Roots of Equations

- Real roots of single variable function.
- Closed domain methods (bracketing).

Examples of nonlinear equations

- Equations of a single variable:

$$x^2 - 6x + 9 = 0$$
$$x^2 - \cos(x) = 0$$
$$\exp(x)\ln(x^2) - x\cos(x) = 0$$

- Equations of two variables:

$$y(x^3 - 1) = x^4$$
$$x^2 + y^2 = 1$$

Introduction

Given a continuous non linear function $f(x)$, find the value $x = c$ such that $f(c) = 0$.

Introduction

Given a continuous non linear function $f(x)$, find the value $x = c$ such that $f(c) = 0$.

The non-linear equation $f(x) = 0$ may be a:

- An algebraic equation (roots of polynomials).

Introduction

Given a continuous non linear function $f(x)$, find the value $x = c$ such that $f(c) = 0$.

The non-linear equation $f(x) = 0$ may be a:

- An algebraic equation (roots of polynomials).
- A transcendental equation.

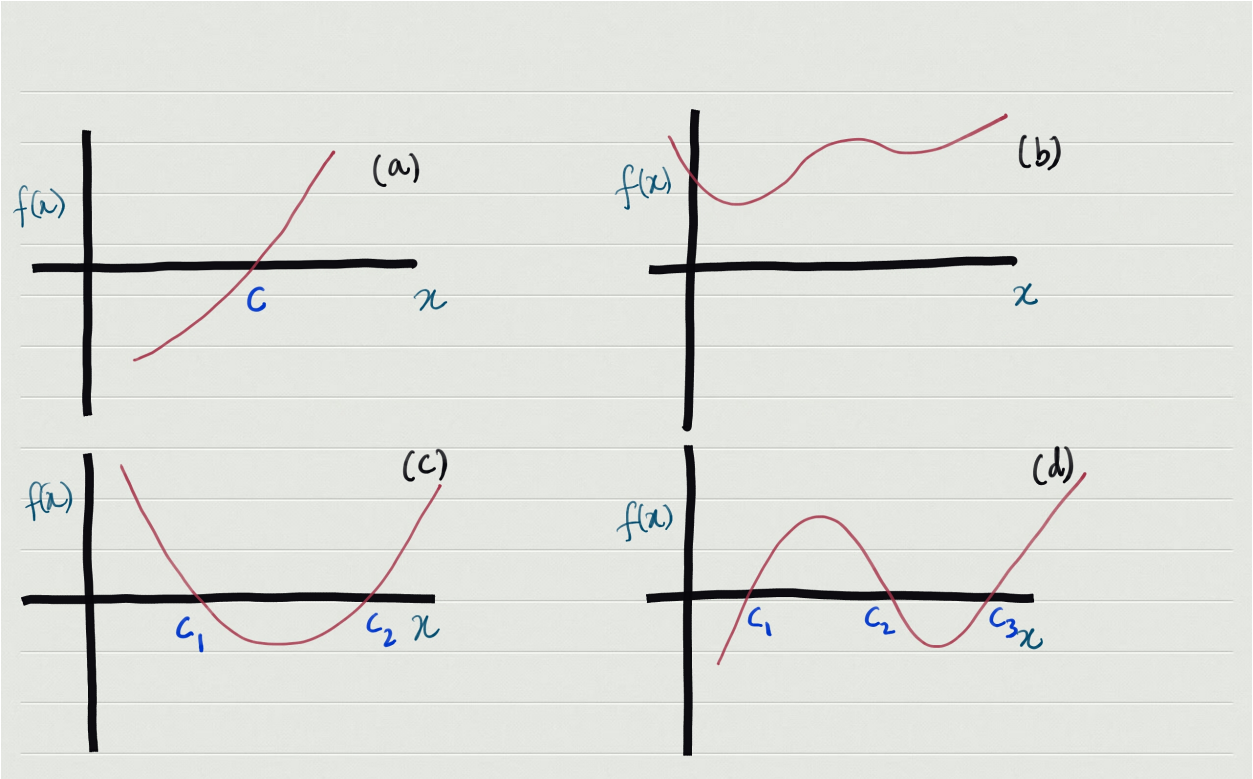
Introduction

Given a continuous non linear function $f(x)$, find the value $x = c$ such that $f(c) = 0$.

The non-linear equation $f(x) = 0$ may be a:

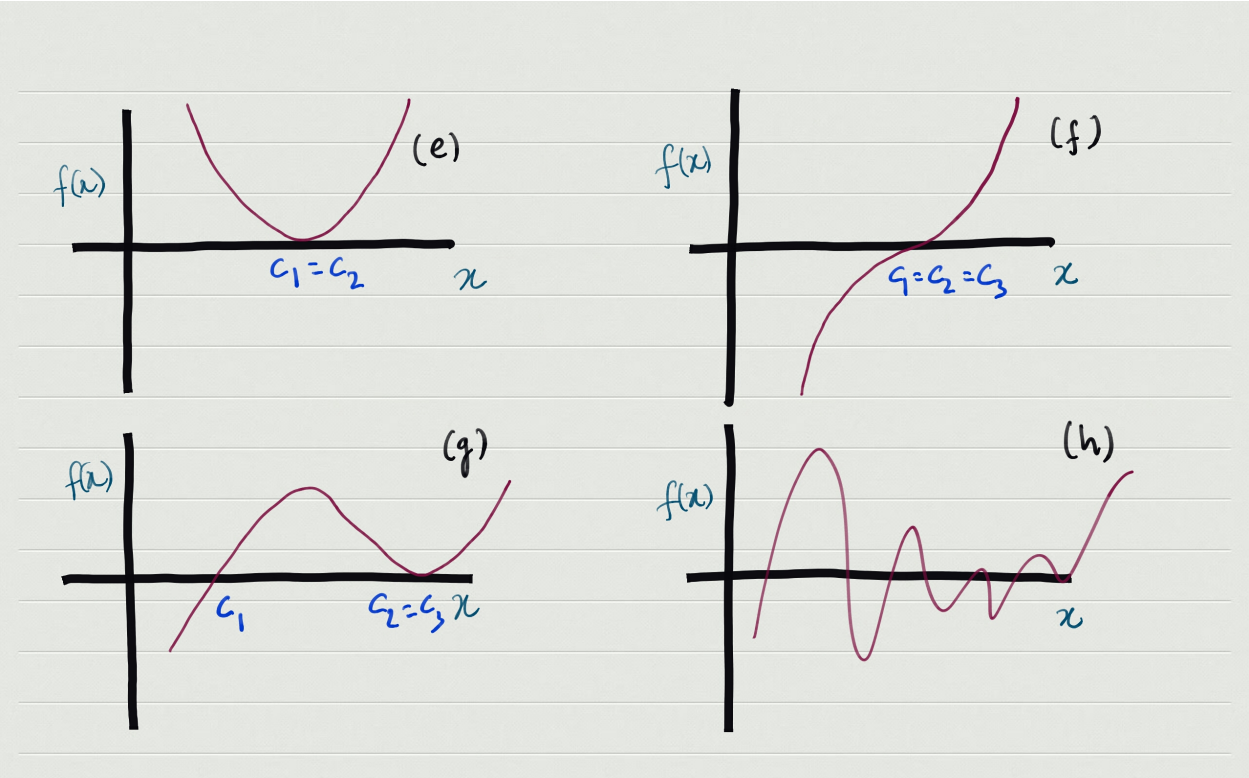
- An algebraic equation (roots of polynomials).
- A transcendental equation.
- ...

Behaviour of non-linear functions



(a) A single root.
(b) No real roots exist (complex roots might).
(c) Two simple roots.
(d) Three simple roots.

Behaviour of non-linear functions



(e) Two multiple roots.

(f) Three multiple roots.

(g) One simple root and two multiple roots.

(h) Multiple roots.

Some prelimenaries

- All non-linear equations have to be solved iteratively!

Some prelimenaries

- All non-linear equations have to be solved iteratively!
- We always start out with a guess – an approximate root.

Some prelimenaries

- All non-linear equations have to be solved iteratively!
- We always start out with a guess – an approximate root.
- Starting from the guess we iterate – the better the guess, the better our chances of converging to the solution and fewer the number of iterations required to converge to the solution.

Some prelimenaries

- All non-linear equations have to be solved iteratively!
- We always start out with a guess – an approximate root.
- Starting from the guess we iterate – the better the guess, the better our chances of converging to the solution and fewer the number of iterations required to converge to the solution.
- So, it is crucial to have a good guess...

Bounding the solution

- Bounding the solution involves finding a rough estimate of the solution that can be used as in initial guess in an iterative process that refines the solution to a specified tolerance.

Bounding the solution

- Bounding the solution involves finding a rough estimate of the solution that can be used as in initial guess in an iterative process that refines the solution to a specified tolerance.
- If possible, the root should be bracketed between two points at which the value of the non-linear function changes sign!

Bounding the solution

- Getting a good guess – seems like a chicken and egg problem – but not so.

Bounding the solution

- Getting a good guess – seems like a chicken and egg problem – but not so.
- GRAPH the functions to get an idea!! This is the best way to get an idea of the solution and get an approximate solution. In case of multiple solutions it also ensures that you have the solution that you want!

.

Bounding the solution

- Getting a good guess – seems like a chicken and egg problem – but not so.
- GRAPH the functions to get an idea!! This is the best way to get an idea of the solution and get an approximate solution. In case of multiple solutions it also ensures that you have the solution that you want! *People who omit the sketch throw away the most important problem solving tool they possess – geometric intuition.*

Bounding the solution

- Getting a good guess – seems like a chicken and egg problem – but not so.
- GRAPH the functions to get an idea!! This is the best way to get an idea of the solution and get an approximate solution. In case of multiple solutions it also ensures that you have the solution that you want! *People who omit the sketch throw away the most important problem solving tool they possess – geometric intuition.*
- Incremental search – ie start with different guesses and search.

Bounding the solution

- Getting a good guess – seems like a chicken and egg problem – but not so.
- GRAPH the functions to get an idea!! This is the best way to get an idea of the solution and get an approximate solution. In case of multiple solutions it also ensures that you have the solution that you want! *People who omit the sketch throw away the most important problem solving tool they possess – geometric intuition.*
- Incremental search – ie start with different guesses and search.
- Past experience with the problem or a similar one.

Bounding the solution

- Getting a good guess – seems like a chicken and egg problem – but not so.
- GRAPH the functions to get an idea!! This is the best way to get an idea of the solution and get an approximate solution. In case of multiple solutions it also ensures that you have the solution that you want! *People who omit the sketch throw away the most important problem solving tool they possess – geometric intuition.*
- Incremental search – ie start with different guesses and search.
- Past experience with the problem or a similar one.
- Solution of a simplified approximate model.

Bounding the solution

- Getting a good guess – seems like a chicken and egg problem – but not so.
- GRAPH the functions to get an idea!! This is the best way to get an idea of the solution and get an approximate solution. In case of multiple solutions it also ensures that you have the solution that you want! *People who omit the sketch throw away the most important problem solving tool they possess – geometric intuition.*
- Incremental search – ie start with different guesses and search.
- Past experience with the problem or a similar one.
- Solution of a simplified approximate model.
- Previous solution in a sequence of solutions.

Iterative refinement of the solution

- Iterative refining the solution involves determining the solution to a specified tolerance by a systematic procedure.

Iterative refinement of the solution

- Iterative refining the solution involves determining the solution to a specified tolerance by a systematic procedure.
- *There are numerous pitfalls in finding the roots of non linear equations*.

Iterative refinement of the solution

- Iterative refining the solution involves determining the solution to a specified tolerance by a systematic procedure.
- *There are numerous pitfalls in finding the roots of non linear equations*.
- An important question is when to stop the iteration –

$$\text{Absolute error:} |f_{i+1} - f_i|$$

$$\text{Relative error:} \left|\frac{f_{i+1} - f_i}{f_{i+1}}\right|$$

Types of Methods

There are two types of methods for finding roots of non linear equations:

- Closed domain (bracketing) methods.
- Open domain (non-bracketing) methods.

Closed domain (bracketing) method

- These are methods that start with two values of x, a and b, which bracket the root in the interval $[a, b]$.

Closed domain (bracketing) method

- These are methods that start with two values of x, a and b, which bracket the root in the interval $[a, b]$.
- If $f(a)$ and $f(b)$ have opposite signs (and if the function is continuous), then there must be atleast one root in $[a, b]$.

Closed domain (bracketing) method

- These are methods that start with two values of x, a and b, which bracket the root in the interval $[a, b]$.
- If $f(a)$ and $f(b)$ have opposite signs (and if the function is continuous), then there must be atleast one root in $[a, b]$.
- Most common closed domain methods:
  - Bisection method – interval halving.
  - False position method.

Closed domain (bracketing) method

- These are methods that start with two values of x, a and b, which bracket the root in the interval $[a, b]$.
- If $f(a)$ and $f(b)$ have opposite signs (and if the function is continuous), then there must be atleast one root in $[a, b]$.
- Most common closed domain methods:
  - Bisection method – interval halving.
  - False position method.
- In general, bracketing methods are quite robust – ie they are guaranteed to give a solution as the solution is bracketted in the interval.

Bisection Method

This is the simplest and the most robust method!

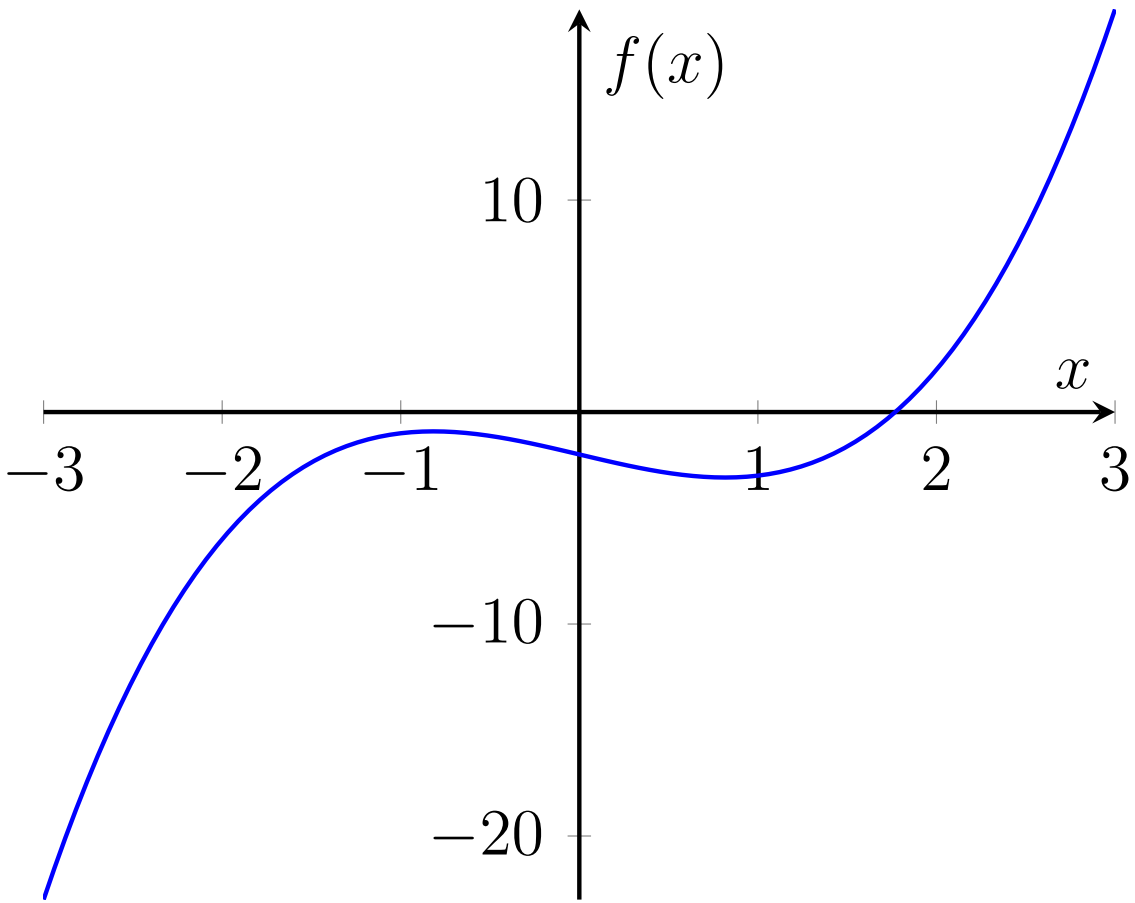
- $f(x)$ a continuous function on $[a, b]$.
- $f(x)$ changes sign between a and b ie $f(a)f(b) < 0$.

Bisection Method

This is the simplest and the most robust method!

- $f(x)$ a continuous function on $[a, b]$.
- $f(x)$ changes sign between a and b ie $f(a)f(b) < 0$.

Consider an example: $f(x) = x^3 - 2x - 2$



Bisection Method: Algorithm

- Divide $[a, b]$ into two equal parts with $c = \frac{a+b}{2}$.

Bisection Method: Algorithm

- Divide $[a, b]$ into two equal parts with $c = \frac{a+b}{2}$.
-

$$\begin{aligned} &< 0 \text{ there is a root in } [a, c] \implies a = a, b = c \\ f(a)f(c) &> 0 \text{ there is a root in } [b, c] \implies a = c, b = b \\ &= 0 \text{ root is at } c \end{aligned}$$

Bisection Method: Algorithm

- Divide $[a, b]$ into two equal parts with $c = \frac{a+b}{2}$.
-

$$\begin{aligned} &< 0 \text{ there is a root in } [a, c] \implies a = a, b = c \\ f(a)f(c) &> 0 \text{ there is a root in } [b, c] \implies a = c, b = b \\ &= 0 \text{ root is at } c \end{aligned}$$

- Interval halving is an iterative procedure.

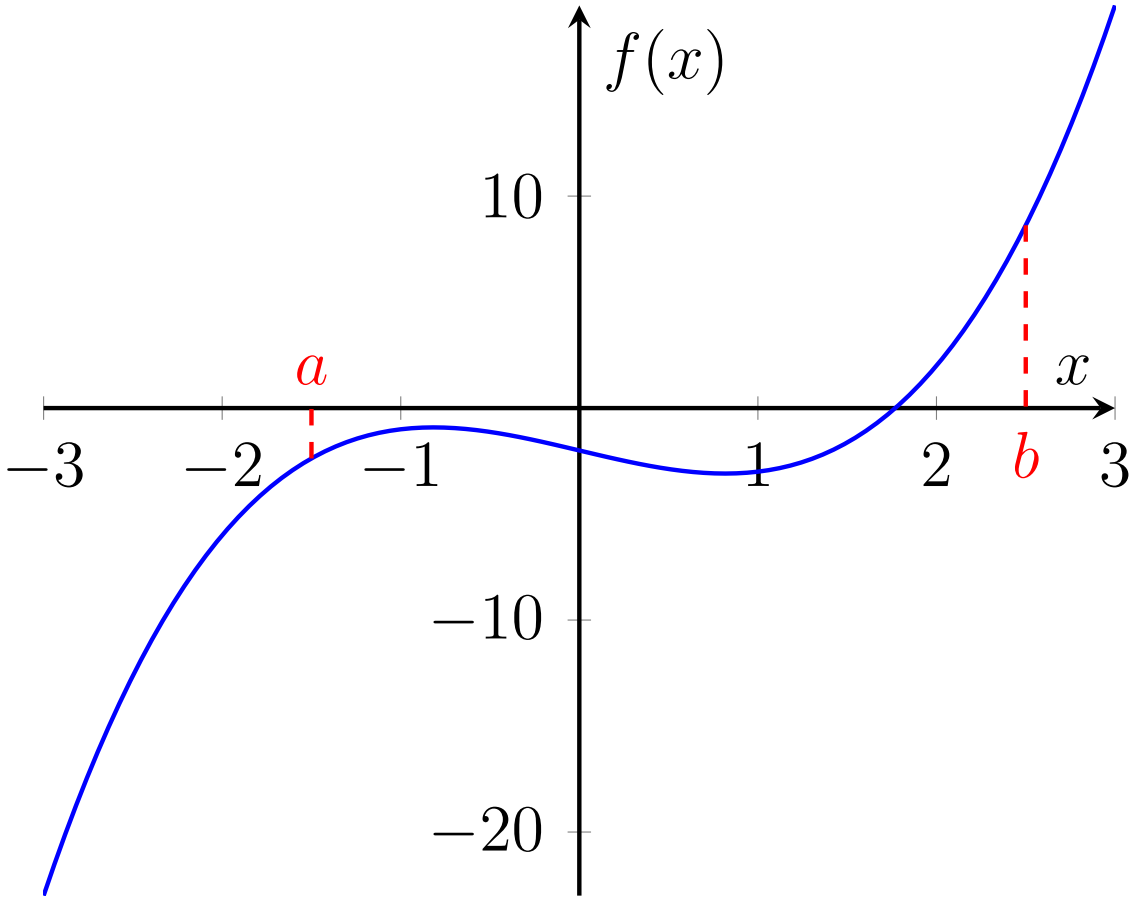
Bisection Method: Algorithm

- Divide $[a, b]$ into two equal parts with $c = \frac{a+b}{2}$.
-

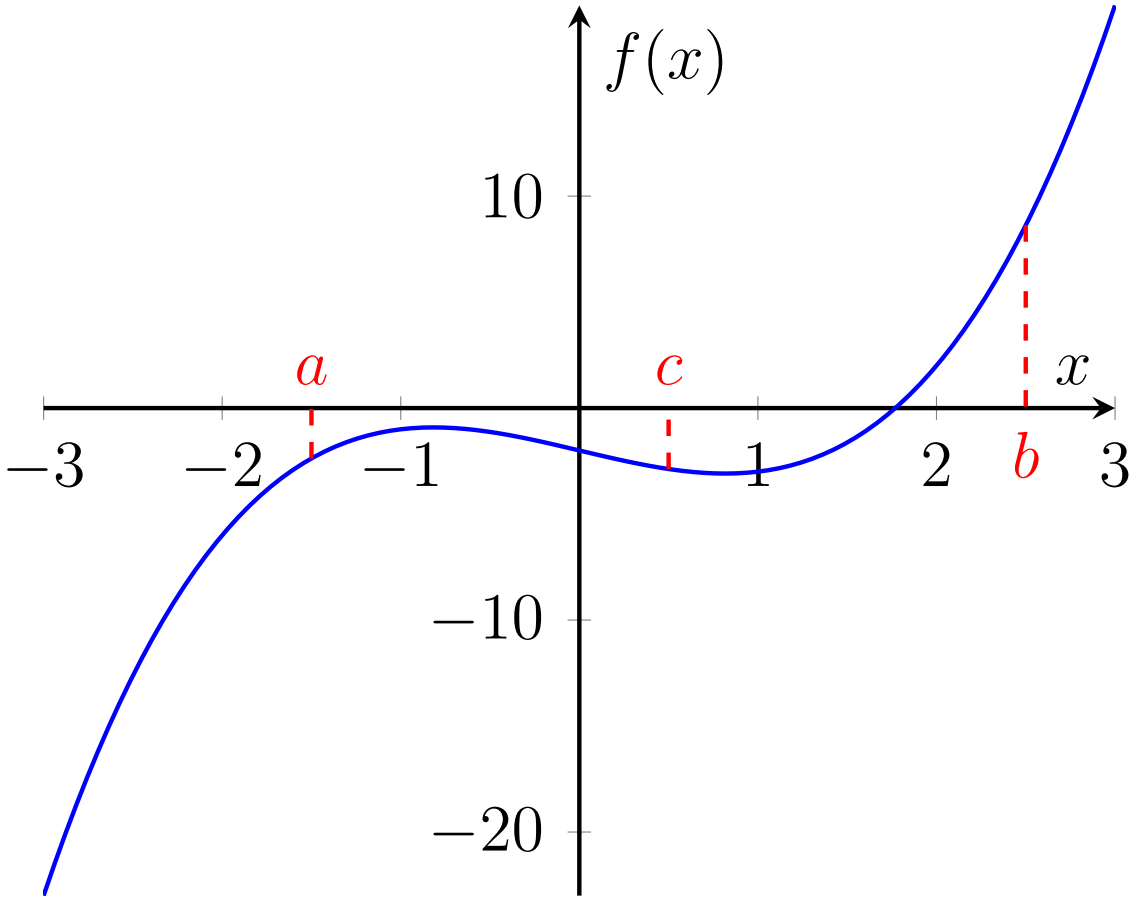
$$\begin{aligned} &< 0 \text{ there is a root in } [a, c] \implies a = a, b = c \\ f(a)f(c) &> 0 \text{ there is a root in } [b, c] \implies a = c, b = b \\ &= 0 \text{ root is at } c \end{aligned}$$

- Interval halving is an iterative procedure.
- Continue iterations till $|b - a| < \text{tol}$ or $|f(c)| < \text{tol}$ or both

Bisection Method: Example



Bisection Method: Example



Bisection Method: analysis

- Each iteration reduces the original interval $[a, b]$ by a factor of 2. After n iterations, the size of the interval is $\frac{(b-a)}{2^n}$.

Bisection Method: analysis

- Each iteration reduces the original interval $[a, b]$ by a factor of 2. After n iterations, the size of the interval is $\frac{(b-a)}{2^n}$.
- As a result, the root remains bracketed and the method is guaranteed to converge.

Bisection Method: analysis

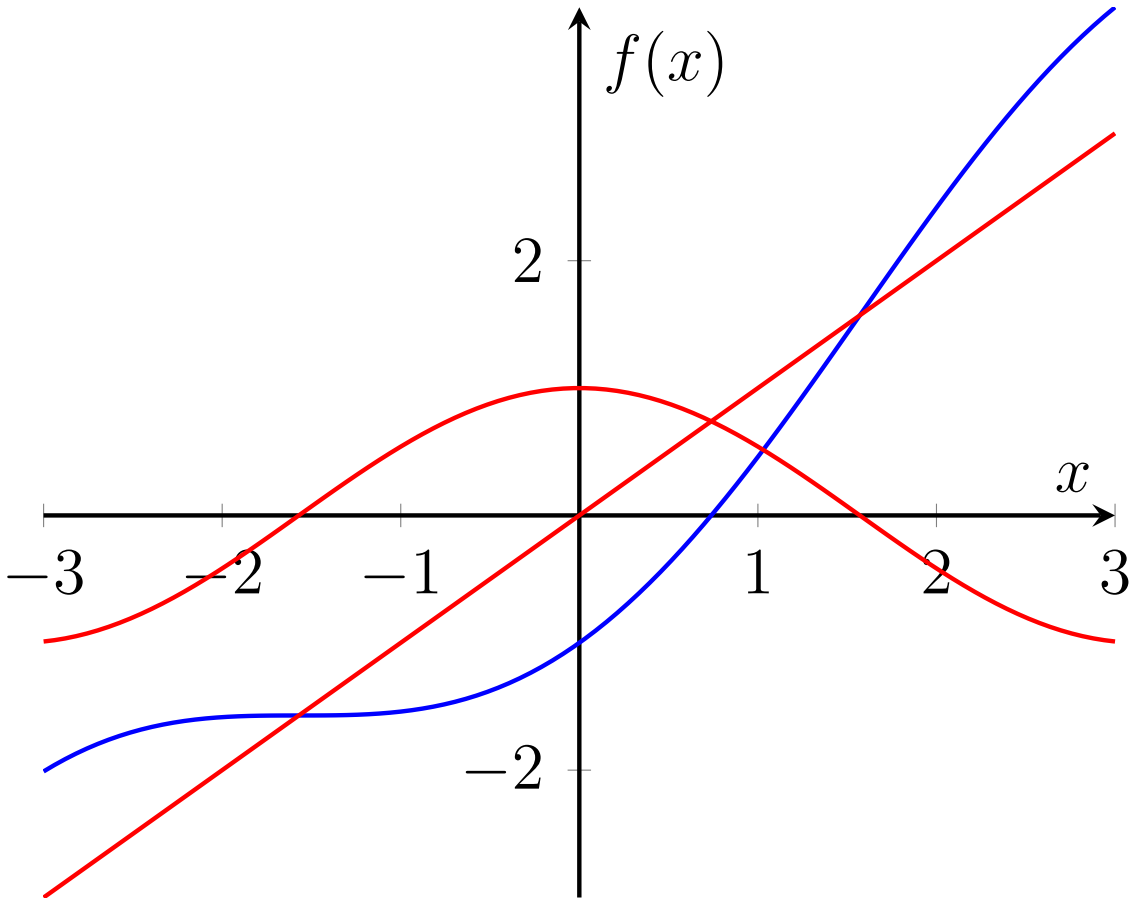
- Each iteration reduces the original interval $[a, b]$ by a factor of 2. After n iterations, the size of the interval is $\frac{(b-a)}{2^n}$.
- As a result, the root remains bracketed and the method is guaranteed to converge.
- However, the convergence of the method can be quite slow.

Bisection Method: analysis

- Each iteration reduces the original interval $[a, b]$ by a factor of 2. After n iterations, the size of the interval is $\frac{(b-a)}{2^n}$.
- As a result, the root remains bracketed and the method is guaranteed to converge.
- However, the convergence of the method can be quite slow.
- Notice that this method does not use any information about the function's behaviour!

Bisection method: Example

Consider an example: $f(x) = x - \cos(x)$



Bisection method: example

i	a	f(a)	b	f(b)	c	f(c)
1	0.000000	-1.000000	4.000000	4.653644	2.000000	2.416147
2	0.000000	-1.000000	2.000000	2.416147	1.000000	0.459698
3	0.000000	-1.000000	1.000000	0.459698	0.500000	-0.377583
4	0.500000	-0.377583	1.000000	0.459698	0.750000	0.018311
5	0.500000	-0.377583	0.750000	0.018311	0.625000	-0.185963
6	0.625000	-0.185963	0.750000	0.018311	0.687500	-0.085335
7	0.687500	-0.085335	0.750000	0.018311	0.718750	-0.033879
8	0.718750	-0.033879	0.750000	0.018311	0.734375	-0.007875
9	0.734375	-0.007875	0.750000	0.018311	0.742188	0.005196
10	0.734375	-0.007875	0.742188	0.005196	0.738281	-0.001345
11	0.738281	-0.001345	0.742188	0.005196	0.740234	0.001924
12	0.738281	-0.001345	0.740234	0.001924	0.739258	0.000289
13	0.738281	-0.001345	0.739258	0.000289	0.738770	-0.000528
14	0.738770	-0.000528	0.739258	0.000289	0.739014	-0.000120
15	0.739014	-0.000120	0.739258	0.000289	0.739136	0.000085
16	0.739014	-0.000120	0.739136	0.000085	0.739075	-0.000017
17	0.739075	-0.000017	0.739136	0.000085	0.739105	0.000034
18	0.739075	-0.000017	0.739105	0.000034	0.739090	0.000008
19	0.739075	-0.000017	0.739090	0.000008	0.739082	-0.000005
20	0.739082	-0.000005	0.739090	0.000008	0.739086	0.000002
21	0.739082	-0.000005	0.739086	0.000002	0.739084	-0.000001
22	0.739084	-0.000001	0.739086	0.000002	0.739085	0.000000

Root is: 0.739085197449

False Position Method

- In this method, the function $f(x)$ is assumed to be a linear function $g(x)$ in the interval $[a, b]$.

False Position Method

- In this method, the function $f(x)$ is assumed to be a linear function $g(x)$ in the interval $[a, b]$.
- The root of the linear function $g(x), x = c$ is taken to be the next approximation of the root of the original function $f(x)$.

False Position Method

- In this method, the function $f(x)$ is assumed to be a linear function $g(x)$ in the interval $[a, b]$.
- The root of the linear function $g(x), x = c$ is taken to be the next approximation of the root of the original function $f(x)$.
- The root of the linear function, $g(x)$, $x = c$ is not the root of the non linear function $f(x)$. It is a *false position* and hence the name.

False Position Method

- In this method, the function $f(x)$ is assumed to be a linear function $g(x)$ in the interval $[a, b]$.
- The root of the linear function $g(x), x = c$ is taken to be the next approximation of the root of the original function $f(x)$.
- The root of the linear function, $g(x)$, $x = c$ is not the root of the non linear function $f(x)$. It is a *false position* and hence the name.
- Unlike bisection, this method uses some information about the function $f(x)$.

False position method: Algorithm

The slope of linear function, $g'(x)$, is given by:

$$g'(x) = \frac{f(b) - f(a)}{b - a}$$

Assuming $f(c) = 0$, one can also write $g'(x)$ as:

$$g'(x) = \frac{f(b) - f(c)}{b - c} \implies c = b - \frac{f(b)}{g'(x)}$$

Combining with the first equation:

$$c = b - f(b) * \frac{b - a}{f(b) - f(a)}$$

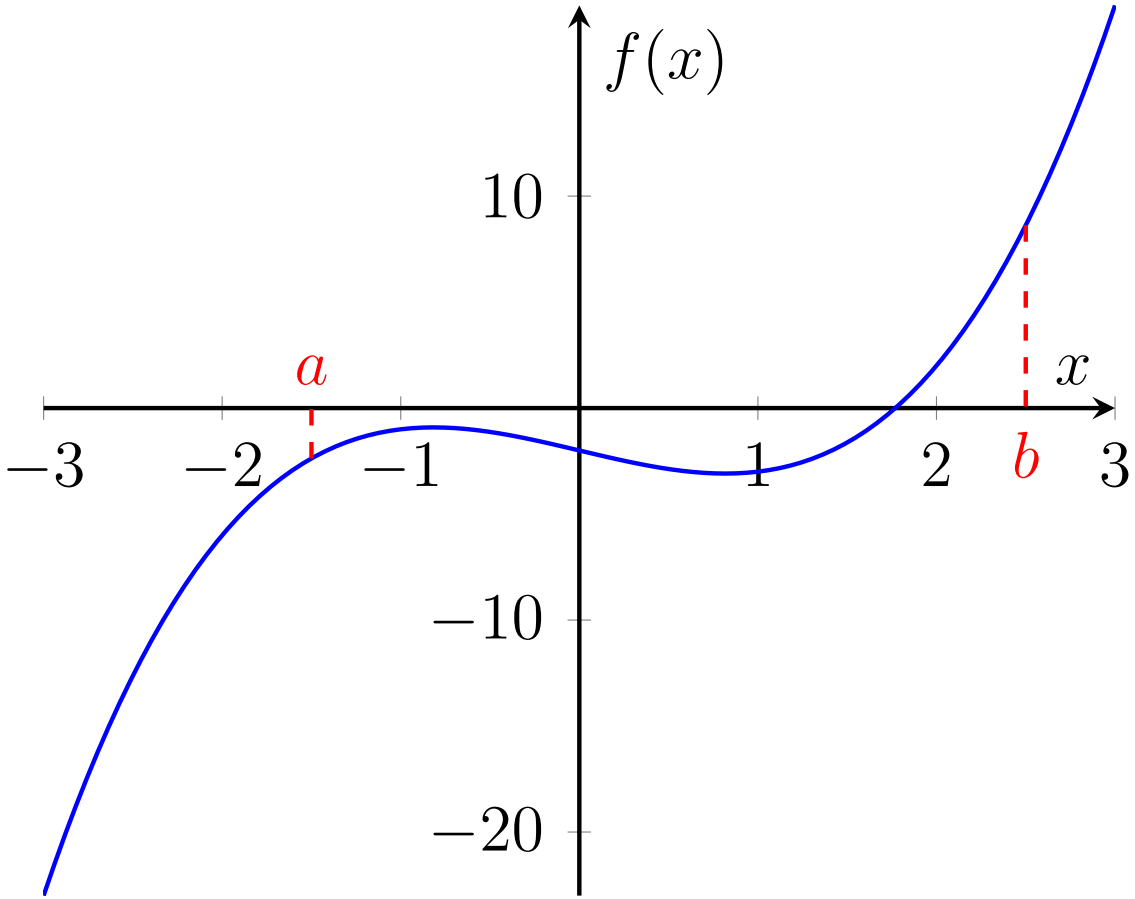
$$c = \frac{a * f(b) - b * f(a)}{f(b) - f(a)}$$

Then just like the bisection method:

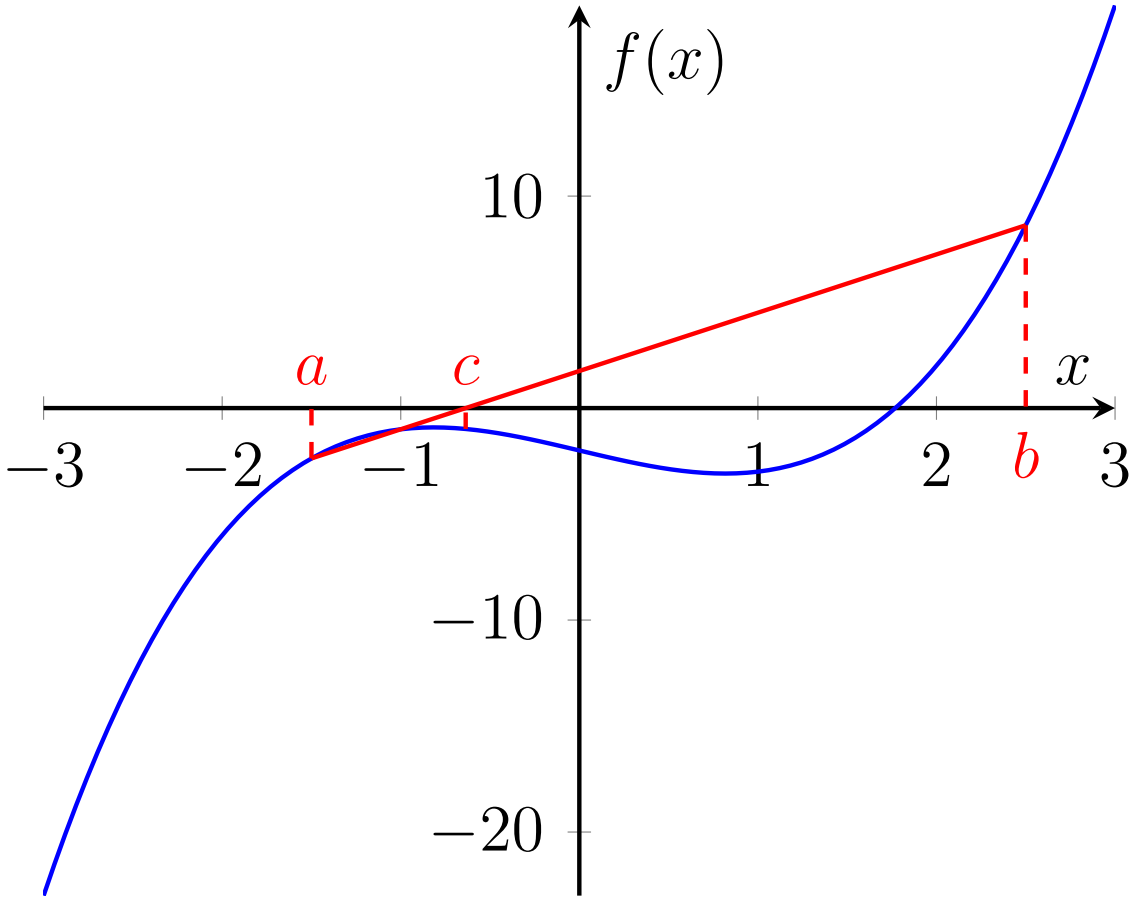
$$\text{if } f(a) * f(c) < 0 \; a = a, b = c$$

$$\text{if } f(a) * f(c) > 0 \; a = c, b = b$$

False position method: Example



False position method: Example



False position method: example

i	a	f(a)	b	f(b)	c	f(c)
1	0.000000	-1.000000	4.000000	4.653644	0.707508	-0.052475
2	0.707508	-0.052475	4.000000	4.653644	0.744221	0.008605
3	0.707508	-0.052475	0.744221	0.008605	0.739049	-0.000061
4	0.739049	-0.000061	0.744221	0.008605	0.739085	-0.000000

```
Root is: 0.739085092149
```

False position used 4 iterations compared to 22 in bisection!
Generally false position converges much faster than bisection.

Open domain methods

- Newton's iteration.
- Method of secants.
- Muller's method.
- Fixed point iteration.

Open domain methods

- Methods do not restrict the root to remain trapped in a closed interval.

Open domain methods

- Methods do not restrict the root to remain trapped in a closed interval.
- As a result, these methods are not as *robust* as bracketing methods – these methods may not converge at all.

Open domain methods

- Methods do not restrict the root to remain trapped in a closed interval.
- As a result, these methods are not as *robust* as bracketing methods – these methods may not converge at all.
- They use information about the non linear function to refine the estimates of the root – can be considerably more efficient than bracketing methods.

Newton's method

- This method is also called the Newton–Raphson method.

Newton's method

- This method is also called the Newton–Raphson method.
- This method uses the derivatives $f'(x)$ of the function $f(x)$ to accelerate convergence for solving $f(x) = 0$.

Newton's method

- This method is also called the Newton–Raphson method.
- This method uses the derivatives $f'(x)$ of the function $f(x)$ to accelerate convergence for solving $f(x) = 0$.
- It always converges *if the initial approximation is sufficiently close to the root*. Its rate of convergence is quadratic!

Newton's method

- This method is also called the Newton–Raphson method.
- This method uses the derivatives $f'(x)$ of the function $f(x)$ to accelerate convergence for solving $f(x) = 0$.
- It always converges *if the initial approximation is sufficiently close to the root*. Its rate of convergence is quadratic!
- However, it not only needs $f(x)$, it also needs the derivative $f'(x)$.

Newton's method: Key idea

A continuous function $f(x)$ can be expanded around a point x is terms of a Taylor's series:

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

Newton's method: Key idea

A continuous function $f(x)$ can be expanded around a point x is terms of a Taylor's series:

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

As we want the solution for $f(x) = 0$, we can write the following:

$$f(x) = 0 = f(x_0) + (x - x_0)f'(x_0)$$

neglecting higher order terms.

Newton's method: Key idea

A continuous function $f(x)$ can be expanded around a point x is terms of a Taylor's series:

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

As we want the solution for $f(x) = 0$, we can write the following:

$$f(x) = 0 = f(x_0) + (x - x_0)f'(x_0)$$

neglecting higher order terms.
Using this we can write:

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

Newton's method: Algorithm

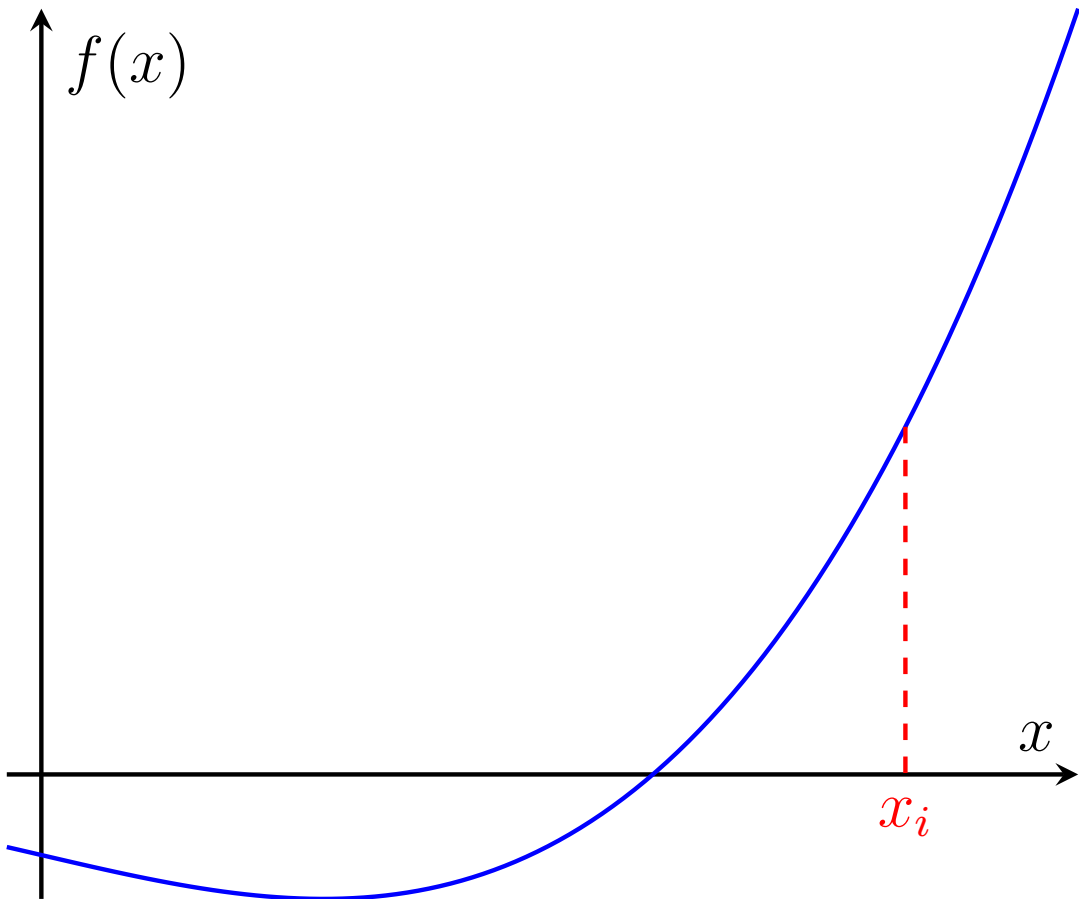
The iterations use $f(x)$ and $f'(x)$ to proceed in the following way:

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

Newton's method: Algorithm

The iterations use $f(x)$ and $f'(x)$ to proceed in the following way:

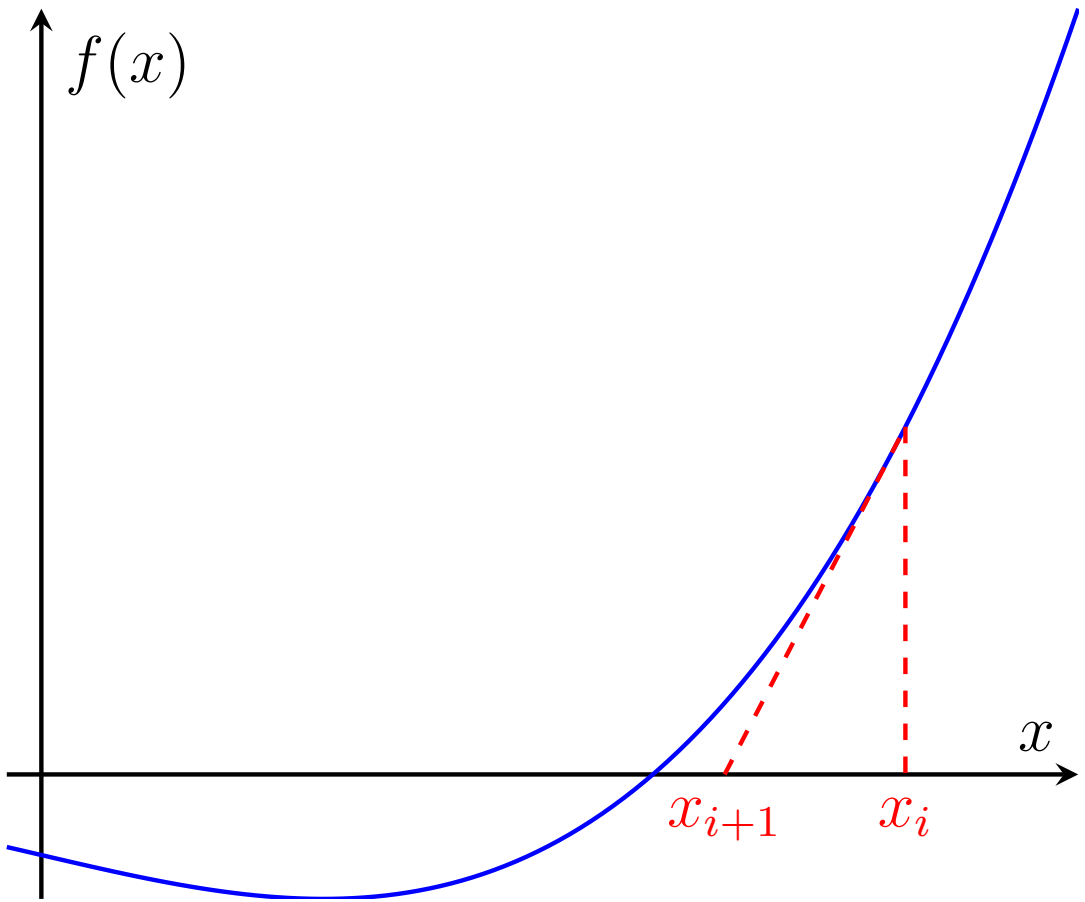
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



Newton's method: Algorithm

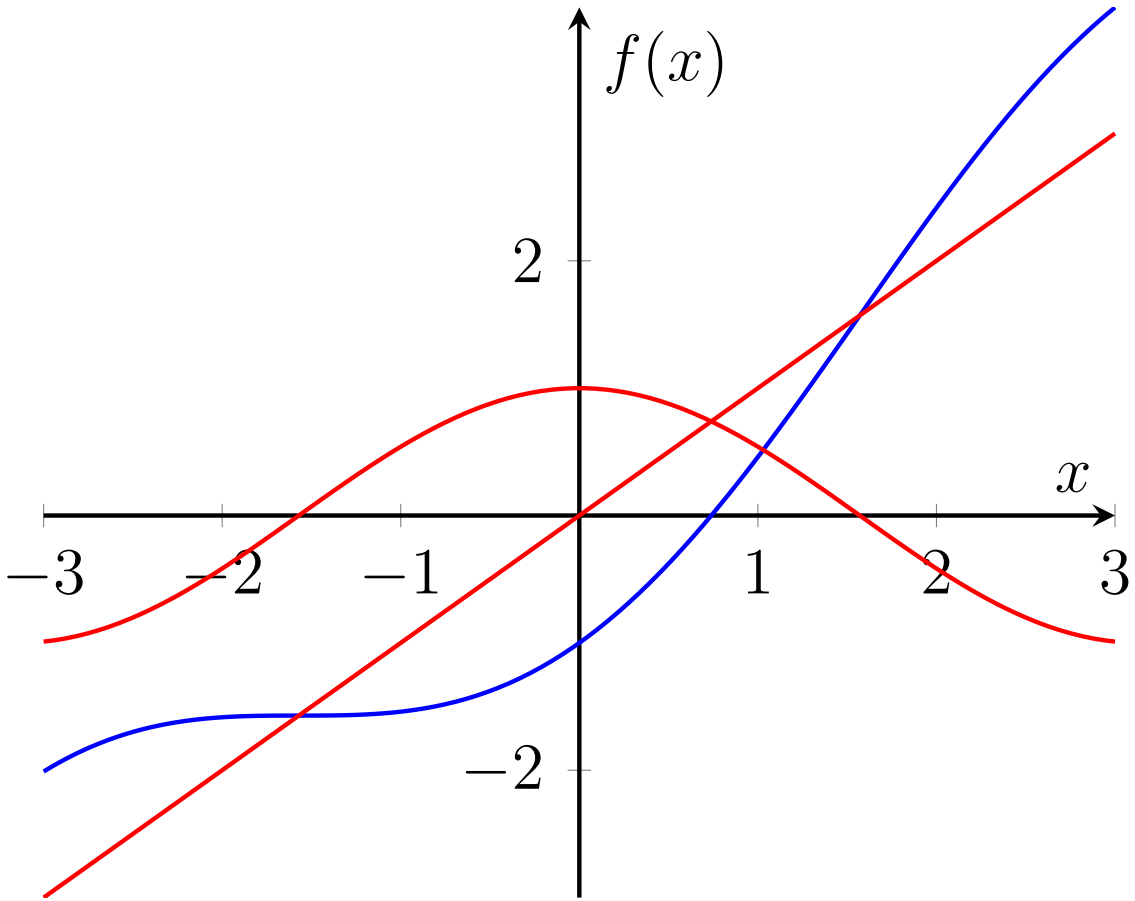
The iterations use $f(x)$ and $f'(x)$ to proceed in the following way:

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



Newton's method: Example

Consider an example: $f(x) = x - \cos(x)$



Initial guess $x = 1.0$.

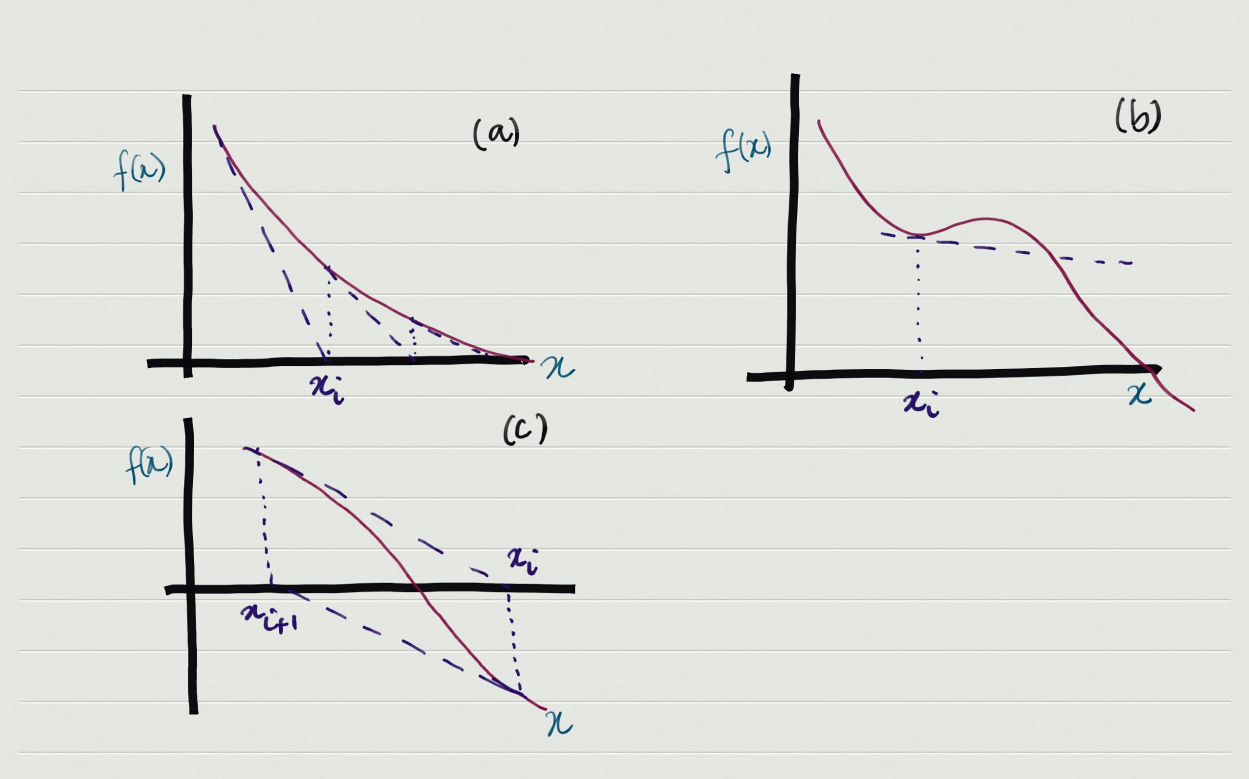
Newton's method: example

```
  i     x_i       f(x_i)     f'(x_i)   x_{i+1}   f(x_{i+1})
  1   1.000000  0.459698   1.841471   0.750364   0.018923
  2   0.750364  0.018923   1.681905   0.739113   0.000046
  3   0.739113  0.000046   1.673633   0.739085   0.000000

Root is: 0.739085133385
```

- False position used 4 iterations and bisection – 22 iterations!
- Newton's method has excellent local convergence properties.
- However, global convergence properties can be quite poor – due to neglect of higher order terms in the Taylor series expansion.

Possible problems



(a) Very slow approach to the root – $f'(x) \to 0$ near the root.

(b) Difficulty with local minima – may send the next iteration x_{i+1} very far from the root.

(c) Lack of convergence for asymmetric functions: $f(a+x) = -f(a-x)$

Newton's method: General comments

- It is an excellent method for polishing roots obtained from other methods!

Newton's method: General comments

- It is an excellent method for polishing roots obtained from other methods!
- The method not only requires the function, but also its derivative. Besides being an additional computational expense, in some cases, it might not be possible to calculate the derivative easily...

Newton's method: General comments

- It is an excellent method for polishing roots obtained from other methods!
- The method not only requires the function, but also its derivative. Besides being an additional computational expense, in some cases, it might not be possible to calculate the derivative easily...
- If the derivative becomes small – the method may end up not converging – as the next step might be far away from the root.

Method of secants

- The secant method is a variant of Newton's method – for the case when the evalulation of the derivative is difficult/impossible.

Method of secants

- The secant method is a variant of Newton's method – for the case when the evalulation of the derivative is difficult/impossible.
- The non linear function $f(x)$ is aproximated locally by the linear function $g(x)$, which is the secant to $f(x)$, and the root of $g(x)$ is taken as an improved approximation to the root of $f(x)$.

Method of secants: Algorithm

The derivative of the function $f(x)$ at point x_i can be approximated as:

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

Method of secants: Algorithm

The derivative of the function $f(x)$ at point x_i can be approximated as:

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

Then using Newton's method:

$$\begin{aligned} x_{i+1} &= x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \\ &= \frac{x_{i-1}f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})} \end{aligned}$$

Method of secants: Algorithm

The derivative of the function $f(x)$ at point x_i can be approximated as:

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

Then using Newton's method:

$$\begin{aligned} x_{i+1} &= x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \\ &= \frac{x_{i-1}f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})} \end{aligned}$$

- Needs two initial points to start.

Method of secants: Algorithm

The derivative of the function $f(x)$ at point x_i can be approximated as:

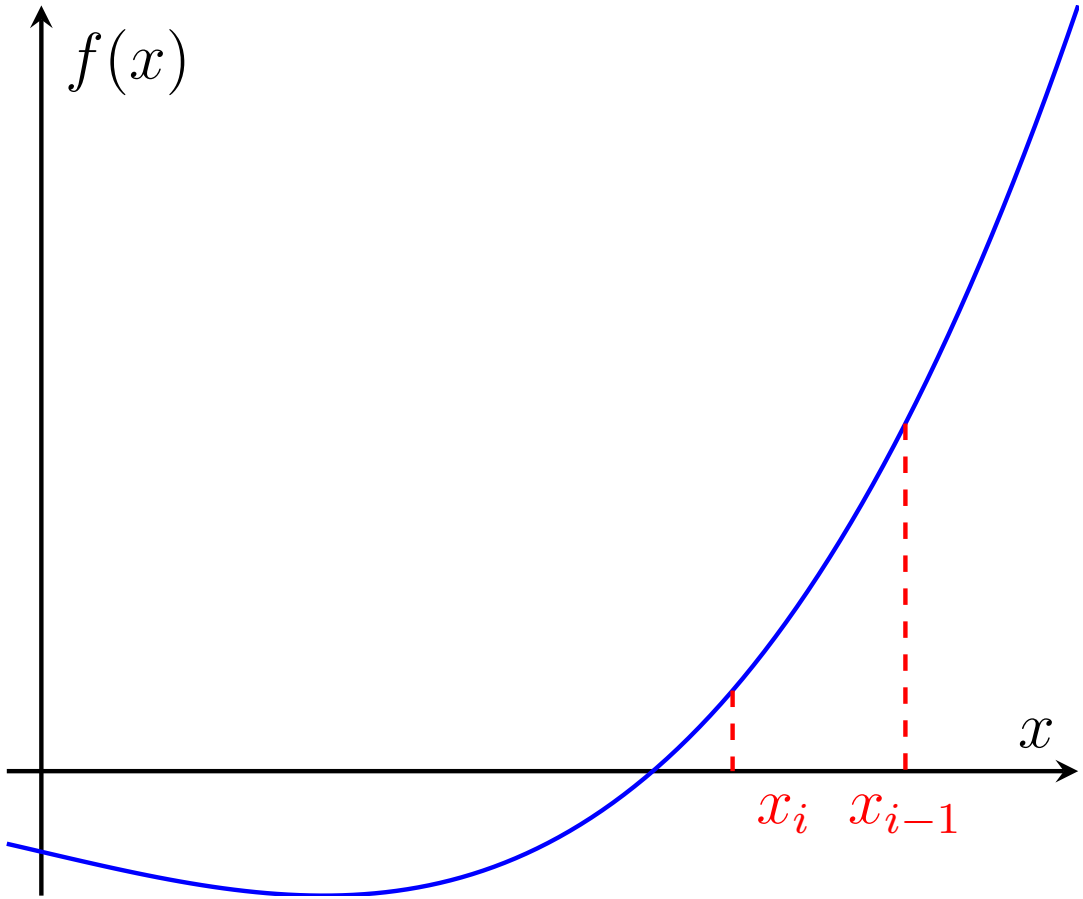
$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

Then using Newton's method:

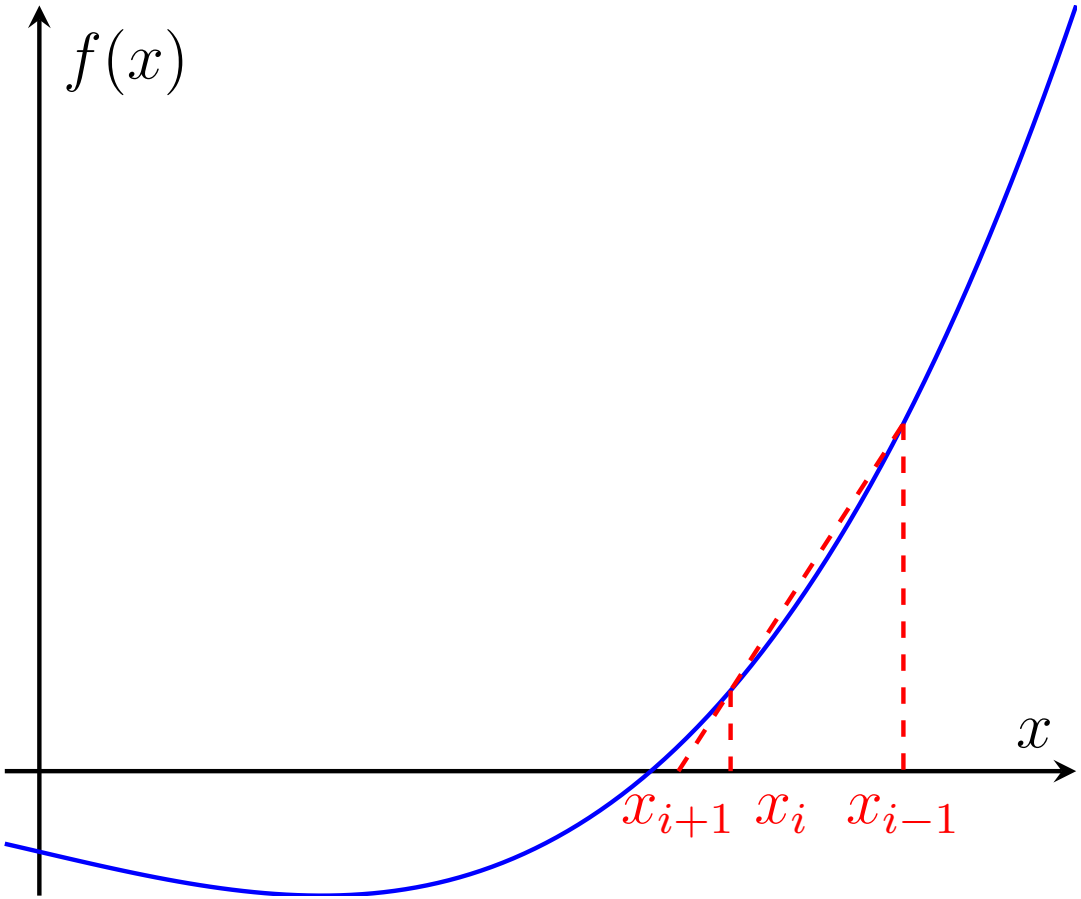
$$\begin{aligned} x_{i+1} &= x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \\ &= \frac{x_{i-1}f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})} \end{aligned}$$

- Needs two initial points to start.
- Same as False position method – except for the points used!

Method of secants: Algorithm



Method of secants: Algorithm



Method of secants: example

i	x_i	f(x_i)	x_{i-1}	f(x_{i-1})	x_{i+1}	f(x_{i+1})
1	1.000000	0.459698	1.500000	1.429263	0.762936	0.040126
2	0.762936	0.040126	1.000000	0.459698	0.740264	0.001974
3	0.740264	0.001974	0.762936	0.040126	0.739091	0.000010
4	0.739091	0.000010	0.740264	0.001974	0.739085	0.000000

Root is: 0.73908513481

- False position used 4 iterations and bisection – 22 iterations!
- Newton's method took 3 iterations.
- However, the question as to which method is more efficient depends not just on the number of iterations – as in the Newton's iteration, one also has to evaluate the derivative!

Muller's method

- The Muller's method is a variant of method of secants.

Muller's method

- The Muller's method is a variant of method of secants.
- The non linear function $f(x)$ is aproximated locally by the *quadratic* function $g(x)$, and the root of $g(x)$ is taken as an improved approximation to the root of $f(x)$.

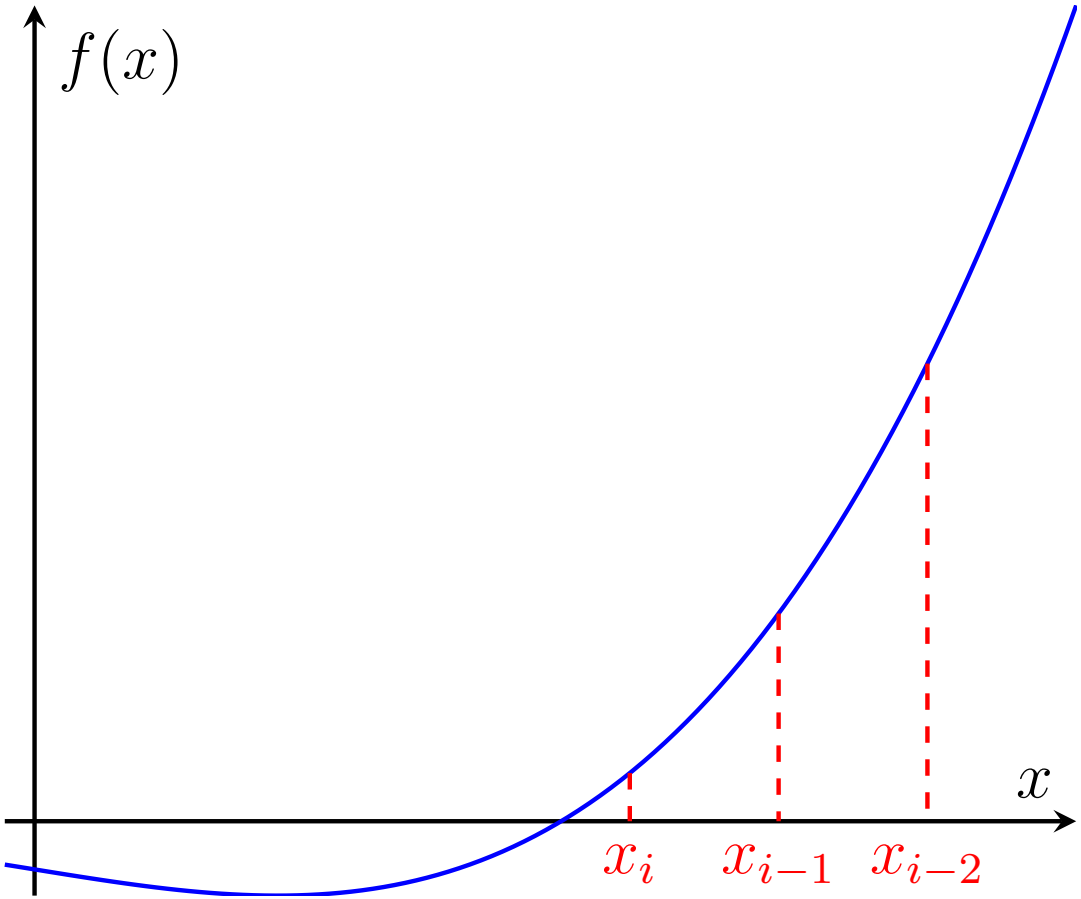
Muller's method

- The Muller's method is a variant of method of secants.
- The non linear function $f(x)$ is aproximated locally by the *quadratic* function $g(x)$, and the root of $g(x)$ is taken as an improved approximation to the root of $f(x)$.
- The only difference between Muller's method and method of secants is that the $g(x)$ is quadratic function in Muller's method and linear function in secant method!

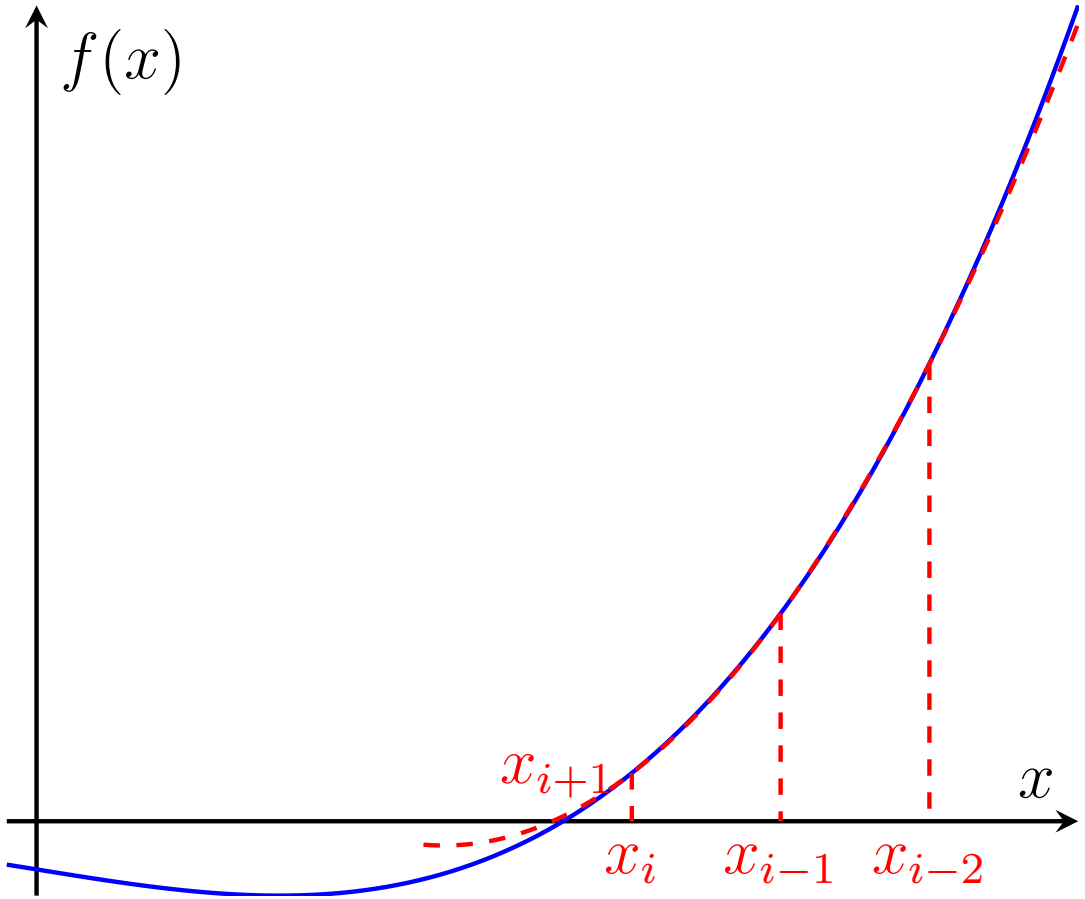
Muller's method

- The Muller's method is a variant of method of secants.
- The non linear function $f(x)$ is aproximated locally by the *quadratic* function $g(x)$, and the root of $g(x)$ is taken as an improved approximation to the root of $f(x)$.
- The only difference between Muller's method and method of secants is that the $g(x)$ is quadratic function in Muller's method and linear function in secant method!
- Three initial approximations are required to start the algorithm (as opposed to two in secant method).

Muller's method: Algorithm



Muller's method: Algorithm



Fixed point iteration

Suppose that you can bring an equation $g(x) = 0$ in the form $x = f(x)$. This fixed point equation, under certain conditions, can be solved using iteration.

Fixed point iteration

Suppose that you can bring an equation $g(x) = 0$ in the form $x = f(x)$. This fixed point equation, under certain conditions, can be solved using iteration.

- Start with an approximation x_0 of the root.

Fixed point iteration

Suppose that you can bring an equation $g(x) = 0$ in the form $x = f(x)$. This fixed point equation, under certain conditions, can be solved using iteration.

- Start with an approximation x_0 of the root.
- Calculate x_0, x_1 $\ldots$ x_n such that

$$x_1 = f(x_0)$$
$$x_2 = f(x_1)$$
$$x_3 = f(x_2)$$

Fixed point iteration

Suppose that you can bring an equation $g(x) = 0$ in the form $x = f(x)$. This fixed point equation, under certain conditions, can be solved using iteration.

- Start with an approximation x_0 of the root.
- Calculate x_0, x_1 ... x_n such that

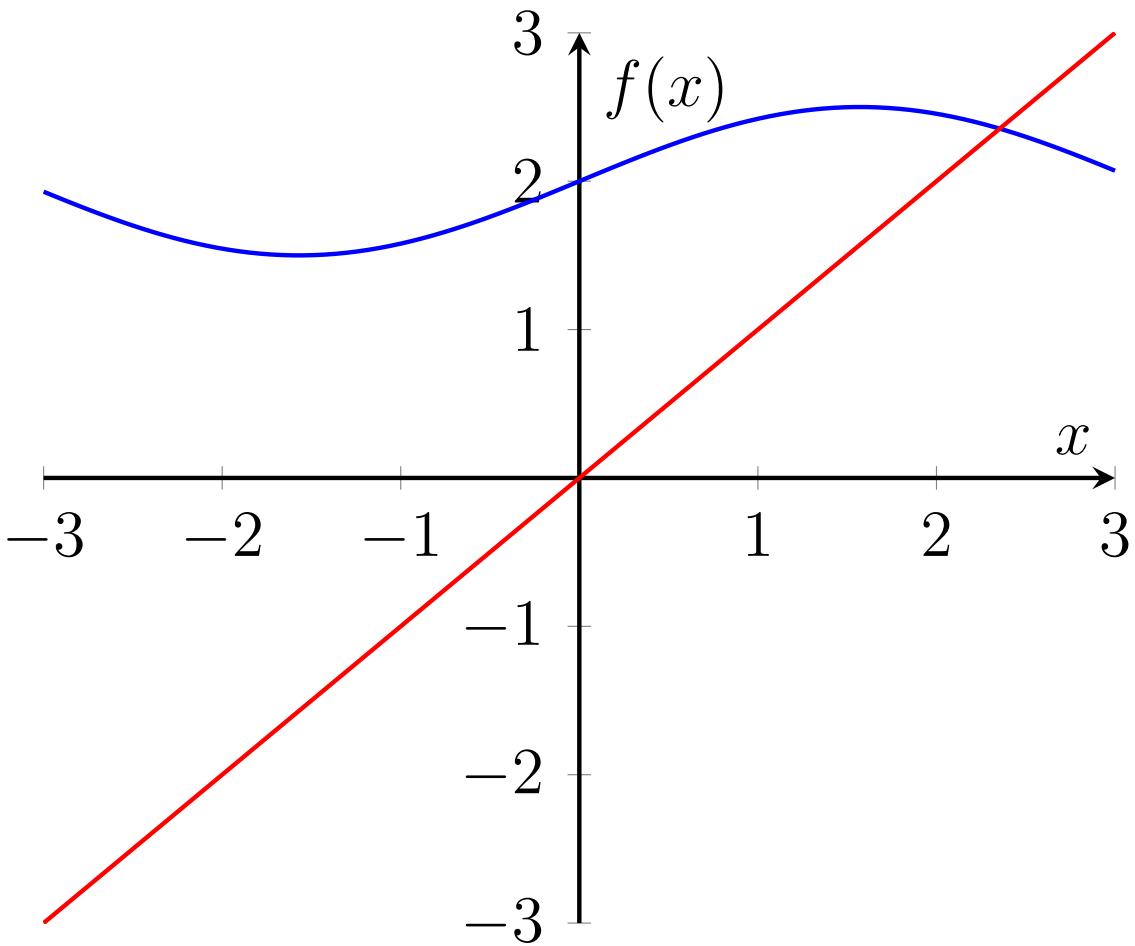
$$x_1 = f(x_0)$$
$$x_2 = f(x_1)$$
$$x_3 = f(x_2)$$

- If the sequence x_0, x_1 ... x_n belongs to an interval I, where $|f'(x)| < k < 1$ then the sequence has a limit L and L is the only root of $x = f(x)$ in the interval I.

Fixed point iteration: example

Consider an example: $f(x) = 2 + \frac{\sin(x)}{2}$

$$|f'(x)| = |\frac{\cos(x)}{2}| < 1.$$



Initial guess $x = 2.0$.

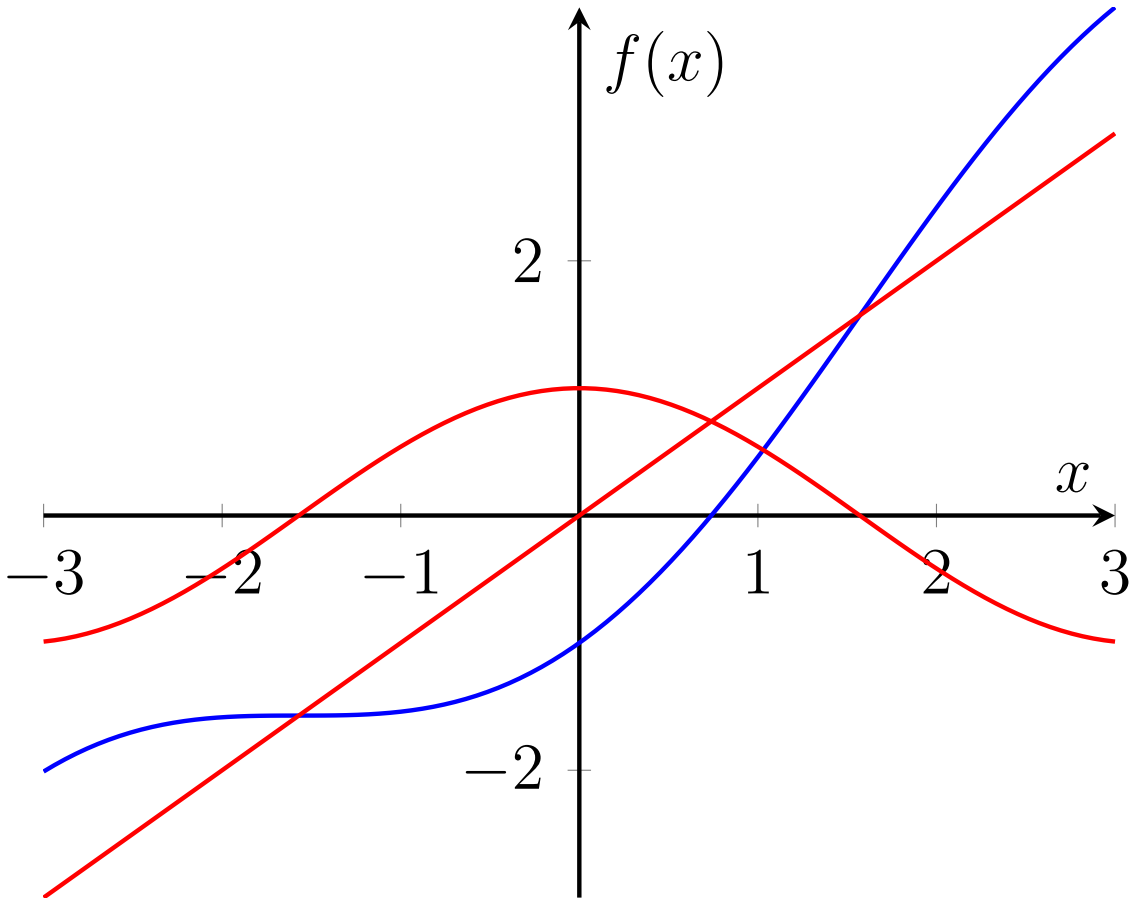
Fixed point iteration: example

```
 i     x_i       f(x_i)
 1   2.000000  2.454649
 2   2.454649  2.317089
 3   2.317089  2.367106
 4   2.367106  2.349675
 5   2.349675  2.355851
 6   2.355851  2.353675
 7   2.353675  2.354443
 8   2.354443  2.354172
 9   2.354172  2.354268
10   2.354268  2.354234
11   2.354234  2.354246
12   2.354246  2.354242
13   2.354242  2.354243

Root is: 2.35424314498
```

Fixed point iteration: Example

Consider an example: $f(x) = x - \cos(x)$



Initial guess $x = 1.0$.

Fixed point iteration: example

i	x_i	f(x_i)
1	1.000000	0.540302
2	0.540302	0.857553
3	0.857553	0.654290
4	0.654290	0.793480
5	0.793480	0.701369
6	0.701369	0.763960
7	0.763960	0.722102
8	0.722102	0.750418
9	0.750418	0.731404
10	0.731404	0.744237
11	0.744237	0.735605
12	0.735605	0.741425
13	0.741425	0.737507
14	0.737507	0.740147
15	0.740147	0.738369
16	0.738369	0.739567
17	0.739567	0.738760
18	0.738760	0.739304
19	0.739304	0.738938
20	0.738938	0.739184
21	0.739184	0.739018
22	0.739018	0.739130
23	0.739130	0.739055
24	0.739055	0.739106
25	0.739106	0.739071
26	0.739071	0.739094
27	0.739094	0.739079
28	0.739079	0.739089
29	0.739089	0.739082
30	0.739082	0.739087
31	0.739087	0.739084
32	0.739084	0.739086
33	0.739086	0.739085

Root is: 0.739084549575

Fixed point iteration

- We want a root of the function $g(x) = 0$ and we know a first approximation x_0 of the root.

Fixed point iteration

- We want a root of the function $g(x) = 0$ and we know a first approximation x_0 of the root.
- We write $x = x + rg(x)$

Fixed point iteration

- We want a root of the function $g(x) = 0$ and we know a first approximation x_0 of the root.
- We write $x = x + rg(x)$
- We choose an r-value such that $1 + rg'(x_0) \approx 0$. Let $r_0 = \text{the chosen value}$.

Fixed point iteration

- We want a root of the function $g(x) = 0$ and we know a first approximation x_0 of the root.
- We write $x = x + rg(x)$
- We choose an r-value such that $1 + rg'(x_0) \approx 0$. Let $r_0 = \text{the chosen value}$.
- We apply iteration on $x = x + r_0 g(x)$ starting with $x = x_0$.

Fixed point iteration: example

We rewrite our function as $x = x - 0.54 * (x - \cos(x))$.

```
i     x_i        f(x_i)
1   1.000000   0.751763
2   0.751763   0.740273
3   0.740273   0.739199
4   0.739199   0.739096
5   0.739096   0.739086
6   0.739086   0.739085

Root is: 0.73908523492
```

- Huge difference in terms of the efficiency!!! From 33 iterations to 6 iterations.
- Careful choice/massaging of the equation can yield remarkable gains.

Open domain methods: General remarks

- All these methods converge rapidly in the vicinity of the root. When the derivative is difficult to compute or is time-consuming, the secant method is very efficient.

Open domain methods: General remarks

- All these methods converge rapidly in the vicinity of the root. When the derivative is difficult to compute or is time-consuming, the secant method is very efficient.
- In sensitive cases, all these methods may misbehave! It may be required to use bracketing methods.

Open domain methods: General remarks

- All these methods converge rapidly in the vicinity of the root. When the derivative is difficult to compute or is time-consuming, the secant method is very efficient.
- In sensitive cases, all these methods may misbehave! It may be required to use bracketing methods.
- Plotting of the functions can help in indetifying such cases.

Open domain methods: General remarks

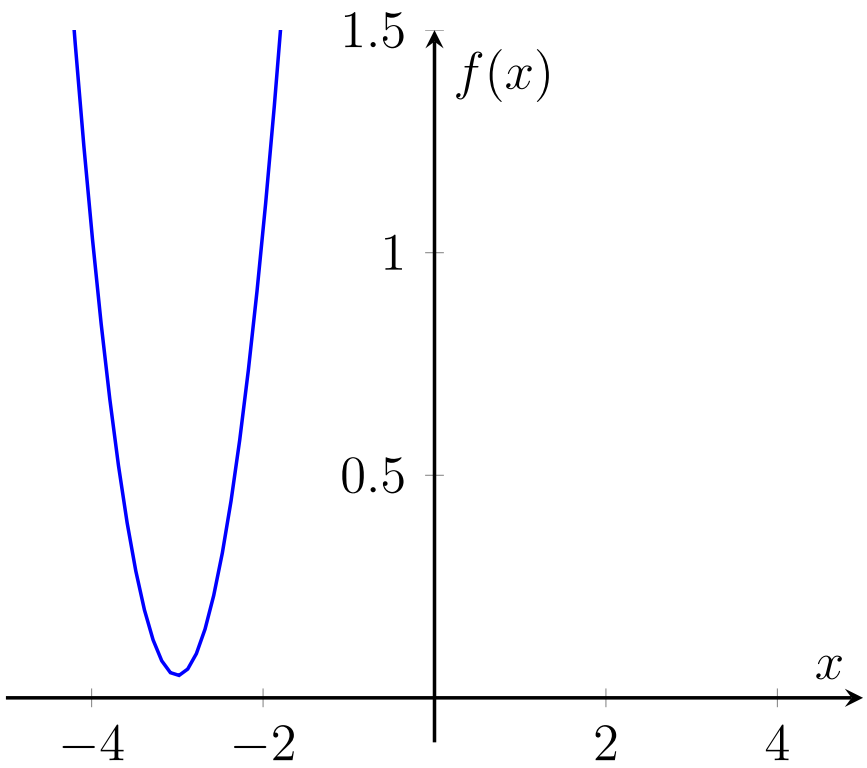
- All these methods converge rapidly in the vicinity of the root. When the derivative is difficult to compute or is time-consuming, the secant method is very efficient.
- In sensitive cases, all these methods may misbehave! It may be required to use bracketing methods.
- Plotting of the functions can help in indetifying such cases.
- *All of these methods can find complex roots simply by using complex arithmetic!*

- Complications.
- Roots of polynomials.
- Non-linear systems of equations with multiple variables.
- Summary.

Complications

- There are no roots at all!

> The hardest thing of all is to find a black cat in a dark room, especially if there is no cat.
>
> Confucius.

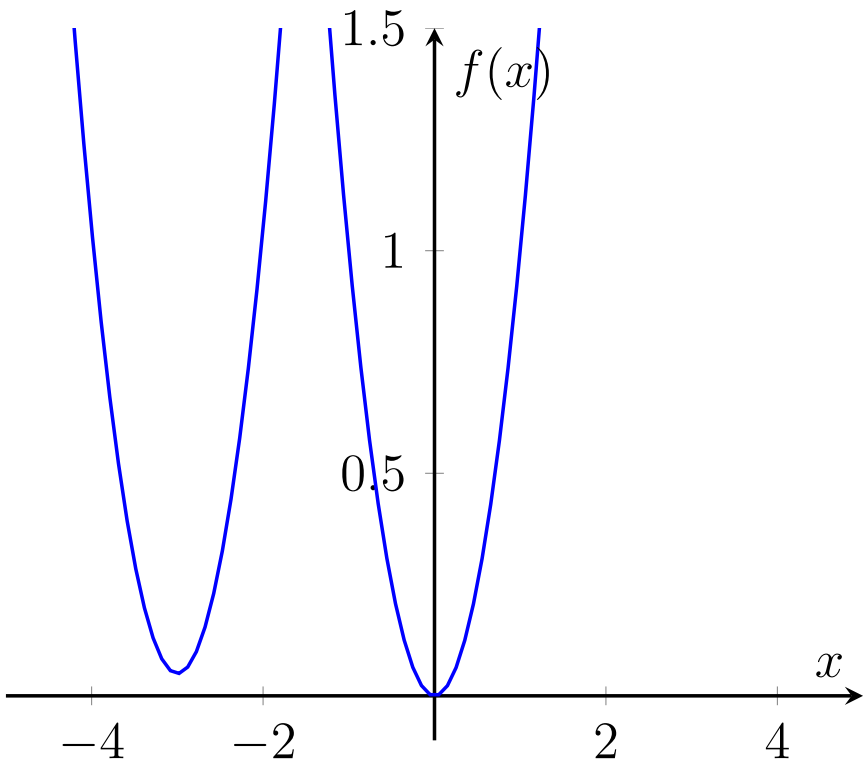


Complications

- There are no roots at all!

> The hardest thing of all is to find a black cat in a dark room, especially if there is no cat.
>
> Confucius.

- There is one root but the function does not change sign.

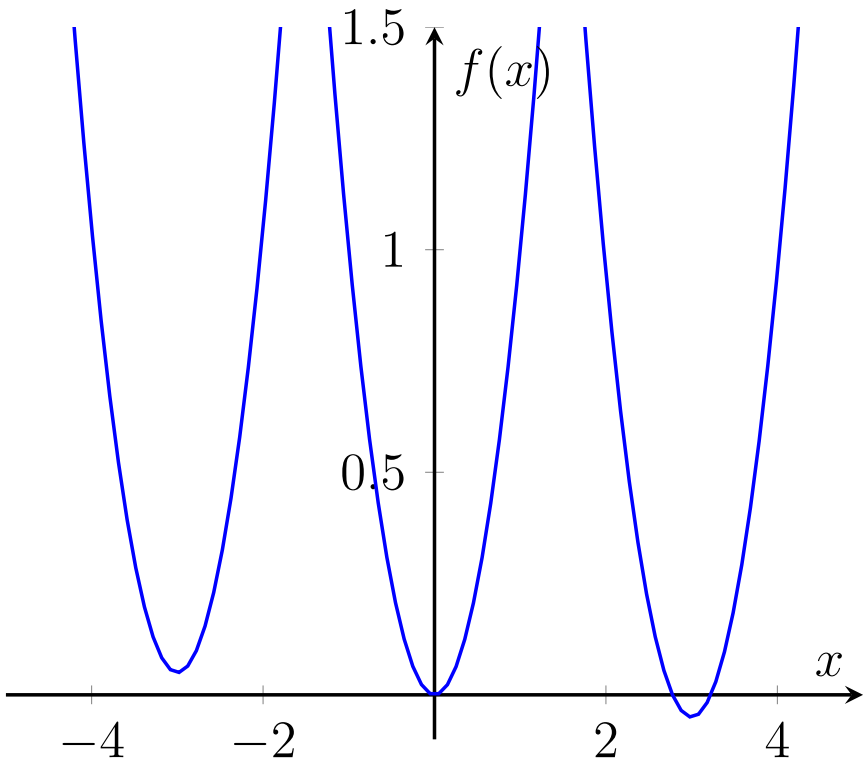


Complications

- There are no roots at all!

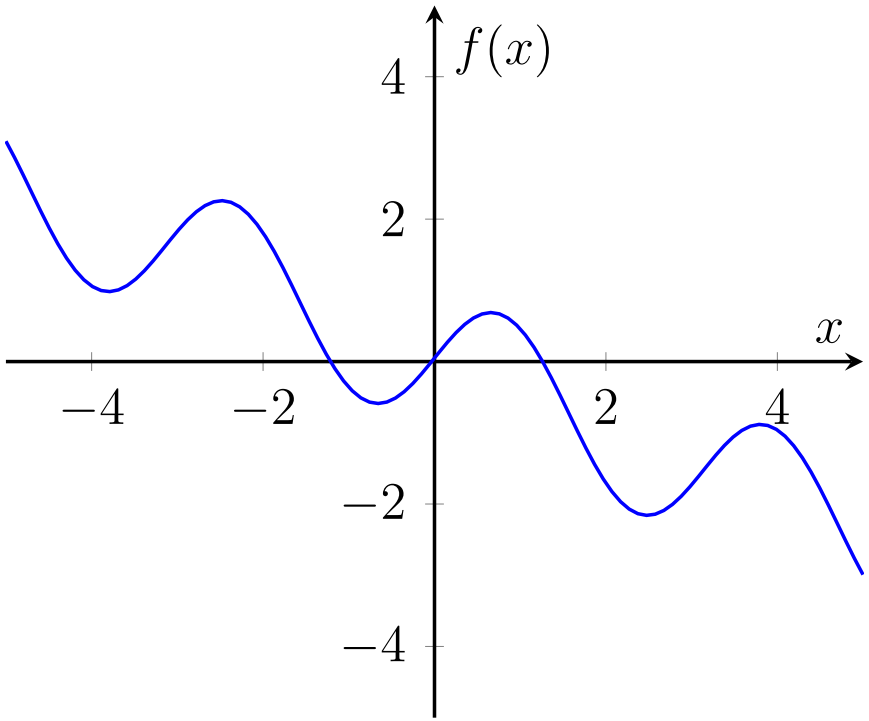
> The hardest thing of all is to find a black cat in a dark room, especially if there is no cat.
>
> Confucius.

- There is one root but the function does not change sign.
- There are two or more roots in the interaval $[a, b]$.



More complications

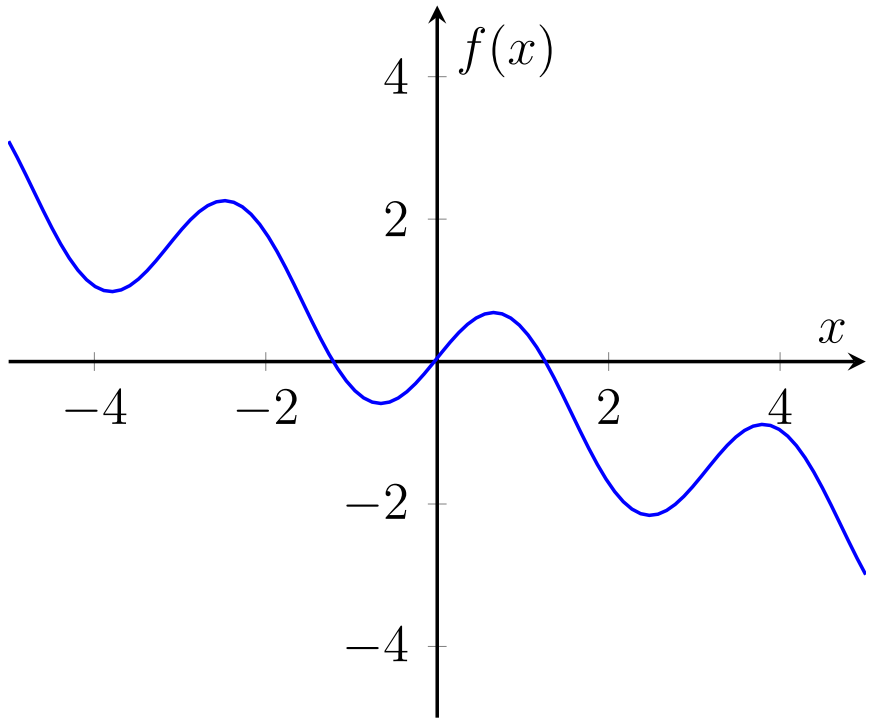
There are many roots.



What will happen if you use bracketing methods for this?

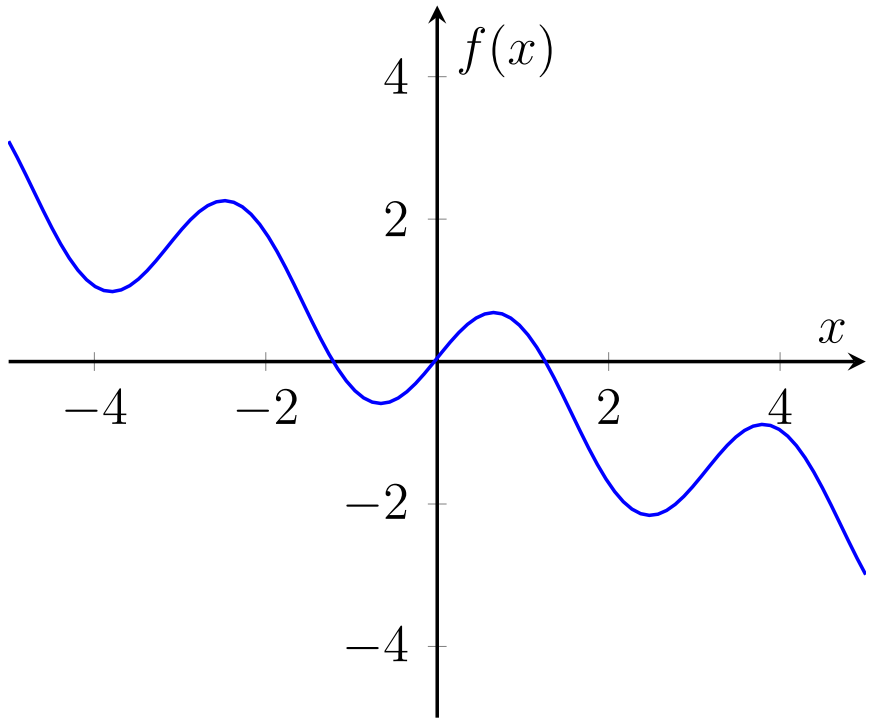
Multiple roots: Brute force method

- The brute force method is a good method for dealing with multiple roots.



Multiple roots: Brute force method

- The brute force method is a good method for dealing with multiple roots.
- One splits the original interval $[a, b]$ into smaller intervals with some step size (say h) and then applies the previously mentioned methods in each of the sub-intervals.



Step size in Brute force method

- If the step size is too large – one may miss multiple roots.

Step size in Brute force method

- If the step size is too large – one may miss multiple roots.
- Choosing too small a step size will result in too many computations.

Step size in Brute force method

- If the step size is too large – one may miss multiple roots.
- Choosing too small a step size will result in too many computations.
- A graphical analysis is very helpful in deciding the step size h.

Step size in Brute force method

- If the step size is too large – one may miss multiple roots.
- Choosing too small a step size will result in too many computations.
- A graphical analysis is very helpful in deciding the step size h.
- *It may be good to evaluate roots with h and then with $h/10$ to confirm that the number of roots remains unchanged.*

Some general comments on roots of polynomials

- The fundamental theorem of algebra states that n^{th} degree polynomial has exactly n zeros or roots.

Some general comments on roots of polynomials

- The fundamental theorem of algebra states that n^{th} degree polynomial has exactly n zeros or roots.
- These roots can be real or complex.

Some general comments on roots of polynomials

- The fundamental theorem of algebra states that n^{th} degree polynomial has exactly n zeros or roots.
- These roots can be real or complex.
- If the coefficients are all real, then the complex roots always occur in conjugate pairs.

Some general comments on roots of polynomials

- The fundamental theorem of algebra states that n^{th} degree polynomial has exactly n zeros or roots.
- These roots can be real or complex.
- If the coefficients are all real, then the complex roots always occur in conjugate pairs.
- The roots may be simple (ie. single) or repeated (ie multiple).

Bracketing methods

- Bracketing methods cannot be used to find repeated roots with even multiplicity because the polynomial function does not change sign at such roots.

Bracketing methods

- Bracketing methods cannot be used to find repeated roots with even multiplicity because the polynomial function does not change sign at such roots.
- They can however be used to find repeated roots with odd multiplicity.

Bracketing methods

- Bracketing methods cannot be used to find repeated roots with even multiplicity because the polynomial function does not change sign at such roots.
- They can however be used to find repeated roots with odd multiplicity.
- Open domain methods are often more efficient than closed domain methods.

Open domain methods

- These methods can be used to find roots – both complex roots as well as real roots.

Open domain methods

- These methods can be used to find roots – both complex roots as well as real roots.
- For complex roots – one just needs to use complex arithemetic – the algorithms remain the same.

Open domain methods

- These methods can be used to find roots – both complex roots as well as real roots.
- For complex roots – one just needs to use complex arithemetic – the algorithms remain the same.
- There are various modifications of the Newton's method which are often used to find roots. For eg. Bairstow's method.

Non linear system of equations: two variables

- A non linear system of equations of two variables:

$$f(x, y) = 0$$
$$g(x, y) = 0$$

Non linear system of equations: two variables

- A non linear system of equations of two variables:

$$f(x, y) = 0$$
$$g(x, y) = 0$$

- There are no good general methods for solving multiple non linear equations.

Non linear system of equations: two variables

- A non linear system of equations of two variables:

$$f(x, y) = 0$$
$$g(x, y) = 0$$

- There are no good general methods for solving multiple non linear equations.
- Extension of bracketing methods is not obvious to systems of non linear equations.

Non linear system of equations: two variables

- A non linear system of equations of two variables:

$$f(x, y) = 0$$
$$g(x, y) = 0$$

- There are no good general methods for solving multiple non linear equations.
- Extension of bracketing methods is not obvious to systems of non linear equations.
- Open domain methods (especially Newton's method) are easily extended to solve such systems.

Non linear system of equations: two variables

- A non linear system of equations of two variables:

$$f(x, y) = 0$$
$$g(x, y) = 0$$

- There are no good general methods for solving multiple non linear equations.
- Extension of bracketing methods is not obvious to systems of non linear equations.
- Open domain methods (especially Newton's method) are easily extended to solve such systems.
- One always needs a good guess!!!

Extension of Newton's method to two variables

We are given two equations:

$$f(x, y) = 0$$
$$g(x, y) = 0$$

We have to find the solution (x^*, y^*) such that:

$$f(x^*, y^*) = 0$$
$$g(x^*, y^*) = 0$$

Taylor expanding about (x^*, y^*):

$$f(x, y) = f(x^*, y^*) + (x - x^*)f'_x + (y - y^*)f'_y + \dots$$
$$g(x, y) = g(x^*, y^*) + (x - x^*)g'_x + (y - y^*)g'_y + \dots$$

Extension of Newton's method to two variables

Keeping only first-order terms and given $f(x^*, y^*) = 0$ and $g(x^*, y^*) = 0$, one has a system of linear equations for x^* and y^* such that:

$$x^* = x + \frac{f'_y g(x,y) - g'_y f(x,y)}{f'_x g'_y - f'_y g'_x}$$

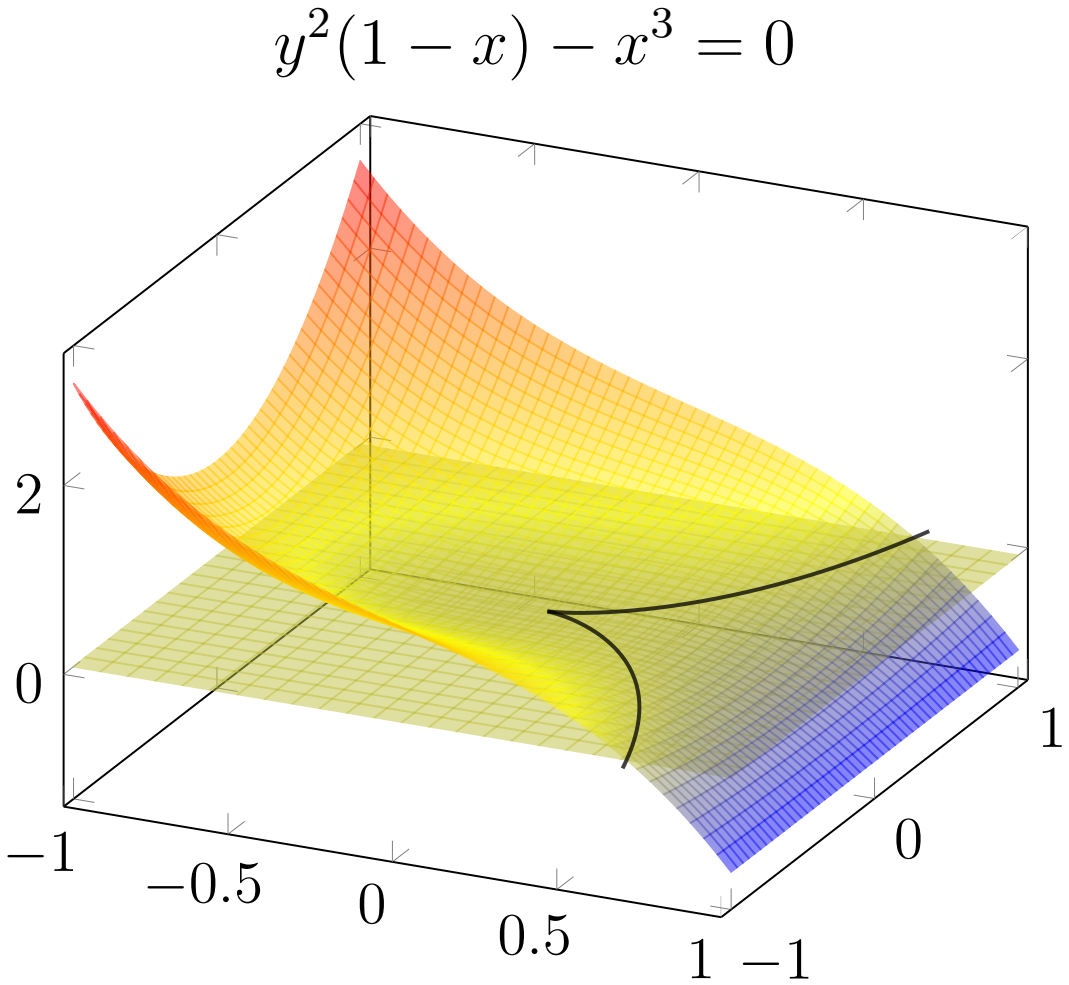
$$y^* = y + \frac{g'_x f(x,y) - f'_x g(x,y)}{f'_x g'_y - f'_y g'_x}$$

This method can be easily generalized to solving n non linear equations.

Non linear systems of equations: Example

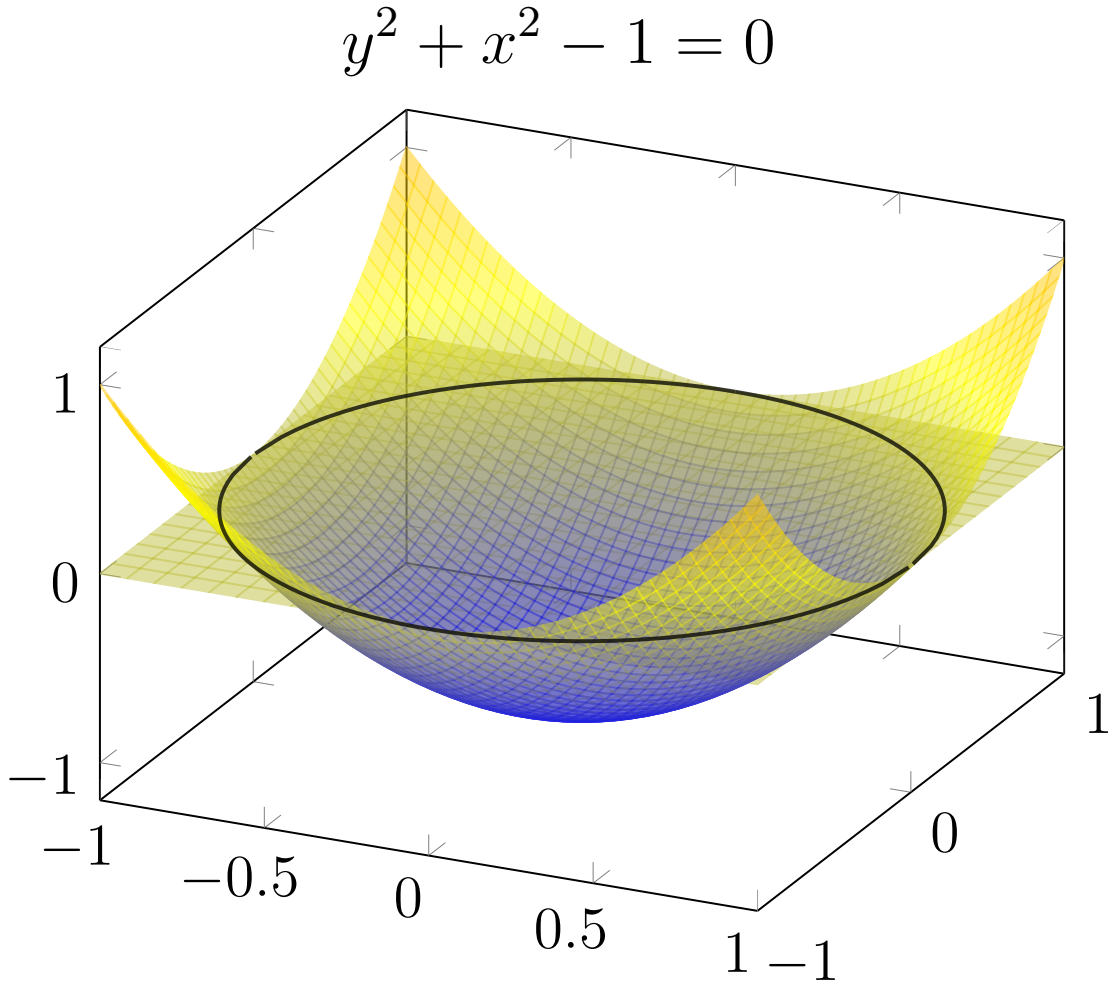
$$f(x, y) = y^2(1 - x) - x^3 = 0$$
$$g(x, y) = y^2 + x^2 - 1 = 0$$

Non linear systems of equations: Example



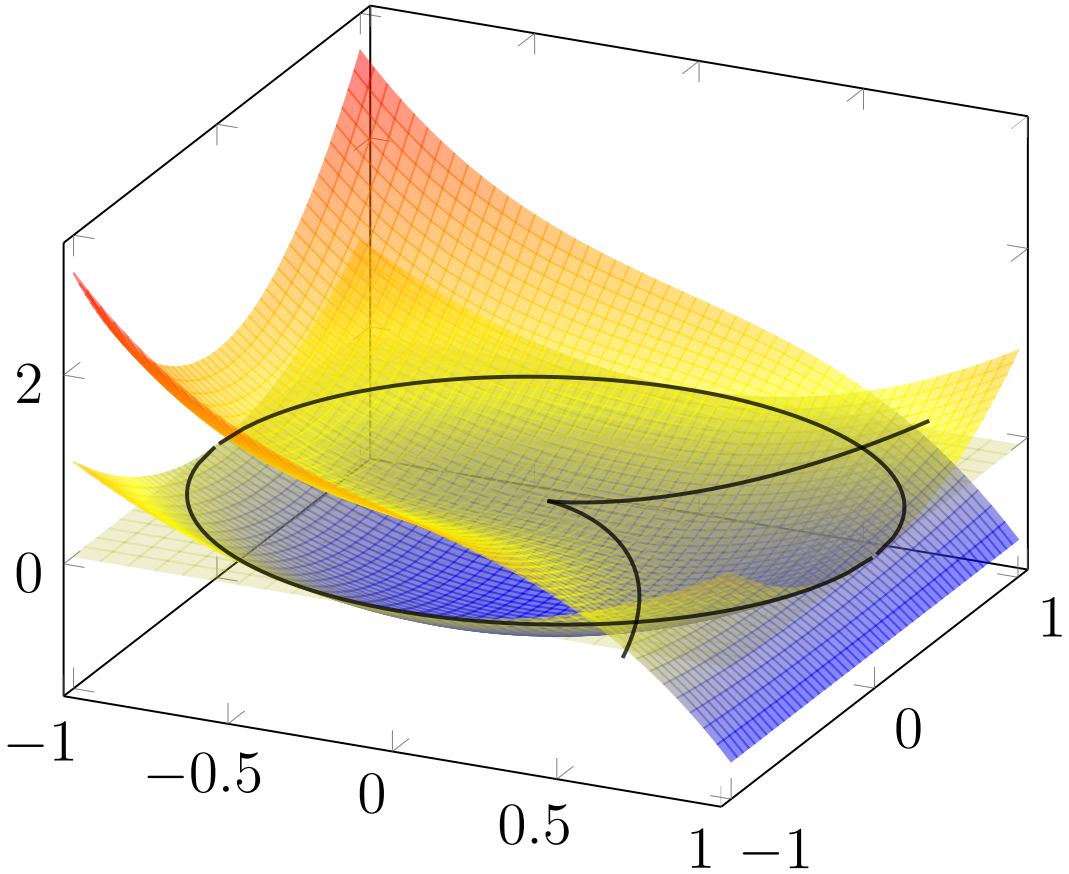
$h(x, y) = 0$ is also shown. The black line shows the intersection of the surface with $h(x, y) = 0$.

Non linear systems of equations: Example



$h(x, y) = 0$ is also shown. The black line shows the intersection of the surface with $h(x, y) = 0$.

Non linear systems of equations: Example



$h(x, y) = 0$ is also shown.

Newton's method: example

```
  i     x          y          f(x,y)     g(x,y)      dx          dy
  1  1.000000   1.000000 -1.000000   1.000000 -0.250000 -0.250000
  2  0.750000   0.750000 -0.281250   0.125000 -0.119048  0.035714
  3  0.630952   0.785714 -0.023352   0.015448 -0.012757  0.000414
  4  0.618195   0.786128 -0.000298   0.000163 -0.000161  0.000023
  5  0.618034   0.786151 -0.000000   0.000000
```

```
Root is: 0.618034012048126, 0.786151376237367
```

```
  i     x          y          f(x,y)     g(x,y)      dx          dy
  1  1.000000  -1.000000 -1.000000   1.000000 -0.250000  0.250000
  2  0.750000  -0.750000 -0.281250   0.125000 -0.119048 -0.035714
  3  0.630952  -0.785714 -0.023352   0.015448 -0.012757 -0.000414
  4  0.618195  -0.786128 -0.000298   0.000163 -0.000161 -0.000023
  5  0.618034  -0.786151 -0.000000   0.000000
```

```
Root is: 0.618034012048126, -0.786151376237367
```

Pitfalls of root finding

- Lack of good initial guess.

Pitfalls of root finding

- Lack of good initial guess.
- Convergence to the wrong root.

Pitfalls of root finding

- Lack of good initial guess.
- Convergence to the wrong root.
- Closely spaced roots/ multiple roots/ Inflection points.

Pitfalls of root finding

- Lack of good initial guess.
- Convergence to the wrong root.
- Closely spaced roots/ multiple roots/ Inflection points.
- Complex roots.

Pitfalls of root finding

- Lack of good initial guess.
- Convergence to the wrong root.
- Closely spaced roots/ multiple roots/ Inflection points.
- Complex roots.
- Ill conditioning of the non linear equations

Pitfalls of root finding

- Lack of good initial guess.
- Convergence to the wrong root.
- Closely spaced roots/ multiple roots/ Inflection points.
- Complex roots.
- Ill conditioning of the non linear equations
- Slow convergence

Some thoughts

- All the methods work for simple/smoothly varying problems if:

Some thoughts

- All the methods work for simple/smoothly varying problems if:
- One starts from a good initial guess.

Some thoughts

- All the methods work for simple/smoothly varying problems if:
- One starts from a good initial guess.
- Efficiency is important if the problem has to be solved multiple times.

Some thoughts

- All the methods work for simple/smoothly varying problems if:
- One starts from a good initial guess.
- Efficiency is important if the problem has to be solved multiple times.
- Anticipate complex roots.

Some thoughts

- All the methods work for simple/smoothly varying problems if:
- One starts from a good initial guess.
- Efficiency is important if the problem has to be solved multiple times.
- Anticipate complex roots.
- Trade off between your time and computer's time...

Some thoughts

- All the methods work for simple/smoothly varying problems if:
- One starts from a good initial guess.
- Efficiency is important if the problem has to be solved multiple times.
- Anticipate complex roots.
- Trade off between your time and computer's time...
- Choosing the right method is difficult – something that works in one equation can miserably fail in another.

Some thoughts

- All the methods work for simple/smoothly varying problems if:
- One starts from a good initial guess.
- Efficiency is important if the problem has to be solved multiple times.
- Anticipate complex roots.
- Trade off between your time and computer's time...
- Choosing the right method is difficult – something that works in one equation can miserably fail in another.
- If possible, visualize the data – this helps a lot.