2 Introduction to Matlab

2.1 Hands-on tutorial

In this tutorial you will learn how to set up and evaluate a function with multiple arguments. Exercises demonstrate the use of fzero (root finding) and quad (numerical integration), with basic data manipulation and display commands. The exercises that follow apply the commands and methodologies to problems very frequently encountered in material and energy balancing.

In this course, it will be sufficient to include *all* code (main program and functions or sub-routines) in one program file, which itself must be a function and have a ".m" extension. Therefore, the first line of any Matlab program (function) you write will most often begin with a declaration such as

```
function myprogram
```

which must be saved in a text file with a ".m" extension, *e.g.*,

```
myprogram.m
```

Functions or "sub-routines" as they are called in FORTRAN, must be appended at the end of this file, *i.e.*, following the main program commands entered after the first line. However, if you write a program that does not require any sub-routines, then you need not make this main program a function, *i.e.*, you will not need to begin the program with a function declaration. Finally, while it is not necessary, sometimes you may find it helpful to put sub-routines (functions) in their own files (*e.g.*, heat capacity correlations and saturated vapour pressure formulas), especially if you anticipate using these in other programs.

Here, we will consider the function:

$$y = \sin(kx - \omega t),$$

where $k = 2\pi/L$ and $\omega = 2\pi/T$ are the wavenumber and angular frequency, with x and t the position and time, respectively. We will consider x and t as *variables*, and k and ω as *parameters*.

- If we want to evaluate this function in Matlab for various values of x and t, we need only pass x and t as arguments:

```
function y=wave(x,t)

k=2.0*pi/1.0 % gives unit wavelength
omega=2.0*pi/1.0 % gives unit period
y=sin(k*x-omega*t);

return
```

  Note that the foregoing function may be placed either at the end of your main program or in a separate file named

```
wave.m
```

- To produce a table of this function for various values of x and t, set up vectors of x and t as follows:

```
x=linspace(-5,5,10)
t=linspace(0,1,10)
```

What do the three arguments of linspace do? Vary them and examine the output, and try the command-line help facility.

Note, the foregoing and subsequent commands may be typed at the command line or placed (and subsequently edited) in your main program.

- Now evaluate the function using two "for" loops (you needn't type the "comments" below):

```
for i=1:length(t) % do t second
   for j=1:length(x) % do x first
      ymat(i,j)=wave(x(j),t(i)) % this a matrix whose first index
    % identifies the position, and whose second index identifies the time.
   end
end
```

While the foregoing and subsequent commands could be typed at the command line, it is expedient to place them in your main program.

Check the result by printing the values in ymat:

```
for i=1:length(t) % do t first
   for j=1:length(x) % do x next
      sprintf('value of the function wave for...
      x=%e and t=%e is ymat=%e\n', x(j), t(i), ymat(i,j))
   end
end
```

- To plot just the first column or row of ymat, execute

```
plot(x,ymat(:,1),'o')
```

Try

```
plot(x,ymat(1,:),'o')
```

What happens with

```
plot(t,ymat(1,:),'o')
```

Why?

Now try

```
hold on
for i=1:length(t)
   plot(x,ymat(i,:),'o-')
end
```

- Let's answer the following question: What value of x at $t = 10$ yields $y = 0$?

  Execute

  ```
  x0=fzero(@(x) wave(x,10),0)
  ```

  Why does

  ```
  x0=fzero(@(x) wave(x,10),10.2)
  ```

  give a different answer?

- What about finding a value of t at $x = 5$ that yields $y = 0$?

  ```
  x0=fzero(@(t) wave(5,t),1.7)
  ```

- Evaluate the integral of function "wave" with respect to x from $x = 0$ to 0.5, with $t = 0$:

  ```
  quad(@(x) wave(x,0),0,0.5)
  ```

  How close is the answer to the exact result? Hint: Do the integration by hand.

- Let's see how well quad does when trying to integrate this periodic function over a range that spans many periods:

  ```
  xp=linspace(0,2,100) % let's integrate from zero to 2,
  % at 100 points between

  num=xp % this will store the numerical values from quad

  for i=1:length(xp)
     num(i)=quad(@(x) wave(x,0),0,xp(i))
  end
  ```

- Plot the result in a new figure:

```
figure(2)
plot(xp, num, 'o')
```

- Now compare the numerical solution with the exact one:

```
exa=xp % this will store the exact result
%(by doing the integration analytically)
% exact answer is -cos(2*pi*x)/(2*pi)+1/(2*pi).

for i=1:length(xp)
   exa(i)=-cos(2*pi*xp(i))/(2*pi)-(-cos(2*pi*0)/(2*pi))
end
hold on
plot(xp, exa, '-')
```

- Try re-running the integration, extending the integration range out to 10 or 20. What happens?

  Answer: Numerical errors are accumulated, and, in general, these build up with a wider range of integration. In general, Matlab does an impressive job of minimizing these these errors. With this periodic function, you will see it occasionally makes a huge error (and it tells you), but seems to fix the problem. With non-periodic functions, like heat-capacity relationships, the integration will be much more reliable.

- Try integrating $y = x + x^2 + x^3$ from 10 to 100:

```
intnum=quad(@fofx,10,100)
```

  and compare the result with the exact value:

```
intexa=0.5*100^2+1/3*100^3+1/4*100^4-...
0.5*10^2+1/3*10^3+1/4*10^4
```

  The relative error is

```
(intnum-intexa)/intnum*100
```

  percent.

  You will need to add the following function to execute the numerical integration above:

```
function y=fofx(x)
y=x+x.^2+x.^3;
return
```

2.2 Exercises

As instructed in the tutorial, you are to complete the following exercises. These demonstrate many of the useful tasks required to successfully complete the much more extensive computing projects later in the term. Note than any answers provided below may contain minor, but important, errors, such as sign errors and incorrect indices, etc. Consider whether your own code is shorter/longer, easier/more difficult to modify/follow, or faster/slower to execute? Your group is to provide a hard-copy of its code and an accompanying printout of the output as proof that is provides the correct answers.

1. What is the purpose of the period (.) preceding *, ^and / ?
2. Write a program that integrates the heat capacity relationship for water vapour (in your text book) from $T = 80°$C to 320°C. Compare the change in enthalpy with what you get when using the steam tables.
3. Write a program that solves the Antoine vapour pressure relationship for water (in your text book), giving the boiling temperature of water at pressures of 0.5, 0.75, 1 and 2 atm.
4. Write a program that plots the heat capacity of water vapour [kJ/(kg K)] as a function of temperature (°C), and try to identify the range of temperatures where the heat capacity equation (polynomial) is likely to be accurate. Augment your graph with a title, y- and x-axis labels with name, symbol and units: *e.g.*, temperature, T (deg C). Next, use the heat capacity of water vapour to calculate the specific enthalpy, relative to liquid water at $T = 0°$C (reference state) as a function of temperature. Produce a list/table of the relative enthalpy at 10 temperatures above 0°C, equally spaced 10°C apart, *i.e.*, at 0, 10, 20, ... , 100°C. Show the results with units of kJ/kg and compare them with values listed in the steam tables. Why are there differences?
5. Using fzero, solve

   $$a/(x-10) + bx^2 - 1/[c\ln(x)] = dx$$

   for x, where $a = 1$, $b = 2$, $c = 3$ and $d = 4$.

There are many ways to achieve the correct answers. While some are very quick and difficult to modify and understand at a later date, others are longer but have the advantage of being easy to use and reuse. You will need to find a compromise between the time required to program a solution (and debug the program), and the functionality, considering, for example, whether you will reuse the program (or parts of it). The examples below adopt various degrees of effort, demonstrating different programming styles based on material introduced in the hands-on tutorial. There are minor extensions, which you can learn more about by using the "help" command, *e.g.*, help plot.

```
function matlabexercises

% Question 2

T1=80+273;
T2=320+273;
delh=quad(@cpwatervap,T2,T1)
```

```
sprintf('int cp(T) dt = %e J/mol\n',delh)
sprintf('int cp(T) dt = %e J/kg\n',delh/18)

% direct from temperature listing
H1=2646;
% linear interpolation from pressure listing
H2=-2709.3+(2699.5-2709.3)/(321.4-318.0)*(320-318.0);
sprintf('value from steam tables = %e - %e = %e kJ/kg\n', H2, H1, H2-H1)

% Question 3

p=[0.5 0.75 1 2]*101.3 % kPa

for i=1:3
btemp(i)=fzero(@(T) zero(p(i),T), 300);
end
answer=[p' btemp']
save -ascii output.txt answer % if you want to plot in excel, for example
load output.txt
% let's load it back and plot in matlab
who % what variable name did matlab give to the file we loaded?
t=output(:,1)
p= output(:,2)
plot(p/101.3,t-273,'-o')
ylabel('boiling temperature (degC)')
xlabel('pressure (atm)')

% Question 4

temp=linspace(-1000,3000,1000) % deg C
cp=cpwatervap(temp+273)
figure(2)
plot(temp,cp/18)
ylabel('heat capacity, cp [J/(kgK)]')
xlabel('temperature, T (deg C)')
sprintf('Looks like range 0 to about 1500 deg C is reasonable!\n')

temp=linspace(0,100,11) % deg C
for i=1:11
deltah(i)=quad(@cpwatervap,0+273,temp(i)+273)/18
end
table=[temp' deltah'+2501.6]
for i=1:11
sprintf('%0.5g %0.5g\n', temp(i), deltah(i)-2501.6)
end
```

```
% Question 5

x=linspace(1e-6,12,10000)
figure(3)
plot(x,zerop(x))
sprintf('This non-linear function has multiple roots in the range!\n')
root1=fzero(@zerop,0.9)
root2=fzero(@zerop,2)

function y=zerop(x)
a=1;
b=2;
c=2;
d=4;
y=a./(x-10)+b.*x.^2-1./(c.*log(x))-d.*x;
return

function cp=cpwatervap(T)
% returns cp in J/mol with T in K
a=3.4047e+01;
b=-9.65064e-03;
c=3.2998e-05;
d=-2.04467e-08;
e=4.30228e-12;
cp=a+b.*T+c.*T.^2+d.*T.^3+e.*T.^4;
return

function psat=psatwater(T)
% returns psat in kPa with T in K
a=16.5362;
b=3985.44;
c=-38.9974;
psat=a-b./(T+c);
psat=exp(psat);
return

function y=zero(p,T)
y=p-psatwater(T);
return
```