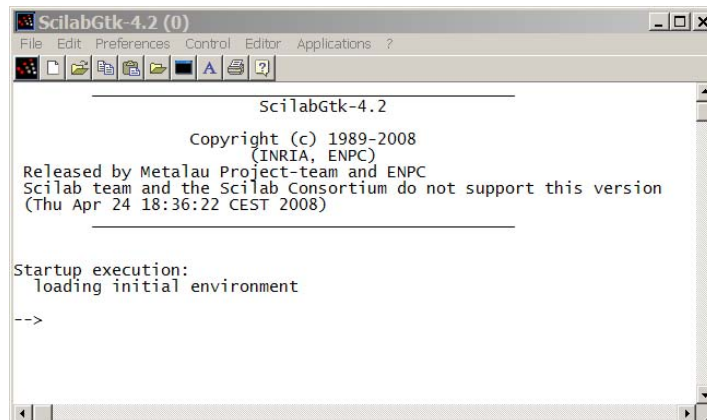


Scilab é um ambiente para programação, cálculo numérico e traçado de gráficos, distribuído gratuitamente, desenvolvido pelo Scilab group, constituído principalmente por pesquisadores do Institut Nationale de Recherche en Informatique et en Automatique - INRIA e do Ecole Nationale des Ponts et des Chaussées - ENPC da França. Pode ser baixado de <http://www.scilab.org>.

Scicos é um *toolbox* do Scilab para modelamento e simulação de sistemas dinâmicos e permite o uso de editores de diagrama de blocos.

Os exemplos deste texto foram testados com o ScilabGtk 4.2.



1. Ajuda: na dúvida, digitar "help" no prompt -->

```
--> help
--> help sin // help mais específico sobre o uso de sin
```

2. Alguns comandos básicos

```
--> [%pi %e %eps 5+7*%i %t %f] // algumas constantes pre-definidas
ans =
```

```
3.1415927    2.7182818    2.220D-16    5. + 7.i    1.    0
```

```
--> mickey=atan(sqrt(sin(5)^2+cos(5)^2))*180/%pi
mickey =
```

```
45.
```

```
--> x=7; // acrescentando ";" desabilita display, mas 7 foi atribuido a x
```

```
--> x // verificando se x esta com valor 7 mesmo...
```

```
x =
```

```
7.
```

```
--> a=rand(2,4) // matriz aleatória 2 x 4
a =
```

```
0.2113249    0.0002211    0.6653811    0.8497452
0.7560439    0.3303271    0.6283918    0.6857310
```

```
--> a=1:0.4:5 // matriz linha com elementos 1 a 5 de 0.4 em 0.4
a =
```

```
1.    1.4    1.8    2.2    2.6    3.    3.4    3.8    4.2    4.6    5.
```

```
--> matriz_a=[1 2 3 ; 6 5 4 ; 9 7 8]
matriz_a =
```

```

1.    2.    3.
6.    5.    4.
9.    7.    8

```

```

--> b = [ 1 2 3 ]
b =

```

```

1.    2.    3.

```

```

--> c=matriz_a*b' // b' denota transposto de b
c =

```

```

14.
28.
47.

```

```

--> jacarei=inv(matriz_a)*b' // inv(a) é a matriz inversa de a
jacarei =

```

```

- 0.0476190
0.3809524
0.0952381

```

```

--> pinda=spec(matriz_a) // spec(a) é um vetor com os auto-valores de a
pinda =

```

```

14.847652
- 1.6863603
0.838708

```

```

--> guara=[eye(2,2) 15*ones(2,2);linspace(80,74,4);logspace(1,3,4); zeros(1,4)]
// como montar matrizes mais complicadas a partir de submatrizes mais simples
guara =

```

```

1.    0.    15.    15.
0.    1.    15.    15.
80.   78.   76.    74.
10.   46.415888 215.44347 1000.
0.    0.    0.    0.

```

```

-->poli_caract=det(s*eye(3,3)-a) // modo facil de obter polinomio caracteristico
poli_caract =

```

```

      2    3
- 6 + 11s - 6s + s

```

```

--> who // lista as entidades já definidas

```

| Name | Type | Size | Bytes |
|----------|----------|---------|-------|
| guara | constant | 4 by 4 | 144 |
| ans | constant | 1 by 5 | 56 |
| linspace | function | | 1744 |
| pinda | constant | 3 by 1 | 64 |
| jacarei | constant | 3 by 1 | 40 |
| c | constant | 3 by 1 | 40 |
| b | constant | 1 by 3 | 40 |
| matriz_a | constant | 3 by 3 | 88 |
| a | constant | 1 by 11 | 104 |

```

--> clear // verifique o que aconteceu apos o clear usando who

```

```

Marizes: rank, inv, cond, det, spec, trace, svd, norm,...

```

```

3. Polinômios: degree, numer, denom, poly, varn, coeff, pdiv, ldiv, simp, horner,
derivat, intpoly, roots,... (tente help)

```

```

--> y=%s // y foi definido como o polinomio 1*s^1

```

```

y =

s

--> quarto_grau=y^4+3*y^3+7*y^2+4*y // lembrar que y = 1*s^1
quarto_grau =

      2      3      4
4s + 7s + 3s + s

--> outro_polin=poly([1 2 3 4 5 6],"ty","coeff") // usando poly e coeficientes
outro_polin =

      2      3      4      5
1 + 2ty + 3ty + 4ty + 5ty + 6ty

--> polin_raizes=poly([1 1 1 5],"z","roots") // usando poly e raizes
polin_raizes =

      2      3      4
5 - 16z + 18z - 8z + z

--> roots(polin_raizes) // verificando se as raizes estao OK
ans =

1.00000001
0.99999999
1.00000000
5.

--> mais_polin=(z-1)^3*(z-5) // podem ser feitas operações com polinômios
mais_polin =

      2      3      4
5 - 16z + 18z - 8z + z

--> roots(quarto_grau) // roots() calcula as raízes do polinômio
ans =

0
- 0.7537338
- 1.1231331 + 2.0113392i
- 1.1231331 - 2.0113392i

```

4. Variáveis Lógicas

```

-->cos(%pi) > 1 // deve resultar Falso = F
ans =

F

--> cos(%pi)<1 // deve resultar Verdadeiro = T
ans =

T

```

5. Funções

```

--> function z=bodosum(x,y) ; z=sin(x*pi)*cos(y*pi) ; endfunction
--> ola=bodosum(0.5,0.6)
ola =

- 0.3090170

--> function z=mais_bodosum(x,y) // como usar "if then else"
--> if x>=0 then z=sin(y); else z=log(-x); end;

```

```

--> endfunction
--> ole=mais_bodosum(-%e)
ole =

    1.

--> function zmatr=ainda_mais_bodosum()
--> zmatr=rand(3,3)
--> endfunction

--> ainda_mais_bodosum
ans =

    0.8782165    0.6623569    0.5442573
    0.0683740    0.7263507    0.2320748
    0.5608486    0.1985144    0.2312237

--> function z=phatorial(x) // ilustra uso do "for"
--> z=1;
--> for k=2:x
--> z=z*k;
--> end
--> endfunction

--> function z=phactorial(x) // ilustra uso do "while"
--> k=1; z=1;
--> while k<= int(x)
--> z=z*k;
--> k=k+1;
--> end
--> endfunction

--> function z=phactor(x) // ilustra possibilidade de "recursao"
--> if x==1 then z=1;
--> else z=x*factor(x-1);
--> end
--> endfunction

--> [phatorial(5) phactorial(7) phactor(9)]
ans =

    120.    5040.    362880.

--> deff(' [graus]=converte(radianos)', 'graus=radianos*180/%pi')
// outra sintaxe para definir funcoes.
--> converte(1) // 1 rad = 57.3 graus aproximadamente...
ans =

    57.29578

--> scipad(); // a função pode ser armazenada no disco. Para editar, chamar o pad

    Digitar na janela do pad:
function [r,fi]=trapolar(x,y)
r=sqrt(x^2+y^2)
if x==0 then fi=%pi/2; else fi=atan(y/x); end
endfunction
    Salvar, por exemplo, com o nome trapolar.sci

--> exec('trapolar.sci') // usar dir, chdir, pwd para ir para o diretorio correto
--> [r,fi]=trapolar(1,1)
fi =

    0.7853982
r =

    1.414213

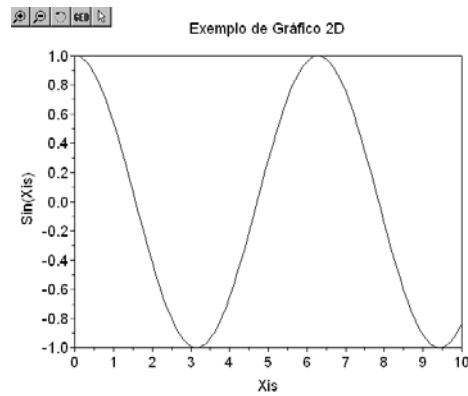
```

6. Impressão de dados

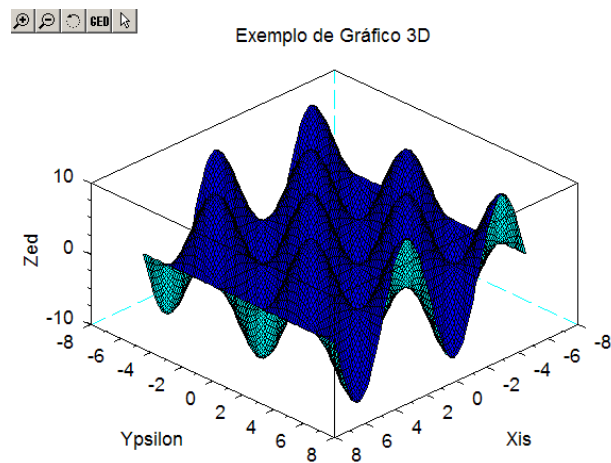
```
--> mprintf('formatados: \n %5d  %12.8f',[%pi %pi]) // saída de resultados  
formatados:  
      3      3.14159265
```

7. Gráficos

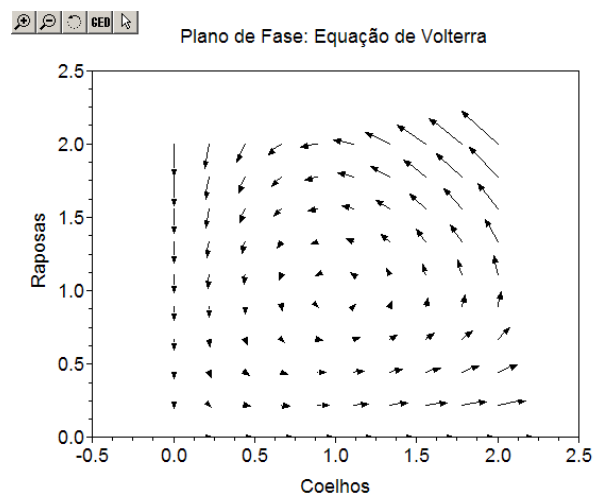
```
--> xis=0:0.1:10 ; plot2d(xis,cos(xis)) // graficos 2D
```



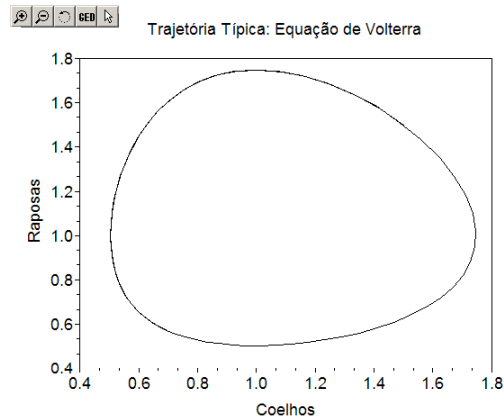
```
--> t=linspace(-2*pi,2*pi,100); plot3d(t,t,10*sin(t)'*cos(t)) // graficos 3D
```



```
--> function [xdot]=volterra(t,x); xdot=[x(1)-x(1)*x(2) ; -x(2)+x(1)*x(2)];  
endfunction  
--> x1=linspace(0,2,10);x2=x1;fchamp(volterra,0,x1,x2) // Campo de Vetores
```



```
--> t=0:0.1:10 ; y=ode([1 0]',0,t,volterra) // solucao de EDO
```



8. Controle Clássico

```
--> s=%s; num=25 ; den=s^2+4*s+2
--> gs=num/den // função de transferencia gs
gs =
```

$$\frac{25}{25 + 4s + s^2}$$

```
--> sistema=syslin('c',gs) // define sistema continuo a partir de gs
sistema =
```

$$\frac{25}{25 + 4s + s^2}$$

```
--> [a,b,c,d]=abcd(sistema) // recupera matrizes a,b,c e d do sistema
```

$$\begin{aligned} d &= 0. \\ c &= 0.5 \quad 0. \\ b &= 0. \\ &= 5. \\ a &= 0. \quad 10. \\ &= -2.5 \quad -4 \end{aligned}$$

```
--> sist_estados=syslin('c',a,b,c,d) // definir sistema a partir de a,b,c,d
```

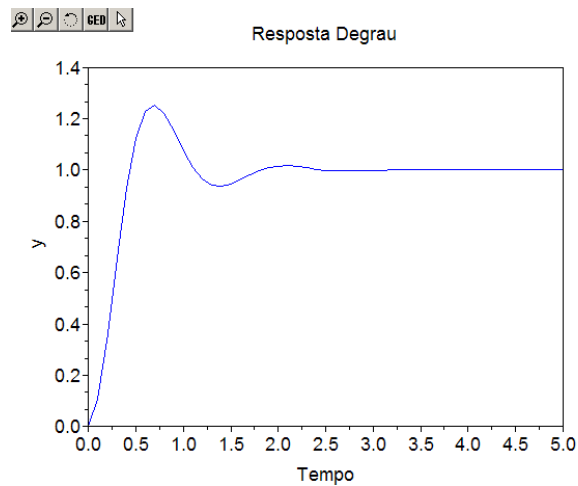
```
--> sis_tf=ss2tf(sist_estados) // verificar se gs foi recuperado
sis_tf =
```

$$\frac{25 - 6.661D-16s}{25 + 4s + s^2}$$

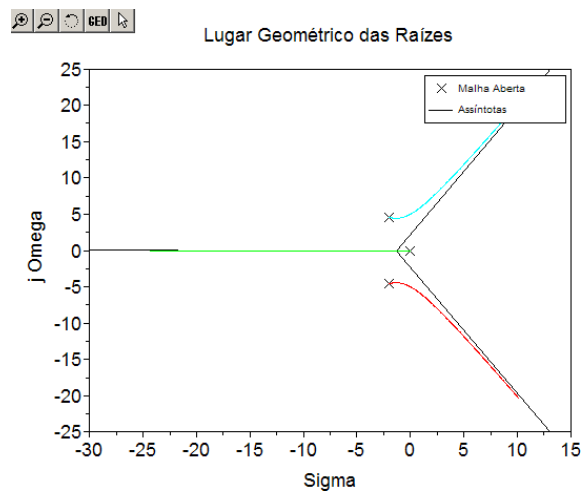
```
--> sys_disc=horner(sistema,(z+1)/(z-1)) // transformação bilinear s=(z+1)/(z-1)
sis_disc =
```

$$\frac{25 - 50z + 25z^2}{22 - 48z + 30z^2}$$

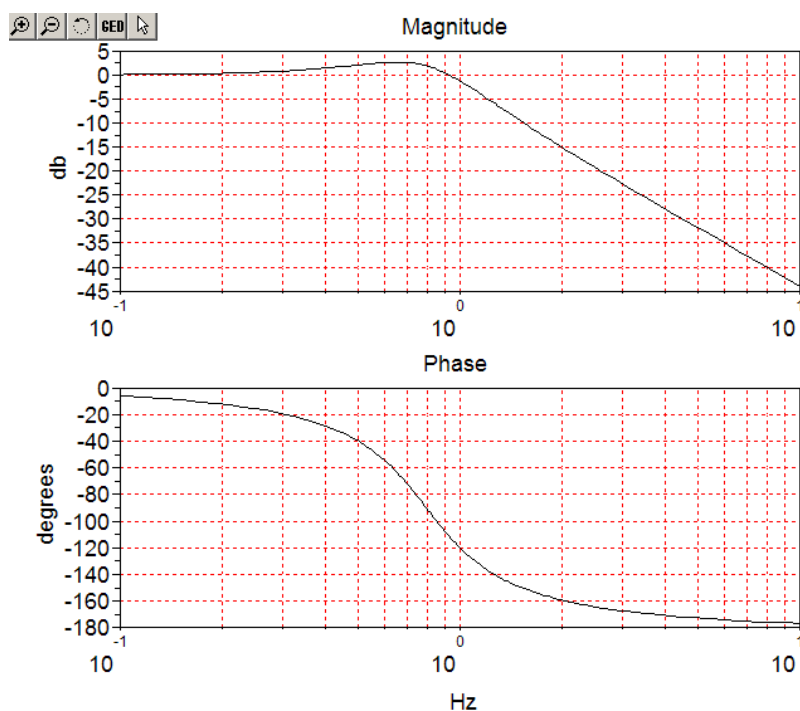
```
--> t=0:0.1:5 ; y=csim('step',t,sistema) ; plot(t,y) // resposta degrau
```



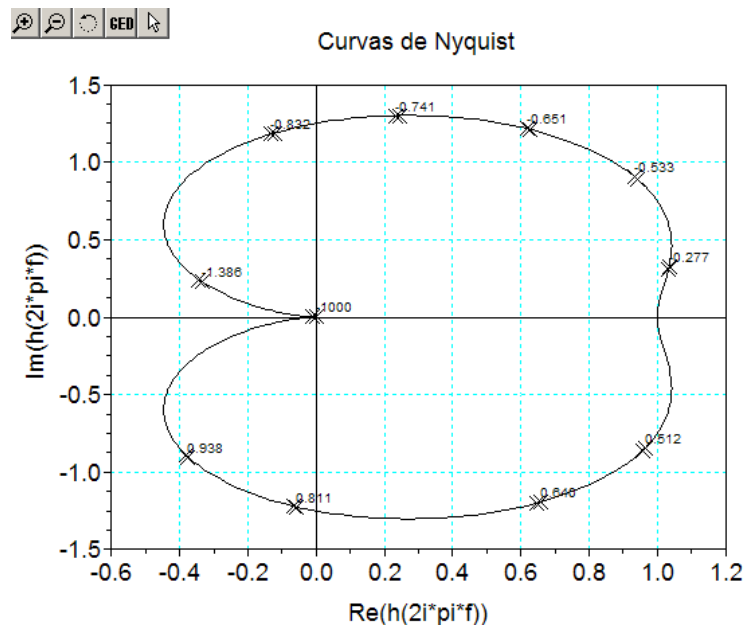
```
--> den=s^3+4*s^2+25*s ; malha_aberta=syslin('c',num/den) ; evans(malha_aberta)
```



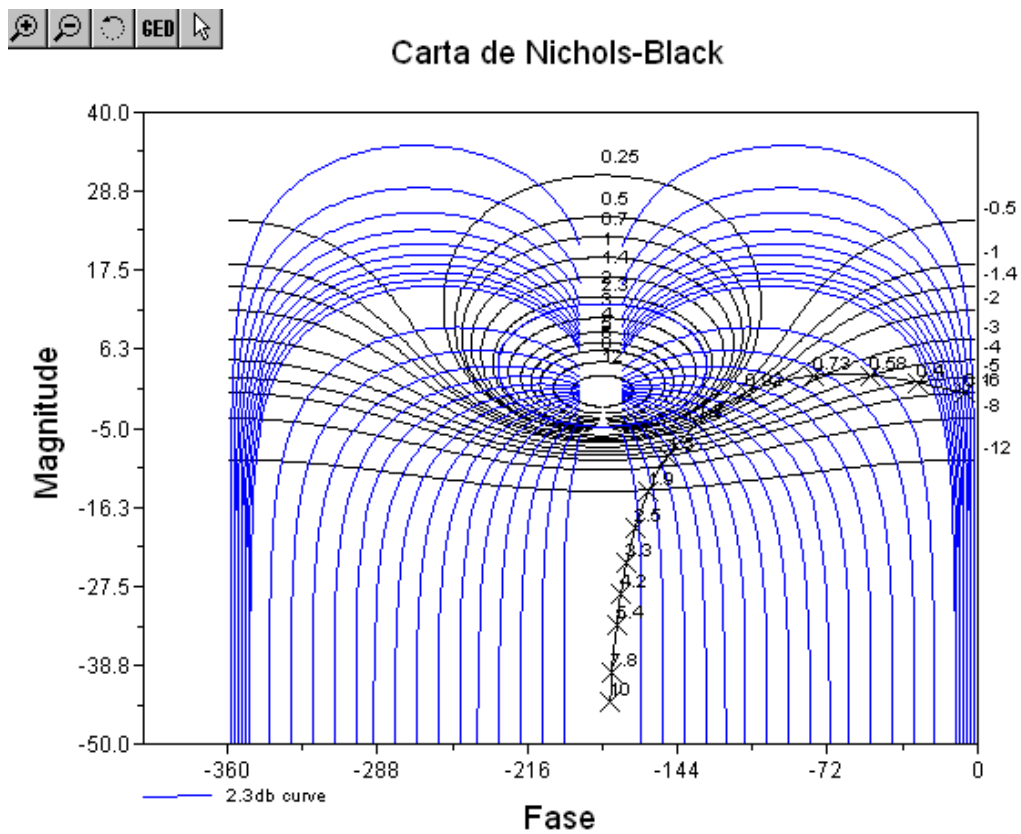
```
--> sistema=syslin('c',25/(s^2+4*s+25)) ; bode(sistema,0.1,10)
```



```
--> nyquist(sistema)
```



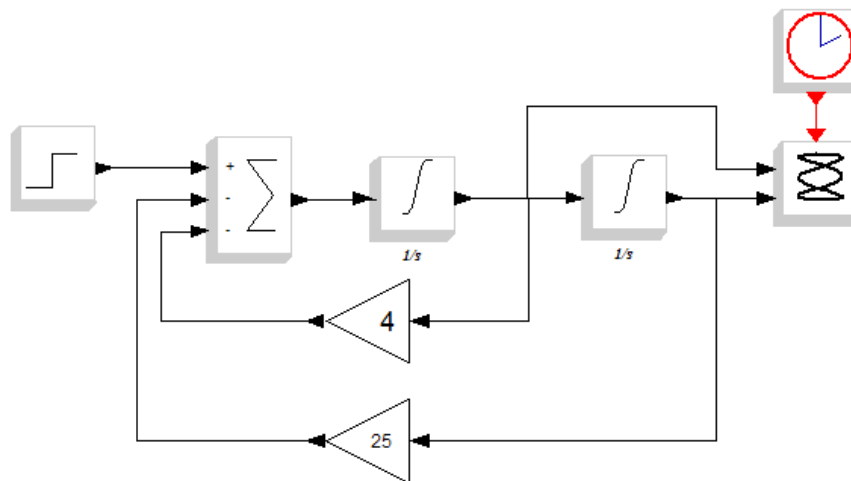
```
--> chart // traça a Carta de Nichols-Black
--> black(sys,0.1,10) // traça a resposta em frequencia Magnitude x Fase
```



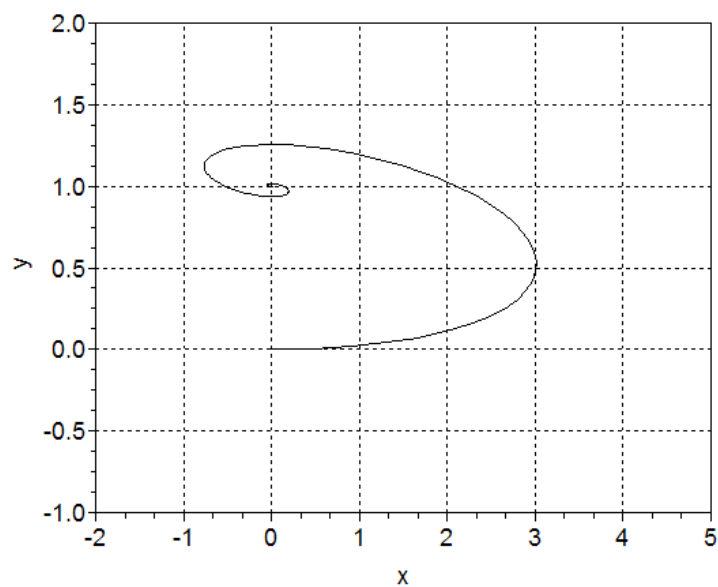
9. Simulação usando Scicos

Para ativar o Scicos, selecionar "Scicos" no botão "Applications" da barra do Scilab. Os diversos blocos podem ser copiados dos "Palette/Palletes" (Sinks: Visualizados $Y \times t$, $Y \times X$, enviar para a área de trabalho; Sources: Geradores de sinais diversos, transferir da área de trabalho, geradores de função tabulada; Linear: Integradores, Atrasadores, Equação na forma de espaço de estados, funções de transferência num/den, somadores, ganhos)

(i) Sistema Linear de Segunda Ordem



Simulação Scicos



(ii) Equação de Van der Pol { $a = 2$, IC = (0.1,0) }

