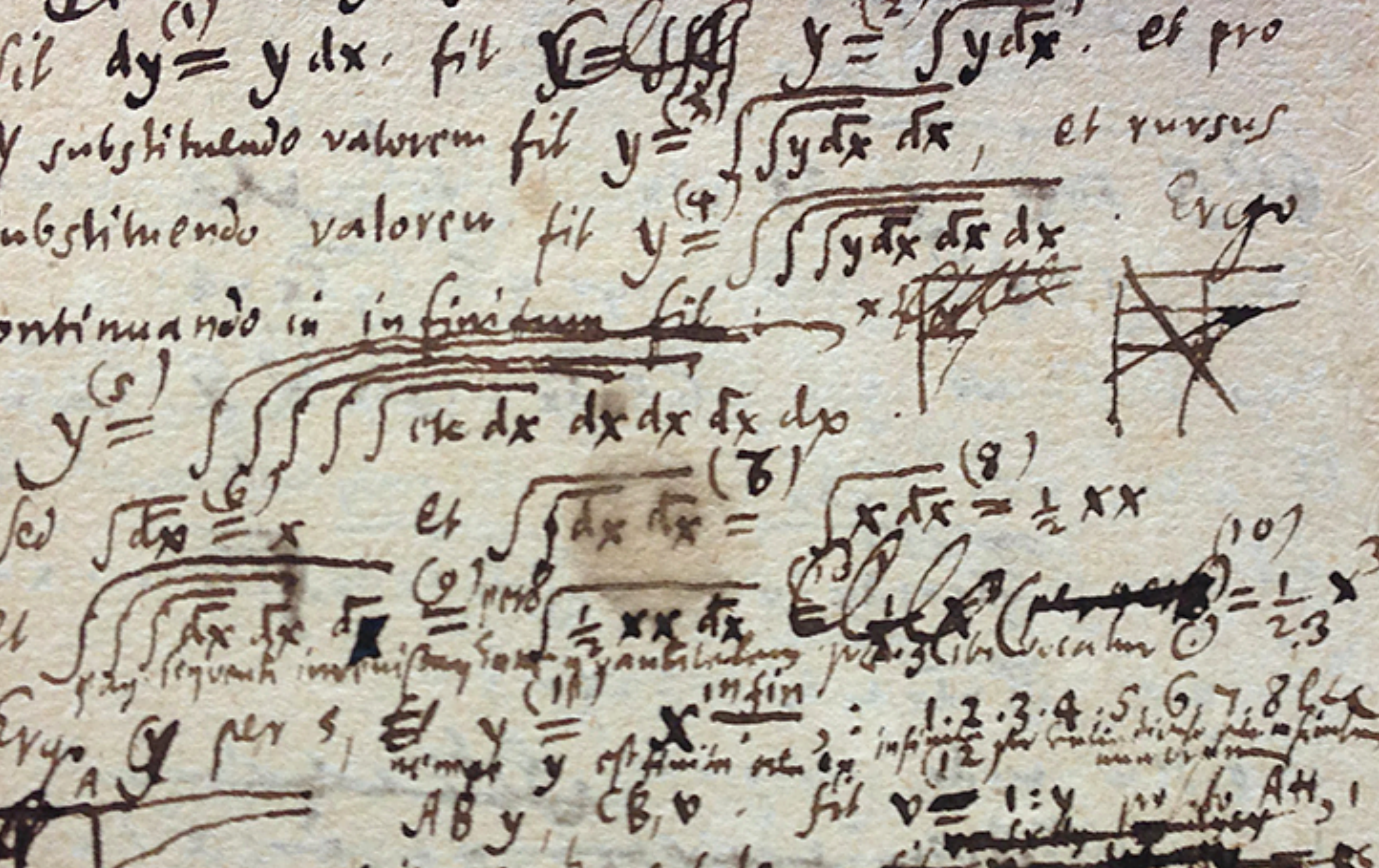


Leibnitz handwritten formulae from http://blog.stephenwolfram.com/2013/05/dropping-in-on-gottfried-leibniz/

# Solving non-linear equations numerically

ISC-3313

# Solving equations

How to find x? $$ax^2 + bx = -c$$

Finding the root $$ax^2 + bx + c = 0$$

## root finding

Web definitions

A root-finding algorithm is a numerical method, or algorithm, for finding a value x such that f(x) = 0, for a given function f. Such an x is called a root of the function f. This article is concerned with finding scalar, real or complex roots, approximated as floating point numbers. ...

http://en.wikipedia.org/wiki/Root_finding

# Finding the root

Rearrange formula

$$ax^2 + bx + c = 0$$

$$x^2 + \frac{b}{a}x + \frac{c}{a} = 0$$

$$x^2 + \frac{b}{a}x = -\frac{c}{a}$$

$$x^2 + \frac{b}{a}x + \left(\frac{b}{2a}\right)^2 = -\frac{c}{a} + \left(\frac{b}{2a}\right)^2$$

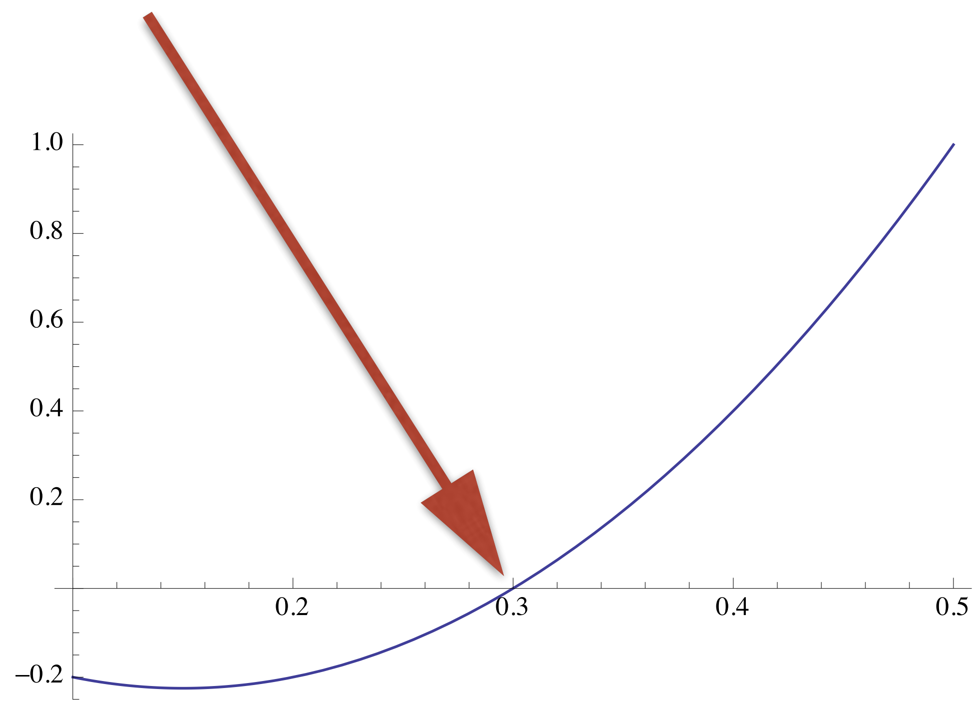
$$\left(x + \frac{b}{2a}\right)^2 = \frac{b^2 - ac}{4a^2}$$

$$x + \frac{b}{2a} = \pm\sqrt{\frac{b^2 - ac}{4a^2}}$$

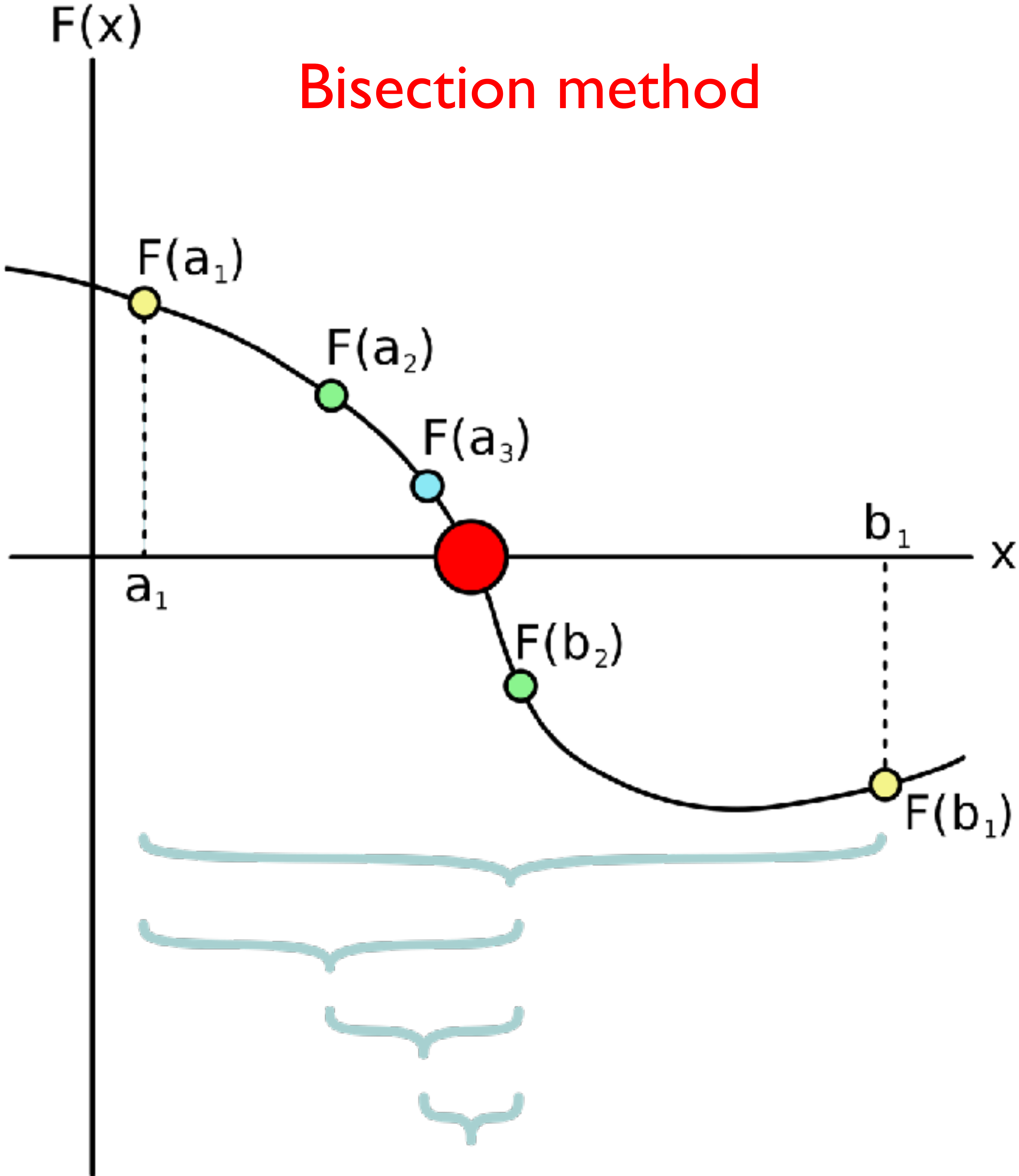
$$x = \frac{-b \pm \sqrt{b^2 - ac}}{2a}$$

# Finding the root

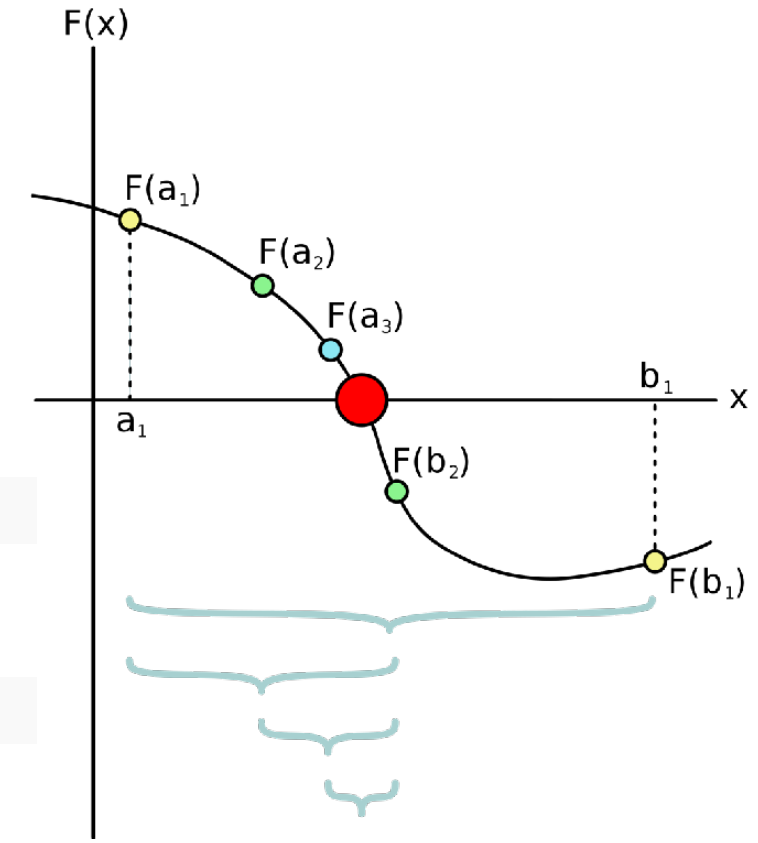
How to find x numerically? $$ax^2 + bx + c = 0$$



# Bisection method



# Bisection method




```
INPUT: Function f, endpoint values a, b, tolerance TOL,
maximum iterations NMAX

CONDITIONS: a < b, either f(a) < 0 and f(b) > 0 or f(a) >
0 and f(b) < 0

OUTPUT: value which differs from a root of f(x)=0 by less
than TOL
```

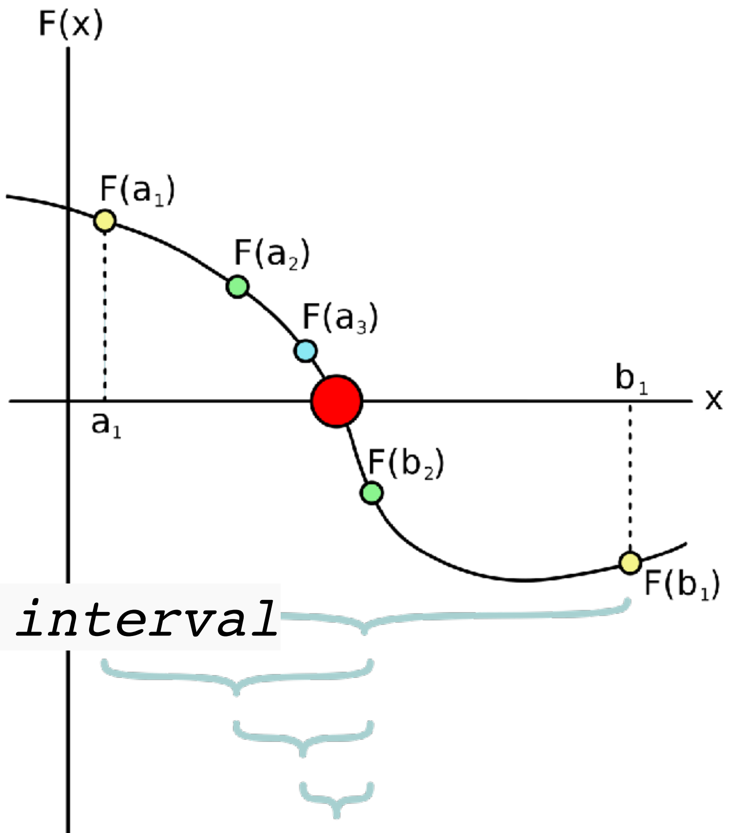
## Algorithm in pseudocode

```
N ← 1
While N ≤ NMAX # limit iterations to prevent infinite loop
  c ← (a + b)/2 # new midpoint
  If f(c) = 0 or (b – a)/2 < TOL then # solution found
    Output(c)
    Stop
  EndIf
  N ← N + 1 # increment step counter
  If sign(f(c)) = sign(f(a)) then a ← c else b ← c # new interval
EndWhile
Output("Method failed.") # max number of steps exceeded
```

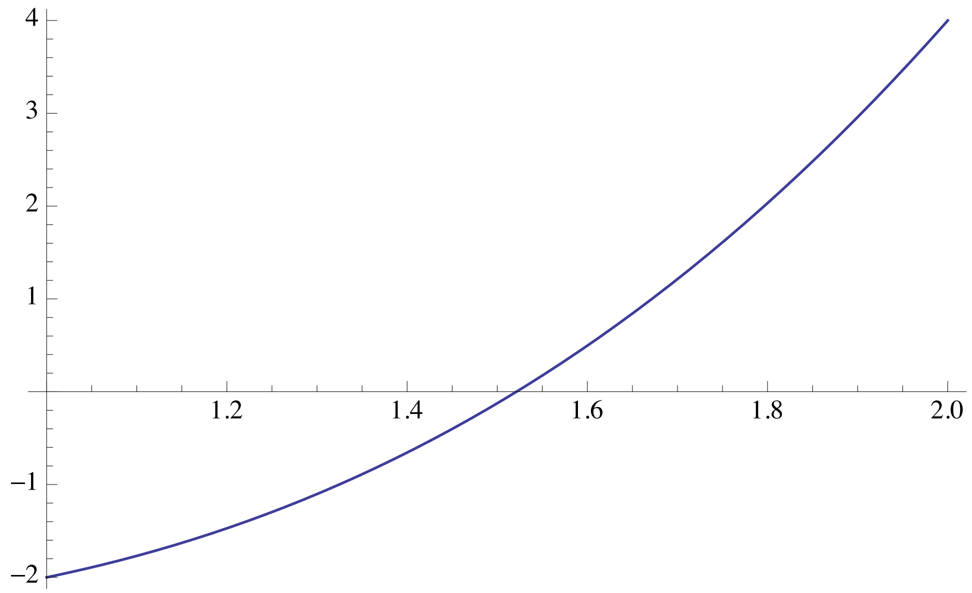
# Bisection method

Fill in the sequence of event during class

```
N ← 1
While N ≤ NMAX # limit iterations to prevent infinite loop
  c ← (a + b)/2 # new midpoint
  If f(c) = 0 or (b – a)/2 < TOL then # solution found
    Output(c)
    Stop
  EndIf
  N ← N + 1 # increment step counter
  If sign(f(c)) = sign(f(a)) then a ← c else b ← c # new interval
EndWhile
Output("Method failed.") # max number of steps exceeded
```



# Bisection method: Example




Suppose that the bisection method is used to find a root of the polynomial

$$f(x) = x^3 - x - 2$$

First, two numbers **a** and **b** have to be found such that **f(a)** and **f(b)** have opposite signs. For the above function, **a=1** and **b=2** satisfy this criterion, as

$$f(1) = 1^3 - 1 - 2 = -2$$

and

$$f(2) = 2^3 - 2 - 2 = 4$$

Because the function is continuous, there must be a root within the interval [1, 2].

In the first iteration, the end points of the interval which brackets the root are **a=1** and **b=2**, so the midpoint is **c=(a+b)/2 = (1+2)/2 = 1.5.** The function value at the midpoint is **-0.125**. Because **f(c)** is negative, **a=1** is replaced with **a=1.5** for the next iteration to ensure that **f(a)** and **f(b)** have opposite signs. As this continues, the interval between and will become increasingly smaller, converging on the root of the function.

| Iteration | $a_n$ | $b_n$ | $c_n$ | $f(c_n)$ |
|---|---|---|---|---|
| 1 | 1 | 2 | 1.5 | -0.125 |
| 2 | 1.5 | 2 | 1.75 | 1.6093750 |
| 3 | 1.5 | 1.75 | 1.625 | 0.6660156 |
| 4 | 1.5 | 1.625 | 1.5625 | 0.2521973 |
| 5 | 1.5 | 1.5625 | 1.5312500 | 0.0591125 |
| 6 | 1.5 | 1.5312500 | 1.5156250 | -0.0340538 |
| 7 | 1.5156250 | 1.5312500 | 1.5234375 | 0.0122504 |
| 8 | 1.5156250 | 1.5234375 | 1.5195313 | -0.0109712 |
| 9 | 1.5195313 | 1.5234375 | 1.5214844 | 0.0006222 |
| 10 | 1.5195313 | 1.5214844 | 1.5205078 | -0.0051789 |
| 11 | 1.5205078 | 1.5214844 | 1.5209961 | -0.0022794 |
| 12 | 1.5209961 | 1.5214844 | 1.5212402 | -0.0008289 |
| 13 | 1.5212402 | 1.5214844 | 1.5213623 | -0.0001034 |
| 14 | 1.5213623 | 1.5214844 | 1.5214233 | 0.0002594 |
| 15 | 1.5213623 | 1.5214233 | 1.5213928 | 0.0000780 |

# Newton Method

We assume that we have a function f(x) that is differentiable between the boundaries a and b, then we approximate f(x) using its tangent so that

$$y = f(x)$$

is approximated by

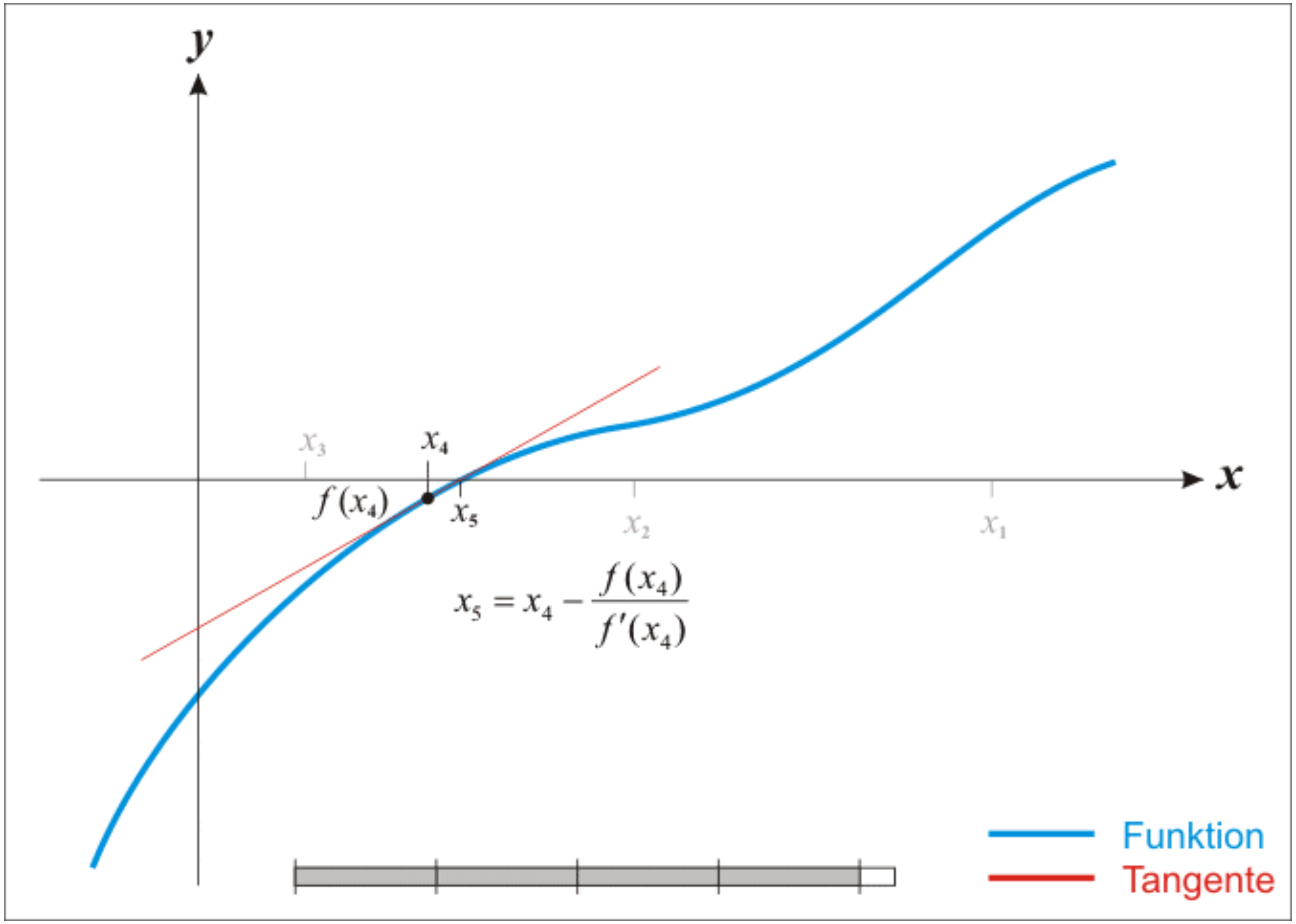
$$y = f'(x_n)(x - x_n) + f(x_n)$$

when we set y=0 then we can solve for x and name it $x_{n+1}$

$$0 = f'(x_n)(x - x_n) + f(x_n)$$

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

# Newton Method



# Newton Method: Example

### Square root of a number [edit]

Consider the problem of finding the square root of a number. There are many methods of computing square roots, and Newton's method is one.

For example, if one wishes to find the square root of 612, this is equivalent to finding the solution to

$$x^2 = 612$$

The function to use in Newton's method is then,

$$f(x) = x^2 - 612$$

with derivative,

$$f'(x) = 2x.$$

With an initial guess of 10, the sequence given by Newton's method is

$$\begin{array}{lclclcl}
x_1 & = & x_0 - \dfrac{f(x_0)}{f'(x_0)} & = & 10 - \dfrac{10^2 - 612}{2 \cdot 10} & = & 35.6 \\
x_2 & = & x_1 - \dfrac{f(x_1)}{f'(x_1)} & = & 35.6 - \dfrac{35.6^2 - 612}{2 \cdot 35.6} & = & \underline{2}6.395505617978\ldots \\
x_3 & = & \vdots & = & \vdots & = & \underline{24.7}90635492455\ldots \\
x_4 & = & \vdots & = & \vdots & = & \underline{24.7386}88294075\ldots \\
x_5 & = & \vdots & = & \vdots & = & \underline{24.7386337537}67\ldots
\end{array}$$

Where the correct digits are underlined. With only a few iterations one can obtain a solution accurate to many decimal places.

# Trouble with finding the root

- bisection will always find the root when the two boundaries are on opposite sites of zero.

- bisection is slow, convergence is linear

- Newton method is converging very fast when started near the root, in fact its precision improves quadratically.

- Newton method only works when the function has a derivative in all points and when no minima or maxima are near the root.

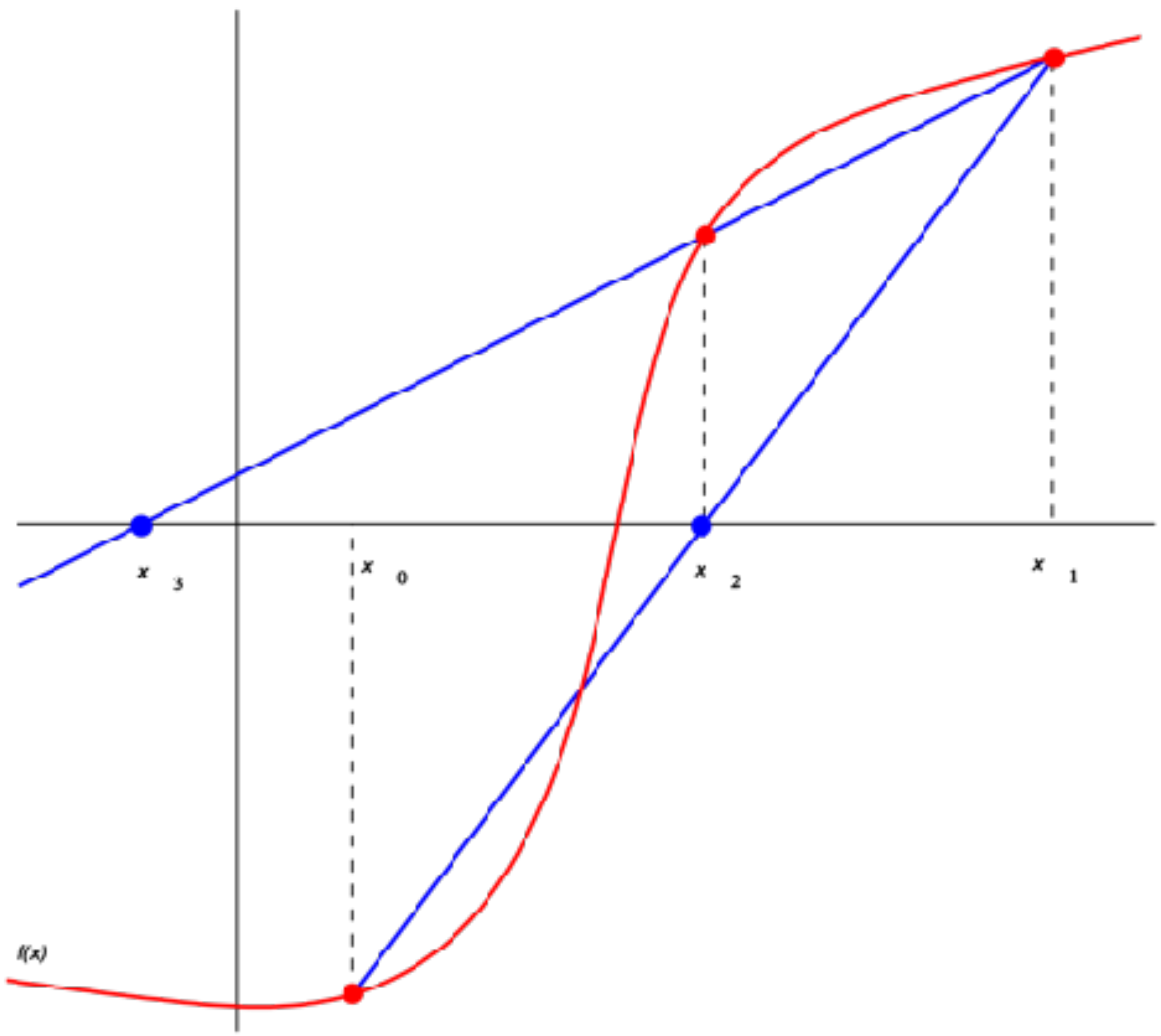
- Newton can be trapped when the derivative is zero (e.g. at a minimum) then the recursion

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

  will fail. Problems also occur with when cycles lead one update to an earlier value.

- Many hybrid methods that combine the fast convergence of the newton method with the certainly of convergence of the bisection method.

# Secant Method



# Secant Method

http://mathfaculty.fullerton.edu/mathews//a2001/Animations/RootFinding/SecantMethod/Secantff.html