

MATH 246: MULTI-FILE PROGRAMS AND TESTING THE SECANT METHOD

To go beyond fairly simple tasks, it is best to organize programs in several files, and with MATLAB programs, I recommend that amongst these files there is a *master script*: a MATLAB M-file script that it the only one you run directly, and which organizes the overall task, invoking other scripts and functions to perform various parts of the task.

Here, we will use the example of testing the secant method, so there are two main files: one containing a MATLAB function

```
function [root,errorest,froot,steps]=secant(fstring,a,b,errortol)
```

defined in M-file `secant.m` and a top-level command

```
>> secant-test
```

defined by a MATLAB M-file script `secant-test.m`.

This script will run a variety of tests of that function and produces an output diary, maybe named `yourname-secant-test-diary.txt`.

Note that for testing purposes, the MATLAB function `secant` should offer additional optional output values `froot`, giving the value of the specified function at the final approximate root given in `froot`, and `steps`, giving the number of iteration steps done.

The master script should handle as much as possible of the production of your final output file for submission, including adding a title and your name atop that file, using commands like `disp`. You will still need to add observations on the results of each test, either in that file or separately. A possible skeleton of such a master script is

```
% test-secant
% Brenton LeMesurier
diary LeMesurier-secant-test-diary.txt
disp 'Secant method testing'
disp 'Brenton LeMesurier'
[root,errorest,fofroot,steps]=secant('x-cos(x)', 0, 1, 1e-12)
...
diary off
```

but of course you will do many more test cases!

This organization will make it far quicker and easier to rerun a collection of tests after the program is modified, or to make modifications to one or two test cases and reproduce the updated output of the whole test suite.

A subsequent step would be to add comparisons between the secant and bisection methods, using a single master script to run test cases on both and gather the results in a single diary:

```
...
[root,errorest,froot,steps]=bisection('x-cos(x)', 0, 1, 1e-12)
[root,errorest,froot,steps]=secant('x-cos(x)', 0, 1, 1e-12)
...
```

This would require slight modification of your previous function `bisection` to add those additional optional output quantities.

SOME TEST CASES

Choosing a good suite of test cases is part of the mathematical task of assessing numerical algorithms, so I leave some choices to be made, but some natural choices come from the previous testing of the bisection method. Since a major hazard with the secant method is problems when the initial interval is too large, experiment with varying the initial intervals, and I add one test case that is sensitive to this problem.

- (1) $f(x) = x - \cos(x)$, $a=0$, $b=1$, $\text{errortol}=1\text{e-}12$.
- (2) $f(x) = x - \sin(x)$, $a=0$, $b=1$, $\text{errortol}=1\text{e-}12$. [Root at one endpoint.]
- (3) $f(x) = x^3 - x$, $a=-2$, $b=3$, $\text{errortol}=1\text{e-}12$. [Multiple roots in the interval.]
- (4) $f(x) = 1/(x - 1)$, $a=0$, $b=3$, $\text{errortol}=1\text{e-}12$. [No root, but a sign change.]
- (5) $f(x) = x^2 - 2x + 1$, $a=0$, $b=3$, $\text{errortol}=1\text{e-}12$. [Non-simple (double) root, with no sign change.]
- (6) $f(x) = x^3 - 3.3x^2 + 3.63x - 1.331$, $a=0$, $b=2$, $\text{errortol}=1\text{e-}3, 1\text{e-}4, 1\text{e-}5, 1\text{e-}6, 1\text{e-}7$ etc., until you have problems. [Non-simple (triple) root, with sign change]
- (7) $f(x) = x^3 - 3.3x^2 + 3.63x - 1.33$, $a=0$, $b=2$, $\text{errortol}=1\text{e-}12$. [A slight perturbation of the above, causing significant changes.]
- (8) $f(x) = (x - 2)e^{-x}$, $a=1$, $b=10$, $\text{errortol}=1\text{e-}12$. [Unique root at 2, but ...]