

Lab-2: Monday, Feb. 21, 2005

Complex Algorithms that Approximate a Solution

This lab will give you experience with complex algorithms that approximate a solution via successive improvements.

You will be wise to save your code occasionally throughout the lab session, in case Scheme crashes and you need to restart.

1. File `~carberry/NewtonRaphson.scm` contains the Scheme code for the Newton-Raphson method of approximating the root of an equation. Copy it into your directory.
2. The Secant Method is another method of approximating the root of an equation. It uses the two most recent approximations to the root, namely x_{i-1} and x_i , and computes the new approximation as

$$x_{i+1} = x_i - f(x_i) \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$

Modify the Newton-Raphson code so that it computes the root of an equation via the Secant method. You will need to do the following:

- (a) Change the name of the procedure to `Secant-root` instead of `NR-root`.
- (b) You no longer need to pass the derivative as a parameter.
- (c) However, you do need to include both the current guess (which we will call `oldguess`) and a prior guess (which we will call `olderguess`) as formal parameters of `Secant-root`. So your procedure should now be

```
(define Secant-root (lambda( f olderguess oldguess errortol n)
```

- (d) Rewrite procedure `improve` to reflect the Secant method's way of computing a new approximation to the root.
3. Define a procedure `testfn1` to compute $f(x) = \frac{x^3}{16} - 4$
 4. Define a procedure `testfn2` to compute $f(x) = x^3 - 3x + 1.872$
 5. Execute your `Secant-root` procedure to find the root of the first $f(x)$ defined in `testfn1`, using 0 and 5 as your current guess and your prior guess at the root. Use .0001 as your error tolerance and 15 as your maximum number of iterations.
 6. Without making any changes to your procedure `Secant-root`, execute your `Secant-root` procedure to compute the root of the $f(x)$ defined in `testfn2`. Once again, use .0001 as your error tolerance and 15 as your maximum number of iterations.
 7. Save your code for `Secant-root` in a file.
 8. Now rename your procedure as `Secant-root-rev` — be sure to rename it both in the definition and wherever it is called.
 9. Revise your code for `Secant-root-rev` so that your procedures for `improve` and for `good-enough?` are local to procedure `Secant-root-rev`.
 10. Now revise your code so that as many variables as possible are *free* variables in procedures `improve` and `good-enough?`
 11. Execute `Secant-root-rev` to compute the root of each of your $f(x)$ functions.
 12. Copy your code for `Secant-root` and for `Secant-root-rev` into the lab submission web page, and submit it for lab-2. Be sure to click on `Confirm` once you are happy with the results.