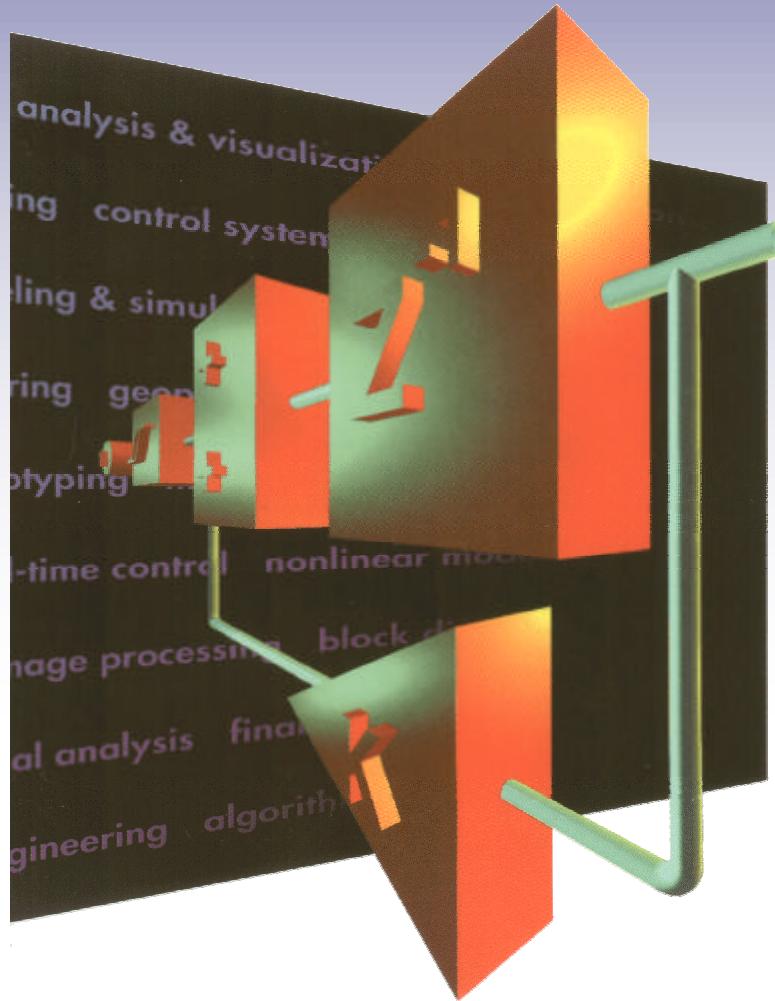


Curso de SIMULINK 2.0

Modelagem, Simulação e Análise de Sistemas Dinâmicos

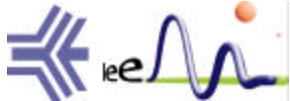


Faculdade de
Engenharia



Laboratório de
Engenharia Elétrica

Programa Prodenge / Sub-Programa Reenge
Universidade do Estado do Rio de Janeiro



AGRADECIMENTOS

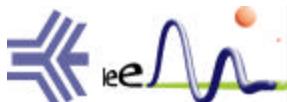
Estas breves notas sobre o SIMULINK versão 2.0 baseadas nas obras "The Student Edition of SIMULINK" e "Mastering SIMULINK" dos autores James B. Dabney e Thomas L. Harman resultam do trabalho dedicado de alunos da Faculdade de Engenharia da UERJ, tanto de forma direta como indireta. De forma direta envolveu-se no trabalho o aluno e bolsista de Iniciação Tecnológica do Projeto REENGE César Cunha de Souza. Um extenso grupo de pessoas se envolveu também ativamente dando suporte de hardware, software e ainda o valioso apoio pessoal nas tarefas diárias do laboratório. Neste grupo incluem-se não só alunos como os também bolsistas Hélio Justino Mattos Filho e Karla Karraz Valder, os estagiários Fábio da Silva Porto, Flávia Delduque Lima, Hellen Nathalia Trevisan, Marcos Paulo dos Santos, Valdeir Gomes de Araújo Filho, como também os funcionários do Laboratório de Engenharia Elétrica, cujos membros contribuíram valorosamente dando suporte e infra-estrutura para que este trabalho fosse bem sucedido. Um muito obrigado à equipe do LEE formada por Alberto Avelar Santiago, André Vallim Stachlewski, Antônio Marcos Medeiros Corrêa, José Emílio Gomes, Jair Medeiros Júnior, João Elias Souza da Costa, Luiz Roberto Franco Fagundes Filho, Marcos Augusto Mafra, Paulo Bulkool Batalheiro, Sueli Ferreira dos Santos e a Sra. Carla Aparecida Caldas de Almeida. Um reconhecimento especial deve ser feito ao diretor da Faculdade de Engenharia Dr. Nival Nunes de Almeida, coordenador geral do REENGE por ter possibilitado inúmeras atividades não só no LEE em particular mas em toda a Faculdade de Engenharia. À Prof.^a Maria Eugênia Mosconi de Gouveia, vice-diretora da Faculdade de Engenharia, que em trabalho conjunto com o diretor vem se empenhando em viabilizar as solicitações de estágio interno no LEE. Um muito obrigado também à aqueles colaboradores silenciosos que de forma direta ou indireta contribuíram para o êxito deste trabalho. O nosso agradecimento ao CNPq que mediante os recursos alocados pela FINEP, patrocinou as bolsas que permitiram este trabalho.

Bernardo Severo da Silva Filho
Orientador e Chefe do Lab. de Engenharia Elétrica



ÍNDICE

Apresentação.....	1
Capítulo 1 – Introdução Teórica.....	2
1.1 – Diagrama em Blocos.....	2
1.1.1 – Símbolos.....	3
1.2 – Transformada de Laplace.....	3
1.2.1 – Definição da Transformada de Laplace.....	4
1.2.2 – Transformação Inversa.....	4
1.2.3 – Propriedades da Transformada de Laplace.....	5
1.3 – Transformada Z.....	6
1.3.1 – Definição da Transformada Z.....	7
1.3.2 – Transformada de Funções Comuns.....	8
1.3.3 – Inversão da Transformada Z.....	8
1.3.4 – Propriedades da Transformada Z.....	9
1.3.5 – A Função de Transformada Discreta no Tempo.....	10
Capítulo 2 – Conhecendo o SIMULINK.....	11
2.1 – Acessando o SIMULINK.....	11
2.2 – Construindo um Modelo Simples.....	11
2.3 – Outro Modelo.....	14
2.4 – Usando o HELP do SIMULINK.....	21
Capítulo 3 – Construindo Modelos SIMULINK.....	23
3.1 – Elementos de Modelos.....	23
3.2 – Manipulando Blocos.....	24
3.3 – Fontes.....	26
3.3.1 – Fontes Comuns.....	26
3.3.2 – Importando do MATLAB.....	28
3.3.3 – Importando Arquivos Gerados no MATLAB.....	29
3.4 – Dispositivos de Saída.....	29
3.4.1 – Osciloscópio.....	29
3.4.1.1 – Dando ZOOM na Tela do Osciloscópio.....	30
3.4.1.2 – Propriedades do Osciloscópio.....	31
3.4.2 – Gráfico XY.....	33
3.5 – Configurando a Simulação.....	33
3.5.1 – Solver Page.....	34
3.5.1.1 – Solver Type.....	35
3.5.1.2 – Opções de Saída.....	37
3.5.2 – Página Workspace I/O.....	38
3.5.2.1 – Vetores de Estado Internos do SIMULINK.....	38
3.5.2.2 – Salvar para a Área de Trabalho.....	39
3.5.2.3 – Estados.....	39
3.5.2.4 – Save Options.....	39



3.5.3 – Página de Diagnósticos.....	40
3.6 – Executando uma Simulação.....	41
3.7 – Imprimindo um Modelo.....	42
3.7.1 – Imprimindo um Modelo Utilizando os Menus.....	42
3.7.2 – Enviando o Modelo para um Documento.....	42
3.7.3 – Utilizando o Comando <i>Print</i> do MATLAB.....	42
Capítulo 4 – Sistemas Contínuos no Tempo.....	45
4.1 – Sistemas Escalares Lineares.....	45
4.1.1 – Bloco Integrador.....	45
4.1.2 – Bloco Função de Transferência.....	50
4.2 – Vetores em Sistemas Lineares.....	52
4.2.1 – Linhas de Sinais Vetoriais.....	52
4.2.2 – Espaço de Estados.....	54
4.2.3 – Bloco de Espaço de Estados.....	56
4.3 – Modelando Sistemas Não Lineares.....	58
4.3.1 – Blocos de Função.....	61
4.3.1.1 – Bloco Fcn.....	62
4.3.1.2 – Bloco MATLAB Fcn.....	62
Capítulo 5 – Sistemas Discretos no Tempo.....	67
5.1 – Visão Geral.....	67
5.2 – Sistemas Discretos no Tempo Lineares Escalares.....	68
5.2.1 – Atraso Unitário.....	68
5.2.2 – Integrador Discreto no Tempo.....	69
5.2.2.1 – Integração Trapezoidal.....	70
5.2.3 – Bloco de Função de Transferência Discreta.....	71
5.3 – Blocos Lógicos.....	73
5.4 – Sistemas Discretos no Tempo Vetoriais.....	75
5.5 – Sistemas Discretos com Diferentes Taxas de Amostragem Simultâneas.....	77
5.6 – Sistemas Híbridos.....	79
Capítulo 6 – Subsistemas e Máscaras.....	82
6.1 – Subsistemas SIMULINK.....	82
6.1.1 – Encapsulando um Subsistema.....	84
6.1.2 – Bloco de Subsistema.....	85
6.2 – Blocos de Máscaras.....	88
6.2.1 – Convertendo um Subsistema em um Sistema com Máscara.....	89
6.2.2 – Página de Documentação do Editor de Máscaras.....	91
6.2.2.1 – Campo Tipo de Máscara.....	91
6.2.2.2 – Campo Descrição do Bloco.....	91
6.2.2.3 – Bloco de Auxílio.....	91
6.2.3 – Página de Inicialização do Editor de Máscara.....	92
6.2.3.1 – Campo Tipo de Máscara.....	92
6.2.3.2 – Seção de Prompt da Caixa de Diálogo do Bloco.....	92
6.2.3.3 – Comandos de Inicialização.....	97



Curso de Introdução ao SIMULINK

6.2.3.4 – Configurando os Blocos do Subsistema.....	97
6.2.3.5 – Variáveis Locais.....	98
6.2.4 – Página de Ícone do Editor de Máscara.....	100
6.2.4.1 – Campo Moldura do Ícone.....	101
6.2.4.2 – Campo Transparência do Ícone.....	101
6.2.4.3 – Campo Rotação do Ícone.....	102
6.2.4.4 – Campo Coordenadas de Desenho.....	102
6.2.4.5 – Comandos de Desenho.....	103
6.2.5 – Olhando sob a Máscara e Removendo Máscaras.....	106
6.2.6 – Criando uma Biblioteca de Blocos.....	107
6.3 – Subsistemas com Execução Condicionada.....	107
6.3.1 – Subsistemas com Habilitação.....	107
6.3.2 – Subsistemas com Gatilho.....	112
6.3.3 – Subsistemas com Habilitação e Gatilho.....	113
6.3.4 – Subsistemas Discretos com Execução Condicionada.....	113
Capítulo 7 – Animação no SIMULINK.....	114
7.1 – Toolbox de Animação.....	114
7.2 – Usando a Toolbox de Animação.....	114
7.2.1 – Propriedades dos Objetos de Animação.....	116
7.2.2 – Configurando uma Animação.....	117
7.2.3 – Propriedades da Figura.....	117
7.2.3.1 – Escala da Figura.....	117
7.2.4 – Modificando uma Animação.....	117
7.2.5 – Configurando Entradas Iniciais.....	117
7.3 – Salvando e Carregando Arquivos de Animação.....	118
Bibliografia.....	119



Apresentação

SIMULINK é um programa utilizado para modelagem, simulação e análise de sistemas dinâmicos. O programa se aplica a sistemas lineares e não lineares, contínuos e/ou discretos no tempo.

Utiliza uma interface gráfica com o usuário para construção dos modelos a partir de diagramas em blocos, através de operações de clique-e-arraste do mouse. Com esta interface podem-se criar modelos da mesma forma que se faz com papel e caneta. SIMULINK é o resultado de uma longa evolução de pacotes de simulação anteriores que necessitavam a formulação de equações diferenciais ou de equações de diferenças em linguagens de programação. Inclui bibliotecas de blocos contendo fontes, visualizadores, componentes lineares, não lineares e conectores, com a opção de criação ou personalização de blocos.

Após a definição do modelo, a simulação pode ser feita com diferentes algoritmos de resolução, escolhidos a partir dos menus do SIMULINK ou da linha de comando do MATLAB. Os menus são particularmente convenientes para o trabalho interativo, enquanto a linha de comando tem sua utilidade na simulação repetitiva a qual se deseja somente mudar parâmetros. Usando osciloscópios (Scopes) ou outros visualizadores, têm-se o resultado gráfico da simulação enquanto esta está sendo executada. Os resultados da simulação podem ser exportados para o MATLAB para futuro processamento ou visualização.

As ferramentas de análise de modelos incluem ferramentas de linearização e ajuste (Trimming) que podem ser acessadas a partir da linha de comando do MATLAB, assim como várias ferramentas do MATLAB e suas TOOLBOXES específicas. Sendo o MATLAB e o SIMULINK integrados, pode-se simular, analisar e revisar os modelos em qualquer dos dois ambientes.



Capítulo 1 – Introdução Teórica

1.1 - Diagrama de Blocos

A representação dos sistemas físicos por meio de equações nem sempre deixa clara a relação entre as funções de entrada e de saída desses sistemas. É portanto conveniente e desejável sistematizar a descrição matemática de um sistema, de tal forma que aquela relação seja expressa claramente.

Uma forma de apresentação das equações diferenciais de um sistema consiste no emprego de **Diagramas de Bloco**, em que cada bloco representa uma operação matemática, associando pares entrada-saída.

Quando o sistema é linear, ou puder ser linearizado, é possível tomar as transformadas de Laplace das equações do sistema, considerando condições iniciais nulas.

A relação entre cada grandeza de saída e a correspondente grandeza de entrada se chama função de transferência.

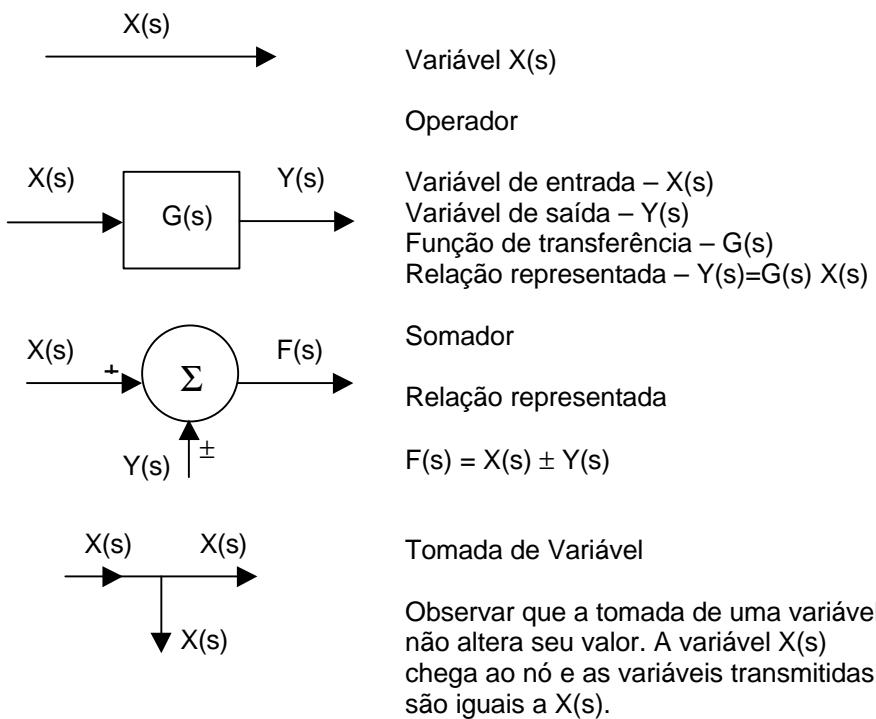
Usando as transformadas de Laplace, essas funções são em geral, funções de s . Quando essas funções são colocadas em vários blocos, o diagrama é chamado *Diagrama de Bloco*.

Em geral, os diagramas de bloco são úteis na visualização das funções dos diversos componentes do sistema, bem como permitem estudos de “*signal-flow*”.

Os diagramas de bloco são mais fáceis de desenhar do que os circuitos que eles representam. Partindo-se de um diagrama de bloco, é possível, mediante a utilização de regras especiais, denominadas “Álgebra dos diagramas de Bloco” reduzir o diagrama a um único bloco e, assim, achar a função global de transferência do problema, sem necessidade de resolver o sistema inicial de equações diferenciais que, algumas vezes, exige muito tempo devido ao elevado número de equações envolvidas.

1.1.1 - Símbolos

Os símbolos utilizados na técnica de diagramas de bloco são muito simples, e se encontram representados a seguir:



1.2 - Transformada de Laplace

A solução de equações diferenciais com excitações descontínuas ou de ordem superior a dois é muito trabalhosa através do método clássico. Além disso, a introdução de condições para a determinação das constantes de integração requer a solução de um sistema de equações algébricas em número igual à ordem da equação diferencial. Com o objetivo de facilitar e sistematizar a solução de equações diferenciais ordinárias lineares, a coeficientes constantes, utiliza-se exaustivamente o método da transformada de Laplace. Podem ser enumeradas as seguintes vantagens deste método moderno, que utiliza as transformadas:

1. Ele inclui as condições iniciais ou de contorno.
2. O trabalho envolvido na solução é simplesmente algébrico.
3. O trabalho é sistematizado.
4. A utilização de tabelas de transformadas reduz o volume de trabalho requerido.
5. Pode-se tratar excitações apresentando descontinuidades.
6. Os componentes transitório e de regime permanente da solução são obtidos simultaneamente.



A desvantagem dos métodos com transformadas é que se forem utilizados mecanicamente, sem conhecimento da teoria realmente envolvida, conduzem algumas vezes a resultados errôneos. Além disso, algumas equações particulares podem ser resolvidas mais simplesmente e com menos trabalho através do método clássico.

1.2.1 - Definição da Transformada de Laplace

A transformada direta de Laplace de uma função $f(t)$, seccionalmente contínua no intervalo $(0, \infty)$, com valor $f(t)$ é dada por:

$$L[f(t)] = \int_0^{\infty} f(t)e^{-st} dt = F(s)$$

O valor da integral resulta numa função $F(s)$, tendo s como variável. Este parâmetro s é uma grandeza complexa da forma $s + jw$. Deve-se ressaltar que os limites de integração são zero e infinito e que, portanto, não interessam os valores de $f(t)$ para instantes negativos ou nulos.

1.2.2 - Transformação Inversa

A aplicação da transformada de Laplace transforma uma equação diferencial em equação algébrica. A partir da equação algébrica, obtém-se prontamente a transformada da função resposta. Para completar a solução deve-se encontrar a transformada inversa. Em alguns casos, a operação de transformação inversa

$$f(t) = L^{-1}[F(s)]$$

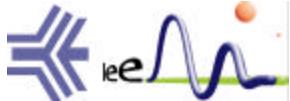
pode ser efetuada por referência direta a tabelas de transformadas ou por meio de programas de computador. Quando não é possível encontrar em tabelas a transformada da resposta, a técnica geral consiste em exprimir $F(s)$ sob a forma de uma soma de frações parciais com coeficientes constantes. As frações parciais apresentam no denominador monômios ou binômios e suas transformadas são encontradas diretamente em tabelas. A transformada inversa completa é a soma das transformadas inversas das frações.

A transformada da resposta $F(s)$ pode ser expressa em geral através da relação entre dois polinômios $P(s)$ e $Q(s)$. Considere-se que estes polinômios sejam de graus w e n , respectivamente, e que são ordenados segundo as potências decrescentes da variável s ; assim,

$$F(s) = \frac{P(s)}{Q(s)} = \frac{a_w s^w + a_{w-1} s^{w-1} + \dots + a_1 s + a_0}{s^n + b_{n-1} s^{n-1} + \dots + b_1 s + b_0}$$

Os elementos a e b são constantes reais e o coeficiente da maior potência em s no denominador é reduzido à unidade. Serão consideradas somente as funções $F(s)$ que são frações próprias para as quais n é maior que w . O primeiro passo é fatorar $Q(s)$ em fatores do primeiro grau e em binômios com coeficientes reais:

$$F(s) = \frac{P(s)}{Q(s)} = \frac{P(s)}{(s - s_1)(s - s_2) \dots (s - s_k) \dots (s - s_n)}$$



Curso de Introdução ao SIMULINK

Os valores $s_1, s_2 \dots s_n$, finitos que anulam o denominador, são chamados zeros do denominador. Estes valores de s , que podem ser reais ou complexos, também tornam $|F(s)|$ infinito e, e decorrência são chamados pólos de $F(s)$. Em consequência, os valores $s_1, s_2 \dots s_n$ são referidos como zeros do denominador ou pólos finitos da função completa, isto é, há n pólos de $F(s)$.

A transformada $F(s)$ pode ser expressa como uma soma de frações. Se os pólos são simples (não repetidos), o número de frações é igual a n (número de pólos de $F(s)$). Em tais casos a função $F(s)$ pode ser expressa sob a forma

$$F(s) = \frac{P(s)}{Q(s)} = \frac{A_1}{s - s_1} + \frac{A_2}{s - s_2} + \dots + \frac{A_k}{s - s_k} + \dots + \frac{A_n}{s - s_n}$$

O problema agora é a determinação das constantes A_1, A_2, \dots, A_n correspondentes aos pólos $s_1, s_2 \dots s_n$. Os coeficientes A_1, A_2, \dots, A_n são chamados resíduos de $F(s)$ nos pólos correspondentes.

1.2.3 - Propriedades das Transformadas de Laplace

Função	Transformada
$Af(t)$	$AF(s)$
$f_1(t)+f_2(t)$	$F_1(s)+F_2(s)$
$\frac{d}{dt} f(t)$	$sF(s)-f(0)$
$\frac{d^2}{dt^2} f(t)$	$s^2F(s)-sf(0)-f'(0)$
$\frac{d^n}{dt^n} f(t)$	$s^nF(s)-\sum_{k=1}^n s^{n-k} f^{(k-1)}(0)$
$\int f(t)dt$	$\frac{F(s)}{s} + \frac{1}{s} \left[\int f(t)dt \right]_{t=0}$
$\int_0^t f(t)dt$	$\frac{F(s)}{s}$
$e^{-at} f(t)$	$F(s+a)$
$f(t-a)1(t-a),$ $1(t-a)=\begin{cases} 1, & t > a \\ 0, & t \leq a \end{cases}$	$e^{-as} F(s)$



$$tf(t) \quad - \frac{dF(s)}{ds}$$

$$t^2f(t) \quad \frac{d^2}{ds^2} F(s)$$

$$\frac{1}{t} f(t) \quad \int_s^{\infty} F(s) ds$$

$$f\left(\frac{t}{a}\right) \quad aF(as)$$

$$f_1(t)*f_2(t) = F_1(s).F_2(s)$$
$$\int_0^t f_1(t-l)f_2(l)dl =$$
$$= \int_0^t f_1(t)f_2(t-l)dl$$

$$f(t).g(t) \quad \frac{1}{2\pi j} F(s) * G(s)$$

* indica o produto de convolução.

1.3 - Transformada Z

Os problemas de engenharia que utilizam a transformada Z surgiram da análise de sistemas de controle a dados amostrados, durante a Segunda Guerra Mundial, quando tais sistemas adquiriram proeminência. Sistemas a dados amostrados operam com funções discretas no tempo (ou amostrados), pou uma ou duas razões. Pode ser que os dados sejam disponíveis somente em instantes discretos (como num radar de exploração). É possível, por outro lado, que o uso dos dados amostrados permita projetar um sistema desempenho dinâmico melhor que o possível com dados contínuos no tempo. Como a transformada de Laplace era a principal ferramenta para estudar sistemas de controle contínuos no tempo, sua extensão a sistemas de dados amostrados era inteiramente natural.

A introdução da transformada Z é motivada pelas mesmas considerações que fazem a transformada de Laplace útil: as equações de diferenças que governam o comportamento do sistema discreto são transformadas em equações algébricas, freqüentemente mais simples de resolver que as equações originais, e capazes de dar melhor visão daquele comportamento.

Em análise de sistemas discretos no tempo por meio da transformada Z, os mesmos passos são empregados: os sinais discretos no tempo são substituídos por suas respectivas transformadas; o sistema é representado por uma “função de transferência discreta no tempo” que, como no caso contínuo, pode ser obtida pela combinação algébrica das funções de transferência discretas de seus componentes. A transformada Z da resposta discreta no tempo será igual ao produto da função de transferência do sistema pela transformada Z do sinal de entrada. Como último passo, a resposta real no domínio do tempo (discreta) será obtida por inversão da transformada Z da saída.

As principais diferenças entre a transformada de Laplace e a transformada Z são: a função elementar na qual os sinais são decompostos para obter a transformada Z é z^n (z uma variável complexa), em vez de e^{st} ; a transformada Z é definida como uma soma em vez de integral.

1.3.1 - Definição da transformada Z

A transformada Z de um sinal discreto no tempo, $f(n)$, é definida como uma série de potências em z^{-1} cujos coeficientes são amplitudes do sinal. Como no caso da transformada de Laplace, podemos definir uma transformada Z unilateral e uma bilateral. Estas definições são, respectivamente,

$$L_U[f(n)] = F_U(z) = \sum_{n=0}^{\infty} f(n) z^{-n}$$

$$L_B[f(n)] = F_B(z) = \sum_{n=-\infty}^{\infty} f(n) z^{-n}$$

Para ser útil em análise de sistemas, é desejável que estas séries sejam exprimíveis em forma fechada. Como o coeficiente de z^{-n} no desenvolvimento em série é $f(n)$, segue-se que o desenvolvimento da forma fechada em uma série de potências em z^{-n} “gerará” o sinal. Portanto, a transformada Z pode ser descrita como a “função geratriz” do sinal discreto no tempo ao qual ela corresponde. As potências negativa z^{-n} são mais usadas que as positivas z^n porque isso está de acordo com o uso mais freqüente em engenharia. Na literatura matemática sobre funções geratrizes e em alguma literatura sobre sistemas de controle a dados amostrados, os coeficientes das potências positivas de z, geralmente correspondem aos valores de $f(n)$ para $n > 0$.

Se $f(n) = 0$ para $n < 0$, as transformadas unilateral e bilateral são idênticas; porém, se $f(n) \neq 0$ para algum $n < 0$ as duas definições produzirão expressões diferentes. A transformada Z bilateral obviamente incorpora informações sobre os valores $f(n)$ em todos os instantes discretos em que é definida. A unilateral, só nos instantes não negativos.

1.3.2 - Transformadas de Funções Comuns

Função $f(n)$	Transformada Z $F(z)$	Raio de Convergência r_a
$d(n - k)$	z^{-1}	0
$m_1(n)$	$\frac{1}{1 - z^{-1}}$	1
a^n	$\frac{1}{1 - az^{-1}}$	a
n	$\frac{z^{-1}}{(1 - z^{-1})^2}$	1

1.3.3 - Inversão da Transformada Z

Para determinar o sinal $f(n)$ correspondente a uma dada $F(z)$, isto é, para gerar $f(n)$, é necessário desenvolver $F(z)$ em série de potências de z^{-1} . Como, em qualquer região de convergência, o desenvolvimento em série de potências é única. O método a empregar é uma questão de conveniência.

Obviamente, o método mais simples é procurar numa tabela adequada a função discreta no tempo no tempo correspondente à transformada que se quer inverter. É possível inverter a maioria das transformadas Z que aparecem em problemas de engenharia, por meio de uma pequena tabela e a técnica de decomposição em frações parciais.

Uma técnica mais elegante e geral é de usar a fórmula de inversão:

$$f(n) = \frac{1}{2\pi j} \oint_C F(z) z^{n-1} dz$$

onde C é um círculo de raio $r > r_a$, isto é, um círculo envolvendo todas as singularidades de $F(z)z^{n-1}$.

O cálculo real da equação anterior é muito eficientemente executado por meio do cálculo de resíduos. Especificamente, a equação pode ser escrita como:

$$f(n) = \text{soma de resíduos do produto } F(z) z^{n-1},$$

em seus pólos internos ao círculo de convergência de $F(z)$.

Observe que, quando $n = 0$, $F(z) z^{-1}$ pode ter um pôlo na origem mesmo quando $F(z)$ não tem o resíduo de $F(z) z^{-1}$ deve ser aí calculado.



Um terceiro método, que pode ser aplicado quando $F(z)$ é racional, é exprimir numerador e denominador de $F(z)$ em polinômios em z^{-1} ; então, usando a “divisão contínua” da álgebra, dividir o numerador pelo denominador, e assim obter uma série em potências z^{-1} . Este método que não tem similar na transformada de Laplace é muito eficiente quando falta conhecimento dos pólos de $F(z)$ e as tabelas de transformada Z ou a fórmula de inversão não podem ser usadas. Somente os valores de $f(n)$, e não sua expressão geral, podem ser assim obtidos.

1.3.4 - Propriedades da transformada Z

Propriedade	Função do tempo	Transformada Z
Linearidade	$af(n) + bg(n)$	$aF(z) + bF(z)$
Diferenças a. Avançadas	$\Delta^k f(n)$	$(z-1)^k F(z) -$ $- z \sum_{j=0}^{k-1} (z-1)^{k-j-1} \Delta^j f(0)$
b. atrasadas	$\nabla^k f(n)$	$(1-z^{-1})^k F(z) -$ $- \sum_{j=0}^{k-1} (1-z^{-1})^{k-j-1} \nabla^j f(-1)$
Somas	$\sum_{k=-\infty}^n f(k)$	$\frac{1}{1-z^{-1}} F(z) +$ $+ \frac{1}{1-z^{-1}} \sum_{k=-\infty}^{-1} f(k)$
Translação	$F(n+k) \quad k > 0$	$z^k F(z) - z^k \sum_{j=0}^{k-1} f(j) z^{-j}$
Multiplicação por a^n	$a^n f(n)$	$f(a^{-1}z)$
Multiplicação por n^k	$n^k f(n)$	$\left(z^{-1} \frac{d}{dz^{-1}} \right) F(z)$
Convolução	$y(n) = \sum_{k=0}^n h(n-k)x(k)$	$Y(z) = H(z)X(z)$
Produto de duas funções	$f(n)g(n)$	$\frac{1}{2\pi j} \oint_C \frac{F(w)G(zw^{-1})}{w} dw$
Mudança de escala	$f(an) \quad a = \text{inteiros}$	$F(z^{-a})$
Valor inicial	$f(0) = \lim_{z \rightarrow \infty} F(z)$	
Valor final	$f(\infty) = \lim_{z \rightarrow 1} (1-z^{-1}) F(z)$ <i>se $(1-z^{-1})F(z)$ é analítico para $z \geq 1$</i>	



1.3.5 - A Função de Transferência Discreta no Tempo

Como a função de transferência de sistema contínuo, a função de transferência discreta no tempo é definida para um sistema fixo e discreto como a transformada Z da resposta unitária $h(n)$. Assim,

$$H(z) = Z[h(n)]$$

Para um sistema de entradas e saídas múltiplas, podemos definir a matriz de transferência discreta no tempo $H(z)$ cujo elemento $H_{ij}(z)$ é a transformada Z da correspondente resposta unitária $h_{ij}(n)$.

Como consequência da propriedade de convolução, a relação de entrada e saída no domínio do tempo:

$$y(n) = \sum_{k=0}^n h(n-k)x(k)$$

é convertida na relação algébrica no “domínio z”:

$$Y(z) = H(z)Z(z)$$

Esta propriedade simplifica consideravelmente a análise dos sistemas fixos discretos no tempo; ela permite que tais elementos sejam analisados calculando as funções de transferência de cada um dos componentes e os combinando de acordo com as regras algébricas conhecidas. A resposta do sistema a uma dada entrada é então obtida invertendo o produto da função de transferência pela transformada da entrada. A técnica acompanha a de sistemas contínuos.

Em muitos casos, não é necessário executar esta última etapa; muitas propriedades úteis da resposta podem ser obtidas diretamente da função de transferência discreta no tempo. Dentre tais propriedades está a estabilidade do sistema e os valores final e inicial da resposta.

A função de transferência de um sistema contínuo no tempo pode ser interpretada como o quociente da resposta à uma exponencial complexa e^{st} por esta entrada fictícia e^{st} . Uma interpretação similar é possível para a transformada Z. Considere um sistema H ao qual a seqüência de entrada $\{..., z^{-2}, z^{-1}, 1, z, z^2, ...\}$ é aplicada começando em $n = -\infty$. A resposta é:

$$y(n) = \sum_{k=-\infty}^n h(n-k)z^k.$$

Com a mudança de índice $m = n - k$, fica

$$y(n) = \sum_{m=0}^{\infty} h(m)z^{n-m} = z^n H(z).$$

Portanto segue-se que

$$H(z) = \frac{\text{resposta à } z^n}{z^n}.$$

Capítulo 2 – Conhecendo o SIMULINK

2.1 - Acessando o SIMULINK

Para acessar o SIMULINK deve-se primeiro abrir o MATLAB, pois apesar de ser uma aplicação específica, este não trabalha independente e utiliza suas ferramentas de cálculo.

A partir do Windows 95/98, deve-se clicar duas vezes no ícone do MATLAB. Aberto o programa deve-se então clicar no ícone “Start Simulink” na barra de ferramentas do MATLAB ou digitar “simulink” na linha de comando e pressionar enter logo em seguida, como mostrado a seguir:

```
>> simulink <enter>
```

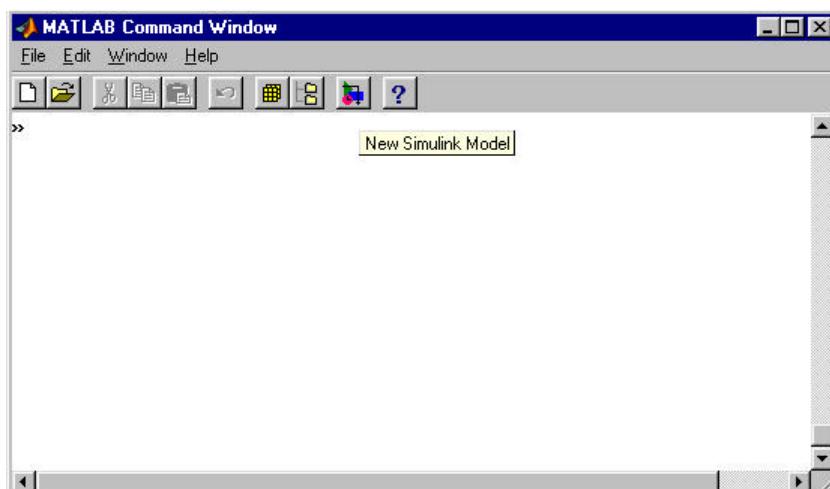
2.2 - Construindo um Modelo Simples

Exemplificando a utilização do SIMULINK, temos um modelo a criar. Este deve resolver a equação diferencial:

$$\dot{x} = \sin(t),$$

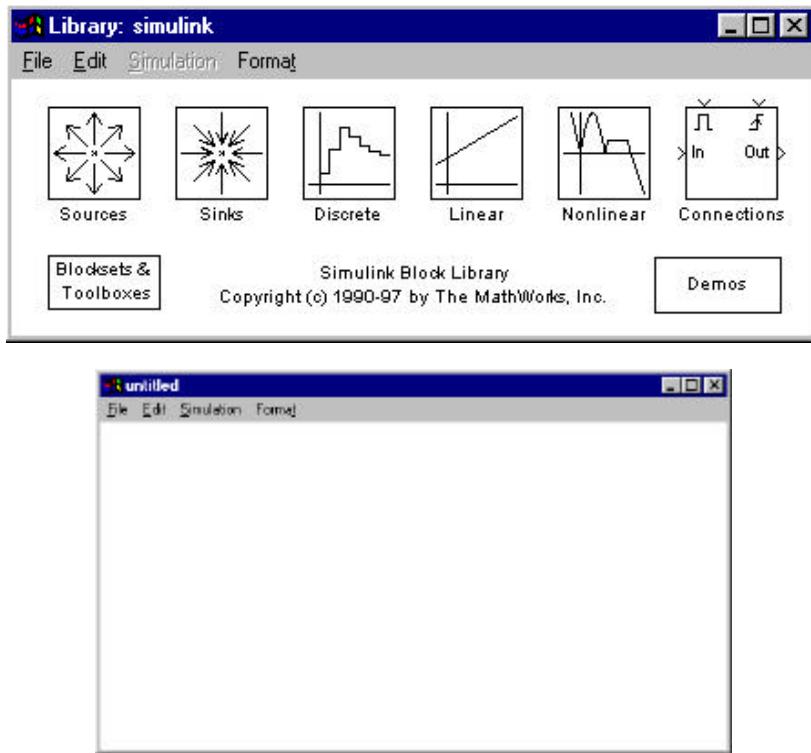
onde $x(0) = 0$.

Sendo o SIMULINK uma extensão do MATLAB, este deve então ser carregado a partir do MATLAB. Inicie o SIMULINK clicando no seu ícone na barra de ferramentas do MATLAB, como mostrado na figura :

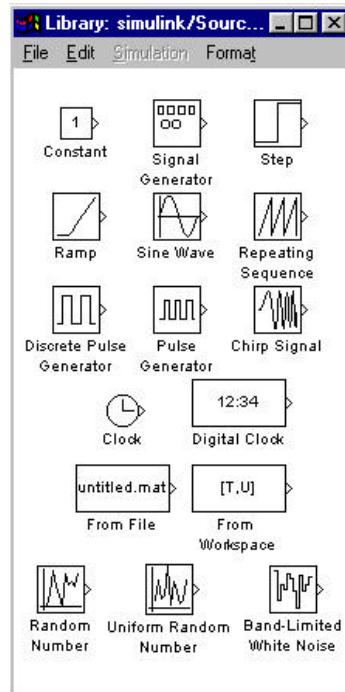


- (1) Todos os comandos do MATLAB são em letras minúsculas.
- (2) O MATLAB é sensível ao tipo de fonte (maiúsculo ou minúsculo). Por exemplo: a variável x é diferente à variável X.

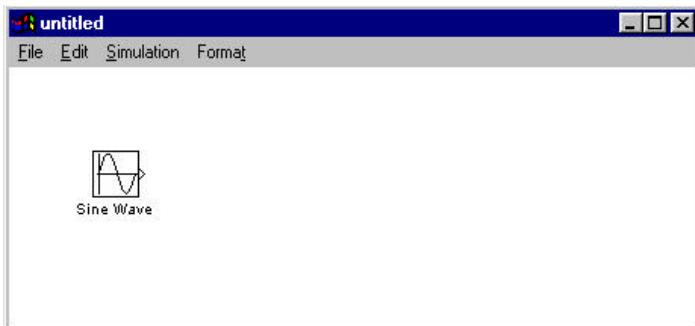
Duas janelas se abrirão na tela. A primeira janela é a biblioteca de blocos do SIMULINK mostrado na figura. A segunda é uma janela em branco para construção do modelo, nomeada untitled até que seja salvo com outro nome.



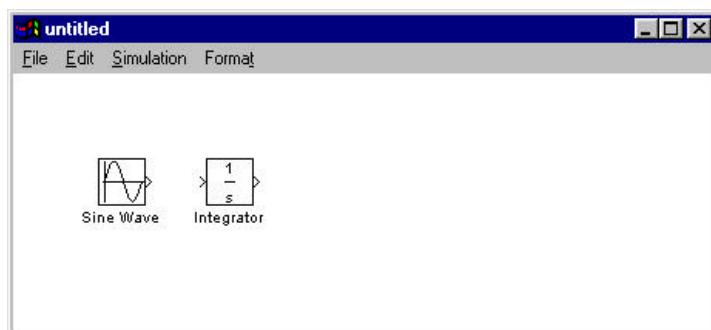
Dê um click duplo no ícone Sources na janela de bibliotecas do SIMULINK.



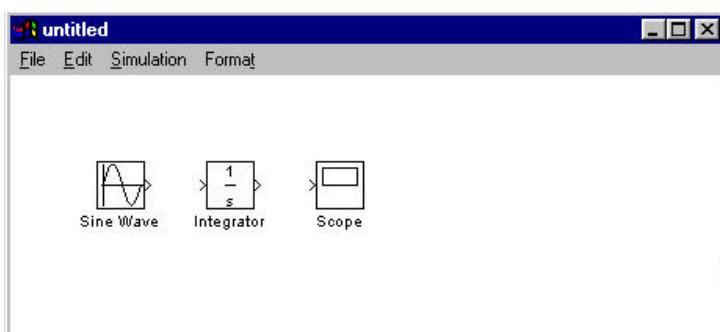
Arraste o bloco de onda senoidal (Sine Wave) para a janela do modelo. Uma cópia deste bloco deve ser criada nesta janela.



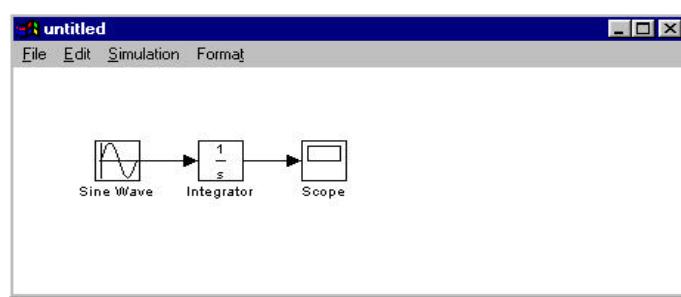
Abra a biblioteca de blocos lineares e arraste um bloco integrador (Integrator) para a janela do modelo.



Abra a biblioteca dispositivos de saída (Sinks) e arraste um SCOPE para a janela do modelo em construção.

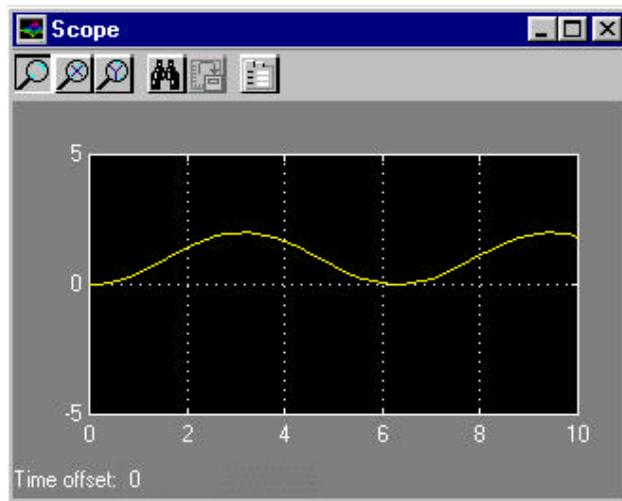


A seguir, conecte os blocos para completar o modelo como na figura a seguir:



Dê um duplo click no bloco SCOPE e na barra de menu do SIMULINK clique **SIMULATION:START**. A simulação será executada, resultando no gráfico gerado no bloco SCOPE, mostrado a seguir:

Obs.: A integral é definida entre t_0 e t_F . Para $t_0 = 0$, $\cos(t)=1$.



Para verificar se o gráfico gerado representa a solução da equação diferencial desejada, deve-se resolver a mesma analiticamente, cujo resultado é:

$$x(t) = 1 - \cos(t),$$

que corresponde ao gráfico apresentado.

2.3 - Outro Modelo

O modelo anterior serviu como exemplo de implementação no SIMULINK, mas está longe de representar um caso usual de utilização do software devido à pequena quantidade de blocos e ligações. Agora será usado um modelo de um processo biológico para ilustrar vários níveis adicionais de dificuldade na implementação.

Scheinerman descreveu um modelo simples do crescimento de bactérias isoladas do ambiente externo num pote. Admite-se que as bactérias nascem numa taxa proporcional ao número de bactérias presentes e que elas morrem a uma taxa proporcional ao quadrado do número de bactérias presentes. Se x representa o número de bactérias presentes, a taxa em que as bactérias nascem é definida por

$$\text{Taxa de Natalidade} = bx$$

E a taxa em que elas morrem

$$\text{Taxa de Mortalidade} = px^2$$

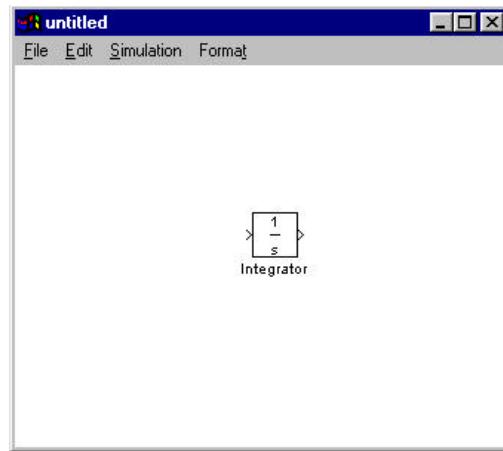
A taxa total de mudança na população de bactérias é a diferença entre a natalidade e a mortalidade de bactérias. O sistema pode ser então descrito pela equação diferencial a seguir:

$$\dot{x} = bx - px^2$$

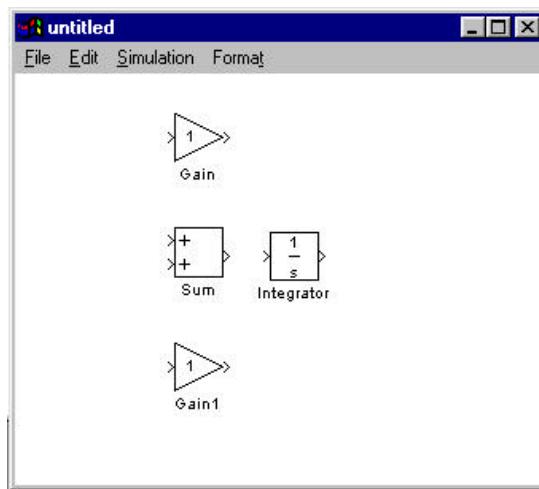
Partindo disto será então construído o modelo do sistema dinâmico supondo que $b=1$ bactéria/hora e $p=0,5$ bactéria/hora. Será determinado o números de bactérias contidas no pote após 1 hora, admitindo que inicialmente existiam 100 bactérias presentes.

Crie uma nova janela de modelo na barra de menu escolhendo **FILE:NEW**.

Este é um sistema de 1^a ordem, o que quer dizer que requer somente um integrador para resolver a equação diferencial. A entrada do integrador é \dot{x} e a saída é x . Abra a biblioteca linear e arraste o integrador para a janela do modelo, seguindo a posição mostrada na figura:

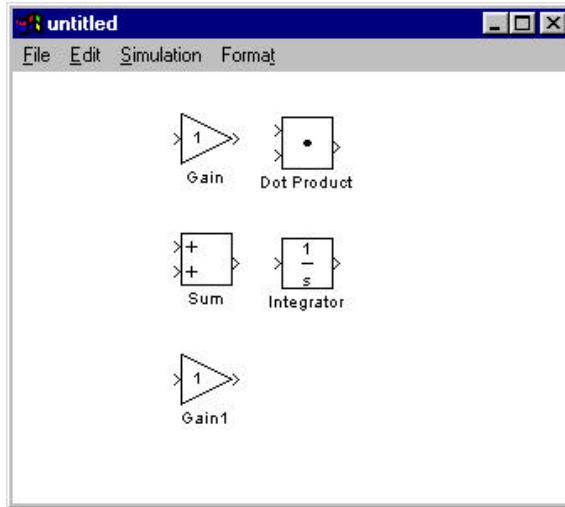


Ainda na biblioteca Linear arraste dois blocos de ganhos (Gain) para a janela do modelo e posicione-os como na figura. O SIMULINK exige que cada bloco tenha seu nome único. Devido a isto, o segundo bloco de ganho será nomeado GAIN1. Arraste ainda um bloco de soma (Sum) e a seguir feche a janela da biblioteca linear.

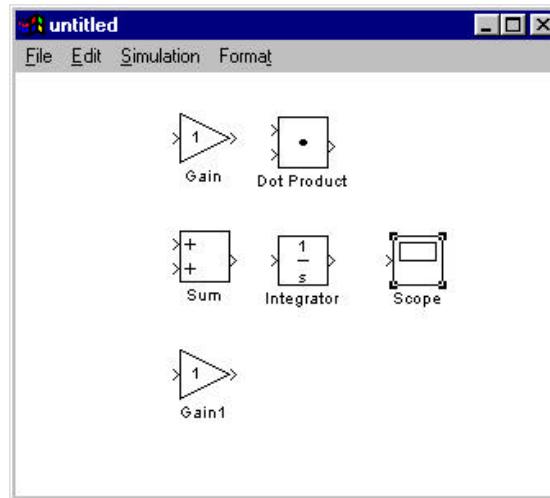


É boa técnica fechar todas as janelas que não estão sendo utilizadas. Isto favorece uma melhor utilização da memória disponível no microcomputador.

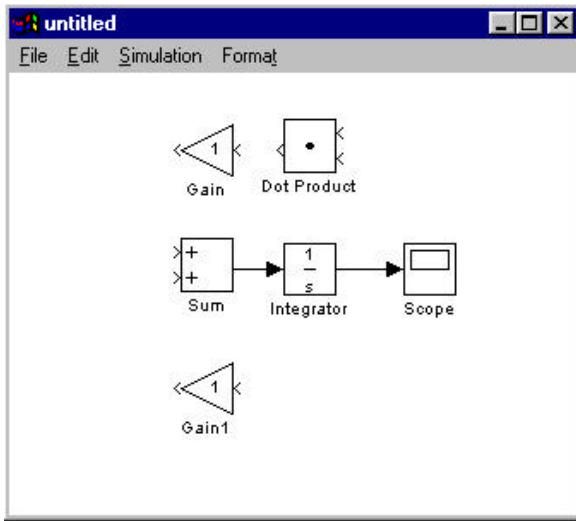
Abra agora a biblioteca de blocos não lineares (Nonlinear) e arraste um bloco de produto (product) para a posição mostrada. Este bloco será utilizado para calcular o valor de x^2 .



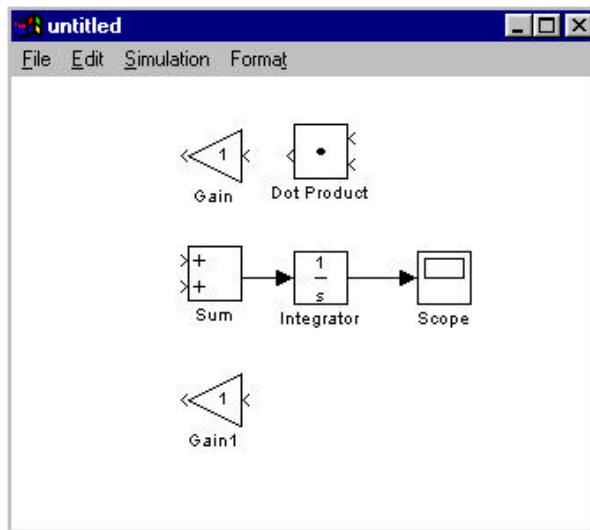
Abra a seguir a biblioteca dispositivos de saída (Sinks) e arraste um bloco SCOPE para a janela do modelo seguindo a posição mostrada.



A orientação padrão do SIMULINK de todos os blocos é posicionar entradas à esquerda e saídas à direita. Porém este modelo será muito mais legível se invertermos os blocos de ganho e produto. Iniciando com o Produto, deve-se primeiro clicar sobre ele de modo a selecioná-lo. Pequenos quadros pretos aparecerão nas quinas do bloco indicando seleção. No menu do SIMULINK, escolha **FORMAT:FLIP BLOCK**. Agora as entradas estão à direita e as saídas à esquerda. Repita a operação de inversão para cada bloco de Ganho. O modelo agora deve estar semelhante à figura:



Trace agora uma linha de sinal da saída do bloco de soma para a entrada do integrador e outra da saída do integrador para a entrada do SCOPE.

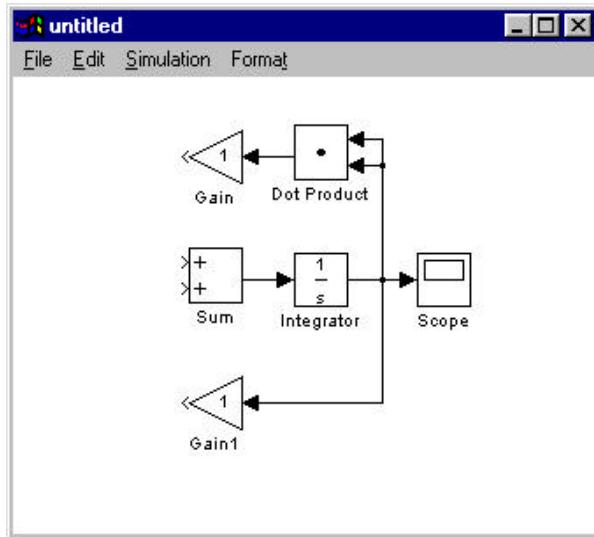


A seguir é necessário conectar a linha que liga o integrador ao SCOPE ao bloco de ganho situado na parte inferior da janela, pois esta linha contém o valor de x . Para fazê-lo, pressione a tecla CTRL do teclado e clique na linha de sinal. O cursor do mouse irá mudar para uma cruz. Conserve a tecla do mouse pressionada enquanto faz a ligação e solte agora a tecla CTRL. Leve a linha até a entrada do bloco de ganho. O SIMULINK automaticamente ajusta a linha com um ângulo de 90°.

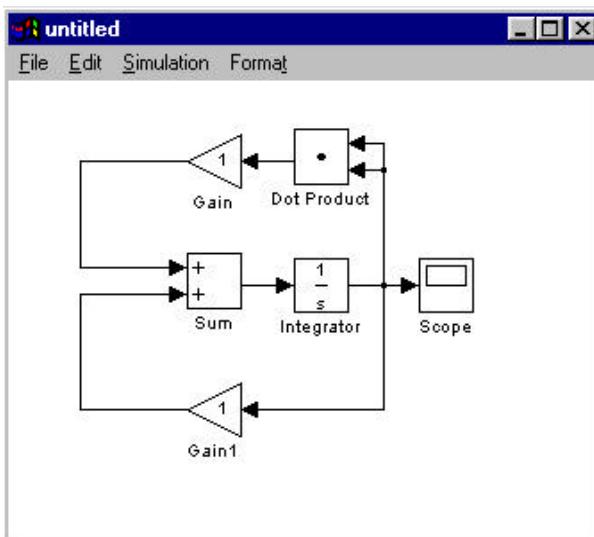
Se o mouse possuir três botões as operações de clicar e arrastar podem ser feitas utilizando o botão direito.

Repita a operação ligando a linha de sinal Integrador-SCOPE até a entrada superior do bloco de produto. Da linha de sinal que liga a entrada superior do bloco de produto repita a operação de ligação para a entrada inferior do mesmo bloco, de modo que o bloco execute a operação $x \cdot x = x^2$. Conecte agora a saída do bloco

de produto à entrada do ganho na parte superior da janela de modelo. Sua janela agora deve estar da seguinte forma:

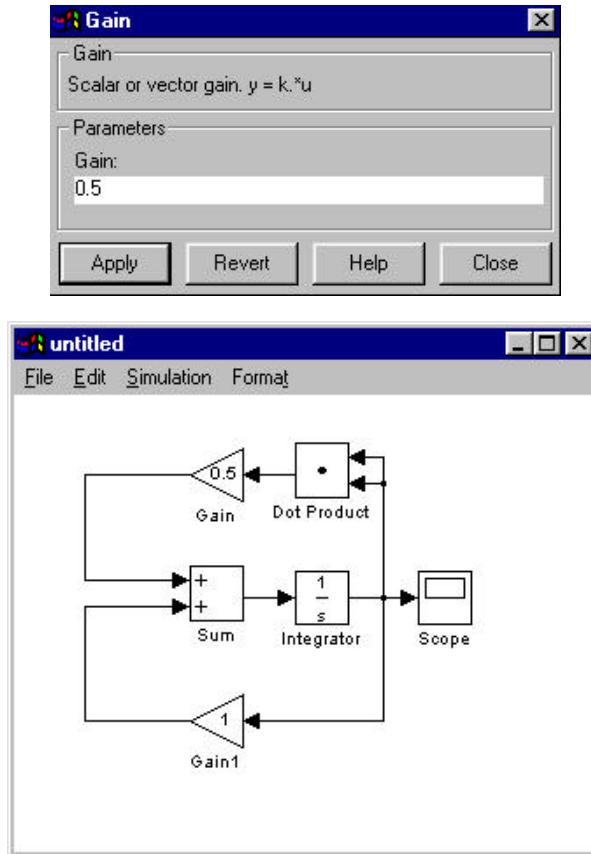


Conecte agora a saída do ganho superior à entrada superior do bloco de soma e a saída do ganho inferior à entrada inferior do mesmo bloco de soma.

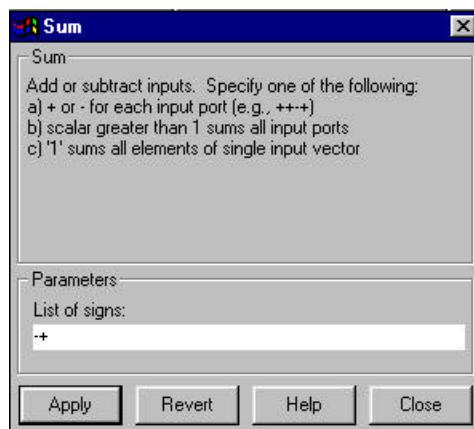


O modelo agora está completo, mas os blocos devem ser configurados (parametrizados) para que este represente o sistema desejado. O SIMULINK tem como *default* para os blocos de ganho o valor de 1.0, para o bloco de soma duas entradas positivas e para o integrador o valor inicial 0.0. O valor inicial do integrador representa o número inicial de bactérias presentes no pote.

Será iniciada agora a parametrização com os blocos de ganho. Dê um duplo clique no ganho da parte superior e mude o valor de 1.0 para 0.5 na caixa de diálogo que irá aparecer, a seguir clique em **Close**. Note que o valor do ganho do bloco muda para 0.5 no diagrama em blocos.

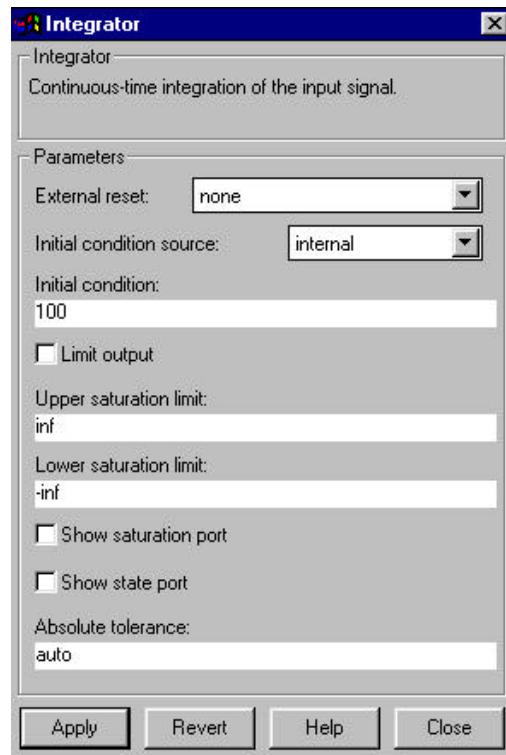


Agora dê um duplo clique no bloco de soma e no campo **List of signs** mude de **++** para **-+** na caixa de diálogo que abrirá. Os sinais representam os próprios sinais de entrada no bloco. A seguir clique em **Close**. Note agora que no bloco de soma o sinal superior é negativo e o inferior é positivo, sendo então a saída a diferença das entradas que representa \dot{x} de acordo com a equação diferencial após substituir os valores de p e b .

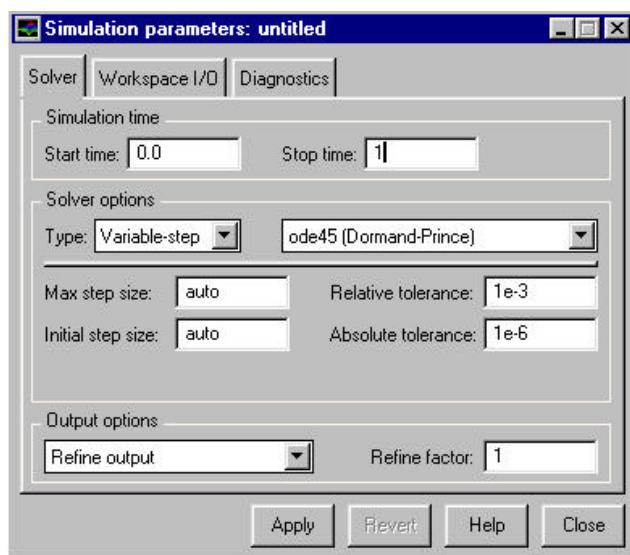


No SIMULINK, o sentido \downarrow no diagrama é representado pelo sentido \rightarrow das caixas de diálogos.

Para finalizar a configuração, deve-se definir o número inicial de bactérias. Para isto, dê um duplo clique no integrador e no campo **Initial condition** mude para 100, e após clique **Close**.



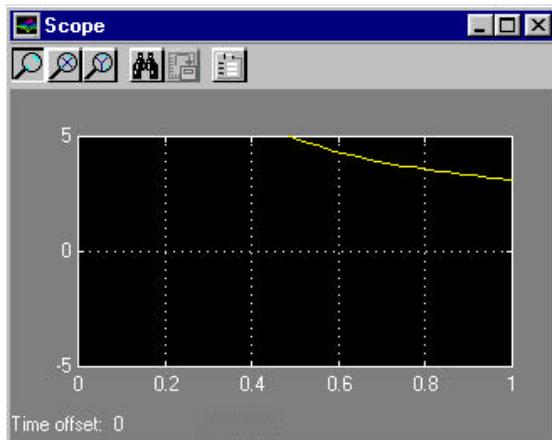
A duração da simulação é definida no tempo *default* de 0 a 10. Neste caso, deseja-se saber o resultado após 1 hora. Para mudar este tempo, seleciona-se na barra de menu a opção **Simulation:Parameters** e no campo **Stop Time** digita-se 1, fechando em **Close** logo a seguir.



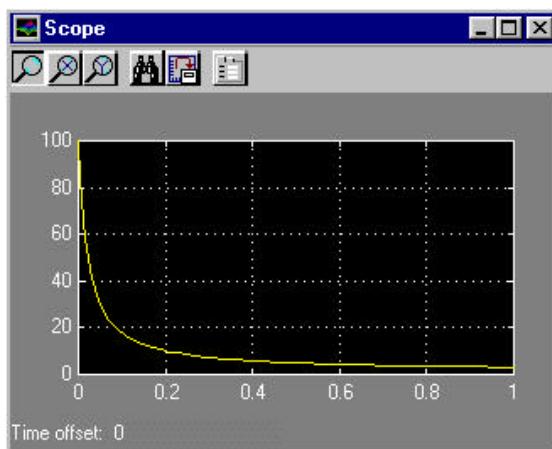
É sempre aconselhável salvar o modelo antes de executar a simulação.

O modelo agora está completo e pronto para ser executado. Para salvar na barra de menu clique em **File:Save** e entre com o nome desejado. O modelo será salvo com o nome digitado e a extensão *.mdl*, e seu nome aparecerá na barra de título da janela de edição.

Abra agora o SCOPE com um duplo clique e a seguir na barra de menu, clique em **Simulation:Start** para iniciar a execução.



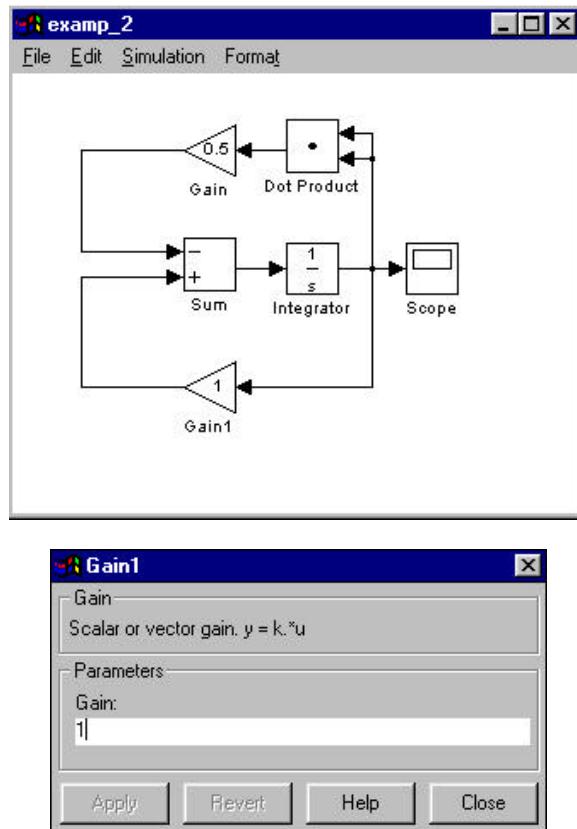
O SCOPE nem sempre mostra a figura numa boa escala para visualização. O botão **Autoscale** na barra de ferramentas do SCOPE redimensiona a escala para acomodar todos os valores.



2.4 - Usando o Help do SIMULINK

O SIMULINK possui um extensivo sistema de help on-line. Os arquivos de help foram desenvolvidos para serem visualizados por navegadores internet como Netscape ou Internet Explorer. Uma detalhada documentação on-line para todos os blocos do SIMULINK está disponível no *Block-Browser*. Um detalhado help também está disponível clicando no botão de help na caixa de diálogo que se abre quando se seleciona **Simulantion:Parameters** na barra de menu.

Para se consultar o help sobre um bloco qualquer deve-se inicialmente dar um clique duplo sobre o bloco desejado. A seguir clica-se no botão de help que aparece na caixa de diálogo que se abre. O seu Navegador Internet irá abrir o *Block Browser* correspondente.



Na janela do *Block Browser* existem 3 quadros. O superior contém um ícone para cada biblioteca do SIMULINK. O ícone selecionado é destacado com uma sombra.

O quadro inferior esquerdo contém ícones para todos os blocos contidos na biblioteca selecionada no quadro superior. Clicando num bloco nesta janela o help correspondente aparece no quadro inferior direito.

No quadro superior existe ainda um campo de pesquisa denominado **Search**, no qual pode se localizar rapidamente um bloco, mesmo quando não se sabe a que biblioteca este pertence.

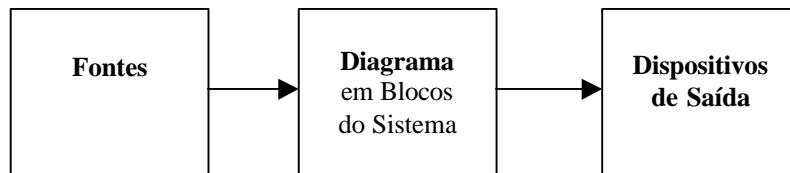
O help do SIMULINK contém informações valiosas. É boa prática utilizá-lo com freqüência.

Outra boa prática é o uso do notebook em anotações durante a aula. Isto facilita a execução de certas rotinas, por muitas vezes modificando somente parâmetros de configuração.

Capítulo 3 – Construindo Modelos SIMULINK

3.1 - Elementos de Modelos

Um modelo SIMULINK consiste de 3 tipos de componentes: Fontes, o sistema a ser modelado e dispositivo de saída.



Elementos de um Modelo SIMULINK

O elemento central, o sistema, é a representação de um diagrama em blocos de um sistema dinâmico a ser modelado no SIMULINK.

As fontes são as entradas aplicadas ao sistema dinâmico. Podem incluir constantes, geradores de funções como senóides ou degrau, ou ainda sinais personalizados pelo usuário criados no MATLAB. São encontrados na biblioteca de fontes (sources).

A saída do sistema é entregue aos dispositivos de saída. Alguns exemplos de saídas são gráficos, osciloscópios e arquivos de saída. Tais blocos são encontrados na biblioteca de Dispositivos de Saída.

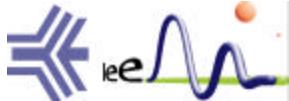
Freqüentemente, em modelos SIMULINK um ou mais desses 3 elementos pode faltar. Por exemplo, pode-se desejar modelar o comportamento na ausência de forças de um sistema inicialmente fora de sua condição de equilíbrio. Tal modelo não deve ter entradas mas deve conter blocos de sistema, tais como ganho, integradores etc, e provavelmente dispositivos de saída. Também é possível construir modelos que possuem fontes e dispositivos de saída, mas nenhum bloco de sistema. Suponha por exemplo que se necessita de um sinal que seja composto da soma de vários outros sinais. Tais sinais podem facilmente ser gerados usando as fontes do SIMULINK e enviados ao MATLAB ou a um arquivo no disco rígido.



3.2 - Manipulando Blocos

A tabela a seguir contém as operações básicas de manipulação de blocos como redimensionar, rotacionar, copiar e renomear blocos.

Selecionar objeto (blocos ou linhas de sinal)	Clique no objeto com o botão esquerdo do mouse.
Selecionar outro objeto	Pressione a tecla SHIFT e clique no outro objeto.
Selecionar com uma caixa de seleção	Clique com o botão esquerdo do mouse no local onde se deseja que seja uma das quinas da caixa de seleção. Continue com a tecla do mouse pressionada e arraste a caixa para encobrir a área desejada.
Copiar um bloco de uma biblioteca ou de outro modelo	Selecione o bloco e arraste para a janela do modelo para o qual se quer copiar.
Inverter blocos	Selecione o bloco e no menu Format:Flip Block . Tecla de atalho: CTRL-f .
Rotacionar blocos	Selecione o bloco e no menu Format:Rotate Block . Tecla de atalho: CTRL-r .
Redimensionar blocos	Selecione o bloco e arraste o canto.
Adicionar sombra	Selecione o bloco e no menu Format>Show Drop Shadow .
Editar o nome de um bloco	Clique no nome.
Ocultar o nome de um bloco	Selecione o nome, no menu Format:Hide Name
Inverter o nome de um bloco	Selecione o nome, no menu Format:Flip Name
Apagar objetos	Selecione o objeto, no menu Edit:Clear . Tecla de atalho: Del .
Copiar objetos para a área de transferência	Selecione o objeto, no menu Edit:Copy . Tecla de atalho: CTRL-c .
Recortar objetos para a área de transferência	Selecione o objeto, no menu Edit:Cut . Tecla de atalho: CTRL-x .



Curso de Introdução ao SIMULINK

Colar objetos a partir da área de transferência No menu **Edit:Copy**. Tecla de atalho: **CTRL-v**.

Traçar uma linha de sinal Arrastar com o mouse da saída do bloco para a entrada do outro.

Traçar uma linha de sinal em segmentos Arraste com o mouse da saída do bloco até o primeiro ponto. Repetir deste ponto até o seguinte e assim por diante.

Traçar uma linha ligada a outra Mantenha a tecla **CTRL** pressionada e clique sobre a linha de origem. Tecla de atalho: Clicar com o botão direito do mouse a partir da linha de origem.

Separar uma linha Selecionar a linha. Mantendo a tecla **SHIFT** pressionada, clique e arraste o novo vertex para a posição desejada.

Mover um segmento de linha Clique e arraste o segmento desejado.

Mover um vertex de um segmento. Clique e arraste o vertex desejado.

Nomear uma linha de sinal Duplo clique na linha e digite o nome.

Mover o nome de uma linha de sinal Clique e arraste o nome para a posição desejada.

Copiar o nome de uma linha de sinal Mantendo a tecla **CTRL** pressionada, arraste o nome para a posição desejada. Tecla de atalho: clique e arraste com o botão direito do mouse para a posição desejada.

Sinais de propagação numa linha de sinal Dê um nome aos sinais conhecidos (entrada) com um único caracter e nas linhas em que se deseja saber seu conteúdo, deve-se digitar somente o caracter "<". Após isso, no menu **Edit:Update Diagram**.

Acrescentar anotação ou observação no modelo Dê um duplo clique no local em que se deseja e digite o texto.

3.3 - Fontes

As entradas de um modelo são chamadas fontes (Sources) e podem ser encontradas na biblioteca de fontes. Um bloco de fonte não possui entrada e deve possuir pelo menos uma saída. A documentação detalhada de cada fonte pode ser encontrada no help do SIMULINK. No texto que segue serão mencionadas somente os tipos mais comuns e utilizados de fontes. Serão ainda discutidas as operações de importação do MATLAB e de arquivos que contenham dados os quais se deseja inserir no modelo. Tal facilidade permite que se tenha qualquer tipo de sinal de entrada, exista ele no SIMULINK ou não.

3.3.1 - Fontes Comuns

Muitos tipos de sinais de entrada utilizados em modelos de sistemas dinâmicos estão disponíveis na biblioteca de fontes.

O bloco Constante (Constant) produz um sinal fixo que possui a magnitude escolhida com um duplo clique sobre o bloco.

O bloco Degrau (Step) produz uma função degrau. Pode-se configurar o instante em que se aplica o degrau, assim como sua magnitude antes e depois da transição.

O bloco de Onda Senoidal (Sine Wave) permite que se configure a amplitude, a fase e a freqüência da onda senoidal.

O Gerador de Sinais (Signal Generator) pode produzir ondas senoidais, quadradas, dente de serra ou sinais aleatórios.

Sinais mais complexos podem ser gerados a partir da combinação destes apresentados.

Exemplo: Impulso Unitário

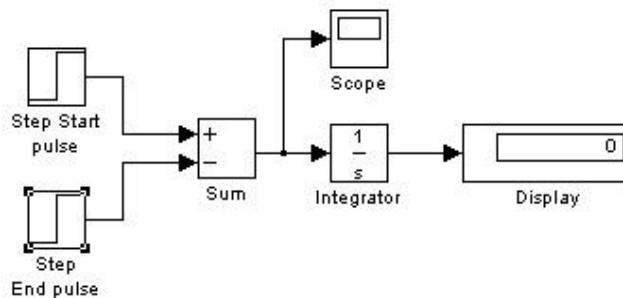
Um sinal muito utilizado para determinar o comportamento dinâmico de sistemas é o *Impulso Unitário*, também conhecido como *Função Delta* ou *Função Delta de Dirac*. O Impulso Unitário ($\delta(t-a)$) é definido como um sinal de duração igual a zero, tendo as seguintes propriedades:

$$\begin{cases} \delta(t-a) = 0, & t \neq a \\ \int_{-\infty}^{\infty} \delta(t) dt = 1 \end{cases}$$

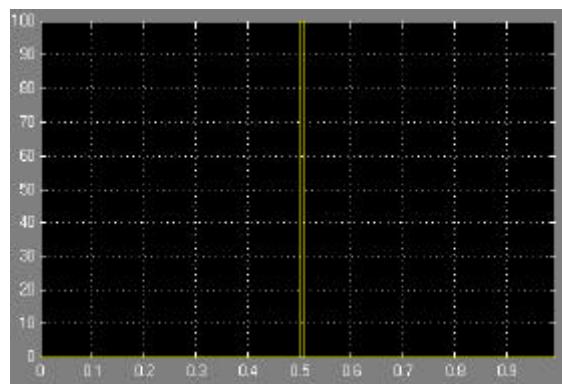
Embora o impulso unitário seja um sinal que teoricamente não existe, existem boas aproximações do caso ideal que são muito comuns. Exemplos físicos são colisões, como uma roda se chocando com o meio-fio ou um bastão rebatendo uma bola ou ainda mudanças instantâneas de velocidade como a de uma bala sendo disparada de um rifle. Outra utilidade da função impulso é a determinação da dinâmica do sistema. O movimento causado em um sistema que sofre uma força impulsional unitária é a própria dinâmica inerente ao sistema. Partindo disto, pode-se utilizar a

resposta à um impulso unitário de um sistema complexo para se determinar sua freqüência natural e suas características de vibração.

Pode-se criar uma aproximação de um impulso unitário utilizando duas fonte de função degrau e um bloco de soma. A idéia é produzir num tempo definido “a” um pulso de duração muito curta “d” e de magnitude “M”, tal que $M \cdot d = 1$. A dificuldade consiste em se definir um valor apropriado para d. Deve ser um valor pequeno quando comparado à mais rápida dinâmica do sistema. Porém, se for muito curto, pode ocorrer problemas numéricos como erros de aproximação. Se for muito longo a simulação não será adequada à um impulso verdadeiro. Usualmente, valores adequados podem ser determinados experimentalmente.

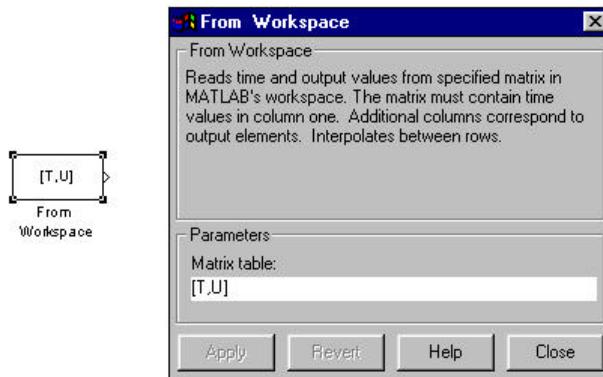


O modelo na figura acima deve ser ajustado para simular um impulso unitário ocorrido aos 0.5 segundos de simulação com uma duração de 0.01 segundos e a magnitude de 100. A fonte degrau na parte superior deve ter a seguinte configuração: **Step time:** 0.5, **Initial value:** 0, **Final value:** 100. Já a fonte situada na parte inferior deve ser configurada da seguinte forma: **Step time:** 0.51, **Initial value:** 0, **Final value:** 100. A simulação deve ser configurada para terminar em 1 segundo. O gráfico gerado no SCOPE é mostrado na figura abaixo. A saída do integrador contém o valor da integral do impulso no decorrer do tempo e é mostrada no bloco Display. Se a simulação foi correta tal valor deve ser 1, o que condiz com o valor teórico.



3.3.2 - Importando do MATLAB (From Workspace Block)

Este bloco permite ao usuário criar seu próprio sinal de entrada. O bloco e sua caixa de diálogo são mostrados em sua figura a seguir.



Na configuração deste bloco deve-se definir quais serão as matrizes tabela utilizadas como fonte de sinal. O valor *default* do SIMULINK são as matrizes [T,U], que devem ser previamente definidas no MATLAB antes da execução da simulação. A primeira coluna da matriz deve ser a variável independente que corresponde ao tempo na simulação estritamente crescente. As colunas seguintes são os valores das variáveis independentes correspondentes à valores das variáveis dependentes da primeira coluna. As saídas serão produzidas por interpolação linear ou extrapolação utilizando as matrizes definidas.

Exemplo

Para ilustrar o uso do bloco vamos supor que se deseja gerar um sinal definido por:

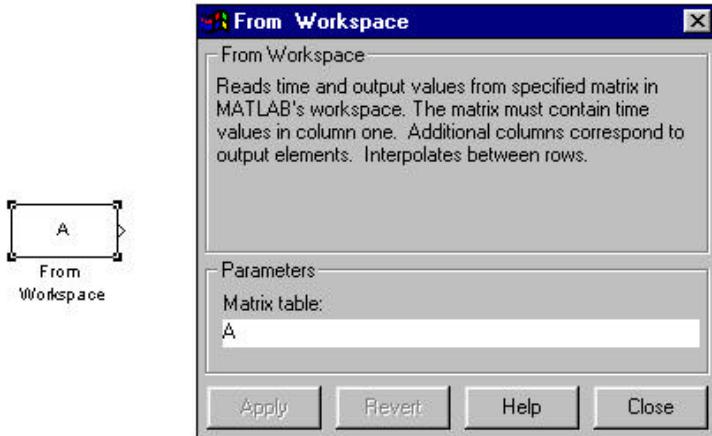
$$u(t) = t^2$$

Inicialmente deve-se gerar a tabela que contenha os valores da função no MATLAB. A seguir temos os comandos necessários que devem ser executados na área de trabalho do MATLAB:

```
>> t=0:0.1:100;           % Variável independente
>> u=t.^2;                % Variável dependente
>> A=[t',u'];              % Formato da tabela
```

É importante notar que os sinais devem ser carregados em colunas, o que exige que se tenha a matriz A composta das matrizes transpostas de u e t.

Criada a tabela, deve-se agora configurar o bloco para que este receba os valores desejados. Com um duplo clique sobre o bloco, deve-se digitar o nome da matriz definida no MATLAB, neste caso “A”, e a seguir clicar no botão **Close**. O nome da matriz aparecerá sobre o bloco e este está pronto para ser usado como uma fonte no SIMULINK.



3.3.3 - Importando Arquivos Gerados no MATLAB (From File Input Block)

Este bloco é muito similar ao anterior. A diferença básica é que a matriz é agora carregada a partir de um arquivo gerado no MATLAB, podendo agora o sinal de entrada ser salvo para diversos usos posteriores. Outra diferença importante é que os sinais devem agora ser carregados em linhas ao invés do caso anterior, em que eram carregados em colunas.

Aproveitando o resultado anterior, podemos executar no MATLAB os seguintes comandos:

```
>> B=A' % Variável independente  

>> save exemp B % Variável dependente
```

Agora pode-se configurar o bloco para receber os valores digitando no campo **File Name** o nome completo do bloco, inclusive sua extensão (.mat), no nosso caso **exemp.mat**.

3.4 - Dispositivos de Saída

São as maneiras de se ver ou armazenar os resultados de um modelo. O osciloscópio e o gráfico XY produzem gráficos a partir de dados do modelo. O bloco Display produz uma amostragem digital do valor contido em sua entrada. Pode-se ainda enviar os dados para a área de trabalho do MATLAB utilizando o bloco To Workspace Block ou ainda armazenar os dados em arquivos do MATLAB para usos posteriores. O bloco de parada (Stop block) causa a parada da simulação quando sua entrada for diferente de zero. Uma referência detalhada de cada bloco pode ser encontrada no help do SIMULINK. No texto que segue serão discutidas particularidades do bloco SCOPE e do gráfico XY.

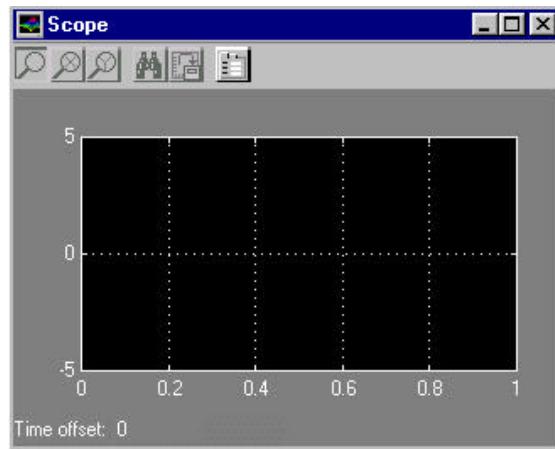
3.4.1 - Osciloscópio (Scope)

Este bloco emula um osciloscópio. Plota um gráfico do sinal de entrada, podendo este ser um escalar ou um vetor. Ambas as escalas, vertical e horizontal podem ser ajustadas para uma melhor visualização. A escala vertical mostra o valor atual correspondente ao sinal de entrada. A escala horizontal sempre inicia em zero e

termina no valor especificado em **Time Range**. Se por exemplo a faixa de escala horizontal é 10 e o tempo corrente é 100, o sinal de entrada correspondente ao período de 90 a 100 é que será mostrado, muito embora a escala horizontal esteja continue mostrando de 0 a 10. O osciloscópio simplesmente mostra o sinal durante a simulação, e é desprovido de métodos de salvar a imagem gerada para impressão ou inclusão em outro documento. Mas tem a possibilidade de enviar os dados para a área de trabalho do MATLAB para possível análise ou plotagem, usando por exemplo o comando **plot** do MATLAB.

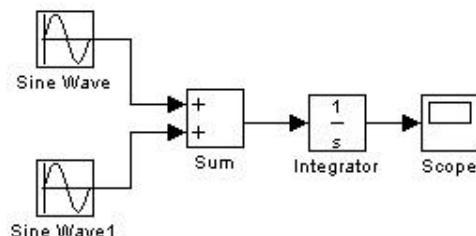
Pode-se inserir o SCOPE no modelo sem qualquer conexão em sua entrada e deve ser configurado como Bloco SCOPE Flutuante (Floating Scope Block). Este SCOPE flutuante será então usado para se visualizar qualquer sinal do modelo durante a execução da simulação com um simples clique sobre a linha em que se deseja conhecer o sinal.

A figura abaixo ilustra o bloco SCOPE. Nota-se que há uma barra de ferramentas que contém seis ícones ao longo do topo da janela. Esses botões permitem dar um *zoom* numa porção da figura, autodimensionar a figura, salvar a configuração para futuro uso e abrir a caixa de diálogo com as propriedades do bloco SCOPE.



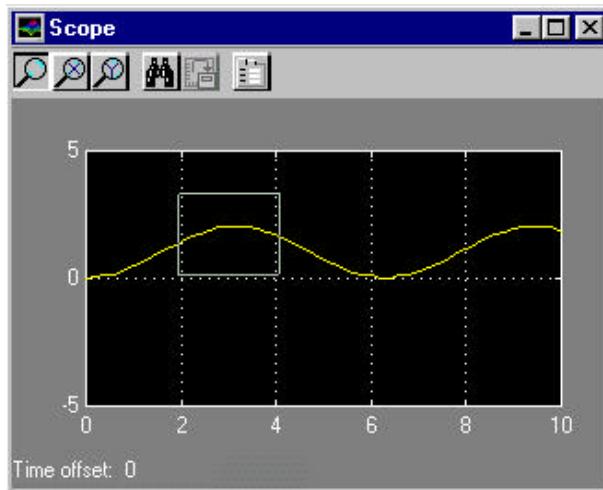
3.4.1.1 - Dando Zoom na tela do Osciloscópio

Considere o modelo mostrado na figura. O bloco gerador de onda senoidal situado na parte superior está configurado para produzir o sinal $\sin(t)$ e o outro bloco para gerar o sinal $0.4\sin(10t)$.

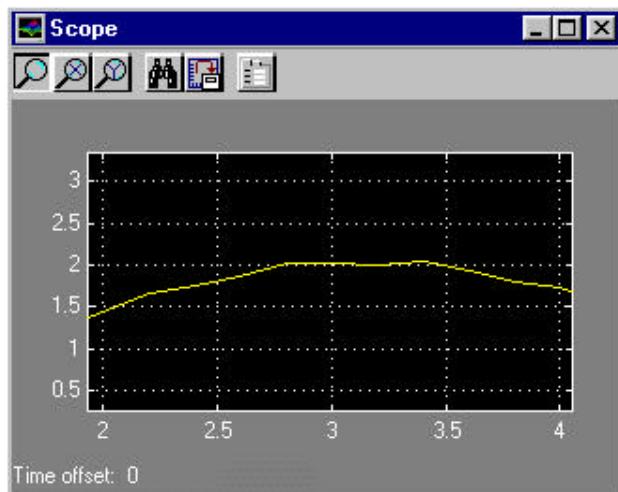


Abre-se a tela do Osciloscópio com um duplo clique sobre ele. Executa-se então a simulação e o resultado obtido deve ser semelhante ao mostrado na figura. Pode-se

redimensionar a figura clicando no botão **Autoescale**. Suponha que se deseja analisar o intervalo de tempo entre 2 e 4 segundos. Para isto, clica-se no botão **Zoom**. Após isto deve-se com o mouse então envolver a área desejada como mostrado na figura.



A tela agora só mostrará o intervalo envolvido.

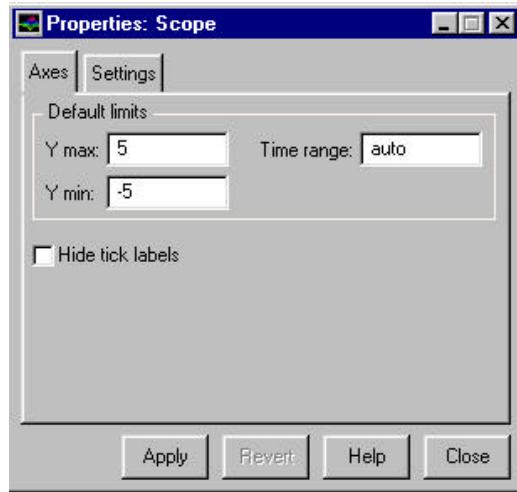


3.4.1.2 - Propriedades do Osciloscópio

O botão **Open properties window** abre a caixa de diálogo com as propriedades do osciloscópio, caixa esta que contém duas páginas.

A primeira página (Axes) contém campos para se definir os valores máximos (**Ymax**) e mínimos (**Ymin**) da variável dependente.

A escala de tempo (**Time Range**) pode ser configurada para um valor particular ou para seu modo automático. Se o modo automático for escolhido, a escala de tempo terá o mesmo tamanho da duração da simulação especificada na caixa de diálogo **Simulation:Parameters**.

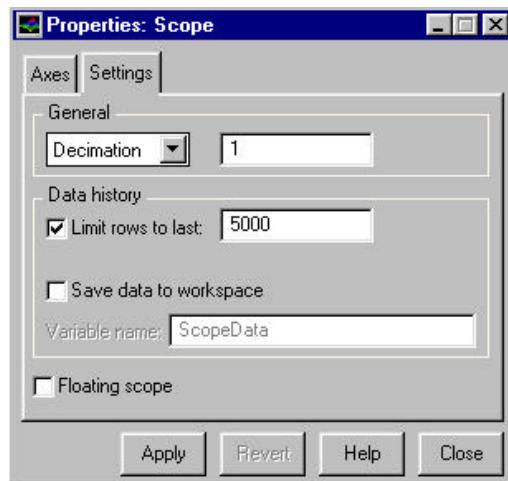


A segunda página (Settings) contém campos para se controlar o número de pontos mostrados e salvar os dados para a área de trabalho do MATLAB.

A seção **General** da página consiste numa caixa contendo duas opções: **Decimation** e **Sample time**.

Se a opção escolhida for **Decimation** o campo que contém o fator deve ser um número inteiro. Escolhendo-se **Decimation** e definindo-se 1 no campo ao lado da caixa, todo os pontos na entrada do bloco serão plotados. Se define-se 2 para o campo, a plotagem é feita com pontos alternados e assim por diante.

Se for escolhida a opção **Sample time**, o espaçamento absoluto entre os pontos a serem plotados deve ser digitado no campo.

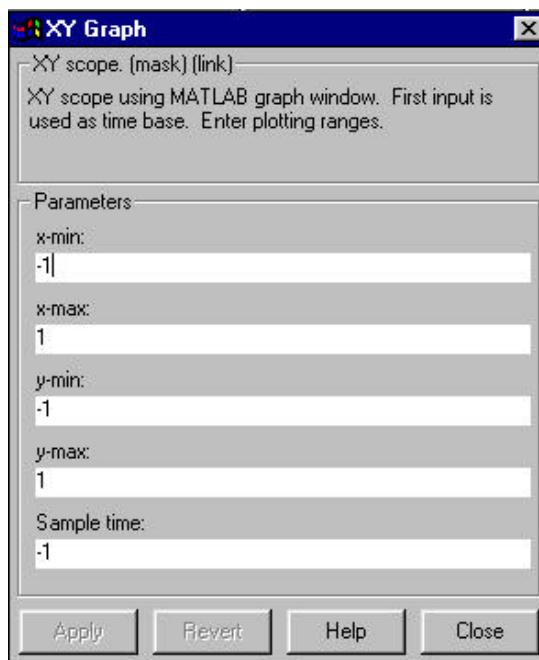


O osciloscópio armazena os pontos num buffer. Assinala-se a opção **Limits rows to last** e digita-se o valor específico para se definir o tamanho do buffer(o default é 5000). As operações de Autoescala, zoom e salvar os dados para a área de trabalho são feitas com este buffer. Se por exemplo ajusta-se o tamanho do buffer para 1000 pontos e a simulação gera 2000 pontos, somente os últimos 1000 estarão disponíveis para estas operações quando a simulação terminar.

Como dito anteriormente, o osciloscópio não possui ferramentas para se imprimir ou enviar sua tela para outros programas diretamente, como um processador de textos por exemplo. Porém, os dados enviados ao osciloscópio podem ser armazenados numa variável do MATLAB e enviados à área de trabalho, podendo então ser utilizadas as muitas capacidades de tratamento destes dados pelo MATLAB. Para isto deve-se primeiro assinalar a opção **Save data to workspace** e digitar o nome da variável do MATLAB. Depois da simulação terminada, os dados mostrados na tela do osciloscópio serão armazenados na variável definida. A variável será composta de uma coluna contendo os valores de tempo e outra coluna para cada sinal de entrada. Se por exemplo o sinal de entrada for um vetor de sinal com duas componentes, a variável terá 3 colunas e o número de linhas igual ao número de pontos mostrados na tela do osciloscópio.

3.4.2 - Gráfico XY

O bloco de gráfico XY produz um gráfico idêntico ao gráfico produzido pelo comando **plot** do MATLAB. O gráfico XY aceita dois escalares como entrada. Deve-se configurar as faixas horizontais e verticais utilizando a caixa de diálogo do bloco. A caixa de diálogo do bloco gráfico XY é mostrada na figura que se segue.



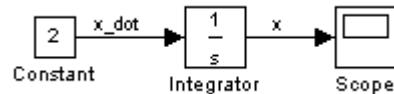
3.5 - Configurando a Simulação

Um modelo SIMULINK é essencialmente um programa de computador que define um grupo de equações diferenciais e de diferenças. Quando se escolhe **Simulation:Start** na barra de menu da janela de modelo, o SIMULINK resolve o grupo de equações diferenciais e de diferença numericamente, utilizando um dos seus algoritmos de resolução.

Antes de se executar uma simulação, deve-se configurar vários parâmetros, como tempo inicial e final, **Step size** da simulação e várias tolerâncias. Pode-se escolher

vários algoritmos de integração de alta qualidade. Pode-se também configurar o SIMULINK para obter e enviar dados da área de trabalho do MATLAB.

Considere-se o modelo mostrado na figura



Este modelo representa a seguinte equação diferencial:

$$\dot{x} = 2$$

Defini-se para condição inicial do integrador valor 1 no campo **Initial condition**. A seguir, escolhe-se no menu da janela do modelo **Simulation:Parameters** e configura-se o tempo inicial para 0 e o tempo final para 5. A seguir se executa a simulação. O SIMULINK determinará numericamente os valores da integral para resolver

$$x(t) = 1 + \int_0^t 2 dt$$

e plota os valores de $x(t)$ no intervalo de 0 a 5. Um algoritmo de integração numérica que resolve este tipo de problema (frequentemente chamado de *problema de valor inicial*) é chamado de *algoritmo de resolução de equações diferenciais ordinárias (Ordinary differential equation solver)*.

Para se configurar os parâmetros da simulação escolhe-se **Simulation:Parameters** na barra de menu da janela do modelo. A caixa de diálogo que se abrirá contém três páginas: *Solver*, *workspace I/O* e *Diagnostics*.

Na primeira página (*Solver*) seleciona-se e configura-se o algoritmo de resolução da equação diferencial. A segunda página (*Workspace I/O*) contém parâmetros opcionais que permitem obter dados de inicialização da área de trabalho do MATLAB e enviar certos dados da simulação para variáveis previamente definidas da área de trabalho do MATLAB. A terceira página (*Diagnostics*) é usada para selecionar alguns métodos de diagnósticos muito utilizados para se determinar problemas na simulação. Cada página será discutida mais detalhadamente.

3.5.1 - Solver Page

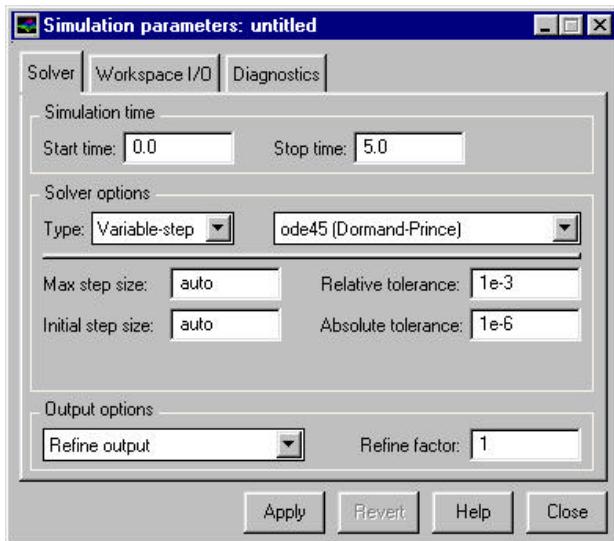
Esta página consiste em três seções. A primeira, *Simulation time*, contém campos para se definir o tempo inicial e final da simulação. Os valores *default* são 0 e 10.

As opções **Solver** contém campos para se selecionar os algoritmos numéricos de integração para se resolver as equações diferenciais e configurar parâmetros que controlam o **Step Size** de integração. Os métodos de solução são agrupados em duas categorias: **Passo variável** e **passo fixo**. Diferentes algoritmos estão disponíveis para cada categoria.

Se a opção escolhida for a de **Passo variável**, estão disponíveis campos para se configurar o tamanho do máximo passo de integração, o passo inicial e as tolerâncias absolutas e relativas.

Se a opção escolhida for a de **Passo fixo**, há um único campo no qual se define o tamanho do passo de integração.

A seção **Output options** controla o espaçamento de tempo nos pontos na trajetória de saída.



3.5.1.1 - Solver Type

O SIMULINK fornece diferentes tipos de algoritmos de solução para equações diferenciais. A maioria destes algoritmos são resultados de recentes pesquisas em integração numérica e estão entre os mais rápidos e precisos métodos disponíveis.

Geralmente se usam métodos de passo variáveis, pois eles ajustam continuamente o passo de integração maximizando a eficiência enquanto mantém uma precisão especificada.

Os métodos disponíveis são listados e sucintamente discutidos:

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. Às 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-Basforth-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.



3.1.5.2 - Opcões de Saída (Output Options)

As opções de saída da página só estão disponíveis para métodos de passo variáveis e controlam o espaçamento entre pontos na trajetória de saída. Estas opções não se aplicam à métodos de passo fixo. Os campos de opções de saída contém uma caixa com três opções: refinar a saída (Refine output), produzir uma saída adicional (Produce additional output), e produzir somente uma saída especificada (Produce specified output only).

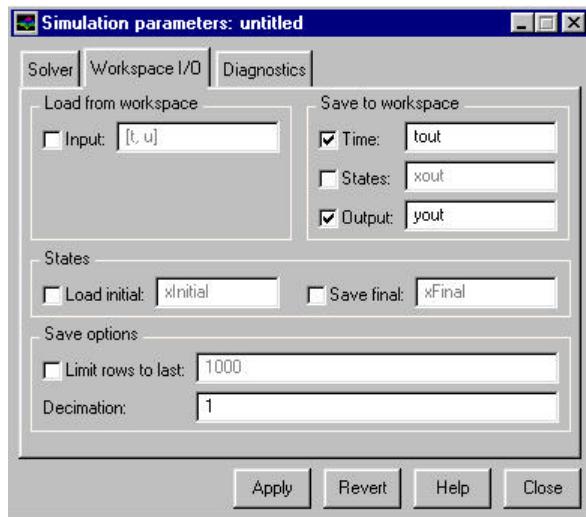
A opção Refinar a saída força o algoritmo a adicionar pontos intermediários entre os pontos da solução fazendo sucessivos passos. O SIMULINK computa os pontos intermediários utilizando interpolação, o que é muito mais rápido do que reduzir o tamanho do passo de integração. É uma boa opção quando se deseja que a trajetória de saída pareça mais homogênea e não se necessita que o espaço entre os pontos seja fixo. Se esta opção for selecionada deve-se ainda definir o fator de refinamento (Refine factor). Este fator deve ser inteiro. O SIMULINK divide cada intervalo de integração pelo fator na saída. Por exemplo, se o fator for 2, o ponto médio de cada intervalo de integração será adicionado na trajetória de saída por interpolação.

A opção de produzir uma saída adicional (Produce additional output) permite forçar o SIMULINK a incluir certos pontos de tempo na trajetória de saída, em adição aos pontos da solução no fim de cada intervalo de integração. Se esta opção for selecionada, deve-se então preencher o campo intitulado Tempos De Saída (Output times). Este campo deve conter um vetor que lista os tempos adicionais para os quais se deseja conhecer a saída. Se se deseja, por exemplo, incluir a saída a cada 10 segundos de intervalo e o tempo inicial é 0 e o final é 100, o vetor deve ser [0:10:100].

A opção de produzir somente uma saída especificada (Produce specified output only) é utilizada quando se deseja produzir uma trajetória de saída contendo somente valores específicos de tempo; por exemplo quando se deseja comparar as diferenças de trajetórias e avaliar os efeitos de mudanças de alguns parâmetros. Se esta opção for selecionada, deve-se preencher o campo chamado tempos de saída (Output times), o qual deve conter um vetor com os instantes de tempo para os quais se deseja conhecer a saída.

3.5.2 - Página Workspace I/O

A segunda página permite obter e enviar dados para área de trabalho do MATLAB.



Na seção Carregar da área de trabalho (Load from workspace), a caixa Entrada (Input) faz com que o SIMULINK importe do MATLAB o tempo e valores previamente armazenados nas variáveis definidas no campo logo ao lado da caixa. Esta opção trabalha em conjunto com o bloco Import encontrado na biblioteca de conexões (Connections Library). Este bloco deve ser configurado para aceitar dados escalares ou vetores. Define-se então o nome das matrizes tempo (default t) e sinais de entrada (default u) no campo entrada (Input). A primeira matriz (t) é um vetor coluna de valores de tempo e a segunda (u) consiste em uma coluna para cada variável de entrada, com cada linha correspondente à matriz de tempo. Se houver mais de um bloco Import, as colunas da matriz serão ordenadas com o número correspondente ao assinalado no bloco. A primeira coluna corresponderá ao bloco com menor número e a última coluna ao bloco com maior número. Para um vetor no bloco Import, deve haver uma coluna na matriz u para cada elemento do vetor de entrada.

3.5.2.1 - Vetores de estado internos do SIMULINK

Antes de se discutir as seções Salvar para a área de trabalho (Save to Workspace) e Estados (States) deve-se discutir brevemente falar das variáveis de estado internas do SIMULINK.

Um modelo SIMULINK pode ser entendido como um grupo de equações diferenciais ou de diferença de primeira ordem, possivelmente não lineares simultâneas. Além das variáveis de estado associadas a cada integrador, há também variáveis de estado implícitas especificadas associadas com os blocos de função de transferência (Transfer function block), os blocos de Espaço de Estados (State – Space block), certos blocos não lineares, alguns blocos discretos e muitos blocos existentes na biblioteca de Extras.

Freqüentemente é necessário ter acesso às variáveis de estado do modelo e o SIMULINK fornece mecanismos que facilitam estas tarefas. Utilizar a página Workspace I/O é provavelmente a maneira mais fácil de acessar estas variáveis.



3.5.2.2 - Salvar para a área de trabalho

Esta seção contém três campos, cada um ativado com a caixa correspondente.

A opção Tempo (Time) envia a variável indenpendente para a área de trabalho com um nome definido (o default é tout).

A opção Estados (States) envia todas as variáveis de estado dos modelos para a área de trabalho do MATLAB numa matriz especificada (o default é xout).

Já a opção Saída (Output) trabalha em conjunto com o bloco Outport de maneira análoga ao bloco Inport, já discutido previamente.

3.5.2.3 - Estados (States)

A seção Estados (States) pode forçar o SIMULINK a carregar valores iniciais de todas as variáveis de estados internas a partir do MATLAB e também enviar os valores finais da variáveis de estado internas para a área de trabalho. Todas as variáveis de estado do SIMULINK possuem valores iniciais default, na maioria dos casos 0. Os estados associados à blocos integradores devem ser configurados pela própria caixa de diálogo do bloco.

Ao se especificar valores de estados iniciais nesta página, o SIMULINK desconsidera qualquer valor default de inicialização, incluindo valores iniciais configurados nos blocos integradores. A caixa Load initial configura o valor inicial de um vetor de estado do modelo para os valores especificados no vetor de entrada (default xInitial), predefinido na área de trabalho do MATLAB. O vetor de inicialização deve ter o tamanho do vetor de estado do modelo.

Na caixa Save Final é possível fazer com que o SIMULINK salve os valores finais das variáveis de estado num vetor especificado (default xFinal) para a área de trabalho do MATLAB. A saída desta opção é um vetor que pode ser utilizado no futuro como condições iniciais numa outra simulação, o que corresponde a continuar de onde a anterior foi interrompida.

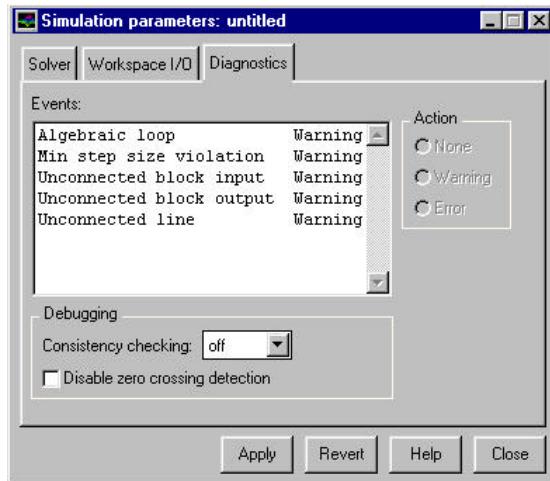
3.5.2.4 - Save Options

Esta seção da página contém dois campos e trabalham em conjunto com a seção Save to workspace.

O primeiro campo (Limit rows to last) define-se o número de pontos que será enviado para o MATLAB. Por exemplo, se no campo for definido 1000, os últimos 1000 pontos serão enviados para a área de trabalho do MATLAB. O campo Decimation configura o intervalo entre os pontos que serão enviados. Este campo só deve conter números inteiros. Se, por exemplo, for definido o valor 1 neste campo, todos os pontos serão enviados para o MATLAB.

3.5.3 - Página de Diagnósticos

Esta página permite selecionar a ação tomada para cinco condições e também inclui opções para o controle automático de saída de blocos (Consistency Checking) e desabilitar a detecção da transição de zero.



Há três escolhas para a resposta de cada um das cinco condições excepcionais. A primeira **None** instrui ao SIMULINK a ignorar o evento correspondente. A segunda escolha, **Warning** (default) faz com que o SIMULINK emita uma mensagem de erro cada vez que o evento ocorrer. A opção final **Error** faz com que o SIMULINK aborte a simulação toda vez que o evento correspondente ocorrer. Os eventos são detectados quando a simulação é executada. Cada um dos eventos serão brevemente discutidos.

Um Loop Algébrico (Algebraic loop) é um evento no qual a entrada de um bloco num dado instante de tempo é dependente da saída do mesmo bloco no mesmo instante de tempo. Este tipo de problema causa perdas de velocidade na execução da simulação e em alguns casos pode causar erros na execução da simulação. É comumente aconselhável configurar **Warning** como resposta para loops algébricos, mas se um loop algébrico for detectado e a performance for aceitável, pode-se configurar **None** como resposta.

Uma violação de tamanho do passo mínimo (Min step size violation) ocorre quando a solução tende a usar um passo de integração menor do que o passo mínimo. Como nem sempre é possível modificar o tamanho do passo mínimo em muitos métodos de passo variável deve-se então trocar para um método de ordem superior ao utilizado, que em geral utiliza um maior passo de integração. Uma outra opção é aumentar o valor da tolerância absoluta e relativa na primeira página de configuração. Este tipo de erro deve ser sempre configurada para **Warning** ou **Error** porque este tipo de erro indica que a simulação não está produzindo a precisão desejada.

Um erro Entrada de Bloco Desconectada (Unconnected block input) ocorre quando um bloco tem uma entrada não utilizada. Geralmente é resultado de um erro na construção do modelo. Deve ser sempre configurada para **Warning** ou **Error**. Se a omissão da entrada de um bloco for intencional é uma boa prática conectar esta entrada a um bloco Terra (Ground).



Um erro Saída de Bloco Desconectada (Unconnected block output) acontece quando uma saída de um bloco não está conectada à entrada de outro. Este tipo de erro frequentemente não causa erro à simulação, mas é muito simples de se resolver. Para evitar este tipo de erro deve-se conectar todas as saídas não utilizadas a um bloco **Terminator** encontrado na biblioteca de conexões (Connections). Este evento deve ser configurado para **Warning** ou **Error**.

O último evento é Linha desconectada (unconnected line) pode também ser causado num erro na construção do modelo. Este evento ocorre quando o final de uma linha de sinal não está conectada a um bloco. Deve ser configurado para **Error**.

A Checagem de consistência (Consistency checking) é uma característica de depuração que detecta certos erros de programação em alguns blocos personalizados. Não é necessário se utilizar esta função quando se tem somente blocos do próprio SIMULINK, o que faz com que a simulação seja executada mais lentamente. Em geral, esta opção deve ser configurada em **Off**.

Um grande número de blocos do SIMULINK possuem comportamento discontinuo. Como exemplo temos o bloco Sign localizado na biblioteca Nonlinear. Sua saída é 1 se a entrada for positiva, 0 se a entrada for nula e -1 se a entrada for negativa. O bloco então possui uma descontinuidade em zero. Se for utilizado um método de passo variável, o SIMULINK irá ajustar o passo de integração quando a entrada do bloco se aproximar de zero para que a mudança ocorra no tempo certo. Este processo é chamado *Zero crossing detection*. Pode-se determinar se um bloco necessita desta detecção na tabela de Características do bloco em suas configurações.

Esta detecção melhora a precisão da simulação, mas pode ocasionar uma lentidão da mesma. Ocasionalmente um sistema pode flutuar rapidamente pela descontinuidade, um fenômeno conhecido por reticência (chatter). Quando isto ocorre, o progresso da simulação pode efetivamente parar, já que o passo de integração é reduzido a valores muito pequenos. O modelo então será executado lentamente quando o mesmo incluir um ou mais blocos com este tipo de detecção intrínseca. Ao selecionar a opção **Disable zero crossing detection** na página de diagnósticos pode aumentar significativamente a velocidade de execução, mas pode também reduzir drasticamente a precisão da simulação. Tal ação deve então ser utilizada somente como uma ferramenta para se verificar se está ocorrendo a reticência. Se a opção for selecionada e a velocidade de execução aumentar consideravelmente, deve-se então localizar a causa da reticência e corrigir o problema.

3.6 - Executando uma Simulação

Pode-se controlar a execução de um modelo no menu **Simulation** na barra de menus da janela do modelo.

Para se iniciar uma execução, clica-se em **Simulation:Start**. Pode-se parar a simulação permanentemente selecionando-se **Simulation:Stop**. Para parar a execução temporariamente clica-se em **Simulation:Pause**, e para continuar a execução do ponto de parada seleciona-se **Simulation:Continue**.

Enquanto a simulação está sendo executada, pode-se modificar diversos parâmetros. Por exemplo o ganho de um bloco de ganho, modificar o algoritmo



utilizado na solução, modificar os parâmetros de integração como o tamanho do passo mínimo. Pode-se ainda selecionar uma linha de sinal que passará então a ser a entrada de um Osciloscópio que esteja sendo configurado como flutuante (Floating Scope Block). Isto permite que se cheque vários sinais durante o progresso da simulação.

3.7 - Imprimindo um Modelo

Há uma variedade de métodos para se imprimir modelos SIMULINK. O mais simples é enviar o modelo diretamente para a impressora. Há ainda a possibilidade de se enviar o modelo para um processador de textos ou outro tipo de arquivo, ou ainda copiar para a área de transferência ou salvar como um arquivo EPS (Encapsulated PostScript File), utilizado pela maioria dos programas processadores de textos ou imagens.

3.7.1 - Imprimindo um modelo utilizando os menus

A maneira mais rápida de se obter uma cópia impressa do modelo é enviando o mesmo diretamente para a impressora utilizando os menus. Clicando-se em **File:Printer Setup** na barra de menu da janela do modelo pode-se configurar a impressora como se deseja. A seguir, clicando-se em **File:Print** o modelo será enviado diretamente para a impressora. O SIMULINK ajusta o modelo para que o mesmo caiba na folha. O usuário não tem controle sobre o tamanho do modelo na impressão.

3.7.2 - Enviando o Modelo para um Documento

O SIMULINK permite que se insira seus modelos em vários tipos de programas modernos como processadores de texto, programas de apresentação, editores de imagem etc.

Os modelos podem ainda ser salvos como arquivos bitmaps, Windows metafiles ou EPS.

Para se obter uma figura como arquivo bmp ou wmf, primeiro se deve clicar **Edit:Copy Model** na barra de menu da janela do modelo. A seguir deve-se inserir a figura no documento desejado de acordo com o programa utilizado.

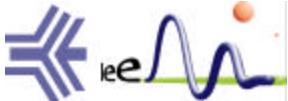
Arquivos Metafile são preferíveis porque neste formato a imagem não perde resolução quando se amplia, mas nem todos os programas trabalham bem com este tipo de imagem. Salvar como arquivo Bitmap é uma opção mais confiável.

Para se utilizar a opção de arquivo EPS deve-se primeiro salvar a imagem como um arquivo EPS utilizando o comando *print* brevemente discutido a seguir. A seguir inserí-lo no documento desejado utilizando procedimentos padrões de cada programa utilizado.

3.7.3 - Utilizando o Comando *PRINT* do MATLAB

Este comando permite enviar a imagem de modelos para uma impressora, para a área de transferência ou para um arquivo numa variedade de formatos. A sintaxe do comando é

```
print -smodel -ddevice filename
```



Curso de Introdução ao SIMULINK

Model é uma string do MATLAB que contém o nome do modelo SIMULINK que deve estar aberto. O modelo é mostrado no título da janela do modelo. Se o nome do modelo contiver espaços, deve então estar entre apóstrofos ('):

```
print -s'Spring Mass System' -ddevice filename
```

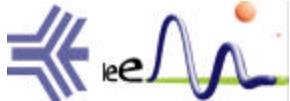
Se o nome do modelo tiver um Enter deve então estar entre colchetes ([]) com o Enter separado por apóstrofos (') e representado pelo seu código ASCII (13):

```
print -s['Damped' 13 'Spring Mass System'] -ddevice filename
```

device é uma string MATLAB que especifica o tipo de dispositivo de saída. Pode incluir impressoras, arquivos e a área de transferência do Windows. A tabela a seguir inclui os dispositivos e seus códigos.

Device	Descrição do dispositivo
ps	PostScript
psc	PostScript Colorido
ps2	PostScrip Nível 2
psc2	PostScrip Nível 2 Colorido
eps	Encapsulated PostScript (arquivo)
epsc	Encapsulated PostScript colorido (arquivo)
eps2	Encapsulated PostScript Nível 2 (arquivo)
epsc2	Encapsulated PostScript Nível 2 Colorido (arquivo)
win	Impressora Padrão
winc	Impressora Padrão Colorida
meta	Área de transferência – Formato Metafile
bitmap	Área de transferência – Formato Bitmap
setup	O mesmo que selecionar File:Printer Setup na barra de menu.

Filename é uma string do MATLAB que deve conter o nome do arquivo que será salvo. É opcional, já que nem sempre se salva como arquivo no disco.



Exemplo

O objetivo deste exemplo é primeiro imprimir na impressora padrão um modelo entitulado *xydemo.mdl* e a seguir salvar a imagem do modelo como um arquivo EPS colorido.

Primeiro deve-se abrir o modelo no SIMULINK. A seguir, na área de trabalho do MATLAB digita-se:

```
print -s'xydemo'
```

O modelo será impresso na impressora padrão do Windows.

A seguir digita-se:

```
print -s'xydemo' -depsc xydemo.eps
```

O modelo então será salvo como um arquivo EPS e poderá ser importado por qualquer programa que aceite este tipo de arquivo.



Capítulo 4 – Sistemas Contínuos no Tempo

Após estudar os capítulos anteriores, o usuário já deve se sentir bem mais confortável com os mecanismos de construção e execução de modelos no SIMULINK. Este capítulo explora o uso do SIMULINK na modelagem de sistemas contínuos.

Um sistema contínuo é um sistema dinâmico que pode ser descrito por equações diferenciais. A maioria dos sistemas e processos são contínuos.

Os sistemas mais simples são escalares e pode-se admitir que são lineares e invariantes no tempo. Inicialmente o texto tratará destes tipos de sistemas utilizando blocos da biblioteca linear. A seguir será mostrado modelos mais complicados utilizando vetores de sinais e ainda blocos da biblioteca Nonlinear para se modelar sistemas contínuos não lineares.

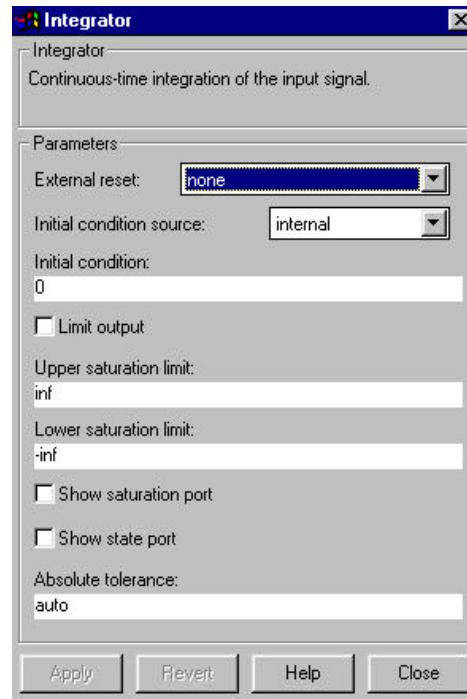
4.1 - Sistemas Escalares Lineares

Podem ser modelados utilizando blocos na biblioteca linear. Estes blocos são fáceis de usar, mas o Integrador tem vital importância na elaboração de diversos sistemas. Primeiro será feita uma explicação detalhada do bloco integrador e a seguir será ilustrada a modelagem de sistemas contínuos escalares com dois exemplos

4.1.1 - Bloco Integrador (Integrator)

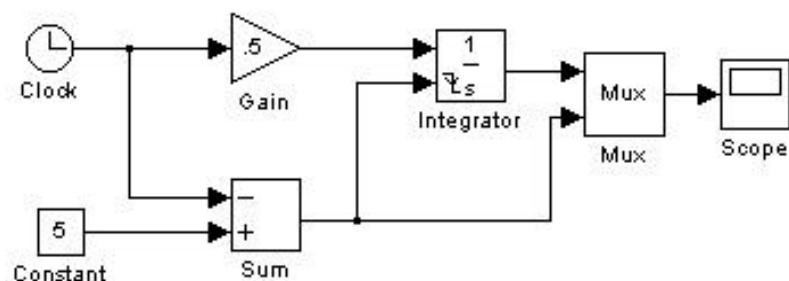
Este bloco pode ser configurado como um integrador simples ou um integrador com reset. Um integrador com reset leva a saída à condição inicial toda vez que seu sinal de reset dispara. Pode-se também configurar o integrador de tal forma que sua saída fique sempre dentro de certos limites pré-definidos. Pode-se ainda configurar o integrador para que sua saída inicial seja um valor configurado na sua caixa de diálogo ou que ele receba o valor inicial através de uma entrada adicional.

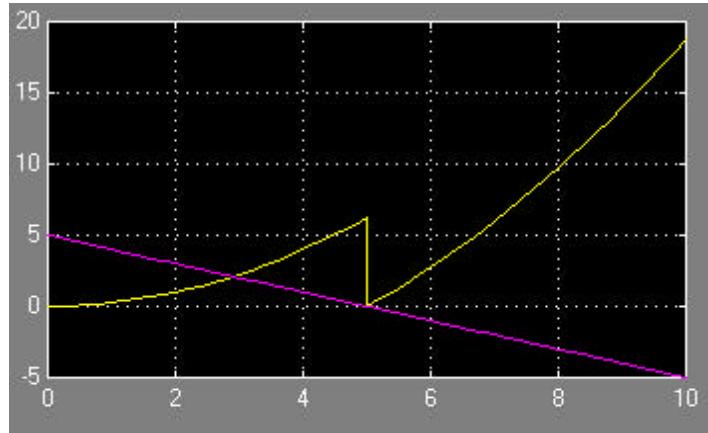
Com um duplo clique no integrador abre-se sua caixa de diálogo. Para se utilizar um integrador padrão a única entrada requerida nesta caixa é a condição inicial (Initial Condition), que tem como default 0.



As opções de reset externo (External reset) estão na caixa de diálogo do bloco integrador. São elas: **None**, **Rising**, **falling** e **either**.

Se a opção escolhida for **None** (default) o reset externo é desabilitado. Se qualquer uma das outras for escolhida o bloco integrador é agora um integrador com reset. Se a opção escolhida for **rising**, a saída do integrador é levada à condição inicial quando o sinal de reset passa pelo zero no sentido crescente. Se a opção for **falling**, a situação se repete, só que agora quando o sinal de reset cruza o zero no sentido descendente. A opção **either** não leva em conta o sentido do sinal de reset. A figura a seguir ilustra um modelo simples de um integrador com sinal de reset configurado para **falling**.





Quando o sinal de reset cruza o zero, a saída do integrador retorna ao seu valor inicial e a simulação continua.

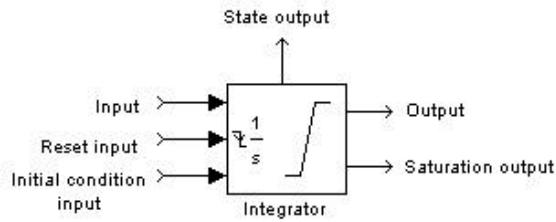
Configurando a Fonte de condição inicial para externa, surgirá uma entrada adicional no integrador. O valor contido neste entrada será utilizado pelo integrador como condição inicial quando a simulação for iniciada ou quando o integrador for resetado.

A opção de limitar a saída (Limit output) faz com que o bloco funcione como um integrador limitado. Nos campos Limite superior de saturação (Upper saturation limit) e Limite inferior de saturação (Lower saturation limit) definem o intervalo no qual o integrador irá funcionar. Os valores default são $-\infty$ e ∞ .

A opção Mostrar porta de saturação (Show saturation port) habilita uma saída adicional no bloco que indica o estado da saturação. Se este sinal for -1 a saída do integrador está abaixo do limite inferior de saturação, se for 1 está acima do limite superior e se for 0 está dentro do intervalo de trabalho definido pelos limites.

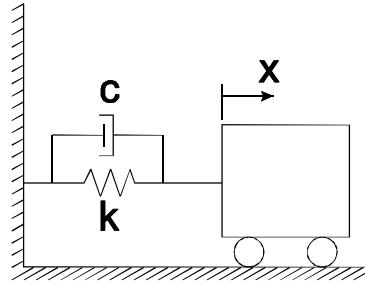
A opção Mostrar porta de estado (Show state port) cria uma saída adicional no bloco. Esta saída contém o estado do integrador que é o mesmo sinal de saída do integrador. Esta saída é necessária em duas ocasiões: se a saída de um bloco integrador realimenta (feedback) o mesmo bloco como reset ou condição inicial, a porta de estado é usada ao invés da saída comum do bloco; e se deseja utilizar a saída do integrador como acionador de um subsistema com execução condicionada a este bloco, deve-se então usar a saída de estado para este acionamento.

O campo Tolerância absoluta permite ao usuário desprezar a configuração de tolerância absoluta definida no menu **Simulation:Parameters**, discutida anteriormente.



Exemplo

Considere o sistema amortecido de segunda ordem



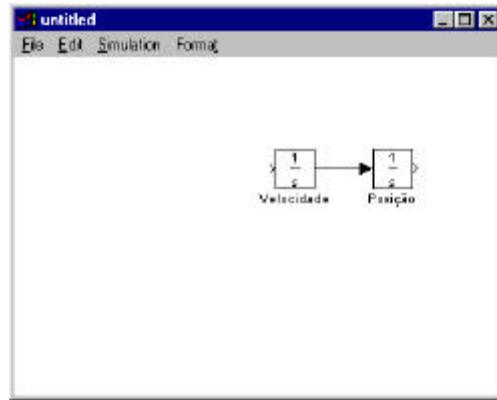
Admite-se que o coeficiente de amortecimento $c=1.0$, a constante da mola $k=2$ N/m e a massa do carro $m=5$ Kg. Não há entradas no sistema. Considerando-se a deflexão inicial igual 1m da posição de equilíbrio.

Para modelar o sistema deve-se primeiro escrever a equação do movimento. Utilizando uma aproximação Newtoniana, nota-se que há somente duas forças agindo no carro: a força da mola e a força de amortecimento. A força da mola é kx e a força de amortecimento é $c\dot{x}$. A força responsável pela aceleração do carro é $m\ddot{x}$. Desde que não há forças externas, a soma destas três forças deve ser zero. Pode-se então escrever a equação:

$$m\ddot{x} + c\dot{x} + kx = 0$$

Sendo um sistema de segunda ordem, necessita-se então de dois integradores para modelar seu comportamento. Abre-se então uma nova janela de modelo para iniciar a construção.

Acrescenta-se dois integradores como na figura, nomeando-os velocidade e posição.



A saída do integrador Velocidade é \dot{x} , já que sua entrada é \ddot{x} . Reescrevendo a equação, tem-se:

$$\ddot{x} = -\frac{c}{m}\dot{x} - \frac{k}{m}x$$

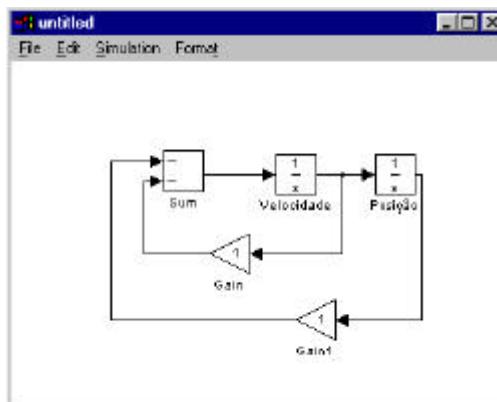
Substituindo os valores:

$$\ddot{x} = -0.2\dot{x} - 0.4x$$

Com $x(0) = 1$, $\dot{x}(0) = 0$.

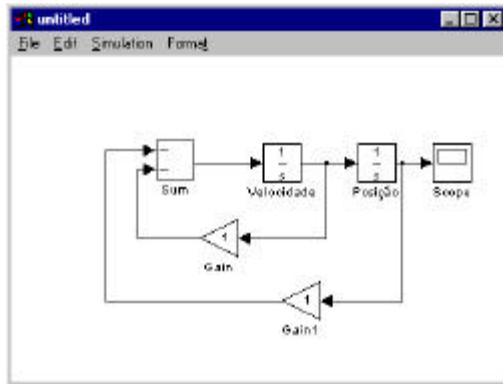
Acrescenta-se agora um bloco de soma para calcular o valor de \ddot{x} , configurando-o com dois sinais de menos (--).

Deve-se agora acrescentar dois blocos de ganho para calcular a taxa em que ocorre o amortecimento e a taxa com que a mola exerce força sobre a massa. É necessário ainda acrescentar as linhas de sinal como mostrado na figura a seguir:



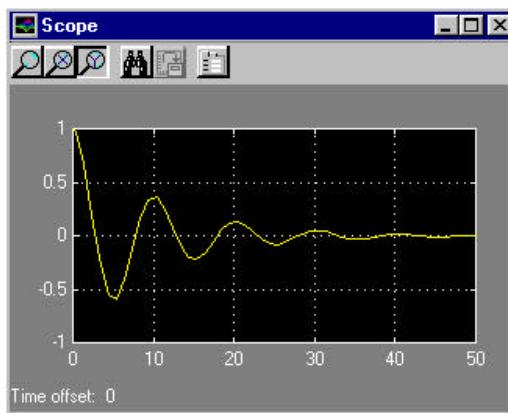
Configura-se a condição inicial do bloco integrador que corresponde à velocidade para 0 e a condição inicial que corresponde à posição da massa para 1.

Falta ainda um dispositivo de saída para que se possa obter a resposta do sistema. Acrescenta-se então um osciloscópio. O modelo final deve ficar da seguinte forma:



Na barra de menu escolhe-se **Simulation:Parameters** e ajusta-se o tempo final (Stop time) para 50. Pode-se agora então dar início à simulação clicando-se **Simulation:Start**.

A resposta pode ser visualizada na tela do osciloscópio.



4.1.2 - Blocos Função de Transferência

A notação de função de transferência é freqüentemente utilizada no projeto e modelagem de sistemas de controle. A *Função de Transferência* pode ser definida como a razão da transformada de Laplace da entrada de um sistema pela transformada de Laplace da saída, considerando as condições iniciais nulas. Desta forma, a função de transferência nos fornece uma descrição conveniente do sistema dinâmico.

O SIMULINK fornece dois blocos para se implementar funções de transferência: O bloco Função de Transferência (Transfer Fcn) e o bloco Zeros-Pólos (Zero-Pole). Os dois blocos são equivalentes, diferindo apenas na notação utilizada para se representar a função de transferência.

O bloco Função de Transferência tem na caixa de diálogo dois campos: Numerador (Numerator) e denominador (Denominator). O numerador contém os coeficientes do numerador da função de transferência em ordem decrescente de potências de s . O denominador contém os coeficientes do denominador de forma semelhante ao numerador.

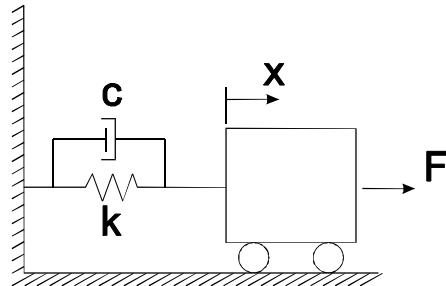
O bloco Zeros-Pólos possuem uma caixa de diálogo que contém três campos: Zeros, Pólos e Ganhos. O campo Zeros contém os zeros do numerador da função de transferência. O campo Pólos contém os zeros do denominador da função e o Ganho ajusta a função. O help do SIMULINK possui uma documentação detalhada dos blocos.

Exemplo

A figura a seguir descreve o mesmo exemplo anterior do sistema massa-mola amortecido com exceção da função força F .

Considere-se que o sistema está inicialmente em equilíbrio estático ($x = 0, \dot{x} = 0$) e a função força é um degrau de 1 N. Ignorando a fricção, a equação do movimento é então:

$$m\ddot{x} + c\dot{x} + kx = F$$



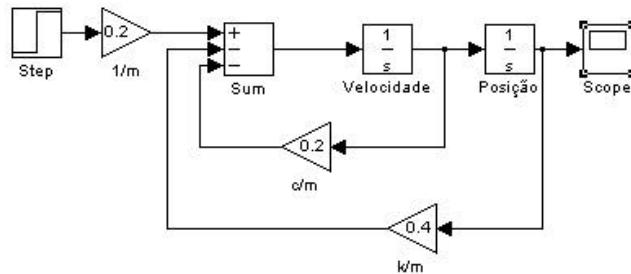
Tomando a transformada de Laplace e ignorando as condições iniciais, tem-se:

$$ms^2 X(s) + csX(s) + kX(s) = F(s)$$

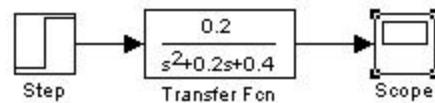
A razão da transformada de Laplace da saída ($X(s)$) pela transformada de Laplace da entrada ($F(s)$) é a função de transferência ($G(s)$):

$$G(s) = \frac{X(s)}{F(s)} = \frac{\frac{1}{m}}{s^2 + \frac{c}{m}s + \frac{k}{m}}.$$

A figura a seguir mostra um modelo SIMULINK deste sistema utilizando blocos lineares primitivos:



Agora temos o mesmo sistema utilizando o bloco função de transferência. O campo Numerador da caixa de diálogo deve conter **[0.2]** e o Denominador **[1 0.2 0.4]**.



4.2 - Vetores em Sistemas Lineares

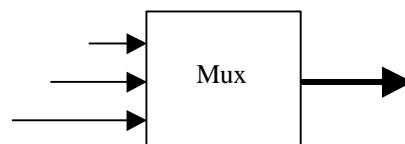
Nos exemplos anteriores as linhas de sinal continham somente sinais escalares. Muitas vezes é conveniente se utilizar sinais vetoriais, pois eles permitem a construção de modelos mais compactos e fáceis de se entender.

Discutiremos alguns mecanismos que utilizam sinais vetoriais e a seguir os utilizaremos com o bloco Espaço de Estados (State-Space).

4.2.1 - Linhas de Sinais Vetoriais

Pode-se combinar vários sinais escalares para se formar um vetor de sinal utilizando o bloco Mux (multiplexador) da biblioteca Connections.

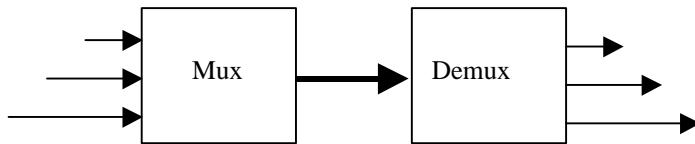
Antes de se utilizar um bloco Mux, deve-se configurar o número de entradas na caixa de diálogo do bloco.



Para facilitar a identificação de linhas de sinais vetoriais deve-se clicar em **Format:Wide Vector Lines** na barra de menu da janela do modelo.

Os componentes do vetor de sinal são $u(1)$, $u(2)$, ..., $u(n)$, onde n é o número de componentes. A entrada superior do bloco é $u(1)$ e a inferior é $u(n)$.

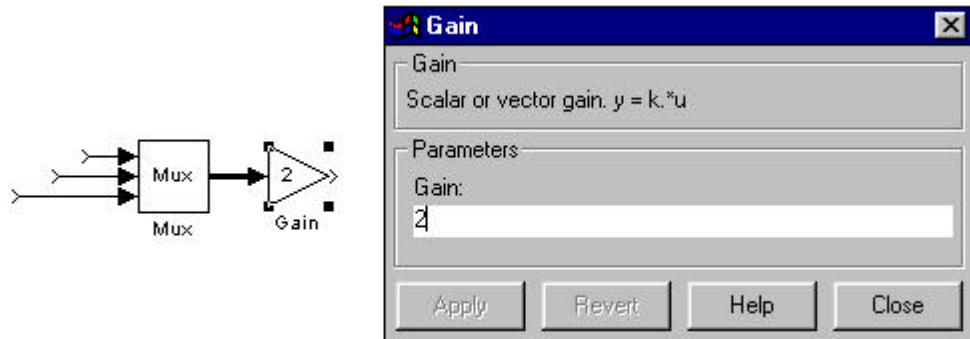
O bloco Demux permite ao usuário separar o vetor de sinal em sinais escalares. Deve ser configurado com o número correto de saídas. A notação do Demux é idêntica à do Mux, $u(1)$ na saída superior e $u(n)$ na saída inferior.



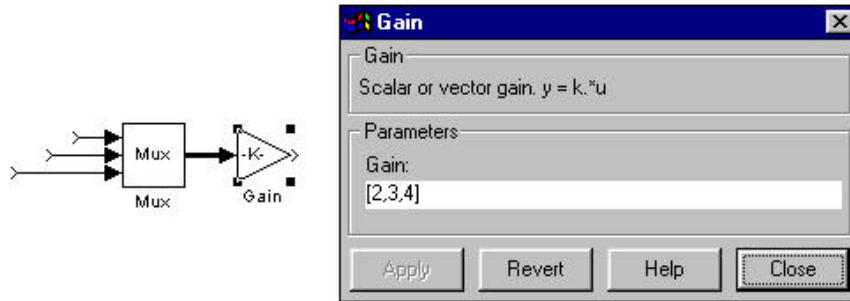
A maioria dos blocos SIMULINK trabalham com entradas vetoriais. O comportamento do bloco com uma entrada vetorial dependerá simultaneamente do tipo de bloco e da configuração do bloco.

Blocos lineares com entradas vetoriais produzem saídas vetoriais na mesma dimensão da entrada. As configurações de blocos lineares com entradas vetoriais devem ter a mesma dimensão da entrada ou ser escalar. No caso de parâmetros escalares, o SIMULINK fará automaticamente a *expansão escalar*, que irá produzir um vetor parâmetro implícito da mesma dimensão da entrada que terá todos os elementos iguais ao valor escalar.

Como exemplo tem-se uma parte de um modelo SIMULINK. A saída do bloco de Ganho é um vetor com 3 elementos no qual cada elemento corresponde ao vetor de entrada multiplicado por 2.



Note agora que para o ganho configurado com o vetor **[2,3,4]**, o primeiro elemento da saída é o primeiro elemento da entrada multiplicado por 2, o segundo elemento da saída é o segundo elemento da entrada multiplicado por 3 e o terceiro elemento é a terceira entrada multiplicada por 4.



A expansão escalar somente se aplica a blocos com entrada escalar. Por exemplo, se a entrada do bloco de ganho do exemplo anterior é um sinal escalar, o primeiro elemento da saída será a entrada multiplicada por 2, o segundo será a entrada multiplicada por 3 e o terceiro multiplicada por 4.

Alguns blocos como o Bloco Função (Fcn block) da biblioteca Nonlinear produzem somente saídas escalares, independente da entrada. O help de cada bloco tem uma explicação detalhada sobre o comportamento com sinais vetoriais.

4.2.2 - Espaço de Estados (State-Space)

Antes de se descrever o bloco Espaço de Estados deve-se discutir o conceito de variáveis de estado. Um *vetor de estado* é um conjunto de *variáveis de estado* suficiente para descrever o estado da dinâmica de um sistema. A forma geral do modelo espaço de estado de um sistema dinâmico é:

$$\dot{x} = f(x, u, t),$$

onde x é o vetor de estado, u o vetor de entrada e t o tempo. A equação é chamada *equação de estado do sistema*. Se define também a saída do sistema:

$$y = g(x, u, t).$$

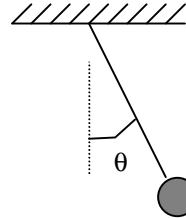
Esta equação é chamada *equação de saída*. Observe-se que se utilizam caracteres minúsculos em negrito para destacar vetores e caracteres maiúsculos em negrito para representar matrizes.

As chamadas *variáveis de estado natural* são quantidades simples como posição, velocidade, temperatura ou corrente elétrica. Para sistemas mecânicos as variáveis de estado natural são posições e velocidades. Para circuitos elétricos podem ser usadas tensões e correntes.

Porém as variáveis de estado natural não são as únicas variáveis de estado disponíveis. Pode-se ainda ter uma combinação linear independente de um grupo de variáveis de estado para se ter novas variáveis de estado válidas.

Exemplo

Considere o pêndulo mostrado:



O estado dinâmico do pêndulo é definido pela sua posição e sua velocidade. Uma escolha natural de variáveis de estado é o ângulo de deflexão e a taxa de mudança de deflexão:

$$x_1 = \theta$$

$$\dot{x}_2 = \dot{\theta}$$

A aproximação das variáveis de estado é uma particularidade muito utilizada na modelagem de sistemas lineares porque se tem a vantagem de utilizar a notação de matrizes para descrever sistemas muito complexos em formatos mais simples. Pode-se ainda calcular a resposta do sistema utilizando aritmética matricial.

Considere o sistema massa-mola amortecedor apresentado num exemplo anterior.

Este é um sistema de segunda ordem com um grau de liberdade, deve-se então escolher duas variáveis de estado. Escolhe-se

$$x_1 = x$$

$$x_2 = \dot{x}$$

As relações entre as duas variáveis de estado são

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}F$$

Estas equações podem também ser escritas na notação matricial:

$$\dot{x} = Ax + Bu$$



Onde

$$\begin{aligned}\mathbf{x} &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ \mathbf{A} &= \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \\ u &= F\end{aligned}$$

A matriz \mathbf{A} é freqüentemente chamada *matriz de sistema*. A matriz de sistema é sempre quadrada. A matriz \mathbf{B} é a *matriz de entrada*. O número de linhas na matriz de entrada é o número de variáveis de estado, e o número de colunas é o número de entradas. A equação que envolve as matrizes acima é a equação de estado do sistema para um sistema linear. Note que as matrizes de sistema e de entrada não são as características únicas do sistema. Diferentes escolhas de variáveis de estado resultam em diferentes matrizes de estado e de entrada.

Para se definir as variáveis de entrada deve-se considerar os estados internos de um sistema. As variáveis de estado não são necessariamente as saídas do sistema. As saídas podem consistir num subconjunto de estados, ou podem consistir numa combinação linear dos estados do sistema e suas entradas. A equação de saída para um sistema linear é:

$$y = \mathbf{C}\mathbf{x} + \mathbf{D}u$$

Se a escolha da saída do sistema como sendo a posição da massa (x), tem-se:

$$\begin{aligned}y &= x_1 \\ \mathbf{C} &= [1 \ 0] \\ \mathbf{D} &= 0\end{aligned}$$

\mathbf{C} é chamada *matriz de saída*. \mathbf{D} é chamada *matriz de transmissão direta* porque se ela não for nula, a entrada será diretamente transmitida para a saída.

4.2.3 - Bloco Espaço de Estados (State-Space block)

Este bloco implementa um modelo linear de um sistema ou uma parte de um sistema no espaço de estados. A caixa de diálogo do bloco contém campos para cada uma das quatro matrizes de espaço de estados (\mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D}) e ainda um quinto campo para condições iniciais. Cada campo deve conter uma matriz no formato MATLAB.

Exemplo

Considere novamente o sistema massa mola comentado anteriormente. Deseja-se agora conhecer a resposta do sistema ao impulso. Substituindo os parâmetros do modelo nas equações, temos:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -0.4x_1 - 0.2x_2 + 0.2d(t),\end{aligned}$$

onde $d(t)$ é a função impulso unitário. A matriz de sistema é

$$A = \begin{bmatrix} 0 & 1 \\ -0.4 & -0.2 \end{bmatrix},$$

e a matriz de entrada é

$$B = \begin{bmatrix} 0 \\ 0.2 \end{bmatrix}.$$

Define-se como saída a posição do bloco,

$$C = [1 \ 0]$$

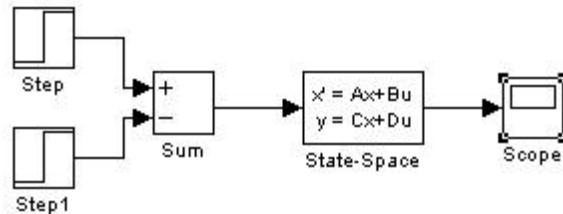
Neste caso não há transmissão direta, então:

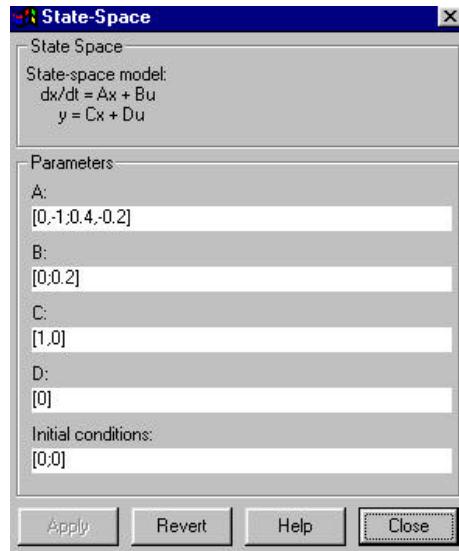
$$D = 0.$$

Pode-se aproximar o impulso unitário como sendo uma função degrau positiva seguida de uma função degrau negativa:

$$d(t) \approx 100u(t) - 100u(t - 0.01).$$

A figura a seguir mostra o modelo SIMULINK desta formulação de equações de movimento e a caixa de diálogo do bloco Espaço de Estados. Note que deve-se incluir condições iniciais para cada variável de estado.



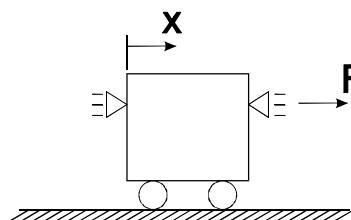


4.3 - Modelando Sistemas Não Lineares

O SIMULINK fornece uma variedade de blocos para a modelagem de sistemas não lineares. Esses blocos estão na biblioteca Nonlinear. O comportamento destes blocos não lineares com respeito a entradas vetoriais pode variar. Alguns blocos produzem saídas vetoriais na mesma dimensão da entrada. Outros blocos produzem somente saídas escalares, ou alternando entre escalares ou vetoriais, dependendo da dimensão das entradas. Uma explicação detalhada do comportamento de cada bloco está no help do SIMULINK.

Exemplo

Para ilustrar o uso de diversos blocos da biblioteca Nonlinear, será mostrado a modelagem do movimento de um carro acionado por foguetes, como mostrado na figura:



O carro é impulsionado por dois foguetes em oposição. O controlador aciona o motor esquerdo se a soma da velocidade e da posição do carro for negativa, e acionar o motor direito se a soma for positiva. O objetivo do controlador é levar o carro ao repouso na origem. Este tipo de controle é chamado *Controle Bang-Bang*.

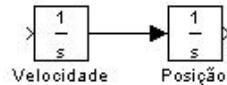
A massa do carro é 5 Kg e a força motor é 1 N.

A princípio, a equação do movimento do carro é:

$$m\ddot{x} = F .$$

Este é um sistema de segunda ordem, então dois integradores serão necessários para se calcular a posição do carro.

Abre-se uma nova janela de modelo e acrescenta-se a ela dois integradores da biblioteca Linear. Os integradores recebem os nomes *Velocidade* e *Posição*, como mostrado:



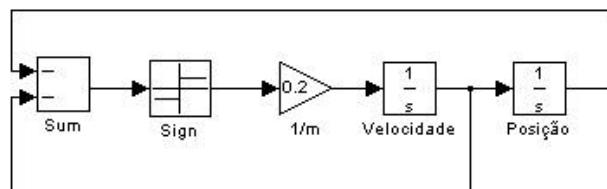
A entrada do primeiro integrador é a aceleração. Resolvendo a equação do movimento para a aceleração, tem-se:

$$\ddot{x} = \frac{F}{m}$$

Acrescentam-se agora um bloco Ganho que multiplica a força motora do foguete por $\frac{1}{m}$. A entrada do bloco Ganho será a força motora F.

A força motora F é 1.0 se a soma da velocidade e posição for negativa e -1.0 se a soma for positiva. Pode-se construir um conveniente *Controlador Bang-Bang* utilizando um bloco de Soma (Sum) da biblioteca Linear e um Bloco Sign da biblioteca Nonlinear. A saída de um bloco Sign é 1.0 se a entrada for positiva, -1.0 se a entrada for negativa e 0.0 se a entrada for exatamente 0.

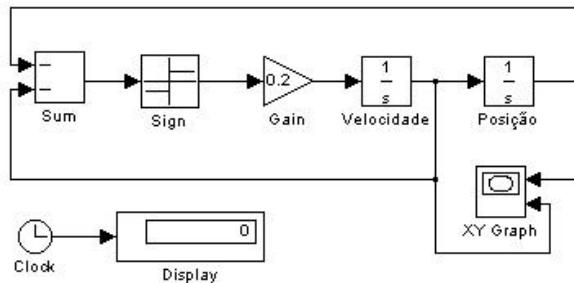
Acrescenta-se então os blocos Sign e de Soma como mostrado. O bloco de Soma é configurado com dois sinais de menos (- -) porque a força motora é oposta ao sinal da soma da velocidade com a posição.



Para visualizar o resultado da simulação utiliza-se um gráfico XY (XY Plot) para se plotar um Gráfico no Plano de Fase (Phase Plot) com o progresso da simulação. Um Gráfico no Plano de Fase é um gráfico da *velocidade x posição*.

Acrescenta-se então um Gráfico XY. Conecta-se a saída do integrador *Posição* à entrada X (superior) e a saída do integrador *Velocidade* à entrada Y (inferior).

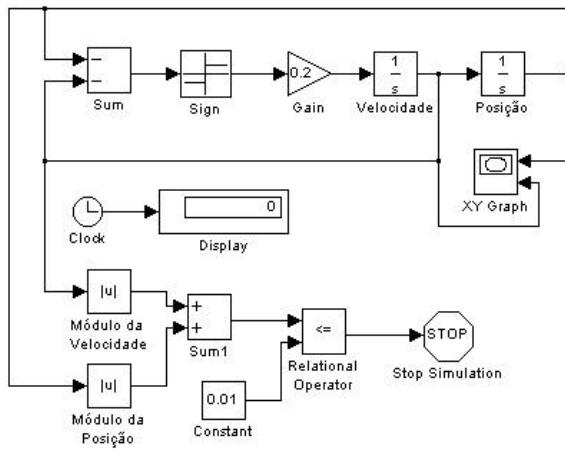
Para monitorar o tempo no progresso da simulação, acrescenta-se um bloco Clock da biblioteca de fontes (Sources) e um bloco Display da biblioteca de dispositivos de saída (Sinks) e conectam-se os dois conforme a figura:



Não se sabe quanto tempo a solução demora para atingir a origem, então por conveniência, adiciona-se uma lógica ao modelo que pare a simulação quando o objetivo for atingido. Esta lógica irá parar a simulação quando a soma dos valores absolutos da velocidade e da posição cair a um valor menor que 0.01. Isto pode ser feito com êxito adicionando um bloco de Parada da Simulação (Stop Block) da biblioteca de dispositivos de saída (Sinks). Este bloco força o SIMULINK parar a simulação quando sua entrada for diferente de zero. Deseja-se que a entrada do bloco seja zero até que

$$|x| + |\dot{x}| \leq 0.01.$$

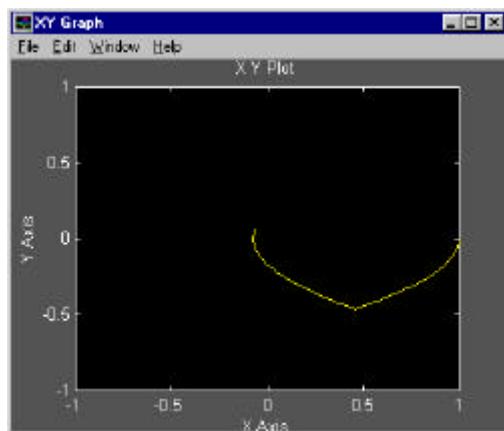
Acrescentam-se então dois blocos de Módulo (Abs) da biblioteca Nonlinear e um bloco de soma da biblioteca Linear ao modelo. A seguir acrescenta-se um bloco Operador Relacional (Relational Operator) e escolhe-se \leq na lista **Operator**. Insere-se agora um bloco Constante (Constant) da biblioteca de fontes e configura-se seu valor para 0.01. Finalmente se acrescenta um bloco de Parada (Stop Simulation) e conecta-se as linhas de sinal como mostrado na figura.



Supondo que no início da simulação o carro estava em repouso afastado de 1 N para a direita. Então a condição inicial do integrador Velocidade é 0 e a do integrador Posição é 1.

O bloco Sign chaveia a saída instantaneamente de acordo com a entrada. Contudo qualquer sistema físico de chaveamento realizável leva um tempo finito para mudar a saída. Para modelar tal comportamento pode-se utilizar um algoritmo de tamanho de passo fixo (fixed-step-size). Vamos supor que o tempo de chaveamento seja 0.05 segundos, clica-se então em **Simulation:Parameters** na barra de menu e na opção **Type** escolhe-se Passo fixo (Fixed-step) e ODE5 (Dormain-Prince). A seguir configura-se o tamanho do passo para 0.05 e o tempo final (Stop time) 200.

A simulação agora está pronta para ser executada. Clica-se então em **Simulation:Start** e a simulação será executada até o carro atingir o repouso na origem. O gráfico XY deve ser semelhante à figura:



Ao se olhar com atenção nota-se que a trajetória da fase se aproxima da origem mas oscila na linha $\dot{x} = -x$. Este fenômeno é chamado *reticência*.

Tente substituir o bloco Sign por um bloco de Saturação (Saturation) configurando o limite superior (Upper limit) para 0.05 e o limite inferior (Lower limit) para -0.05. Tente também substituir o bloco Sign por um bloco de Zona Morta (Dead Zone),

com o início (Start of dead zone) em -0.05 e fim (End of dead zone) em 0.05. Ao se utilizar qualquer um destes dois blocos no lugar do bloco Sign pode-se então configurar a simulação para um método de passo variável já que nenhum bloco do sistema muda instantaneamente de -1 para +1.

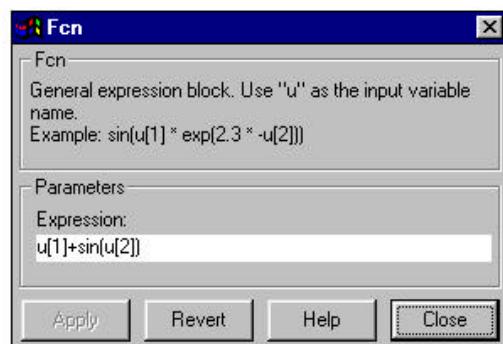
4.3.1 - Blocos de Função (Function Blocks)

Duas utilidades particulares de blocos não lineares são os blocos Fcn (C function) e MATLAB Fcn. Ambos executam operações matemáticas especificadas na entrada, mas os blocos tem algumas diferenças importantes.

O SIMULINK executa o bloco Fcn muito mais rápido do que o MATLAB Fcn. Porém o bloco Fcn não aceita operações matriciais. Devido à vantagem na velocidade, é sempre aconselhável utilizar o bloco Fcn quando as capacidades especiais do bloco MATLAB Fcn não forem necessárias.

4.3.1.1 - Bloco Fcn

O bloco Fcn é mostrado na figura a seguir:



A caixa de diálogo contém um único campo que deve conter uma expressão na sintaxe da linguagem C. A expressão é então executada nos elementos do vetor de entrada do bloco $u[n]$, onde n é a posição do elemento desejado.

Vale lembrar que se devem utilizar os colchetes ("[" e "]") no lugar dos parênteses ("(" e ")"') utilizados no MATLAB.

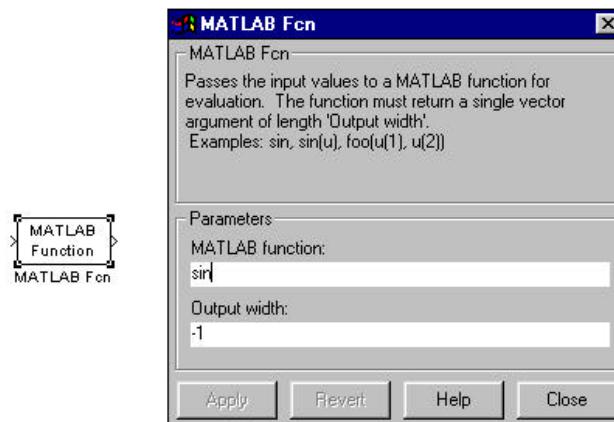
O bloco pode ainda utilizar como parâmetro qualquer variável previamente definida na área de trabalho do MATLAB. Se a variável for um escalar, ela deve então ser referida pelo seu nome. Por exemplo, se há uma variável escalar "a" e um vetor com duas entradas no bloco Fcn, uma expressão válida poderia ser $a*(sin(u[1])+u[2])$. Se a variável da área de trabalho for um vetor ou matriz, deve-se então utilizar para esta variável a sintaxe adequada ao MATLAB para se referir a esta variável como $a(1)$ ou $a(2,3)$.

O bloco Fcn pode executar todas as funções matemáticas escalares padrões como *sin*, *abs*, *tan* etc, operações relacionais com a sintaxe da linguagem C ($==$, $!=$, $>$, $<$, \geq , \leq) e as operações lógicas ainda na sintaxe C ($\&\&$ - AND lógico e $\|$ - OR lógico). O bloco Fcn produz sempre uma saída escalar.

4.3.1.2 - Bloco MATLAB Fcn

Este bloco é mais poderoso do que o anterior pois pode executar operações matriciais e produzir saídas vetoriais. Em contrapartida é muito mais lento do que o bloco Fcn, sendo então usado somente quando o bloco Fcn não for adequado.

O bloco MATLAB Fcn é mostrado na figura que se segue:

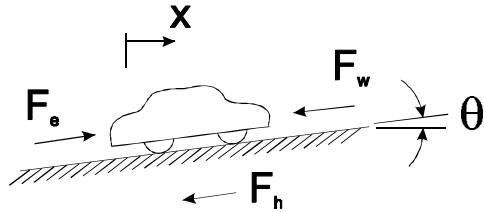


A caixa de diálogo contém dois campos. O primeiro deve conter uma expressão válida com a sintaxe MATLAB. O valor da expressão é a saída do bloco. O enésimo elemento no vetor de entrada do bloco é $u(n)$, similar ao bloco Fcn. Se a função MATLAB não tiver argumentos como a da figura, a operação é executada em todos os elementos de entrada. O segundo campo especifica a dimensão do vetor de saída. Entre com -1 se deseja que a saída tenha o mesmo tamanho da entrada. Independentemente de se especificar explicitamente a dimensão da saída ou de se utilizar o valor default, a dimensão do vetor de saída com o resultado da expressão no campo MATLAB *function* deve ser a mesma especificada em *Output Width*.

O exemplo a seguir ilustra o uso de blocos não lineares na construção de um modelo simples de um carro e um controle de ganho proporcional. O modelo inclui arraste aerodinâmico, a força gravitacional na subida de rampas e ainda o vento.

Exemplo

Considere o deslocamento do automóvel numa pista montanhosa como mostrado na figura que segue:



Há três forças agindo no carro: a força motriz produzida pelo motor do carro e transmitida aos pneus (ou força de frenagem se esta for negativa) (F_e), a força aerodinâmica (incluindo o vento) (F_w) e a componente tangencial da gravidade quando o carro sobe ou desce as rampas (F_h). Aplicando a segunda lei de Newton, a equação do movimento do automóvel pode ser escrita da seguinte forma:

$$m\ddot{x} = F_e - F_w - F_h$$

onde m representa a massa do automóvel e x a distância percorrida. F_e deve ter limites máximos e mínimos. O limite máximo é a máxima força que o motor pode transmitir pelas rodas para a pista e o limite mínimo é a máxima força de freio. Então $-2000 \leq F_e \geq 1000$, sendo a massa do carro 100 N.

A força aerodinâmica é o produto do coeficiente de arraste (C_D), a área frontal do automóvel (A) e a pressão dinâmica (P), onde

$$P = \frac{rV^2}{2}$$

e r representa a densidade do ar e V a soma da velocidade do automóvel e do vento (V_w). Supondo que

$$\frac{C_D A r}{2} = 0.001$$

e que a velocidade do vento varia senoidalmente de acordo com a função

$$V_w = 20 \sin(0.01t)$$

então a força aerodinâmica pode ser aproximada para

$$F_w = 0.001(\dot{x} + 20 \sin(0.01t))^2.$$

Supondo ainda que o ângulo da pista varie senoidalmente com a distância de acordo com a função

$$q = 0.0093 \sin(0.001x).$$

A componente tangencial da gravidade será então

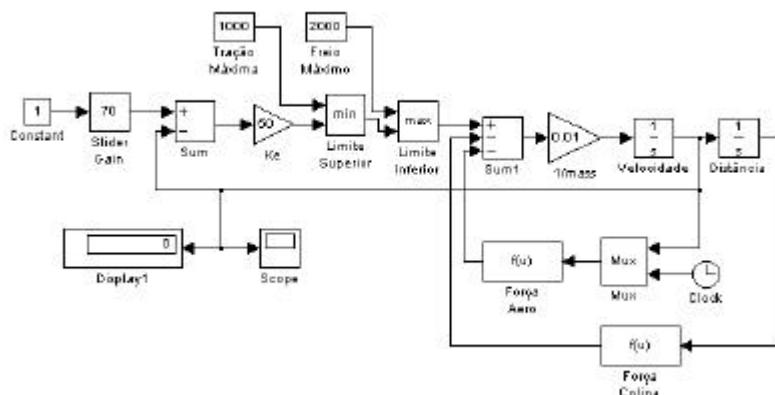
$$F_h = 30 \operatorname{sen}(0.0001x).$$

O controlador utilizado será do tipo controle proporcional com a seguinte lei de controle

$$F_c = K_e (\dot{x}_{DESEJADO} - \dot{x})$$

F_c é a força do motor comandada de tração ou freio, $\dot{x}_{DESEJADO}$ é a velocidade controlada (ft/s) e K_e é o ganho de realimentação (feedback). Segundo esta lei, a força comandada do motor será proporcional ao erro de velocidade. A força atual do motor (F_e), como dito antes, é limitada pela força máxima de tração e força máxima de frenagem. O ganho de realimentação utilizado será $K_e = 50$.

Um modelo SIMULINK para este sistema é mostrado na figura que se segue:



A entrada do controlador proporcional é a velocidade desejada do automóvel em m/s. Este controlador foi implementado com um bloco de ganho do tipo Slider com a entrada ligada à um valor constante. Este bloco permite que se modifique o ganho no decorrer da simulação.

O controlador proporcional consiste de um bloco de soma (Sum) que calcula o erro da velocidade (diferença entre a velocidade comandada e a atual) e um bloco de ganho.

Os limites superior e inferior da força motora são impostos utilizando blocos MinMax. As constantes de nome Max Tração e Max Freio junto com os blocos MinMax são utilizados para se ilustrar o uso destes blocos. Esta parte do modelo poderia ser substituída por um bloco de Saturação (Saturation) da biblioteca Nonlinear. (Por que o aluno não tenta fazê-lo?)

As forças não lineares devidas à rampa e à aerodinâmica são calculadas pelos blocos Fcn. O campo **Expression** da caixa de diálogo do bloco de nome Força Aero



Curso de Introdução ao SIMULINK

contém $0.001*(u[1]+20*\sin(0.01*u[2]))^2$. Para o bloco da força devido ao acidente deve ser $30*\sin(0.0001*u[1])$.

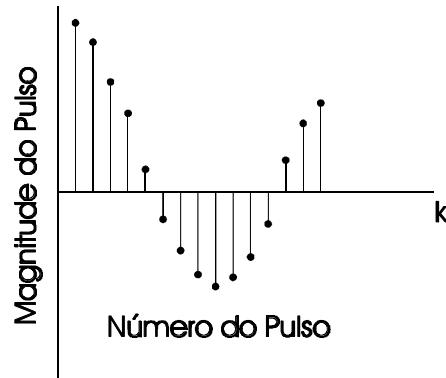
O bloco Display serve como um velocímetro (em m/s) e a velocidade é mostrada graficamente no osciloscópio.

Este modelo é um bom exemplo de um sistema levemente rígido. Para se observar o efeito desta rigidez, a simulação deve ser executada utilizando ODE45 e a seguir deve ser repetida utilizando ODE15S.

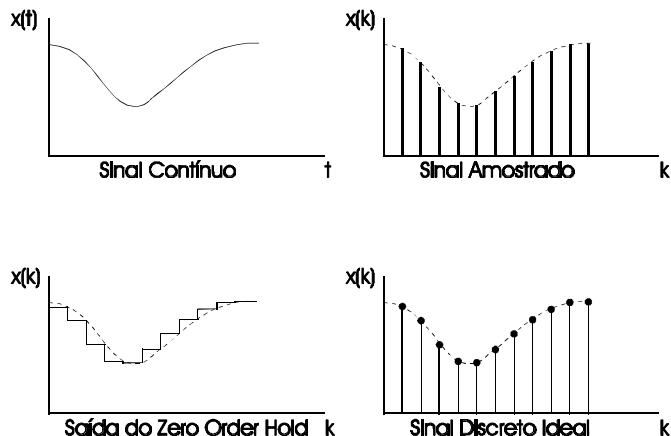
Capítulo 5 – Sistemas Discretos no Tempo

5.1 - Visão Geral

Um sistema discreto é um sistema que pode ser representado utilizando equações de diferença e que opera com sinais discretos. Um sinal discreto pode ser representado como uma seqüência de pulsos, como na figura.



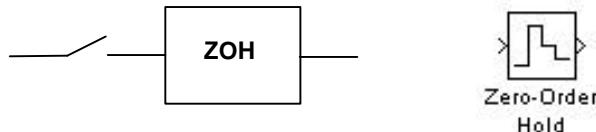
Um sistema discreto tem como entrada um ou mais sinais discretos e produz um ou mais sinais discretos como saída. Um sistema discreto no tempo é um sistema discreto que só permite mudanças em instantes específicos de tempo. Na maioria dos sistemas de controle discreto os sinais não são inherentemente discretos. Nesses sistemas, os sinais discretos são extraídos de sinais contínuos por um processo conhecido como *amostragem*. A figura a seguir ilustra o processo de amostragem:



Este tipo de amostragem mostrado na figura anterior utiliza dois dispositivos: um amostrador e um extrapolador de ordem zero (*zero-order hold*).

O amostrador fecha uma chave periodicamente por um instante. Cada vez que a chave é fechada, um pulso de duração muito curta (teoricamente zero) e de magnitude igual ao sinal de entrada é produzido. O espaçamento entre os pulsos é o período de amostragem (T).

O extrapolador de ordem zero recebe este sinal e mantém na saída o último valor de sua entrada, produzindo um sinal em degraus. Um controlador discreto requer uma seqüência de números na entrada. Um conversor A/D (análogo – digital) é um dispositivo que converte o sinal em degraus da saída do extrapolador de ordem zero para uma seqüência de números, representada pela seqüência de pulsos representada na figura. A combinação do amostrador e extrapolador de ordem zero é mostrada na figura que segue com o bloco SIMULINK equivalente.



O texto se refere a sinais que variam com o tempo utilizando a notação $x(t)$. A notação correspondente para sinais discretos é $x(k)$, onde k é o número ordinal do pulso. O mapeamento do espaço amostrado para o tempo contínuo é

$$x(t) = x(kT),$$

onde T é o período amostral (sampling).

5.2 - Sistemas Discretos no Tempo Lineares Escalares

Modelar este tipo de sistemas é muito similar a modelar sistemas contínuos. Modelos discreto no tempo podem usar blocos de ganho e soma da biblioteca linear. Estes blocos se comportam da mesma maneira em sistemas discretos como em sistemas contínuos. A biblioteca Discreta (Discrete) contém os blocos análogos ao integrador e funções de transferência contínuos.

Em cada bloco discreto admite-se que há um amostrador na entrada e um extrapolador de ordem zero na saída. Blocos discretos possuem o parâmetro adicional de configuração **Sample time**. **Sample time** pode ser tanto um valor escalar que será o intervalo entre as amostras quanto um vetor com dois elementos que definem o intervalo entre as amostras e um deslocamento (offset or time skew). Por exemplo, se o bloco tiver um intervalo de amostragem de 1.5 segundos e nenhum **offset**, **Sample time** deve ser configurado para 1.5. Se o tempo de amostragem for 0.75 segundos com 0.25 segundos de **offset**, o campo **Sample time** deve conter [0.75, 0.25].

Todas os algoritmos de solução mostrados em **Simulation:Parameters Solver options** são compatíveis com sistemas discretos. Uma opção especial de nome “discreto (sem estados contínuos)” é a melhor escolha para sistemas puramente discretos, pois é otimizada para este tipo de sistema.

5.2.1 - Atraso unitário (Unit Delay)

Este é um bloco discreto no tempo fundamental. As vezes é chamado registrador de deslocamento (shift register) ou elemento de atraso de tempo (time-delay element). A saída do bloco atraso unitário é a entrada no instante de amostragem anterior. O atraso unitário representa a equação diferença:

$$y(k) = x(k-1)$$

onde y é a seqüência de saída e x é a seqüência de entrada. A caixa de diálogo do bloco atraso unitário contém dois campos. O primeiro é a Condição Inicial (Initial Condition). Este campo deve conter o valor da saída do bloco no início da simulação. O segundo campo é o **Sample time**.

Exemplo

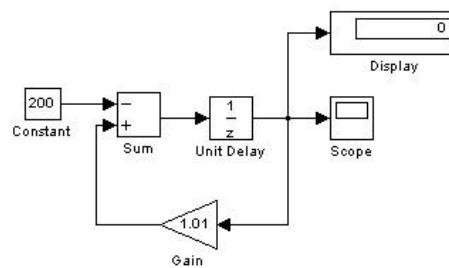
Neste exemplo será modelada a amortização do financiamento de um automóvel. No fim de cada mês, o balanço do empréstimo é igual à soma do balanço no início do mês mais os juros do mês menos o pagamento do mês. Se o balanço do fim do mês for representado por $b(k)$, o balanço será então, no final do mês k

$$b(k) = rb(k-1) - p(k)$$

onde $r = 1 + i$ e i é a taxa de juros mensal.

Sabe-se que o valor inicial do financiamento é R\$15.000,00, a taxa de juros é 1% ao mês (12% ao ano), e a mensalidade é R\$200,00. Calcule o valor do financiamento para 100 pagamentos.

A figura a seguir mostra o modelo SIMULINK para este sistema.



O bloco atraso unitário calcula $b(k-1)$. A condição inicial deste bloco é o valor total do financiamento (15.000). O **Sample time** do bloco atraso unitário deve ser 1.

Para este sistema pode-se utilizar o método de passo fixo Discreto (Sem estados contínuos), o tempo inicial deve ser 0 e o tempo final 100. Depois da simulação executada, o bloco Display irá mostrar o valor total do financiamento.

5.2.2 - Integrador no Tempo Discreto (Discrete-Time Integrator)

A saída deste bloco é uma aproximação da integral no tempo do sinal de entrada. É uma aproximação discreta do integrador contínuo. A saída do Integrador no Tempo Discreto é aproximadamente

$$y(k) = y(k-1) + \int_{T(k-1)}^{Tk} u(t) dt$$

onde $u(t)$ é a entrada do integrador, $y(k)$ é a saída e T é o período de amostragem (Sample Period).

Os campos da sua caixa de diálogo são os mesmos do integrador contínuo, com mais dois campos adicionais. Os dois campos adicionais são Método de Integração (Integrator method) e período de amostragem (Sample time).

O campo Método de Integração possui três opções: Euler adiantado (Forward Euler), Euler atrasado (Backward Euler) e Trapezoidal. Sendo este integrador um bloco discreto, ele possui então um amostrador e um extrapolador de ordem zero como entrada. A cada passo de integração, ele então tem acesso a somente dois valores de $u(t)$: $u(Tk)$ na entrada e $u(T(k-1))$ gerado por um atrasador unitário interno. Cada uma das três opções nos métodos de integração aproxima $u(t)$ de maneira diferente.

5.2.2.1 - Integração Trapezoidal

A integração trapezoidal é baseada na aproximação

$$u(t) = \frac{u(Tk) + u(T(k-1))}{2}$$

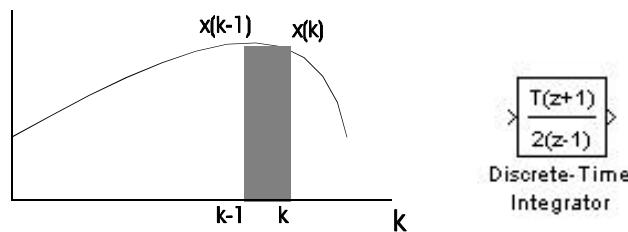
A integração trapezoidal aproxima então

$$y(k) = y(k-1) + \frac{Tu(Tk) + Tu(T(k-1))}{2}.$$

A função de transferência Z correspondente é

$$\frac{Y(z)}{U(z)} = \frac{T(z+1)}{2(z-1)}.$$

Isto está retratado a seguir



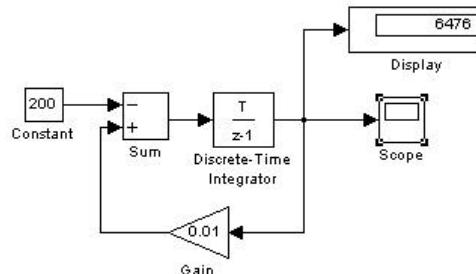
Exemplo

Considere o financiamento do automóvel descrito no exemplo anterior. O balanço do financiamento pode ser calculado por

$$b(k) = b(k-1) + \int_{T(k-1)}^{Tk} (ib(k-1) - p) dt$$

Examinando o integrando, pode-se ver que o problema pode ser resolvido utilizando a Integração de Euler adiantado (Forward Euler Integration). A figura mostra o

modelo revisado para se utilizar o integrador no tempo discreto com o método de integração de Euler adiantado.



A condição inicial do integrador deve ser configurada para 15.000 e o período de amostragem para 1. Note que neste caso, o ganho de retroação é 0.01.

5.2.3 - Blocos de Função de Transferência Discreta

Uma função de transferência discreta é análoga à função de transferência contínua. Esta função é definida como a razão da transformada Z do sinal de entrada de um sistema pela transformada Z do sinal de saída deste sistema. A biblioteca Discreta fornece 3 blocos que implementam funções de transferência discreta: Filtro Discreto (Discrete Filter), Função de Transferência Discreta (Discrete Transfer Fcn) e Zeros-Pólos Discretos (Discrete Zero-Pole). Estes blocos são equivalentes, diferindo apenas nas definições dos coeficientes dos polinômios do numerador e do denominador.

O bloco Filtro Discreto requer vetores de coeficientes dos polinômios em potências ascendentes de z^{-1} . O bloco Função de Transferência Discreta (Discrete Transfer Fcn) requer vetores com os coeficientes dos polinômios em potências descendentes de z . Os dois blocos são idênticos, diferindo somente na maneira de se representar a função de transferência no ícone do bloco. O bloco Pólos-Zeros Discreto requerem vetores de zeros (numerador) e pólos (denominador) da função de transferência e o ganho que dá dimensão da função de transferência.

Exemplo

Sistemas de controle freqüentemente contêm filtros para remover ruídos em altas freqüências dos sinais de entrada. O MATLAB possui uma Toolbox de processamento de sinais que fornece uma grande variedade de algoritmos de projetos de filtros. O usuário pode incluir um filtro projetado no MATLAB num modelo SIMULINK utilizando um bloco Função de Transferência Discreta (Discrete Transfer Fcn).

Suponha que se deseja projetar um sistema de controle com um ruído senoidal na entrada e precisa-se filtrar este ruído.

O período de amostragem (Sampling Period) é 0.1 segundo e se deseja remover sinais com uma freqüência superior a 0.2 Hz.

A ordem de um filtro é, por definição, o número de pólos existentes na sua função de transferência. É importante frisar que quanto maior for a ordem de um filtro mais a sua resposta se aproximará das curvas ideais.

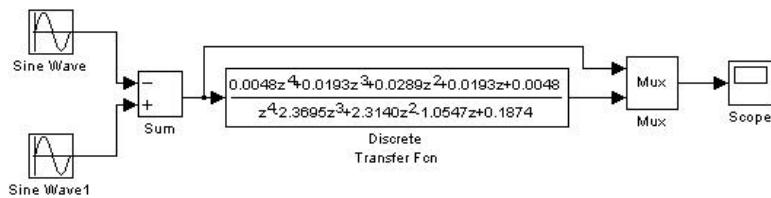
A resposta Butterworth também é denominada “resposta plana”. Esta denominação se deve ao fato de que as curvas obtidas não possuem nenhum tipo de ondulação (ripple). A resposta plana máxima ocorre nas proximidades do ponto $f=0$ Hz. Para o caso desejado, um filtro Butterworth de quarta-ordem é suficiente. O comando MATLAB para o projeto é *butter*. As seguintes linhas de comando na área de trabalho do MATLAB fornecem os coeficientes da função de transferência do filtro Butterworth de quarta-ordem com frequência de corte em 0.2 Hz.

```
>> [B,A]=butter(4,0.2)

B =
    0.0048 0.0193 0.0289 0.0193 0.0048

A =
    1.0000 -2.3695 2.3140 -1.0547 0.1874
```

O filtro pode ser testado com o modelo SIMULINK mostrado na figura:



O bloco de onda senoidal superior é configurado para uma freqüência de 0.5 rad/s e uma amplitude de 1. O bloco inferior tem freqüência 10 rad/s e amplitude 0.4 e representa o ruído de alta freqüência indesejado.

O campo Numerador da caixa de diálogo do bloco Função de Transferência deve conter o vetor $[0.0048 \ 0.0193 \ 0.0289 \ 0.0193 \ 0.0048]$ e o denominador $[1.0000 \ -2.3695 \ 2.3140 \ -1.0547 \ 0.1874]$. Sample time pode ser 0.1. O modelo pode ser configurado para um método de resolução discreto, já que não há nenhum estado contínuo. O tempo final pode ser 20 para visualização total da onda. O bloco Mux cria um vetor contendo o sinal antes e depois da filtragem. Com a opção Salvar dado para a área de trabalho (Save data to workspace) assinalada no osciloscópio os sinais exibidos na tela do mesmo podem ser plotados de maneira mais legível com o comando *plot* do MATLAB. Os comandos necessários para isto são listados a seguir:

```
>> t = ScopeData(:,1);
>> y_raw = ScopeData(:,2);
>> y_filt = ScopeData(:,3);
>> subplot(2,1,1)
```

```

>> plot(t,y_raw);

>> title('Sinal sem filtragem')

>> grid

>> subplot(2,1,2)

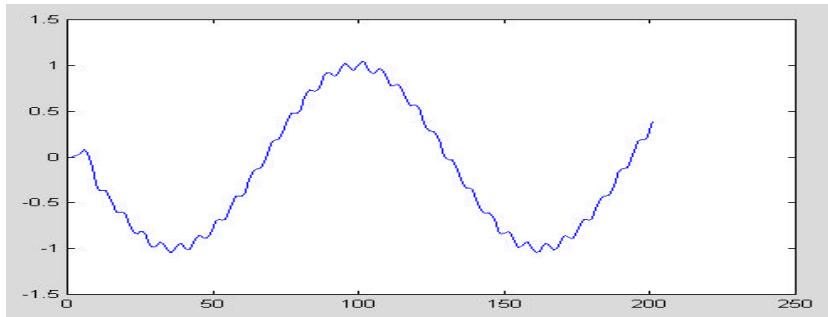
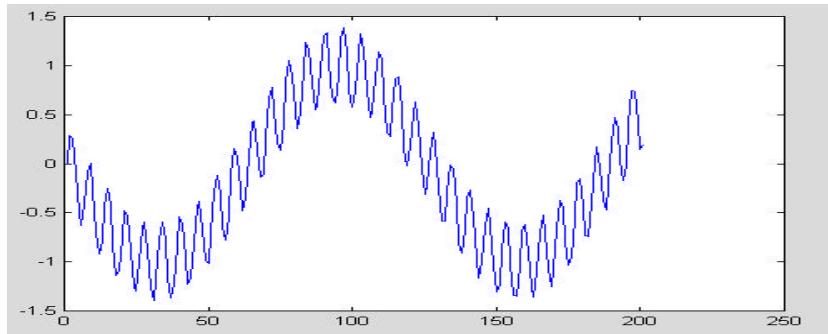
>> plot(t,y_filt);

>> title('Sinal filtrado')

>> xlabel('Tempo (s)')

>> grid

```



5.3 - Blocos Lógicos

Os blocos Operador Lógico e Lógica Combinacional são encontrados na biblioteca Nonlinear e são muito utilizados na modelagem de sistemas discretos. Não possuem amostradores na entrada e nem extrapolador de ordem zeros na saída. Apesar de poderem trabalhar com modelos contínuos, são mais comumente utilizados em modelos discretos no tempo.

O bloco operador lógico pode ser configurado para executar qualquer operação lógica: E (AND), OU (OR), NÃO (NAND), NÃO OU (NOR), OU EXCLUSIVO (XOR) ou NÃO(NOT). Este bloco pode ser configurado para ter qualquer número de entradas. Se os sinais de entrada forem vetores, todas as entradas devem ter o mesmo tamanho e a saída também será um vetor.

O bloco Lógica Combinacional implementa uma tabela verdade. A entrada deste bloco é um vetor de n elementos. A tabela verdade deve ter 2^n linhas, arranjadas de modo que os valores de uma linha de entrada forneça um índice à tabela. Então se houver somente uma entrada, deve haver então duas linhas, a primeira correspondendo à entrada de 0 e a segunda à entrada de 1. Se houver duas entradas, deve existir então quatro linhas na tabela.

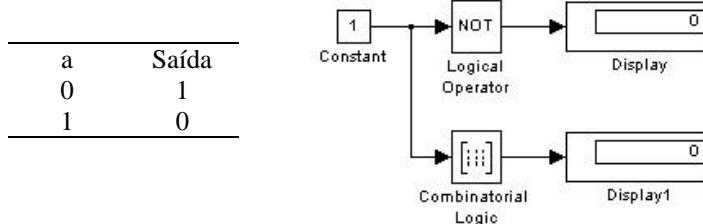
Nos parâmetros do bloco, Tabela Verdade (Truth table) consiste de um ou mais vetores coluna, no qual cada linha representa a saída correspondente ao índice da linha. Cada coluna na tabela verdade produz uma saída diferente e um único bloco Lógica Combinacional pode implementar múltiplos operadores lógicos. A saída do bloco não deve conter somente 0 ou 1, qualquer número é aceitável.

Exemplo

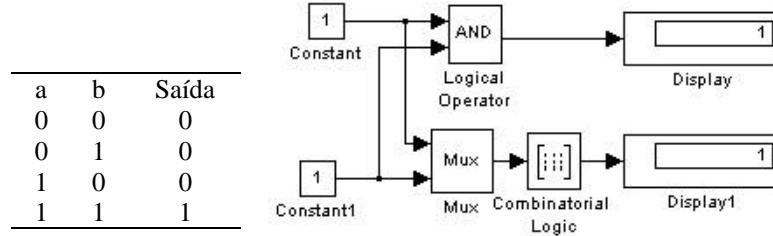
Deseja-se implementar as três expressões que seguem utilizando blocos de Lógica Combinacional e de operadores lógicos, comparando as duas respostas.

Expressão	Equivalente
\bar{a}	NÃO a (NOT a)
ab	$a \text{ E } b$ ($a \text{ AND } b$)
$ab + bc$	$(a \text{ AND } b) \text{ OR } (b \text{ AND } c)$ $(a \text{ AND } b) \text{ OU } (b \text{ E } c)$

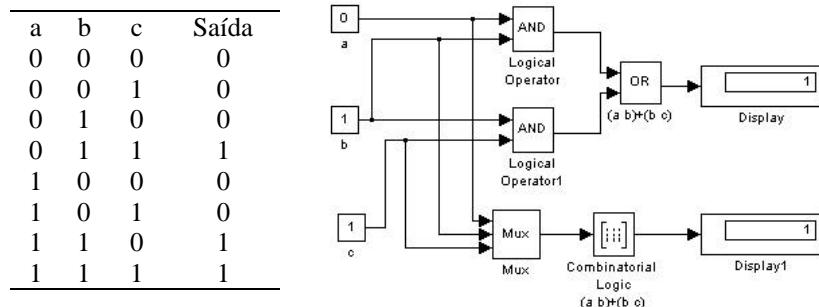
A tabela verdade para \bar{a} é mostrada a seguir. O parâmetro Tabela Verdade equivalente do bloco Lógica Combinacional é a saída da coluna, [1;0]. Um modelo SIMULINK que implementa esta expressão usando ambos os tipos de blocos é mostrado na figura abaixo:



A tabela verdade para ab é mostrada ao lado da figura do modelo implementado. Note que se escolhermos a como coluna da esquerda, o vetor de entrada deve ser $[a,b]$. O parâmetro Tabela Verdade é a coluna de saída da tabela mostrada, ou seja, $[0;0;0;1]$. O vetor de entrada é composto utilizando um Mux na entrada com os componentes em ordem $[a,b]$.



A expressão lógica final $ab + bc$ tem a tabela seguinte verdade:



O parâmetro Tabela Verdade deve conter o vetor coluna $[0;0;0;1;0;0;1;1]$.

5.4 - Sistemas Discretos no Tempo Vetoriais

O conceito de Espaço de Estados discutido no capítulo anterior é também muito utilizado em sistemas discreto no tempo. Assim como as variáveis de estado em sistemas contínuos representam derivadas ou combinações lineares de derivadas, as variáveis de estado em sistemas discretos representam porções de seqüências ou combinações de seqüências. A equação discreta equivalente é a forma geral de um modelo discreto de Espaço de Estado de um sistema dinâmico:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}, \mathbf{u}, k)$$

onde \mathbf{x} , como antes é o vetor de estado e \mathbf{u} é o vetor de entrada. A equação de saída correspondente é

$$y(k) = \mathbf{g}(\mathbf{x}, \mathbf{u}, k)$$

A notação matricial para sistemas discretos lineares é similar ao modelo para sistemas contínuos. As mesmas 4 matrizes (matriz de sistema, matriz de entrada, matriz de saída e matriz de transmissão direta) são utilizadas. As equações de sistema passam a ser então:



$$\mathbf{x}(k+1) = \mathbf{Ax}(k) + \mathbf{Bu}(k)$$

e a equação de saída:

$$\mathbf{y}(k) = \mathbf{Cx}(k) + \mathbf{Du}(k).$$

Uma vez obtida a equação no espaço de estados, o modelo do sistema pode então ser implementado no SIMULINK utilizando o bloco Espaço de Estado Discreto encontrado na biblioteca Discreta. O uso deste bloco é análogo ao bloco Espaço de Estado discutido no capítulo anterior.

Exemplo

Este exemplo consiste em desenvolver matrizes no Espaço de Estado para uma sistema descrito pela função de transferência

$$y(k+1) = y(k) - 2y(k-1) + u(k).$$

As variáveis de estado definidas são

$$x_1(k) = y(k-1)$$

$$x_2(k) = y(k)$$

As equações de estado são

$$x_1(k+1) = x_2(k)$$

$$x_2(k+1) = x_2(k) - 2x_1(k) + u(k)$$

Na forma matricial tem-se

$$\mathbf{x}(k+1) = \begin{bmatrix} 0 & I \\ -2 & I \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u.$$

A matriz de sistema é então

$$\mathbf{A} = \begin{bmatrix} 0 & I \\ -2 & I \end{bmatrix}$$

e a matriz de entrada

$$\mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Já que não há transmissão direta, \mathbf{D} é zero. Se a saída é $y(k)$, então

$$\mathbf{C} = [0 \ 1].$$

5.5 - Sistemas Discretos com Diferentes Taxas de Amostragem Simultâneas

Muitos sistemas discretos envolvem subsistemas que operam em diferentes taxas (rates), e com diferentes fases. Por exemplo, um computador comum tem numerosos subsistemas discretos como a unidade central de processamento, as controladoras de interface serial e paralela, os drives, as controladoras de vídeo e ainda dispositivos de entrada como teclado e mouse.

Modelos de sistemas de comunicação e processamento de sinais consistem de subsistemas que operam em diferentes taxas (rates). Modelar sistemas multi-taxas discretos com o SIMULINK é muito similar à modelar sistemas com uma única taxa. É evidente que sistemas multi-taxas requerem uma atenção especial nos valores de tempos de amostragem e de offsets.

O SIMULINK utiliza cores diferentes para diferentes taxas, o que é de muito ajuda em sistemas multirate. Ele codifica automaticamente as linhas de sinal correspondente à pelo menos cinco taxas de amostragem diferentes. Para ativar esta função clica-se no menu **Format:Sample time colors** na barra de menu. Se for feita alguma mudança no modelo após se ativar esta função, deve-se então clicar em **Edit:Update Diagram** para atualizar as cores. A tabela a seguir lista as cores disponíveis e seus significados:

Cor	Significado
Preto	Blocos Contínuos
Magenta	Blocos Constantes
Amarelo	Híbridos (vários Sample times ou blocos contínuos e discretos juntos)
Vermelho	Taxa de amostragem mais rápida
Verde	Segunda taxa mais rápida
Azul	Terceira taxa mais rápida
Azul Claro	Quarta taxa mais rápida
Verde Escuro	Quinta taxa mais rápida
Ciano	Taxas de amostragem disparadas (usado com subsistemas disparáveis)

Blocos inherentemente contínuos no tempo (como integradores) e blocos inherentemente discretos (como o atraso unitário) são assinalados com cores dadas pela tabela. Linhas de sinal e blocos que não sejam inherentemente nem contínuos nem discretos (como ganho) são assinalados com as cores que se baseiam nas taxas de amostragem dos sinais em suas entradas. Assim, se uma linha de sinal vem da saída de um integrador, esta será preta. Se por ela passar um sinal vindo de um bloco atraso unitário, sua cor vai depender da taxa de amostragem do bloco

atraso unitário. Se a entrada de um bloco como o de soma consistir de vários sinais com taxas de amostragem de todos os sinais múltiplos inteiros da taxa de amostragem mais rápida, o bloco recebe então a cor correspondente ao sinal mais rápido. Se os tempos de amostragem não forem múltiplos inteiros do sinal mais rápido, o bloco recebe a cor preta.

Somente os cinco sinais com taxas de amostragem mais rápidas recebem cores próprias. Se houver mais de cinco sinais com diferentes taxas de amostragem, todos os blocos além do 5º receberão a cor amarela.

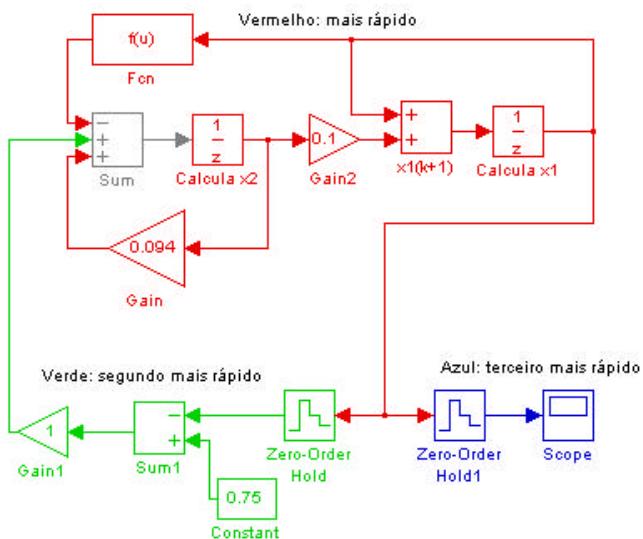
Exemplo

Sistemas de controle para processamento discreto freqüentemente operam com freqüências menores que a freqüência de atualização do processo, respeitando as limitações da velocidade do computador. Sistemas de visualização usualmente são atualizados numa freqüência suficientemente baixa para que a visualização seja legível. Como um exemplo simples de um sistema multi-taxa, supõe-se que algum processo a ser controlado comporta-se de acordo com as seguintes equações discretas no espaço de estados:

$$x_1(k+1) = x_1(k) + 0.1x_2(k)$$

$$x_2(k+1) = -0.05 \operatorname{sen} x_1(k) + 0.094x_2(k) + u(k)$$

onde $u(k)$ é a entrada. O processo possui um tempo de amostragem de 0.1 segundo. O processo será controlado utilizando um controle proporcional com um tempo de amostragem de 0.25 segundos, e atualização do dispositivo de visualização de 0.5 segundo. Um modelo SIMULINK deste sistema é mostrado na figura que se segue.

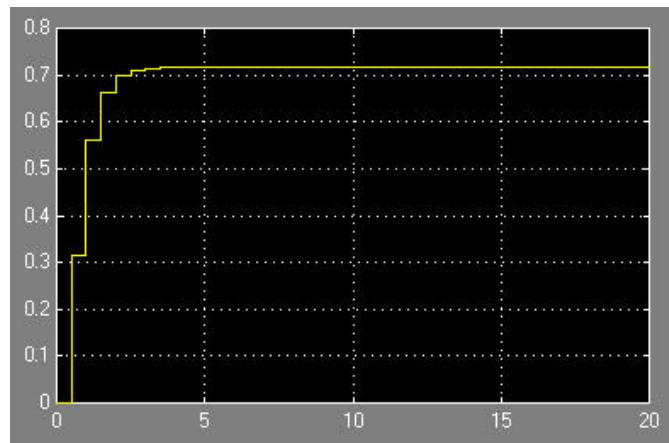


Os blocos na seção do modelo com linhas de sinal vermelho estão configurados para um tempo de amostragem de 0.1 segundos. Esta seção representa o processo dinâmico.

A seção com linhas de sinal verde é o controlador. Este é o controlador proporcional. O bloco Extrapolador de ordem zero faz com que o controlador atualize o sinal em intervalos de 0.25 segundos. O controlador produz um sinal de saída que é proporcional à diferença entre o *setpoint* (0.75) e o valor de entrada do bloco Extrapolador de ordem zero (x_1) no tempo de amostragem mais recente.

A seção com linhas de sinal azul é o dispositivo de visualização, aqui um bloco osciloscópio. O bloco Extrapolador de ordem zero nesta seção está configurado para um tempo de amostragem igual a 0.5 segundo.

Executando a simulação, obtém-se o seguinte resultado:



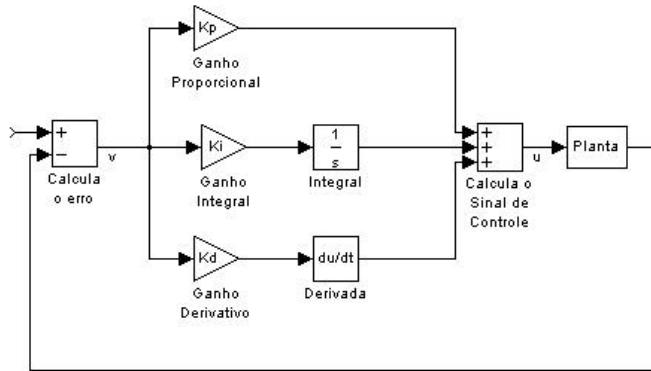
5.6 - Sistemas híbridos

Sistemas híbridos consistem de componentes discretos e contínuos. O sistema híbrido mais comum é o processo físico controlado por lógica discreta ou um computador. Modelar sistemas híbridos no SIMULINK é simples, como ilustrado no exemplo a seguir.

Exemplo

Para ilustrar a construção de um modelo híbrido é proposto substituir o controlador contínuo de um exemplo anterior por um controlador proporcional-integral-derivativo (PID) discreto, com um tempo de amostragem de 0.5 segundos.

A figura ilustra um controlador PID contínuo.



O controlador consiste de 3 seções, as quais operam com a diferença (v) entre o sinal de saída do processo a controlar e o valor de *offset*.

A seção proporcional produz um sinal proporcional à diferença entre o valor desejado e a saída do sistema no valor atual. A saída proporcional é então

$$u_p = K_p v$$

A seção integral do controlador PID tem por objetivo reduzir o erro estacionário. Este componente produz uma saída que é proporcional à integral no tempo do sinal de erro:

$$u_i = K_i \int_0^t v dt$$

Um problema com a seção integral é que se a resposta do processo a variação do sinal de entrada (v) for relativamente lenta, u_i pode crescer rapidamente. Este fenômeno é chamado *wind-up*. O Wind-up pode ser evitado adicionando limites inferior e superior ao valor de u_i .

A seção derivativa proporciona amortecimento. Sua saída é proporcional a taxa de mudança de \dot{v} :

$$u_d = K_d \dot{v}$$

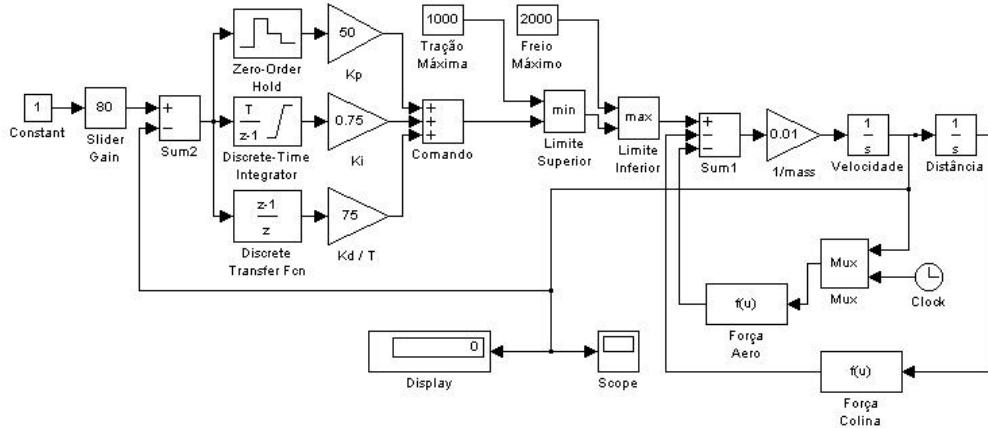
Num controlador PID discreto substitui-se a seção integral por um integrador discreto e a seção derivativa por uma aproximação discreta da derivada. Um aproximação numérica de primeira ordem da derivada é

$$u_d(k) = \frac{v(k) - v(k-1)}{T}$$

A função de transferência da aproximação da derivada é

$$\frac{U_d(z)}{V(z)} = \frac{K_d}{T} \left(\frac{z-1}{z} \right)$$

A figura a seguir mostra o modelo SIMULINK de um automóvel utilizando um controlador PID discreto com $K_p = 50$, $K_i = 0.75$ e $K_d = 75$.

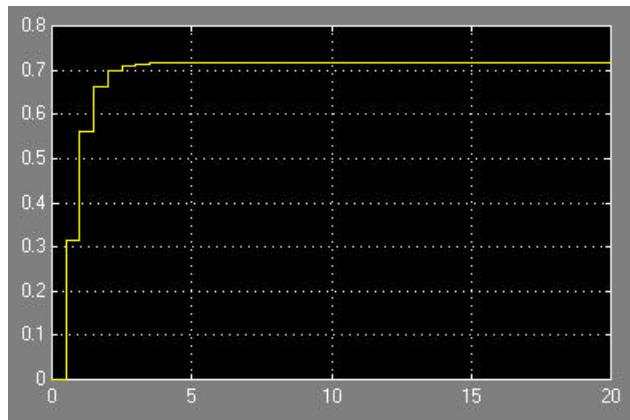


O modelo é idêntico ao modelo do exemplo anterior, com exceção do controlador. A parte proporcional consiste de um bloco Extrapolador de ordem zero e um bloco ganho proporcional. O ganho proporcional (50) neste controlador é o mesmo utilizado anteriormente no controlador proporcional contínuo.

A parte integral do controlador PID consiste de um Integrador no tempo discreto e um bloco de ganho (0.75). Este bloco Integrador no tempo discreto deve ser configurado para trabalhar com limites inferior e superior para a saída em ± 100 . Isto é feito clicando-se na opção Limite de Saída (Limit Output).

A parte derivativa do controlador é construída utilizando um bloco Função de Transferência Discreta e um bloco de ganho (75).

Neste exemplo a simulação foi configurada para um tempo final de 1000 segundos e o ganho Slider foi ajustado para uma velocidade de 80 m/s. O osciloscópio deve produzir então a seguinte imagem:

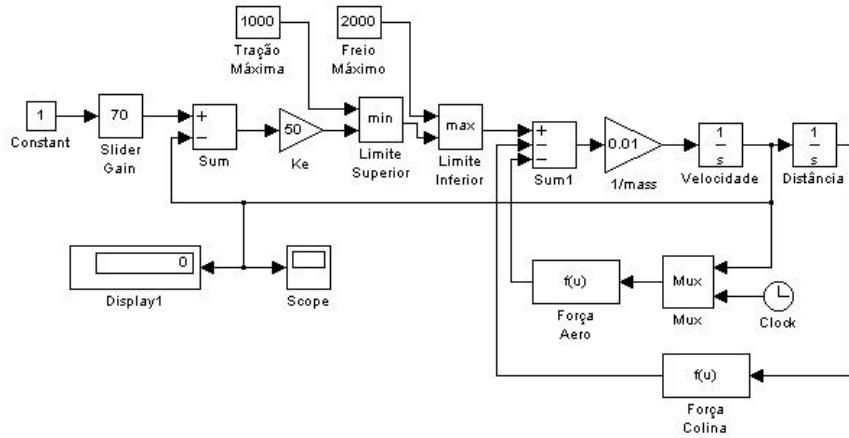


Capítulo 6 – Subsistemas e Máscaras

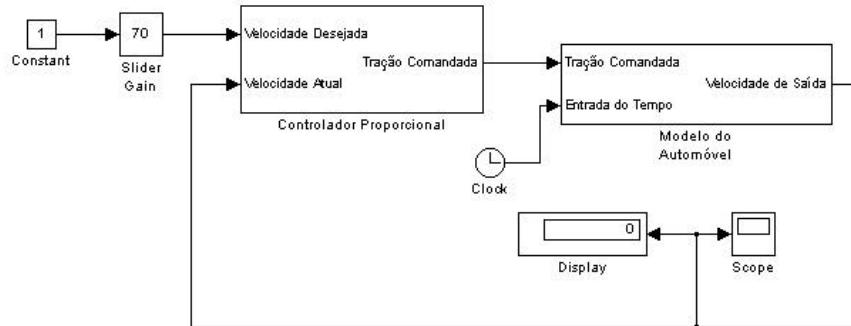
6.1 - Subsistemas SIMULINK

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.

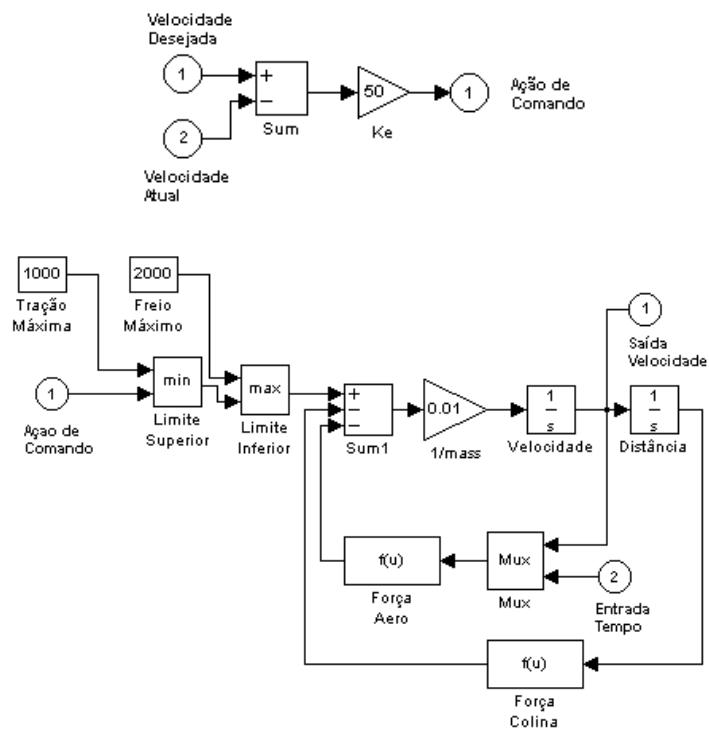
A medida que os modelos crescem e ficam complexos, tendem a ficar difíceis de se entender e executar manutenção. Subsistemas resolvem este problema fazendo que um complexo e grande modelo em grupos hierárquicos de modelos menores. Como um exemplo tem-se o modelo do automóvel já utilizado em exemplos 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 **debugg** cada vez que o subsistema for usado em um novo modelo.

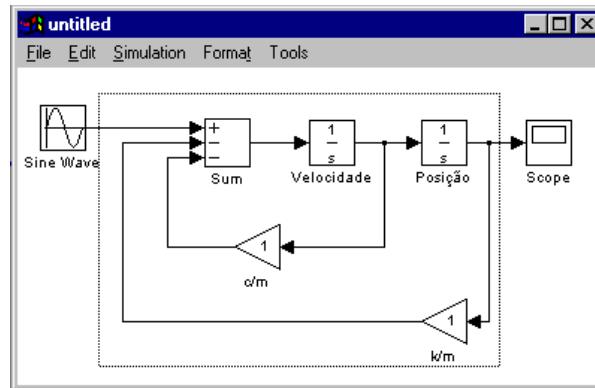


Existem dois métodos de construir subsistemas SIMULINK. O primeiro método é encapsular uma parte de um sistema já existente em um modelo pelo menu **Edit>Create Subsystem**. O segundo método é utilizar um bloco Subsistema (Subsystem) da biblioteca de Conexões (Connections). Os dois métodos serão discutidos.

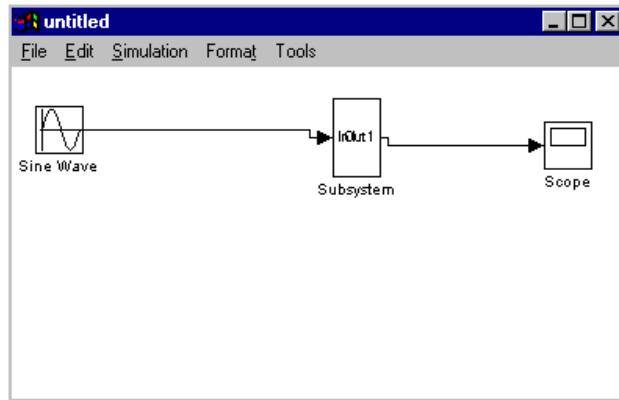
6.1.1 - Encapsulando um Subsistema

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

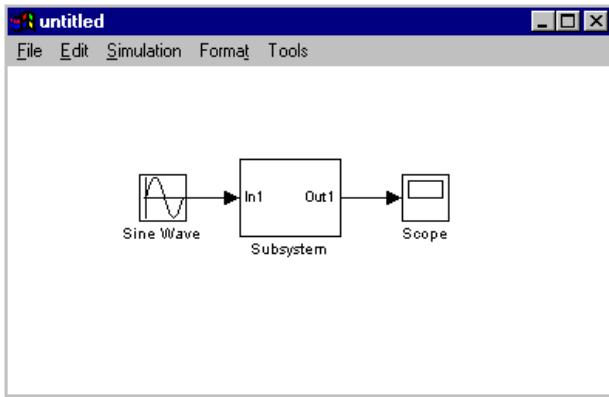
Selecione todos os blocos e linhas de sinal a serem incluídos no subsistema usando uma caixa de contorno com o mouse. Freqüentemente é necessário rearranjar alguns blocos para que a caixa inclua somente os blocos desejados para o subsistema.



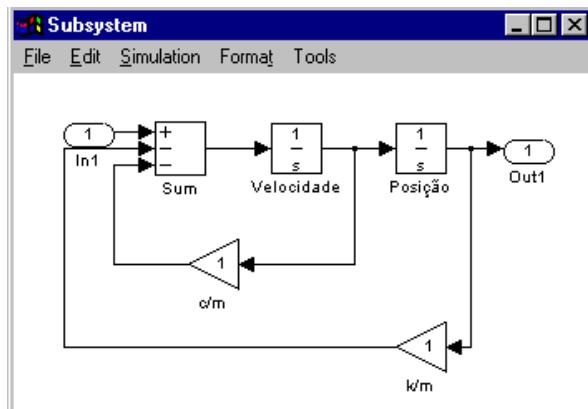
Clique em **Edit>Create Subsystem** na barra de menu da janela do modelo. O SIMULINK irá então substituir os blocos selecionados por um bloco Subsistema com entrada e saída para cada sinal de entrada ou saída.



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 Import 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 selecionar 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

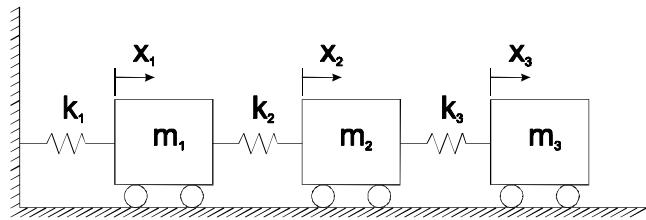
Se ao se construir um modelo o usuário souber que irá precisar de um subsistema, é conveniente construir o subsistema diretamente numa janela. Isto elimina a necessidade de se rearranjar os blocos que irão compor o subsistema de modo a enquadrá-los no espaço disponível. Isto permite que o usuário inicie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

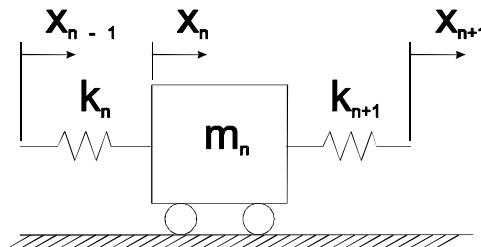
um duplo clique sobre ele abre-se a janela do subsistema. Nesta janela deve-se construir o subsistema utilizando os procedimentos padrões para construção de modelos. Utiliza-se blocos Import para os sinais de entrada do sistema e blocos Outport para os sinais de saída. Se desejado, pode-se mudar os nomes 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.



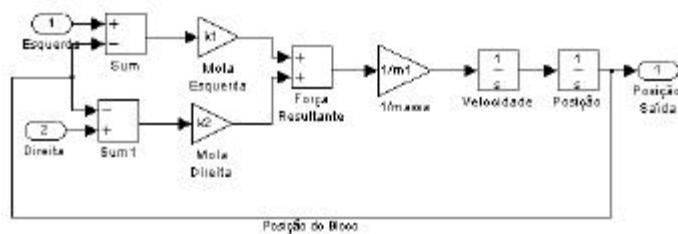
O modelo será construído a partir de blocos subsistemas que irão modelar cada carro, como na figura:



A equação do movimento para um carro é:

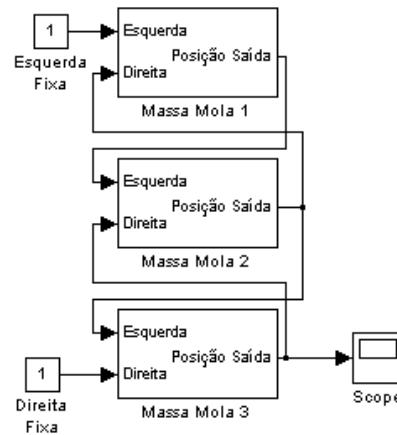
$$\ddot{x}_n = \frac{1}{m_n} [k_n(x_{n-1} - x_n) + k_{n+1}(x_n - x_{n+1})]$$

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 exempl_1.m, o MATLAB irá abrir o modelo SIMULINK quando você digitar o comando exempl_1 na área de trabalho do MATLAB.

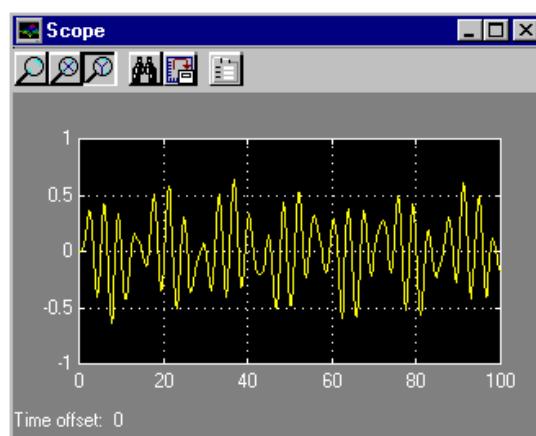
Os parâmetros dos blocos de cada cópia do subsistema devem ser agora configurados. Para o carro 1, o valor do ganho do bloco de ganho Mola da Esquerda deve ser k_1 e o do bloco de ganho Mola da Direita k_2 . A seguir, o bloco 1/massa

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.

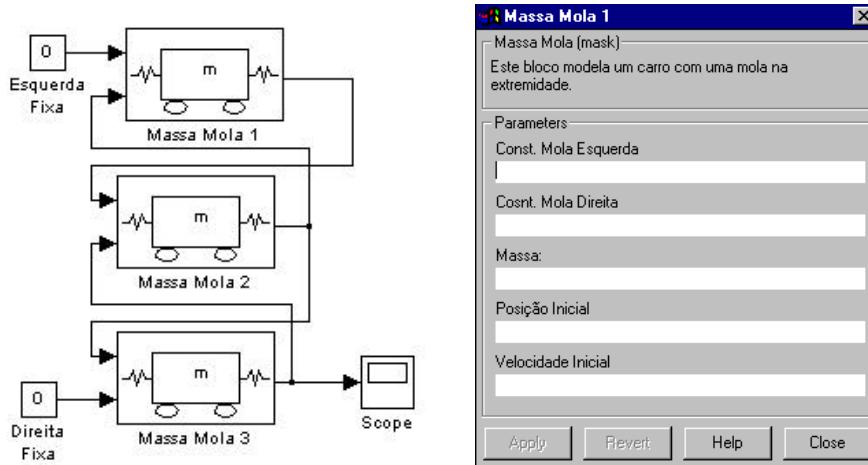
O bloco osciloscópio deve ser configurado para salvar os dados mostrados para a área de trabalho do MATLAB, com o tempo inicial 0 e o final 100. Após executar a simulação, os dados do osciloscópio podem ser plotados no MATLAB, resultando na trajetória do carro 3 mostrado logo a seguir:



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.

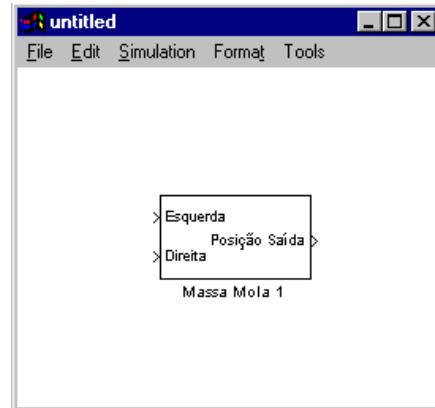
O processo de produção de um bloco com máscara pode ser resumido em:

1. Construir um subsistema como já discutido.
2. Selecionar o bloco subsistema e clica-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.

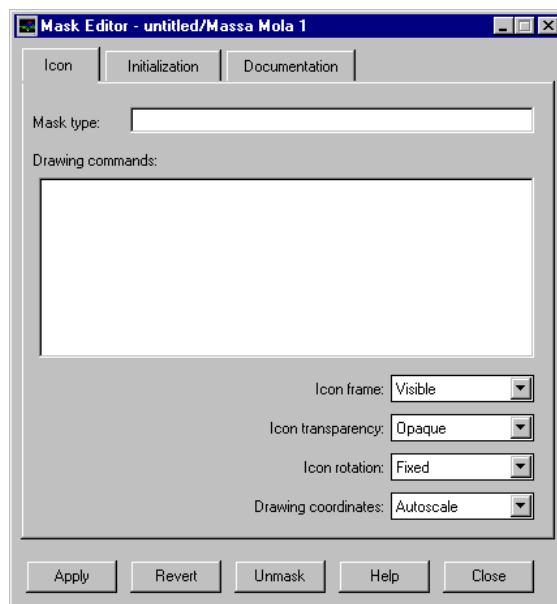
O primeiro passo na criação de um subsistema com máscara é criar um subsistema utilizando procedimentos descritos anteriormente. Para ilustrar o processo, será construído um bloco Massa-Mola com máscara a partir de um dos subsistemas no modelo do exemplo anterior.

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.

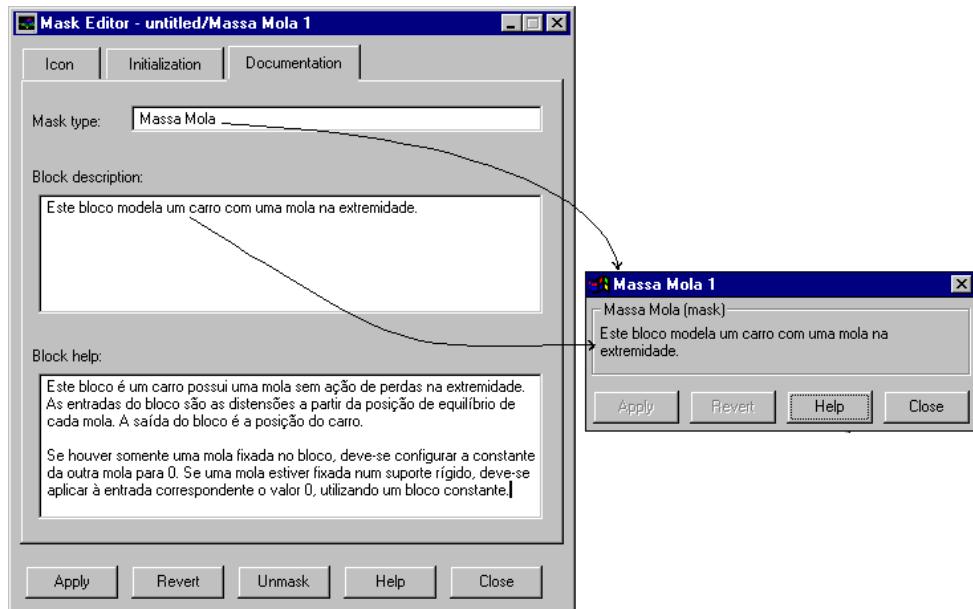
A caixa de diálogo do editor de máscara irá aparecer. Note que o editor de máscaras tem três páginas. Cada página será discutida em subseções.



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

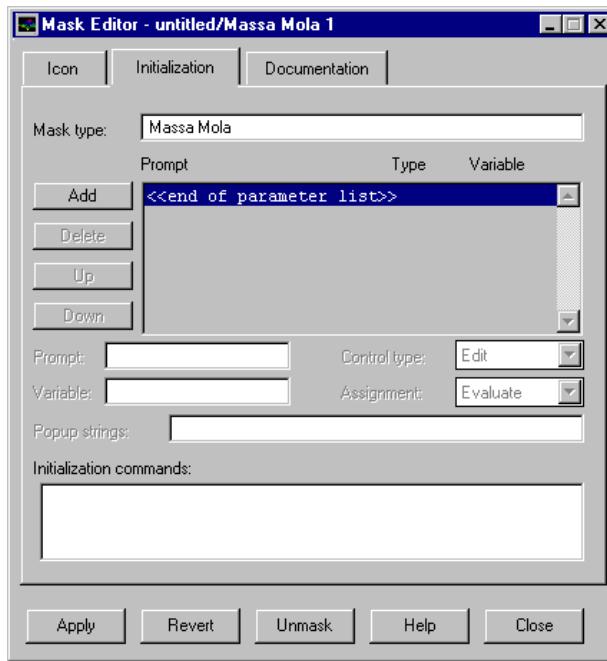
O segundo campo é mostrado na área interna da caixa. Este campo deve conter uma breve descrição do bloco, seu propósito e qualquer lembrete necessário à utilização do bloco.

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.

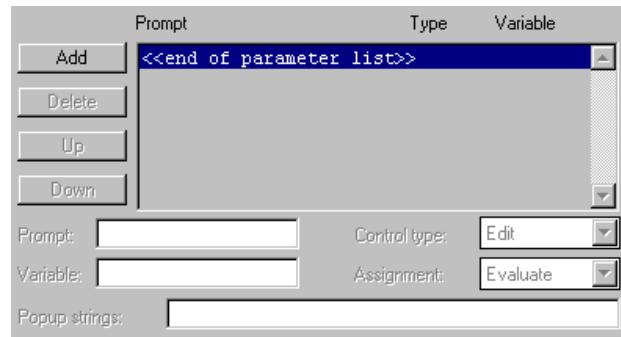
6.2.3.2 - Seção de Prompt da Caixa de Diálogo do Bloco

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.

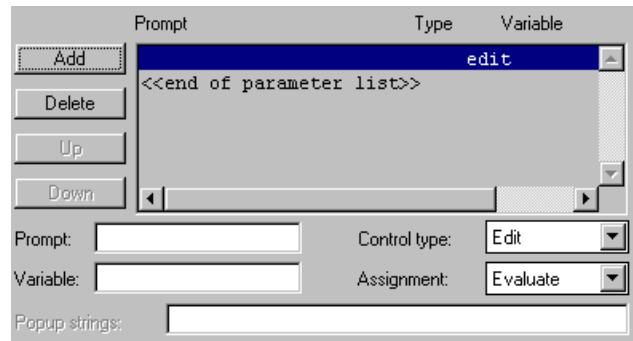
Para adicionar o primeiro campo no bloco Massa-Mola deve-se seguir os passos:

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

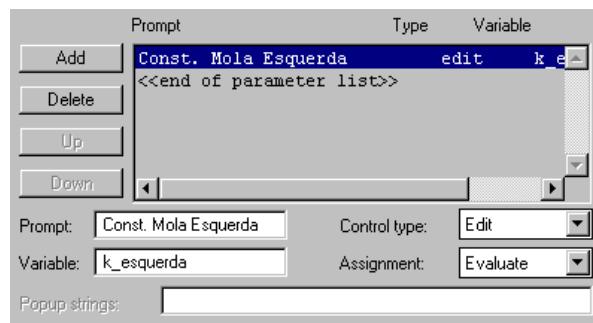


Uma linha em branco será inserida na lista de parâmetros.

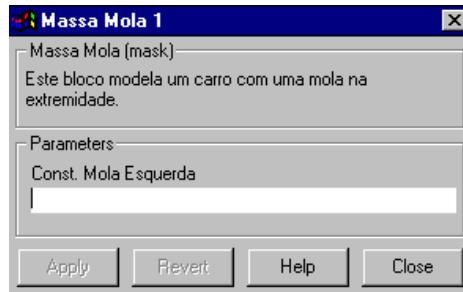


No campo Prompt digite: Constante da mola da esquerda.

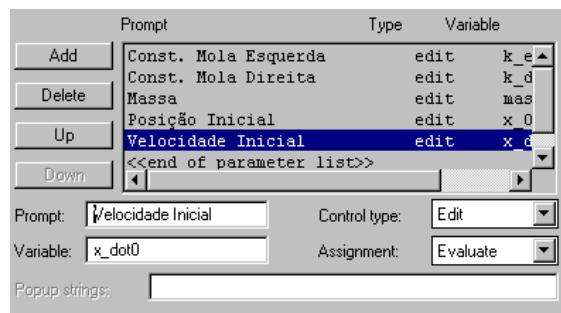
No campo Variable field digite: k_esquerda. Note que o prompt e o nome da variável aparecem na lista de parâmetros.



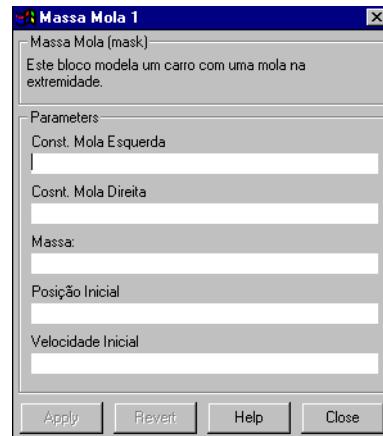
Clique a seguir em Close. Com um duplo clique no bloco Massa –Mola a caixa de diálogo irá abrir contendo agora o campo para a entrada da variável k_esquerda.



Acrescente agora as variáveis massa com o prompt Massa, k_direita com o prompt Constante da Mola Direita, x0 com o prompt Posição Inicial e x_dot0 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 selecionada. Os botões Up e Down são utilizados para mover as linhas para se encontrar a disposição adequada dos campos na caixa de diálogo.

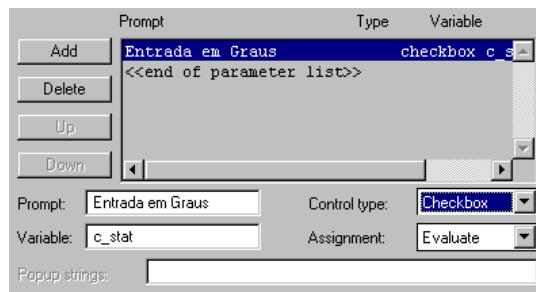
O campo Control type é uma lista que contém três opções: **Edit**, **Checkbox** e **Popup**. **Edit** produz um campo de entrada de dados. É o tipo de campo mais comum. **Checkbox** produz um campo que possui somente duas possibilidades de valores, dependendo da condição da caixa, se está selecionada 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'.

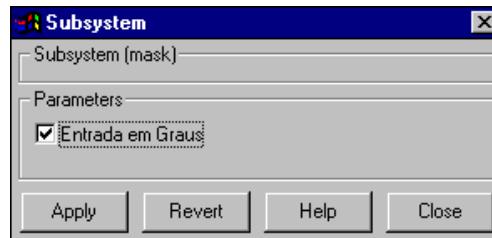
Escolher Checkbox no campo Control type produz uma caixa de checagem na caixa de diálogo. A variável associada irá assumir dois valores dependendo da escolha feita em Assignment. Se a escolha for Evaluate a variável poderá assumir 0 se a caixa não for marcada ou 1 se a caixa for marcada. Se a escolha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

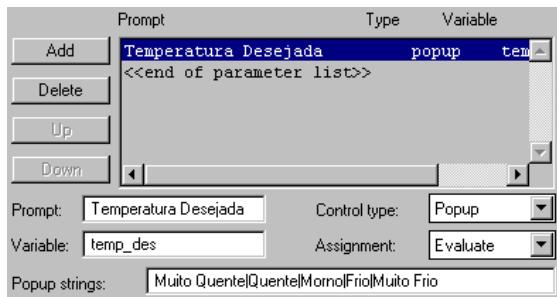
A lista de opções deve estar no campo Popup strings. As opções devem ser digitadas em seqüência separados por "|". Por exemplo se as opções forem muito quente, quente, morno, frio e gelado, no campo Popup strings deve conter Muito quente | Quente | Morno | Frio | Gelado.

Exemplo

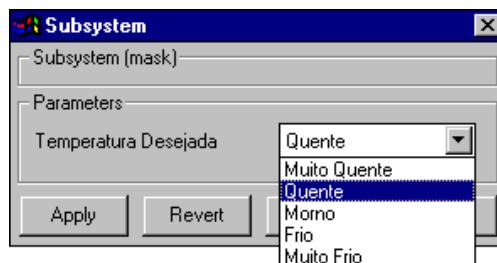
Suponha que se deseja agora criar um bloco com máscara que produza um sinal de saída definido numa lista que contenha as opções Muito quente, Quente, Morno, Frio e Gelado.

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

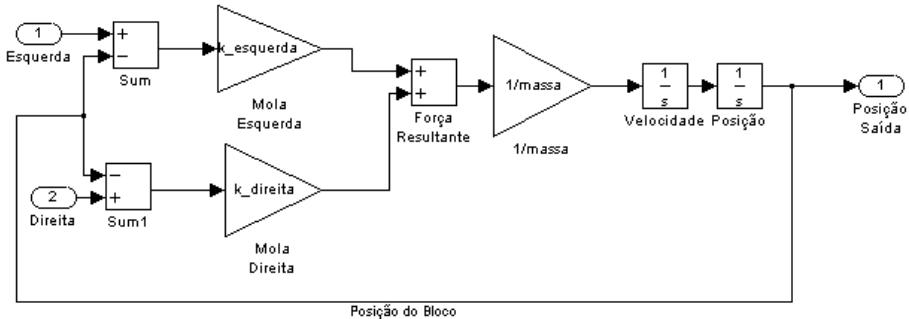
O ícone do bloco Massa-mola necessita de duas rodas. Para preparar o desenho das rodas, deve-se criar dois vetores, um contendo as coordenadas – x de um pequeno círculo e outro as coordenadas – y. Estas variáveis serão usadas na Página Ícone (Icon Page) para desenhar as rodas.

```
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.

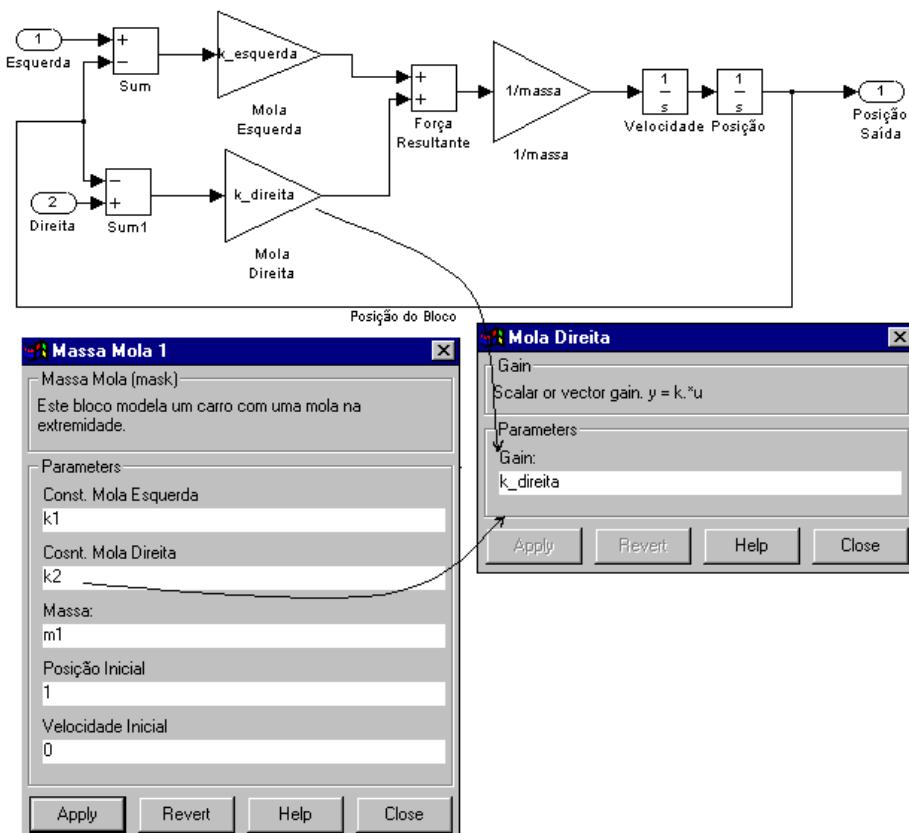
As variáveis internas de um subsistema com máscara são criadas e assumem valores definidos no Editor de Máscaras usando comandos de inicialização. Cada campo define uma variável interna acessível somente no subsistema. As variáveis internas adicionais devem ser definidas no campo Initialization Commands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

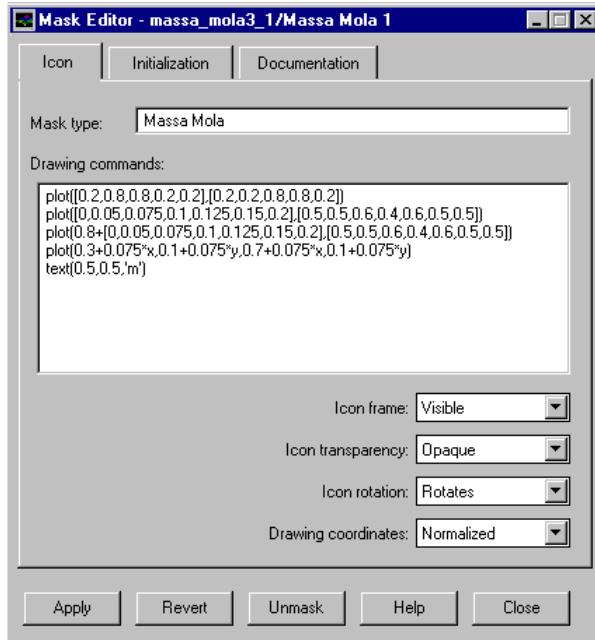
```
temp_list = [120,100,85,70,50];
```

```
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

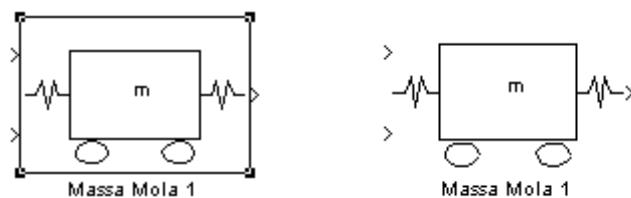
Esta página permite ao usuário personalizar ícones para seus blocos com máscara. É usada para criar e personalizar o ícone no bloco do carro para o sistema Massa-Mola (lembrando que x e y foram definidos no campo Initialization Commands). Esta página consiste de seis campos. O primeiro campo é idêntico 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 quatro campos a seguir é configurar o ícone. Explicar o campo Drawing Commands se torna fácil se antes forem discutidos estes campos de configuração.

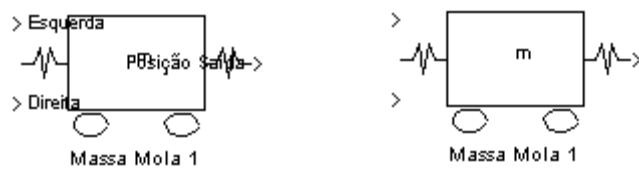
6.2.4.1 - Campo Moldura do Ícone (Icon Frame)

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.



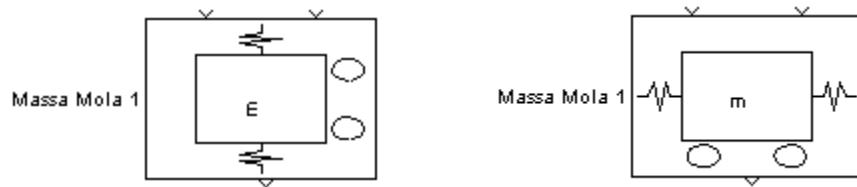
6.2.4.2 - Campo Transparência do Ícone

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 Import e Outport aparecem. Ao selecionar Opaco, estes nomes ficam escondidos.



6.2.4.3 - Campo Rotação do Ícone

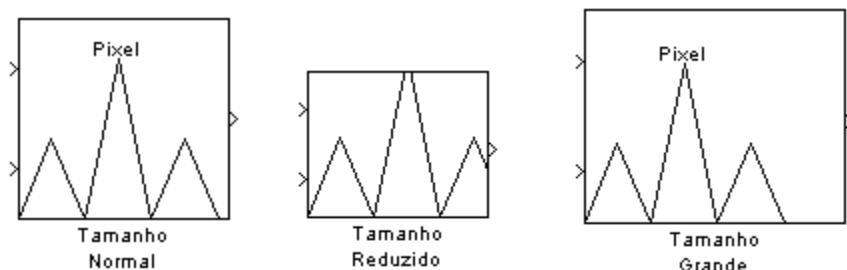
O terceiro campo é uma lista com as opções: Fixo e Rotacionar. Este campo determina o comportamento do ícone quando os comandos *Edit:Flip block* e *Format:Rotate block* são executados. Se a opção escolhida for fixa, quando o bloco for rotacionado ou invertido o ícone original conserva a mesma orientação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



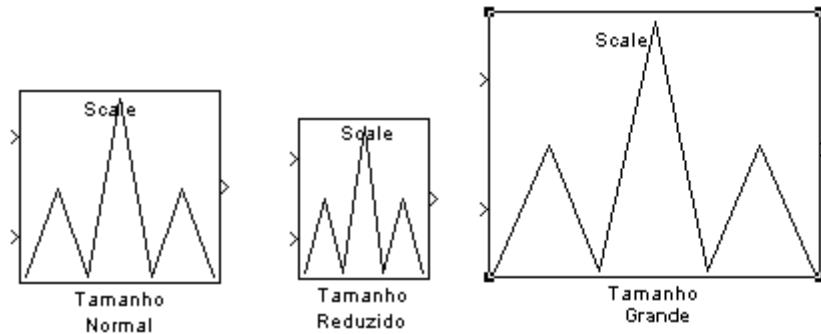
6.2.4.4 - Campo Coordenadas de Desenho (Drawing coordinates)

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.

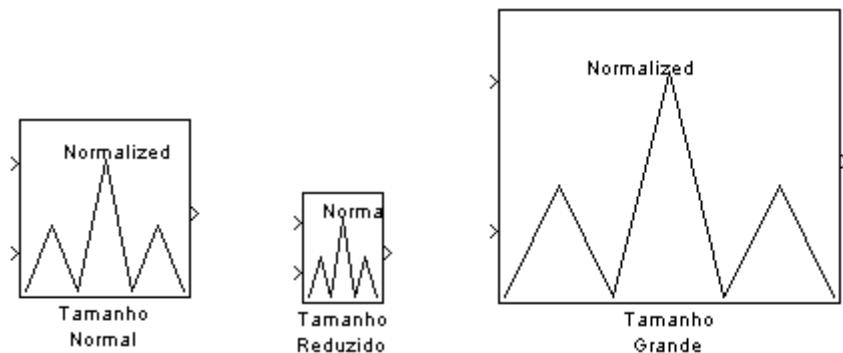
Pixel é uma escala absoluta e irá resultar em um ícone que não pode ser redimensionado. As coordenadas do canto inferior esquerdo são (0,0). As unidades são pixels, e o tamanho de um ícone irá depender somente da resolução do monitor. A figura mostra o resultado com a opção Pixel.



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 múltiplos 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 cercado 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

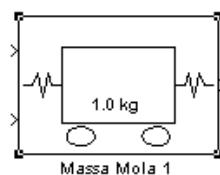
Suponha que se deseja mudar o ícone do carro no sistema Massa-Mola para que a massa do carro seja mostrada no ícone acima das rodas, em unidades de massa (kg). Adiciona-se então a seguinte linha de comando no campo Initialization Commands na página de inicialização do editor de máscaras.

```
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

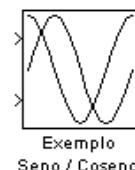
Suponha que se deseja mostrar funções seno e cosseno num ícone. Os seguintes comandos devem então ser acrescentados no campo Initialization Commands para produzirem os vetores necessários.

```
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 sinal. 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.

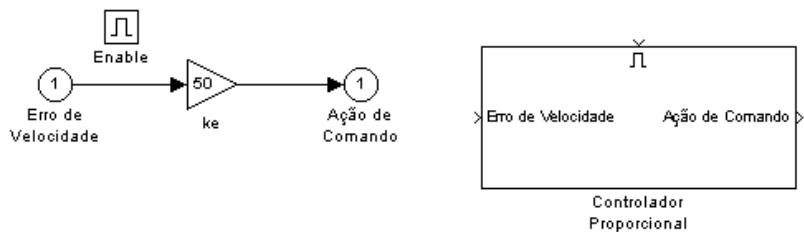
6.3.1 - Subsistemas com Habilitação (Enabled)

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.

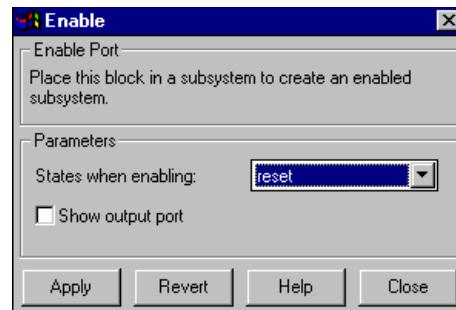
Se o usuário converter os vários subsistemas contendo as dinâmicas de vôo em subsistemas com opção de habilitação, somente os subsistemas ativos serão

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.

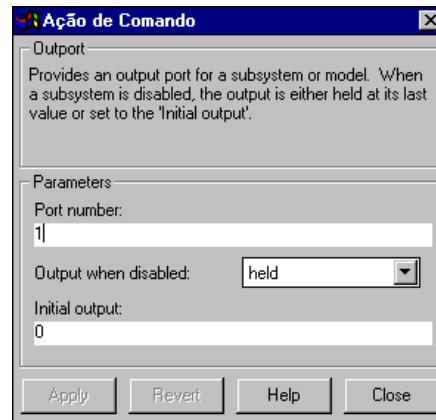


A caixa de diálogo do bloco de habilitação é a que se segue:



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 disable) 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.

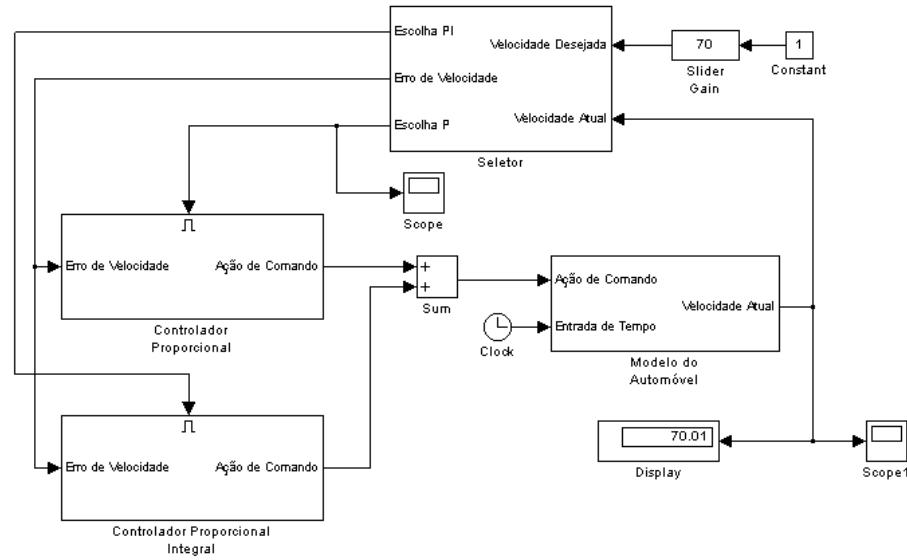
Exemplo

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

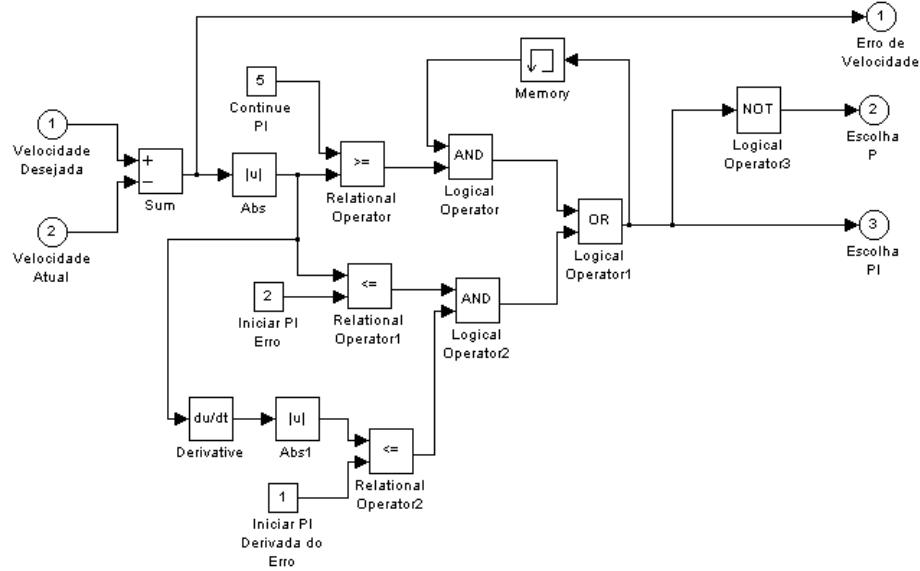
$$v_{err} = |\dot{x}_{desejado} - \dot{x}|$$

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/s², 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:

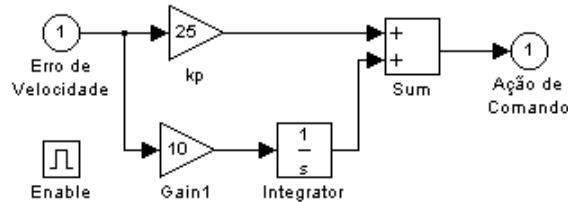


O subsistema seletor é mostrado na figura:



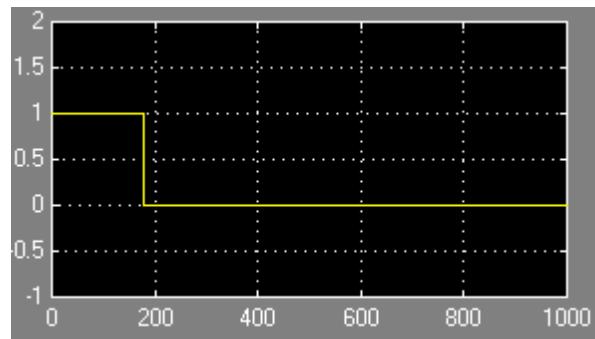
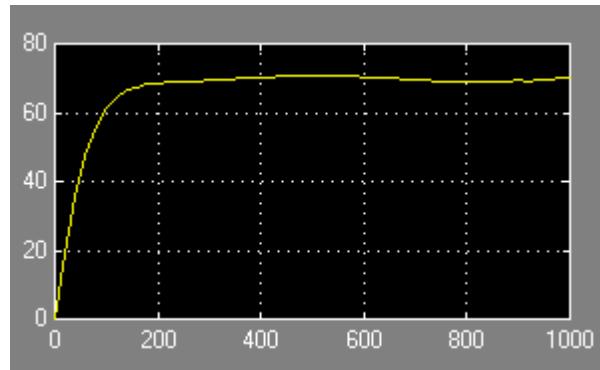
Este produz duas saídas. A saída PI é levada à 1.0 se as condições para o controlador PI forem satisfeitas, e 0.0 na situação contrária. A saída P é sempre o inverso lógico da saída PI.

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



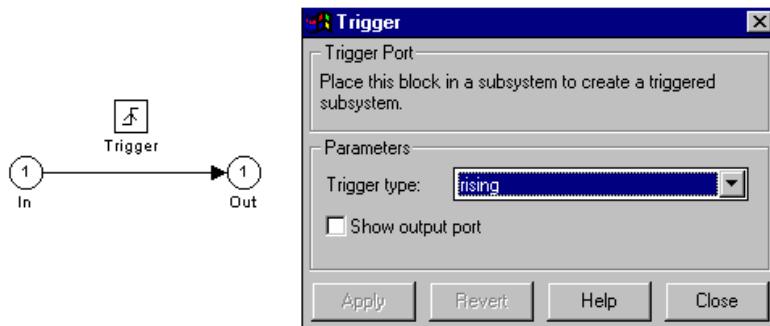
O bloco de Enable e o bloco Outport são configurados para reset.

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:



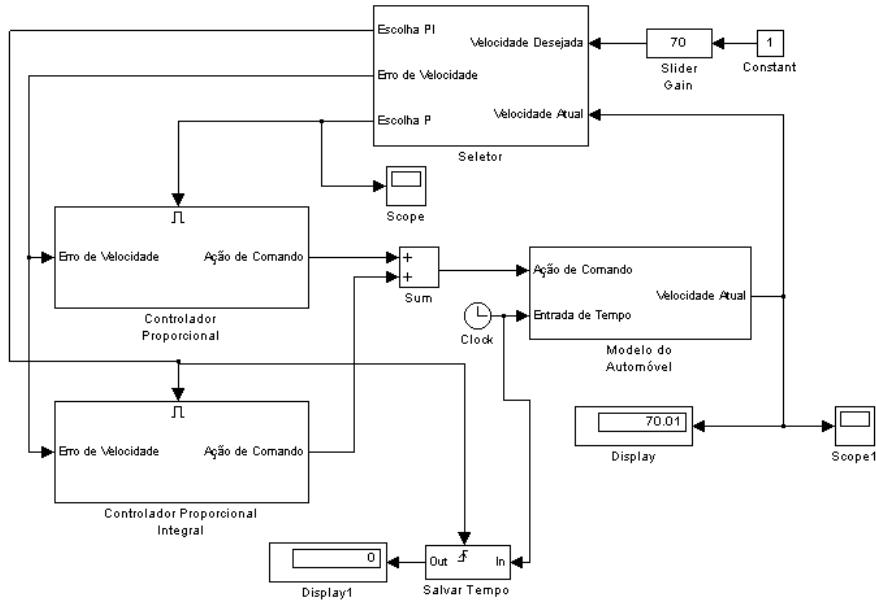
O primeiro campo da caixa (Trigger type) possui três opções: rising, falling e either. Se a opção escolhida for rising, um sinal de gatilho é definido quando este cruzar o zero enquanto estiver crescente. Se falling for escolhido, o sinal de gatilho passa a ser definido quando a entrada Trigger cruzar 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 herdam 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.

Capítulo 7 – Animação no SIMULINK

7.1 - Toolbox de Animação

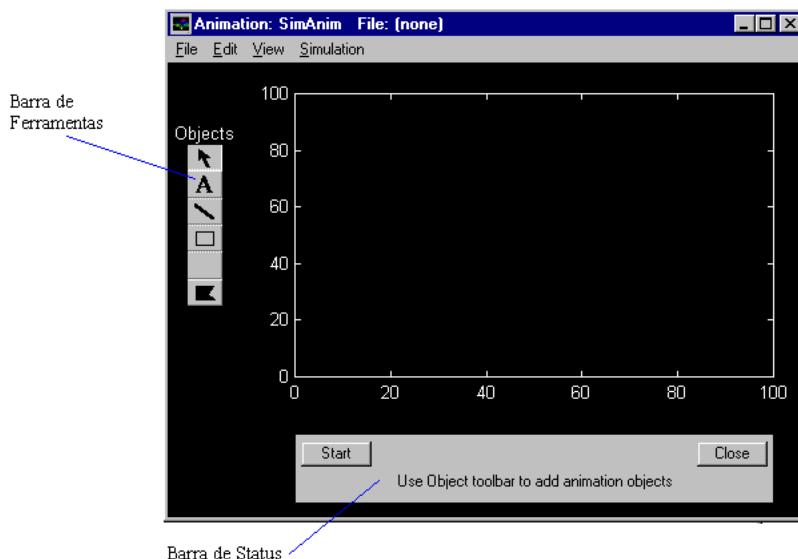
Esta toolbox é uma extensão do SIMULINK que permite ao usuário criar animações gráficas, de maneira similar à construção de um modelo no SIMULINK.

Para construir uma animação, deve-se copiar objetos gráficos para a janela de figuras e em seguida configurar as propriedades dos objetos usando suas caixas de diálogos.

7.2 - Usando a Toolbox de Animação

Para adicionar uma animação a um modelo deve-se clicar na biblioteca de nome **Blocksets & Toolboxes** e a seguir clicar em **Simulation Animation**. Uma janela irá abrir contendo um bloco de nome **SimAnim**. Deve-se arrastar este bloco para a janela do modelo em construção.

Um clique duplo neste bloco irá abrir a janela com a figura a seguir;



A janela de animação consiste de três áreas: A barra de ferramentas, a barra de status e a figura de animação.

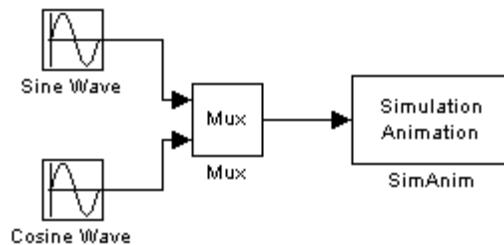
A barra de ferramentas consiste de um grupo de ícones representando os objetos de animação disponíveis: ponto, retângulo, texto, linha e retalho. A barra de status mostra o estado da animação e fornece botões para controlar a simulação e finalizar a animação. Quando a simulação está sendo executada, a barra de status contém um botão de Stop, uma caixa de verificação com a opção de mostrar o rastro da simulação e um botão para limpar este rastro. A figura de animação é a área em que a animação é criada e mostrada.

Para construir uma animação, deve-se clicar em um dos ícones da barra de ferramentas e a seguir clicar na figura no local desejado para o objeto representado

pelo ícone. Uma caixa de diálogo do objeto se abrirá. Deve-se então entrar com os dados de configuração nesta caixa de diálogo, clicar em Apply e a seguir fechar a caixa. Deve-se repetir estes passos para cada objeto na figura de animação. O usuário pode executar a simulação usando a barra de status clicando em **Start** ou em **Simulation:Start** na barra de menu da janela do modelo.

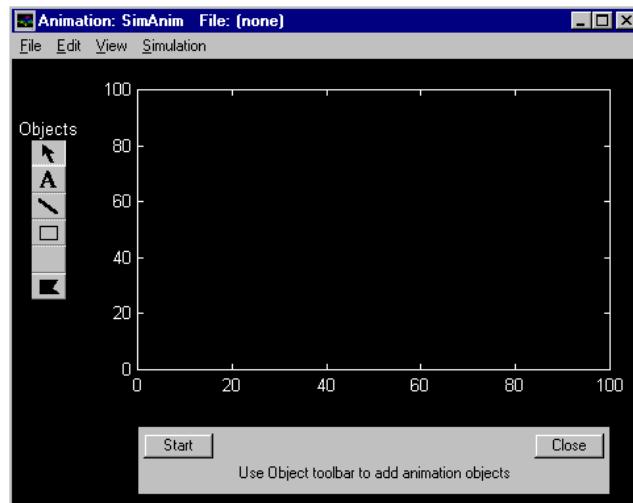
Exemplo

Deseja-se criar uma animação usando a Toolbox de Animação utilizando o modelo o modelo a seguir;

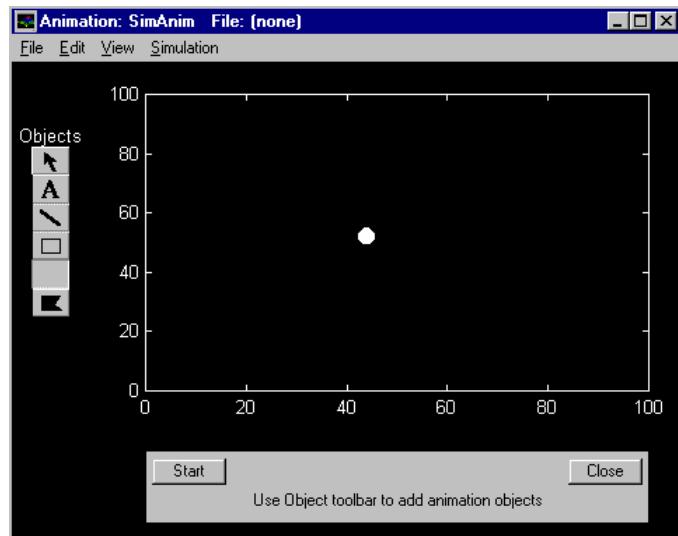


O modelo consiste em dois blocos de ondas senoidais (Sine Wave) conectados à um Mux para produzir um vetor como sinal. O bloco inferior deve ser configurado com a fase igual a $\pi/2$.

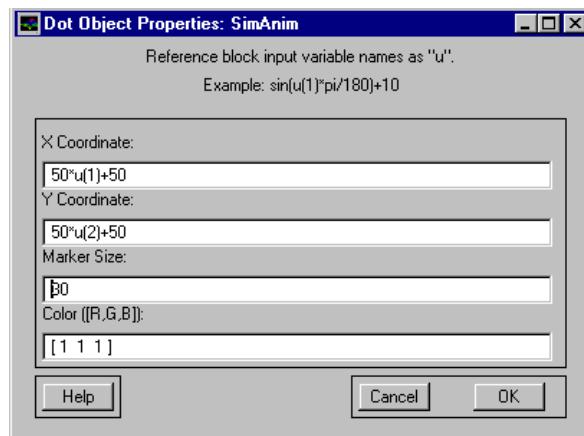
Um duplo clique no bloco Animation abre a figura a seguir:



Clica-se a seguir no ícone do ponto na barra de ferramentas:



A caixa de diálogo do objeto pode ser aberta com um duplo clique sobre este. Configura-se os campos desta caixa como mostrado na figura, clicando a seguir em **Ok**.



Executa-se agora a simulação clicando em **Start**. O ponto deve-se mover em círculo.

7.2.1 - Propriedades dos Objetos de Animação

Cada campo de propriedades contidos na caixa de diálogo pode conter expressões consistindo de constantes, variáveis correntes definidas na área de trabalho do MATLAB e elementos do vetor de entrada (u). Pode-se fazer da localização de um objeto uma função da entrada, ou ainda, a cor ou o tamanho do objeto como uma função da entrada.

7.2.2 - Configurando uma animação

O menu **View** contém várias opções muito úteis na configuração da aparência de uma figura de animação.

7.2.3 - Propriedades da Figura

O menu **View** fornece opções para mostrar ou ocultar muitas propriedades de figura. Pode-se ocultar ou não os eixos, o reticulado (grid), a borda etc. Há também opções para mostrar ou ocultar a barra de status e a barra de ferramentas.

7.2.3.1 - Escala da Figura

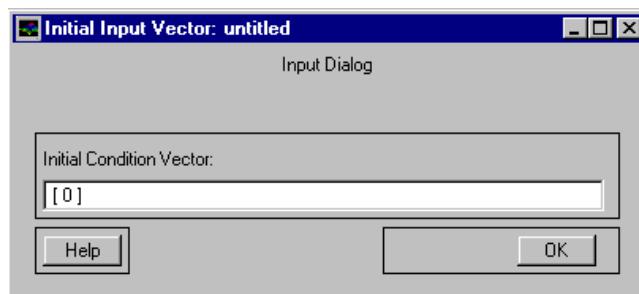
View:AutoScale permite controlar ou não a escala de uma figura automaticamente. O conteúdo da opção autoscale pode ser muito útil determinando limites apropriados para axis. Uma vez que o usuário encontrou limites aceitáveis, a aparência da animação pode ser melhorada escolhendo-se **Autoscale** para **off** e a seguir usando **View:Change Axis Limits** para um ajuste fino dos limites.

7.2.4 - Modificando uma Animação

Após executar uma simulação o usuário pode adicionar objetos ou modificar alguns já existentes. Antes de modificar a figura, deve-se clicar em **Simulation:Reset** na barra de menu da janela de animação. Pode-se modificar um objeto já existente com um duplo clique sobre o mesmo.

7.2.5 - Configurando Entradas Iniciais

O usuário pode configurar valores iniciais para as entradas do bloco de animação ($u(1)$, $u(2)$ etc) clicando em **Simulation:Set Initial Inputs**. Uma caixa de diálogo se abrirá.



Deve-se entrar com os valores iniciais da entrada na forma de um vetor. Esta capacidade é muito usada devido ao fato de que o bloco de animação não tem acesso às entradas antes da simulação começar. Inicializando as entradas prevê-se um grande transiente no início da animação. Configurar as entradas na caixa de diálogo não causa qualquer efeito nas próprias entradas; seus valores não se propagam para blocos anteriores ao bloco de animação. O valor inicial padrão é zero.



7.3 - Salvando e Carregando Arquivos de Animação

O menu **File** da janela de animação contém opções para salvar ou carregar uma animação. A informação que uma Toolbox de animação precisa para gerar uma figura de animação está armazenada na forma de um arquivo .mat do MATLAB.

Quando o usuário abre uma janela de animação com um duplo clique num bloco de animação, e se pela última vez que o usuário acessou este bloco ele salvou uma animação associada com o mesmo, haverá um Prompt perguntando se o usuário deseja usar a animação anterior ou iniciar uma nova.



Bibliografia

- The Student Edition of SIMULINK
James B. Dabney / Thomas L. Harman
- Mastering SIMULINK
James B. Dabney / Thomas L. Harman
- Sistemas Lineares – Vol. 2
Ralph J. Schwarz / Bernard Friedland