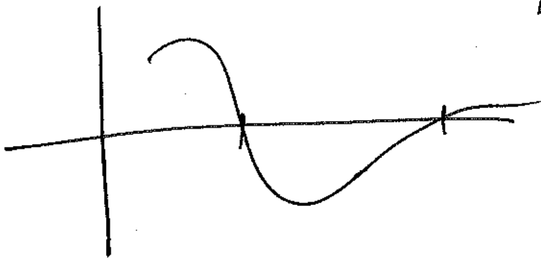
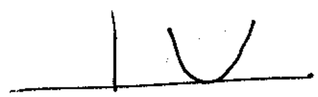
Root Finding:

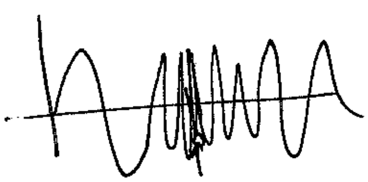


Aim: Given some function $f(x)$, find values x_0 where $f(x_0) = 0$

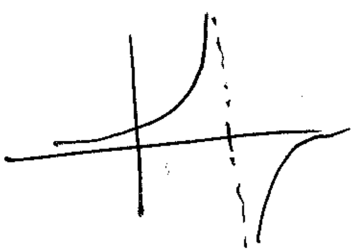
General Method: find where $f(x)$ changes sign ... this will be the root.

Possible problems:

①  no change of sign

② $\cancel{\frac{\sin x}{x}}$ $\sin\frac{1}{x}$ 

this function has infinitely (?) many zeros near $x=0$.

③  Change of sign occurs at a divergence of a function.

As a result, you have to be reasonably smart when root finding. Practically speaking, this means selecting bounderies ~~that~~ $\{x_1, x_2\}$ that you know bracket the root @ x_0.

Bracketing Method: very simple, not very efficent.

Take the interval $[x_1, x_2]$ and subdivide into n intervals (~~so that you have a~~ & find where $f(x)$ changes sign. This gives the root to tolerance $= \frac{x_2 - x_1}{n}$

This process is order n $(O(n))$

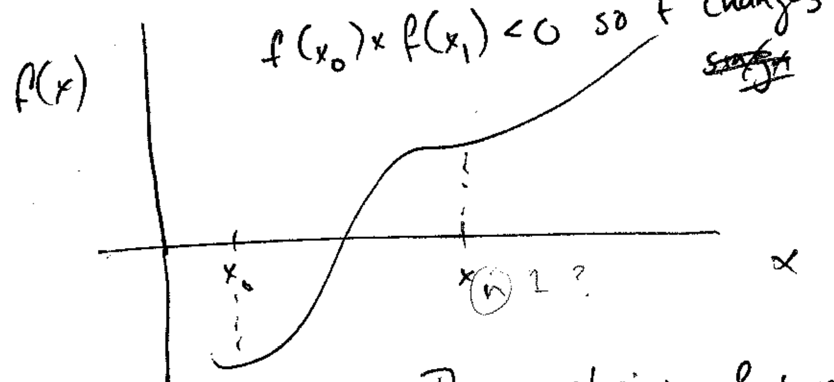
Root Finding:

This is one of the most basic numerical tasks, finding solutions to $f(x) = 0$

(Note that Mathematica has a built-in function "FindRoot").

We will deal with this as a 1-D problem, but in general f could be an N-dimensional vector field that is itself a function of an N-dim vector.

Bracket Method:

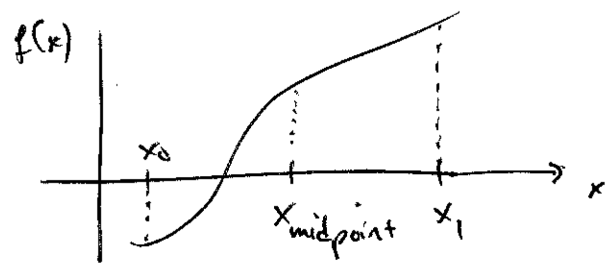


Simple case: $f(x)$ is well behaved & we know it has a root in the range $x \in [x_0, x_n]$

The obvious feature is that the fctn changes sign! The bracket method ~~tha~~ uses the simple approach of splitting the interval into segments of size ϵ, and ~~sp~~ stepping from $x_0 \to x_1$ until a change of sign is found. This is the location of the root, to tolerance ϵ. The number of steps needed to find the root is on the order of the number of divisions... so $N \sim \frac{1}{\epsilon}$ or $\epsilon \sim \frac{1}{N}$

steps to find root

Bisection Method: Again, let's assume we know $f(x) = 0$ in the range $x \in (x_0, x_1)$. The bisection method uses the value @ midpoint of the interval to speed things up! Explained graphically:



By evaluating the fctn @ $f(x_{midpoint})$ we see it changes sign; therefore, the root must be in the range $(x_0, x_{midpoint})$!

So let $x_0 \rightarrow x_0$ & $x_{mid} \rightarrow x_1$ & repeat the process.

Clearly this is superior to the bracket method since we eliminate half the interval with each step!

Mathematically, we have the root on an interval ϵ_n on the n^{th} step. On the $n+1$ step,

$$\epsilon_{n+1} = \frac{\epsilon_n}{2} \text{ "linear convergence"}$$

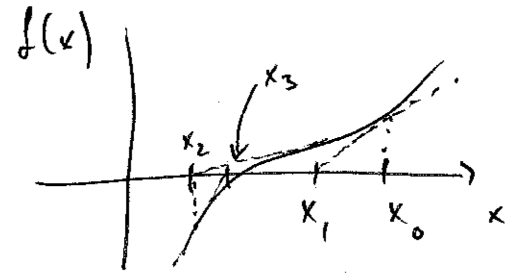
; if our desired tolerance is ϵ, we can relate to the initial bracket interval $\epsilon_0 = x_1 - x_0$ by

$$\epsilon_N = \left(\frac{\epsilon_0}{2^N}\right)$$

where N is the number of steps to converge.

$\Rightarrow$ $N = \log_2 \frac{\epsilon_0}{\epsilon}$ is generally less the $\frac{1}{\epsilon}$, so this method is far more efficient than the bracket method.

Newton-Raphson Method: This method needs only 1 value to start, & use information about the derivative to speed convergence to the root. Graphically:



ie we make a linear approximation to estimate the location of the root & use that point to further refine our estimate.

For a given starting point x_0, we need to determine our estimate x_1 of the zero. The tangent line has an equation

$$y = mx + b \qquad m = f'(x_0)$$ determined either analytically or numerically.

We also know $f(x_0) = mx_0 + b$ so

$$b = f(x_0) - mx_0 = f(x_0) - f'(x_0)\,x_0$$

The line $y = mx + b$ intersects the $y=0$ axis at

$$0 = mx + b \Rightarrow x_{root} = \frac{-b}{m} = \frac{-(f(x_0) - f'(x_0)\,x_0)}{f'(x_0)}$$

$$\Rightarrow x_1 = -\frac{f(x_0)}{f'(x_0)} + x_0 \qquad \text{or} \qquad \boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}$$

We iterate this process until the difference

$$|x_{n+1} - x_n| < \epsilon \quad \text{(tolerance)}.$$

How does this converge?

Define the root r, ie $f(r)=0$ & thus error

$$\epsilon_n = x_n - r$$

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

$$= (x_n - r) - \frac{f(x_n)}{f'(x_n)}$$

$$= \epsilon_n - \frac{f(x_n)}{f'(x_n)}$$

Now Taylor expand $f(x_n) = f(r+\epsilon_n)$ & $f'(x_n) = f'(r+\epsilon_n)$

$$f(r+\epsilon_n) = \cancelto{0}{f(r)} + \epsilon_n f'(r) + \epsilon_n^2 \frac{f''(r)}{2} + \dots$$

$$f'(r+\epsilon_n) = f'(r) + \epsilon_n f''(r) + \dots$$

plug in dropping terms $O(\epsilon^2)$ or smaller ...

$$\epsilon_{n+1} = \epsilon_n - \frac{\epsilon_n f'(r)}{f'(r) + \epsilon_n f''(r)} = \epsilon_n - \frac{\epsilon_n}{1+\epsilon_n f''(r)/f'(r)}$$

$\frac{1}{1+x} \approx 1-x$ for x small

$$\approx \epsilon_n - \epsilon_n \left(1 - \epsilon_n \frac{f''(r)}{f'(r)}\right)$$

$$\Rightarrow \boxed{\epsilon_{n+1} = \epsilon_n^2 \frac{f''(r)}{f'(r)}}$$

The error decreases Quadradically.

"geometric convergence"

In terms of the initial error ϵ_0,

$$\epsilon_N = \left(\frac{f''(r)}{f'(r)}\right)^{(2^N-1)} \times \epsilon_0^{2^N} \quad \text{or} \quad \boxed{\epsilon_N \sim \epsilon_0^{2^N}}$$ drops very fast!

In terms of steps, $\ln \epsilon_n = 2^N \ln \epsilon_0$

$$\ln(\ln \epsilon_n) = N \ln 2 + \ln(\ln \epsilon_0)$$

$\Rightarrow N \sim \ln(\ln \epsilon)$ where ϵ is the desired accuracy.