SOFTWARE LIVRE NA EDUCAÇÃO EM ENGENHARIA DE CONTROLE: UM ESTUDO DE CASO NA ANÁLISE E PROJETO DE CONTROLE DE UM PÊNDULO AMORTECIDO

A. G. LEMES[*], F. D. PRADO[*], D. KIAMETIS[*], A. S. SILVEIRA[*], A. A. R. COELHO[*], C. MEZA[**]

** Departamento de Automação e Sistemas - Universidade Federal de Santa Catarina*
Caixa Postal 476, 88040-900, Florianópolis, SC, Brasil
E-mails: {annelise, prado, dionicios, toninho, aarc}@das.ufsc.br

*** Departamento de Engenharia Eletrônica – Instituto de Tecnologia de Costa Rica*
Cartago, Costa Rica
E-mail: cmeza@ietec.org

Abstract— This paper aims to show the advantages of using Free Open Source Software (FOSS) for automatic control systems in education. Accordingly, this paper presents two control tasks for a process control course that can be done entirely with FOSS. These activities consist of the modeling and digital control of a damped pendulum.

Keywords— control education, free open source software, dynamic systems simulation, real-time control.

Resumo— Este artigo procura mostrar as reais potencialidades de uso de software livre na educação de sistemas de controle automático. Apropriadamente, este artigo ilustra o conteúdo de duas tarefas de controle para um curso de controle de processos que pode ser implementado inteiramente com software livre. As atividades consistem em modelagem e controle digital de um pêndulo amortecido.

Palavras-chave— educação em controle, software livre, simulação de sistemas dinâmicos, controle em tempo real.

1 Introdução

Efetivamente o ensino da engenharia de controle de processos (modelagem, análise, projeto do controlador, implementação) não pode ser concretizado somente em aulas teóricas com pouco ou nenhum ensaio prático. As simulações dinâmicas de processos, além de motivarem os alunos, proporcionam um conhecimento consistente do ponto de vista dos conteúdos teóricos ministrados em classe. Adicionalmente, para viabilizar as simulações numéricas e práticas, os estudantes compartilham da utilização de algum tipo de ferramenta baseada em computador, importante também dos pontos de vista do engenheiro da indústria ou do pesquisador da universidade.

Os avanços em software e hardware têm proporcionado o desenvolvimento de recursos de aprendizado por computador mais flexível com interação homem-máquina amigável, interfaces gráficas com o usuário e elevado grau de interatividade. Assim, é comum a utilização de software tipo CAD (Computer Aided Design) na análise e projeto de sistemas de controle em ambientes acadêmicos e industriais.

Algumas décadas atrás várias universidades optaram pelo desenvolvimento de softwares proprietários para sistemas de controle automático, não somente para dar suporte aos cursos de engenharia de controle, mas também como modo de ensino de uma ferramenta computacional que mais se aproximava do padrão da indústria. Na atualidade a maioria dos cursos de controle de processos emprega o software comercial de simulação Matlab/Simulink em atividades de ensino, aulas de laboratório ou em estudos de caso. Embora o software científico Matlab/Simulink represente uma padronização na educação de controle de processos, o custo de aquisição (obtenção e manutenção da licença de uso) está fora do padrão financeiro de muitas universidades e companhias da América Latina (Pires e Rogers, 2002).

Uma opção conveniente para os propósitos da universidade e da indústria é a utilização de software de distribuição livre (FOSS). O conceito de software livre está apoiado nos seguintes aspectos: i) liberdade de conhecer e modificar o código fonte; ii) liberdade de redistribuir e de uso sem restrições. FOSS para sistemas de controle tem atingido uma maturidade e desenvolvimento nos últimos anos em paises como Itália, França, Alemanha, Suíça, e são uma alternativa de utilização em termos dos consistentes resultados de programação e facilidade de instalação pelo usuário a partir da internet. Adicionalmente, apoiado nesta idéia, um grupo de professores e pesquisadores da América Latina implementaram um rede para troca de experiência e conhecimento, denominada HeDiSC (http://hedisc.ietec.org), onde existe o real interesse do FOSS para sistemas de controle, incentivando projetos acadêmicos e parcerias com a indústria. Embora outros softwares livres possam ser empregados, no momento somente a plataforma computacional ScicosLab, que inclui o ambiente Scicos, está efetivamente em desenvolvimento e representa uma alternativa FOSS para o Matlab/Simulink como um meio de ensino para os cursos e praticantes de controle de processos (Meza *et al.*, 2009).

Visando reforçar e validar a integração do ambiente computacional ScicosLab em cursos relacionados ao controle automático, o presente artigo apresenta um material de apoio desenvolvido e implementado no ScicosLab com experiências típicas da teoria de controle de processos em um sistema mecatrônico (pêndulo amortecido).

2 Material para Cursos de Engenharia de Controle Via FOSS

O material apresentado nesta seção é interessante para dar suporte em cursos da teoria de controle, entre os quais, sinais e sistemas lineares, sistemas não-lineares, controle clássico linear (por variáveis de estados ou por alocação de pólos), identificação e controle avançado. É um típico estudo de caso em um pêndulo amortecido que pode contribuir com vários exercícios ou experimentos para um período letivo de 4 meses e sendo perfeitamente implementado em FOSS. Uma das vantagens de utilização do FOSS para determinados tipos de tarefas é que os estudantes podem livremente adquirir e instalar o software em seus próprios computadores melhorando assim o aprendizado. Como mencionado anteriormente, existem diversas ferramentas computacionais FOSS que podem ser empregadas na educação da engenharia de controle. Neste caso, emprega-se o ScicosLab, que é um software de distribuição livre baseado no Scilab (programação por código) mas com uma versão mais estável e poderosa da plataforma Scicos (programação por diagramas) (Pires e Rogers, 2002; Pendharkar, 2005; Tona, 2006).

2.1 Recurso da Experimentação: Pêndulo

O pêndulo amortecido é um popular experimento de laboratório que pode ser utilizado para explicar aspectos como modelagem, análise e técnicas de projeto de controle. As características dinâmicas não-lineares e visuais deste sistema mecatrônico motivam o emprego no estudo do controle da posição com controle por realimentação (Bucher e Balemi, 2008).

O pêndulo amortecido contém uma barra vertical onde no ponto de pivô existe um potenciômetro para medição da posição angular, como pode ser visto na figura 1. No ponto extremo da barra existe um sistema propulsor composto por um motor DC e uma hélice. Quando uma voltagem é aplicada ao propulsor, a posição angular da barra é modificada. O objetivo é posicionar a barra para um ângulo especificado com uma dinâmica desejada.

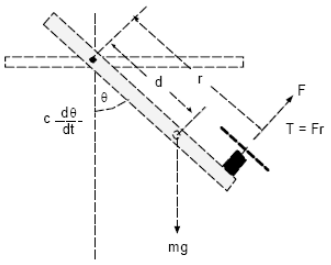

Figura 1. Processo pêndulo amortecido.

A equação dinâmica não-linear de movimento do pêndulo amortecido está caracterizada por

$$J\frac{d^2\theta(t)}{dt^2} + c\frac{d\theta(t)}{dt} + mgd\sin\theta(t) = T(t) \qquad (1)$$

Os parâmetros do sistema estão descritos na tabela 1, sendo $\theta(t)$ a posição angular (variável controlada) e $T(t)$ o torque de comando (sinal de controle).

Tabela 1. Parâmetros do pêndulo amortecido.

Símbolo	Descrição
J	Momento de inércia
c	Amortecimento
m	Massa do pêndulo
d	Distância para o centro de gravidade do pivô do pêndulo
r	Comprimento da barra em relação ao motor/hélice

Essencialmente, a aproximação para um ângulo pequeno é utilizada para linearização da equação dinâmica, onde o torque é proporcional a voltagem aplicado no motor e o resultado está na figura 2 (Fadali e Visioli, 2009).

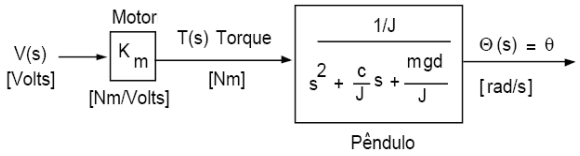

Figura 2. Modelo linearizado do pêndulo amortecido.

2.2 Recurso da Simulação: ScicosLab

Os requisitos necessários para realizar a análise, projeto e simulação numérica e experimental estão basicamente apoiados no ScicosLab 4.3, que é um software científico e desenvolvido por um grupo de pesquisa denominado *Metalau-INRIA*. Atualmente o ambiente computacional SicosLab é utilizado nos meios acadêmico e industrial (aplicações científicas e da engenharia) e, conforme ilustra a figura 3, é extremamente amigável. Por outro lado, existe um programa similar denominado Scilab que é desenvolvido pelo *Consórcio Scilab*. Ambos, ScicosLab e Scilab incluem o Scicos mas com diferentes versões de código. A versão atual do Scilab do *Consórcio Scilab* é muito instável, especialmente quando se utiliza o ambiente Scicos (Najafi *et al.*, 2005; Silva e Cunha, 2006; Meza *et al.*, 2009).

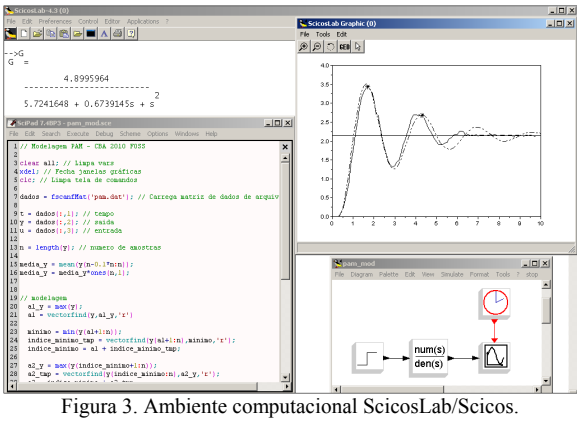

Figura 3. Ambiente computacional ScicosLab/Scicos.

2.2.1 Plataforma ScicosLab

O ScicosLab inclui um grande número de bibliotecas (toolboxes) que englobam funções gráficas, integração numérica, álgebra linear, otimização e outras. Existem também bibliotecas específicas para engenharia, como controle e processamento de sinais. O ScicosLab tem uma linguagem de programação própria que permite a criação de programas numéricos. Algumas características do ScicosLab são: linguagem de programação de alto nível, editor dedicado, centenas de funções matemáticas, possibilidade de adicionar programas a partir de várias linguagens (C, Fortran), vários toolboxes (álgebra linear, polinomial, estatística, controle clássico, identificação, entre outros) (Tona, 2006; Layec, 2009).

2.2.2 Plataforma Scicos

Scicos é uma ferramenta que permite a construção de modelos de sistemas dinâmicos contínuos e discretos através de diagrama de blocos similar ao Simulink do Matlab. Diversos blocos padrões estão disponíveis e organizados em grupos (palettes) específicos, como: i) Sources (degrau, rampa, senoide); ii) Sinks (osciloscópio, indicador numérico); iii) Linear (integrador, função de transferência contínua ou discreta), entre outros. É possível montar superblocos dos modelos de sistemas através da estrutura de blocos hierárquico. Uma amostra de blocos do Scicos recomendado para simulações dinâmicas está mostrada na figura 4 (Silva e Cunha, 2006).

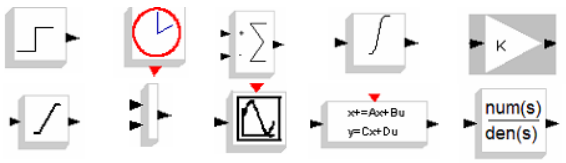

Figura 4. Blocos básicos do Scicos para simulação.

2.3 ScicosLab Integrado com um Sistema Real

As simulações via ScicosLab e Scicos podem ser utilizadas para interagir com sistemas reais nas seguintes diferentes formas:

i) Hardware in the Loop: O comissionamento de sistemas experimentais nem sempre é uma tarefa trivial e pode consumir horas de trabalho, resultados inadequados e situações de perigo. A técnica denominada Hardware in the Loop pode ser utilizada para avaliar e calibrar sistemas simulados numericamente no computador que está conectado com um controlador digital dedicado ou industrial.

ii) Gerador de Código Interno: O gerador de código (*menu: "Tools + Code Generation"*) converte a representação de diagrama do Scicos em um arquivo de código C de propósito geral para fins de aplicativos digitais dedicados. O código emprega um dispositivo de I/O padrão como interface. O usuário deve adaptar o código C para interagir adequadamente com outros dispositivos I/O (placas de aquisição de dados) e resolver os aspectos de sincronização e tempo real.

iii) Interface para Aplicação em Tempo Real: O RTAI-Lab (Real-Time Application Interface) é um projeto aberto integrando o código gerado por um CACSD (Computer Aided Control System Design) numa tarefa em tempo real do Linux RTAI. No presente momento, enquadra-se no ambiente CACSD quatro propostas: os comerciais Matlab/Simulink/RealTime Workshop (RTW), EicasLab, MatrixX, e o software livre ScicosLab/Scicos (Bucher e Balemi, 2005 e 2008; Mannori *et al.*, 2006; Meza *et al.*, 2009). RTAI-Lab proporciona também uma aplicação GUI (Graphical User Interface) para monitoramento remoto e controle em tempo real com freqüência de amostragem de *1 ... 10kHz*. Algumas aplicações em sistemas reais têm sido reportadas em processos mecatrônicos (pêndulo de Furuta, sistema torcional), máquina de fricção, tanques acoplados e aplicativo de precisão em posição (Bucher e Balemi, 2005 e 2008).

iv) Interface para Placa de Aquisição de Dados: A aproximadamente 15 anos o Departamento de Automação e Sistemas (DAS) da Universidade Federal de Santa Catarina (UFSC) decidiu construir suas próprias plantas para atividades de laboratório de controle uma vez que plantas comerciais semelhantes têm custo exagerado e de difícil manutenção. Desde então processos em escala de laboratório e placas de aquisição de dados estão presentes em várias disciplinas práticas do Curso de Engenharia de Controle e Automação. Tecnologias para o Ensino de Laboratório de Controle é uma área de pesquisa no DAS e muitos experimentos, artigos e teses têm sido desenvolvidos, http://www.das.ufsc.br (Coelho, 2009).

No Laboratório de Controle e Automação do DAS existem placas de aquisição de dados (DAS-01USB) que foram desenvolvidas especialmente para experimentações em controle de processos. O hardware das placas utiliza fundamentalmente *1* microprocessador PIC16F877A (Microchip), conversores D/A TLC5618AI (Texas Instruments) e para realizar a interface de comunicação com o computador, um conversor USB-FIFO, modelo FT245BL (FTDI).

Para que os sistemas operacionais (SO), como Windows e Linux, reconheçam o FT245BL, é fornecido pelo fabricante, e encontra-se disponível na internet (http://www.ftdichip.com/), um driver denominado ftd2xx.sys que realiza a comunicação entre o SO e o FT245BL e uma biblioteca de ligação dinâmica denominada ftd2xx (.dll para Windows e .so para Linux) que permite a interface com programas aplicativos. Portanto, é possível realizar a leitura e escrita de dados no FT245BL realizando chamadas das funções existentes na biblioteca. No entanto, o ScicosLab não consegue se comunicar diretamente com a ftd2xx.dll e uma biblioteca auxiliar precisa ser elaborada.

Utilizando uma distribuição gratuita do Microsoft Visual C++ 2008 Express Edition (http://www.microsoft.com/express), foi criada a biblioteca de interface ftd2xx_scicoslab.dll conforme descrito em FOSS-ADCON-DAS.

Para utilizar a nova biblioteca, é necessário ligar a mesma ao ScicosLab. Para isto, pode-se criar um *script* de inicialização como o da tabela 2.

Tabela 2. Script de inicialização da placa.

```
// loader.sce
id_d2xx = link('ftd2xx_scicoslab.dll','load_d2xx','c');
  link(id_d2xx,'free_d2xx','c');
  link(id_d2xx,'openUSB','c');
  link(id_d2xx,'closeUSB','c');
  link(id_d2xx,'readUSB','c');
  link(id_d2xx,'writeUSB','c');
  link(id_d2xx,'SetLatencyTimer','c');
  link(id_d2xx,'PurgeTxRx','c');

function openUSB()
  status = call('openUSB','out',[1,1],1,'i');
  if(status == 0) then
    disp('USB device opened');
  end
endfunction
function data_read = readUSB(n_bytes_read)
  [d_r,s_r,b_r] = call('readUSB',n_bytes_read,1,'i',
         'out',[1,1],2,'d',[1,1],3,'i',[1,1],4,'i');
endfunction
function writeUSB(n_bytes,data)
  [s_w,b_w] = call('writeUSB',n_bytes,1,'i',data,2,'i',
                   'out',[1,1],3,'i',[1,1],4,'i');
endfunction
function boot_daq()
  load_d2xx; openUSB; SetLatencyTimer(2);
  purgeRxTx(1); purgeRxTx(2); writeUSB(1,1);
endfunction
```

É importante ressaltar que a ftd2xx_scicoslab.dll pode ser usada para comunicar outros dispositivos da FTDI que suportam a ftd2xx.dll.

O Scicos possui um bloco denominado *Scifunc* que tem a capacidade de executar funções do ScicosLab e associar os resultados retornados aos canais de entrada e saída do bloco. Deste modo, pode-se inicializar, escrever e ler na DAS-01USB utilizando as funções definidas na tabela 2. Além disto, o usuário pode editar o ícone do bloco *Scifunc* através do menu de propriedades e guardar o bloco configurado como um item da paleta ou como um arquivo do Scicos. A figura 8 apresenta um bloco *Scifunc* editado para exercer as funções A/D e D/A no Scicos.

3 Identificação e Projeto de Controle

Para ilustrar a factibilidade de uso da plataforma ScicosLab em atividades da teoria de controle, simulações de modelagem e projeto do controlador, aplicados ao pêndulo amortecido, são apresentadas.

3.1 Identificação do Pêndulo Amortecido

Identificação consiste na determinação de um modelo matemático que representa os aspectos essências do sistema, relacionados através de uma função de transferência, equação diferencial ou a diferenças. O modelo final é uma forma do conhecimento da relação existente entre os sinais de entrada e saída. Uma vez obtido o modelo matemático do processo é possível, por exemplo, analisar e projetar um sistema de controle de modo a garantir as especificações de desempenho em malha fechada. Na prática utiliza-se a hipótese básica para construção de modelos de que processos reais, em geral, não necessitam obrigatoriamente de modelos complexos. Assim, um modelo matemático reduzido é adequado em diferentes aplicações práticas (Coelho e Coelho, 2004).

Para respostas oscilatórias o seguinte modelo contínuo é utilizado, sendo *y(t)* a saída, *u(t)* a entrada, ζ o fator de amortecimento e w_n a freqüência natural.

$$\frac{d^2y(t)}{dt^2} + 2\zeta w_n \frac{dy(t)}{dt} + w_n^2 y(t) = K_p w_n^2 u(t) \qquad (2)$$

Na identificação dos parâmetros é necessário obter a curva de reação da resposta ao degrau e medir o ganho estático, $Kp=\Delta y/\Delta u$, o período de oscilação, *To*, e os dois primeiros picos da resposta, *a1* e *a2*, conforme ilustra a figura 5.

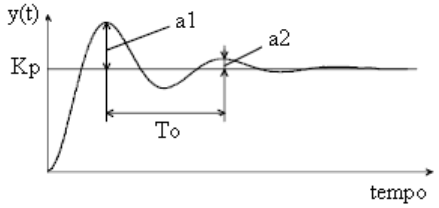

Figura 5. Estimação do modelo de segunda ordem.

Com a estimação por inspeção das constantes da figura 5 os parâmetros do modelo de segunda ordem são calculados pelas seguintes relações:

$$\zeta = \frac{1}{\sqrt{1+\left\{\frac{2\pi}{\ln\left(a2/a1\right)}\right\}^2}} \quad ; \quad \omega_n = \frac{2\pi}{To\sqrt{1-\zeta^2}} \qquad (3)$$

Para realização da tarefa de modelagem linear aplica-se uma tensão de entrada de *2.5 volts* e registra-se a posição angular do pêndulo amortecido em *volts*. Com a geração das medidas em arquivo texto carrega-se e executa-se este arquivo de dados com o código de programação da tabela 3.

Tabela 3. Código ScicosLab da modelagem com dados de campo.

```
clear all; // limpa variáveis
xdel; // fecha janelas gráficas
clc; // limpa tela de comandos
dados = fscanfMat('pam.dat'); // carrega dados
t = dados(:,1); // tempo
y = dados(:,2); // saída
u = dados(:,3); // entrada
n = length(y); // número de amostras
media_y = mean(y(n-0.1*n:n));
media_y = media_y*ones(n,1);
// modelagem
a1_y = max(y);
a1 = vectorfind(y,a1_y,'r');
minimo = min(y(a1+1:n));
ind_min_tmp = vectorfind(y(a1+1:n),minimo,'r');
ind_minimo = a1 + ind_min_tmp;
a2_y = max(y(ind_minimo+1:n));
```

```
a2_tmp = vectorfind(y(ind_minimo:n),a2_y,'r');
a2 = ind_minimo + a2_tmp;
ts = 0.1; // período de amostragem
T0 = (a2 - a1)*ts;
pi = %pi;
qsi = 1 / (sqrt(1+((2*pi)/(log(a2/a1)))^2));
wn = 2*pi / T0*sqrt(1-(qsi)^2);
Kp = media_y(n) / 2.5; // degrau u = 2.5;
// modelo estimado
s = poly(0,'s');
G = Kp*(wn^2) / (s^2 + 2*qsi*wn*s + wn^2);
```

Para completar a atividade de modelagem a realização da figura 6 deve ser implementada no ambiente Scicos (*num(s) = 4.9*, *den(s) = s^2 + 0.674s + 5.724*). O resultado obtido com o diagrama de blocos do Scicos é ilustrado na figura 7, onde pode ser observado que a resposta dinâmica do modelo linear estimado é similar as medidas de observação do pêndulo amortecido na faixa de operação selecionada.

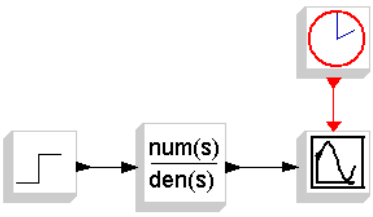

Figura 6. Diagrama do Scicos com a dinâmica linear do pêndulo.

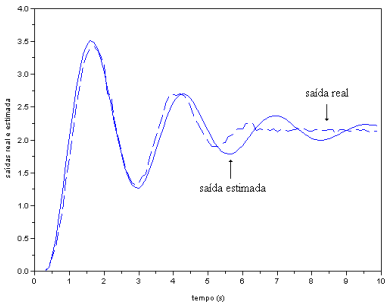

Figura 7. Respostas da posição angular real e estimada.

3.2 Projeto de Controle do Pêndulo Amortecido

Com o modelo linear estimado do pêndulo amortecido é possível projetar um controlador linear para estabilizar em diferentes pontos de operação. A síntese do controlador está apoiada no controlador por realimentação de estados e observador de estados de ordem completa. O projeto do observador visa garantir que as estimativas de todos os estados estejam disponíveis para realimentação.

A representação do pêndulo por variáveis de estados na forma discreta assume a seguinte forma:

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k) \\ y(k) &= Cx(k) \end{aligned} \qquad (4)$$

onde A_d e B_d são obtidas das relações

$$A_c = \begin{bmatrix} 0 & 1 \\ -\frac{mgd}{J} & -\frac{c}{J} \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ \frac{K_m}{J} \end{bmatrix}$$

e $x(k) = [x_1(k)\ x_2(k)]^T \approx [\theta\ \dot{\theta}]^T$. O controle por realimentação de estados é

$$u(k) = -Kx(k) + Fy_r(k) \qquad (5)$$

sendo $y_r(k)$ a referência e K o ganho de realimentação de estados, selecionado com base na equação característica de malha fechada, $A_{mf} = (A_d - B_d K)$, cujas raízes determinam o comportamento dinâmico e com ajuste pela técnica da alocação de pólos. Para garantir um adequado comportamento servo (a saída deve rastrear referências constantes), então assumir $F = [C(I_2 - A_{mf})^{-1} B_d]^{-1}$. Como o estado $x_2(k)$ não está acessível para realimentação, é necessário estimá-lo. A equação do estimador de ordem completa é dada por

$$\hat{x}(k+1) = A_d \hat{x}(k) + B_d u(k) + L[y(k) - C\hat{x}(k)] \qquad (6)$$

sendo L o ganho do estimador e os autovalores de $A_o = (A_d - LC)$ devem ter módulo menor do que um (Fadali e Visioli, 2009).

No projeto do controlador por realimentação de estados para o pêndulo amortecido tem-se a seguinte parametrização: $K = [0.74\ \ 0.98]$, $L = [1\ \ 1]^T$, $\hat{x}(0) = [0\ \ 0]^T$, $F = 2.2$, $ts = 0.1\ s$ (período de amostragem). Embora os resultados experimentais tenham sido obtidos para o ambiente de diagrama de blocos do Scisos (figura 8), é possível viabilizar todo o projeto e experimentação em código de programação via ScicosLab.

O processo é submetido a um ensaio de *10 s* para uma mudança de referência de posição correspondente a *2 volts*. Os resultados da figura 9 ilustram os comportamentos dinâmicos da posição angular e do observador de estados. A saída da planta rastreia a referência com uma energia de controle conservativa e o estimador tem bom desempenho.

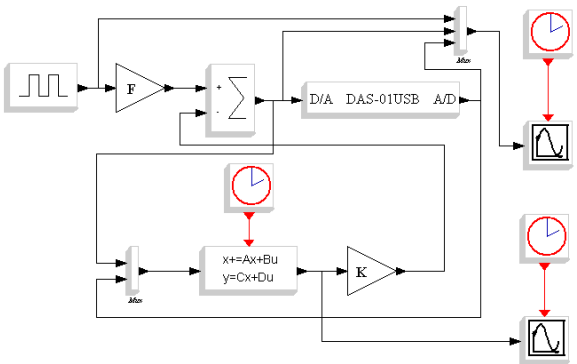

Figura 8. Diagrama do Scicos com o controlador por realimentação de estados em tempo real no pêndulo.

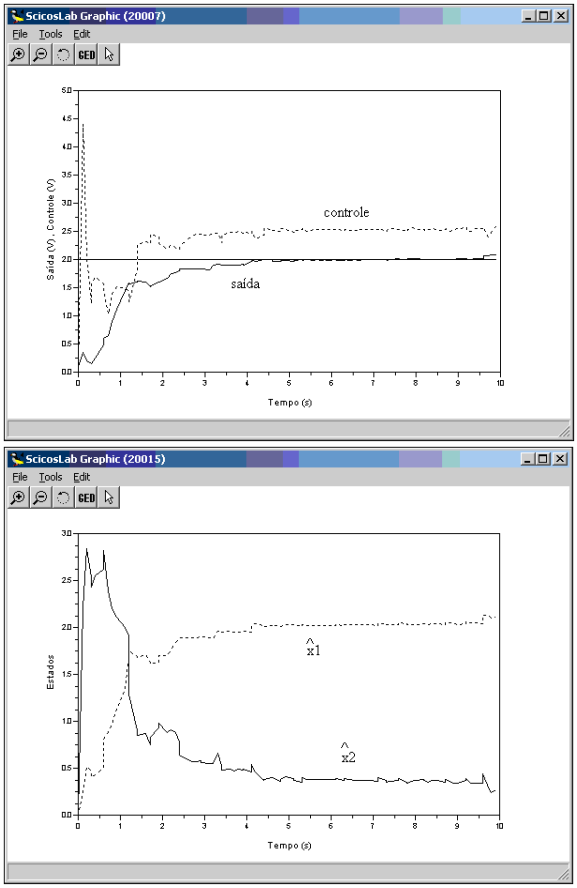

Figura 9. Respostas da posição angular e do estimador de estados.

Embora o sistema de controle do pêndulo tenha sido projetado, no caso do controlador por realimentação de estados, usando alocação de pólos e observador de estados, outros projetos de controle clássico e avançado podem também ser implementados, o que torna a plataforma (planta+ScicosLab) eficiente para atividades da engenharia de controle de processos.

4 Conclusão

Este artigo descreveu, em forma de atividades práticas, como o ambiente computacional ScicosLab pode ser utilizado no ensino da engenharia de controle de processos. Aspectos da modelagem e controle digital direto para um pêndulo amortecido foram apresentados e desenvolvidos inteiramente em FOSS. Os autores acreditam que a disponibilidade atual de softwares livres para sistemas de controle automático é suficientemente estável para os propósitos de ensino. Os resultados apresentados pelas comunidades internacional e nacional indicam um claro benefício no uso de FOSS em controle de processos, quer na academia ou indústria, onde os custos do laboratório são extremamente reduzidos (não existe a necessidade do pagamento da licença de uso). No que diz respeito ao aprendizado dos alunos, não se observa uma degradação com a educação quando se redirecionam as simulações, exercícios e estudos de caso totalmente para a plataforma FOSS.

Agradecimentos

Agradecemos a UFSC, DAS GPqTCA e CNPq (PIBIC e 478828/2009-8) pelo apoio para a realização deste trabalho.

Referências Bibliográficas

Coelho, A. A. R.; Coelho, L. S. (2004). Identificação de Sistemas Dinâmicos Lineares, Editora da UFSC.

Bucher, R.; Balemi, S. (2005). Scilab/Scicos and Linux RTAI - A Unified Approach, IEEE Conference on Control Applications, Toronto, Canada, pp. 1121-1126.

Bucher, R.; Balemi, S. (2008). CAN-Bus Based Rapid Control Prototyping System for Education Laboratories, 17th World Congress of IFAC, Seoul, Korea, pp. 9761-9766.

Coelho, A. A. R. (2009). Connecting Practices for Control Education: Teaching with Home-Made Plants, 1st HeDiSC Workshop, Lugano, Switzerland (http://hedisc.ietec.org/).

Fadali, M. S.; Visioli, A. (2009). Digital Control Engineering: Analysis and Design, Elsevier.

FOSS-ADCON-DAS (2009). Disponível em www.das.ufsc.br/~aarc/FOSS-ADCON/DAS01USB.

Layec, A. (2009). ScicosLab: A Free Scientific Software Package, 1st HeDiSC Workshop, Lugano, Switzerland (http://hedisc.ietec.org/).

Mannori, S.; Nikoukhah, R.; Steer, S. (2006). Free and Open Source Software for Industrial Process Control Systems, Internal Report, http://www-rocq.inria.fr/scicos/scicoshil.html.

Meza, C.; Romero, J. A. A.; Bucher, R.; Balemi, S. (2009). Free Open Source Software in Control Engineering Education: A Case Study in the Analysis and Control Design of a Rotatory Inverted Pendulum. 14th IEEE Intern. Conf. on Emerging Technologies and Automation Education, Palma de Mallorca, Spain.

Najafi, M.; Nikoukhah; Steer, S.; Furic, S. (2005). New Features and New Challenges in Modeling and Simulation in Scicos, IEEE Conf. on Control Appl., Toronto, Canada, pp. 1109-1114.

Pendharkar, I. (2005). Rltool for Scilab: A Public Domain Tool for SISO System Design, IEEE Control Systems Magazine, vol. 25, pp.23-25.

Pires, P. S. M.; Rogers, D. A. (2002). Free/Open Source Software: An Alternative for Engineering Students, 32nd IEEE Frontiers in Education Conf., Boston, MA, USA, pp. 7-11.

Silva, E. M.; Cunha, J. P. V. S. (2006). Scilab, Sicos e Rltool: Softwares Livres no Ensino de Engenharia Elétrica, XVI Congresso Brasileiro de Automatica, Salvador, Bahia, pp. 1620-1625.

Tona, P. (2006). Teaching Process Control with Scilab and Scicos, American Control Conf., Minneapolis, Minnesota, USA, pp. 280-285.