Systems Modelling and Control: Tutorial 1

Due on March 14, 2016 at 3:10pm

Damien Hill

S.Reynolds (262538)

Problem 1

Matlab array indexing

Question 2

The line of code:

```
xx(3:7)
```

returns a slice out the original vector **xx** from index 3 to index 7.

The line of code:

```
length(xx)
```

returns the length of the vector as 12.

The line of code:

```
xx(2:2:length(xx))
```

returns a slice from the vector **xx** which starts at index 2 and takes every second value from the vector until the final element.

Question 3

The following line of code will replace every odd element in the vector with the constant -77:

```
x(1:2:end) = -77
```

Problem 2

MATLAB Script Files

Question 1

The piece of code:

```
cos(pi*kset/4)
```

uses the power of vectorisation in order to avoid using a for loop. It is one of the principal draw cards in MATLAB. Whilst computationally expensive with respect to memory, it allows for rapid prototyping.

Question 2

The following piece of code vectorises the for loop:

```
x = sin([0:7]*(pi/4))
```

Question 5

The following code was added to the script for the additional plot:

```
% Original code
t = -2:0.05:3;
x = sin(2*pi*0.789*t);
plot(t,x), grid on
title('TESTPLOT of SINUSOID')
xlabel('TIME(sec)')

% Additional 3 lines of code
hold on
x1 = cos(2*pi*0.789*t);
plot(t,x1,'r')
```

Problem 3

Functions

QUESTION 1

The function:

```
function x = cosgen(f,dur)
% cosgen fucntion to generate a cosine wave
% usage:
%   x = cosgen(f,dur)
%   f = desired frequency
%   dur = duration of the waveform in seconds

t = [0:1/(20*f):dur];
y = cos(2*pi*f*t);
```

should be written as:

```
function x = cosgen(f,dur)
% cosgen fucntion to generate a cosine wave
% usage:
%   x = cosgen(f,dur)
%   f = desired frequency
%   dur = duration of the waveform in seconds

t = [0:1/(20*f):dur];
x = cos(2*pi*f*t);
```

The problem with the code was that the output was being assigned to the variable **y**.

The function:

```
function [sum,prod] = sumprod(x1,x2)
% sumprod function to add and multiply two complex numbers
% usage:
%   [sum,prod] = sumprod(x1,x2)
%   x1 = a complex number
%   x2 = another complex number
%   sum = sum of x1 and x2
%   prod = product of x1 and x2
sum = z1 + z2;
prod = z1*z2;
```

should be written as:

```
function [sum,prod] = sumprod(x1,x2)
% sumprod function to add and multiply two complex numbers
% usage:
%   [sum,prod] = sumprod(x1,x2)
%   x1 = a complex number
%   x2 = another complex number
%   sum = sum of x1 and x2
%   prod = product of x1 and x2
sum = x1 + x2;
prod = x1*2;
```

The problem is that the variables that were specified as the function arguments weren't actually used and instead unspecified variables were used to evaluate sum and product.

Question 2

Very simply, after specifying the variables **f**, **dur**, **x1** and **x2** with numerical values, we would call the functions as follows:

```
values = cosgen(f,dur)
[value1,value2] = sumprod(x1,x2)
```

Note that the variables we specified and entered in arguments do not have the requirement to be named as they were.

Question 3

The following function is an implementation of the unit function:

$$u(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}$$

```
function u = unit(t)
% unit step function
% usage:
%   u = unit(t)
%   enter the time vector t as an arguement and the output
%   recieved will be 1 for every value of t greater than 0.
u = t>0;
```

Question 4

The following function is an implementation of the ramp function:

$$r(t) = \begin{cases} t & t \geq 0 \\ 0 & t < 0 \end{cases}$$

```
function r = ramp(t)
% ramp function
% usage:
%   r = ramp(t)
%   enter the time vector t as an arguement and the output
%   will be t for every value of t greater than 0.
r = t.*(t>0);
```

Problem 4

Using Functions

Question 1

The following script was written to produce the following plots output seen in figure 1:

```
t = linspace(-2,5,2000);

y1 = unit(t);
subplot(2,2,1)
plot(t,y1)

y2 = ramp(t);
subplot(2,2,2)
plot(t,y2)

piecewise = t.*(t>=0 & t<=2);
y3 = exp(2*t).*sin(6*t).*unit(t);
subplot(2,2,3)
plot(t,y3.*piecewise)

y4 = 2*(ramp(t+0.5)-2*ramp(t)+ramp(t-0.5));
subplot(2,2,4)
plot(t,y4)
```

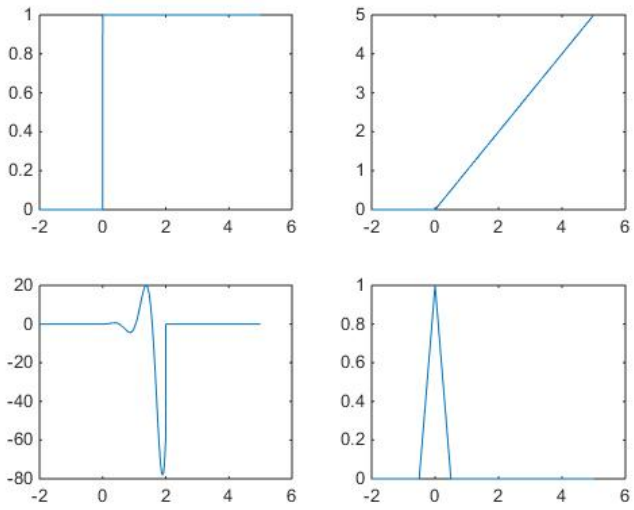

Figure 1: Plots

Problem 5

MATLAB Demos

Question 1

The following code was run with the variable **fs** set at 8000, 22050 and 44100, however, there was no discernible differences in the tones emitted.

```
dur = 1.0;
fs = 8000;
t = 0:(1/fs):dur;
x = sin(2*pi*2000*t);
soundsc(x,fs)
```

The following code was run, letting the default sample rate of 8192 take the place of the variable **fs** in the **soundsc()** function. When it was run this was the tone pitch of sound got lower.

```
dur = 1.0;
fs = 8000;
t = 0:(1/fs):dur;
x = sin(2*pi*2000*t);
soundsc(x)
```