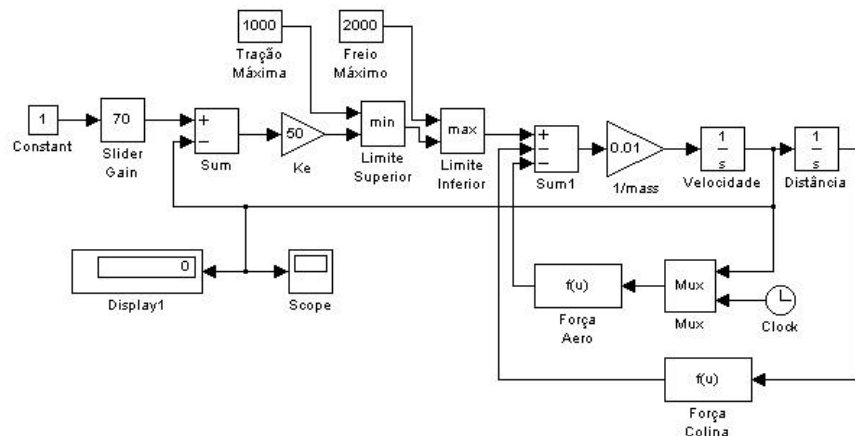


## Capítulo 6 – Subsistemas e Máscaras

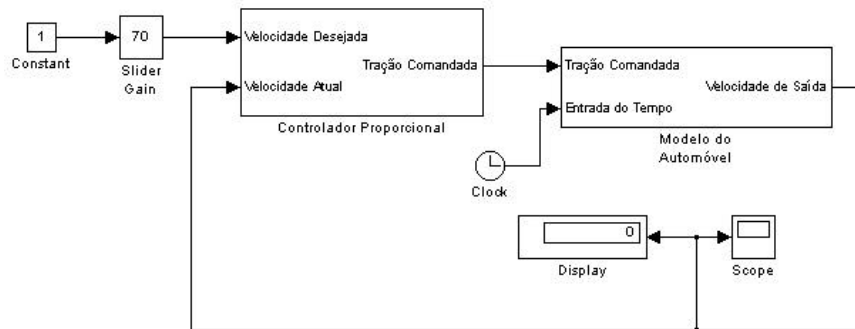
### 6.1 - Subsistemas SIMULINK

A maioria de linguagens de programação utilizados em engenharia incluem a capacidade de se utilizar *subprogramas*. Em FORTRAN existem subrotinas e subprogramas funções. Em C, os subprogramas são chamados funções, subprogramas MATLAB funções em arquivo M. O SIMULINK possui uma utilidade semelhante chamado *subsistemas*. Duas grandes razões para se utilizar subprogramas são a abstração dos detalhes e a reutilização do software.

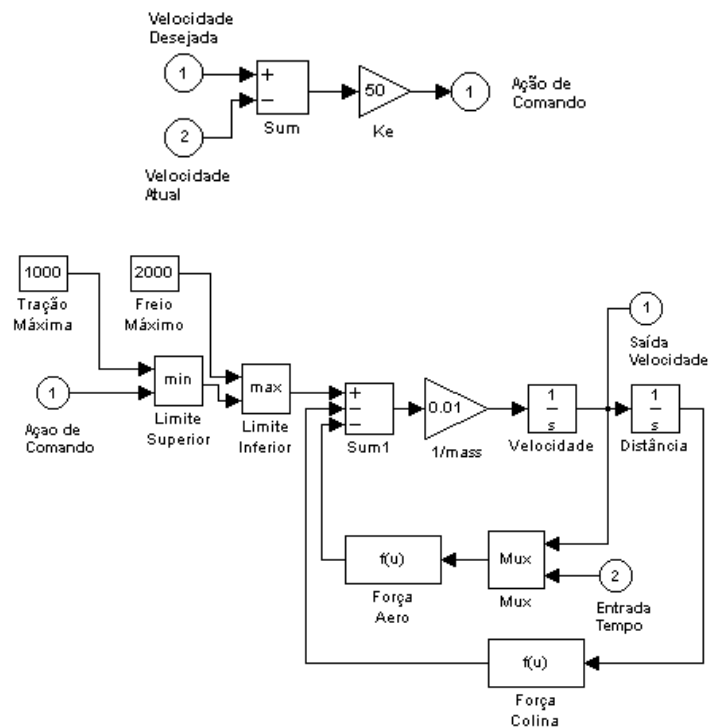
A medida que os modelos crescem e ficam complexos, tendem a ficar difíceis de se entender e executar manutenção. Subsistemas resolvem este problema fazendo que um complexo e grande modelo em grupos hierárquicos de modelos menores. Como um exemplo tem-se o modelo do automóvel já utilizado em exemplos anteriores. O modelo SIMULINK é mostrado na figura que segue.



Este modelo consiste basicamente em duas partes: A dinâmica do automóvel e o controlador. Examinando o modelo, não é tão claro determinar quais blocos representam a dinâmica do automóvel e quais representam o controlador. Na figura seguinte, foi feita a conversão das partes do automóvel e do controlador em subsistemas. Nesta versão, a estrutura conceitual é clara, mas os detalhes do controlador e da dinâmica do automóvel estão escondidas nos subsistemas. Esta estrutura hierárquica é um exemplo da abstração do software.



Subsistemas podem também ser vistos como componentes reutilizáveis de modelos futuros. Suponha que se deseja comparar as diferenças de projetos de controladores utilizando o mesmo modelo de dinâmica do automóvel. Ao invés de se construir um modelo completo cada vez que se utiliza um controlador é muito mais conveniente construir somente a parte do modelo referente ao controlador. Isto não só economiza um tempo considerável mas garante ao usuário que ele está usando exatamente a mesma dinâmica do automóvel. Uma vantagem importante na reutilização do software é a de que uma vez o sistema verificado funcionando corretamente, não é necessário repetir os testes e o processo de **debug** cada vez que o subsistema for usado em um novo modelo.

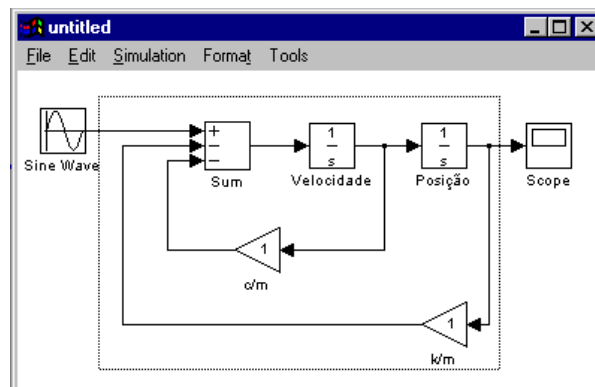


Existem dois métodos de construir subsistemas SIMULINK. O primeiro método é encapsular uma parte de um sistema já existente em um modelo pelo menu **Edit>Create Subsystem**. O segundo método é utilizar um bloco Subsystem (Subsystem) da biblioteca de Conexões (Connections). Os dois métodos serão discutidos.

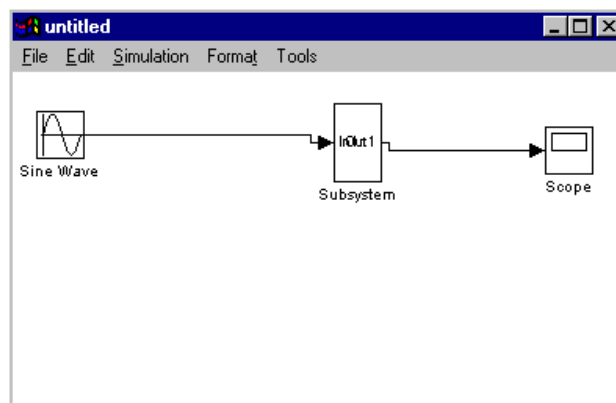
### 6.1.1 - Encapsulando um Subsistema

Para encapsular uma parte de um sistema SIMULINK já existente em um subsistema, deve-se proceder da seguinte forma:

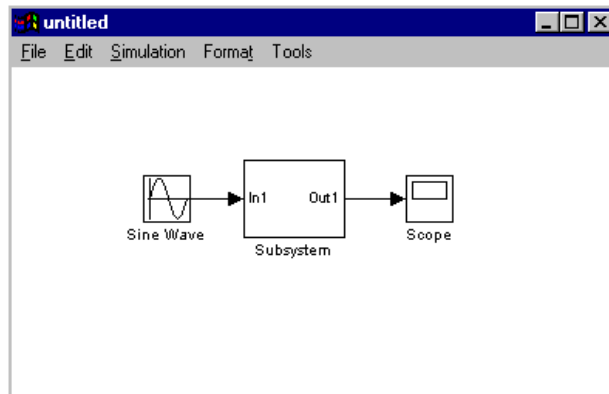
Selecione todos os blocos e linhas de sinal a serem incluídos no subsistema usando uma caixa de contorno com o mouse. Frequentemente é necessário rearranjar alguns blocos para que a caixa inclua somente os blocos desejados para o subsistema.



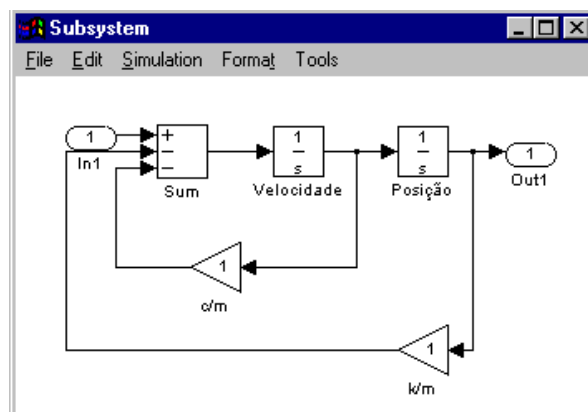
Clique em **Edit:Create Subsystem** na barra de menu da janela do modelo. O SIMULINK irá então substituir os blocos selecionados por um bloco Subsystema com entrada e saída para cada sinal de entrada ou saída.



O SIMULINK dará nomes *default* às portas de entrada e saída. Redimensione então o bloco para que os nomes das portas sejam legíveis e rearranje o modelo da maneira desejada.



Para visualizar ou editar o subsistema deve-se dar um duplo clique no bloco. Uma nova janela se abrirá e o seu conteúdo será o subsistema. Além dos blocos originais surgirão blocos Inport e Outport nos terminais de entrada e saída. Mudando o nome destes blocos, pode-se mudar os nomes das entradas no ícone do bloco. Feche então a janela quando terminar a edição do subsistema.



O comando **Edit>Create Subsystem** não possui a operação inversa. Uma vez encapsulado um grupo de blocos em um subsistema, não há como reverter o processo. Por isso é sempre uma boa idéia salvar o modelo antes de criar o subsistema. Se o usuário decidir que não deve aceitar o novo formato para seu sistema, deve então fechar a janela atual e abrir aquela salva antes de executada a operação de encapsulamento. Para se reverter manualmente este processo pode-se seleccionar todos os blocos contidos no subsistema e copiá-los para uma nova janela ou para a janela original do modelo.

## 6.1.2 - Blocos de Subsistema

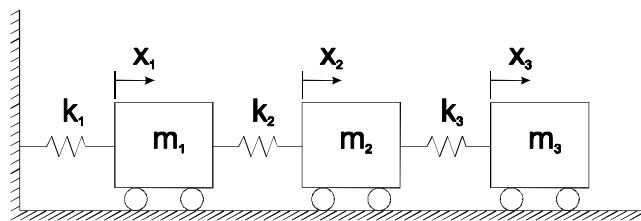
Se ao se construir um modelo o usuário souber que irá precisar de um subsistema, é conveniente construir o subsistema diretamente numa janela. Isto elimina a necessidade de se rearranjar os blocos que irão compor o subsistema de modo a enquadrá-los no espaço disponível. Isto permite que o usuário inicie a construção do modelo após o subsistema ser encapsulado.

Para criar um subsistema utilizando um Bloco Subsistema, inicialmente se deve arrastar um destes blocos da biblioteca de Conexões (Connections). A seguir, com

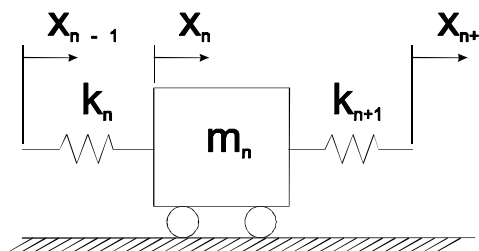
um duplo clique sobre ele abre-se a janela do subsistema. Nesta janela deve-se construir o subsistema utilizando os procedimentos padrões para construção de modelos. Utiliza-se blocos Inport para os sinais de entrada do sistema e blocos Outport para os sinais de saída. Se desejado, pode-se mudar os nomes destes blocos para identificar a finalidade de cada entrada e saída. Pode-se agora fechar o subsistema quando este estiver concluído. Não é necessário salvá-lo antes de fechar, pois o subsistema é parte do modelo e é salvo toda vez que o modelo for salvo.

**Exemplo**

Suponha que se deseja modelar um sistema massa-mola composto de carros conectados como mostra a figura a seguir.



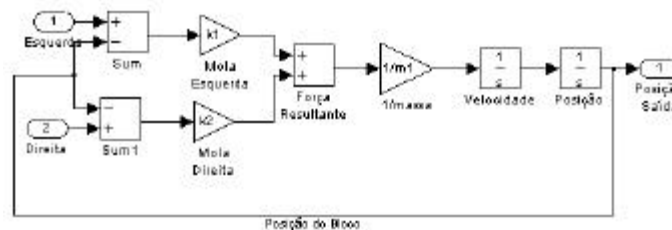
O modelo será construído a partir de blocos subsistemas que irão modelar cada carro, como na figura:



A equação do movimento para um carro é:

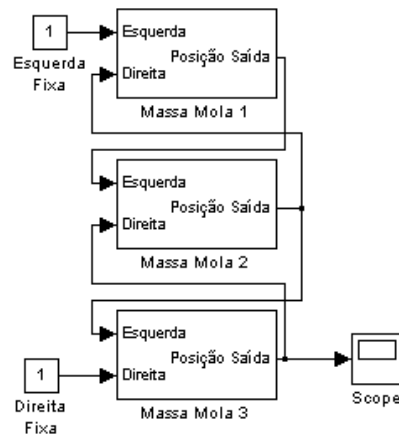
$$\ddot{x}_n = \frac{1}{m_n} [k_n (x_{n-1} - x_n) + k_{n+1} (x_n - x_{n+1})]$$

Utilizando o procedimento discutido anteriormente, constrói-se o subsistema mostrado na figura.



Este subsistema irá modelar o carro 1. As entradas de um único carro são  $x_{n-1}$  (posição do carro da esquerda) e  $x_{n+1}$  (posição do carro da direita). A saída do subsistema é  $x_n$  (posição do carro). Deve-se notar que cada mola está relacionada a dois blocos tipo subsistema, ao da esquerda e ao da direita da mola.

Estando o subsistema completo, pode-se fechar sua janela. A seguir é necessário fazer duas cópias do bloco subsistema e conectá-las como na figura:



Neste caso é conveniente entrar com as constantes da mola ( $k_1$ ,  $k_2$  e  $k_3$ ) e as massas dos carros ( $m_1$ ,  $m_2$  e  $m_3$ ) como variáveis MATLAB, e determinar seus valores utilizando um arquivo `.m` do MATLAB (nomeado `set_x4a.m`). A figura a seguir ilustra as linhas de comando que devem constar no arquivo:

```
% Configura as constantes das molas e as massas dos carros

k1 = 1;

k2 = 2;

k3 = 4;

m1 = 1;

m2 = 3;

m3 = 2;
```

Este arquivo deve ser executado na área de trabalho do MATLAB antes da execução da simulação. Note que o arquivo tem a extensão `.m` e deve ter um nome diferente daquele do modelo SIMULINK. Por exemplo, se o modelo SIMULINK for designado por `exemp_1.m`, o MATLAB irá abrir o modelo SIMULINK quando você digitar o comando `exemp_1` na área de trabalho do MATLAB.

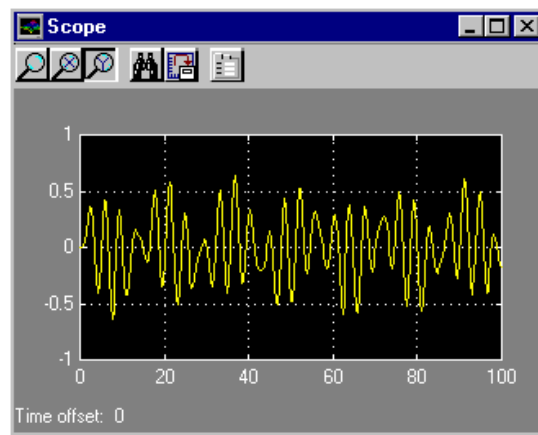
Os parâmetros dos blocos de cada cópia do subsistema devem ser agora configurados. Para o carro 1, o valor do ganho do bloco de ganho Mola da Esquerda deve ser  $k_1$  e o do bloco de ganho Mola da Direita  $k_2$ . A seguir, o bloco 1/massa

deve ser configurado para ter o ganho  $1/m_1$ . A velocidade inicial deve ser 0 e a posição inicial 1.

Para o carro 2, o valor de ganho para o bloco Mola da Esquerda deve ser  $k_2$  e o do bloco Mola da Direita  $k_3$ . O ganho do bloco  $1/\text{massa}$  deve ser  $1/m_2$ . A velocidade inicial deste bloco é 0 e a posição inicial é também 0.

Para o carro 3, o valor de ganho do bloco Mola da Esquerda deve ser  $k_3$  e o bloco Mola da Direita 0, já que não há mola à direita deste carro. O ganho do bloco  $1/\text{massa}$  deve ser  $1/m_3$ . A velocidade inicial é configurada para 0 e a posição inicial 0.

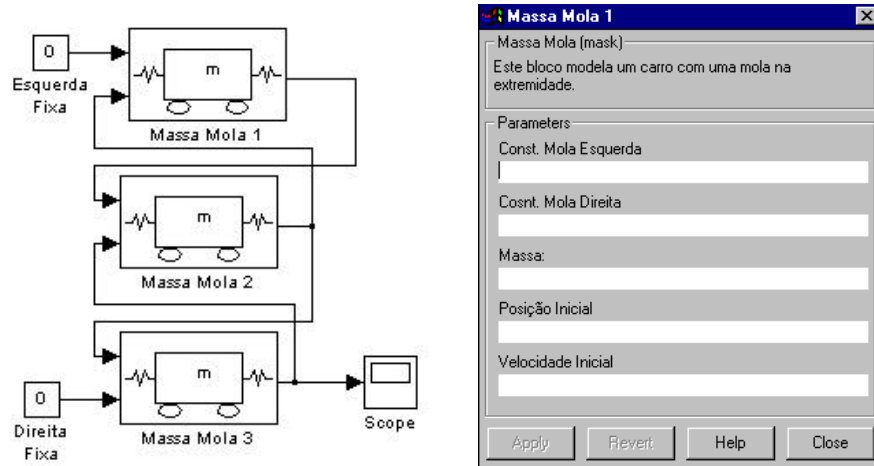
O bloco osciloscópio deve ser configurado para salvar os dados mostrados para a área de trabalho do MATLAB, com o tempo inicial 0 e o final 100. Após executar a simulação, os dados do osciloscópio podem ser plotados no MATLAB, resultando na trajetória do carro 3 mostrado logo a seguir:



### 6.2 - Blocos com Máscara

Mascarar blocos é uma capacidade do SIMULINK que estende o conceito de abstração. Máscaras permitem ao usuário tratar um subsistema como um bloco simples. Um bloco com máscara deve ter um ícone próprio, e uma caixa de diálogo com parâmetros de configuração para entrada de dados assim como os blocos das bibliotecas do SIMULINK. Os parâmetros de configuração devem ser usados diretamente para inicializar os blocos pertencentes ao subsistemas, ou então utilizados para calcular dados para inicializar os blocos.

Para se entender os conceitos de máscaras, considere o modelo mostrado na figura:



Este modelo é equivalente ao modelo do exemplo anterior, porém muito mais fácil de se entender. Um clique duplo no bloco de nome Massa-Mola 1 abre a caixa de diálogo mostrada. Ao invés de se abrir a caixa de diálogo de cada bloco de ganho e cada integrador para configurar os parâmetros, o usuário pode entrar com todos os parâmetros do subsistema com a caixa de diálogo do próprio subsistema.

Esta seção explica os passos para se criar um subsistema com máscara. Os exemplos irão mostrar como criar um subsistema massa-mola com máscara. Exemplos adicionais ilustrarão outras características de máscaras.

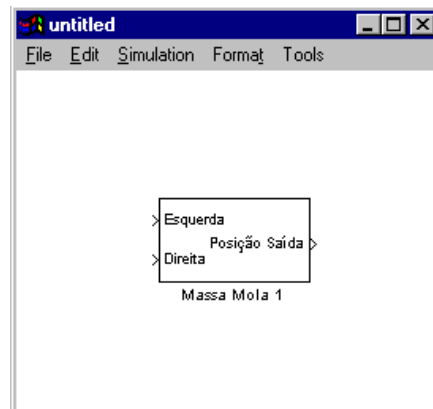
O processo de produção de um bloco com máscara pode ser resumido em:

1. Construir um subsistema como já discutido.
2. Selecionar o bloco subsistema e clicar-se em Edit>Create Mask na barra de menu do modelo.
3. Utilizar o editor de máscara (Mask Editor) para configurar a documentação da máscara, a caixa de diálogo e opcionalmente um ícone próprio para o bloco.

### 6.2.1 - Convertendo um Subsistema em um Sistema com Máscara.

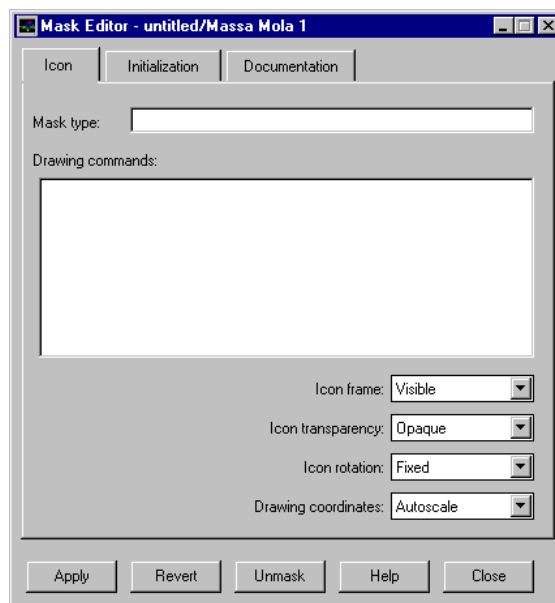
O primeiro passo na criação de um subsistema com máscara é criar um subsistema utilizando procedimentos descritos anteriormente. Para ilustrar o processo, será construído um bloco Massa-Mola com máscara a partir de um dos subsistemas no modelo do exemplo anterior.

Abra o modelo do exemplo anterior. A seguir, abra uma nova janela de modelo. Arraste uma cópia do bloco Massa-Mola 1 para a nova janela de modelo.



Selecione o bloco e clique em Edit>Create Mask na barra de menu do modelo.

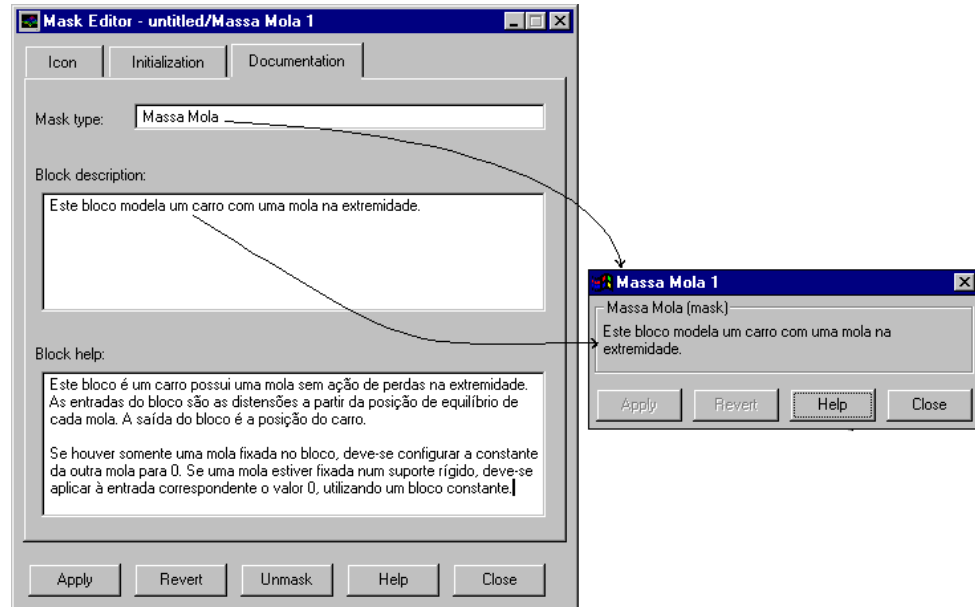
A caixa de diálogo do editor de máscara irá aparecer. Note que o editor de máscaras tem três páginas. Cada página será discutida em subseções.



Antes de continuar, deve-se salvar o modelo utilizando o nome spm\_msk.

### 6.2.2 - Página de Documentação do Editor de Máscaras

A página de documentação mostrada na figura a seguir contém três campos. Todos os campos nesta página são opcionais.



Após preencher os três campos, clique em Close. A partir deste momento, um duplo clique no bloco Massa-Mola irá abrir a caixa de diálogo mostrada na figura. Pode-se retornar ao editor de máscaras clicando em Edit:Edit Mask na barra de menu.

Cada campo desta página será detalhado a seguir.

#### 6.2.2.1 - Campo Tipo de Máscara

O conteúdo deste primeiro campo será mostrado no título da caixa de diálogo do bloco com máscara. Deve-se notar que há dois títulos na parte superior esquerda da caixa. O nome na barra de título é o nome do bloco selecionado. O nome na parte interna da caixa é o tipo do bloco.

#### 6.2.2.2 - Campo Descrição do Bloco

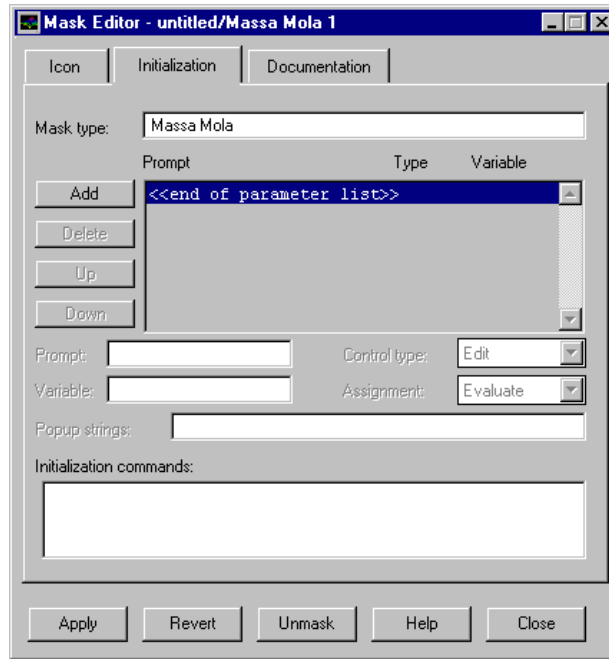
O segundo campo é mostrado na área interna da caixa. Este campo deve conter uma breve descrição do bloco, seu propósito e qualquer lembrete necessário à utilização do bloco.

#### 6.2.2.3 - Bloco de Auxílio (Help)

O conteúdo do terceiro campo será mostrado pelo help do MATLAB quando o botão de help do bloco for pressionado. Este campo deve conter informações detalhadas a respeito do uso, configuração e limitações do bloco com máscara e dos subsistemas internos ao bloco.

### 6.2.3 - Página de Inicialização do Editor de Máscara

A página de inicialização é usada para configurar parâmetros do bloco e dos subsistemas internos ao bloco.



Esta página pode ser dividida em três seções.

A primeira contém o campo Mask Type.

A seção central contém um grupo de campos que definem os campos na caixa de diálogo do bloco, a variável local correspondente a cada campo.

A seção inferior contém o campo Comandos de Inicialização. Este campo deve ser usado para definir variáveis adicionais para configurar parâmetros na caixa de diálogo ou na página de Ícone, discutida mais tarde.

#### 6.2.3.1 - Campo Tipo de Máscara

A parte superior desta página contém o campo Tipo de Máscara, que é idêntico ao campo de mesmo nome na página Documentação. Este campo pode ser definido ou editado nesta página, ou em outras páginas do editor de máscara. Modificando em qualquer página, a mudança será feita em todas as outras páginas.

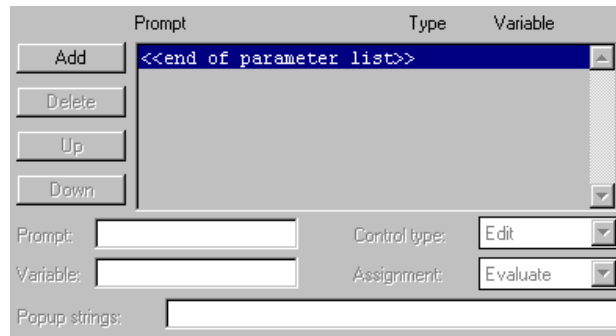
#### 6.2.3.2 - Seção de Prompt da Caixa de Diálogo do Bloco

A seção central desta página é usada para criar, editar e deletar campos da caixa de diálogo que se abre quando se dá um duplo clique no bloco. Contém uma lista dos campos contidos na caixa, botões para adicionar, deletar e mover campos e ainda mais cinco campos usados para configurar a caixa de diálogo.

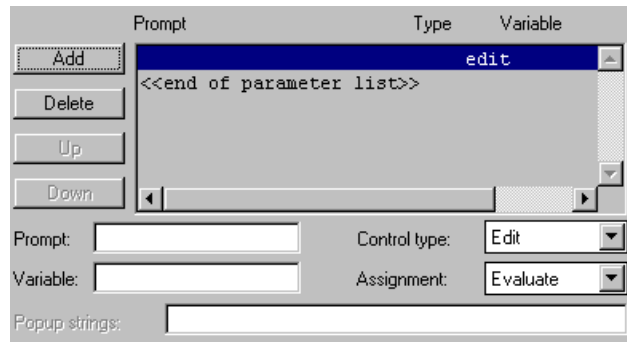
Para adicionar o primeiro campo no bloco Massa-Mola deve-se seguir os passos:

Selecione o bloco e abra o editor de máscara clicando em Edit:Edit Mask na barra de menu do modelo.

Clique em <<end of parameter list>> e a seguir clique em Add

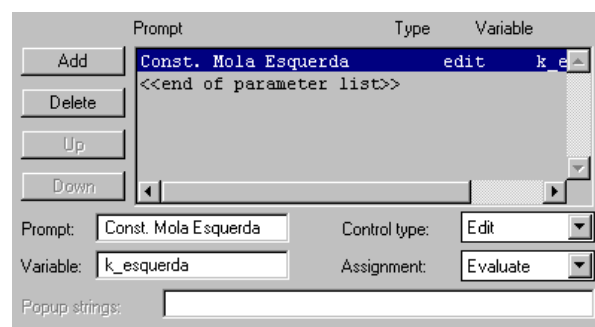


Uma linha em branco será inserida na lista de parâmetros.

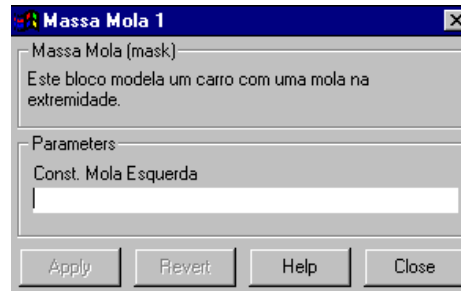


No campo Prompt digite: Constante da mola da esquerda.

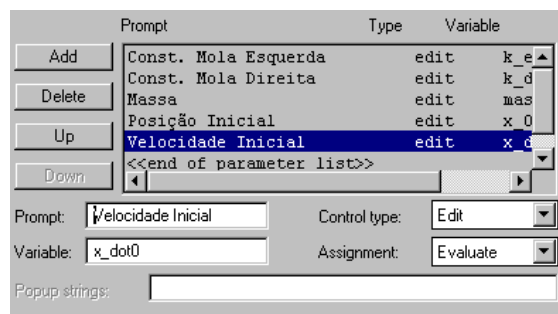
No campo Variable field digite: k\_esquerda. Note que o prompt e o nome da variável aparecem na lista de parâmetros.



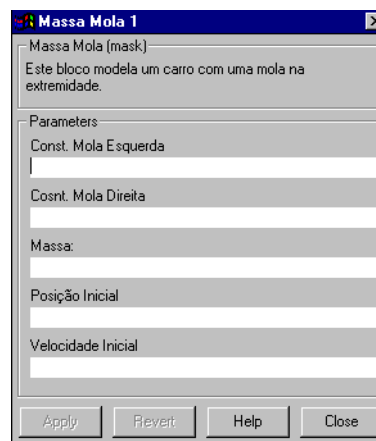
Clique a seguir em Close. Com um duplo clique no bloco Massa –Mola a caixa de diálogo irá abrir contendo agora o campo para a entrada da variável k\_esquerda.



Acrescente agora as variáveis massa com o prompt Massa,  $k_{direita}$  com o prompt Constante da Mola Direita,  $x_0$  com o prompt Posição Inicial e  $\dot{x}_0$  com o prompt Velocidade Inicial.



Estes passos completam o processo de criação dos campos da caixa de diálogo dos blocos. Os resultados podem ser vistos com um duplo clique no bloco.



Os botões e campos desta página serão agora discutidos com mais detalhes.

Há quatro botões à esquerda da lista. Estes botões são usados para adicionar, apagar e reorganizar a caixa de diálogo. Para criar um novo prompt deve-se clicar na posição em que se deseja que a nova variável seja criada antes anteriormente à esta posição. Se o novo prompt deve ser criado no fim da caixa de diálogo basta clicar em <<end of parameter list>> e a seguir clicar em Add. Uma linha em branco irá aparecer na lista. O botão Delete apaga a linha seleccionada. Os botões Up e Down são utilizados para mover as linhas para se encontrar a disposição adequada dos campos na caixa de diálogo.

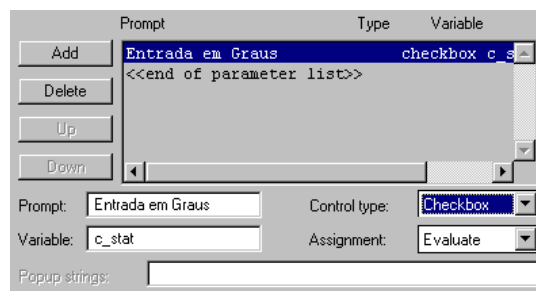
O campo Control type é uma lista que contém três opções: **Edit**, **Checkbox** e **Popup**. **Edit** produz um campo de entrada de dados. É o tipo de campo mais comum. **Checkbox** produz um campo que possui somente duas possibilidades de valores, dependendo da condição da caixa, se está selecionada ou não. **Popup** produz uma lista de escolhas contidas no campo Popup strings.

O valor atribuído à variável interna associada com o campo na caixa de diálogo do bloco dependerá do conteúdo do campo Assignment. Deve ser configurado como Evaluate ou Literal. Se a primeira opção for a escolhida, a variável associada com o campo irá conter o valor da expressão no campo. Por exemplo, se o campo contiver k1 e esta variável for definida na área de trabalho do MATLAB com o valor 2.0, a variável associada com este campo irá assumir o valor 2.0. Se for escolhida a segunda opção, a variável associada com o campo assumirá a string 'k1'.

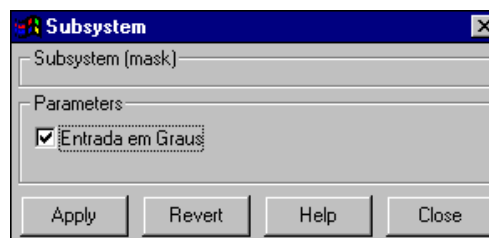
Escolher Checkbox no campo Control type produz uma caixa de checagem na caixa de diálogo. A variável associada irá assumir dois valores dependendo da escolha feita em Assignment. Se a escolha for Evaluate a variável poderá assumir 0 se a caixa não for marcada ou 1 se a caixa for marcada. Se a escolha for Literal a variável poderá então assumir 'no' se a caixa não for marcada ou 'yes' se a caixa for marcada.

### Exemplo

Suponha que se deseje configurar um subsistema com máscara de tal forma que sua caixa de diálogo possua uma caixa de checagem permitindo especificar as entradas angulares em graus ou radianos. O valor da variável associada com a caixa (c\_stat) deve ser 0 se a caixa não for marcada e 1 se a caixa for marcada. A figura que se segue mostra como deve ser a página de inicialização para produzir o efeito desejado.



Neste exemplo o campo Mask type contém Exemplo Check box e o campo Block description Este bloco ilustra uma opção em check box.



Se o campo Control type for configurado para Popup, o campo Popup strings é usado para definir uma lista de escolhas. A variável associada ao campo irá assumir um valor dependendo da escolha em Assignment.

Se Assignment for configurado para Evaluate, a variável associada ao campo assumirá o número ordinal selecionado. Se, por exemplo, for selecionada a primeira opção, o valor da variável será 1. Se a escolha for a segunda opção, a variável assumirá então 2. Se Assignment for configurado para Literal, a variável irá conter o valor correspondente à opção selecionada.

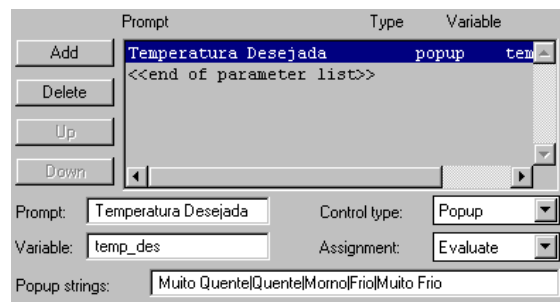
A lista de opções deve estar no campo Popup strings. As opções devem ser digitadas em seqüência separados por "|". Por exemplo se as opções forem muito quente, quente, morno, frio e gelado, no campo Popup strings deve conter Muito quente | Quente | Morno | Frio | Gelado.

### Exemplo

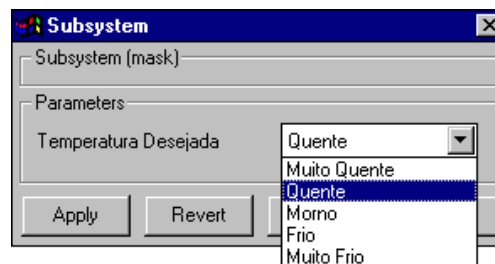
Suponha que se deseja agora criar um bloco com máscara que produza um sinal de saída definido numa lista que contenha as opções Muito quente, Quente, Morno, Frio e Gelado.

Inicialmente abre-se uma nova janela de modelo e arrasta-se um bloco Constante para a janela do modelo. Seleciona-se então o bloco e clicando a seguir em Edit>Create Subsystem na barra de menu da janela. Seleciona-se agora o subsistema e clicando em seguida em Edit>Create Mask. O campo Mask type deve conter exemplo Popup e Block description: Este bloco ilustra a opção Popup.

Configura-se então a página de Inicialização do editor de máscara como mostra a figura:



A seguir, com um duplo clique no bloco, abre-se a caixa de diálogo. Clicando na caixa de opção pode-se selecionar a escolha desejada.



### 6.2.3.3 - Comandos de Inicialização

A seção inferior da página de inicialização é o campo Initialization commands. Este campo pode conter uma ou mais declarações na sintaxe do MATLAB que atribuem valores a variáveis do MATLAB usadas para configurar blocos no subsistema com máscara. Nas declarações podem ser usados quaisquer operadores do MATLAB, funções internas ou externas e funções de controle de fluxo como if, while e end. O grupo de variáveis utilizadas neste campo são locais; variáveis definidas na área de trabalho do MATLAB não são acessíveis neste campo.

O campo Initialization commands mostra inicialmente 4 linhas mas pode conter o qualquer número de linhas que seja necessário. Pode-se navegar pelas linhas utilizando as teclas do cursor.

Cada comando neste campo deve ser terminado por um ponto-e-vírgula (;). Se for omitido, em um comando, o resultado deste comando será mostrado na área de trabalho do MATLAB cada vez que o comando for executado. Isto pode ser conveniente para a depuração na execução da simulação.

#### Exemplo

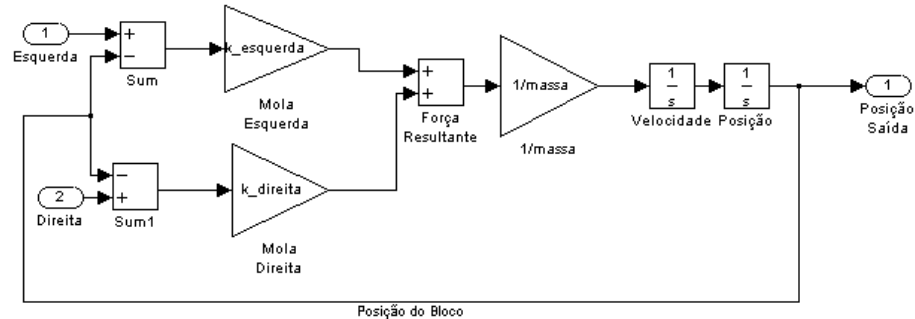
O ícone do bloco Massa-mola necessita de duas rodas. Para preparar o desenho das rodas, deve-se criar dois vetores, um contendo as coordenadas – x de um pequeno círculo e outro as coordenadas – y. Estas variáveis serão usadas na Página Ícone (Icon Page) para desenhar as rodas.

```
Initialization commands:
t=[0:0.5:2*pi];
x=cos(t);
y=sin(t);
```

### 6.2.3.4 - Configurando os Blocos do Subsistema

Os blocos no subsistema com máscara devem ser configurados para o uso de variáveis definidas na página de inicialização. Para configurar os blocos no subsistema Massa-Mola, seleciona-se o subsistema, a seguir escolhe-se Edit:Look Under Mask na barra de menu da janela do modelo. Com um clique duplo no bloco de ganho de nome Mola Esquerda, e configura-se o ganho para k\_esquerda. Repete-se o procedimento para o bloco Mola Direita, com o ganho k\_direita, assim como o bloco 1/mass para 1/mass. A condição inicial para o integrador Velocidade deve ser x\_dot0 e do integrador Posição x0.

O subsistema deve agora estar semelhante ao mostrado a seguir. Deve-se agora fechar a janela do subsistema e salvar o modelo.



### 6.2.3.5 - Variáveis Locais

Uma diferença importante entre subsistemas com e sem máscara está no conjunto de variáveis na caixa de diálogo para os blocos do subsistema.

Blocos em subsistemas sem máscara podem usar qualquer variável MATLAB definida na área de trabalho do MATLAB. Esta característica é usada para inicializar subsistemas como no primeiro exemplo deste capítulo. Os blocos em subsistemas com máscara não podem acessar variáveis da área de trabalho do MATLAB. Um bloco num subsistema com máscara possui suas próprias variáveis com nomes próprios, independentes da área de trabalho do MATLAB e qualquer outro subsistema num modelo SIMULINK. Isto é uma característica extremamente valiosa de subsistemas com máscaras, pois elimina a possibilidade de conflitos de nomes não intencionais.

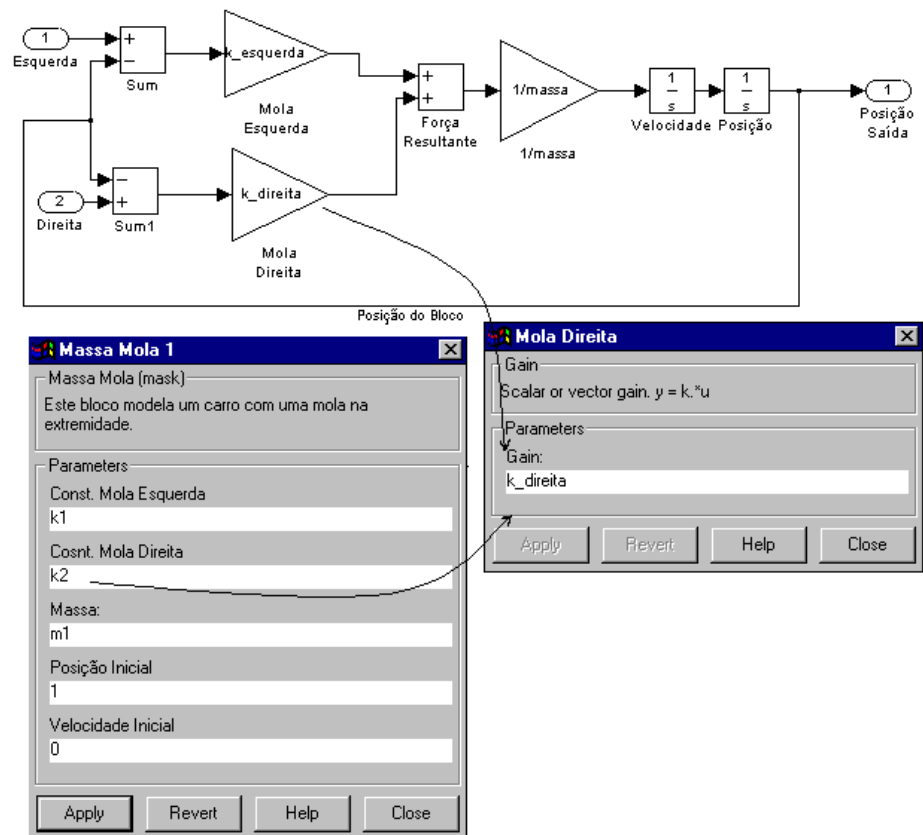
As variáveis internas de um subsistema com máscara são criadas e assumem valores definidos no Editor de Máscaras usando comandos de inicialização. Cada campo define uma variável interna acessível somente no subsistema. As variáveis internas adicionais devem ser definidas no campo Initialization Commands na página de inicialização.

As conexões entre a área de trabalho do MATLAB e um subsistema com máscara são o conteúdo dos campos das caixas de diálogo dos blocos contidos no subsistema. Um campo de entrada numa caixa de diálogo deve conter constantes ou expressões usando variáveis definidas na área de trabalho do MATLAB. O valor deste conteúdo é atribuído à variável interna associada a este campo. Esta variável interna pode ser usada para definir outras variáveis internas no campo Initialization Commands.

O modelo Massa-mola com três carros mostrado anteriormente possui subsistemas configurados como mostrado na figura anterior. A constante da mola para a mola da esquerda em cada instante é definida como `k_esquerda`. Porém, o conteúdo do campo `k_esquerda` em cada caso é diferente. No primeiro bloco este valor assume `k1`, `k_esquerda` para o segundo bloco deve ser `k2` e o terceiro bloco deve ser `k3`.

## Exemplo

Para ilustrar o uso de variáveis internas num subsistema com máscara, considere a atribuição de um valor à constante da mola direita no subsistema do carro. Na caixa de diálogo, o campo Constante da Mola da Direita está associado à variável interna `k_direita`. Na figura que se segue têm-se o bloco Ganho de nome Mola da Direita é configurado como `k_direita`. Quando os comandos já mencionados forem executados na área de trabalho, o valor do ganho para o bloco de ganho Mola da Direita assume o valor particular de 2.



### Exemplo

Para o subsistema com máscara do exemplo das temperaturas, deseja-se configurar o valor da variável `temp_val` como a seguir:

Opção do Menu	<i>Temp_val</i>
Muito quente	120
Quente	100
Morno	85
Frio	70
Gelado	50

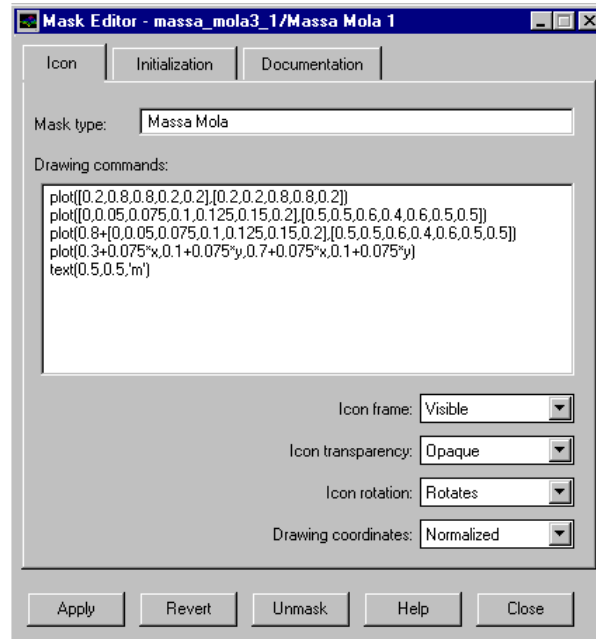
Para executar esta tarefa, deve-se inserir os seguintes comandos no campo Initialization Commands.

```
temp_list = [120,100,85,70,50];  
temp_val = temp_list(temp_des);
```

A primeira linha de comando cria um vetor de temperaturas. A segunda linha usa a variável `temp_des` como índice deste vetor. Clica-se a seguir em Close e a seguir com o subsistema selecionado Edit:Look Under Mask e configura-se o bloco Constant para a string `temp_val`.

### 6.2.4 - Página de Ícone do Editor de Máscara

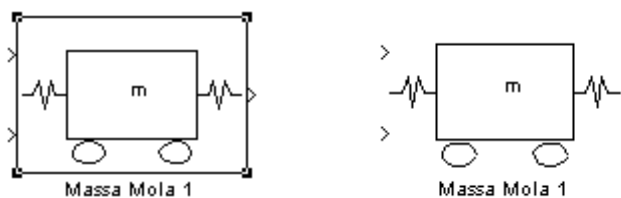
Esta página permite ao usuário personalizar ícones para seus blocos com máscara. É usada para criar e personalizar o ícone no bloco do carro para o sistema Massa-Mola (lembrando que `x` e `y` foram definidos no campo Initialization Commands). Esta página consiste de seis campos. O primeiro campo é idêntico ao campo Mask Type já mencionado em outras páginas. Drawing commands é um campo de linhas múltiplas no qual devem ser digitados comandos do MATLAB para desenhar e nomear o ícone. A figura a seguir mostra os comandos necessários para se desenhar o ícone do bloco Massa-Mola.



O objetivo dos quatros campos a seguir é configurar o ícone. Explicar o campo Drawing Commands se torna fácil se antes forem discutidos estes campos de configuração.

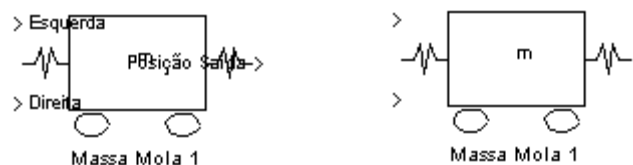
### 6.2.4.1 - Campo Moldura do Ícone (Icon Frame)

O primeiro campo de configuração é uma lista contendo duas opções: Visível e Invisível. A figura a seguir ilustra o bloco Massa-Mola com e sem moldura.



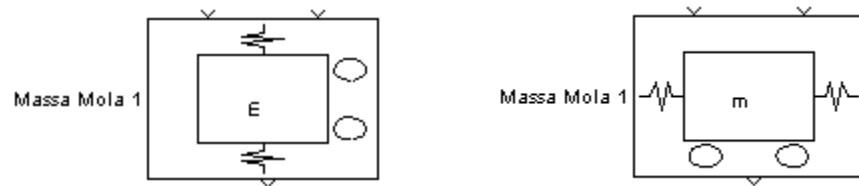
### 6.2.4.2 - Campo Transparência do Ícone

O segundo campo de configuração do ícone é uma lista também com duas opções: Transparente e Opaco. A figura mostra o bloco com as duas configurações possíveis para este ícone. Note que com a opção transparente selecionada os nomes dos blocos Inport e Outport aparecem. Ao selecionar Opaco, estes nomes ficam escondidos.



### 6.2.4.3 - Campo Rotação do Ícone

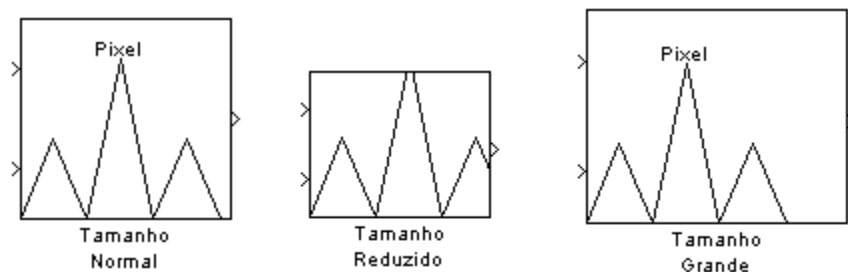
O terceiro campo é uma lista com as opções: Fixo e Rotacionar. Este campo determina o comportamento do ícone quando os comandos Edit:Flip block e Format:Rotate block são executados. Se a opção escolhida for fixa, quando o bloco for rotacionado ou invertido o ícone original conserva a mesma orientação do bloco. A figura mostra as diferentes situações. A opção Fixa é a mais usada, particularmente em blocos que contenham textos.



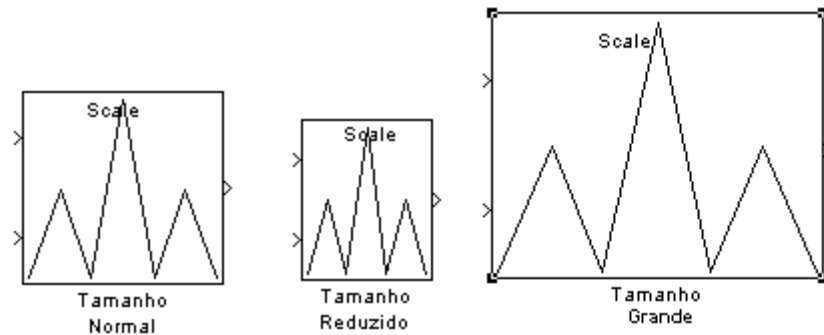
### 6.2.4.4 - Campo Coordenadas de Desenho (Drawing coordinates)

O campo final de configuração do ícone determina a escala usada na plotagem dos gráficos do ícone e a posição do texto no ícone. O campo é uma lista com três opções: Pixel, Autoescale, e Normalized.

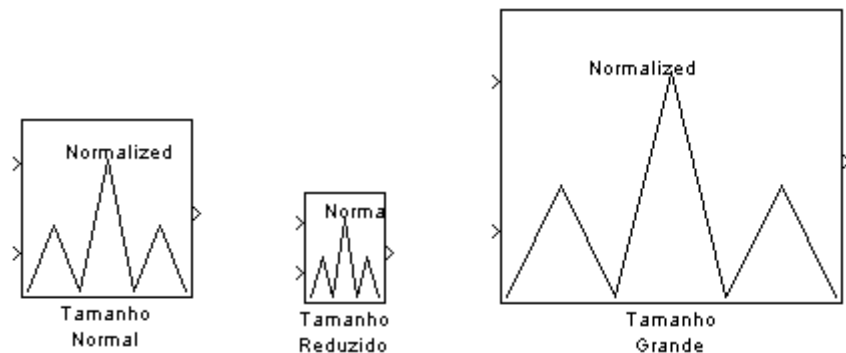
Pixel é uma escala absoluta e irá resultar em um ícone que não pode ser redimensionado. As coordenadas do canto inferior esquerdo são (0,0). As unidades são pixels, e o tamanho de um ícone irá depender somente da resolução do monitor. A figura mostra o resultado com a opção Pixel.



A opção Autoescale ajusta o tamanho do ícone para que caiba exatamente na moldura do bloco. A figura mostra um bloco com sua máscara com a opção autoescale. Note que o texto no ícone não muda de tamanho quando o bloco é redimensionado.



Normalized especifica que a escala do desenho assume de 0.0 a 1.0 tanto no eixo vertical quanto no horizontal. As coordenadas do canto inferior esquerdo do ícone são (0,0) e do canto superior direito (1,1). Quando o bloco é redimensionado, as coordenadas também são redimensionadas. O texto não modifica com o tamanho do ícone.



#### 6.2.4.5 - Comandos de Desenho (Drawing Commands)

Vários comandos MATLAB podem ser inseridos neste campo para se personalizar o ícone do bloco. A tabela a seguir lista estes comandos:

Comando	Descrição
<code>disp(string)</code>	Mostra a <i>string</i> no centro do ícone
<code>text(x,y,string)</code>	Mostra a <i>string</i> com início em (x,y)
<code>fprintf(string,list)</code>	Mostra o resultado do comando <code>fprintf</code> no centro do ícone
<code>plot(x_vetor,y_vetor)</code>	Desenha um gráfico no ícone
<code>dpoly(num,denom)</code>	Mostra uma função de transferência no centro do ícone
<code>dpoly(num,denom,'z')</code>	Mostra uma função discreta de transferência em potências ascendentes de z
<code>dpoly(num,denom,'z^-')</code>	Mostra uma função discreta de transferência em potências descendentes de z
<code>droots(zeros,poles,gain)</code>	Mostra uma função de transferência no formato zeros-pólos-ganhos

Três dos comandos mostram textos no ícone. O mais simples, `disp(string)`, mostra uma cadeia de caracteres (string) no centro do ícone. Este comando é muito útil para se inserir uma descrição simples no centro do ícone. O comando `text(x,y,string)` permite colocar uma string em qualquer lugar do ícone, usando o sistema de coordenadas especificado no campo Drawing coordinates. O terceiro comando de texto, `fprintf`, é idêntico ao comando `fprintf` do MATLAB (veja no help do MATLAB as opções deste comando). Usando este comando pode-se inserir strings definidas nas caixas de diálogos ou no campo Initialization Commands na página de inicialização. Para acrescentar uma nova linha de texto, pode-se usar a opção (`\n`), produzindo textos de múltiplas linhas. Como o comando `disp`, `fprintf` coloca o texto no centro do ícone.

Uma cadeia de caracteres usada para visualizar um texto num ícone deve ser uma string literal ou então uma variável string do MATLAB. Uma string literal é uma seqüência de caracteres visualizáveis cercada por apóstrofos simples. Por exemplo, para inserir o nome "Bloco Especial" no centro do ícone do bloco, deve-se fazer uso do comando

```
Disp('Bloco Especial')
```

Uma variável string é uma variável do MATLAB que representa uma cadeia de caracteres ao invés de um número. O MATLAB fornece várias funções para criação e manipulação de variáveis strings. Estas funções podem ser usadas no campo Initialization Commands na página de inicialização para criar strings para uso dos comandos de exibição. Uma função muito útil é `sprintf`. Este comando é muito similar ao `fprintf`, mas escreve a string em uma variável ao invés de escrever na tela ou em um arquivo.

### Exemplo

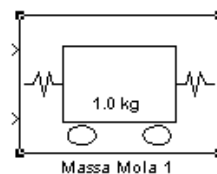
Suponha que se deseja mudar o ícone do carro no sistema Massa-Mola para que a massa do carro seja mostrada no ícone acima das rodas, em unidades de massa (kg). Adiciona-se então a seguinte linha de comando no campo Initialization Commands na página de inicialização do editor de máscaras.

```
b_label = sprintf('%1.1f kg',m);
```

A seguir adiciona-se a seguinte linha de comando no campo Drawing Commands na página de ícone.

```
text(0.3,0.35,b_label);
```

O bloco deve então ficar como o da figura:



O comando `plot(x_vetor,y_vetor)` mostra gráficos no ícone do bloco. Este comando é similar ao comando `plot` do MATLAB, mas tem poucas opções. O comando `plot` do editor de máscaras não suporta opções para se configurar estilos de linhas, cores, e não irá plotar matrizes de duas dimensões. Este comando espera pares de vetores especificando seqüências de coordenadas  $x$  e  $y$ . Pode haver mais de um par de vetores em um simples comando `plot`, e pode também haver mais de um comando `plot` para a criação de um ícone.

### Exemplo

Suponha que se deseja mostrar funções seno e cosseno num ícone. Os seguintes comandos devem então ser acrescentados no campo Initialization Commands para produzirem os vetores necessários.

```
x_vetor = [0:0.05:1] ;
```

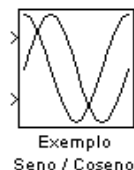
```
y_sin = 0.5 + 0.5*sin(2*pi*x_vetor) ;
```

```
y_cos = 0.5 + 0.5*cos(2*pi*x_vetor) ;
```

A seguir, acrescenta-se o seguinte comando ao campo Drawing Commands:

```
plot(x_vetor,y_sin,x_vetor,y_cos)
```

O bloco deve agora ter a aparência:



### Exemplo

O bloco operador lógico pode ser configurado para implementar portas AND ou OR, mas o ícone do bloco continua sendo um retângulo com a função lógica descrita no ícone. Substituindo cópias destes blocos por subsistemas com máscaras, pode-se produzir blocos com aparência semelhante à convenção utilizada para estas portas.

Para produzir a porta AND, deve-se iniciar com um bloco Operador Lógico configurado para implementar a função AND. Seleciona-se o bloco e clica-se em `Edit>Create subsystem` na barra de menu da janela do modelo. A seguir clica-se em `Edit>Create mask`. Na página de inicialização do editor de máscaras deve-se acrescentar no campo Initialization Commands a seguinte linha:

```
t = -pi/2:0.1:pi/2;
```

O campo Icon Frame deve estar configurado como Invisible e Drawing coordinates como Normalized. No campo Drawing commands deve conter as linhas:

```
plot([0.5 0 0 0.5],[0 0 1 1],0.5+0.5*cos(t),0.5+0.5*sin(t))  
text(0.05,0.65,'a');  
text(0.05,0.2,'b');  
text(0.75,0.45,'ab');
```

Para produzir a porta OR, o procedimento é similar, substituindo somente as linhas em Drawing Commands:

```
plot([0 0],[0 1],t,0.5*t.^2,t,1-0.5*t.^2);  
text(0.05,0.65,'a');  
text(0.05,0.2,'b');  
text(0.75,0.45,'a+b');
```

Os blocos Portas Lógicas devem agora estar como:



Os comandos finais, `dpoly(num,denom)` e `droots(zeros,poles,gain)`, mostram uma função de transferência no ícone do bloco.

`dpoly` mostra uma função de transferência na forma polinomial. Os argumentos `num` e `denom` são vetores contendo os coeficientes do numerador e do denominador da função de transferência em funções descendentes de `s`. `dpoly` somente irá mostrar funções de transferências em potências descendentes de `z` ou ascendentes de `1/z`. Para mostrar funções em potências descendentes de `z`, deve-se utilizar o comando `dpoly(num,denom,'z')`. Para mostrar funções em potências ascendentes de `1/z`, deve-se usar `dpoly(num,denom,'z-')`.

`droots` mostra funções de transferências na forma fatorada de zeros-pólos. `zeros` é um vetor contendo os zeros da função de transferência (raízes do numerador), e `poles` é um vetor contendo os pólos da função de transferência (raízes do denominador). O ganho deve ser um escalar.

### 6.2.5 - Olhando sob a Máscara e Removendo Máscaras

Existem dois comandos adicionais para se lidar com máscaras. Eles permitem que se observe um subsistema sob a máscara e que se apague a máscara.

Para examinar um subsistema com máscara, seleciona-se o bloco e a seguir clica-se em `Edit:Look under mask`.

Para converter um bloco com máscara em um bloco sem máscara seleciona-se o bloco abrindo o editor de máscaras logo em seguida. Clica-se no botão Unmask na parte inferior do editor. Se o usuário mudar de idéia após remover a máscara, clicando-se em Edit>Create mask, as informações da antiga máscara estarão disponíveis. Mas, uma vez fechado o modelo após remover a máscara, torna-se impossível recuperar tais informações.

### 6.2.6 - Criando uma Biblioteca de Blocos

Cada biblioteca do SIMULINK é um subsistema com máscara contendo um número de blocos do SIMULINK não conectados entre si. Para verificar isto selecione o bloco da biblioteca de Fontes clicando a seguir em Edit>Edit Mask. As bibliotecas do SIMULINK são modelos contendo inúmeros blocos, nenhum dos quais contendo entradas ou saídas.

O usuário pode criar suas próprias bibliotecas. A maneira mais simples de fazer isto é copiar blocos para uma janela de modelo vazia e salvar o modelo a seguir. Para usar esta biblioteca deve-se clicar em File:Open no SIMULINK ou entrar com o nome da biblioteca na área de trabalho do MATLAB.

O usuário pode criar uma biblioteca a partir de um subsistema contendo um ou mais blocos não conectados com linhas de sinais. Estes subsistemas podem ter máscaras para ficarem semelhantes às bibliotecas do SIMULINK. O subsistema com máscara pode ter ícones personalizados, mas não podem conter caixa de diálogo (a seção de prompt na página de inicialização deve ficar vazia), e o campo de descrição do bloco (Bloco description field) deve ficar em branco. Se isto não acontecer, ao se dar um duplo clique no ícone da biblioteca, ao invés de abrir a biblioteca contendo os blocos, aparecerá uma caixa de diálogo.

## 6.3 - Subsistemas com Execução Condicionada

A biblioteca de Conexões fornece dois blocos que podem causar a um subsistema execução condicionada. O bloco Habilita (Enable) faz com que um subsistema seja executado somente se uma entrada de controle for positiva. O bloco de disparo (Triggered) é um bloco de subsistema que causa a execução do subsistema quando o sinal de disparo for recebido. Acrescentar estes dois blocos ao subsistema faz com que este seja executado toda vez que o sinal de disparo for recebido somente quando a entrada de habilitação for positiva.

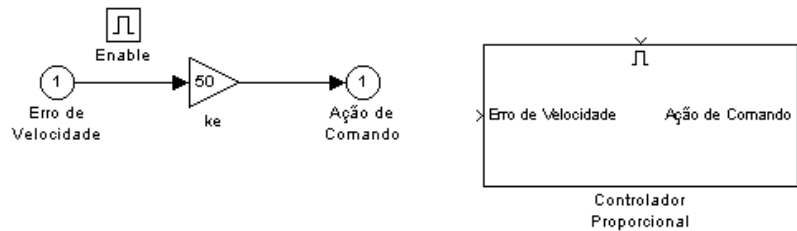
### 6.3.1 - Subsistemas com Habilitação (Enabled)

Estes subsistemas permitem ao usuário modelar sistemas que possuem vários modos de operação ou fases. Por exemplo, a aerodinâmica de um avião supersônico na configuração de decolagem é diferente de sua aerodinâmica para o mesmo avião num vôo supersônico. O sistema de controle digital de vôo para um avião deste tipo emprega diferentes algoritmos de controle para diferentes regimes de vôo. Um modelo SIMULINK do avião e do sistema de controle pode precisar incluir os dois regimes de vôo. É possível modelar este tipo de sistema usando blocos lógicos ou blocos com chaveamento. Se for usada esta aproximação, todo bloco do modelo será estimado a cada passo de integração. A inclusão destes blocos não contribuem para o desempenho da simulação.

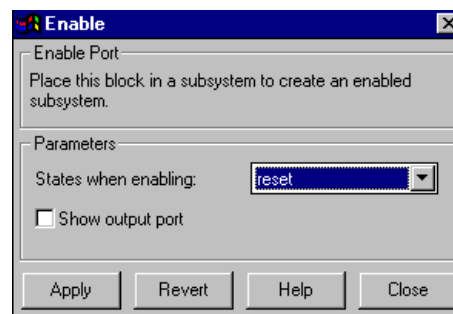
Se o usuário converter os vários subsistemas contendo as dinâmicas de vôo em subsistemas com opção de habilitação, somente os subsistemas ativos serão

calculados durante uma simulação particular. Isto pode significar uma melhoria computacional significativa.

Um subsistema é convertido em um subsistema com opção de habilitação com a adição do bloco Enable da biblioteca de Conexões. A figura que se segue ilustra um controlador proporcional simples convertido em um subsistema com opção de habilitação.

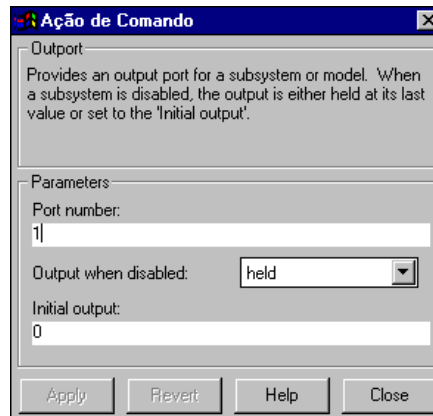


A caixa de diálogos do bloco de habilitação é a que se segue:



A caixa possui dois campos. O primeiro campo (States when enabling) permite escolher a situação do subsistema quando este é habilitado. É uma caixa com duas opções: reset e held. A primeira opção faz com que os estados internos do subsistema seja resetado às suas condições iniciais cada vez que o bloco é habilitado. A segunda opção faz com que todas as variáveis internas ao subsistema assumam os valores da última vez que o sistema foi executado. O segundo campo (Show output port) é uma caixa de verificação. Quando selecionada, o bloco Enable terá uma porta de saída. Esta porta passa adiante o sinal recebido na entrada de Enable quando o bloco é habilitado.

É também de vital importância configurar os blocos de saída (Outport) de um subsistema . A caixa de diálogo do bloco tem três campos:



O primeiro campo (Port Number) determina a ordem que as portas irão aparecer no ícone do subsistema. O segundo campo (Output when disabled) contém duas opções: reset e held. A opção reset faz com que a saída seja resetada para o valor contido no terceiro campo (Initial output). A opção held faz com que a saída mantenha o seu último valor de saída antes do sistema ser desabilitado.

Um subsistema com enable é habilitado quando o sinal de entrada na porta de enable for positivo. Se o sinal for um vetor, o subsistema é habilitado se qualquer dos elementos do vetor for positivo.

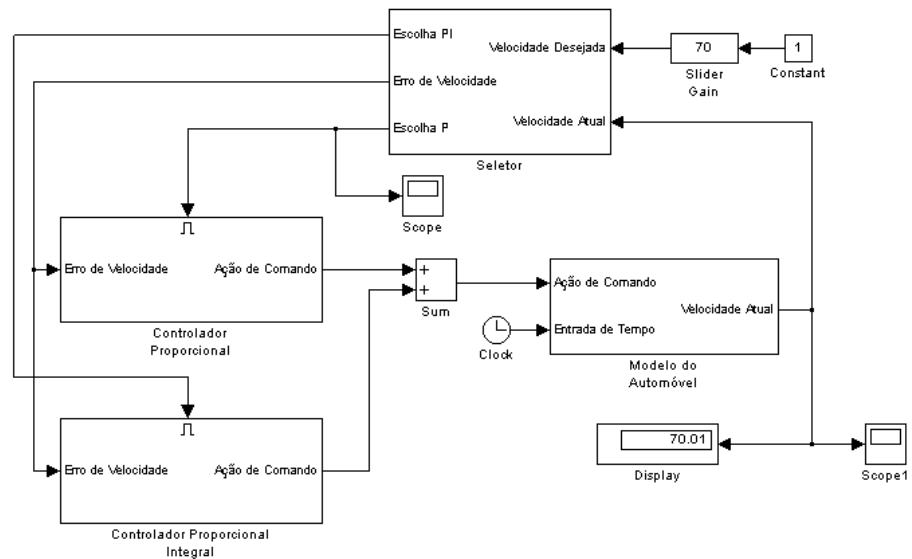
### Exemplo

Para ilustrar o uso de subsistemas com opção de habilitação, suponha que se deseja modificar o controlador de velocidade do automóvel do exemplo já citado para um controlador que possui dois modos de operação, dependendo do erro da velocidade. Se o valor absoluto do erro

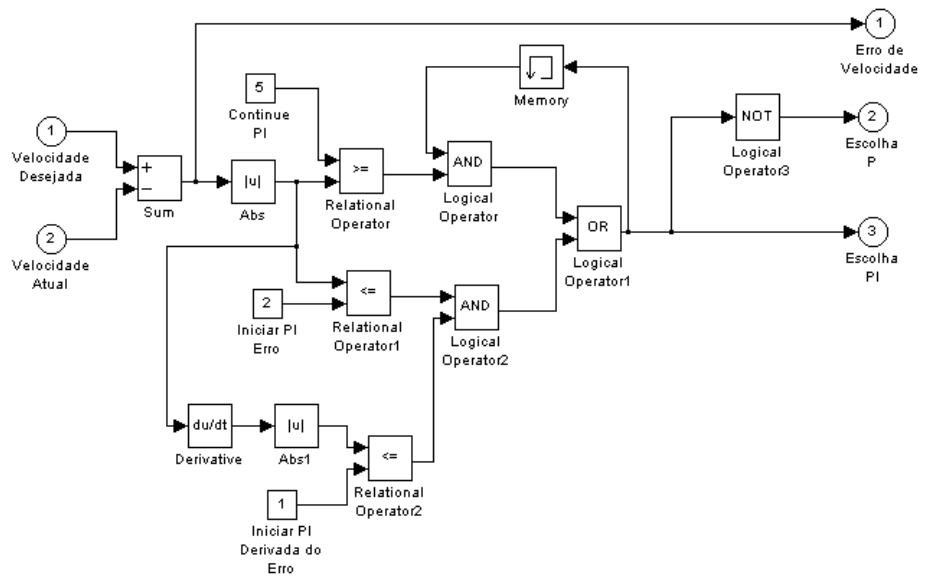
$$v_{err} = |\dot{x}_{desejado} - \dot{x}|$$

for menor do que o limiar de 2 m/s e o valor absoluto da taxa de variação do erro  $\dot{v}_{err}$ , for menor que um limiar de 1m/s<sup>2</sup>, deseja-se que o controlador seja um controlador Proporcional-Integral (PI). Uma vez que o controlador PI está habilitado, este deve permanecer até  $v_{err}$  ser menor que um valor de limiar de 5 m/s. De outra forma, o controlador proporcional (P) deve estar habilitado.

O modelo SIMULINK é mostrado na figura a seguir:

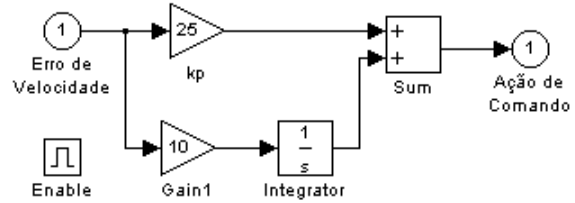


O subsistema seletor é mostrado na figura:



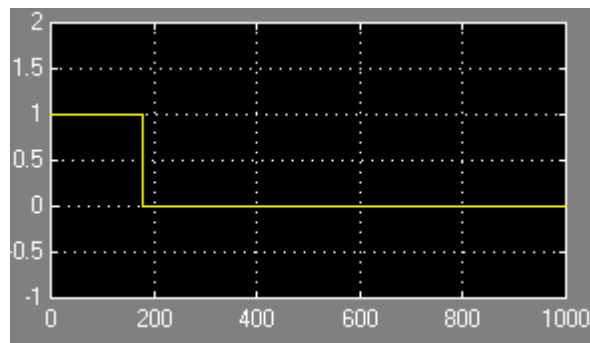
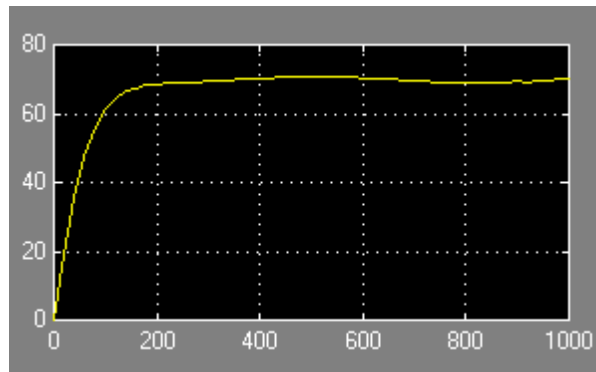
Este produz duas saídas. A saída PI é levada à 1.0 se as condições para o controlador PI forem satisfeitas, e 0.0 na situação contrária. A saída P é sempre o inverso lógico da saída PI.

O subsistema do controlador proporcional já foi mostrado anteriormente. O subsistema PI é mostrado na figura que segue;



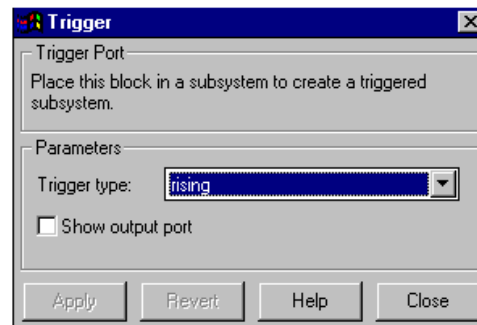
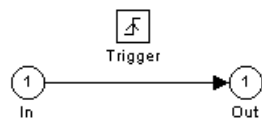
O bloco de Enable e o bloco Outport são configurados para reset.

Executando a simulação, a trajetória da velocidade e a saída P devem ser as seguintes:



### 6.3.2 - Subsistemas com Gatilho (Triggered)

Um subsistema com gatilho é executado cada vez que um sinal de gatilho é recebido. Este tipo de subsistema e a caixa de diálogo do bloco Trigger são mostrados a seguir:



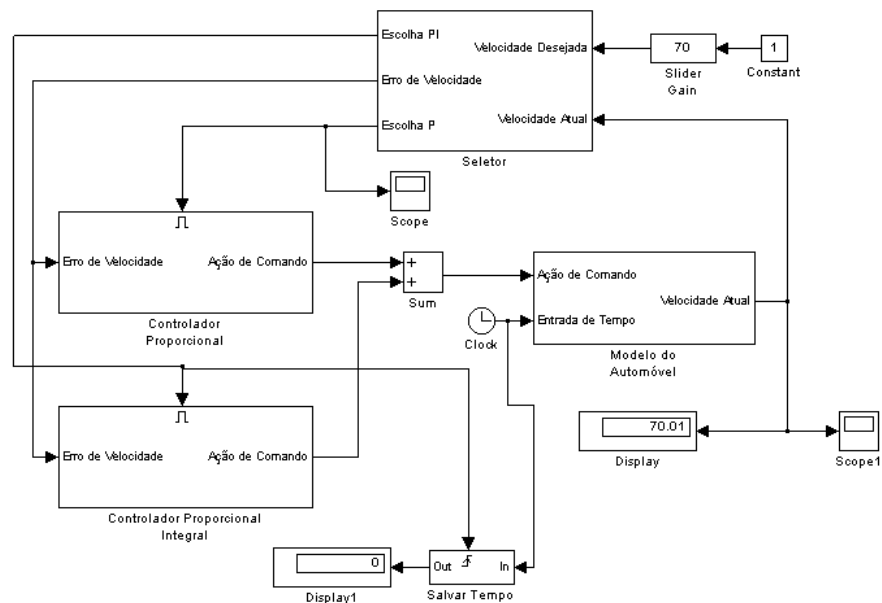
O primeiro campo da caixa (Trigger type) possui três opções: rising, falling e either. Se a opção escolhida for rising, um sinal de gatilho é definido quando este cruzar o zero enquanto estiver crescente. Se falling for escolhido, o sinal de gatilho passa a ser definido quando a entrada Trigger cruzar o zero no sentido decrescente. Either o sinal é definido quando cruzar o zero em ambos os sentidos. O segundo campo (Show output port) é uma caixa de verificação que permite escolher se uma saída de trigger irá aparecer no bloco. Esta saída, como no bloco de enable, passa adiante o sinal gatilho de entrada do subsistema.

Um subsistema com gatilho mantém seu valor de saída após o sinal de trigger ser recebido. O valor inicial de saída de um subsistema com gatilho é configurado nos blocos Outport do subsistema.

O sinal de gatilho pode ser escalar ou vetor. Se o sinal for um vetor, o subsistema é disparado quando qualquer elemento satisfizer as condições selecionadas em Trigger type.

#### Exemplo

O subsistema com gatilho ilustrado na figura anterior transmite a entrada para a saída quando um sinal de gatilho for recebido, e está configurado (bloco Outport) para manter sua saída até um outro sinal de trigger ser recebido. O bloco Outport é inicializado com 0. A figura a seguir ilustra o exemplo anterior com a adição deste subsistema com sua saída ligada a um bloco Display da biblioteca de dispositivos de saída (Sinks). O sinal de gatilho é conectado ao sinal de habilitação do controlador PI. O Display irá mostrar 0 até que o controlador PI seja ativado, e a seguir, mostrará o tempo da mais recente habilitação deste controlador.



### 6.3.3 - Subsistemas com Habilitação e Gatilhos

Adicionar os blocos Enable e Trigger simultaneamente a um subsistema, produz um subsistema com opções de habilitação e disparo. Este subsistema possuirá ambas as entradas Enable e Trigger. Este subsistema irá se comportar como um subsistema com opção de disparo, mas o sinal de trigger será ignorado a menos que o sinal de Enable seja positivo.

### 6.3.4 - Subsistemas Discretos com Execução Condicionada

Todos os tipos de execução condicionada devem servir para blocos contínuos ou discretos. Blocos discretos em um subsistema com habilitação serão executados com base no seu tempo de amostragem. Eles usam a mesma referência de tempo do resto do modelo SIMULINK; o tempo no subsistema é referenciado no início da simulação, e não na ativação do subsistema. Consequentemente, uma saída dependente de um bloco discreto não irá mudar necessariamente no instante da habilitação do subsistema.

Blocos discretos em subsistemas com gatilho devem ter seus tempos de amostragem configurados para  $-1$ , indicando que eles herdaram seus tempos de amostragem do sinal de controle. Nota-se que blocos discretos que incluem tempos de atraso ( $z^{-1}$ ) mudam seu estado cada vez que o subsistema é disparado com o sinal de Trigger.