Universidade Federal de Minas Gerais - Departamento de Engenharia Eletrônica

ELT062 - OFICINA DE SIMULAÇÃO ANALÓGICA E DIGITAL EM CONTROLE

TUTORIAL 1

Instrutor: Alexandre R. Mesquita

Parte 1

Nosso objetivo é simular a dinâmica populacional dada pelo *mapa logístico*

$$x[k+1] = rx[k](1-x[k]),\ k = 0, 1, 2, \ldots$$

onde $x[k]$ representa a população de uma certa espécie no ano k e r é uma constante entre 1 e 4. A equação acima modela o fenômeno de competição intraespecífica: quando há poucos indivíduos, isto é, $x[k] << 1$, a população cresce a uma taxa aproximada de r. Quando a população se torna demasiado grande ($x[k] > 1 - 1/r$), esta começa a decrescer devido à competição entre seus indivíduos. Por sinal, o valor $1 - 1/r$ é um ponto de equilíbrio populacional, isto é, se $x[0] = 1 - 1/r$, então $x[k]$ permanecerá constante neste valor para todos os anos posteriores.

Note que este se trata de um sistema autônomo (sem entradas) e que nosso interesse é estudar a resposta devido à condição inicial.

Para simular essa dinâmica, crie um script de matlab chamado `tut1a.m`. Iniciamos o script com os comandos

```
clear
close all
clc
```

usados para limpar as variáveis do *workspace*, fechar as figuras abertas e limpar a linha de comando. Em seguida definimos o valor de r, o horizonte de simulação e a população inicial.

```
r=2.7;
T=30;          %período simulado em anos
x(1)=0.95;
```

Em seguida, usamos um loop do tipo `for` para iterar os valores da população ao longo dos anos.

```
for it=1:T-1
   x(it+1)=r*x(it)*(1-x(it));
end
```

Para gerar um gráfico da população em função do tempo, definimos a variável tempo e usamos a função `stem`, apropriada para plotar sinais discretos.

```
tempo=(1:T)-1;
stem(tempo,x)
xlabel('tempo (anos)') %dando nome aos eixos
ylabel('populacao')
```

Note que o método `plot` deve ser evitado aqui porque este interpolaria entre os valores de tempo discreto. Contudo, como estamos trabalhando com um sistema a

tempo discreto, não faz sentido apresentar valores populacionais para tempos não inteiros.

Queremos comparar a solução obtida com o valor no ponto de equilíbrio populacional. Para tanto, vamos traçar uma reta indicando o valor de equilíbrio sobre o gráfico anterior.

```
xeq=1-1/r;
hold on  %mantém que o gráfico anterior ao plotarmos a nova curva
plot(tempo,xeq*ones(length(tempo),1),'r')
legend('simulacao', 'equilibrio')
```

Podemos ajustar os eixos do gráfico usando a linha

```
axis([-1 31 0 1])
```

Salvamos a figura gerada no arquivo de Matlab simulamapalogistico.fig, que poderá ser aberto posteriormente no Matlab.

```
saveas(gcf,'simumapalogistico','fig')
```

Alternativamente, podemos exportar a figura para o arquivo simulamapalogistico.png, que poderá ser usado em arquivos de texto.

```
print -dpng simumapalogistico
```

Entre outros formatos disponíveis estão pdf, tiff, jpg, eps.

Execute o script e note como a população converge para o valor de equilíbrio após um curto transitório.

Parte 2

Nosso próximo objetivo é analisar como a dinâmica populacional dada pelo mapa logístico se comporta em função do parâmetro r. Para isso, crie um script chamado tut1b.m iniciado pelos comandos

```
clear
close all
clc

r=[0.5; 1.5; 2.5; 3.2; 3.5; 3.6; 3.8; 4];
T=30;
x=0.95*ones(size(r));
```

Note que desta vez usaremos um vetor com valores de r e simularemos cada linha do vetor x independentemente de forma que cada linha corresponda a um valor de r.

```
X=x;
for it=1:T-1
   x=r.*x.*(1-x);  %note a multiplicação linha a linha .*
   X=[X x];
end
```

Outra opção seria criar um for loop para percorrer os diferentes valores de r. Contudo, essa opção será em geral mais lenta pois o Matlab é otimizado para realizar operações com vetores. Assim, sempre que possível devemos preferir operações vetoriais em detrimento de for loops.

Outra observação interessante é como geramos a matriz X por meio de concatenação de matrizes. A matriz gerada é tal que cada linha contém a simulação correspondente a um valor de r.

Em seguida, queremos plotar as órbitas simuladas para cada valor de r em diferentes gráficos numa mesma figura. Para tal, usamos o comando subplot.

```
tempo=(1:T)-1;

for i=1:length(r)
   subplot(length(r)/2,2,i)
   stem(tempo,X(i,:))
   title(['r=' num2str(r(i))])
end
```

Analise as linhas de código acima e tente entender como elas geram a figura desejada. Note o uso da função num2str, que converte uma variável numérica do tiplo double em uma variável de texto (string), que pode ser usada para dar título ao gráfico. Por fim, salvamos a figura gerada em um arquivo de formato png.

```
print -dpng simumapalogistico2
```

Execute o script. Note o comportamento periódico do sistema para diversos valores de r. Para $r = 3.2$, por exemplo, temos comportamento periódico com período 2, isto é, passado um curto transitório, a resposta se repete a cada 2 instantes de tempo. Já para $r = 2.5$, temos comportamento periódico com período 1. Tente calcular o período de oscilação para os valores de $r = 3.5, 3.6$ e 4.

Parte 3

Tendo analisado os efeitos do parâmetro r, queremos agora analisar o efeito das condições iniciais. Faremos isso para o sistema dado pela equação

$$x[k+1] = 2x[k] \mod 1$$

Lembrando que o operador $x \mod y$ retorna o resto da divisão de x por y. Na prática se $x[k]$ é um número no intervalo $[0, 1]$ representado na base binária [p. ex. $(1/4)_{10} = (0.01)_2$], o mapa acima consiste em deslocar os bits de uma casa para a esquerda (multiplicação por 2) e descartar o bit mais significativo (mod 1). Por isso esse mapa é comumente conhecido como *bit shift map*. Veremos que esse mapa pode ser utilizado para gerar números pseudo-aleatórios em computadores.

Assim como na base decimal, números racionais possuem representação com dizímas periódicas e números irracionais possuem representação com dízimas que nunca se repetem. Por consequência, se $x[0]$ for racional, $x[k]$ terá um comportamento periódico. Se $x[0]$ for irracional, $x[k]$ nunca se repetirá e, mais ainda, pode-se mostrar que $x[k]$ "visitará" praticamente todos os pontos no intervalo $(0, 1)$.

Crie um script com nome tut1c.m. Como anteriormente, vamos simular diversas realizações do sistema usando um vetor de condições iniciais. Iniciamos com

```
clear
close all
clc

T=50;
x0=[1/3;3/5;5/7;7/9;9/11];
x=[x0; x0+1e-4];
```

Temos aqui um conjunto de 10 condições iniciais, sendo que cinco pares estão separados por um valor muito pequeno. Simulamos usando as linhas

```
X=x;
for it=1:T-1
   x=mod(2*x,1);
   X=[X x];
end
```

Os gráficos são gerados de maneira semelhante ao que fizemos anteriormente.

```
tempo=(1:T)-1;
for i=1:size(x0,1)
   subplot(size(x0,1),2,2*(i-1)+1)
   stem(tempo,X(i,:))
   title(['x0=' num2str(x0(i))])
   subplot(size(x0,1),2,2*(i))
   stem(tempo,X(i+size(x0,1),:))
end

print -dpng simumabitshiftmap
```

Dedique algum tempo para entender a indexação usada acima.

Execute o script e observe o comportamento periódico para as condições iniciais "racionais". Observe que as mesmas condições iniciais perturbadas de 10^{-4} produzem resposta semelhante por cerca de 11 instantes de tempo. Daí em diante, as respostas são aparentemente aperiódicas e muito distintas das soluções não perturbadas. Para entender o que está acontecendo, basta notar que a perturbação de 10^{-4} é multiplicada por 2^{11} após 11 deslocamentos de bits, ou seja, essa perturbação passa a ter um efeito equivalente a uma perturbação de 0.2048.

Comentário

Aqui cabem duas perguntas importantes acerca da validade da simulação de sistemas dinâmicos:

(1) *O que é possível concluir a partir de uma única simulação de um sistema dinâmico se comportamentos tão diferentes são observados para diferentes condições iniciais?*

A princípio, para capturar completamente o comportamento de um sistema dinâmico, teríamos que simulá-lo para todas as condições iniciais. Isso é impraticável porque o número de condições iniciais é infinito. Infelizmente, é comum encontrar sistemas que apresentam um comportamento uniforme e estável para quase todas as condições iniciais mas são instáveis para um pequeno conjunto de condições iniciais que passariam inexploradas na maioria as simulações. Por exemplo, para o mapa logístico com $r = 1.5$, temos um comportamento de período 2 para quase todas as condições iniciais, exceto para $x_0 = 0, 1, 1/3$ e $2/3$.

Em geral, precisamos adquirir algum conhecimento teórico sobre um determinado sistema antes de confiar cegamente nas simulações do mesmo. Felizmente, para muitos sistemas é possível mostrar propriedades que apontam um número pequeno de possíveis comportamentos, o que permite confiar mais no valor das simulações. Por exemplo, sabemos que, devido à

dissipação de energia, um pêndulo simples eventualmente se estabilizará na posição de menor energia para "qualquer" condição inicial.

(2) *O que é possível concluir a partir de simulações de um sistema dinâmico se comportamentos tão diferentes são observados devido a pequenas perturbações (às quais a simulação está fatalmente sujeita devido aos arredondamentos numéricos)?*

De fato, como no caso das previsões meteorológicas, há sistemas cujo comportamento só pode ser previsto com acurácia por um curto horizonte de tempo. Esse horizonte é de 11 instantes de tempo no caso do nosso *bit shift map*. Sistemas com essa propriedade de curta previsibilidade são comumente chamados de sistemas caóticos. Contudo, se lembrarmos que sistemas reais também estão sujeitos a perturbações, podemos esperar que nossas simulações nos deem ao menos uma ideia qualitativa do comportamento do sistema. Felizmente, nem todos os sistemas dinâmicos são caóticos. Como na resposta ao item anterior, há sistemas para os quais podemos mostrar propriedades teóricas que nos garantem simulações precisas para horizontes de tempo arbitrariamente grandes.