

More on functions in Matlab: Passing functions as arguments

By Gilberto E. Urroz, September 2004

Passing functions as arguments to other functions in Matlab

We indicated that there are two basic types of user-defined Matlab functions, namely, (1) inline functions, and (2) file functions. We also indicated that inline functions are single-line functions that can be defined with a string, while file functions are separate *m* files.

When passing functions as arguments of other functions we have used the following:

- For inline functions: type the name of the function
- For *m*-file functions: type the name of the function between quotes (which requires that the name of the *m* file be the same name as the function)

These are not, however, all the ways that functions can be passed as arguments of other functions in Matlab. *Moler, C. (2002)* in page 9, Chapter 4, of his book *Numerical Computing with Matlab*, points out five different forms of passing a function as argument of another function in Matlab 7.0. These are:

- **Inline object:** same as inline functions. The name of an inline object or inline function is typed without quotes when used as argument.
- **Name of an *m*-file:** the name of an *m*-file function must be typed between quotes when used as arguments.
- **Function handle:** a function handle uses the character '@' preceding the name of a pre-defined or *m*-file function, e.g., @sin, @f00. (This method does not work in *Matlab SE 5.3*)
- **Expression string:** instead of defining an inline object or an *m*-file function, you can simply type the string that defines the function, between quotes, and use it as an argument.
- **Symbolic expression:** similar to expression strings, but used specifically with symbolic functions, e.g., *solve* or *dsolve*, etc.

Examples of the different ways of passing a function as arguments are shown next. The functions in some of the following examples are passed as arguments of function *fzero*.

Inline object

To find the solution to the equation $e^{-x} \sin(x/2) = 0.75$, we create the function

$$f_{00}(x) = e^{-x} \sin(x/2) - 0.75,$$

and proceed to solve $f_{00}(x) = 0$, using function *fzero* as follows (using Matlab SE 5.3). The initial guess for the solution is $x = 2.5$:

```
» f00 = inline('exp(-x)*sin(x/2)-0.75')
f00 =
    Inline function:
    f00(x) = exp(-x)*sin(x/2)-0.75

» fzero(f00,2.5)
Zero found in the interval: [-6.551, 8.9].

ans = -6.2860
```

If the expression to be solved contains more than one variable, we can always pass the other variables to *fzero* as illustrated next. Suppose that the equation to solve is:

$$f_{01}(x) = A e^{-x/L} \sin(x/M) - R,$$

with $A = 2$, $L = 5$, $M = 3$, and $R = 1.5$. Here is a way to solve for x starting with an initial guess of 1.2:

```

» f01 = inline('A*exp(-x/L)*sin(x/M)-R','x','A','L','M','R')

f01 =
    Inline function:
    f01(x,A,L,M,R) = A*exp(-x/L)*sin(x/M)-R

» fzero(f01,1.2,[],2,5,3,1.5)
Warning: Cannot determine from calling sequence whether to use new or
grandfathered FZERO function. Using new; if call was grandfathered
FZERO syntax, this may give unexpected results.

> In C:\MATLAB_SE_5.3\toolbox\matlab\funfun\fzero.m (parse_call) at line 393
   In C:\MATLAB_SE_5.3\toolbox\matlab\funfun\fzero.m at line 104
Zero found in the interval: [-11.088, 9.8889].

ans = -9.7458

```

When passing more than one argument with a function into function *fzero*, *Matlab SE 5.3* always produces a warning as shown above. As an alternative, you can replace all the known numerical values in the expression defining the inline object (or inline function) using strings, as shown next:

```

» s1 = 'A*exp(-x/L)*sin(x/M)-R'
s1 = A*exp(-x/L)*sin(x/M)-R

» s1 = strrep(s1,'A',num2str(2))
s1 = 2*exp(-x/L)*sin(x/M)-R

» s1 = strrep(s1,'L',num2str(5))
s1 = 2*exp(-x/5)*sin(x/M)-R

» s1 = strrep(s1,'M',num2str(3))
s1 = 2*exp(-x/5)*sin(x/3)-R

» s1 = strrep(s1,'R',num2str(1.5))
s1 = 2*exp(-x/5)*sin(x/3)-1.5

» f02 = inline(s1,'x')
f02 =
    Inline function:
    f02(x) = 2*exp(-x/5)*sin(x/3)-1.5

» fzero(f02,1.2)
Zero found in the interval: [-11.088, 9.8889].
ans = -9.7458

```

With this approach, the warning received earlier is no longer there. Also, such warning is not produced when using *Matlab 7.0*, e.g., for function *f01*:

```

>> fzero(f01,1.2,[],2,5,3,1.5)
ans = -9.7458

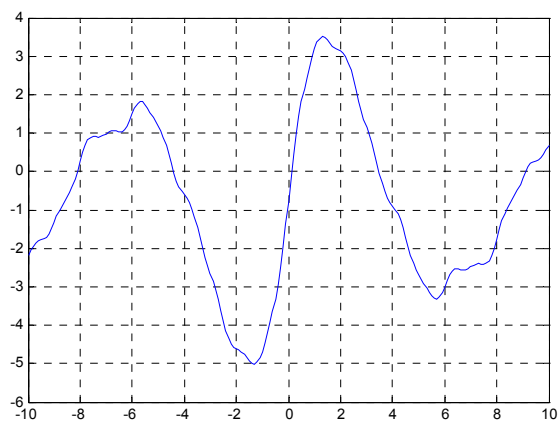
```

To illustrate passing function handles or m-file names as arguments, we create the following function as an *m* file in Matlab (the file is named *fex1.m*):

```
function [w] = fex1(x)
w = zeros(size(x));
for j = 1:10
    w = w + j/10*sin(2*pi*x/j);
end;
w = w - 0.75*ones(size(x));
```

To show the behavior of the function, we produce the following plot:

```
>> x = [-10:0.1:10]; y = fex1(x);
>> plot(x,y);
>> grid on;
```



We will use the function *fex1* as argument of function *fzero* in the next couple of examples.

Function handle

Here are the commands for finding a zero for function $fex1(x) = 0$ in *Matlab 7.0* using a function handle as argument:

```
>> fzero(@fex1,1.2)
ans = 0.1211
```

M-file name

Here are the commands for finding a zero for function $fex1(x) = 0$ in *Matlab 7.0* using a function handle as argument:

```
>> fzero('fex1',1.2)
ans = 0.1211
```

Expression string

In the following example, we use an expression string as argument of the function *fzero*:

```
>> fzero('x^3-3*x+6',0.2)
ans = -2.3553
```

Symbolic expression

If a variable name, say *x*, has been declared as symbolic (by using, for example, *syms x*), any expression involving *x* is a symbolic expression and it can be passed on as argument to a symbolic function. Here is an example:

```
>> syms x % Declare variable as symbolic
>> c = [2, 3, -1]; % Coefficients of a polynomial
>> p = poly2sym(c,'x') % Convert vector to a symbolic polynomial

p =
2*x^2+3*x-1

>> solve(p,'x') % Use symbolic function 'solve' to solve for x

ans =
-3/4+1/4*17^(1/2)
-3/4-1/4*17^(1/2)

>> pretty(ans) % "Pretty" print for solution

[ 1/2]
[- 3/4 + 1/4 17 ]
[ ]
[ 1/2]
[- 3/4 - 1/4 17 ]
```

In some cases, you can pass a symbolic expression to numerical functions, for example:

```
>> syms x
>> fzero('x^5+sqrt(4)*x-1/(1+sqrt(3))',1.2)
ans = 0.1829
```

However, the following is not permitted:

```
>> syms x
>> c = [2, 3, -1];
>> p = poly2sym(c,'x')
p = 2*x^2+3*x-1

>> fzero(p,1.2)
??? Error using ==> error
When the first input is a message identifier, the second input
must be a string.

Error in ==> fzero at 157
error('MATLAB:fzero:InvalidFUN',msg)
```