Introdução ao Scilab

Cap Carrilho

Fevereiro de 2004

Objetivos

- Conhecer o software Scilab como ferramenta de CASD.
- Manipular algumas funcionalidades importantes do Scilab, a saber:
  - Programação;
  - Gráficos;
  - Ambiente SCICOS, e
  - Interfaceamento com Fortran e C.

Objetivos

- Usar o Scilab para resolver problemas de
  - Controle e
  - Processamento de sinais.

Sumário

1. Introdução

- O que é o Scilab?
  - Ferramenta para o desenvolvimento de sistemas de controle automático e processamento de sinais.
  - Criado e mantido pelo INRIA.

Destaques do Scilab

- Diversos objetos
  - Matrizes, polinômios, razões de polinômios, funções de transferência, equações de estados etc.
- Inúmeras funções primitivas básicas
  - Álgebra linear, solução de EDOs, otimização, controle automático, processamento de sinais etc.
- Ambiente de programação que permite a criação de novas funções ou bibliotecas pelo usuário.
- Ambiente gráfico SCICOS.
- Interface com funções escritas em C e Fortran.

Observações gerais

- Software grátis com código fonte aberto.
  - Versão 2.7 disponível em http://scilabsoft.inria.fr/.
- Binários disponíveis para diversas plataformas:
  - Diversas versões de Unix e Windows.
- Documentação
  - Também disponível no site http://scilabsoft.inria.fr/doc.html

Comentários sobre o uso

- Scilab
  - Software com código livre e distribuição gratuita.
  - *Scilab Consortium*: intenção de expansão.
- Matlab
  - Software comercial e pago ($$$).
  - Padrão no meio acadêmico (ensino e pesquisa) e indústria.
- Relação de compromisso
  - Uso legalizado para ensino e para pesquisa.
  - Na pesquisa, poder comunicar-se com outros grupos.

Experiência pessoal

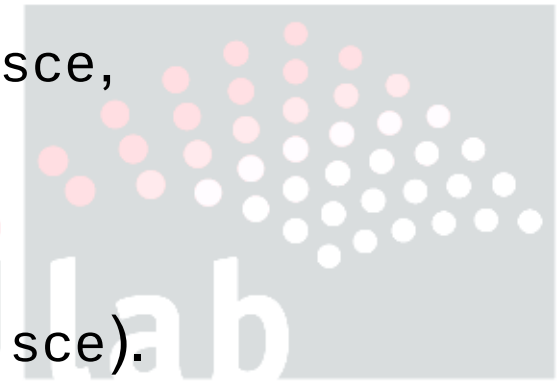
- UFSC
  - Uso de software livre recomendado.
  - Projeto *Disciplina Livre* do GUFSC – grupo de usuários de software livre da UFSC.
  - Uso ainda considerável do Matlab em pesquisa.
- CMU
  - Matlab com licenças distintas para ensino e para pesquisa.
  - Dois episódios de modificação dos termos da licença.
  - Experiência com os desenvolvedores do CheckMate.

Uma sessão inicial

- Introdução à interface.
- Básico de manipulação de números e matrizes.
- Manipulação de polinômios e funções de transferência.
- Sistemas lineares.
- Funções definidas na linha de comando.
- Interface com o sistema e programas em C.
- Solução de EDO e manipulação de funções.
- Arquivos de *script* `sessao0i.sce` i=1,...,6.

2. Tipos de dados

- Constantes especiais (`constantes_especiais.sce`).
- Escalares, Vetores e Matrizes (`matrizes.sce`).
- Polinômios (`polinomios.sce`).
- Sistemas lineares (`sistema_linear.sce`).
- Algumas especificidades:
  - Listas (`listas.sce`).
  - Matrizes de booleanos e inteiros (`boolean.sce`, `inteiro.sce`).
  - Vetores N-dimensionais (`vetores_n.sce`).
  - Indexação de matrizes e listas (`indexacao.sce`).



3. Programação

- O Scilab fornece ao usuário a possibilidade de criar e usar novas funções.
- Permite o desenvolvimento de programas especializados que podem se integrar no pacote do Scilab de forma simples e modular (bibliotecas).

Ferramentas de programação

- Operadores `==`, `<`, `>`, `<=`, `>=` e `<>`.
- Laço `for` e laço `while`.
- Condicionais `if-then-else` e `select-case`.
- Exemplo: `programacao.sce`

Estrutura de uma função

- Sintaxe

```
function [y1,...,yn] = foo (x1,...,xm)
...
endfunction
```

- `xi` são os argumentos de entrada.
- `yj` são os argumentos de saída.

Definição de funções

- As funções podem ser definidas *in line* ou em arquivos (extensão `.sci`).
- Um arquivo pode conter diversas funções.
- As funções são objetos do Scilab.
- Comandos `getf`, `exec` e `exists`.

Exemplo: fatorial

$$n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$$

- Implementação do cálculo do fatorial de n:

  k = 1

  Para i de 1 até n faça k = k.i

  fatorial(n) = k

- Implementação recursiva:

  $fatorial(n) = n \cdot fatorial(n-1) \quad n \geqslant 1$

  $fatorial(n) = 1 \quad n < 1$

Exemplo: Cálculo do determinante

- Expansão em cofatores e menores:

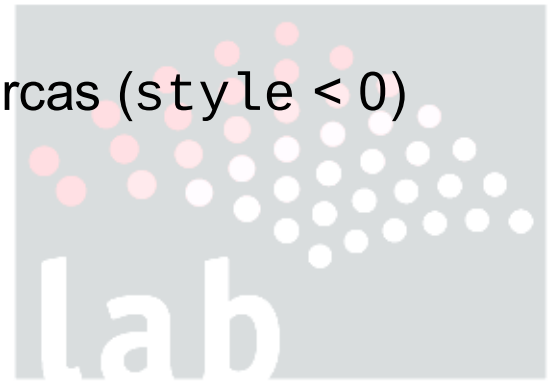
$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix} = (-1)^{1+1} \cdot 1 \cdot \begin{vmatrix} 5 & 6 \\ 8 & 9 \end{vmatrix} + (-1)^{1+2} \cdot 2 \cdot \begin{vmatrix} 4 & 6 \\ 7 & 9 \end{vmatrix} + (-1)^{1+3} \cdot 3 \cdot \begin{vmatrix} 4 & 5 \\ 7 & 8 \end{vmatrix}$$

4. Manipulação de gráficos

- Gráficos 2D genéricos
  - `plot2d(x, y [, opt])`
  - `x` : vetor coluna com valores para o eixo horizontal.
  - `y` : vetor coluna ou matriz com valores para o eixo vertical.
- Exemplo
  - Desenhar o gráfico de y = x.sen(x) para x entre -50 e 50.
  - Desenhar também as linhas y = x e y = -x.

Escala e estilo do gráfico

- Comandos opcionais `plot2di`:
  - `i = 1` : logarítmico
  - `i = 2` : constante por partes
  - `i = 3` : barras
  - `i = 4` : setas
  - Exemplo: `graficos_plot2di.sce`
- Estilo de ponto
  - Parâmetro `style` define cores (`style` > 0) e marcas (`style` < 0) diferentes
  - Identificação de cores e marcas com xset()
  - Exemplo: `graficos_estilos2d.sce`



Molduras, legendas, divisões e subdivisões dos eixos

- Estilos de eixo
  - `axesflag = 5` (par de eixos passando por (0,0))
- Legendas para as curvas
  - `leg = "curva1@curva2@..."`
- Limites do gráfico
  - `rect = [ xmin, ymin, xmax, ymax]`
- Número de divisões e subdivisões
  - `nax = [ nx, Nx, ny, Ny]`
- Exemplo: `graficos_legendas2d.sce`

Cabeçalhos e apresentação

- Grade
  - `xgrid()`
- Título do gráfico
  - `xtitle("Gráfico","Eixo x","Eixo y")`
- Ajuste dos parâmetros do gráfico
  - `xset()`
- Título ao fundo do gráfico
  - `titlepage("Título")`
- Exemplo: `graficos_cabecalhos2d.sce`

Gráficos: generalidades

- Exportação de gráficos para outros formatos:
  - LaTeX
  - Xfig
  - Gif
- Outros exemplos
  - Gráficos para controle: `graficos_controle.sce`.
  - Gráficos 3D: `graficos_exemplos3di.sce` com `i=1,...,4`.
- Mais detalhes
  - *Introduction to Scilab* e manuais.

Exemplo: Série de Fourier para a onda quadrada

$$x(t) = \sin wt + \sin\frac{3wt}{3} + \sin\frac{5wt}{5} + \ldots + \sin\frac{(2n-1)wt}{2n-1}$$

- n: número de termos da série
- w: freqüência em rad/s (w = 2πT)

5.1. Aplicação: controle automático

- Solução de equações diferenciais ordinárias
  - `y = ode([tipo,] y0, t0, t, fc)`
  - `y0` : condições iniciais para y
  - `t0` : tempo inicial
  - `fc` : função externa ou lista, informa as derivadas
  - `[tipo]` : método de solução
    - `adams`: non-stiff predictor corrector Adams method (padrão)
    - `stiff`: stiff backward differentiation formula (BDF) (padrão)
    - `rk`: Runge Kutta adaptativo de ordem 4
    - `rkf`: Runge Kutta 4 e 5
    - Outros: `fix`, `root` e `discrete`.

Sintaxe de fc

- Equações do tipo $\dot{y} = f(y)$
  - fc é uma função do Scilab que define a derivada.
- Exemplo

$$\dot{y} = 2 \cdot y^2 + t \cdot \sin y$$

- A função fica definida como:

```
function dy = f(t,y)
   dy = 2*y^2 + t*sin(y)
endfunction
```

Sintaxe de fc (cont.)

- Equações do tipo $\dot{y} = f(y, u)$
  - fc é uma lista `list(f,u1,u2,...,un)` onde f é uma função do Scilab que define as derivadas e ui são funções do Scilab que definem as entradas para f.
- Exemplo
  - Equação diferencial

$$\dot{y} = 2 \cdot y^2 + y \cdot u(t)$$

  - Entrada

$$u(t) = 5 \cdot \sin(4t)$$

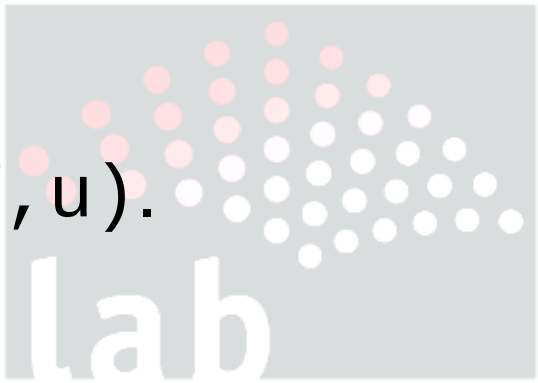
Sintaxe de fc (cont.)

- As funções são definidas como

```
function dy = f(t,y,u)
   dy = y^2 + y*u(t);
endfunction

function uc = u(t)
   uc = 5*sin(4*t);
endfunction
```

- E a chamada de fc fica sendo `list(f,u)`.



Exemplos

- Resolver a equação diferencial abaixo para $t \in [0,\pi]$ com y(0) = 0.

$$\dot{y} = y^2 - y \cdot \sin t + \cos t$$

- Resolver a equação diferencial abaixo para $t \in [0,1]$ com u(t) = sen (5t) e x(0) = [1 0]'.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot u$$

Funções de Controle Clássico

- Um sistema linear é definido por sua função de transferência ou suas matrizes de espaço de estados:
  - `[s1] = syslin(dom, A, B, C[, D[, x0]])`
  - `[s1] = syslin(dom, num, den)`
  - `[s1] = syslin(dom, H)`
- Exemplo

$$H(s)=\frac{0,2\cdot s^2+0,3\cdot s+1}{(s^2+0,4\cdot s+1)\cdot(s+0,5)}$$

Análise clássica

- Mostrar pólos e zeros
  - `trfmod(s1[, job])`
- Traça o *root locus*
  - `evans(s1)`
- Transformação em espaço de estados e vice-e-versa
  - `s2 = tf2ss(s1)` e `s3 = ss2tf(s2)`
- Autovalores da matriz do espaço de estados do sistema (pólos)
  - `spec(s2.A)`

Resposta temporal

- Simulação (resposta temporal) de um sistema linear
  - `[ y[, x] ] = csim( u, t, s1[, x0])`
- u é a entrada de controle que pode ser
  - uma função `[inputs] = u (t)`
  - uma lista `list(u1, p1, p2, ...)`, onde `[inputs] = u ( t, p1, p2, ...)`
  - A palavra `impuls` para a resposta ao impulso
  - A palavra `step` para a resposta ao degrau

Análise no domínio da freqüência

- Cálculo da resposta em freqüência
  - `[x] = freq(A, B, C[, D], f)`
  - `[x] = freq(num, den, f)`
- Diagrama de Bode
  - Amplitude em dB e fase da resposta em freqüência
  - `bode(s1)`
- Margem de ganho e margem de fase
  - `g_margin(s1)`
  - `p_margin(s1)`

Análise no domínio da freqüência (cont.)

- Diagrama de Nyquist
  - Parte real versus parte imaginária da resposta em freqüência
  - `nyquist(s1)`
- Carta de Nichols
  - `black(s1)`

Projeto de controladores

- Locação de pólos para realimentação total de estados
  - `ppol(s1.A, s1.B, polos)`
- Controlador LQR
  - `lqr2stan(s1, Q, R)`
- Controlador ótimo pelo critério da norma H_2
  - `lqr(s1)`

5.2. Aplicação: processamento de sinais

- Diversas funções para processamento de sinais.

Exemplo: FFT

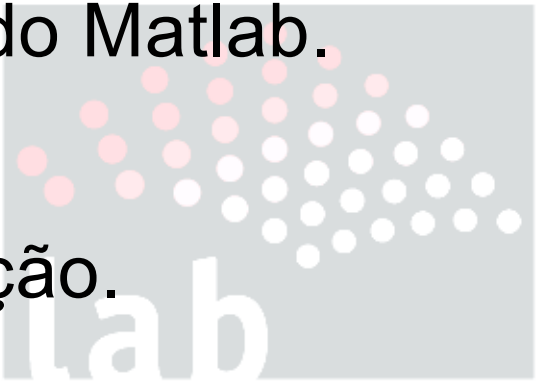
- Uso da FFT para análise de sinais estocásticos no domínio da freqüência
- Definir sinal senoidal
  - `t = (0:0.1:20)';`
  - `x = sin(3*t)/2;`
- Criar sinal de ruído usando números aleatórios
  - `r = rand(x);`
- Adicione o ruído ao sinal x
  - `xr = x + r;`

Exemplo: FFT (cont.)

- Completar xr para 1024 pontos (sinal xrf) e tomar a transformada de Fourier
  - `Xrf = fft(xrf, 1)`
- Observar o pico correspondente ao sinal senoidal

6. Outras aplicações

- Controle Automático
  - LMI, controle robusto, modelagem ARMA, identificação etc.
- Processamento de sinais
  - Communications Toolbox, arquivos de som etc.
- Otimização
- Interface com programas C e Fortran.
- Tradutor para programas e toolboxes do Matlab.
- Metanet: manipulação de grafos.
- Scicos: diagramas de blocos e simulação.



7. Conclusões

- O Scilab é uma ferramenta para o projeto de sistemas em controle automático e processamento de sinais.
- É um software livre.
- Diversas funcionalidades somadas à possibilidade de expansão na forma de bibliotecas.
- Opção ao uso do Matlab tanto no meio acadêmico quanto na indústria.

Muito Obrigado!

Contato:

aecc@epq.ime.eb.br