SIMULINK

erig@utfpr.edu.br

1 INTRODUÇÃO

Esta apostila tem o objetivo de servir como referência básica para um curso de técnicas de programação baseadas em Matlab e Simulink. Esta referência, aliada a atividades de laboratório, deve propiciar ao aluno uma base de conhecimento necessária para o bom aproveitamento em um curso de automação e controle. Este material tem não tem a pretensão de ser completo ou suficiente em qualquer dos temas abordados. Referências adicionais são feitas com o desejo que os alunos busquem o importante aprofundamento nos diversos temas aqui explorados.

2 O AMBIENTE SIMULINK

O que é o Simulink

Simulink é um software associado ao Matlab usado para modelagem, simulação e análise de sistemas dinâmicos. Permite trabalhar com sistemas lineares e não lineares, contínuos ou discretos. O Simulink apresenta uma interface gráfica com o usuário (graphical user interface GUI) para construção de modelos usando diagramas de blocos. Estes blocos estão distribuídos em diversas bibliotecas, permitindo a construção de modelos complexos com o mínimo de trabalho.

Executando um modelo térmico

O Simulink apresenta uma série de modelos usados para demonstração. Um destes modelos é a simulação de uma planta térmica de uma casa. Para executar este exemplo digite na linha de comando do Matlab o comando:

```
>>thermo
```

O modelo criado em Simulink é mostrado na figura 1. A seta vermelha aponta para o botão usado para iniciar a simulação deste modelo. Pressionando este botão são gerados os gráficos da figura 2, onde são apresentadas as temperaturas internas e externas da casa e a evolução do custo da energia com o tempo.

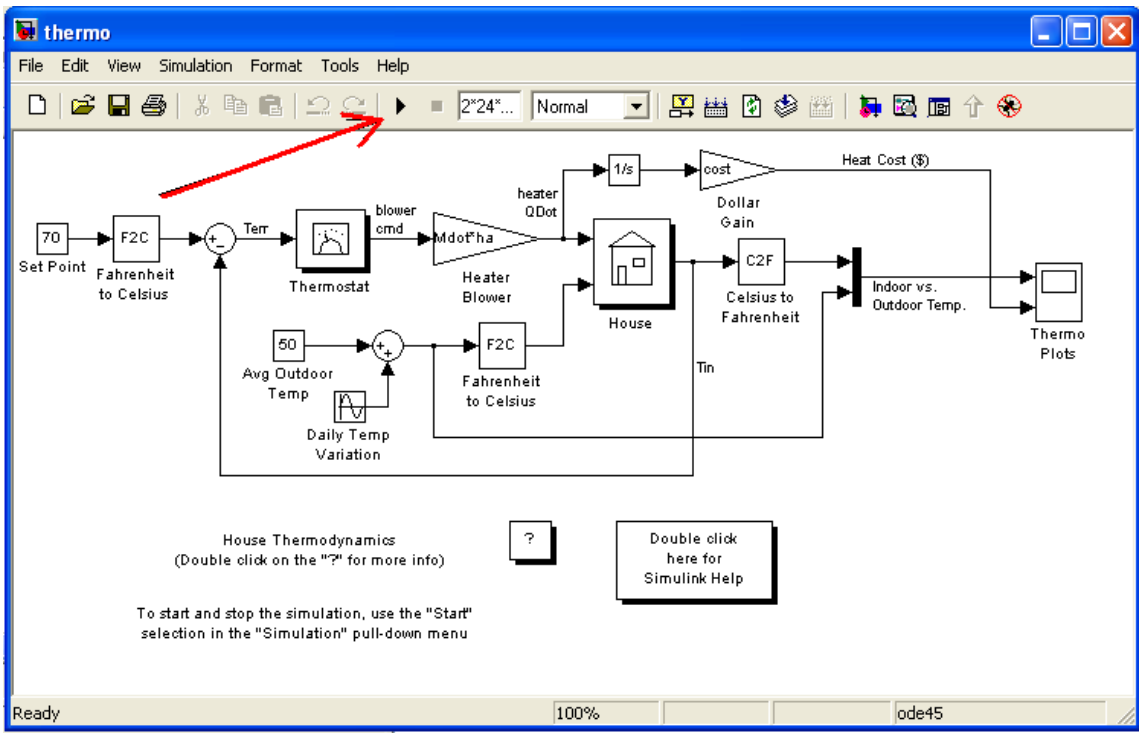


Figura 1 – Modelo em Simulink do controle de temperatura em uma casa.

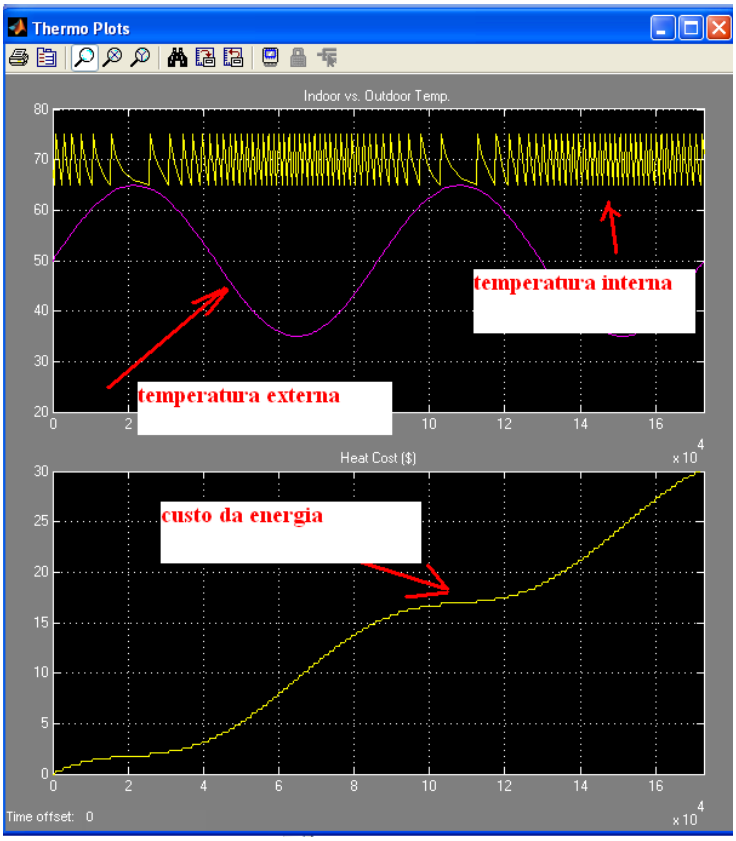


Figura 2 - Curvas de resposta do sistema térmico.

Alem da simulação propriamente dita é possível observar algumas propriedades do projeto usando Simulink:

- É possível "clicar" sobre diversos blocos criados para obter maiores detalhes sobre eles. Por exemplo: tente "clicar" sobre o bloco da casa, sobre o bloco termostato ou sobre o bloco de "Simulink Help" para observar o que ocorre.
- É possível mudar o set-point do sistema diretamente no bloco "set point". Mude o valor de temperatura e execute o modelo novamente.
- Existe um campo ao lado a tecla de inicio de execução que pode ser usado para mudar o tempo de simulação.

Outros modelos de demonstração (figura 3) podem ser acessados com o comando:

>> simulink 3

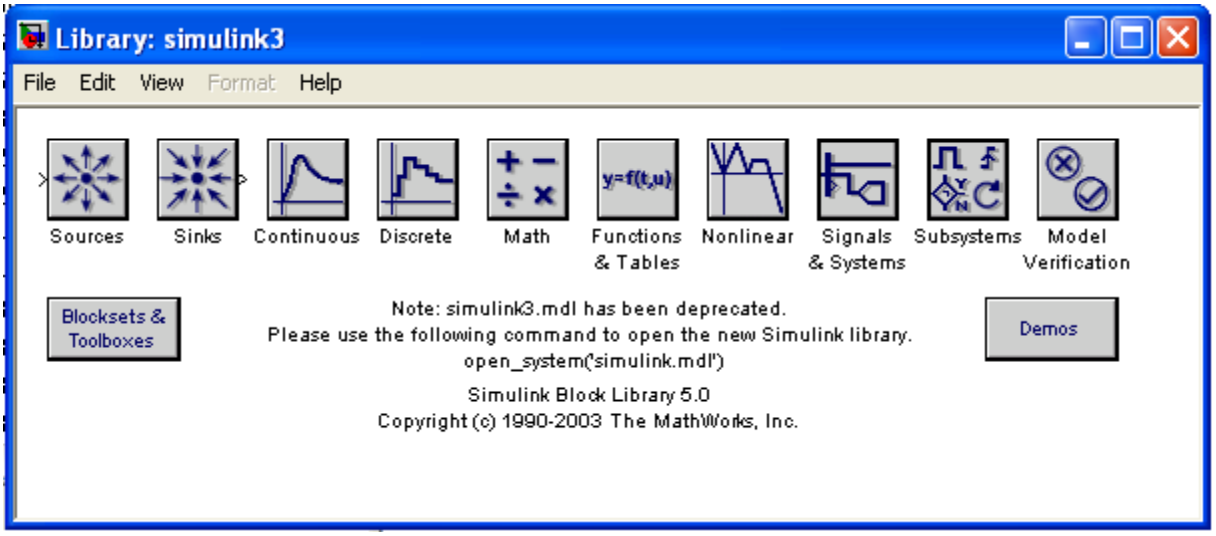


Figura 3 - Tela com diversas bibliotecas do Simulink e vários modelos de demonstração.

3 CRIANDO UM MODELO

O Simulink pode ser acessado usando-se o comando: >> Simulink

Que abre a tela da figura 4, representado as bibliotecas disponíveis.

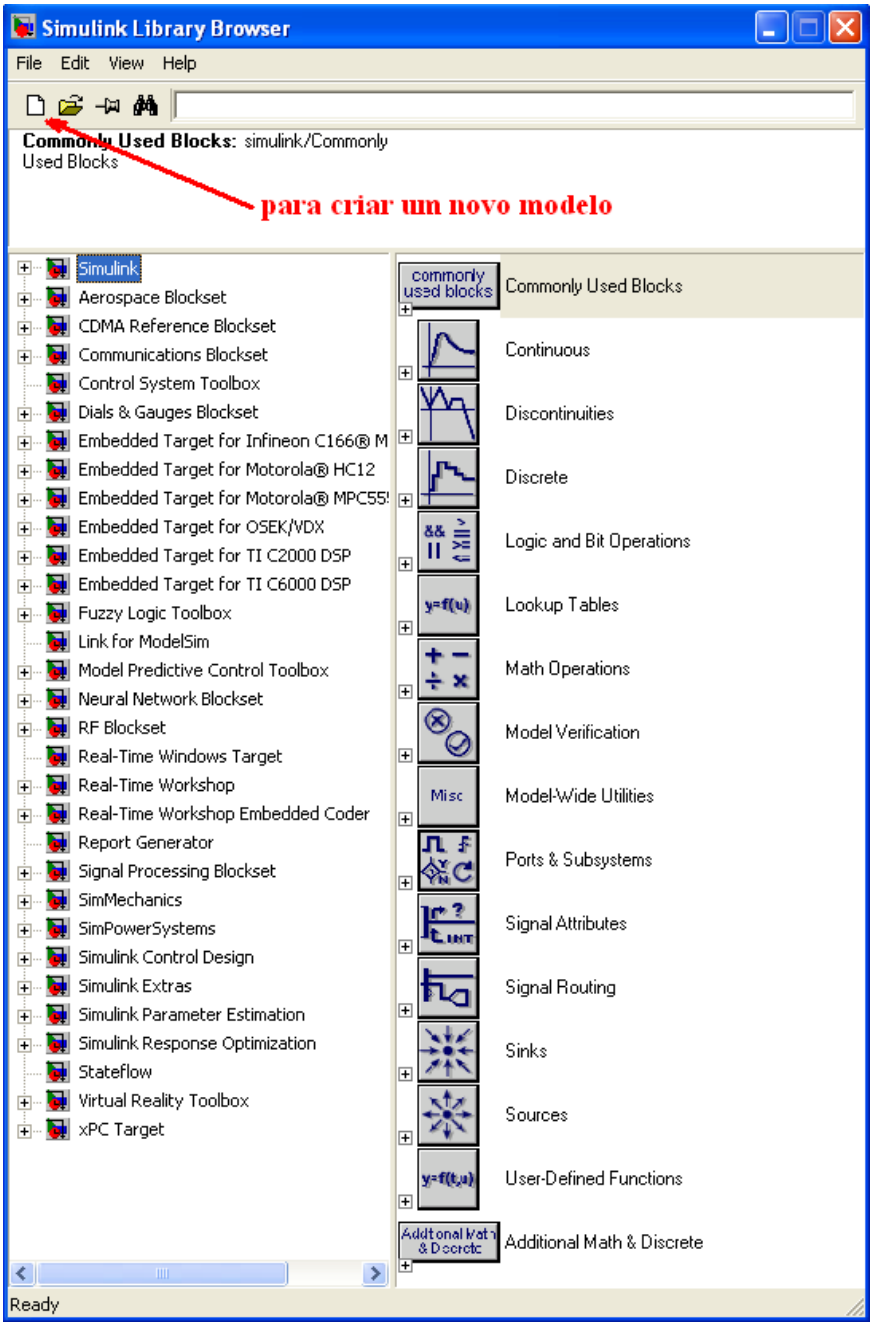


Figura 4 – Tela inicial do Simulink.

A seta vermelha, na figura 4, ressalta o ícone usado para criar um novo modelo. Se pressionado este ícone é criado uma nova janela em branco. Nesta janela será criado o novo modelo através da inserção progressiva de blocos e de linhas ligando entradas e saídas dos mesmos.

O modelo pode ser iniciado pela escolha do bloco de entrada função seno (Sine Wave block), representado na figura 5. Observe que este bloco faz parte de uma biblioteca chamada de Sources. Esta biblioteca apresenta vários blocos que podem ser usados como sinais de entrada em sistemas a serem simulados. Este bloco é "arrastado" para área de trabalho recém criada: "clique" sobre este bloco na biblioteca e mantenha o botão do mouse pressionado até que o bloco esteja na posição desejada.

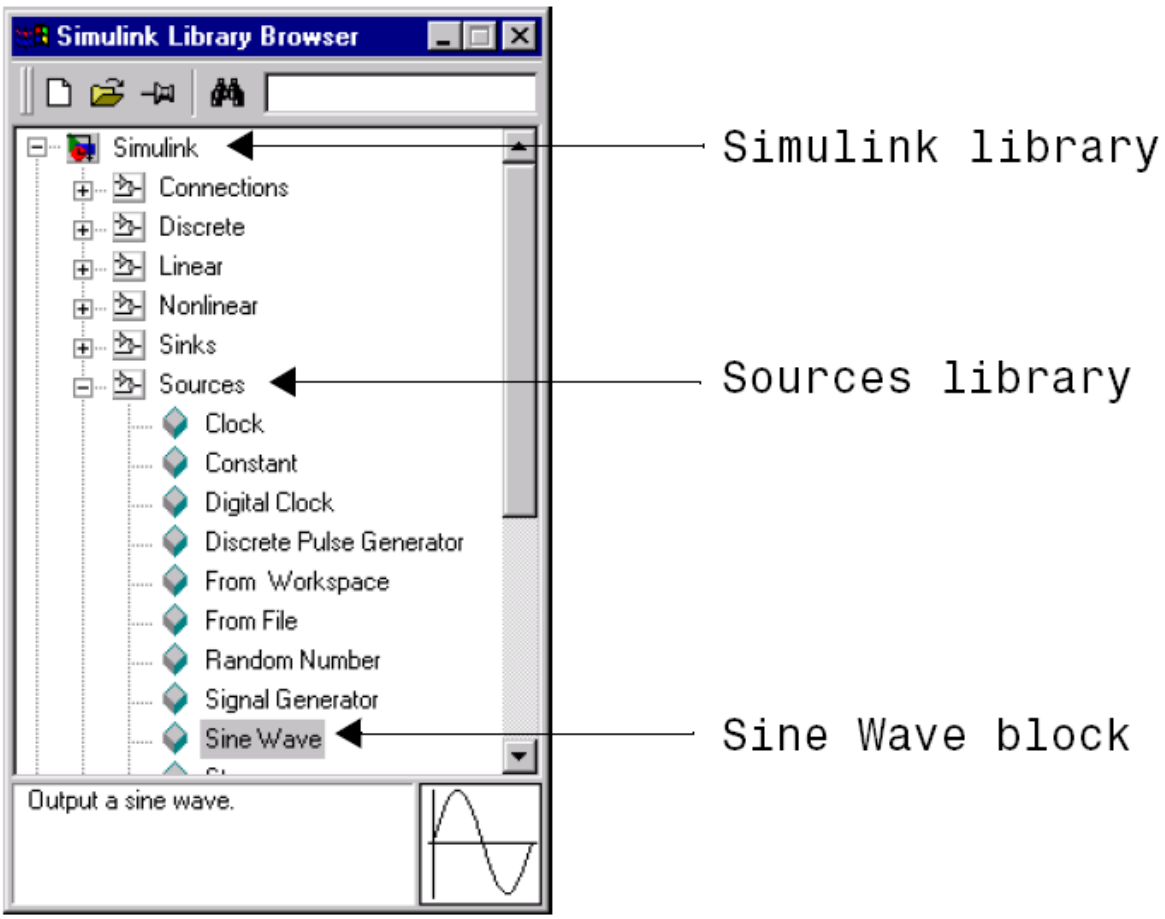


Figura 5 – Seleção do bloco "Sine Wave" na biblioteca "Sources" do Simulink.

A área de trabalho deve ficar como na figura 6.

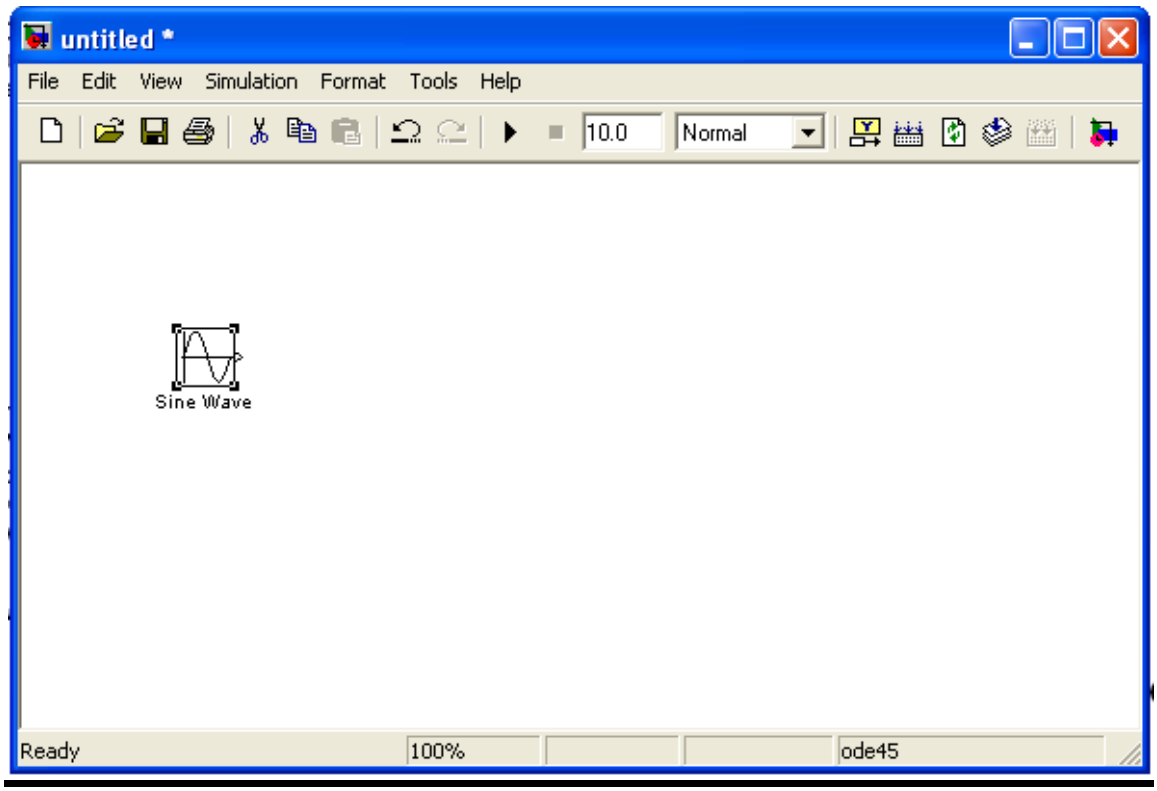


Figura 6 – Área de trabalho do modelo depois da inserção do bloco de entrada senoidal.

As figuras 7,8 e 9 apresentam os blocos seguintes a serem inseridos na área de trabalho. Na figura 7 é inserido o bloco integrador (“Integrator”), localizado na biblioteca “Continuous”. Na figura 8 é inserido o bloco “Scope”, localizado na biblioteca “Sinks”. A biblioteca “Sinks” apresenta uma coleção de blocos que são usados para visualização de resultados de saída. Na figura 9 é inserido o bloco “Mux”, localizado na biblioteca “Signal Routing”.

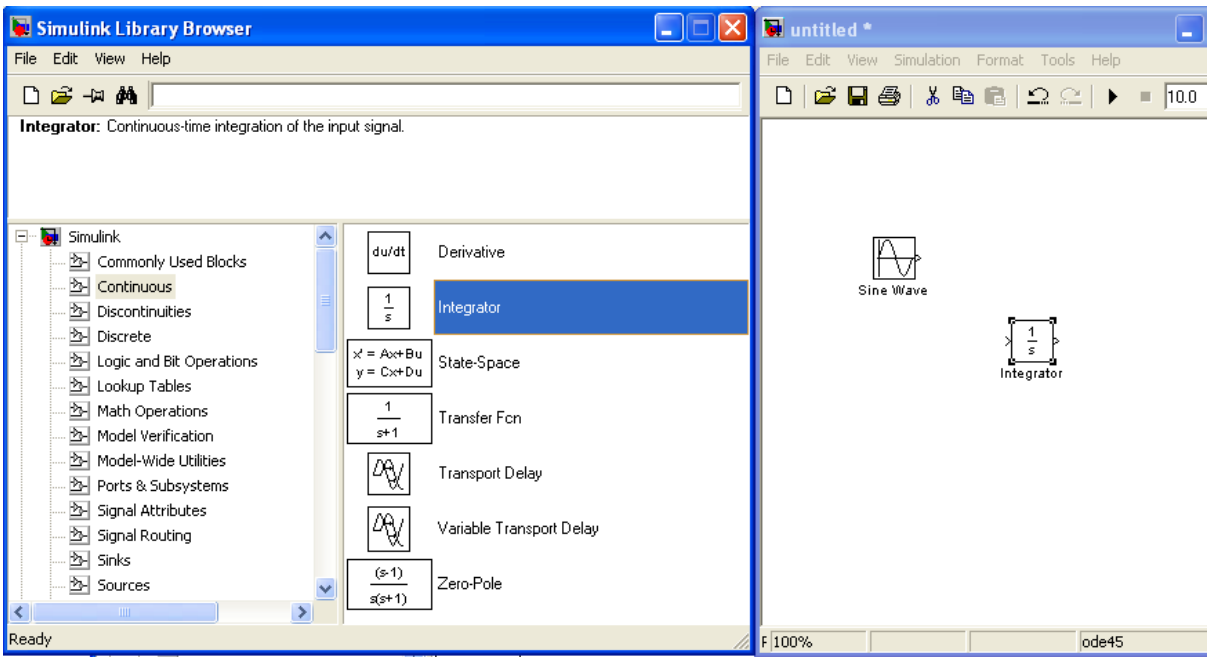


Figura 7 - Inserção do bloco "Integrator" na área de trabalho.

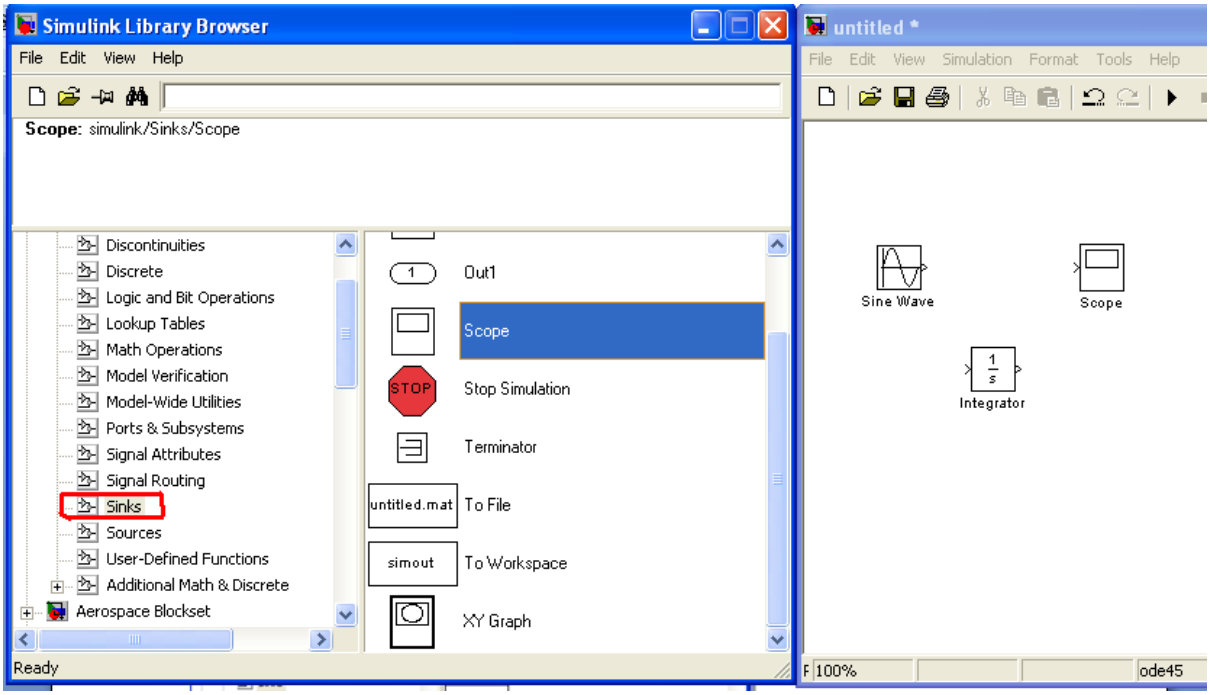


Figura 8 - Inserção do bloco "Scope" na área de trabalho.

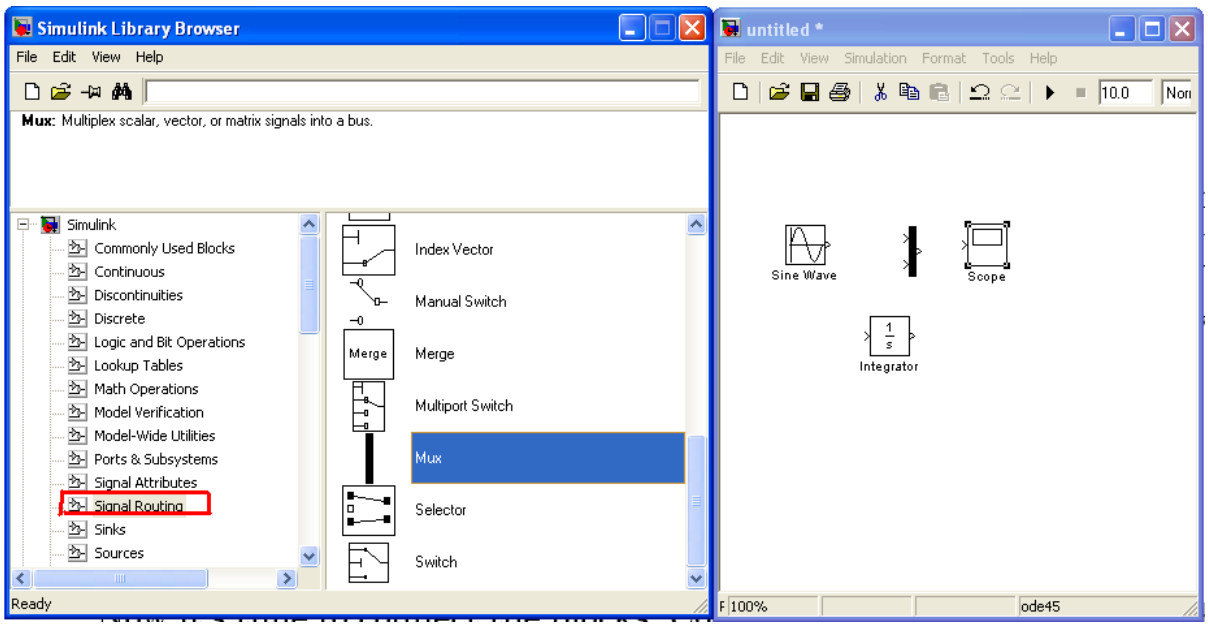


Figura 9 - Inserção do bloco "Mux" na área de trabalho.

Agora os blocos devem ser interligados com linhas. Por exemplo, a saída do bloco "Sin Wave" deve ser ligada à entrada do bloco "Integrator" e a uma das entradas do bloco "Mux". Basta "clicar" o mouse sobre uma saída e, mantendo o botão pressionado, levar a linha até a entrada desejada. A tecla CRTL pode ser usada para criar uma derivação de uma linha. A área de trabalho deve apresentar o aspecto da figura 10.

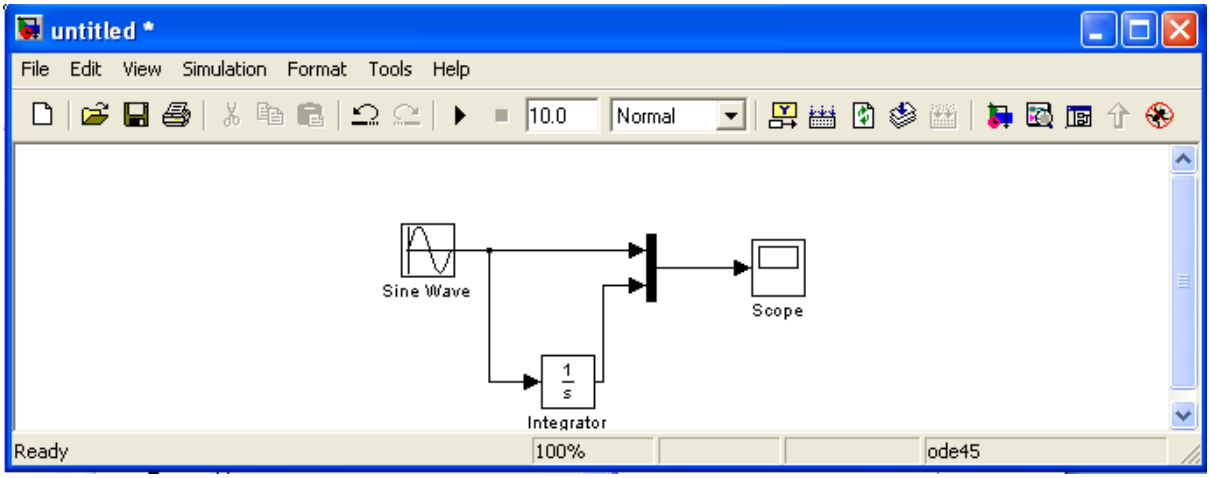


Figura 10 – Área de trabalho com os blocos interligados.

Antes da simulação é importante definir alguns parâmetros, como por exemplo o tempo total da simulação. Isto é feito no menu Simulation-Configuration Parameters, segundo a figura 11. No caso, serão usados 10 segundos.

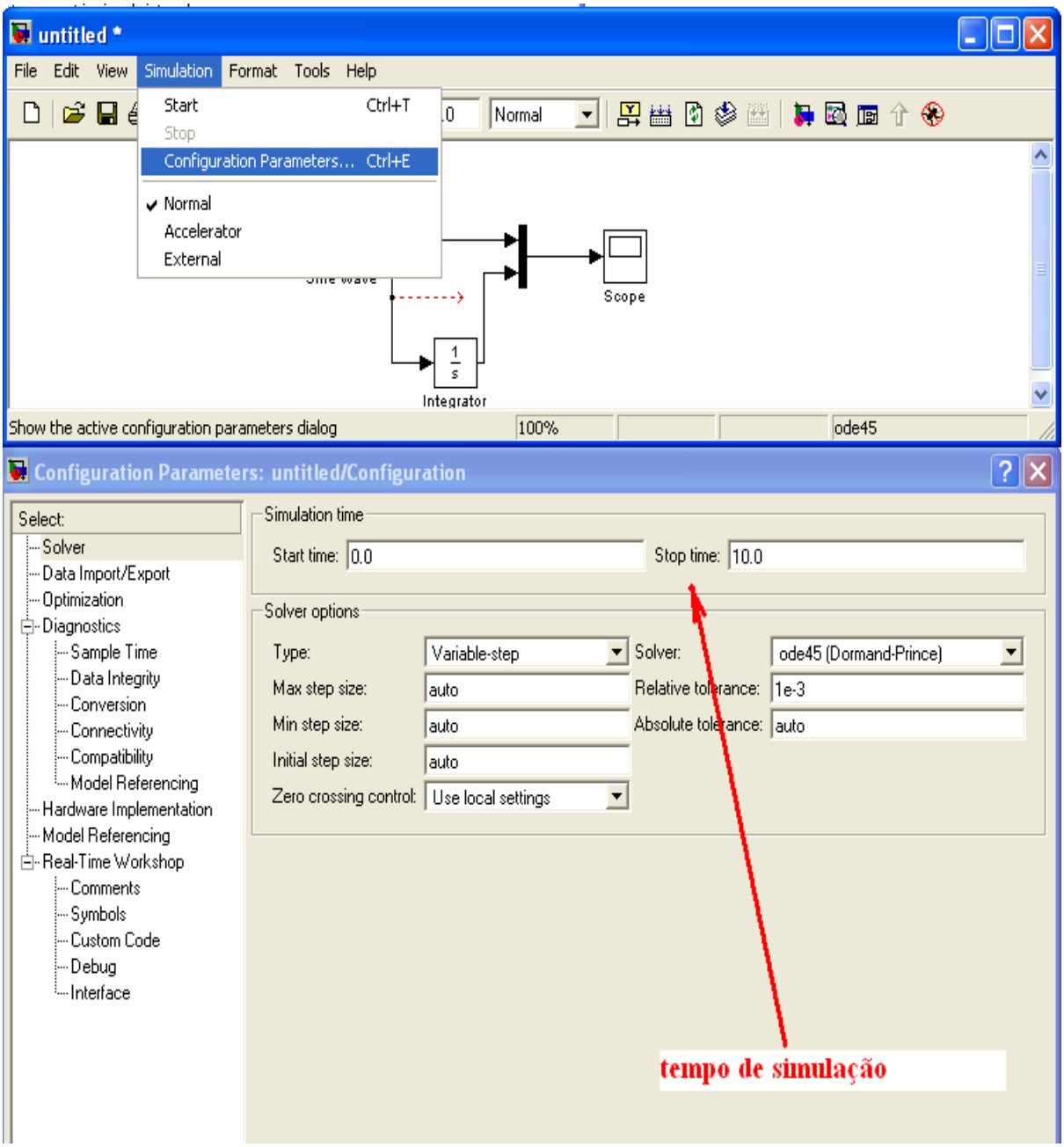


Figura 11 – Tela de configuração de parâmetros.

O resultado da simulação pode ser visto na figura 12.

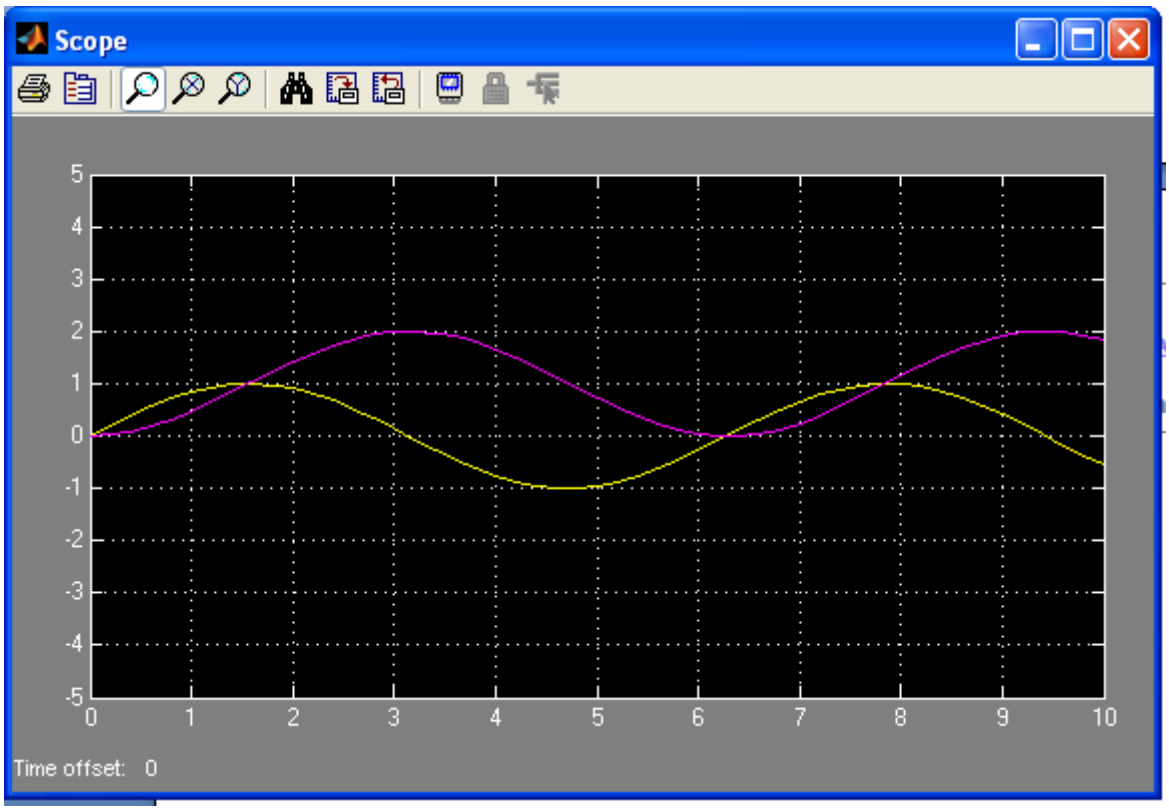


Figura 12 – Resultado da simulação do modelo criado.

Salve o modelo com o nome teste1. Será criado o arquivo teste1.mdl. Este modelo poderá ser recuperado quando for necessário.

Métodos de solução

Os diversos métodos de solução de equações diferenciais são listados na tabela seguinte (UERG):

MÉTODO DE SOLUÇÃO	CARACTERÍSTICAS
ODE45	Excelente método de propósito geral de passo simples. É baseado nos métodos de Dormand-Prince e de Runge-Kutta para Quarta/Quinta ordem. ODE45 é o método default e é geralmente uma boa primeira opção.
ODE23	Usa os métodos Bogacki-Shampine e Runge-Kutta de Segunda/Terceira ordem. Ás vezes funciona melhor do que ODE45 na presença de pouca flexibilidade. Geralmente requer um passo menor do que ODE45 para atingir a mesma precisão.
ODE113	Utiliza o método de ordem variável de Adams-Bashforth-Moulton. Já que ODE113 utiliza as soluções de pontos em tempos anteriores para se determinar a solução do tempo corrente, deve então produzir a mesma precisão dos métodos ODE45 ou ODE23 com menor número de cálculos e com isto tendo uma melhor performance. Não é apropriado para sistemas com descontinuidades.
ODE15S	Sistema de ordem variável de multi passos para sistemas inflexíveis. É baseado em pesquisas recentes que utilizam fórmulas numéricas de diferença. Se a simulação executar lentamente utilizando ODE45, tente ODE15S.
ODE23S	Ordem fixa de passo simples para sistema inflexíveis. Devido ao fato de ser um método de passo simples, em muitas das vezes é mais rápido do que ODE15S. Se um sistema parecer inflexível é uma boa idéia tentar ambos os métodos para este tipo de sistema para se determinar qual dos dois tem melhor performance.
Fixed-and-Variable-Step Discrete	Método especial para sistemas que contém estados descontínuos.
ODE5	Versão de passo fixo de ODE45.
ODE4	Fórmulas clássicas de Quarta ordem de Runge-Kutta utilizando passo de tamanho fixo.
ODE3	Versão de passo fixo de ODE23.
ODE2	Método de Runge-Kutta de passo fixo de Segunda ordem, também conhecido por Método de Heun.
ODE1	Método de Euler utilizando passo de tamanho fixo.

Exercícios:

1 - Baseado no modelo anterior crie um modelo que some o sinal de entrada senoidal e o sinal de saída do integrador. O sinal resultante deve aparecer na mesma tela.

2 – Crie um novo modelo que apresente o resultado gráfico de dois sinais: um sinal triangular de 1kHz e um sinal senoidal de 500 Hz.

4 OUTROS RECURSOS

O Simulink apresenta uma série de recursos adicionais que podem ser usados para facilitar a integração com outros programas feitos no Matlab bem como permitir a criação de modelos de forma mais fácil.

Usando o Workspace

A passagem de parâmetros entre o Workspace do Matlab trabalho pode ser feita a qualquer momento. Qualquer bloco do Simulink aceita variáveis criadas em linhas de comando do MATLAB. Por exemplo crie a variável na linha de comando do Matlab:

```
>> a = 2;
```

Agora, no exemplo da figura 10, modifique o,parâmetro do bloco "sin wave", como mostrado na figura 13. Observe que o bloco em questão teve o parâmetro relacionado com a amplitude da senoide associado com a variável "a" criada na linha de comando do Matlab. Isto pode ser muito útil para configuração de simulações de modelos.

De forma igualmente útil é possível passar dados para o Workspace do Matlab. Para tanto, se utiliza o bloco "To Workspace", segundo a figura 14. Nesta figura é adicionado o bloco "To Workspace" no exemplo anterior.

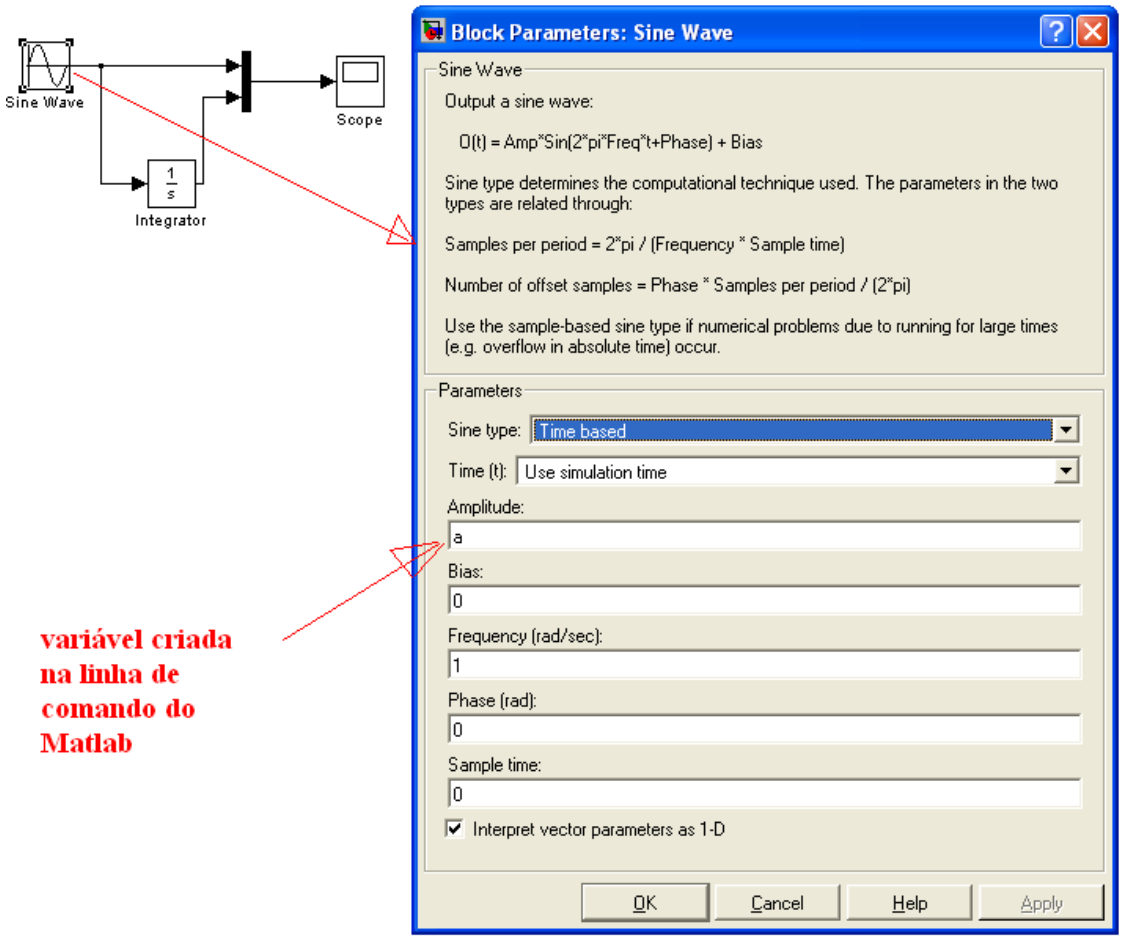


Figura 13 – Exemplo de passagem de parâmetro.

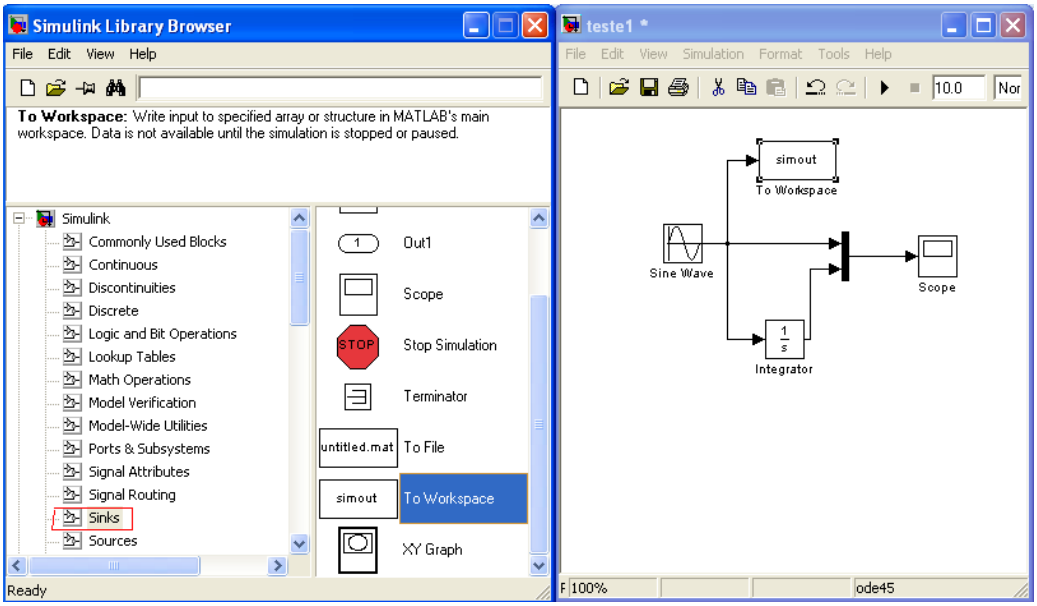


Figura 14 – Inserindo o bloco "To workspace".

Dentro do bloco "To Workspace" é possível mudar algumas propriedades. A figura 15 mostra as duas mudanças a serem feitas: o nome da variável e o formato de saída. O objetivo é criar uma variável tipo array com o nome saída1 que será salva no Workspace. A propósito, qualquer bloco do Simulink apresenta um help de detalha os parâmetros usados no mesmo.

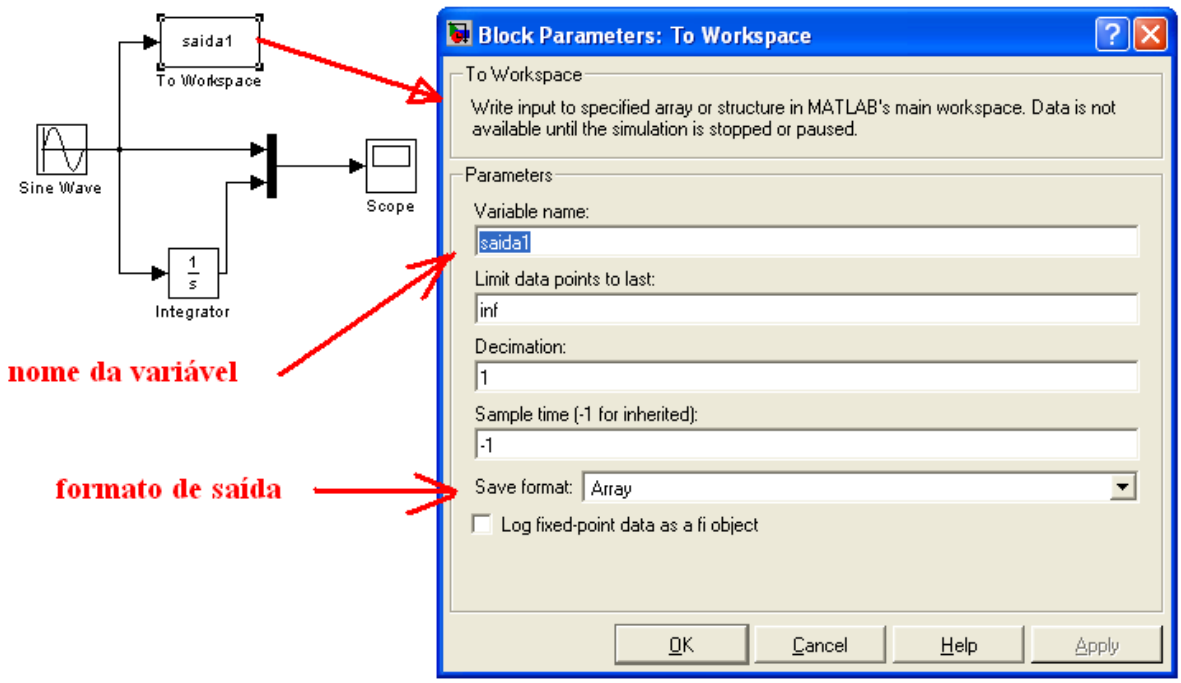


Figura 15 – Modificação dos parâmetros do bloco "To Workspace".

Execute a simulação do modelo e depois, na linha de comando do Matlab:

```
>> plot (saida1)
```

Cuja execução gerará o gráfico da figura 16.

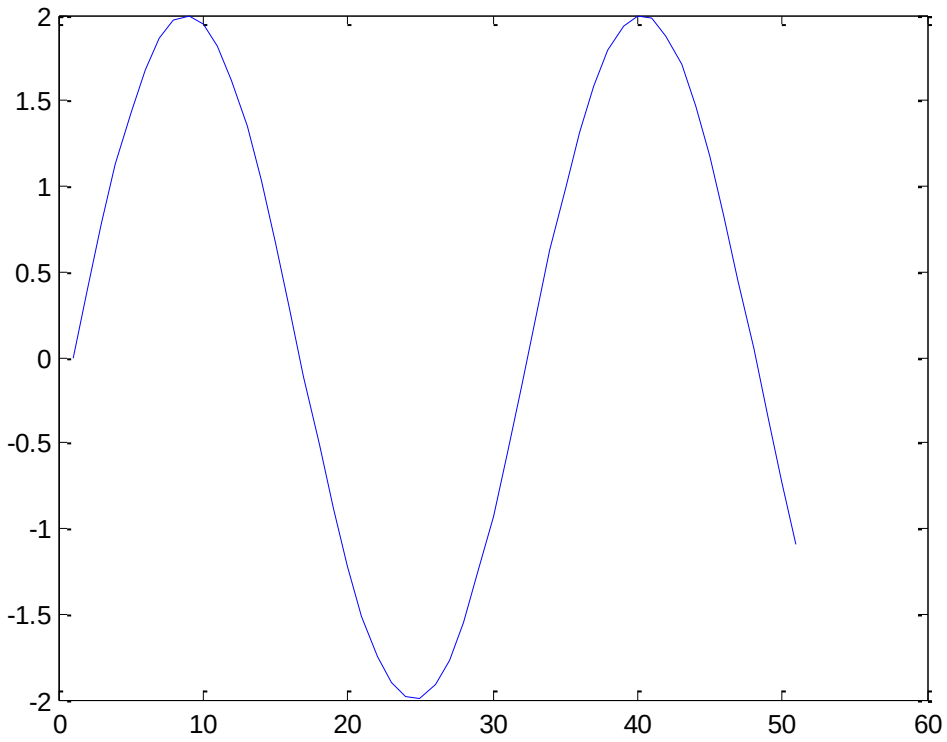


Figura 16 – Gráfico do vetor saída1, gerado no Simulink.

Executando um modelo em linha de comando do Matlab

A execução de um modelo em linha de comando do Matlab pode ser feita com o seguinte comando:

>> sim (‘modelo’)

Este comando irá simular o modelo criado, usando inclusive as variáveis definidas no Workspace. Um outro exemplo de execução:
>> [t,x,y] = sim(‘modelo’, tempolimite)

Onde t é o vetor de tempos usado na simulação, x e y são os vetores de resposta do modelo. Neste caso é considerado que o modelo apresenta duas variáveis sendo mandadas para o Workspace. O parâmentro "tempolimite" define o tempo de simulação. Por exemplo no modelo teste1.mdl da figura 15, execute a seguinte linha de comando:

```
>> [t,x] = sim('teste1', 20);
>> plot (x)
```

O resultado é mostrado na figura 17, onde se observa que o modelo foi executado além do tempo inicialmente previsto e que o valor de saída foi armazenado no vetor x, com uma cópia na variável saida1.

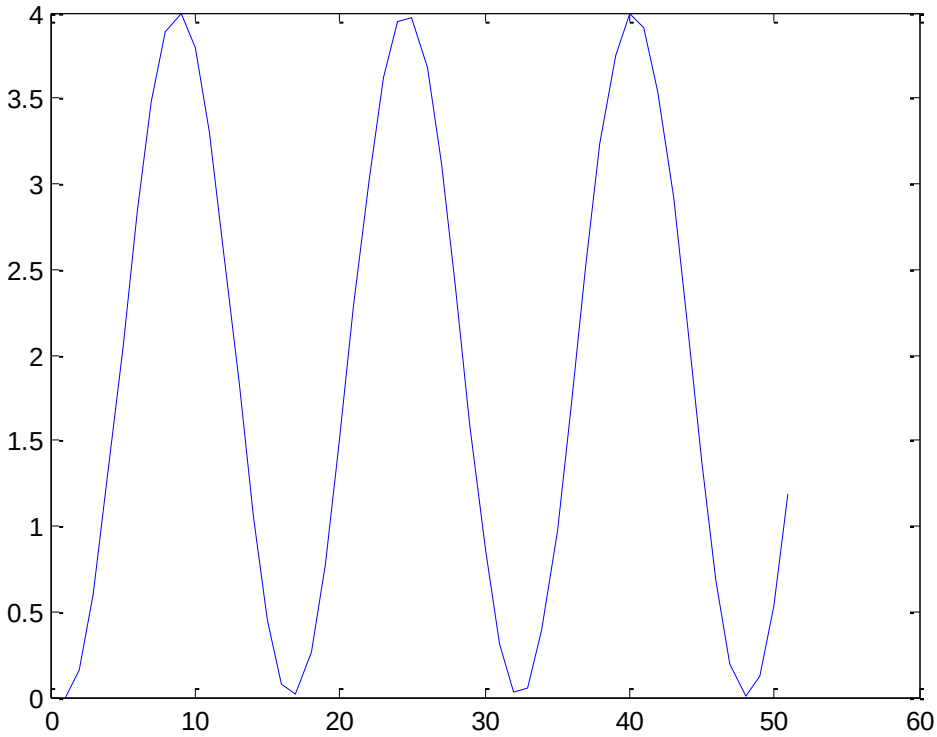


Figura 17 – Resultado da execução do modelo por linha de comando.

Criando um subsistema

Um subsistema pode ser visto como uma sub-rotina em um programa de computador. Além de organizar o projeto de um modelo de forma hierárquica, permitindo uma fácil visão de seus elementos principais, o uso de subsistemas permite a reutilização de blocos já implementado, facilitando a expansão e correção de modelos. Considere o exemplo engine.mdl, representado na figura 18.

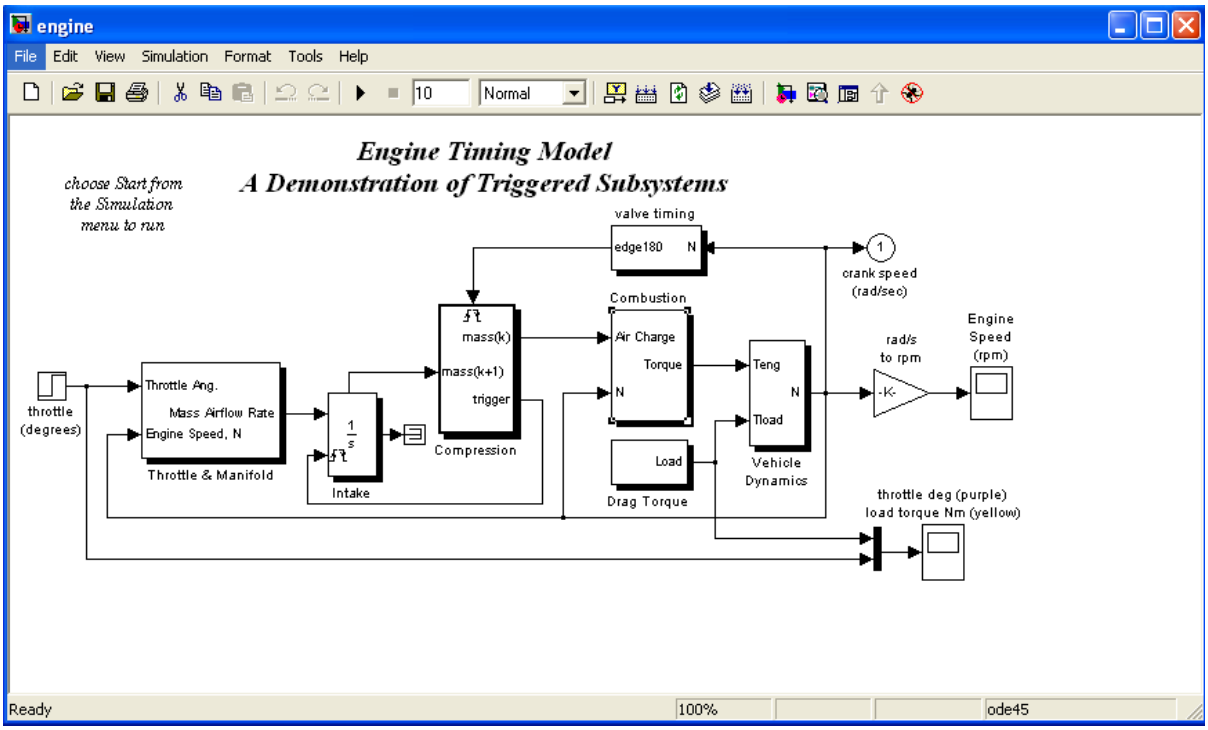


Figura 18 – Modelo de um motor a combustão.

Neste modelo de um motor a combustão diversos blocos foram criados para definir os módulo básicos do motor: combustão, dinâmica do veículo, compressão e assim por diante. Ao "clicar" duas vezes o bloco "combustion" é apresentada a janela da figura 19.

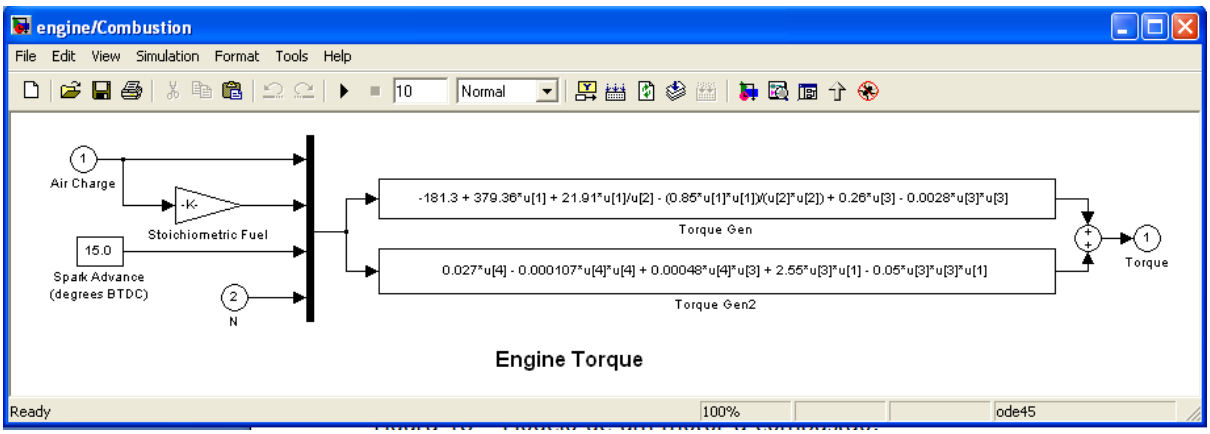


Figura 19 – Subsistema do motor a combustão.

Neste subsistema observam-se dois sinais de entrada: air charge e spark advance (entradas de ar e combustível) e uma saída: torque. Estes sinais são definidos como e entrada e saída usando-se as bibliotecas Sources (bloco In1) e Sinks (bloco Out1), de acordo com a figura 20.

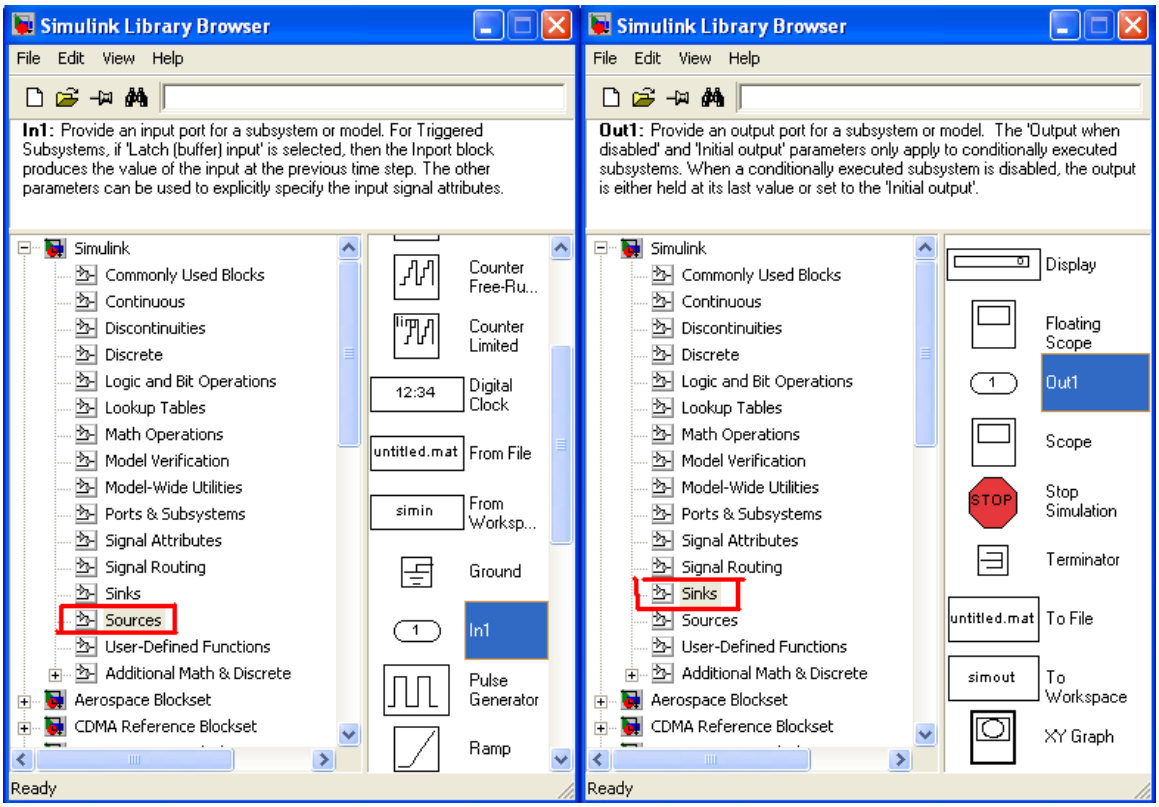
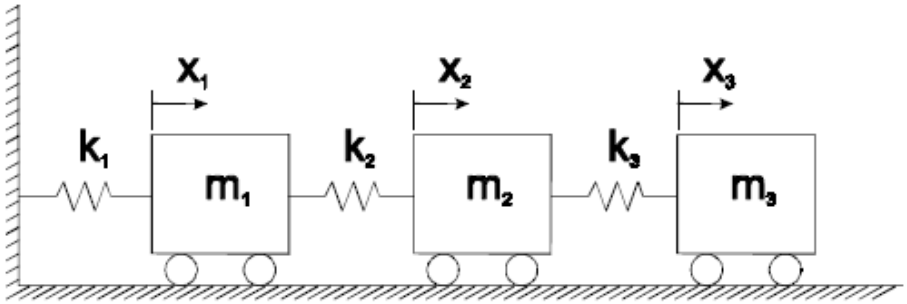
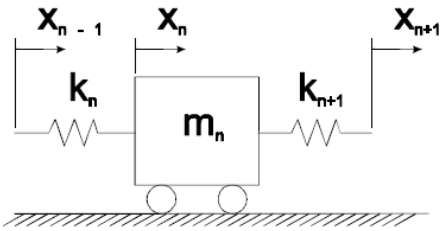


Figura 20 – Blocos usados para representar entradas e saída de um subsistema.

Como exemplo, pode-se criar o sistema (Eng. Elétrica - UERJ):



Onde a equação do movimento de um único carrinho será:



$$\ddot{x}_n = \frac{1}{m_n}\left[k_n\left(x_{n-1} - x_n\right) + k_{n+1}\left(x_n - x_{n+1}\right)\right]$$

O modelo em simulink será dado pela figura 21 e 22.

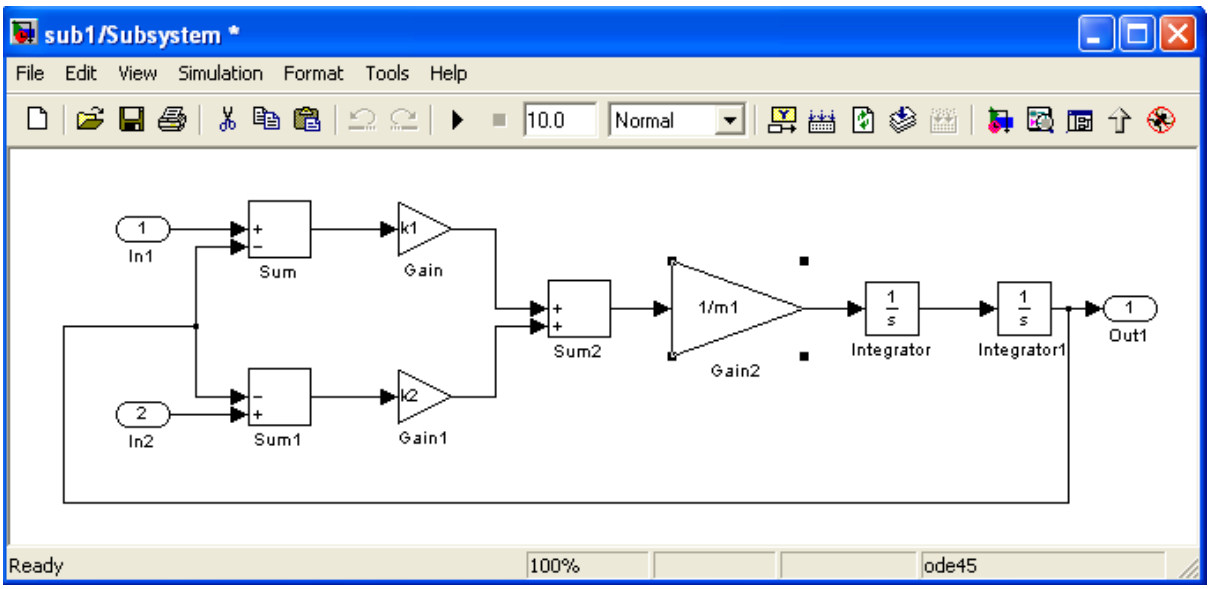


Figura 21 – Exemplo de subsistema.

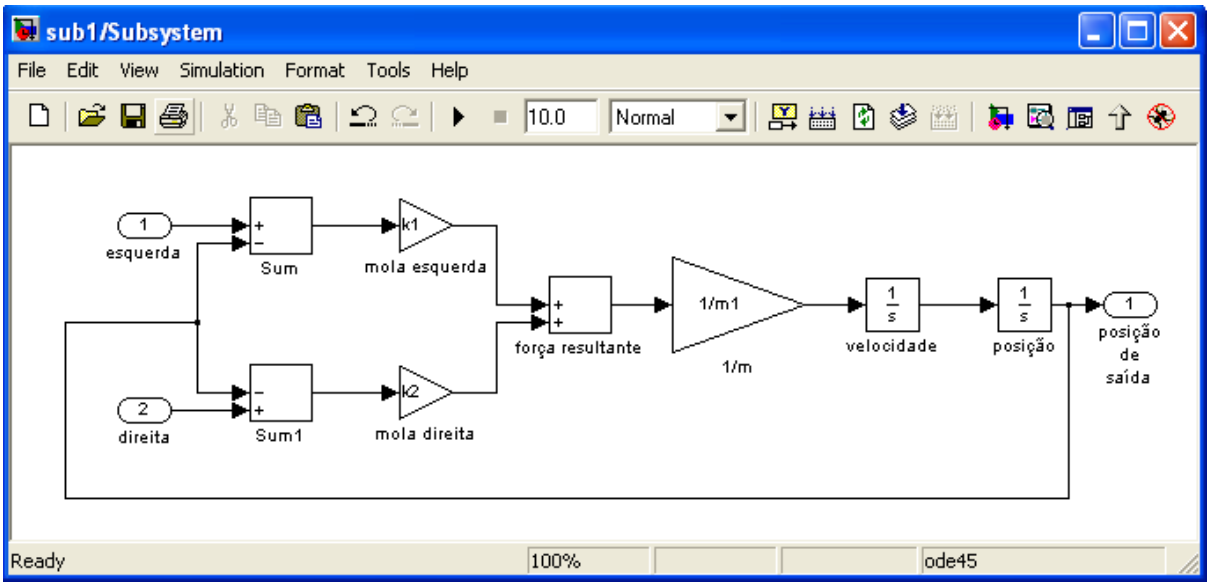


Figura 22 – Exemplo de subsistema com novos nomes de blocos inseridos.

Marcando-se todos os blocos desejados como subsistema pode criar o subsistema através do menu Edit – Create Subsystem visto na figura 23. O resultado é visto na figura 24.

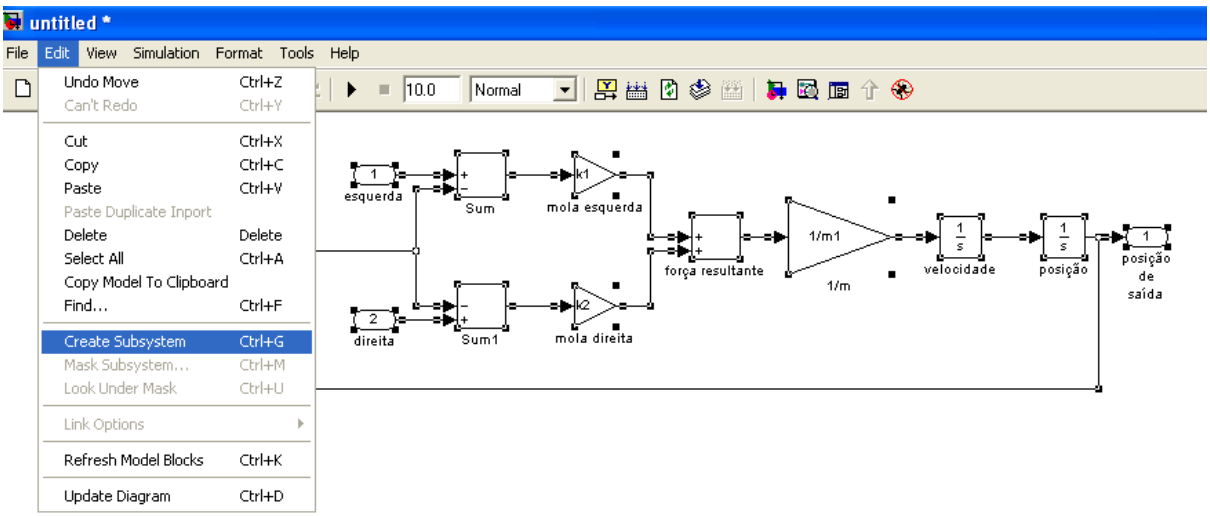


Figura 23 – Criação do subsistema.

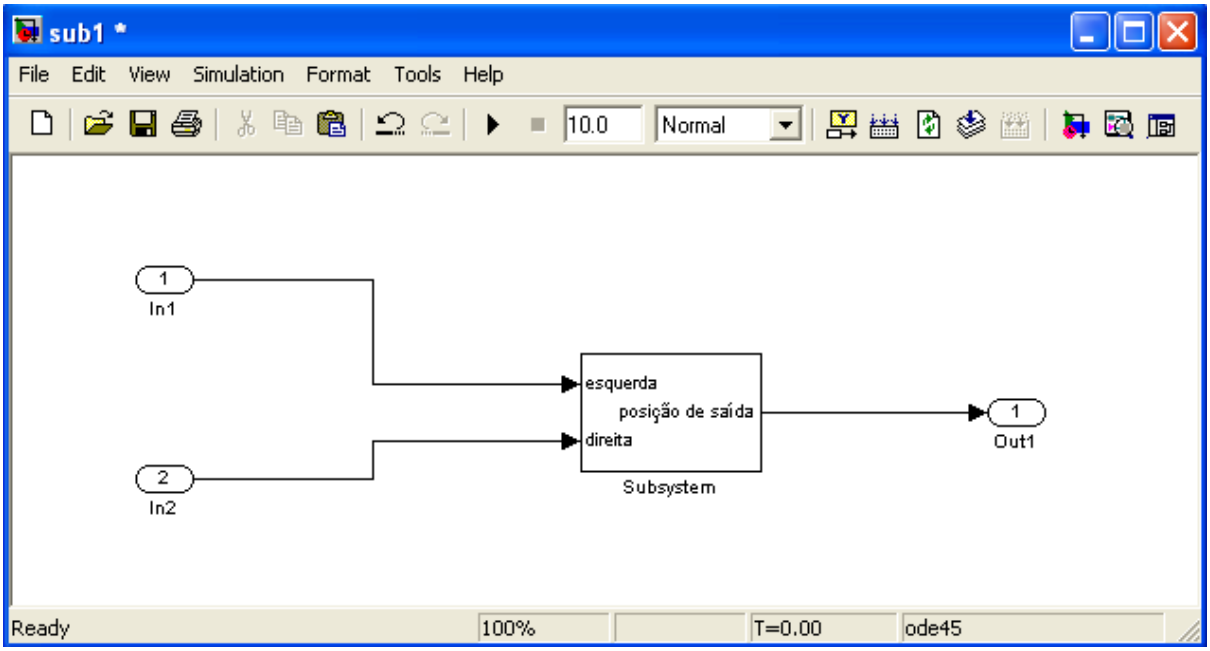


Figura 24 – Subsistema criado.

Finalmente, é possível cria o sistema completo, segundo a figura 25. Os parâmetros dos blocos de cada cópia do subsistema devem ser configurados:

- carro 1: o valor do ganho do bloco de ganho Mola da Esquerda deve ser k1 e o do bloco de ganho Mola da Direita k2. O bloco 1/massa deve ser configurado para ter o ganho 1/m1. A velocidade inicial deve ser 0 e a posição inicial 1.
- carro 2: o valor de ganho para o bloco Mola da Esquerda deve ser k2 e o do bloco Mola da Direita k3. O ganho do bloco 1/massa deve ser 1/m2. A velocidade inicial deste bloco é 0 e a posição inicial é também 0.
- carro 3: o valor de ganho do bloco Mola da Esquerda deve ser k3 e o bloco Mola da Direita 0, já que não há mola à direita deste carro. O ganho do bloco 1/massa deve ser 1/m3. A velocidade inicial é configurada para 0 e a posição inicial 0.

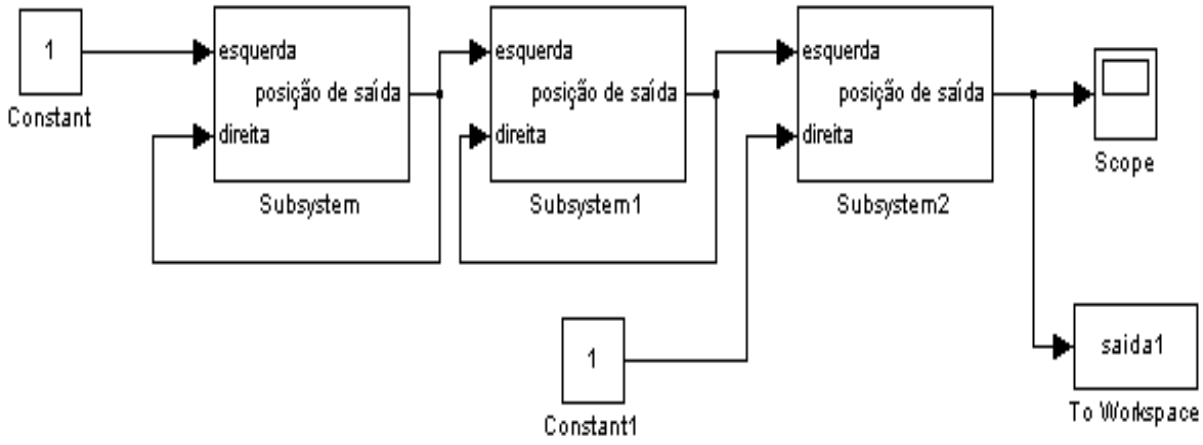


Figura 25 – Sistema completo usando três subsistemas.

Este sistema (salvo como sistema1.mdl) pode ser executado com os seguintes comandos do Matlab:

```
>> k1 = 1, k2 = 2, k3 = 4;
>> m1 = 1, m2 = 3, m3 = 2
>> sim('sistema1',100);
>> plot (saida1)
```

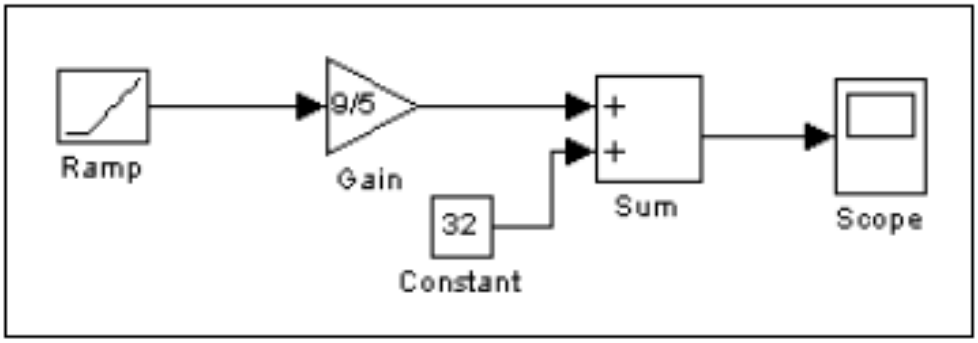
5 OUTROS EXEMPLOS

Convertendo Celsius para Fahrenheit

A fórmula que converte graus Celsius to Fahrenheit é:

$$T_F = 9/5(T_C) + 32$$

O modelo que pode criar esta conversão é mostrado na figura seguinte:

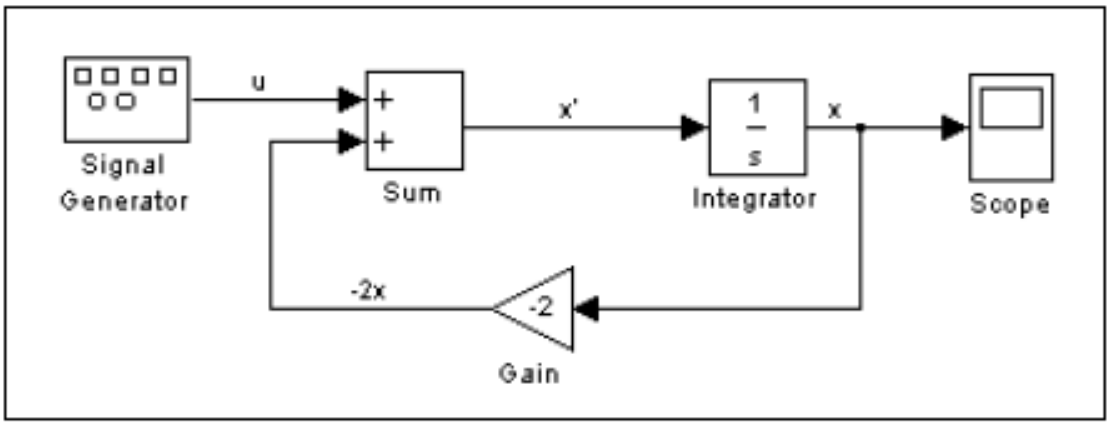


Modelo de conversão de graus Celsius to Fahrenheit.

Exercício: Execute este modelo e verifique os resultados. Mude a entrada para uma constante gerada pelo Matlab e a saída para uma constante a ser lida pelo Matlab.

Modelo de uma equação diferencial

O modelo da equação diferencial $x'(t) = -2x(t) + u(t)$, onde u(t) é a função excitação pode ser dado pela figura seguinte:



Modelo de uma equação diferencial.

Exercício: Verifique a resposta do modelo para diferentes sinais de entrada.