 <i>Instituto Nacional de Telecomunicações</i>	2º Relatório	Turma: C213__	Data:
	Controle de sistemas dinâmicos		
Nome:			

Plotagem de gráficos 2D

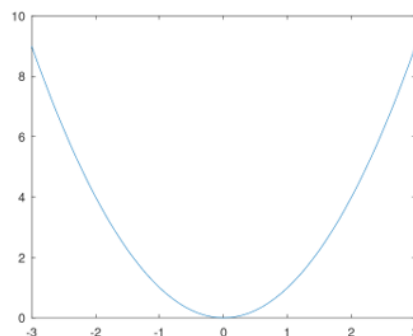
1) Plotagem de gráficos no *MatLab* e no *Octave*

Um dos recursos disponíveis no *MatLab* e no *Octave* que facilitam a visualização de resultados é a plotagem de gráficos. Um gráfico possibilita analisar a resposta de um sistema a diversos estímulos de entrada de forma dinâmica.

1.1) Como criar um gráfico

Os passos para se plotar um gráfico podem ser divididos em: preparação dos dados, chamada da função que define o gráfico e a configuração da aparência desejada para a figura. Os comandos a seguir ilustram cada um dos passos:

```
>> %Preparacao dos dados
>> x = -3:0.1:3;
>>
>> %Definicao da lei de formacao
>> y = x.^2;
>>
>> %A lei de formacao define uma parabola
>> plot(x,y)
>>
```



Na etapa de preparação dos dados o vetor *X* exige certa atenção, pois, na plotagem dos gráficos, os *softwares* desenharam a forma da figura valor a valor. Dessa forma, se o passo entre um valor e outro do vetor *X* for muito grande o gráfico pode ficar distorcido ao invés de respeitar a curva esperada. Além disso, um passo muito pequeno pode resultar em um vetor com uma quantidade elevada de valores, resultando em um erro na criação do vetor na janela de comandos. Logo o passo entre os valores do vetor deve ser preciso, geralmente 0.1 é um valor adequado.

Na definição da lei de formação da função que será plotada, se *X* é um vetor *Y* também será um vetor. É imprescindível que os dois vetores sejam do mesmo tamanho para que não aconteça nenhum erro na plotagem.

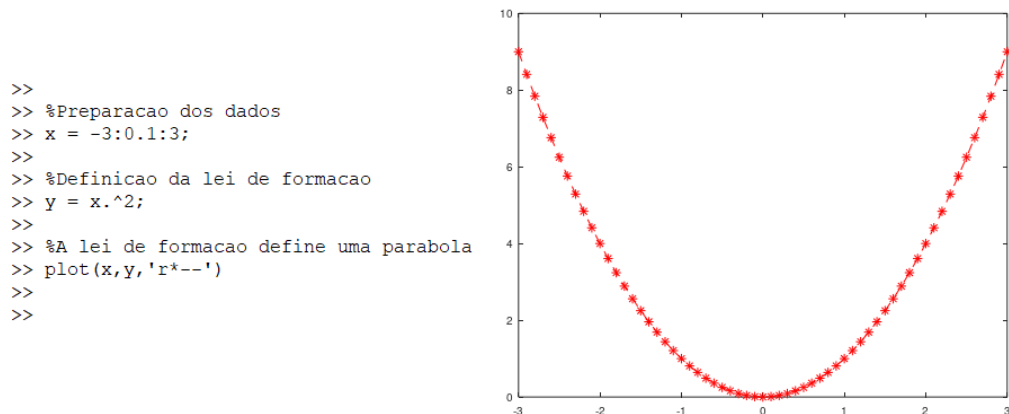
O vetor *X* é o vetor da entrada, logo é o ponto em que o tamanho dos vetores pode ser determinado. Já o tamanho do vetor *Y* depende do tamanho do vetor *X*. Quando uma lei de formação que define *Y* em função de *X* é criada, deve ser passado ao *software* que ele deve usar todos os valores do vetor *X* para criar o vetor *Y*. Por isso, na definição da lei de formação o vetor *X* aparece seguido de um ponto, sendo que o ponto após o nome do vetor é quem determina que todos os valores do vetor sejam usados para criar o vetor *Y*.

1.2) Configuração da aparência do gráfico

Além de plotar o gráfico de Y em função de X, algumas configurações podem ser feitas nos comandos para ajustar a aparência do gráfico, como, por exemplo, a cor e o tamanho da linha, deixar a linha em outro formato, alterar para linha pontilhada etc. A tabela a seguir ilustra comandos que podem ser utilizados como parâmetro no *plot* para alterar a aparência do gráfico:

Cor da linha	Tipo de marcador	Traçado da linha
y (amarelo)	. (ponto)	: (pontilhada)
m (magenta)	o (círculo)	-. (ponto – traço)
c (azul claro)	x (x's)	-- (tracejada)
r (vermelho)	+ (cruz)	- (sólida)
g (verde)	* (estrela)	
b (azul)	s (quadrado)	
k (prata)	d (losango)	
	v (triângulo p/ baixo)	
	^ (triângulo p/ cima)	
	< (triângulo p/ esquerda)	
	> (triângulo p/ direita)	
	p (pentagrama)	
	h (hexagrama)	

O código a seguir mostra um exemplo em que a aparência do gráfico foi alterada para apresentar uma linha vermelha tracejada, com marcações em estrela:



Caso a linha esteja fina, a propriedade *linewidth* altera a espessura da linha. O padrão do *MatLab* e do *Octave* plota os gráficos com linha de espessura 1. O comando segue a estrutura:

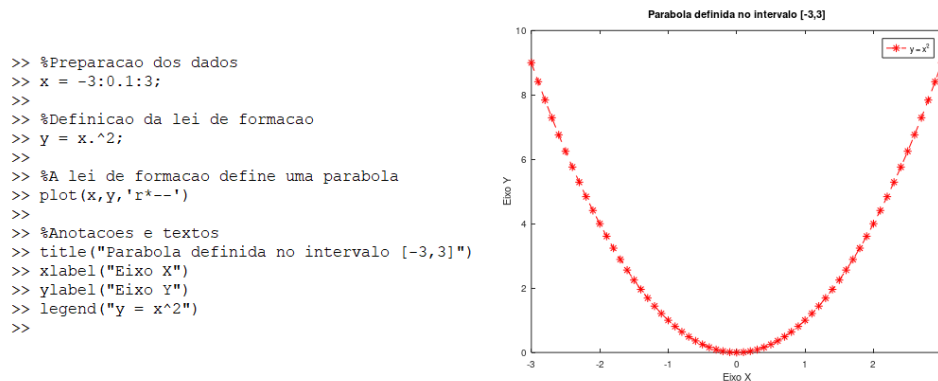
```
>> plot(x,y,'r*--','linewidth',tamanho)
```

Além das configurações de aparência, a figura plotada também pode ter suas propriedades alteradas, colocando descrições em forma de texto no gráfico:

Propriedade	Sintaxe
	>> title("texto")

Inserir um título do gráfico	
Configura os eixos x, y, z	<code>>> eixo_label("texto")</code>
Coloca legenda no gráfico	<code>>> legend("texto 1", "texto 2", ...)</code>
Habilita grade	<code>>> grid on</code>

O gráfico a seguir é o mesmo do anterior com legenda e títulos inseridos:

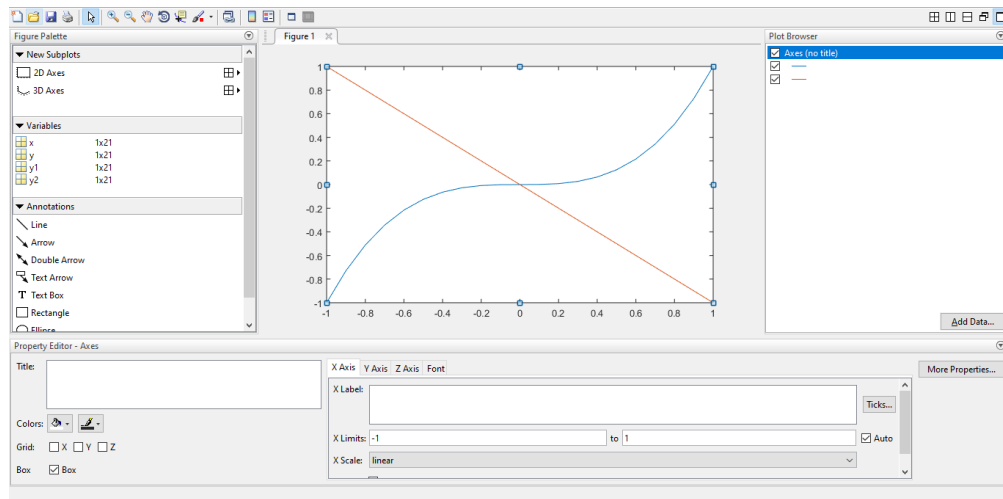


Os parâmetros de texto não permitem caracteres especiais nos comandos.

No *MatLab*, há uma funcionalidade disponível que permite alterar o gráfico manualmente. Para isso há um botão chamado “*Show Plot Tools and Dock Figure*”.



Ao clicar no botão a janela a seguir é aberta e as propriedades do gráfico podem ser alteradas, sendo possível inserir títulos, nomear os eixos, legendas, grid etc.



1.3) Mini janela de zoom

Além da configuração das propriedades, é possível inserir uma janela no gráfico para que seja criado um pequeno “*box*” de zoom da região plotada. Para isso, a função `axes` cria um eixo sobre o do gráfico já existente. Tem-se para essa função:

```

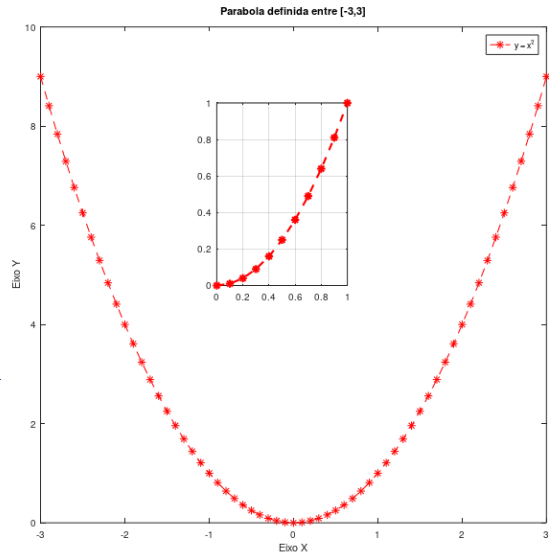
>> axes('Position', [xorigem yorigem comprimento_x comprimento_y])

```

```

>> %Preparacao dos dados
>> x = -3:0.1:3;
>>
>> %Definicao da lei de formacao
>> y = x.^2;
>>
>> %A lei de formacao define uma parabola
>> plot(x,y,'r*--')
>>
>> %Anotacoes e textos
>> title("Parabola definida entre [-3,3]")
>> xlabel("Eixo X")
>> ylabel("Eixo Y")
>> legend("y = x^2")
>>
>> %Mini janela de zoom
>> axes('Position', [0.4 0.5 0.2 0.3])
>>
>> %Plote o grafico novamente, agora na janela
>> plot(x,y,'r*--','linewidth',1.5)
>>
>> %Defina os limites de x e y na janela
>> xlim([0 1])
>> ylim([0 1])
>>
>> grid on

```



1.4) Visualização de mais de um gráfico ao mesmo tempo

Em algumas situações, será necessário plotar mais de um gráfico ao mesmo tempo para que seja possível comparar a resposta de dois sistemas distintos em relação a uma entrada. Além disso, alguns tipos de gráficos não permitem que outros gráficos sejam plotados dentro da mesma figura. Para isso, alguns comandos permitem que isso seja feito.

Em casos gerais, a própria função *plot* é capaz de mostrar mais de um gráfico. Para isso, basta utilizar os vetores correspondentes, como no exemplo a seguir. Vale ressaltar que os eixos devem ser passados à função, mesmo quando o vetor do eixo X for o mesmo para todos os gráficos.

```

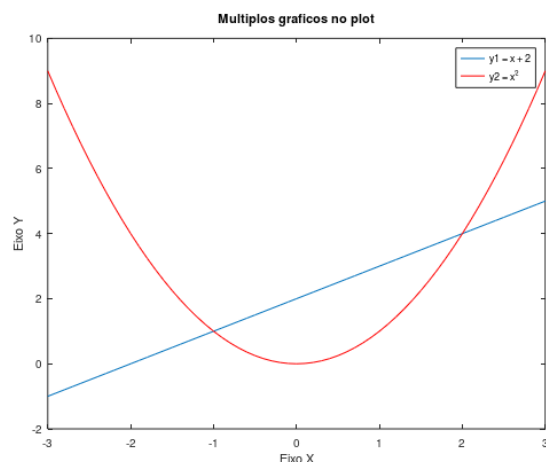
>> plot(x1,y1,'formato1',x2, y2,'formato2',...)

```

```

>> x = -3:0.1:3;
>> y1 = x.*1 + 2;
>> y2 = x.^2;
>>
>> plot(x, y1, x, y2, 'r')
>>
>> title("Multiplos graficos no plot")
>> xlabel("Eixo X")
>> ylabel("Eixo Y")
>> legend("y1 = x + 2", "y2 = x^2")

```



O formato do gráfico é opcional, sendo que, se nada for colocado, o gráfico será plotado no seu formato padrão: azul, com espessura da linha igual a 1 etc. Mas, como são muitos gráficos, é aconselhável que as formas tenham características diferentes e que cada curva seja identificada na legenda.

Nesse caso da função *plot*, os gráficos são gerados dentro de um mesmo comando. Para casos em que os gráficos sejam plotados em linhas de comando diferentes, o

comando *hold on* é suficiente. Essa função “segura” o gráfico que já está na figura aberta e não permite que um outro gráfico seja plotado por cima.

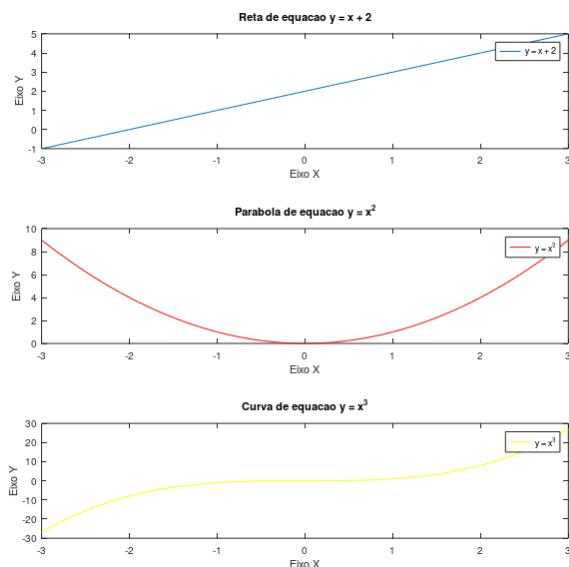
A função *plot* e o comando *hold on* conseguem plotar gráficos diferentes na mesma figura. Para outros gráficos mais específicos, esse tipo de ação não é suportado, pois a configuração da função faz com que ela seja a única na figura. Nessas situações, para plotar outros gráficos é necessário abrir outra figura com o comando *figure()*;

Outra forma de se fazer isso é com uma função chamada *subplot*. Essa função cria uma matriz de gráficos e exibe todos ao mesmo tempo na tela. Assim como o comando *hold on*, para alguns gráficos mais específicos a função *subplot* não funcionará. A sintaxe da função *subplot* é a seguinte:

```
>> subplot(linhas, colunas, posicao)
```

Nessa sintaxe, linhas e colunas representam a dimensão da matriz que será criada dentro da figura, e *posicao* é posição absoluta da matriz em que o gráfico será plotado. Veja o exemplo a seguir de uma figura plotada utilizando essa função:

```
>> x = -3:0.1:3;
>>
>> y1 = x.*1 + 2;
>> y2 = x.^2;
>> y3 = x.^3;
>>
>> subplot(3,1,1)
>> plot(x,y1)
>> xlabel("Eixo X")
>> ylabel("Eixo Y")
>> title("Reta de equacao y = x + 2")
>> legend("y = x + 2")
>>
>> subplot(3,1,2)
>> plot(x,y2,'r')
>> xlabel("Eixo X")
>> ylabel("Eixo Y")
>> title("Parabola de equacao y = x^2")
>> legend("y = x^2")
>>
>> subplot(3,1,3)
>> plot(x,y3,'y')
>> xlabel("Eixo X")
>> ylabel("Eixo Y")
>> title("Curva de equacao y = x^3")
>> legend("y = x^3")
```



1.5) Exportando os gráficos plotados

Os gráficos gerados podem ser exportados em formatos de arquivos compatíveis com o *software*. Isso será bastante útil para exportar os gráficos e enviá-los em dia de prova/tarefa. A tabela a seguir mostra alguns formatos de arquivo aceitos para exportar a imagem e o comando utilizado para fazer cada extensão:

Extensão	Comando
<i>MatLab figure</i> (abre em qualquer <i>MatLab</i>)	–
<i>Octave figure</i> (abre em qualquer <i>Octave</i>)	–
Formato PNG (não perde resolução)	<code>>> print -dpng graf.png</code>
Formato SVG (não perde resolução)	<code>>> print -dsvg graf.svg</code>
Formato JPG (não perde resolução)	<code>>> print -djpg graf.jpg</code>
Formato JPEG (não perde resolução)	<code>>> print -djpeg graf.jpeg</code>

Para salvar os arquivos nas extensões do *MatLab* – extensão *fig*, e do *Octave* – extensão *ofg*, basta ir em *Files* em clicar para salvar. Em todos os comandos de exportação, “*graf*” é o nome do arquivo exportado, podendo ser alterado na linha de código para um nome diferente. Para que o gráfico possa ser exportado a figura que contém o *plot* deve estar aberta. O arquivo exportado é inserido na pasta selecionada como o diretório atual.

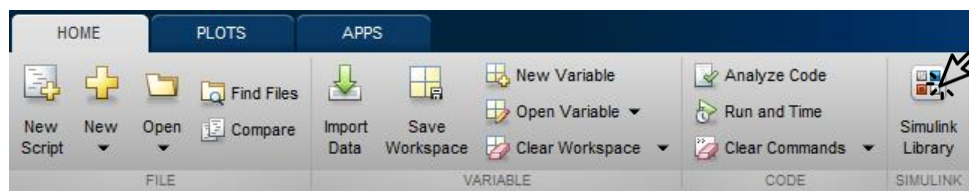
Para fechar o gráfico que foi aberto, a figura pode ser fechada diretamente na janela de visualização ou então com o comando *close* na janela de comandos.

2) Introdução aos gráficos no *Simulink*

2.1) O que é o *Simulink*

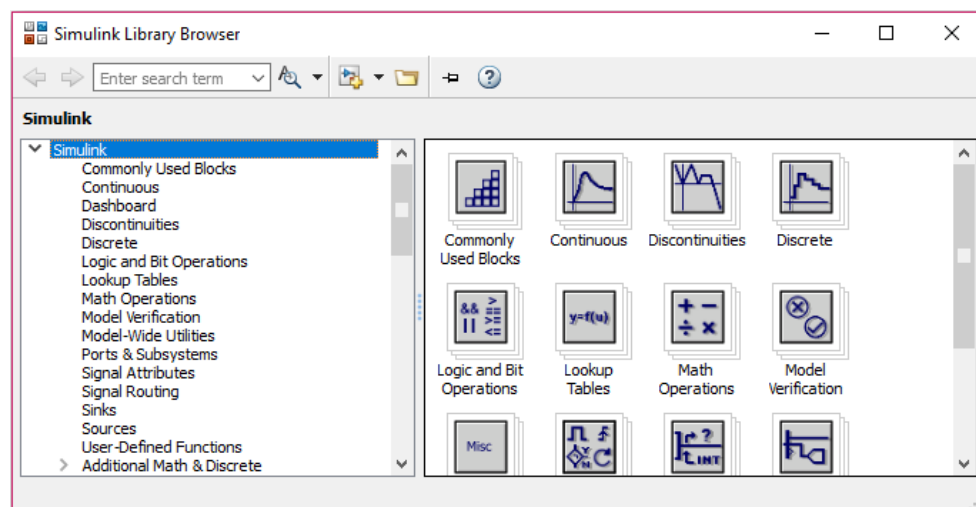
O *Simulink* é um aplicativo dentro do *MatLab* que permite analisar o comportamento de sistemas dinâmicos a partir da construção do modelo matemático do sistema utilizando diagrama em blocos simulados. Tal funcionalidade não existe dentro do *Octave*, sendo substituído pelo *software SciLab*.

Para acessar o *Simulink* dentro do *MatLab* digite o nome do aplicativo na janela de comandos ou então clique sobre o ícone do aplicativo na guia *HOME*:



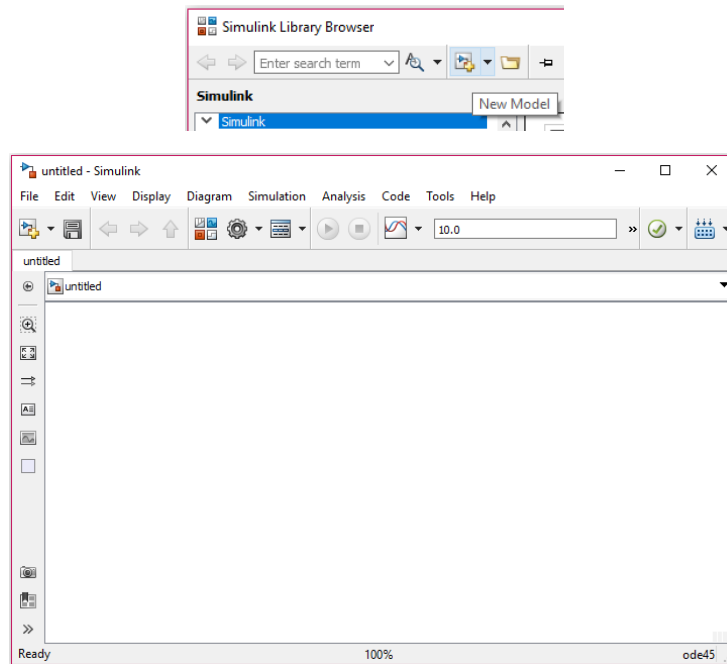
Ou então digite: >> simulink

Na janela inicial do *Simulink*, são exibidos os ícones que representam cada um dos diagramas em bloco que serão utilizados para compor os modelos dos sistemas. Os blocos são divididos por bibliotecas de acordo com suas funções.

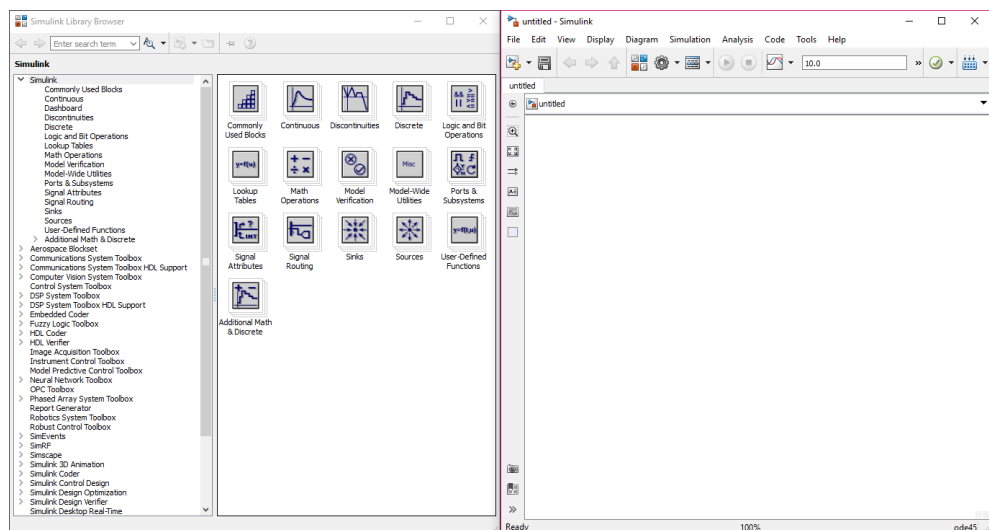


2.2) Criando o modelo de um sistema

Para fazer uma nova simulação é necessário criar um novo modelo e após isso uma janela vazia em que os blocos serão inseridos e o sistema será simulado.

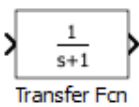









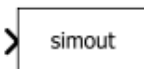




Como dica, deixe as duas abas abertas em uma tela dividida, uma com os blocos e outra com a janela de simulação, desta forma:



2.3) Principais blocos

A tabela a seguir, mostra alguns blocos importantes que serão muito utilizados ao longo do curso. Nessa listagem é mostrada em qual biblioteca se encontra cada bloco, qual a sua função e qual a representação do bloco:

Biblioteca	Função	Bloco
<i>Continuous</i>	Função de transferência	

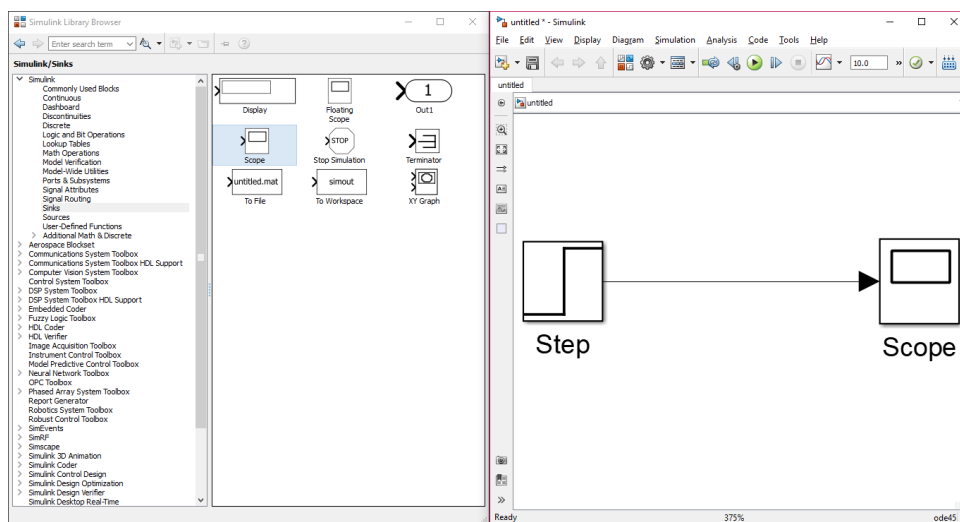
<i>Continuous</i>	Atraso de transporte	 Transport Delay
	Função Derivativa	 Derivative
	Função Integrativo	 Integrator
<i>Math Operations</i>	Ganho	 Gain
	Somador	 Sum
<i>Signal Routing</i>	Multiplexador	 Mux
<i>Sinks</i>	Mostra a saída do sistema	 Scope
	Saída do sistema no <i>Workspace</i>	 To Workspace
<i>Source</i>	Tempo de amostra para o <i>Workspace</i>	 Clock
	Sinal tipo rampa	 Ramp
	Sinal senoidal	 Sine Wave
	Sinal tipo degrau	 Step

Alguns blocos podem ter sua configuração alterada com um duplo clique sobre ele dentro da janela. Por exemplo, o bloco somador pode ser alterado para um bloco subtrator ao mudar a entrada + para -, e assim vale para outros blocos.

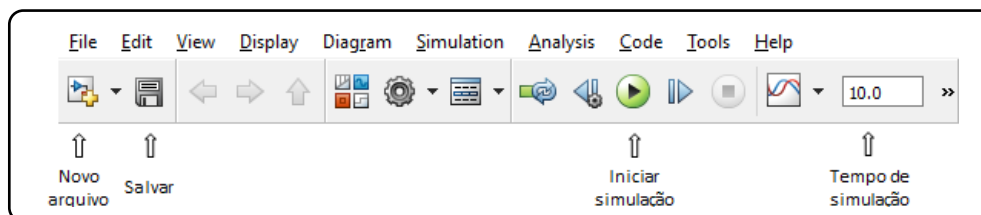
2.4) Simulação

Para mostrar como visualizar o gráfico na saída, será usada uma entrada simples, um degrau unitário. Na biblioteca **Sources** procure pelo bloco **Step**, clique e arraste o bloco para dentro da janela do modelo. Após o **Step**, na biblioteca **Sinks** procure pelo bloco **Scope**, clique e arraste para dentro da janela do modelo. Como dica, aperte duas vezes a barra de espaço para ajustar o modelo ao tamanho da tela.

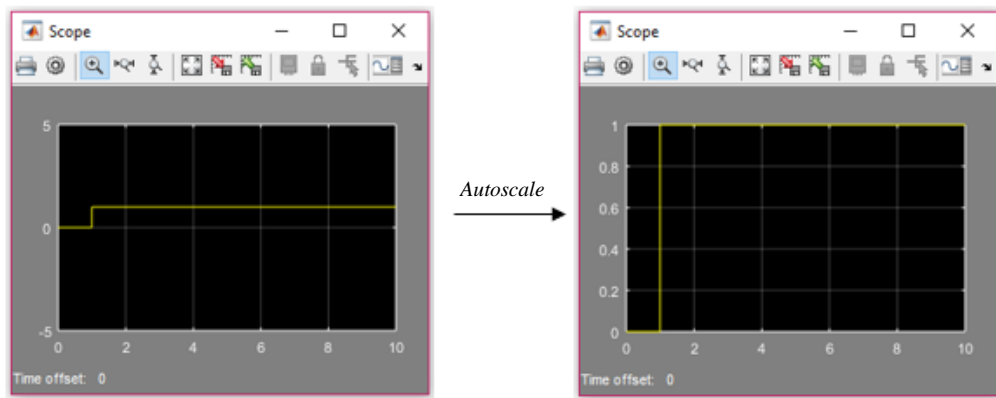
Para conectar os blocos, clique no ponto de saída do sinal do bloco (representado por uma flecha) com o botão esquerdo do *mouse* e arraste a conexão até a entrada do outro bloco. Faça isso ligando a saída do **Step** com a entrada do **Scope**.



O modelo criado pode ser salvo para que seja usado depois ou apenas para simulá-lo. Para isso, clique na opção **Save** no menu do *Simulink* ou então aperte **Ctrl + S**. Salve o arquivo com o nome que desejar, sendo que os arquivos do *Simulink* terão extensão *slx*. O menu de ferramentas do *Simulink* oferece opções para realizar essas tarefas de forma rápida:

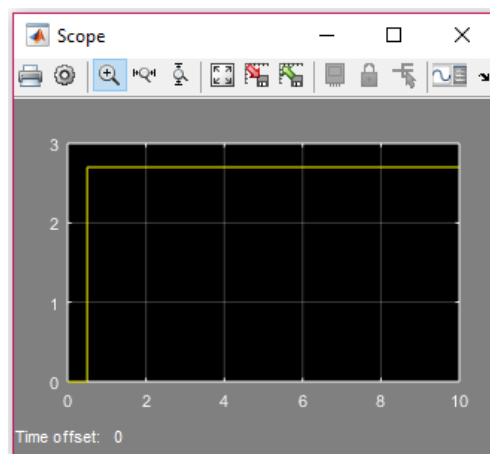
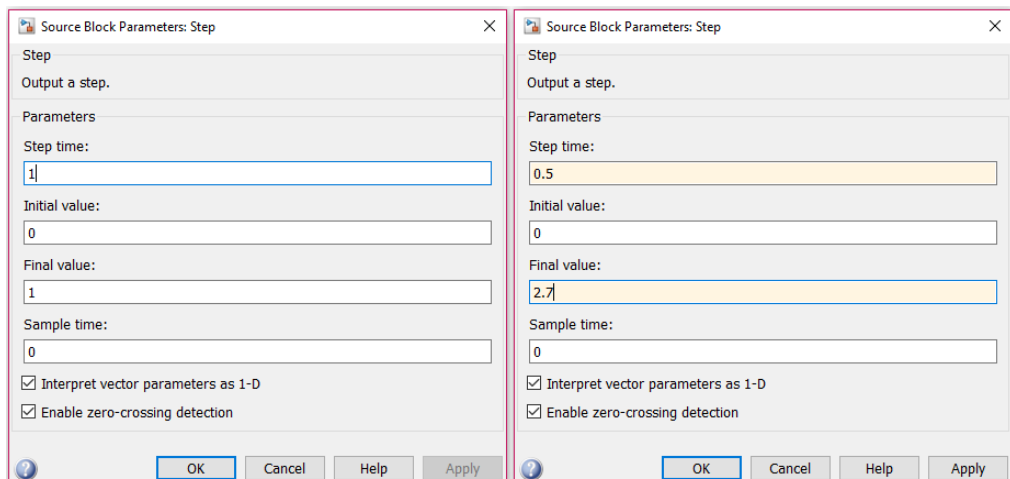


Para realizar a simulação do sistema montado, deve-se escolher o tempo de simulação (mostrado na nota acima) e clicar em **Run**. Para visualizar o sinal, dê um duplo clique no **Scope** e a resposta do sistema será mostrada na tela. Clique em **Autoscale** para ajustar o sinal automaticamente ao tamanho da janela.

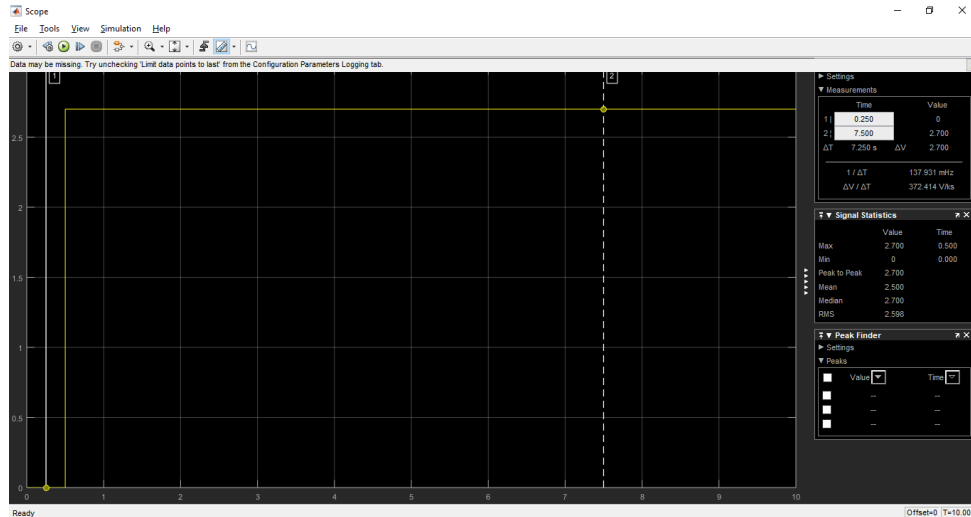


O sinal visto na tela do **Scope** mostra o padrão de uma entrada do tipo degrau: o início do bloco é em 1 segundo e a amplitude é unitária. Com um duplo clique sobre o bloco é possível alterar essas configurações na janela que é aberta. Como dito anteriormente, essa mudança de configurações serve para todos os blocos.

Veja no exemplo a seguir a saída do **Scope** com o tempo de início do sinal mudado para 0,5 segundos e a amplitude do degrau mudada para 2,7. Lembre-se que o **MatLab** é um *software* de linguagem universal, logo o operador decimal é o ponto, e não a vírgula.

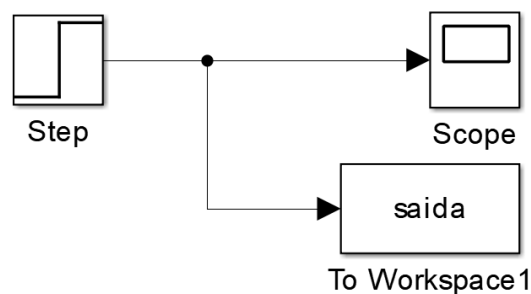


Há uma opção de cursor que ajuda na visualização dos valores de posição de um ponto qualquer sobre o gráfico no **Scope**. Para isso, clique na última opção na janela do **Scope**, chamada de “*Try Time Scope*”. No ícone “*Cursor Measurements*” as opções de medida podem ser utilizadas, basta escolher qual opção melhor satisfaz a necessidade.



2.4) Exportando gráficos para o *Workspace*

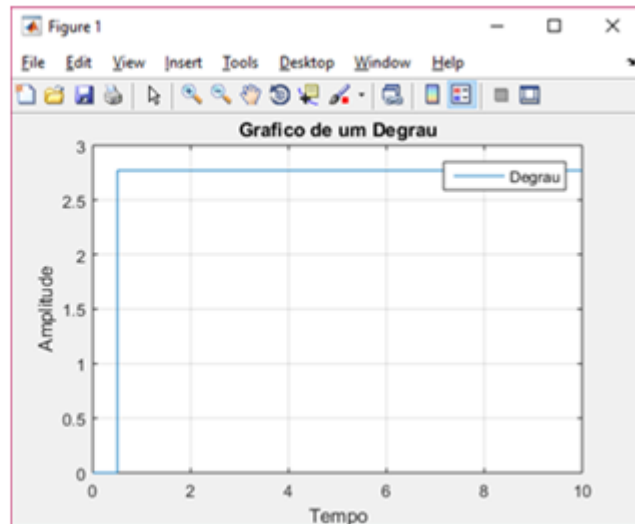
Na biblioteca **Sinks** um bloco chamado **To Workspace** pode melhorar os gráficos de resposta e exportar em arquivos com extensão jpg, png, ou outras extensões de arquivos de imagem. Ao arrastar o bloco para a janela do modelo, o nome do bloco pode ser alterado com um duplo clique sobre ele mudando o campo *Variable Name* na janela que é aberta. Veja o exemplo:



O bloco **To Workspace** armazena os dados referentes à saída do sistema ao longo da simulação, para isso deve ser configurado para salvar o resultado no formato de **Array**, que nada mais é do que um vetor. Ao alterar a configuração, clique em **OK** e execute novamente a simulação. Assim serão criados dois vetores, um deles terá o mesmo nome da *Variable Name* do bloco e o outro será o t_{out} .

O vetor t_{out} é criado automaticamente, e guarda o tempo da simulação. Com os vetores salvos no *Workspace*, o gráfico pode ser plotado na própria janela de comandos e suas configurações podem ser alteradas das formas vistas anteriormente.

```
>> plot(tout,saida)
>> grid on
>> legend('Degrau')
>> xlabel('Tempo')
>> ylabel('Amplitude')
>> title('Grafico de um Degrau')
```



3) Introdução ao *SciLab*

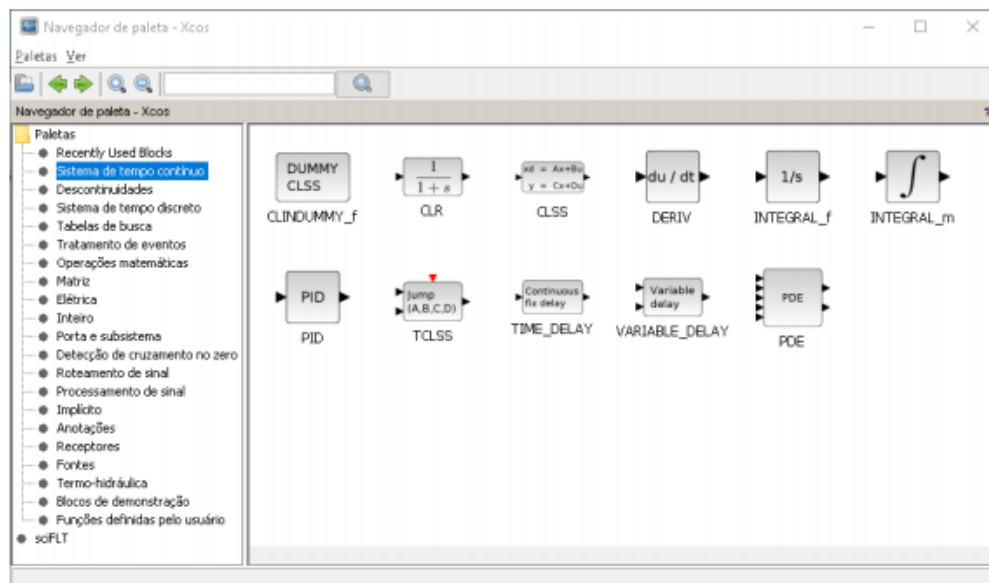
3.1) Ambiente Xcos

O programa Xcos dentro do *SciLab* permite a simulação de sistemas dinâmicos, tanto de natureza contínua quanto de natureza discreta. Além disso, possui recursos de pós-processamento gráfico que permitem ao usuário realizar a apresentação da resposta dinâmica do sistema. Funciona como a janela do modelo do *Simulink*.

Ao inicializar o *SciLab* clique sobre o ícone do programa Xcos na barra de ferramentas. Duas janelas são abertas: o ambiente de simulação e o navegador de paleta, que apresenta as bibliotecas com os blocos para a simulação.

3.2) Biblioteca de blocos

Assim como no *Simulink*, os blocos dentro do *SciLab* são divididos em bibliotecas organizadas de acordo com a sua função.

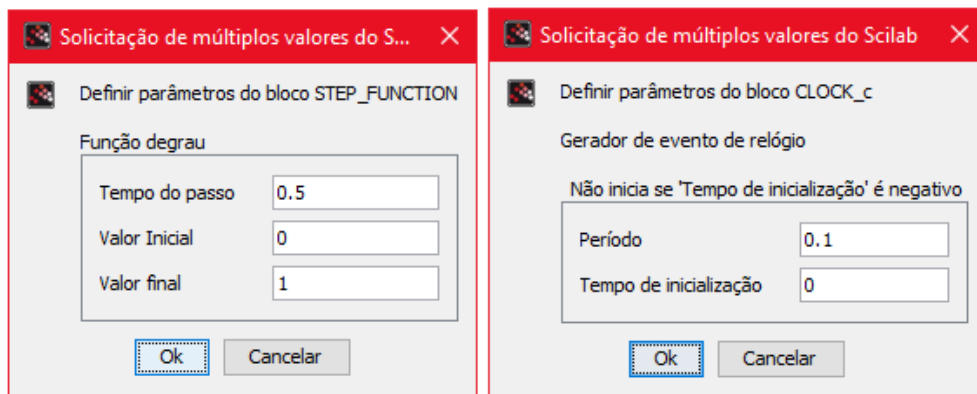


Para adicionar um bloco ao Xcos, clique duas vezes sobre ele e o bloco aparecerá na janela de simulação. Você pode arrastar ou redimensionar os blocos da forma que

desejar. Por meio das setas nas entradas e nas saídas dos blocos, conecte um bloco ao outro para formar o sistema a ser simulado.

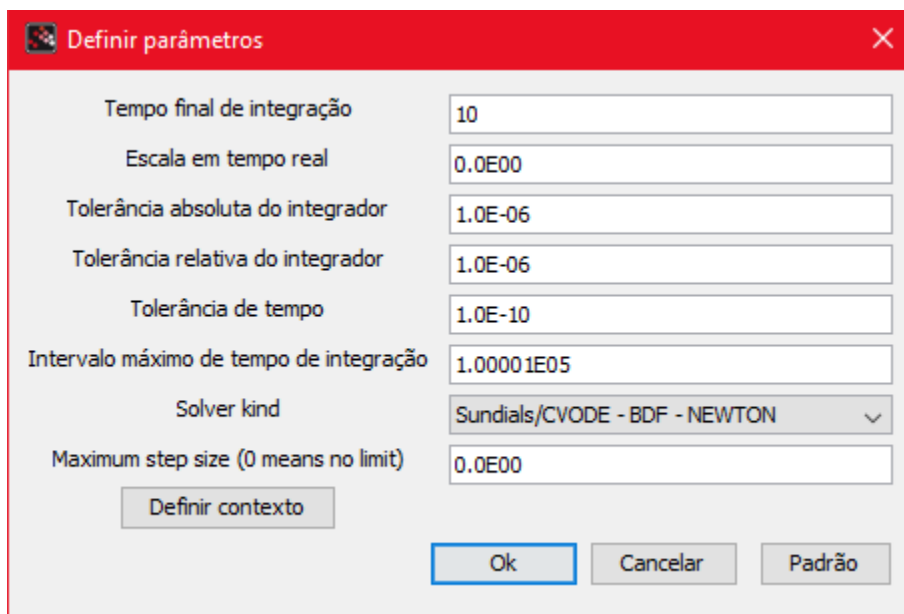
Assim como foi feito no *Simulink*, será simulado um degrau unitário. Na biblioteca de fontes, insira um bloco **Step** na janela da simulação. Na biblioteca de receptores, insira um **CScope** no ambiente. Diferente do *Simulink*, o *SciLab* necessita de um bloco de **Clock** para o **CScope**, então vá novamente na biblioteca de fontes e insira um bloco **Clock** no ambiente.

Os blocos apresentam parâmetros que podem ter seus valores mudados de acordo com a necessidade. Clique duas vezes sobre o bloco e mude o parâmetro que for necessário na janela que abrir.



Na configuração do **Clock** o valor do período é de suma importância. Esse valor simboliza qual será o intervalo de tempo entre os instantes em que um novo valor será gerado para a construção do gráfico. Dessa forma, se o valor do período for alto o gráfico terá uma forma distorcida. Valores bons para o período são 0,1 e 0,05.

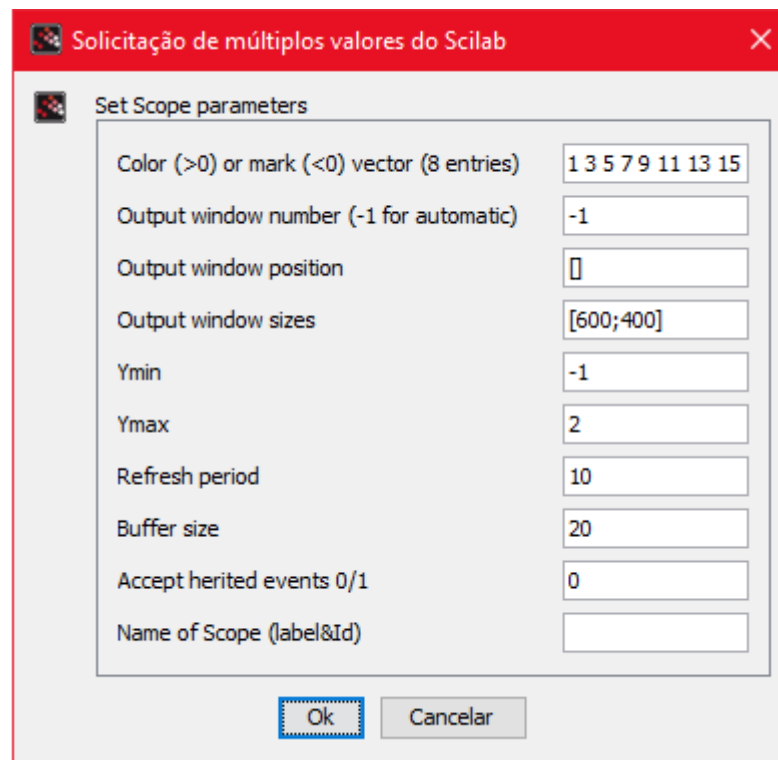
Outro parâmetro importante a ser configurado é o tempo de simulação. Na barra de opções do ambiente vá em simulação, e depois clique em configuração. Na janela que será aberta altere o parâmetro “Tempo final de integração”. Esse parâmetro corresponde ao tempo total que o sinal será mostrado na tela do **CScope**.



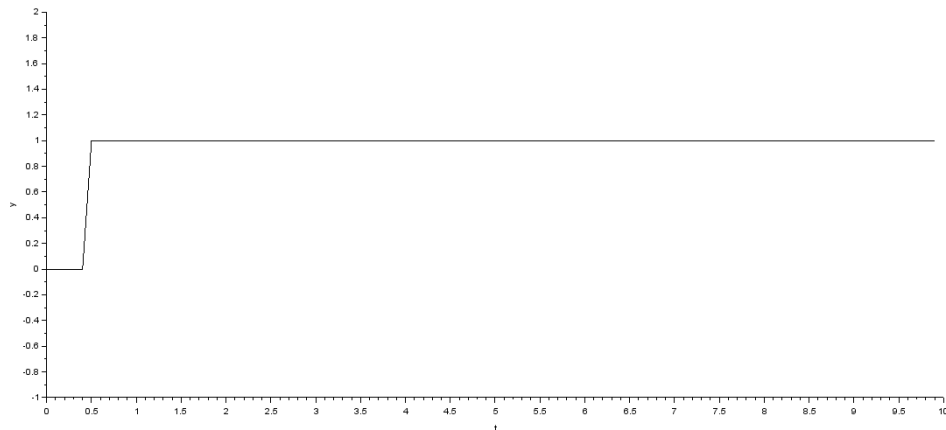
De forma geral, o tempo total de simulação depende do sistema. De acordo com o tempo de acomodação, sistemas podem entrar em regime permanente mais rápido ou mais lentamente. Mas, em qualquer dos casos, esse tempo não será muito alto. Você pode simular uma primeira vez e depois ir alterando o tempo de simulação para enquadrar totalmente o sinal na tela do **CScope**.

Para visualizar o sinal de forma correta, os parâmetros do *clock* e do **CScope** devem ser alterados. Clique duas vezes sobre o bloco do **CScope** e altere os valores de $Y_{\text{MÁX}}$ e $Y_{\text{MÍN}}$ para valores que melhores se ajustam ao sinal desejado na saída. Você pode rodar a simulação uma primeira vez, observar a faixa de valores de Y, pausar a simulação, alterar as configurações no **CScope** e depois rodar a simulação novamente.

Ainda na janela dos parâmetros do **CScope** configure o parâmetro *Refresh period* para o mesmo valor ajustado no parâmetro do tempo de integração, mostrado na janela anterior. As outras configurações de simulação podem ser mantidas com os valores que o próprio *SciLab* mostrar na tela.



Clique em **Run** e o gráfico aparecerá em uma janela que será aberta na tela. Você pode alterar a configuração do gráfico fazendo com que ele fique ajustado na tela. Na barra de ferramentas do gráfico, na guia editar, você pode alterar as propriedades do gráfico, como, por exemplo, nomear os eixos X e Y, alterar a cor da curva etc.



Após gerar o gráfico, você pode salvá-lo em um formato de imagem. Para isso, na janela que foi aberta pelo gráfico, clique em arquivo e vá na opção de exportar. Renomeie o arquivo e verifique se a extensão está em um formato de imagem. Selecione o diretório que for mais adequado e clique para exportar.

Exercícios

- (1) Simule uma onda senoidal com amplitude $4,55V$ e frequência de $2Hz$. Ajuste o tempo para que no *Scope* que sejam vistos três ciclos do sinal na tela.
- (2) Simule uma entrada do tipo rampa com inclinação 3.
- (3) Simule uma onda senoidal de amplitude $2V$ e frequência de $1Hz$ com um ganho de 0,8. Ajuste o tempo para que no *Scope* sejam vistos três ciclos do sinal na tela. Encontre o valor de pico do sinal de entrada após passar pelo bloco de ganho.