```
% Output argument is:
% xyout: The values of x and y at x=a, x=(a+b)/2, and x=b
% listed in a 3 by 2 matrix.

x=linspace(a,b,100);
y=feval(Fun,x);
xyout(1,1)=a; xyout(2,1)=(a+b)/2; xyout(3,1)=b;
xyout(1,2)=y(1);
xyout(2,2)=feval(Fun,(a+b)/2);
xyout(3,2)=y(100);
plot(x,y)
xlabel('x'), ylabel('y')
```

Using the imported function to calculate $f(x)$ at 100 points.

Using the imported function to calculate $f(x)$ at the midpoint.

Passing a user-defined function into another function by using a string expression:

The following demonstrates how to pass a user-defined function into a function function by typing the name of the imported function as a string in the input argument. The function $f(x) = e^{-0.17x}x^3 - 2x^2 + 0.8x - 3$ from Section 7.9.1, created as a user-defined function named `Fdemo`, is passed into the user-defined function `funplotS`. Note that the name `Fdemo` is typed in a string for the input argument `Fun` in the user-defined function `funplotS`.

```
>> ydemoS=funplotS('Fdemo',0.5,4)
ydemoS =
    0.5000   -2.9852
    2.2500   -3.5548
    4.0000    0.6235
```

The name of the imported function is typed as a string.

In addition to the display of the numerical output in the Command Window, the plot shown in Figure 7-3 is displayed in the Figure Window.

7.10 SUBFUNCTIONS

A function file can contain more than one user-defined function. The functions are typed one after the other. Each function begins with a function definition line. The first function is called the primary function and the rest of the functions are called subfunctions. The subfunctions can be typed in any order. The name of the function file that is saved should correspond to the name of the primary function. Each of the functions in the file can call any of the other functions in the file. Outside functions, or programs (script files), can call only the primary function. Each of the functions in the file has its own workspace, which means that in each the variables are local. In other words, the primary function and the subfunctions cannot access each other's variables (unless variables are

declared to be global).

Subfunctions can help in writing user-defined functions in an organized manner. The program in the primary function can be divided into smaller tasks, each of which is carried out in a subfunction. This is demonstrated in Sample Problem 7-4.

Sample Problem 7-4: Average and standard deviation

Write a user-defined function that calculates the average and the standard deviation of a list of numbers. Use the function to calculate the average and the standard deviation of the following list of grades:
80 75 91 60 79 89 65 80 95 50 81

Solution

The average x_{ave} (mean) of a given set of n numbers $x_1, x_2, \ldots, x_n$ is given by:

$$x_{ave} = (x_1 + x_2 + \ldots + x_n) / n$$

The standard deviation is given by:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{i=n} (x_i - x_{ave})^2}{n-1}}$$

A user-defined function, named `stat`, is written for solving the problem. To demonstrate the use of subfunctions, the function file includes `stat` as a primary function, and two subfunctions called `AVG` and `StandDiv`. The function `AVG` calculates x_{ave}, and the function `StandDiv` calculates σ. The subfunctions are called by the primary function.The following listing is saved as one function file called `stat`.

```
function [me SD] = stat(v)                          The primary function.
n=length(v);
me=AVG(v,n);
SD=StandDiv(v,me,n);

function av=AVG(x,num)                              Subfunction.
av=sum(x)/num;

function Sdiv=StandDiv(x,xAve,num)                  Subfunction.
xdif=x-xAve;
xdif2=xdif.^2;
Sdiv= sqrt(sum(xdif2)/(num-1));
```

The user-defined function `stat` is then used in the Command Window for calculating the average and the standard deviation of the grades: