Parameter Passing in Matlab

Many of the Matlab "function functions" (such as fzero, fminbnd, and quad) are built for functions of a single variable. Unfortunately, there are many situations – often called "parameter testing" – in which you would like to use a function of several variables. Although the functions have more than one variable, the action of the Matlab procedure is with respect to just one of those variables. Thus, fzero finds a zero with respect to just one variable, fminbnd minimizes with respect to just one variable, and quad integrates in just one variable.

The need is fairly common and Matlab has allowed for it. The only difficulty is that the means by which Matlab allows function names to be used as parameters is changing. Furthermore, under the new scheme, there are two approaches – making a total of three if we count old and new.

1. Old:

Suppose you wanted to integrate a function $testf(x,\beta,z) = \sin(\beta z x)/(x-z)^{\beta}$ with respect to x. You need only write a Matlab function:

```
function y = testf (x, beta, z)
y =sin (beta*z*x)./(x-z).^beta;
```

(Notice the "./" and ".^". The procedure quad requires a function that acts on an array x. That artifact has nothing to do with the issue of sending multiple parameters in.).

Then to get $\int_{\beta}^{2\beta} \sin(\beta z x)/(x-z)^{\beta}\,dx$ you invoke quad as

```
quad(@testf, beta, 2*beta, [], [], beta, z)
```

The bracket pairs are necessary to skip you over to the sixth and seventh positions - that's where the extra parameters begin. Notice also that the variable of integration (in this case x) must be the first variable in the function specification and the others must follow in the same order they occur (i.e., the first variable in testf is x and since beta precedes z in testf, they are specified in the same order in the invocation of quad). Obviously beta and z must be defined prior to this usage of quad.

2. New (Nested Functions):

I see this as pretty convoluted and include it only to be complete. The idea is that you embed the definition of the function to be used inside a special function that also invokes the "function function". (Wasn't that clear?) The definition of the function actual uses just one variable – the others are just local variables at the time. It is called "nested functions" here but elsewhere it may be called an "internal function".

If we continue with our example from above, we could do it this way:

```
function r = parameterIntegrate (beta, z)
%
r = quad(@testf, beta, 2*beta);
%
function y = testf (x)
y =sin (beta*z*x)./(x-z).^beta;
end
end
```

Notice here the pair of "ends". You need the one to close off the internal function testf– otherwise the boundary between the internal and the external ones get blurred. You need the second one because if you ever close an internal function with an end, Matlab insists that you close the outer one similarly.

3. New (Anonymous Functions):

This approach is pretty clean. You don't need to string the extra parameters out at the end and you don't need to nest. You just make it explicit what the parameter list is in the function. Thus with the three variable testf written just as back in the Old approach, you need only invoke:

```
quad(@(x) testf (x, beta, z), beta, 2*beta)
```