Chapter 1

Solving nonlinear equations

1.1 Bisection

1.1.1 Introduction

Linear equations are of the form:

$$\textit{find } x \textit{ such that } ax + b = 0$$

and are easy to solve. Some nonlinear problems are also easy to solve, e.g.,

$$\textit{find } x \textit{ such that } ax^2 + bx + c = 0.$$

Cubic and quartic equations also have solutions for which we can obtain a formula. But most equations to not have simple formulae for this soltuions, so numerical methods are needed.

References

- Süli and Mayers [1, Chapter 1]. We'll follow this pretty closely in lectures.
- Stewart (*Afternotes ...*), [3, Lectures 1–5]. A well-presented introduction, with lots of diagrams to give an intuitive introduction.
- Moler (Numerical Computing with MATLAB) [2, Chap. 4]. Gives a brief introduction to the methods we study, and a description of MATLAB functions for solving these problems.
- The proof of the convergence of Newton's Method is based on the presentation in [5, Thm 3.2].

Our generic problem is:

Let f *be a continuous function on the interval* $[a, b]$.
Find $\tau = [a, b]$ *such that* $f(\tau) = 0$.

Here f is some specified function, and τ is the *solution* to $f(x) = 0$.

This leads to two natural questions:

(1) How do we know there is a solution?

(2) How do we find it?

The following gives *sufficient* conditions for the existence of a solution:

Proposition 1.1.1. *Let* f *be a real-valued function that is defined and continuous on a bounded closed interval* $[a, b] \subset \mathbb{R}$. *Suppose that* $f(a)f(b) \leqslant 0$. *Then there exists* $\tau \in [a, b]$ *such that* $f(\tau) = 0$.

Take notes:

OK – now we know there is a solution τ to $f(x) = 0$, but how to we actually solve it? Usually we don't! Instead we construct a sequence of estimates $\{x_0, x_1, x_2, x_3, \dots\}$ that *converge* to the true solution. So now we have to answer these questions:

(1) How can we construct the sequence $x_0, x_1, \dots$?

(2) How do we show that $\lim_{k\to\infty} x_k = \tau$?

There are some subtleties here, particularly with part (2). What we would like to say is that at each step the error is getting smaller. That is

$$|\tau - x_k| < |\tau - x_{k-1}| \qquad \text{for } k = 1, 2, 3, \dots.$$

But we can't. Usually all we can say is that the *bounds* on the error is getting smaller. That is: *let* ε_k *be a bound on the error at step* k

$$|\tau - x_k| < \varepsilon_k,$$

then $\varepsilon_{k+1} < \mu\varepsilon_k$ for some number $\mu \in (0, 1)$. It is easiest to explain this in terms of an example, so we'll study the simplest method: *Bisection*.

1.1.2 Bisection

The most elementary algorithm is the "*Bisection Method*" (also known as "Interval Bisection"). Suppose that we know that f changes sign on the interval $[a, b] = [x_0, x_1]$ and, thus, $f(x) = 0$ has a solution, τ, in $[a, b]$. Proceed as follows

1. Set x_2 to be the midpoint of the interval $[x_0, x_1]$.

2. Choose one of the sub-intervals $[x_0, x_2]$ and $[x_2, x_1]$ where f change sign;

3. Repeat Steps 1–2 on that sub-interval, until f sufficiently small at the end points of the interval.

This may be expressed more precisely using some *pseudocode*.

Method 1.1.2 (Bisection)**.**
Set eps to be the stopping criterion.
If $|f(a)| \leqslant eps$, return a. Exit.
If $|f(b)| \leqslant eps$, return b. Exit.
Set $x_0 = a$ and $x_1 = b$.
Set $x_L = x_0$ and $x_R = x_1$.
Set $k = 1$
while($|f(x_k)| > eps$)
    $x_{k+1} = (x_L + x_R)/2$;
    if $(f(x_L)f(x_{k+1}) < eps)$
        $x_R = x_{k+1}$;
    else
        $x_L = x_{k+1}$
    end if;
    $k = k + 1$
end while;

Example 1.1.3. *Find an estimate for $\sqrt{2}$ that is correct to 6 decimal places.*
Solution: Try to solve the equation $f(x) := x^2 - 2 = 0$ on the interval $[0, 2]$. Then proceed as shown in Figure 1.1 and Table 1.1.

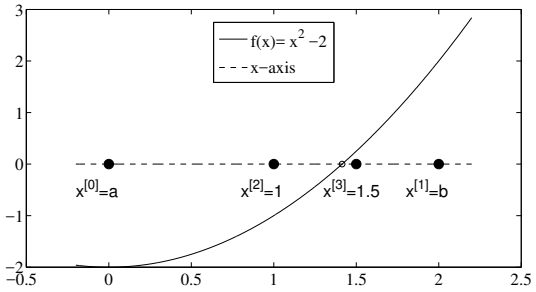

Fig. 1.1: Solving $x^2 - 2 = 0$ with the Bisection Method

Note that at steps 4 and 10 in Table 1.1 the error actually *increases*, although the bound on the error is decreasing.

1.1.3 The bisection method works

The main advantages of the Bisection method are

- It will always work.
- After k steps we know that

Theorem 1.1.4.

$$|\tau - x_k| \leqslant (\frac{1}{2})^{k-1}|b - a|, \quad \textit{for } k = 2, 3, 4, ...$$

k	x_k	$\lvert x_k - \tau \rvert$	$\lvert x_k - x_{k-1} \rvert$
0	0.000000	1.41	
1	2.000000	5.86e-01	
2	1.000000	4.14e-01	1.00
3	1.500000	8.58e-02	5.00e-01
4	1.250000	1.64e-01	2.50e-01
5	1.375000	3.92e-02	1.25e-01
6	1.437500	2.33e-02	6.25e-02
7	1.406250	7.96e-03	3.12e-02
8	1.421875	7.66e-03	1.56e-02
9	1.414062	1.51e-04	7.81e-03
10	1.417969	3.76e-03	3.91e-03
⋮	⋮	⋮	⋮
22	1.414214	5.72e-07	9.54e-07

Table 1.1: Solving $x^2 - 2 = 0$ with the Bisection Method

Take notes:

A disadvantage of bisection is that it is not as efficient as some other methods we'll investigate later.

1.1.4 Improving upon bisection

The bisection method is not very efficient. Our next goals will be to derive better methods, particularly the *Secant Method* and *Newton's method*. We also have to come up with some way of expressing what we mean by "better"; and we'll have to use Taylor's theorem in our analyses.

1.1.5 Exercises

Exercise 1.1. Does Proposition 1.1.1 mean that, if there is a solution to $f(x) = 0$ in $[a, b]$ then $f(a)f(b) \leqslant 0$? That is, is $f(a)f(b) \leqslant 0$ a *necessary* condition for their being a solution to $f(x) = 0$? Give an example that supports your answer.

Exercise 1.2. Suppose we want to find $\tau \in [a, b]$ such that $f(\tau) = 0$ for some given f, a and b. Write down an estimate for the number of iterations K required by the bisection method to ensure that, for a given ε, we know $|x_k - \tau| \leqslant \varepsilon$ for all $k \geqslant K$. In particular, how does this estimate depend on f, a and b?

Exercise 1.3. How many (decimal) digits of accuracy are gained at each step of the bisection method? (If you prefer, how many steps are need to gain a single (decimal) digit of accuracy?)

Exercise 1.4. Let $f(x) = e^x - 2x - 2$. Show that there is a solution to the problem: *find* $\tau \in [0, 2]$ *such that* $f(\tau) = 0$.
Taking $x_0 = 0$ and $x_1 = 2$, use 6 steps of the bisection method to estimate τ. Give an upper bound for the error $|\tau - x_6|$. (You may use a computer program to do this).

1.2 The Secant Method

1.2.1 Motivation

Looking back at Table 1.1 we notice that, at step 4 the error *increases* rather *decreases*. You could argue that this is because we didn't take into account how close x_3 is to the true solution. We could improve upon the bisection method as described below. The idea is, given x_{k-1} and x_k, take x_{k+1} to be the zero of the line intersects the points $\big(x_{k-1}, f(x_{k-1})\big)$ and $\big(x_k, f(x_k)\big)$. See Figure 1.2.

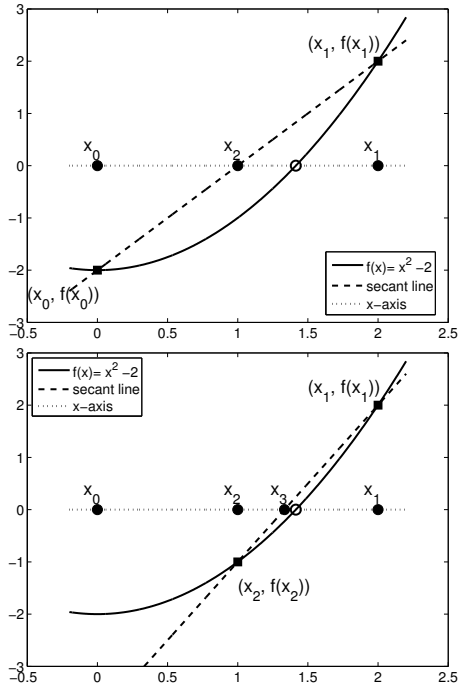

Fig. 1.2: The Secant Method for Example 1.2.2

Method 1.2.1 (Secant). [1] *Choose* x_0 *and* x_1 *so that there is a solution in* $[x_0, x_1]$. *Then define*

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}. \tag{1.2.1}$$

Example 1.2.2. *Use the Secant Method to solve the nonlinear problem* $x^2 - 2 = 0$ *in* $[0, 2]$. The results are shown in Table 1.2. By comparing Tables 1.1 and 1.2, we see that for this example, the Secant method is *much* more efficient than Bisection. We'll return to why this is later.

The Method of Bisection could be written as the weighted average

$$x_{k+1} = (1 - \sigma_k)x_k + \sigma_k x_{k-1}, \quad \text{with } \sigma_k = 1/2.$$

[1] The name comes from the name of the line that intersects a curve at two points. There is a related method called "false position" which was known in India in the 3rd century BC, and China in the 2nd century BC.

k	x_k	$\lvert x_k - \tau\rvert$
0	0.000000	1.41e
1	2.000000	5.86e-01
2	1.000000	4.14e-01
3	1.333333	8.09e-02
4	1.428571	1.44e-02
5	1.413793	4.20e-04
6	1.414211	2.12e-06
7	1.414214	3.16e-10
8	1.414214	4.44e-16

Table 1.2: Solving $x^2 - 2 = 0$ using the Secant Method

We can also think of the Secant method as being a *weighted average*, but with σ_k chosen to obtain faster convergence to the true solution. Looking at Figure 1.2 above, you could say that we should choose σ_k depending on which is smaller: $f(x_{k-1})$ or $f(x_k)$. If (for example) $|f(x_{k-1})| < |f(x_k)|$, then probably $|\tau - x_{k-1}| < |\tau - x_k|$. This gives another formulation of the Secant Method.

$$x_{k+1} = (1 - \sigma_k)x_k + \sigma_k x_{k-1}, \tag{1.2.2}$$

where

$$\sigma_k = \frac{f(x_k)}{f(x_k) - f(x_{k-1})}.$$

When its written in this form it is sometimes called a *relaxation method*.

1.2.2 Order of Convergence

To compare different methods, we need the following concept:

Definition 1.2.3 (Linear Convergence). *Suppose that* $\tau = \lim_{k\to\infty} x_k$. *Then we say that the sequence* $\{x_k\}_{k=0}^{\infty}$ *converges to* τ **at least linearly** *if there is a sequence of positive numbers* $\{\varepsilon_k\}_{k=0}^{\infty}$, *and* $\mu \in (0, 1)$, *such that*

$$\lim_{k\to\infty} \varepsilon_k = 0, \tag{1.2.3a}$$

and

$$|\tau - x_k| \leqslant \varepsilon_k \quad \textit{for} \quad k = 0, 1, 2, \ldots. \tag{1.2.3b}$$

and

$$\lim_{k\to\infty} \frac{\varepsilon_{k+1}}{\varepsilon_k} = \mu. \tag{1.2.3c}$$

So, for example, the bisection method converges at least linearly.

The reason for the expression "*at least*" is because we usually can only show that a set of upper bounds for the errors converges linearly. If (1.2.3b) can be strengthened to the equality $|\tau - x_k| = \varepsilon_k$, then the $\{x_k\}_{k=0}^{\infty}$ converges linearly, (not just "at least" linearly).

As we have seen, there are methods that converge more quickly than bisection. We state this more precisely:

Definition 1.2.4 (Order of Convergence). *Let* $\tau = \lim_{k\to\infty} x_k$. *Suppose there exists* $\mu > 0$ *and a sequence of positive numbers* $\{\varepsilon_k\}_{k=0}^{\infty}$ *such that (1.2.3a) and and (1.2.3b) both hold. Then we say that the sequence* $\{x_k\}_{k=0}^{\infty}$ *converges* with at least order q *if*

$$\lim_{k\to\infty} \frac{\varepsilon_{k+1}}{(\varepsilon_k)^q} = \mu.$$

Two particular values of q are important to us:

(i) If $q = 1$, and we further have that $0 < \mu < 1$, then the rate is *linear*.

(ii) If $q = 2$, the rate is *quadratic* for any $\mu > 0$.

1.2.3 Analysis of the Secant Method

Our next goal is to prove that the *Secant Method* converges. We'll be a little lazy, and only prove a suboptimal linear convergence rate. Then, in our MATLAB class, we'll investigate exactly how rapidly it really converges.

One simple mathematical tool that we use is the *Mean Value Theorem* Theorem 0.2.1 . See also [1, p420].

Theorem 1.2.5. *Suppose that* f *and* f' *are real-valued functions, continuous and defined in an interval* $I = [\tau - h, \tau + h]$ *for some* $h > 0$. *If* $f(\tau) = 0$ *and* $f'(\tau) \neq 0$, *then the sequence* (1.2.1) *converges at least linearly to* τ.

Before we prove this, we note the following

- We wish to show that $|\tau - x_{k+1}| < |\tau - x_k|$.
- From Theorem 0.2.1, there is a point $w_k \in [x_{k-1}, x_k]$ such that
$$\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} = f'(w_k). \qquad (1.2.4)$$
- Also by the MVT, there is a point $z_k \in [x_k, \tau]$ such that
$$\frac{f(x_k) - f(\tau)}{x_k - \tau} = \frac{f(x_k)}{x_k - \tau} = f'(z_k). \qquad (1.2.5)$$
Therefore $f(x_k) = (x_k - \tau)f'(z_k)$.
- Using (1.2.4) and (1.2.5), we can show that
$$\tau - x_{k+1} = (\tau - x_k)\left(1 - f'(z_k)/f'(w_k)\right).$$
Therefore
$$\frac{|\tau - x_{k+1}|}{|\tau - x_k|} \leqslant \left|1 - \frac{f'(z_k)}{f'(w_k)}\right|.$$
- Suppose that $f'(\tau) > 0$. (If $f'(\tau) < 0$ just tweak the arguments accordingly). Saying that f' *is continuous in the region* $[\tau - h, \tau + h]$ means that, for any $\varepsilon > 0$ there is a $\delta > 0$ such that
$$|f'(x) - f'(\tau)| < \varepsilon \text{ for any } x \in [\tau - \delta, \tau + \delta].$$
Take $\varepsilon = f'(\tau)/4$. Then $|f'(x) - f'(\tau)| < f'(\tau)/4$. Thus
$$\frac{3}{4}f'(\tau) \leqslant f'(x) \leqslant \frac{5}{4}f'(\tau) \quad \text{for any } x \in [\tau-\delta, \tau+\delta].$$
Then, so long as w_k and z_k are both in $[\tau-\delta, \tau+\delta]$
$$\frac{f'(z_k)}{f'(w_k)} \leqslant \frac{5}{3}.$$

Take notes:

(See also details in Section 1.2.5).

Given enough time and effort we *could* show that the Secant Method converges faster that linearly. In particular, that the order of convergence is $q = (1 + \sqrt{5})/2 \approx 1.618$. This number arises as the only positive root of $q^2 - q - 1$. It is called the *Golden Mean*, and arises in many areas of Mathematics, including finding an explicit expression for the Fibonacci Sequence: $f_0 = 1$, $f_1 = 1$, $f_{k+1} = f_k + f_{k-1}$ for $k = 2, 3, \ldots$. That gives, $f_0 = 1$, $f_1 = 1$, $f_2 = 2$, $f_3 = 3$, $f_4 = 5$, $f_5 = 8$, $f_6 = 13$, ...

A rigorous proof depends on, among other things, and error bound for polynomial interpolation, which is the first topic in MA378. With that, one can show that $\varepsilon_{k+1} \leqslant C\varepsilon_k\varepsilon_{k-1}$. Repeatedly using this we get:

- Let $r = |x_1 - x_0|$ so that $\varepsilon_0 \leqslant r$ and $\varepsilon_1 \leqslant r$,
- Then $\varepsilon_2 \leqslant C\varepsilon_1\varepsilon_0 \leqslant Cr^2$
- Then $\varepsilon_3 \leqslant C\varepsilon_2\varepsilon_1 \leqslant C(Cr^2)r = C^2r^3$.
- Then $\varepsilon_4 \leqslant C\varepsilon_3\varepsilon_2 \leqslant C(C^2r^3)(Cr^2) = C^4r^5$.
- Then $\varepsilon_5 \leqslant C\varepsilon_4\varepsilon_3 \leqslant C(C^4r^5)(C^2r^3) = C^7r^8$.
- And in general, $\varepsilon_k = C^{f_k-1}r^{f_k}$.

1.2.4 Exercises

Exercise 1.5. ⋆ Suppose we define the Secant Method as follows.

> *Choose any two points* x_0 *and* x_1.
> *For* $k = 1, 2, \ldots$, *set* x_{k+1} *to be the point where the line through* $\big(x_{k-1}, f(x_{k-1})\big)$ *and* $\big(x_k, f(x_k)\big)$ *that intersects the* x*-axis.*

Show how to derive the formula for the secant method.

Exercise 1.6. ⋆

(i) Is it possible to construct a problem for which the bisection method will work, but the secant method will fail? If so, give an example.

(ii) Is it possible to construct a problem for which the secant method will work, but bisection will fail? If so, give an example.

1.2.5 Appendix (Proof of convergence of the secant method)

Here are the full details on the proof of the fact that the Secant Method converges at least linearly (Theorem 1.2.5). Before you read it, take care to review the notes from that section, particularly (1.2.4) and (1.2.5).

Proof. The method is

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}.$$

We'll use this to derive an expression of the error at step $k+1$ in terms of the error at step k. In particular,

$$\begin{aligned}
\tau - x_{k+1} &= \tau - x_k + f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \\
&= \tau - x_k + f(x_k)/f'(w_k) \\
&= \tau - x_k + (x_k - \tau)f'(z_k)/f'(w_k) \\
&= (\tau - x_k)\Big(1 - f'(z_k)/f'(w_k)\Big).
\end{aligned}$$

Therefore

$$\frac{|\tau - x_{k+1}|}{|\tau - x_k|} \leqslant \big|1 - \frac{f'(z_k)}{f'(w_k)}\big|.$$

So it remains to be shown that

$$\big|1 - \frac{f'(z_k)}{f'(w_k)}\big| < 1.$$

Lets first assume that $f'(\tau) = \alpha > 0$. (If $f'(\tau) = \alpha < 0$ the following arguments still hold, just with a few small changes). Because f' is continuous in the region $[\tau - h, \tau + h]$, for any given $\varepsilon > 0$ there is a $\delta > 0$ such that $|f'(x) - \alpha| < \varepsilon$ for and $x \in [\tau - \delta, \tau + \delta]$. Take $\varepsilon = \alpha/4$. Then $|f'(x) - \alpha| < \alpha/4$. Thus

$$\alpha\frac{3}{4} \leqslant f'(x) \leqslant \alpha\frac{5}{4} \quad \text{for any } x \in [\tau - \delta, \tau + \delta].$$

Then, so long as w_k and z_k are both in $[\tau - \delta, \tau + \delta]$

$$\frac{f'(z_k)}{f'(w_k)} \leqslant \frac{5}{3}.$$

This gives

$$\frac{|\tau - x_{k+1}|}{|\tau - x_k|} \leqslant \frac{2}{3},$$

which is what we needed. □

1.3 Newton's Method

1.3.1 Motivation

These notes are loosely based on Section 1.4 of [1] (i.e., Süli and Mayers, Introduction to Numerical Analysis). See also, [3, Lecture 2], and [5, §3.5] The Secant method is often written as

$$x_{k+1} = x_k - f(x_k)\phi(x_k, x_{k-1}),$$

where the function ϕ is chosen so that x_{k+1} is the root of the secant line joining the points $\big(x_{k-1}, f(x_{k-1})\big)$ and $\big(x_k, f(x_k)\big)$. A related idea is to construct a method $x_{k+1} = x_k - f(x_k)\lambda(x_k)$, where we choose λ so that x_{k+1} is the point where the tangent line to f at $(x_k, f(x_k))$ cuts the x-axis. This is shown in Figure 1.3. We attempt to solve $x^2 - 2 = 0$, taking $x_0 = 2$. Taking the x_1 to be zero of the tangent to $f(x)$ at $x = 2$, we get $x_1 = 1.5$. Taking the x_2 to be zero of the tangent to $f(x)$ at $x = 1.5$, we get $x_2 = 1.4167$, which is very close to the true solution of $\tau = 1.4142$.

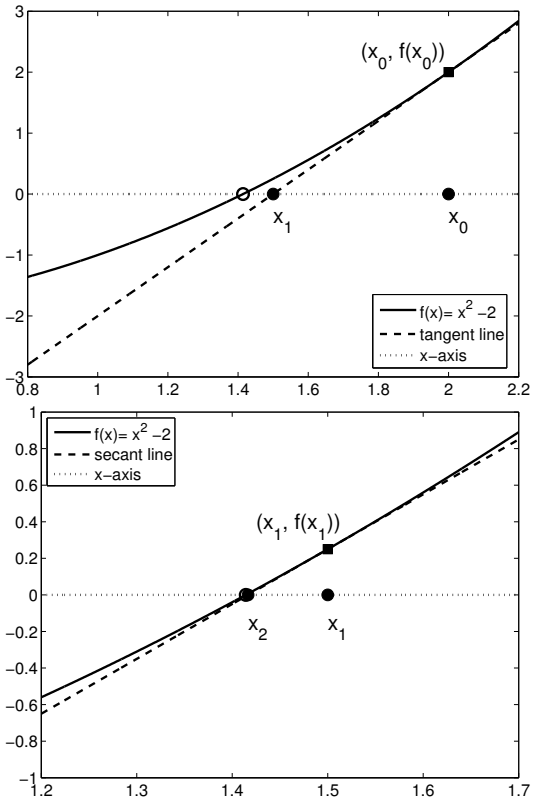

Fig. 1.3: Estimating $\sqrt{2}$ by solving $x^2 - 2 = 0$ using Newton's Method

Method 1.3.1 (Newton's Method[2]).

1. Choose any x_0 in $[a, b]$,
2. For $i = 0, 1, \ldots,$ set x_{k+1} to the root of the line through x_k with slope $f'(x_k)$.

By writing down the equation for the line at $\big(x_k, f(x_k)\big)$ with slope $f'(x_k)$, one can show (see Exercise 1.7-(i)) that the formula for the iteration is

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{1.3.6}$$

Example 1.3.2. *Use Newton's Method to solve the nonlinear problem* $x^2 - 2 = 0$ *in* $[0, 2]$. The results are shown in Table 1.3. For this example, the method becomes

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - 2}{2x_k} = \frac{1}{2}x_k + \frac{1}{x_k}.$$

k	x_k	$\lvert x_k - \tau\rvert$	$\lvert x_k - x_{k-1}\rvert$
0	2.000000	5.86e-01	
1	1.500000	8.58e-02	5.00e-01
2	1.416667	2.45e-03	8.33e-02
3	1.414216	2.12e-06	2.45e-03
4	1.414214	1.59e-12	2.12e-06
5	1.414214	2.34e-16	1.59e-12

Table 1.3: Solving $x^2 - 2 = 0$ using Newton's Method

By comparing Table 1.2 and Table 1.3, we see that for this example, the Newton's method is more efficient again than the Secant method.

Deriving Newton's method geometrically certainly has an intuitive appeal. However, to analyse the method, we need a more abstract derivation based on a **Truncated Taylor Series**.

Take notes:

[2] 

Sir Isaac Newton, 1643 - 1727, England. Easily one of the greatest scientist of all time. The method we are studying appeared in his celebrated *Principia Mathematica* in 1687, but it is believed he had used it as early as 1669.

1.3.2 Newton Error Formula

We saw in Table 1.3 that Newton's method can be much more efficient than, say, Bisection: it yields estimates that converge far more quickly to τ. Bisection converges

(at least) linearly, whereas Newton's converges *quadratically*, that is, with *at least order* $q = 2$.

In order to prove that this is so, we need to

1. write down a recursive formula for the error;
2. show that it converges;
3. then find the limit of $|\tau - x_{k+1}|/|\tau - x_k|^2$.

Step 2 is usually the crucial part.

There are two parts to the proof. The first involves deriving the so-called "Newton Error formula". Then we'll apply this to prove (quadratic) convergence. In all cases we'll assume that the functions f, f' and f'' are defined and continuous on the an interval $I_\delta = [\tau-\delta, \tau+\delta]$ around the root τ. The proof we'll do in class comes directly from the above derivation (see also Epperson [5, Thm 3.2]).

Theorem 1.3.3 (Newton Error Formula)**.** *If* $f(\tau) = 0$ *and*

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},$$

then there is a point η_k *between* τ *and* x_k *such that*

$$\tau - x_{k+1} = -\frac{(\tau - x_k)^2}{2}\frac{f''(\eta_k)}{f'(x_k)},$$

Take notes:

Example 1.3.4. As an application of Newton's error formula, we'll show that the number of correct decimal digits in the approximation doubles at each step.

Take notes:

1.3.3 Convergence of Newton's Method

We'll now complete our analysis of this section by proving the convergence of Newton's method.

Theorem 1.3.5. *Let us suppose that* f *is a function such that*

- f *is continuous and real-valued, with continuous* f''*, defined on some close interval* $I_\delta = [\tau-\delta, \tau+\delta]$,
- $f(\tau) = 0$ *and* $f''(\tau) \neq 0$,
- *there is some positive constant* A *such that*

$$\frac{|f''(x)|}{|f'(y)|} \leqslant A \quad \textit{for all} \quad x, y \in I_\delta.$$

Let $h = \min\{\delta, 1/A\}$. *If* $|\tau - x_0| \leqslant h$ *then Newton's Method converges quadratically.*

Take notes:

1.3.4 Exercises

Exercise 1.7. ⋆ Write down the equation of the line that is tangential to the function f at the point x_k. Give an expression for its zero. Hence show how to derive Newton's method.

Exercise 1.8. (i) It is possible to construct a problem for which the bisection method will work, but Newton's method will fail? If so, give an example.

(ii) It is possible to construct a problem for which Newton's method will work, but bisection will fail? If so, give an example.

Exercise 1.9. (i) Write down Newton's Method as applied to the function $f(x) = x^3 - 2$. Simplify the computation as much as possible. What is achieved if we find the root of this function?

(ii) Do three iterations by hand of Newton's Method applied to $f(x) = x^3 - 2$ with $x_0 = 1$.

Exercise 1.10. (This is taken from Exercise 3.5.1 of Epperson). If f is such that $|f''(x)| \leqslant 3$ and $|f'(x)| \geqslant 1$ for all x, and if the initial error in Newton's Method is less than $1/2$, give an upper bound for the error at each of the first 3 steps.

Exercise 1.11. Here is (yet) another scheme called *Steffenson's Method*: Choose $x_0 \in [a, b]$ and set

$$x_{k+1} = x_k - \frac{\big(f(x_k)\big)^2}{f\big(x_k + f(x_k)\big) - f(x_k)} \text{ for } k = 0, 1, 2, \ldots.$$

(a) ⋆ Explain how this method relates to Newton's Method.

(b) [Optional] Write a program, in MATLAB, or your language of choice, to implement this method. Verify it works by using it to estimate the solution to $e^x = (2 - x)^3$ with $x_0 = 0$. Submit your code and test harness as Blackboard assignment. *No credit is available for this part, but feedback will be given on your code. Also, it will help you prepare for the final exam.*

Exercise 1.12. ⋆ (This is Exercise 1.6 from Süli and Mayers) The proof of the convergence of Newton's method given in Theorem 1.3.5 uses that $f'(\tau) \neq 0$. Suppose that it is the case that $f'(\tau) = 0$.

(i) What can we say about the root, τ?

(ii) Starting from the Newton Error formula, show that

$$\tau - x_{k+1} = \frac{(\tau - x_k)}{2} \frac{f''(\eta_k)}{f''(\mu_k)},$$

for some μ_k between τ and x_k. (*Hint: try using the MVT*).

(iii) What does the above error formula tell us about the convergence of Newton's method in this case?

1.4 Fixed Point Iteration

1.4.1 Introduction

Newton's method can be considered to be a particular instance of a very general approach called *Fixed Point Iteration* or *Simple Iteration*.

The basic idea is:

> *If we want to solve* $f(x) = 0$ *in* $[a, b]$*, find a function* $g(x)$ *such that, if* τ *is such that* $f(\tau) = 0$*, then* $g(\tau) = \tau$*.*
> *Next, choose* x_0 *and set* $x_{k+1} = g(x_k)$ *for* $k = 0, 1, 2 \ldots$.

Example 1.4.1. Suppose that $f(x) = e^x - 2x - 1$ and we are trying to find a solution to $f(x) = 0$ in $[1, 2]$. We can reformulate this problem as

> *For* $g(x) = \ln(2x + 1)$*, find* $\tau \in [1, 2]$ *such that* $g(\tau) = \tau$.

If we take the initial estimate $x_0 = 1$, then Simple Iteration gives the following sequence of estimates.

k	x_k	$\lvert\tau - x_k\rvert$
0	1.0000	2.564e-1
1	1.0986	1.578e-1
2	1.1623	9.415e-2
3	1.2013	5.509e-2
4	1.2246	3.187e-2
5	1.2381	1.831e-2
⋮	⋮	⋮
10	1.2558	6.310e-4

To make this table, I used a numerical scheme to solve the problem quite accurately to get $\tau = 1.256431$. (In general we don't know τ in advance–otherwise we wouldn't need such a scheme). I've given the quantities $|\tau - x_k|$ here so we can observe that the method is converging, and get an idea of how quickly it is converging.

We have to be quite careful with this method: **not every choice is g is suitable**.

Suppose we want the solution to $f(x) = x^2 - 2 = 0$ in $[1, 2]$. We could choose $g(x) = x^2 + x - 2$. Taking $x_0 = 1$ we get the iterations shown opposite.

k	x_k
0	1
1	0
2	-2
3	0
4	-2
5	0
⋮	⋮

This sequence doesn't converge!

We need to refine the method that ensure that it will converge. Before we do that in a formal way, consider the following...

Example 1.4.2. Use the Mean Value Theorem to show that the fixed point method $x_{k+1} = g(x_k)$ converges if $|g'(x)| < 1$ for all x near the fixed point.

Take notes:

This is an important example, mostly because it introduces the "tricks" of using that $g(\tau) = \tau$ and $g(x_k) = x_{k+1}$. But it is not a rigorous theory. That requires some ideas such as the *contraction mapping theorem*.

1.4.2 A short tour of fixed points and contractions

A variant of the famous Fixed Point Theorem[3] is :

> *Suppose that* $g(x)$ *is defined and continuous on* $[a, b]$*, and that* $g(x) \in [a, b]$ *for all* $x \in [a, b]$*. Then there exists a point* $\tau \in [a, b]$ *such that* $g(\tau) = \tau$*. That is,* $g(x)$ *has a* fixed point *in the interval* $[a, b]$.

Try to convince yourself that it is true, by sketching the graphs of a few functions that send all points in the interval, say, $[1, 2]$ to that interval, as in Figure 1.4.

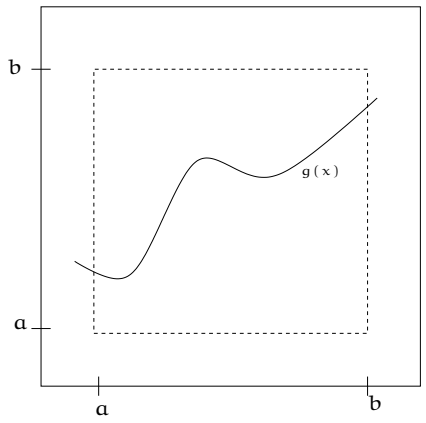

Fig. 1.4: Sketch of a function $g(x)$ such that, if $a \leqslant x \leqslant b$ then $a \leqslant g(x) \leqslant b$

The next ingredient we need is to observe that g is a *contraction*. That is, $g(x)$ is continuous and defined on $[a, b]$ and there is a number $L \in (0, 1)$ such that

$$|g(\alpha) - g(\beta)| \leqslant L|\alpha - \beta| \text{ for all } \alpha, \beta \in [a, b]. \quad (1.4.7)$$

[3] LEJ Brouwer, 1881–1966, Netherlands

Theorem 1.4.3 (Contraction Mapping Theorem)**.**
Suppose that the function g *is a real-valued, defined, continuous, and*

(a) it maps every point in $[a, b]$ *to some point in* $[a, b]$*;*

(b) and it is a contraction on $[a, b]$*,*

then

(i) g *has a fixed point* $\tau \in [a, b]$*,*

(ii) the fixed point is unique,

(iii) the sequence $\{x_k\}_{k=0}^{\infty}$ *defined by* $x_0 \in [a, b]$ *and* $x_k = g(x_{k-1})$ *for* $k = 1, 2, \ldots$ *converges to* τ*.*

Proof:
Take notes:

1.4.3 Convergence of Fixed Point Iteration

We now know how to apply to Fixed-Point Method and to check if it will converge. Of course we can't perform an infinite number of iterations, and so the method will yield only an approximate solution. Suppose we want the solution to be accurate to say 10^{-6}, how many steps are needed? That is, how large must k be so that

$$|x_k - \tau| \leqslant 10^{-6}?$$

The answer is obtained by first showing that

$$|\tau - x_k| \leqslant \frac{L^k}{1 - L}|x_1 - x_0|. \tag{1.4.8}$$

Take notes:

Example 1.4.4. If $g(x) = \ln(2x + 1)$ and $x_0 = 1$, and we want $|x_k - \tau| \leqslant 10^{-6}$, then we can use (1.4.8) to determine the number of iterations required.

Take notes:

This calculation only gives an upper bound for the number of iterations. It is correct, but not necessarily *sharp*. In practice, one finds that 23 iterations is sufficient to ensure that the error is less than 10^{-6}. Even so, 23 iterations a quite a lot for such a simple problem. So can conclude that this method is not as fast as, say, Newton's Method. However, it is perhaps the most generalizable.

1.4.4 Knowing When to Stop

Suppose you wish to program one of the above methods. You will get your computer to repeat one of the iterative methods until your solution is sufficiently close to the true solution:

```
x[0] = 0
tol = 1e-6
i=0
while (abs(tau - x[i]) > tol)  // This is the
                               // stopping criterion
  x[i+1] = g(x[i])   // Fixed point iteration
  i = i+1
end
```

All very well, except you don't know τ. If you did, you wouldn't need a numerical method. Instead, we could choose the stopping criterion based on how close successive estimates are:

```
while (abs(x[i-1] - x[i]) > tol)
```

This is fine if the solution is not close to zero. E.g., if its about 1, would should get roughly 6 accurate figures. But is $\tau = 10^{-7}$ then it is quite useless: x_k could be ten times larger than τ. The problem is that we are estimating the *absolute* error.

Instead, we usually work with *relative error*:

```
while (abs (x[i-1]-x[i] / x[i] ) > tol)
```

1.4.5 Exercises

Exercise 1.13. Is it possible for g to be a contraction on $[a, b]$ but not have a fixed point in $[a, b]$? Give an example to support your answer.

Exercise 1.14. Show that $g(x) = \ln(2x + 1)$ is a contraction on $[1, 2]$. Give an estimate for L. (Hint: Use the Mean Value Theorem).

Exercise 1.15. Consider the function $g(x) = x^2/4 + 5x/4 - 1/2$.

(i) It has two fixed points – what are they?

(ii) For each of these, find the largest region around them such that g is a contraction on that region.

Exercise 1.16. Although we didn't prove it in class, it turns out that, if $g(\tau) = \tau$, and the fixed point method given by

$$x_{k+1} = g(x_k),$$

converges to the point τ (where $g(\tau) = \tau$), and

$$g'(\tau) = g''(\tau) = \cdots = g^{(p-1)}(\tau) = 0,$$

then it converges with order p.

(i) Use a Taylor Series expansion to prove this.

(ii) We can think of Newton's Method for the problem $f(x) = 0$ as fixed point iteration with $g(x) = x - f(x)/f'(x)$. Use this, and Part (i), to show that, if Newton's method converges, it does so with order 2, providing that $f'(\tau) \neq 0$.

1.5 LAB 1: the bisection and secant methods

The goal of this section is to help you gain familiarity with the fundamental tasks that can be accomplished with MATLAB: defining vectors, computing functions, and plotting. We'll then see how to implement and analyse the Bisection and Secant schemes in MATLAB.

You'll find many good MATLAB references online. I particularly recommend:

- Cleve Moler, *Numerical Computing with MATLAB*, which you can access at `http://uk.mathworks.com/moler/chapters`
- Tobin Driscoll, *Learning MATLAB*, which you can access through the NUI Galway library portal.

MATLAB is an interactive environment for mathematical and scientific computing. It the standard tool for numerical computing in industry and research.

MATLAB stands for Matrix Laboratory. It specialises in matrix and vector computations, but includes functions for graphics, numerical integration and differentiation, solving differential equations, etc.

MATLAB differs from most significantly from, say, Maple, by not having a facility for abstract computation.

1.5.1 The Basics

MATLAB an *interpretive* environment – you type a command and it will execute it immediately.

The default data-type is a matrix of double precision floating-point numbers. A scalar variable is an instance of a 1×1 matrix. To check this set,
`>> t=10` and use `>> size(t)`
to find the numbers of rows and columns of `t`.

A vector may be declared as follows:
`>> x = [1 2 3 4 5 6 7]`
This generates a vector, x, with $x_1 = 1$, $x_2 = 2$, etc. However, this could also be done with `x=1:7`

More generally, if we want to define a vector $x = (a, a+h, a+2h, \ldots, b)$, we could use `x = a:h:b;`
For example
`>> x=10:-2:0` gives $x = (10, 8, 6, 4, 2, 0)$.
If h is omitted, it is assumed to be 1.

The i^{th} element of a vector is access by typing `x(i)`. The element of in row `i` and column `j` of a matrix is given by `A(i,j)`

Most "scalar" functions return a matrix when given a matrix as an argument. For example, if x is a vector of length n, then `y = sin(x)` sets y to be a vector, also of length n, with $y_i = \sin(x_i)$.

MATLAB has most of the standard mathematical functions: `sin`, `cos`, `exp`, `log`, etc.
In each case, write the function name followed by the argument in round brackets, e.g.,
`>> exp(x)` for e^x.

The `*` operator performs matrix multiplication. For element-by-element multiplication use `.*`
For example,
`y = x.*x` sets $y_i = (x_i)^2$.
So does `y = x.^2`. Similarly, `y=1./x` set $y_i = 1/x_i$.

If you put a semicolon at the end of a line of MATLAB, the line is executed, but the output is not shown. (This is useful if you are dealing with large vectors). If no semicolon is used, the output is shown in the command window.

1.5.2 Plotting functions

Define a vector
`>> x=[0 1 2 3]` and then set `>> f = x.^2 -2`
To plot these vectors use:
`>> plot(x, f)`
If the picture isn't particularly impressive, then this might be because Matlab is actually only printing the 4 points that you defined. To make this more clear, use
`>> plot(x, f, '-o')`
This means to plot the vector `f` as a function of the vector x, placing a circle at each point, and joining adjacent points with a straight line.

Try instead: `>> x=0:0.1:3` and `f = x.^2 -2` and plot them again.

To define function in terms of *any* variable, type:
```
>> F = @(x)(x.^2 -2);
```
Now you can use this *function* as follows:
```
>> plot(x, F(x));
```
Take care to note that MATLAB is *case sensitive*.

In this last case, it might be helpful to also observe where the function cuts the x-axis. That can be done by also plotting the line joining, for example, the points $(0, 0)$, and $(3, 0)$:
```
>> plot(x,F(x), [0,3], [0,0]);
```
Tip: *Use the* `>> help` *menu to find out what the* `ezplot` *function is, and how to use it.*

1.5.3 Programming the Bisection Method

Revise the lecture notes on the *Bisection Method*.
Suppose we want to find a solution to $e^x - (2-x)^3 = 0$ in the interval $[0, 5]$ using Bisection.

- Define the function `f` as:
  ```
  >> f = @(x)(exp(x) - (2-x).^3);
  ```
- Taking $x_1 = 0$ and $x_2 = 5$, do 8 iterations of the Bisection method.
- Complete the table below. You may use that the solution is (approximately)
  $\tau = 0.7261444658054950$.

k	x_k	$\lvert\tau - x_k\rvert$
1		
2		
3		
4		
5		
6		
7		
8		

Implementing the Bisection method by hand is very tedious. Here is a program that will do it for you. You don't need to type it all in; you can download it from www.maths.nuigalway.ie/MA385/lab1/Bisection.m

```
3  clear; % Erase all stored variables
4  fprintf('\n\n---------\n Using Bisection\n');
5  % The function is
6  f = @(x)(exp(x) - (2-x).^3);
7  fprintf('Solving f=0 with the function\n');
8  disp(f);
9
10
11 tau = 0.72614446580549503614; % true solution
12 fprintf('The true solution is %12.8f\n', tau);
13
14 %% Our initial guesses are x_1=0 and x_2 =2;
15 x(1)=0;
16 fprintf('%2d | %14.8e | %9.3e \n', ...
17    1, x(1), abs(tau - x(1)));
18 x(2)=5;
19 fprintf('%2d | %14.8e | %9.3e \n', ...
20    2, x(2), abs(tau - x(2)));
21 for k=2:8
22    x(k+1) = (x(k-1)+x(k))/2;
23    if ( f(x(k+1))*f(x(k-1)) < 0)
24       x(k)=x(k-1);
25    end
26    fprintf('%2d | %14.8e | %9.3e\n', ...
27       k+1, x(k+1), abs(tau - x(k+1)));
28 end
```

Read the code carefully. If there is a line you do not understand, then ask a tutor, or look up the on-line help. For example, find out what that `clear` on Line 3 does by typing `>> doc clear`

Q1. Suppose we wanted an estimate x_k for τ so that $|\tau - x_k| \leqslant 10^{-10}$.

(i) In §1.1 we saw that $|\tau - x_k| \leqslant (\frac{1}{2})^{k-1}|b - a|$. Use this to estimate how many iterations are required in theory.

(ii) Use the program above to find how many iterations are required in practice.

1.5.4 The Secant method

Recall the the Secant Method in (1.2.1).

Q2 (a) Adapt the program above to implement the secant method.

(b) Use it to find a solution to $e^x - (2-x)^3 = 0$ in the interval $[0, 5]$.

(c) How many iterations are required to ensure that the error is less than 10^{-10}?

. . . . . .

Q3 Recall from Definition 1.2.4 the *order of convergence* of a sequence $\{\varepsilon_0, \varepsilon_1, \varepsilon_2, \ldots\}$ is q if

$$\lim_{k\to\infty} \frac{\varepsilon_{k+1}}{\varepsilon_k^q} = \mu,$$

for some constant μ.

We would like to verify that $q = (1+\sqrt{5})/2 \approx 1.618$. This is difficult to do computationally because, after a relatively small number of iterations, the round-off error becomes significant. But we can still try!

Adapt the program above so that at each iteration it displays

$$\frac{|\tau - x_{k+1}|}{|\tau - x_k|}, \quad \frac{|\tau - x_{k+1}|}{|\tau - x_k|^{1.618}}, \quad \frac{|\tau - x_{k+1}|}{|\tau - x_k|^2},$$

and so deduce that the order of converges is greater than 1 (so better than bisection), less than 2, and roughly $(1+\sqrt{5})/2$.

1.5.5 To Finish

Before you leave the class upload your MATLAB code for the Secant Method (Q2 an Q3) to "**Lab 1**" in the "Assignments and Labs" section Blackboard. This file must include your name and ID number and comments. Ideally, it should incorporate your name or ID into the file name (e.g., `Lab1_Donal_Duck.m`). Include your answer to Q1 and Q2 as text.

1.5.6 Extra

The bisection method is popular because it is robust: it will always work subject to minimal constraints. However, it is slow: if the Secant works, then it converges much more quickly. How can we combine these two algorithms to get a fast, robust method? Consider the following problem:

$$\text{Solve } 1 - \frac{2}{x^2 - 2x + 2} = 0 \quad \text{on } [-10, 1].$$

You should find that the bisection method works (slowly) for this problem, but the Secant method will fail. So write a hybrid algorithm that switches between the bisection method and the secant method as appropriate.

Take care to document your code carefully, to show which algorithm is used when.

How many iterations are required?