```
>> FdemoAnony=@(x) exp(-0.17*x).*x.^3-2*x.^2+0.8*x-3
FdemoAnony =
    @(x) exp(-0.17*x).*x.^3-2*x.^2+0.8*x-3

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

Create an anonymous function for $f(x)$.

Enter the name of the anonymous function (`FdemoAnony`).

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.

7.9.2 Using a Function Name for Passing a Function into a Function Function

A second method for passing a function into a function function is by typing the name of the function that is being imported as a string in the input argument of the function function. The method that was used before the introduction of function handles can be used for importing user-defined functions. As mentioned, function handles are easier to use and more efficient and should be the preferred method. Importing user-defined functions by using their name is covered in the present edition of the book for the benefit of readers who need to understand programs written before MATLAB 7. New programs should use function handles.

When a user-defined function is imported by using its name, the value of the imported function inside the function function has to be calculated with the `feval` command. This is different from the case where a function handle is used, which means that there is a difference in the way that the code in the function function is written that depends on how the imported function is passed in.

The `feval` command:

The `feval` (short for "function evaluate") command evaluates the value of a function for a given value (or values) of the function's argument (or arguments). The format of the command is:

```
variable = feval('function name', argument value)
```

The value that is determined by `feval` can be assigned to a variable, or if the command is typed without an assignment, MATLAB displays `ans =` and the value of the function.

- The function name is typed as string.
- The function can be a built-in or a user-defined function.
- If there is more than one input argument, the arguments are separated with commas.

- If there is more than one output argument, the variables on the left-hand side of the assignment operator are typed inside brackets and separated with commas.

Two examples using the `feval` command with built-in functions follow.

```
>> feval('sqrt',64)
ans =
     8
>> x=feval('sin',pi/6)
x =
    0.5000
```

The following shows the use of the `feval` command with the user-defined function `loan` that was created earlier in the chapter (Figure 7-2). This function has three input arguments and two output arguments.

A $50,000 loan, 3.9% interest, 10 years.

```
>> [M,T]=feval('loan',50000,3.9,10)
M =
        502.22
```

Monthly payment.

```
T =
      60266.47
```

Total payment.

Writing a function function that accepts a function by typing its name as an input argument:

As already mentioned, when a user-defined function is imported by using its name, the value of the function inside the function function has to be calculated with the `feval` command. This is demonstrated in the following user-defined function function that is called `funplotS`. The function is the same as the function `funplot` from Section 7.9.1, except that the command `feval` is used for the calculations with the imported function.

A name for the function that is passed in.

```
function xyout=funplotS(Fun,a,b)
% funplotS makes a plot of the function Fun which is passed
in
% when funplotS is called in the domain [a, b].
% Input arguments are:
% Fun: The function to be plotted. Its name is entered as
string expression.
% 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=feval(Fun,x);   Using the imported function to calculate f(x) at 100 points.
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);   Using the imported function to
xyout(3,2)=y(100);               calculate f(x) at the midpoint.
plot(x,y)
xlabel('x'), ylabel('y')
```

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 =                            The name of the imported
    0.5000   -2.9852                function is typed as a string.
    2.2500   -3.5548
    4.0000    0.6235
```

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