To complete the lab exercises, you will also need to download the `bisectNew.m`, `showBisectFx.m` and `showFzeroFx.m` files.

Root-finding with Nested User-Defined Functions

1. Write a single m-file that uses `bisect` and an embedded function to find the roots of

$$f(x) = 1.4 - \frac{1}{x^{2.71} - 3} - 0.234x^{1.2} \tag{1}$$

   (a) First, plot $f(x)$ on the interval $1.5 \leq x \leq 5$. How many roots are in this interval?

   (b) Add a call to `bisect` to refine one or more of the roots.

   (c) Replace the call to `bisect` with `bisectNew` so that you can request both the root and the number of function evaluations necessary to find the root.

```
[r,fcount] = bisectNew(@fx,xb(i,:));
```

   How many iterations are required to find the root? Does the size of the bracket interval affect the number of iterations necessary for convergence?

2. Make a copy of the m-file from the preceding exercise and replace the call to `bisectNew` with a call to `fzero`. Test that the same root(s) are returned by both `bisect` and `fzero`.

   (a) Use the fourth *output* parameter from `fzero` to request a data structure with information on the iterations.

```
[r,~,~,output] = fzero(@fx,xb(i,:));
```

   The ~ symbol is a placeholder that tells MATLAB not to return a value, and to skip that position in the output list.

   `output` is a *data structure* that stores several variables in a container. Both `output`, the container, and the individual elements, called *fields*, can be assigned to other variables. The values stored in fields of a data structure can be obtained and set using a "dot" notation, as demonstrated below.

   When using `fzero` we are interested in the total number of iterations by the root-finding algorithm and the total number of function evaluations used by `fzero`. These values are extracted from the `output` data structure as follows.

```
output.iterations      %  number of iterations of the root-finding algorithm
output.funcCount       %  number of times that f(x) was evaluated
```

   For example, you could add those values to an `fprintf` statement like this;

```
fprintf('%8.5f  %8.5f  %12.3e  %6d  %6d\n',x0,r,fr,output.iterations,output.funcCount)
```

(b) How many iterations (`output.iterations`) and how many function evaluations (`output.funcCount`) are required to find the root? Does the size of the bracket interval affect the number of iterations or number of function evaluations necessary for convergence?

(c) What happens when you replace a bracket with a single value as a guess for the root?

3. When methanol is produced by CO and H_2, the equilibrium *extent of reaction* is ξ, where

$$\frac{\xi(3-2\xi)^2}{(1-\xi)^3} = K \tag{2}$$

and $K = 249.2$. In the following exercises, write your solution so that the value of K can be changed. In other words, leave K as a variable that is determined each time your m-file is called, as opposed to hard-coding a constant value for k.

(a) Use `bisect` to find the value of ξ

(b) Use `fzero` to find the value of ξ.

(c) Extra challenge: Use `newton` to find the value of ξ.

Comparing the Performance of `bisect` and `fzero`

1. The `showBisectFx` function implements a solution to the first exercise in this lab (problem 1 on the preceding page). Run `showBisectFx` and explain the results.

2. The `showFzeroFx` function implements a solution to the second exercise in this lab (problem 2 on the preceding page). Run `showFzeroFx` and explain the results.

(a) When `fzero` is given a single guess instead of a bracket, does it always find the "closest" root?

(b) When `fzero` is given a single guess at the root instead of a bracket, how does the work done by `fzero` compare to the work done by `bisect`?

(c) When `fzero` is given a bracket of the root, how does the work done by `fzero` compate to the work done by `bisect`?