MATLAB Tutorial Solutions:

Q1: Symbolic vs. Numeric Variables

This distinction is very important to how MATLAB performs calculations, so we need to be comfortable working with both kinds of variable. We will require both kinds depending on the question and context.

(a) Type *1/3* in the command window, and press enter. What is the output?

```
1/3
```

```
ans = 0.3333
```

This gives us a decimal approximation of 1/3, so 0.3333333333.

(b) Now, type *sym(1/3)* Do you see a different result?

```
sym(1/3)
```

```
ans =
```

$$\frac{1}{3}$$

This has returned the *precise* value of 1/3 as a fraction.

(c) Lets's store some variables. Type *x = 1/3.* Is *x* numeric or symbolic?

```
x = 1/3
```

```
x = 0.3333
```

x has been assigned the numeric value of 1/3=0.3333333333, so it is numeric.

(d) Type *y = sym(1/3).* Is *y* numeric or symbolic?

```
y = sym(1/3)
```

```
y =
```

$$\frac{1}{3}$$

This has assigned *y* to the symbolic interpretation of *1/3*, and so it is exactly *1/3.*

(e) We can obtain the value of a symbolic variable as a decimal approximation. Type *double(y)*

```
double(y)
```

```
ans = 0.3333
```

This asks for the value of *y*, but converts the output to a decimal approximation (i.e. it stated the value as a numeric variable). Note that *y* itself has not changed.

(f) Type *clear* What has this command done?

```
clear
```

This clears the workspace, so *x* and *y* have been forgotten.

(g) Store *2/3* in Matlab as a decimal approximation, stored as a variable called *z*

```
z = 2/3
```

```
z = 0.6667
```

(h) Obtain the value of *z* as a precise fraction.

```
sym(z)
```

```
ans =
```

$$\frac{2}{3}$$

(i) Store *4/3* in Matlab as a precise fraction, stored as a variable called *w*

```
w=sym(4/3)
```

```
w =
```

$$\frac{4}{3}$$

(j) Obtain the value of *w* as a decimal approximation and store it in a variable called *w2*

```
w2=double(w)
```

```
w2 = 1.3333
```

(k) We can declare a variable as symbolic without giving it a specific value. Try typing *syms p*

```
syms p
```

There is no specific output here, but note that *p* has appeared as a symbolic variable in the workspace, so now we could use it to differentiate a function of *p* for example.

Q2: Calculations

```
clear
```

(a)

```
15 + 39/2
```

```
ans = 34.5000
```

(b)

```
(15 + 39)/2
```

```
ans = 27
```

(c)

```
k = 6 * 104 - 2.^(-3)
```

```
k = 623.8750
```

(d)

```
cos(pi)
```

```
ans = -1
```

(e)

```
y = (exp(-3)+sqrt(19)).^(-2)
```

```
y = 0.0514
```

(f)

```
syms x
f = int(4*x+1,x)
```

f = $x\,(2\,x+1)$

(g)

```
syms x
int(sin(3*x),x,0,2*pi)
```

ans = 0

(h)

```
syms t
M = int(t*cos(2*t/pi),t,-5,3)
```

M =

$$\frac{3\,\pi\,\sin\left(\frac{6}{\pi}\right)}{2} - \frac{5\,\pi\,\sin\left(\frac{10}{\pi}\right)}{2} + \frac{\pi^2\cos\left(\frac{6}{\pi}\right)}{4} - \frac{\pi^2\cos\left(\frac{10}{\pi}\right)}{4}$$

```
% to get an approximate answer:
double(M)
```

```
ans = 6.4146
```

(i)

```
syms x
y = int( (3*x^2+2*x-7)/(3+sin(2*x)) , x, -pi, pi )
```

y =

$$\int_{-\pi}^{\pi} \frac{3\,x^2+2\,x-7}{\sin(2\,x)+3}\,\mathrm{d}x$$

```
double(y)
```

ans = 7.0809

Q3: Graphs and curve plotting (Part I)

```
clear
```

(a)

```
syms x
```

(b)

```
y1 = sin(x)
```

y1 = $\sin(x)$

```
y2 = sin(2*x+1)
```

y2 = $\sin(2\,x+1)$

(c)

```
fplot(x,y1)
```

(d)

```
hold on
```

(e)

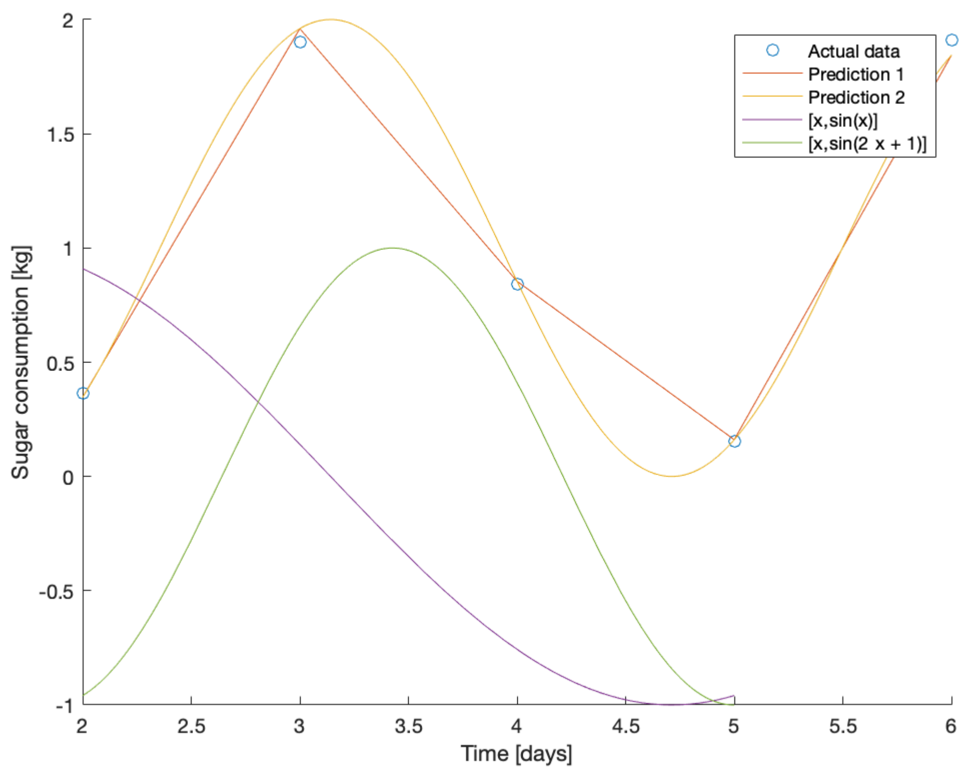
```
fplot(x,y2)
```

(f)

```
xlim([2 6])
```

(g)

```
hold off
```



(h)

```
fplot(x,y1,[2 6])
```

(i)

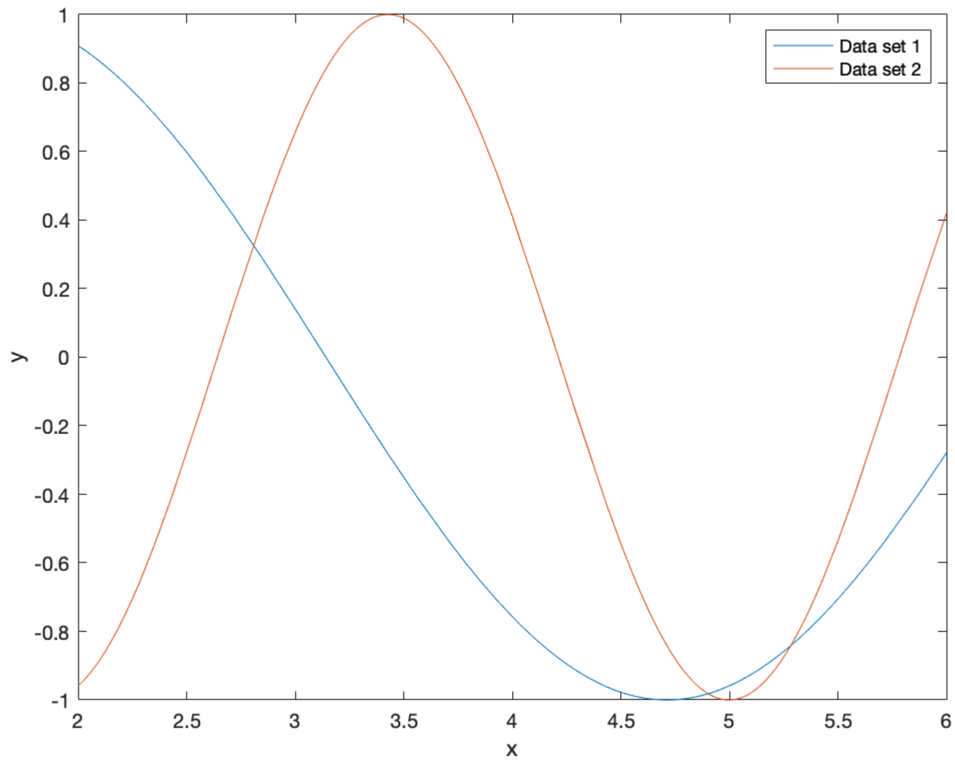
```
hold on
fplot(x,y2,[2 6])
```

(j)

```
xlabel('x')
ylabel('y')
```

(k)

```
legend('Data set 1','Data set 2')
```



Q4: Graphs and curve plotting (Part II)

```
clear
```

(a)

```
t = [1 2 3 4 5 6 7 8 9 10]
```

```
t = 1×10
     1     2     3     4     5     6     7     8     9    10
```

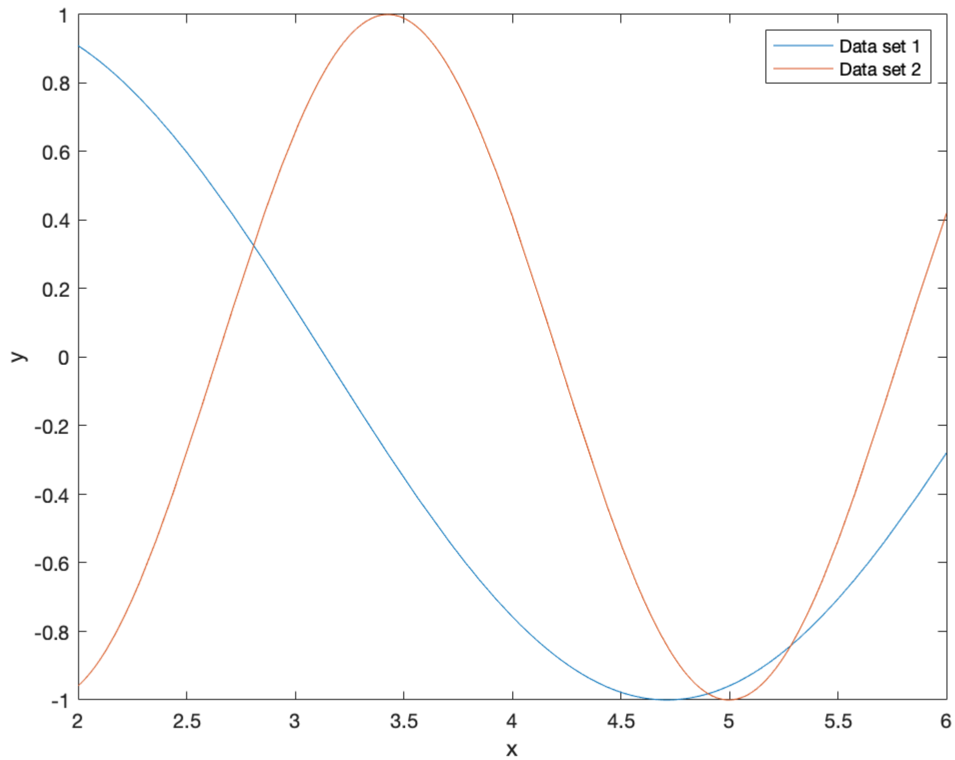
```
M = [0.573 0.366 1.900 0.841 0.155 1.910 1.187 0.040 1.703 1.411]
```

```
M = 1×10
    0.5730    0.3660    1.9000    0.8410    0.1550    1.9100    1.1870    0.0400 ⋯
```

(b) Including the semicolon ; means that the variable M will not be printed to the command window. This is especially useful if you are inputting very large data sets, and want to easily see the previous commands you've written without scrolling through many lines of data.

(c) We need to use "hold off" first, so that Matlab doesn't superimpose this scatter plot on our graph from Q2!

```
hold off
```




```
scatter(t,M)
```

(d)

```
Mpred = cos(2*t)+1
```

```
Mpred = 1×10
    0.5839    0.3464    1.9602    0.8545    0.1609    1.8439    1.1367    0.0423 ⋯
```

```
hold on
plot(t,Mpred)
```

(e)

```
t2 = linspace(1, 10, 1000)
```

```
t2 = 1×1000
    1.0000    1.0090    1.0180    1.0270    1.0360    1.0450    1.0541    1.0631 ⋯
```

```
Mpred2 = cos(2*t2)+1
```

```
Mpred2 = 1×1000
    0.5839    0.5675    0.5514    0.5353    0.5195    0.5037    0.4882    0.4728 ⋯
```

```
plot(t2, Mpred2)
```

(f)

```
legend('Actual data','Prediction 1','Prediction 2')
```

```
xlabel('Time [days]')
ylabel('Sugar consumption [kg]')
```