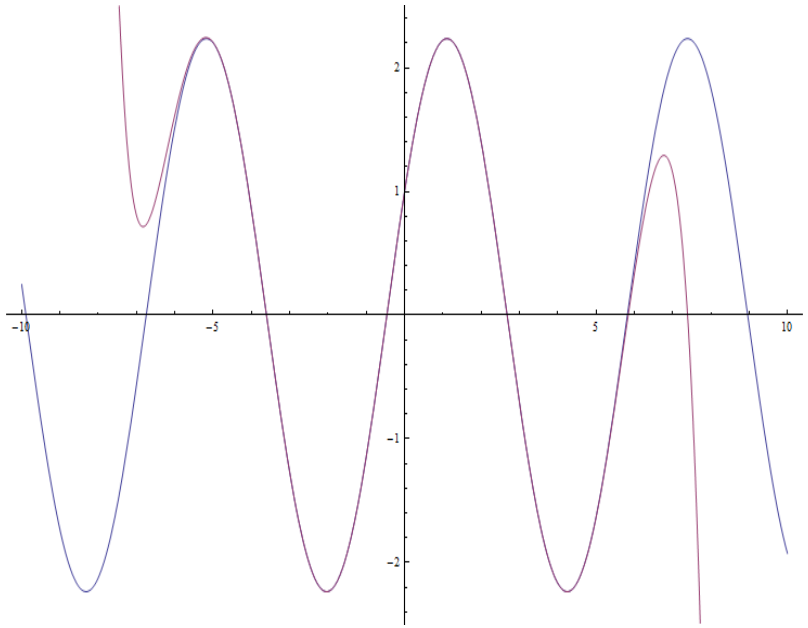
Taylor Series (revision)

Power Series: A different way of thinking of functions

Another way of *representing functions* is as an infinite sum of *powers of x.*

E.g. instead of writing

$$f(x) = \cos(x) + 2\sin(x)$$



We can write

$$f(x) = 1 + 2x - \frac{1}{2}x^2 - \frac{1}{3}x^3 + \frac{1}{24}x^4 + \cdots$$

We can write any smooth function this way. Such a representation is called a *power series.*

All we need to find are the *coefficients* in the series.

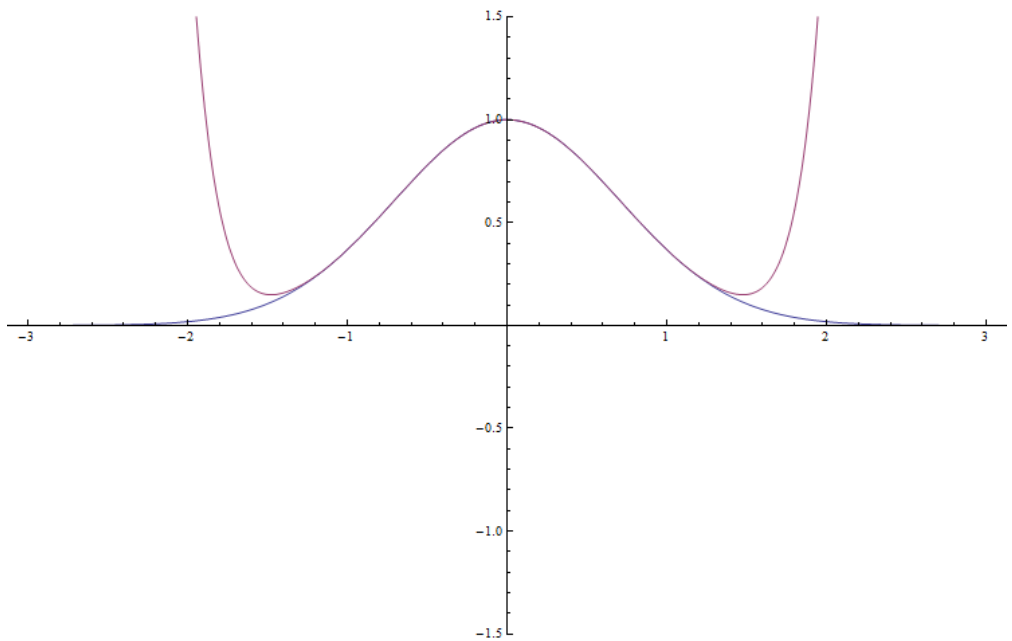
Another example:

The function

$$f(x) = e^{-x^2}$$

Can be represented by the series

$$f(x) = 1 - x^2 + \frac{1}{2}x^4 - \frac{1}{6}x^6 + \cdots$$



Why would we do this?

1. Because the series representation is *often much simpler to deal with*.

Example:
If we know the coefficients of the series, then we can differentiate/integrate very easily.

$$f(x) = e^{-x^2}$$

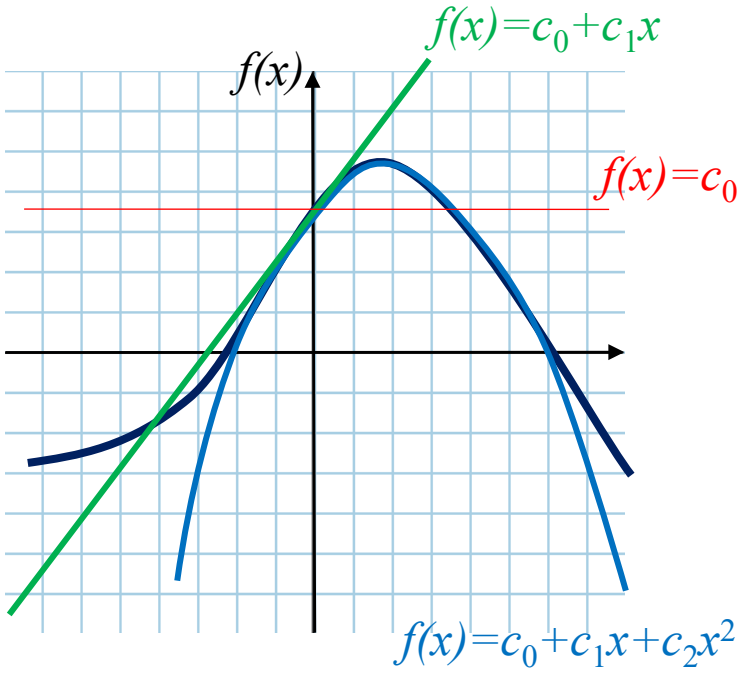
$$f(x) = 1 - x^2 + \frac{1}{2}x^4 - \frac{1}{6}x^6 + \cdots$$

2. It gives us a powerful way to evaluate functions. (This is in fact how a lot of functions are evaluated)

E.g. f(0.1)=

We can think of a series representation as a sum of polynomials.

$$f(x) = c_0 + c_1 x + c_2 x^2 + \cdots$$



The 1st term specifies the *value* of f at x = 0.

The 2nd term specifies the *first derivative* of f at x = 0.

The 3rd term specifies the *second derivative* of f at x = 0.

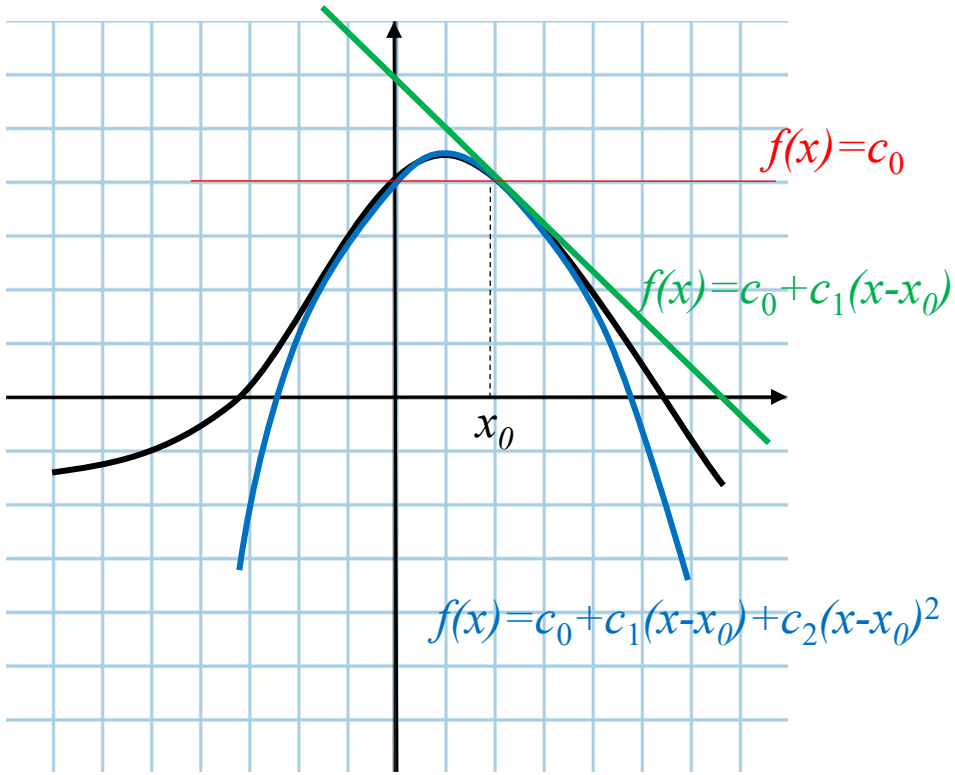
As we add more terms, the series converges to the "real" function.
A truncated series is *most accurate near the point x = 0.*

Taylor Series

The power series expansion of a function $f(x)$ about a point $x=x_0$ is called the *Taylor series* of $f(x)$ at x_0.

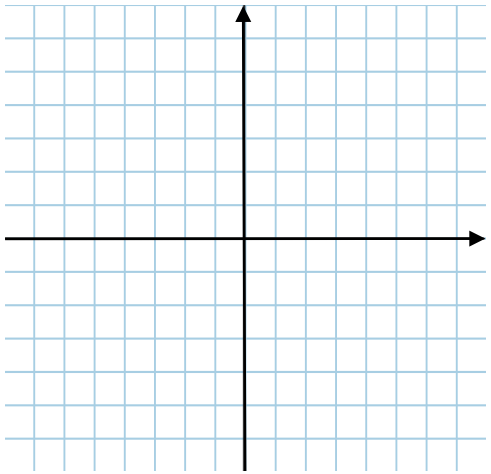
$$f(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)^2 + \cdots$$

$$f(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)^2 + \cdots$$



The Taylor series expansion of a function f(x) about x_0 is:

$$f(x) = \sum_{k=0}^{\infty} \frac{1}{k!} f^{(k)}(x_0)(x - x_0)^k$$



Or:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2 + \cdots$$

Solution of nonlinear equations (a.k. root-finding)

Why this is sometimes difficult

Newton’s method (and why it can go wrong)

The importance of Bracketing

Bisection

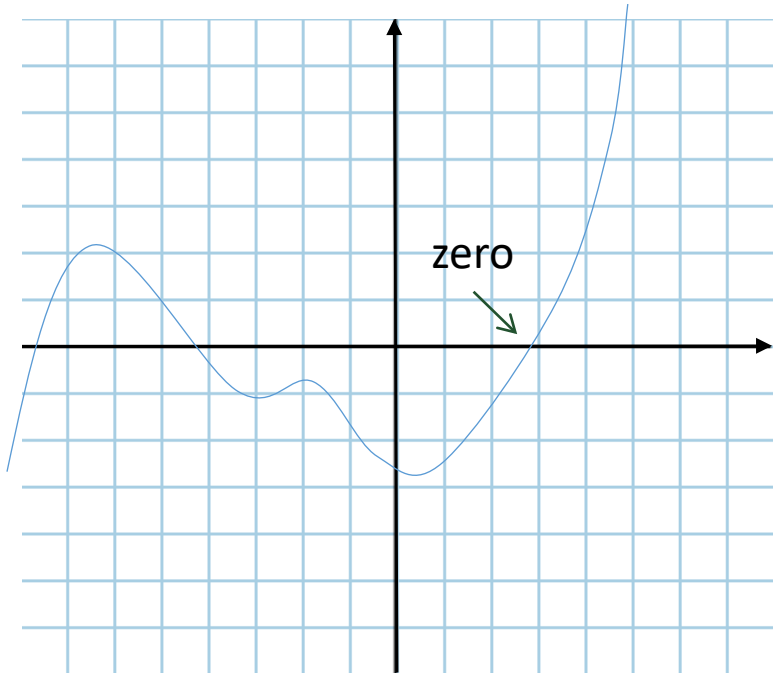
The secant and false-position methods

Brent’s method (overview)

Very often we would like to find the *solutions* to equations of the form

$$f(x) = 0$$

This is often called *root-finding* or *zero finding.*



If $f(x)$ is a quadratic equation (or even a cubic or quartic) then there is an *analytic solution.* However things are not usually that simple, and often we can only compute f(x) *numerically.*

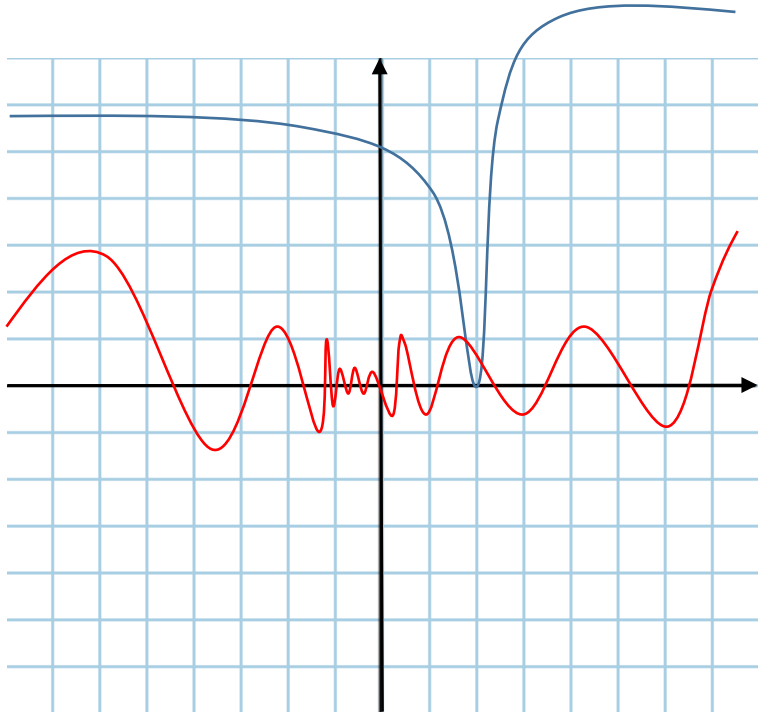
Things that make root-searching difficult:

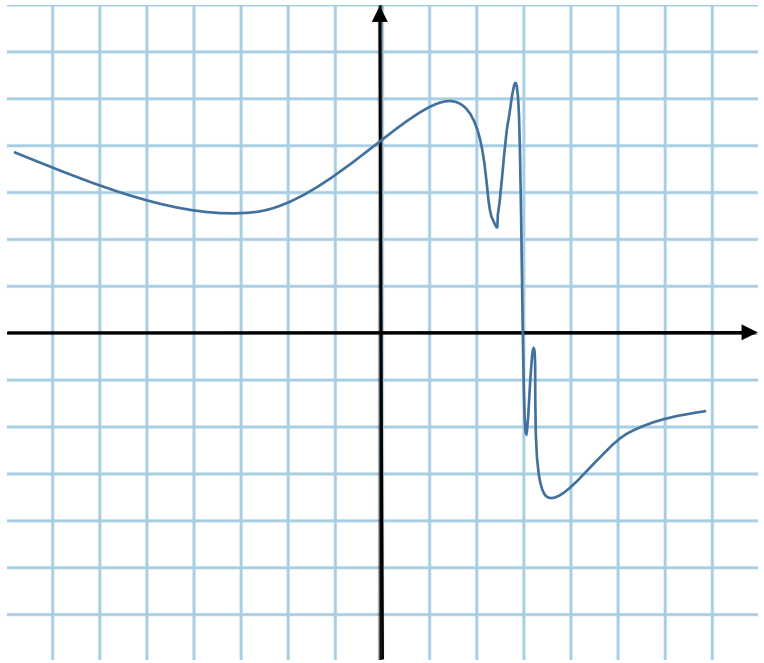
1. Function doesn't change sign

2. Function has a lot of zeros in a small area

3. Function has very large derivatives in the vicinity of a zero

4. Function is "pathological" or non-smooth





Newton's Method

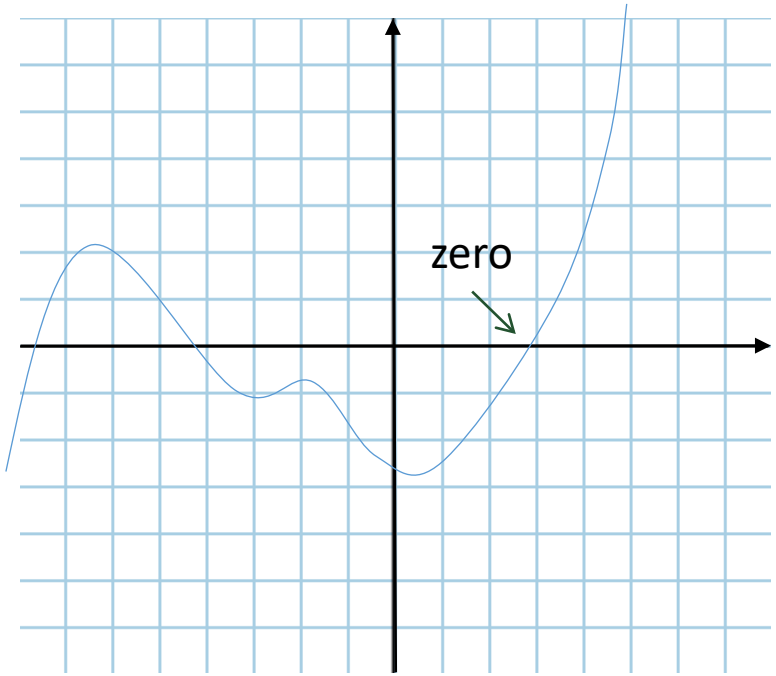
Newton's method is almost the simplest algorithm that achieves rapid convergence to the root.

Advantages:

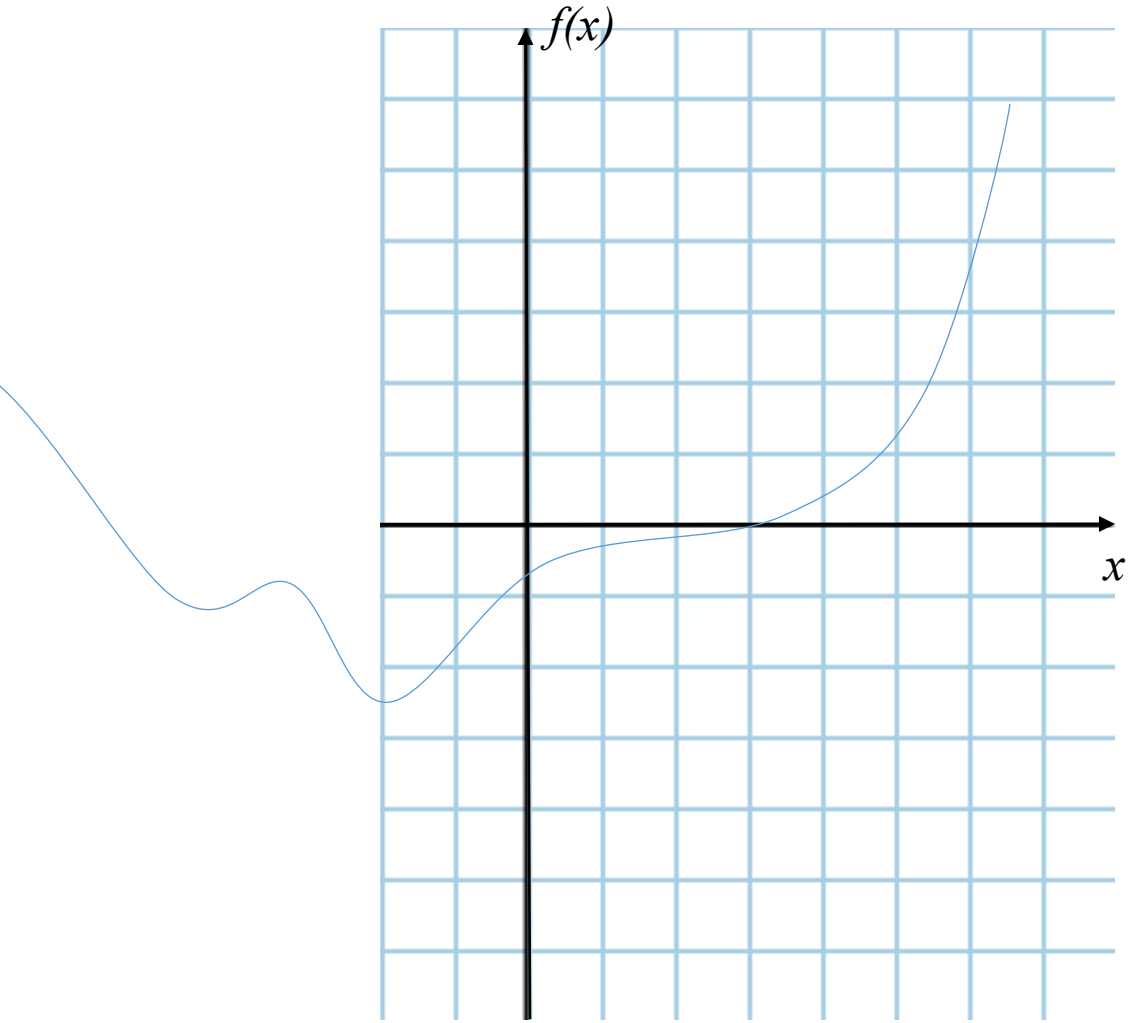
1. Simple to implement
2. Converges extremely quickly

Disadvantages:

1. Requires access to the derivatives
2. Is not a *bracketed* algorithm
3. In some situations can get "stuck"

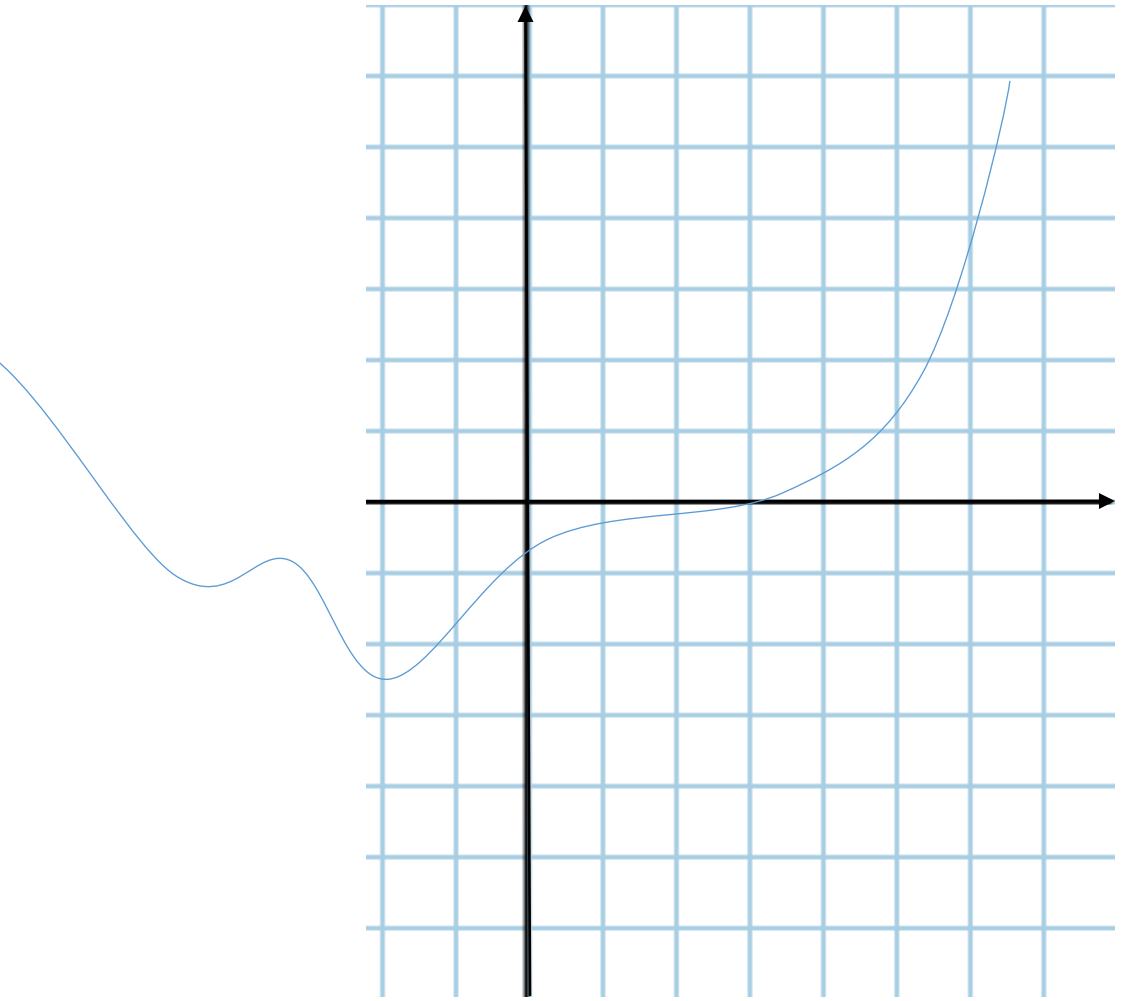


The idea is to start with an initial guess near the root, and use the derivatives to get a better guess.



The algorithm:

1. Start with a point x_0
2. Draw a tangent to the curve, and find where this intersects the x axis
3. This point becomes the next best guess. Repeat



Newton's method algorithm:

1. Start with a point x_n
2. The next point is

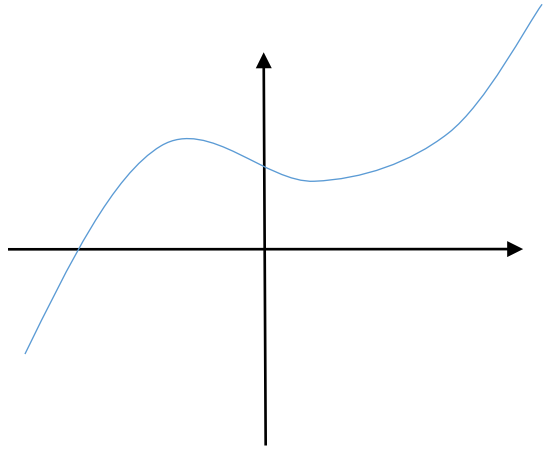
$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

3. Repeat

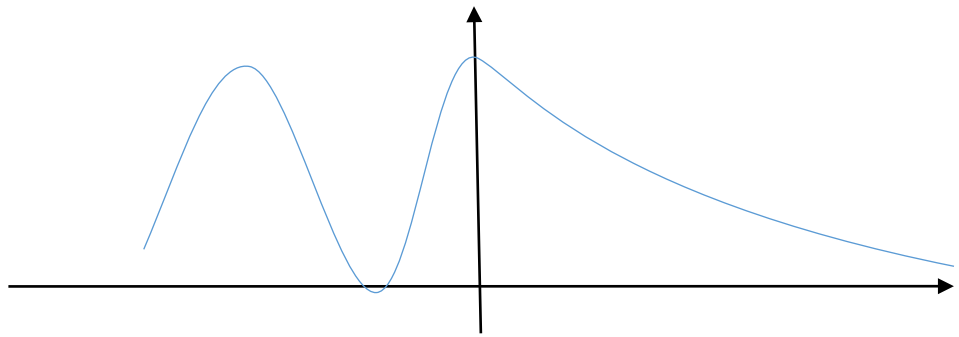
Things that can go wrong
with Newton's method:

1. The derivative can become very small,
causing the predictions to rocket off somewhere

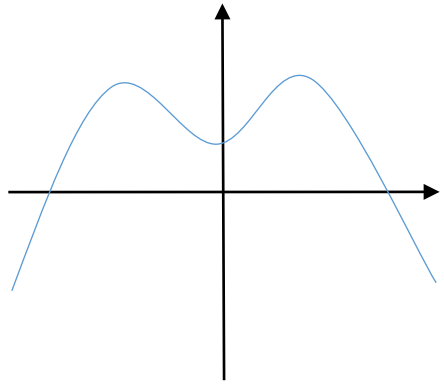
(i.e. the algorithm *exceeds the brackets* of the root)



2. The solutions might *run away*



2. You can land in a "cyclic" situation
and it doesn't converge at all.



Convergence analysis of Newton's method

How fast does this converge to the root?

To answer this, we would like to know *how much the error changes with each iteration.*

Imagine that the zero is at a point α. The absolute error in the method at any iteration is then given by

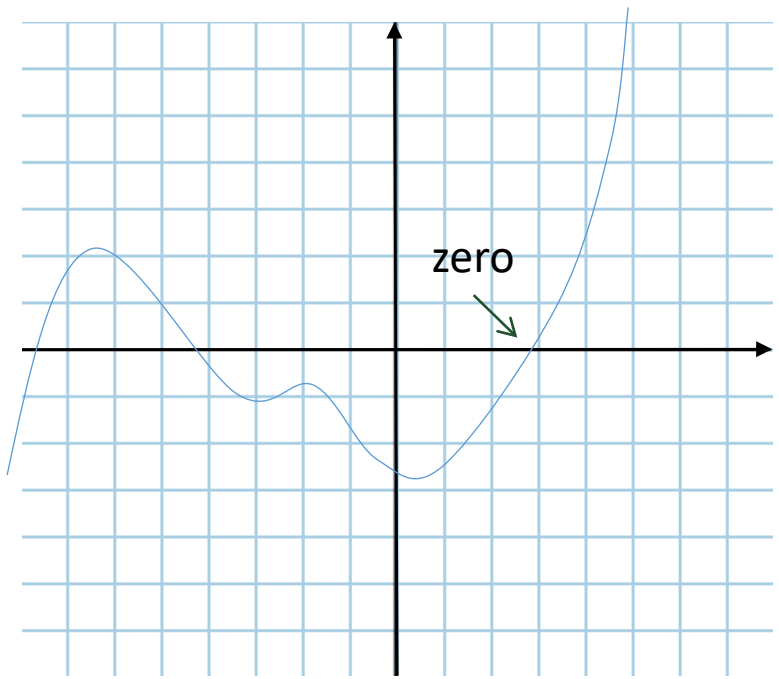
$$\epsilon_n = |x_n - \alpha|$$

The Taylor expansion of f about x_n is

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2}f''(x_n)(x - x_n)^2 + \cdots$$

evaluated at α this is

$$0 = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2}f''(x_n)(\alpha - x_n)^2 + \cdots$$

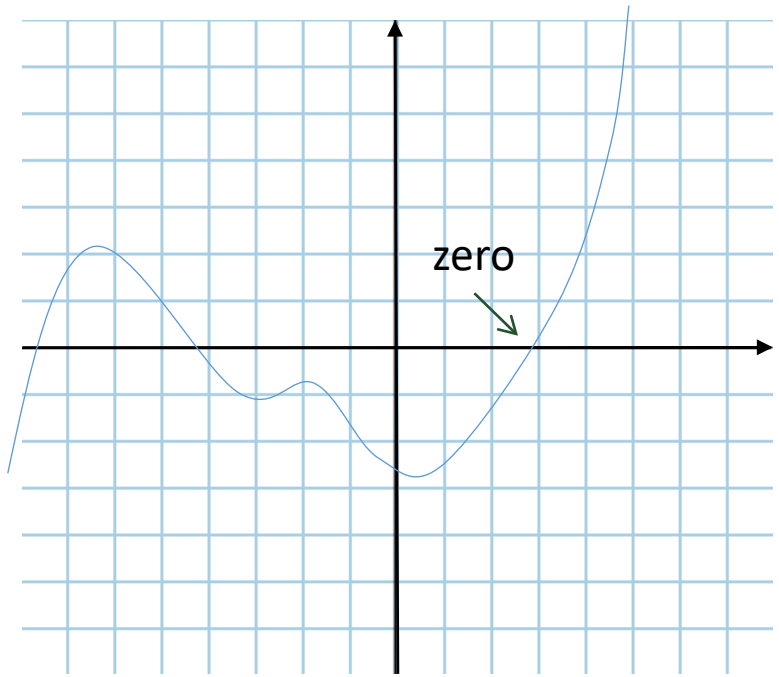


$$0 = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2}f''(x_n)(\alpha - x_n)^2 + \cdots$$

Bisection

What happens if we're happy with slower convergence, but would like something that is guaranteed to find a root? A good algorithm is then the bisection method.

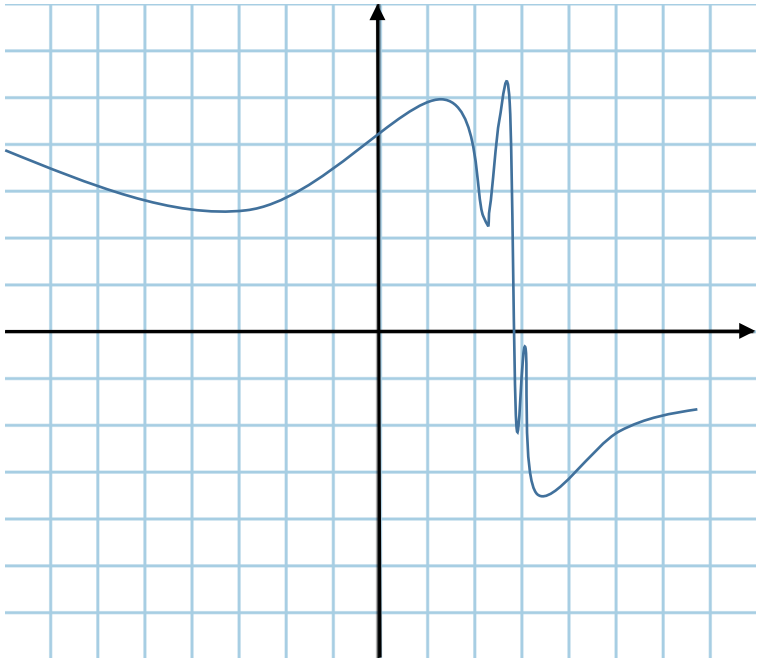
The idea: suppose we start off knowing that the root is between two points. Such a root is said to be *bracketed.*



Bisection Algorithm:

1. Bisect the interval
2. Choose the sub-interval that contains the root
3. Repeat

How to we do this?
We choose the interval for which the function *changes sign.*



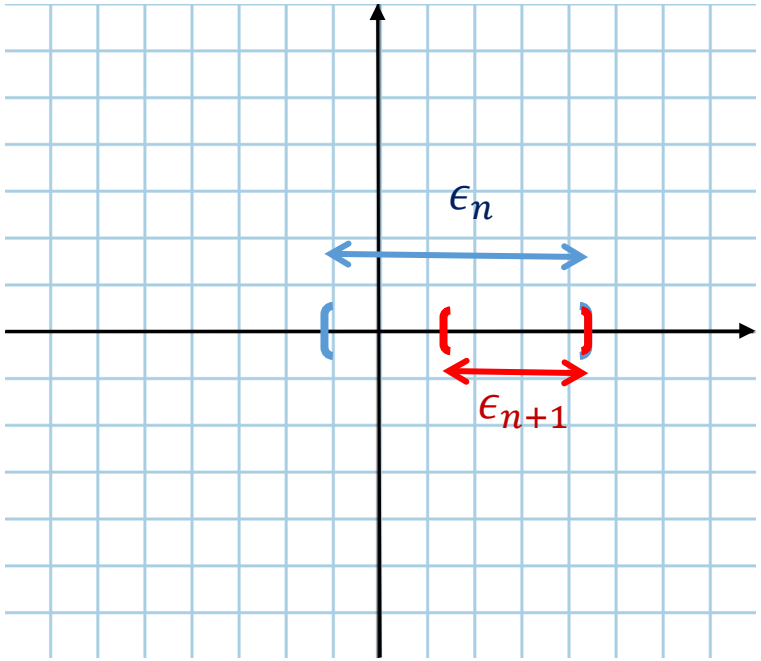
Bisection converges *linearly*. That is, if after n iterations
The root is known to be in an interval of size

$$\epsilon_n$$

Then after the *next step* it the interval will shrink to

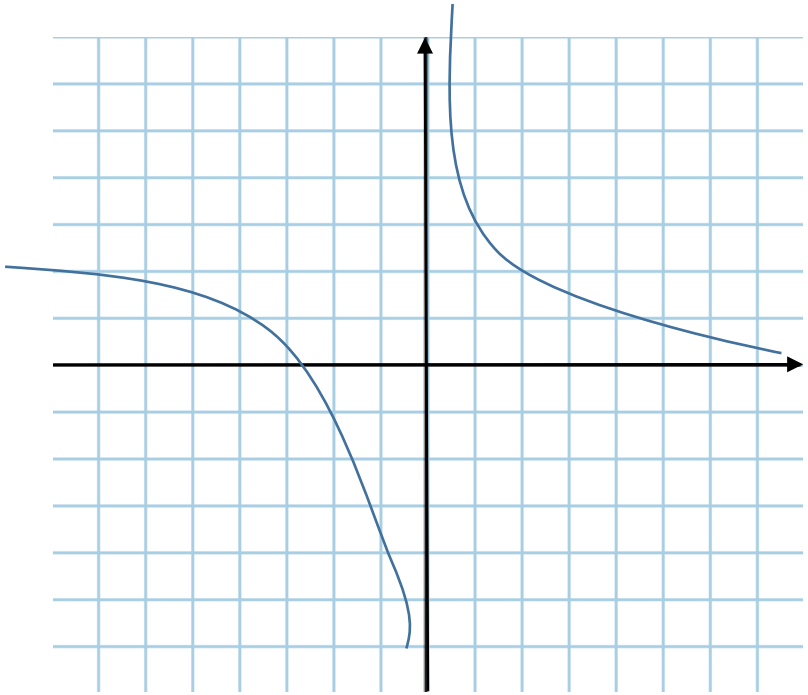
$$\epsilon_{n+1} = \frac{1}{2}\epsilon_n$$

Because this is “to the power one”
we say that this is linear convergence.



Bisection is powerful but slow. It will always find at least one root, provided the root is bracketed.

Note: If the function is not smooth then the bisection may converge on the singularity.

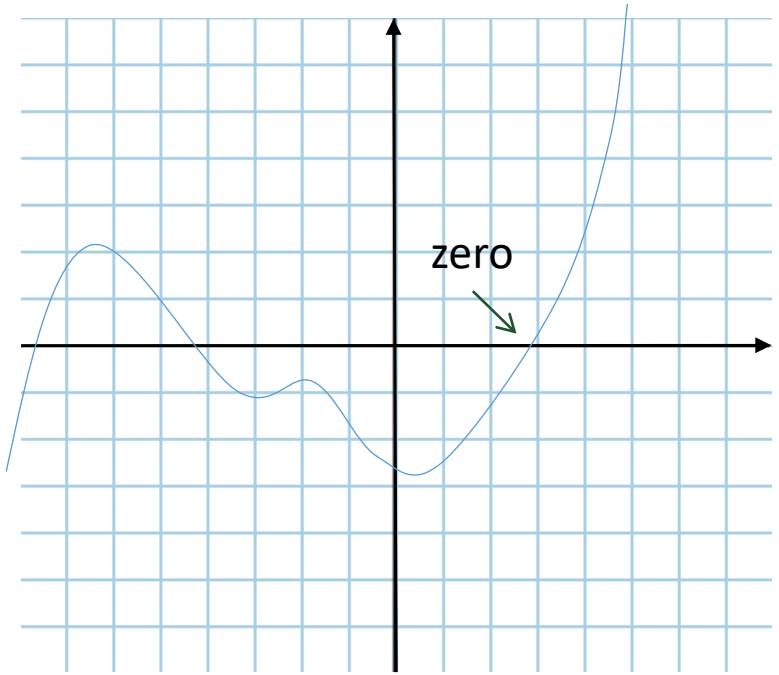


Bracketing

A useful step in many root-finding procedures is *bracketing*. We say
That a root is *bracketed* if there is an interval where we know the root is there.

Basic algorithm for bracketing roots inside an interval:

1. Divide the interval into a lot of sub-segments
2. Test each adjacent pair for a change in sign
3. If no roots are found, then split each interval in two, and repeat



Important Tip: Before applying any root-finding procedure, you should run a Bracketing procedure.

The Secant Method

The secant method can achieve *superlinear convergence* while remaining relatively simple.

The Secant Method

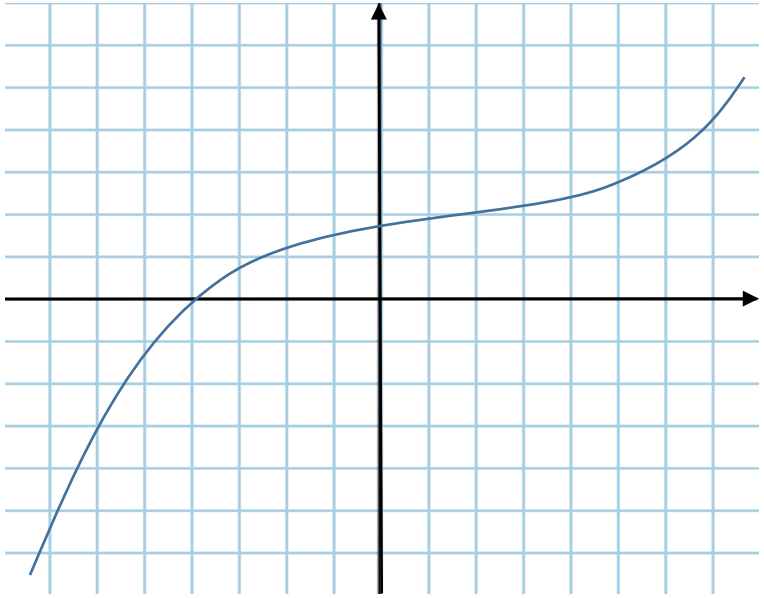
1. Begin with the two most recently computed points x_n and x_{n-1}

2. Construct the secant line between them and find the point x_{n+1} where it crosses the y axis

4. Repeat

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

Derivation of the secant method formula:



The secant method *converges with order*

$$\epsilon_{n+1} = const \times \epsilon_n^{1.618034}$$

So it is better than bisection, but not as good as Newton's method.

The main *disadvantages are:*

1. The root does not remain bracketed

2. Cyclic behaviour is also possible.

The False Position Method

Very similar to the secant method, except *we keep the root bracketed*

The False Position Method

1. Begin with the two points x_n and x_{n-1}

2. Construct the secant line between them and find the point x_{n+1} where it crosses the y axis

3. Keep as the new two points whichever pair $\{x_n, x_{n+1}\}$ or $\{x_{n-1}, x_{n+1}\}$ brackets the root

4. Repeat

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

The convergence of the false position method is hard to compute (and can change over the calculation) but is definitely super-linear.

Brent's method (a.k.a the VanWijngaarden-Dekker-Brent Method)

Brent's method combines the sureness of bisection with (strong) super-linear convergence.
It's complicated to implement, so we just give an overview here of a scaled-down version:

Brent's method (simplified)

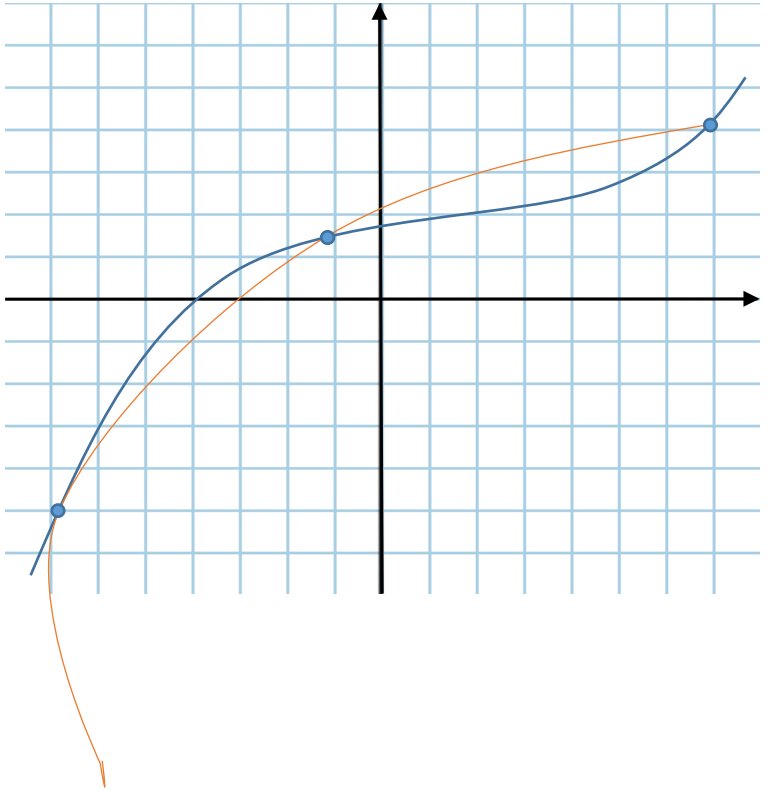
1. Start with 3 points

2. Fit an *inverse parabola* to these three points

3. Check if the y=0 crossing of the parabola
lands inside the brackets.

   3a. If so, great! Take the
   Smallest three points that bracket the root, and repeat.

   3b. If not, do a bisection and repeat



Formulas for the inverse Quadratic Interpolation:

If our three points are a,b, and c, then the new point x where the Inverse parabola crosses the axis is (from numerical recipes, Eq. 9.3.2)

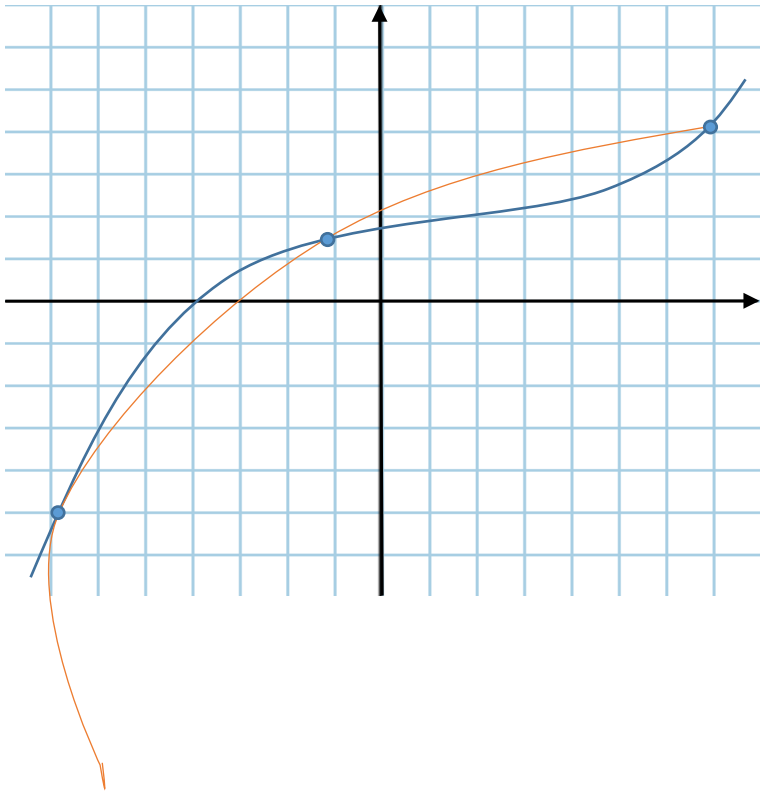
$$x = b + P/Q$$

where

$$P = S\left[T(R-T)(c-b) - (1-R)(b-a)\right]$$
$$Q = (T-1)(R-1)(S-1)$$

and

$$R \equiv f(b)/f(c), \qquad S \equiv f(b)/f(a), \qquad T \equiv f(a)/f(c)$$



For a good explanation and derivation, see Oscar Veliz's video
https://www.youtube.com/watch?v=-bLSRiokgFk

Other important methods:

Dekker's method:
Basically like Brent's method, but uses a secant step instead of an inverse quadratic
(superlinear, but not very)

Ridder's method:
Uses exponential interpolation to find the next root.
(quadratic convergence)

Halley's method:
Newton's method, but 2nd order (so uses a parabola instead of a straight line)
(*cubic* convergence)

Assignment 1 (Due end of Week 3):

Create a nonlinear equation solver that uses Newton's method, but using a numerical derivative.

- Should be based on your Lab solutions to Labs 1 and 2
- The instructions are online under the Assignments" tab – read them carefully