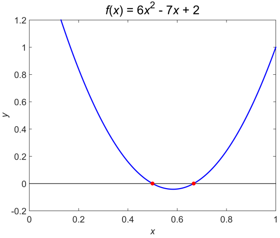
Chapter 3: Methods for calculating the roots

In Chapter 3, we mainly focus on the following contents:

- Bisection method – read section 3.1 page 114–120.
- Newton's method – read section 3.2 page 125–131.
- Secant method – read section 3.3 page 142–144.

This Chapter lists three methods to calculate a root of an equation, especially a nonlinear equation. If we can find a root of an equation, then we also can calculate the minimum for a function, since a function f reaches to its minimum as $f'(x) = 0$.

Root (page 114–116)



A number r, which satisfies $f(r) = 0$, is called as a **root** of that equation, or a **zero** (or **root**) of that function.

Example: The equation $6x^2 - 7x + 2 = 0$ has the roots $1/2$ and $2/3$.

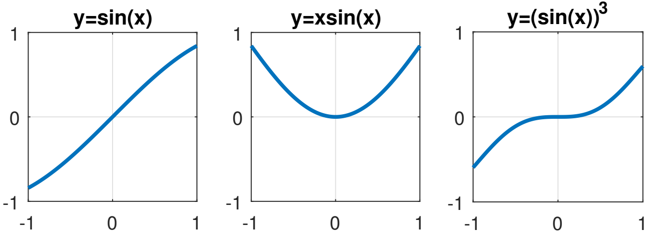
In the most cases we cannot find a root r analytically – then we need numerical methods to get a *good approximation* of the root!

Suppose that $f(x)$ is *continuous*. Let

$$a < b \qquad \text{and} \qquad f(a) \cdot f(b) < 0,$$

i.e. $f(a)$ and $f(b)$ have opposite signs. It then follows that f has a root in the interval $[a, b]$.

Multiple root (page 131)



Simple root: $f(r) = 0$ and $f'(r) \neq 0$:

$$f_1(x) = \sin(x) \quad \Rightarrow \quad f_1'(x) = \cos(x)$$

Double root: $f(r) = f'(r) = 0$ and $f''(r) \neq 0$:

$$f_2(x) = x\,\sin(x) \;\Rightarrow\; f_2'(x) = \sin(x) + x\,\cos(x),\; f_2''(x) = 2\cos(x) - x\,\sin(x)$$

Triple root: $f(r) = f'(r) = f''(r) = 0$ and $f^{(3)}(r) \neq 0$.

$$f_3(x) = \sin^3(x) \quad \Rightarrow \quad f_3^{(3)}(x) = 6\cos^3(x) - 21\cos(x)^2\sin(x)$$

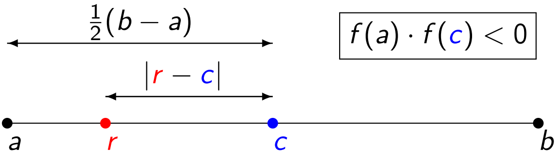
Here, we only focus on **simple root**, i.e. $f'(r) \neq 0$.

Idea of bisection algorithm (page 116–119)

Given two values a and b, where $a < b$ and $f(a)$ and $f(b)$ have opposite signs.

We calculate the *midpoint* $c = (a + b)/2$ of the interval $[a, b]$. If $f(a) \cdot f(c) < 0$, then we know that there is a root of f in the interval $[a, c]$, and the absolute error is bounded as

$$|r - c| \leq \frac{1}{2}(b - a) \ .$$



Along the same way, as the interval becomes smaller and smaller, we approach to a root of f.

Bisection algorithm (page 116–117)

Bisection algorithm reduces the width of the interval by half in each iteration.

Start-interval: $[a, b]$

Compute: $fa = f(a)$, $fb = f(b)$

Repeat until $b - a$ is small enough:

Compute the midpoint $c = \frac{1}{2}(a + b)$ and $fc = f(c)$

if $fa \cdot fc < 0$ (A root exists between a and c)

$b = c$ and $fb = fc$

otherwise

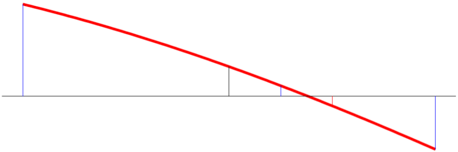
$a = c$ and $fa = fc$

end

Compute root: $c = \frac{1}{2}(a + b)$

Absolute error after n iterations:

$$|r - c| \leq \frac{b - a}{2^{n+1}}$$



Example

The function $f(x) = x^2 - 2$ has a zero at $r = \sqrt{2} \approx 1.4142$. We use the bisection algorithm with the starting interval $[a_0, b_0] = [0, 6]$ to estimate it:

n	a_n	b_n	$\lvert b_n - a_n \rvert$	c_n	$\lvert r - c_n \rvert$
0	0.0	6.0	6.0	3.0	1.586
1	0.0	3.0	3.0	1.5	0.086
2	0.0	1.5	1.5	0.75	0.664
3	0.75	1.5	0.75	1.125	0.289
4	1.125	1.5	0.375	1.3125	0.102
5	1.3125	1.5	0.18755	1.4063	0.0080
6	1.40625	1.5	0.09375	1.4531	0.0389
7	1.40625	1.4531	0.046875	1.4297	0.0155
8	1.40625	1.4297	0.023438	1.4180	0.0038

Example

The function $f(x) = x^2 - 2$ has a zero at $r = \sqrt{2} \approx 1.4142$. We use the bisection algorithm with the starting interval $[a_0, b_0] = [0, 6]$ to estimate it:

n	a_n	b_n	$\lvert b_n - a_n \rvert$	c_n	$\lvert r - c_n \rvert$
0	0.0	6.0	6.0	3.0	1.586
1	0.0	3.0	3.0	1.5	0.086
2	0.0	1.5	1.5	0.75	0.664
3	0.75	1.5	0.75	1.125	0.289
4	1.125	1.5	0.375	1.3125	0.102
5	1.3125	1.5	0.18755	1.4063	0.0080
6	1.40625	1.5	0.09375	1.4531	0.0389
7	1.40625	1.4531	0.046875	1.4297	0.0155
8	1.40625	1.4297	0.023438	1.4180	0.0038

Notice that the absolute error $\lvert r - c_n \rvert$ does not necessarily decrease monotonically.

There are another two examples on the bisection algorithm in page 117–118.

Convergence analysis (page 119–120)

If we use the midpoint $c_0 = (a_0 + b_0)/2$ as our estimate of r, we have

$$|r - c_0| \leq \frac{b_0 - a_0}{2} \; .$$

Then, in the next iteration we have

$$|r - c_1| \leq \frac{b_1 - a_1}{2} = \frac{\frac{1}{2}(b_0 - a_0)}{2} = \frac{b_0 - a_0}{2^2} \; .$$

After n iterations, the maximum error will be reduced by a factor 2^n:

$$|r - c_n| \leq \frac{b_n - a_n}{2} = \frac{\frac{1}{2}(b_{n-1} - a_{n-1})}{2} \leq \cdots \leq \frac{b_0 - a_0}{2^{n+1}} = \frac{1}{2^n} \, \frac{b_0 - a_0}{2} \; .$$

Do we have **linear convergence**, i.e, there is a constant $C \in [0, 1)$ such that

$$\forall n \geq 1 : \quad |c_{n+1} - r| \leq C \, |c_n - r| \; ?$$

Convergence analysis (page 119–120)

If we use the midpoint $c_0 = (a_0 + b_0)/2$ as our estimate of r, we have

$$|r - c_0| \leq \frac{b_0 - a_0}{2} \ .$$

Then, in the next iteration we have

$$|r - c_1| \leq \frac{b_1 - a_1}{2} = \frac{\frac{1}{2}(b_0 - a_0)}{2} = \frac{b_0 - a_0}{2^2} \ .$$

After n iterations, the maximum error will be reduced by a factor 2^n:

$$|r - c_n| \leq \frac{b_n - a_n}{2} = \frac{\frac{1}{2}(b_{n-1} - a_{n-1})}{2} \leq \cdots \leq \frac{b_0 - a_0}{2^{n+1}} = \frac{1}{2^n}\,\frac{b_0 - a_0}{2} \ .$$

Do we have **linear convergence**, i.e, there is a constant $C \in [0, 1)$ such that

$$\forall n \geq 1 : \quad |c_{n+1} - r| \leq C\,|c_n - r| \ ?$$

NO! – The example in the former slide shows that the error can increase in the next iteration (i.e., $C > 1$).

How many iterations? (page 119–120)

How many iterations, n, of the bisection algorithm are needed to guarantee that the error is bounded by a given tolerance ϵ? Then it is necessary to solve the following inequality for n:

$$|r - c_n| \leq \frac{b-a}{2^{n+1}} < \epsilon \ .$$

We obtain:

$$\frac{b-a}{\epsilon} < 2^{n+1} \qquad \Leftrightarrow \qquad \log \frac{b-a}{2\epsilon} < n \log 2 \qquad \Rightarrow$$

$$n = \left\lceil \frac{\log(b-a) - \log(2\epsilon)}{\log 2} \right\rceil \qquad \text{(The result is rounded up)}$$

Example: $b - a = 3$ and $\epsilon = 10^{-4}$ give

$$n = \left\lceil \log \frac{3}{2 \times 10^{-4}} \Big/ \log 2 \right\rceil = \lceil 9.615/0.693 \rceil = \lceil 13.87 \rceil = 14 \ .$$

Summary of the bisection algorithm

Some disadvantages of the bisection algorithm:

- It is robust, but converges slowly.
- It cannot find some multiple roots, for example:

$$f(x) = (x-1)^2 = 0 \; .$$

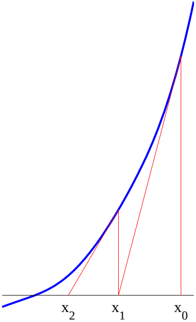
- If f is not continuous, we cannot use bisection algorithm.
- It cannot be generalized to the case with more than one variable.

Hence, we need more advanced methods:

- For example: Newton's method and secant method
- It generates a sequence $x_0 \to x_1 \to x_2 \cdots$ to the root r.
  (Not a sequence of intervals where there is a root.)

Newton's method (page 126)

We start with an approximation x_0. The next approximation x_1 is the intersection of the line $\ell(x) = f'(x_0)\,(x - x_0) + f(x_0)$ with the x-axis.



That is, set $\ell(x) = 0$, and obtain $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$.

Naturally, this process can be repeated to produce a sequence:
$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$, $x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$, ...

Newton's method (page 126–127)

Based on Taylor's Theorem (page 27) at $x = x_0$ we have:

$$f(x_0 + h) = f(x_0) + h\, f'(x_0) + \tfrac{1}{2}\, h^2\, f''(x_0) + \cdots$$

We want to find h such that $f(x_0 + h) = 0$, but it's not easy.

Newton's method (page 126–127)

Based on Taylor's Theorem (page 27) at $x = x_0$ we have:

$$f(x_0 + h) = f(x_0) + h\,f'(x_0) + \tfrac{1}{2}\,h^2\,f''(x_0) + \cdots$$

We want to find h such that $f(x_0 + h) = 0$, but it's not easy. $\rightarrow$ Ignore all higher-order terms in the Taylor series and find h satisfying

$$f(x_0) + h\,f'(x_0) = 0$$

which gives

$$h = -\frac{f(x_0)}{f'(x_0)} \qquad = \text{same formula as in the former slide.}$$

Algorithm: $x_0 =$ starting point; for $n = 0, 1, 2, \ldots$

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

$\varsigma_n = |x_{n+1} - x_n|$ is an *estimate* for the error $|r - x_n|$, so stop as $\varsigma_n < \epsilon$.

Example: convergence of Newton's method

Apply Newton's method on the example in slide 6: $f(x) = x^2 - 2$ with $x_0 = 3$.

n	x_n	$\lvert r - x_n \rvert$
0	3.000000000000000	1.6
1	1.833333333333333	0.42
2	1.462121212121212	0.048
3	1.414998429894803	$7.8 \cdot 10^{-4}$
4	1.414213780047198	$2.2 \cdot 10^{-7}$
5	1.414213562373112	$1.7 \cdot 10^{-14}$
$r =$	1.414213562373095	

Notice the doubling of correct digits in every iteration, which corresponds to the error squared in every iteration.

Convergence analysis (page 129–130)

The formula of x_{n+1} from Newton's method gives:

$$e_{n+1} \equiv r - x_{n+1} = \underbrace{r - x_n}_{e_n} + \frac{f(x_n)}{f'(x_n)} = \frac{e_n f'(x_n) + f(x_n)}{f'(x_n)} \ .$$

By Taylor's Theorem (page 27), there exists a point ξ_n situated between r and x_n for which:

$$0 = f(r) = f(x_n + e_n) = f(x_n) + e_n\, f'(x_n) + \tfrac{1}{2}\, e_n^2\, f''(\xi_n) \qquad \Leftrightarrow$$

$$e_n f'(x_n) + f(x_n) = -\tfrac{1}{2}\, e_n^2\, f''(\xi_n) \ .$$

Substitute in e_{n+1}:

$$e_{n+1} = \frac{-\frac{1}{2}\, e_n^2\, f''(\xi_n)}{f'(x_n)} = -\frac{1}{2} \left(\frac{f''(\xi_n)}{f'(x_n)} \right) e_n^2 \ .$$

This is, at least quantitively, the sort of equation we want.

Convergence analysis – quadratic convergence

Suppose that at some iterate x_n we have $|e_n| = |r - x_n| \le \delta \Rightarrow |r - \xi_n| \le \delta$.

Define a function

$$c(\delta) \equiv \frac{1}{2} \frac{\max_{|r-x|\le\delta} |f''(x)|}{\min_{|r-x|\le\delta} |f'(x)|}$$

Then, we have (from the former slide)

$$\underline{\underline{|e_{n+1}|}} = \frac{1}{2} \left| \frac{f''(\xi_n)}{f'(x_n)} \right| e_n^2 \,\underline{\underline{\le\ c(\delta)\, e_n^2}} \le c(\delta)\, |e_n|\, \delta = \delta\, c(\delta)\, |e_n| \,. \qquad (*)$$

In the same way, if $|e_{n-1}| = |r - x_{n-1}| \le \delta$, then we have

$$|e_n| \le \delta\, c(\delta)\, |e_{n-1}|$$

Continue backward until the first iteration, and obtain that if $|r - x_0| \le \delta$ we have

$$|e_{n+1}| \le \big(\delta\, c(\delta)\big)^{n+1} |e_0| \,.$$

Hence, Newton's method always converges, if x_0 is close enough to r, i.e., $\delta\, c(\delta) < 1$. Its convergent rate is quadratic according to $(*)$.

Example: quadratic convergence

Example: $f(x) = x^2 - 2$ with $x_0 = 3$.

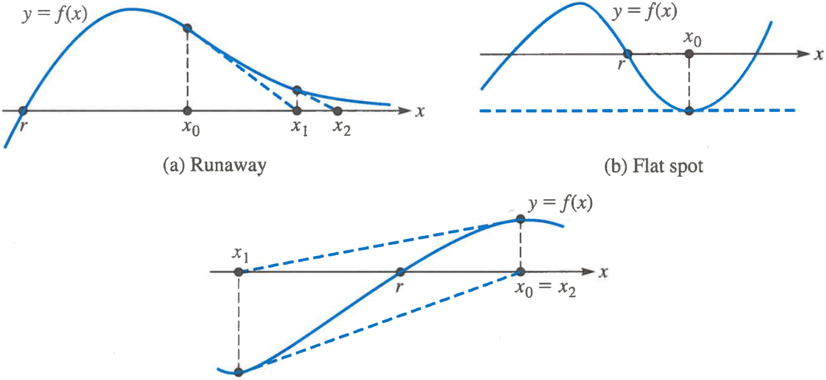
Define $|e_n| = |r - x_n|$.

n	x_n	$\lvert e_n \rvert$	$\lvert e_n/e_{n-1}^2 \rvert$
0	3.0000000000000000	1.6	–
1	1.8333333333333333	0.42	0.1667
2	1.4621212121212122	0.048	0.2727
3	1.4149984298948028	$7.8 \cdot 10^{-4}$	0.3420
4	1.4142137800471977	$2.2 \cdot 10^{-7}$	0.3534
5	1.4142135623731118	$1.7 \cdot 10^{-14}$	0.3515
$r =$ 1.4142135623730950			

Notice that $|e_n/e_{n-1}^2|$ tends to a constant ≈ 0.35 as n increases.

Discussion of Newton's method (page 131)

The convergence of Newton's method depends on the hypothesis that the starting point x_0 is *sufficiently close* to a root. If x_0 is a bit far away from r, the method can fail.



(a) Runaway

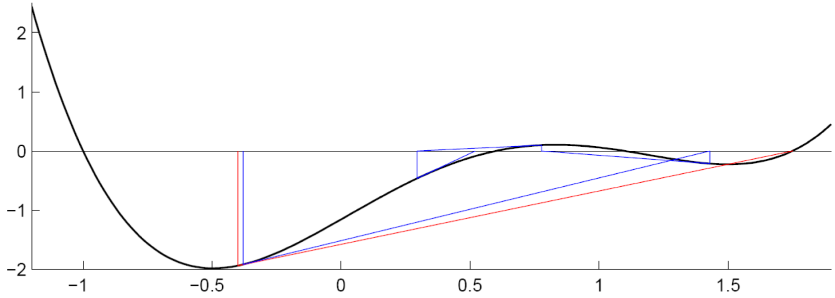
(b) Flat spot

(c) Cycle

Newton's method is not robust

When the starting point is sufficiently close to a root, Newton's method converges fast.

But if x_0 is far from r, for example it can happen:



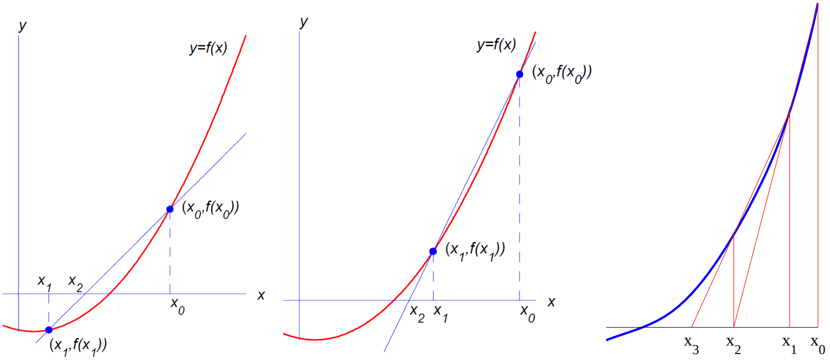
There are two very close starting points, but Newton's method finds two roots, which are far away from each other.

Newton's method is not robust!

Secant method (page 142–143)

In some cases it is difficult to obtain $f'(x_n)$, for example, it's too expensive to compute $f'(x_n)$ or we are lack of knowledge of the function and its derivatives, then Newton's method cannot be used.

Alternatively: We approximate $f'(x_n)$ by the slope of the *secant line* of two points x_{n-1} and x_n.



Secant method (page 142–143)

Graphical. The equation of the secant line from $(x_{n-1}, f(x_{n-1}))$ and $(x_n, f(x_n))$ is:

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

If we set $y = 0$, then we obtain

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

As an approximation of Newton's method Use an approximation of $f'(x_n)$:

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

Newton's iteration can be approximated by:

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

Same example – with secant method

Secant method: $f(x) = x^2 - 2$ with $x_0 = 3$, $x_1 = 2$.

n	x_n	$\lvert r - x_n \rvert$
0	3.000000000000000	1.6
1	2.000000000000000	0.59
2	1.599999999999999	0.19
3	1.444444444444444	0.030
4	1.416058394160584	0.0018
5	1.414233059257159	$1.9 \cdot 10^{-5}$
6	1.414213575081493	$1.3 \cdot 10^{-8}$
7	1.414213562373182	$8.7 \cdot 10^{-14}$
r =	1.414213562373095	

It does not converge as fast as Newton's method.

Convergence of secant method (formula (6) page 145)

The order of convergence of the secant method can be expressed in terms of the inequality

$$|e_{n+1}| \le C\,|e_n|^{\phi}, \qquad C < 1 \quad \text{(a constant)}$$

where $\phi = \frac{1}{2}(1+\sqrt{5}) \approx 1.62$ is the *golden ratio*. Since $\phi > 1$, we say that the method has *superlinear convergence*, which is faster than linear.

(In the book, the proof is not solid, so we skip it.)

Python: functions for computing roots

The Python package Scipy has the functions `fsolve` and `brentq` that calculate roots:

- `x = fsolve(fun,x0)` finds a root of the function `fun` near `x0`.
- `x = brentq(fun,x1,x2)` finds a root in the interval [`x1`,`x2`].

The signs of `fun(x1)` and `fun(x2)` must be different.

Example: find a root of $f(x) = x\sin(x)$:

```
# import root-finding functions
>>> from scipy.optimize import fsolve, brentq
# define the function as lambda-function
>>> f = lambda x: x*np.sin(x)
# compute root
>>> root = brentq(f,2,4)
root =
3.1416
```

Comparison of methods (page 147)

- **Bisection** is robust but slow. If an error tolerance has been given in advance, the number of iterations required by bisection method can be estimated.
- **Bisection** requires two starting points with different signs of $f(x)$, and f only need be continuous.
- **Newton's method** requires f'.
- **Newton's method** and **secant method** converge faster than bisection method, but require a starting point that is "sufficiently close" to a root.
- **Secant method** is nearly as fast as Newton's method and does not require knowledge of the derivative f'.
- **All methods** requires the user to know roughly about the desired root and how many roots to be examined.

First homework assignment

- It includes 4 multiple choice questions and 2 normal exercise questions.
- The first exercise question is from an exam on data fitting.
- The second exercise question: Newton's method works well for simple root, but not good for double root. Why?

First homework assignment

- It includes 4 multiple choice questions and 2 normal exercise questions.
- The first exercise question is from an exam on data fitting.
- The second exercise question: Newton's method works well for simple root, but not good for double root. Why? Due to the term $\frac{f(x)}{f'(x)}$.

First homework assignment

- It includes 4 multiple choice questions and 2 normal exercise questions.
- The first exercise question is from an exam on data fitting.
- The second exercise question: Newton's method works well for simple root, but not good for double root. Why? Due to the term $\frac{f(x)}{f'(x)}$.
  - In this exercise, we will apply Taylor expansion to modify Newton's method to deal with the double root.
  - It will lead you to study on convergence theoretically and numerically.

First homework assignment

- It includes 4 multiple choice questions and 2 normal exercise questions.
- The first exercise question is from an exam on data fitting.
- The second exercise question: Newton's method works well for simple root, but not good for double root. Why? Due to the term $\frac{f(x)}{f'(x)}$.
  - In this exercise, we will apply Taylor expansion to modify Newton's method to deal with the double root.
  - It will lead you to study on convergence theoretically and numerically.
- For MC, you only need list your answers. But for exercise questions, you should include important intermediate steps.
- Next week we will have 4-hour exercise, so we will meet at databars directly.
- Upload the homework before **26/9 10pm** on DTU Learn.