Revista de Ciências Exatas e Tecnologia

Vol. IV, Nº. 4, Ano 2009

SOFTWARE LIVRE APLICADO AO ENSINO DE ENGENHARIA E COMPUTAÇÃO

Eberval Oliveira Castro
Faculdade Anhanguera de Jundiaí
eberval.oliveira@unianhanguera.edu.br

Ângela C. de Oliveira Lühmann
Faculdade Anhanguera de Jundiaí
angela.luhmann@unianhanguera.edu.br

RESUMO

Nos últimos anos, tem-se notado no Brasil um movimento pela substituição de ferramentas de *software* proprietárias, de código fechado, por soluções livres, de código aberto. Dentre as vantagens obtidas podem ser citadas a maior independência tecnológica, a economia com licenciamento de *software* (saída de divisas para o exterior) e a possibilidade de adaptação (reuso) de código já existente. Este trabalho apresenta uma ferramenta de *software* livre chamada GNU Octave, aplicada ao ambiente acadêmico para apoio em aulas expositivas e experimentais. Exemplos são apresentados com o objetivo de demonstrar o seu no ensino de disciplinas de engenharia.

Palavras-Chave: Octave; simulação numérica; álgebra linear; controle.

ABSTRACT

In the last years, it is observed in Brazil a movement for replacement of closed source proprietary tools by open code free solution. Among the advantages obtained can be cited the greater technological independence, economy with software licensing (the flight of foreign currency abroad) and possibility of adaptation (reuse) previously made code. This paper presents a free software tool called GNU Octave, applied to the academic support of lectures and experiments. Examples are given in order to demonstrate their teaching in engineering disciplines.

Keywords: Octave; numerical simulation; linear algebra; control.

Anhanguera Educacional

Correspondência/Contato
Alameda Maria Tereza, 2000
Valinhos, São Paulo
CEP 13.278-181
rc.ipade@unianhanguera.edu.br

Coordenação
Instituto de Pesquisas Aplicadas e Desenvolvimento Educacional - IPADE

Informe Técnico
Recebido em: 29/10/2009
Avaliado em: 25/07/2010

Publicação: 21 de dezembro de 2010

1. INTRODUÇÃO

Nos últimos anos tem-se notado no Brasil um movimento de substituição de ferramentas de *software* proprietárias, de código fechado, por soluções livres, de código aberto. Dentre as vantagens obtidas podem ser citadas: a maior independência tecnológica, a economia com licença de *software* (saída de divisas para o exterior) e a possibilidade de adaptação (reuso) de código já existente. Este movimento visa minimizar a dependência tecnológica criada através da adoção de formatos fechados, frequentemente desenvolvidos por empresas estrangeiras, e a massificação de soluções de um único desenvolvedor, que além de prejudicar a livre concorrência através do monopólio de mercado, geram problemas relacionados à segurança da informação pelo armazenamento em formatos de padrão proprietário e fechado, muitas vezes não dominado pelo usuário final.

Programas de computador podem ser tratados tanto como produtos quanto como serviços, ou ainda como uma mescla das duas coisas. Quando a licença de uso de um determinado *software* é vendida e uma mídia contendo os binários do programa é entregue ao comprador, pode-se dizer que o mesmo está sendo comercializado na forma de produto. Por outro lado, pode-se contratar uma empresa para desenvolver uma solução de *software* para uma determinada demanda, o que caracterizando uma prestação de serviços.

Grandes empresas como Banco Itaú, Carrefour, Casas Bahia, Pão de Açúcar, Terra, Varig, Mais Indústrias de Alimentos, Philips, Mitsubishi e Deutsch Bank adotaram *software* livre como alternativa nos seus sistemas (BRASIL, 2009). Mais recentemente, o Banco do Brasil, que já era um grande utilizador de soluções de código aberto, adotou *software* livre em todos os seus terminais eletrônicos. O Governo Federal Brasileiro também tem sido um grande ator no fomento ao uso do *software* livre (FALCÃO et al., 2005), sendo considerado, por exemplo, uma referência mundial na informatização do processo eleitoral através do sistema de urnas eletrônicas, as quais operam usando *software* livre, conferindo segurança, transparência e rapidez no processamento do voto eletrônico (SANTANA, 2002).

Um outro aspecto muito importante a ser ressaltado é a questão econômica do mercado de *software*. O setor de TI movimenta no Brasil cerca de 3 bilhões de dólares anualmente. Deste total, aproximadamente 1/3 é para o pagamento de licenças de *software* proprietário. Dados do relatório oficial da Associação Brasileira das Empresas de Software (ABES) mostram que 58% dos programas usados no Brasil é ilegal (ABES, 2009). Segundo o mesmo relatório, a regularização movimentaria quantia em torno de 1,6 bilhão de

dólares, sendo que 400 milhões dólares apenas em impostos e uma outra grande parte em pagamento de *royalties*, representando mais fuga de divisas do país.

Por outro lado, o *offshore outsourcing*[1] tem ganhado força no Brasil, fazendo do país um grande exportador de soluções de TI ao lado de Índia e China. Segundo dados da Associação Brasileira de Empresas de Tecnologia da Informação e Comunicação, o Brasil exportou 1,4 bilhão de dólares no ano de 2008 na área de TI (BRASSCOM, 2009). Estes dados deixam muito claras as vantagens de investimentos no setor de TI a fim de promover o desenvolvimento nacional.

O uso do *software livre* é, portanto, uma alternativa técnica e economicamente viável tanto para o ambiente corporativo quanto para o acadêmico. Vale ressaltar que o termo "*software* livre" não significa simplesmente gratuidade, ele representa liberdade de uso. Tal liberdade é regida por licenças internacionais, dentre as quais a mais conhecida é a GPL (*General Public License* – Licença Pública Geral) que pode ser resumida nas quatro liberdades a seguir:

1. Liberdade para **executar** o programa, qualquer que seja o seu propósito.
2. Liberdade de **estudar** como o programa funciona, e adaptá-lo a necessidades particulares.
3. Liberdade de **redistribuir**, inclusive vender, cópias de modo a ajudar outras pessoas.
4. Liberdade de **modificar** o programa e distribuir estas modificações, de modo que toda a comunidade se beneficie.

Embora o movimento pró *software* livre seja visto por alguns como o braço comunista do mundo da informática, ele nasceu em berço capitalista, nos Estados Unidos da América, na década de 80, sob os ideais de Richard Stallman do MIT (*Massachusetts Institute of Technology* – Instituto de Tecnologia de Massachusetts) (FSF, 2009). A liberdade é a bandeira principal deste movimento e sua sobrevivência ao longo de todos estes anos se dá através de uma comunidade democrática e colaborativa.

Este trabalho apresenta uma alternativa de *software* livre para desenvolvimento de simulações e práticas laboratoriais na área de engenharia, mais especificamente para simulação numérica. Exemplos foram elaborados a fim de ilustrar o uso da ferramenta proposta capaz de servir como apoio no ensino de disciplinas como Álgebra Linear, Geometria Analítica, Estatística, Cálculo Vetorial, Equações Diferenciais, Cálculo Numérico, Controle Clássico, Controle Moderno, Controle Digital, Princípios de Comunicação entre outras. Tal adoção em detrimento a soluções proprietárias conduz à

[1] *Offshore outsourcing* é a prática de contratar uma organização externa para execução de algumas funções empresariais num país diferente daquele onde os produtos ou serviços são efetivamente desenvolvidos ou fabricados.

imediata economia com licenças de *software* além de permitir ao aluno fazer uso d*a* ferramenta em seu computador pessoal sem custos, o que é, na maioria das vezes, proibitivo para o estudante no caso da adoção de ferramentas proprietárias.

2. LINGUAGENS PARA COMPUTAÇÃO NUMÉRICA

Embora possa-se resolver diversos problemas numéricos usando bibliotecas específicas para linguagens de uso geral como SciPy para linguagem Python ou BLAS e LAPACK para diversas linguagens, o uso de linguagens de uso específico torna-se vantajoso pois a concepção das mesmas leva em conta a aplicação específica a que serão submetidas. No caso de aplicações em ciências exatas, linguagens de computação numérica são fundamentais em diversas áreas.

Linguagens de Computação Numérica são linguagens de uso específico muito usadas na solução de problemas como simulação numérica de modelos dinâmicos e projetos de sistemas de engenharia. Há uma grande variedade de linguagens deste tipo dentre as quais podemos citar linguagem R, Fortran, GNU Octave, Scilab, Matlab, Mathematica e PDL.

Dentre as ferramentas de computação numérica mais usadas nas áreas de exatas estão os interpretadores de linguagem matemática de alto nível. Embora a maioria dos currículos de nível superior atuais utilize o ambiente MathWorks Matlab no ensino laboratorial, existem alternativas livres que podem servir bem aos propósitos de uma ampla gama de disciplinas, tais como: Álgebra Linear, Geometria Analítica, Estatística, Otimização Linear e Não-Linear, Teoria de Controle, Análise Linear de Sistemas, Inteligência Artificial, Processamento Digital de Sinais, Análise por Elementos Finitos, Economia, Bioinformática entre outras. A introdução de ferramentas deste tipo torna o conteúdo mais atrativo e facilmente compreensível para o aluno, materializando muitos dos conceitos apresentados em sala de aula e promovendo maior fixação dos mesmos. Neste trabalho será apresentado um ambiente alternativo ao Matlab – GNU Octave – juntamente com exemplos práticos para que o leitor interessado possa fazer uso rápido dos seus recursos.

3. GNU OCTAVE

GNU Octave é um projeto mantido pelo Departamento de Engenharia Química da Universidade de Wisconsin, EUA, distribuído sob a GPL. Além de oferecer razoável compatibilidade com o Matlab, este ambiente permite instalação de pacotes a partir de

repositórios na Internet, adicionando funcionalidades ao interpretador de acordo com as necessidades do usuário através do comando `pkg`. Estão disponíveis versões pré-compiladas tanto para Linux quanto para Microsoft Windows que podem ser baixadas diretamente da página do mantenedor do projeto (EATON, 2008) ou ainda pode ser compilada uma versão otimizada para o equipamento em que será executado o programa. Embora os processos de instalação/compilação não sejam abordados neste trabalho (por estarem fora do seu escopo), vasta documentação pode ser obtida online sobre como proceder.

Na Figura 1, mostra-se a tela inicial do Octave para Linux e, na última linha, o *prompt* de comando indicando que o programa está pronto para receber entradas.

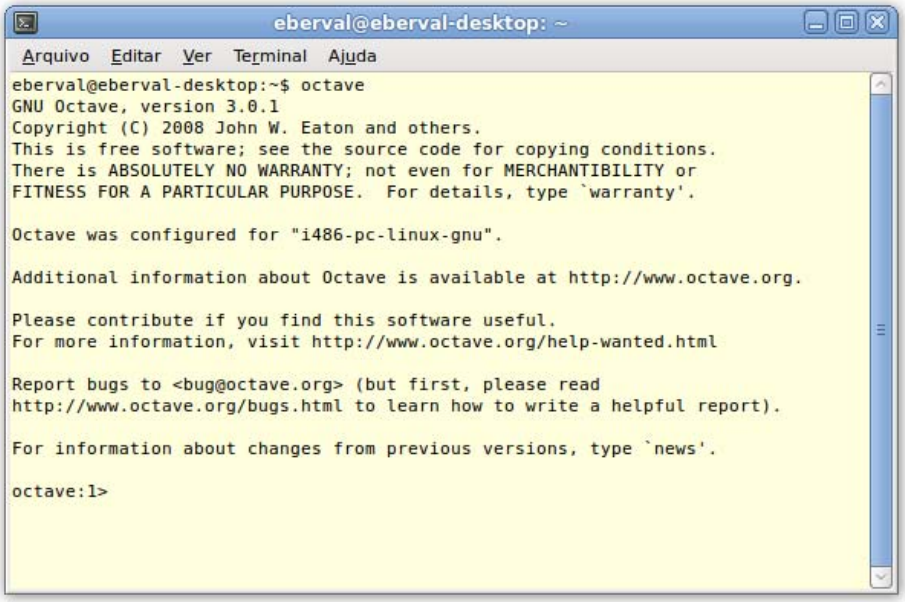

Figura 1 – Tela inicial (*prompt*) do GNU Octave.

Embora o Octave seja originalmente um ambiente em modo texto, existem interfaces gráficas disponíveis para o mesmo, a exemplo do QtOctave, além de pacotes de integração entre o interpretador e processadores de texto como GNU Emacs.

4. EXEMPLO: ÁLGEBRA MATRICIAL

O exemplo abaixo mostra como gerar uma matriz aleatória usando a função `rand()`, mostrar esta matriz na forma de uma imagem através da função `imagesc()`, alterar o mapa de cores através da função `colormap()`, calcular a inversa da matriz e verificar graficamente se o cálculo está correto através da visualização gráfica do produto da matriz por sua inversa. Cada comando está comentado (através do símbolo de porcentagem "%") para melhor compreensão.

```
octave:1> n = 80;         % define uma variável “n” com valor 80
octave:2> a = rand(n);    % gera matriz aleatória de ordem “n”
octave:3> imagesc(a);     % exibe uma imagem baseada na matriz “a”
octave:4> colormap(hot); % altera o mapa de cores para “hot”
octave:5> axis square;    % altera a escala dos eixos do gráfico
```

Após a sequência de comandos anterior, será exibida uma janela contendo a imagem apresentada na Figura 2, representando a matriz na forma gráfica. Os elementos de menor magnitude (próximos de zero) são representados por pontos mais escuros e os elementos de maior magnitude (próximos de um), por pontos mais claros.

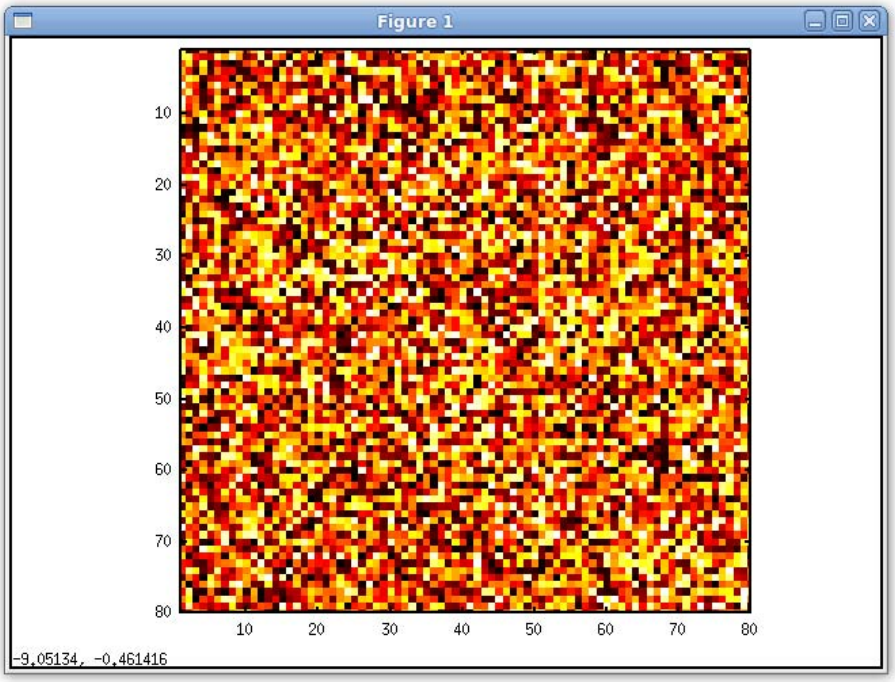

Figura 2 – Visualização gráfica da matriz “a”.

O cálculo da matriz inversa e sua representação na forma gráfica pode ser realizada através dos comandos a seguir:

```
octave:6> b = inv(a);    % calcula a inversa da matriz “a”
octave:7> imagesc(b);    % exibe uma imagem baseada na matriz b
octave:8> axis square;   % altera a escala dos eixos do gráfico
```

Estes últimos três comandos resultam na imagem que pode ser vista na Figura 3.

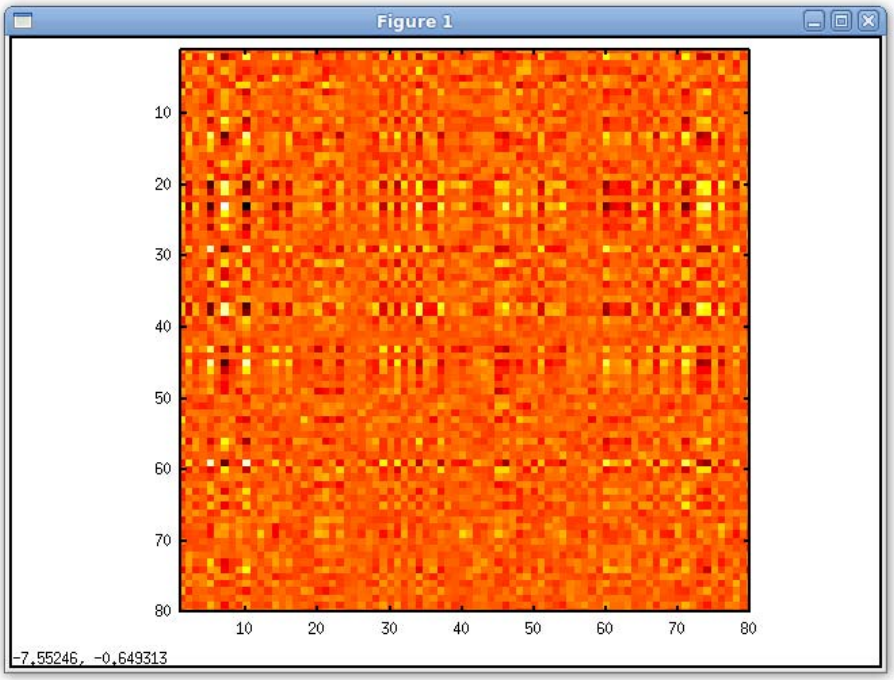

Figura 3 – Visualização gráfica da matriz inversa de “a”.

Neste ponto, para que se possa verificar que a matriz inversa foi calculada corretamente, basta calcular o produto entre a matriz “a” e a sua inversa, o que deve resultar na matriz identidade de ordem igual a da matriz original. Esta verificação pode ser realizada por meio dos comandos a seguir, resultando na imagem apresentada na Figura 4.

```
octave:9> imagesc(a*b);  % exibe uma imagem baseada em a*b
octave:10> axis square;  % altera a escala dos eixos do gráfico
```

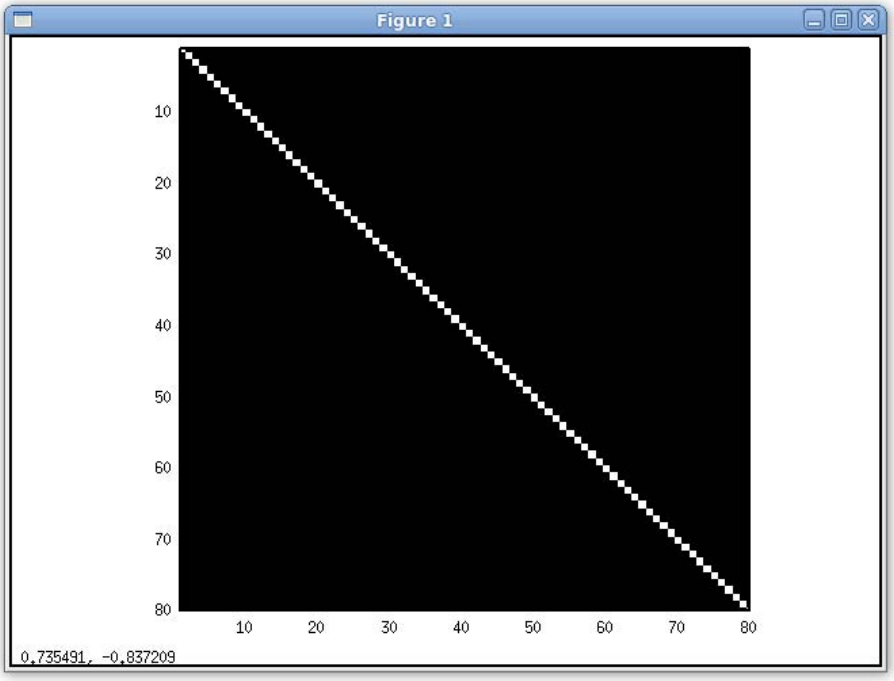

Figura 4 – Resultante do produto da matriz “a” por sua inversa.

5. EXEMPLO: CONTROLE DE SISTEMAS DINÂMICOS LINEARES

Outra área na qual o Octave se apresenta como ferramenta útil é nos temas abordados no Controle de Sistemas Dinâmicos Lineares, conforme o diagrama de blocos da Figura 5.

Figura 5 – Diagrama de blocos de um sistema dinâmico linear.

Existe um pacote de ferramentas especialmente desenvolvido para esta finalidade que pode ser instalado posteriormente à instalação do Octave bastando para isso baixar o pacote *Octave Control Systems Toolbox* (OCST) na seção "Control" do site oficial de desenvolvimento colaborativo (OCTAVE-FOURGE, 2009) e executando o comando abaixo no diretório onde foi baixado o arquivo do pacote[2]:

```
octave:11> pgk install control-1.0.11.tar.gz
```

O OCST foi criado no intuito oferecer uma interface simples para o usuário a um conjunto útil de ferramentas.

Como outras ferramentas para projeto de sistema de controle auxiliado por computador, o OCST permite ao usuário entrar com a descrição do sistema dinâmico na sua forma preferida (espaço de estados, função de transferência ou formato polos-zeros). Uma estrutura de dados de variáveis simples permite armazenar a descrição de sistemas em tempo contínuo, em tempo discreto ou ainda sistemas mistos (dados amostrados).

Esta descrição de variáveis simples de sistemas dinâmicos simplifica grandemente tanto o código do OCST quanto a interface com o usuário, uma vez que uma variável é passada por sistema, ao invés da representação interna usada na estrutura de dados.

Como resultado, tem-se a redução na quantidade de erros na chamada a funções e, adicionalmente, todas as funções OCST são escritas para fornecer mensagens de alerta significativas de modo a ajudar o usuário a corrigir seus erros de programação.

A estrutura de dados usada no OCST é chamada de "estrutura de dados de sistema" e é construída com um dos comandos a seguir:

- **fir** – função de transferência FIR (*Finite Impulse Response*) do sistema.

[2] O Octave aceita comandos de *Shell*. Caso seja necessário mudar de diretório, basta usar o comando interno: cd <nome_do_diretório>.

- **ss** – matrizes de espaço de estados do sistema.
- **tf** – função de transferência SISO (*Single Input Single Output*) do sistema.
- **zp** – coeficientes de zero/pólo/avanço do sistema.

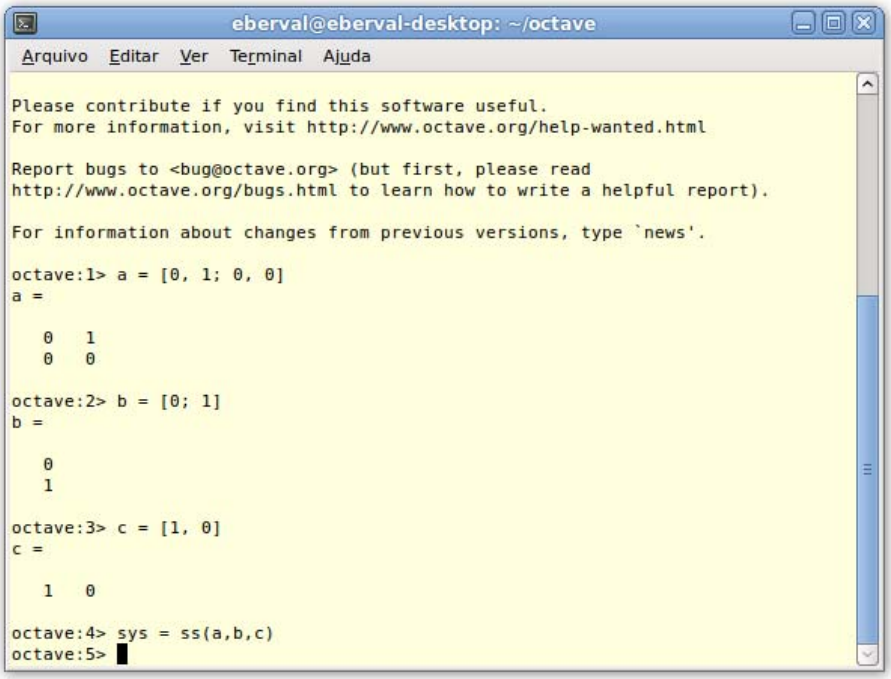

Figura 6 – Definição e criação da estrutura de dados do sistema.

Tomando como exemplo o modelo do controlador da cabeça de leitura de um disco rígido (MATHWORKS, 2009), conforme apresentado na Figura 7, aplicando as Leis de Newton tem-se que:

$$J\frac{d^2\theta}{dt^2} + C\frac{d\theta}{dt} + K\theta = K_i i \tag{1}$$

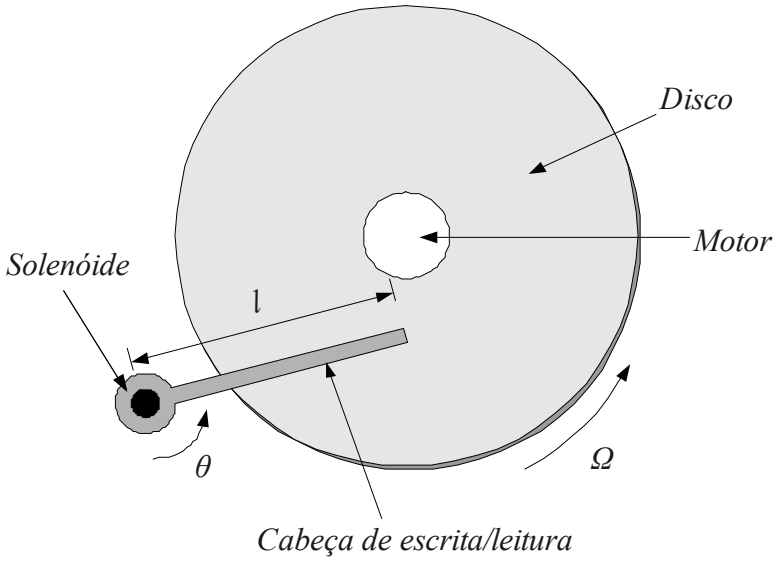

Figura 7 – Controlador de disco rígido.

onde J é a inércia do conjunto da cabeça de escrita/leitura, C é o coeficiente de atrito viscoso nas juntas, K é a constante de elasticidade da mola de retorno, K_i é a constante de torque do motor, θ é a posição angular da cabeça e i é a corrente de entrada. Aplicando a *Transformada de Laplace* à Equação 1 tem-se:

$$H(s) = \frac{K_i}{J\,s^2 + C\,s + K} \tag{2}$$

Usando os valores J = 0.01 kg.m², C = 0.004 Nm/(rad/s), K = 10 Nm/rad e K_i = 0.05 Nm/rad pode-se construir uma representação no Octave por meio dos comandos mostrados na Figura 8.

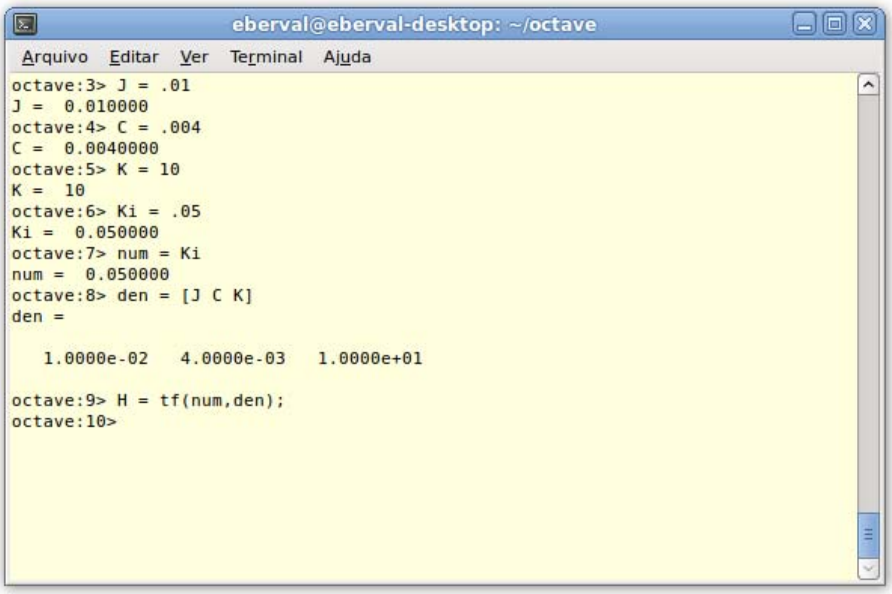

Figura 8 – Modelo do controlador de disco rígido no Octave.

Neste ponto é possível discretizar o sistema a fim de projetar um controlador digital. Para tanto, é necessário que o projeto seja realizado no domínio discreto. Considerando que a planta esteja equipada com um conversor A/D com segurador de ordem zero (*zero-order holder*) conectado à sua entrada, podemos realizar a discretização por meio dos comandos a seguir:

```
octave:11> Ts = .005; % período de amostragem = 0.005 segundos
octave:12> Hd = c2d(H,Ts);
```

Agora pode-se comparar os diagramas de Bode dos modelos através do comando abaixo. O resultado pode ser visto na Figura 9.

```
octave:13> Hd = c2d(H,Ts);
```

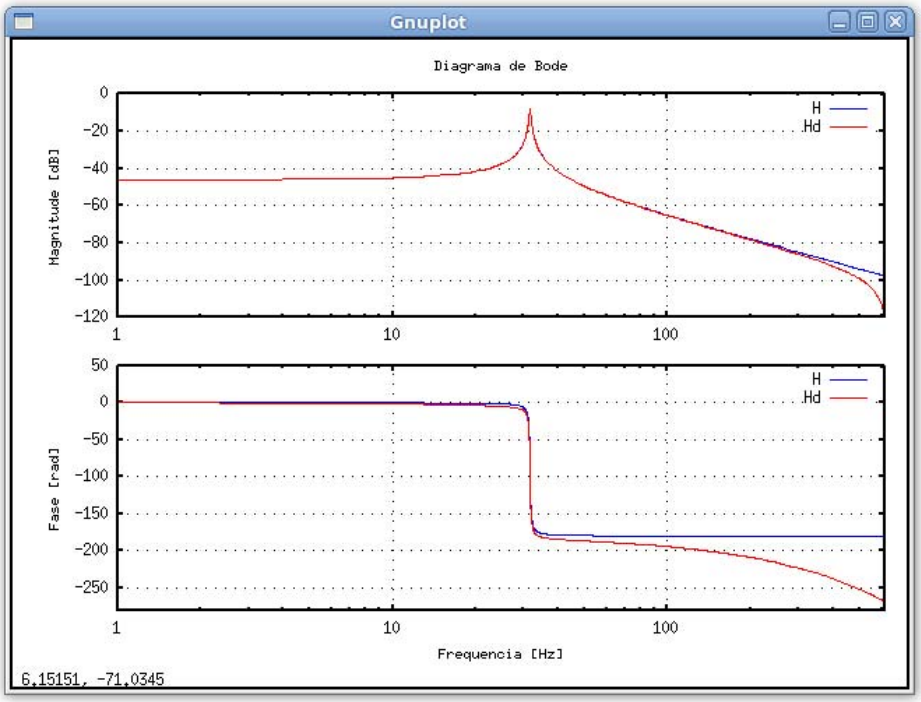

Figura 9 – Diagrama de Bode do sistema.

Uma vez que a representação do modelo tenha sido criada, é possível usar o Octave para estudar o sistema através dos recursos oferecidos pelo OCST.

A análise da resposta temporal do sistema, especificamente a resposta ao degrau, pode ser gerada no Octave usando a função `step()`. A Figura 10 mostra o resultado do comando `step(Hd)`[3].

Observando a Figura 10 percebe-se que o sistema oscila muito, provavelmente devido a um amortecimento muito pequeno. Esta verificação pode ser feita analisando os polos de malha aberta através da função `damp()` conforme o comando 26 na janela mostrada na Figura 11. A partir da saída do comando, observa-se que ambos os polos possuem um amortecimento equivalente muito pequeno e estão bastante próximos do círculo unitário.

[3] Uma importante configuração a ser realizada no Octave a fim de aumentar o tempo de simulação é alterar a variável N_MAX no arquivo /usr/share/octave/3.0.1/m/control/base/__stepimp__.m de 2000 para 10000. Esta alteração permite a produção de gráficos com mais informação em detrimento do tempo de simulação.

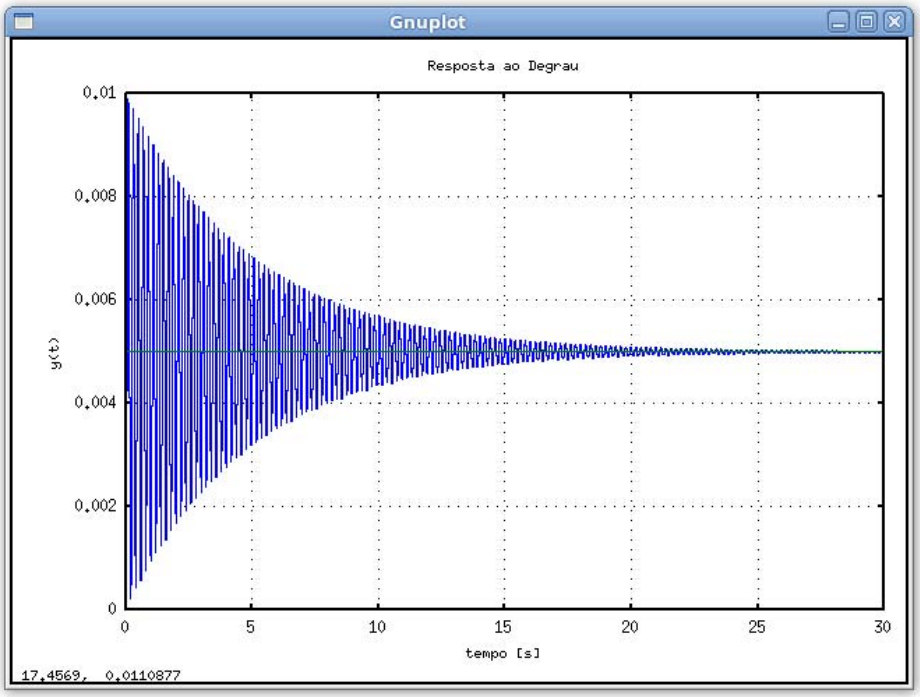

Figura 10 – Resposta ao degrau do sistema.

Para projetar um compensador simples (Figura 11), pode-se utilizar da técnica do lugar das raízes para encontrar um ganho de realimentação adequado.

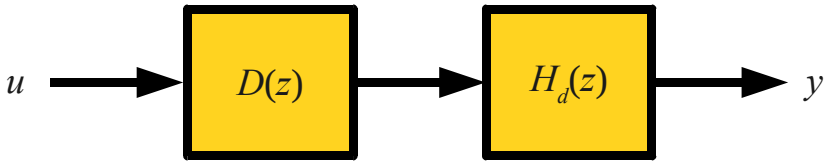

Figura 11 – Modelo de malha aberta do sistema compensado.

O lugar das raízes pode ser rapidamente obtido usando Octave conforme o comando 27 mostrado na janela da Figura 12. Os comandos adicionais têm a finalidade de ajustar a visualização e desenhar o círculo unitário. O resultado final está mostrado na Figura 13 com a representação gráfica do lugar das raízes. Os polos estão representados em marcas vermelhas e os zeros, em marcas verdes. Observando-se a Figura 13, nota-se que os polos se afastam rapidamente do círculo unitário, tornando o sistema instável.

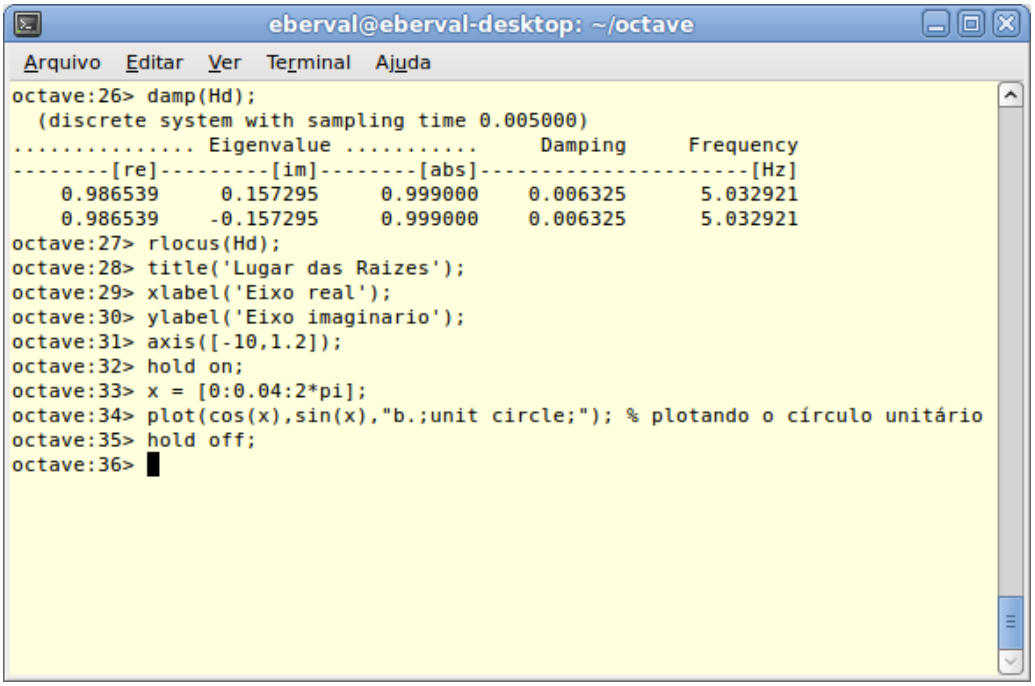

Figura 12 – Polos de malha aberta e lugar das raízes.

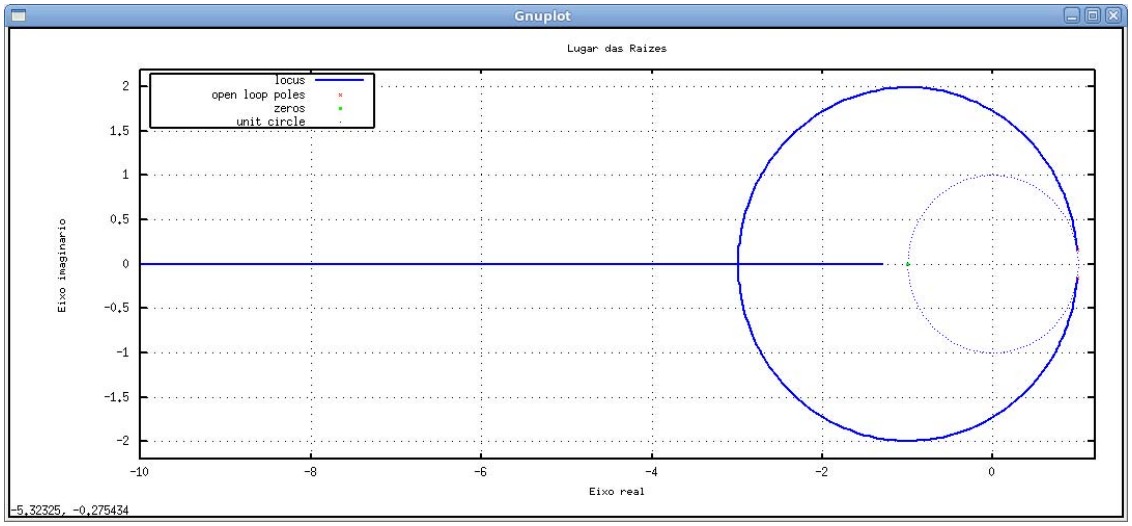

Figura 13 – Lugar das raízes.

Pode-se adicionar um compensador de modo a estabilizar este sistema conforme mostrado na Figura 11. Adicionando o compensador $D(z)$ dado na equação a seguir:

$$D(z) = \frac{z + a}{z + b} \tag{3}$$

Tomando a = – 0,85 e b = 0, na intenção de melhorar a resposta do sistema, é introduzido o compensador no modelo do Octave através dos comandos a seguir:

```
D = zp(0.85,0,1,Ts);   % Cria uma estrutura em polos e zeros
oloop = sysmult(Hd,D); % Produto do sistema pelo compensador
```

Traçando novamente e diagrama de Bode para o sistema compensado juntamente com o sistema original, obtém-se o resultado na Figura 14. A linha vermelha representa a o sistema original e a azul, o sistema compensado em malha aberta.

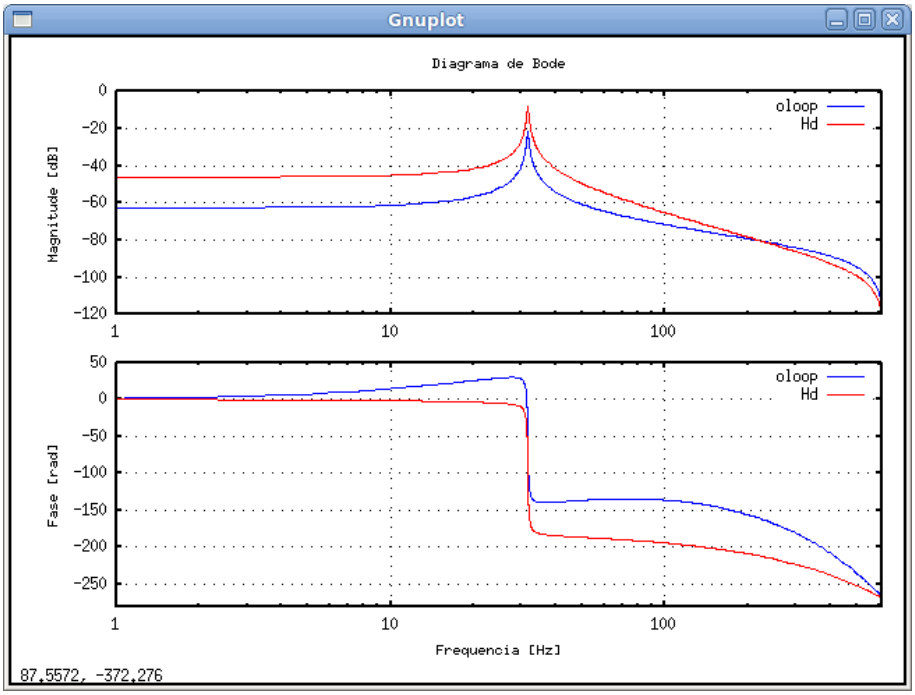

Figura 14 – Diagrama de Bode comparativo do sistema compensado.

Para gerar o gráfico do lugar das raízes do sistema compensado foi utilizada uma função adicional em linguagem Octave, chamada `zgrid()`, a fim de mostrar a variação do ganho e facilitar a escolha dos parâmetros do projeto. A função original, escrita por Tarmigan Casebolt (2007), sofreu pequenas adaptações para atender as necessidades deste trabalho e torná-la mais didática – foram retiradas as legendas e configurada a cor magenta para as linhas indicativas do mapa. O código da função modificada pode ser consultado no Apêndice A. O resultado obtido está representado na Figura 15.

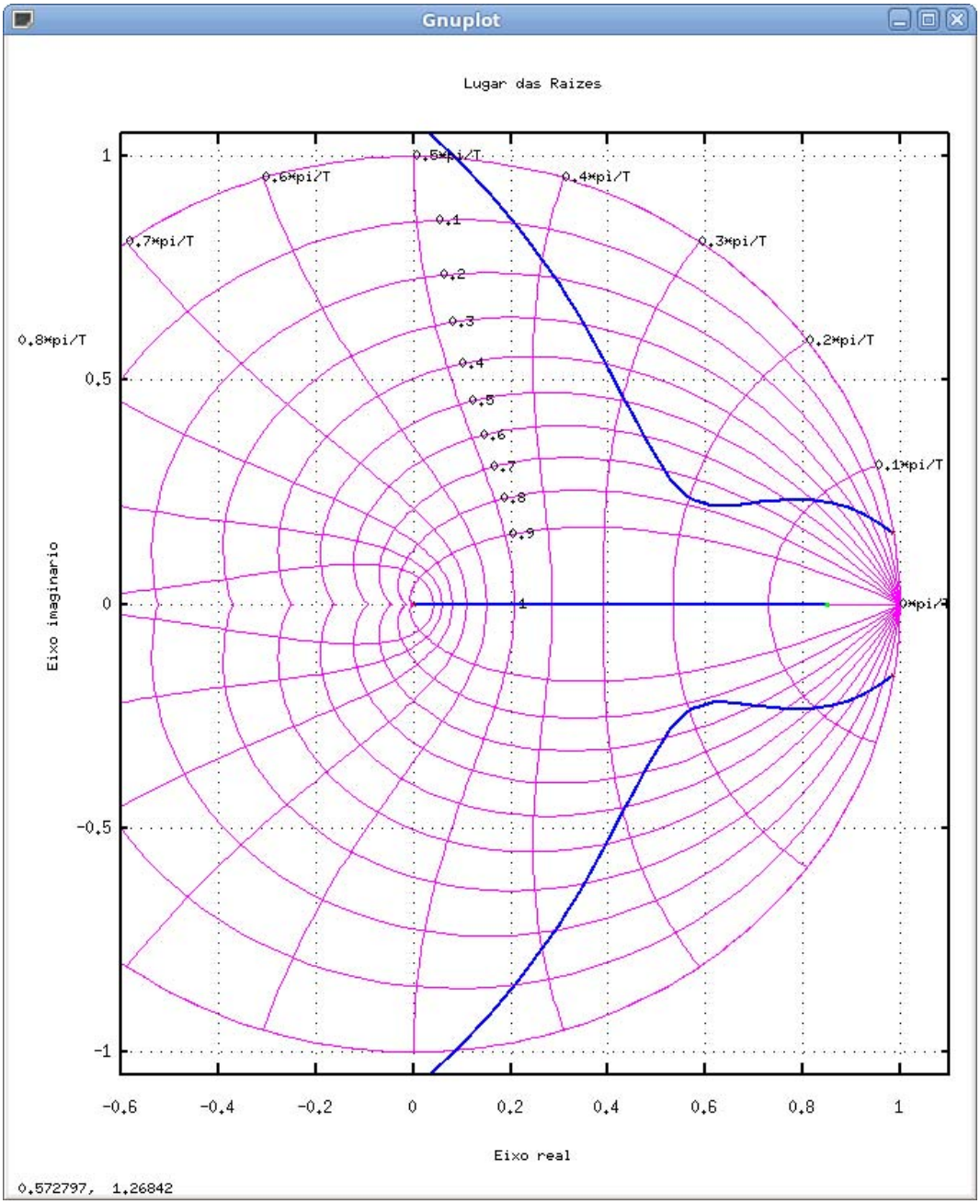

Figura 15 – Lugar das raízes do sistema compensado. Pontos verdes representam zeros e pontos vermelhos representam polos de malha aberta.

Na Figura 16, apresenta-se a resposta ao degrau unitário. Nota-se que o tempo de acomodação do sistema é agora inferior a 70 ms. Pode-se obter resultados ainda melhores utilizando filtragem, mas este assunto está fora do escopo deste texto. Para mais informações, sugere-se bibliografia especializada (OGATA, 2003).

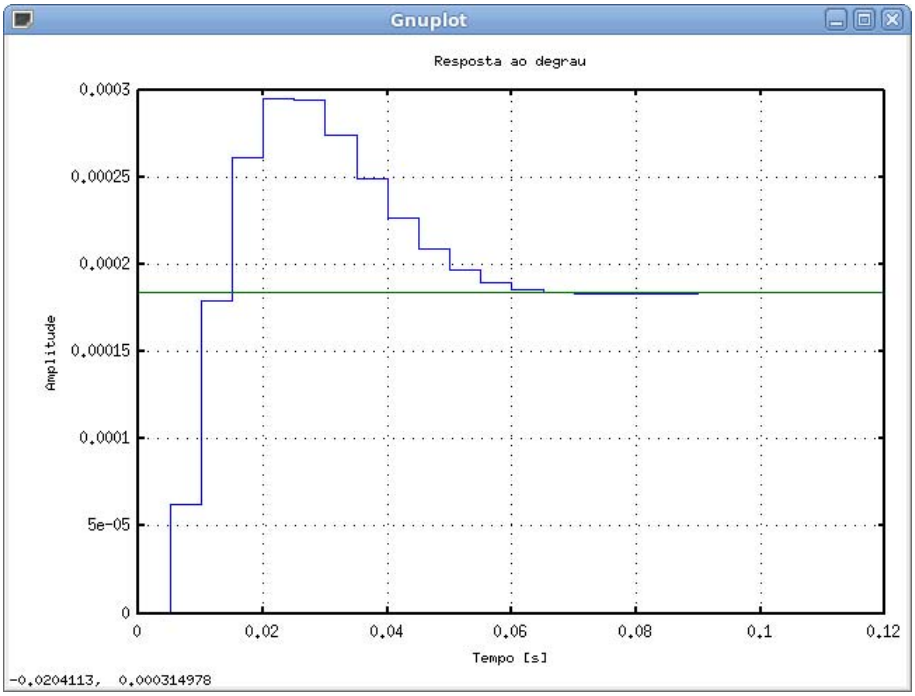

Figura 16 – Resposta ao degrau do sistema em malha fechada.

6. CONCLUSÕES

O presente trabalho demonstrou como o ensino de engenharia pode ser apoiado por ferramentas de *software* livre que podem, em muitos casos, substituir satisfatoriamente ferramentas proprietárias. Foram apresentadas as vantagens, inclusive econômicas, decorrentes da adoção desta abordagem por parte da instituição de ensino, bem como exemplos de aplicação do GNU Octave em Álgebra Linear e Controle Digital de Sistemas Dinâmicos. Embora estejam disponíveis centenas de ferramentas livres, sua adoção no Brasil ainda é modesta nos ambientes acadêmicos universitários. A simulação de circuitos em modo texto, por exemplo, pode ser realizada através de Spice; desenho técnico bidimensional auxiliado por computador, através do QCAD; simulação de circuitos elétricos em modo gráfico, através do Oregano; simulação de redes de computadores, de alta velocidade e convergentes, através do GNS3; projetos PCB, através do PCB Designer; como alternativa ao GNU Octave, o Scilab francês. A demonstração de uso de algumas destas soluções pode ser tema para trabalhos futuros.

REFERÊNCIAS

ABES. **Pirataria de software no Brasil**. Relatório Oficial (CNI). 2009. 6p. Disponível em: <http://www.abes.org.br/old/gruptrab/antipira_comsumo/relofipiratariaswbr-cni.pdf>. Acesso em: jun. 2009.

BRASIL. **Software livre**: mudando para melhor. Presidência da República. Casa Civil. Governo Federal. Disponível online em: <http://www.softwarelivre.gov.br/publicacoes/ cartilhaempdf>. Acesso em: 13 jun. 2009.

BRASSCOM. **Exportações de TI tem um salto de 75% em 2008**. Disponível em: <http://www.brasscom.com.br/brasscom/content/view/full/2849>. Acesso em: jun. 2009.

CASEBOLT, Tarmigan. **Zgrid**: Plots the locus of constant damping and natural frequencies for the discrete root locus. 2007. Disponível em: <https://www-old.cae.wisc.edu/pipermail/ octave-maintainers/attachments/20070718/c7172a17/ attachment.obj>. Acesso em: jun. 2009.

EATON, J.W. **Octave**. Disponível em: <http://www.gnu.org/software/octave/>. Acesso em: jun. 2009.

FALCÃO, J.; FERRAZ JUNIOR, T.S.; LEMOS, R.; MARANHÃO, J.; DE SOUSA, C.A.P.; SENNA, E.G. **Estudo sobre o software livre**. Presidência da República. Casa Civil. Instituto Nacional de Tecnologia da Informação. Rio de Janeiro, 18 mar. 2005.

FSF. Free Software Foundation. Disponível em: <http://www.fsf.org/>. Acesso em: jun. 2009.

MATHWORKS, The. **Hard-disk read/write head controller**. Disponível em: <http:// www.mathworks.com/access/helpdesk/help/toolbox/control/index.html?/access/helpdesk/help/toolbox/control/&http://www.mathworks.com/products/control/>. Acesso em: jun. 2009.

OCTAVE-FOURGE. **Extra packages for GNU Octave**. Disponível em: <http://octave.sourceforge.net/>. Acesso em: jun. 2009.

OGATA, Katsuhiko. **Engenharia de controle mordeno**. 4.ed. Pearson Brasil, 2003.

SANTANA, M.N.P. **Sobre a urna eletrônica**. Monografia - Departamento de Ciência da Computação. Universidade de Brasília, 2002.

APÊNDICE A

Código da função `zgrid()`

```
## Copyright (C) 2007 Tarmigan Casebolt
##
## This program is free software; you can redistribute it and/or modify
## it under the terms of the GNU General Public License as published by
## the Free Software Foundation; either version 2 of the License, or
## (at your option) any later version.
##
## This program is distributed in the hope that it will be useful,
## but WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
## GNU General Public License for more details.
##
## You should have received a copy of the GNU General Public License
## along with this program; if not, write to the Free Software
## Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

## -*- texinfo -*-
## @deftypefn {Function File} zgrid (@var{zeta}, @var{wn})
## Plots the locus of constant damping and natural frequencies for the
## discrete root locus.
##
## INPUTS:
## @var{zeta} and @var{wn} are vectors containing damping ratio and
## natural frequencies that you care about.
## @var{zeta} should be between 0 and 1.
## @var{wn} is the sampled natural frequency relative to the sampling period.
## If you would like to plot a real frequency, use @var{wn} = w_real / T_sample
##
## It can also be called with no input arguments to show the general contours
## within the unit circle.
##
## EXAMPLE:
## @example
## rlocus(zp([.5i -.5i],[.6 .6],1,1));
## zgrid
## axis tight
## axis square
## legend('off')
##
## ts = .1;
## rlocus(zp([-.9 .6],[.3 .75 .9],1,ts));
## zgrid(.6,.01*pi/ts)
## axis([-1 1 -1 1]); axis square; legend('off')
## @end example
##
## @strong{references}
## @itemize
## @item Franklin, Powell, and Emami-Naeimi @cite{Feedback Control of Dynamic Systems},
2002, p656
## @end itemize
## @seealso{sgrid, rlocus}
## @end deftypefn

function zgrid(varargin)

texton = 1;  # if you change this to 0, the text won't be printed
             # if you change this to 1, the numbers are printed
             # if you change this to 2, the names and numbers are printed

if (nargin == 0)
  zeta = .1:.1:1;
  wn = (0:.1:1)*pi;
elseif (nargin == 2)
  zeta = varargin{1};
  wn = varargin{2};
else
  warning('You must enter 0 or 2 input arguments.')
  return
end

oldaxis  = axis;
grid('off')

old_hold = ishold;
hold on;

wn_a = linspace(0,2*pi,100);
zeta_a = linspace(0,1,50);
```

```
# plot the lines for z
[zz,wwn]=meshgrid(zeta,wn_a);
s = -zz.*wwn + i*wwn.*sqrt(1-zz.^2);
z = e.^s;

# Do this so that we don't wrap around.
# z(imag(z)<0) = real(z(imag(z)<0));
# We do this ugly for loop so that we don't make an ugly dark line
# on the real axis.
for q=1:size(z,2)
  y = z(:,q);
  y(imag(y)<0) = real(y(find( imag(y)<0 ,1,'first')));
  z(:,q) = y;
end

plot(z,'m','linewidth',.25)
plot(conj(z),'m','linewidth',.25)

if texton > 0
  # put in text labels for zeta
  for q=1:size(z,2)
    loc = z(ceil(end/4),q);
    if texton == 2
      text(real(loc),imag(loc),['zeta = ',num2str(zeta(q))])
    else
      text(real(loc),imag(loc),[num2str(zeta(q))])
    end
  end
end

# plot the lines for wn.
[zz,wwn]=meshgrid(zeta_a,wn);
s = -zz.*wwn + i*wwn.*sqrt(1-zz.^2);
z = e.^s;

plot(z.','m','linewidth',.25)
plot(conj(z.'),'m','linewidth',.25)

if texton
  # put in the text labels for wn
  for q=1:size(z,1)
    if texton == 2
      text(real(z(q,1)),.03+imag(z(q,1)),...
           ['wn = ',10,num2str(wn(q)/pi),'*pi/T'])
    else
      text(real(z(q,1)),imag(z(q,1)),[num2str(wn(q)/pi),'*pi/T'])
    end
  end
end

# always plot the unit circle
t = linspace(0,2*pi,200);
plot(sin(t),cos(t),'m','linewidth',.5)

if ~old_hold
  hold off
end

legend('off');

%!demo
%! G = tf([1],[1 10 100]);    # model of the continuous plant
%! ts = .02;                  # sampling period
%! Gd = c2d(G,ts);
%! figure; subplot(121)
%! rlocus(Gd);
%! zgrid;                     # This time when we call zgrid, we just
%!                            # plot the general contours
%! axis([-1 1 -1 1]);
%! axis square; legend('off');
%! K = zp(.7,-.1,1,ts);       # make the compensator
%! subplot(122);
%! rlocus(sysmult(Gd,K));
%! t_rise = 200;
%! wn = (1.8/t_rise) / ts;
%! zgrid(.6,wn);              # This time we plot specific values
%! axis([-1 1 -1 1]); axis square; legend('off')
```