Universidade Federal de Minas Gerais - Departamento de Engenharia Eletrônica

ELT062 - OFICINA DE SIMULAÇÃO ANALÓGICA E DIGITAL EM CONTROLE

TUTORIAL 5

Instrutor: Alexandre R. Mesquita

PARTE 1

A partir desta aula começaremos a usar o ambiente gráfico de simulação Simulink. Os algoritmos de simulação do Simulink são os mesmos que usamos para solução numérica de EDOs nos arquivos .m. O que passa a ser diferente é a maneira de programar, que se dá de forma gráfica no Simulink.

Para abrir o Simulink, digite simulink na linha de comando do Matlab. Esse comando abrirá uma janela com uma biblioteca de blocos que podem ser usados para criar modelos em Simulink.

Clique no botão New model (página em branco) para criar um novo modelo e salve seu modelo como mod5a.mdl. Usando blocos da biblioteca Commonly Used Blocks, crie um diagrama idêntico ao da Figura 1.

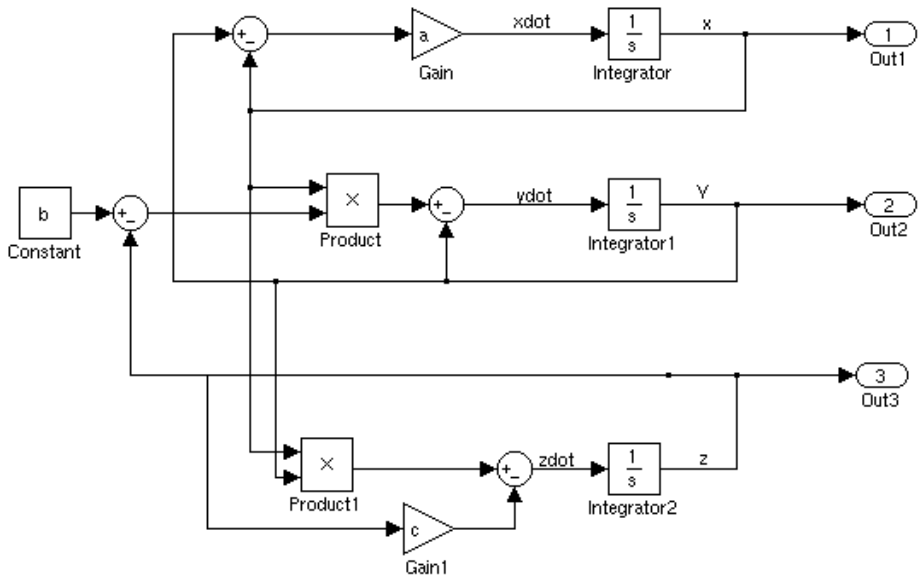

FIGURA 1. Diagrama de blocos para o atrator de Lorenz

Para criar o diagrama, basta arrastar os blocos da biblioteca para o seu modelo e ligar as saídas e entradas de cada bloco como indicado. Para criar um novo ramo saindo de uma ligação já existente, segure a tecla 'ctrl' ('control' no MacOs), clique na ligação desejada e arraste. Dê um clique duplo sobre os blocos Gain e Constant para ajustar seus valores conforme indicado na figura. O bloco redondo com '+-' é obtido a partir

do bloco com '++' dando um clique duplo sobre ele e substituindo um dos sinais por '-' na janela que aparece.

O diagrama de blocos acima representa um sistema dinâmico usado para estudar fenômenos meteorológicos. Esse sistema dá origem ao famoso atrator de Lorenz, que constitui um dos primeiros exemplos de comportamento caótico observados na natureza.

Para entender como o Simulink funciona, observe o diagrama de blocos e tente escrever xdot, ydot e zdot em função de x, y, z e b. Por exemplo, podemos seguir os sinais que dão origem a xdot para concluir que

$$\dot{x} = a(y - x) \tag{1}$$

Faça o mesmo com os demais estados.

Assim, usando blocos como integradores, ganhos, constantes, produtos e outras funções matemáticas, podemos escrever diversas equações diferenciais no Simulink. Note que as variáveis de saída dos integradores constituem a escolha natural para variáveis de estado do sistema.

A maneira como essas simulações são criadas no Simulink é reminiscente das simulações em computadores analógicos. Num computador analógico, podemos implementar integradores usando um capacitor e um amp-op, implementamos ganhos usando trimpots ou amp-ops, implementamos sinais de entrada usando geradores de sinais, implementamos somadores com amp-ops e diversos resistores. Dessa forma, cada bloco no diagrama acima poderia ser implementado num computador analógico usando um circuito apropriado.

Para facilitar a simulação de nosso modelo, crie um arquivo tut5a.m, que será usado para definir alguns parâmetros de nossa simulação. Em geral, não é necessário simular modelos do Simulink a partir de m-files, sendo bastante clicar na opção 'Start' no menu 'Simulation'. Contudo, a execução a partir de m-files nos dará certa flexibilidade na alteração de parâmetros do modelo. Começamos nosso m-file definindo os parâmetros usados no atrator de Lorenz.

```
clear;
close all;
clc;

a=10;
b=28;
c=8/3;

T=40;  %horizonte de simulacao
```

Em seguida, definimos as condições iniciais de cada integrador e executamos o modelo.

```
%modifica a propriedade InitialCondition do objeto/bloco mod5a/Integrator
set_param('mod5a/Integrator','InitialCondition','1')
set_param('mod5a/Integrator1','InitialCondition','2')
set_param('mod5a/Integrator2','InitialCondition','3')

sim('mod5a',T)   %executa o modelo por um tempo T
```

Note que também seria possível definir as condições iniciais dando um clique duplo sobre os blocos dos integradores. Quando não são definidas pelo usuário, o valor padrão usado para as condições iniciais é 0.

Após a simulação, são criados um vetor de tempo tout e um vetor de saídas yout. Cada coluna de yout corresponde a um dos blocos de saída Out1, Out2, Out3.

Usando os resultados de simulação, plotamos o diagrama de fase tridimensional que exibe o atrator de Lorenz. Plotamos ainda um dos sinais no tempo.

```
figure
plot3(yout(:,1),yout(:,2),yout(:,3))
xlabel('x')
ylabel('y')
zlabel('z')
title('Atrator Caotico de Lorenz')
grid

figure
plot(tout,yout(:,1))
xlabel('t')
ylabel('x(t)')
```

Execute o arquivo tut5a.m e observe as figuras obtidas. A Figura 2 revela um movimento persistente, mas não-periódico. Na Figura 1, experimente usar a ferramenta/botão Rotate3D para visualizar o atrator a partir de diferentes ângulos.

Adicione um comentário (usando %) ao início do arquivo para descrever o conjunto de equações diferenciais simuladas incluindo (1).

PARTE 2

Nosso próximo objetivo é simular um sistema de controle de rolamento de um satélite. Seja $\theta(t)$ o ângulo de rolamento (ângulo no eixo-z) do satélite no instante t. Um modelo simplificado para a dinâmica de rolamento é dado por

$$J\ddot{\theta} = \tau(t)$$

onde J é o momento de inércia para o eixo-z e $\tau(t)$ é o torque devido aos propulsores do satélite. A natureza dos propulsores é tal que não é possível controlar a intensidade do torque, mas apenas sua direção. Dessa forma, temos que $\tau(t)$ assume valores em $\{-M, M\}$, onde M representa o máximo torque devido aos propulsores.

Nosso objetivo é atingir $\theta(t) = \theta_{ref}$, onde θ_{ref} é um ângulo de referência necessário para a correta operação do satélite. Para isso propomos a seguinte estratégia de controle:

$$\tau(t) = \begin{cases} M & \text{se } \theta(t) < \theta_{ref} \\ -M & \text{se } \theta(t) > \theta_{ref} \end{cases}$$

ou seja, se o ângulo atual é maior que o desejado, damos propulsão no sentido de diminuir tal ângulo; se o ângulo é menor do que o desejado, damos propulsão para criar um torque positivo.

O sistema de controle descrito está implementado no arquivo mod5b.mdl. Abra o arquivo e tente entender como ele corresponde ao comportamento descrito. A comparação entre o ângulo de referência e o ângulo de rolamento é feita pelo bloco '+-' e então injetada num relé. Quando a entrada é positiva, o relé tem 1 como saída; quando a entrada é negativa, o relé tem saída -1.

Crie um arquivo tut5b.m para auxiliá-lo na execução da simulação. Iniciamos definindo os parâmetros do sistema, bem como o tipo de sinal de referência.

```
clear;
close all;
clc;

M=1000;
J=100;

T=100; %horizonte de simulacao

set_param('mod5b/Signal Generator','Frequency','0.01')
set_param('mod5b/Signal Generator','Waveform','square')
```

Em seguida realizamos a simulação e plotamos os resultados.

```
sim('mod5b',T)

figure
plot(tout,yout(:,1),tout,yout(:,3),'r')
xlabel('t')
ylabel('\theta(t)')

figure
plot(yout(:,1),yout(:,2))
hold on
plot([20 20],[min(yout(:,2)) max(yout(:,2))],'m')
plot([-20 -20],[min(yout(:,2)) max(yout(:,2))],'m')
```

Execute o arquivo e observe os resultados. Na Figura 1, vemos que o ângulo de rolamento oscila em torno do ângulo de referência, sendo a amplitude de oscilação bem maior quando o ângulo de referência é de 20^o. Na Figura 2, observamos o comportamento do sistema no espaço de fase. Em especial, notamos que o chaveamento do relé (ou dos propulsores) ocorre quando a trajetória corta as linhas em magenta, que correspondem aos valores de referência.

Algo que deve ser notado também é que a simulação é bem grosseira, tendo curvas com poucos pontos de amostragem, o que dá uma aparência de que as trajetórias são constituídas por retas. Para remediar esse problema, podemos forçar o Simulink a usar um passo de simulação menor. Isso será feito configurando o máximo passo de simulação permitido. Substitua a linha correspondente à simulação pelas linhas:

```
options=simset('MaxStep',0.1);
sim('mod5b',T,options)
```

Com esses comandos, impedimos que passos de simulação maiores que 0.1 sejam adotados. Execute o arquivo novamente e note como as trajetórias se tornaram mais suaves.

Comentário: Nos exercícios computacionais, tentaremos melhorar esse sistema de controle, que claramente não é satisfatório para a operação do satélite dado que a amplitude de oscilação é muito grande. Uma explicação para o pobre desempenho desse sistema de controle está no fato de que ele não leva em conta a velocidade angular. Quando o controlador decide chavear porque $\theta > \theta_{ref}$, é tarde demais porque o sistema já acumulou muita energia (velocidade) na direção contrária. Idealmente, queremos levar em conta quão rápido $\theta(t)$ se aproxima de θ_{ref}.

Outro ponto importante dessa aula é que esta foi a primeira vez que consideramos sistemas com descontinuidades. A descontinuidade introduzida pelo relé pode afetar a simulação de maneira significativa. Em geral, para obter-se uma boa simulação é importante identificar com precisão os instantes em que as descontinuidades ocorrem. Para isso, o Simulink usa métodos chamados de *Zero-Crossing Detection* para encontrar os pontos de descontinuidade. Como você pode imaginar, achar os instantes de descontinuidade envolve resolver equações implícitas, o que torna as simulações muito mais lentas que o habitual. Se sabemos por alguma razão que não é importante calcular certos tempos de descontinuidade, podemos tornar nossa simulação mais rápida dizendo ao Simulink que não é necessário calcular tais tempos. Para isso, bastar abrir a janela correspondente ao bloco de descontinuidade e desmarcar a opção Enable zero-crossing detection.

No exercício computacional, lidaremos com o problema oposto. Teremos um sistema sujeito a *chattering*, que consiste no chaveamento muito rápido do relé. Teoricamente, assim que o relé chaveia num certo sentido, ele cria uma condição que implica chaveamento no sentido oposto e assim por diante. Dessa forma, teremos chaveamentos instantâneos entre $+1$ e -1.

Esse chaveamento instantâneo tornará impossível para o algoritmo de simulação detectar o instante de chaveamento com precisão. O algoritmo retornará uma mensagem dizendo que há um número muito grande de cruzamentos consecutivos de zero. Para resolver esse problema, precisamos diminuir a acurácia exigida na detecção dos tempos de chaveamento. Como chaveamentos instantâneos não existem em um sistema real, não estaremos nos desviando muito do comportamento real do sistema.