

# 1 Taylor

Let  $f \in C^n[a, b]$  and assume  $f^{(n+1)}$  exists in  $(a, b)$ . Then for any  $c, x \in [a, b]$  there is some  $\zeta$  between  $c$  and  $x$  s.t.

$$f(x) = \sum_{k=0}^n \frac{f^{(k)}(c)(x-c)^k}{k!} + E_n(x) \quad (1)$$

where

$$E_n(x) = \frac{f^{(n+1)}(\zeta)(x-c)^{n+1}}{(n+1)!}$$

Equation (1) is called the Taylor expansion of  $f$  around  $c$ .

**Observation.** The famous *mean value theorem* is simply the case  $n = 0$  of Taylor's expansion: if  $f \in C[a, b]$  and  $f'$  exists on  $(a, b)$ , then for  $x, c \in [a, b]$

$$f(x) = f(c) + f'(\zeta)(x-c)$$

where  $\zeta$  is between  $c$  and  $x$ . Take  $x = b, c = a$  and the theorem appears:

$$f(b) - f(a) = f'(\zeta)(b-a)$$

We typically extend the Taylor approximation of  $f$  around a point  $r$ , where  $r = x + h$  is an approximation some value of interest  $x$ . This is useful because said approximation gives

$$f(r) = f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \dots + \frac{f^{(n)}(x)}{n!}h^n + E_n(h)$$

In other words, this strategy allows us to extend  $f(r)$  in terms of  $x$  and  $h$ , the approximation and its error. Usually,  $r, h$  are unknown but  $h$  can be bounded.

## 2 Alg. de Horner: Polynomial evaluation

Consider

$$p(x) = \sum_{i=0}^n a_i x^i$$

We wish to compute  $p(k)$  for a given  $k \in \mathbb{R}$  minimizing the number of operations. Directly computing  $a_0 + a_1 k + \dots$  leads to  $n$  sums. The  $i$ th term requires computing  $k^i$ , which means  $i$  product operations, for a total of  $\sum_{i=1}^n i = \frac{n(n+1)}{2}$  products. The total number of operations is then

$$\Theta = n + n(n+1)/2$$

The associated complexity is  $O(n^2)$ .

Horner's method consists of re-writing  $p(x)$  so that the number of products is reduced. One writes

$$p(x) = a_0 + x b_0$$

where  $b_{n-1} = a_n$  and for  $0 \leq i < n-1$ :

$$b_{i-1} = a_i + x b_i$$

Let  $p(x) = 3 + 5x - 4x^2 + 0x^3 + 6x^4$ , giving  $n = 4$ . Then  $b_3 = 6$  and

$$\begin{aligned} b_2 &= a_3 + x b_3 = 6x, & b_1 &= a_2 + x b_2 = -4 + x(6x), \\ b_0 &= a_1 + x b_1 = 5 + x(-4 + x(6x)) \end{aligned}$$

This finally gives

$$p(x) = 3 + x b_0 = 3 + x(5 + x(-4 + x(6x)))$$

Here, one must perform  $n$  sums again but only  $n$  products. Thus, there are  $\Theta = n + n = 2n$  operations, giving a complexity of  $O(n)$  (in the operation space). See the algorithm below:

```

input  $n; a_i, i = 0, \dots, n; x$ 
 $b_{n-1} \leftarrow a_n$ 
for  $i = n - 2$  to  $i = 0$ 
     $b_i = a_{i+1} + x * b_{i+1}$ 
od
 $y \leftarrow a_0 + x * b_0$ 
return  $y$ 

```

It is easy to see in this code that the **for** loop performs  $n - 1$  iterations, in each of which a single sum and a single product are computed. The  $n$ th sum and  $n$ th product are performed in the computation of  $y$ , the final result.

A more polished version includes the last computation (the one in the assignment of  $y$ ) within the loop and makes no use of indexes:

```

input  $n; a_i, i = 0, \dots, n; x$ 
 $b \leftarrow a_n$ 
for  $i = n - 2$  to  $i = -1$ 
     $b = a_{i+1} + x * b$ 
od
return  $b$ 

```

In Python,

```

def horner(coefs, x):
    n = len(coefs)-1
    b = coefs[n]

    for i in reversed(range(-1, n-1)):
        b = coefs[i+1] + x*b

    return b

```

It is trivial to adapt the code so that it returns the coefficients  $b_0, \dots, b_{n-1}$  and not the final result, if needed.

### 3 Error

Let  $r, \bar{r}$  be two real numbers s.t. the latter is an approximation of the first. We define the **error** of the approximation to be  $r - \bar{r}$ , and

$$\Delta r = |r - \bar{r}|, \quad \delta r = \frac{\Delta r}{|r|}$$

With  $r$  unknown the strategy is to work with a known bound of  $r$ .

## 4 Non-linear equations

The general problem is to find members of the set  $\mathcal{R}_f$  of roots of  $f \in \mathbb{R} \rightarrow \mathbb{R}$ . The numerical strategy is to iteratively approximate some  $r \in \mathcal{R}_f$  until some pre-established threshold in the error of approximation is met.

More formally, the numerical strategy produces a sequence  $\{x_k\}_{k \in \mathbb{N}}$  which satisfies

- $\lim_{k \rightarrow \infty} \{x_k\} = r$  for some  $r \in \mathcal{R}_f$
- Either  $e(x_k) < e(x_{k-1})$  or, more strongly,  $\lim_{k \rightarrow \infty} e(x_k) = 0$ , where  $e(x_k)$  is some appropriate measure of the error of approximation.

### 4.1 Bisection

A very simple procedure: if a root exists in  $[a, b]$ , it iteratively shrinks  $[a, b]$  in halves (keeping the halves which contain the root) until the interval is of sufficiently small length or the root is found.

**Theorem 1** (Intermediate value). If  $f$  is continuous in  $[a, b]$  and  $f(a)f(b) < 0$ , then  $\exists r \in \mathcal{R}_f$  s.t.  $r \in [a, b]$ .

Assume  $f$  is continuous. A root exists in  $[a, b]$  if  $f(a)f(b) < 0$  (**Theorem 1**). If that is the case, the midpoint  $(a + b)/2$  is taken as the approximation  $x_0$ . It is also trivial to observe that  $x_0$  is *at most* at a distance of  $(b - a)/2$  from the real root, so  $e_0 = |x_0 - r| \leq (b - a)/2$ .

If  $f(x_0) = 0$  the procedure must end because a root was found. Otherwise, suffices to find which half of the interval contains a root computing  $f(a)f(c)$  and, if needed,  $f(c)f(b)$ .

The iterations may stop after reaching a maximum number of steps, when  $|f(c)|$  is sufficiently close to zero, or when the error bound  $|e_k| \leq (b_k - a_k)/2$  (where  $[a_k, b_k]$  is the interval of this iteration) is sufficiently small.

(!) The algorithm not always converges. Take  $f(x) = 1/x$ . Clearly, it has no root. Yet setting  $a = -1, b = 1$  in the initial iteration falsely passes the test. (The problem obviously is that  $f$  is not continuous in  $[-1, 1]$ .) If one sets

**Input :**  $a, b, \delta, M, f$

**Output :** Tupla de la forma:  $(r, \text{cota de error})$

$f_a \leftarrow f(a)$

$f_b \leftarrow f(b)$

**if**  $f_a * f_b > 0$

**return** ?

**fi**

**for**  $i = 1$  **to**  $i = M$  **do**

$c \leftarrow a + (b - a)/2$

$f_c \leftarrow f(c)$

**if**  $f_c = 0$  **then**

**return**  $(c, 0)$

**fi**

$\epsilon = \frac{b - a}{2}$

**if**  $\epsilon < \delta$  **then**

**break**

**fi**

**if**  $f_a * f_c < 0$  **then**

$b \leftarrow c$

$f_b = f(b)$

**else**

$a \leftarrow c$

$f_a = f(a)$

**fi**

**od**

**return**  $(c, \epsilon)$

```

def bisection(f : callable, a : float, b : float, delta : float, M : int):

    s, e = f(a), f(b) # function values at (s)tart, (e)nd of interval

    if s*e > 0:
        raise ValueError("Interval [a, b] contains no root.")

    for i in range(M):

        c = a + (b-a)/2
        m = f(c) # value of f at (m)idpoint

        if m == 0:
            return c, 0

        e = (b-a)/2
        if e < delta:
            return c, e

        if s*m < 0:
            b = c
            e = f(b)
        else:
            a = c
            s = f(a)

    return c, e

```

**Theorem 2.** If  $\{[a_i, b_i]\}_{i=0}^{\infty}$  are the intervals generated by the bisection method on iterations  $i = 0, 1, \dots$ , then:

1.  $\lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} b_n$  is a member of  $\mathcal{R}_f$ .
2. If  $c_n = \frac{1}{2}(a_n + b_n)$ ,  $r = \lim_{n \rightarrow \infty} c_n$ , then  $|r - c_n| \leq \frac{1}{2^{n+1}}(b_0 - a_0)$

**Proof. (1)** It is clear that  $a_i \leq a_{i+1}$  and  $b_i \geq b_{i+1}$ , since the interval on each iteration shrinks in one direction.

$\therefore a_n, b_n$  are monotonous.

But clearly  $a_n$  is bounded by  $b_0$  and  $b_n$  is bounded by  $a_0$ .

$\therefore a_n, b_n$  are monotonous and bounded.

$\therefore$  Their limits exist.

It is also clear that the interval shrinks to half its size on each iteration:

$$b_n - a_n = \frac{1}{2}(b_{n-1} - a_{n-1}), \quad n \geq 1 \quad (1)$$

By recurrence on (1),

$$b_n - a_n = \frac{1}{2^n}(b_0 - a_0), \quad n \geq 0 \quad (2)$$

Then

$$\lim_{n \rightarrow \infty} a_n - \lim_{n \rightarrow \infty} b_n = \lim_{n \rightarrow \infty} (a_n - b_n) = \lim_{n \rightarrow \infty} \frac{1}{2^n}(b_0 - a_0) = 0 \quad (3)$$

$\therefore \lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} b_n$ .

Since the limit of  $a_n, b_n$  exists and  $f$  is by assumption continuous, the composition limit theorem applies and:

$$\begin{aligned} & \lim_{n \rightarrow \infty} (f(a_n) \cdot f(b_n)) \\ &= \lim_{n \rightarrow \infty} f(a_n) \cdot \lim_{n \rightarrow \infty} f(b_n) \quad \{\text{Product of limits}\} \\ &= f\left(\lim_{n \rightarrow \infty} a_n\right) \cdot f\left(\lim_{n \rightarrow \infty} b_n\right) \quad \{\text{Composition limit theorem}\} \\ &= [f(r)]^2 \quad \left\{r = \lim_{n \rightarrow \infty} a_n\right\} \end{aligned} \quad (4)$$

The invariant of the algorithm is  $f(a_n)f(b_n) < 0$ . But due to the last result,

$$\lim_{n \rightarrow \infty} f(a_n)f(b_n) \leq 0 \iff [f(r)]^2 \leq 0 \iff f(r) = 0$$

$\therefore r = \lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} b_n$  is a root.



(2) Follows directly from result (2)

$$\begin{aligned} |r - c_n| &= \left| r - \frac{1}{2}(b_n - a_n) \right| \\ &\leq \left| \frac{1}{2}(b_n - a_n) \right| \\ &= \left| \frac{1}{2^{n+1}}(b_0 - a_0) \right| \end{aligned} \quad \{\text{Result (2)}\}$$

## 4.2 Newton's method

Assume  $r \in \mathcal{R}_f$  and  $r = x + h$ , with  $x$  an approximation of  $r$  and  $h$  its error. Assume  $f''$  exists and is continuous in some  $I$  around  $x$  s.t.  $r \in I$ . What we explained on Taylor expansions around a point gives:

$$0 = f(r) = f(x + h) = f(x) + f'(x)h + O(h^2)$$

If  $x$  is sufficiently close to  $r$ ,  $h$  is small and  $h^2$  even smaller, so that  $O(h^2)$  is unconsiderable:

$$0 \approx f(x) + hf'(x)$$

Therefore,

$$h \approx -\frac{f(x)}{f'(x)} \tag{1}$$

From this follows that  $r = x + h$  is approximated by

$$r \approx x - \frac{f(x)}{f'(x)}$$

Since the approximation in (5) truncated the terms of  $O(h^2)$  complexity, this new approximation is closer to  $r$  than  $x$  originally was. In other words,  $x - f(x)/f'(x)$  is a better approximation to  $r$  than  $x$  itself.

Thus, if  $x_0$  is an original approximation, we can define

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2}$$

to produce a sequence of approximations. This is the fundamental idea of Newton's method.

```

Input:  $x_0, M, \delta, \epsilon;$ 
 $v \leftarrow f(x_0)$ 
if  $|v| < \epsilon$  then return  $x_0$  fi
for  $k = 1$  to  $k = M$  do
     $x_1 \leftarrow x_0 - \frac{v}{f'(x_0)}$ 
     $v \leftarrow f(x_1)$ 
    if  $|x_1 - x_0| < \delta \vee v < \epsilon$  then
        return  $x_1$ 
    fi
     $x_0 \leftarrow x_1$ 
od
return  $x_0$ 

```

The predicate  $|x_1 - x_0| < \delta$  checks whether our algorithm is adjusting  $x$  in a negligible degree. If that is the case, we should stop.

**Theorem 3.** If  $f''$  continuous around  $r \in \mathcal{R}_f$  and  $f'(r) \neq 0$ , then there is some  $\delta > 0$  s.t. if  $|r - x_0| \leq \delta$ , then:

- $|r - x_n| \leq \delta$  for all  $n \geq 1$ .
- $\{x_n\}$  converges to  $r$
- The convergence is quadratic, i.e. there is a constant  $c(\delta)$  and a natural  $N$  s.t.  $|r - x_{n+1}| \leq c |r - x_n|^2$  for all  $n \geq N$ .

**Proof.** Let  $e_n = r - x_n$  be the error in the  $n$ th approximation. Assume  $f''$  is continuous and  $f(r) = 0, f'(r) \neq 0$ . Then

$$\begin{aligned}
 e_{n+1} &= r - x_{n+1} \\
 &= r - \left( x_n - \frac{f(x_n)}{f'(x_n)} \right) \\
 &= r - x_n + \frac{f(x_n)}{f'(x_n)} \\
 &= \frac{e_n f'(x_n) + f(x_n)}{f'(x_n)}
 \end{aligned} \tag{3}$$

Thus, the error at any given iteration is a function of the error at the previous iteration. Now consider the expansion of  $f(r)$  as

$$f(r) = f(x_n - e_n) = f(x_n) + e_n f'(x_n) + \frac{e_n^2 f''(\zeta_n)}{2} \quad (4)$$

for  $\zeta_n$  between  $x_n$  and  $r$ . This equation gives

$$e_n f'(x_n) + f(x_n) = -\frac{1}{2} f''(\zeta_n) e_n^2 \quad (5)$$

The expression in (5) is the numerator in (3), whereby we obtain via substitution:

$$e_{n+1} = -\frac{1}{2} \frac{f''(\zeta_n) e_n^2}{f'(x_n)} \quad (6)$$

Equation (6) ensures that the error scales quadratically. Now we wish to bound the error expression in (6). To bound  $e_{n+1}$ , we take  $\delta > 0$  to define a neighbourhood of length  $\delta$  around  $r$ . For any  $x$  in this neighbourhood, (6) reaches its maximum when the numerator is maximized and the denominator is minimized:

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

In other words,  $c(\delta)$  is the maximum value which  $e_{n+1}$  can take if  $\zeta_n, x_n$  are assumed to belong to the neighbourhood. Now we make two assumptions:

1.  $x_0$  belongs to the neighbourhood, i.e.  $|x_0 - r| \leq \delta$
2.  $\delta$  is sufficiently small so that  $\varrho := \delta c(\delta) < 1$ .

Note that, since  $\zeta_0$  is between  $x_0$  and  $r$ , assumption (1) ensures that  $\zeta_0$  is also in the neighbourhood, i.e.  $|r - \zeta_0| \leq \delta$ . Then we have:

$$|e_0| = \frac{1}{2} |f''(\zeta_0)/f'(x_0)| \leq c(\delta)$$

Then:

$$\begin{aligned} |x_1 - r| &= |e_1| \\ &= \left| e_0^2 \cdot \frac{1}{2} f''(\zeta_0)/f'(x_0) \right| \\ &\leq |e_0^2| c(\delta) && \left\{ \frac{1}{2} f''(\zeta_0)/f'(x_0) \leq c(\delta) \right\} \\ &\leq |e_0| \delta c(\delta) && \{|e_0| \leq \delta\} \\ &= |e_0| \varrho && \{\varrho = \delta c(\delta)\} \\ &< |e_0| && \{\varrho < 1\} \\ &\leq \delta \end{aligned}$$

$\therefore |e_1| < |e_0| \leq \delta$ , which means the error decreases. This argument may be repeated inductively, giving:

$$\begin{aligned}
|e_1| &\leq \varrho |e_0| \\
|e_2| &\leq \varrho |e_1| \leq \varrho^2 |e_0| \\
|e_3| &\leq \varrho |e_2| \leq \varrho^3 |e_0| \\
&\vdots
\end{aligned}$$

In general,  $|e_n| \leq \varrho^n |e_0|$ . And since  $0 \leq \varrho < 1$ , we have  $\varrho^n \rightarrow 0$  when  $n \rightarrow \infty$ , entailing that  $|e_n| \rightarrow 0$  when  $n \rightarrow \infty$ .

**Theorem 4.** If  $f''$  is continuous in  $\mathbb{R}$ , and if  $f$  is increasing, convex, and has a root, then said root is unique and Newton's method converges to it from any starting point.

Recall that  $f$  is convex if  $f''(x) > 0$  for all  $x$ . Graphically, it is convex if the line connecting two arbitrary points of  $f$  lies above the curve of  $f$  between those two points.

### 4.3 Secant method

In Newton's method,

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

The function of interest is  $f$ . We cannot escape computing  $f(x_n)$ , but it would be desirable to avoid the computation of  $f'(x_n)$ , which may potentially be expensive. Since

$$f'(x) = \lim_{h \rightarrow x} \frac{f(x) - f(h)}{x - h}$$

it is natural to suggest

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

Graphically, this means we are not using the line tangent to the point  $(x_n, f(x_n))$  but the line secant to the points  $(x_n, f(x_n))$  and  $(x_{n-1}, f(x_{n-1}))$ . The point  $x_{n+1}$  is then the value of  $x$  where this secant line has a root.

## 4.4 Fixed point iteration

The key observation is this: if  $r \in \mathcal{R}_f$ , then  $g(x) = x - kf(x)$  has  $r$  as fixed point, for any  $k \in \mathbb{R}$ . Inversely, if  $g$  has a fixed point in  $r$ , then  $r \in \mathcal{R}_f$ .

**Theorem 5.** (1) Let  $g \in C[a, b]$  and assume  $g(x) \in [a, b]$  for all  $x \in [a, b]$ . Then there is a fixed point of  $g$  in  $[a, b]$ .

(2) If, on top of previous conditions,  $g$  is differentiable in  $(a, b)$  and there is some  $k < 1$  s.t.  $|g'(x)| \leq k$  for all  $x \in (a, b)$ , then the fixed point referred in (1) is unique.

**Theorem 6** (Mean value theorem). Let  $f : [a, b] \rightarrow \mathbb{R}$  continuous and differentiable on  $(a, b)$  with  $a < b$ . Then there is some  $c \in (a, b)$  s.t.

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

The interpretation is simple: consider the line secant to  $f$  on  $a, b$ . The theorem ensures that there is some point  $c$  s.t. the line tangent to  $c$  is parallel to said secant (equal slopes).

**Proof.** (1) If  $a$  or  $b$  are fixed points the proof is done so assume otherwise. Since  $g(x) \in [a, b]$ , we have  $g(a) > a$  and  $g(b) < b$ .

Take  $\varphi(x) = g(x) - x$ , which is continuous and defined in  $[a, b]$ . Then

$$\varphi(a) = g(a) - a > 0, \quad \varphi(b) = g(b) - b < 0$$

Then  $\varphi(a)\varphi(b) < 0$ . Then, by the intermediate value theorem,  $\varphi$  has a root in  $(a, b)$ . In otherwords, there is at least one  $p$  s.t.

$$\varphi(p) = g(p) - p = 0$$

$\therefore g(p) = p$  is a fixed point of  $g$ .

(2) Assume two distinct fixed points  $p, q$  exist in  $[a, b]$ . The mean value theorem ensures the existence of some  $\zeta$  between  $p, q$  (and thus in  $[a, b]$ ) s.t.

$$g'(\zeta) = \frac{g(a) - g(b)}{a - b} \iff g'(\zeta)(a - b) = g(a) - g(b) \quad (1)$$

By hypothesis,  $|g'(x)| \leq k < 1$ . Since  $p, q$  are assumed to be fixed points, equation (1) gives:

$$\begin{aligned} |p - q| &= |g(p) - g(q)| \\ &= |g'(\zeta)| |p - q| \\ &\leq k |p - q| < |p - q| \end{aligned}$$

But this is absurd. The contradiction arises from assuming  $p, q$  to be distinct. Therefore, the fixed point is unique.

The fixed point algorithm begins with an approximation  $p_0$ . Then,

$$p_n = g(p_{n-1})$$

If  $g$  is continuous and the sequence converges, then it converges to a fixed point, since:

$$p := \lim_{n \rightarrow \infty} p_n = \lim_{n \rightarrow \infty} g(p_{n-1}) = g\left(\lim_{n \rightarrow \infty} p_{n-1}\right) = g(p)$$

```

Input:  $p, M, \delta$ 
 $p_{\text{previous}} = p$ 
for  $i = 1$  to  $i = M$  do
     $p \leftarrow g(p)$ 
    if  $|p - p_{\text{previous}}| < \delta$  then
        return  $p$ 
    fi
     $p_{\text{previous}} = p$ 
od
return  $p$ 

```

**Theorem 7.** Let  $g \in C[a, b]$  be a self-map of  $[a, b]$  differentiable in  $(a, b)$ . Assume there is a constant  $0 < k < 1$  s.t.  $|g'(x)| \leq k$  for all  $x \in (a, b)$ .

For all  $p_0 \in [a, b]$ , the sequence  $p_n = g(p_{n-1})$  converges to the unique fixed point  $p$  in  $(a, b)$ .

**Proof.** The mean value theorem ensures that

$$\begin{aligned}
 |p_n - p| &= |g(p_{n-1}) - g(p)| \\
 &= |g'(\zeta_n)| |p_{n-1} - p| \\
 &\leq k |p_{n-1} - p|
 \end{aligned}$$

with  $\zeta_n \in (a, b)$ . More succinctly, with  $e_n := p_n - p$ ,

$$|e_n| \leq k |e_{n-1}| \leq k |e_{n-2}| \leq \dots \leq k |e_0|$$

By recurrence,

$$|e_n| \leq k^n |e_0|$$

Since  $0 < k < 1$ ,  $k^n \rightarrow 0$  when  $n \rightarrow \infty$ , which entails  $|e_n| \rightarrow 0$  when  $n \rightarrow \infty$ . It follows that  $\{p_n\} \rightarrow p$  when  $n \rightarrow \infty$ .

Now let us consider the error of this method. Take  $p_n = p + e_n$  and consider the Taylor expansion of  $g$  around  $p$  evaluated at  $p_n = p + e_n$ :

$$g(p_n) = g(p + e_n) = \sum_{i=1}^{m-1} \frac{g^{(i)}(p)}{i!} e_n^i + \frac{g^{(m)}(\zeta_n)}{(n+1)!} e_n^m \quad (2)$$

See that in (2),  $n$  corresponds to the iteration we are dealing with, and thus  $\zeta_n$  and  $e_n$  depend on it. On the contrary,  $m$  is the degree to which we expand the series of  $g$  around  $p$  evaluated at  $p_n$ . We also assume that  $\zeta_n$  lies between  $p_n$  and  $p$ .

By definition,  $g(p_n) = p_{n+1}$  so (2) is nothing but an expression for this value. Assume  $g^{(k)}(p) = 0$  for  $k = 1, 2, \dots, m-1$ , but  $g^{(m)}(p) \neq 0$ . Then

$$\begin{aligned} e_{n+1} &= p_{n+1} - p \\ &= g(p_n) - g(p) \\ &= \frac{g^{(m)}(\zeta_n)}{m!} e_n^m \end{aligned}$$

More succinctly,

$$e_{n+1} = \frac{g^{(m)}(\zeta_n)}{m!} e_n^m$$

Then

$$\lim_{n \rightarrow \infty} \left| \frac{e_{n+1}}{e_n^m} \right| = \frac{|g^{(m)}(p)|}{m!}$$

which is a constant. In conclusion, if the derivatives of  $g$  are null in  $p$  up to the order  $m-1$ , the method has an order of convergence of at least  $m$ . Three results follow from this fact.



## 5 P2

(1) Let  $f(x) = (x+2)(x+1)^2x(x-1)^3(x-2)$ . To which root does the bisection method converge on the following intervals?

$$[-1.5, 2.5], \quad [-0.5, 2.4], \quad [-0.5, 3], \quad [-3, -0.5]$$

(a) The midpoint of  $I_0 = [-1.5, 2.5]$  is  $c_0 := (2.5 - 1.5)/2 = 1/2$ . Since  $f(a)f(c) < 0$ , we have  $I_1 = [-1.5, 0.5]$ . The midpoint of  $I_1$  is  $c_1 = -0.5$ , so  $I_2$  will be  $[-0.5, 0.5]$ . The only root in this interval is  $r = 0$ , so the algorithm converges to it.

(b) The midpoint of  $I_0 = [-0.5, 2.4]$  is  $c := (2.4 - 0.5)/2 = 0.95$ . Then  $I_1 = [-1.5, 0.95]$ . Same logic gives  $c_1 = -0.725$  and then  $I_2 = [-0.725, 0.95]$ . The only root here is zero again.

(c, d) Same.

(2) We wish to find a root of  $f$  in  $[a, b]$  using bisection method and ensuring that the error is not greater than  $\epsilon \in \mathbb{R}^+$ .

(a) Estimate the number of iterations sufficient to meet the criterion.

(b) What is the number of iterations for  $a = 0, b = 1, \epsilon = 10^{-5}$ ?

Let  $e_n = x_n - r$ . It is trivial to note that  $|e_n| \leq \frac{b_n - a_n}{2}$ . Furthermore, the length of  $I_1$  is half the length of  $I_0$ , that of  $I_2$  is half that of  $I_1$ , etc. In other words,

$$|e_0| \leq \frac{b-a}{2}, \quad |e_1| \leq \frac{b-a}{2^2}, \quad |e_2| \leq \frac{b-a}{2^3}, \dots$$

In general,

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

Imposing

$$|e_n| \leq \frac{b-a}{2^{n+1}} \leq \epsilon$$

we satisfy our criterion, but we wish to express this bound in terms of  $n$ . Now, clearly,

$$\begin{aligned} \frac{b-a}{2^{n+1}} &\leq \epsilon \\ \iff \frac{b-a}{\epsilon} &\leq 2^{n+1} \\ \iff \log_2 \left( \frac{b-a}{\epsilon} \right) - 1 &\leq n \\ \iff \log_2 \left( \frac{b-a}{\epsilon} \right) &\leq n \\ \iff \frac{\ln \left( \frac{b-a}{\epsilon} \right)}{\ln 2} &\leq n \end{aligned}$$

which is our final answer.

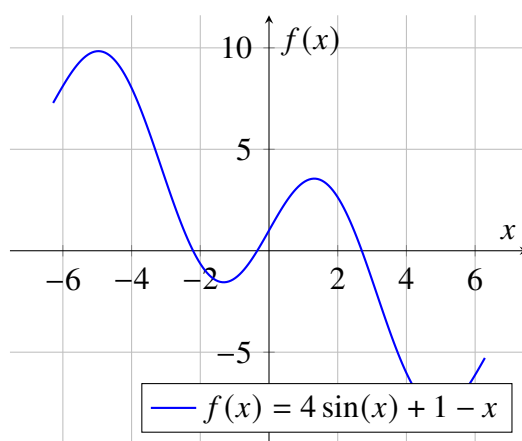
(b) For  $a = 0, b = 1, \epsilon = 10^{-5}$ , we need

$$n \geq \frac{\ln\left(\frac{1}{10^{-5}}\right)}{\ln 2} \approx 16.609$$

so  $n = 17$  would suffice.

(3) Determine graphically some root of  $f(x) = 4 \sin x + 1 - x$  and perform three iterations of the bisection method to approximate. How many steps are needed to ensure an error less than  $10^{-3}$ ?

Let us unveil the full power of LaTeX:



I'm too lazy to perform the steps of the algorithm. The number of steps needed again are given by

$$n \geq \frac{\ln\left(\frac{4-2}{10^{-3}}\right)}{\ln 2} \approx 10.96$$

so taking  $n = 11$  suffices.

(4) Let  $a > 0$ . Computing  $\sqrt{a}$  is equivalent to finding the root of  $f(x) = x^2 - a$ .

(a) Show that Newton's sequence for this case is

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{a}{x_n} \right)$$

(b) Prove that for any  $x_0 > 0$ , the approximations  $\{x_n\}$  satisfy  $x_n \geq \sqrt{a}$  for  $n \geq 1$ .

(c) Prove  $\{x_n\}$  is decreasing.

(d) Conclude that the sequence converges to  $\sqrt{a}$

(a) In Newton's algorithm,

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

Clearly,

$$f'(x) = \frac{d}{dx}(x^2 - a) = 2x$$

Therefore,

$$\begin{aligned} x_{n+1} &= x_n - \frac{x_n^2 - a}{2x_n} \\ &= x_n - \frac{1}{2} \left( x_n - \frac{a}{x_n} \right) \\ &= \frac{1}{2}x_n + \frac{1}{2} \frac{a}{x_n} \\ &= \frac{1}{2} \left( x_n + \frac{a}{x_n} \right) \quad \blacksquare \end{aligned}$$

(b) Let  $x_0 > 0$ . Recall that, among all Pythagorean means, the arithmetic mean is the greatest, assuming positively-valued vectors. In particular, it is greater or equal to the geometric mean:

$$\frac{1}{N} \sum_{i=1}^n y_i \geq \sqrt[n]{\prod_{i=1}^n y_i}$$

for any set of points  $y_1, \dots, y_n$  all positive. In particular,

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{a}{x_n} \right) \geq \sqrt{x_n \frac{a}{x_n}} = \sqrt{a} \quad \blacksquare$$

(c)

$$\begin{aligned} \frac{1}{2} \left( x_n + \frac{a}{x_n} \right) &\leq x_n \\ \iff x_n + \frac{a}{x_n} &\leq 2x_n \\ \iff \frac{a}{x_n} &\leq x_n \\ \iff a &\leq x_n^2 \\ \iff \sqrt{a} &\leq x_n \end{aligned}$$

which is true due to point (b).

(d) Let  $e_n = x_n - \sqrt{a}$ . We have shown  $\{x_n\}$  to be decreasing and bounded below by  $\sqrt{a}$ . Therefore, it converges to a limit  $L$  (with  $L$  the infimum of  $\{x_n\}$ ). Then

$$\lim_{n \rightarrow \infty} x_n = \frac{1}{2} \lim_{n \rightarrow \infty} \left( x_{n-1} + \frac{a}{x_{n-1}} \right) = \frac{1}{2} L + \frac{a}{2L}$$

This induces the equation

$$\begin{aligned} L = \frac{L}{2} + \frac{a}{2L} &\iff \frac{L}{2} = \frac{a}{2L} \\ &\iff L^2 = a \\ &\iff L = \sqrt{a} \quad \blacksquare \end{aligned}$$

**(5)** Propose an iteration formula to approximate  $\frac{1}{\sqrt{a}}$ , with  $a > 0$ , using Newton's method. Decide the number of iterations needed so that the relative error in the approximation is less than  $10^{-4}$  when starting from  $x_0 = 1$  and taking  $a = 5$ .

Error:  $e_n = r - x_n$ , quadratic, i.e.  $|r - x_{n+1}| \leq c|r - x_n|^2$ .

(a. Iteration formula) Let  $a > 0$  and assume we wish to approximate  $1/\sqrt{a}$ . Let  $\varphi = \frac{1}{a}$ , so that  $\frac{1}{\sqrt{a}} = \sqrt{\varphi}$ . We see that we can express the problem of finding the reciprocal of a root in terms of a simple root.

We know from the previous exercise that the iteration formula for  $\sqrt{\varphi}$  is

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{\varphi}{x_n} \right)$$

Now take  $x_0 = 1$  and  $a = 5$ , so that  $\varphi = \frac{1}{5}$ . The relative error of approximation on iteration  $n$  is

$$e_n = \frac{\left| x_n - \frac{1}{\sqrt{5}} \right|}{\frac{1}{\sqrt{5}}}$$

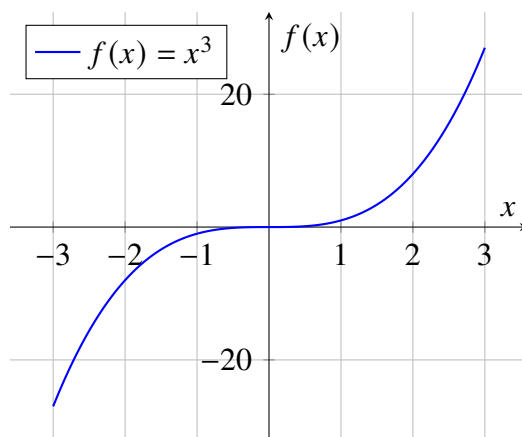
Brute-forcing allows us to see that  $x_0, x_1, x_2, x_3$  do not meet the criterion, but

$$x_4 = 0.4472137791286728 \text{ ( jaja )}$$

has  $e_4 < 10^{-4}$ .

(6) Propose an iteration formula for  $\sqrt[3]{R}$  where  $R > 0$ . Plot the function to see where the procedure converges.

Observe that finding  $\sqrt[3]{R}$  is equivalent to finding a root of  $f(x) = x^3 - R$ .



But  $f(x)$  is simply a vertical displacement of  $x^3$ , so  $\frac{d}{dx}x^3 = \frac{d}{dx}f(x)$  (which holds algebraically). In particular, the derivative of  $x^3$  approaches 0 as  $x \rightarrow 0$ , meaning that Newton's method will fail to converge for intervals of length  $L$  around 0 (with  $L$  unspecified). The graph suggests that an appropriate value for  $L$  is 1.

That said, since  $\frac{d}{dx}f(x) = \frac{d}{dx}x^3$  (in other words, since the derivative of the function is independent of  $R$ ), and  $\frac{d}{dx}x^3 = 3x^2$ , we propose

$$x_{n+1} = x_n - \frac{x_n^3}{3x_n^2} = x_n - \frac{x_n}{3} = \frac{2x_n}{3}$$