Universidade Federal de Minas Gerais - Departamento de Engenharia Eletrônica

ELT062 - OFICINA DE SIMULAÇÃO ANALÓGICA E DIGITAL EM CONTROLE

TUTORIAL 2

Instrutores: Alexandre R. Mesquita e Bruno O. S. Teixeira

Parte 1

Nosso objetivo é simular o sistema multivariável chamado *Brusselator*, que descreve a seguinte reação química

$$\begin{aligned} A &\longrightarrow X_1 \\ 2X_1 + X_2 &\longrightarrow 3X_1 \\ B + X_1 &\longrightarrow X_2 + C \\ X_1 &\longrightarrow D \end{aligned}$$

Nosso foco é na evolução das espécies químicas X_1 e X_2. Assumimos que as espécies A, B, C e D são abundantes, de forma que não há modificação perceptível de suas concentrações ao longo do intervalo de tempo estudado.

Considere a terceira reação acima. Queremos modelar a quantidade de produto gerado num pequeno intervalo de tempo de duração h. O produto é gerado toda vez que duas moléculas reagentes se chocam com a quantidade certa de energia. Dessa forma, supondo que as moléculas estão suficientemente misturadas, podemos assumir que a probabilidade de uma reação ocorrer é proporcional ao número de moléculas de B vezes o número de moléculas de X_1. Dessa forma, o número médio de reações que ocorre num intervalo de tempo h é proporcional a $h[B][X_1]$ (usamos $[\cdot]$ para indicar o número de moléculas de uma determinada espécie). Como assumimos que $[B]$ permanece inalterado ao longo do tempo, podemos dizer o que número médio de produtos X_2 e C gerados pela reação num tempo h é dado por

$$b[X_1]h, \tag{1}$$

onde $b > 0$ é uma constante. De forma semelhante, o número médio de moléculas de X_1 consumidas pela quarta reação é dado por

$$d[X_1]h, \tag{2}$$

para uma constante $d > 0$. O número médio de moléculas X_1 criadas pela primeira reação é

$$ch, \tag{3}$$

para uma constante $c > 0$. Para a segunda reação, temos duas moléculas de X_1 juntando-se a uma molécula de X_2 para gerar uma nova molécula de X_1. Dessa forma, o número médio de X_2 consumido e X_1 gerado é dado por

$$a[X_1]^2[X_2], \tag{4}$$

para uma constante $a > 0$.

Dessa forma, podemos escrever a seguinte equação para a evolução do número médio de moléculas de cada espécie:

(5)

$$[X_1](kh+h) = [X_1](kh) + h(c + a[X_1](kh)^2[X_2](kh) - d[X_1](kh) - b[X_1](kh))$$
$$[X_2](kh+h) = [X_2](kh) + h(-a[X_1](kh)^2[X_2](kh) + b[X_1](kh))$$

Dedique algum tempo para entender como cada reação contribui para os termos da equação acima. Para simular a dinâmica acima, crie um m-file com nome tut2a.m. Começamos por definir a função principal e fechar as figuras já abertas:

```
function tut2a
close all
```

A seguir, definimos o tempo h, o horizonte de simulação e a condição inicial x_0.

```
h=0.03;
T=4200;
x0=[4000;300];
```

Aqui usamos um vetor cujos componentes indicam o número de moléculas de X_1 e X_2. A simulação segue como abaixo.

```
X=x0;
x=x0;
for it=1:T-1
   x=x+h*taxa(x);
   x=max(x,0);    %usado para evitar números negativos de moléculas
   X=[X x];
end
```

Aqui nós criamos uma função chamada taxa para fornecer a taxa de crescimento de cada componente em função do número de moléculas. Essa função deve ser colocada no fim do arquivo após a função tut2a.

```
function f=taxa(x)
a=1e-6;
b=2.5;
c=1e3;
d=1;
f(1,1)=c+a*x(1)^2*x(2)-(d+b)*x(1);
f(2,1)=-a*x(1)^2*x(2)+b*x(1);
```

Note que esta função tem como entrada um vetor bidimensional com o número de moléculas e como saída a taxa bidimensional de crescimento para cada espécie. Voltando à função principal, plotamos o resultado simulado como nos programas anteriores

```
tempo=((1:T)-1)*h;
figure(1)
subplot(2,1,1)
plot(tempo,X(1,:))
subplot(2,1,2)
plot(tempo,X(2,:))
```

Como nosso sistema dinâmico possui duas variáveis que evoluem no tempo, é interessante observar como essas variáveis evoluem conjuntamente no plano $[X_1] \times [X_2]$.

Para tanto, plotamos $[X_1](kh)$ no eixo-x e $[X_2](kh)$ no eixo-y. Chamamos a esse tipo de gráfico de diagrama no espaço de fase, sendo o espaço de fase o conjunto de possíveis variáveis do sistema dinâmico.

```
figure(2)
plot(X(1,:),X(2,:),'-')
xlabel('X_1')
ylabel('X_2')
title('Diagrama de fase para o Brusselator')
```

Execute o arquivo e observe o resultado da simulação. Observe o comportamento periódico do sistema. No diagrama de fase, note que a trajetória do sistema converge para uma curva fechada à qual chamamos de ciclo-limite.

Parte 2

Nosso próximo objetivo é explorar o comportamento predominantemente aleatório de reações químicas quando a concentração de moléculas é baixa. Neste caso, faz pouco sentido falar da média de moléculas produzidas ou consumidas por uma reação já que este número seria muito pequeno e não aproximaria a realidade. Assim, quando os valores dados pelas equações (1), (2), (3) e (4) acima são bem menores que 1, esses valores, em vez de representar a média de produtos gerados, são melhor interpretados como sendo a probabilidade de ocorrência de uma reação. Queremos realizar uma simulação que capture essa nova interpretação.

Em nossa simulação, precisamos que, em cada intervalo de tempo, a reação $B + X_1 \longrightarrow X_2 + C$, por exemplo, aconteça com probabilidade $b[X_1]h$. Quando a reação ocorrer, $[X_1]$ será diminuído de 1 e $[X_2]$ será aumentado de 1.

Se a probabilidade dessa reação ocorrer fosse $1/2$, o que faríamos seria "jogar uma moeda" e, caso o resultado fosse cara, incrementaríamos $[X_2]$ e decrementaríamos $[X_1]$. Como a probabilidade de uma reação pode em princípio ser qualquer número entre 0 e 1, precisamos de uma "moeda" um pouco mais sofisticada.

Para gerar eventos com probabilidade p no Matlab, usamos o seguinte artifício. Geramos um número aleatório P uniformemente distribuído no intervalo $[0, 1]$ usando a função rand. Isto significa que há uma mesma probabilidade de o número gerado ser qualquer número nesse intervalo. Então declaramos que o evento ocorre se $p > P$. Intuitivamente, se p for um número próximo de 1, haverá uma grande chance de $p > P$ e de termos a ocorrência do evento. Por outro lado, se p for muito próximo a zero, haverá uma chance muito pequena de que $p > P$ e, por conseguinte, de que o evento ocorra.

Para realizar nossa simulação, salve o arquivo tut2a.m como tut2b.m. Neste arquivo, paralelamente à simulação da média x, vamos simular xr, que corresonde ao sistema dinâmico sujeito a reações probabilísticas conforme descrito acima. Adicione a seguinte linha de código antes do for loop para definir a condição inicial dessa variável.

```
xr=x0;
Xr=x0;
```

Definimos as taxas para cada reação usando as funções abaixo, que devem ser adicionadas ao final do m-file.

```
function r=r1(x)
global c
```

```
r=c;

function r=r2(x)
global a

r=a*x(1)^2*x(2);

function r=r3(x)
global b

r=b*x(1);

function r=r4(x)
r=d*x(1);
```

Como os parâmetros a, b, c e d eram definidos apenas dentro da função taxa, precisamos torná-los variáveis globais de forma que possam ser utilizados pelas demais funções. Por isso usamos a declaração de variáveis globais no início das funções acima. Contudo, ainda temos que declarar essas variáveis como globais dentro da função taxa. Para isso, adicione a seguinte linha ao início dessa função.

```
global a b c k
```

(a variável k será introduzida posteriormente). Tendo definido as taxas de cada reação, podemos agora simular essas reações de forma probabilística usando o procedimento descrito acima. Adicione o seguinte código dentro do for loop.

```
%contabilizamos a variacao do numero de
%moleculas na variavel aux
aux=[0;0];
if rand<h*r1(xr)      %decide se 1a reacao deve ocorrer
aux=aux+[1;0];     %caso positivo, incrementamos x1
end

if rand<h*r2(xr)      %decide se 2a reacao deve ocorrer
aux=aux+[1;-1];     %caso positivo, incrementamos x1
                  % e decrementamos x2
end

if rand<h*r3(xr)      %decide se 3a reacao deve ocorrer
aux=aux+[-1;1];
end

if rand<h*r4(xr)      %decide se 4a reacao deve ocorrer
aux=aux+[-1;0];
end

%por fim, contabilizamos a variacao no numero
% de moleculas e salvamos o resultado em Xr
xr=max(xr+aux,0);
Xr=[Xr xr];
```

Queremos que os resultados de nossa simulação probabilística sejam plotados por sobre os gráficos da simulação da média. Para isso, adicione as seguintes linhas logo após o código usado para plotar a simulação da média.

```
figure(1)
subplot(2,1,1)
hold on
plot(tempo,Xr(1,:),'r')
subplot(2,1,2)
hold on
plot(tempo,Xr(2,:),'r')
figure(2)
hold on
plot(Xr(1,:),Xr(2,:),'r')
```

Nossa motivação para simular reações químicas de forma probabilística veio da necessidade de simular reações com um pequeno número de moléculas. Nosso próximo passo é criar um mecanismo para mudar a escala das equações que descrevem nosso sistema dinâmico. Para isso, definimos um fator de mudança de escala k. Queremos mudar os parâmetros de a, b, c, d de forma que a equação (5) produza os mesmos resultados mas com um número de moléculas k vezes menor. Analisando (5), vemos que esse objetivo é atingido quando redefinimos

$$a = k^2 a_0,\ b = b_0,\ c = c_0/k,\ d = d_0,$$

onde a_0, b_0, c_0 e d_0 correspondem aos parâmetros na escala original. Para efetuar essa mudança de escala, substitua as linhas que definem esses parâmetros na função `taxa` por

```
a=k^2*1e-6;
b=2.5;
c=1e3/k;
d=1;
```

Para definir o fator de mudança de escala como variável global, adicione as linhas

```
global k
k=50;
```

ao início da função principal. Nosso próximo passo é escolher h pequeno o bastante para garantir que `hr2(x)` seja menor que 1, de forma a definir uma probabilidade. Substitua as linhas que definem h, T e x_0 por

```
T=3e6/k;
h=1e-5*k;
x0=[1000;1000]/k;
```

Note como variamos o horizonte de simulação de forma que o tempo total de simulação hT seja constante.

Por fim, queremos garantir a reprodutibilidade de nossa simulação. Os números aleatórios gerados pela função `rand` são chamados de números pseudo-aleatórios porque eles apenas aparentam ser aleatórios, mas na verdade é possível prevê-los. O algoritmo usado por `rand` consiste em simular um sistema dinâmico como o *bit shift map* da aula anterior até que o resultado pareça "aleatório" e "imprevisível". Na verdade, se iniciarmos a simulação desse sistema dinâmico com a mesma condição inicial, vamos sempre obter a mesma sequência de números (pseudo)-aleatórios. O

Matlab permite que escolhamos essa condição inicial ('seed') usando o comando abaixo:

```
RandStream.setDefaultStream(RandStream('mt19937ar','seed',246));
```

Adicione esta linha ao início da função principal para definir 246 como condição inicial. Dessa forma, toda vez que executarmos nossa simulação, a função `rand` retornará os mesmos valores, o que nos permitirá reproduzir simulações com exatamente o mesmo resultado. Se não configurarmos a condição inicial, observaremos resultados de simulação diferentes cada vez que executarmos a simulação.

Execute a simulação e observe o resultado. Note como a média aproxima grosseiramente a simulação probabilística. Experimente mudar k para 100 e 200. Observe que aproximação da simulação probabilística pelas médias é muito pior, mas o comportamento oscilatório permanece. Um comportamento muito mais próximo da média seria observado para $k = 1$. Contudo, a duração dessa simulação seria muito longa para nossa prática.