma função, da forma *varCorpo.especificarMassa(massa)*, onde massa seria um valor em ponto flutuante. Para obter o valor da massa de um dado corpo

[2]Aqui o termo usuário refere-se ao usuário de uma dada classe. Este usuário é um programador que pode estar utilizando uma classe definida por ele ou por outra pessoa qualquer.

o usuário utilizaria a função *obterMassa* da forma *massa=varCorpo.obterMassa()*. Neste caso, massa será utilizada para armazenar o valor, em ponto flutuante, da massa do corpo. Note que a abordagem de objetos torna a forma de armazenagem dos dados (e a maneira como esses dados são processados) irrelevante ao usuário, o que é a grande vantagem da orientação à objetos, visto que o desenvolvedor das classes pode modificar a forma como esses dados são armazenados e processados sem influenciar as outras partes do sistema. Em suma, um objeto encapsula os dados, caracterizando seu estado e procedimentos, caracterizando seu comportamento, sendo que alguns desses procedimentos determinam sua interface.

O programa desenvolvido foi projetado dentro dos preceitos de orientação à objetos, onde são utilizadas classes modelando diretamente certos objetos do domínio da dinâmica de multicorpos. A Fig (6.1) mostra esquematicamente o programa desenvolvido.

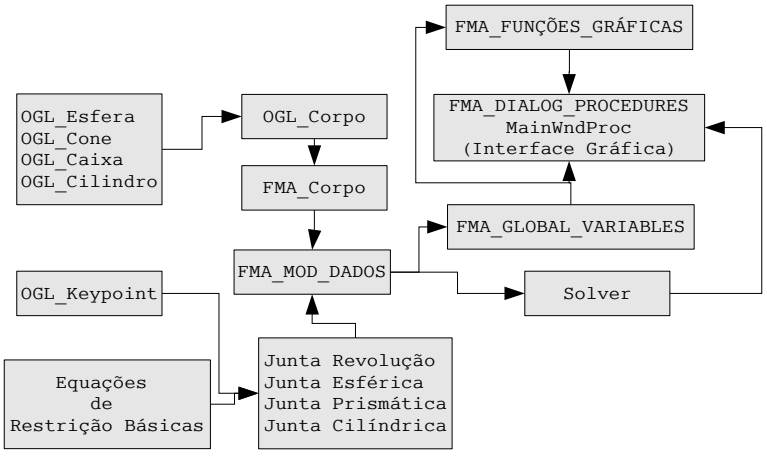



FIGURA 6.1 – Estrutura do programa desenvolvido.

Na Fig. (6.1) podemos identificar cinco classes que tem o nome com o prefixo OGL_. Este prefixo indica que essas classes implementam a parte gráfica do programa, sendo as

letras do prefixo indicativas do nome da biblioteca utilizada para contrução das imagens, denominada de OpenGL[3]. Essas classes fornecem toda funcionalidade necessária à visualização dos corpos e pontos. As classes OGL_Esfera, OGL_Cone, OGL_Caixa e OGL_Cilindro são utilizadas na classe OGL_Corpo, que define a parte gráfica para os corpos. As equações básicas de restrição fornecem a funcionalidade requerida para implementação das juntas mecânicas. Aqui é importante fazer uma diferenciação na composição de classes e objetos. Note que a equação de restrição de uma junta *possui* equações básicas de restrição (composição de objetos), ao passo que uma junta de revolução é *um tipo* de junta (composição de classe). Essas relações de posse e especialização são cruciais na representação da complexidade dos sistemas reais (ECKEL, 2000), (ECKEL, 1999). Objetivando permitir a inserção e retirada dos elementos do programa pelo usuário é necessário adotar algum tipo dinâmico de estrutura de dados. O mais simples desses tipos, sendo aquele adotado no programa, é a lista encadeada. Desta maneira, são definidas algumas dessas listas no programa, estando essas listas incumbidas de representar os corpos e as juntas. Essas definições estão em FMA_MOD_DADOS, responsável pela definição da estrutura de dados utilizada pelo programa. Essas estruturas de dados são utilizadas pelo *solver* e por FMA_Global_Variables, definindo as variáveis globais para interface gráfica. Como o *solver* é chamado pelo usuário, temos que FMA_Dialog_Procedures, caracterizando as rotinas para as caixas de diálogo e MainWndProc, representando a principal função de interface gráfica do programa, devem estar conectadas ao *solver*. Por fim, FMA_Funcoes_Graficas implementa as rotinas gráficas para todo o sistema.

Esta versão do programa simula corpos sujeitos à atração gravitacional, permitindo ainda a inserção de outras forças, através de sistemas massa-mola-amortecedor-atuador

[3]OpenGL provém de Open Graphics Library. O *site* oficial desta biblioteca é www.opengl.org.

lineares e angulares. As forças para os atuadores desses conjuntos podem ser descritas por meio de arquivos de funções Matlab. Também não há limitação quanto ao número de corpos. De fato, no próximo capítulo, destinado ao estudo de casos, será apresentado um sistema multicorpos possuindo uma cadeia fechada com um total de 30 corpos. Embora a inserção de forças constantes seja simples de ser implementada, a verdadeira importância na inserção de forças está relacionada com a questão do controle dos sistemas multicorpos, onde essas forças tornam-se funções, geralmente complexas, da posição e velocidade, estando normalmente modeladas em programas específicos, destinados para este fim. Esta implementação está condicionada não só à representação computacional das forças. É necessário desenvolver também todo um mecanismo de comunicação com algum programa de controle, como o Matlab por exemplo. O programa desenvolvido implementa uma mecanismo de comunicação com o Matlab, permitindo a modelagem adequada das forças relacionadas ao controle dos sistemas multicorpos. A Fig. (6.2) ilustra o programa em funcionamento, onde é mostrado um mecanismo de quatro barras.

Como dito anteriormente, as animações mostradas pelo programa são construídas utilizando-se a biblioteca gráfica OpenGL. Esta biblioteca possui funções básicas, destinadas ao desenho de entidades gráficas primitivas. Estas são o ponto, reta e polígono, sendo a curvatura dos objetos mais gerais obtida através de inúmeros polígonos. Na implementação OpenGL para Windows, ambiente onde o programa opera, não há funções para desenhar objetos mais gerais, como esferas por exemplo. Estes devem ser implementados pelo próprio programador. A biblioteca gráfica OpenGL também possui um mecanismo sofisticado de seleção, permitindo ao programador implementar rotinas viabilizando ao usuário selecionar os objetos diretamente com o *mouse*. Embora esta biblioteca tenha

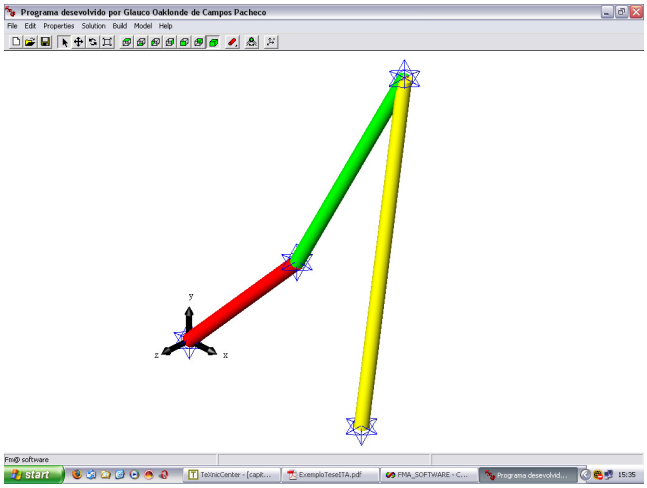



FIGURA 6.2 – Imagem ilustrando o programa em funcionamento.

sido escrita na linguagem de programação C, bem como a *API*[4] do sistema operacional Windows, o compilador adotado, o Compaq Visual Fortran, possui blocos de interface para todas essas funções. Logo, torna-se possível utilizar essas funções na linguagem Fortran, com o mínimo de alterações. Em (PETZOLD, 1998) existe uma das mais completas exposições da *API* do sistema operacional Windows. Sobre a biblioteca OpenGL, tem-se em (BOARD, 2003), uma apresentação bem detalhada, elaborada pelos próprios desenvolvedores desta biblioteca. Contudo, esta apresentação não prima por nenhum ambiente computacional em particular. Na verdade esses desenvolvedores implementaram

[4]O termo API significa Application Programming Interface, constituindo funções visando utilizar as funcionalidades do sistema operacional.

uma biblioteca OpenGL específica, independente de funções de qualquer sistema operacional, possuindo funções básicas para criação de janelas. Caso o programador deseje utilizar essas rotinas gráficas juntamente com as funções de um dado sistema operacional, é necessário saber implementar essas rotinas para esse sistema específico. Felizmente, no caso do Windows, o programador encontra em (LAWRENCE, 2002) uma boa fonte de informações.

Como vimos anteriormente, listas encadeadas representando certos objetos do sistema multicorpos foram definidas, onde cada objeto está definido em sua respectiva classe. Essas classes possuem basicamente dois tipos de rotinas. Existem aquelas relacionadas à simulação, como funções para obter o jacobiano, etc e outro tipo de funções, relacionadas à interface gráfica. Embora cada uma dessas classes implemente funcionalidades específicas, elas devem realizar, de um ponto de vista mais abstrato, muitas atividades semelhantes, como desenhar o respectivo objeto na tela. Devido ao tamanho do programa desenvolvido, torna-se necessária a utilização de técnicas de programação que facilitem sua elaboração. Como ressaltado em (KUNZ, April, 1997), a orientação a objetos é uma técnica de análise, projeto e programação que encaixa-se perfeitamente na análise de sistemas multicorpos. Embora a orientação a objetos esteja amplamente difundida, é raro encontrarmos programas em Fortran 90 implementando essas técnicas. Infelizmente, existe uma visão pejorativa de antiguidade em relação ao Fortran. Isto se deve em grande parte à enorme popularidade do Fortran 77. Este sim pode ser considerado como incapaz de implementar sem maiores transtornos técnicas de orientação a objetos. No caso do Fortran 90 ocorre o contrário. Esta versão do Fortran permite a implementação de inúmeras técnicas de orientação a objetos. Em (AKIN, 2003) o leitor pode encontrar a implementação de várias dessas técnicas. Implementar orientação a objetos no Fortran é importante, visto que

esta linguagem normalmente produz, para códigos numéricos, executáveis mais rápidos do que aqueles obtidos através da linguagem C++. Contudo, infelizmente, existe certa rivalidade entre defensores de uma ou outra linguagem. Na opinião do autor, o importante é saber que é possível implementar os conceitos de orientação a objetos em programas escritos em Fortran 90, o que permite, por exemplo, implementar as rotinas numéricas em Fortran e a interface gráfica em outra linguagem, dentro do contexto de orientação a objetos. De fato, a programação com múltiplas linguagem ganha mais importância a cada dia. Isto é confirmado observando-se a nova norma do Fortran, liberada ainda como rascunho, assegurando a interoperabilidade entre Fortran e C.

## 6.2 Programa Computacional - Parte II

Nesta parte vamos apresentar importantes rotinas relacionadas à interface gráfica do programa desenvolvido. Uma das características mais interessantes do programa desenvolvido é o fato deste ser multitarefa. Esta característica pode ser confirmada quando o programa está resolvendo numericamente as equações de movimento enquanto permite ao usuário a completa interação com a interface gráfica. Para entendermos melhor a vantagem de um programa multitarefa, temos que compreender o funcionamento conceitual de um programa Windows. Para tal, vamos apresentar como código computacional aquele que constitui o mínimo necessário à criação de uma janela no Windows. Este está apresentado a seguir:

```
%todo função programa window precisa registrar a classe de uma janela. A função WinMain é responsável pelo
%registro da classe da janela do aplicativo.
integer*4 function WinMain( hInstance, hPrevInstance, lpszCmdLine, nCmdShow )
%abaixo são dadas diretrizes para o compilador
!DEC$ IF DEFINED(_X86_)
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_WinMain@16' :: WinMain
!DEC$ ELSE
```

```
!DEC$ ATTRIBUTES STDCALL, ALIAS : 'WinMain' :: WinMain
!DEC$ ENDIF
    %o módulo dfwina possui blocos de interface para praticamente todas as API's do Windows.
    use dfwina
    use Variaveis_Globais
    implicit none
    integer*4 hInstance
    integer*4 hPrevInstance
    integer*4 lpszCmdLine
    integer*4 nCmdShow
    include 'dfghj.fi'
    %o tipo T_WNDCLASS será preenchido e utilizado para registrar a classe
    type (T_WNDCLASS)       wc
    %o tipo T_MSG é responsável pelo armazenamento das mensagens enviadas pelo sistema operacional
    type (T_MSG)            mesg
    integer*4               ret
    logical*4               lret
    integer                 haccel
    character(SIZEOFAPPNAME) lpszClassName
    character(SIZEOFAPPNAME) lpszIconName
    character(SIZEOFAPPNAME) lpszAppName
    character(SIZEOFAPPNAME) lpszMenuName
    character(SIZEOFAPPNAME) lpszAccelName
    ghInstance = hInstance
    ghModule = GetModuleHandle(NULL)
    ghwndMain = NULL
    lpszClassName ="dfghj"C
    lpszAppName ="dfghj"C
    lpszIconName ="dfghj"C
    lpszMenuName ="dfghj"C
    lpszAccelName ="dfghj"C
    !  If this is the first instance of the application, register the
    !  window class(es)
    if (hPrevInstance .eq. 0) then
        !  Main window
         %abaixo preenchemos o tipo wc
         wc%lpszClassName = LOC(lpszClassName)
         %o campo lpfnWndProc, preenchido abaixo é o mais importante, especificando a rotina responsável
         %pelo processamento das mensagens enviadas pelo sistema operacional
         wc%lpfnWndProc = LOC(MainWndProc)
         wc%style = IOR(CS_VREDRAW , CS_HREDRAW)
         wc%hInstance = hInstance
         wc%hIcon = LoadIcon( hInstance, LOC(lpszIconName))
         wc%hCursor = LoadCursor( NULL, IDC_ARROW )
         wc%hbrBackground = ( COLOR_WINDOW+1 )
         wc%lpszMenuName = NULL
         wc%cbClsExtra = 0
         wc%cbWndExtra = 0
         %registrando a janela
         if (RegisterClass(wc) == 0) goto 99999
```

```
    end if
    %após registrarmos a classe da janela, vamos criar uma janela baseada nesta classe
    ghwndMain = CreateWindowEx(  0, lpszClassName,                \&
                                 lpszAppName,                     \&
                                 INT(WS_OVERLAPPEDWINDOW),        \&
                                 CW_USEDEFAULT,                   \&
                                 0,                               \&
                                 CW_USEDEFAULT,                   \&
                                 0,                               \&
                                 NULL,                            \&
                                 ghMenu,                          \&
                                 hInstance,                       \&
                                 NULL                             \&
                               )
    if (ghwndMain == 0) goto 99999
    lret = ShowWindow( ghwndMain, nCmdShow )
    %agora processaremos as mensagens
    do while( GetMessage (mesg, NULL, 0, 0) ) %lendo o stack de mensagens gerado pelo sistema operacional
       if ( TranslateAccelerator (mesg%hwnd, haccel, mesg) == 0) then
           lret = TranslateMessage( mesg )
           ret  = DispatchMessage( mesg )
       end if
    end do
    WinMain = mesg.wParam
    return
99999 \&
    ret = MessageBox(ghwndMain, "Error initializing application dfghj"C, \&
                     "Error"C, MB_OK)
    WinMain = 0
end
!
!*******************************************************************************
!
%o endereço da função abaixo foi dado como o campo lpfnWndProc do tipo wc.
%Isto significa que TODAS as janelas criadas com a classe registrada terão suas
%mensagens enviadas para a rotina abaixo.
integer function MainWndProc ( hWnd, mesg, wParam, lParam )
!DEC$ IF DEFINED(_X86_)
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_MainWndProc@16' :: MainWndProc
!DEC$ ELSE
!DEC$ ATTRIBUTES STDCALL, ALIAS : 'MainWndProc' :: MainWndProc
!DEC$ ENDIF
    use dfwina
    use Variaveis_Globais
    implicit none
    integer*4 hWnd
    integer*4 mesg
    integer*4 wParam
    integer*4 lParam
    include 'resource.fd'
```

```
%variáveis
integer*4           ret
integer*4 :: mouseX, mouseY, mouseOldX, mouseOldY
logical :: Botao_Esquerdo_Pressionado , Zoom
character(SIZEOFAPPNAME)  lpszName, lpszHelpFileName, lpszContents, lpszMessage
character(SIZEOFAPPNAME)  lpszHeader
%procurando que mensagem foi enviada
select case ( mesg )
  case (WM_CREATE)
     Botao_Esquerdo_Pressionado = .false.
     Zoom = .false.
     MainWndProc = 0
     return
  case (WM_LBUTTONDOWN)
     mouseOldX = LOWORD(lParam)
     mouseOldY = HIWORD(lParam)
     Botao_Esquerdo_Pressionado = .true.
     MainWndProc = 0
     return
  case (WM_LBUTTONUP)
     Botao_Esquerdo_Pressionado = .false.
     MainWndProc = 0
     return
  case (WM_MOUSEMOVE)
     mouseX = LOWORD(lParam)
     mouseY = HIWORD(lParam)
     if (Botao_Esquerdo_Pressionado == .true. .AND. Zoom == .true. ) then
        call Zoom_Sub ( mouseX , mouseY , mouseOldX , mouseOldY )
        mouseOldX = mouseX
        mouseOldY = mouseY
        MainWndProc = 0
        return
  case (WM_DESTROY) %fechar o aplicativo
      call PostQuitMessage( 0 )
      MainWndProc = 0
      return
! WM_COMMAND: user command
  case (WM_COMMAND)
    select case ( IAND(wParam, 16\#ffff ) )
        case (IDM_EXIT)
            ret = SendMessage( hWnd, WM_CLOSE, 0, 0 )
            MainWndProc = 0
            return
        case (IDM_Zoom)
            Zoom = .true.
            MainWndProc = 0
            return
        %processamento padrão
        case DEFAULT
            MainWndProc = DefWindowProc( hWnd, mesg, wParam, lParam )
```

```
                return

        end select

    %processamento padrão para as outras mensagens

      case default

          MainWndProc = DefWindowProc( hWnd, mesg, wParam, lParam )

    end select

end
```

O código está todo comentado para facilitar sua compreensão. As implementações de interfaces gráficas distinguem-se daquelas de modo texto, mais comuns a engenheiros, no sentido que as primeiras devem permitir maior interação com o usuário. Os desenvolvedores dessas interfaces (e do código por trás delas) ao invés de tentar classificar o usuário, definiram a forma como este interage com o computador. Esta interação é representada por meio de eventos. Assim, por exemplo, quando o usuário movimenta o mouse, ocorre um evento. Esses eventos geram mensagens para os aplicativos. Dessa forma, um programa para interface gráfica deve responder e agir de acordo com as mensagens necessárias. Observando a função MainWndProc, mostrada acima, podemos observar que esta função é a responsável pelo processamento das mensagens. Para ambientar o leitor com essa nova perspectiva, foi inserido no código apresentado acima uma implementação rudimentar do efeito de zoom na tela onde o modelo multicorpos é desenhado. Do ponto de vista do usuário, em qualquer programa implementando este efeito, deve-se selecionar o zoom clicando no menu ou toolbar para então, com o mouse, clicar com o botão esquerdo na tela movendo-o, mantendo o botão pressionado, até obter o zoom desejado. Do ponto de vista do programa, este deverá primeiro permitir ao usuário a seleção do zoom. Isto é feito criando-se uma entrada no menu ou toolbar. Tanto a entrada no menu quanto o botão da toolbar vão ser associados, pelo programador, a uma constante inteira, que será a mensagem a ser enviada pelo sistema operacional ao programa quando o usuário selecionar algum desses itens. No caso do código apresentado, a mensagem é identificada

pela constante inteira IDM_ZOOM, que claramente deve ser diferente de todas criadas pelo programador ou reservadas para o sistema. Quando o usuário clica o botão esquerdo do mouse ou o movimenta, o sistema operacional envia as mensagens WM_LBUTTONDOWN e WM_MOUSEMOVE, respectivamente. A forma como o programa irá responder depende da funcionalidade que este deverá apresentar ao usuário. No nosso caso em estudo, quando o usuário clica com o botão esquerdo o programa armazena o ponto na tela onde o evento ocorreu e quando o usuário move o mouse o programa efetua o zoom, se necessário, chamando a subrotina Zoom_Sub. Note que o código apresentado é apenas um esboço de um código completo para tal fim, visto que, numa implementação mais completa, deve-se capturar o mouse, bem como, no caso do programa desenvolvido, deve-se alterar parâmetros nas matrizes de projeção definidas em OpenGL, tornando este código bem mais extenso. Contudo, o esboço apresentado possui a vantagem da simplicidade, sendo ainda capaz de reter o conceito da programação Windows através de sua API.

Agora, retornaremos à questão da necessidade do programa ser multitarefa. Quando uma mensagem deve ser enviada para um programa, esta não é enviada à função responsável pelo seu processamento diretamente. Essas mensagens são postas numa pilha de mensagens, sendo lidas *assim que possível*. Essa pilha é lida constantemente pelo loop implementado no fim da função WinMain, mostrada no código acima. Além do sistema operacional enviar mensagens à pilha de mensagens do programa, o sistema operacional também gerencia esta pilha. Assim, estando o programa ocupado com um dado processamento demorado (resolvendo as equações de movimento por exemplo) essas mensagens não serão lidas. Do ponto de vista do usuário, se este tentar realizar o zoom durante a simulação das equações de movimento, as mensagens não serão processadas. Naturalmente, o usuário continuará tentando realizar o zoom e o sistema operacional ao postar

as mensagens, verificará que nenhuma delas está sendo processada (nenhuma mensagem será retirada da pilha pelo programa), resultando numa condição que infelizmente é bem conhecida dos usuários Windows: a tela do aplicativo ficará toda branca e ao lado do nome do aplicativo será escrita a mensagem *The program is not responding*. Para evitar esta infeliz situação, deve-se permitir a execução multitarefa do programa. A entidade responsável pela execução de códigos computacionais denomina-se *thread*. Toda vez que um dado executável é inicializado, o sistema operacional cria um thread para o aplicativo. Visando permitir uma execução multitarefa, o programa deverá criar outros threads e associá-los com funções específicas. No nosso caso, o programa desenvolvido cria um thread para a função do solver, bem como para a função play, responsável pela animação. Dessa forma, a interface gráfica continua respondendo ao usuário enquanto realiza essas funções.

Embora as interfaces gráficas tenham uma vida relativamente longa, o código por trás dessas interfaces muda constantemente. De fato, ocorrem melhorias e certas modificações em algumas rotinas. Por exemplo, no caso do Windows, embora sua API permaneça, existem atualmente muitas funções obsoletas e outras tantas novas. Não podemos deixar de comentar também que um programa Windows hoje em dia é provavelmente escrito em C++, com o uso da *Microsoft Foundation Classes*, constituída de classes, escritas em C++, encapsulando funcionalidades específicas da interface gráfica. Contudo, ainda deve-se dominar o uso direto das API's, visto que essas classes utilizam as funções desta API diretamente. Assim, neste capítulo foi escolhido apresentar um programa básico permitindo ao leitor ter uma noção do conceito de programação Windows e foi apresentada a forma como o programa desenvolvido é estruturado, dando uma visão mais detalhada deste.

# 7 Estudo de Casos

Este capítulo divide-se em duas partes. A primeira delas trata da validação do programa desenvolvido, onde são analisados três casos. O primeiro baseia-se na simulação de pêndulos planos, com 2 até 10 corpos, permitindo a análise do método de solução desenvolvido no quesito complexidade computacional. No segundo caso é estudado um sistema, ainda plano, possuindo uma cadeia fechada. No último caso da primeira parte, abordaremos a importante questão dos métodos numéricos que preservam a energia mecânica dos sistemas conservativos, utilizando para tal um pêndulo duplo espacial. Em todos esses casos, os resultados obtidos serão comparados com aqueles fornecidos pelo programa comercial ADAMS. Em todos esses casos os corpos são cilindros com comprimento $l = 1m$ e raio $r = 0.025m$, sendo o material considerado como aço, com densidade $\rho = 7801Kg/m^3$. Na segunda parte deste capítulo trataremos de uma aplicação do programa desenvolvido a um robô com 3 graus de liberdade. Neste caso, utilizaremos o sistema de comunicação implementado no programa desenvolvido que permite a comunicação deste com o Matlab[1]. Assim, o programa desenvolvido fornece ao Matlab certas variáveis medidas, definidas pelo usuário, colocando-as no espaço de trabalho do Matlab, como também executa comandos Matlab para fornecer as forças necessárias ao controle de movimento do robô. Dessa forma, é possível gerar um arquivo .m implementando a lei de controle, no caso

[1]O Matlab é um produto da *Mathworks* (www.mathworks.com)

em estudo obtida a partir da téncica denominada Controle por Torque Computado. Por fim, cabe ressaltar que em todas as simulações efetuadas neste capítulo não foram realizadas qualquer animação para obter-se os tempos computacionais de solução das equações de movimento, visto que esses algoritmos de animação são complexos e tomam um considerável tempo computacional do tempo total de solução (cerca de 10-30%). O computador utilizado para todas as simulações foi um computador PC com processador Athlon XP 2000, 256MB de RAM, rodando Windows XP com Service Pack 1.

## 7.1 Parte I - Validação

### 7.1.1 Caso I.1 - Pêndulo

Nesta seção simularemos, por um tempo de 100 segundos, pêndulos planos com dois até 10 corpos. Esses pêndulos terão como condição inicial velocidade nula, estando alinhados com a horizontal. A Tabela. (7.1) ilustra os tempos em segundos para solução das equações de movimento para os dois programas. Como parâmetros da simulação, foi especificada uma tolerância de $10^{-7}$, utilizando-se, no caso do ADAMS, dois métodos de integração. O primeiro deles, denominado ABAM, implementa, à semelhança do programa desenvolvido, as fórmulas de Adams. O outro é o algoritmo de solução padrão do ADAMS, denominado de GSTIFF, consistindo no método Gear, onde foi utilizada a estabilização de índice (formulação SI2).

Observando-se a tabela, constata-se que o ADAMS foi, praticamente em todos os casos, 100% mais lento que o programa desenvolvido, quando utilizado o integrador GSTIFF. Embora o método ABAM seja mais eficiente, este também foi cerca de 50% mais lento

TABELA 7.1 – Tempos gastos na simulação do sistema (segundos).

| *Nº Corpos* | *Programa Desenvolvido* | *ADAMS ABAM* | *ADAMS GSTIFF* |
|---|---|---|---|
| 2 | 6.25 | 12.87 | 9.07 |
| 3 | 10.0 | 18.76 | 15.71 |
| 4 | 15.7 | 27.87 | 24.48 |
| 5 | 21.32 | 37.50 | 37.85 |
| 6 | 29.60 | 47.49 | 52.31 |
| 7 | 38.55 | 60.14 | 67.93 |
| 8 | 46.81 | 73.68 | 89.67 |
| 9 | 57.02 | 88.21 | 119.19 |
| 10 | 68.36 | 105.50 | 152.33 |

que o programa desenvolvido.

Embora o ADAMS possua formulações um pouco mais eficientes, como não estabilizar o índice (formulação SI3) no integrador GSTIFF, fica difícil para o *solver* obedecer a tolerância especificada com essa formulação, resultando, na prática, na interrupção do *solver* durante o processo de solução. Quando não houve tal interrupção, os resultados obtidos diferiram substancialmente daqueles fornecidos pelos outros integradores utilizados. A interrupção da integração no caso da formulação SI3 ocorre devido ao integrador escolher um passo pequeno visando satisfazer a tolerância especificada, tornando a matriz do jacobiano das equações de movimento singular. No caso da diferença nos resultados obtidos com a formulação SI3, essas diferenças se caracterizaram por saltos na solução. Esses saltos ocorrem devido à mudança freqüente do passo do integrador, visto que essas mudanças de passo provocam a perda de acurácia do método GSTIFF. A solução neste caso é utilizar a formulação SI2, estabilizando o índice das equações diferenciais-algébricas de movimento.

Outra alternativa seria utilizar o integrador WSTIFF, que não sofre perda de acurácia com mudanças de passo, visto que este método, ao contrário do GSTIFF, obtém os coeficientes do método de integração a cada mudança de passo (RAMPALLI, 2000). Contudo,

o integrador WSTIFF, embora mais estável que o GSTIFF, neste caso não forneceu resultados satisfatórios, principalmente para os pêndulos com um maior número de corpos. Isto não foi decorrência do método, mas sim de sua implementação, visto que o ADAMS, na formulação SI3, realiza análise de erro somente nas coordenadas, deixando suas derivadas sem qualquer tipo de controle explícito de erro. Quando utilizou-se tolerâncias maiores, menos restritivas, o ADAMS forneceu resultados significativamente diferentes daqueles obtidos com uma tolerância mais apertada, o que não ocorreu com o programa desenvolvido.

Deve-se notar ainda que o programa desenvolvido aloca memória dinamicamente, ao passo que o ADAMS aloca toda a memória necessária no início da simulação, onde foi especificado que os dois programas armazenariam os dados dos pêndulos a cada 0.025 segundo. Logo, neste aspecto, o programa desenvolvido possui uma desvantagem nos testes de comparação de eficiência. Para ilustrar melhor os resultados obtidos com a simulação, além do tempo gasto na solução das equações de movimento, vamos mostrar gráficos específicos para o pêndulo com dez corpos, obtidos para uma simulação de 10 segundos. A Fig. (7.1) ilustra o pêndulo com 10 corpos construído no programa desenvolvido (a) e no ADAMS (b), durante a simulação, no tempo $t = 6.125s$

No Anexo A estão apresentados os gráficos para o último corpo do pêndulo. Nesses gráficos são comparados os resultados obtidos com o programa desenvolvido com aqueles gerados utilizando-se o integrador ABAM. Nas Figs. (B.1-B.6) estão os gráficos para a coordenada $x$ e suas derivadas e $y$ e suas derivadas, respectivamente, onde a letra (a) indica os gráficos obtidos com o programa desenvolvido, enquanto a letra (b) indica esses mesmos gráficos, fornecidos pelo ADAMS.

As Figs. (B.7-B.8) ilustram, respectivamente, os gráficos obtidos para a velocidade

**(a)** **(b)**

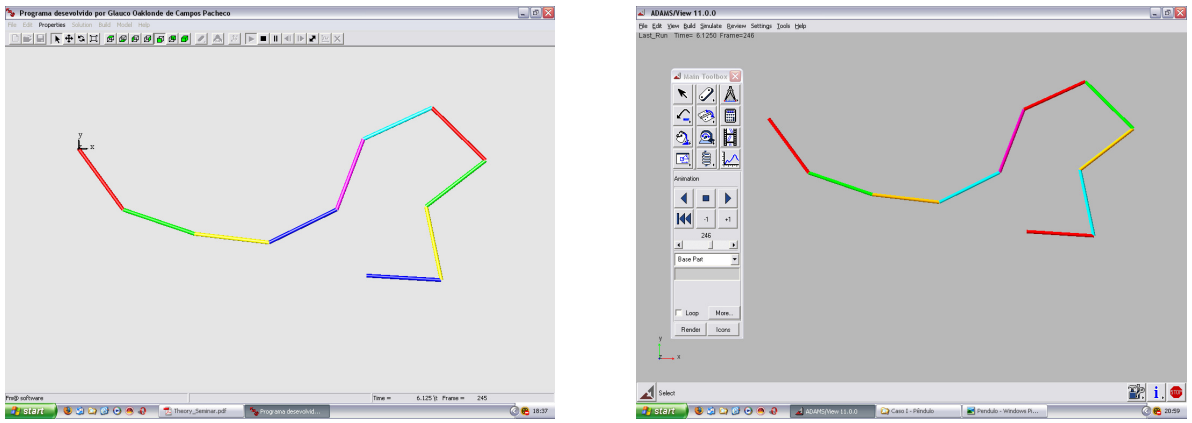


FIGURA 7.1 – Gráficos do pêndulo no instante $t = 6.125s$

e aceleração angulares do último corpo do pêndulo. As letra (a) indica os gráficos da velocidade e aceleração angulares para o programa desenvolvido, ao passo que a letra (b) indica esses gráficos para o ADAMS.

Por fim, temos na Fig (B.9) os gráficos do passo utilizado pelos programas utilizados. Nestas figuras, a letra (a) indica o resultado obtido com o programa desenvolvido, (b) o resultado fornecido pelo ADAMS com o integrador GSTIFF e a letra (c) ilustra o resultado obtido utilizando-se o integrador ABAM do ADAMS.

Observando-se os gráficos para o passo mostrados na Fig (B.9), verifica-se que o maior passo utilizado foi o do programa desenvolvido, o que representa uma grande vantagem, já que a tolerância especificada é cumprida com um maior passo. Deve-se notar que o método ABAM empregado no programa ADAMS implementa as fórmulas de Adams e possui ordem 12, como o programa desenvolvido. Contudo, no caso do ADAMS, o passo utilizado foi menor. Isto se deve ao fato do ADAMS utilizar aproximações numéricas para obter certos termos das equações de movimento. Essas aproximações são necessárias visto que o ADAMS utiliza, para descrever a orientação dos corpos, os ângulos de Eu-

ler, introduzindo, dessa forma, uma complexidade consideravelmente superior na geração dos termos necessários às análises cinemática e dinâmica como, por exemplo, o jacobiano das equações de restrição[2]. Dessa forma, essas aproximações diminuem a ordem de convergência dos integradores do ADAMS. O passo utilizado no caso do integrador GSTIFF foi bem menor, visto que este integrador implementa as fórmulas de Gear, baseadas no método BDF, sendo muito estáveis para baixas ordens, permitindo a integração numérica de sistemas rígidos. Contudo, essas fórmulas são instáveis para ordens maiores que 7 (SHAMPINE, 1994). Como a ordem do método GSTIFF é menor, este é menos acurado e deve tomar um menor passo visando satisfazer as tolerâncias especificadas, explicando o resultado obtido. Observando-se os gráficos obtidos, constata-se, neste caso em estudo, que o programa desenvolvido forneceu excelentes resultados, sendo ainda muito mais eficiente que o ADAMS.

### 7.1.2 Caso I.2 - Cadeia Fechada

O segundo caso estuda um sistema plano possuindo uma cadeia fechada, com um total de 30 corpos e 28 graus de liberdade. Este exemplo permite aplicarmos o programa desenvolvido a sistemas com equações redundantes. De fato, o sistema em estudo apresenta três equações redundantes. O sistema será simulado por 20 segundos com uma tolerância padrão igual a do caso I, isto é, $10^{-7}$. Neste caso utilizaremos praticamente todos os integradores disponíveis no ADAMS, de forma a permitir uma ampla comparação de resultados. Foram realizadas 5 simulações com o ADAMS. As duas primeiras utilizam o integrador ABAM. Contudo, na segunda é especificada, excepcionalmente, uma tolerância mais restritiva, no caso de $10^{-12}$. Neste caso, denominaremos este integrador da forma

[2]Note que no programa desenvolvido estes termos são obtidos diretamente a partir de fórmulas pré-determinadas, graças ao uso dos parâmetros de Euler.

TABELA 7.2 – Tempos gastos na simulação do sistema

| *Programa Desenvolvido* | *ADAMS ABAM* | *ADAMS ABAM*$^{\dagger}$ | *ADAMS* $GSTIFF - SI3$ | *ADAMS* $GSTIFF - SI2$ | *ADAMS* $WSTIFF - SI3$ |
|---|---|---|---|---|---|
| 35.96 | 73.71 | 126.17 | 85.79 | 271.53 | 62.94 |

TABELA 7.3 – Correspondência dos integradores

| *Programa Desenvolvido* | *ADAMS ABAM*$^{\dagger}$ | *ADAMS ABAM* | *ADAMS* $GSTIFF - SI2$ | *ADAMS* $GSTIFF - SI3$ | *ADAMS* $WSTIFF - SI3$ |
|---|---|---|---|---|---|
| (a) | (b) | (c) | (d) | (e) | (f) |

*ABAM*$^{\dagger}$, para diferenciá-lo. A terceira e quarta simulações foram conduzidas escolhendo-se o integrador padrão GSTIFF, onde na quarta simulação foi utilizada a estabilização de índice (formulação SI2). A quinta e última simulação foi obtida com o integrador WSTIFF, sem qualquer estabilização de índice[3]. A Tabela (7.2) ilustra os tempos obtidos nas simulações. Como condição inicial foi dada velocidade nula com a cadeia aberta na horizontal.

Observando-se a Tabela (7.2), constata-se que o programa desenvolvido continuou sendo o mais rápido nesta simulação, sendo novamente cerca de 90% mais que rápido que o integrador mais eficiente do ADAMS. Nas figuras que se seguem nesta seção, os resultados obtidos com cada integrador serão identificados por letras. A Tabela (7.3) mostra essa correspondência

Para compararmos os resultados obtidos, vamos mostrar gráficos para as variáveis $x$, $\dot{x}$, $\ddot{x}$, $y$, $\dot{y}$, $\ddot{y}$, a velocidade angular, aceleração angular para o último corpo da cadeia aberta (corpo na cor abóbora no ADAMS) e, finalmente, o passo utilizado nas integrações numéricas. As Figs. (C.1-C.9), apresentadas no Anexo B, mostram esses gráficos na ordem citada. A Fig. (7.2), mostrada neste capítulo, ilustra o sistema em questão no

[3]A versão utilizada do ADAMS ainda não suportava a estabilização de índice para o integrador WSTIFF.

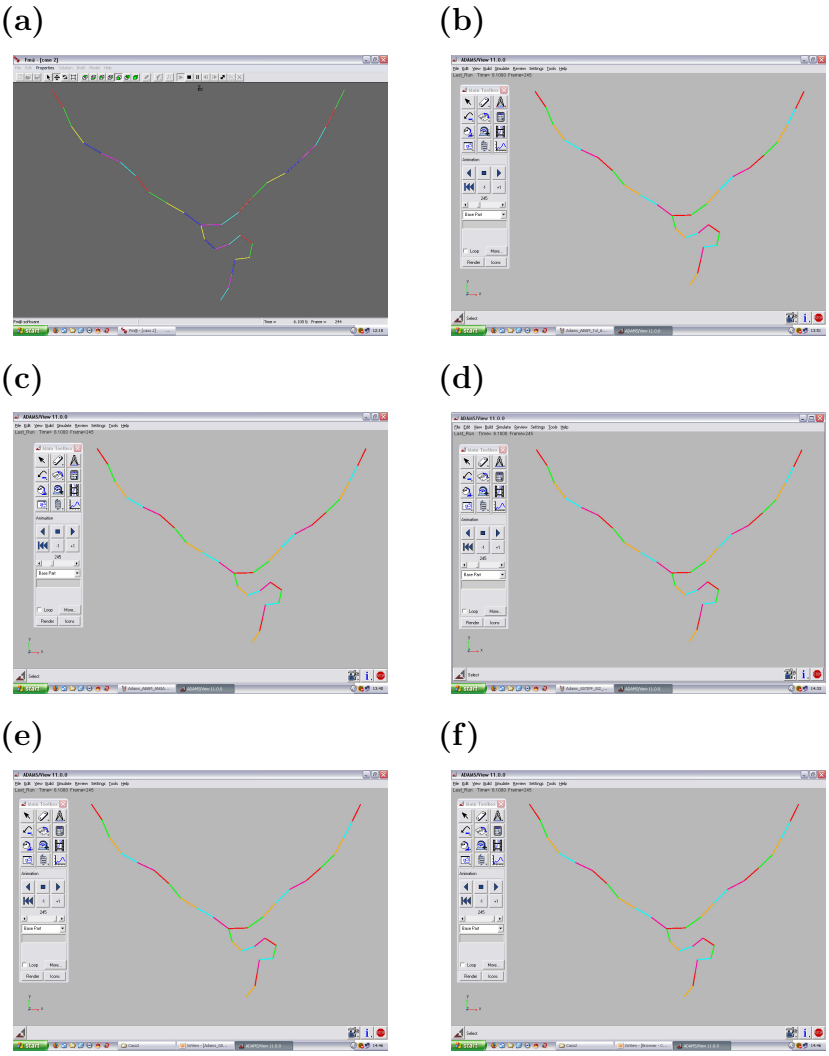



FIGURA 7.2 – Modelo durante animação no tempo $t = 6.1s$

tempo $t = 6.1s$

Observando-se os resultados obtidos, verifica-se que o ADAMS forneceu resultados significativamente diferentes entre si de acordo com os integradores utilizados. Observando-se o resultado mais acurado gerado pelo ADAMS, obtido com o integrador ABAM, com tolerância $10^{-12}$, identificado nos gráficos pela letra (b), vemos que este resultado assemelha-se bem com aquele obtido com o programa desenvolvido. Na análise dos gráficos, deve-se levar em consideração a complexidade do modelo simulado que embora seja plano, possui 30 corpos.

Uma observação interessante deve-se aos resultados mostrados nas letras (e) e (f). Estas letras indicam os integradores GSTIFF e WSTIFF, respectivamente, ambos na formulação SI3 e sem estabilização de índice. O interessante é que a única diferença entre esses integradores resulta que, no caso do WSTIFF, os coeficientes da fórmula são calculados a cada mudança de passo. Isso acarreta num aumento de complexidade do preditor deste método.

Contudo, embora o integrador WSTIFF seja computacionalmente mais dispendioso, este foi cerca de 27% mais rápido que o GSTIFF. Isto se deve ao fato do preditor do integrador WSTIFF ser mais acurado, acomodando sem problemas as mudanças de passo. No caso do integrador GSTIFF, como seu preditor perde acurácia conforme muda-se o passo[4], isto força o integrador a utilizar um menor passo, visando atender a tolerância especificada.

Por fim, nota-se que os resultados obtidos com o programa desenvolvido estão de acordo com aqueles fornecidos pelo ADAMS. Este caso abordou um sistema de cadeia fechada, onde existem equações redundantes, servindo para mostrar que o *solver* implementado no programa desenvolvido é tolerante a tais equações.

### 7.1.3 Caso I.3 - Sistema Tridimensional

O último caso aborda um sistema tridimensional. Este constitui-se num pêndulo com 2 corpos conectados por juntas esféricas. A Fig. (7.3) ilustra o sistema na sua configuração inicial, onde é especificada velocidade nula.

Este modelo permitirá compararmos os resultados obtidos pelo programa desenvolvido com aqueles fornecidos pelo ADAMS a fim de conhecermos melhor um importante

[4]Tecnicamente, diz-se que a ordem de convergência do método fica reduzida.

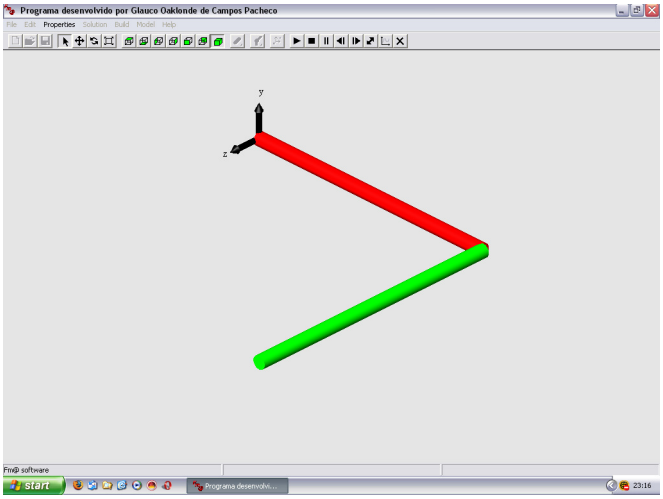



FIGURA 7.3 – Pêndulo em sua configuração inicial

aspecto dos integradores: a conservação de energia. Como o pêndulo em estudo é um sistema conservativo, sua energia mecânica total deverá ser constante. Neste caso, utilizaremos somente o integrador GSTIFF do ADAMS na formulação SI2, com estabilização de índice, visto que, somente neste caso, o ADAMS realiza análise de erro nas velocidades. Também serão mostrados, à semelhança do caso anterior, gráficos para as coordenadas do centro de massa, da velocidade e aceleração angulares e do passo. Os gráficos referentes às coordenadas e velocidade e aceleração angulares são referentes ao último corpo do pêndulo, mostrado em verde na Fig (7.3). Todos esses gráficos estão dispostos no Anexo C nas Figs (D.1-D.17), onde a letra (a) indica o programa desenvolvido e a letra (b) indica os resultados obtidos com o ADAMS. Como pode-se constatar a partir da Fig. (D.17), o programa desenvolvido fornece uma solução que, além de satisfazer as restrições, também assegura a conservação de energia, o que não ocorre com o ADAMS. Embora apresentemos somente gráficos para o integrador GSTIFF (SI2), todos os integradores disponíveis no ADAMS foram testados e nenhum forneceu uma solução satisfatória em termos da conservação de energia.

## 7.2 Parte II - Aplicação

Nesta seção abordaremos um problema de robótica, estudando um robô com três graus de liberdade rotacionais fixo a um referencial inercial. A Fig. (7.4) ilustra o robô

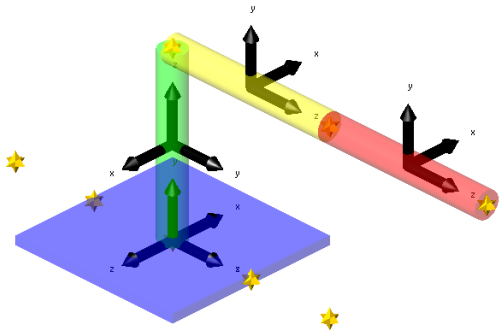


FIGURA 7.4 – Robô montado de três graus de liberdade.

O robô possuirá um atuador para cada uma de suas três juntas. Para descrevermos o movimento do robô, utilizaremos um sistema coordenado com direção $x$ coincidente com o eixo de simetria do último cilindro em vermelho e eixo $y$ vertical. O movimento do sistema ocorre da seguinte forma: o primeiro elo (em verde) pode girar, em relação à base fixa, em torno do eixo $y$, e os dois últimos elos podem girar em torno do eixo $z$. Note que o programa implementado utiliza coordenadas absolutas para descrever o movimento de cada parte do robô. Desta forma, poderiam ocorrer problemas relativos à questão de controle, já que este utiliza o espaço das juntas. Contudo, visando facilitar a questão do controle, foram introduzidos os conjuntos linear e angular de mola-amortecedor-atuador. Assim, esses conjuntos definem automaticamente variáveis lineares a angulares para os conjuntos linear e angular, respectivamente, logo que esses são criados. Isto torna a questão de controle independente das coordenadas utilizadas internamente pelo programa, restando ao usuário a tarefa de selecionar vetores e pontos diretamente na tela do computador para

gerar esses conjuntos. Logo, dependendo do conjunto escolhido, o usuário define forças e torques em função dessas coordenadas (deslocamentos e ângulos) e o programa elaborado relaciona essas coordenadas com as absolutas. A formulação para a mola e amortecedor pode ser encontrada em qualquer livro de mecânica computacional (SHABANA, 2001), (NIKRAVESH, 1988).

Assim, resta a questão da introdução das expressões definindo as forças/torques dos atuadores. Essas podem ser *qualquer* comando que o usuário possa introduzir diretamente na janela de comandos do Matlab. Esses comandos podem ser inclusive o nome de funções ou arquivos de comandos Matlab (.m), contanto que essas funções estejam no caminho de busca do Matlab. No exemplo do robô em estudo, serão utilizadas funções Matlab para obter os torques dos atuadores. Cabe ressaltar que antes de executar esses comandos (funções), o programa desenvolvido insere todas as variáveis dos conjuntos mola-amortecedor-atuador no espaço de trabalho Matlab, permitindo que a expressão para o torque (força) de um dado conjunto possa depender de outras variáveis, condição necessária à implementação de controladores. Do exposto, fica claro que o usuário necessita ter instalado o Matlab em sua máquina para poder fazer uso de suas funções[5].

Visando obter as expressões dos torques para os atuadores, devemos primeiro gerar um modelo dinâmico do robô descrito em função das coodenadas das juntas. Como o programa desenvolvido não realiza essa função, deveríamos gerar esse modelo manualmente. Contudo, como a geração manual das equações de movimento é propensa a erros, vamos fornecer um arquivo de comandos Maple[6] para realizar tal função. O método de funcionamento é muito simples, bastando ao usuário escolher quais coordenadas serão utilizadas na descrição do sistema multicorpos em estudo e fornecer, em relação ao referencial iner-

[5]A versão utilizada neste trabalho é a 7.0 para Windows.
[6]O Maple é um produto da Maplesoft (www.maplesoft.com).

cial, a posição do centro de massa de cada um dos corpos, onde esse vetor posição deve necessariamente estar descrito numa base de vetores fixa neste referencial inercial, bem como os componentes do vetor velocidade angular expresso na mesma base, fixa ao corpo, utilizada na representação matricial do tensor de inércia. Os comandos implementados no arquivo geram, utilizando as equações de Lagrange, as equações de movimento a partir dessa descrição, restando ao usuário a inserção de forças/torques atuantes no modelo, tarefa em geral bem mais simples que a formulação manual das equações de movimento. Este programa está ilustrado no apêndice A, seção *Formulação Simbólica*. As equações de movimento fornecidas pelo programa estão ilustradas abaixo nas Figs. (7.5-7.6)

Equação de movimento para phi(t)

$$\frac{1}{4}\Big(\big((2\,m\,l_4{}^2-8\,Ixx+8\,Iyy)\cos(\theta_2(t))^2+4\,m\,l_3\,l_4\cos(\theta_2(t))+5\,m\,l_3{}^2-m\,l_4{}^2\big)\cos(\theta_1(t))^2$$
$$-2\,\big((m\,l_4{}^2-4\,Ixx+4\,Iyy)\cos(\theta_2(t))+2\,m\,l_3\,l_4\big)\sin(\theta_1(t))\sin(\theta_2(t))\cos(\theta_1(t))+(-4\,Iyy+4\,Ixx-m\,l_4{}^2)\cos(\theta_2(t))^2+8\,Ixx+m\,l_4{}^2+4\,Iyy\big)$$
$$\left(\frac{d^2}{dt^2}\phi(t)\right)-\left(\frac{d}{dt}\phi(t)\right)\Big(\Big(\big((m\,l_4{}^2-4\,Ixx+4\,Iyy)\cos(\theta_2(t))+m\,l_3\,l_4\big)\sin(\theta_2(t))\cos(\theta_1(t))^2$$
$$+\sin(\theta_1(t))\left(\left(m\,l_4{}^2-4\,Ixx+4\,Iyy\right)\cos(\theta_2(t))^2+m\,l_3\,l_4\cos(\theta_2(t))+2\,Ixx-2\,Iyy-\frac{1}{2}m\,l_4{}^2\right)\cos(\theta_1(t))$$
$$-\frac{1}{2}\cos(\theta_2(t))\sin(\theta_2(t))\left(m\,l_4{}^2-4\,Ixx+4\,Iyy\right)\Big)\left(\frac{d}{dt}\theta_2(t)\right)+\left(\frac{d}{dt}\theta_1(t)\right)\Big(\big((m\,l_4{}^2-4\,Ixx+4\,Iyy)\cos(\theta_2(t))+2\,m\,l_3\,l_4\big)\sin(\theta_2(t))\cos(\theta_1(t))^2$$
$$+\left(\left(m\,l_4{}^2-4\,Ixx+4\,Iyy\right)\cos(\theta_2(t))^2+2\,m\,l_3\,l_4\cos(\theta_2(t))-\frac{1}{2}m\,(-5\,l_3{}^2+l_4{}^2)\right)\sin(\theta_1(t))\cos(\theta_1(t))$$
$$-\frac{1}{2}\big((m\,l_4{}^2-4\,Ixx+4\,Iyy)\cos(\theta_2(t))+2\,m\,l_3\,l_4\big)\sin(\theta_2(t))\Big)\Big)=0$$

FIGURA 7.5 – Equações de movimento descritas no espaço das juntas.

onde $\phi(t)$, $\theta_1(t)$ e $\theta_2(t)$ representam o ângulo de rotação do elo 1 em relação à base fixa, o ângulo de rotação do elo 2 em relação ao elo 1 e, finalmente, o ângulo de rotação do elo 3 em relação ao elo 2, nesta ordem. A partir deste modelo é simples contruirmos a lei de controle por torque computado. Escrevendo as equações de movimento no espaço das juntas dadas acima da forma

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = \tau \tag{7.1}$$

podemos esquematizar o controle por torque computado da forma ilustrada na Fig. (7.7)

Equação de movimento para theta[1](t)

$$\frac{1}{4}(m\,l_4{}^2+5\,m\,l_3{}^2+4\,m\,l_3\,l_4\cos(\theta_2(t))+8\,Izz)\left(\frac{d^2}{dt^2}\theta_1(t)\right)+\frac{1}{4}(2\,m\,l_3\,l_4\cos(\theta_2(t))+4\,Izz+m\,l_4{}^2)\left(\frac{d^2}{dt^2}\theta_2(t)\right)+\frac{1}{4}($$
$$2\sin(\theta_1(t))\cos(\theta_1(t))(m\,l_4{}^2-4\,Ixx+4\,Iyy)\cos(\theta_2(t))^2$$
$$+(2\sin(\theta_2(t))(m\,l_4{}^2-4\,Ixx+4\,Iyy)\cos(\theta_1(t))^2+4\,m\,l_3\cos(\theta_1(t))\,l_4\sin(\theta_1(t))-\sin(\theta_2(t))(m\,l_4{}^2-4\,Ixx+4\,Iyy))\cos(\theta_2(t))$$
$$-m\,(-4\,l_3\cos(\theta_1(t))^2\,l_4\sin(\theta_2(t))+\sin(\theta_1(t))(-5\,l_3{}^2+l_4{}^2)\cos(\theta_1(t))+2\,l_3\,l_4\sin(\theta_2(t))))\left(\frac{d}{dt}\phi(t)\right)^2-\frac{1}{2}$$
$$\left(l_3\,l_4\sin(\theta_2(t))\left(\frac{d}{dt}\theta_2(t)\right)^2+2\,l_3\left(\frac{d}{dt}\theta_1(t)\right)l_4\sin(\theta_2(t))\left(\frac{d}{dt}\theta_2(t)\right)+g\,(-3\,l_3\cos(\theta_1(t))-l_4\cos(\theta_1(t))\cos(\theta_2(t))+l_4\sin(\theta_1(t))\sin(\theta_2(t)))\right)$$
$$m=0$$

Equação de movimento para theta[2](t)

$$\frac{1}{4}(2\,m\,l_3\,l_4\cos(\theta_2(t))+4\,Izz+m\,l_4{}^2)\left(\frac{d^2}{dt^2}\theta_1(t)\right)+\frac{1}{4}(4\,Izz+m\,l_4{}^2)\left(\frac{d^2}{dt^2}\theta_2(t)\right)+\frac{1}{4}(2\sin(\theta_1(t))\cos(\theta_1(t))(m\,l_4{}^2-4\,Ixx+4\,Iyy)\cos(\theta_2(t))^2$$
$$+(2\sin(\theta_2(t))(m\,l_4{}^2-4\,Ixx+4\,Iyy)\cos(\theta_1(t))^2+2\,m\,l_3\cos(\theta_1(t))\,l_4\sin(\theta_1(t))-\sin(\theta_2(t))(m\,l_4{}^2-4\,Ixx+4\,Iyy))\cos(\theta_2(t))$$
$$-(-2\,m\,l_3\,l_4\sin(\theta_2(t))\cos(\theta_1(t))+\sin(\theta_1(t))(m\,l_4{}^2-4\,Ixx+4\,Iyy))\cos(\theta_1(t)))\left(\frac{d}{dt}\phi(t)\right)^2$$
$$-\frac{1}{2}\left(-l_3\left(\frac{d}{dt}\theta_1(t)\right)^2\sin(\theta_2(t))+g\,(-\cos(\theta_1(t))\cos(\theta_2(t))+\sin(\theta_1(t))\sin(\theta_2(t)))\right)m\,l_4=0$$

FIGURA 7.6 – Equações de movimento descritas no espaço das juntas.

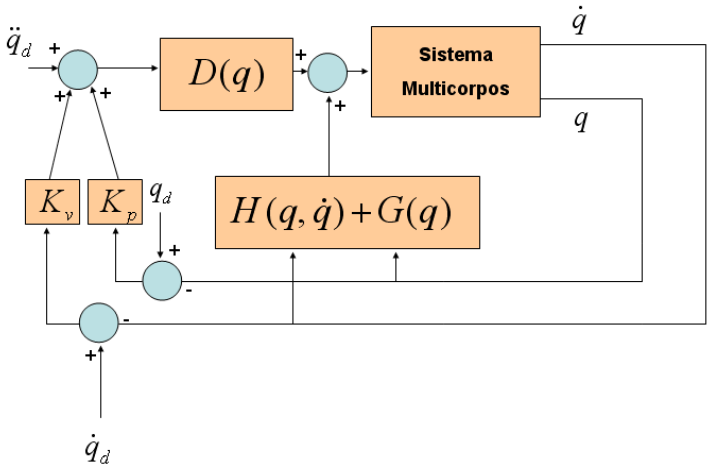



FIGURA 7.7 – Esquematização do controle por torque computado.

O relacionamento dos termos **D**, **H** e **G**, caracterizando as forças de inércia, os efeitos quadráticos na velocidade (Coriolis e centrífugas) e a atração gravitacional, respectivamente, com as equações de movimento no espaço das juntas é imediato. Vamos, portanto, apresentar diretamente as leis de controle através das funções *Torque_Robo_Phi.m*, *Torque_Robo_theta1.m* e *Torque_Robo_theta2.m*, todos mostrados na última seção do apêndice A. Desses arquivos, vê-se que cada parte possui massa igual a 8.82Kg e comprimento de 0.4m. Como essas partes possuem raio igual a 0.03m, obtemos os momentos de inércia

indicados nos arquivos. O movimento desejado para o robô consiste na prescrição dos ângulos das juntas de acordo com as seguintes funções

$$\phi(t) = 30.0 * (1.0 - e^{-1.3158*t}) \tag{7.2}$$

$$\theta_1(t) = 35.0 * (1.0 - e^{-1.3158*t}) \tag{7.3}$$

$$\theta_2(t) = 10.0 * (1.0 - e^{-1.3158*t}) \tag{7.4}$$

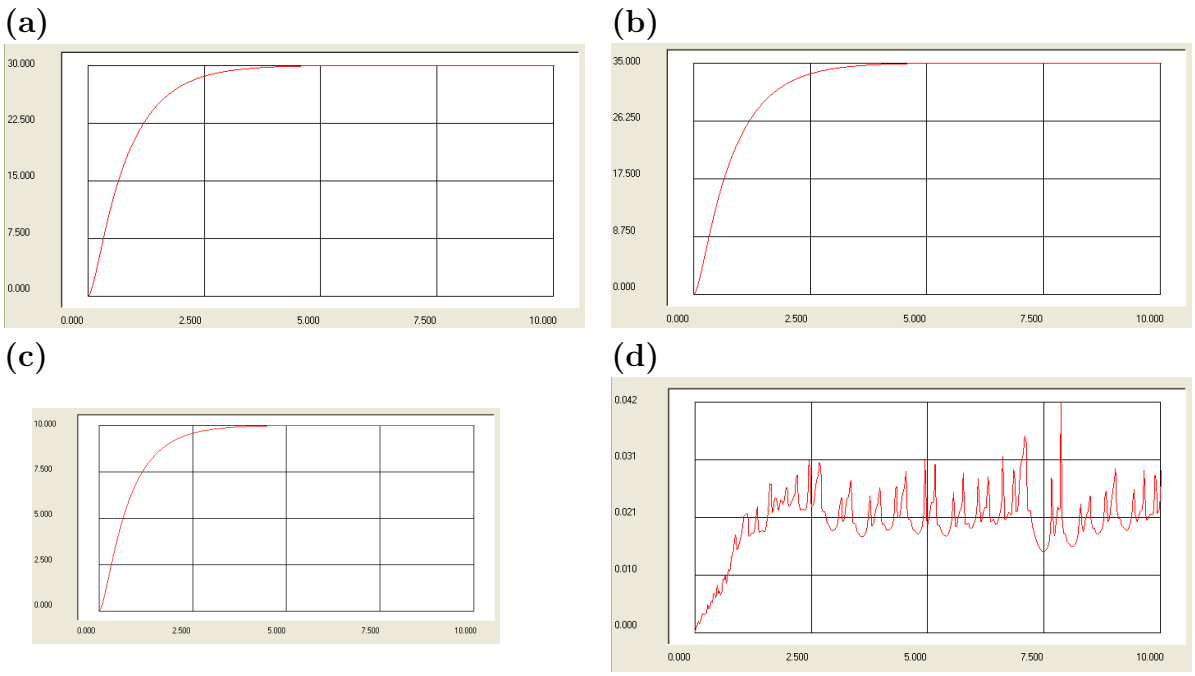



FIGURA 7.8 – Gráficos para $\phi$, $\theta_1$, $\theta_2$ e para o passo do integrador.

Os resultados obtidos para os ângulos das juntas, bem como o passo utilizado pelo integrador estão mostrados na Fig. (7.8), onde as letras (a), (b), (c) e (d) indicam os ângulos $\phi$, $\theta_1$, $\theta_2$ e o passo do integrador, nesta ordem. A Fig. (7.9) ilustra o robô durante quatro momentos distintos ao longo de seu movimento.

Observando-se os resultados obtidos, constata-se que estes se assemelham com o de-

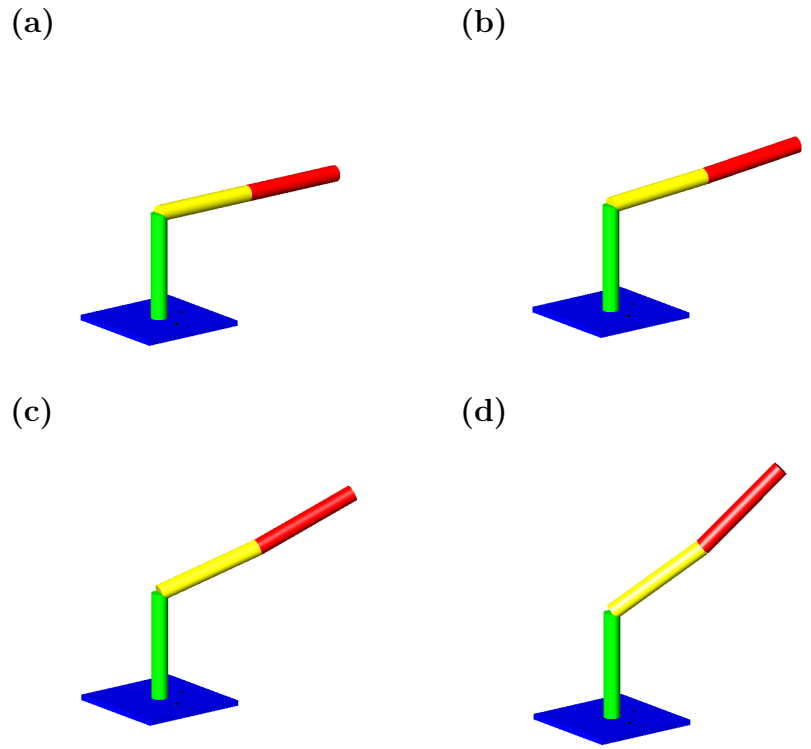



FIGURA 7.9 – Figuras ilustrando o movimento do robô até sua posição final.

sejado. Contudo, o controlador implementado, através da técnica de torque computado poderá não fornecer resultados satisfatórios em aplicações reais, onde os parâmetros do sistema sofrem inevitáveis variações, bem como existem variações durante a vida útil de operação desses robôs, forçando a utilização de técnicas de controle robustas, adaptativas,....

Entretanto, o presente exemplo serve para ressaltar a importante funcionalidade do programa desenvolvido pelo autor, que é sua capacidade de comunicação com o Matlab, abrindo caminho para a possibilidade de implementação de controladores mais sofisticados.

# 8 Conclusão

Nesta dissertação foi estudada a dinâmica computacional de sistemas multicorpos rígidos. Do ponto de vista teórico, foi apresentada a técnica de formulação das equações de movimento baseada nas coordenadas absolutas, sendo mostrados também alguns métodos para solução numérica dessas equações. Neste contexto, foi apresentado um aperfeiçoamento de um método existente baseado em ortogonalização, caracterizando a contribuição científica deste trabalho.

Do ponto de vista prático, foi implementado um programa computacional para simulação de sistemas multicorpos rígidos. O programa desenvolvido apresenta uma elevada abrangência, sendo capaz de simular uma ampla classe de sistemas rígidos holonômicos espaciais com qualquer número de cadeias fechadas. Dentre as limitações do programa, podemos citar a incapacidade deste em simular situações de contato de qualquer tipo. Assim, condições de contato sem impacto (rolamento) não são possíveis de serem simuladas, bem como situações de impacto mais gerais.

No programa elaborado, a interface com o usuário é bastante amigável, seguindo o padrão das aplicações para Windows, onde o programa opera. Esta interface permite ao usuário a seleção de corpos diretamente na tela, como também a realização de animações (inclusive durante a própria simulação) dos sistemas multicorpos, utilizando-se para tal

da biblioteca gráfica OpenGL. Dessa forma, essas animações tiram proveito do *hardware* gráfico disponível. Nos testes realizados para validação, o programa desenvolvido mostrou-se muito eficiente e acurado, fornecendo resultados semelhantes ao programa comercial ADAMS.

Do ponto de vista das aplicações, o programa implementado permite a simulação de sistemas rígidos holonômicos, em situações sem impacto, onde as forças (torques) aplicadas podem ser funções das coordenadas, suas derivadas e de ângulos/distâncias (e suas derivadas) definidos pelo usuário, com o auxílio de sensores, onde essas forças podem ser funções MATLAB tendo, portanto, toda a funcionalidade deste aplicativo, como funções da toolbox de controle, por exemplo. Dessa forma, é possível simular robôs, máquinas e mecanismos em geral possuindo controladores. Logo, embora o intuito original deste trabalho seja, além do estudo teórico dos sistemas multicorpos, desenvolver mais uma ferramenta computacional de auxílio na área de robótica do Instituto, pode-se verificar que a abrangência do programa desenvolvido, a partir dos estudos teóricos realizados, está além daquela contemplada pela robótica. Por fim, deve-se ressaltar que o programa desenvolvido pelo autor será liberado de acordo com a licença GPL[1], assegurando que *qualquer* pessoa possa modificar e redistribuir livremente o programa desenvolvido.

[1]GPL significa General Public License, utilizada em programas ditos livres. Mais informações sobre a licença podem ser encontradas em *http://www.gnu.org/copyleft/gpl.html*

# Referências Bibliográficas


AKIN, E. **Object-Oriented Programming via Fortran 90/95**. 1. ed. Cambridge: Cambridge University Press, 2003.

BAE, D.-S.; HAUG, E. E. A recursive formulation for constrained mechanical systems: Part i. open loop systems. **Mechanical Structures and Machines**, v. 15, n. 3, p. 359–382, 1987.

BAE, D.-S.; HAUG, E. E. A recursive formulation for constrained mechanical systems: Part i. closed loop systems. **Mechanical Structures and Machines**, v. 15, n. 4, p. 481–506, 1988.

BAE, D.-S.; HAUG, E. E. A recursive formulation for constrained mechanical systems: Part i. parallel processor implementation. **Mechanical Structures and Machines**, v. 16, n. 2, p. 249–269, 1988.

BARUH, H. **Analytical Dynamics**. 1. ed. New York: McGraw-Hill, 1999.

BAUMGARTE, J. Stabilization of constraints and integrals of motion. **Methods in Applied Mechanics**, v. 1, n. 2, p. 239–258, 1972.

BEER, E. R. J. F. P. **Mecânica Vetorial para Engenheiros - Cinemática e Dinâmica**. 5. ed. São Paulo: Makron, 1994.

BEER, E. R. J. F. P. **Mecânica Vetorial para Engenheiros - Estática**. 5. ed. São Paulo: Makron, 1994.

BLAJER, W. An orthonormal tangent space method for constrained multibody systems. **Computer Methods In Applied Mechanics and Engineering**, v. 121, n. 2, p. 45–57, 1995.

BLAJER, W. A geometric unification of constrained system dynamics. **Multibody System Dynamics**, v. 1, n. 1, p. 3–21, 1997.

BOARD, O. A. R. **OpenGL Programming Guide: The Official Guide to Learning OpenGL**. 4. ed. New York: Addison-Wesley Professional, 2003.

BOOCH JAMES RUMBAUGH, I. J. G. **The Unified Modeling Language User Guide**. 1. ed. Cambridge: Addison Wesley, 1998.

COAD, P.; YOURDON, E. **Object-Oriented Analysis**. 2. ed. New Jersey: Prentice Hall, 1990.

COAD, P.; YOURDON, E. **Object-Oriented Design**. 1. ed. New Jersey: Prentice Hall, 1992.

COAD, P.; YOURDON, E. **Object-Oriented Programming**. 1. ed. New Jersey: Prentice Hall, 1993.

DUFF, I. S.; ERISMAN, A. M.; REID, J. K. **Direct Methods for Sparse Matrices**. 1. ed. New York: Oxford University Press, 1989.

ECKEL, B. **Thinking in C++ Volume II**. 1. ed. New Jersey: Prentice Hall, 1999.

ECKEL, B. **Thinking in C++ Volume I**. 2. ed. New Jersey: Prentice Hall, 2000.

FONSECA, A. **Curso de Mecânica - Volume I**. 1. ed. Rio de Janeiro: Ao Livro Técnico, 1958.

FONSECA, A. **Curso de Mecânica - Volume II**. 1. ed. Rio de Janeiro: Ao Livro Técnico, 1958.

GOLDSTEIN, H. **Classical Mechanics**. 3. ed. New York: Addison Wesley, 2000.

GREENWOOD, D. T. **Classical Dynamics**. 1. ed. Michigan: Dover, 1977.

GREENWOOD, D. T. **Advanced Dynamics**. 1. ed. Cambridge: Cambridge University Press, 2002.

HWANG, R. S.; HAUG, E. E. Topological analysis of multibody systems for recursive dynamics formulations. **Mechanical Structures and Machines**, v. 17, n. 2, p. 239–258, 1989.

JALóN, J. G. de; BAYO, E. **Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge**. 1. ed. New York: Springer, 1988.

KANE, T. R.; LEVINSON, D. A. **Dynamics: Theory and Applications**. 1. ed. New York: McGraw-Hill, 1985.

KUNZ, D. L. An object-oriented approach to multibody system analysis. **AIAA paper No. 97-1275, AIAA/ASME/ASCE,AHS/ASC 38 Structures, Structural Dynamics and Materials Conference, Kissimmee, Florida**, April, 1997.

LANCZOS, C. **The Variational Principles of Mechanics**. 4. ed. Toronto: Dover, 1986.

LAWRENCE, N. **Compaq Visual Fortran A Guide to Creating Windows Applications**. 1. ed. New York: Digital Press, 2002.

LIMA, E. L. **Álgebra Linear**. 7. ed. Rio de Janeiro: IMPA, 2004.

LURIE, A. I. **Analytical Dynamics**. 1. ed. New York: Springer, 2002.

MEIROVITCH, L. **Methods of Analytical Dynamics**. 2. ed. New York: Dover, 2004.

NEGRUT, D.; HARRIS, B. **ADAMS Theory in a Nutshell**. [S.l.], 2001. Disponível em:
<www.mscsoftware.com/support/university/guest_lectures/me543/adamsUofM.pdf>.

NIKRAVESH, P. E. **Computer-Aided Analysis of Mechanical Systems**. 1. ed. New Jersey: Prentice Hall PTR, 1988.

PETZOLD, C. **Programming Windows**. 5. ed. New York: Microsoft Press, 1998.

RAMPALLI, R. **ADAMS/Solver Theory Seminar**. [S.l.], 2000. Disponível em:
<www.mscsoftware.com/support/university/faculty_only/Training_Materials-/Theory_Seminar.pdf>.

SAAD, Y. **Iterative Methods for Sparse Linear Systems**. 3. ed. New York: Wiley, 2000.

SHABANA, A. A. **Computational Dynamics**. 2. ed. New York: Wiley-Interscience, 2001.

SHAMPINE, L. **Numerical Solution of Ordinary Differential Equations**. 1. ed. New York: Chapman & Hall, 1994.

SHILOV, G. E. **Linear Algebra**. 1. ed. Moscow: Dover, 1977.

SOLOW, D. **How to Read and Do Proofs**. 4. ed. New York: Wiley, 2005.

STRANG, G. **Linear Algebra and It's Applications**. 3. ed. New York: Brooks/Cole, 1988.

STRICHARTZ, R. S. **The Way of Analysis**. Revised. Toronto: Jones and Bartlett, 2000.

TASORA, A. An optimized lagrangian-multiplier approach for interactive multibody simulation in kinematic and dynamical digital prototyping. **Dipartimento di Ingegneria Strutturale (DIS) - Sezione Transporti e Movimentazione - Politecnico di Milano, Italy**, 2000.

TENENBAUM, R. A. **Dinâmica**. 1. ed. Rio de Janeiro: UFRJ, 1997.

VELLEMAN, D. J. **How to Prove It**. 1. ed. Cambridge: Cambridge, 1998.

WEHAGE, R. A. **Generalized Coordinate Partitioning in Dynamic Analysis of Mechanical Systems**. 1980. Tese (PhD. Thesis) — University of Iowa, Iowa City, 1980.

YU, Q.; CHEN, I. M. A direct violation correction method in numerical simulation of constrained multibody systems. **Computational Mechanics**, v. 26, n. 2, p. 52–57, 2000.

ZHENKUAN, P. An automatic correction method for position and velocity constraint equations violations in multibody systems. **Dynamics of Multibody Systems and Control**, v. 26, n. 2, p. 31–35, 1996.

ZORICH, V. A. **Mathematical Analysis I**. 1. ed. Moscow: Springer, 2000.

ZORICH, V. A. **Mathematical Analysis II**. 1. ed. Moscow: Springer, 2000.

# Apêndice A - Programa Maple

## A.1 SemEstabilização.mws

Programa SemEstabilização.mws

>

> restart;

> with(plots):

> with(plottools):

> eq1:=D(D(x))(t)*x(t)+D(D(y))(t)*y(t)+D(D(z))(t)*z(t)=

-2*D(x)(t)-2*D(y)(t)-2*D(z)(t);

> eq2:=D(D(x))(t)-x(t)/z(t)*D(D(z))(t)=x(t)/z(t)*g;

> eq3:=D(D(y))(t)-y(t)/z(t)*D(D(z))(t)=y(t)/z(t)*g;

> g:=9.81;

> r:=5;

> x_0:=r/sqrt(2);

> y_0:=0.0;

> z_0:=-r/sqrt(2);

> xp_0:=0.0;

> yp_0:=sqrt(-(x_0$\hat{2}$+y_0$\hat{2}$)/z_0*g);

> zp_0:=0.0;

> ci:=x(0)=x_0,D(x)(0)=xp_0,y(0)=y_0,D(y)(0)=yp_0,z(0)=z_0,D(z)(0)=zp_0;

> eq1;

> eq2;

> eq3;

> ci;

>Solution:=dsolve(eq1,eq2,eq3,ci,x(t),y(t),z(t),type=numeric,method=classical[rk4]

,corrections=4,maxfun=-1,output=listprocedure);

> odeplot(Solution,[t,z(t)],0..5,numpoints=1000);

> odeplot(Solution,[t,diff(z(t),t)],0..5,numpoints=1000);

> ax:=subs(Solution,x(t)):

> ay:=subs(Solution,y(t)):

> az:=subs(Solution,z(t)):

> plot([sqrt(ax(t)$\hat{}$2+ay(t)$\hat{}$2+az(t)$\hat{}$2),5],t=0..1,labels=[t,raio]);

## A.2 ComEstabilização.mws

Programa ComEstabilização.mws

>

> restart;

> with(plots):

> with(plottools):

>

eq1:=D(D(x))(t)*x(t)+D(D(y))(t)*y(t)+D(D(z))(t)*z(t)=

```
-2*D(x)(t)-2*D(y)(t)-2*D(z)(t)-2*alpha*(D(x)(t)*x(t)+
D(y)(t)*y(t)+D(z)(t)*z(t))-beta^2/2*(x(t)^2+y(t)^2+z(t)^2-r^2);
> eq2:=D(D(x))(t)-x(t)/z(t)*D(D(z))(t)=x(t)/z(t)*g;
> eq3:=D(D(y))(t)-y(t)/z(t)*D(D(z))(t)=y(t)/z(t)*g;
> g:=9.81;
> r:=5;
> alpha:=10.0;
> beta:=10.0;
> x_0:=r/sqrt(2);
> y_0:=0.0;
> z_0:=-r/sqrt(2);
> xp_0:=0.0;
> yp_0:=sqrt(-(x_0^2+y_0^2)/z_0*g);
> zp_0:=0.0;
> ci:=x(0)=x_0,D(x)(0)=xp_0,y(0)=y_0,D(y)(0)=yp_0,z(0)=z_0,D(z)(0)=zp_0;
> eq1;
> eq2;
> eq3;
> ci;
>Solution:=dsolve(eq1,eq2,eq3,ci,x(t),y(t),z(t),type=numeric,method=classical[rk4]
,corrections=4,maxfun=-1,output=listprocedure);
> odeplot(Solution,[t,z(t)],0..30,numpoints=1000);
> odeplot(Solution,[t,diff(z(t),t)],0..30,numpoints=1000);
> ax:=subs(Solution,x(t)):
```

> ay:=subs(Solution,y(t)):

> az:=subs(Solution,z(t)):

> plot([sqrt(ax(t)$\hat{2}$+ay(t)$\hat{2}$+az(t)$\hat{2}$),5],t=0..30,labels=[t,raio]);

## A.3 ComEstabilização02.mws

Programa ComEstabilização02.mws

>

> restart;

> with(plots):

> with(plottools):

> with(LinearAlgebra):

> r:=5;

> g:=9.81;

>F:=(t,x,y,z,u,v,w)->evalf(LinearSolve(Matrix([[1,0,0,0,0,0],[0,1,0,0,0,0],[0,0,1,0,0,0
],[0,0,0,x,y,z],[0,0,0,1,0,-x/z],[0,0,0,0,1,-y/z]]),Vector([u,v,w,-u$\hat{2}$-v$\hat{2}$-w$\hat{2}$-2*alpha*
(x*u+y*v+z*w)-beta$\hat{2}$/2*(x$\hat{2}$+y$\hat{2}$+z$\hat{2}$-r$\hat{2}$),x/z*g,y/z*g])));

> t0:=0.0;

> x_0:=r/sqrt(2);

> y_0:=0.0;

> z_0:=-r/sqrt(2);

> u_0:=0.0;

> v_0:=sqrt(-(x_0$\hat{2}$+y_0$\hat{2}$)/z_0*g);

> w_0:=0.0;

```
> TheEnd:=false;

> LastStep:=false;

> t:=t0;

> x_1:=x_0;

> y_1:=y_0;

> z_1:=z_0;

> u_1:=u_0;

> v_1:=v_0;

> w_1:=w_0;

> tsim:=30;

> tol:=evalf(10^(-3));

> betasim:=0.9;

> cont:=0;

> tempo[cont]:=t0;

> x[cont]:=x_1;

> y[cont]:=y_1;

> z[cont]:=z_1;

> u[cont]:=u_1;

> v[cont]:=v_1;

> w[cont]:=w_1;

> cont:=cont+1;

> h:=0.01;

> alpha:=1/h;

> beta:=sqrt(2)/h;
```

```
> for i while TheEnd=false do
> f0:=F(t,x_1,y_1,z_1,u_1,v_1,w_1);
f1:=F(t+h/3.0,x_1+h/3.0*f0[1],y_1+h/3.0*f0[2],z_1+h/3.0*f0[3],u_1+h/3.0*f0[4],
v_1+h/3.0*f0[5],w_1+h/3.0*f0[6]);
f2:=F(t+h/3.0,x_1+h/6.0*(f0[1]+f1[1]),y_1+h/6.0*(f0[2]+f1[2]),z_1+h/6.0*(f0[3]+
f1[3]),u_1+h/6.0*(f0[4]+f1[4]),v_1+h/6.0*(f0[5]+f1[5]),w_1+h/6.0*(f0[6]+f1[6]));
f3:=F(t+h/2.0,x_1+h/8.0*(f1[1]+3.0*f2[1]),y_1+h/8.0*(f1[2]+3.0*f2[2]),z_1+
h/8.0*(f1[3]+3.0*f2[3]),u_1+h/8.0*(f1[4]+3.0*f2[4]),v_1+h/8.0*(f1[5]+3.0*f2[5])
,w_1+h/8.0*(f1[6]+3.0*f2[6]));
f4:=F(t+h,x_1+h/2.0*(f0[1]-3.0*f2[1]+4.0*f3[1]),y_1+h/2.0*(f0[2]-3.0*f2[2]+
4.0*f3[2]),z_1+h/2.0*(f0[3]-3.0*f2[3]+4.0*f3[3]),u_1+h/2.0*(f0[4]-3.0*f2[4]+
4.0*f3[4]),v_1+h/2.0*(f0[5]-3.0*f2[5]+4.0*f3[5]),w_1+h/2.0*(f0[6]-3.0*f2[6]+
4.0*f3[6]));
y_4:=Vector([x_1,y_1,z_1,u_1,v_1,w_1])+h/2.0*(f0-3.0*f2+4.0*f3);
> y_:=Vector([x_1,y_1,z_1,u_1,v_1,w_1])+h/6.0*(f0+4.0*f3+f4);
est:=evalf(Norm(y_-y_4,Euclidean)/5.0);
> if (est¡evalf(tol)) then
> tempo[cont]:=t+h;
> x[cont]:=y_[1]:
> y[cont]:=y_[2]:
> z[cont]:=y_[3]:
> u[cont]:=y_[4]:
> v[cont]:=y_[5]:
> w[cont]:=y_[6]:
```

```
> passo[cont]:=h:

> cont:=cont+1:

> t:=t+h:

> x_1:=y_[1]:

> y_1:=y_[2]:

> z_1:=y_[3]:

> u_1:=y_[4]:

> v_1:=y_[5]:

> w_1:=y_[6]:

> end if:

> if (abs(est)>1e-12) then

> h_:=h*evalf(0.9*tol/est)^(1/4);

> else

> h_:=1.8*h;

> end if:

> if (h_>2.0*h) then

> h_:=2.0*h;

> end if:

> if (h_¡h/4.0) then

> h_:=h/4.0;

> end if:

> h:=h_;

> alpha:=1/h:

> beta:=sqrt(2)/h:
```

```
> if (LastStep=true) then
> TheEnd:=true;
> next:
> end if;
> if (t>tsim) then
> h:=h-(t-tsim);
> LastStep:=true;
> end if;
> end do:
> cont;
> Solraio:=[t0,sqrt(x_0^2+y_0^2+z_0^2)]:
> Solz:=[t0,z_0]:
> Solw:=[t0,w_0]:
> StepSize:=[t0,passo[1]]:
> for i from 1 to cont-1 do
> Solraio:=Solraio,[tempo[i],sqrt(x[i]^2+y[i]^2+z[i]^2)]:
> Solz:=Solz,[tempo[i],z[i]]:
> Solw:=Solw,[tempo[i],w[i]]:
> StepSize:=StepSize,[tempo[i],passo[i]]:
> end do:
> listplot([Solraio],labels=[tempo,raio],view=[0..30, 5.000006..5.000009]);
> listplot([Solz],labels=[tempo,z]);
> listplot([Solw],labels=[tempo,zp]);
> listplot([StepSize],labels=[tempo,passo],view=[0..30, 0.00544..0.00548]);
```

## A.4 ComProjecao.mws

Programa ComProjeção.mws

>

> restart;

> with(plots):

> with(plottools):

> with(LinearAlgebra):

> m:=10.0;

> g:=9.81;

> r:=5.0;

Inserindo a função F tal que y'=F(y)

>

> F:=proc(y)

>Sol:=evalf(LinearSolve(Matrix([[m,0,0,2*y[1]],[0,m,0,2*y[2]],[0,0,m,2*y[3]],

>[2*y[1],2*y[2],2*y[3],0]]),Vector([0,0,-m*g,-2*y[4]^2-2*y[5]^2-2*y[6]^2]))):

> evalf(Vector([y[4],y[5],y[6],Sol[1..3]]))

> end proc;

> F_DeltaY:=proc(y)

>SolDy:=evalf(LinearSolve(Matrix([[1,0,0,2*y[1]],[0,1,0,2*y[2]],[0,0,1,2*y[3]],

>[2*y[1],2*y[2],2*y[3],0]]),Vector([0,0,0,-y[1]^2-y[2]^2-y[3]^2+r^2]))):

> evalf(Vector([SolDy[1..3]]))

> end proc;

> F_DeltaYp:=proc(y)

>SolDyp:=evalf(LinearSolve(Matrix([[1,0,0,2*y[1]],[0,1,0,2*y[2]],[0,0,1,2*y[3]],

>[2*y[1],2*y[2],2*y[3],0]]),Vector([0,0,0,-2*y[1]*y[4]-2*y[2]*y[5]-2*y[3]*y[6]])):

> evalf(Vector([SolDyp[1..3]]))

> end proc;

Inserindo as condições iniciais

>

> x0:=evalf(r/sqrt(2));

> y0:=0.0;

> z0:=-evalf(r/sqrt(2));

> y:=Vector([x0,y0,z0,0.0,sqrt(-(x0$\hat{2}$+y0$\hat{2}$)/z0*g),0.0]);

> TheEnd:=false;

> LastStep:=false;

> h:=0.5;

> tsim:=30;

> cont:=1;

> tol:=1e-3;

> t:=0.0;

> Projecao:=true;

> for i while TheEnd=false do

> f0:=F(y);

> f1:=F(y+h/3.0*f0);

> f2:=F(y+h/6.0*(f0+f1));

> f3:=F(y+h/8.0*(f1+3.0*f2));

> y_4:=evalf(y+h/2.0*(f0-3.0*f2+4.0*f3));

> f4:=F(y_4);

```
> y_:=evalf(y+h/6.0*(f0+4.0*f3+f4));

> est:=evalf(Norm(y_-y_4,Euclidean)/5.0);

> if (est¡evalf(tol)) then

> if (Projecao=true) then

> Fim_Proj:=false:

> while Fim_Proj¡>true do

> y_[1..3]:=y_[1..3]+F_DeltaY(y_):

> if (abs(y_[1]^2+y_[2]^2+y_[3]^2-r^2) ¡ evalf(tol)) then

> Fim_Proj:=true:

> end if:

> end do:

> y_[4..6]:=y_[4..6]+F_DeltaYp(y_):

> end if:

> tempo[cont]:=t+h;

> sol[cont]:=evalf(y_):

> passo[cont]:=h:

> cont:=cont+1:

> t:=t+h:

> y:=y_

> end if:

> if (abs(est)>1e-12) then

> h_:=h*evalf(0.9*tol/est)^(1/4);

> else

> h_:=1.8*h;
```

```
> end if:
> if (h_>2.0*h) then
> h_:=2.0*h;
> end if:
> if (h_¡h/4.0) then
> h_:=h/4.0;
> end if:
> h:=h_;
> if (LastStep=true) then
> TheEnd:=true;
> next:
> end if;
> if (t>tsim) then
> h:=h-(t-tsim);
> LastStep:=true;
> end if;
> end do:
> cont;
> Solraio:=[0.0,sqrt(sol[1][1]^2+sol[1][2]^2+sol[1][3]^2)]:
> Solz:=[0.0,sol[1][3]]:
> Solv:=[0.0,sol[1][5]]:
> Solw:=[0.0,sol[1][6]]:
> StepSize:=[0.0,passo[1]]:
> for i from 1 to cont-1 do
```

> Solraio:=Solraio,[tempo[i],sqrt(sol[i][1]$\hat{2}$+sol[i][2]$\hat{2}$+sol[i][3]$\hat{2}$)]:

> Solz:=Solz,[tempo[i],sol[i][3]]:

> Solv:=Solv,[tempo[i],sol[i][5]]:

> Solw:=Solw,[tempo[i],sol[i][6]]:

> StepSize:=StepSize,[tempo[i],passo[i]]:

> end do:

> listplot([Solraio],labels=[tempo,raio],view=[0..30,5-1e-6..5+1e-6]);

> listplot([Solz],labels=[tempo,z],view=[0..30,evalf(-5*sqrt(2)/2-2e-2)..

>evalf(-5*sqrt(2)/2+1e-2)]);

> listplot([Solw],labels=[tempo,zp]);

> listplot([StepSize],labels=[tempo,passo],view=[0..30, 0.0..0.25]);

## A.5 Formulação Simbólica

> restart;

> with(LinearAlgebra):

> with(plots):

href é a referência para a energia potencial

>

> href:=0:

g é a gravidade

>

> g:=g:

Abaixo, devemos entrar com uma matriz de dimensão nx13, onde n é o número de corpos. A estrutura de uma dada linha i dessa matriz, referente ao corpo i, é da forma [x , y , z , wx , wy , wz , m , Ixx , Ixy , Ixz , Iyy , Iyz , Izz], onde x = x(var1(t),var2(t),... , y = y(var1(t),var2(t),... , z = z(var1(t),var2(t),... , w[] = w[](var1(t),var2(t),... , etc. As coordenadas x, y, z são do centro de massa e os componenetes do vetor velocidade angular devem estar representados numa base fixa ao corpo igual aquela utilizada na representação matricial do tensor de inércia. Como as variáveis do centro de massa bem como os componenetes do vetor velocidade angular são escritos em função das funções (graus de liberdade) var1(t),..., devemos inserir todas essas variáveis no vetor CoordVec.

>

>

> NCorpos:=3:

```
> NCol:=13:
> Data:=Matrix(NCorpos,NCol):
> x:=0:
> y:=l[2]/2:
> z:=0:
> wx:=-diff(phi(t),t):
> wy:=0:
> wz:=0:
> Data[1,1..NCol]:=¡x,y,z,wx,wy,wz,m,Ixx,0,0,Iyy,0,Izz>:
> x:=l[3]/2*cos(theta[1](t))*cos(phi(t)):
> y:=l[2]+l[3]/2*sin(theta[1](t)):
> z:=l[3]/2*cos(theta[1](t))*sin(phi(t)):
> wx:=-diff(phi(t),t)*sin(theta[1](t)):
> wy:=-diff(phi(t),t)*cos(theta[1](t)):
> wz:=diff(theta[1](t),t):
> Data[2,1..NCol]:=¡x,y,z,wx,wy,wz,m,Ixx,0,0,Iyy,0,Izz>:
> x:=l[3]*cos(theta[1](t))*cos(phi(t))+l[4]/2*cos(theta[1](t)+theta[2](t))*cos(phi(t)):
> y:=l[2]+l[3]*sin(theta[1](t))+l[4]/2*(sin(theta[1](t)+theta[2](t))):
> z:=l[3]*cos(theta[1](t))*sin(phi(t))+l[4]/2*(cos(theta[1](t)+theta[2](t)))*sin(phi(t)):
> wx:=-diff(phi(t),t)*(sin(theta[1](t)+theta[2](t))):
> wy:=-diff(phi(t),t)*(cos(theta[1](t)+theta[2](t))):
> wz:=diff(theta[1](t),t)+diff(theta[2](t),t):
> Data[3,1..NCol]:=¡x,y,z,wx,wy,wz,m,Ixx,0,0,Iyy,0,Izz>:
Entrando com o vetor das coordenadas
>
> CoordVec:=Matrix([[phi(t),theta[1](t),theta[2](t)]]):
Obtendo as equações de movimento
> printf("Obtendo o Lagrangeano do sistema...");
> L:=0:
> for i from 1 to RowDimension(Data) do
> ECtrans:=1/2*Data[i,7]*(diff(Data[i,1],t)^2+diff(Data[i,2],t)^2+diff(Data[i,3],t)^2):
In:=Matrix([[Data[i,8],Data[i,9],Data[i,10]],[Data[i,9],Data[i,11],Data[i,12]],[Data[i,10],Data[i,12],Data[i,13]]]):
> Omega:=Matrix([[Data[i,4],Data[i,5],Data[i,6]]]):
> ECrot:=1/2*MatrixMatrixMultiply(Omega,MatrixMatrixMultiply(In,Transpose(Omega))):
> EP:=Data[i,7]*g*(Data[i,2]+href):
> L:=L+ECtrans+ECrot[1,1]-EP:
> end do:
> printf("Transformando as variáveis temporais em algébricas...");
> L1:=L:
> TransMatx:=Matrix(3,ColumnDimension(CoordVec)):
> for i from 1 to ColumnDimension(CoordVec) do
> TransMatx[1,i]:=CoordVec[1,i]:
> TransMatx[2,i]:=var[i]:
> TransMatx[3,i]:=dvar[i]:
> L1:=subs(TransMatx[1,i]=TransMatx[2,i],diff(TransMatx[1,i],t)=TransMatx[3,i],L1):
> end do:
> L1:=expand(L1):
> printf("Derivando o Lagrangeano");
> for i from 1 to ColumnDimension(TransMatx) do
> EqMovData[i,1]:=diff(L1,TransMatx[3,i]):
```

```
> EqMovData[i,2]:=diff(L1,TransMatx[2,i]):
> end do:
> printf("Transformando as variáveis algébricas em temporais...");
> for i from 1 to ColumnDimension(TransMatx) do
> for j from 1 to ColumnDimension(TransMatx) do
EqMovData[i,1]:=subs(TransMatx[2,j]=TransMatx[1,j],TransMatx[3,j]=diff(TransMatx[1,j],t),EqMovData[i,1]):
> EqMovData[i,2]:=subs(TransMatx[2,j]=TransMatx[1,j],TransMatx[3,j]=diff(TransMatx[1,j],t),EqMovData[i,2]):
> end do:
> end do:
> printf("Obtendo as equações de movimento");
> for i from 1 to ColumnDimension(TransMatx) do
> EqMov[i]:=simplify(simplify(diff(EqMovData[i,1],t)-EqMovData[i,2]=0,'symbolic'),'size'):
> end do:
> printf("Escrevendo equações de movimento");
> for i from 1 to ColumnDimension(TransMatx) do
> printf("Equação de movimento para %a",TransMatx[1,i]);
> EqMov[i];
> end do;
```

# A.6 Funções Matlab

## A.6.1 Torque_Robo_Phi.m

```
function T=Torque_Robo_Phi ( phi , der_phi , theta1 , der_theta1 , theta2 , der_theta2 , tempo)
%
m = 8.8227231446355;
Ixx = .3970225415e-2;
Iyy = .11962142129798;
Izz = .11962142129798;
l = 0.4;
g = 9.80665;
%
K_phi = 10.0;
K_theta1 = 10.0;
K_theta2 = 10.0;
%
phi = phi*pi/180;
theta1 = theta1*pi/180;
theta2 = theta2*pi/180;
%
phi_d = 30.0 * ( 1.0 - exp(-1.3158*tempo) )*pi/180.0;
der_phi_d = 30.0 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
der2_phi_d = - 30.0 * 1.3158 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
%
```

```
theta1_d = 35.0 * ( 1.0 - exp(-1.3158*tempo) )*pi/180.0;
der_theta1_d = 35.0 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
der2_theta1_d = - 35.0 * 1.3158 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
%
theta2_d = 10.0 * ( 1.0 - exp(-1.3158*tempo) )*pi/180.0;
der_theta2_d = 10.0 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
der2_theta2_d = - 10.0 * 1.3158 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
%
erro_phi = (phi_d-phi);
erro_der_phi = (der_phi_d - der_phi);
%
erro_theta1 = (theta1_d-theta1);
erro_der_theta1 = (der_theta1_d - der_theta1);
%
erro_theta2 = (theta2_d-theta2);
erro_der_theta2 = (der_theta2_d - der_theta2);
%
T =
.2500000000*(((8.*Iyy-8.*Ixx+2.*m*l^2)*cos(theta2)^2+4.*m*l^2*cos(theta2)+4.*m*l^2)*cos(theta1)^2+8.*sin(theta1)*
((-1.*Iyy+Ixx-.2500000000*m*l^2)*cos(theta2)-.5000000000*m*l^2)*sin(theta2)*cos(theta1)+
(-4.*Iyy+4.*Ixx-1.*m*l^2)*cos(theta2)^2+8.*Ixx+m*l^2+4.*Iyy)*(der2_phi_d+K_phi*(erro_phi+erro_der_phi
))+4.*der_phi*((((-1.*Iyy+Ixx-.2500000000*m*l^2)*cos(theta2)-.5000000000*m*l^2)*sin(theta2)*cos(theta1)^2
+sin(theta1)*((-1.*Iyy+Ixx-.2500000000*m*l^2)*cos(theta2)^2-.5000000000*m*l^2*cos(theta2)-.5000000000*m*l^2
)*cos(theta1)-.5000000000*((-1.*Iyy+Ixx-.2500000000*m*l^2)*cos(theta2)-.5000000000*m*l^2)*
sin(theta2))*der_theta1+(((-1.*Iyy+Ixx-.2500000000*m*l^2)*cos(theta2)-.2500000000*m*l^2)*sin(theta2)
*cos(theta1)^2+sin(theta1)*((-1.*Iyy+Ixx-.2500000000*m*l^2)*cos(theta2)^2-.2500000000*m*l^2*
cos(theta2)-.5000000000*Ixx+.5000000000*Iyy+.1250000000*m*l^2)*cos(theta1)-.5000000000*cos(theta2)*
sin(theta2)*(-1.*Iyy+Ixx-.2500000000*m*l^2))*der_theta2);
```

## A.6.2 Torque_Robo_theta1.m

```
function T=Torque_Robo_theta1 ( phi , der_phi , theta1 , der_theta1 , theta2 , der_theta2 , tempo)
%
m = 8.8227231446355;
Ixx = .3970225415e-2;
Iyy = .11962142129798;
Izz = .11962142129798;
l = 0.4;
g = 9.80665;
%
K_phi = 10.0;
K_theta1 = 10.0;
K_theta2 = 10.0;
%
phi = phi*pi/180;
theta1 = theta1*pi/180;
theta2 = theta2*pi/180;
```

```
%
phi_d = 30.0 * ( 1.0 - exp(-1.3158*tempo) )*pi/180.0;
der_phi_d = 30.0 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
der2_phi_d = - 30.0 * 1.3158 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
%
theta1_d = 35.0 * ( 1.0 - exp(-1.3158*tempo) )*pi/180.0;
der_theta1_d = 35.0 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
der2_theta1_d = - 35.0 * 1.3158 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
%
theta2_d = 10.0 * ( 1.0 - exp(-1.3158*tempo) )*pi/180.0;
der_theta2_d = 10.0 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
der2_theta2_d = - 10.0 * 1.3158 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
%
erro_phi = (phi_d-phi);
erro_der_phi = (der_phi_d - der_phi);
%
erro_theta1 = (theta1_d-theta1);
erro_der_theta1 = (der_theta1_d - der_theta1);
%
erro_theta2 = (theta2_d-theta2);
erro_der_theta2 = (der_theta2_d - der_theta2);
%
T =
.2500000000*(6.*m*l^2+4.*m*l^2*cos(theta2)+8.*Izz)*(der2_theta1_d+K_theta1*
(erro_theta1+erro_der_theta1))+.2500000000*(4.*Izz+m*l^2+2.*m*l^2*
cos(theta2))*(der2_theta2_d+K_theta2*(erro_theta2+erro_der_theta2))+.2500000000*
(-8.*cos(theta1)*sin(theta1)*(-.2500000000*m*l^2+Ixx-1.*Iyy)*cos(theta2)^2+(-8.*
sin(theta2)*(-.2500000000*m*l^2+Ixx-1.*Iyy)*cos(theta1)^2+4.*m*l^2*cos(theta1)*sin(theta1)+
4.*sin(theta2)*(-.2500000000*m*l^2+Ixx-1.*Iyy))*cos(theta2)+4.*m*(l^2*cos(theta1)^2*sin(theta2)+
sin(theta1)*l^2*cos(theta1)-.5000000000*l^2*sin(theta2)))*der_phi^2+.5000000000*(-1.*l^2*
sin(theta2)*der_theta2^2-2.*l^2*der_theta1*sin(theta2)*der_theta2+g*(3.*l*cos(theta1)-1.*l*
sin(theta1)*sin(theta2)+l*cos(theta1)*cos(theta2)))*m;
```

### A.6.3 Torque_Robo_theta2.m

```
function T=Torque_Robo_theta2 ( phi , der_phi , theta1 , der_theta1 , theta2 , der_theta2 , tempo)
%
m = 8.8227231446355;
Ixx = .3970225415e-2;
Iyy = .11962142129798;
Izz = .11962142129798;
l = 0.4;
g = 9.80665;
%
K_phi = 10.0;
K_theta1 = 10.0;
K_theta2 = 10.0;
```

```
%
phi = phi*pi/180;
theta1 = theta1*pi/180;
theta2 = theta2*pi/180;
%
phi_d = 30.0 * ( 1.0 - exp(-1.3158*tempo) )*pi/180.0;
der_phi_d = 30.0 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
der2_phi_d = - 30.0 * 1.3158 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
%
theta1_d = 35.0 * ( 1.0 - exp(-1.3158*tempo) )*pi/180.0;
der_theta1_d = 35.0 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
der2_theta1_d = - 35.0 * 1.3158 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
%
theta2_d = 10.0 * ( 1.0 - exp(-1.3158*tempo) )*pi/180.0;
der_theta2_d = 10.0 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
der2_theta2_d = - 10.0 * 1.3158 * 1.3158 * exp(-1.3158*tempo)*pi/180.0;
%
erro_phi = (phi_d-phi);
erro_der_phi = (der_phi_d - der_phi);
%
erro_theta1 = (theta1_d-theta1);
erro_der_theta1 = (der_theta1_d - der_theta1);
%
erro_theta2 = (theta2_d-theta2);
erro_der_theta2 = (der_theta2_d - der_theta2);
%
T =
.2500000000*(4.*Izz+2.*m*l^2*cos(theta2)+m*l^2)*(der2_theta1_d+K_theta1*
(erro_theta1+erro_der_theta1))+.2500000000*(m*l^2+4.*Izz)*(der2_theta2_d+K_theta2*
(erro_theta2+erro_der_theta2))+.2500000000*(-8.*sin(theta1)*cos(theta1)*(-1.*Iyy+Ixx-
.2500000000*m*l^2)*cos(theta2)^2+(-8.*sin(theta2)*(-1.*Iyy+Ixx-.2500000000*m*l^2)*cos(theta1)^2+
2.*m*l^2*cos(theta1)*sin(theta1)+4.*sin(theta2)*(-1.*Iyy+Ixx-.2500000000*m*l^2))*cos(theta2)+
4.*cos(theta1)*(.5000000000*m*l^2*sin(theta2)*cos(theta1)+sin(theta1)*(-1.*Iyy+Ixx-.2500000000*
m*l^2)))*der_phi^2+.5000000000*(l*der_theta1^2*sin(theta2)+g*(-1.*sin(theta1)*sin(theta2)+cos(theta1)*
cos(theta2)))*m*l;
```

# Anexo A - Solução Numérica

Neste anexo será apresentada a solução numérica de um ponto de vista mais aplicado, onde serão feitas referências esfecíficas à implementação computacional. A solução aqui mostrada refere-se ao método que utiliza a decomposição $\mathbf{LDL}^T$ na matriz dos coeficientes do sistema linear fornecendo as acelerações e multiplicadores de Lagrange, Eq. (4.20), mostrada abaixo

$$\begin{bmatrix} \overline{\mathbf{M}} & (\boldsymbol{\varphi}_{\mathrm{q}})^T \\ \boldsymbol{\varphi}_{\mathrm{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_{\alpha} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{Q}} \\ \overline{\mathbf{Q}}_{\mathrm{d}} \end{bmatrix}. \tag{A.1}$$

Como a matriz dos coeficientes da Eq. (A.1) é simétrica e a matriz $\overline{\mathbf{M}}$ é constante e diagonal, esta matriz pode ser representada da forma mostrada na Fig. (A.1), onde a matriz é dividida em partes. A parte indicada pelo número um é constituída pela matriz diagonal $\overline{\mathbf{M}}$, o número dois indica o jacobiano $\boldsymbol{\varphi}_{\mathrm{q}}$ e o número três indica a matriz composta de zeros. Como a matriz $\overline{\mathbf{M}}$ é diagonal, somente as matrizes indicadas pelos números dois e três terão seus valores alterados durante o processo de fatoração. A fatoração $\mathbf{LDL}^T$ implementada é realizada utilizando-se a fórmula de Doolittle.

Agora, vamos obter as fórmulas de Doolittle. Supondo $\mathbf{A}$ uma matriz quadrada genérica inversiva temos que, ao fatorarmos a matriz $\mathbf{A}$ na forma $\mathbf{A} = \mathbf{LU}$, com $\mathbf{L}$

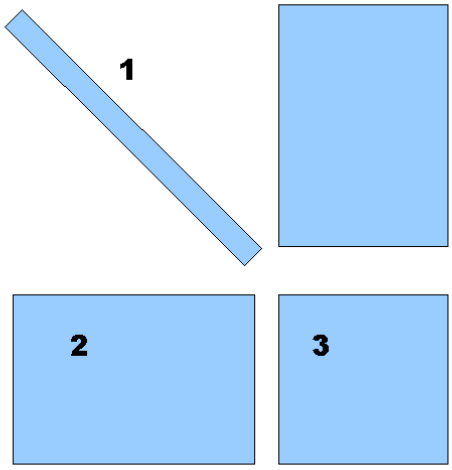



FIGURA A.1 – Gráfico ilustrando a matriz dos coeficientes da Eq. (A.1)

triangular inferior com diagonal unitária e **U** triangular superior, um dado termo $a_{ij}$ da matriz **A** será escrito em função dos termos das matrizes **L** e **U** da seguinte forma (DUFF; ERISMAN; REID, 1989)

$$a_{ij} = \sum_{p=1}^{min(i,j)} l_{ip}u_{pj}. \tag{A.2}$$

Rearranjando a Eq. (A.2) podemos escrever

$$l_{ij} = \frac{1}{u_{jj}}(a_{ij} - \sum_{p=1}^{j-1} l_{ip}u_{pj}), \quad i > j \tag{A.3}$$

$$u_{ij} = (a_{ij} - \sum_{p=1}^{i-1} l_{ip}u_{pj}), \quad i \leq j. \tag{A.4}$$

As Eqs. (A.3, A.4) permitem a obtenção das matrizes **L** e **U** a partir da matriz **A**. Para tal, esta obtenção deve ser realizada linha a linha, onde determina-se a primeira linha das matrizes **L** e **U** e assim sucessivamente. Uma variação da decomposição **A** = **LU** é

a decomposição $\mathbf{A} = \mathbf{LDQ}$, onde $\mathbf{D}$ é uma matriz diagonal e $\mathbf{Q}$ é uma matriz triangular superior com os elementos da diagonal unitários. Obviamente, temos que $\mathbf{DQ} = \mathbf{U}$. Caso a matriz $\mathbf{A}$ seja simétrica, podemos escrever $\mathbf{A} = \mathbf{LDQ} = \mathbf{A}^T = (\mathbf{LDQ})^T = \mathbf{Q}^T\mathbf{DL}^T \Rightarrow \mathbf{Q} = \mathbf{L}^T$. Logo, para uma matriz simétrica, podemos utilizar as fórmulas de Doolittle para obter as matrizes $\mathbf{L}$ e $\mathbf{D}$ tais que $\mathbf{A} = \mathbf{LDL}^T$. Neste caso os termos $d_{ii}$ da matriz diagonal $\mathbf{D}$ são obtidos a partir da fórmula dada pela Eq. (A.4), já que $d_{ii} = u_{ii}$. Contudo, deve-se observar que os termos $u_{pj}, p < j$ *não* são gerados, visto que somente os termos da diagonal devem ser obtidos. Como esses termos aparecem nas Eqs. (A.3, A.4), observando-se que, para $p < j$, $u_{pj} = l_{jp}d_{pp}$, podemos reescrever as Eqs. (A.3, A.4) da seguinte maneira

$$l_{ij} = \frac{1}{d_{jj}}(a_{ij} - \sum_{p=1}^{j-1} l_{ip}l_{jp}d_{pp}), \quad i > j \tag{A.5}$$

$$d_{ii} = (a_{ii} - \sum_{p=1}^{i-1} l_{ip}l_{jp}d_{pp}). \tag{A.6}$$

Observando-se a Fig. (A.2), vemos que, ao aplicarmos a decomposição $\mathbf{LDL}^T$ à matriz dos coeficientes, fazendo uso das fórmulas de Doolittle, a parte indicada pelo número um fica inalterada. De fato, a decomposição começa a partir da primeira linha após a matriz diagonal de massa $\overline{\mathbf{M}}$. Nesta parte, os elementos referentes ao jacobiano, indicados na Fig. (A.2) com o número dois, são obtidos com o uso da Eq. (A.5), cujo resultado é

$$l_{ij} = \frac{a_{ij}}{d_{jj}}, \tag{A.7}$$

indicando que os valores $l_{ij}$ referentes à parte de número dois na Fig. (A.2), oriundos da decomposição $\mathbf{LDL}^T$, são obtidos simplesmente pela divisão dos elementos do jacobiano

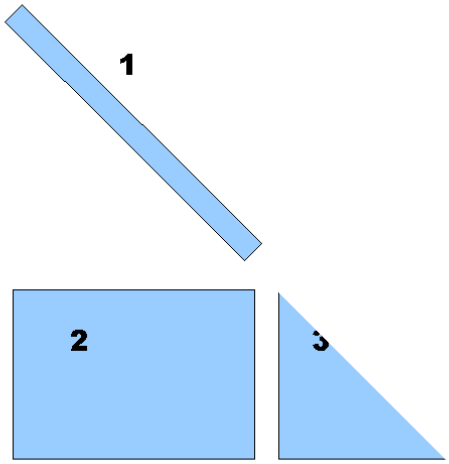



FIGURA A.2 – Gráfico ilustrando a matriz dos coeficientes da Eq. (A.1)

pelos elementos da matriz de massa na mesma coluna. A Fig. (A.3) ilustra o processo, onde uma coluna da matriz dos coeficientes é destacada. Note que é mais eficiente armazenar diretamente os inversos dos elementos da matriz de massa diagonal $\overline{\mathbf{M}}$, visto que a operação de divisão requer um número muito maior de ciclos do processador para ser efetuada do que a operação de multiplicação.

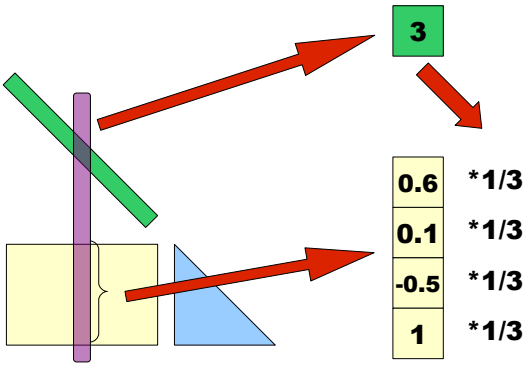



FIGURA A.3 – Gráfico ilustrando o processo de fatoração

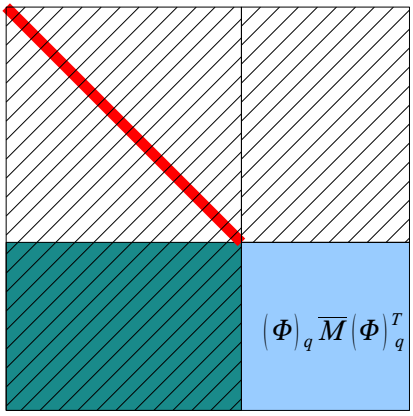



FIGURA A.4 – Gráfico ilustrando o processo de fatoração.

A obtenção dos termos referentes à parte indicada com o número três na Fig. (A.2) requer um pré-processamento, visto que é necessário conhecer antecipadamente o padrão de preennchimento desta matriz durante o processo de fatoração, já que esta matriz é, após o processo de fatoração, esparsa. Note que este pré-processamento é necessário por questões de eficiência, pois, além desta matriz ser, ao final do processo de fatoração, esparsa, este padrão de preenchimento é também invariante durante a simulação. Assim, é melhor adotar uma estrutura de dados que seja gerada inicialmente do que utilizar uma estrutura dinâmica de dados[1]. O padrão inicial de preenchimento é obtido através do emprego de matrizes booleanas. Observando-se que, caso fizéssemos uso da fatoração tradicional de Gauss, quando a sub-matriz a ser fatorada fosse a matriz indicada pelo número três na Fig. (A.2), esta matriz seria igual a $\boldsymbol{\varphi}_{\mathrm{q}}\overline{\mathbf{M}}(\boldsymbol{\varphi}_{\mathrm{q}})^T$, uma matriz de fato simétrica, positiva definida. A Fig. (A.4) ilustra quando a sub-matriz do processo de eliminação de Gauss coincide com a matriz indicada pelo número três na Fig. (A.2). Neste

[1]Note-se que a estrutura de dados utilizada é alocada inicialmente, fazendo uso de arrays. No caso de utilizarmos uma estrutura dinâmica (listas encadeadas) para acomodar os preenchimentos, teríamos um decréscimo severo de eficiência, visto que arrays são alocados sequencialmente na memória do computador, viabilizando o acesso mais rápido aos elementos da matriz, representados nesses arrays.

ponto, os preenchimentos originaram na matriz originalmente nula a matriz $\boldsymbol{\varphi}_{\mathrm{q}}\overline{\mathbf{M}}(\boldsymbol{\varphi}_{\mathrm{q}})^T$. Ao aplicarmos a eliminação de Gauss nesta matriz teríamos ainda outros preenchimentos.

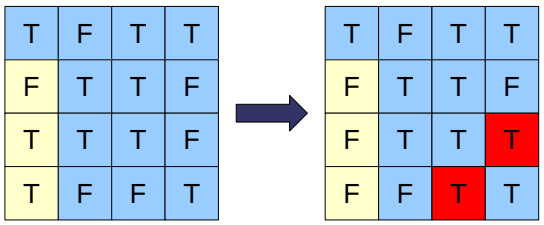



FIGURA A.5 – Gráfico ilustrando o processo de fatoração

O programa implementado ao realizar o pré-processamento para obtenção desses preenchimentos primeiro forma uma matriz booleana para o jacobiano, onde esta matriz terá um elemento verdadeiro quando no jacobiano este for diferente de zero. Após a obtenção desta matriz ocorre a multiplicação booleana desta pela sua transposta[2]. Obtido o resultado desta multiplicação matricial, o programa aplica uma rotina de eliminação Gaussiana à matriz booleana resultante da multiplicação matricial conduzida anteriormente. A Fig. (A.5) ilustra a eliminação de Gauss aplicada a uma matriz booleana. Os números nos campos em vermelho constituem os preenchimentos. Esta eliminação é realizada escolhendo-se sempre os pivos da diagonal, já que estes produzem uma fatoração estável, devido ao fato da matriz ser positiva definida (DUFF; ERISMAN; REID, 1989). Escolhendo-se os pivos na diagonal preserva-se a simetria durante o processo de fatoração. Na busca do melhor pivo na diagonal a cada passo, visando à minimização dos preenchimentos, utiliza-se o método de grau mínimo, onde escolhe-se o pivo pertencente à linha com o menor número de entradas. Este critério de escolha é uma modificação do critério de Markowitz, adaptado à matrizes simétricas, positivas definidas. A propriedade

[2]Note que a matriz $\overline{\mathbf{M}}$ é irrelevante dada à natureza das operações (booleanas) envolvidas.

da simetria da matriz $\boldsymbol{\varphi}_{\mathsf{q}}\overline{\mathbf{M}}(\boldsymbol{\varphi}_{\mathsf{q}})^T$ é trivial de ser observada. Contudo, o fato desta matriz ser positiva definida é mais sútil à primeira vista. Assim, antes de darmos continuidade à questão da fatoração da matriz dos coeficientes, vamos provar que a matriz $\boldsymbol{\varphi}_{\mathsf{q}}\overline{\mathbf{M}}(\boldsymbol{\varphi}_{\mathsf{q}})^T$ é, de fato, positiva definida.

**Teorema 1** Seja $\boldsymbol{\varphi}_{\mathsf{q}} : R^n \rightarrow R^l$ uma tranformação linear tal que $dim(\mathsf{N})\left(\boldsymbol{\varphi}_{\mathsf{q}}\right) = n - l$, onde $\mathsf{N}\left(\boldsymbol{\varphi}_{\mathsf{q}}\right)$ é o núcleo de $\boldsymbol{\varphi}_{\mathsf{q}}$ e $dim(\mathsf{N})\left(\boldsymbol{\varphi}_{\mathsf{q}}\right)$ é a dimensão do núcleo. Seja ainda $\overline{\mathbf{M}} : R^n \rightarrow R^n$ uma transformação linear positiva definida. Afirmamos que a transformação linear $\boldsymbol{\varphi}_{\mathsf{q}}\overline{\mathbf{M}}\boldsymbol{\varphi}_{\mathsf{q}}^{*} : R^l \rightarrow R^l$, onde $\boldsymbol{\varphi}_{\mathsf{q}}^{*} : R^l \rightarrow R^n$ é a adjunta de $\boldsymbol{\varphi}_{\mathsf{q}}$, é positiva definida.

**Prova** *Para todo* $\mathbf{v} \in R^l$ *seja* $\mathbf{u} = \boldsymbol{\varphi}_{q}^{*}\mathbf{v}$. *Como* $\overline{\mathbf{M}}$ *é positiva definida, temos que qualquer que seja* $\mathbf{u}$, $\langle \mathbf{u}, \overline{\mathbf{M}}\mathbf{u}\rangle \geq 0$ *e* $\langle \mathbf{u}, \overline{\mathbf{M}}\mathbf{u}\rangle = 0$ *se, e somente se,* $\mathbf{u} = \mathbf{0}$. *Como* $\langle \mathbf{v}, \boldsymbol{\varphi}_{q}\overline{\mathbf{M}}\boldsymbol{\varphi}_{q}^{*}\mathbf{v}\rangle = \langle \boldsymbol{\varphi}_{q}^{*}\mathbf{v}, \overline{\mathbf{M}}\boldsymbol{\varphi}_{q}^{*}\mathbf{v}\rangle$, *basta provar que* $\boldsymbol{\varphi}_{q}^{*}\mathbf{v} = \mathbf{0}$ *se, e somente se,* $\mathbf{v} = \mathbf{0}$. *Como* $dim(\boldsymbol{Im}(\boldsymbol{\varphi}_{q})) = dim(\boldsymbol{Im}(\boldsymbol{\varphi}_{q}^{*})) = l$, *onde* $\boldsymbol{Im}(\boldsymbol{\varphi}_{q})$ *é a imagem de* $\boldsymbol{\varphi}_{q}$, *temos que* $dim(N)\left(\boldsymbol{\varphi}_{q}^{*}\right) = 0$, *o que prova o teorema.* $\triangleleft$

Note que a abordagem empregada na formulação e prova do teorema consiste em transformações lineares. Embora esta abordagem seja a mais apropriada num curso de Álgebra Linear, onde estudam-se os espaços vetoriais a as transformações lineares entre eles, em cursos de engenharia, infelizmente, é comum a apresentação da Álgebra Linear através do conceito de matrizes. No nosso caso, o uso direto das transformações lineares, além de permitir uma compacidade maior em relação à prova, cuja validade independe de sistemas de coordenadas, permite um maior aprofundamento dos conceitos da Álgebra Linear. Em (LIMA, 2004), (SHILOV, 1977) há uma boa fonte de consulta e estudo sobre Álgebra Linear. Obseve também que a única propriedade utilizada da transformação linear $\overline{\mathbf{M}} : R^n \rightarrow R^n$ é o fato desta ser positiva definida. Dessa forma, pode-se, na realização matricial desta

transformação linear, utilizar uma matriz de massa não necessariamente diagonal.

Para aplicarmos a fórmula de Doolittle à matriz representada com o número três na Fig. (A.2) devemos observar que esta matriz é inicialmente nula. Dessa forma, os elementos $a_{ij}$ nas fórmulas de Doolittle serão iguais a zero. Uma grande vantagem na implementação computacional realizada do método $\mathbf{LDL}^T$ reside no fato deste eliminar a necessidade de zerar a matriz cada vez que formos resolver o sistema linear para as acelerações, Eq. (A.1). De fato, a matriz de massa é constante. Na parte do jacobiano não há preenchimentos, permitindo que os elementos novos do jacobiano sejam escritos sobre os antigos e, conhecendo antecipadamente o padrão de preenchimento da matriz nula (número três na Fig. A.2), podemos simplesmente, à semelhança do jacobiano, escrever os novos valores sobre os antigos.

Na implementação computacional, a estrutura de dados utilizada para armazenar os elementos da matriz dos coeficientes é constituída num array cujos elementos são objetos que possuem como um dos membros um array alocável. Esses arrays armazenarão os elementos tanto do jacobiano como os de preenchimento da matriz nula, respectivamente indicados pelos números dois e três na Fig. (A.2). A Fig. (A.6) ilustra o array alocado para os objetos, ilustrado em verde, onde cada objeto possui um array alocado armazenando tanto os elementos do jacobiano quanto os preenchimentos da matriz inicialmente nula.

Agora, vamos introduzir o *solver* do ponto de vista esquemático. A Fig. (A.7) ilustra o fluxograma do solver implementado. Nas figuras mostrando o funcionamento do solver, as setas em vermelho indicam o caminho seguido caso a variável de teste seja verdadeira. Inicialmente o solver chama rotinas para realizar a alocação de dados e realizar a análise do modelo, identificando equações de restrição redundantes e alterando a ordem das equações de restrição visando à minimização de preenchimentos. Nesta parte é realizada a análise de

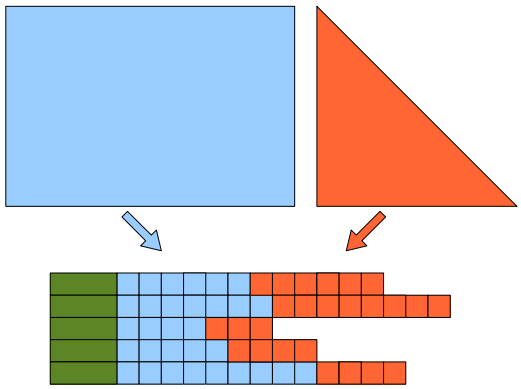


FIGURA A.6 – Gráfico ilustrando a estrutura de dados implementada

pré-processamento para a fatoração da matriz dos coeficientes, anteriormente apresentada neste anexo. Realizada esta análise, o array de valores anteriormente calculados, utilizados no método preditor-corretor, é formado fazendo uso da fórmula de Merson. Após esta fase, o solver entra em sua parte principal, constituída de um loop principal, com alguns loops aninhados. De fato, a maioria dos códigos intensivos do ponto de vista de operações em ponto flutuante possuem esta forma padrão. Neste loop principal podemos identificar três partes importantes, responsáveis pelo avanço do integrador, pela realização da projeção e pelo armazenamento de dados. Cada uma dessas partes possui loops aninhados. No caso do integrador, é necessário verificar se a tolerância especificada foi cumprida. Caso o passo seja rejeitado, volta-se ao início do integrador e realiza-se o avanço deste com um passo menor até que a integração seja bem sucedida ou que o passo seja inferior a um valor pré-estabelecido. A projeção, mostrada em fluxograma na Fig. (A.8), possui um loop aninhado representando a parte iterativa, de solução para as posições, do método. A parte de alocação de dados possui um loop visto que o integrador pode dar um passo que ultrapasse mais de um ponto de armazenagem de dados, já que a freqüência desta armazenagem é especificada pelo usuário.

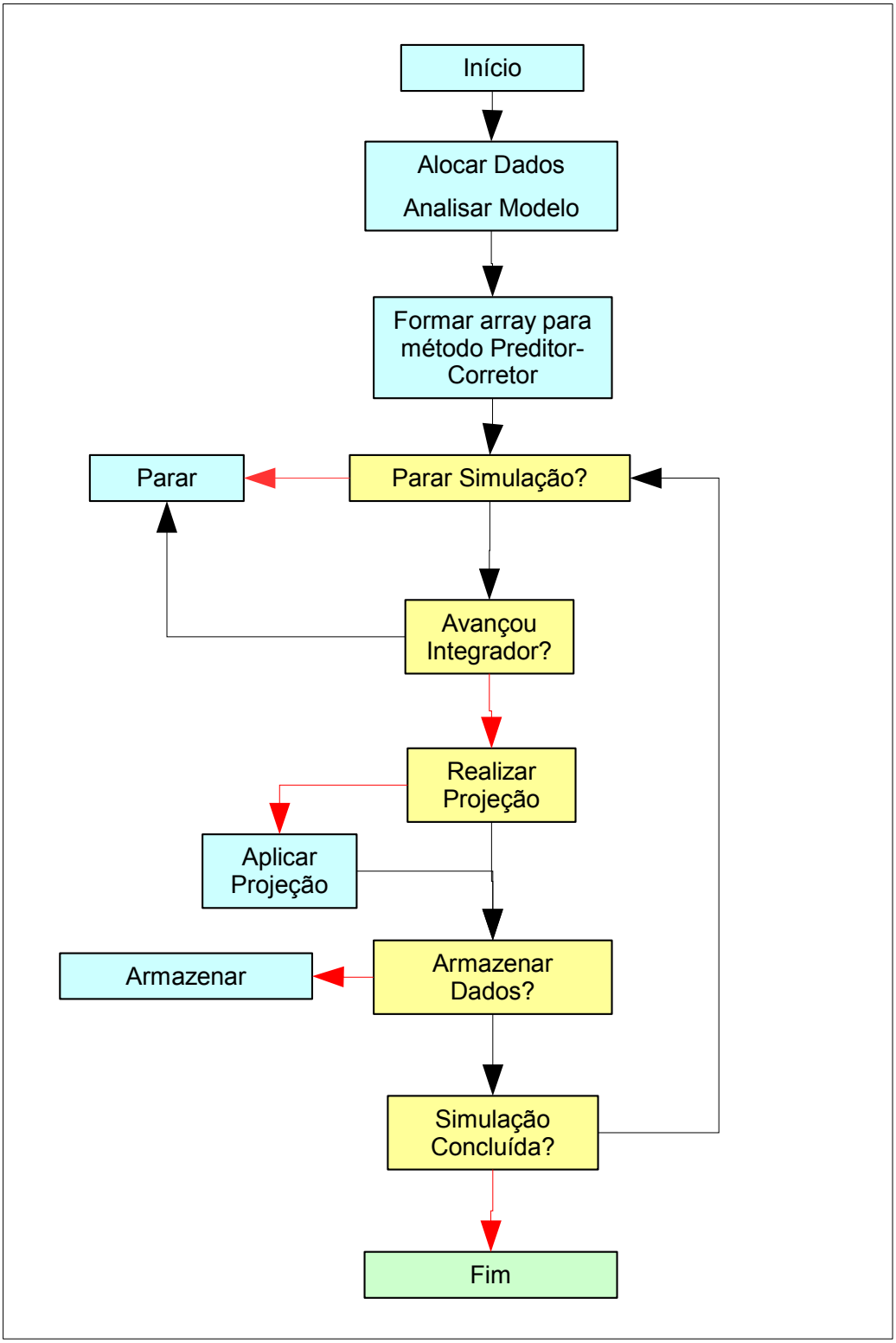



FIGURA A.7 – Fluxograma do solver implementado.

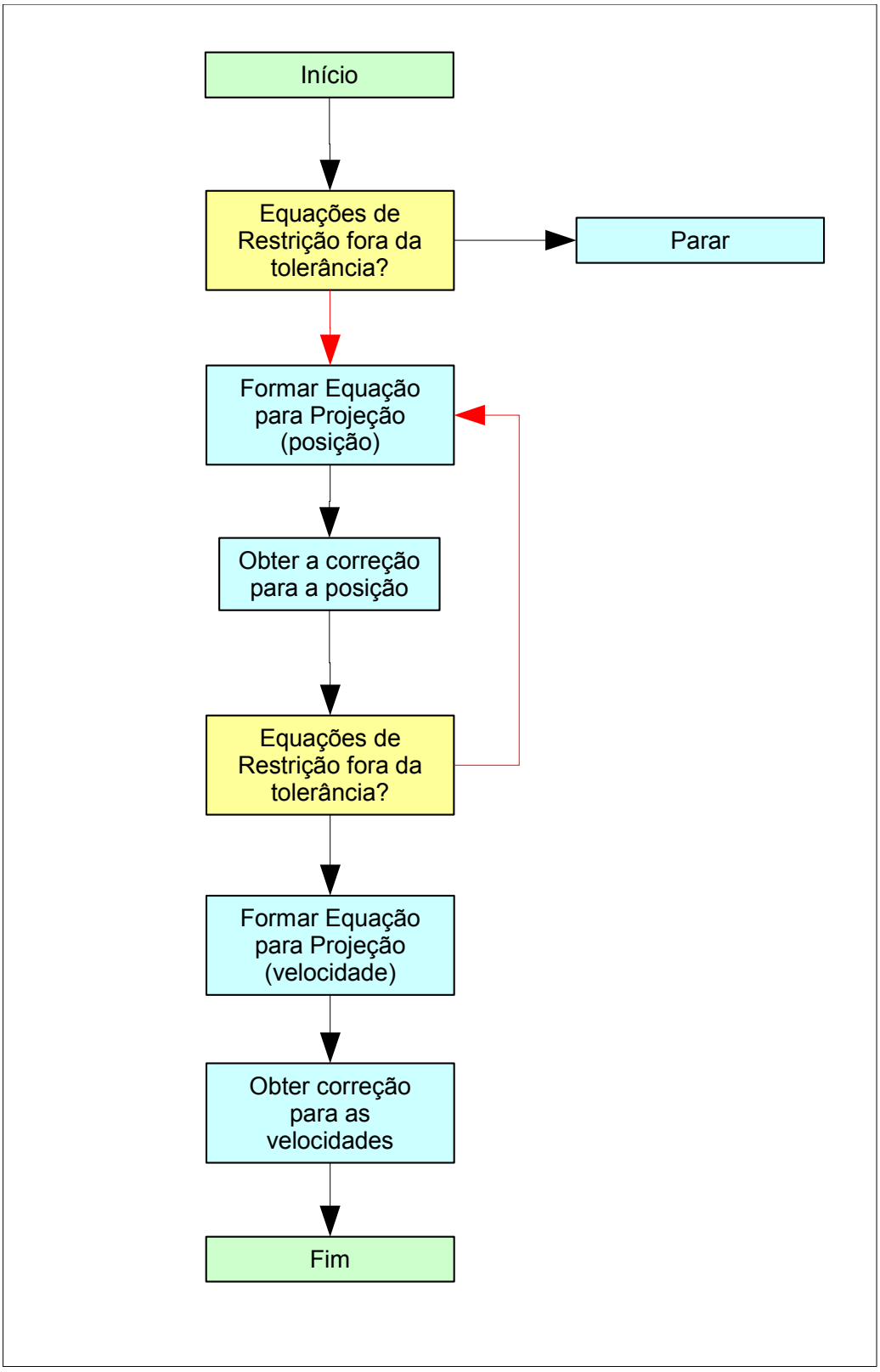
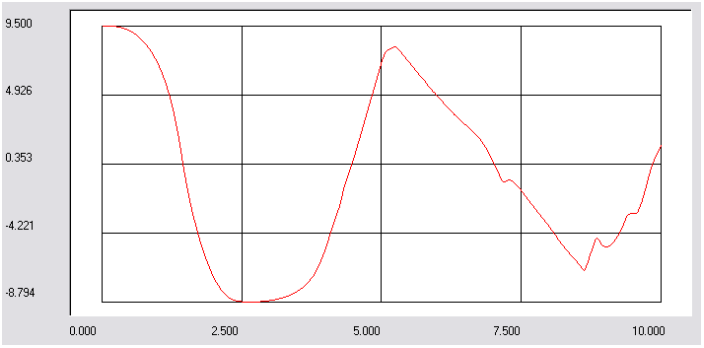



FIGURA A.8 – Fluxograma indicando o método de projeção implementado.

# Anexo B - Estudo de Caso I - Figuras

**(a)**




**(b)**

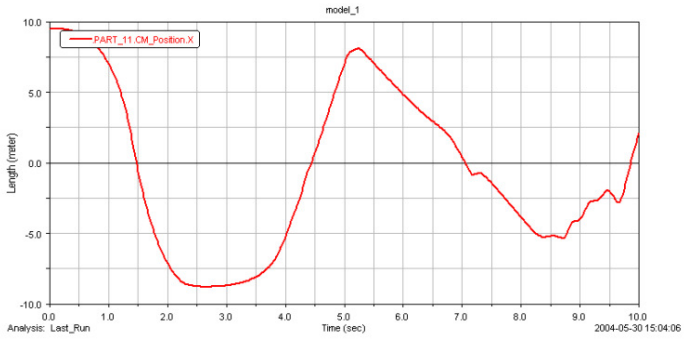
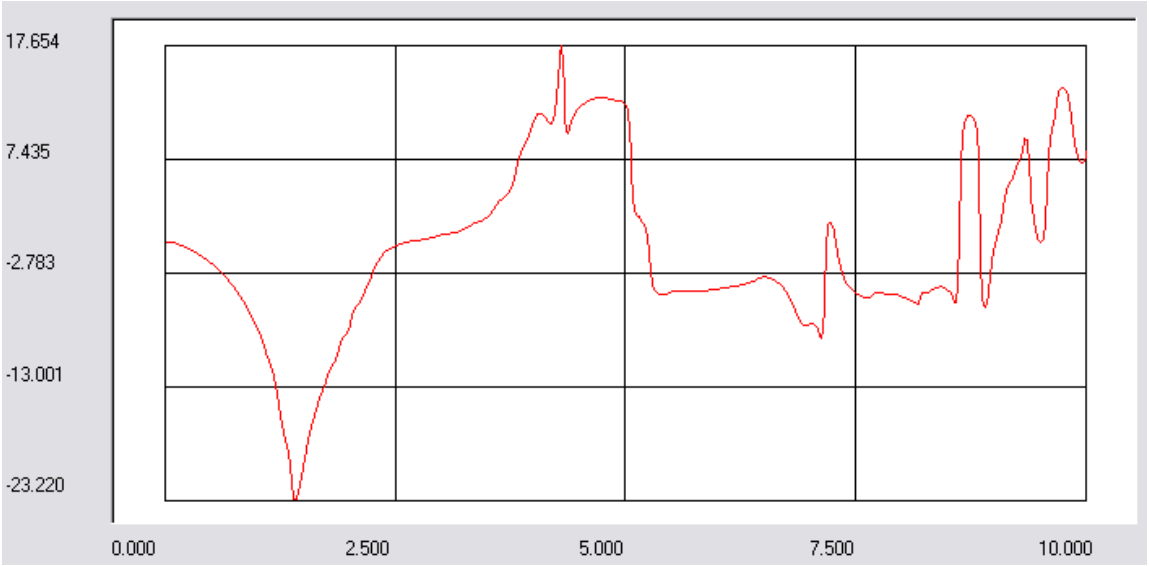



FIGURA B.1 – Gráficos para $x$

**(a)**




**(b)**

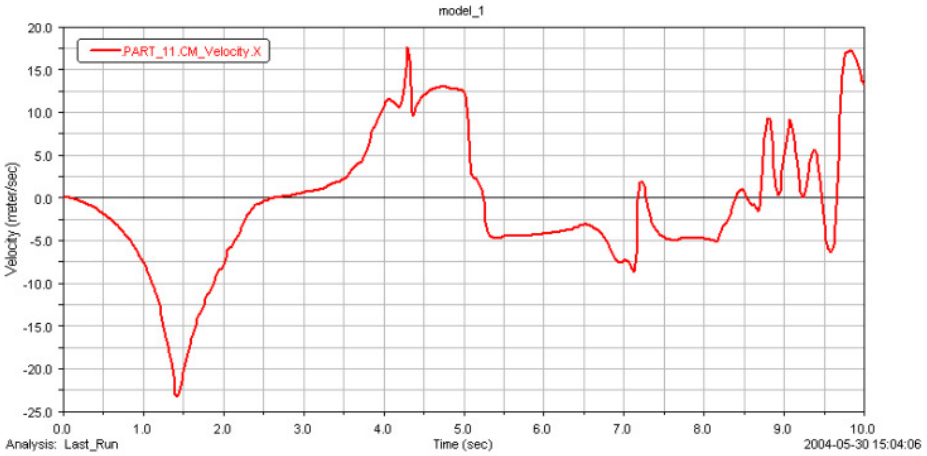
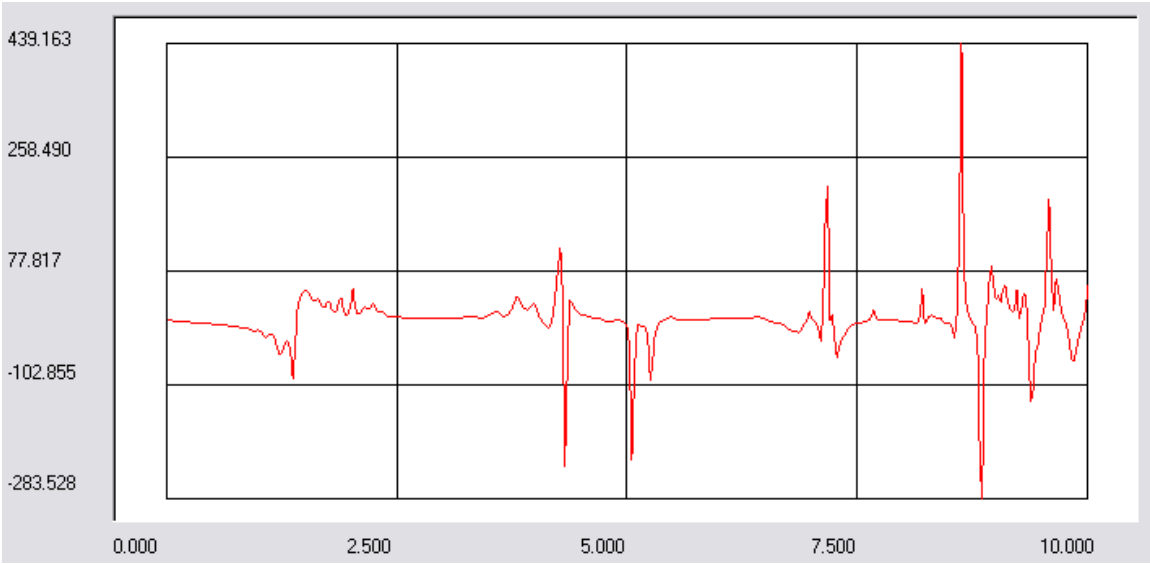



FIGURA B.2 – Gráficos para $\dot{x}$

**(a)**




**(b)**

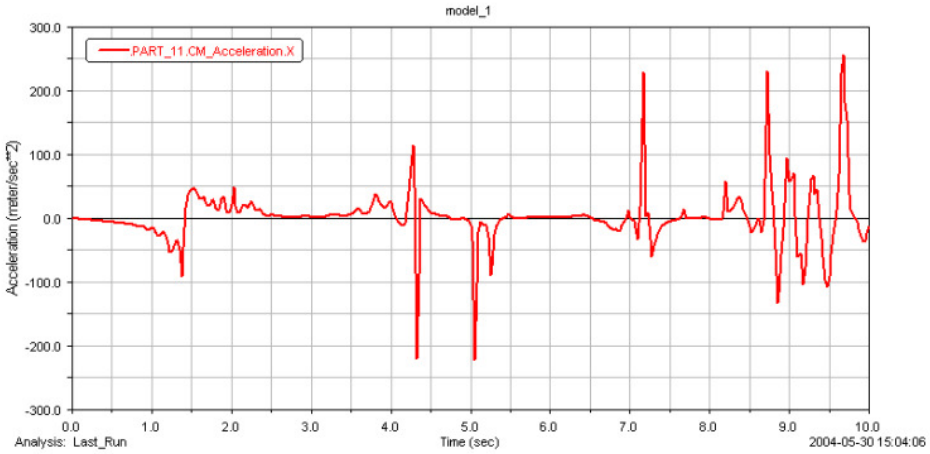
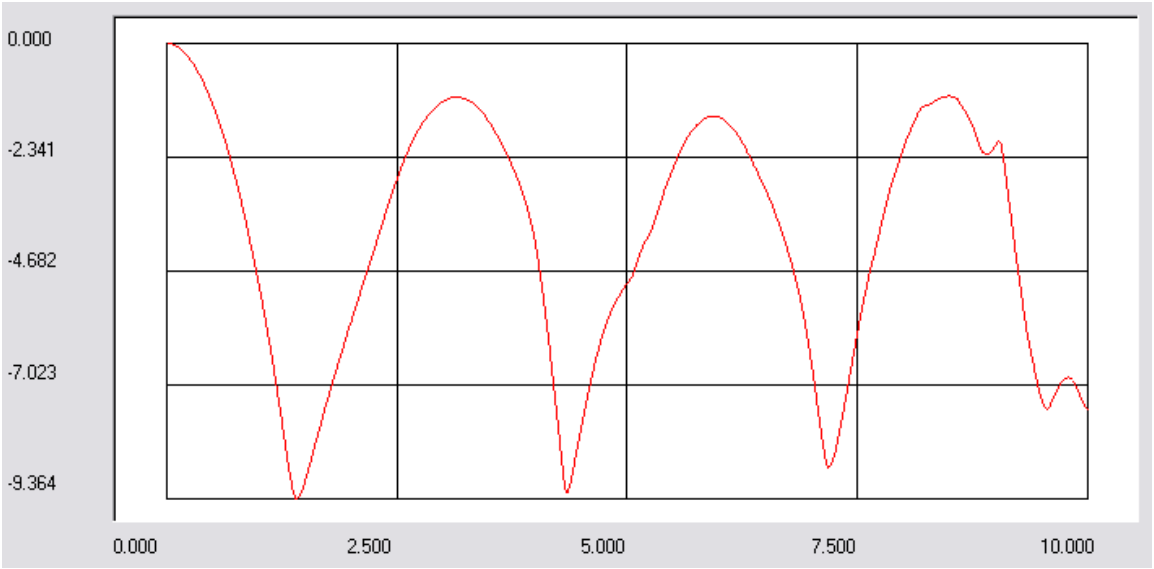



FIGURA B.3 – Gráficos para $\ddot{x}$

**(a)**




**(b)**

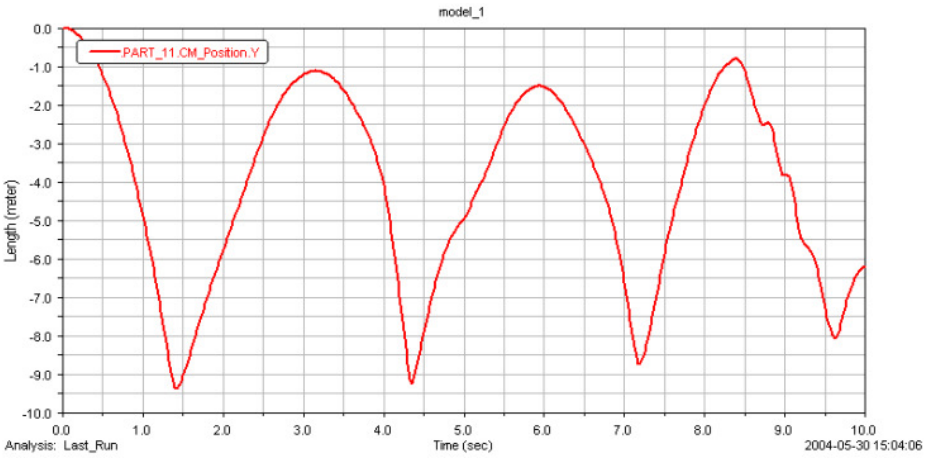
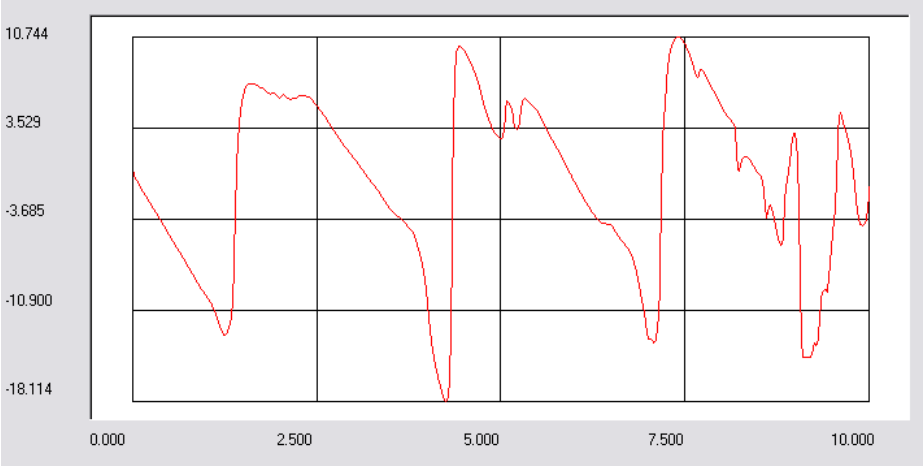



FIGURA B.4 – Gráficos para $y$

**(a)**




**(b)**

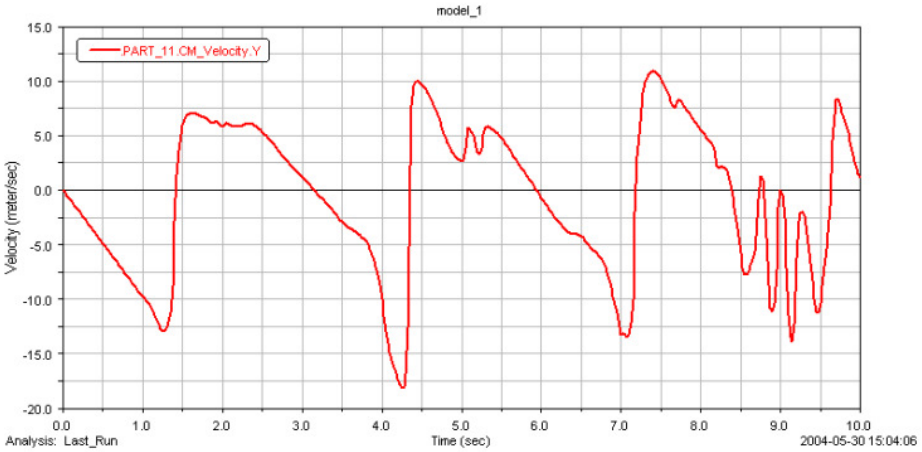
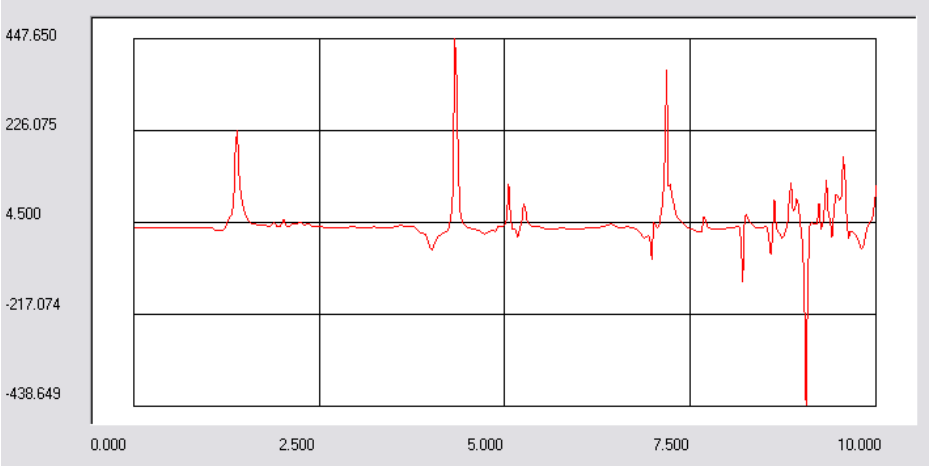



FIGURA B.5 – Gráficos para $\dot{y}$

**(a)**




**(b)**

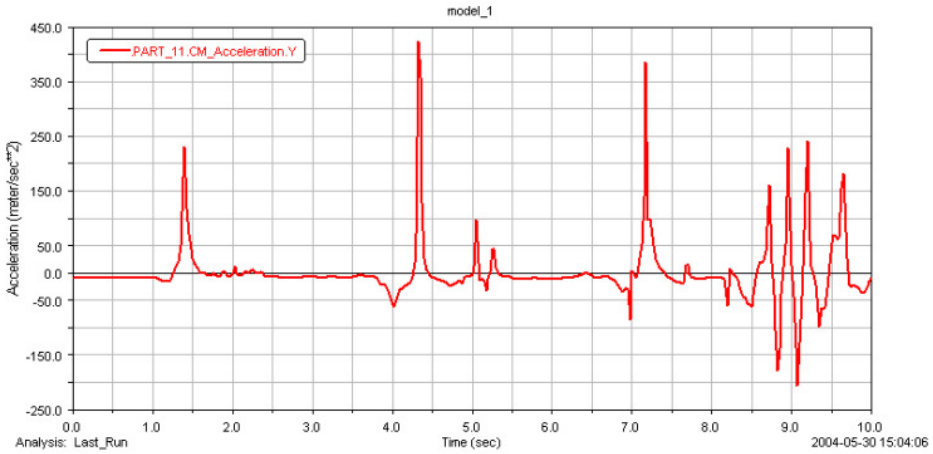
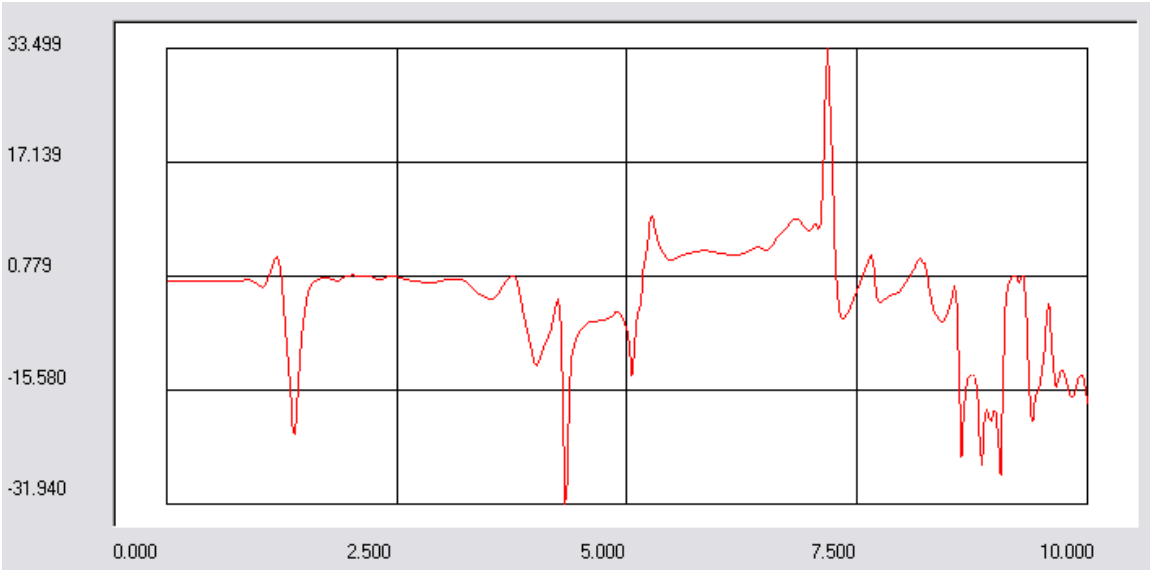



FIGURA B.6 – Gráficos para $\ddot{y}$

**(a)**




**(b)**

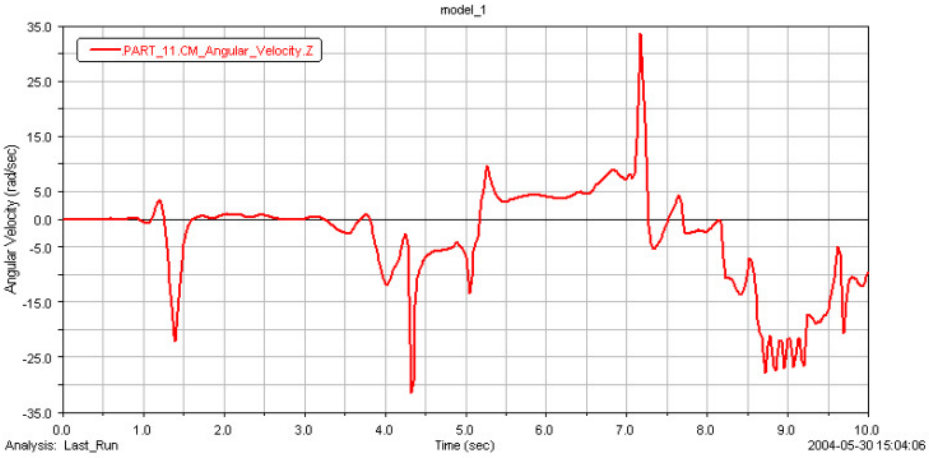
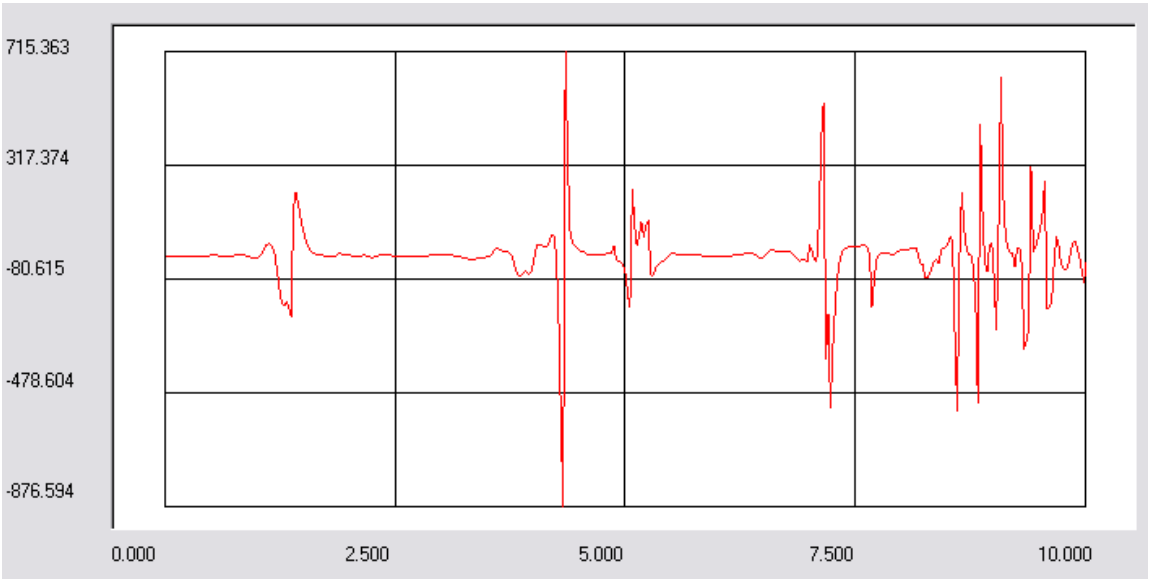



FIGURA B.7 – Gráficos para a velocidade angular

**(a)**




**(b)**

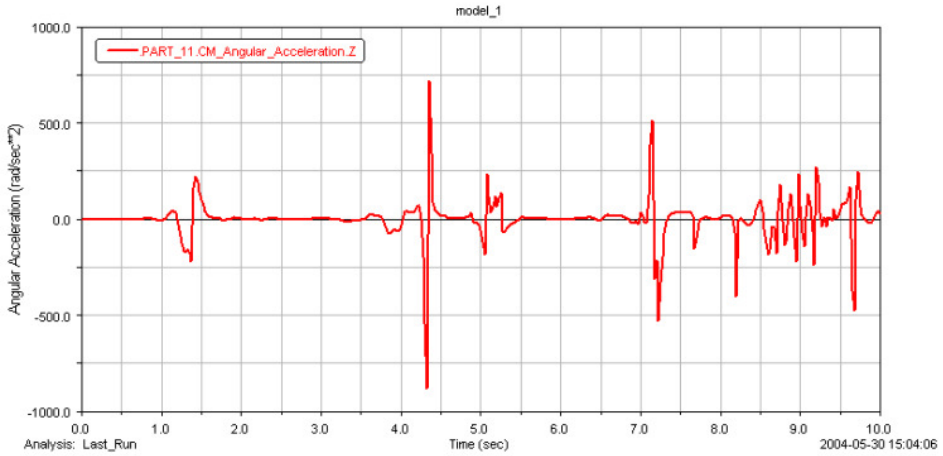
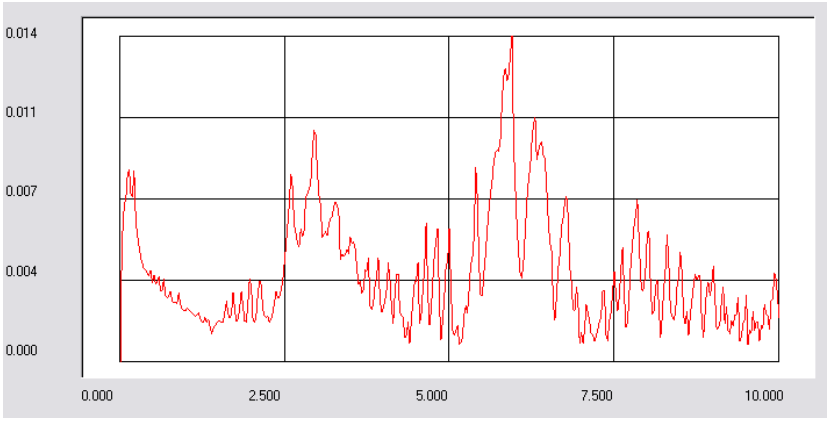
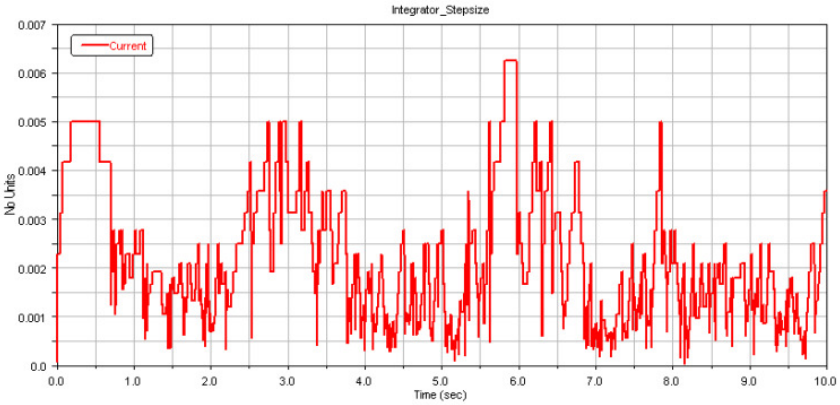



FIGURA B.8 – Gráficos para a aceleração angular

**(a)**




**(b)**




**(c)**

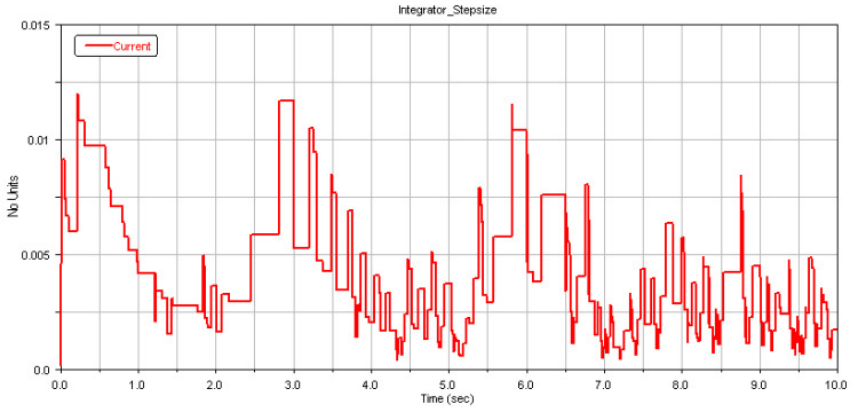
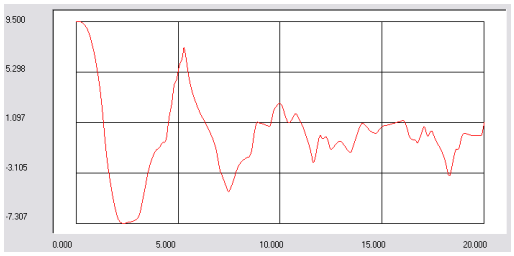


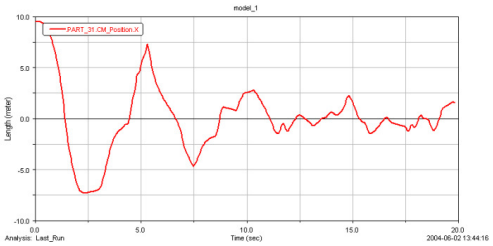

FIGURA B.9 – Gráficos para o passo

# Anexo C - Estudo de Caso II - Figuras

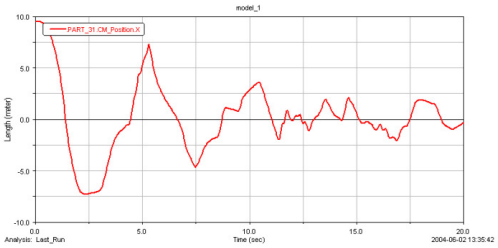
**(a)**



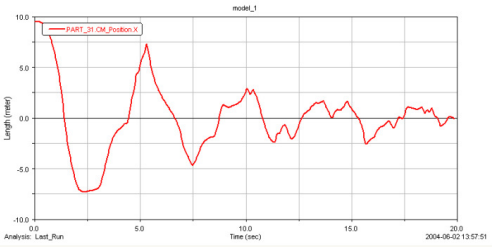

**(b)**



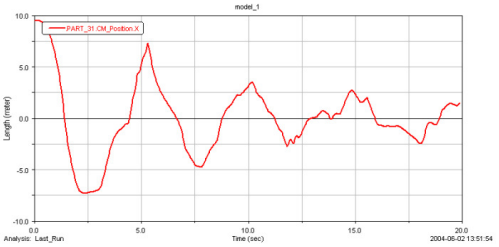

**(c)**




**(d)**




**(e)**




**(f)**

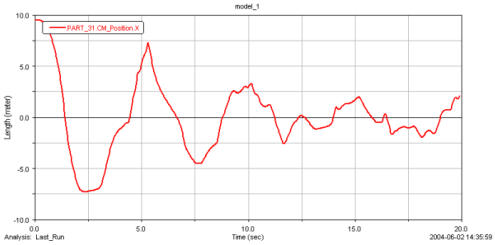
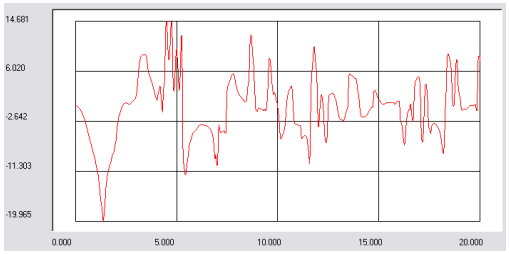
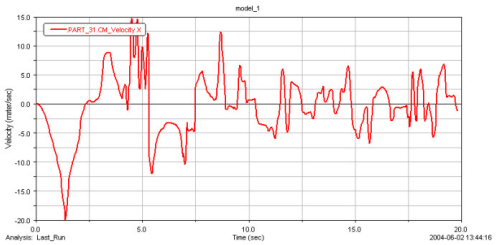



FIGURA C.1 – Gráficos para $x$

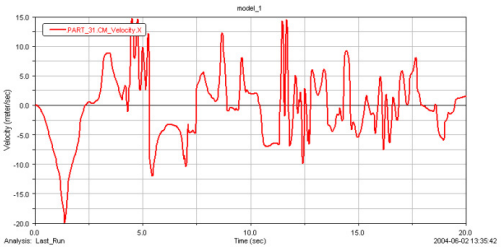
**(a)**



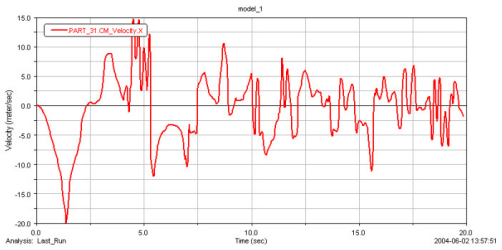

**(b)**



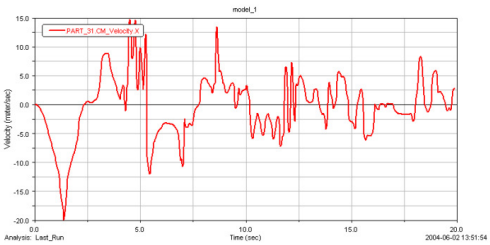

**(c)**




**(d)**




**(e)**




**(f)**

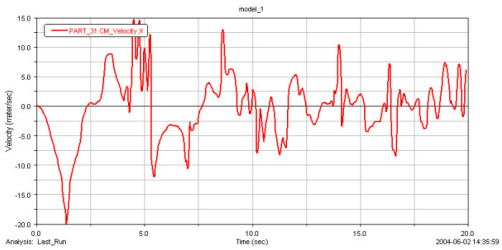
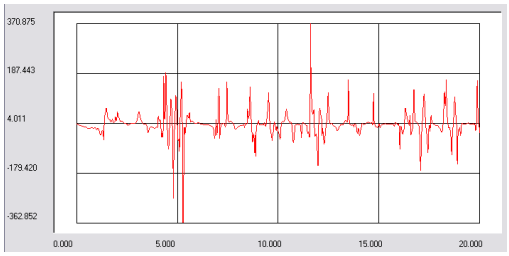
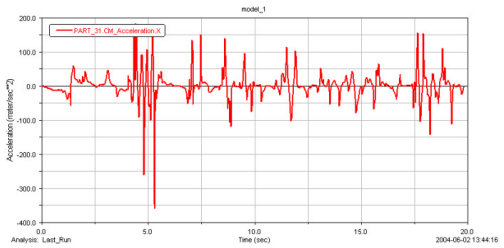



FIGURA C.2 – Gráficos para $\dot{x}$

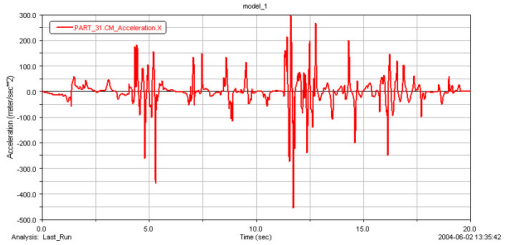
**(a)**



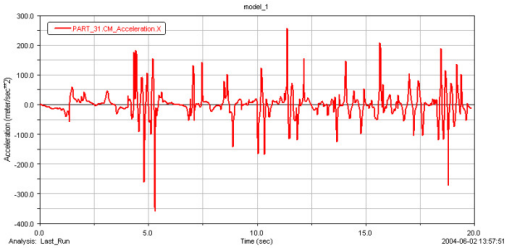

**(b)**



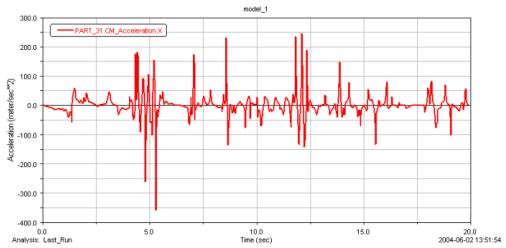

**(c)**




**(d)**




**(e)**




**(f)**

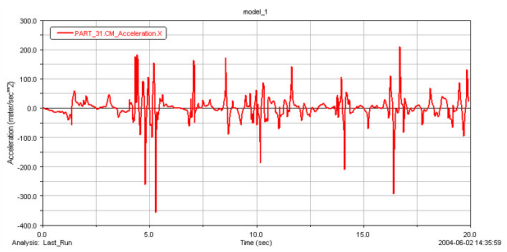
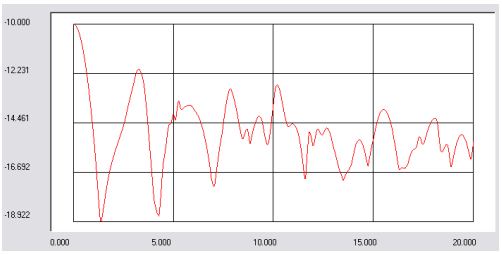
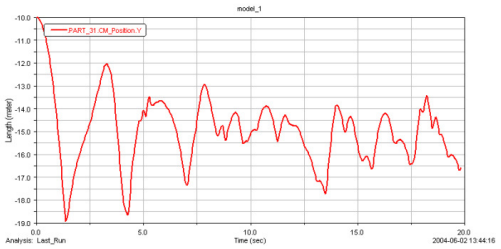



FIGURA C.3 – Gráficos para $\ddot{x}$

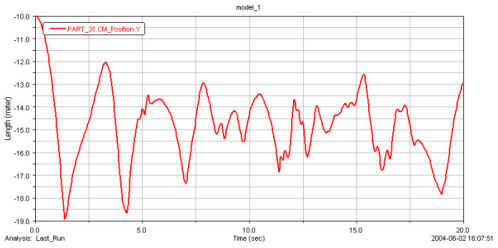
**(a)**



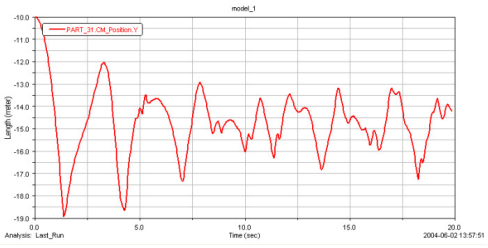

**(b)**



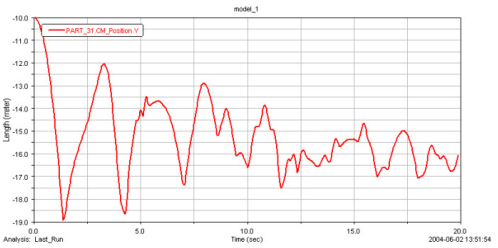

**(c)**




**(d)**




**(e)**




**(f)**

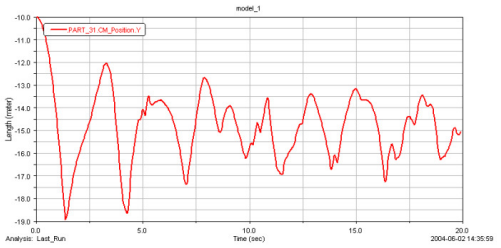
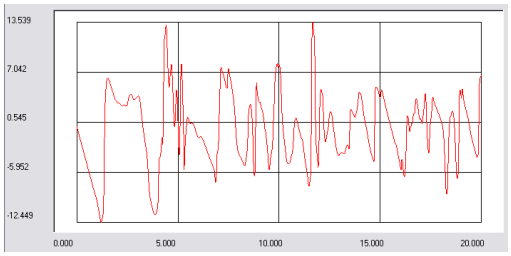
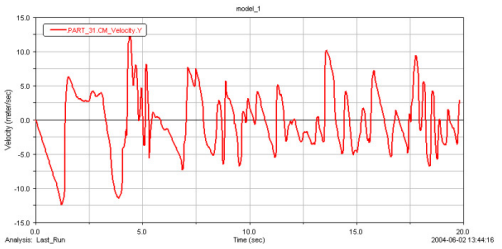



FIGURA C.4 – Gráficos para $y$

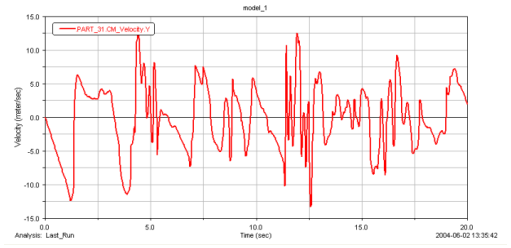
**(a)**



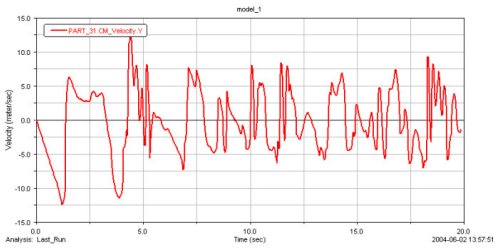

**(b)**



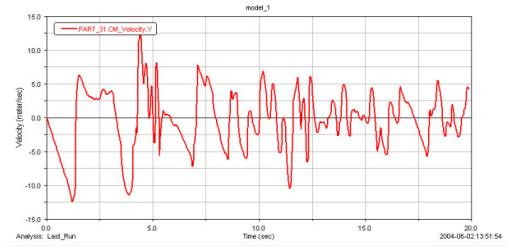

**(c)**




**(d)**




**(e)**




**(f)**

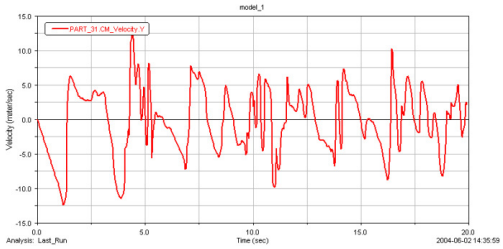
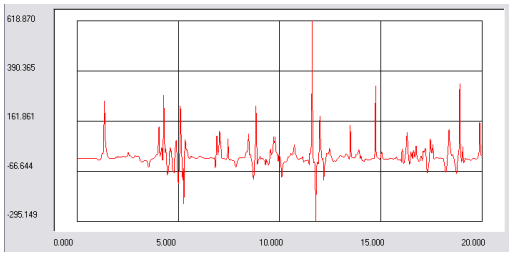
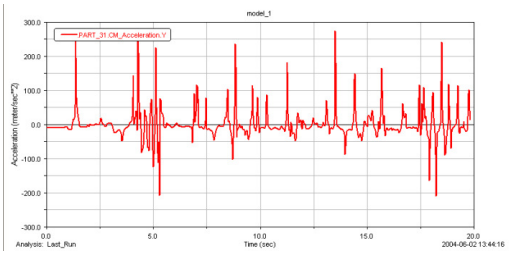



FIGURA C.5 – Gráficos para $\dot{y}$

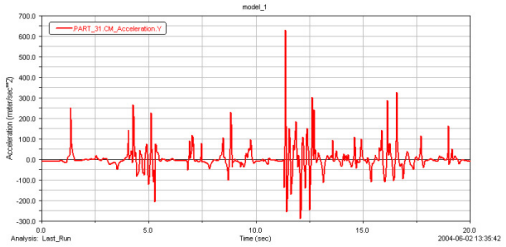
**(a)**



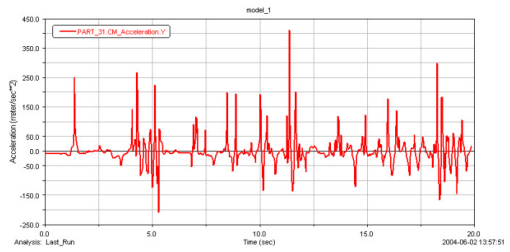

**(b)**



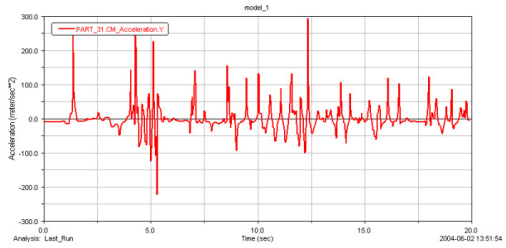

**(c)**




**(d)**




**(e)**




**(f)**

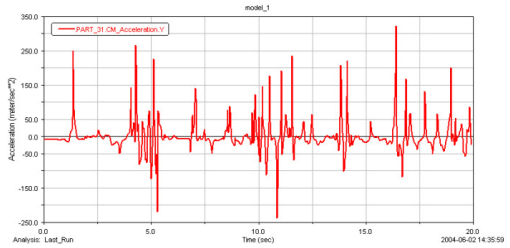
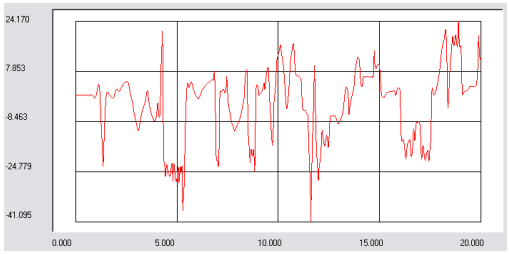
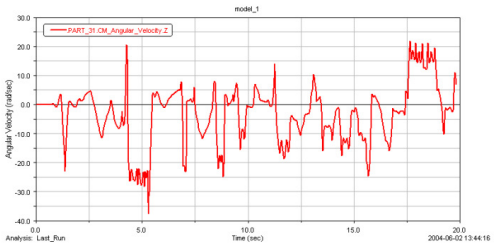



FIGURA C.6 – Gráficos para $\ddot{y}$

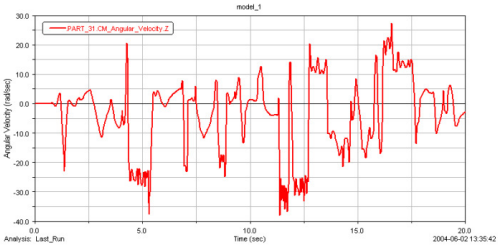
**(a)**



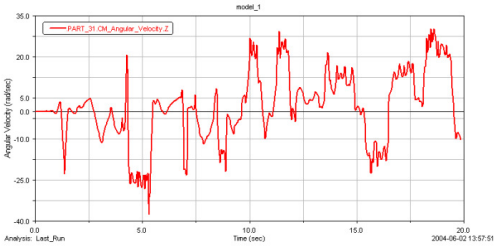

**(b)**



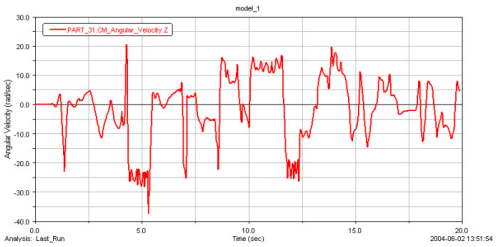

**(c)**




**(d)**




**(e)**




**(f)**

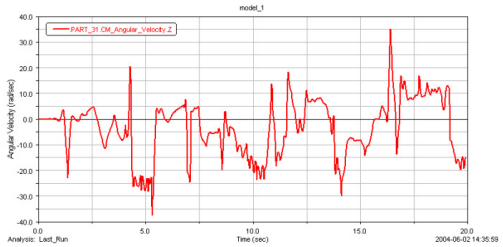
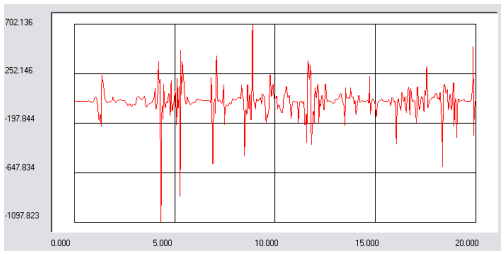
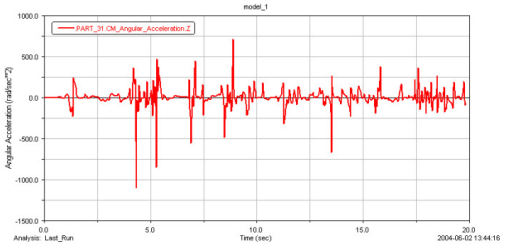



FIGURA C.7 – Gráficos para a velocidade angular

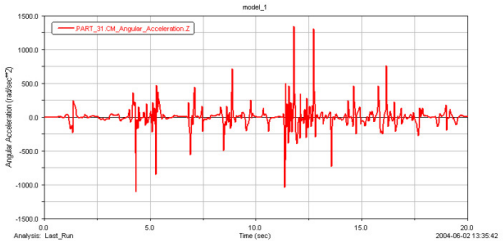
**(a)**



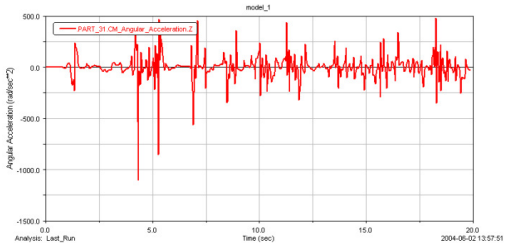

**(b)**



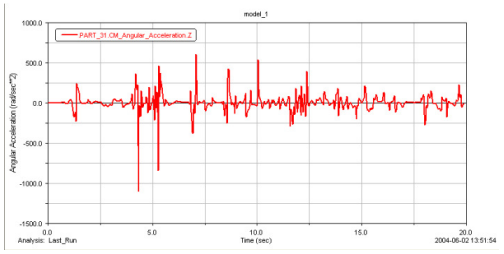

**(c)**




**(d)**




**(e)**




**(f)**

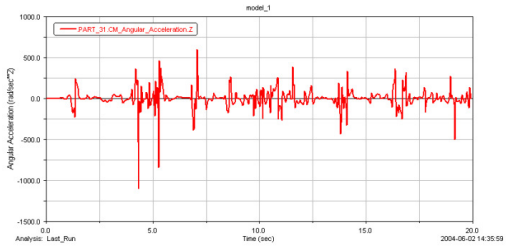
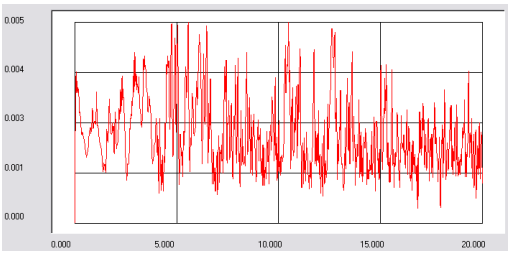
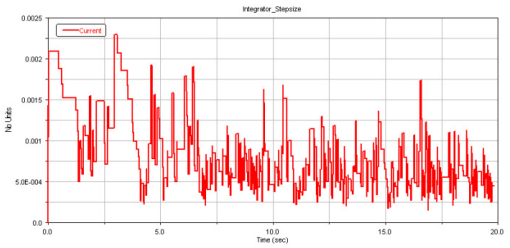



FIGURA C.8 – Gráficos para a aceleração angular

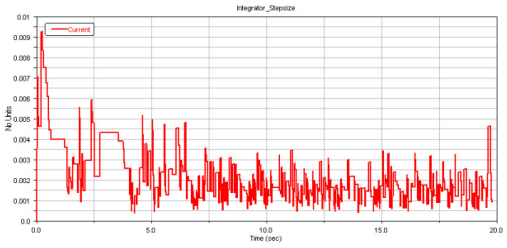
**(a)**



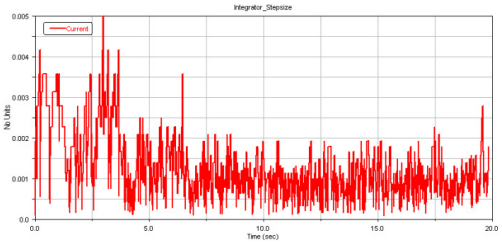

**(b)**



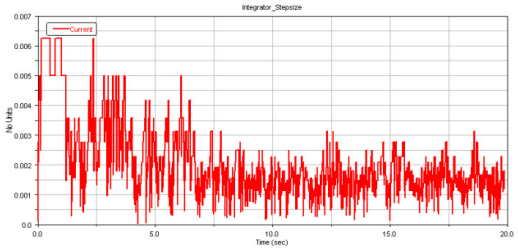

**(c)**




**(d)**




**(e)**




**(f)**

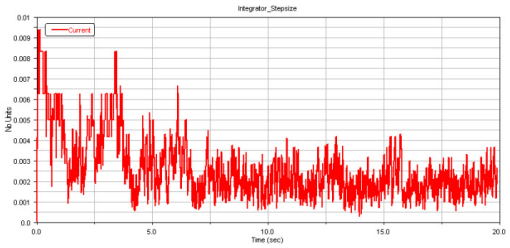
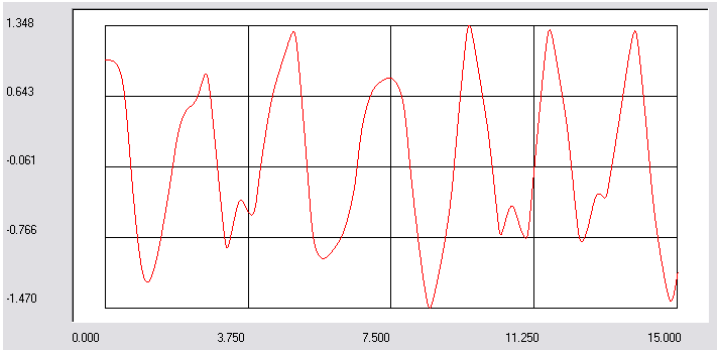



FIGURA C.9 – Gráficos para o passo do integrador

# Anexo D - Estudo de Caso III - Figuras

**(a)**




**(b)**

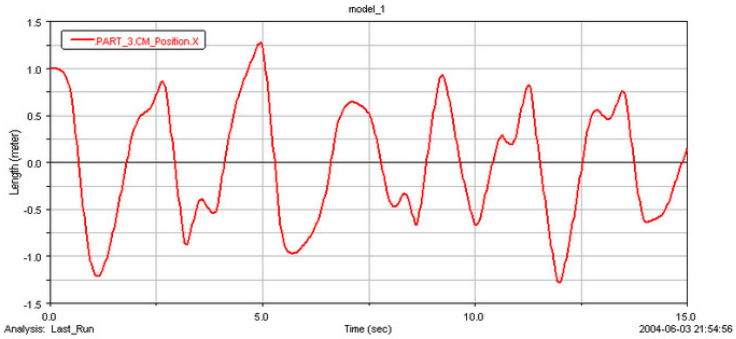
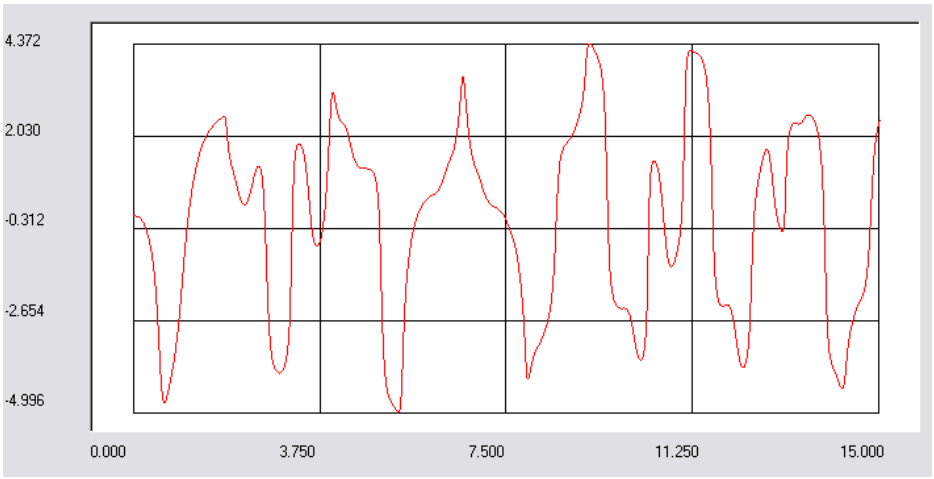



FIGURA D.1 – Gráficos para $x$

**(a)**




**(b)**

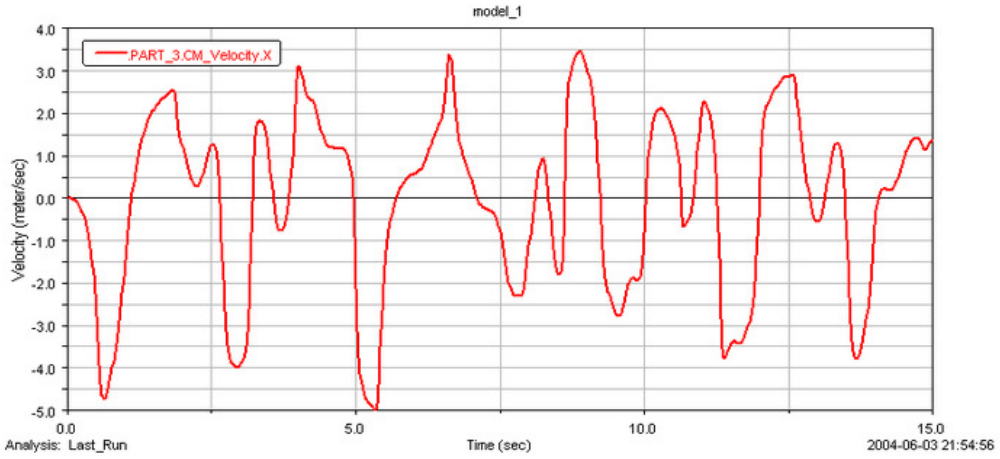
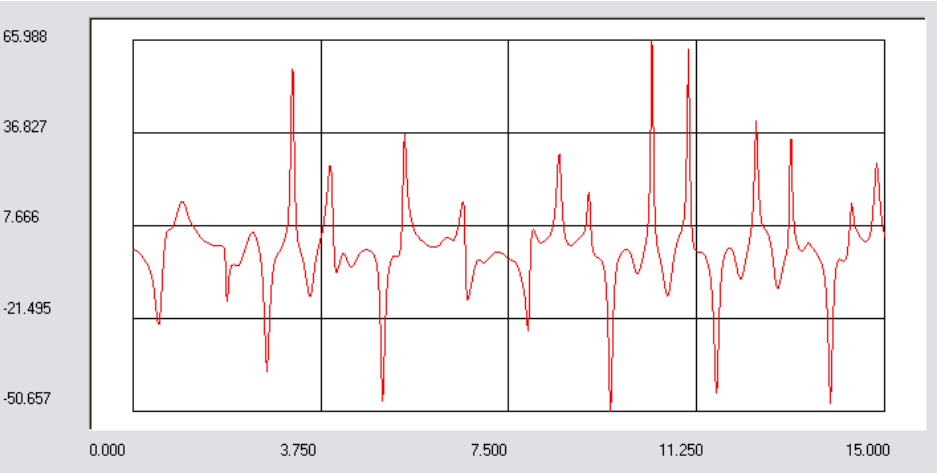



FIGURA D.2 – Gráficos para $\dot{x}$

**(a)**




**(b)**

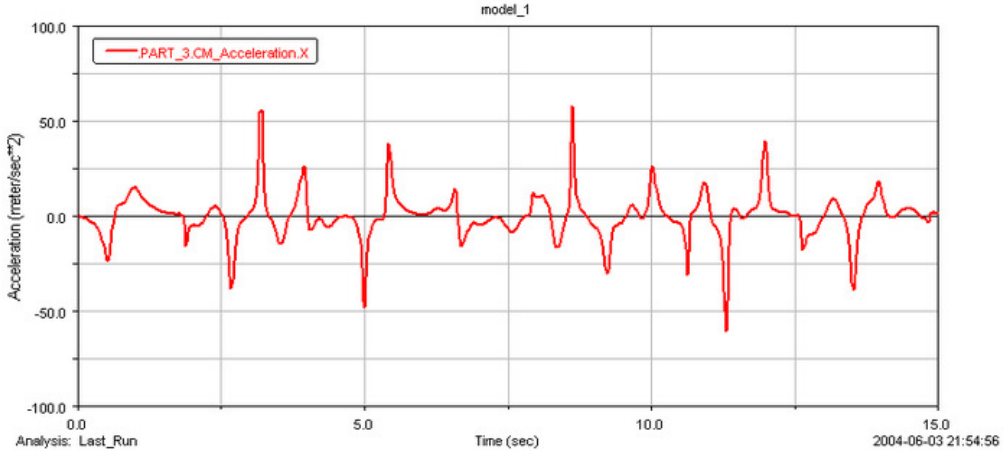
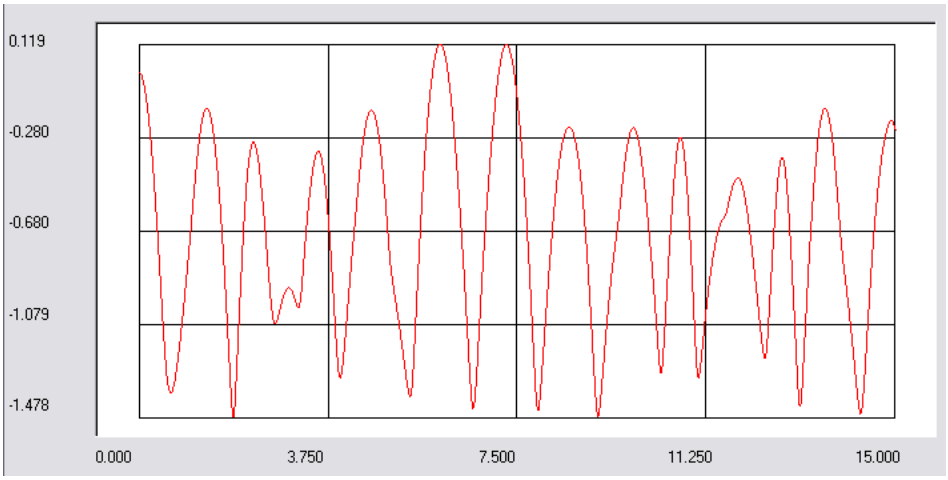



FIGURA D.3 – Gráficos para $\ddot{x}$

**(a)**




**(b)**

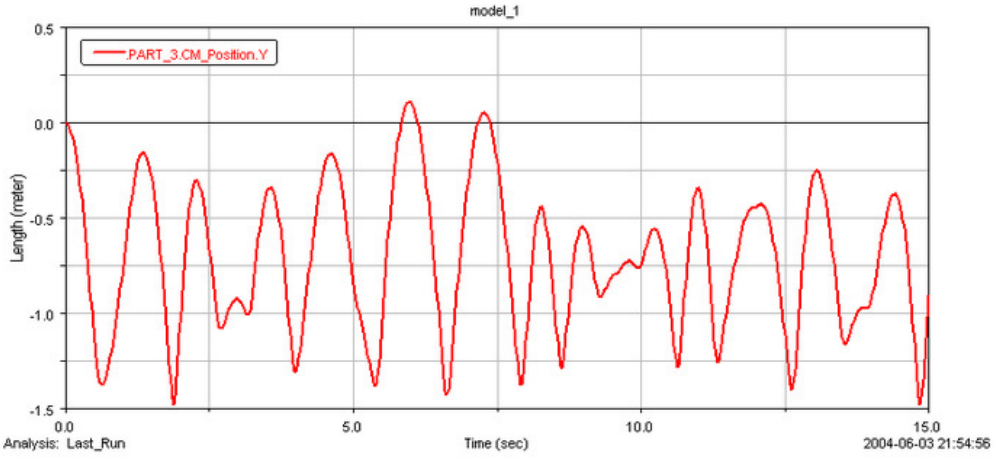
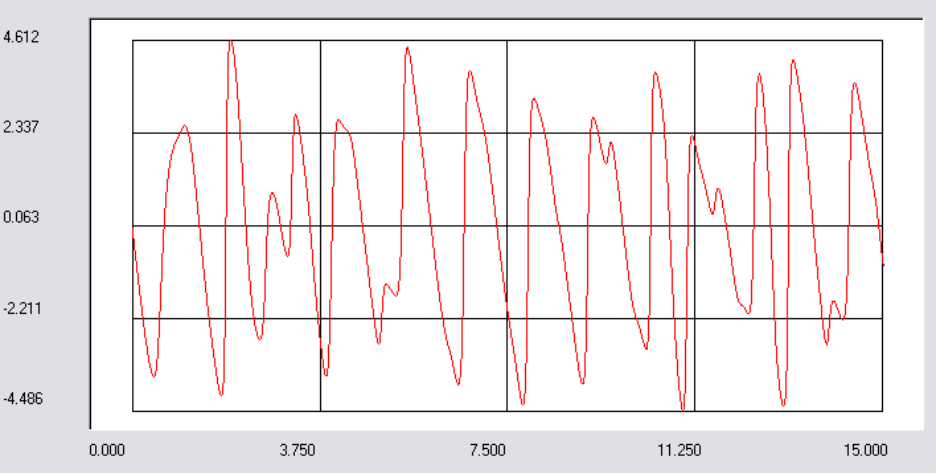



FIGURA D.4 – Gráficos para $y$

**(a)**




**(b)**

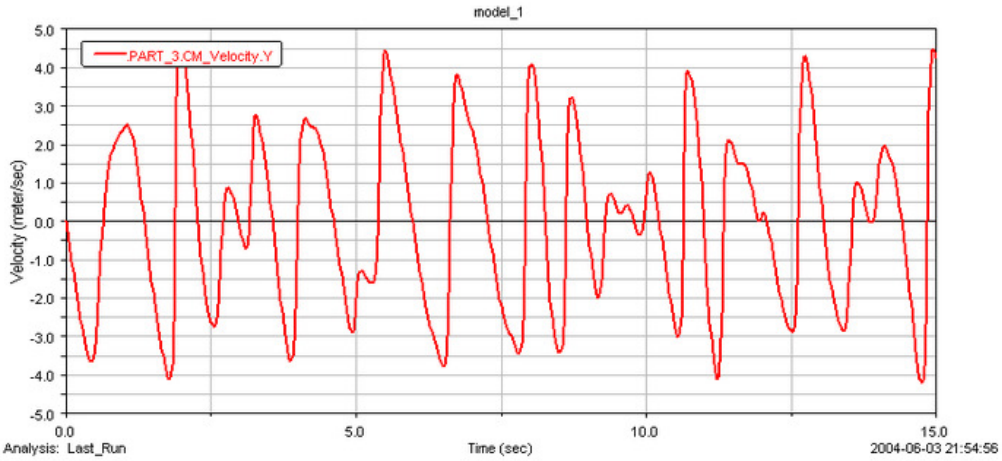
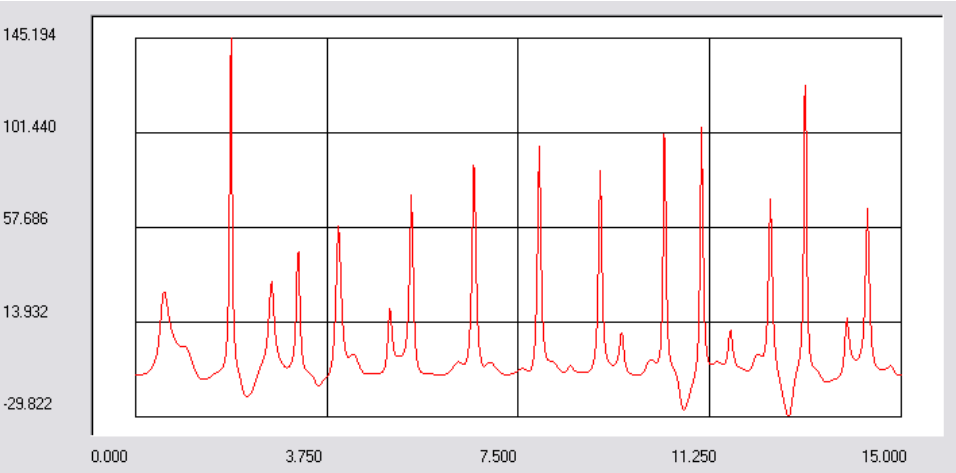



FIGURA D.5 – Gráficos para $\dot{y}$

**(a)**




**(b)**

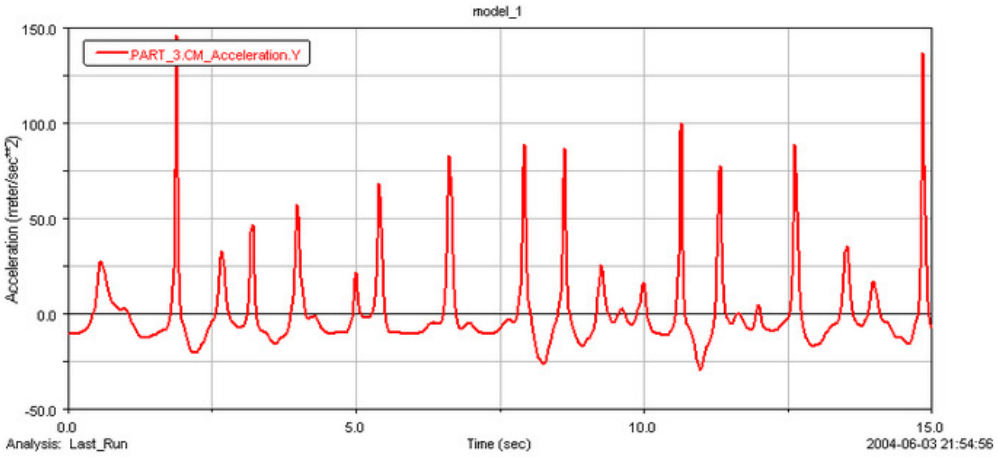
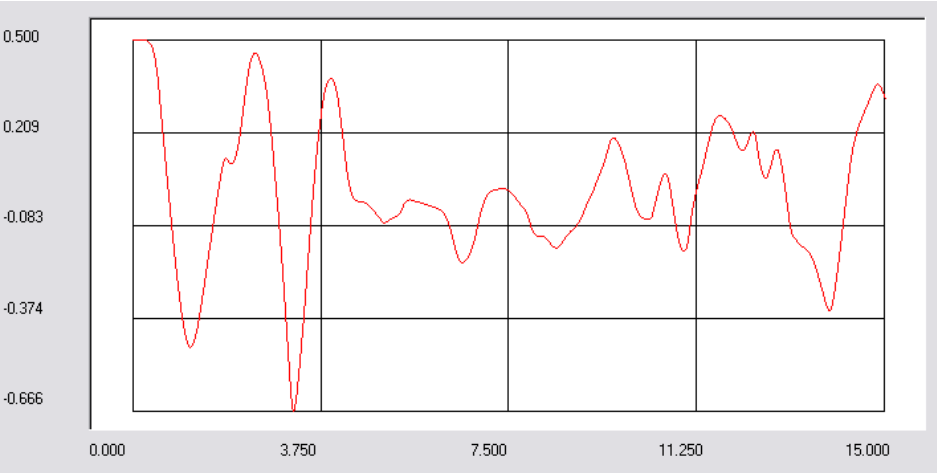



FIGURA D.6 – Gráficos para $\ddot{y}$

**(a)**




**(b)**

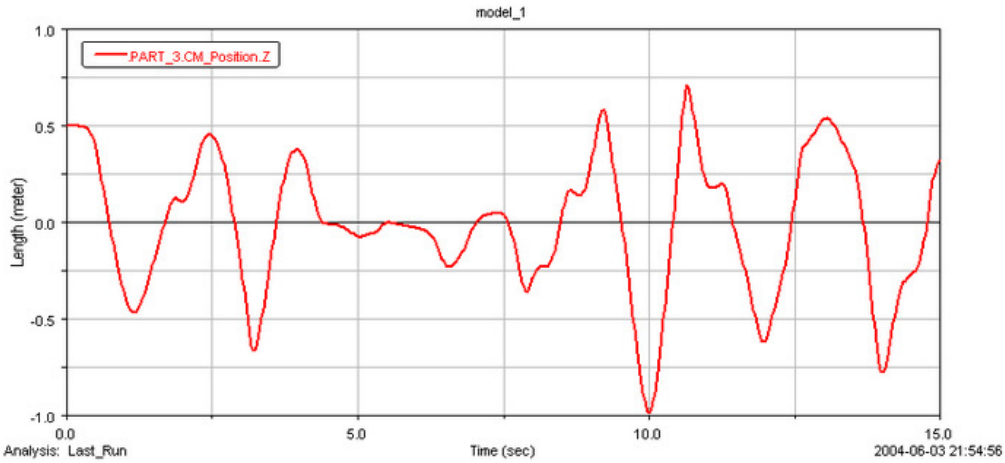
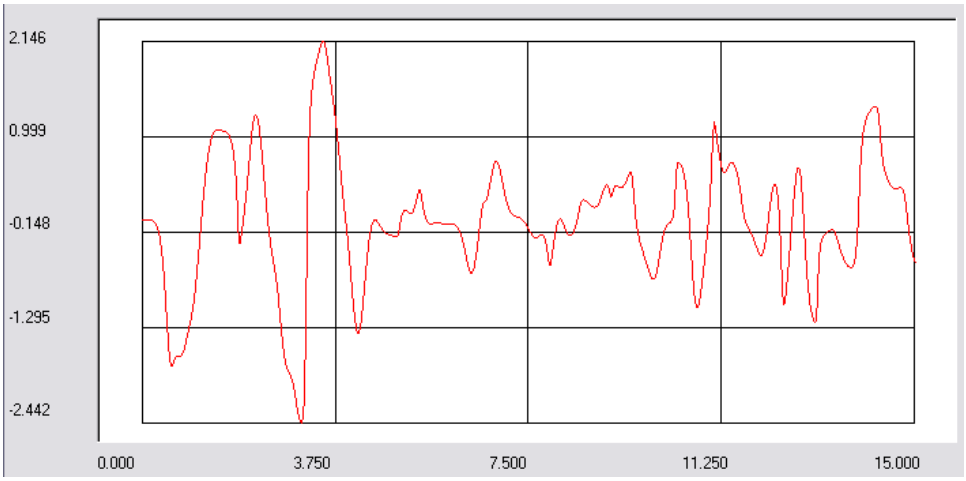



FIGURA D.7 – Gráficos para $z$

**(a)**




**(b)**

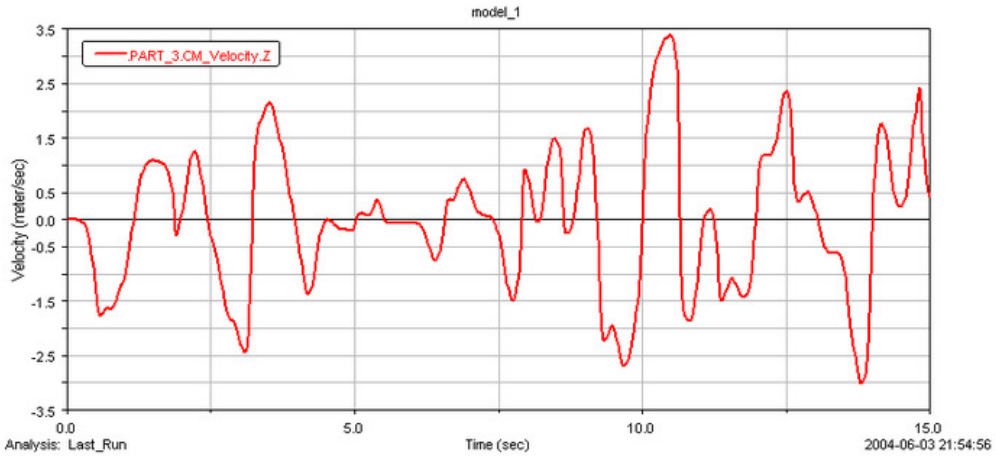
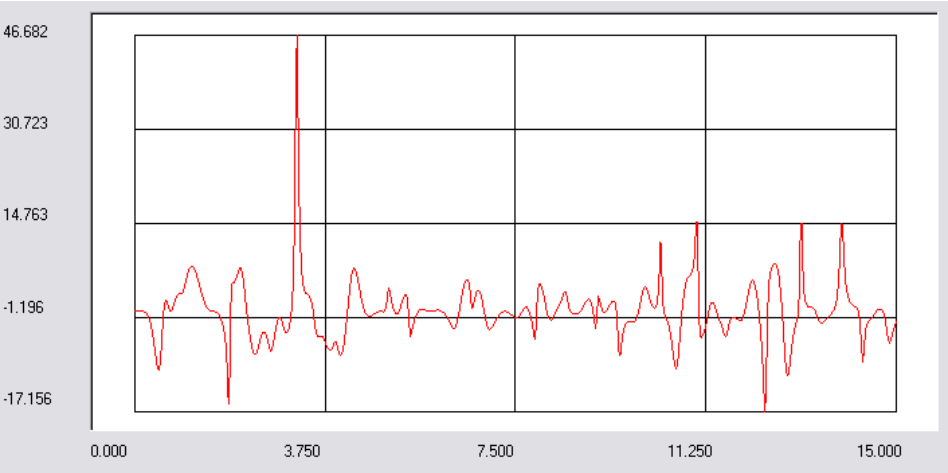



FIGURA D.8 – Gráficos para $\dot{z}$

**(a)**




**(b)**

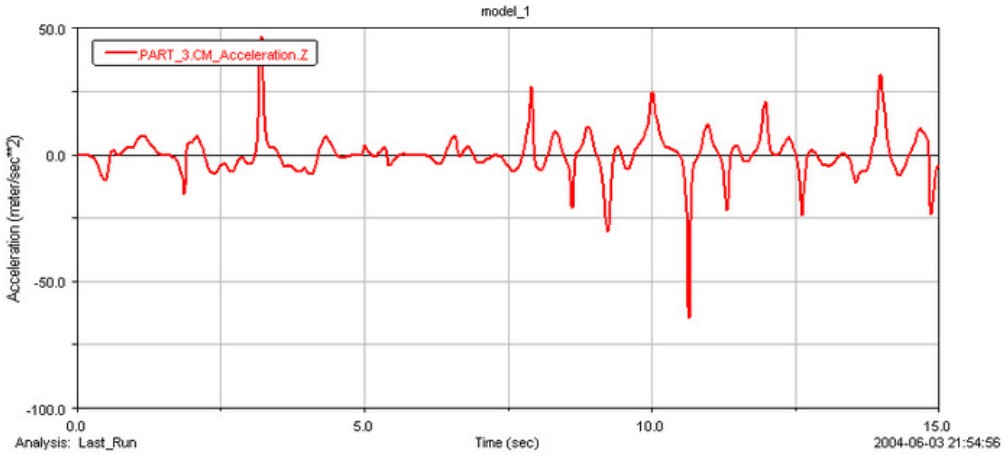
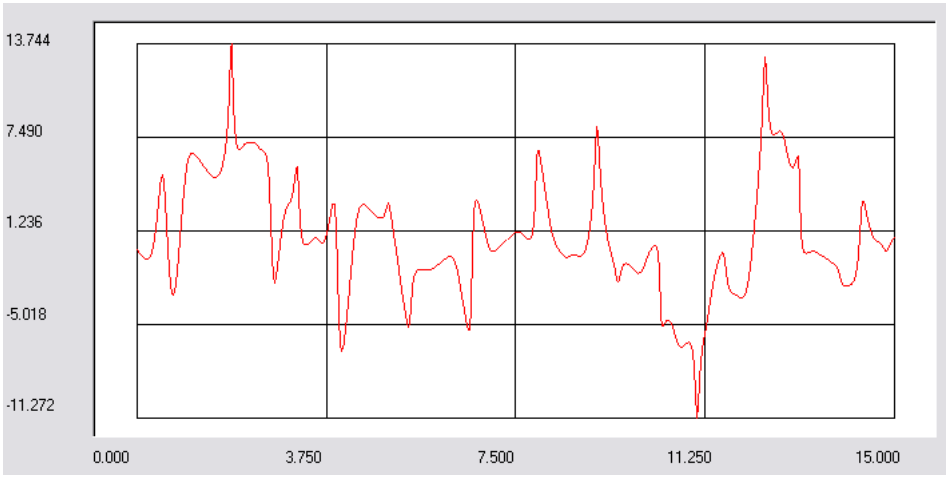



FIGURA D.9 – Gráficos para $\ddot{z}$

**(a)**




**(b)**

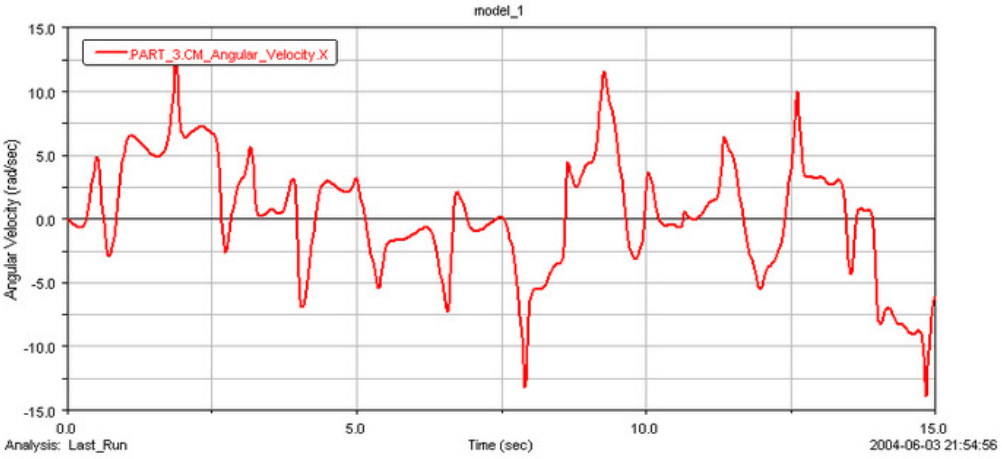
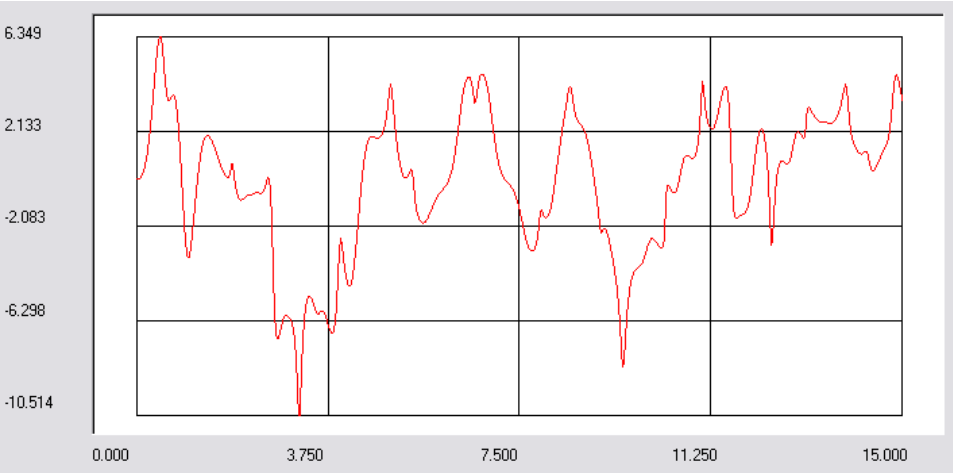



FIGURA D.10 – Gráficos para o componente x da velocidade angular

**(a)**




**(b)**

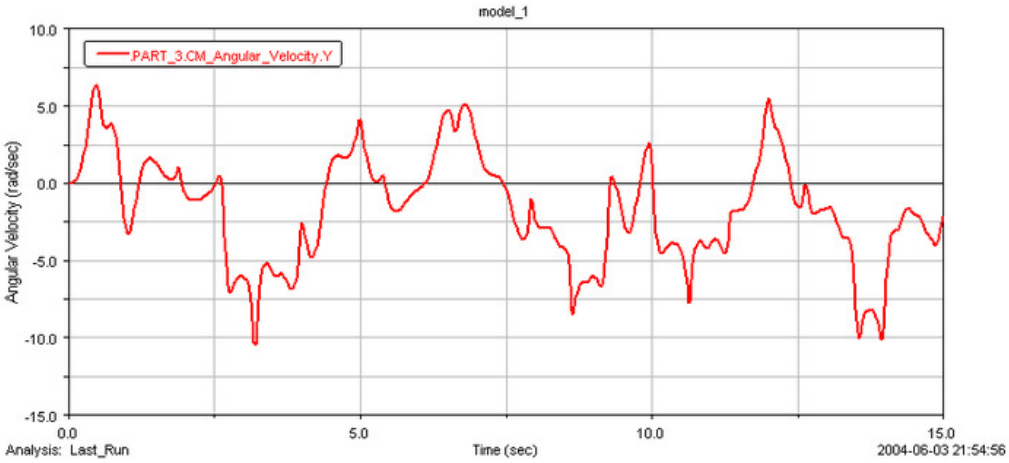
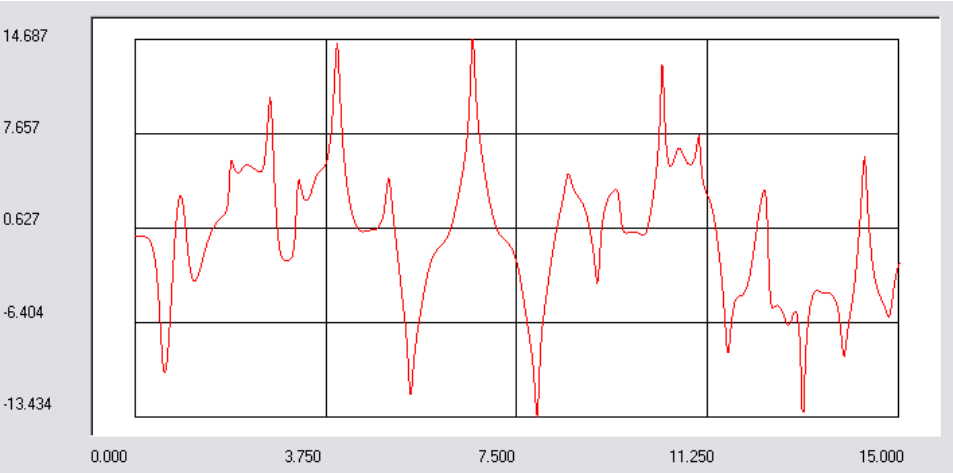



FIGURA D.11 – Gráficos para o componente y da velocidade angular

**(a)**




**(b)**

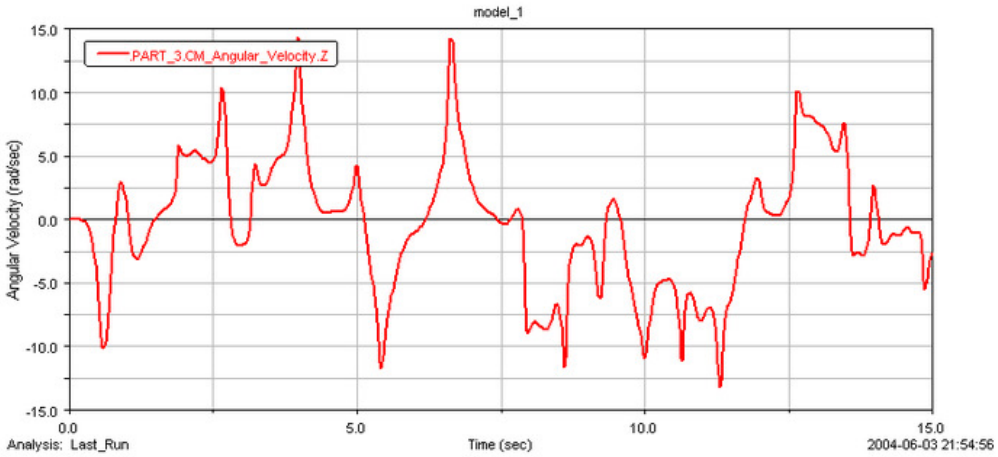
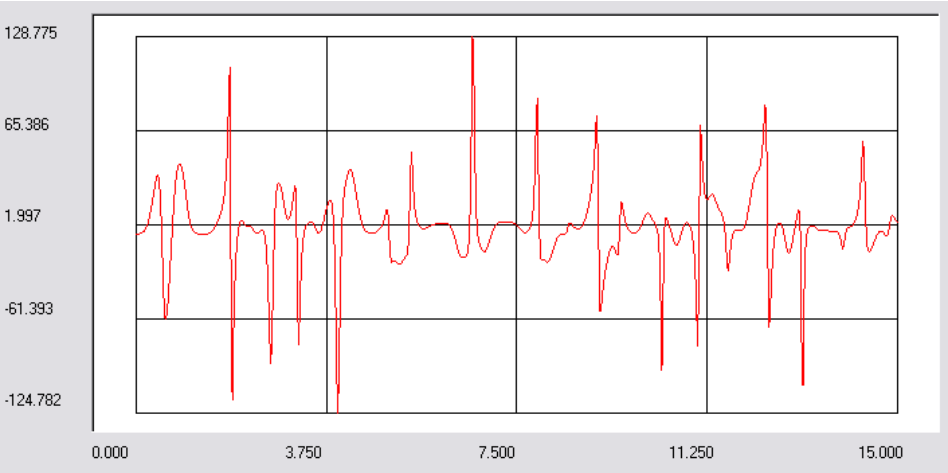



FIGURA D.12 – Gráficos para o componente z da velocidade angular

**(a)**




**(b)**

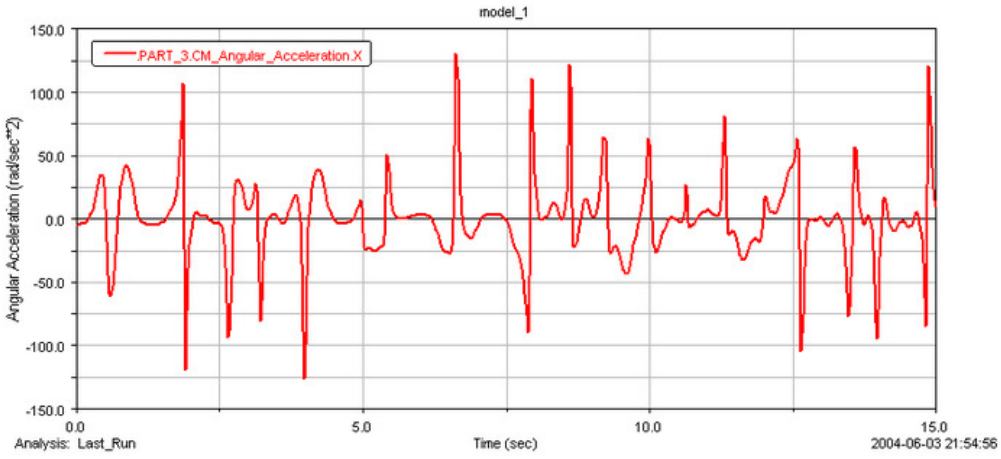
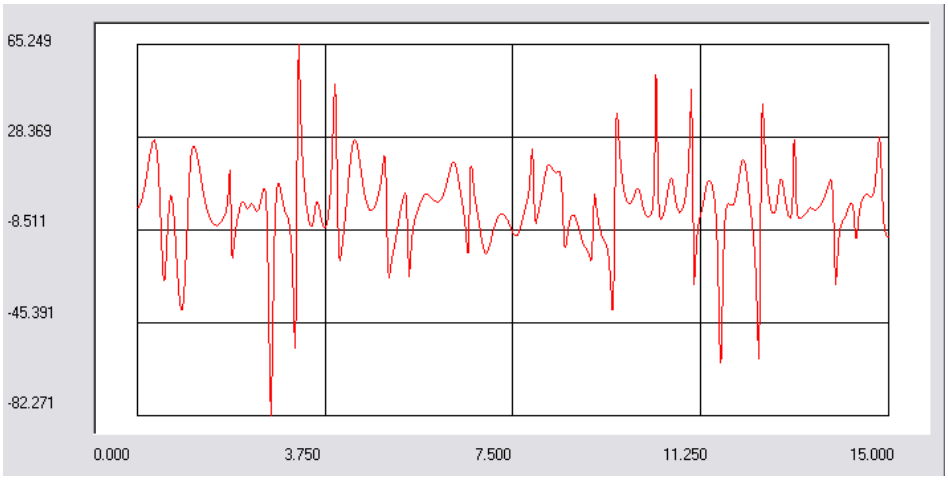



FIGURA D.13 – Gráficos para o componente x da aceleração angular

**(a)**




**(b)**

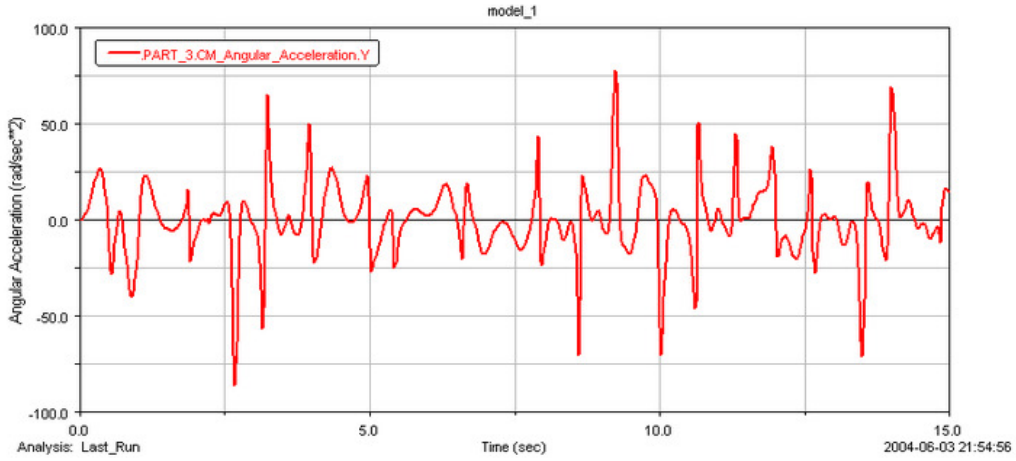
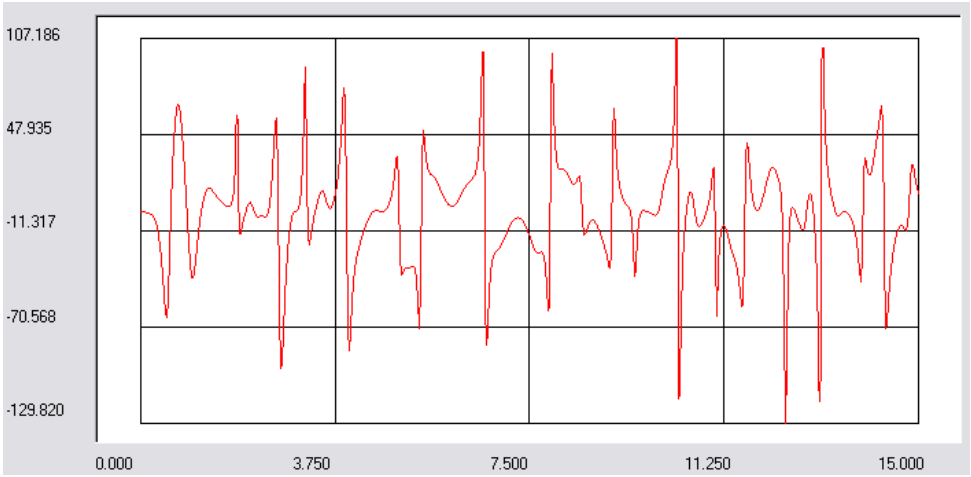



FIGURA D.14 – Gráficos para o componente y da aceleração angular

**(a)**




**(b)**

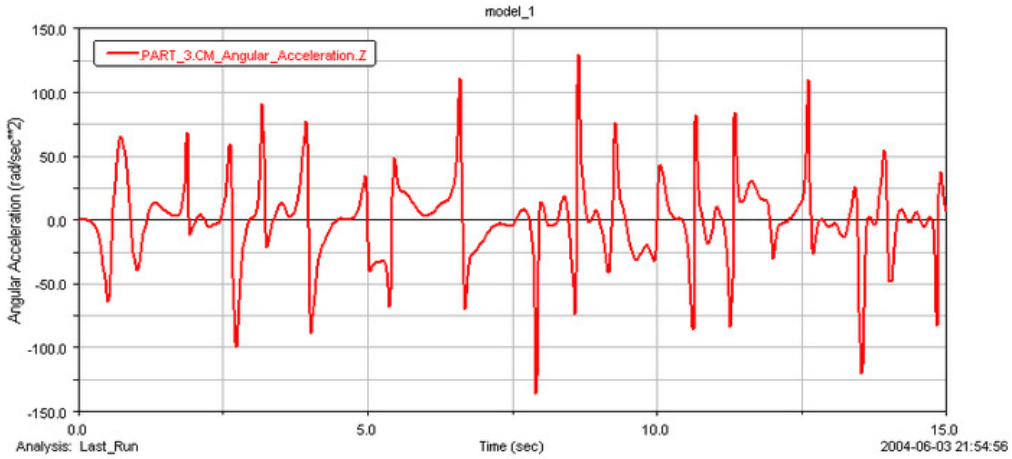
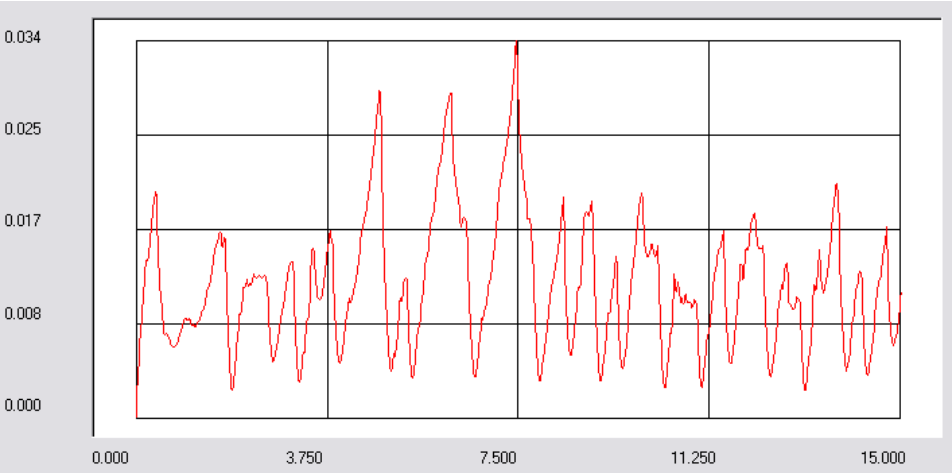



FIGURA D.15 – Gráficos para o componente z da aceleração angular

**(a)**




**(b)**

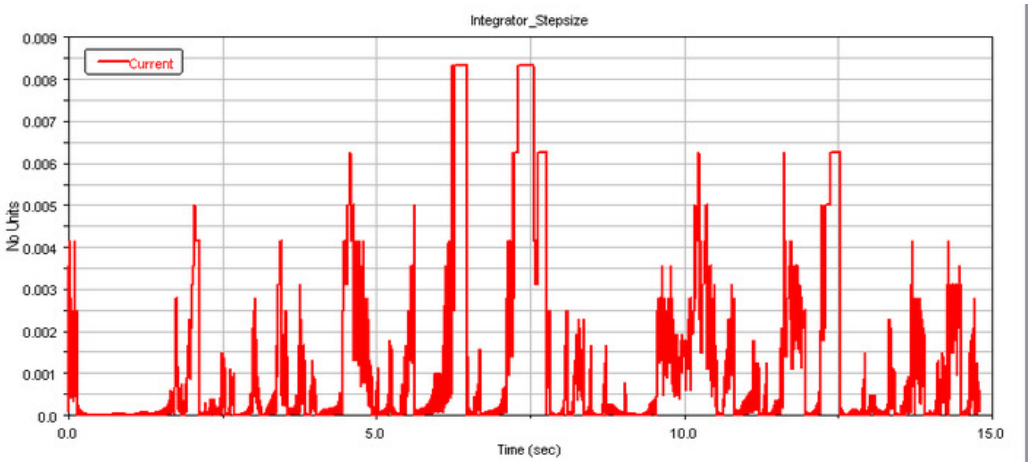
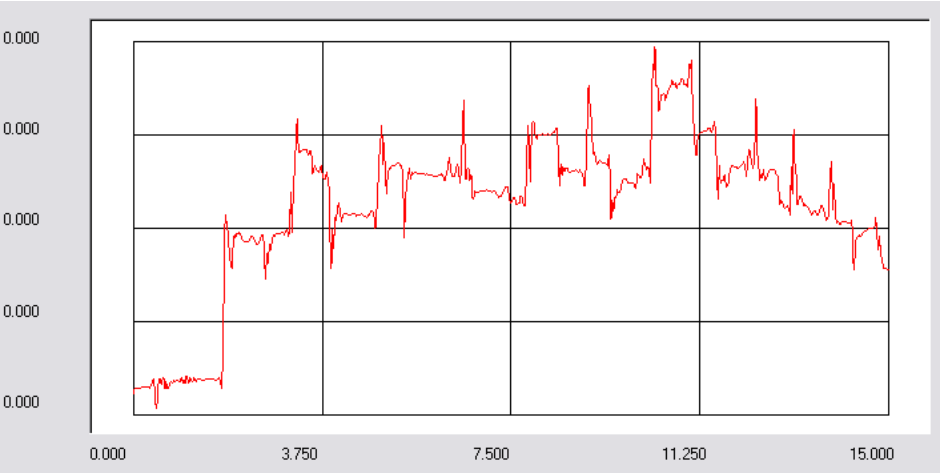



FIGURA D.16 – Gráficos para o passo do integrador

**(a)**




**(b)**

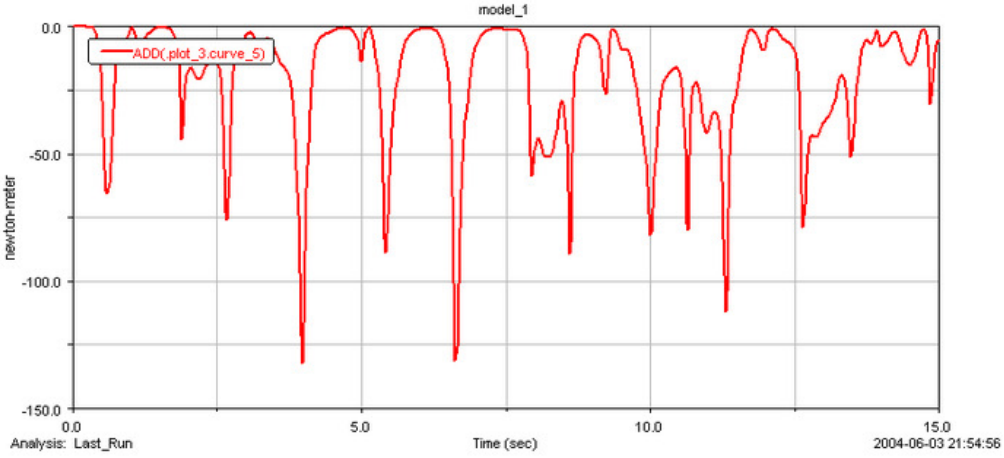



FIGURA D.17 – Gráficos para a energia mecânica total

# FOLHA DE REGISTRO DO DOCUMENTO

| 1. CLASSIFICAÇÃO/TIPO | 2. DATA | 3. DOCUMENTO Nº | 4. Nº DE PÁGINAS |
|---|---|---|---|
| TM | 28 de setembro de 2006 | CTA/ITA - IEM/TM-025/2006 | 210 |

5. TÍTULO E SUBTÍTULO:

Análise Computacional de Sistemas Multicorpos

6. AUTOR(ES):

**Glauco Oaklonde de Campos Pacheco**

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica. Divisão de Engenharia Mecânica-Aeronáutica – ITA/IEM

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Dinâmica; Computacional; Multicorpos

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Simulação computadorizada; Modelos matemáticos; Análise estrutural dinâmica; Robótica; Controle; Engenharia mecânica.

10. APRESENTAÇÃO: (**X**) **Nacional** ( ) **Internacional**

ITA, São José dos Campos, 2006, 210 páginas

11. RESUMO:

Neste texto será apresentada a teoria básica da dinâmica computacional, juntamente com alguns métodos de solução para as equações diferenciais-algébricas de movimento. Neste contexto, uma melhoria de um método de ortogonalização existente para solução numérica das equações de movimento será apresentada, caracterizando a contribuição científica deste texto. Do ponto de vista aplicado, foi desenvolvido um programa computacional visando a simulação computacional de sistemas multicorpos. O método implementado neste programa permite a simulação de sistemas rígidos holonômicos arbitrários, conectados por juntas de vários tipos, onde esses sistemas rígidos podem conter cadeias fechadas. As únicas exceções referem-se a situações de contato e impacto, não incorporadas no programa desenvolvido. A simulação de sistemas flexíveis não foi implementada. Contudo, como o programa permite a criação de conjuntos mola-amortecedor lineares e angulares, a representação de barras flexíveis por meio de parâmetros concentrados é simples de ser introduzida. O programa desenvolvido permite ainda ao usuário a inserção de funções MATLAB, abrindo caminho para a implementação de controladores para os sistemas multicorpos. O programa elaborado foi validado utilizando-se alguns exemplos teste, onde cada exemplo concentra-se num aspecto distinto, como complexidade computacional do método de solução, estudo de cadeias fechadas, bem como a análise de sistemas tridimensionais, onde os resultados obtidos com o programa desenvolvido foram comparados, em termos de acurácia e eficiência, com aqueles fornecidos pelo programa comercial ADAMS. Foi realizada ainda uma aplicação do programa a um robô com três graus de liberdade, controlado pela técnica de torque computado, visando ilustrar a possibilidade de integração do programa desenvolvido com o Matlab. Por fim, cabe ressaltar que o intuito do programa implementado é que este sirva como mais uma ferramenta disponível para a análise computacional na área de robótica do Instituto. Contudo, devido à generalidade do programa, capaz de simular sistemas multicorpos além daqueles comumente encontrados no domínio da robótica, é esperado que este sirva para as inúmeras áreas onde a análise computacional de sistemas multicorpos se faz necessária.

12. GRAU DE SIGILO:

(**X**) **OSTENSIVO** ( ) **RESERVADO** ( ) **CONFIDENCIAL** ( ) **SECRETO**