Chapter 5
Numerical Root Findings

Root of a continuous function

- The roots of a function $f(x)$ are defined as the values for which the value of the function becomes equal to zero. So, finding the roots of $f(x)$ means solving the equation $f(x) = 0$.
- The value of $x=r$ such that $f(r)=0$ is the root for the function f.
- Given a continuous function in an interval, how do we find it roots?

Bisection method

- We shall refer to the lecture note by Dr Dana Mackey, Dublin Institute of Technology: http://www.maths.dit.ie/~dmackey/lectures/Roots.pdf

Bisection method, figure

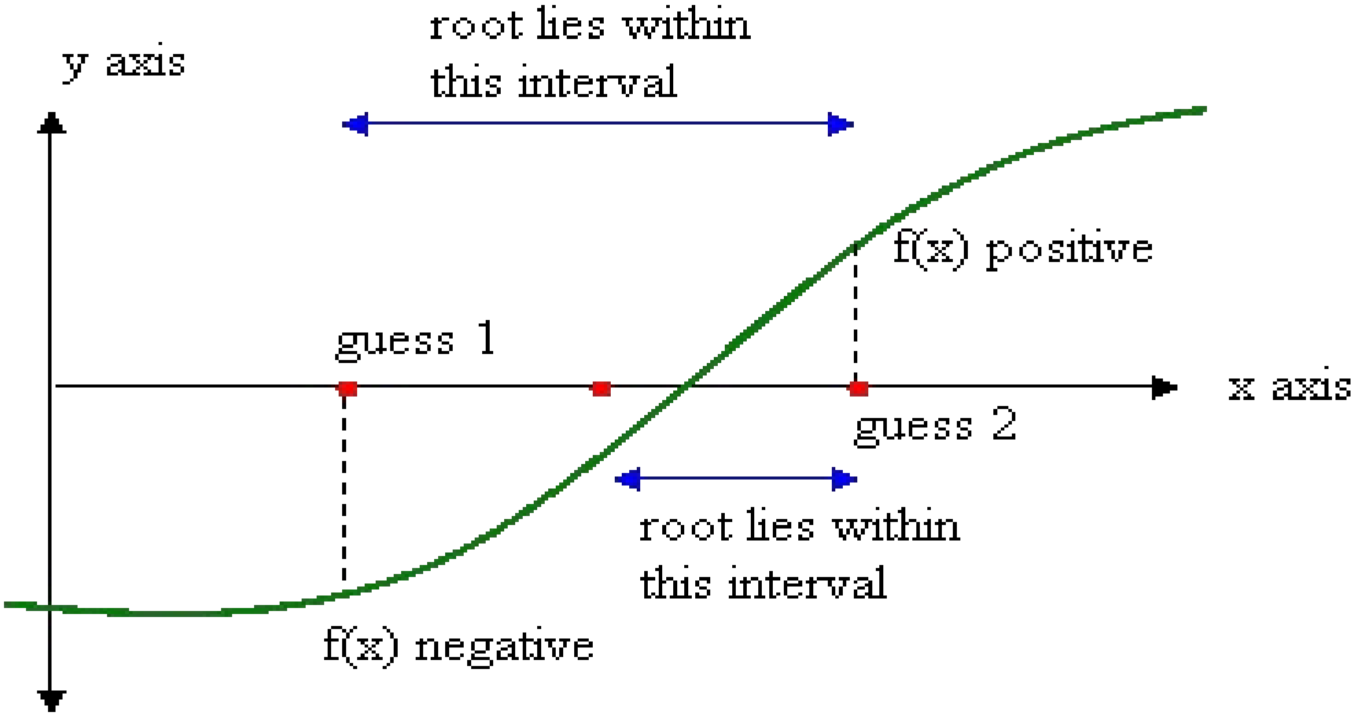


Figure credit:
http://cse.unl.edu/~sincovec/Matlab/Lesson%2010/CS211%20Lesson%2010%20-%20Program%20Design.htm

The algorithm of bisection method

- Suppose we wish to find the root for f(x), and we have an error tolerance of ε (the absolute error in calculating the root must be less that ε).
- Step 1: Find two numbers a and b at which f has different signs.
- Step 2: Define $c = (a + b)/2$.
- Step 3: If $|f(c)| \leq \varepsilon$ then accept c as the root and stop.
- Step 4: If $f(a)f(c) \leq 0$ then set c as the new b. Otherwise, set c as the new a. Return to step 1.

Exercise

- Find a root of the equation
- $x^6 - x - 1 = 0$

  accurate to within $\varepsilon = 0.001$.
- We will need to implement the algorithm using **While** command.
- See bisection_rootfinding.nb.
- The function contains two roots but the code only finds one. As an exercise, modify it to find the other root manually.

Exercise

- Modify the code bisection_rootfinding.nb so that it can automatically find both roots of the equation

  $$x^6 - x - 1 = 0$$

  accurate to within $\varepsilon = 0.001$, without manual intervention.

Exercise

Modify your code further so that, given any continuous function $f(x)$, it can

(i) Count the number of roots in a domain [a,b].

(ii) Evaluate each of these roots one by one in sequence.

Try your code on the following functions

(i) $f(x) = e^x - x - 2$, for all x.

(ii) $f(x) = x^3 + 2x^2 - 3x - 1$, for all x

(iii) $f(x) = (1/x) \sin x$, for all x.

(iv) $f(x) = \tan(\pi x) - x - 6$, for 0 <= x <= 2Pi.

Use $\varepsilon = 0.001$. You code is suppose to be able to find out the roots in all the functions automatically and without manual intervention.

Newton-Raphson Method

Recall that the equation of a straight line is given by the equation

$$y = mx + n \tag{1}$$

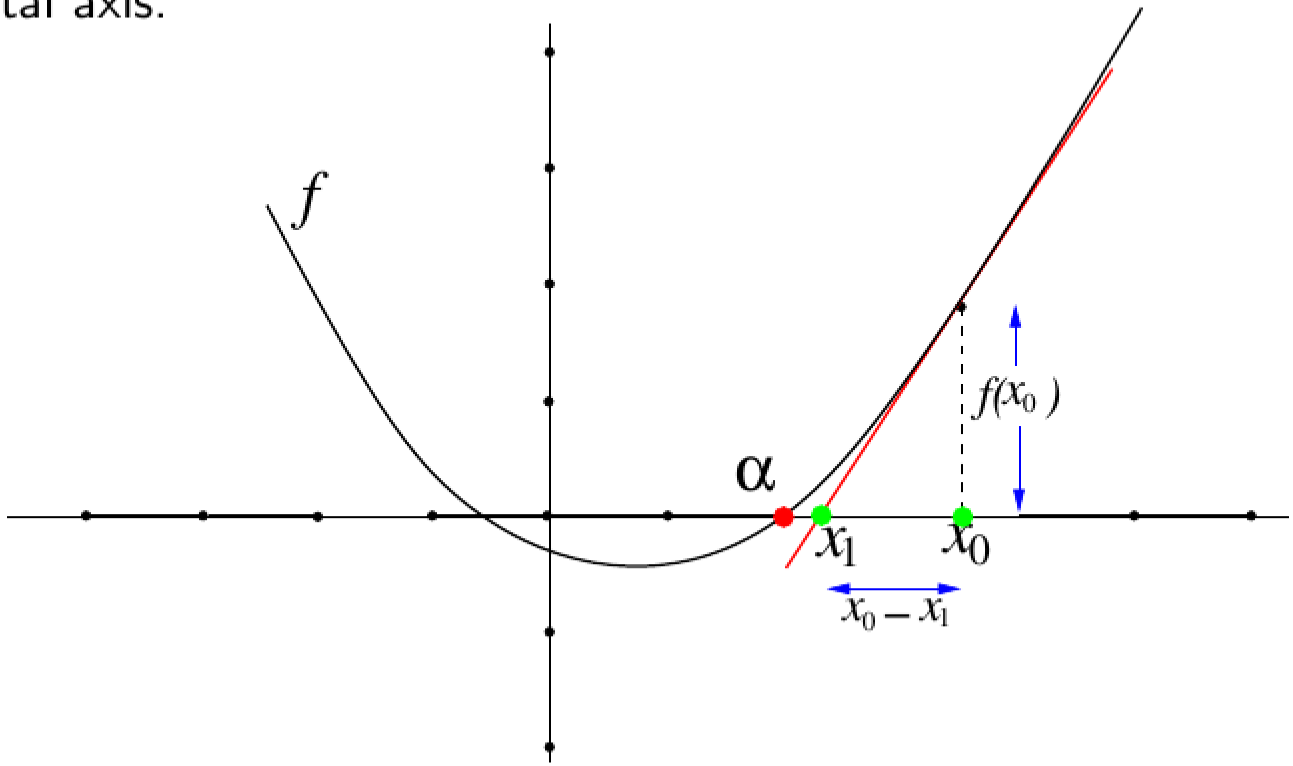
where m is called the *slope* of the line. (This means that all points (x, y) on the line satisfy the equation above.)

If we know the slope m and one point (x_0, y_0) on the line, equation (1) becomes

$$y - y_0 = m(x - x_0) \tag{2}$$

Idea behind Newton's method

Assume we need to find a root of the equation $f(x) = 0$. Consider the graph of the function $f(x)$ and an initial estimate of the root, x_0. To improve this estimate, take the tangent to the graph of $f(x)$ through the point $(x_0, f(x_0)$ and let x_1 be the point where this line crosses the horizontal axis.



According to eq. (2) above, this point is given by

$$x_1 = x_0 - \frac{f(x_0}{f'(x_0)}$$

where $f'(x_0)$ is the derivative of f at x_0. Then take x_1 as the next approximation and continue the procedure. The general iteration will be given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

and so on.

Newton-Raphson Method

- The code Newton_method_root_finding.nb finds a root of the following equations using Newton method based on an initial guess:

(i) $f(x) = e^x - x - 2$

(ii) $f(x) = x^3 + 2x^2 - 3x - 1$

(iii) $f(x) = (1/x) \sin x$

(iv) $f(x) = \tan(\pi x) - x - 6$

Exercise

Modify the code Newton_method_root_finding.nb so that it can find all the roots using Newton method for all the following functions automatically and without manual intervention:

(i) $f(x) = e^x - x - 2$, for all x.

(ii) $f(x) = x^3 + 2x^2 - 3x - 1$, for all x

(iii) $f(x) = (1/x) \sin x$, for all x.

(iv) $f(x) = \tan(\pi x) - x - 6$, for 0 <= x <= 2Pi.

Use $\varepsilon = 0.001$.

Mathematica built-in function to find roots

Syntax: **FindRoot**. **NSolve**.

- **NSolve** find multiple solutions automatically, but may fail in certain types of equations. Best used for algebraic equations and polynomials.
- **FindRoot** finds only one root at a time, and needs an initial guess value. More robust than **Nsolve**.

Example of **NSolve**

- See Math_built_in_Nsolve.nb.
- NSolve[x^5 - 2 x + 3 == 0, x]
- NSolve[x^5 - 2 x + 3 == 0, x, Reals]
- NSolve[(x^2 - 1) (x^4 - 1) == 0, x, Reals]
- NSolve[Sqrt[x] + 3 x^(1/3) == 5, x, Reals]
- NSolve[E^x - x == 7, x, Reals]
- NSolve[E^(2 E^x) - Log[x^2 + 1] - 20 x == 11, x, Reals]
- NSolve[2 x^(123451/67890) - x^2 + 4 Sqrt[x] - 4 x - 9/8 ==
-   0, x, Reals]
- NSolve[E^(2 x) + x^4 + 4 (x^2 + 1) == (2 x^2 + 4) E^x, x, Reals]
- NSolve[10 Sin[Tan[E^-x^2]] - x == 3, x, Reals]
- NSolve[2 Sin[Exp[x]] - Cos[Pi x] == 3/2 && -1 < x < 1, x, Reals]

Example of using **FindRoot** and **NSolve**

We would like to try using FindRoot and Nsolve on the following examples (previously solved using Newton and bisection method):

(i) $f(x) = e^x - x - 2$, for all x.

(iii) $f(x) = (1/x) \sin x$, for all x.

(iv) $f(x) = \tan(\pi x) - x - 6$, for 0 <= x <= 2Pi.

See Math_built_in_findroots_NSolve.nb

You are now a proud root finder

After all these exercises, now you should be confident to proclaim to the whole word that:

Given me any single variable function, and I'll find you their roots at a click. =)