A function function, which accepts another function (imported function), includes in its input arguments a name that represents the imported function. The imported function name is used for the operations in the program (code) of the function function. When the function function is used (called), the specific function that is imported is listed in its input argument. In this way different functions can be imported (passed) into the function function. There are two methods for listing the name of an imported function in the argument list of a function function. One is by using a function handle (Section 7.9.1), and the other is by typing the name of the function that is being passed in as a string expression (Section 7.9.2). The method that is used affects the way that the operations in the function function are written (this is explained in more detail in the next two sections). Using function handles is easier and more efficient, and should be the preferred method.

7.9.1 Using Function Handles for Passing a Function into a Function Function

Function handles are used for passing (importing) user-defined functions, built-in functions, and anonymous functions into function functions that can accept them. This section first explains what a function handle is, then shows how to write a user-defined function function that accepts function handles, and finally shows how to use function handles for passing functions into function functions.

Function handle:

A function handle is a MATLAB value that is associated with a function. It is a MATLAB data type and can be passed as an argument into another function. Once passed, the function handle provides means for calling (using) the function it is associated with. Function handles can be used with any kind of MATLAB function. This includes built-in functions, user-defined functions (written in function files), and anonymous functions.

- For built-in and user-defined functions, a function handle is created by typing the symbol @ in front of the function name. For example, `@cos` is the function handle of the built-in function `cos`, and `@FtoC` is the function handle of the user-defined function `FtoC` that was created in Sample Problem 7-2.
- The function handle can also be assigned to a variable name. For example, `cosHandle=@cos` assigns the handle `@cos` to `cosHandle`. Then the name `cosHandle` can be used for passing the handle.
- As anonymous functions (see Section 7.8.1), their name is already a function handle.

Writing a function function that accepts a function handle as an input argument:

As already mentioned, the input arguments of a function function (which accepts another function) includes a name (dummy function name) that represents the imported function. This dummy function (including a list of input arguments enclosed in parentheses) is used for the operations of the program inside the function function.

- The function that is actually being imported must be in a form consistent with the way that the dummy function is being used in the program. This means that both must have the same number and type of input and output arguments.

The following is an example of a user-defined function function, named `funplot`, that makes a plot of a function (any function $f(x)$ that is imported into it) between the points $x = a$ and $x = b$. The input arguments are `(Fun,a,b)`, where `Fun` is a dummy name that represents the imported function, and `a` and `b` are the end points of the domain. The function `funplot` also has a numerical output `xyout`, which is a 2×3 matrix with the values of x and $f(x)$ at the three points $x = a$, $x = (a+b)/2$, and $x = b$. Note that in the program, the dummy function `Fun` has one input argument `(x)` and one output argument `y`, which are both vectors.

A name for the function that is passed in.

```
function xyout=funplot(Fun,a,b)
% funplot makes a plot of the function Fun which is passed in
% when funplot is called in the domain [a, b].
% Input arguments are:
% Fun:  Function handle of the function to be plotted.
% a:  The first point of the domain.
% b:  The last point of the domain.
% 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=Fun(x);
xyout(1,1)=a; xyout(2,1)=(a+b)/2; xyout(3,1)=b;
xyout(1,2)=y(1);
xyout(2,2)=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.

As an example, the function $f(x) = e^{-0.17x}x^3 - 2x^2 + 0.8x - 3$ over the domain $[0.5, 4]$ is passed into the user-defined function `funplot`. This is done in two ways: first by writing a user-defined function for $f(x)$, and then by writing $f(x)$ as an anonymous function.

Passing a user-defined function into a function function:

First, a user-defined function is written for $f(x)$. The function, named `Fdemo`, calculates $f(x)$ for a given value of x and is written using element-by-element operations.

```
function y=Fdemo(x)
y=exp(-0.17*x).*x.^3-2*x.^2+0.8*x-3;
```

Next, the function `Fdemo` is passed into the user-defined function function `funplot`, which is called in the Command Window. Note that a handle of the user-defined function `Fdemo` is entered (the handle is `@Fdemo`) for the input argument `Fun` in the user-defined function `funplot`.

```
>> ydemo=funplot(@Fdemo,0.5,4)
ydemo =
    0.5000   -2.9852
    2.2500   -3.5548
    4.0000    0.6235
```

Enter a handle of the user-defined function `Fdemo`.

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

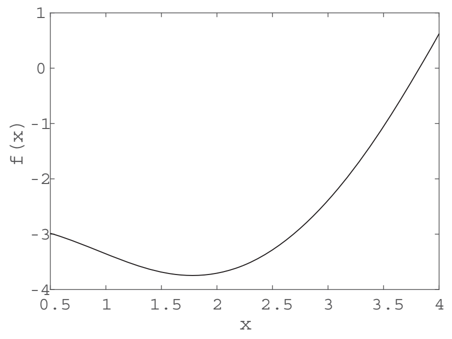

Figure 7-3: A plot of the function $f(x) = e^{-0.17x}x^3 - 2x^2 + 0.8x - 3$.

Passing an anonymous function into a function function:

To use an anonymous function, the function $f(x) = e^{-0.17x}x^3 - 2x^2 + 0.8x - 3$ first has to be written as an anonymous function, and then passed into the user-defined function `funplot`. The following shows how both of these steps are done in the Command Window. Note that the name of the anonymous function `FdemoAnony` is entered without the @ sign for the input argument `Fun` in the user-defined function `funplot` (since the name is already the handle of the anonymous function).