PÓLO UNIVERSITÁRIO DE VOLTA REDONDA
ESCOLA DE ENGENHARIA INDUSTRIAL METALÚRGICA DE VOLTA REDONDA

TRABALHO DE CONCLUSÃO DE CURSO

Estudo das Vibrações Causadas por uma Máquina de Estampagem Implementadas em Ambiente MATLAB

ALUNO: Aníbal Gabriel Martins Vilela
ORIENTADOR: Prof. Diomar Cesar Lobão

Volta Redonda
2016

UNIVERSIDADE FEDERAL FLUMINENSE
PÓLO UNIVERSITÁRIO DE VOLTA REDONDA
ESCOLA DE ENGENHARIA INDUSTRIAL METALÚRGICA DE VOLTA REDONDA
CURSO DE GRADUAÇÃO EM ENGENHARIA MECÂNICA

Aníbal Gabriel Martins Vilela

Estudo das Vibrações Causadas por uma Máquina de Estampagem Implementadas em Ambiente MATLAB

Projeto apresentado ao Curso de Engenharia Metalúrgica da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Engenheiro Mecânico.

ORIENTADOR: Prof. Diomar Cesar Lobão

Volta Redonda
2016

Aníbal Gabriel Martins Vilela

Estudo das Vibrações Causadas por uma Máquina de Estampagem Implementadas em Ambiente MATLAB

Projeto apresentado ao Curso de Engenharia Metalúrgica da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Engenheiro Mecânico.

Defendido em 30 de março de 2016.

Prof. Diomar Cesar Lobão (Orientador)

UFF — Universidade Federal Fluminense

Prof. João Ercio Miranda Júnior

UFF — Universidade Federal Fluminense

Prof. José Flávio Silveira Feiteira

UFF — Universidade Federal Fluminense

Volta Redonda

2016

Agradecimentos

Às aulas inspiradoras do Prof. Paiva, que influenciaram a escolha do tema desse trabalho.

Ao meu orientador, Prof. Lobão, pela paciência de sanar minhas dúvidas e por indicar o caminho a seguir.

À minha família, pela compreensão e paciência.

Resumo

Esse trabalho apresenta o projeto de uma fundação para uma máquina de estampagem de portas de carro, implementando os algoritmos em MATLAB e Octave.

Palavras-chave: Fundação de Máquinas, Vibrações Mecânicas, MATLAB, Octave

Abstract

This work presents the project of a foundation for a stamping machine specilized in car doors, implementing the algorithms in MATLAB and Octave.

Key-words: MATLAB, Octave, Machine Foundations, Mechanical Vibrations

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Abreviações

EDO	Equação Diferencial Ordinária
GdL	Grau de Liberdade
REPL	*Read, Eval, Print, Loop*
RMS	*Root Mean Square*, fator largura de banda
SI	Sistema Internacional

Conjuntos Numéricos

$\mathbb{C}$	Conjunto dos números complexos
$\mathbb{R}$	Conjunto dos números reais

Constantes

e	Número de Euler ($e \approx 2.718$)
i	Constante imaginária ($i = \sqrt{-1}$)
$\underline{g}$	Vetor aceleração da gravidade (em unidades do SI, $g \approx 9.81\,\mathrm{m/s^2}$)

Funções

atan	Arco-tangente do ângulo
cos	Co-seno do ângulo
cosh	Co-seno hiperbólico do ângulo
det	Determinante da matriz
err	Erro máximo da variável
sen	Seno do ângulo

senh Seno hiperbólico do ângulo

square Onda quadrada construída a partir do sinal de uma onda senóide

Letras Gregas

δt Pequeno intervalo de tempo; nos métodos de Runge-Kutta, representa o passo

Δ Representa a variação de altura da energia potencial gravitacional

Ω Frequência do forçamento

ω_n Frequência natural do sistema

$\mathbf{\Lambda}$ Matriz diagonal que contém o quadrado das frequências naturais

$\underline{\alpha}$ Vetor aceleração angular

$\underline{\omega}$ Vetor velocidade angular

$\underline{\theta}$ Vetor deslocamento angular

ξ Constante adimensional de amortecimento

Letras Romanas

K Energia cinética

D Energia dissipada

T Transmissibilidade

U Energia potencial

$\mathbf{C}$ Matriz de amortecimento

$\mathbf{I}$ Matriz identidade

$\mathbf{K}$ Matriz de rigidez

$\mathbf{M}$ Matriz de inércia

$\mathbf{V}$ Matriz modal normalizada

$\underline{a}$ Vetor aceleração linear

$\underline{Q}$ Vetor de forçamentos generalizados

$\underline{q}$ Vetor de coordenadas generalizadas

$\underline{v}$ Vetor velocidade linear

$\underline{x}$ Vetor deslocamento linear

c Constante de amortecimento

D Diâmetro do desenvolvimento

d Diâmetro da chapa

f Frequência de um evento

F_a Módulo da força gerada por um amortecedor

F_c Força de corte

F_e Módulo do forçamento externo aplicado ao corpo

F_m Módulo da força gerada por uma mola

F_P Módulo da força peso

k Constante da mola

K_i Nos métodos de Runge-Kutta, representam os coeficientes angulares das retas

K_s Tensão de ruptura ao cisalhamento

L Dimensões lineares

m Massa

p Perímetro do corte

t Instante de tempo

Subscritos

a Componentes de deslocamento angular e variação linear

c Relacionado ao processo de corte

e Relacionado à estrutura

emb Relacionado ao processo de embutimento

end Valor final da variável

h Parcela correspondente à solução homogênea do problema

i	Qualquer um dos possíveis valores da variável em um intervalo conhecido
ini	Valor inicial da variável
l	Componentes de variação e deslocamento lineares
max	Valor máximo da variável em um intervalo conhecido
p	Parcela correspondente à solução particular do problema
res	Referente à ressonância do sistema
$rk2$	Variável calculada segundo o método de Runge-Kutta de 2^a ordem
$rk3$	Variável calculada segundo o método de Runge-Kutta de 3^a ordem
s	Referente a uma propriedade equivalente do solo
0	Condição inicial arbitrária fornecida para a variável

Superscritos

n	Elemento na iteração atual
$n+1$	Elemento na próxima iteração
T	Transposto do elemento indicado

Parte I

Introdução

Capítulo 1

Noções Básicas do Estudo Proposto

1.1 Motivação

Máquinas mecânicas, equipamentos elétricos[1] ou até mesmo estruturas mecânicas estão sujeitas a problemas relacionados a vibrações, seja por pequenas massas que resultam em forçamentos periódicos, ou efeitos de vento ou outros fluidos interagindo com geometrias sólidas, gerando forças aleatórias. Ônibus e carros, por exemplo, podem apresentar vibrações por causa do motor, causando desde leves desconfortos aos seus passageiros até o falhas em suas estruturas, que podem levar a acidentes graves.

Na indústria em geral, o estudo das vibrações em qualquer tipo de equipamento é feito através da análise dos resultados obtidos por um acelerômetro colocado junto ao equipamento. Contudo, é possível uma análise prévia com conclusões significativas ainda na fase de projeto, com modelagens matemáticas relativamente simples, especialmente com o auxílio de computadores e métodos numéricos já bem estabelecidos no mundo acadêmico[2].

O presente trabalho visa apresentar, de forma teórica, ferramentas matemáticas e numéricas suficientes para modelar sistemas vibratórios reais, baseando-se no estudo de uma máquina de estampagem industrial. Ainda que tenha um enfoque maior nas ferramentas numéricas apresentadas pelo MATLAB, são citados ao longo do texto alternativas a elas, em diversas linguagens de programação, que poderiam ser usadas de forma semelhante.

Além de apresentar uma solução utilizando ferramentas prontas, já presentes na distribuição padrão do MATLAB, é apresentada uma implementação do método de Runge-Kutta de 4^a ordem para um problema específico, utilizando, apenas, a função residente de impressão de gráficos, `plot`.

Ainda, no Apêndice B, são apresentadas três alternativas a essa implementação, utilizando as linguagens Go, Haskell e OCaml — as duas últimas escolhidas por focarem no

[1] Embora esses estejam fora do escopo desse estudo.

[2] Por exemplo, os famosos métodos de Runge-Kutta, foram desenvolvidos em torno de 1900, pelos matemáticos alemães de mesmo sobrenome.

paradigma de programação funcional.

Como ferramenta didática, o Apêndice C apresenta um simulador para o sistema massa-mola-amortecedor sem forçamento, que demonstra as diversas formas de comportamento desse sistema. Seu código é disponibilizado gratuitamente *online*

1.2 Origens do MATLAB

No final da década de 1970, Cleve Moler, então professor da Universidade do Novo México, lecionando análise numérica e teoria de matrizes, queria dar ao seus alunos acesso às bibliotecas EISPACK[1] e LINPACK[2] sem que esses tivessem de escrever programas em FORTRAN.

Figura 1.2.1: Logo do MATLAB, apresentando a característica membrana em L.
Fonte: Site oficial da MathWorks.

Assim, seguindo a metodologia do livro *Algorithms + Data Structures = Programs* (Niklaus Wirth, Prentice Hall, 1976), Cleve, utilizando FORTRAN e partes do LINPACK e EISPACK, desenvolveu a primeira versão do MATLAB.

Figura 1.2.2: Gráficos na primeira versão do MATLAB.
Fonte: Site oficial da MathWorks.

Até então, o único tipo de dado era a matriz, e o comando `help` listava as quase 80 funções disponíveis. Não havia suporte a *M-files* ou *toolboxes*; se alguém quisesse adicionar uma função ao programa, tinha de modificar o código-fonte em FORTRAN e recompilar tudo. Os primeiros gráficos consistiam em imprimir asteriscos na tela, conforme demostra a figura 1.2.2, retirada da página *The Origins of MATLAB*[3]

Em sua primeira versão, o FORTRAN MATLAB era portável e poderia ser compilado para rodar em vários computadores disponíveis no final da década de 1970 e começo da década de 1980.

[1] `http://www.netlib.org/eispack`, um conjunto de subrotinas escrito para o FORTRAN para o cálculo de autovalores e autovetores para nove conjuntos de matrizes.

[2] `http://www.netlib.org/linpack`, um conjunto de subrotinas escrito para o FORTRAN para análise e solução de equações lineares e equações lineares de mínimos quadrados.

[3] `http://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html`

O primeiro "computador pesoal" usado por Cleve foi um Tektronix 4081, adquirido em 1978, o primeiro microcomputador com um processador de 32 bits. A máquina ocupava uma mesa inteira e possuía apenas 64 kilobytes de memória. A Tektronix, contudo, tinha um compilador de FORTRAN, e, portanto, era capaz de rodar o MATLAB — embora o processo de compilação fosse um pouco mais complicado, dependendo de *memory overlay*[①].

Cleve visitou Stanford em 1979 e lecionou CS237, o curso de análise numérica para graduandos, fazendo com que os alunos utilizassem o MATLAB para algumas tarefas. A turma em si consistia de alunos da Matemática, Ciência da Computação e Engenharia; os dois primeiros grupos não se impressionaram com o novo programa, por ser baseado em FORTRAN, além de não ser um sistema particularmente robusto e não refletir as pesquisas da época na área de análise numérica. Contudo, para os estudantes de Engenharia, que estudavam sistemas de controle e processamento de sinais, a ênfase que o MATLAB tinha em matrizes se tornava muito útil.

Figura 1.2.3: Computador Tektronix 4081 de 1978.
Fonte: Blog de Cleve Moler.

Jack Little, um Engenheiro de Controle, foi o principal desenvolvedor da primeira versão comercial do FORTRAN MATLAB, e, quando a IBM anunciou seu primeiro computador pessoal em 1981, ele e seu colega Steve Bangert reescreveram o MATLAB em C, adicionando *M-files*, *toolboxes* e gráficos melhorados.

E, em 1984, Cleve Moler, Jack Little e Steve Bangert fundaram na Califórnia a MathWorks.

[①]Uma técnica que permite a reutilização dos mesmos espaços da memória em diferentes estágios do programa; por exemplo, quando uma função não é mais necessária, esse espaço passa a ser utilizado parcial ou totalmente por outra função.

1.3 Situação do Trabalho

Esse estudo divide-se em três partes principais:

I. Situa o trabalho no meio acadêmico, apresentando as motivações e a notação utilizada ao longo do estudo.

II. Discorre, com exemplos, sobre as diversas ferramentas teóricas necessárias para o estudo do problema proposto.

III. Apresenta o problema proposto e a solução encontrada para o mesmo.

Após isso, os apêndices oferecem informações complementares e adicionais ao texto principal:

A. Apresenta as implementações de arquivos auxiliares utilizados ao longo do texto. Alguns deles são, de fato, necessários para que os *scripts* descritos ao longo do texto funcionem normalmente, porém desviariam a atenção dos arquivos principais; assim, sempre que necessário, são indicadas referências para a seção correspondente presente nesse apêndice.

B. Ilustra diferentes implementações do método de Runge-Kutta de quarta ordem em algumas linguagens não discutidas ao longo do texto. De modo a apresentar diferentes formas de implementação do método, foram escolhidas três linguagens não convencionais na área de engenharia: Go, Haskell e OCaml.

C. Apresenta a animação criada para ilustrar o comportamento do sistema massa-mola-amortecedor sem forçamento, discutido ao longo do texto; como o código-fonte é disponibilizado *on-line*, e esse tipo de implementação foge do propósito desse estudo, ele não é discutido em detalhe.

1.4 Notação

A notação utilizada foi escolhida de modo a facilitar a formatação e visualização do documento, no formato impresso, virtual, e, ainda, na apresentação de slides.

- vetores: $\underline{x}$

  Contrariando a notação vetorial tradicional, a indicação de vetor é representada sublinhando-se a variável que o representa. Isso deve-se à melhor apresentação tipográfica de entidades como a derivada temporal ($\underline{\dot{x}}$), que ficariam confusas segundo a notação tradicional.

- elemento de um vetor: $\underline{x}[n]$, onde $n = 1, 2, 3, ...$

  Aproveita-se da notação vetorial, porém indicando qual elemento será apresentado entre colchetes, remetendo à sintaxe equivalente em algumas linguagens de programação; ao contrário dessas, contudo, a indexação está baseada na unidade.

- tensores e matrizes: $\mathbf{I}$

  Segue a notação matricial tradicional, tendo em vista que destaca o elemento dos demais na equação

- elemento de um tensor ou de uma matriz: $\mathbf{I}[m, n]$, onde $m, n = 1, 2, 3, ...$

  Aproveita-se da notação tensorial, porém, assim como na notação vetorial, indica entre colchetes a linha e a coluna do elemento a ser referenciado, seguindo, também, a sintaxe de algumas linguagens de programação.

- energias: K (cinética), U (potencial) e D (dissipada)

  Segue a notação tradicional estadunidense[①], por ser mais concisa e distinguível no texto.

Os comandos do MATLAB citados ao longo do trabalho geralmente são referenciados pela sua variante mais conhecida, caso outras existam — como é o caso dos comandos `ode23`, `ode23s`, `ode45` e similares. Nesse caso, entende-se que as variantes tenham funcionamento similar à citada (ainda que difiram em seus resultados) e, portanto, não precisam ser demonstradas. Como cada uma dessas apresenta implementações e otimizações para a solução de problemas diferentes dos apresentados, recomenda-se a leitura de sua documentação.

[①]Segundo o dicionário Houaiss, "dos Estados Unidos da América; norte-americano". O termo comumente utilizado como sinônimo, "americano", refere-se a algo relativo ao continente Americano e não ao país.

Capítulo 2

Noções Básicas de MATLAB

Ao contrário de linguagens de programação mais tradicionais — como C, FORTRAN ou Java — o MATLAB apresenta uma linguagem interpretada, um ambiente de desenvolvimento próprio e uma extensa documentação *on* e *offline*. Por ser uma linguagem interpretada, o desenvolvimento de programas pode ser feito de modo interativo através do REPL[①] ou com o auxílio das *M-files*[②]. Isso o torna altamente produtivo, tendo em vista que o REPL pode ser usado como ambiente de testes e depuração, enquanto as *M-files* funcionam como o programa em si.

Porém, por ser uma linguagem interpretada, a velocidade de execução dos programas é mais lenta do que se esses fossem escritos em linguagens compiladas. Essa desvantagem, contudo, é aceitável frente ao ganho de velocidade de desenvolvimento, uma vez que dois passos do ciclo *editar, compilar, "linkar" e testar*[③] são substituídos pelo interpretador, que traduz o programa linha a linha, ao invés de compilar e "linkar" as bibliotecas utilizadas. Esse processo também leva a uma captura de erros mais precisa, indicando ao programador em qual linha foi detectado o erro.

Além disso, o MATLAB conta com uma série de otimizações que fazem os programas executarem de forma mais eficiente e veloz, e com mecanismos de paralelização de fácil implementação, especialmente úteis em máquinas com vários núcleos de processamento.

2.1 MATLAB, Scilab e GNU Octave

Os *softwares* Scilab[④] e GNU Octave[⑤] são alternativas *open-source* ao MATLAB, e procuram possuir uma boa compatibilidade com esse.

A sintaxe das três linguagens são parecidas o suficiente para que se possa ter uma boa compatibilidade entre si; assim, a não ser que se diga o contrário, refere-se nesse

[①] Em inglês: "*Read, Eval, Print, Loop*" ("Ler, Avaliar, Imprimir, Repetir", em tradução livre).
[②] Arquivos contendo instruções sequenciais para gerar determinado resultado.
[③] Do termo original em inglês: "*edit, compile, link and test*".
[④] `http://www.scilab.org/`
[⑤] `http://www.gnu.org/software/octave/`

apenas ao MATLAB. Mais informações sobre as alternativas podem ser encontradas em seus respectivos *sites*, bem como um comparativo de compatibilidade.

Nota-se que certas funcionalidades não estão presentes nos três, e problemas de compatibilidade não são incomuns. Porém, os referidos problemas costumam encontrar-se bem documentados ou facilmente encontrados nos sites oficiais, em fóruns *online*, ou em artigos entitulados "Diferenças entre Octave, Scilab e MATLAB" em sites na internet.

Também é notável, embora de menor importância, que os três sistemas apresentam *prompts* diferenciados: `>>>`, no caso do MATLAB; `-->` no Scilab, e `octave:1>` no Octave. A única diferença entre eles, afora a aparência, ocorre no Octave, onde é mostrado o número do comando na seção atual — e, por apresentar maior clareza, esse será o prompt utilizado nesse documento, quando se fizer necessário apresentar uma seção interativa de algum dos ambientes.

2.2 Ajuda e Documentação

Além da documentação online[①], o MATLAB oferece o comando `help` que disponibiliza páginas de ajuda para seus comandos. Sua utilização é bem simples, bastando digitar qual o comando deseja-se analizar, como, por exemplo, o operador `^`, na primeira linha do Algoritmo abaixo, ou o comando `square` — comando esse que, no ambiente Octave, não é uma função nativa, requerendo a instalação do pacote `signal` (a Seção 2.3 abaixo discorre em mais detalhes sobre pacotes).

Algoritmo 2.2.1: Comandos de ajuda

```
1  octave:1> help ^
2
3   -- Operator: ^
4       Power operator.  This may return complex results for real inputs.
5       Use 'realsqrt', 'cbrt', 'nthroot', or 'realroot' to obtain real
6       results when possible.
7
8       See also: power, **, .^, .**, realpow, realsqrt, cbrt, nthroot.
9
10
11 Additional help for built-in functions and operators is
12 available in the online version of the manual.  Use the command
13 'doc <topic>' to search the manual index.
14
15 Help and information about Octave is also available on the WWW
16 at http://www.octave.org and via the help@octave.org
17 mailing list.
18 octave:2> pkg load signal
```

[①] http://www.mathworks.com/help/matlab/

```
octave:3> help square
'square' is a function from the file /home/ryu/octave/signal-1.3.2/square.m

 -- Function File: S = square (T, DUTY)
 -- Function File: S = square (T)
     Generate a square wave of period 2 pi with limits +1/-1.

     If DUTY is specified, the square wave is +1 for that portion of the
     time.

                           on time
        duty cycle = ------------------
                     on time + off time

     See also: cos, sawtooth, sin, tripuls.


Additional help for built-in functions and operators is
available in the online version of the manual.  Use the command
'doc <topic>' to search the manual index.

Help and information about Octave is also available on the WWW
at http://www.octave.org and via the help@octave.org
mailing list.
octave:4>
```

Essa documentação está disponível *offline* e fornece facilmente a utilização de comandos e suas opções, frequentemente contendo exemplos práticos.

2.3 Instalação de Pacotes no GNU Octave

Alguns dos algoritmos apresentados são definidos em termos de funções residentes no MATLAB que não estão presentes na instalação padrão do GNU Octave; contudo, esse disponibiliza um grande número de funcionalidades extras através do Octave-Forge[①].

Para os *scripts* apresentados, assume-se a instalação de três desses pacotes:

- `odepkg`: pacote para solução de equações diferenciais (dentre outras, contém a função `ode45`);
- `control`: dependência do pacote `signal`;
- `signal`: ferramentas para processamento de sinais (dentre outras, contém o comando `square`).

[①] `http://octave.sourceforge.net/`

Esses podem ser instalados, respectivamente, com os seguintes comandos dentro do ambiente Octave:

```
pkg install -forge odepkg
pkg install -forge control
pkg install -forge signal
```

A página *Packages*[1] no site do Octave-Forge fornece uma lista completa dos pacotes disponíveis.

O comando `pkg list` mostra informações sobre quais pacotes estão atualmente instalados no sistema, bem como suas versões. Para que suas funcionalidades estejam disponíveis no ambiente de trabalho, é necessário importá-los; o comando para isso é

```
pkg load package-name[s]
```

Após uma importação, o comando `pkg list` indicará quais pacotes estão disponíveis no ambiente atual, como demonstra o Algoritmo 2.3.2.

Algoritmo 2.3.2: Importação de pacotes no GNU Octave

```
1  octave:1> pkg list
2  Package Name  | Version | Installation directory
3  --------------+---------+-----------------------
4       control  |   2.8.1 | /home/ryu/octave/control-2.8.1
5        odepkg  |   0.8.5 | /home/ryu/octave/odepkg-0.8.5
6        signal  |   1.3.2 | /home/ryu/octave/signal-1.3.2
7  octave:2> pkg load odepkg
8  octave:3> pkg list
9  Package Name  | Version | Installation directory
10 --------------+---------+-----------------------
11      control  |   2.8.1 | /home/ryu/octave/control-2.8.1
12       odepkg *|   0.8.5 | /home/ryu/octave/odepkg-0.8.5
13       signal  |   1.3.2 | /home/ryu/octave/signal-1.3.2
14 octave:4>
```

Caso se deseje retirar as funcionalidades de um pacote no ambiente, basta utilizar o comando `pkg unload package-name`.

Por exemplo, o comando `ode45` está definido no ambiente do Algoritmo 2.3.2; porém, retirando o pacote (com o comando da linha 28), o Octave não reconhece esse símbolo, como demonstra o resultado do comando da linha 29.[2]

Algoritmo 2.3.3: Retirada de pacotes no GNU Octave

```
14 octave:4> ode45
15 'ode45' is a function from the file /home/ryu/octave/odepkg-0.8.5/ode45.m
```

[1] `http://octave.sourceforge.net/packages.php`

[2] Grande parte do retorno foi suprimido na linha 27 por questões de legibilidade.

```
16
17  -- Function File: [] = ode45 (@FUN, SLOT, INIT, [OPT], [PAR1, PAR2,
18           ...])
19  -- Command: [SOL] = ode45 (@FUN, SLOT, INIT, [OPT], [PAR1, PAR2, ...])
20  -- Command: [T, Y, [XE, YE, IE]] = ode45 (@FUN, SLOT, INIT, [OPT],
21           [PAR1, PAR2, ...])
22
23      This function file can be used to solve a set of non-stiff ordinary
24      differential equations (non-stiff ODEs) or non-stiff differential
25      algebraic equations (non-stiff DAEs) with the well known explicit
26      Runge-Kutta method of order (4,5).
27 [...]
28 octave:5> pkg unload odepkg
29 octave:6> ode45
30 warning: Octave provides lsode for solving differential equations.  For more
31 information try 'help lsode'.  Matlab-compatible ODE functions are
32 provided by the odepkg package.  See
33 <http://octave.sourceforge.net/odepkg/>.
34
35 Please read <http://www.octave.org/missing.html> to learn how you can
36 contribute missing functionality.
37
38
39 error: 'ode45' undefined near line 1 column 1
40 octave:7>
```

Existe, ainda, o comando `pkg update`, que atualiza todos os pacotes instalados.

2.4 Variáveis

Diferentemente de linguagens de programação compiladas, o MATLAB não requer declarações explícitas das variáveis a serem usadas, e irá criá-las conforme a necessidade. Também por isso, essas não requerem identificação de tipo, sendo esse inferido automaticamente pelo contexto. Assim, a definição de uma nova variável no MATLAB corresponde à associação de variáveis em linguagens como C e Java, conforme demonstrado na linha 2 do Algoritmo 2.4.4.

Uma lista contendo todas as variáveis definidas no ambiente atual pode ser gerada com o comando `whos`. Esse comando também mostra informações adicionais dessas, como tamanho da matriz, tipo e tamanho em bytes.

Algoritmo 2.4.4: Definição de variáveis

```
1 octave:1> whos
2 octave:2> x = 10;
3 octave:3> whos
```

```
4  Variables in the current scope:
5
6     Attr Name        Size                     Bytes  Class
7     ==== ====        ====                     =====  =====
8          x           1x1                          8  double
9
10 Total is 1 element using 8 bytes
11
12 octave:4>
```

Na interface gráfica do Octave ou no MATLAB, essas informações costumam ser apresentadas na sub-janela *Workspace*[①], presente por padrão ao iniciar o ambiente. Esse comando é especialmente útil caso a *Workspace* esteja fechada ou o programa esteja sendo utilizado através de um terminal *shell*.

2.5 Cálculos Básicos

2.5.1 Aritmética

Cálculos básicos seguem a notação tradicional da matemática. No exemplo 2.5.5, as linhas 1 e 3 demonstram a definição de duas variáveis, `x` e `y`.

Por padrão, o MATLAB imprime na tela o resultado dos comandos, o que pode ser problemático em certas cituações — como vetores de grandes dimensões. Para suprimir esse retorno, basta encerrar um comando com um ponto-e-vírgula, como ilustrado na linha 5, que também demonstra a associação de valores a variáveis[②].

A varíavel `ans`, definida automaticamente pelo MATLAB no retorno de um comando, representa o resultado do cálculo anterior realizado, e pode ser referido nos comandos seguintes.

Nas linhas 8 a 16 são demonstrados alguns dos operadores básicos, como adição, subtração e exponenciação, bem como a aplicação de uma função que não possui um operador próprio, `sqrt` — que também demonstra a capacidade do MATLAB de lidar com números complexos; as linhas 18 e 20 demonstram a precedência dos operadores e a avaliação das expressões.

Algoritmo 2.5.5: Operações básicas

```
1 octave:1> x = 2
2 x = 2
3 octave:2> y = 3
4 y = 3
```

[①] Ou "Área de Trabalho", em tradução literal.

[②] Nota-se que essa sintaxe é a mesma para a criação de variáveis. De fato, o MATLAB não requer declaração de variáveis, e irá criá-las ou associar valores a elas conforme a necsessidade.

```
octave:3> y = 4;
octave:4> x + y
ans = 6
octave:5> x - y
ans = -2
octave:6> x*y
ans = 8
octave:7> x/y
ans = 0.5
octave:8> sqrt(x-y)
ans =  0.00000 + 1.41421i
octave:9> x**y
ans = 16
octave:10> x + y*2 - 2**y + 5*sqrt(4)
ans = 4
octave:11> (x) + ((y)*2) - (2**(y)) + (5*(sqrt(4)))
ans = 4
octave:12>
```

2.5.2 Vetores

Para o MATLAB, vetores são apenas matrizes que possuem somente uma linha (no caso de um vetor coluna) ou somente uma coluna (no caso de um vetor linha). Assim, as operações definidas para esses elementos podem ser aplicadas, também, a matrizes.

Em algumas operações com escalares, os operadores precedidos de um ponto (como, por exemplo, ".+" e ".-"), conhecidos como *element-wise operators*, correspondem aos mesmos sem o ponto precedente. Os operadores *element-wise* são utilizados para aplicar uma determinada operação a cada elemento de um vetor.

Algoritmo 2.5.6: Vetores

```
octave:1> x = [1, 2, 3]
x =

   1   2   3

octave:2> x = [1; 2; 3]
x =

   1
   2
   3

octave:3> x = tranpose(x)
x =
```

```
15
16    1   2   3
17
18  octave:4> 2.*x
19  ans =
20
21    2   4   6
22
23  octave:5> x .+ 2
24  ans =
25
26    3   4   5
27
28  octave:6> [1, 2, 3] * [2; 3; 1]
29  ans = 11
30  octave:7> [1, 2, 3] * [2, 3, 1]
31  error: operator *: nonconformant arguments (op1 is 1x3, op2 is 1x3)
32  octave:7> [1, 2, 3] ^ 2
33  error: for A^b, A must be a square matrix. Use .^ for elementwise power.
34  octave:7> [1, 2, 3] .^ 2
35  ans =
36
37    1   4   9
38
39  octave:8>
```

Mesmo nos casos citados anteriormente, é considerado boa prática a utilização da variante *element-wise* do operador aplicado, por conta de possíveis erros, como os demonstrados nas linhas 31 e 33.

É notável, também, que, como a transposição é uma operação muito comum, ela possui um operador próprio, "`'`". Assim, `transpose(x)` é equivalente a `x'`.

2.6 Funções

Para definir-se uma nova função no MATLAB, é necessário criá-la em um arquivo separado, cujo nome corresponda ao nome da função. Apesar da sintaxe contra-intuitiva, funções são muito úteis em problemas reais, quando deseja-se compor uma nova funcionalidade para o MATLAB, ou quando deseja-se reduzir um determinado problema em partes menores.

Considerando que `par1`, ..., `parN` são os parâmetros da função e que `ans1`, ..., `ansN` são os valores que se deseja retornar dela, seu cabeçalho pode ser escrito como:

```
function [ans1, ans2, ..., ansN] = file_name(par1, par2, ..., parN)
```

onde `file_name` deve, obrigatoriamente, corresponder ao nome do arquivo onde essa função está definida. Também, nesse arquivo, não pode conter nada além da definição da função.

Assim, uma função que recebe dois parâmetros e sempre retorna o primeiro pode ser escrita como:

Algoritmo 2.6.7: return_first.m

```
function val = return_first(x,y)
    val = x;
end
```

Para acessar essa função pelo MATLAB, é necessário que esse esteja no mesmo diretório do arquivo que a define.

Algoritmo 2.6.8: Funções

```
octave:1> return_first(2,3)
ans =  2
octave:2> return_first([2, 2, 2], 3)
ans =

   2   2   2

octave:3>
```

A Parte II à seguir trata dos ferramentas teóricas para o estabelecimento do estudo de vibrações.

Parte II

Ferramentas Teóricas

Capítulo 3

Mecânicas Newtoniana e Lagrangiana

Das etapas de um estudo das vibrações mecânicas de um sistema, a obtenção de suas equações de movimento é a primeira e mais importante. As equações de movimento representam, matematicamente, o modelo teórico de um sistema real; quanto mais esse modelo conseguir se aproximar da realidade, mais precisas serão as respostas obtidas no final do processo.

Existem dois métodos principais para encontrar tais equações: o Método de Newton-Euler (derivado da mecânica clássica), que leva em consideração as forças e os momentos atuantes no sistema, e o chamado Método da Energia (derivado da mecânica Lagrangiana), que considera as diferentes parcelas de energia atuantes no sistema.

3.1 Formalização Newtoniana

Conhecido como Método de Newton-Euler, deriva da mecânica clássica, e consiste em analisar as forças e momentos atuantes em cada corpo do sistema, escrevendo as equações de Newton, referente às forças atuantes, (3.1.1), e Euler, referente aos momentos, (3.1.2), para cada um [1, 2].

Como resultado do equacionamento, obtém-se as equações de movimento e as forças de reação do sistema.

Em um sistema tridimensional, escreve-se, para cada corpo do sistema:

$$\sum_{i=1}^{n} \underline{F}_i = m\ \underline{\ddot{x}} \tag{3.1.1}$$

$$\sum_{i=1}^{n} \underline{M}_i = \mathbf{I}\ \underline{\ddot{\theta}} \tag{3.1.2}$$

onde:

- F_i módulo do i-ésimo forçamento externo aplicado ao corpo;

- m massa do corpo;
- x deslocamento linear;
- M_i módulo do i-ésimo momento externo aplicado ao corpo;
- $\mathbf{I}$ tensor de inércia;
- θ deslocamento angular.

Considerando um sistema cujos componentes são elementos lineares, as forças atuantes podem ser escritas como:

- força peso:

$$\underline{F}_P = m\ \underline{g} \tag{3.1.3}$$

- força exercida por uma mola:

$$\underline{F}_m = \underbrace{k_l\ \underline{x}}_{\substack{\text{deslocamento}\\ \text{linear}}} + \underbrace{k_a\ \underline{\theta}}_{\substack{\text{deslocamento}\\ \text{angular}}} \tag{3.1.4}$$

- força exercida por um amortecedor:

$$\underline{F}_a = \underbrace{c_l\ \underline{\dot{x}}}_{\substack{\text{deslocamento}\\ \text{linear}}} + \underbrace{c_a\ \underline{\dot{\theta}}}_{\substack{\text{deslocamento}\\ \text{angular}}} \tag{3.1.5}$$

3.2 Formalização Lagrangiana

Conhecido como Método da Energia, Método de Euler-Lagrange ou, simplesmente, Método de Lagrange, é um método derivado do Cálculo Variacional, considerando os princípios dos Trabalhos Virtuais, de Hamilton, e da Energia Potencial Mínima, e consiste em calcular as parcelas de energia associadas ao sistema e escrever a equação de Lagrange para ele [1, 2].

Essa equação, em sua forma mais geral, pode ser escrita como:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathrm{K}}{\partial \underline{\dot{q}}}\right) - \frac{\partial \mathrm{K}}{\partial \underline{q}} + \frac{\partial \mathrm{U}}{\partial \underline{q}} + \frac{\partial \mathrm{D}}{\partial \underline{\dot{q}}} = \underline{Q} \tag{3.2.1}$$

onde:

- $\underline{q}$ é o vetor de coordenadas generalizadas;
- $\underline{Q}$ é o vetor de forçamentos generalizados;

- K, U e D representam, respectivamente, as parcelas de energia cinética, potencial e dissipada do sistema.

O vetor das coordenadas generalizadas, $\underline{q}$, representa um conjunto de coordenadas absolutas que é capaz de descrever plenamente o sistema em questão; por sua vez, o vetor de forçamentos generalizados, $\underline{Q}$, representa os forçamentos associados às coordenadas escolhidas.

As parcelas de energia, considerando um sistema cujos componentes são lineares, podem ser escritas como:

- energia cinética:

$$\mathrm{K} = \underbrace{\frac{1}{2} m |\underline{v}|^2}_{\text{translação}} + \underbrace{\frac{1}{2} \underline{\omega}^T \mathbf{I} \underline{\omega}}_{\text{rotação}} \tag{3.2.2}$$

- energia potencial:

$$\mathrm{U} = \underbrace{P\Delta}_{\text{gravitacional}} + \underbrace{\frac{1}{2} k_l |\underline{x}|^2}_{\substack{\text{elástica} \\ \text{linear}}} + \underbrace{\frac{1}{2} k_a |\underline{\theta}|^2}_{\substack{\text{elástica} \\ \text{angular}}} \tag{3.2.3}$$

  Onde Δ representa a variação de altutude do sistema, da posição de equilíbrio para a de movimento; assim, a energia potencial gravitacional, U_G, considerando a altura inicial como referência ($h_i = 0$) pode ser reescrita como:

$$\begin{aligned} \mathrm{U}_G &= P\Delta && (3.2.4) \\ &= m\ g\ (h_f - h_i) && (3.2.5) \\ &= m\ g\ h_f && (3.2.6) \end{aligned}$$

- energia dissipada:

$$\mathrm{D} = \underbrace{\frac{1}{2} c_l |\underline{\dot{x}}|}_{\text{linear}} + \underbrace{\frac{1}{2} c_a |\underline{\dot{\theta}}|}_{\text{angular}} \tag{3.2.7}$$

Para um sistema conservativo (onde não há dissipação de energia, ou $\mathrm{D} = 0$), a equação de Lagrange, (3.2.1), pode ser simplificada para[2]:

$$\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial L}{\partial \underline{\dot{q}}} \right) - \frac{\partial L}{\partial \underline{q}} = \underline{Q} \tag{3.2.8}$$

onde L, chamado de lagrangeano, é a diferença entre a energia cinética e a potencial:

$$L = \mathrm{K} - \mathrm{U} \tag{3.2.9}$$

Como resultado do equacionamento, o Método da Energia fornece apenas as equações de movimento do sistema, sem informações sobre as forças de reação.

3.3 Estudos de Casos

Segue-se alguns exemplos de problemas clássicos da teoria de vibrações mecânicas, cujo propósito é a obtenção das equações de movimento dos sistemas. Esses problemas foram retirados das referências [1, 2], e foram desenvolvidos escolhendo-se, arbitrariamente, um dos métodos anteriormente apresentados.

Essa referência apresenta os sistemas da forma mais geral possível, por isso seu desenvolvimento é inteiramente literal e o sistema massa-mola-amortecedor de 2 GdLs é generalizado para um caso de n GdLs.

3.3.1 Sistemas com 1 GdL

Massa-mola

Como representado na figura 3.3.1, o sistema massa-mola representa um objeto de massa m fixado à um referencial imóvel e capaz de se mover em apenas uma direção.

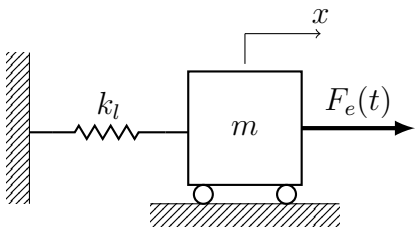

Figura 3.3.1: Diagrama do sistema massa-mola.

Como a força peso (P) será cancelada com a normal (N), apenas a força exercida pela mola no bloco é considerada; portanto, a equação de Newton, (3.1.1), para esse caso será escrita como:

$$-F_m + F_e = m\ddot{x} \tag{3.3.1}$$

onde F_m representa a força exercida pela mola no bloco, e F_e é a força externa atuante.

Considerando uma mola linear[①], $F_m = k_l x$. Assim, a equação de movimento do

[①]Considera-se que um elemento linear seja aquele que não adicione não-linearidade ao sistema; graficamente, a variação da constante elástica com a deformação, pode ser representada como uma reta no

sistema é, simplesmente:

$$m\ddot{x} + k_l x = F_e \tag{3.3.2}$$

Massa-mola-amortecedor

Análogo ao caso anterior, porém o sistema considera a dissipação de energia no amortecedor, colocado em paralelo à mola, como representado na figura 3.3.2.

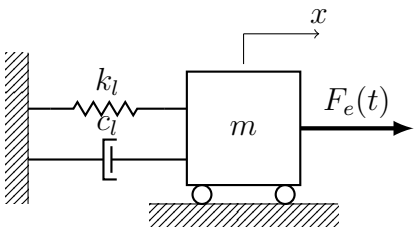

Figura 3.3.2: Diagrama do sistema massa-mola-amortecedor.

Como o sistema somente possui movimento na direção x — ou seja, sua força peso (P) será numericamente equivalente à normal (N) —, a análise do sistema se reduz a identificar as forças atuantes nessa direção.

Assim, a equação de Newton (3.1.1) para esse caso será:

$$-F_m - F_a + F_e = m\ddot{x} \tag{3.3.3}$$

onde F_m representa a força exercida pela mola no bloco, F_a a força exercida pelo amortecedor, e F_e é a força externa atuante.

Portanto, assumindo que o amortecedor e a mola sejam componentes lineares[①], a equação anterior se torna:

$$m\ddot{x} + c_l\dot{x} + k_l x = F_e \tag{3.3.4}$$

Pêndulo simples

O sistema consiste em uma massa m, presa a um referencial imóvel por uma haste inextensível, capaz de se mover em duas dimensões, como representa a figura 3.3.3.

Como não há forçamento nesse sistema, o vetor das forças generalizadas, $\underline{Q}$, será nulo; o vetor de coordenadas generalizadas, por sua vez, será:

$$\underline{q} = \{\theta\} \tag{3.3.5}$$

As parcelas de energia são:

plano cartesiano.

[①]Idem à nota anterior, contudo também aplicado à constante de amortecimento.

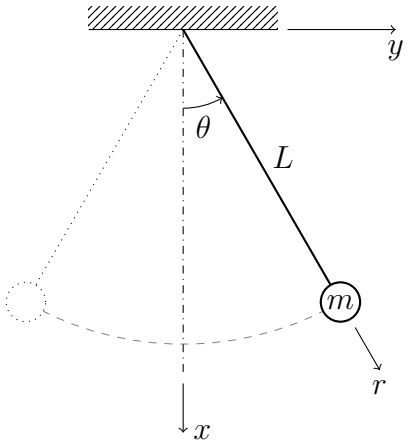

Figura 3.3.3: Diagrama do pêndulo simples.

- cinética

$$\mathrm{K} = \frac{1}{2}\, m|\underline{v}|^2 \tag{3.3.6}$$

Pode-se descrever a posição pontual do pêndulo da forma:

$$\underline{x} = \begin{Bmatrix} L\,\cos(\theta) \\ L\,\mathrm{sen}(\theta) \end{Bmatrix} \tag{3.3.7}$$

A velocidade do pêndulo, por sua vez, pode ser escrita como:

$$\underline{v} = \dot{\underline{x}} = \begin{Bmatrix} -L\,\dot{\theta}\,\mathrm{sen}(\theta) \\ L\,\dot{\theta}\,\cos(\theta) \end{Bmatrix} \tag{3.3.8}$$

Assim, seu módulo é:

$$|\underline{v}|^2 = \underline{v}[1]^2 + \underline{v}[2]^2 \tag{3.3.9}$$

$$= \dot{\theta}^2 L^2 \tag{3.3.10}$$

Assim, a parcela de energia cinética do sistema é:

$$\mathrm{K} = \frac{1}{2}\, m\,\dot{\theta}^2 L^2 \tag{3.3.11}$$

- potencial

$$\mathrm{U} = P\Delta \tag{3.3.12}$$

$$= mgL(1 - \cos(\theta)) \tag{3.3.13}$$

Como o sistema desconsidera dissipações, pode-se escrever a equação de Lagrange na forma conservativa, (3.2.8). Assim, o Lagrangeano desse sistema será:

$$L = \mathrm{K} - \mathrm{U} \tag{3.3.14}$$

$$= \frac{mL^2\dot{\theta}^2}{2} - mgL(1 - \cos(\theta)) \tag{3.3.15}$$

Derivando-se o Lagrangeano em relação ao deslocamento angular, θ, tem-se:

$$\frac{\partial L}{\partial \dot{\theta}} = mL^2\dot{\theta} \tag{3.3.16}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = mL^2\ddot{\theta} \tag{3.3.17}$$

$$\frac{\partial L}{\partial \theta} = mgL\mathrm{sen}(\theta) \tag{3.3.18}$$

Portanto, a equação de movimento do pêndulo será:

$$mL^2\ddot{\theta} + mgL\mathrm{sen}(\theta) = 0 \tag{3.3.19}$$

$$mL\left(L\ddot{\theta} + g\ \mathrm{sen}(\theta)\right) = 0 \tag{3.3.20}$$

$$L\ddot{\theta} + g\ \mathrm{sen}(\theta) = 0 \tag{3.3.21}$$

Assim, dividindo a equação anterior por L:

$$\ddot{\theta} + \frac{g}{L}\mathrm{sen}(\theta) = 0 \tag{3.3.22}$$

Pêndulo invertido

Esses sistema é uma modificação do exemplo anterior, que considera um pêndulo de haste indeformável e uma mola torcional que o leva à posição de equilíbrio instável. A força gravitacional, ao contrário do pêndulo simples, haje para forçar o sistema para longe do ponto de equilíbrio, enquanto a mola tende a levar o sistema ao equilíbrio.

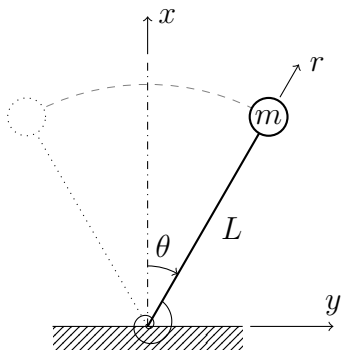

Figura 3.3.4: Diagrama do pêndulo invertido.

Como não há forçamento nesse sistema, o vetor das forças generalizadas, $\underline{Q}$, será nulo; o vetor de coordenadas generalizadas, por sua vez, será:

$$\underline{q} = \{\theta\} \tag{3.3.23}$$

As parcelas de energia desse sistema são:

- cinética

$$\mathrm{K} = \frac{1}{2} m |\underline{v}|^2 \tag{3.3.24}$$

A posição em cada instante de tempo do pêndulo pode ser escrita como:

$$\underline{x} = \begin{cases} L\ \cos(\theta) \\ L\ \mathrm{sen}(\theta) \end{cases} \tag{3.3.25}$$

Assim, a velocidade do pêndulo será dada por:

$$\underline{v} = \dot{\underline{x}} = \begin{cases} -L\ \dot{\theta}\ \mathrm{sen}(\theta) \\ L\ \dot{\theta}\ \cos(\theta) \end{cases} \tag{3.3.26}$$

O módulo da velocidade, então, será:

$$|\underline{v}|^2 = \underline{v}[1]^2 + \underline{v}[2]^2 \tag{3.3.27}$$

$$= \dot{\theta}^2 L^2 \tag{3.3.28}$$

Assim, a parcela de energia cinética do sistema será:

$$\mathrm{K} = \frac{1}{2} m\ \dot{\theta}^2 L^2 \tag{3.3.29}$$

- potencial

$$\mathrm{U} = P\Delta + \frac{1}{2} k_a |\underline{\theta}|^2 \tag{3.3.30}$$

$$= mg(L\cos(\theta) - L) + \frac{1}{2} k_a \theta^2 \tag{3.3.31}$$

$$= mgL(\cos(\theta) - 1) + \frac{1}{2} k_a \theta^2 \tag{3.3.32}$$

Como o sistema desconsidera dissipações, pode-se escrever a equação de Lagrange na

forma conservativa, (3.2.8). Assim, o lagrangeano desse sistema será:

$$L = \mathrm{K} - \mathrm{U} \tag{3.3.33}$$

$$= \frac{1}{2}m\ \dot{\theta}^2 L^2 - \left[mgL(\cos(\theta) - 1) + \frac{1}{2}k_a\theta^2\right] \tag{3.3.34}$$

Derivando-se o lagrangeano, tem-se:

$$\frac{\partial L}{\partial \dot{\theta}} = mL^2\dot{\theta} \tag{3.3.35}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = mL^2\ddot{\theta} \tag{3.3.36}$$

$$\frac{\partial L}{\partial \theta} = mgL\mathrm{sen}(\theta) - k_a\theta \tag{3.3.37}$$

Assim, substituindo as parcelas na equação de Lagrange conservativa, (3.2.8), tem-se que a equação de movimento do sistema é:

$$mL^2\ddot{\theta} + k_a\theta - mgL\mathrm{sen}(\theta) = 0 \tag{3.3.38}$$

3.3.2 Sistemas com n GdLs

Pêndulo extensível

Esse problema é uma extensão do pêndulo simples, uma vez que a única diferença entre eles é uma mola, que permite a extensão ou contração da haste ao longo do movimento. Seria uma representação mais fiel da realidade, uma vez que não existem hastes ou fios completamente inextensíveis.

Para o problema, considera-se que a mola permita apenas deslocamentos radiais.

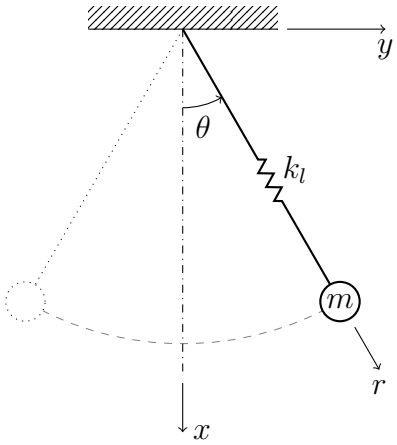

Figura 3.3.5: Diagrama do pêndulo extensível.

Como no cado do pêndulo simples, o vetor de forçamentos, $\underline{Q}$, será nulo; o vetor das

coordenadas generalizadas, contudo, será:

$$\underline{q} = \begin{Bmatrix} r \\ \theta \end{Bmatrix} \tag{3.3.39}$$

onde r representa o deslocamento ao longo da haste do pêndulo, e θ o deslocamento angular da massa m.

As parcelas de energia são:

- cinética

$$\mathrm{K} = \frac{1}{2} m |\underline{v}|^2 \tag{3.3.40}$$

Pode-se descrever a posição pontual do pêndulo da forma:

$$\underline{x} = \begin{Bmatrix} r\ \cos(\theta) \\ r\ \mathrm{sen}(\theta) \end{Bmatrix} \tag{3.3.41}$$

A velocidade do pêndulo, por sua vez, pode ser escrita como:

$$\underline{v} = \underline{\dot{x}} = \begin{Bmatrix} \dot{r}\ \cos(\theta) - r\ \dot{\theta}\ \mathrm{sen}(\theta) \\ \dot{r}\ \mathrm{sen}(\theta) + r\ \dot{\theta}\ \cos(\theta) \end{Bmatrix} \tag{3.3.42}$$

Seu módulo, portanto, será:

$$|\underline{v}|^2 = \dot{x}^2 + \dot{y}^2 \tag{3.3.43}$$

$$= \dot{r} + r^2\ \dot{\theta}^2 \tag{3.3.44}$$

Portanto:

$$\mathrm{K} = \frac{1}{2} m \dot{r} + r^2 \dot{\theta}^2 \tag{3.3.45}$$

- potencial

$$\mathrm{U} = P\Delta + \frac{1}{2} k_l |\underline{x}^2| \tag{3.3.46}$$

$$= \underbrace{mg}_{P}\ \underbrace{r(1 - \cos(\theta))}_{\Delta} + \frac{k_l r^2}{2} \tag{3.3.47}$$

Como esse é um sistema conservativo, pode-se fazer uso da euqção de Lagrange sim-

plificada, (3.2.8); assim, o Lagrangeano para esse sistema é:

$$L = \mathrm{K} - \mathrm{U} \tag{3.3.48}$$

$$= \left[\frac{1}{2}m\dot{r} + r^2\dot{\theta}^2\right] - \left[mgr(1-\cos(\theta)) + \frac{k_l r^2}{2}\right] \tag{3.3.49}$$

- primeira equação de movimento ($\underline{q}[1] = \theta$)

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \tag{3.3.50}$$

Onde:

$$\frac{\partial L}{\partial \dot{\theta}} = mr^2\dot{\theta} \tag{3.3.51}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = 2\ m\ r\ \dot{r}\ \dot{\theta} + m\ r^2\ \ddot{\theta} \tag{3.3.52}$$

$$\frac{\partial L}{\partial \theta} = -m\ g\ r\ \mathrm{sen}(\theta) \tag{3.3.53}$$

Com os resultados das parcelas, tem-se:

$$2mr\dot{r}\dot{\theta} + mr^2\ddot{\theta} + mgr\mathrm{sen}(\theta) = 0 \tag{3.3.54}$$

Assim, a primeira equação de movimento será:

$$r\ddot{\theta} + 2\dot{r}\dot{\theta} + g\ \mathrm{sen}(\theta) = 0 \tag{3.3.55}$$

- segunda equação de movimento ($\underline{q}[2] = r$)

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{r}}\right) - \frac{\partial L}{\partial r} = 0 \tag{3.3.56}$$

Onde:

$$\frac{\partial L}{\partial \dot{r}} = m\dot{r} \tag{3.3.57}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{r}}\right) = m\ \ddot{r} \tag{3.3.58}$$

$$\frac{\partial L}{\partial r} = m\ r\ \dot{\theta}^2 - m\ g + m\ g\ \cos(\theta) - k\ r \tag{3.3.59}$$

Assim, a segunda equação de movimento será:

$$m\ddot{r} - mr\dot{\theta}^2 + mg\,(1-\cos(\theta)) + k\ r = 0 \tag{3.3.60}$$

Portanto, o sistema de equações que representa o movimento do pêndulo extensível pode ser escrito como:

$$\begin{cases} r\ddot{\theta} + 2\dot{r}\dot{\theta} + g\ \text{sen}(\theta) = 0 \\ m\ddot{r} - mr\dot{\theta}^2 + mg\,(1 - \cos(\theta)) + k\ r = 0 \end{cases} \tag{3.3.61}$$

Massa-mola-amortecedor

Esse problema é uma extensão do sistema massa-mola-amortecedor apresentado anteriormente, apresentando duas massas ligadas em série por conjuntos de molas e amortecedores, coforme mostra o diagrama abaixo.

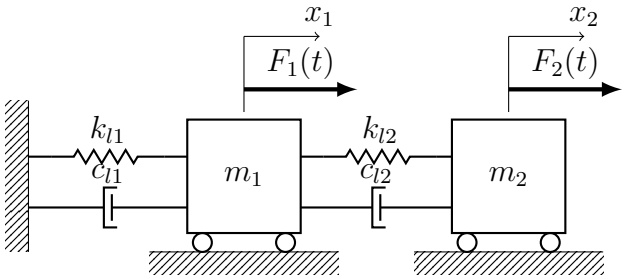

Figura 3.3.6: Diagrama do sistema massa-mola-amortecedor para 2 GDLs.

Considerando um sistema de componentes lineares e aplicando a segunda Lei de Newton para cada uma das massas do sistema, tem-se:

- corpo m_1

$$-k_{l1}x_1 - c_{l1}\dot{x}_1 + k_{l2}\,(x_2 - x_1) + c_{l2}\,(\dot{x}_2 - \dot{x}_1) + F_1(t) = m_1\ddot{x}_1 \tag{3.3.62}$$

$$m_1\ddot{x}_1 + c_{l1}\dot{x}_1 - c_{l2}\,(\dot{x}_2 - \dot{x}_1) + k_{l1}x_1 - k_{l2}\,(x_2 - x_1) = F_1(t) \tag{3.3.63}$$

$$m_1\ddot{x}_1 + c_{l1}\dot{x}_1 + c_{l2}\,(\dot{x}_1 - \dot{x}_2) + k_{l1}x_1 + k_{l2}\,(x_1 - x_2) = F_1(t) \tag{3.3.64}$$

- corpo m_2

$$-k_{l2}\,(x_2 - x_1) - c_{l2}\,(\dot{x}_2 - \dot{x}_1) + F_2(t) = m_2\ddot{x}_2 \tag{3.3.65}$$

$$m_2\ddot{x}_2 + c_{l2}\,(\dot{x}_2 - \dot{x}_1) + k_{l2}\,(x_2 - x_1) = F_2(t) \tag{3.3.66}$$

Portanto, o sistema de equações que descreve o movimento do sistema massa-mola-amortecedor é:

$$\begin{cases} m_1\ddot{x}_1 + (c_{l1} + c_{l2})\,\dot{x}_1 - c_{l2}\dot{x}_2 + (k_{l1} + k_{l2})\,x_1 - k_{l2}x_2 = F_1(t) \\ m_2\ddot{x}_2 - c_{l2}\dot{x}_1 + c_{l2}\dot{x}_2 - k_{l2}x_1 + k_{l2}x_2 = F_2(t) \end{cases} \tag{3.3.67}$$

Pode-se reescrever esse sistema de equações na forma matricial[2]:

$$\underbrace{\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}}_{\mathbf{M}} \begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{Bmatrix} + \underbrace{\begin{bmatrix} c_{l1}+c_{l2} & -c_{l2} \\ -c_{l2} & c_{l2} \end{bmatrix}}_{\mathbf{C_l}} \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} + \underbrace{\begin{bmatrix} k_{l1}+k_{l2} & -k_{l2} \\ -k_{l2} & k_{l2} \end{bmatrix}}_{\mathbf{K_l}} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} F_1(t) \\ F_2(t) \end{Bmatrix} \quad (3.3.68)$$

onde a matriz $\mathbf{M}$ é conhecida por matriz de inércia, a matriz $\mathbf{C_l}$, como matriz de amortecimento, e a matriz $\mathbf{K_l}$, como matriz de rigidez[2].

Esse problema pode ser facilmente expandido para casos com mais GdLs; por exemplo, no caso de um sistema análogo, porém com três massas, m_1, m_2 e m_3, o sistema de equações que representa o movimento desse sistema seria, em sua forma matricial:

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{Bmatrix} + \begin{bmatrix} c_{l1}+c_{l2} & -c_{l2} & 0 \\ -c_{l2} & c_{l2}+c_{l3} & -c_{l3} \\ 0 & -c_{l3} & c_{l3} \end{bmatrix} \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{Bmatrix} + \begin{bmatrix} k_{l1}+k_{l2} & -k_{l2} & 0 \\ -k_{l2} & k_{l2}+k_{l3} & -k_{l3} \\ 0 & -k_{l3} & k_{l3} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} F_1(t) \\ F_2(t) \\ F_3(t) \end{Bmatrix} \quad (3.3.69)$$

De uma forma geral, um sistema constituído por n massas, s molas e t amortecedores pode ser representado por um conjunto de equações matriciais análogo ao anterior, onde os elementos $\mathbf{K_l}[i,i]$ representam a soma das rigidezes das molas que estão conectadas ao corpo i, enquanto os elementos $-\mathbf{K_l}[i,j]$, onde $i \neq j$, representam a soma das rigidezes das molas que acoplam o corpo i ao corpo j.

A representação da matriz de amortecimento, $\mathbf{C_l}$, é análoga à da matriz de rigidez.

Se o sistema não for acoplado inercialmente[①], as três matrizes serão simétricas, e a de inércia será diagonal; caso contrário, a matriz de inércia deixará de ser diagonal, porém as três continuarão sendo simétricas[2].

[①]Isto é, se duas ou mais massas do sistema não estiverem sobrepostas com movimento relevante entre si.

Ou seja, para um sistema com n GDLs, a equação matricial de movimento seria:

$$\begin{bmatrix} m_{11} & 0 & \cdots & 0 \\ 0 & m_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_{nn} \end{bmatrix} \begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \vdots \\ \ddot{x}_n \end{Bmatrix} + \begin{bmatrix} c_{11} & -c_{12} & \cdots & -c_{1n} \\ -c_{12} & c_{22} & \cdots & -c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -c_{1n} & -c_{2n} & \cdots & c_{nn} \end{bmatrix} \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{Bmatrix}$$
$$+ \begin{bmatrix} k_{11} & -k_{12} & \cdots & -k_{1n} \\ -k_{12} & k_{22} & \cdots & -k_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -k_{1n} & -k_{2n} & \cdots & k_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} F_1(t) \\ F_2(t) \\ \vdots \\ F_n(t) \end{Bmatrix} \quad (3.3.70)$$

onde m_{ij}, c_{ij} e k_{ij} $(i, j = 1, 2, 3, ..., n)$ representam as componentes na posição i, j das matrizes $\mathbf{M}$, $\mathbf{C}$ e $\mathbf{K}$, respectivamente[①]. Suas interpretações são conforme esclarecidas acima.

[①]Seguindo a notação tradicional de matrizes, por ser mais compacta.

Capítulo 4

Solução Analítica das Equações de Movimento

Para a análise de modelos simples, ou para a validação de um código numérico escrito, faz-se necessária a solução analítica das equações de movimento obtidas, de modo a obter dados sobre o comportamento do sistema.

Comumente, sistemas reais são modelados com componentes lineares, pela conveniência proveniente da utilização do princípio da superposição, além da disposição de ferramentas matemáticas bem desenvolvidas para esse caso; em contraste, o comportamento de sistemas não-lineares não é tão bem conhecido, e, geralmente, costumam ser de difícil solução analítica[3].

Para grandes amplitudes de oscilação, todos os componentes tendem a se tornar não-lineares. Nesse estudo, porém, visa-se a minimização das amplitudes de vibração dos sistemas; assim, uma modelagem com componentes lineares satisfaz suas necessidades.

No caso de um sistema de equações lineares, o princípio da superposição pode ser usado para separar a solução do sistema em duas partes: homogênea (que considera somente as condições iniciais do sistema, e é também chamada de solução transiente[①]) e particular (que considera somente os forçamentos, também conhecida por solução permanente[②]). Assim, a solução $x(t)$ do sistema se torna:

$$x(t) = x_h(t) + x_p(t)$$

Essa análise, contudo, somente pode ser utilizada se o sistema de equações for linear.

[①] Pois seus efeitos tendem a ser reduzidos com o tempo.

[②] Ao contrário do anterior, seus efeitos duram enquanto o sistema estiver sob forçamento externo.

4.1 Solução Homogênea

Representa a parcela evanescente da solução, pois ela desaparecerá com o tempo e o sistema irá estagnar em sua posição de equilíbrio.

Para o cálculo da solução homogênea, considera-se um novo sistema de equações idêntico ao original, porém sem os termos forçantes.

4.2 Solução Particular

Representa a parcela permanente do comportamento, caso existam forçamentos no sistema, e sua influência existe até que o sistema não esteja sob a ação de forçamentos externos.

4.3 Identidade de Euler

De modo a retirar a parcela imaginária das equações e reescrevê-las como um somatório de senos e co-senos, é possível, nos casos estudados, utilizar-se das seguintes substituições[2]:

$$e^{\pm\vartheta} = \cosh(\vartheta) \pm \operatorname{senh}(\vartheta) \tag{4.3.1}$$

$$e^{\pm i\vartheta} = \cos(\vartheta) \pm i\cdot\operatorname{sen}(\vartheta) \tag{4.3.2}$$

Como exemplo, assume-se uma equação do tipo[①]:

$$u(t) = A_1 e^{i\omega_n t} + A_2 e^{-i\omega_n t} \tag{4.3.3}$$

Aplicando-se a Identidade de Euler, tem-se:

$$u(t) = A_1\left[\cos(\omega_n t) + i\cdot\operatorname{sen}(\omega_n t)\right] + A_2\left[-\cos(\omega_n t) + i\cdot\operatorname{sen}(\omega_n t)\right] \tag{4.3.4}$$

Ou, reagrupando a equação em uma forma mais conveniente:

$$u(t) = (A_1 + A_2)\cdot\cos(\omega_n t) + (A_1 - A_2)\cdot i\cdot\operatorname{sen}(\omega_n t) \tag{4.3.5}$$

Renomeando as constantes da equação anterior[2],

$$\begin{cases} C_1 = A_1 + A_2 \\ C_2 = (A_1 - A_2)\cdot i \end{cases} \tag{4.3.6}$$

[①]Que é, na verdade, a solução para a equação de movimento do sistema massa-mola.

pode-se reescrever a equação para $u(t)$ como:

$$u(t) = C_1 \cdot \cos(\omega_n t) + C_2 \cdot \text{sen}(\omega_n t) \tag{4.3.7}$$

4.4 Batimento

Para o caso de sistemas com forçamento, quando a frequência desse, Ω, se aproxima da frequência de ressonância do sistema, Ω_{res}, ocorre o efeito chamado de batimento[1, 2].

Esse efeito gera ruído sonoro e aumento na amplitude de vibração dos sistemas, potencialmente causando falhas mecânicas dependendo de sua intensidade.

4.5 Ressonância

Para o caso de sistemas com forçamento, caso a frequência desse, Ω, se iguale à frequência de ressonância do sistema, Ω_{res}, ocorre o efeito chamado de ressonância[1, 2].

Esse efeito causa um aumento considerável na amplitude de vibração dos sistemas, podendo, potencialmente, levar à ruptura de mecanismos por falha mecânica.

A ressonância de sistemas com múltiplos GdLs pode ocorrer quando um ou mais corpos sofrem influência de um forçamento cuja frequência se iguale à de ressonância.

4.6 Fator de Qualidade e Largura de Banda

Define-se como fator de qualidade do sistema, o valor da amplitude de movimento quando a frequência do forçamento, Ω, se iguala à de ressonância, Ω_{res}. Esse é o ponto de máxima potência do sistema.

Comumente, não se atinge um valor de máxima potência em um sistema real; assim, define-se a largura de banda como uma faixa de operação para o sistema, usando-se os pontos de meia potência definidos pelo fator RMS.

4.7 Estudos de Casos

Continuando os estudos da seção 3.3, apresenta-se aqui a solução analítica de alguns dos sistemas, aproveitando-se das equações de movimento já obtidas.

Novamente, as equações são resolvidas da forma mais geral possível, por isso as condições iniciais aplicadas às soluções são arbitradas como valores literais e, no caso do problema divergir em tipos de solução diferenciados (como, por exemplo, ocorre no sistema massa-mola-amortecedor), esses são estudados individualmente.

4.7.1 Sistemas com 1 GdL

Pêndulo simples

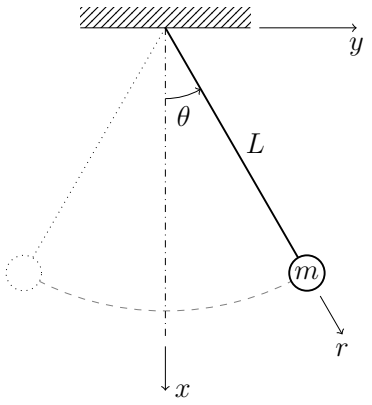

Figura 4.7.1: Diagrama do pêndulo simples.

No estudo de casos do capítulo anterior, apresentou-se a equação de movimento para o pêndulo simples, (3.3.22):

$$\ddot{\theta} + \frac{g}{L}\text{sen}(\theta) = 0 \tag{4.7.1}$$

Para resolver analiticamente essa equação, pode-se linearizar a equação acima considerando pequenas oscilações para o pêndulo. Assim, expandindo $\text{sen}(x)$ por série de Taylor na vizinhança do ponto $x = 0$:

$$\text{sen}(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + \dots \tag{4.7.2}$$

Uma aproximação razoável para linearizar o problema é truncar a relação anterior no primeiro termo, considerando que x seja dado em radianos e $x \ll 1$. Então, pode-se reescrever a equação (4.7.1) como:

$$\ddot{\theta} + \frac{g}{L}\theta = 0 \tag{4.7.3}$$

onde $\omega_n = \sqrt{g/L}$ representa a frequência natural do sistema[4].

Assumindo uma solução do tipo $\theta(t) = Ae^{\lambda t}$:

$$\begin{cases} \dot{\theta} = \lambda A e^{\lambda t} \\ \ddot{\theta} = \lambda^2 A e^{\lambda t} \end{cases} \tag{4.7.4}$$

Substituinto a solução $\theta(t)$ e suas derivadas na equação (4.7.3):

$$\lambda^2 A e^{\lambda t} + \frac{g}{L}\lambda A e^{\lambda t} = 0 \tag{4.7.5}$$

$$A e^{\lambda t}\left(\lambda^2 + \frac{g}{L}\right) = 0 \tag{4.7.6}$$

Para que a igualdade anterior seja verdadeira, uma de duas condições deve acontecer:

$$\begin{cases} A e^{\lambda t} = 0 \\ \lambda^2 + \frac{g}{L} = 0 \end{cases} \tag{4.7.7}$$

Considerando que o primeiro caso fosse verdadeiro, o sistema não entraria em movimento, pois essa parcela representa a amplitude do movimento. Portanto, considerando a segunda possibilidade:

$$\lambda^2 + \frac{g}{L} = 0 \tag{4.7.8}$$

$$\lambda = \pm\sqrt{-\frac{g}{L}} \tag{4.7.9}$$

$$\lambda_{1,2} = \pm i\sqrt{\frac{g}{L}} \tag{4.7.10}$$

ou, ainda, em termos da frequência natural do sistema, ω_n:

$$\lambda_{1,2} = \pm i\omega_n \tag{4.7.11}$$

Assim, a equação de movimento do sistema é:

$$\theta(t) = A e^{\lambda t} \tag{4.7.12}$$

$$\theta(t) = A_1 e^{i\omega_n t} + A_2 e^{-i\omega_n t} \tag{4.7.13}$$

Pode-se, ainda, reescrever essa equalção em termos de $\text{sen}(\omega_n t)$ e $\cos(\omega_n t)$, utilizando a Identidade de Euler[2]:

$$e^{\pm i\vartheta} = \cos(\vartheta) \pm i\cdot\text{sen}(\vartheta) \tag{4.7.14}$$

Voltando à equação (4.7.13), utilizando a Identidade de Euler e renomeando as cons-

tantes:

$$\theta(t) = A_1 e^{i\omega_n t} + A_2 e^{-i\omega_n t} \tag{4.7.15}$$

$$\theta(t) = A_1 \left[\cos(\omega_n t) + i \cdot \text{sen}(\omega_n t)\right] + A_2 \left[\cos(\omega_n t) - i \cdot \text{sen}(\omega_n t)\right] \tag{4.7.16}$$

$$\theta(t) = \underbrace{(A_1 + A_2)}_{C_1} \cos(\omega_n t) + \underbrace{(A_1 - A_2)\, i}_{C_2} \text{sen}(\omega_n t) \tag{4.7.17}$$

a equação de movimento pode ser reescrita como:

$$\theta(t) = C_1 \cos(\omega_n t) + C_2 \text{sen}(\omega_n t) \tag{4.7.18}$$

onde $C_1 = A_1 + A_2$, e $C_2 = (A_1 - A_2)\, i$.

Para determinar as constantes C_1 e C_2, o sistema precisa assumir uma condição inicial não-nula[①]; assim, assumindo as condições iniciais:

$$\begin{cases} \theta(0) = \theta_0 \\ \dot{\theta}(0) = \omega_0 \end{cases} \tag{4.7.19}$$

e aplicando-as à equação de movimento, é possível determinar suas constantes para um caso particular.

Aplicando a condição inicial de deslocamento e resolvendo para C_1, tem-se:

$$\begin{aligned} \theta(0) = \theta_0 &= C_1 \cdot \cos(\omega_n \cdot 0) + C_2 \cdot \text{sen}(\omega_n \cdot 0) && (4.7.20) \\ &= C_1 \cdot 1 + C_2 \cdot 0 && (4.7.21) \\ C_1 &= \theta_0 && (4.7.22) \end{aligned}$$

Para aplicar a condição inicial de velocidade, é necessário derivar a equação de movimento uma vez. Contudo, uma das constantes já foi identificada, fazendo uso disso, tem-se:

$$\begin{aligned} \theta(t) &= C_1 \cos(\omega_n t) + C_2 \text{sen}(\omega_n t) && (4.7.23) \\ &= \theta_0 \cos(\omega_n t) + C_2 \text{sen}(\omega_n t) && (4.7.24) \end{aligned}$$

[①] Já que não há forçamento nesse sistema, o movimento será devido às condições iniciais; assim, se ao menos uma delas não for nula, essa colocará o sistema em movimento. Caso existisse um forçamento, mesmo com condições iniciais nulas, ele seria o responsável pelo movimento.

Derivando a equação acima e aplicando-a no ponto $\dot{\theta}(0)$:

$$\begin{aligned}
\dot{\theta}(t) &= -\theta_0\,\omega_n\,\mathrm{sen}(\omega_n t) + C_2\,\omega_n\cos(\omega_n t) && (4.7.25)\\
\dot{\theta}(0) &= -\theta_0\,\omega_n\,\mathrm{sen}(\omega_n\cdot 0) + C_2\,\omega_n\cos(\omega_n\cdot 0) && (4.7.26)\\
&= -\theta_0\,\omega_n\cdot 0 + C_2\,\omega_n\cdot 1 && (4.7.27)\\
&= C_2\,\omega_n && (4.7.28)
\end{aligned}$$

Aplicando a equação anterior para a condição inicial $\dot{\theta}(0) = \omega_0$ e resolvendo-a para C_2:

$$\begin{aligned}
\omega_0 &= C_2\,\omega_n && (4.7.29)\\
C_2 &= \frac{\omega_0}{\omega_n} && (4.7.30)
\end{aligned}$$

Portanto, a equação de movimento do pêndulo, considerando as condições iniciais $\theta(0) = \theta_0$ e $\dot{\theta}(0) = \omega_0$ é:

$$\theta(t) = \theta_0\cdot\cos(\omega_n t) + \frac{\omega_0}{\omega_n}\cdot\mathrm{sen}(\omega_n t) \tag{4.7.31}$$

É possível, ainda, reescrever essa equação na forma de uma função harmônica defasada[2], aplicando:

$$\begin{cases} A = \sqrt{C_1^2 + C_2^2} \\ \varphi = \mathrm{atan}\left(\frac{C_2}{C_1}\right) \end{cases} \tag{4.7.32}$$

Assim, a equação de movimento será:

$$\begin{aligned}
\theta(t) &= A\cdot\cos(\omega_n t - \varphi) && (4.7.33)\\
&= \sqrt{C_1^2 + C_2^2}\cdot\cos\left(\omega_n t - \mathrm{atan}\left(\frac{C_2}{C_1}\right)\right) && (4.7.34)\\
&= \sqrt{\theta_0^2 + \left(\frac{\omega_0}{\omega_n}\right)^2}\cdot\cos\left(\omega_n t - \mathrm{atan}\left(\frac{\omega_0/\omega_n}{\theta_0}\right)\right) && (4.7.35)\\
&= \sqrt{\frac{\theta_0^2\omega_n^2 + \omega_0^2}{\omega_n^2}}\cdot\cos\left(\omega_n t - \mathrm{atan}\left(\frac{\omega_0}{\omega_n\theta_0}\right)\right) && (4.7.36)
\end{aligned}$$

Que representa um movimento oscilatório sem perda de energia; pode-se verificar mais facilmente isso em um gráfico, assumindo valores numéricos para as constantes:

- $\theta_0 = 0.1$
- $\omega_0 = 0.2$

- $L = 1, 2$
- $\omega_n = \sqrt{g/L}$

escolhendo-se, para ilustrar o problema, dois valores para o comprimento L do pêndulo — no que resulta a figura 4.7.2, gerada com o script descrito no Apêndice A.1.

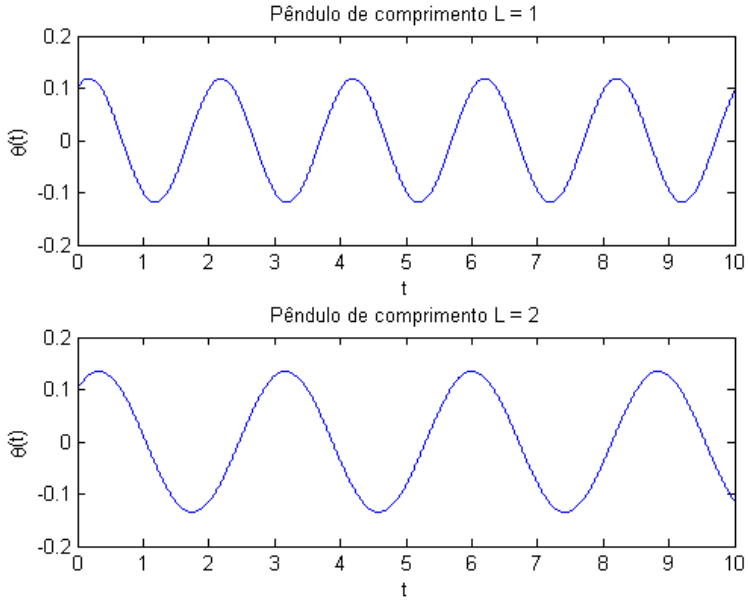

Figura 4.7.2: Solução exata da equação de movimento do pêndulo para $L = 1$ e $L = 2$.

Massa-mola-amortecedor sem forçamento

Voltando com a equação (3.3.4), apresentada anteriormente como a equação do sistema massa-mola-amortecedor:

$$m\ddot{x} + c_l\dot{x} + k_l x = F_e(t) \tag{4.7.37}$$

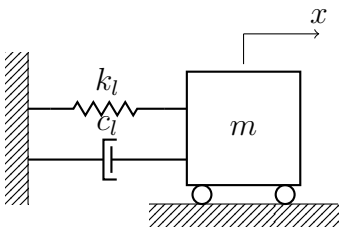

Figura 4.7.3: Diagrama do sistema massa-mola-amortecedor sem forçamento.

Considerando que $F_e(t) = 0$, pode-se assumir que a equação de movimento tenha uma solução do tipo:

$$x(t) = Ae^{\lambda t} \tag{4.7.38}$$

Derivando-se essa expressão em função do tempo, tem-se:

$$\dot{x} = A\lambda e^{\lambda t} \tag{4.7.39}$$

$$\ddot{x} = A\lambda^2 e^{\lambda t} \tag{4.7.40}$$

De modo a facilitar a solução, é possível fazer uma renomeação mais conveniente das variáveis[2]. Assim, dividindo a equação de movimento por m:

$$\ddot{x} + \frac{c_l}{m}\dot{x} + \frac{k_l}{m}x = \frac{F_e(t)}{m} \tag{4.7.41}$$

e fazendo:

$$\begin{cases} \frac{c_l}{m} = 2\xi\omega_n \\ \frac{k_l}{m} = \omega_n^2 \end{cases} \tag{4.7.42}$$

A equação (4.7.37) se torna:

$$\ddot{x} + 2\xi\omega_n\dot{x} + \omega_n^2 x = \frac{F_e(t)}{m} \tag{4.7.43}$$

Para o caso sem forçamento, $F_e(t) = 0$; assim:

$$\ddot{x} + 2\xi\omega_n\dot{x} + \omega_n^2 x = 0 \tag{4.7.44}$$

Substituindo a possível solução na equação anterior:

$$A\lambda^2 e^{\lambda t} + 2\xi\omega_n A\lambda e^{\lambda t} + \omega_n^2 A e^{\lambda t} = 0 \tag{4.7.45}$$

$$Ae^{\lambda t}\left(\lambda^2 + 2\xi\omega_n\lambda + \omega_n^2\right) = 0 \tag{4.7.46}$$

Onde, para que exista movimento, o termo $Ae^{\lambda t}$ (que representa a amplitude de oscilação do movimento[2]) não pode ser nulo; assim, a equação característica desse problema é:

$$\lambda^2 + 2\xi\omega_n\lambda + \omega_n^2 = 0 \tag{4.7.47}$$

Assim, as soluções para a equação anterior são:

$$\lambda_{1,2} = \frac{-2\xi\omega_n \pm \sqrt{4\xi^2\omega_n^2 - 4\omega_n^2}}{2} \tag{4.7.48}$$

$$= \frac{-2\xi\omega_n \pm \sqrt{4\omega_n^2\left(\xi^2 - 1\right)}}{2} \tag{4.7.49}$$

$$= \frac{-2\xi\omega_n \pm 2\omega_n\sqrt{\xi^2 - 1}}{2} \tag{4.7.50}$$

$$= -\xi\omega_n \pm \omega_n\sqrt{\xi^2 - 1} \tag{4.7.51}$$

Voltando com esse resultado na equação assumida como solução do sistema:

$$x(t) = A_1 e^{\lambda_1 t} + A_2 e^{\lambda_2 t} \tag{4.7.52}$$

$$= A_1 e^{\left(-\xi\omega_n + \omega_n\sqrt{\xi^2-1}\right)t} + A_2 e^{\left(-\xi\omega_n - \omega_n\sqrt{\xi^2-1}\right)t} \tag{4.7.53}$$

$$= A_1 e^{-\xi\omega_n t} e^{\omega_n\sqrt{\xi^2-1}\,t} + A_2 e^{-\xi\omega_n t} e^{-\omega_n\sqrt{\xi^2-1}\,t} \tag{4.7.54}$$

$$= e^{-\xi\omega_n t}\left(A_1 e^{\omega_n\sqrt{\xi^2-1}\,t} + A_2 e^{-\omega_n\sqrt{\xi^2-1}\,t}\right) \tag{4.7.55}$$

Por outro lado, a parcela $\sqrt{\xi^2 - 1}$ da solução da equação característica, (4.7.51), leva a três tipos de solução para esse problema[①]:

- sub-amortecimento $(0 < \xi < 1)$

  Voltando à solução da equação característica do problema, (4.7.51) e fazendo $\omega_D = \omega_n\sqrt{\xi^2 - 1}$:

  $$\lambda_{1,2} = -\xi\omega_n \pm \underbrace{\omega_n\sqrt{\xi^2 - 1}}_{\omega_D \in \mathbb{C}} \tag{4.7.56}$$

  Se ξ está entre 0 e 1, ξ^2 será menor do que 1, e, portanto, $\xi^2 - 1$ será menor do que zero — ou seja, $\sqrt{\xi^2 - 1} \in \mathbb{C}$. Assim:

  $$\sqrt{\xi^2 - 1} = \sqrt{-1\cdot(1 - \xi^2)} = \sqrt{-1}\cdot\sqrt{1 - \xi^2} \tag{4.7.57}$$

  $$= \underbrace{\sqrt{1 - \xi^2}}_{\in\mathbb{R}}\cdot i \tag{4.7.58}$$

  Portanto, as raízes da equação característica, (4.7.51), se tornam:

  $$\lambda_{1,2} = -\xi\omega_n \pm \omega_D i \tag{4.7.59}$$

[①]Para o caso em que $\xi = 0$, o sistema passa a ser idêntico ao massa-mola. As raízes da equação característica, (4.7.47), são puramente imaginárias e conjugadas, portanto o sistema oscila sem perda de energia.

Aplicando esse resultado, descobre-se que a equação de movimento desse sistema é:

$$x(t) = e^{-\xi\omega_n t} \cdot \left(A_1 e^{i\omega_D t} + A_2 e^{-i\omega_D t}\right) \tag{4.7.60}$$

Aplicando a Identidade de Euler e a transformação de variáveis:

$$\begin{cases} C_1 = A_1 + A_2 \\ C_2 = (A_1 - A_2) \cdot i \end{cases} \tag{4.7.61}$$

reescreve-se a equação (4.7.60) como:

$$x(t) = e^{-\xi\omega_n t} \cdot [C_1 \cos(\omega_D t) + C_2 \text{sen}(\omega_D t)] \tag{4.7.62}$$

Ou, ainda, na forma de uma função harmônica defasada, a solução para a equação de movimento pode ser escrita como:

$$x(t) = Ae^{-\xi\omega_n t} \cdot \cos(\omega_D t - \varphi) \tag{4.7.63}$$

onde:

$$\begin{cases} A = \sqrt{C_1^2 + C_2^2} \\ \varphi = \text{atan}\left(\frac{C_2}{C_1}\right) \end{cases} \tag{4.7.64}$$

Onde, a equação (4.7.63) representa um movimento oscilatório, porém com perda de energia ao longo do tempo — onde a amplitude do movimento decai conforme a parcela $Ae^{-\xi\omega_n t}$.

- super-amortecimento ($\xi > 1$)

  Voltando à solução da equação característica do problema, (4.7.51) e fazendo $\omega_P = \omega_N\sqrt{\xi^2 - 1}$:

$$\lambda_{1,2} = -\xi\omega_n \pm \underbrace{\omega_n\sqrt{\xi^2 - 1}}_{\omega_P \in \mathbb{R}} \tag{4.7.65}$$

  Como $\xi > 1$, $\xi^2 > 1$, ou seja, $\xi^2 - 1 > 0$. Portanto, $\sqrt{\xi^2 - 1} \in \mathbb{R}$. Assim:

$$\lambda_{1,2} = -\xi\omega_n \pm \omega_P \tag{4.7.66}$$

  Voltando com esse resultado na solução da equação de movimento, a equação (4.7.55)

será reescrita como:

$$x(t) = e^{-\xi\omega_n t} \cdot \left(A_1 e^{\omega_P t} + A_2 e^{-\omega_P t}\right) \tag{4.7.67}$$

Como no caso anterior, pode-se reescrever a equação anterior utilizando a Indentidade de Euler:

$$x(t) = e^{-\xi\omega_n t} \left[C_1 \cdot \cosh(\omega_P t) + C_2 \cdot \text{senh}(\omega_P t)\right] \tag{4.7.68}$$

onde:

$$\begin{cases} C_1 = A_1 + A_2 \\ C_2 = A_1 - A_2 \end{cases} \tag{4.7.69}$$

Como a solução do sistema está no domínio dos números reais, a solução não apresenta uma resposta oscilatória, como representa a Identidade de Euler para o caso $e^{\pm\vartheta}$, conforme apresentado na seção 4.3. Assim, o sistema retorna rapidamente para a posição de equilíbrio, sem que se caracterize uma oscilação.

- amortecimento crítico ($\xi = 1$)

  Ainda que na prática considere-se apenas o caso super-amortecido[2], por apresentar uma resposta idêntica a esse[①] e ser de mais fácil solução, ainda é interessante observar esse caso, pois, ao contrário dos dois anteriores, a solução da equação característica não gera uma base para a solução da EDO original do problema, (4.7.43).

  Assim, voltando à solução da equação característica do problema, (4.7.51) e fazendo $\xi = 1$:

$$\begin{aligned} \lambda_{1,2} &= -\xi\omega_n \pm \omega_n\sqrt{\xi^2 - 1} && (4.7.70)\\ &= -1\omega_n \pm \omega_n\sqrt{1^2 - 1} && (4.7.71)\\ &= -\omega_n \pm \omega_n\sqrt{0} && (4.7.72)\\ &= -\omega_n && (4.7.73) \end{aligned}$$

  Para resolver a EDO que representa o movimento desse sistema, são necessárias duas funções, $f_1(t)$ e $f_2(t)$, que, além de atenderem à equação (4.7.43), devem ser linearmente independentes entre si. A solução da equação característica apresenta

[①] Além do fato de ser complicada a definição de um amortecedor cujo valor do coeficiente adimensional de amortecimento, ξ, seja exatamente igual à unidade.

apenas uma dessas equações:

$$f_1(t) = A_1 e^{-\omega_n t} \tag{4.7.74}$$

assim, deseja-se uma função, $f_2(t)$, tal que:

$$f_2(t) = g(t) \cdot f_1(t) \tag{4.7.75}$$

também seja solução da EDO. A função mais simples que atende esse caso é[2]:

$$\ddot{g}(t) = 0 \tag{4.7.76}$$

$$\dot{g}(t) = A_2 \tag{4.7.77}$$

$$g(t) = A_2 t + A_3 \tag{4.7.78}$$

Voltando com esse resultado na definição de $f_2(t)$:

$$\begin{aligned} f_2(t) &= g(t) \cdot f_1(t) && (4.7.79)\\ &= g(t) \cdot A_1 e^{-\omega_n t} && (4.7.80)\\ &= (A_2 t + A_3) \cdot A_1 e^{-\omega_n t} && (4.7.81)\\ &= A_2 t \cdot A_1 e^{-\omega_n t} + A_3 \cdot A_1 e^{-\omega_n t} && (4.7.82)\\ &= \underbrace{A_1 A_2}_{A_4} t e^{-\omega_n t} + \underbrace{A_1 A_3}_{A_5} e^{-\omega_n t} && (4.7.83)\\ &= A_4 t e^{-\omega_n t} + A_5 e^{-\omega_n t} && (4.7.84) \end{aligned}$$

Com as duas funções, pode-se, então, definir a solução $x(t)$ da EDO:

$$\begin{aligned} x(t) &= f_1(t) + f_2(t) && (4.7.85)\\ &= A_1 e^{-\omega_n t} + A_4 t e^{-\omega_n t} + A_5 e^{-\omega_n t} && (4.7.86)\\ &= \underbrace{(A_1 + A_5)}_{A_6} e^{-\omega_n t} + A_4 t e^{-\omega_n t} && (4.7.87)\\ &= \underbrace{A_6 e^{-\omega_n t}}_{f_1^*} + \underbrace{A_4 t e^{-\omega_n t}}_{f_2^*} && (4.7.88) \end{aligned}$$

Pode-se verificar a ortogonalidade das funções encontradas, aplicando-se o Wrons-

kiano:

$$W = \det\begin{bmatrix} f_1^* & f_2^* \\ \dot{f}_1^* & \dot{f}_2^* \end{bmatrix} \tag{4.7.89}$$

$$= \det\begin{bmatrix} A_6 e^{-\omega_n t} & A_4 t e^{\omega_n t} \\ -\omega_n A_6 e^{-\omega_n t} & A_4 e^{-\omega_n t} - A_4 \omega_n t e^{-\omega_n t} \end{bmatrix} \tag{4.7.90}$$

$$= A_4 A_6 e^{-2\omega_n t} - A_4 A_6 \omega_n t e^{-2\omega_n t} + A_4 A_6 \omega_n t e^{-2\omega_n t} \tag{4.7.91}$$

$$= A_4 A_6 e^{-2\omega_n t} \tag{4.7.92}$$

Nota-se que a componente $e^{-2\omega_n t} \neq 0 \ \forall t$, e, para que as funções $f_1^*(t)$ e $f_2^*(t)$ sejam diferentes de zero, $A_4 \neq 0$ e $A_6 \neq 0$. Portanto, a nova solução $x(t)$ é solução da EDO (4.7.43).

Massa-mola-amortecedor com forçamento

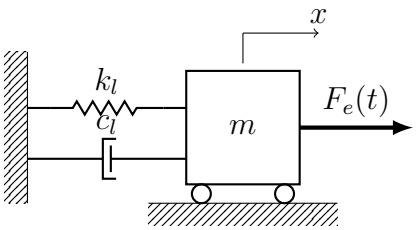

Figura 4.7.4: Diagrama do sistema massa-mola-amortecedor com forçamento.

Quando a EDO que deu origem ao problema for linear, pode-se aplicar o teorema da superposição para a obtenção da solução da equação de movimento do sistema, $x(t)$[2]:

$$x(t) = x_h(t) + x_p(t) \tag{4.7.93}$$

onde:

- $x_h(t)$: solução homogênea do problema;
- $x_p(t)$: solução particular do problema.

Portanto, esse problema se subdivide em três casos:

- $x(t) = x_h(t)$: não existem forçamentos aplicados ao sistema, e o movimento é causado, apenas, pela aplicação de condições iniciais;
- $x(t) = x_p(t)$: não foram aplicadas condições iniciais ao problema, e o movimento é causado, apenas, pela aplicação de forçamentos;
- $x(t) = x_h(t) + x_p(t)$: o movimento do sistema é causado tanto pelas condições iniciais impostas quanto pelos forçamentos aplicados.

Como a solução homogênea desse problema já foi estudada no caso anterior, aqui será apresentada, somente, a parcela particular da solução, $x_p(t)$. Caso existam condições iniciais aplicadas ao problema, basta somar a solução obtida anteriormente à que será apresentada para a obtenção do movimento real do sistema.

Assim, retomando a EDO de movimento do sistema, (3.3.4):

$$m\ddot{x} + c_l\dot{x} + k_l x = F_e(t) \tag{4.7.94}$$

Assumindo que o forçamento aplicado, $F_e(t)$, seja representado por uma função harmônica da forma:

$$F_e(t) = F_e\cos(\Omega t) \tag{4.7.95}$$

E, considerando a seguinte transformação de coordenadas:

$$\begin{cases} k_l = \omega_n^2 \\ c_l = 2\xi\omega_n \\ F_e = f_e\omega_n^2 \end{cases} \tag{4.7.96}$$

a EDO anterior pode ser reescrita como:

$$\ddot{x} + 2\xi\omega_n\dot{x} + \omega_n^2 x = f_e\omega_n^2\cos(\Omega t) \tag{4.7.97}$$

Assim, assumindo que a solução particular do problema seja do tipo:

$$x_p(t) = B_1\cdot\text{sen}(\Omega t) + B_2\cdot\cos(\Omega t) \tag{4.7.98}$$

suas derivadas, portanto, são:

$$\dot{x}_p(t) = B_1\Omega\cos(\Omega t) - B_2\Omega\text{sen}(\Omega t) \tag{4.7.99}$$

$$\ddot{x}_p(t) = -B_1\Omega^2\text{sen}(\Omega t) - B_2\Omega^2\cos(\Omega t) \tag{4.7.100}$$

Substituindo a solução assumida na EDO (4.7.97), tem-se:

$$\begin{aligned} -B_1\Omega^2\text{sen}(\Omega t) - B_2\Omega^2\cos(\Omega t) + 2\xi\omega_n\left[B_1\Omega\cos(\Omega t) - B_2\Omega\text{sen}(\Omega t)\right] \\ + \omega_n^2\left[B_1\cdot\text{sen}(\Omega t) + B_2\cdot\cos(\Omega t)\right] = f_e\omega_n^2\cos(\Omega t) \end{aligned} \tag{4.7.101}$$

Separando os termos em seno da equação (4.7.101):

$$-B_1\Omega^2 - 2\xi\omega_n\Omega B_2 + B_1\omega_n^2 = 0 \tag{4.7.102}$$

$$-B_1\Omega^2 + B_1\omega_n^2 = 2\xi\omega_n\Omega B_2 \tag{4.7.103}$$

$$B_1 = \frac{2\xi\omega_n\Omega}{\omega_n^2 - \Omega^2} B_2 \tag{4.7.104}$$

Separando os termos em co-seno da equação (4.7.101):

$$-B_2\Omega^2 + 2\xi\omega_n\Omega B_1 + B_2\omega_n^2 = f_e\omega_n^2 \tag{4.7.105}$$

$$\left(\omega_n^2 - \Omega^2\right) B_2 + 2\xi\omega_n\Omega B_1 = f_e\omega_n^2 \tag{4.7.106}$$

Substituindo o resultado encontrado para B_1 em (4.7.104) na equação anterior:

$$\left(\omega_n^2 - \Omega^2\right) B_2 + 2\xi\omega_n\Omega \left[\frac{2\xi\omega_n\Omega}{\omega_n^2 - \Omega^2} B_2\right] = f_e\omega_n^2 \tag{4.7.107}$$

$$\left(\omega_n^2 - \Omega^2\right) B_2 + \frac{4\xi^2\omega_n^2\Omega^2}{\omega_n^2 - \Omega^2} B_2 = f_e\omega_n^2 \tag{4.7.108}$$

$$\left[\omega_n^2 - \Omega^2 + \frac{4\xi^2\omega_n^2\Omega^2}{\omega_n^2 - \Omega^2}\right] B_2 = f_e\omega_n^2 \tag{4.7.109}$$

$$\left[\frac{\omega_n^2(\omega_n^2 - \Omega^2) + \Omega^2(\omega_n^2 - \Omega^2) + 4\xi^2\omega_n^2\Omega^2}{\omega_n^2 - \Omega^2}\right] B_2 = f_e\omega_n^2 \tag{4.7.110}$$

$$\left[\frac{(\omega_n^2 - \Omega^2)\cdot(\omega_n^2 - \Omega^2) + 4\xi^2\omega_n^2\Omega^2}{\omega_n^2 - \Omega^2}\right] B_2 = f_e\omega_n^2 \tag{4.7.111}$$

$$B_2 = \frac{f_e\omega_n^2(\omega_n^2 - \Omega^2)}{(\omega_n^2 - \Omega^2)^2 + 4\xi^2\omega_n^2\Omega^2} \tag{4.7.112}$$

Voltando com esse resultado na equação (4.7.104):

$$B_1 = \frac{2\xi\omega_n\Omega}{\omega_n^2 - \Omega^2} \left[\frac{f_e\omega_n^2(\omega_n^2 - \Omega^2)}{(\omega_n^2 - \Omega^2)^2 + 4\xi^2\omega_n^2\Omega^2}\right] \tag{4.7.113}$$

$$B_1 = \frac{2\xi\omega_n^3\Omega f_e}{(\omega_n^2 - \Omega^2)^2 + 4\xi^2\omega_n^2\Omega^2} \tag{4.7.114}$$

Assim, a solução particular da equação de movimento, (4.7.98) se torna:

$$x_p(t) = \frac{2\xi\omega_n^3\Omega f_e}{(\omega_n^2 - \Omega^2)^2 + 4\xi^2\omega_n^2\Omega^2}\cdot\text{sen}(\Omega t) + \frac{f_e\omega_n^2(\omega_n^2 - \Omega^2)}{(\omega_n^2 - \Omega^2)^2 + 4\xi^2\omega_n^2\Omega^2}\cdot\cos(\Omega t) \tag{4.7.115}$$

Usando a transformação de variáveis

$$\begin{cases} B_1 = B\cdot\text{sen}(\phi) \\ B_2 = B\cdot\cos(\phi) \end{cases} \tag{4.7.116}$$

a equação (4.7.115) pode ser reescrita como[2]:

$$x_p(t) = B \cdot \cos(\Omega t - \phi) \tag{4.7.117}$$

Considerando que ϕ pode ser obtido como:

$$\phi = \operatorname{atan}\left(\frac{2\xi\omega_n\Omega}{\omega_n^2 - \Omega^2}\right) \tag{4.7.118}$$

e B da seguinte forma:

$$B = \frac{\omega_n^2 f_e}{(\omega_n^2 - \Omega^2)^2 + 4\xi^2\omega_n^2\Omega^2}\sqrt{(\omega_n^2 - \Omega^2)^2 + 4\xi^2\omega_n^2\Omega^2} \tag{4.7.119}$$

Na equação anterior, B pode ser visto como uma função da frequência do forçamento, ou seja, $B = B(\Omega)$. Assim, utilizando a seguinte manipulação:

$$\frac{\sqrt{a}}{a} \cdot \frac{\sqrt{a}}{\sqrt{a}} = \frac{a}{a\sqrt{a}} = \frac{1}{\sqrt{a}} \tag{4.7.120}$$

pode-se reescrever $B(\Omega)$ como:

$$\begin{aligned}
B(\Omega) &= \frac{\omega_n^2 f_e}{\sqrt{(\omega_n^2 - \Omega^2)^2 + 4\xi^2\omega_n^2\Omega^2}} && (4.7.121)\\
&= \frac{\omega_n^2 f_e}{\sqrt{\omega_n^4 \cdot \left[\left(1 - \frac{\Omega^2}{\omega_n^2}\right)^2 + 4\xi^2\frac{\Omega^2}{\omega_n^2}\right]}} && (4.7.122)\\
&= \frac{\omega_n^2 f_e}{\omega_n^2\sqrt{\left(1 - \frac{\Omega^2}{\omega_n^2}\right)^2 + 4\xi^2\frac{\Omega^2}{\omega_n^2}}} && (4.7.123)\\
&= \frac{f_e}{\sqrt{\left(1 - \frac{\Omega^2}{\omega_n^2}\right)^2 + 4\xi^2\frac{\Omega^2}{\omega_n^2}}} && (4.7.124)
\end{aligned}$$

Fazendo $r = {}^{\Omega}/_{\omega_n}$, e definindo-se a função $G(r)$ como ${}^{B(\Omega)}/_{f_e}$, obtem-se:

$$G(r) = \frac{1}{\sqrt{(1 - r^2)^2 + 4\xi^2 r^2}} \tag{4.7.125}$$

que é o chamado fator de amplificação de frequências[2]. Esse fator representa a amplificação da amplitude do sistema para efeitos como ressonância ou batimento. Para os casos anteriormente estudados, esse fator se torna[1]:

- $\xi = 0$ (sem amortecimento)

$$|G(r)| = \frac{1}{1 - r^2} \tag{4.7.126}$$

- $\xi = 0.5$ (sub-amortecido)

$$\begin{aligned} |G(r)| &= \frac{1}{\sqrt{1 - 2r^2 + r^4 + r^2}} && (4.7.127)\\ &= \frac{1}{\sqrt{r^4 - r^2 + 1}} && (4.7.128) \end{aligned}$$

- $\xi = 1$ (criticamente amortecido)

$$\begin{aligned} |G(r)| &= \frac{1}{\sqrt{1 - 2r^2 + r^4 + 4r^2}} && (4.7.129)\\ &= \frac{1}{1 + r^2} && (4.7.130) \end{aligned}$$

Para encontrar o ponto de máximo da função $G(r)$, faz-se:

$$\frac{\mathrm{d}G(r)}{\mathrm{d}r} = 0 \tag{4.7.131}$$

Assim:

$$\frac{\mathrm{d}}{\mathrm{d}r}\left[\left(1 - r^2\right)^2 + 4\xi^2 r^2\right]^{-1/2} = 0 \tag{4.7.132}$$

$$\left[-2r \cdot 2(1 - r^2) + 8\xi^2 r\right] \cdot \left(-\frac{1}{2}\right)\left[\left(1 - r^2\right)^2 + 4\xi^2 r^2\right]^{-3/2} = 0 \tag{4.7.133}$$

Para que o produto anterior seja nulo, um de seus termos deve sê-lo. Assim, assumindo que o primeiro termo seja igual a zero:

$$-4r(1 - r^2) + 8\xi^2 r = 0 \tag{4.7.134}$$

$$4r\left[2\xi^2 - 1 + r^2\right] = 0 \tag{4.7.135}$$

O que resulta em duas possibilidades:

$$\begin{cases} r = 0 \\ 2\xi^2 - 1 + r^2 = 0 \end{cases} \tag{4.7.136}$$

Como r ser nulo implicaria que a frequência do forçamento, Ω, seria nula, considera-se:

$$2\xi^2 - 1 + r^2 = 0 \tag{4.7.137}$$

$$r = \pm\sqrt{1 - 2\xi^2} \tag{4.7.138}$$

Para respostas em termos da frequência do sistema, desconsidera-se a parte negativa[4]; assim:

$$r = \sqrt{1 - 2\xi^2} \tag{4.7.139}$$

$$\frac{\Omega}{\omega_n} = \sqrt{1 - 2\xi^2} \tag{4.7.140}$$

$$\Omega_{res} = \omega_n\sqrt{1 - 2\xi^2} \tag{4.7.141}$$

Ω_{res} é a frequência de ressonância do sistema.

Retomando-se a definição de $G(\Omega)$:

$$G(\Omega) = \frac{1}{\sqrt{\left(1 - \frac{\Omega^2}{\omega_n^2}\right)^2 + 4\xi^2\frac{\Omega^2}{\omega_n^2}}} \tag{4.7.142}$$

Substituindo, na equação anterior, o valor encontrado para Ω_{res} em (4.7.141):

$$G_{max} = G(\Omega_{res}) = \frac{1}{\sqrt{\left(1 - (1 - 2\xi^2)\right)^2 + 4\xi^2\left(1 - 2\xi^2\right)}} \tag{4.7.143}$$

$$= \frac{1}{\sqrt{4\xi^4 + 4\xi^2 - 8\xi^4}} \tag{4.7.144}$$

$$= \frac{1}{\sqrt{4\xi^2 - 4\xi^4}} \tag{4.7.145}$$

$$= \frac{1}{2\xi\sqrt{1 - \xi^2}} \tag{4.7.146}$$

Fazendo $\xi \ll 1$, $\sqrt{1 - \xi^2} \approx 1$; então, a equação anterior se torna:

$$G_{max} = \frac{1}{2\xi} \tag{4.7.147}$$

Essa amplitude é conhecida como fator de qualidade do sistema[1].

Pode-se, ainda, definir a largura de banda desse sistema como:

$$\text{RMS} = \frac{G_{max}}{\sqrt{2}} \tag{4.7.148}$$

$$= \frac{G(\Omega_{res})}{\sqrt{2}} \tag{4.7.149}$$

Para identificar as frequências que determinam a largura de banda, ω_1 e ω_2, faz-se[2]:

$$\frac{1}{2\xi\sqrt{2-2\xi^2}} = \frac{1}{\sqrt{(1-r^2)^2+4\xi^2r^2}} \tag{4.7.150}$$

$$2\xi\sqrt{2-2\xi^2} = \sqrt{(1-r^2)^2+4\xi^2r^2} \tag{4.7.151}$$

$$4\xi^2(2-2\xi^2) = \left(1-r^2\right)^2+4\xi^2r^2 \tag{4.7.152}$$

$$8\xi^2-8\xi^4 = 1-2r^2+r^4+4\xi^2r^2 \tag{4.7.153}$$

$$r^4+(4\xi^2-2)r^2+(8\xi^4-8\xi^2+1) = 0 \tag{4.7.154}$$

Fazendo $m = r^2$, tem-se:

$$m^2+(4\xi^2-2)m+(8\xi^4-8\xi^2+1) = 0 \tag{4.7.155}$$

Dessa forma, as soluções da equação anterior são:

$$m_{1,2} = \frac{2-4\xi^2 \pm \sqrt{(4\xi^2-2)^2-4(8\xi^4-8\xi^2+1)}}{2} \tag{4.7.156}$$

$$= \frac{2-4\xi^2 \pm \sqrt{(16\xi^4-16\xi^2+4)-(32\xi^4-32\xi^2+4)}}{2} \tag{4.7.157}$$

$$= \frac{2-4\xi^2 \pm \sqrt{16\xi^2-16\xi^4}}{2} \tag{4.7.158}$$

$$= \frac{2-4\xi^2 \pm \sqrt{16\xi^2\left(1-\xi^2\right)}}{2} \tag{4.7.159}$$

$$= \frac{2-4\xi^2 \pm 4\xi\sqrt{1-\xi^2}}{2} \tag{4.7.160}$$

$$= 1-2\xi^2 \pm 2\xi\sqrt{1-\xi^2} \tag{4.7.161}$$

Para o caso em que $\xi \ll 1$, $\sqrt{1-\xi^2} \approx 1$; assim:

$$m \approx 1-2\xi^2 \pm 2\xi \tag{4.7.162}$$

$$\approx (1\pm\xi)^2 \tag{4.7.163}$$

onde ambas as soluções são positivas.

Retornando à variável original, $r = \pm\sqrt{m}$, as soluções da equação original são:

$$r = \pm\sqrt{(1\pm\xi^2)^2} \tag{4.7.164}$$

$$= \pm\left(1\pm\xi^2\right) \tag{4.7.165}$$

Nesse caso, desconsidera-se a parte negativa da solução. Assim, voltando r à sua

definição original, tem-se:

$$\frac{\Omega}{\omega_n} = \left(1 \pm \xi^2\right) \tag{4.7.166}$$

$$\Omega = \omega_n \left(1 \pm \xi^2\right) \tag{4.7.167}$$

Assim, as frequências que definem a largura de banda são:

$$\begin{cases} \omega_1 = \omega_n \left(1 - \xi^2\right) \\ \omega_2 = \omega_n \left(1 + \xi^2\right) \end{cases} \tag{4.7.168}$$

4.7.2 Sistemas com n GdLs

Massa-mola-amortecedor sem forçamento

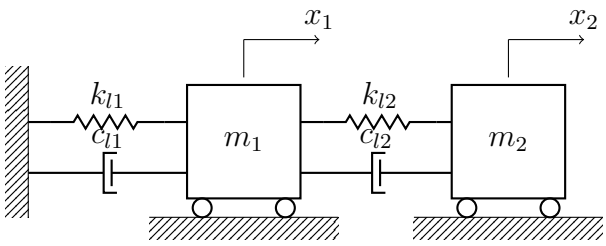

Figura 4.7.5: Diagrama do sistema massa-mola-amortecedor sem forçamento com 2 GDLs.

Retornando à equação de movimento do sistema, obtida em (3.3.68):

$$\mathbf{M}\underline{\ddot{x}} + \mathbf{C}\underline{\dot{x}} + \mathbf{K}\underline{x} = \underline{F}(t) \tag{4.7.169}$$

Existem três principais formas de solucionar essa equação[2]:

- tratando o sistema como EDOs de segunda ordem;
- tratando o sistema como EDOs de primeira ordem;
- amortecimento proporcional.

A implementação dos métodos numéricos de Runge-Kutta exige que o sistema de equações passado seja de primeira ordem; assim, a solução mais conveniente para esse problema seria resolvê-lo pelo segundo método.

Assim, reescrevendo a equação de movimento do sistema como:

$$\begin{cases} m_1\ddot{x}_1 + \left(c_{l1} + c_{l2}\right)\dot{x}_1 - c_{l2}\dot{x}_2 + \left(k_{l1} + k_{l2}\right)x_1 - k_{l2}x_2 = 0 \\ m_2\ddot{x}_2 - c_{l2}\dot{x}_1 + c_{l2}\dot{x}_2 - k_{l2}x_1 + k_{l2}x_2 = 0 \end{cases} \tag{4.7.170}$$

e considerando a seguinte transformação de variáveis:

$$\begin{array}{ll} x_1 = u_1 & x_2 = u_2 \\ \dot{x}_1 = \dot{u}_1 = v_1 & \dot{x}_2 = \dot{u}_2 = v_2 \\ \ddot{x}_1 = \ddot{u}_1 = \dot{v}_1 & \ddot{x}_2 = \ddot{u}_2 = \dot{v}_2 \end{array}$$

pode-se escrever o sistema de equações de movimento como:

$$\begin{cases} \dot{u}_1 = v_1 \\ \dot{u}_2 = v_2 \\ m_1\dot{v}_1 + (c_{l1} + c_{l2})\, v_1 - c_{l2}v_2 + (k_{l1} + k_{l2})\, u_1 - k_{l2}u_2 = 0 \\ m_2\dot{v}_2 - c_{l2}v_1 + c_{l2}v_2 - k_{l2}u_1 + k_{l2}u_2 = 0 \end{cases} \tag{4.7.171}$$

Pode-se separar os termos das equações da seguinte forma:

$$\begin{cases} \dot{u}_1 = v_1 \\ \dot{u}_2 = v_2 \\ \dot{v}_1 = m_1^{-1} \cdot [-(c_{l1} + c_{l2})\, v_1 + c_{l2}v_2 - (k_{l1} + k_{l2})\, u_1 + k_{l2}u_2] \\ \dot{v}_2 = m_2^{-1} \cdot [c_{l2}v_1 - c_{l2}v_2 + k_{l2}u_1 - k_{l2}u_2] \end{cases} \tag{4.7.172}$$

que, por sua vez, pode ser visto na forma matricial:

$$\begin{Bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{v}_1 \\ \dot{v}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -(k_{l1} + k_{l2})\, m_1^{-1} & k_{l2}m_1^{-1} & -(c_{l1} + c_{l2})\, m_1^{-1} & c_{l2}m_1^{-1} \\ k_{l2}m_2^{-1} & -k_{l2}m_2^{-1} & c_{l2}m_2^{-1} & -c_{l2}m_2^{-1} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ v_1 \\ v_2 \end{Bmatrix} \tag{4.7.173}$$

Por conveniência, pode-se reescrever o lado esquerdo da equação com relação a uma matriz identidade. Assim, considerando:

$$p = \begin{Bmatrix} u_1 \\ u_2 \\ v_1 \\ v_2 \end{Bmatrix} \tag{4.7.174}$$

o sistema anterior pode ser escrito como:

$$\mathbf{I}\ \underline{\dot{p}} = \mathbf{A}\ \underline{p} \tag{4.7.175}$$

$$\mathbf{A}\ \underline{p} - \mathbf{I}\ \underline{\dot{p}} = \underline{0} \tag{4.7.176}$$

Analogamente à solução do sistema massa-mola-amortecedor, assume-se que a solução

tenha a forma:

$$\underline{p}(t) = \underline{P}e^{\lambda t} \tag{4.7.177}$$

onde deseja-se determinar os valores de $\underline{\mathrm{P}}$ que satisfaçam o sistema, onde os elementos P_i são constantes, derivando-se a equação anterior:

$$\dot{\underline{p}}(t) = \underline{P}\lambda e^{\lambda t} \tag{4.7.178}$$

e, substituindo em 4.7.176, tem-se que:

$$\mathbf{A}\underline{P}e^{\lambda t} - \mathbf{I}\underline{P}\lambda e^{\lambda t} = \underline{0} \tag{4.7.179}$$

$$[\mathbf{A} - \lambda\mathbf{I}]\,\underline{P} = \underline{0} \tag{4.7.180}$$

Para um sistema de equações lineares, se o determinante da matriz dos coeficientes — no caso, a matriz $\mathbf{A} - \lambda\mathbf{I}$ —, for diferente de zero, esse sistema só admite a solução homogênea[5]. Assim:

$$\det[\mathbf{A} - \lambda\mathbf{I}] = 0 \tag{4.7.181}$$

Por questão de simplificação, assume-se os seguintes valores:

$$\begin{aligned} m_1 &= m_2 = 5 \\ c_{l1} &= c_{l2} = 25 \\ k_{l1} &= k_{l2} = 1000 \end{aligned} \tag{4.7.182}$$

Assim, o sistema torna-se:

$$\det\begin{bmatrix} -\lambda & 0 & 1 & 0 \\ 0 & -\lambda & 0 & 1 \\ -400 & 0 & -10-\lambda & 5 \\ 200 & -200 & 5 & -5-\lambda \end{bmatrix} = 0 \tag{4.7.183}$$

Para esse caso, os autovalores são[①]:

$$\lambda_1 = -6.5451 + 21.9264i \tag{4.7.184}$$

$$\lambda_2 = -6.5451 - 21.9264i \tag{4.7.185}$$

$$\lambda_3 = -0.9549 + 8.6880i \tag{4.7.186}$$

$$\lambda_4 = -0.9549 - 8.6880i \tag{4.7.187}$$

Voltando com esses valores na equação (4.7.180)

$$\begin{bmatrix} -\lambda & 0 & 1 & 0 \\ 0 & -\lambda & 0 & 1 \\ -400 & 200 & -10-\lambda & 5 \\ 200 & -200 & 5 & -5-\lambda \end{bmatrix} \begin{Bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{4.7.188}$$

pode-se encontrar os autovetores do sistema:

- $\lambda_1 = -6.5451 + 21.9264i$

$$\underline{P}_1 = \begin{bmatrix} -0.010\,62 - 0.035\,59i \\ 0.006\,57 + 0.021\,99i \\ 0.849\,84 \\ -0.525\,23 \end{bmatrix} \tag{4.7.189}$$

- $\lambda_2 = -6.5451 - 21.9264i$

$$\underline{P}_2 = \begin{bmatrix} -0.010\,62 + 0.035\,59i \\ 0.006\,57 - 0.021\,99i \\ 0.849\,84 \\ -0.525\,23 \end{bmatrix} \tag{4.7.190}$$

- $\lambda_3 = -0.9549 + 8.6880i$

$$\underline{P}_3 = \begin{bmatrix} -0.006\,53 - 0.059\,40i \\ -0.010\,56 - 0.096\,12i \\ 0.522\,32 \\ 0.845\,14 \end{bmatrix} \tag{4.7.191}$$

[①]Esses valores foram obtidos utilizando-se o comando `eig` do Octave/MATLAB. Mais informações podem ser obtidas na página de ajuda do comando, acessível dentro dos referidos ambientes com o comando `help eig`.

- $\lambda_4 = -0.9549 - 8.6880i$

$$\underline{P}_4 = \begin{bmatrix} -0.006\,53 + 0.059\,40i \\ -0.010\,56 + 0.096\,12i \\ 0.522\,32 \\ 0.845\,14 \end{bmatrix} \tag{4.7.192}$$

De posse desses valores, pode-se retornar à Equação (4.7.177) para obter as equações de movimento:

$$p_i(t) = P_i e^{\lambda t},\ \forall\, i = 1, ..., 4 \tag{4.7.193}$$

Lembrando que

$$\underline{p} = \begin{Bmatrix} u_1 \\ u_2 \\ v_1 \\ v_2 \end{Bmatrix} \tag{4.7.194}$$

portanto, obteve-se por esse método as equações dos deslocamentos e das velocidades apresentados pelo sistema.

Capítulo 5

Métodos Numéricos de Runge-Kutta

Seja considerado um problema de valor inicial definido da seguinte forma:

$$PVI \begin{cases} \dot{y} = f(y,t) \\ y(t_0) = \alpha \end{cases} \tag{5.0.1}$$

A solução analítica das equações de governo dadas por um PVI é a melhor forma de se analisar um sistema mecânico; porém, na prática, com sistemas complexos e não-lineares, faz-se necessária a utilização de métodos numéricos aproximados e precisos para a análise. Dentre a grande variedade de métodos disponíveis, os de Runge-Kutta são os mais utilizados.

Os métodos de Euler (Runge-Kutta de primeira ordem), contudo, não são muito utilizados, pois requererem um passo muito reduzido para uma precisão razoável[6]; por sua vez, os métodos de ordem superior apresentam uma boa precisão, sem a necessidade do conhecimento de derivadas de ordens superiores de $y(x)$, como acontece com o algoritmo de Taylor[6].

Dentre esses, o método de Runge-Kutta de quarta ordem oferece um bom equilíbrio entre custo computacional e precisão, sendo o mais utilizado para problemas de engenharia, ou, pelo menos, como uma primeira aproximação do problema.

Alguns softwares e linguagens de programação oferecem soluções prontas para os métodos de Runge-Kutta:

- MATLAB: oferece vários *solvers* para equações (ou sistemas de equações) diferenciais ordinárias — dentre eles, os comandos `ode45` e `ode23`, são os mais notáveis;
- GNU Octave[①]: através do pacote extra `odepkg`, oferecido pelo Forge[②], oferece funções parecidas com as do MATLAB;

[①] `https://www.gnu.org/software/octave/`, com uma sintaxe parecida com a do MATLAB, tenta ser uma alternativa gratuita a esse.

[②] `http://octave.sourceforge.net/`, oferece pacotes a mais para o Octave, que não acompanham a distribuição principal.

- a linguagem de programação Python[①] oferece os pacotes de computação científica NumPy[②] e SciPy[③] que possuem uma interface em classe para o método de quarta ordem, `scipy.integrate.ode`.

Existe, ainda, o chamado método de Fehlberg[6], que adapta o passo do método conforme a necessidade, e é capaz de gerar uma precisão maior do que o Runge-Kutta de quarta ordem.

5.1 Fluxograma dos Métodos de Runge-Kutta

Independente da ordem, os métodos de Runge-Kutta seguem um padrão genérico, que pode ser melhor representado no fluxograma da Figura 5.1.1.

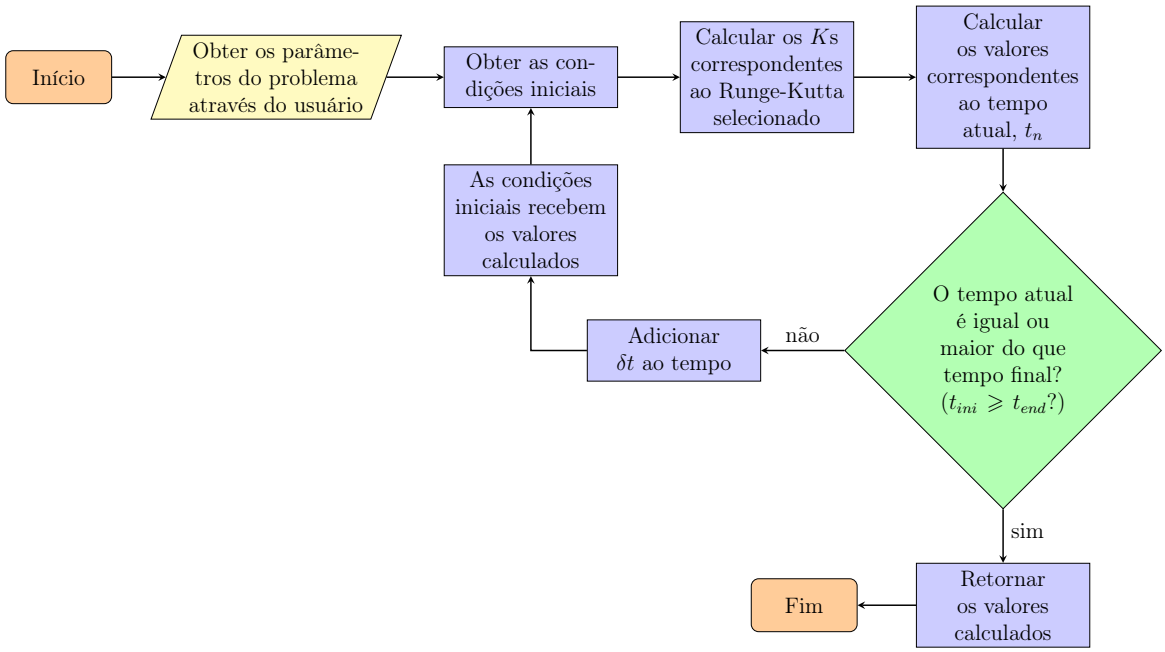

Figura 5.1.1: Fluxograma genérico dos métodos de Runge-Kutta.

5.2 Runge-Kutta 2

Conforme ilustrado na figura 5.2.1, o método consiste em calcular o coeficiente angular das retas tangentes a dois passos de evolução no tempo (retas m_1 e m_2), e aproximar a curva (no diagrama, a curva em vermelho) pela reta cujo ponto inicial corresponde ao

[①] `https://www.python.org/`

[②] `http://www.numpy.org/`, oferece a interface básica para computção científica com o Python, contendo, por exemplo, ferramentas para integração de códigos em C/C++ e FORTRAN, definições de álgebra linear, etc.

[③] `http://www.scipy.org/`, utiliza e amplia a interface do NumPy, contendo implementações de vários modelos numéricos.

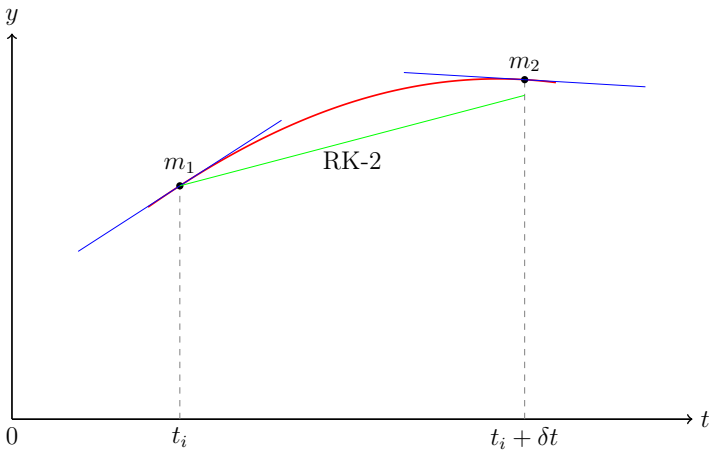

Figura 5.2.1: Esquematização do cálculo de um passo no método de Runge-Kutta de segunda ordem.

primeiro ponto (ponto t_i do diagrama), com inclinação equivalente à média dos coeficientes angulares dos dois pontos calculados (representada pela reta em verde)[7]. Na notação tradicional:

$$y^{n+1} = y^n + \frac{1}{2}(K_1 + K_2) \tag{5.2.1}$$

onde:

- $K_1 = \delta t \cdot f(y^n,\ t^n)$
- $K_2 = \delta t \cdot f(y^n + K_1,\ \underbrace{t^n + \delta t}_{t^{n+1}})$

Uma implementação desse método seria:

Algoritmo 5.2.1: rk2.m

```
clear all; clc; close all;

dt      = 1e-2;
t_ini   = 0;
t_end   = 30;
y_ini1  = 1;
y_ini2  = 2;
yp_ini1 = 1;
yp_ini2 = 0;

ti = [t_ini];
yi = [y_ini1; y_ini2; yp_ini1; yp_ini2];

while ti(end) <= t_end
    tn = ti(end);
```

```
16      yn = yi(:,end);
17
18      k_1 = dt .* mov_eq_vec(tn      , yn       );
19      k_2 = dt .* mov_eq_vec(tn .+ dt, yn .+ k_1);
20
21      yn = yn .+ (k_1 .+ k_2)./2;
22
23      ti = [ti, tn+dt];
24      yi = [yi, yn];
25  end
26
27  yi = transpose(yi);
28
29  plot(ti, yi(:,1));
30  xlabel 'Tempo'
31  ylabel 'Deslocamento da massa m_1'
```

5.3 Runge-Kutta 4

Análogo ao método anterior, porém utiliza quatro pontos e o coeficiente angular da reta é calculado através da soma ponderada com proporções 1-2-2-1[①] (ou seja, as retas intermediárias são mais significativas para o método). Assim:

$$y^{n+1} = y^n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{5.3.1}$$

onde:

- $K_1 = \delta t \cdot f(y^n,\ t^n)$
- $K_2 = \delta t \cdot f(y^n + {}^1\!/_2 \cdot K_1,\ t^n + {}^{\delta t}\!/_2)$
- $K_3 = \delta t \cdot f(y^n + {}^1\!/_2 \cdot K_2,\ t^n + {}^{\delta t}\!/_2)$
- $K_4 = \delta t \cdot f(y^n + K_3,\ \underbrace{t^n + \delta t}_{t^{n+1}})$

Uma implementação desse método pode ser encontrada no Algoritmo 6.2.2, no próximo capítulo.

5.4 Método de Fehlberg

Consiste em, para cada intervalo de tempo, calcular a evolução pelos métodos de Runge-Kutta de segunda e terceira ordem, comparando seus resultados; caso difiram além de um limite, esse ponto é recalculado com um passo menor.

[①] Embora outras sejam utilizadas, essa é a mais comum.

Existem, ainda, variações desse método, que utilizam os métodos de Runge-Kutta de quarta e quinta ordem; o MATLAB, em sua distribuição padrão, possui variações desses, adequadas a diversos tipos de problemas. Contudo, devido ao excelente equilíbrio entre precisão e tempo computacional, ao menos em uma primeira aproximação ao problema, comumente utiliza-se a função `ode45`.

Assim, a formalização matemática do Método de Fehlberg seria:

$$\begin{cases} \delta t^{n+1} = \delta t^n \text{ e } y^{n+1} = \frac{y_{rk2}^{n+1} + y_{rk3}^{n+1}}{2} & \text{se } y_{rk3}^{n+1} - \text{err}(y) \leqslant y_{rk2}^{n+1} \leqslant y_{rk3}^{n+1} + \text{err}(y) \\ \delta t^{n+1} = {}^{\delta t^n}/_2 & \text{caso contrário, até } y^{n+1} \text{ atender à anterior} \end{cases} \tag{5.4.1}$$

onde:

- y_{rk2}^{n+1} é o próximo ponto da função, calculado pelo método Runge-Kutta de segunda ordem;
- y_{rk3}^{n+1} é o próximo ponto da função, calculado pelo método Runge-Kutta de terceira ordem;
- δt^n é o passo para a iteração atual;
- δt^{n+1} é o passo para a próxima iteração;
- $\text{err}(y)$ é a variação máxima aceitável entre os dois métodos para o cálculo do próximo ponto.

Assim, nota-se facilmente que cada novo ponto calculado não corresponde a um passo do método, como acontece no método de Runge-Kutta tradicional, mas sim um subprocesso que calcula q vezes o novo ponto, reduzindo o passo δt^{n+1} para a próxima passagem do mesmo subprocesso tantas vezes quantas forem necessárias. Por causa disso, o Método de Fehlberg também é conhecido como Runge-Kutta de Passo Adaptativo[7].

O número de repetições do subprocesso, q, varia conforme a precisão inicial fornecida (δt_{ini}), com o erro máximo aceitável (δx) e com a equação a ser calculada.

A representação e a interpretação do método para o caso da função `ode45` são análogas.

5.5 Runge-Kutta Paralelo

Segundo a lei de Moore[①], a um custo constante, o poder de processamento dos computadores dobraria a cada dois anos[②]; então, para fazer com que um programa executasse

[①] `http://en.wikipedia.org/wiki/Moore%27s_law`

[②] Na sua forma original, a lei diz, na verdade, que o número de transistores em um circuito integrado seria capaz de dobrar a cada dois anos, considerando um custo constante. Em essência, as duas interpretações são equivalentes.

com o dobro da velocidade, sem aplicar otimizações ao código original, bastaria apenas esperar dois anos e comprar um processador mais avançado.

Porém, com o advento dos processadores com múltiplos núcleos com o Celeron Dual-Core da Intel, além da lei de Moore ter se tornado obsoleta, para que os programas executem com o dobro da velocidade são necessárias alterações no código-fonte para que, ao invés de seguir instruções lineares, esses se dividam em processos (ou *threads*) capazes de serem executados simultaneamente.

Os métodos de Runge-Kutta sofrem uma ligeira desvantagem com relação a isso, pois o cálculo de cada novo ponto depende do cálculo do ponto anterior, o que torna complicada a divisão de tarefas entre múltiplos processadores.

Dentre os apresentados, o método de Fehlberg é o único que pode ser facilmente adaptado para processadores com dois núcleos. Como esse utiliza as soluções de dois métodos para o cálculo de cada novo ponto, esses poderiam ser enviados como instruções em paralelo para o processador[①], e o novo ponto seria calculado assim que essas fossem concluídas. Contudo, essa adaptação utilizaria apenas dois núcleos, enquanto podem ser facilmente encontrados processadores de quatro ou mais núcleos em computadores pessoais[②].

Assim, existem estudos de métodos baseados nos de Runge-Kutta que fazem uso de processamento paralelo para o cálculo de cada passo. Alguns artigos que tratam do assunto são:

- Claus Bendtsen. "Highly Stable Parallel Runge-Kutta Methods". Inglês. Em: *Applied Numerical Mathematics* 21(1) (1996), pp. 1–8
- Clint Dawson et al. "A Parallel Local Timestepping Runge–Kutta Discontinuous Galerkin Method With Applications to Coastal Ocean Modeling". Inglês. Em: *Computer Methods in Applied Mechanics and Engineering* 259 (2013), pp. 154–165
- Matthias Korch e Thomas Rauber. "Parallel Low-Storage Runge-Kutta Solvers for ODE Systems with Limited Access Distance". Inglês. Em: *International Journal of High Performance Computing Applications* 25(2) (maio de 2011), pp. 236–255

5.6 Limitações do Método

Apesar de amplamente utilizado e do poder computacional, os métodos de Runge-Kutta exigem um tratamento anterior das equações desejadas, pois esses requerem que os sistemas sejam compostos apenas por equações de primeiro grau; além disso, podem requerer

[①] Por exemplo, utilizando a instrução `parfor loopvar = initval:endval, statements, end`, no MATLAB.

[②] Por exemplo, alguns modelos dos populares processadores Core i5 (especificamente, os modelos i5-750 e i5-750s) e Core i7 (todos, exceto o i7-980X), apresentam quatro núcleos de processamento; o modelo i7-980X, apresenta seis núcleos.

mais avaliações das funções por passo do que outros métodos o que, dependendo da complexidade das funções, pode reduzir sua eficiência.

Contudo, a alta flexibilidade do passo dos métodos de Runge-Kutta possibilita a solução de problemas de forma mais eficiente, ocupando menos espaço em memória do que, por exemplo, os médodos Adams[11].

Capítulo 6

Implementação Em MATLAB

6.1 Implementação da Função de Movimento

Para a implementação do Runge-Kutta de 4^a ordem, primeiro precisa-se definir o sistema de equações de movimento do problema a ser analisado. Assim, considerando o sistema da figura 6.1.1, o sistema de equações que representa seu movimento é:

$$\begin{cases} m_1\ddot{x}_1 + (c_{l1} + c_{l2})\,\dot{x}_1 - c_{l2}\dot{x}_2 + (k_{l1} + k_{l2})\,x_1 - k_{l2}x_2 = F_1(t) \\ m_2\ddot{x}_2 - c_{l2}\dot{x}_1 + c_{l2}\dot{x}_2 - k_{l2}x_1 + k_{l2}x_2 = F_2(t) \end{cases} \tag{6.1.1}$$

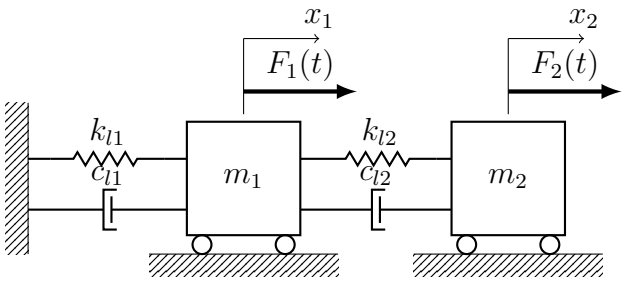

Figura 6.1.1: Diagrama do sistema massa-mola-amortecedor para 2 GDLs.

A implementação do Runge-Kutta espera que `mov-eq.m` retorne um vetor que represente o sistema de equações; além disso, o método de Runge-Kutta espera que as equações diferenciais sejam de primeira ordem. Contudo, é possível transformar uma EDO de ordem 2 em duas EDOs de ordem 1, aplicando uma transformação de variáveis. Assim, fazendo:

$$\begin{aligned} x_1 &= u_1 & x_2 &= u_2 \\ \dot{x}_1 &= \dot{u}_1 = v_1 & \dot{x}_2 &= \dot{u}_2 = v_2 \\ \ddot{x}_1 &= \ddot{u}_1 = \dot{v}_1 & \ddot{x}_2 &= \ddot{u}_2 = \dot{v}_2 \end{aligned}$$

o sistema (6.1.1) se torna:

$$\begin{cases} \dot{u}_1 = v_1 \\ m_1\dot{v}_1 + (c_{l1} + c_{l2})\, v_1 - c_{l2}v_2 + (k_{l1} + k_{l2})\, u_1 - k_{l2}u_2 = F_1(t) \\ \dot{u}_2 = v_2 \\ m_2\dot{v}_2 - c_{l2}v_1 + c_{l2}v_2 - k_{l2}u_1 + k_{l2}u_2 = F_2(t) \end{cases} \tag{6.1.2}$$

Rearranjando as equações, reescreve-se o sistema como:

$$\begin{cases} \dot{u}_1 = v_1 \\ \dot{v}_1 = \frac{1}{m_1}\left[-(c_{l1} + c_{l2})v_1 + c_{l2}v_2 - (k_{l1} + k_{l2})u_1 + k_{l2}u_2 + F_1(t)\right] \\ \dot{u}_2 = v_2 \\ \dot{v}_2 = \frac{1}{m_2}\left[c_{l2}v_1 - c_{l2}v_2 + k_{l2}u_1 - k_{l2}u_2 + F_2(t)\right] \end{cases} \tag{6.1.3}$$

Contudo, esse sistema deve ser traduzido para o MATLAB no arquivo `mov-eq.m`; para isso, cria-se uma função que receba os valores de t, u_1, u_2, v_1 e v_2 e retorne o vetor com os pontos calculados. Por conveniência, os valores das constantes m_1, m_2, c_{l1}, c_{l2}, k_{l1} e k_{l2} serão definidos dentro da função.

Ainda, para que se possa aplicar o método, é necessária a definição dos forçamentos $F_1(t)$ e $F_2(t)$. Portanto, considerando forçamentos harmônicos:

$$F_1(t) = F_1 \cdot \cos(\omega_1 t)$$
$$F_2(t) = F_2 \cdot \cos(\omega_2 t)$$

onde $F_{1,2}$ representa a amplitude dos forçamentos, e $\omega_{1,2}$ suas frequências. Assim, pode-se reescrever o sistema (6.1.3) como:

$$\begin{cases} \dot{u}_1 = v_1 \\ \dot{v}_1 = \frac{1}{m_1}\left[-(c_{l1} + c_{l2})v_1 + c_{l2}v_2 - (k_{l1} + k_{l2})u_1 + k_{l2}u_2 + F_1 \cdot \cos(\omega_1 t)\right] \\ \dot{u}_2 = v_2 \\ \dot{v}_2 = \frac{1}{m_2}\left[c_{l2}v_1 - c_{l2}v_2 + k_{l2}u_1 - k_{l2}u_2 + F_2 \cdot \cos(\omega_2 t)\right] \end{cases} \tag{6.1.4}$$

Para utilizar esse arquivo com o comando `ode45`, deve-se modificar a função definida para que tenha como argumentos, apenas, o intervalo de tempo t e um vetor $\underline{y}$; assim,

assumindo a mudança de variáveis

$$\begin{cases} u_1 = \underline{y}[1] \\ v_1 = \underline{y}[2] \\ u_2 = \underline{y}[3] \\ v_2 = \underline{y}[4] \end{cases} \tag{6.1.5}$$

pode-se reescrever o sistema de equações de movimento como:

$$\begin{cases} \dot{u}_1 = \underline{y}[2] \\ \dot{v}_1 = \frac{1}{m_1}\left[-(c_{l1}+c_{l2})\underline{y}[2] + c_{l2}\underline{y}[4] - (k_{l1}+k_{l2})\underline{y}[1] + k_{l2}\underline{y}[3] + F_1 \cdot \cos(\omega_1 t)\right] \\ \dot{u}_2 = \underline{y}[4] \\ \dot{v}_2 = \frac{1}{m_2}\left[c_{l2}\underline{y}[2] - c_{l2}\underline{y}[4] + k_{l2}\underline{y}[1] - k_{l2}\underline{y}[3] + F_2 \cdot \cos(\omega_2 t)\right] \end{cases} \tag{6.1.6}$$

Para definir essa função no arquivo do MATLAB, faz-se necessária a definição das constantes m_i, k_{li}, c_{li}, F_i e ω_i $(i = 1, 2)$. Uma forma de se fazer isso seria[①]:

Algoritmo 6.1.1: mov_eq_vec.m

```
function mov = mov_eq_vec(t,y)
    F1 = 1;
    F2 = 1.2;
    cl1 = 0.5;
    cl2 = 0.7;
    kl1 = 0.2;
    kl2 = 0.3;
    m1 = 0.2;
    m2 = 0.2;
    omega1 = 2;
    omega2 = 3;

    mov = [ y(2)
            1/m1*(- (cl1 + cl2)*y(2) + cl2*y(4) - (kl1 + kl2)*y(1) + kl2*y(3) ...
                  + F1*cos(omega1*t))
            y(4)
            1/m2*(cl2*y(2) - cl2*y(4) + kl2*y(1) - kl2*y(3) + F2*cos(omega2*t)) ];
end
```

Essa definição pode ser passada para as funções residentes do MATLAB, como será visto a seguir.

[①]Também poderia-se definir esses valores como variáveis globais, ou substituí-los diretamente no vetor `mov`.

6.2 Sem a Utilização de Funções Residentes

Considerando uma EDO descrita no arquivo `mov_eq_vec.m`, conforme deduzida na seção anterior:

$$f = f(t, \underline{y}) \tag{6.2.1}$$

pode-se implementar o método de Runge-Kutta de quarta ordem no MATLAB como um *script* que, dados:

- os intervalos de tempo inicial e final desejados, `t_ini` e `t_end`, respectivamente;
- o intervalo entre cada instante de tempo, `dt`;
- as condições iniciais de deslocamento das duas massas, `y_ini1` e `y_ini2`;
- as condições iniciais de velocidade das massas, `yp_ini1` e `yp_ini2`;

com os dados iniciais do problema, prepara-se o estado inicial das variáveis a serem calculadas:

- `ti`: vetor dos tempos utilizados no cálculo;
- `yi`: matriz dos vetores de deslocamento e velocidade das duas massas;

Em seguida, no loop `while`, os últimos valores calculados são resgatados com o comando `vec(end)` e os novos são gerados com o cálculo dos coeficientes e dos novos pontos. Esses novos valores são, então, alocados no final dos vetores a serem retornados do cálculo como uma nova coluna. Para que o comportamento desse script seja parecido com o dos comandos do MATLAB, é necessário transpor o vetor dos resultados ao final da execução do programa.

Por fim, resta apenas gerar os gráficos desejados com os resultados obtidos, no caso, o deslocamento da massa m_1 em relação ao tempo.

Algoritmo 6.2.2: rk4.m

```
clear all; clc; close all;

dt      = 1e-2;
t_ini   = 0;
t_end   = 30;
y_ini1  = 1;
y_ini2  = 2;
yp_ini1 = 1;
yp_ini2 = 0;

```

```
11 ti = [t_ini];
12 yi = [y_ini1; y_ini2; yp_ini1; yp_ini2];
13
14 while ti(end) <= t_end
15     tn = ti(end);
16     yn = yi(:,end);
17
18     k_1 = dt .* mov_eq_vec(tn        , yn          );
19     k_2 = dt .* mov_eq_vec(tn .+ dt/2, yn .+ k_1./2 );
20     k_3 = dt .* mov_eq_vec(tn .+ dt/2, yn .+ k_2./2 );
21     k_4 = dt .* mov_eq_vec(tn .+ dt  , yn .+ k_3    );
22
23     yn = yn .+ (k_1 .+ 2.*k_2 .+ 2.*k_3 .+ k_4)./6;
24
25     ti = [ti, tn+dt];
26     yi = [yi, yn];
27 end
28
29 yi = transpose(yi);
30
31 plot(ti, yi(:,1));
32 xlabel 'Tempo'
33 ylabel 'Deslocamento da massa m_1'
```

6.3 Fazendo Uso de Funções Residentes no MATLAB

Da mesma forma que anteriormente feito, primeiro define-se as condições iniciais do problema através das variáveis `y_inii` e `yp_inii` ($i = 1, 2$ para o caso apresentado) e define-se o intervalo de tempo desejado através das variáveis `t_ini` e `t_end`. Nesse caso, não é necessário a definição do passo, pois os métodos do MATLAB o adaptam segundo a necessidade, e a passagem do intervalo de tempo desejado e das condições iniciais do problema são feitas através de vetores.

Assim, a solução de um sistema de EDOs pode ser calculada com um *script* do tipo:

Algoritmo 6.3.3: rk45.m

```
1 clear all; clc; close all;
2
3 t_ini   = 0;
4 t_end   = 30;
5 y_ini1  = 1;
6 y_ini2  = 2;
7 yp_ini1 = 1;
8 yp_ini2 = 0;
9
```

```
10 [t, ya] = ode45(@mov_eq_vec,[t_ini, t_end],[y_ini1, y_ini2, yp_ini1, yp_ini2]);
11
12 plot(t, ya(:,1));
13 xlabel 'Tempo'
14 ylabel 'Deslocamento da massa m_1'
```

6.4 Comparação

As duas possibilidades de solução apresentadas anteriormente podem ser utilizadas para os problemas apresentados por esse estudo, ainda que exista uma variação significativa em seus resultados.

Contudo, numa simples comparação, pode-se perceber que, para ambos os casos, o problema foi satisfatoriamente resolvido, conforme apresentado na Figura 6.4.1.

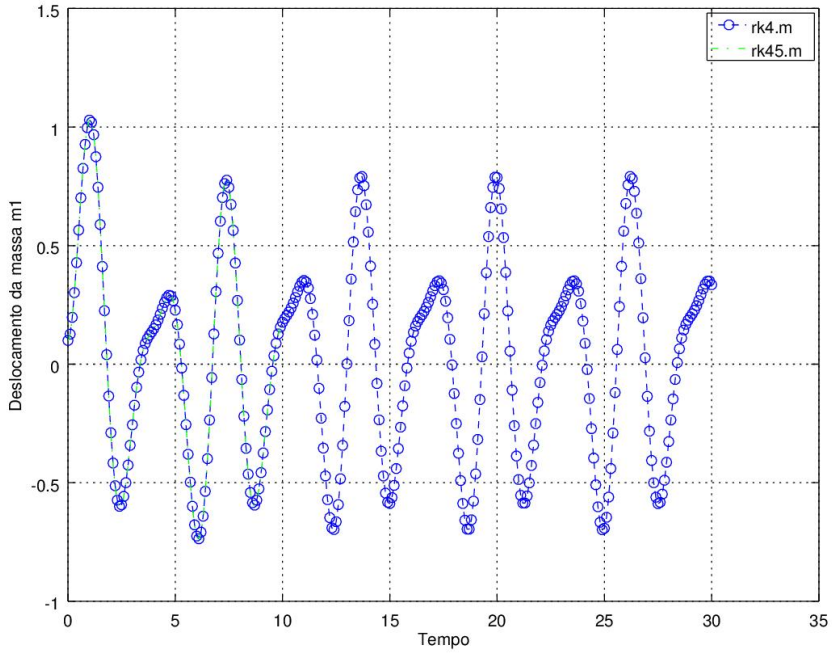

Figura 6.4.1: Comparação entre a implementação do método de Runge-Kutta apresentado e o comando `ode45`.

Parte III

Projeto da Fundação de uma Máquina de Estampagem

Capítulo 7

Problema Proposto

7.1 Descrição do Sistema

Figura 7.1.1: Processo de estampagem de uma folha de alumínio para a fabricação de uma porta de carro.
Fonte: Programa How It's Made.

O processo de estampagem profunda tem por propósito fazer com que uma chapa plana tome a forma de uma matriz sob a imposição de um forçamento. A máquina que realiza esse processo possui cerca de 15 000 kg e realiza um ciclo de estampagem, aproximadamente, a cada dois segundos ($f = 0.5\,\mathrm{Hz}$). Tem-se como objetivo a análise das vibrações causadas pela força de estampagem, e pretende-se projetar a instalação desse equipamento em dois casos característicos para equipamentos industriais:

1. a máquina é montada diretamente ao solo;

2. a máquina é montada sobre uma fundação instalada no solo.

No primeiro caso, procura-se determinar quais as melhores características que um solo deveria ter para que essa instalação fosse possível; no segundo, além da análise das propriedades do solo, procura-se projetar uma fundação de massa ideal para o funcionamento do sistema.

Como não se procura analisar uma máquina de uma linha de produção específica, os dados necessários foram considerados a partir de informações coletadas na literatura (como no caso da frequência de operação da máquina), ou de casos perto do limite (como para o fator de correção m)[12]. Assim, espera-se que os resultados encontrados reflitam uma solução quase ideal para o caso estudado, porém ainda representem uma solução realística.

7.2 Parâmetros do Projeto

As normas ABNT NBR 10082/1987 e Petrobras N-1848 definem o parâmetro velocidade eficaz, v_{ef}, como:

$$v_{ef} = \sqrt{\frac{1}{T} \cdot \int_0^T v^2(t) \mathrm{d}t} \tag{7.2.1}$$

Considerando-se V_i como as amplitudes das velocidades de vibração medidas em diversos pontos do equipamento, para o caso em que V_i são conhecidas, a velocidade eficaz pode ser caracterizada como:

$$v_{ef} = \sqrt{\frac{1}{2} \cdot \left(\sum_{i=1}^{n} V_i^2 \right)} \tag{7.2.2}$$

Também são definidas as seguintes classificações para os equipamentos:

- classe I: partes individuais de motores e máquinas, integralmente conectadas com a máquina completa na sua condição de operação normal (motores elétricos de produção até 15 KW são exemplos típicos de máquinas nesta categoria);
- classe II: máquinas de tamanho médio, (tipicamente motores elétricos de 15 KW até 75 KW de potência sem fundações especiais, motores ou máquinas montados rigidamente até 300 KW) sobre fundações especiais;
- classe III: máquinas motrizes grandes e outras máquinas grandes com massas rotativas montadas sobre fundações rígidas e pesadas, que são relativamente rígidas na direção de medição de vibração;
- classe IV: máquinas motrizes grandes e outras máquinas grandes com massas rotativas, montadas sobre fundações que são relativamente flexíveis na direção de medição

de vibração (por exemplo, conjunto de turbogeradores, especialmente aqueles montados sobre estruturas leves);

- classe V: máquinas e sistemas acionadores mecânicos com forças de inércia não-balanceáveis (devido às partes alternativas), montados sobre fundações que são relativamente rígidas na direção da medição de vibração;

- classe VI: máquinas e sistemas acionadores mecânicos com forças de inércia não-balanceáveis (devido às partes alternativas), montados sobre fundações que são relativamente flexíveis na direção de medições de vibração; máquinas com massas rotativas frouxamente acopladas, tais como eixos batedores em moinho; máquinas, como centrífugas, com desbalanceamentos variáveis capazes de operação como unidades próprias sem componentes de conexão; peneiras vibratórias, máquinas de ensaios dinâmicos de fadiga e excitadores de vibração usados em processos industriais.

Com isso, a Tabela 7.2.1 classifica as faixas de severidade de vibração.

Tabela 7.2.1: Faixas de severidade de vibração, classificadas na norma ABNT NBR 10082/1987.

Faixa de classificação	Valor eficaz da velocidade de vibração (mm/s)	
	acima de	até
0.11	0.071	0.112
0.18	0.112	0.18
0.28	0.18	0.28
0.45	0.28	0.45
0.71	0.45	0.71
1.12	0.71	1.12
1.8	1.12	1.8
2.8	1.8	2.8
4.5	2.8	4.5
7.1	4.5	7.1
11.2	7.1	11.2
18	11.2	18
28	18	28
45	28	45
71	45	71

Essa serve como referência para a criação da Tabela 7.2.2, que define a avaliação de qualidade para as diferentes classes de máquinas.

Tabela 7.2.2: Avaliação da qualidade para diferentes classes de máquina segundo a norma ABNT NBR 10082/1987.

<table>
<tr><th colspan="2">Faixas de severidade de vibração</th><th colspan="4">Exemplos de avaliação de qualidade para classes diferentes de máquinas</th></tr>
<tr><th>Faixa</th><th>Velocidade efetiva (mm/s) nos limites da faixa</th><th>Classe I</th><th>Classe II</th><th>Classe III</th><th>Classe IV</th></tr>
<tr><td>0.28</td><td>0.28</td><td rowspan="3">A</td><td rowspan="3">A</td><td rowspan="4">A</td><td rowspan="5">A</td></tr>
<tr><td>0.45</td><td>0.45</td></tr>
<tr><td>0.71</td><td>0.71</td></tr>
<tr><td>1.12</td><td>1.12</td><td rowspan="2">B</td></tr>
<tr><td>1.8</td><td>1.8</td><td rowspan="2">B</td></tr>
<tr><td>2.8</td><td>2.8</td><td rowspan="2">C</td><td rowspan="2">B</td></tr>
<tr><td>4.5</td><td>4.5</td><td>C</td><td>B</td></tr>
</table>

As classificações da Tabela 7.2.2 são definidas na norma Petrobras N-1848 como:

- A: bom;
- B: satisfatório;
- C: pouco satisfatório;
- D: ruim ou não satisfatório.

Para esse estudo, pretende-se fazer com que o equipamento enquadre-se na classificação "A", com velocidade efetiva $v_{ef} \leqslant 0.28$. Para isso, define-se que a velocidade efetiva seja:

$$v_{ef} = \sqrt{\frac{1}{2}\, v^2} \tag{7.2.3}$$

onde v corresponde à máxima amplitude de velocidade do equipamento (no caso do sistema montado diretamanete ao chão, $v = \dot{y}$; no caso do sistema montado sobre uma fundação em bloco, $v = \dot{y}_1$).

7.3 Análise da Amplitude do Forçamento Aplicado

A força necessária para o processo de embutimento depende de vários fatores, o que torna sua determinação complexa. Alguns desses são:

- tipo de material;
- espessura da chapa;

- profundidade do embutimento;

- lubrificação.

Porém, sabe-se que a força de embutimento, F_{emb}, deve ser menor do que a necessária para cortar a mesma chapa, F_c. Assim, sendo m um fator de correção, tal que $0 \leqslant m \leqslant 1$, pode-se definir a força de embutimento como[13]:

$$F_{emb} = m \cdot F_c \tag{7.3.1}$$

O fator de correção m é tabelado, seguindo a relação dos diâmetros da chapa, d, e do desenvolvimento, D. Alguns dos valores característicos estão dispostos na Tabela 7.3.1.

Tabela 7.3.1: Valores característicos do fator de correção m

Relação entre os diâmetros da peça e do desenvolvimento, $^{d}/_{D}$	Fator de correção, m, a ser aplicado
0.550	1.00
0.575	0.93
0.600	0.86
0.650	0.72
0.700	0.60
0.750	0.50
0.800	0.40

Para peças de seção circular, a geometria do desenvolvimento é um círculo; caso contrário, essa determinação não é simples, podendo exigir cálculos computacionais ou dados experimentais.

Para efeitos desse estudo, assume-se que $m = 0.93$ seja um valor razoável. Isso quer dizer que a máquina estaria trabalhando perto do limite de cortar o material, que corresponderia à maior força que ela poderia realizar sobre a chapa[14]. Essa hipótese pode levar à supervalorização do forçamento aplicado, uma vez que está muito próxima do caso limite, e estudos mais detalhados da máquina seriam necessários para evitar investimentos desnecessários.

A força de corte para uma chapa depende da sua espessura (h, dada em mm), do perímetro do corte (p, dado em mm) e da tensão de ruptura ao cisalhamento do material (K_s, dado em N/mm^2). Considerando isso, a força de corte F_c, dada em N, é definida como:

$$F_c = h \cdot p \cdot K_s \tag{7.3.2}$$

Assim, a força aplicada pela máquina sobre a chapa, F_{emb}, é dada por:

$$F_{emb} = 0.93 \cdot F_c \tag{7.3.3}$$
$$= 0.93 \cdot h \cdot p \cdot K_s \tag{7.3.4}$$

A máquina de estampagem em questão trabalha na produção de portas de carro, que, geralmente, são costruídas em aço SAE 1020 laminados a quente[13]; para a obtenção dos dados faltantes, fez-se três medidas experimentais em alguns modelos distintos de carro[①], e os resultados obtidos foram:

- Celta, ano 2013, modelo de quatro portas:

  média das medidas realizadas: 115x110x60x100 (± 5) mm

  $p = (385 \pm 20)\,\text{mm}$

  espessura das chapas: $(1.5 \pm 0.3)\,\text{mm}$

- Logus, ano 1995, modelo de duas portas:

  média das medidas realizadas: 140x120x120x70 (± 5) mm

  $p = (450 \pm 20)\,\text{mm}$

  espessura das chapas: $(1.7 \pm 0.3)\,\text{mm}$

Assim, uma boa aproximação para o estudo seria uma média desses valores; portanto, define-se o valor do perímetro como $p = (417 \pm 20)\,\text{mm}$ e a espessura das chapas como $h = (1.6 \pm 0.3)\,\text{mm}$.

Com esses dados, e sabendo-se que, para o aço 1020, $K_s = 125\,\text{MPa}$[13], define-se a magnitude da força de embutimento como:

$$F_{emb} = 0.93 \cdot h \cdot p \cdot K_s \tag{7.3.5}$$
$$= 0.93 \cdot 1.6\,\text{mm} \cdot 417\,\text{mm} \cdot 125\,\text{MPa} \tag{7.3.6}$$
$$= 77\,562.0\,\text{N} \tag{7.3.7}$$
$$= 77.562 \cdot 10^3\,\text{N} \tag{7.3.8}$$
$$= 77.562\,\text{kN} \tag{7.3.9}$$

7.4 Análise da Forma do Forçamento

Durante os exemplos dos estudos de casos dos capítulos anteriores, foi considerado um forçamento harmônico seno ou co-senoidal, com uma forma parecida com a representada na Figura 7.4.1.

[①] Para os modelos de quatro portas, considerou-se somente as dimensões das maiores.

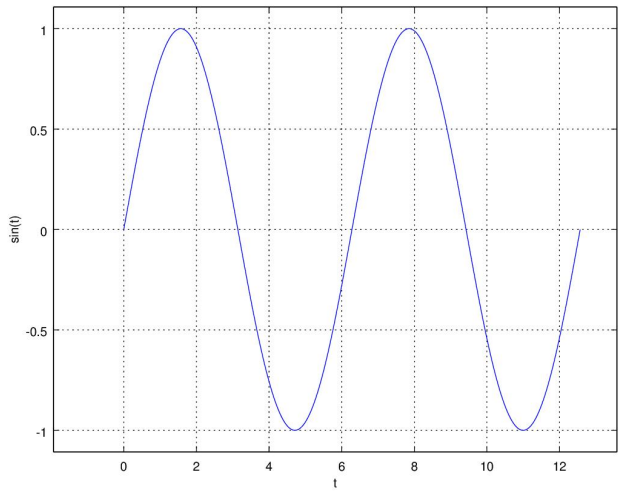

Figura 7.4.1: Forma da onda que representa um forçamento senoidal.

Contudo, para o caso estudado, o forçamento é aplicado rapidamente à placa — o que pode ser representado, matematicamente, na forma de uma onda quadrada. No MATLAB esse tipo de onda pode ser definida com o comando `square`, como demonstra a Figura 7.4.2.

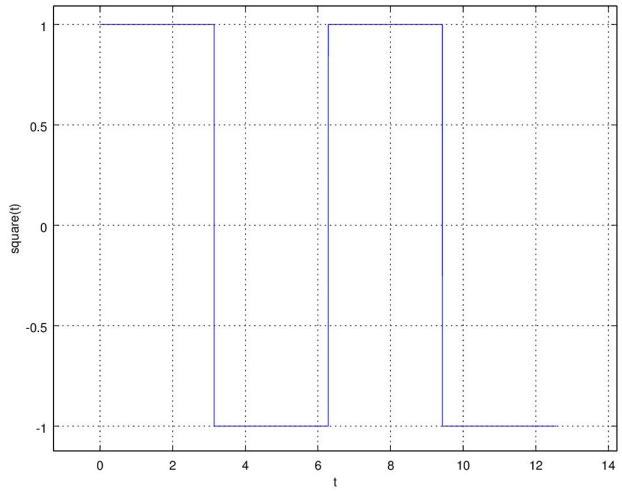

Figura 7.4.2: Forma de um forçamento representado por uma onda quadrada.

Porém, a máquina realiza esse forçamento apenas na direção inferior, enquanto o comando gera uma onda tal que $-1 \leqslant square(t) \leqslant 1$. Usando a indexação lógica (`y(y>0) = 0`), pode-se modificar esses valores e obter-se a Figura 7.4.3, onde $-1 \leqslant square(t) \leqslant 0$.

Da seção anterior, tem-se a definição da magnitude do forçamento. Assim, basta multiplicar cada ponto da onda representada na Figura 7.4.3 por esse escalar para obter-se a onda que representa o forçamento desejado. Por fim, resta definir a frequência dentro

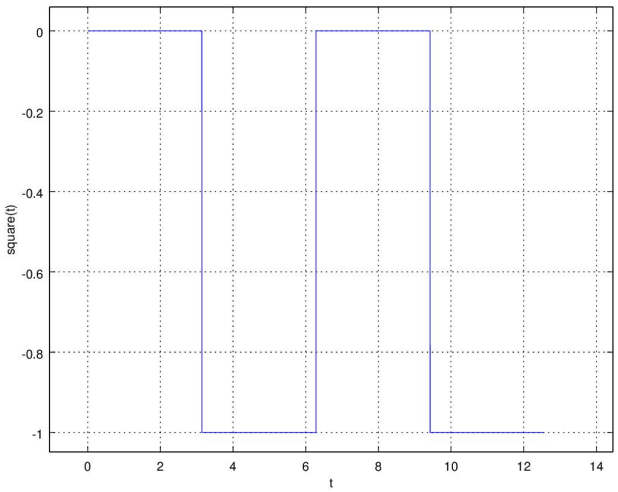

Figura 7.4.3: Forma da onda que representa o forçamento da máquina.

do comando `square`; assim, com os seguintes passos:

Algoritmo 7.4.1: Exemplo de uma Onda Quadrada

```
octave:1> pkg load signal
octave:2> t = 0:0.001:4*pi;
octave:3> freq = 0.5;
octave:4> mag = 77.562;
octave:5> y = square(2*pi*freq*t)*mag;
octave:6> y(y>0) = 0;
octave:7> plot(t, y);
octave:8> xlabel 't';
octave:9> ylabel 'square(t)';
```

obtem-se, como desejado, a Figura 7.4.4, que representa o forçamento aplicado pela máquina, devidamente corrigido para as frequência e magnitude reais do problema.

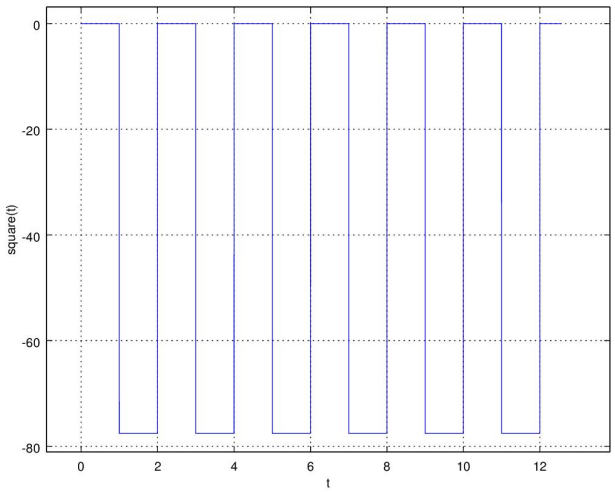

Figura 7.4.4: Forçamento exercido pela máquina.

7.5 Sistema Montado Diretamente ao Chão

Assumindo, como uma primeira aproximação ao problema, que a máquina esteja montada diretamente ao chão, conforme ilustrado na figura 7.5.1.

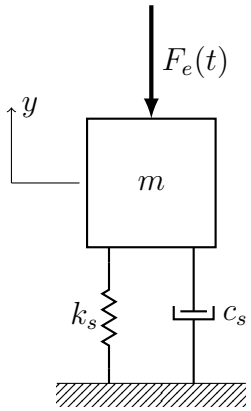

Figura 7.5.1: Diagrama da máquina montada ao chão.

Nesse caso, a força peso se anula com a pré-tensão na mola e no amortecedor[2], reduzindo o sistema ao massa-mola-amortecedor anteriormente estudado. A reação do solo depende de suas propriedades — o que exige uma análise de solo, de responsabilidade do engenheiro civil; porém, é possível determinar propriedades equivalentes, modelando o solo como uma mola de constante k_s e um amortecedor de constante c_s.

Assim, o movimento do sistema pode ser representado pela equação deduzida anteriormente:

$$m\ddot{y} + c_s\dot{y} + k_s y = F_e(t) \tag{7.5.1}$$

Considerando a transformação de coordenadas:

$$\begin{aligned} y &= u \\ \dot{y} &= \dot{u} = v \\ \ddot{y} &= \ddot{u} = \dot{v} \end{aligned}$$

A equação 7.5.1 pode ser reescrita como:

$$\begin{cases} \dot{u} = v \\ \dot{v} = \frac{1}{m}\left(F_e(t) - c_s v - k_s u\right) \end{cases} \tag{7.5.2}$$

Para a utilização com o MATLAB, é necessária uma nova mudança de variáveis:

$$\begin{aligned} u &= \underline{y}[1] \\ v &= \underline{y}[2] \end{aligned}$$

que leva o sistema anterior a ser reescrito como:

$$\begin{cases} \dot{u} = \underline{y}[2] \\ \dot{v} = \frac{1}{m}\left(F_e(t) - c_s \underline{y}[2] - k_s \underline{y}[1]\right) \end{cases} \tag{7.5.3}$$

Para facilitar os cálculos, os dados de m, c_s, k_s e F_e foram divididos por 10^9 (correspondente à ordem de grandeza do forçamento, k), de modo a reduzir a magnitude dos dados do problema.

Transcreve-se, então, a equação do problema na sintaxe do MATLAB como:

Algoritmo 7.5.2: mov_eq_machine_on_floor.m

```
function mov = mov_eq_machine_on_floor(t,y)
    Fe       = square(2*pi*0.5*t)*77.562; % * 1e3 [N]
    Fe(Fe>0) = 0;                         % -1 <= square(t) <= 0
    m        = 1.5e1;                     % * 1e3 [kg]
    k        = 1000;                        % * 1e3 [N/mm]
    c        = 150;                       % * 1e3 [N*s/mm]

    mov = [ y(2),
            1/m*(Fe - c*y(2) - k*y(1)) ];
end
```

onde os dados do problema podem facilmente ser manipulados.

7.5.1 Sistema Não-Amortecido

Como uma primeira aproximação, pretende-se determinar uma faixa de valores aceitável para a constante da mola do sistema. Para facilitar essa determinação, desconsidera-se

qualquer amortecimento do sistema.

Assim, observou-se os valores listados nas tabelas abaixo, considerando os 100 s iniciais da simulação.

Tabela 7.5.1: Resultados encontrados para o sistema montado ao chão utilizando a implementação do método de Runge-Kutta de quarta ordem.

Constante da mola (10^3 N/mm)	Velocidade efetiva (mm/s)
1.00(0)	8.05(7)
10.00(0)	3.39(0)
50.00(0)	2.93(9)
300.00(0)	1.31(9)
500.00(0)	0.66(9)
700.00(0)	0.55(0)
1000.00(0)	0.74(0)
3000.00(0)	0.36(7)
6000.00(0)	0.21(3)

Tabela 7.5.2: Resultados encontrados para o sistema montado ao chão utilizando o comando `ode23s`.

Constante da mola (10^3 N/mm)	Velocidade efetiva (mm/s)
1.00(0)	14.26(7)
10.00(0)	3.49(8)
50.00(0)	2.99(6)
300.00(0)	1.39(7)
500.00(0)	0.82(2)
700.00(0)	0.59(1)
1000.00(0)	0.88(8)
3000.00(0)	0.45(2)
6000.00(0)	0.22(2)

Observa-se que, dos valores analisados, $k_s = 6000 \times 10^3$ N/mm é o que melhor se aproxima do parâmetro de projeto utilizando ambas as ferramentas. Porém, considerando-se que, em sistema reais, existem fatores de amortecimento, escolhe-se $k_s = 1000 \times 10^3$ N/mm e pretende-se encontrar um valor adequado para o amortecimento que faça o sitema atender aos parâmetros do projeto.

7.5.2 Sistema Amortecido

Mantendo-se o valor da constante de amortecimento em $k_s = 1000 \times 10^3\,\mathrm{N/mm}$, tem-se as tabelas abaixo.

Tabela 7.5.3: Resultados encontrados para o sistema montado ao chão ao adicionar-se um fator de amortecimento utilizando a implementação do método de Runge-Kutta de quarta ordem.

Constante de amortecimento (10^3 Ns/mm)	Velocidade efetiva (mm/s)
1.00(0)	0.70(5)
10.00(0)	0.57(4)
100.00(0)	0.26(8)
150.00(0)	0.22(1)

Tabela 7.5.4: Resultados encontrados para o sistema montado ao chão ao adicionar-se um fator de amortecimento utilizando o comando `ode23s`.

Constante de amortecimento (10^3 Ns/mm)	Velocidade efetiva (mm/s)
1.00(0)	0.64(9)
10.00(0)	0.53(0)
100.00(0)	0.26(8)
150.00(0)	0.22(4)

Verifica-se que qualquer valor de amortecimento acima de $c_s = 150 \times 10^3\,\mathrm{Ns/mm}$ satisfaz o requisito apresentado, para ambas as ferramentas utilizadas.

7.6 Sistema Montado sobre uma Fundação em Bloco

Nesse caso, monta-se o equipamento sobre uma fundação maciça de concreto, conforme ilustrado na figura 7.6.1.

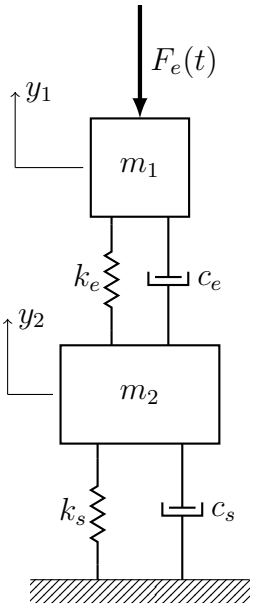

Figura 7.6.1: Diagrama da máquina montada sobre uma fundação em bloco.

Novamente, a pré-tensão nas molas e nos amortecedores anula a força peso em ambos os corpos[2], reduzindo o problema ao massa-mola-amortecedor de 2 GdLs anteriormente estudado. Portanto, o movimento do sistema pode ser representado pela equação anteriormente deduzida:

$$\mathbf{M}\underline{\ddot{y}} + \mathbf{C_l}\underline{\dot{y}} + \mathbf{K_l}\underline{y} = \underline{F}_e(t) \tag{7.6.1}$$

Além disso, com analogia ao caso anterior, pode-se considerar que o bloco seria o que anteriormente foi considerado o solo; assim:

$$\begin{cases} k_e = 1000 \times 10^3\,\mathrm{N/mm} \\ c_e = 150 \times 10^3\,\mathrm{Ns/mm} \end{cases} \tag{7.6.2}$$

Deseja-se obter a massa m_2 e as propriedades do solo, k_s e c_s, de modo que os deslocamentos de ambos não ultrapasse 20 mm de amplitude.

A equação de movimento para esse sistema é:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{Bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{Bmatrix} + \begin{bmatrix} c_e & -c_e \\ -c_e & c_s + c_e \end{bmatrix} \begin{Bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{Bmatrix} + \begin{bmatrix} k_e & -k_e \\ -k_e & k_s + k_e \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \begin{Bmatrix} F_e(t) \\ 0 \end{Bmatrix} \tag{7.6.3}$$

A equação anterior pode ser reescrita como:

$$\begin{cases} m_1\ddot{y}_1 + c_e\dot{y}_1 - c_e\dot{y}_2 + k_e y_1 - k_e y_2 = F_e(t) \\ m_2\ddot{y}_2 - c_e\dot{y}_1 + (c_s + c_e)\,\dot{y}_2 - k_e y_1 + (k_s + k_e)\,y_2 = 0 \end{cases} \tag{7.6.4}$$

Aplicando a transformação de variáveis:

$$\begin{aligned} y_1 &= u_1 & y_2 &= u_2 \\ \dot{y}_1 &= \dot{u}_1 = v_1 & \dot{y}_2 &= \dot{u}_2 = v_2 \\ \ddot{y}_1 &= \ddot{u}_1 = \dot{v}_1 & \ddot{y}_2 &= \ddot{u}_2 = \dot{v}_2 \end{aligned}$$

a equação anterior pode ser reescrita como:

$$\begin{cases} \dot{u}_1 = v_1 \\ \dot{v}_1 = \frac{1}{m_1}\left(-c_e v_1 + c_e v_2 - k_e u_1 + k_e u_2 + F_e\right) \\ \dot{u}_2 = v_2 \\ \dot{v}_2 = \frac{1}{m_2}\left(c_e v_1 - (c_s + c_e) v_2 + k_e u_1 - (k_s + k_e) u_2\right) \end{cases} \tag{7.6.5}$$

Para a utilização com o MATLAB, faz-se a seguinte transformação:

$$\begin{aligned} u_1 &= \underline{y}[1] \\ v_1 &= \underline{y}[2] \\ u_2 &= \underline{y}[3] \\ v_2 &= \underline{y}[4] \end{aligned}$$

e reescreve-se o sistema como:

$$\begin{cases} \dot{u}_1 = \underline{y}[2] \\ \dot{v}_1 = \frac{1}{m_1}\left(-c_e \underline{y}[2] + c_e \underline{y}[4] - k_e \underline{y}[1] + k_e \underline{y}[3] + F_e\right) \\ \dot{u}_2 = \underline{y}[4] \\ \dot{v}_2 = \frac{1}{m_2}\left(c_e \underline{y}[2] - (c_s + c_e)\underline{y}[4] + k_e \underline{y}[1] - (k_s + k_e)\underline{y}[3]\right) \end{cases} \tag{7.6.6}$$

Portanto, transcrevendo no arquivo a ser utilizado pelo MATLAB:

Algoritmo 7.6.3: mov_eq_machine_on_block.m

```
function mov = mov_eq_machine_on_block(t,y)
    F_e = square(2*pi*0.5*t)*77.562; % * 1e3 [N]
    F_e(F_e>0) = 0;                  % -1 <= square(t) <= 0
    % machine
    m_1 = 1.5e1;                     % * 1e3 [kg]
    k_e = 1000;                      % * 1e3 [N/mm]
    c_e = 150;                       % * 1e3 [N*s/mm]
    % block
    m_2 = 1.5*m_1;
    k_s = 5000;                      % * 1e3 [N/mm]
    c_s = 100;                       % * 1e3 [N*s/mm]

```

```
13      mov = [ y(2),
14              1/m_1*(- c_e*y(2) + c_e*y(4) - k_e*y(1) + k_e*y(3) + F_e),
15              y(4),
16              1/m_2*(c_e*y(2) - (c_s + c_e)*y(4) + k_e*y(1) - (k_s + k_e)*y(3)) ];
17  end
```

Assumindo que os resultados encontrados anteriormente são válidos para a massa m_1 (bem como para o amortecedor e a mola a ela ligados), mantém-se os valores observados para k_e e c_e, e pretende-se encontrar valores para k_s, c_s e, por fim, m_2 que satisfaçam as características do projeto.

7.6.1 Sistema Não-Amortecido

Analogamente ao caso anterior, observou-se os valores listados nas tabelas abaixo, considerando os 100 s iniciais da simulação.

Tabela 7.6.1: Resultados encontrados para o sistema montado sobre uma fundação em bloco utilizando a implementação do método de Runge-Kutta de quarta ordem.

Constante da mola (10^3 N/mm)	Velocidade efetiva (mm/s)
1.00(0)	5.48(8)
10.00(0)	2.12(4)
100.00(0)	1.49(3)
1000.00(0)	0.45(0)
2000.00(0)	0.36(8)
5000.00(0)	0.30(5)
6000.00(0)	0.29(5)
7000.00(0)	0.28(7)
9000.00(0)	0.27(5)

Tabela 7.6.2: Resultados encontrados para o sistema montado sobre uma fundação em bloco utilizando o comando `ode23s`.

Constante da mola (10^3 N/mm)	Velocidade efetiva (mm/s)
1.00(0)	7.28(9)
10.00(0)	2.41(6)
100.00(0)	1.46(7)
1000.00(0)	0.47(4)
2000.00(0)	0.36(7)
5000.00(0)	0.30(6)
7000.00(0)	0.29(0)
9000.00(0)	0.27(6)

Observa-se que, dos valores analisados, $k_s = 9000 \times 10^3$ N/mm é o que melhor se aproxima do parâmetro de projeto utilizando ambas as ferramentas. Porém, considerando-se que, em sistema reais, existem fatores de amortecimento, escolhe-se $k_s = 5000 \times 10^3$ N/mm e pretende-se encontrar um valor adequado para o amortecimento que faça o sitema atender aos parâmetros do projeto.

7.6.2 Sistema Amortecido

Considerando, agora, que o sitema sofra amortecimento, porém mantendo-se $k_s = 5000 \times 10^3$ N/mm constante ao longo das análises, tem-se:

Tabela 7.6.3: Resultados encontrados para o sistema montado sobre uma fundação em bloco ao adicionar-se um fator de amortecimento utilizando a implementação do método de Runge-Kutta de quarta ordem.

Constante de amortecimento c_s (10^3 Ns/mm)	Velocidade efetiva (mm/s)
1.00(0)	0.30(5)
10.00(0)	0.30(3)
100.00(0)	0.28(9)
300.00(0)	0.27(0)
500.00(0)	0.26(0)
1000.00(0)	0.24(6)

Tabela 7.6.4: Resultados encontrados para o sistema montado sobre uma fundação em bloco ao adicionar-se um fator de amortecimento utilizando o comando `ode23s`.

Constante de amortecimento c_s (10^3 Ns/mm)	Velocidade efetiva (mm/s)
1.00(0)	0.30(6)
10.00(0)	0.30(5)
100.00(0)	0.28(9)
300.00(0)	0.27(1)
500.00(0)	0.28(8)
1000.00(0)	0.27(7)

Observa-se que, dos valores analisados, $c_s = 300 \times 10^3$ Ns/mm é o que melhor se aproxima do parâmetro de projeto utilizando ambas as ferramentas. Porém, considerando-se que, em sistema reais, existem fatores de amortecimento, escolhe-se $c_s = 100 \times 10^3$ Ns/mm e pretende-se encontrar um valor adequado para a massa da fundação que faça o sitema atender aos parâmetros do projeto.

7.6.3 Massa da fundação

Variando-se, agora, a massa da fundação e mantendo-se constantes os demais valores, pretende-se avaliar o impacto que a relação entre as massas m_1 e m_2 tem sobre o sistema.

Tabela 7.6.5: Resultados encontrados para o sistema montado sobre uma fundação em bloco ao alterar-se a relação entre as massas utilizando a implementação do método de Runge-Kutta de quarta ordem

Relação entre a massa da máquina e a da fundação (m_2/m_1)	Velocidade efetiva (mm/s)
1.50(0)	0.28(2)
2.00(0)	0.27(5)
3.00(0)	0.26(3)
5.00(0)	0.24(8)

Tabela 7.6.6: Resultados encontrados para o sistema montado sobre uma fundação em bloco ao alterar-se a relação entre as massas utilizando o comando `ode23s`

Relação entre a massa da máquina e a da fundação (m_2/m_1)	Velocidade efetiva (mm/s)
1.50(0)	0.28(2)
2.00(0)	0.27(5)
3.00(0)	0.26(1)
5.00(0)	0.24(8)

Verifica-se que qualquer valor de amortecimento acima de $m_2/m_1 = 2$ satisfaz o requisito apresentado, para ambas as ferramentas utilizadas.

Capítulo 8

Conclusão

8.1 Sobre o Estudo

Conforme percebe-se, a utilização de ambientes como MATLAB e Octave auxiliam decisões, facilitam os cálculos do projeto e possuem diversas ferramentas genéricas que são aplicáveis a diversos casos. Além disso, conforme anteriormente citato, um conjunto similar de ferramentas pode ser também encontrado para linguagens de programação tradicionais, e diversos outros produtos também estão disponíveis no mercado.

Além disso, diferentes análises com metodologias mais acuradas poderiam levar a outras conclusões com relação às propriedades citadas, e, portanto, seria interessante verificar o efeito de alterações significativas nas propriedades do solo em simulações mais detalhadas.

8.2 Soluções Propostas

8.2.1 Sistema Montado Diretamente ao Chão

A partir das análises feitas, conclui-se que é possível a instalação do equipamento diretamente ao chão, desde que garantidas propriedades equivalentes a $k_s = 1000 \times 10^3\,\mathrm{N/mm}$ e $c_s = 150 \times 10^3\,\mathrm{Ns/mm}$ ou superiores para o solo.

Nesse caso, necessita-se, ainda, verificar se o equipamento não afetará a estrutura da instalação, equipamentos próximos ou gerará ruídos em excesso. Em caso favorável, essa é a instalação mais simples do equipamento e, para o caso estudado, suficientemente adequada; caso contrário, sugere-se o estudo da instalação sobre uma fundação.

8.2.2 Sistema Montado sobre uma Fundação em Bloco

Para essa análise, utilizou-se dos dados escolhidos para o sistema montado diretamente ao chão, considerando-os equivalentes às propriedades da fundação (ou seja, $k_e = 1000 \times 10^3\,\mathrm{N/mm}$

e $c_e = 150 \times 10^3\,\mathrm{Ns/mm}$). Assim, além desses valores, o solo precisaria ter propriedades equivalentes a $k_s = 5000 \times 10^3\,\mathrm{N/mm}$ e $c_s = 100 \times 10^3\,\mathrm{kNs/mm}$, e considerando que a massa total da fundação seja $m_2 = 30\,000\,\mathrm{kg}$, ou ($m_2/m_1 = 2$).

Como pode-se permitir um maior controle de vibrações adicionando corpos auxiliares à fundação, essa opção oferece uma maior flexibilidade com relação à mudança do ciclo de operação, bem como permite uma melhor redução de ruídos emitidos pelo equipamento.

Parte IV

Apêndices e Bibliografia

Apêndice A

Algoritmos

A.1 Movimento do Pêndulo Simples

Esse script foi feito para demonstrar graficamente a solução do sistema do pêndulo simples segundo dados aleatoriamente escolhidos.

Primeiramente são definidas algumas constantes:

- `theta_0`: deslocamento angular inicial do pêndulo;
- `omega_0`: velocidade angular inicial do pêndulo;
- `g`: aceleração da gravidade;

e o intervalo de tempo desejado, que corresponderá ao eixo `x` das figuras.

Em seguida, é definida a função que representa o movimento do pêndulo, `theta`, como uma função anônima do MATLAB, dependente do instante de tempo e de `omega_n`, para que se possa utilizá-la para os dois casos desejados.

Como pretende-se apresentar o mesmo resultado para dois casos diferentes, defini-se duas sub-figuras com o comando `subplot`. Na figura superior, contempla-se o caso em que $L = 1$ e, na inferior, o caso $L = 2$.

Assim, pode-se escrever o script da seguinte forma:

Algoritmo A.1.1: plot-pendulum.m

```
% script to print the exact solution for the pendulum movement
% using two lengths

clear all; clc; close all;

% constants
theta_0 = 0.1;
omega_0 = 0.2;
g = 9.81;
```

```
10
11 % desired time interval (x axis in both subplots)
12 x = 0:0.001:10;
13
14 % movement equation
15 theta = @(t,omega_n) sqrt((theta_0.^2.*omega_n.^2 + omega_0.^2)/omega_n.^2).*...
16     cos(omega_n.*t - atan(omega_0./(omega_n.*theta_0)));
17
18 % dividing the figure into subplots
19 pen_1 = subplot(2,1,1);
20 pen_2 = subplot(2,1,2);
21
22 % superior subplot (L=1)
23 L = 1;
24 omega_n = sqrt(g/L);
25 y = theta(x,omega_n); % y axis
26 plot(pen_1,x,y);
27 title(pen_1,'Pendulo de comprimento L = 1')
28 xlabel(pen_1,'t');
29 ylabel(pen_1,'\theta(t)');
30
31 % inferior subplot (L=2)
32 L = 2;
33 omega_n = sqrt(g/L);
34 y = theta(x,omega_n); % recalculating y axis
35 plot(pen_2,x,y);
36 title(pen_2,'Pendulo de comprimento L = 2')
37 xlabel(pen_2,'t');
38 ylabel(pen_2,'\theta(t)');
```

Como resultado, tem-se a figura 4.7.2, conforme apresentada ao longo do texto.

A.2 Definição de uma Onda Quadrada

Para a definição do forçamento do problema proposto, faz-se necessária a geração de uma onda quadrada que varie entre -1 e 0. No MATLAB, essa definição pode ser feita através da função `square` e, posteriormente, transformando os valores negativos em zeros com a indexação lógica `y>0`. Como exemplo, o script:

Algoritmo A.2.2: square-wave.m

```
1 octave:1> pkg load signal
2 octave:2> t = 0:0.001:4*pi;
3 octave:3> freq = 0.5;
4 octave:4> mag = 77.562;
5 octave:5> y = square(2*pi*freq*t)*mag;
```

```
octave:6> y(y>0) = 0;
octave:7> plot(t, y);
octave:8> xlabel 't';
octave:9> ylabel 'square(t)';
```

gera, como resultado, a figura A.2.1

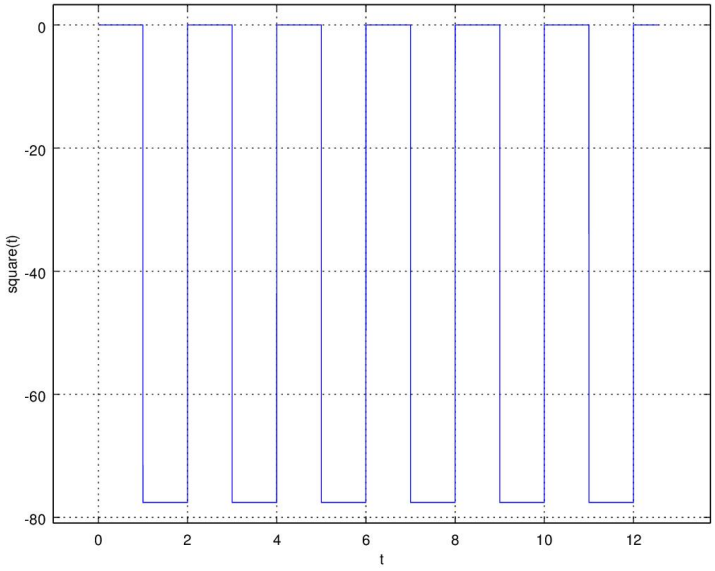

Figura A.2.1: Onda quadrada variando entre os valores 0 e 1.

Apêndice B

Implementações em outras linguagens

A título de ilustração, apresenta-se aqui algumas implementações do méodo de Runge-Kutta de 4^a ordem retirados do banco online Rosetta Code[①] — especificamente, os exemplos da página *Runge-Kutta Method*[②].

Ainda que para um problema diferente do apresentado, a referida página funciona como uma demostração das linguagens e das diferentes formas que podem ser usadas para implemrntar o algoritmo de Runge-Kutta.

Como esse apêndice tem somente o propósito de ilustração, segue-se o exemplo apresentado na página: considerando uma função $y = y(t)$, resolver, implementando o método de Runge-Kutta de 4^a ordem, a equação:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = t \cdot \sqrt{y(t)}$$

sob a condição inicial:

$$y_0 = y(t = 0) = 1$$

Para comparação, a solução analítica fornecida do problema é:

$$y(t) = \frac{1}{16}\left(t^2 + 4\right)^2$$

Para a exemplificação, utilizam-se aqui os códigos exatamente como apresentados no site[③], exluindo-se, apenas, os resultados do programa[④].

[①] `http://www.rosettacode.org`

[②] `http://rosettacode.org/wiki/Runge-Kutta_method`

[③] Embora não às cegas; cada um deles foi testado e verificado.

[④] Tendo em vista que esse apêndice visa, apenas, demonstrar a facilidade de escrever um código em MATLAB; para isso, contudo, foram escolhidas três linguagens não muito convencionais para esse tipo de problema.

B.1 Go

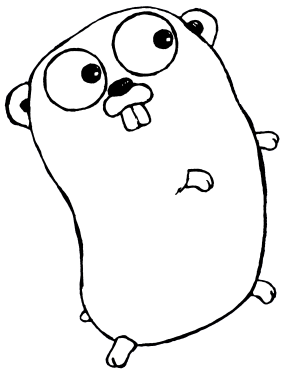

Figura B.1.1: O esquilo, mascote da linguagem Go.
Fonte: Site oficial da linguagem Go.

Segundo Rob Pike, um dos principais engenheiros na Google e co-desenvolvedor da linguagem Go[①], os funcionários da empresa ficavam frustrados com o sistema de desenvolvimento da época, especialmente quando tinham de escrever softwares em larga escala usando C++. O código demorava muito para compilar, e o arquivo final era muito grande; além disso, "muitas das ideias e mudanças de hardware que aconteceram nos últimos 20 anos ainda não tiveram uma chance de influenciar o C++", diz em uma entrevista à O'reilly em junho de 2010[②].

Originalmente desenvolvida dentro da Google, a linguagem Go (também referida como `golang`) atualmente possui uma grande contrubuição da comunidade, uma vez que se tornou *open-source*.

Alguns usos notáveis dessa linguagem:

- a própria Google, em muitos projetos;
- a Dropbox[③], que migrou alguns componentes críticos de Python para Go;
- Docker[④], um conjunto de ferramentas para desenvolvimento de aplicações para Linux.

Algumas bibliografias são:

- Caleb Doxsey. *An Introduction to Programming in Go*. Inglês. 1ª ed. CreateSpace Independent Publishing Platform, set. de 2012. ISBN: 978-14-7835-582-3
- Mark Summerfield. *Programming in Go: Creating Applications for the 21st Century*. Inglês. 1ª ed. Addison-Wesley Professional, maio de 2012. ISBN: 978-03-2177-463-7

①`http://golang.org/`
②`http://radar.oreilly.com/2010/06/does-the-world-need-yet-anothe.html`
③`https://www.dropbox.com/`
④`https://www.docker.com/`

Algoritmo B.1.1: go-rk4.go

```
package main

import (
    "fmt"
    "math"
)

type ypFunc func(t, y float64) float64
type ypStepFunc func(t, y, dt float64) float64

// newRKStep takes a function representing a differential equation
// and returns a function that performs a single step of the forth-order
// Runge-Kutta method.
func newRK4Step(yp ypFunc) ypStepFunc {
    return func(t, y, dt float64) float64 {
        dy1 := dt * yp(t, y)
        dy2 := dt * yp(t+dt/2, y+dy1/2)
        dy3 := dt * yp(t+dt/2, y+dy2/2)
        dy4 := dt * yp(t+dt, y+dy3)
        return y + (dy1+2*(dy2+dy3)+dy4)/6
    }
}

// example differential equation
func yprime(t, y float64) float64 {
    return t * math.Sqrt(y)
}

// exact solution of example
func actual(t float64) float64 {
    t = t*t + 4
    return t * t / 16
}

func main() {
    t0, tFinal := 0, 10 // task specifies times as integers,
    dtPrint := 1        // and to print at whole numbers.
    y0 := 1.            // initial y.
    dtStep := .1        // step value.

    t, y := float64(t0), y0
    ypStep := newRK4Step(yprime)
    for t1 := t0 + dtPrint; t1 <= tFinal; t1 += dtPrint {
        printErr(t, y) // print intermediate result
        for steps := int(float64(dtPrint)/dtStep + .5); steps > 1; steps-- {
```

```
            y = ypStep(t, y, dtStep)
            t += dtStep
        }
        y = ypStep(t, y, float64(t1)-t) // adjust step to integer time
        t = float64(t1)
    }
    printErr(t, y) // print final result
}

func printErr(t, y float64) {
    fmt.Printf("y(%.1f) = %f Error: %e\n", t, y, math.Abs(actual(t)-y))
}
```

B.2 Haskell

Figura B.2.1: Logo da linguagem Haskell.
Fonte: Site oficial da linguagem Haskell.

Após o desenvolvimento da linguagem Miranda, em 1985, cresceu o interesse por linguagens funcionais de avaliação preguiçosa ao ponto de, em torno de dois anos depois, já existirem algumas dezenas de linguagens influenciadas por ela. Durante a conferência FPCA '87 (*Functional Programming Languages and Computer Architecture*) em Portland, Oregon, um consenso entre os participantes decidiu que um comitê deveria ser formado para definir um padrão para esse tipo de linguagem, que deveria servir como base para pesquisas futuras sobre o *design* de linguagens funcionais.

A primeira versão da linguagem Haskell[①] foi definida em 1990; com os esforços do comitê, surgiu uma série de atualizações para a linguagem, até a revisão 1.4. Em 1997, essa série de atualizações culminou no Haskell 98, intencionando uma especificação minimalista e portável da linguagem, em conjunto com uma biblioteca padrão, que serviria para ensino e como base para extensões futuras.

Atualmente, possui duas principais implementações[17]: GHC[②] (*Glasgow Haskell Compiler*) e Hugs[③] (*Haskell User's Gofer System*, que, segundo o site oficial, não está mais em desenvolvimento).

Alguns usos notáveis da linguagem são:

[①] https://www.haskell.org/
[②] https://www.haskell.org/ghc/
[③] https://www.haskell.org/hugs/

- Darks[1], um sistema distribuído de controle de revisão;
- Xmonad[2], um gerenciador de janelas para o *X Window System*.

Algumas referências bibliográficas notáveis são:

- Miran Lipovača. *Learn You a Haskell for Great Good!* Inglês. No Starch Press, abr. de 2011. ISBN: 978-15-9327-283-8
- Alejandro Serrano Mena. *Beginning Haskell: A Project-Based Approach*. Inglês. 1ª ed. Apress, jan. de 2014. ISBN: 978-14-3026-250-3
- Claudio Cesar de Sá e Márcio Ferreira da Silva. *Haskell: Uma Abordagem Prática*. Português. 1ª ed. Novatec, 2006. ISBN: 857-522-095-0

Algoritmo B.2.2: haskell-rk4.hs

```
1  import Data.List
2
3  dv :: Floating a => a -> a -> a
4  dv = (. sqrt). (*)
5
6  fy t = 1/16 * (4+t^2)^2
7
8  rk4 :: (Enum a, Fractional a)=> (a -> a -> a) -> a -> a -> a -> [(a,a)]
9  rk4 fd y0 a h = zip ts $ scanl (flip fc) y0 ts  where
10   ts = [a,h ..]
11   fc t y = sum. (y:). zipWith (*) [1/6,1/3,1/3,1/6]
12     $ scanl (\k f -> h * fd (t+f*h) (y+f*k)) (h * fd t y) [1/2,1/2,1]
13
14 task = mapM_ print
15   $ map (\(x,y)-> (truncate x,y,fy x - y))
16   $ filter (\(x,_) -> 0== mod (truncate $ 10*x) 10)
17   $ take 101 $ rk4 dv 1.0 0 0.1
18
19 main = task
```

B.3 OCaml

Originalmente concebida como *Objective Caml*, atualmente, OCaml[3] é a principal implementação da linguagem Caml, criada em 1996, extendendo-a com o suporte a orientação a objetos.

[1] http://darcs.net/
[2] http://xmonad.org/
[3] https://ocaml.org/

Apesar de ser uma linguagem mais nova, possui uma extensa biblioteca padrão, o que a torna tão útil quanto linguagens mais antigas e bem estabelecidas, como Perl ou Python, além de possuir mecanismos que a tornam aplicável para softwares em larga escala.

Figura B.3.1: Logo da linguagem OCaml.
Fonte: Site oficial da linguagem OCaml.

Alguns usos notáveis da linguagem são:

- o compilador da linguagem de programação Hack[①] e o conjunto de ferramentas e APIs PFFF[②], desenvolvidos pelo Facebook;
- o gerenciador de provas formais Coq[③] que, inclusive, foi utilizado na prova do Teorema do Mapa das Quatro Cores[④].

Algumas referências bibliográficas notáveis são:

- John Whitington. *OCaml from the Very Beginning*. Inglês. Coherent Press, jun. de 2013. ISBN: 978-09-5767-110-2
- Jon D. Harrop. *OCaml for Scientists*. Inglês. 1ª ed. Flying Frog Consultancy Ltd, 2005
- Jason Hickey, Anil Madhavapeddy e Yaron Minsky. *Real World OCaml*. Inglês. 1ª ed. O'Reilly Media, nov. de 2013. ISBN: 978-14-4932-391-2

Algoritmo B.3.3: ocaml-rk4.ml

```
1  let y' t y = t *. sqrt y
2  let exact t = let u = 0.25*.t*.t +. 1.0 in u*.u
3
4  let rk4_step (y,t) h =
5    let k1 = h *. y' t y in
6    let k2 = h *. y' (t +. 0.5*.h) (y +. 0.5*.k1) in
7    let k3 = h *. y' (t +. 0.5*.h) (y +. 0.5*.k2) in
8    let k4 = h *. y' (t +. h) (y +. k3) in
9    (y +. (k1+.k4)/.6.0 +. (k2+.k3)/.3.0, t +. h)
10
11 let rec loop h n (y,t) =
```

①http://hacklang.org/
②https://github.com/facebook/pfff
③https://coq.inria.fr/
④http://pt.wikipedia.org/wiki/Teorema_das_quatro_cores

```
12    if n mod 10 = 1 then
13      Printf.printf "t = %f,\ty = %f,\terr = %g\n" t y (abs_float (y -. exact t));
14    if n < 102 then loop h (n+1) (rk4_step (y,t) h)
15
16  let _ = loop 0.1 1 (1.0, 0.0)
```

Apêndice C

Implementação de uma Animação do Sistema Massa-Mola-Amortecedor em Clojure

C.1 Sobre a Linguagem

A linguagem Clojure[1] difere de muitas ao ser hospedada na JVM[2], no CLR[3] e no JavaScript[4]. É uma linguagem compilada e dinâmica, cuja sintaxe sofre fortes influências da família LISP, com ótimo suporte a concorrência e paralelismo. Predominantemente uma linguagem funcional, Clojure conta com estruturas imutáveis e persistentes (embora exista suporte a mutação de estruturas, caso haja necessidade).

A única dependência do aplicativo desnvolvido é a biblioteca play-clj[5], concebida para facilitar o desenvolvimento de jogos independentes em Clojure que rodem em *desktops* (Windows, OS X e Linux) e, experimentalmente, *mobiles* (Android e iOS). Sua utilização é simples, cabendo ao programador manipular uma lista contendo informações atualizadas sobre a tela atual e as entidades a serem inseridas nela.

[1] `http://clojure.org/`

[2] *Java Virtual Machine*, a máquina virtual onde rodam os programas escritos em Java. Mais informações em `http://en.wikipedia.org/wiki/Java_virtual_machine`.

[3] *Common Language Runtime*, componente do *framework* .NET da Microsoft. Mais informações em `http://en.wikipedia.org/wiki/Common_Language_Runtime`.

[4] Embora essa não tenha um site oficial, informações podem ser encontradas em `https://developer.mozilla.org/en-US/docs/Web/JavaScript` e em `http://www.ecmascript.org/`.

[5] `https://github.com/oakes/play-clj`

C.2 Equações do Sistema

Conforme demonstrado ao longo do texto, esse sistema, sem forçamento, apresenta três tipos de respostas, baseadas nos fatores de amortecimento. Portanto, considerando:

$$\omega_n = \sqrt{\frac{k}{m}} \tag{C.2.1}$$

$$\omega_D = \omega_n\sqrt{1-\xi^2} \tag{C.2.2}$$

$$\omega_P = \omega_n\sqrt{\xi^2-1} \tag{C.2.3}$$

e desconsiderando a aplicação de condições iniciais[①], as possibilidades de resposta do sistema e suas respectivas equações de movimento são[②]:

- sem amortecimento

$$x(t) = A\cdot\text{sen}(\omega_n t - \varphi) \tag{C.2.4}$$

- sub-amortecimento

$$x(t) = A\cdot e^{-\xi\omega_n t}\text{sen}(\omega_D t - \varphi) \tag{C.2.5}$$

- super-amortecimento

$$x(t) = e^{-\xi\omega_n t}\cdot\left(A_1 e^{\omega_P t} + A_2 e^{-\omega_P t}\right) \tag{C.2.6}$$

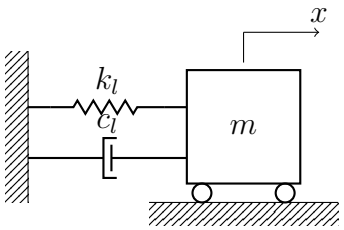

Figura C.2.1: Diagrama do sistema massa-mola-amortecedor sem forçamento.

C.3 Utilização do Programa

Com as informações anteriores, para a construção do programa, basta definir essas funções e manipular a equação de movimento conforme o usuário utiliza as teclas indicadas. Para

[①]De modo a facilitar o controle do aplicativo sobre a amplitude do movimento.

[②]Como as respostas do sistema a um fator de amortecimento crítico ($\xi = 1$) ou super-crítico ($\xi > 1$) são idênticas, e, como na prática é difícil a definição de um amortecedor crítico[23], desconsiderou-se a implementação desse.

mais detalhes, o código-fonte do projeto está disponibilizado no Github e pode ser baixado através do site ou com o comando:

```
git clone https://github.com/DonRyuDragoni/vibration-sim
```

que clona o repositório na pasta `vibration-sim` dentro do diretório atual.

Para a utilização do programa, contudo, é disponibilizado, na página do projeto, o arquivo compilado dentro da pasta `target`[1]. Preferencialmente, deve-se utilizar o executável marcado como *standalone*.

C.4 Demonstração do Programa

Ao inicializar o programa[2], o cronômetro começa a rodar, embora o sistema esteja no modo sem movimento (tecla `r`), como demonstra a figura C.4.1. Ao pressionar qualquer tecla, válida ou não, o programa reinicia o cronômetro a qualquer momento, tornando-o útil para reapresentar as animações. Existem, contudo, cinco teclas que controlam o programa:

- `r`: como dito anteriormente, não há movimento no sistema;
- `n`: o sistema oscila sem perda de energia, representando o movimento do sistema massa-mola (figura C.4.2);
- `l`: o sistema oscila com perda de energia, representando o sistema com amortecimento sub-crítico (figura C.4.3);
- `h`: o sistema não oscila, representando o sistema com amortecimento crítico ou super-crítico (figura C.4.4);
- `q`: encerra o programa.

[1] Ou seja, no link: `https://github.com/DonRyuDragoni/vibration-sim/tree/master/target`

[2] Refere-se aqui à versão 1.0.0 do programa, atual ao tempo de escrita. No página do projeto no Github, citada na seção anterior, podem haver informações mais atualizadas.

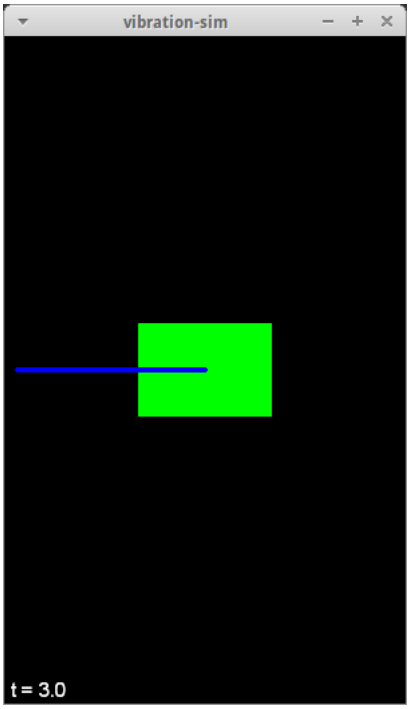

Figura C.4.1: Janela inicial do programa.

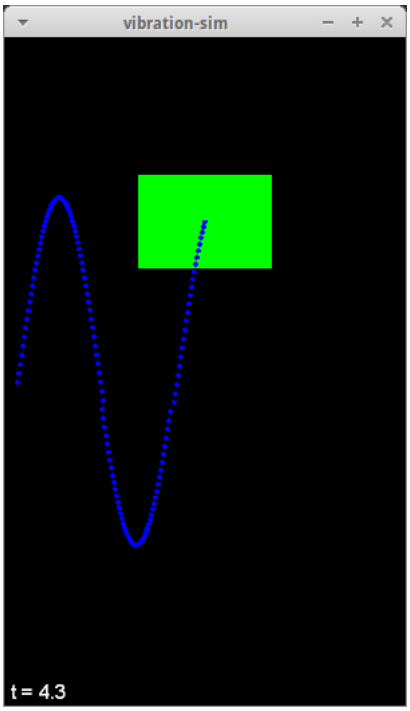

Figura C.4.2: Janela do programa rodando a animação do sistema sem amortecimento.

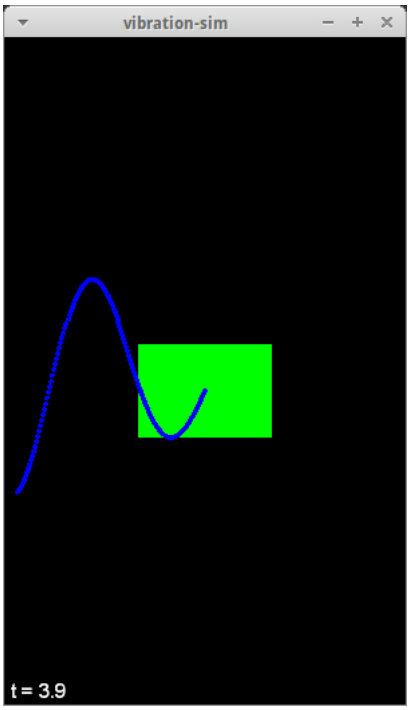

Figura C.4.3: Janela do programa rodando a animação do sistema com baixo amortecimento.

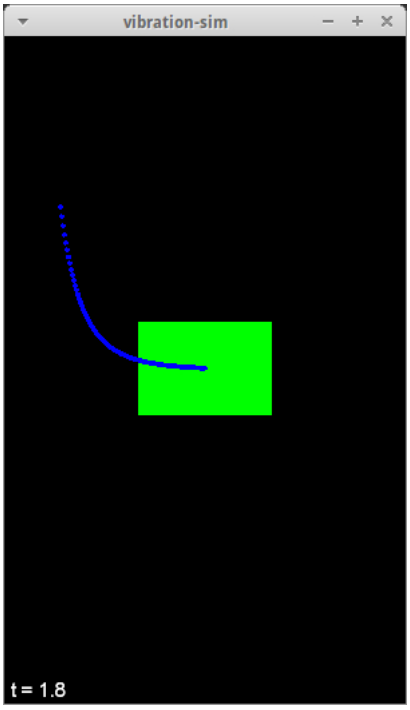

Figura C.4.4: Janela do programa rodando a animação do sistema com alto amortecimento.

Bibliografia

[1] Leonard MEIROVITCH. *Elements Of Vibration Analysis*. Inglês. 2ª ed. McGraw-Hill, 1986.

[2] Arthur Palmeira Ripper NETO. *Vibrações Mecânicas*. Português. E-papers, 2007. ISBN: 978-85-7650-106-0.

[3] K. G. BHATIA. *Foundations for Industrial Machines*. Inglês. 1ª ed. D-CAD Publishers, 2008. ISBN: 978-81-9060-320-1.

[4] Rao V. DUKKIPATI. *MATLAB: An Introduction With Aplications*. Inglês. New Age International Publishers, 2010.

[5] José Luiz BOLDRINI et al. *Álgebra Linear*. Português. 3ª ed. Harper & Row do Brasil, 1980. ISBN: 852-940-202-2.

[6] Samuel Daniel CONTE e Carl de BOOR. *Elementary Numerical Analysis - An Algorithmic Approach*. Inglês. 3ª ed. International Series In Pure And Applied Mathematics. Mcgraw-Hill College, 1980. ISBN: 978-00-7012-447-9.

[7] Hans Petter LANGTANGEN. *A Primer On Scientific Programming With Python*. Inglês. Springer, 2009.

[8] Claus BENDTSEN. "Highly Stable Parallel Runge-Kutta Methods". Inglês. Em: *Applied Numerical Mathematics* 21(1) (1996), pp. 1–8.

[9] Clint DAWSON et al. "A Parallel Local Timestepping Runge–Kutta Discontinuous Galerkin Method With Applications to Coastal Ocean Modeling". Inglês. Em: *Computer Methods in Applied Mechanics and Engineering* 259 (2013), pp. 154–165.

[10] Matthias KORCH e Thomas RAUBER. "Parallel Low-Storage Runge-Kutta Solvers for ODE Systems with Limited Access Distance". Inglês. Em: *International Journal of High Performance Computing Applications* 25(2) (maio de 2011), pp. 236–255.

[11] J. R. CASH e Alan H. KARP. "A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides". Inglês. Em: *ACM Transactions on Mathematical Software* 16 (set. de 1990), pp. 201–222.

[12] Michael F. ASHBY. *Material Selection In Mechanical Design*. Inglês. 3ª ed. Elsevier, 2004.

[13] William D. CALLISTER. *Ciência E Engenharia De Materiais: Uma Introdução*. Português. 7ª ed. LTC, 2008.

[14] Schuler GMBH, ed. *Metal Forming Handbook*. Inglês. 1ª ed. Springer, 1998. ISBN: 978-35-4061-185-1.

[15] Caleb DOXSEY. *An Introduction to Programming in Go*. Inglês. 1ª ed. CreateSpace Independent Publishing Platform, set. de 2012. ISBN: 978-14-7835-582-3.

[16] Mark SUMMERFIELD. *Programming in Go: Creating Applications for the 21st Century*. Inglês. 1ª ed. Addison-Wesley Professional, maio de 2012. ISBN: 978-03-2177-463-7.

[17] Claudio Cesar de SÁ e Márcio Ferreira da SILVA. *Haskell: Uma Abordagem Prática*. Português. 1ª ed. Novatec, 2006. ISBN: 857-522-095-0.

[18] Miran LIPOVAČA. *Learn You a Haskell for Great Good!* Inglês. No Starch Press, abr. de 2011. ISBN: 978-15-9327-283-8.

[19] Alejandro Serrano MENA. *Beginning Haskell: A Project-Based Approach*. Inglês. 1ª ed. Apress, jan. de 2014. ISBN: 978-14-3026-250-3.

[20] John WHITINGTON. *OCaml from the Very Beginning*. Inglês. Coherent Press, jun. de 2013. ISBN: 978-09-5767-110-2.

[21] Jon D. HARROP. *OCaml for Scientists*. Inglês. 1ª ed. Flying Frog Consultancy Ltd, 2005.

[22] Jason HICKEY, Anil MADHAVAPEDDY e Yaron MINSKY. *Real World OCaml*. Inglês. 1ª ed. O'Reilly Media, nov. de 2013. ISBN: 978-14-4932-391-2.

[23] Suresh ARYA, Michael O'NEILL e George PINCUS. *Design of Structures and Foundations for Vibrating Machines*. Inglês. Gulf Publishing Company, maio de 1984. ISBN: 087-201-294-8.

[24] David BEAZLEY e Brian K. JONES. *Python Cookbook*. Português. 3ª ed. O'Reilly Novatec, 2013. ISBN: 978-85-7522-332-1.

[25] Robert D. BLEVINS. *Flow-Induced Vibration*. Inglês. 2ª ed. Krieger Pub Co, abr. de 2001. ISBN: 978-15-7524-183-8.

[26] Michael R. HATCH. *Vibration Simulation Using MATLAB and ANSYS*. Inglês. Chapman & Hall/CRC, 2001. ISBN: 158-488-205-0.

[27] P. SRINIVASULU e C. V. VAIDYANATHAN. *Handbook of Machine Foundations*. Inglês. Tata McGraw-Hill Publishing Limited, 1976.

[28] John W. EATON et al. *GNU Octave Version 3.8.1 Manual: a High-Level Interactive Language for Numerical Computations*. Inglês. CreateSpace Independent Publishing Platform, 2014. ISBN: 144-141-300-6. URL: `http://www.gnu.org/software/octave/doc/interpreter`.