Contents

Numerical Analysis summary
Course with Dr. Quxuan Wang - Winter 2022

Prepared by: Khoi Vo

August 27, 2022

1 Numerical Solutions to Nonlinear Equations

1.1 Convergence: linear and with order p

Definition 1. A nonlinear equation $F(x) = 0$ where $F; \mathbb{R}^d \to \mathbb{R}^d$: $F = (f_1, f_2, \ldots, f_d)$ with:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_d) &= 0 \\ f_2(x_1, x_2, \ldots, x_d) &= 0 \\ &\ldots \\ f_d(x_1, x_2, \ldots, x_d) &= 0 \end{aligned}$$

And each f_i is nonlinear.

Remark 2.

1. It is hard to solve a nonlinear equation.
2. Nonlinear equations may have no solutions, for example:
$$f(x) = x^2 - 4x + 5$$
3. Solutions may not be unique
4. **Most nonlinear CANNOT** be solved analytically.

Question 3. How do we solve numerically?

The Method: **iterative Method**

Definition 4 (Iterative Method)**.** Set $F(\overrightarrow{x}) = 0$ where $F : \mathbb{R}^d \to \mathbb{R}^d$. We find a sequence $\{x_n\}_{n\in\mathbb{N}}$ such that $x_n \to x^*$ and that $F(x^*) = 0$.

Remark 5. The challenge is that does x_n converges **fast enough**?

Definition 6. Consider the little o and big O notation:

1. Little o notation:
$$f(h) = o(g(h)) \text{ if } \frac{|f(h)|}{|g(h)|} \to 0 \text{ as } h \to 0$$

2. Big O notation:

$$f(h) = O(g(h)) \text{ if there exists } \varepsilon, C > 0 : |f(h)| \leq C|g(h)|, \text{ for all } 0 < h < \varepsilon$$

3. For sequences $\{\alpha_n\}_{n=1}^{\infty}$ and $\{\beta\}_{n=1}^{\infty}$, we write $\alpha_n = o(\beta_n)$ if $\frac{|\alpha_n|}{|\beta_n|} \to 0$ as $n \to \infty$.

4. For sequence $\{\alpha_n\}_{n=1}^{\infty}$ and $\{\beta\}_{n=1}^{\infty}$, we write $\alpha_n = O(\beta_n)$ if there exists $N, C > 0$ such that $|\alpha_n| \leq C|\beta_n|$ for all $n > N$.

Remark 7.

1. Big O means "is of the same order as"

2. small o means "is ultimately smaller than"/converges faster than.

Definition 8 (Linear Convergence)**.** Suppose a sequence $\{x_n\}_{n=1}^{\infty}$ that converges to x^*, i.e., $\lim_{n\to\infty} x_n = x^*$, if there exists a $p \in [0, 1]$ such that:

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|} = p$$

then

1. $\{x_n\}_{n=1}^{\infty}$ converges to x^* **linearly** with rate p for $p \in (0, 1)$

2. $\{x_n\}_{n=1}^{\infty}$ converges to x^* **superlinearly** when $p = 0$

3. $\{x_n\}_{n=1}^{\infty}$ converges to x^* **sublinearly** when $p = 1$

Example 9. For the following sequence:

1. $x_n = 1 + 2^{2-n}$: linear

2. $x_{n+1} = 2^{-n} x_n$: superlinear

3. $x_n = \frac{1}{n}$: sublinear

   To see this, we check the following:

   (a) Take the limit $x_n = 1 + 2^{2-n} \xrightarrow{n\to\infty} 1 = x^*$. Therefore:

   $$\lim_{n\to\infty} \frac{x_{n+1} - x^*}{x_n - x^*} = \lim_{n\to\infty} \frac{2^{2-(n+1)}}{2^{2-n}} = \frac{1}{2}$$

   So converges to 1 linearly with rate $\frac{1}{2}$.

   (b) Take the limit $x_{n+1} = \frac{x_n}{2^n} \xrightarrow{n\to\infty} 0 = x^*$. Then

   $$\lim_{n\to\infty} \frac{x_{n+1} - x^*}{x_n - x^*} = \lim_{n\to\infty} \frac{\frac{|x_n|}{2^n}}{|x_n|} = \lim_{n\to\infty} \frac{1}{2^n} = 0$$

   Therefore it converges superlinear.

(c) Take limit $x_n = \frac{1}{n} \xrightarrow{n\to\infty} 0 = x^*$. Then:

$$\lim_{n\to\infty} \frac{x_{n+1} - x^*}{x_n - x^*} = \lim_{n\to\infty} \frac{\frac{1}{n+1}}{\frac{1}{n}} = 1$$

Therefore it converges sublinear.

Definition 10 (Convergence with Order p)**.** If there exists a $p > 1$ and a $C > 1$ such that:

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^p} = C$$

Then $\{x_n\}_{n=1}^{\infty}$ is said to converges to x^* **with order** p.

Remark 11. If $\{x_n\}_{n=1}^{\infty}$ converges to x^* with order p, then it converges to x^* with order q for all $0 < q < p$.

Example 12. Consider the following sequence:

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

where $a > 0$. Show the following:

1. Suppose that the sequence converges, find its limit.
2. Show that the convergence order of $\{x_n\}_{n=1}^{\infty}$ is 2.

The solution is as followed:

1. Suppose the limit is L, then:

$$\begin{aligned} L &= \frac{1}{2}(L + \frac{a}{L}) \\ 2L &= L + \frac{a}{L} \\ L &= \frac{a}{L} \\ L &= \sqrt{a} \end{aligned}$$

2. For the convergence rate:

$$\begin{aligned} \frac{|x_{n+1} - \sqrt{a}|}{|x_n - \sqrt{a}|^2} &= \frac{\frac{1}{2}(x_n + \frac{a}{x_n}) - \sqrt{a}}{|x_n - \sqrt{a}|^2} = \frac{1}{2}\frac{(\sqrt{x_n} - \frac{\sqrt{a}}{\sqrt{x_n}})^2}{|x_n - \sqrt{a}|^2} \\ &= \frac{1}{2x_n}\frac{|x_n - \sqrt{a}|^2}{|x_n - \sqrt{a}|^2} = \frac{1}{2x_n} \xrightarrow{n\to\infty} \frac{1}{2\sqrt{a}} = c > 0 \end{aligned}$$

Then x_n converges to $\sqrt{a}$ with order 2.

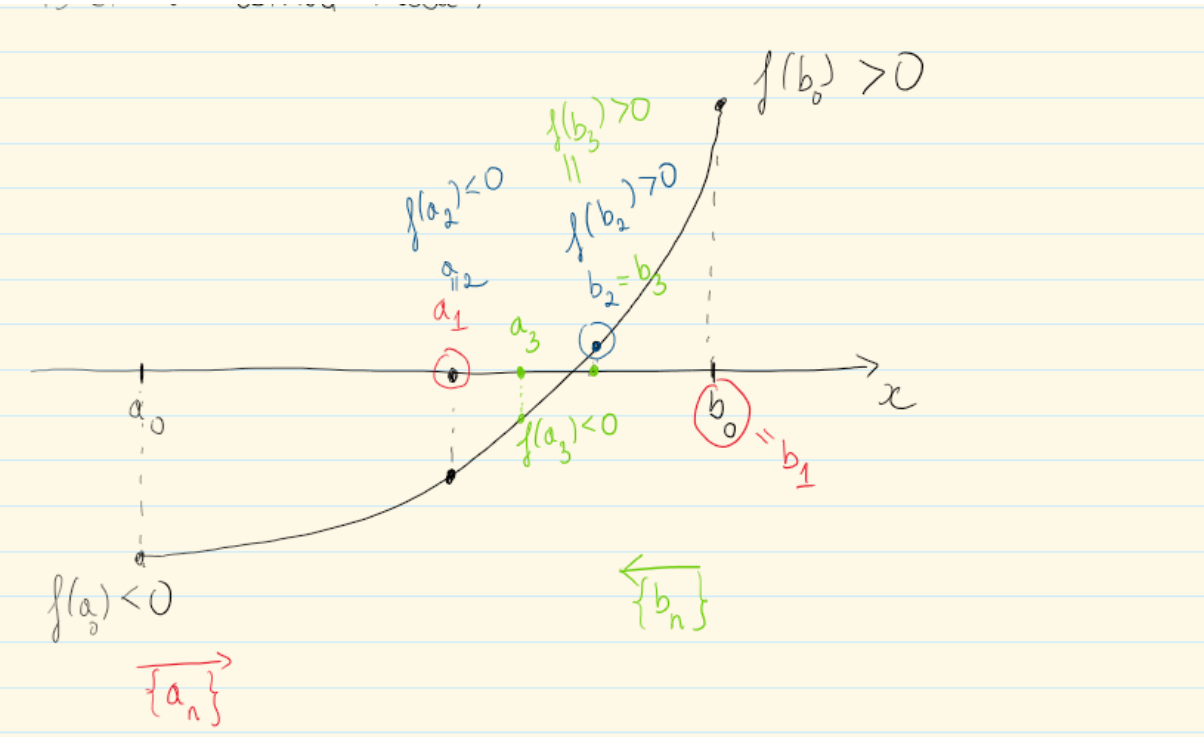


Figure 1: Bisection Method

1.2 Bisection Method

The key idea of this section is based on this theorem:

Theorem 13 (Intermediate value Theorem (IVT)). *If $f \in C[a,b]$, and K is between $f(a)$ and $f(b)$, then there exists a $p \in [a,b]$ such that $f(p) = K$.*

Corollary 14. *If $f \in C[a,b]$ and that $f(a)f(b) < 0$, then there exists $p \in [a,b]$ such that $f(p) = 0$.*

Question 15. How do we find that p?

Bisection method Idea: We "bisect" the interval $[a,b]$ into intervals $[a,b] = [a_0,b_0] \supseteq [a_1,b_1] \supseteq [a_3,b_b] \supseteq \ldots$. At each step, either $a_{n+1} = a_n + \frac{b_n - a_n}{2}$ or $b_{n+1} = b_n + \frac{b_n - a_n}{2}$. This is depends on the value of $f(\frac{b_n-a_n}{2})$. WLOG, say $f(a_0) < 0$ and $f(b_0) > 0$. If $f(\frac{b_0-a_0}{2}) > 0$, then we set $b_1 = \frac{b_0-a_0}{2}$ and keep $a_1 = a_0$. If $f(\frac{b_0-a_0}{2}) < 0$, then we set $a_1 = \frac{b_0-a_0}{2}$ and keep $b_1 = b_0$. Continue in this manner. Basically, the a_n are moving to the right, and the b_n are moving to the left. So the length of $|b_{n+1} - a_{n+1}| = \frac{1}{2}|b_n - a_n|$. Note that

$$\bigcap_n [a_n, b_n] = x^*$$

where $f(x^*) = 0$. See figure (1) for visual and figure (2) for pseudocode.

Remark 16. Note that the pseudocode of Bisection Method already assumed $f(a)f(b) < 0$ and it only changes the left the value $f(a)$, it checked $f(a)f(p) > 0$.

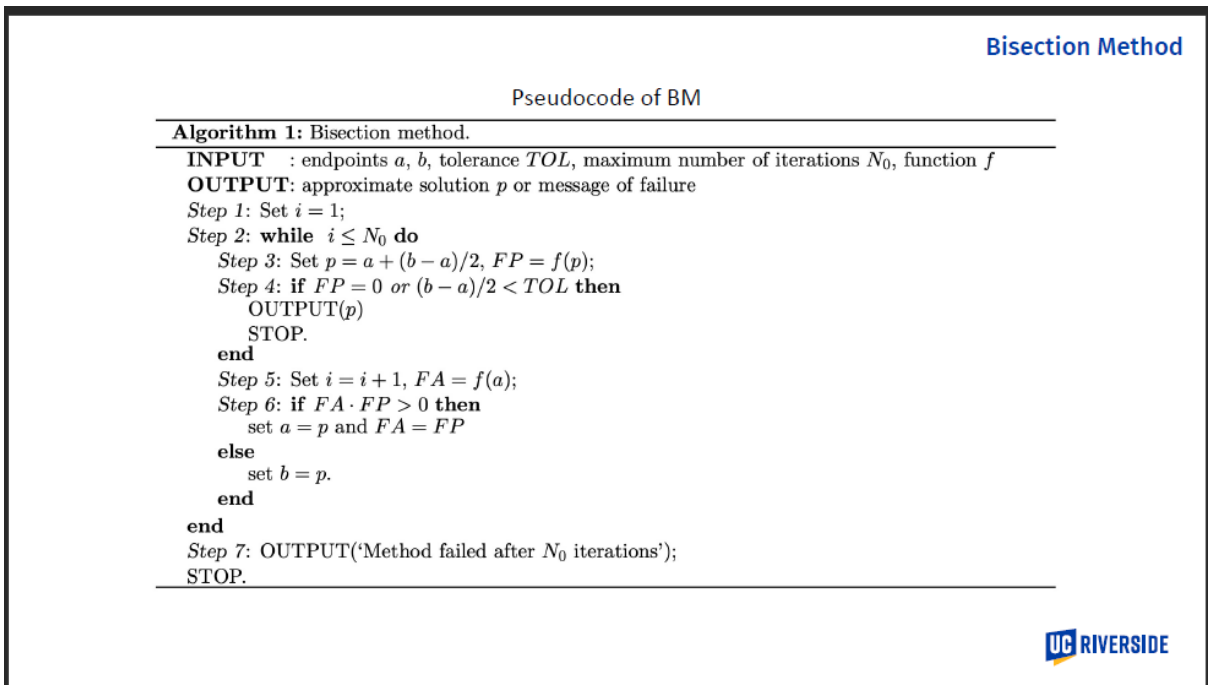


Figure 2: Bisection Method pseudocode

2 Day 2: Bisection Method, Fixed Point Iteration and Newton's Method

2.1 Error in Bisection Method

Since doing bisection method (or any numerical method), we are interested in **error**.

Definition 17 (Error of the Bisection Method)**.** Consider a sequence $\{x_n\}_{n=1}^{\infty}$ that converges to x^*, i.e., $\lim_{n\to\infty} x_n = x^*$.

1. The **absolute error** between the true value (the solution) x^* and the approximate value x_k is given by:
$$|x_k - x^*|$$
2. If $x^* \neq 0$, then the **relative error** of x_k is given by:
$$\frac{|x_k - x^*|}{|x^*|}$$

Theorem 18. *If $f \in C[a,b]$ and $f(a)f(b) < 0$. The bisection algorithm generates $[a_n, b_n]_{n=1}^{\infty}$, and the associated end points:*

$$x_n = \frac{a_n + b_n}{2}$$

The sequences $\{a_n\}_{n=1}^{\infty}$, $\{b_n\}_{n=1}^{\infty}$, and $\{x_n\}_{n=1}^{\infty}$ will all converge to a solution x^* of the equation $f(x^*) = 0$, and the following error estimate holds for the k-th approximate value x_k:

$$|x_k - x^*| \leq \frac{1}{2}(b_k - a_k) = 2^{-(k+1)}(b-a)$$

Proof. Recall the sequence of intervals:

$$[a,b] = [a_0,b_0] \supseteq [a_1,b_1] \supseteq [a_3,b_b] \supseteq \dots [a_n,b_n] \supseteq \dots$$

Consider the $\{a_n\}$, this is an increasing monotonically sequence, and has an upper bound b_0. Therefore $\{a_n\}$ converges. Denote $a^* = \lim_{n\to\infty} a_n$.
Similarly, the sequence b_n decrease monotonically and has a lower bound a_0, so denote $b^* = \lim_{n\to\infty} b_n$. Then:

$$|a^* - b^*| \leq \lim_{n\to\infty} |a_n - b_n| = \lim_{n\to\infty} \frac{1}{2}|a_{n+1} - b_{n+1}| = \dots = \lim_{n\to\infty} \frac{|a_0 - b_0|}{2^{n+1}} = 0$$

Therefore $a^* = b^*$. Denote $x^* = a^* = b^*$. Now note that:

$$a_n \leq x_n \leq b_n$$

Taking squeeze theorem with $\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n = x^*$, therefore $\lim_{n\to\infty} x_n = x^*$.
By Bisection method algorithm, we always have $f(a_n)f(b_n) < 0$, since f is continuous, we have:

$$\lim_{n\to\infty} f(a_n)f(b_n) = [f(x^*)]^2 \leq 0$$

Thus $f(x^*) = 0$. Finally, since $x^* \in \bigcap_{n=0}^{\infty}[a_n,b_n]$ and $x_n \in [a_n,b_n] \cap [a_{n+1},b_{n+1}]$:

$$|x_n - x^*| \leq |a_{n+1} - b_{n+1}| = \frac{1}{2}|a_n - b_n| = \dots = \frac{1}{2^{n+1}}|a_n - b_n|$$

So we get the bound proof as needed. □

2.2 Fixed-Point Iteration

Definition 19 (fixed point problem)**.** The number x^* is a fixed point for a given function g if $g(x^*) = x^*$. The problem of finding x^* is such that $g(x^*) = x^*$ is called the **fixed point problem**.

Example 20. Consider $g(x) = x^2 - 2$. Then $g(-1) = -1$, and $g(2) = 2$, so $-1, 2$ are fixed point of g.

Remark 21. The **root finding problem** is equivalent to **a fixed point problem** of $g(x)$, that is:

$$f(x) = 0 \iff g(x) = x + \alpha f(x)$$

Theorem 22 (Existence and Uniqueness of a fixed point)**.**

1. *If $g(x) \in C[a,b]$ and $g(x) \in [a,b]$ for any $x \in [a,b]$ (g is called a **contraction mapping on** $[a,b]$), then there exists a $x^* \in [a,b]$ such that $g(x^*) = x^*$.*
2. *If, in addition, $g \in C[a,b]$ and there exists $k \in (0,1)$ such that $|g'(x)| \leq k < 1$ for any $x \in (a,b)$, then the fixed point $x^* \in [a,b]$ is unique.*

Proof. First let's look at figure (3) for reference.

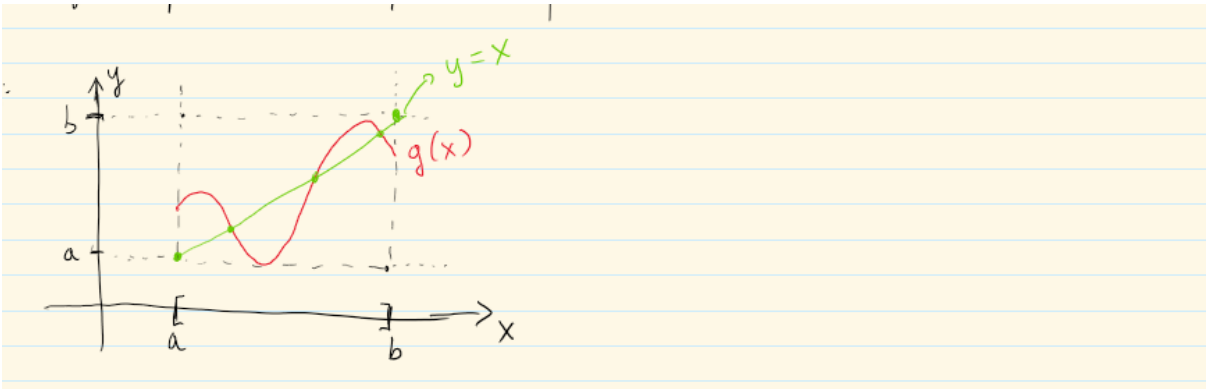


Figure 3: Fixed point theorem proof

1. For showing existence: With $g(x) \in C[a,b]$, and $g([a,b]) \subset [a,b]$, show there exists $x^* \in [a,b]$ such that $g(x^*) = x^*$.
   If $g(a) = a$ or $g(b) = b$ we found our fixed point at the end point. Otherwise, we get $g(a) > a$ and $g(b) < b$.
   Let $h(x) = g(x) - x$. Then $h(a) > 0$ and $h(b) < 0$. Since $h(x) \in C[a,b]$, by IVT, we get the existence of x^* such that $h(x^*) = 0$. Therefore $g(x^*) = x^*$.

2. for showing uniqueness:
   If $x^*, y' \in [a,b]$ with $x^* \neq y^*$, with $g(x^*) = x^*$ and $g(y^*) = y^*$.
   Recall the Mean Value Theorem which said that, if it is differentiable on (a,b): there exists $\xi \in (a,b)$ such that $f'(\xi) = \frac{f(b)-f(a)}{b-a}$.
   Use MVT between x^* and y^*, there exists ξ such that $g'(\xi) = \frac{g(x^*)-g(y^*)}{x^*-y^*} = \frac{x^*-y^*}{x^*-y^*} = 1$
   This is a contradiction to $|g'(x)| \leq k < 1$. Therefore $x^* = y^*$.

Thus we completed the proof of the theorem. □

Algorithm 23 (fixed point iteration)**.** Find fixed point of $g(y)$, i.e., $g(x^*) = x^*$.

1. Choose an initial guess x_0
2. Generate the sequence $\{x_n\}_{n=1}^{\infty}$ by letting $x_n = g(x_{n-1})$ for $n \geq 1$. If $x_n \to x^*$ as $n \to \infty$ and $g(x)$ is continous, then:

$$x^* = \lim_{n\to\infty} x_n = \lim_{n\to\infty} g(x_{n-1}) = g(\lim_{n\to\infty} x_{n-1}) = g(x^*)$$

The stopping criteria is $|x_n - x^*| < \varepsilon$

Theorem 24 (fixed point theorem)**.** *Let $g(x) \in C[a,b]$ such that $g(x) \in [a,b]$ for any $x \in [a,b]$ and $g \in C^1(a,b)$. There exists $k \in (0,1)$ such that $|g'(x)| \leq k < 1$ for any $x \in (a,b)$. Then for any $x_0 \in [a,b]$, the sequence $\{x_n\}_{n=1}^{\infty}$ defined by:*

$$x_n = g(x_{n-1}),\ n \geq 1$$

converges to the unique fixed point of g in $[a,b]$.

Corollary 25. *The **error bound** in the fixed point iteration follows:*

$$|x_n - x^*| \leq k^n \max\{|x_0 - a|, |x_0 - b|\}$$

and

$$|x_n - x^*| \leq \frac{k^n}{1-k}|x_1 - x_0|,\ \forall n \geq 1$$

2.3 Newton's Method

The **goal**: find the root of $f(x) = 0$.
Idea: Convert it to a fixed point problem. We use **Local Linear Approximation.**

Algorithm 26 (**Newton's Method Algorithm**). Let $f \in C^2[a, b]$, we need to find x^* such that $f(x^*) = 0$.

1. Step 1: Choose an initial guess x_0 (note that $f(x_0) \neq 0$).
2. Step 2: For $n \geq 1$:
$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} \text{ for } n \geq 1$$
3. If the stopping criteria is met, then stop; else $n \leftarrow n + 1$ go to Step 2.

The **Stopping criteria** is any of the following:

1. $|x_n - x_{n-1}| < \varepsilon$
2. $\frac{|x_n - x_{n-1}|}{|x_n|} < \varepsilon$ where $p_n \neq 0$
3. $|f(x_n)| < \varepsilon$

Question 27. Why does this have to be local? It need to be local, i.e., x^* need to be closed to x_0 so that we can use Taylor expansion and get (if x_0 is too far, it might diverge):

$$\begin{aligned} f(x^*) &= f'(x_0)(x^* - x_0) + o(|x^* - x_0|) \\ 0 &\approx f(x_0) + f'(x_0)(x^* - x_0) \\ x^* &\approx x_0 - \frac{f(x_0)}{f'(x_0)} \end{aligned}$$

Which we then define $x_1 := \frac{f(x_0)}{f'(x_0)}$ and continue the same process and get $x_2 = \frac{f(x_1)}{f'(x_1)}$. Therefore we get the algorithm

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

See the figure (4) for graphical visualization of Newton Method,

Example 28. Consider $f(x) = x^2 - 3$. Use Newton method to find root within accuracy 10^{-3}.
We get:

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

Recall the three (stopping) criteria:

1. $|x_n - x_{n-1}| < \varepsilon = 10^{-3}$
2. $\frac{|x_n - x_{n-1}|}{|x_n|} < \varepsilon = 10^{-3}$ where $p_n \neq 0$

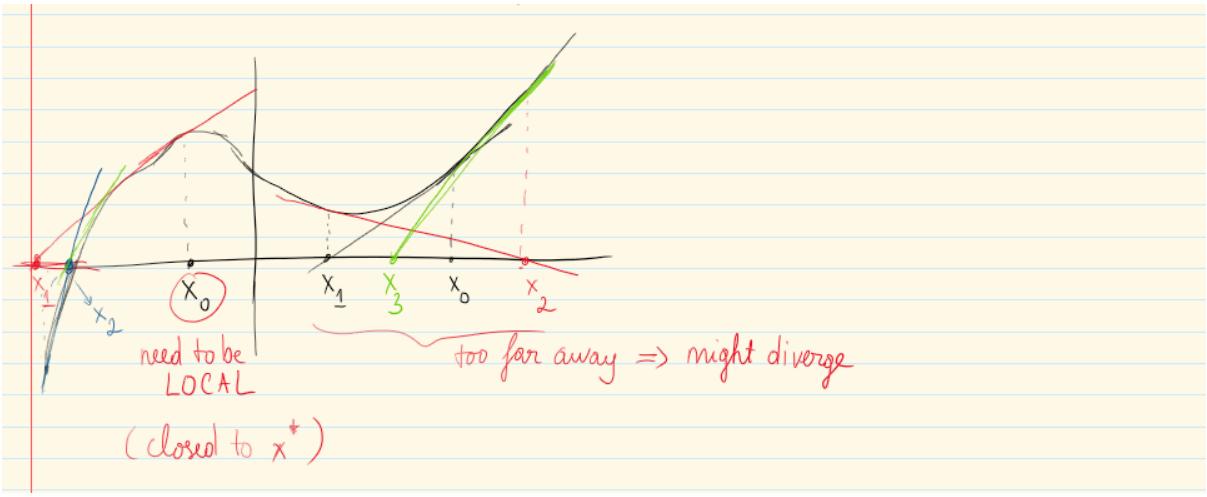


Figure 4: Newton Method Graphical

3. $|f(x_n)| < \varepsilon = 10^{-3}$

We will use criteria (3)- $|f(x_n)| < \varepsilon = 10^{-3}$ for stopping criteria. Now with $x_0 = 1.5$, we see that $f(x_0) = -0.75$ so $x_1 = 1.75$. Then we continue and get $f(x_1) = 0.0625$ and so $x_2 = 1.7321$. Then $f(x_2) = 1.7041.10^{-4} < 10^{-3}$. So we stopped here and choose $x_2 = 1.7321$.

Theorem 29 (convergence of Newton Method)**.** *Let $x^* \in [a, b]$ be a zero of $f \in C^2[a, b]$. If $f'(x^*) \neq 0$ and $f''(x^*) < \infty$, then there exists a $\delta > 0$ such that:*
For all initial guess $x_0 \in (x^* - \delta, x^* + \delta)$, the sequence $\{x_n\}$ generated by Newton method converges quadratically to x^*,i.e., converges to x^* with order 2.

Proof. Let x^* be as in Newton method, that is, $f(x^*) = 0$ and $f'(x^*) \neq 0$. Consider:

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

Then note that $\varphi(x^*) = x^*$. The Newton method is setting $x_{n+1} = \varphi(x_n)$ for $n = 0, 1, \ldots$.
The first question we need to show is that $x_n \xrightarrow{n\to\infty} x^*$.
Let $\rho_n = x_n - x^*$, we need to show $\rho_n \to 0$. First we get:

$$\begin{aligned}\rho_{n+1} &= x_{n+1} - x^* = \varphi(x_n) - \varphi(x^*) \\ &= \varphi'(\xi_n)(x_n - x^*) \text{ , by MVT, for some } \xi_n \text{ in between } x_n \text{ and } x^* \\ &= \varphi'(\xi_n)\rho_n\end{aligned}$$

So

$$|\frac{\rho_{n+1}}{\rho_n}| = |\varphi'(\xi_n)|$$

We claim that this ratio $|\frac{\rho_{n+1}}{\rho_n}| = |\varphi'(\xi_n)| < \frac{1}{2}$, and then we can get $\lim_{n\to\infty} \rho_n = 0$. This claim is true because of this calculation:

$$\varphi'(x) = 1 - \frac{f'.f' - f''.f}{(f')^2} = \frac{f(x^*)f''(x)}{(f'(x))^2}$$

Therefore

$$\varphi'(x^*) = 0 \text{ , because } f(x^*) = 0 \text{ and } f''(x^*) \neq 0 \text{ and } f'(x^*) \neq 0$$

Since φ is continuous, $\varphi'(x^*) = 0$, so there exists a $\delta > 0$ such that for all $\xi \in (x^* - \delta, x^* + \delta)$, we have $|\varphi'(\xi)| < \frac{1}{2}$. This proved the claim.
Now we will show the convergence rate of order 2. By Taylor:

$$f(x^*) = f(x_n) + f'(x_n)(x^* - x_n) + \frac{1}{2}f''(\xi_n)(x^* - x_n)^2$$
$$\text{some } \xi_n \text{ between } x^* \text{ and } x_n$$
$$\Rightarrow f'(x_n)(x_n - x^*) - f(x_n) = \frac{1}{2}f''(\xi_n)(x^* - x_n)^2$$
$$\text{since } f(x^*) = 0$$

Then we divide by $f'(x_n)$ and get:

$$\frac{(x_n - x^*)f'(x_n) - f(x_n)}{f'(x_n)} = \frac{f''(\xi_n)(x^* - x_n)^2}{2f'(x_n)}$$

When taking $n \to \infty$, the term $f(x_n) \to f(x^*) = 0$, therefore:

$$\frac{|x_{n+1} - x^*|}{|x_n - x^*|^2} = \frac{f''(\xi_n)}{2|f'(x_n)|} \xrightarrow{n\to\infty} |\frac{f''(x^*)}{2|f'(x^*)|}| = C > 0$$

Therefore the convergence is of order 2. □

Remark 30. The **limitation of Newton method:**

1. The expression $f(x)$ is unknown
2. $f'(x)$ is expensive to compute
3. value of f may result long numerical calculation, so no f' is available.

Remark 31. The **Solution:** Instead of $f'(x_n)$, we will find some g_k such that $g_k \approx f'(x_n)$, and g_k is much easier to compute. This method is called the **Quasi-Newton Method**.

Definition 32 (**Quasi-Newton Method**)**.** The formula is

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{g_{n-1}} \text{ for } n \geq 1$$

where the $g_n \approx f'(x_n)$.

2.4 Secant Method

Definition 33 (**Secant Method**)**.** The g_n will be calculated as

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

And so we get:

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}f(x_n) \text{ for } n \geq 1$$

Note that this method need 2 initial values x_0 and x_1. See figure (5) for pseudo code, and see figure (6) for an illustrative idea.

Secant Method

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n), \qquad \text{for } n \geq 1$$

Secant Method

Two initial points x_0 & x_1

To find a solution to $f(x) = 0$ given initial approximations p_0 and p_1:

INPUT initial approximations p_0, p_1; tolerance TOL; maximum number of iterations N_0.

OUTPUT approximate solution p or message of failure.

Step 1 Set $i = 2$;
$q_0 = f(p_0)$;
$q_1 = f(p_1)$.

Step 2 While $i \leq N_0$ do Steps 3–6.

Step 3 Set $p = p_1 - q_1(p_1 - p_0)/(q_1 - q_0)$. (*Compute* p_i.)

Step 4 If $|p - p_1| < TOL$ then
OUTPUT (p); (*The procedure was successful.*)
STOP.

Step 5 Set $i = i + 1$.

Step 6 Set $p_0 = p_1$; (*Update* p_0, q_0, p_1, q_1.)
$q_0 = q_1$;
$p_1 = p$;
$q_1 = f(p)$.

Step 7 OUTPUT ('The method failed after N_0 iterations, $N_0 =$', N_0);
(*The procedure was unsuccessful.*)
STOP.

UC RIVERSIDE

Figure 5: Secant Method Pseudocode

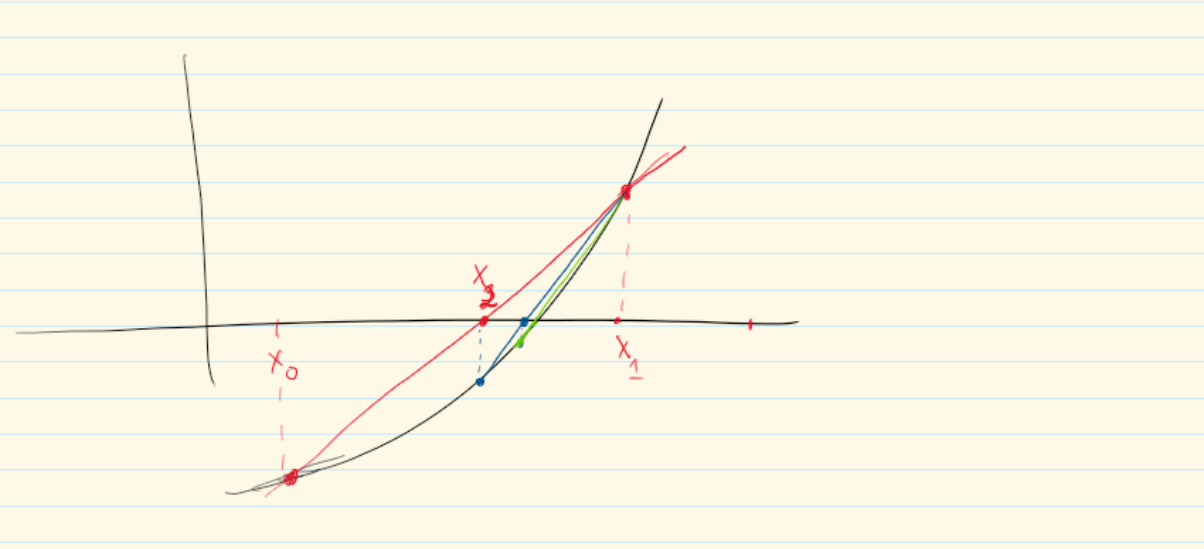


Figure 6: Secant Method graphical

3 Day 3 and 4: Higher dimension methods- Newton system and Steepest Descent Method

3.1 Newton Method for System of Nonlinear equations

The system is

$$\overrightarrow{F}(\overrightarrow{x}) = \overrightarrow{0}$$

Which is equivalent to:

$$\begin{cases} f_1(x_1, \ldots, x_n) & = 0 \\ \ldots \\ f_n(x_1, \ldots, x_n) & = 0 \end{cases}$$

Note the Jacobian equations:

$$f_1(\overrightarrow{x}) = f_1(\overrightarrow{x_k}) + \left[\frac{\partial f_1}{\partial x_1}(\overrightarrow{x_k})(\overrightarrow{x} - \overrightarrow{x_k})_1 \frac{\partial f_1}{\partial x_2}(\overrightarrow{x_k})(\overrightarrow{x} - \overrightarrow{x_k})_2 + \ldots + \frac{\partial f_1}{\partial x_n}(\overrightarrow{x_k})(\overrightarrow{x} - \overrightarrow{x_k})_n + \right] + \ldots +$$

And so we have the Jacobian matrix:

$$F'(\overrightarrow{x}) = \left(\frac{\partial f_i}{\partial x_j} \right)_{ij} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \cdots & \frac{\partial f_1}{\partial x_n} \\ \cdots & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Linear Approximation:

$$\overrightarrow{F}(\overrightarrow{x}) = \overrightarrow{F}(\overrightarrow{x_k}) + \overrightarrow{F'}(\overrightarrow{x_k})(\overrightarrow{x} - \overrightarrow{x_k}) + \ldots$$

Therefore a similar argument like the one dimensional Newton method with the fixed point of

$$G(\overrightarrow{x}) = \overrightarrow{x} - F'(x)^{-1} F(x)$$

Definition 34 (**Newton Method for system of nonlinear equations**).

$$x_k = x_{k-1} - F'(x_{k-1})^{-1} F(x_{k-1}) \text{ for } k \geq 1$$

and the Jacobian matrix:

$$F'(\overrightarrow{x}) = \left(\frac{\partial f_i}{\partial x_j} \right)_{ij}$$

Example 35. Consider the system:

$$\begin{cases} f(x, y) & = 2x + xy - 1 = 0 \\ g(x, y) & = 2y - xy + 1 = 0 \end{cases}$$

With the initial guess

$$\overrightarrow{x_0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

We known the fixed point

$$\overrightarrow{x^*} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

The Jacobian matrix is:

$$\overrightarrow{F'}(\overrightarrow{x}) = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} = \begin{pmatrix} 2+y & x \\ -y & 2-x \end{pmatrix}$$

We begin the iteration process:

1. With the first case

$$\overrightarrow{x_0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

We get the next iteration:

$$\overrightarrow{x_1} = x_0 - (F')^{-1}F(x_0)$$

We get (noting that $F(x_0) = (-1\ 1)^T$):

$$\overrightarrow{x_1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}^{-1} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = -\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}$$

2. The next iteration (note that $F(x_1) = (-\frac{1}{4}\ \frac{1}{4})$):

$$\overrightarrow{x_2} = \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} - \begin{pmatrix} \frac{3}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} \end{pmatrix}^{-1} \begin{pmatrix} -\frac{1}{4} \\ \frac{1}{4} \end{pmatrix} = \begin{pmatrix} \frac{3}{4} \\ -\frac{3}{4} \end{pmatrix}$$

3. Continue the process and get

$$\overrightarrow{x_3} = \begin{pmatrix} 0.875 \\ -0.875 \end{pmatrix}$$

And

$$\overrightarrow{x_4} = \begin{pmatrix} 0.9325 \\ -0.9325 \end{pmatrix}$$

$$\ldots \xrightarrow{n\to\infty} x^* = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

3.2 Some properties of Newton Method for System

Theorem 36 (Convergence of Newton Method of system)**.** *Consider $F : \Omega \subset \mathbb{R}^n \to \mathbb{R}^n$ such that $\overrightarrow{F}(\overrightarrow{x}) = 0$. If $\overrightarrow{F}$ has these conditions:*

1. *There exists an x^* such that $F(x^*) = 0$*
2. *Jacobian $F'(x^*)$ is non-singular*
3. *$F' : \Omega \to \mathbb{R}^n$ is Lipschitz continuous with constant γ, i.e., for all $x, y \in \Omega$:*

$$\|F'(x) - F'(y)\| \leq \gamma\|x - y\|$$

Then there exists $K > 0$ and $\delta > 0$ such that for all $x_0 \in B_\delta(x^*)$, the sequence $\{x_k\}$ generated by the Newton method (of system) satisfies $x_k \in B_\delta(x^*)$, and:

$$\|x_{k+1} - x^*\| \leq K\|x_k - x^*\|^2, \text{ for } k = 0, 1, 2, \ldots$$

That is, Newton method converges quadratically.

Proof. The proof used Fundamental of Calculus (37) and Banach Lemma (38) and another Key Lemma (39). For any $x_k \in B_\delta(x^*)$, we get:

$$x_{k+1} = x_k - F'(x_k)^{-1}F(x_k)$$

Then:

$$\begin{aligned}
x_{k+1} - x^* &= x_k - x^* - F'(x_k)^{-1}F(x_k) \\
&= F'(x_k)^{-1}\Big[F(x_k)(x_k - x^*) - (F(x_k) - F(x^*))\Big] \text{ note that } F(x^*) = 0 \\
&= F'(x_k)^{-1}\left[\int_0^1 F'(x_k)(x_k - x^*)dt - \int_0^1 F'(x^* + t(x_k - x^*))(x_k - x^*)dt\right] \text{ use Fund of Cal} \\
&= F'(x_k)^{-1}\int_0^1 \Big[F'(x_k) - F(x^* + t(x_k - x^*))\Big](x_k - x^*)dt
\end{aligned}$$

Therefore taking the norm:

$$\begin{aligned}
\|x_{k+1} - x^*\| &\leq \|F'(x_k)^{-1}\|\| \int_0^1 \Big[F'(x_k) - F(x^* + t(x_k - x^*))\Big](x_k - x^*)dt\| \\
&= \|F'(x_k)^{-1}\|I
\end{aligned}$$

In the above, let's take a closer look at the norm of the integral:

$$\begin{aligned}
I &\leq \|x_k - x^*\| \int_0^1 \|F'(x_k) - F(x^* + t(x_k - x^*))\|dt \\
&\leq \|x_k - x^*\| \int_0^1 \gamma\|x_k - (x^* + t(x_k - x^*))\|dt \text{ , since } F \text{ is Lipschitz with constant } \gamma \\
&\leq \|x_k - x^*\| \int_0^1 \gamma(1-t)\|x_k - x^*\|dt \\
&= \gamma\|x_k - x^*\|^2 \int_0^1 (1-t)dt = \frac{\gamma}{2}\|x_k - x^*\|^2 \text{ (because } \int_0^1 (1-t)dt = \frac{1}{2})
\end{aligned}$$

Therefore putting it back we get:

$$\|x_{k+1} - x^*\| \leq 2\|F'(x^*)^{-1}\|\frac{\gamma}{2}\|x_k - x^*\|^2 = K\|x_k - x^*\|^2$$

where $K = \|F'(x^*)^{-1}\|\gamma$. □

Theorem 37 (Fundamental Theorem of Calculus)**.** *Let f be differentiable in an open set $\Omega \subset \mathbb{R}^n$ and $x^* \in \Omega$. Then for all $x \in \Omega$ sufficiently closed to x^*, we have:*

$$f(x) = f(x^*) = \int_0^1 f'(x^* + t(x - x^*))(x - x^*)dt$$

Theorem 38 (Banach Lemma)**.** *Let A and B be two $n \times n$ matrices such that B is an approximate inverse of A, namely $\|I - BA\| < 1$, then both A and B are non-singular and:*

$$\|A^{-1}\| \leq \frac{\|B\|}{1 - \|I - BA\|} \text{ and } \|B^{-1}\| \leq \frac{\|A\|}{1 - \|I - BA\|}$$

Theorem 39 (Key Lemma). *Under three conditions in the theorem of convergence of Newton's method, there exists a $\delta > 0$ such that $\forall x \in B_\delta(x^*)$, it holds that:*

$$\begin{aligned} \|F'(x)\| &\leq 2\|F'(x^*)\| \\ \|F'(x)^{-1}\| &\leq 2\|F'(x^*)^{-1}\| \\ \frac{1}{2}\|F'(x^*)^{-1}\|^{-1}\|x - x^*\| \leq \|F(x)\| &\leq 2\|F'(x^*)\|\|x - x^*\| \end{aligned}$$

3.3 Steepest Descent Method

Definition 40 (gradient). For $g : \mathbb{R}^n \to \mathbb{R}$, the **gradient of** g at $x = (x_1\ x_2\ \dots x_n)^T$ is denoted by $\nabla g(x)$ and is defined by:

$$\nabla g(x) = \left(\frac{\partial g}{\partial x_1}(x), \frac{\partial g}{\partial x_2}(x), \dots, \frac{\partial g}{\partial x_n}(x)\right)^T$$

Definition 41 (directional derivative of g at x in the direction of v). The directional derivative of g at x in the direction of v measures the change in the value of the function g relative to the change in the variable in the direction v, it is defined by:

$$D_v g(x) := \lim_{h \to 0} \frac{g(x + hv) - g(x)}{h} = v^T \nabla g(x)$$

Algorithm 42 (Steepest Descent Method).
Goal: Find a local minimum of g.
The idea is:

1. Step 1: Initial point x_0.
2. Step 2: Determine a direction from x_0 that results in a decrease in the value of g.
3. Step 3: Move a little from x_0 along this direction, get a new point x_1.
4. Step 4: Repeat step 2 and 3.

The algorithm is summarized into:

$$x_k = x_{k-1} - \alpha_{k-1} \nabla g(x_{k-1})$$

In which the $\nabla g(x_{k-1})$ is the direction you should move to decrease g, and the α_{k-1} is how much you should move. How we define this α? We first have $\nabla g(x_{k-1})$, then we need to find α_{k-1}. See figure (7) for pictorial illustration.

Algorithm 43 (Line Search). **Line search (in SDM)** is a strategy that first finds a descent direction along which the objective function will be reduced, and then compute a step size that determines how fast the point x should move along that direction:
We choose α_{k-1} so it loosely minimize $g(x_{k-1} + \alpha \nabla g(x_{k-1}))$ over $\alpha \in \mathbb{R}_+$.

Example 44. Use the steepest descent method with $x^{(0)} = (0\ 0\ 0)^T$ to find a reasonable starting approximation to the solution of the nonlinear system:

$$\begin{aligned} f_1(x_1, x_2, x_3) &= 3x_1 - \cos(x_2 x_3) - \frac{1}{2} = 0 \\ f_2(x_1, x_2, x_3) &= x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 = 0 \\ f_3(x_1, x_2, x_3) &= e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} = 0 \end{aligned}$$

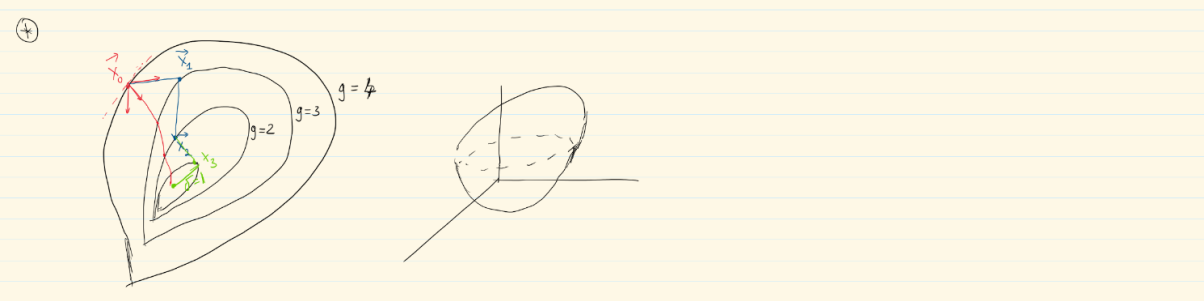


Figure 7: Steepest Descent Method

Step 1: Convert the original root-finding problem to a local min finding problem.
We consider the function $g(x_1, x_2, x_3) = [f_1(x_1, x_2, x_3)]^2 + [f_2(x_1, x_2, x_3)]^2 + [f_3(x_1, x_2, x_3)]^2$.
Step 2: Then the gradient is

$$\begin{aligned}\nabla g(x_1, x_2, x_3) &= \nabla g(x) \\ &= \Bigg(2f_1(x)\frac{\partial f_1}{\partial x_1}(x) + 2f_2(x)\frac{\partial f_1}{\partial x_1}(x) + 2f_1(x)\frac{\partial f_3}{\partial x_1}(x), \\ &\quad 2f_1(x)\frac{\partial f_1}{\partial x_2}(x) + 2f_2(x)\frac{\partial f_1}{\partial x_2}(x) + 2f_1(x)\frac{\partial f_3}{\partial x_2}(x), \\ &\quad 2f_1(x)\frac{\partial f_1}{\partial x_3}(x) + 2f_2(x)\frac{\partial f_1}{\partial x_3}(x) + 2f_1(x)\frac{\partial f_3}{\partial x_3}(x)\Bigg) = 2J(x)^T F(x)\end{aligned}$$

Step 3: The initial point and its gradient (unit vector): For $x^{(0)} = (0\ 0\ 0)^T$, we have:

$$g(x^{(0)}) = 111.975 \text{ and } z_0 = \|\nabla g(x^{(0)})\|_2 = 419.554$$

Let

$$z = \frac{1}{z_0}\nabla g(x^{(0)}) = (-0.0214514, -0.0193062, 0.999583)^T$$

where z is the unit vector.
Step 4: Find an approximated min of g on the curve from $g(x_0)$ to $g(x_0 - z)$. We use a quadratic polynomial to approximate this curve.
With $\alpha_1 = 0$ we get $g_1 = g(x^{(0)} - \alpha_1 z) = g(x^{(0)}) = 111.975$. We also choose arbitrarily let $\alpha_3 = 1$ so that:

$$g_3 = g(x^{(0)} - \alpha_3 z) = 93.5649$$

Because $g_3 < g_1$, we accept α_3 and set $\alpha_2 = \frac{\alpha_3}{2} = 0.5$. Thus

$$g_2 = g(x^{(0)} - \alpha_2 z) = g(x^{(0)} - 0.5z) = 2.53557$$

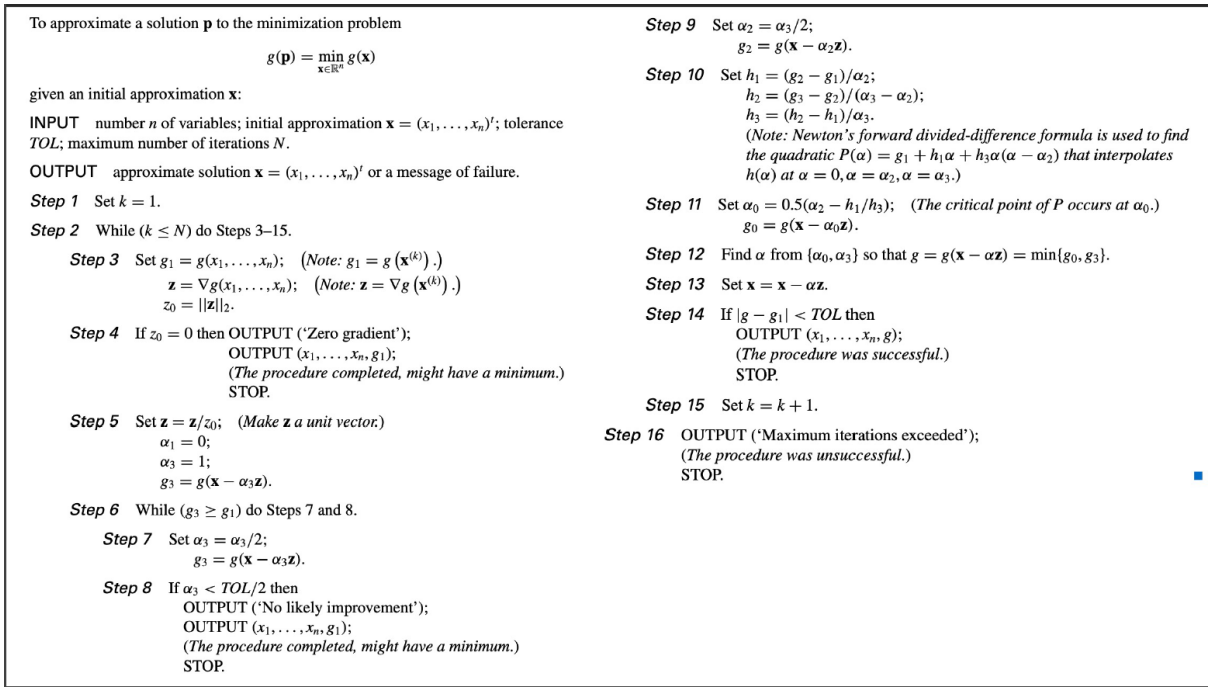
We now find the quadratic polynomial that interpolates the date $(0, 111.975)$, $(1, 93.5649)$, and$(0.5, 2.53557)$. It is most convenient to use **Newton's forward divided-difference interpolating polynomial** for this purpose, which has this form:

$$P(\alpha) = g_1 + h_1\alpha + h_3\alpha(\alpha - \alpha_2)$$

This will interpolate the function

$$g(x^{(0)} - \alpha\nabla g(x^{(0)})) = g(x^{(0)} - \alpha z)$$

at $\alpha_1 = 0$, $\alpha_2 = 0.5$ and $\alpha_3 = 1$ as follow:



To approximate a solution $\mathbf{p}$ to the minimization problem

$$g(\mathbf{p}) = \min_{\mathbf{x}\in\mathbb{R}^n} g(\mathbf{x})$$

given an initial approximation $\mathbf{x}$:

INPUT number n of variables; initial approximation $\mathbf{x} = (x_1, \ldots, x_n)^t$; tolerance TOL; maximum number of iterations N.

OUTPUT approximate solution $\mathbf{x} = (x_1, \ldots, x_n)^t$ or a message of failure.

Step 1 Set $k = 1$.

Step 2 While $(k \le N)$ do Steps 3–15.

Step 3 Set $g_1 = g(x_1, \ldots, x_n)$; (*Note:* $g_1 = g\left(\mathbf{x}^{(k)}\right)$.)
$\mathbf{z} = \nabla g(x_1, \ldots, x_n)$; (*Note:* $\mathbf{z} = \nabla g\left(\mathbf{x}^{(k)}\right)$.)
$z_0 = ||\mathbf{z}||_2$.

Step 4 If $z_0 = 0$ then OUTPUT ('Zero gradient');
OUTPUT $(x_1, \ldots, x_n, g_1)$;
(*The procedure completed, might have a minimum.*)
STOP.

Step 5 Set $\mathbf{z} = \mathbf{z}/z_0$; (*Make* $\mathbf{z}$ *a unit vector.*)
$\alpha_1 = 0$;
$\alpha_3 = 1$;
$g_3 = g(\mathbf{x} - \alpha_3\mathbf{z})$.

Step 6 While $(g_3 \ge g_1)$ do Steps 7 and 8.

Step 7 Set $\alpha_3 = \alpha_3/2$;
$g_3 = g(\mathbf{x} - \alpha_3\mathbf{z})$.

Step 8 If $\alpha_3 < TOL/2$ then
OUTPUT ('No likely improvement');
OUTPUT $(x_1, \ldots, x_n, g_1)$;
(*The procedure completed, might have a minimum.*)
STOP.

Step 9 Set $\alpha_2 = \alpha_3/2$;
$g_2 = g(\mathbf{x} - \alpha_2\mathbf{z})$.

Step 10 Set $h_1 = (g_2 - g_1)/\alpha_2$;
$h_2 = (g_3 - g_2)/(\alpha_3 - \alpha_2)$;
$h_3 = (h_2 - h_1)/\alpha_3$.
(*Note: Newton's forward divided-difference formula is used to find the quadratic* $P(\alpha) = g_1 + h_1\alpha + h_3\alpha(\alpha - \alpha_2)$ *that interpolates* $h(\alpha)$ *at* $\alpha = 0, \alpha = \alpha_2, \alpha = \alpha_3$.)

Step 11 Set $\alpha_0 = 0.5(\alpha_2 - h_1/h_3)$; (*The critical point of* P *occurs at* α_0.)
$g_0 = g(\mathbf{x} - \alpha_0\mathbf{z})$.

Step 12 Find α from $\{\alpha_0, \alpha_3\}$ so that $g = g(\mathbf{x} - \alpha\mathbf{z}) = \min\{g_0, g_3\}$.

Step 13 Set $\mathbf{x} = \mathbf{x} - \alpha\mathbf{z}$.

Step 14 If $|g - g_1| < TOL$ then
OUTPUT $(x_1, \ldots, x_n, g)$;
(*The procedure was successful.*)
STOP.

Step 15 Set $k = k + 1$.

Step 16 OUTPUT ('Maximum iterations exceeded');
(*The procedure was unsuccessful.*)
STOP.

Figure 8: steepest descent pseudocode

1. $\alpha_1 = 0$: the $g_1 = 111.975$
2. $\alpha_2 = 0.5$ and $g_2 = 2.53557$ and $h_1 = \frac{g_2 - g_1}{\alpha_2 - \alpha_1} = -218.878$
3. $\alpha_3 = 1$ and $g_3 = 93.5649$ and $h_2 = \frac{g_3 - g_2}{\alpha_3 - \alpha_2} = 182.059$ and $h_3 = \frac{h_2 - h_1}{\alpha_3 - \alpha_1} = 400.937$

Thus

$$P(\alpha) = 111.975 - 218.878\alpha + 400.937\alpha(\alpha - 0.5)$$

We use $P'(\alpha) = 0$ and solve for $\alpha = \alpha_0 = 0.522959$. Since $g_0 = g(x^{(0)})$ Now once we found this α_0, recall that we need to find the min value of the polynomial P, so we evaluate $g_0 = g(x^{(0)} - \alpha_0 z) = 2.32762$ and this value g_0 is smaller than g_1 and g_2, so we will this α_0 and get our next iteration:

$$x^{(1)} = x^{(0)} - \alpha_0 z = x^{(0)} - 0.522959z = (0.0112182, 0.0100964, -0.522741)^T$$

and

$$g(x^{(1)}) = 2.32762$$

Algorithm 45. The pseudocode is found in figure (8).

4 Day 5: Numerical Linear Algebra

The main idea are to:

1. Find $\overrightarrow{x}$ such that $A\overrightarrow{x} = \overrightarrow{b}$
2. Find $(\lambda, \overrightarrow{v})$ such that $A\overrightarrow{v} = \lambda\overrightarrow{b}$

4.1 LU Decomposition

4.1.1 Gaussian Elimination

Definition 46 (Gaussian elimination)**.** Gaussian elimination is a process to reduce a full $n \times n$ system of equations:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n} &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots a_{nn}x_n &= b_n \end{aligned}$$

into an **upper diagonal** system of equations:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{nn}x_n &= b_1 \\ \tilde{a}_{22}x_2 + \ldots + \tilde{a}_{2n}x_n &= \tilde{b}_2 \\ &\ldots \\ \tilde{a}_{nn}x_n &= \tilde{b}_n \end{aligned}$$

Gaussian elimination is equivalent to:

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \rightarrow \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ 0 & \tilde{a}_{22} & \ldots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \tilde{a}_{nn} \end{pmatrix}$$

Example 47. Perform Gaussian Elimination of A where

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 4 \\ 3 & 5 & 7 \end{pmatrix}$$

First we change:

$$R_2 \leftrightarrow R_2 - 2R_1$$

Notice the coefficient 2 in front of the operation of R_1, and we always want to keep $-$ sign in between. This is a change from R_2 related to R_1, so we get a coefficient $l_{21} = 2$ (which we will use later in the lower L matrix):

$$A \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 3 & 5 & 7 \end{pmatrix}$$

Next we would like to get the first entry in the third row to zero, and we perform this operation:

$$R_3 \leftrightarrow R_3 - 3R_1$$

Again, notice the coefficient 3 in front of the operation of R_1, and we always want to keep $-$ sign in between. This is a change from R_3 related to R_1, so we get a coefficient $l_{31} = 3$ (which we will use later in the lower L matrix):

$$A \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 2 & 4 \end{pmatrix}$$

Next we would like to turn entries a_{32} into zero, and we can do this by:

$$R_3 \leftrightarrow R_3 - (1)R_2$$

Again, notice the coefficient 1 in front of the operation of R_2, and we always want to keep $-$ sign in between. This is a change from R_3 related to R_2, so we get a coefficient $l_{32} = 1$ (which we will use later in the lower L matrix):

$$A \to \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{pmatrix}$$

which is now an upper diagonal matrix.

Definition 48 (LU factorization)**.** Let A be an $n \times n$ matrix. If all n leading principal minors:

$$A_k = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{pmatrix}, \ k = 1, 2, \dots, n$$

of the $n \times n$ matrix A are non-singular, then A will have an LU factorization:

$$A = LU$$

where L is a lower triangular matrix with 1 as its diagonal entries and U is an upper triangular matrix.

Remark 49. An LU factorization is **not** unique. There are infinite possible of L and U matrices.

Remark 50. Note that the matrix obtained in the Gaussian elimination process is already an upper triangular matrix and we had our U in the LU factorization. To find the L, there are many ways:

1. One way is that we can assume the remaining elements in L (besides the zeros and ones) are artificial variables, then make equations $A = LU$ and solve for this to find the artificial variables. However, in this course, we will **not** use this method.

2. The other method is the multiplier coefficients that we obtained when doing Gaussian elimination will be the entries in the L matrix (the l_{ij}). This is the method we will use.

Example 51. Perform LU factorization for A where:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 4 \\ 3 & 5 & 7 \end{pmatrix}$$

In the Gaussian elimination example, we already get out upper triangular matrix:

$$U = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{pmatrix}$$

Now for the L matrix, we also got from the Gaussian elimination example the entries $l_{21} = 2$, $l_{31} = 3$ and $l_{32} = 1$, hence the lower diagonal matrix is:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 1 & 1 \end{pmatrix}$$

Now that we are able to do LU factorization, what does it help?

Example 52. Find $\overrightarrow{x} = (x_1\ x_2\ x_3)^T$ such that:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 4 \\ 3 & 5 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix}$$

We know that $A = LU$, so $Ax = c$ where $c = (1\ 3\ 5)^T$ is the same with $LUx = c$. We first try to solve for $y = (y_1\ y_2\ y_3)^T$ such that $Ly = c$. (Note that $y = Ux$):

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix}$$

This gives $y_1 = 1$, $2y_1 + y_2 = 3$ and $3y_1 + y_2 + y_3 = 5$. So $y_2 = 0$ and $y_3 = 1$. So $y = (1\ 0\ 1)^T$. Next we solve $Ux = y$:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

This gives $2x_3 = 1$, $x_1 + x_2 + x_3 = 1$ and $2x_2 + 2x_3 = 0$, so $x_1 = 1$, $x_2 = -\frac{1}{2}$ and $x_3 = \frac{1}{2}$.

Algorithm 53. For computer to do this LU decomposition, we have the following pseudocode:
To factor the $n \times n$ matrix $A = \{a_{ij}\}$:

1. INPUT: dimension n, the entries a_{ij} of A, the $l_{11} = l_{22} = \ldots = l_{nn} = 1$ of L diagonal matrix.
2. OUTPUT: the other entries in L and the entries in U.

The steps are:

1. Step 1: Initialize L to an identity matrix, dimension $n \times n$ and $U = A$
2. Step 2: For $i = 1, \ldots, n$ do step 3
3. Step 3: For j=$i + 1, \ldots, n$ do step 4-5:
4. Step 4: Set $l_{ji} = \frac{u_{ji}}{u_{ii}}$
5. Step 5: Perform $U_j = (U_j - l_{ji}U_i)$ where the U_i,U_j represent the i and j rows of the matrix U respectively.

Algorithm 54. Python/Numpy immplementation:

```
def lu(A):

    #Get the number of rows
    n = A.shape[0]

    U = A.copy()
    L = np.eye(n, dtype=np.double)

    #Loop over rows
    for i in range(n):

        #Eliminate entries below i with row operations
        #on U and reverse the row operations to
        #manipulate L
        factor = U[i+1:, i] / U[i, i]
        L[i+1:, i] = factor
        U[i+1:] -= factor[:, np.newaxis] * U[i]

    return L, U
```

Algorithm 55. A details pseudocode:

1. Step 1: Select l_{11} and u_{11} such that $l_{11}u_{11} = 1$. If $l_{11}u_{11} = 0$ then OUTPUT 'Factorization impossible'; STOP.
2. Step 2: For $j = 2, \ldots, n$: set:
   - $U_{1j} = \frac{a_{1j}}{l_{11}}$; (first row of U)
   - $l_{j1} = \frac{a_{j1}}{u_{11}}$ (first column of L).
3. Step 3: For $i = 2, \ldots, n-1$ do step 4 and 5:
   (a) Step 4: Select l_{ii} and u_{ii} satisfying $l_{ii}u_{ii} = a_i i - \sum_{k=1}^{i-1} l_{ik}u_{ki}$. If $l_{ii}u_{ii}$ then OUTPUT 'Factorization impossible;. STOP.
   (b) Step 5: For $j = i+1, \ldots, n$:
   - Set $u_{ij} = \frac{1}{l_{ii}}\left[a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}\right]$ (ith row of U)
   - Set $l_{ji} = \frac{1}{u_{ii}}\left[a_{ji} - \sum_{k=1}^{i-1} l_{jk}u_{ki}\right]$ (ith column of L)
4. Step 6: Select l_{nn} and u_{nn} satisfying $l_{nn}u_{nn} = a_{nn} - \sum_{k=1}^{n-1} l_{nk}u_{kn}$ (Note if $l_{nn}u_{nn} = 0$, then $A = LU$ but A is singular)
5. Step 7:
   - OUTPUT: l_{ij} for $j = 1, \ldots, i$ and $i = 1, \ldots, n$
   - OUTPUT u_{ij} for $j = 1, \ldots, i$ and $i = 1, \ldots, n$
   - STOP

4.2 LDU Factorization

Definition 56. The LDU factorization of an $n \times n$ matrix A is $A = LDU$ where:

- L: Lower triangular matrix with 1 diagonal
- D: diagonal matrix
- U: Upper triangular matrix with 1 diagonal.

Remark 57. While the LU factorization is **not** unique, the LDU factorization is **unique**.

Algorithm 58. To find the LDU factorization from the LU factorization, we do as follow:
Suppose $A = \tilde{L}\tilde{U}$. Then we will set:

$$D = \text{diag}(\tilde{U})$$

Then we find

$$U = D^{-1}\tilde{U}$$

And we get:

$$A = \tilde{L}DU$$

Example 59. Find LDU factorization of

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 4 \\ 3 & 5 & 7 \end{pmatrix}$$

We know that $A = \tilde{L}\tilde{U}$ where

$$\tilde{L} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 1 & 1 \end{pmatrix}$$

and

$$\tilde{U} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{pmatrix}$$

So the digonal matrix is

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

Therefore we get :

$$U = D^{-1}\tilde{U} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Then $A = \tilde{L}DU$ is the LDU factorization of A.

4.3 LDL^T factorization for $A = A^T$

In the case that $A = A^T$, that is, matrix A is symmetric, we have the LDL^T factorization.

Definition 60. The LDL^T factorization of an $n \times n$ symmetric matrix $A = A^T$ is $A = LDL^T$ where:

- L is lower triangular matrix with 1 diagonal
- D diagonal matrix

This is because when we get $A = A^T$, the usual LDU factorization gives:

1. $A = LDU$ so $A^T = U^T D^T L^T = U^T D L^T$
2. Comparing $A = A^T$ gives:
$$LDU = U^T D L^T$$

By uniqueness of LDU factorization, we must have: $L = U^T$ and $U = L^T$. Therefore $A = LDL^T$.

5 Numerical Linear Algebra (continued): SPD Matrices

5.1 Cholesky Factorization

Definition 61 (symmetric positive definite (SPD))**.** A matrix A is said to be **symmetric positive definite (SPD)** if $A = BB^T$ for some nonsingular matrix B.
Equivalently, A is said to be **symmetric positive definite (SPD)** if A is symmetric and:
$$x^T Ax > 0, \ \forall x \neq 0$$

Definition 62 (Cholesky Factorization)**.** The **Cholesky Factorization** of an $n \times n$ SPD matrix A is LL^T where L is a lower triangular matrix.

Remark 63. The diagonal entries of L in the Cholesky Factorization is **not necessarily** all 1 (Not like the entries of L in LU factorization).

Proof. A is SPD. Recall we can have $A = LDU$ uniquely. Because A is symmetric, we also have: $A^T = (LDU)^T = U^T D L^T$, and since $A = A^T$, so $LDU = U^T D L^T$. Because LDU factorization is unique, we must have $L = U^T$ and $U = L^T$. Therefore, we can write:
$$A = LDL^T$$
Denote $D = \text{diag}(d_1 \ d_2 \dots d_n)$. We next will show all $d_i > 0$.
Consider $\overrightarrow{x}_i = (L^T)^{-1}\overrightarrow{e}_i$. Because A is SPD, we get $\overrightarrow{x}_i^T A \overrightarrow{x}_i > 0$, applying to this particular $\overrightarrow{x}_i = (L^T)^{-1}\overrightarrow{e}_i$:

$$\begin{aligned} 0 < \overrightarrow{x}_i^T A \overrightarrow{x}_i &= \overrightarrow{e}_i (L^{-1}) A (L^T)^{-1} \overrightarrow{e}_i \\ &= \overrightarrow{e}_i^T (L^{-1}) L D L^T (L^T)^{-1} \overrightarrow{e}_i \\ &= \overrightarrow{e}_i^T D \overrightarrow{e}_i = d_i \end{aligned}$$

Since all the diagonal $d_i > 0$, we can take its square root, denote $\bar{D} = \text{diag}(\sqrt{d_1} \dots \sqrt{d_n})$. We also note that $D = (\bar{D})^2$. Now:

$$A = L\bar{D}\bar{D}^T L^T = (L\overline{D})^T(\overline{LD})^T = \tilde{L}\tilde{L}^T$$

where $\tilde{L} = L\bar{D}$ is a lower triangular matrix. □

5.2 Power Method

Let A be an SPD matrix. From theory, there exists eigenpairs $\{(v_k, \lambda_k)\}_{k=1}^n$ such that

$$Av_k = \lambda_k v_k$$

Assume that $\{v_k\}$ forms an orthonormal basis. In addition, also assume that

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$$

The power method will depends on the largest term $|\lambda_1|$.
For any $\overrightarrow{x}$, we can write:

$$\overrightarrow{x} = \sum_{k=1}^{n} \alpha_k \overrightarrow{v}_k$$

Then:

$$A\overrightarrow{x} = \sum_{k=1}^{n} \alpha_k (A\overrightarrow{v}_k) = \sum_{k=1}^{n} \alpha_k \lambda_k \overrightarrow{v}_k$$

If $\alpha_1 \neq 0$: (note that $A^m\overrightarrow{x} = \sum_{k=1}^n \alpha_k \lambda^m \overrightarrow{v}_k$):

$$\begin{aligned}
\frac{A^m\overrightarrow{x}}{\|A^m\overrightarrow{x}\|} &= \frac{\sum_{k=1}^n \alpha_k \lambda_k^m \overrightarrow{v}_k}{\sqrt{\sum_{k=1}^n \alpha_k^2 \lambda_k^{2m}}} \\
&= \frac{\sum_{k=1}^n \alpha_k \left(\frac{\lambda_k}{\lambda_1}\right)^m \overrightarrow{v}_k}{\sqrt{\sum_{k=1}^n \alpha_k^2 \left(\frac{\lambda_k}{\lambda_1}\right)^{2m}}} \\
&\xrightarrow{m\to\infty} \frac{\alpha_1}{|\alpha_1|}\overrightarrow{v}_1
\end{aligned}$$

This is because when $m \to \infty$, the term $\frac{(\lambda_k}{\lambda_1})^m$ goes to zero except $k = 1$.

1. **Power Method I:** Consider

$$x_k = Ax_{k-1},\ \mu_k = \frac{\langle Ax_k, x_k\rangle}{\langle x_k, x_k\rangle}$$

2. **Power Method II:** Consider:

$$w_k = Ax_{k-1},\ \mu_k = \langle Ax_k, x_k\rangle,\ x_k = \frac{w_k}{\|w_k\|}$$

5.3 QR Factorization

This is a way to find all eigenvalues simultaneously.

Definition 64 (QR factorization)**.** A **QR factorization** is such that $A = QR$, where Q is orthogonal ,i.e., $Q^T = Q^{-1}$ and R is upper triangular.

Definition 65 (Gram-Schmidt Process)**.** Let A be an $n \times n$ matrix. We will factor A into:

$$A = [a_1 \ a_2 \dots a_n] = [q_1 \ q_2 \dots q_n] \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix}$$

where $r_{ij} = q_i.a_j = q_i^T.a_j$ and $r_{ii} = \|v_j\|$ and the algorithm is:

$$\begin{aligned} v_1 &= a_1, \ q_1 = \frac{v_1}{\|v_1\|} \\ v_2 &= a_2 - \langle q_1, q_2 \rangle q_1, \ q_2 = \frac{v_2}{\|v_2\|} \\ &\vdots \\ v_k &= a_k - \sum_{j=1}^{k-1} \langle q_j, a_k \rangle q_j \ q_k = \frac{v_k}{\|q_k\|} \end{aligned}$$

Algorithm 66. The pseudocode for QR Factorization:
For $j = 1, \dots, n$:

$$\begin{aligned} &v_j = a_j \\ &\quad \text{for } i = 1, \dots j-1 : \\ &\quad r_{ij} \leftrightarrow q_i^T a_j \\ &\quad v_j \leftarrow v_i - r_{ij} q_i \\ &\quad r_{jj} = \|v_j\|_2 \\ &\quad q_j = \frac{v_j}{r_{jj}} \end{aligned}$$

Return $Q = [q_1 \ \dots Q_n]$ and $R = (r_{ij})_{n \times n}$.

Example 67. QR factorization:

$$A = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}$$

We begin by finding first q_1, which is just the "unit" version of $v_1 = a_1$:

$$v_1 = a_1 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

So

$$q_1 = \frac{v_1}{\|v_1\|} = \frac{1}{\sqrt{10}} \begin{pmatrix} 3 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{3}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} \end{pmatrix}$$

And we also get our $r_{11} = \|v_1\| = \sqrt{10}$. Next we get our second vector v_2 by inner product:

$$\begin{aligned} v_2 &= a_2 - \langle q_1, a_2 \rangle q_1 \\ &= \begin{pmatrix} 2 \\ 2 \end{pmatrix} - \langle \begin{pmatrix} \frac{3}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix} \rangle \begin{pmatrix} \frac{3}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} \end{pmatrix} \\ &= \begin{pmatrix} \frac{-2}{5} \\ \frac{6}{5} \end{pmatrix} \end{aligned}$$

And the q_2 is the unit vector from v_2 so

$$q_2 = \begin{pmatrix} \frac{-1}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} \end{pmatrix}$$

And we get our $r_{22} = \|v_2\| = \sqrt{\frac{8}{5}}$. For the r_{12} term, it is the inner product of q_1 and a_2:

$$r_{12} = \langle \begin{pmatrix} \frac{3}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix} \rangle = \sqrt{\frac{32}{5}}$$

So we get the Q matrix:

$$Q = [q_1 \ q_2] = \begin{pmatrix} \frac{3}{\sqrt{10}} & \frac{-1}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \end{pmatrix}$$

And the R matrix is:

$$R = \begin{pmatrix} \sqrt{10} & \sqrt{\frac{32}{5}} \\ 0 & \sqrt{\frac{8}{5}} \end{pmatrix}$$

So

$$A = QR \iff \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} \frac{3}{\sqrt{10}} & \frac{-1}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \end{pmatrix} \begin{pmatrix} \sqrt{10} & \sqrt{\frac{32}{5}} \\ 0 & \sqrt{\frac{8}{5}} \end{pmatrix}$$

5.4 QR Iteration

Definition 68 (QR iteration)**.** Let A be an $n \times n$ matrix.

1. Set $A_0 = A$.
2. Do the iteration, at the k-th step, assume that we can QR factorization A_k into $A_k = Q_k R_k$ where Q_k is orthogonal (i.e., $Q_k^T = Q_k^{-1}$) and R_k is an upper triangular matrix.
3. Set $A_{k+1} = R_k Q_k$ and continue the iteration process.

Note that the A_k are similar so they preserve the eigenvalues (see remark later).
Then under certain conditions, the $A_k \to \overline{A}$ in which $\overline{A}$ is an upper triangular matrix. The $\overline{A}$ is called the **Schur form of** A in which its diagonal entries will give the eigenvalues of A.

Definition 69 (similar)**.** Two matrices A and B is called **similar** if there exists an invertible matrix P such that $B = P^{-1}AP$.

Remark 70. Why does the A_k are similar? We note that:

$$A_{k+1} = R_k Q_k = (Q_k^{-1} Q_k) R_k Q_k = Q_k^{-1}(Q_k R_k) Q_k = Q_k^{-1} A_k Q_k$$

And Q_k is invertible because of the QR factorization. Therefore we get A_{k+1} and A_k is similar. In addition, similar matrices have same eigenvalues.

Definition 71 (Schur Decomposition)**.** Let A be a complexed value $n \times n$ matrix. If $A = QUQ^{-1}$ where Q is unitary (i.e., $QQ^* = I$) and U is an upper triangular matrix, then this is the Schur form of A.

Question 72. How do we proceed? Unfortunately, Gram-Schmidt process for QR factorization is **slow and unstable**. So we will use a different method, which we referred to as **Householder method**.

5.5 Householder Method

Definition 73 (Householder Reflection Method)**.** Recall we wish to find the eigenvalues of a matrix A. The earlier method Gram-Schmidt is slow and unstable. We will introduce a different method, which we can call Householder Reflection method, and in general is divided into 2 steps:

1. Step 1: (the Householder Reflection) We find P orthogonal matrix such that PAP^{-1} is a Hessenburg matrix.
2. Step 2: Use rotation to perfom QR factorization on the upper Hessenburg matrix H. Then we update QR iteration for H.

5.5.1 Step 1: Find Hessenburg matrix

Definition 74 (upper Hessenburg matrix)**.** H is an **upper Hessenburg matrix** if $a_{ij} = 0$ for all i, j with $i > j + 1$.

$$H = \begin{pmatrix} * & * & \dots & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{pmatrix}$$

Definition 75 (Householder Reflection- Householder Matrix)**.** Given a matrix A, we will find a matrix P that is orthogonal such that:

$$PAP^{-1} = H$$

where H is an Hessenburg matrix. The matrix P is called **Householder matrix**.

Remark 76 (How to find Householder matrix)**.** In addition, a Householder matrix must satisfy:

$$P = I - vv^T$$

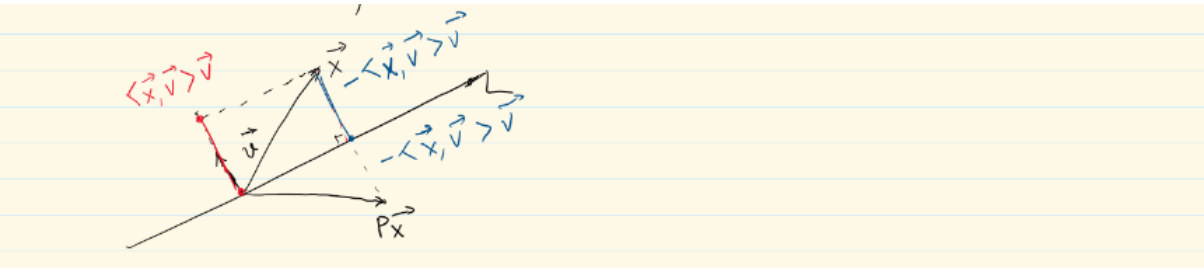


Figure 9: reflection matrix P applied to a vector x

where $\|v\| = 1$.
P is called a reflection matrix because (see figure (9)) of what happen when applying P to a vector x:

$$\begin{aligned} Px &= x + 2(-\langle x, v\rangle v) \\ &= x - 2\langle x, v\rangle v \\ &= (I - 2vv^T)x \end{aligned}$$

Remark 77 (properties of P)**.** P has some properties:

1. Hamiltonian: $P = P^T$
2. unitary: $P^{-1} = P^T$, or equivalently, $PP^T = I$
3. involuntary: $P = P^{-1}$
4. -1 is an eigenvalue of P with multiplicity 1: $Pv = -v$.
   This is because in $\mathbb{R}^n$ there are $n-1$ linearly independent unit vectors orthogonal to v. Thus 1 is an eigenvalue of P with multiplicity $n-1$.
5. $\det(P) = 1$

Algorithm 78. How do we find P and at the same time turn A into an Hessenburg matrix H? The theory behind it is pretty complicated to explain, it might be better to see summary method and an example in later part.
We begin with:

$$A = \begin{matrix} * & * & * & * \\ w_{11} & * & * & * \\ w_{12} & * & * & * \\ w_{13} & * & * & * \end{matrix}$$

where the red is a vector in $w_1 = (w_{11}\ w_{12}\ w_{13})^T \in \mathbb{R}^3$
We need to transform that A matrix into a new matrix:

$$H_1 = \begin{matrix} * & * & * & * \\ v_{11} & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{matrix}$$

We need to find a P_1 which is a 3 by 3 matrix such that it turns the original w_1 into the new $v_1 = (v_{11}\ 0\ 0)^T \in \mathbb{R}^3$. Actually, we need to find P_1 such that $P_1(w_1) = \|w_1\|e_1 = v_{11}e_1$. How can we do this? See algorithm (79) later. Now suppose you can find such a 3 by 3 matrix P_1, then how do we turn A into H_1? Recall that both A and H_1 are 4 by 4 matrix. We did it my multiplying A with a "complemented matrix $\overline{P_1}$ (which is now is 4 by 4) and its inverse, where the complemented matrix $\overline{P_1}$ is obtained from our P_1 by:

$$\overline{P} = \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \hline \mathbf{0} & P_1 \end{array}\right)$$

Then we get our H_1 by multiplying: $H_1 = \overline{P_1}A\overline{P_1}^{-1}$. But do we have to find the inverse matrix $\overline{P_1}^{-1}$? The good news is **no**. We do not need to find the inverse. This is because $\overline{P}$ is a Householder matrix, so $\overline{P}^{-1} = \overline{P}$. Therefore, we get our matrix H_1 by:

$$H_1 = \overline{P_1}A\overline{P_1}$$

Now H_1 is still not of Hessenburg type, so we need to continue to turn $H_1 = A_1$ of the form: $H_1 = A_1 = \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ 0 & w_{21} & * & * \\ 0 & w_{22} & * & * \end{pmatrix}$ into: $H_2 = \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{pmatrix}$

How do we do this process? Similarly, we need to find a P_2 2 by 2 matrix making the red vector $w_2 = (w_{21}\ w_{22})^T \in \mathbb{R}^2$ such that $P_2(w_2) = \|w_2\|e_1 = (*\ 0) \in \mathbb{R}^2$. After we found the P_2, we turn it into the complemented matrix $\overline{P_2}$ which is 4 by 4 by:

$$\overline{P_2} = \left(\begin{array}{cc|c} 1 & 0 & \mathbf{0} \\ 0 & 1 & \\ \hline \multicolumn{2}{c|}{\mathbf{0}} & P_2 \end{array}\right)$$

And then find our H_2 as before by:

$$H = H_2 = \overline{P_2}A_1\overline{P_2}$$

And note that H_2 is now of the form of an upper Hessenburg matrix. and our original matrix A is similar to this $H_2 = H$ matrix. Why?
Because

$$\begin{aligned} H &= \overline{P_2}A_1\overline{H_1} = \overline{P_2P_1}A\overline{P_1P_2} \\ &= PAP = PAP^T \end{aligned}$$

where $P = \overline{P_2P_1}$ and is orthogonal (since $\overline{P_2}$ and $\overline{P_1}$ are orthogonal).

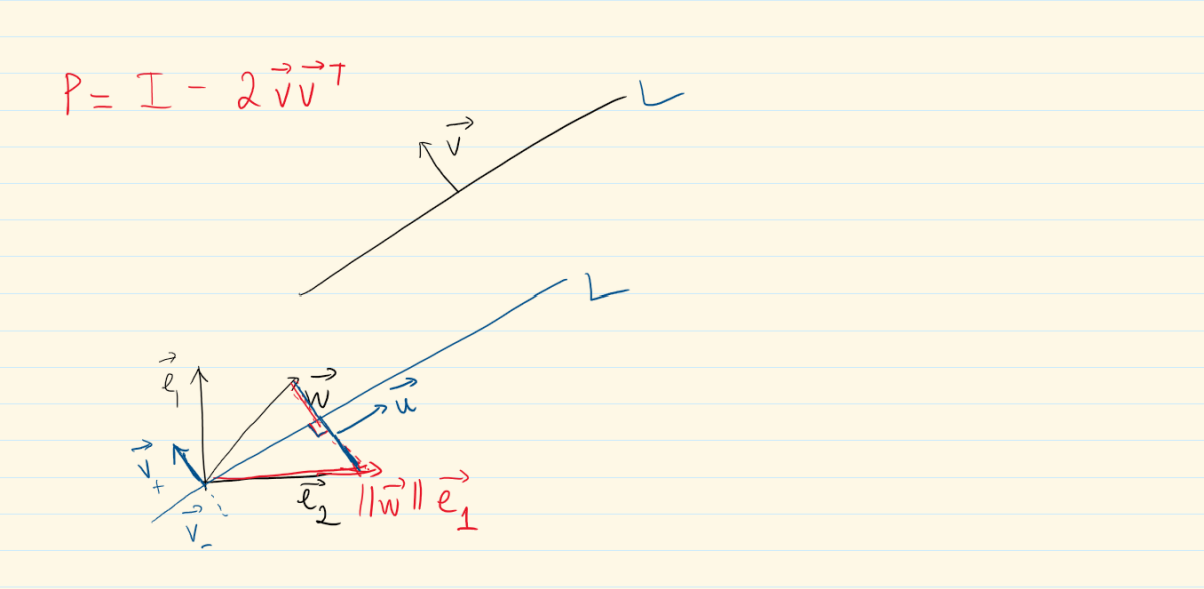


Figure 10: algorithm to find Householder matrix P

Algorithm 79 (turn any vector into a scale of directional vector)**.** Suppose we have a vector

$$w = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

How do we find a matrix P such that $P(w) = \|w\|e_1$? The matrix P (note the P is 3 by 3) will satisfy:

$$P = I - 2vv^T$$

where the v is the unit vector obtained from the $u = w - \|w\|e_1$. (i.e., $v = \frac{1}{\|u\|}u$). How is this possible? (see picture (10).

$$\begin{aligned} w + u &= \|w\|e_1 \\ \implies u &= \|w\|e_1 - w \end{aligned}$$

We find a v such that $v \parallel u$ and $\|v\| = 1$, and the easiest way to do such is to normalize u:

$$\begin{aligned} v &= \pm\frac{u}{\|u\|} \\ &= \pm\frac{\|w\|e_1 - w}{\|\|w\|e_1 - w\|} \end{aligned}$$

Example 80. Consider the matrix

$$A = \begin{pmatrix} 4 & 1 & -1 & 2 \\ 1 & 2 & 0 & 1 \\ -2 & 0 & 3 & -2 \\ 2 & 1 & -2 & -1 \end{pmatrix}$$

Note that $A = A^T$. Find the P such that $PAP^T = H$, i.e., transform matrix A into an upper Hessenburg matrix H.
We begin by looking at the first 3 entries in the first column, the red entries $w = (1\text{-}2\ 2)^T$:

$$A = \begin{matrix} 4 & 1 & -2 & 2 \\ 1 & 2 & 0 & 1 \\ -2 & 0 & 3 & -2 \\ 2 & 1 & -2 & -1 \end{matrix}$$

In $\mathbb{R}^3$ we now have: $u = w - \|w\|e_1 = (1\ \ -2\ 2)^T - 3(1\ 0\ 0)^T = (-2\ \ -2\ 2)^T$. Then we get:

$$v = \frac{1}{\sqrt{3}} \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}$$

Therefore we get our 3 by 3 matrix P:

$$\begin{aligned} P &= I_3 - 2vv^T \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{2}{3} \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} (-1\ \ -1\ 1) \\ &= \begin{pmatrix} \frac{1}{3} & -\frac{2}{3} & \frac{2}{3} \\ -\frac{2}{3} & -\frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{pmatrix} \end{aligned}$$

Then to find our first step $H_1 = A_1$ we take:

$$\left(\begin{array}{c|c} 1 & \mathbf{0} \\ \hline \mathbf{0} & P \end{array}\right) A \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \hline \mathbf{0} & P \end{array}\right) = \begin{pmatrix} 4 & 3 & 0 & 0 \\ 3 & \frac{10}{3} & -\frac{4}{3} & -1 \\ 0 & -\frac{4}{3} & -1 & -\frac{4}{3} \\ 0 & -1 & -\frac{4}{3} & \frac{5}{3} \end{pmatrix} = A_1$$

We should also check that $A_1 = A_1^T$. Now next we proceed with the same steps applied to this matrix and the second column red entries:

$$A_1 = \begin{matrix} 4 & 3 & 0 & 0 \\ 3 & \frac{10}{3} & -\frac{4}{3} & -1 \\ 0 & -\frac{4}{3} & -1 & -\frac{4}{3} \\ 0 & -1 & -\frac{4}{3} & \frac{5}{3} \end{matrix}$$

Now in $\mathbb{R}^2$, the $w = (-\frac{4}{3}\ \ -1)^T$ will give:

$$u = w - \|w\|e_1 = \begin{pmatrix} -\frac{4}{3} \\ -1 \end{pmatrix} - \frac{5}{3} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -3 \\ -1 \end{pmatrix}$$

which then give the unit vector:

$$v = \frac{1}{\sqrt{3}} \begin{pmatrix} -3 \\ -1 \end{pmatrix}$$

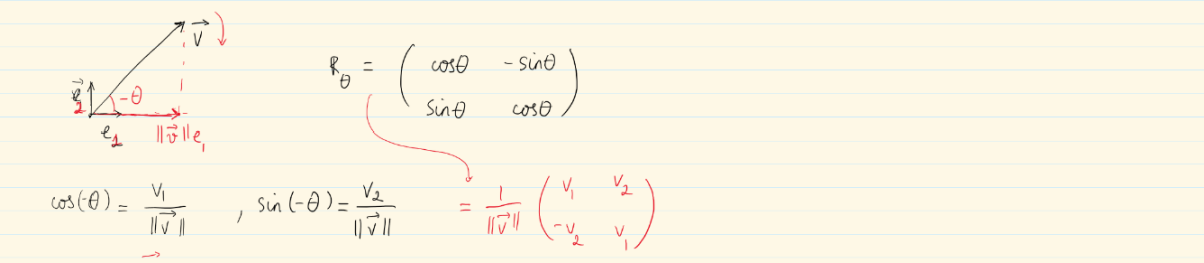


Figure 11: rotation by angle θ

Then the 2 by 2 matrix P_2 is:

$$P_2 = I_2 - 2vv^T = \begin{pmatrix} -\frac{4}{5} & -\frac{3}{5} \\ -\frac{3}{5} & \frac{4}{5} \end{pmatrix}$$

Therefore putting this P_2 to create a 4 by 4 matrix and multiply with A_2 we get:

$$\left(\begin{array}{cc|c} 1 & 0 & \mathbf{0} \\ 0 & 1 & \\ \hline \mathbf{0} & & P \end{array}\right) A \left(\begin{array}{cc|c} 1 & 0 & \mathbf{0} \\ 0 & 1 & \\ \hline \mathbf{0} & & P \end{array}\right) = \begin{pmatrix} 4 & 3 & 0 & 0 \\ 3 & \frac{10}{3} & \frac{5}{3} & 0 \\ 0 & \frac{5}{3} & -\frac{33}{25} & -\frac{68}{75} \\ 0 & 0 & -\frac{68}{75} & \frac{149}{75} \end{pmatrix} = H$$

5.5.2 Step 2: The iteration process

Now we will proceed with the steps to **transform an upper Hessenburg matrix H into an upper triangular matrix R.**

Definition 81 (rotation matrix)**.** A 2-dimensional **rotation matrix** is defined as:

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

For any $v \in \mathbb{R}^2$, $R_\theta v$ rotates the vector v counter-clockwise by an angle θ.

Lemma 82. For any $v = (v_1\ v_2) \in \mathbb{R}^2$, if we take θ such that

$$\cos(-\theta) = \frac{v_1}{\|v\|} \text{ and } \sin(-\theta) = \frac{v_2}{\|v\|}$$

Then $R_\theta = \|v\|e_1$. see figure (11) for pictorial.

Now we will the above lemma in our final step of iteration by rotation. Let H^0 be the Hessenburg matrix we found in step 1. We will first turn the vector $v = (v_1\ v_2)^T$ into an $v' = (\overline{v_1}\ 0)^T$ matrix in $\mathbb{R}^2$:

$$H^0 = \begin{pmatrix} v_1 & * & * & * \\ v_2 & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{pmatrix} \rightarrow H^1 = \begin{pmatrix} \overline{v_1} & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{pmatrix}$$

How can we do this? We first find our rotation R_1 in which we turn vector v into a scale of e_1:

$$R_{\theta_1} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \frac{1}{\|v\|} \begin{pmatrix} v_1 & v_2 \\ -v_2 & v_1 \end{pmatrix}$$

And putting this R into the block with identity matrix:

$$R_1 = \left(\begin{array}{c|cc} R_{\theta_1} & \multicolumn{2}{c}{\mathbf{0}} \\ \hline \multirow{2}{*}{$\mathbf{0}$} & 1 & 0 \\ & 0 & 1 \end{array}\right)$$

Then take R_1 multiply with H^0 will give H^1. In short:

$$R_1 H^0 = H^1$$

We continue this method for:

$$H^1 = \begin{matrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{matrix} \rightarrow H^2 = \begin{matrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{matrix}$$

In which we can obtained H^2 from multiplying H^1 with R_2 where

$$R_2 = \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline 0 & R_{\theta_2} & 0 \\ \hline 0 & 0 & 1 \end{array}\right)$$

And then continue:

$$H^2 = \begin{matrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{matrix} \rightarrow H^3 = \begin{matrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{matrix}$$

in which we can otained H^3 by multiplying H^2 with R_3, where:

$$R_3 = \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & R_{\theta_3} \end{array}\right)$$

Notice that H^3 is now already in the upper triangular form, which we could rename it as R^0 and see why is this a QR factorization of H^0. Recall in all of our steps we get:

$$R_1 R_2 R_3 H^0 = H^3 = R^0$$

So

$$H^0 = R_1^{-1} R_2^{-1} R_3^{-1} R^0$$

If we denote $Q = R_1^{-1} R_2^{-1} R_3^{-1}$, since each R_i satisfying $R_i R_i^T = I$, then we get $R_i^{-1} = R_i^T$ (for $i = 1, 2, 3$). So Q is an orthogonal matrix, and R^0 is an upper triangular matrix. Therefore we get:

$$H^0 = QR^0$$

Remark 83. A summary of method:

1. Step 1: Use Householder reflection to get an upper Hessenburg matrix and name it H^0:
$$P_{n-1} \dots P_2 P_1 A P_1 P_2 \dots P_{n-2} = H^0$$
2. Step 2: Apply the QR iteration on H^k by using rotation R_k, and the H^k will converge to an upper triangular matrix H^* from which you can get our eigenvalues.

Example 84. Find the eigenvalues of
$$A = \begin{pmatrix} 4 & 1 & -1 & 2 \\ 1 & 2 & 0 & 1 \\ -2 & 0 & 3 & -2 \\ 2 & 1 & -2 & -1 \end{pmatrix}$$

Recall earilier we already have the Householder matrix H^0 from A:
$$H^0 = \begin{matrix} 4 & 3 & 0 & 0 \\ 3 & \frac{10}{3} & \frac{5}{3} & 0 \\ 0 & \frac{5}{3} & -\frac{33}{25} & -\frac{68}{75} \\ 0 & 0 & -\frac{68}{75} & \frac{149}{75} \end{matrix}$$

So we get $\cos(-\theta) = \frac{4}{5}$ and $\sin(-\theta) = \frac{3}{5}$. Thus we now have our rotation matrix:
$$\begin{pmatrix} \frac{4}{5} & \frac{3}{5} \\ -\frac{3}{5} & \frac{4}{5} \end{pmatrix}$$

and therefore our first R_1 matrix is:
$$R_1 = \begin{pmatrix} \frac{4}{5} & \frac{3}{5} & 0 & 0 \\ -\frac{3}{5} & \frac{4}{5} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Taking the multiplication of matrix we get:
$$R_1 H^0 = H^1 = \begin{pmatrix} 5 & \frac{22}{5} & 1 & 0 \\ 0 & \frac{13}{15} & \frac{4}{3} & 0 \\ 0 & \frac{5}{3} & -\frac{33}{25} & -\frac{68}{75} \\ 0 & 0 & -\frac{68}{75} & \frac{149}{75} \end{pmatrix}$$

Now from this
$$H^1 = \begin{matrix} 5 & \frac{22}{5} & 1 & 0 \\ 0 & \frac{13}{15} & \frac{4}{3} & 0 \\ 0 & \frac{5}{3} & -\frac{33}{25} & -\frac{68}{75} \\ 0 & 0 & -\frac{68}{75} & \frac{149}{75} \end{matrix}$$

we get our $\cos(-\theta) = 0.46$ and $\sin(-\theta) = 0.89$ and so the rotation matrix is:

$$\begin{pmatrix} 0.46 & 0.89 \\ -0.89 & 0.46 \end{pmatrix}$$

and this gives our R_2 matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.46 & 0.89 & 0 \\ 0 & -0.89 & 0.46 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Continue multiplying:

$$R^2 H^1 = R^2 R^1 H^0$$

and eventually we will get:

$$H^* = \begin{pmatrix} 6.2953 & 0 & 0 & 0 \\ 0 & -2.4249 & 0 & 0 \\ 0 & 0 & 2.551 & 0 \\ 0 & 0 & 0 & 0.5412 \end{pmatrix}$$

Notice that we get all zeros on top and below of the diagonal because $A = A^T$ (in general, you will only get an upper triangular matrix).

6 Numerical Linear Algebra: solving Ax=b

6.1 Iterative methods

To solve $Ax = b$, we generate a sequence of x_k such that $\lim_{k\to\infty} x_k = x$:

1. The equivalent linear system:

$$Ax = b \Longleftrightarrow x = Bx + c$$

2. from the equivalent linear system, we have the **fixed point iteration**:

$$x^{(k+1)} = Bx^{(k)} + c$$

6.2 Linear Operating splitting method

We can try to factor $A = P - R$ where P is easier to invert matrix. Then :

$$\begin{aligned} Ax = b &\Longleftrightarrow (P - R)x = b \\ &\Longleftrightarrow Px = Rx + b \\ &\Longleftrightarrow x = P^{-1}Rx + P^{-1}b \end{aligned}$$

Where if we denote $B = P^{-1}R$ and $c = P^{-1}b$ we get this in the form:

$$x = Bx + c$$

and so we can do the iteration method:

$$x^{(k+1)} = Bx^{(k)} + c$$

6.2.1 Jacobi Method

Definition 85 (Jacobi method)**.** The Jacobi method is when we let P to be the diagonal of A. Let $A = (a_{ij})$ and assume $A = L + D + U$ where $D = \text{diag}(a_{11} \dots a_{nn})$ is the diagonal of A, and L is the upper triangular matrix from A, and L is the lower triangular matrix from A.

$$L = \begin{pmatrix} 0 & 0 & 0 \\ a_{21} & 0 & \\ \vdots & \ddots & \\ a_{n1} & a_{n2} & \dots 0 \end{pmatrix}$$

and

$$U = \begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ & \ddots & \vdots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{pmatrix}$$

The Jacobian method is:

$$\begin{aligned} Ax = b &\iff (D + L + U)x = b \\ &\iff Dx = -(L + U)x + b \\ &\iff x = -D^{-1}(L + U)x + D^{-1}b \end{aligned}$$

Therefore the iteration is:

$$x^{(k+1)} = -D^{-1}(L + U)x^{(k)} + D^{-1}b$$

So the coordinate entries are:

$$a_i^{(k+1)} = \frac{1}{a_{ij}} \left[\sum_{j=1, j \neq i}^{n} (-a_{ii} x_j^{(k)}) + b_i \right]$$

Remark 86. We observe the following:

1. Proper row exchanges may be needed to ensure that $a_{ii} \neq 0$ (since we have division by a_{ii}).
2. a suggested stopping criteria might be:

$$\frac{\|x^{(k)} - x^{(k-1)}\|}{\|x^{(k)}\|} < \varepsilon$$

Example 87. Solve this system:

$$\begin{cases} 2x_1 - x_2 & = 1 \\ -x_1 + 2x_2 & = 1 \end{cases}$$

The fixed point is

$$x^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

the matrix

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

and

$$D = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

The Jacobi is then:

$$x_1^{(k+1)} = -\frac{1}{2}(-x_2^{(k)} + 1)$$

and

$$x_2^{(k+1)} = -\frac{1}{2}(-x_1^{(k)} + 1)$$

Applying this we found:

$$x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$x^{(1)} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

$$x^{(2)} = \begin{pmatrix} \frac{3}{4} \\ \frac{3}{4} \end{pmatrix}$$

$$x^{(3)} = \begin{pmatrix} \frac{7}{8} \\ \frac{7}{8} \end{pmatrix} \to \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Definition 88 (strictly diagonal dominant matrix)**.** A is strictly diagonal dominant matrix if

$$|a_{ii}| > \sum_{j \neq 1} |a_{ij}|$$

Definition 89 (infinity norm of a matrix)**.** The infinite norm of a matrix A is defined as:

$$\|A\|_\infty = \sup_{x \in \mathbb{R}^n} \frac{\|Bx\|_\infty}{\|x\|_\infty}$$

Definition 90. If A is a strictly diagonal dominant matrix, then $\|B_J\| < 1$ where $B_J = -D^{-1}(L + U)$. Therefore the Jacobi method will always converge (i.e., it always work!).

6.2.2 Gauss-Seidel Method (GS)

Definition 91 (Gauss-Seidel Method (GS))**.** Suppose $A = (a_{ij})$, and let $A = D + L + U$ as before. But now we do:

$$\begin{aligned} Ax = b &\iff (D + L + U)x = b \\ &\iff (D + L)x = -Ux + b \\ &\iff x = -(D + L)Ux + (L + D)^{-1}b \end{aligned}$$

And the iteration is then obtained by:

$$x^{(k+1)} = -(L + D)^{-1}Ux^{(k)} + (L + D)^{-1}b$$

The iteration matrix of this method is $B_{GS} = -(L+D)^{-1}U$. And the entries in this iteration matrix is:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left[\sum_{j>i}\left(-a_{ij}x_j^{(k)}\right) + \sum_{j<i}\left(-a_{ij}x_j^{(k+1)}\right) + b_i\right]$$

We tend to keep the terms $a_{ii}x_i$ on the left, for instance, with $A \in \mathbb{R}^{3\times 3}$ as in:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \implies \begin{cases} a_{11}x_1^{(k+1)} &= -a_{12}x_2^{(k)} - a_{13}x_3^{(k)} + b_1 \\ a_{22}x_2^{(k+1)} &= -a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} + b_2 \\ a_{33}x_3^{(k+1)} &= -a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)} + b_3 \end{cases}$$

Theorem 92. *If A is strictly diagonal dominant, for any choice of $x^{(0)}$, both the Jacobi and Gauss-Seidel methods converges to the unique solution of $Ax = b$.*

Remark 93. The GS method converges twice as fast as Jacobi.

Example 94. Solve this system:

$$\begin{cases} 2x_1 - x_2 &= 1 \\ -x_1 + 2x_2 &= 1 \end{cases}$$

The fixed point is

$$x^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

the matrix

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

and

$$D = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Recall the Jacobi method B_J has

$$\begin{cases} x_1^{(k+1)} &= \frac{1}{2}x_2^{(k)} - \frac{1}{2} \\ x_2^{(k+1)} &= \frac{1}{2}x_1^{(k)} - \frac{1}{2} \end{cases}$$

If we did our GS method, we would get:

$$x_1^{(k+1)} = \frac{1}{2}x_2^{(k)} + \frac{1}{2}$$

and

$$\begin{aligned} x_2^{(k+1)} &= \frac{1}{2}x_1^{(k+1)} + \frac{1}{2} \\ &= \frac{1}{4}x_2^{(k)} + \frac{3}{4} \end{aligned}$$

Therefore if we start with

$$x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \to x^{(1)} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \to x^{(2)} = \begin{pmatrix} \frac{7}{8} \\ \frac{15}{16} \end{pmatrix} \to x^{(3)} = \begin{pmatrix} \frac{31}{32} \\ \frac{63}{64} \end{pmatrix}$$

6.2.3 When can we find x?

Definition 95 (spectrum and spectral radius)**.** We have these terms:

1. **Spectrum**: The spectrum of a matrix B is $\sigma(B) = \{\lambda | \lambda I - B \text{ is singular}\}$
2. **Spectral Radius** the spectral radius of a matrix B is defined as $\rho(B) = \sup\{|\lambda| : \lambda \in \sigma(B)\}$.

Lemma 96 (Gelfand Lemma)**.** The spectral radius can be computed as:

$$\rho(A) = \lim_{n\to\infty} \|A^n\|^{\frac{1}{n}}$$

for any induced norm $\|.\|$.
For example, we can take the induced (infinite) norm:

$$\|A\|_\infty = \frac{\sup \|Ax\|_\infty}{\|x\|_\infty}$$

Theorem 97. *The following are equivalent:*

1. $\rho(B) < 1$
2. $(I - B)^{-1}$ *exists and equal to* $\sum_{n=1}^{\infty} B^n$
3. *For all* x_0*, the* $x_{n+1} = Bx_n + P^{-1}b$ *will converge to a unique* x^* *where* $x^* = Bx^* + P^{-1}b$*, or* $Ax^* = b$*.(where* $A = P - R$*)*

Proof. We started with showing:

1. Show (1) $\implies$ (2):

   - First step: Let $S_m := \sum_{i=0}^{m} B^i$. Consider the given norm $\|.\|$. Then take any $\varepsilon > 0$ such that $\rho(B) + \varepsilon < 1$. Then by Gelfand Theorem, we get, there exists a k_0 such that for all $k \geq k_0$:

     $$\|B^k\|^{\frac{1}{k}} \leq \rho(B) + \varepsilon < 1$$

     If we denoted $C_0 = \rho(B) + \varepsilon < 1$, then the above (raising to power k) gives:

     $$\|B^k\| \leq C^k < 1, \ \forall k \geq k_0$$

     Then when we take infinite sum we get:

     $$\sum_{i=0}^{\infty} \|B^i\| = \sum_{i=0}^{k_0-1} \|B^i\| + \sum_{i=k_0}^{\infty} \|B^i\|$$
     $$\leq C + \sum_{i=k_0}^{\infty} C_0^i = M < \infty \text{ since } C_0 < 1, \text{ so the sum } \sum_{i=k_0}^{\infty} C_0^i \text{ is finite}$$

- Second step: Now, for all $n \geq m > k_0$:

$$\begin{aligned}\|S_n - S_m\| = \|\sum_{i=m+1}^{n} B^i\| &\leq \sum_{i=m+1}^{n} \|B^i\| \\ &= \|B^{m+1}\|\Big(\sum_{i=0}^{n-m-1} \|B^i\|\Big) \\ &\leq \|B^{m+1}\|\Big(\sum_{i=0}^{\infty} \|B^i\|\Big) \\ &\leq C_0^{m+1}.M\end{aligned}$$

- Step 3: Because $M < \infty$ and $C_0 < 1$ so $C_0^{m+1} \xrightarrow{m\to\infty} 0$. Thus we must have $\|S_n - S_m\| \to 0$.
  By completeness (of $\mathbb{R}$), we must have that $\lim_{n\to\infty} S_n = S^*$ exists.
  Next:

$$\begin{aligned}S_m(I-B) = \Big(\sum_{i=0}^{m} B^i\Big)(I-B) &= \sum_{i=0}^{m} B^i - \sum_{i=0}^{m} B^{i+1} \\ &= B^0 - B^{m+1} = I - B^{m+1} = (I-B)S_m\end{aligned}$$

  In short, $S_m(I-B) = (I-B)S_m = I - B^{m+1}$.
  Since $\lim_{m\to\infty} S_m = S^*$ and $\lim_{m\to\infty} \|B^{m+1}\| \leq \lim_{m\to\infty} C_0^{m+1} = 0$, when passing the above to limit give:

$$S^*(I-B) = I = (I-B)S^*$$

  Therefore we get:

$$(I-B)^{-1} = S^* = \sum_{n=0}^{\infty} B^n$$

  and this is what we needed for part (2).

2. Show (2) $\Longrightarrow$ (3): From (2) we get $\lim_{m\to\infty} S_m = S^*$, and $\lim_{m\to\infty} \|B^m\| = 0$, so:

$$\begin{aligned}x_m &= Bx_{m-1} + P^{-1}b = B(Bx_{m-2} + P^{-1}b) + P^{-1}b \\ &= \ldots = B^m x_0 + S_m P^{-1}b \text{ , by induction}\end{aligned}$$

   Therefore

$$\begin{aligned}\|x_m - x_n\| &= \|(B^m - B^n)x_0 + (S_m - S_n)P^{-1}b\| \\ &\leq \|B^m - B^n\|\|x_0\| + \|S_m - S_n\|P^{-1}b\| \to 0 \\ &\quad \text{(since } \|B^m - B^n\| \to 0 \text{ and } \|S_{m-1} - S_n\| \to 0)\end{aligned}$$

   Therefore by completeness, x_n must converge, so $\lim_{n\to\infty} x_n = x^*$. In particular, we have:

$$\begin{aligned}x^* &= (\lim_{n\to\infty} B^n)x_0 + (\lim_{n\to\infty} S_{n-1})P^{-1}b \\ &= S^*P^{-1}b \\ &\quad \text{(since } \lim_{n\to\infty} B^m = 0 \text{ and } \lim_{n\to\infty} S_{n-1} = S^*) \\ &= (I-B)^{-1}P^{-1}b\end{aligned}$$

In short we get:

$$\begin{aligned}(I-B)x^* &= P^{-1}b\\ x^* - Bx^* &= P^{-1}b\\ x^* &= Bx^* + P^{-1}b\\ Px^* &= PBx^* + b\\ Px^* &= Rx^* + b \text{ where } B = P^{-1}R \text{ and } A = P - R\end{aligned}$$

And this showed part (3).

3. Show (3) $\Longrightarrow$ (1): For all x_0, let us denote $x_0 = v + x^*$ and we have:

$$\begin{aligned}x_n - x^* &= (Bx_{n-1} + P^{-1}b) - (Bx^* + P^{-1}b)\\ &= B(x_{n-1} - x^*) = \ldots = B^n(x_0 - x^*) = B^n v\end{aligned}$$

In short we get $B^n v = x_n - x^* \xrightarrow{n\to\infty} 0$. Thus (by Gelfand Theorem, $\rho(B) = \lim_{n\to\infty} \|B^n\|^{\frac{1}{n}} = \lim_{n\to\infty} \left(\sup_{\|x\|=1} \|B^n x\| \right)^{\frac{1}{n}}$, there is an eigenvalue of B, so $\rho(B)$ which is the supremum of the set of all eigenvalues of B exists) there must exists a w such that:

$$Bw = \pm\rho(B)w \text{ , with } \|w\| = 1$$

Then raising this to n power:

$$B^n w = \pm(\rho(B))^n w \xrightarrow{n\to\infty} 0$$

Since $w \neq 0$, we must conclude that:

$$(\rho(B))^n \xrightarrow{n\to\infty} 0$$

And therefore we must have $\rho(B) < 1$.

Thus we show (1) $\Longrightarrow$ (2) $\Longrightarrow$ (3) $\Longrightarrow$ (1). □

6.3 Conjugate Gradient Method

6.3.1 Gershgorin Disc Method

Theorem 98 (Gershogorin Disc Theorem)**.** *For all $\lambda \in \sigma(A)$, there exists an i such that:*

$$|a_{ii} - \lambda| \leq \sum_{j\neq i} |a_{ij}|$$

Proof. The existence of λ means that there is such an eigenvector v such that $Av = \lambda v$ and $\|v\| = 1$. There exists an i such that $v_i = 1$ (i coordinate of v) and $|v_j| \leq 1$ for all

$i \neq j$. Then:

$$\begin{aligned}
(Av)_i &= (\lambda v)_i \\
\sum a_{ij} v_j &= \lambda v_i \\
(\lambda - a_i i) v_i &= \sum_{j \neq i} a_{ij} v_j \\
|\lambda - a_{ii}||v_i| &\leq \sum_{j \neq i} |a_{ij}||v_j| \\
&\qquad \text{since } |v_i| = 1 \text{ and } |v_j| < 1 \\
|\lambda - a_{ii}| &< \sum_{j \neq i} |a_{ij}|
\end{aligned}$$

This completed the proof. □

Corollary 99. *If A is **strictly diagonal dominant (SDD)**, then:*

$$|\lambda_j| < 1 \text{ and } |\lambda_{GS}| < 1$$

Proof. Recall **SDD** mean that $|a_{ii}| > \sum_{j \neq 1} |a_{ij}|$. The Jacobi method matrix is B_J with $b_{ii} = 0$ and $b_{ij} = \frac{a_{ij}}{a_{ii}}$. Apply the Gershgorin Disc Theorem to B_J gives: for all $\lambda_J \in \sigma(B_J)$, there exists an i such that:

$$\begin{aligned}
|\lambda_J| = |b_{ii} - \lambda_J| \leq \sum_{j \neq i} |b_{ij}| = \sum_{j \neq i} \frac{|a_{ij}|}{|a_{ii}|} < 1 \\
\text{since } A \text{ is SDD}
\end{aligned}$$

Now for the GS method, $B_{GS} = (D - L)^{-1} U$ satisfying:

$$\begin{aligned}
\det(\lambda_{GS} I - (D - L)^{-1} U) = 0 \\
\det(\lambda_{GS}(D - L) - U) = 0
\end{aligned}$$

Therefore 0 is an eigenvalue of the matrix $\lambda_{GS}(D - L)$. So we use Gershogorin Disc Theorem again and get:

$$\begin{aligned}
|\lambda_{GS} a_{ii} - 0| &\leq \sum_{j<i} |a_{ij}||\lambda_{GS}| + \sum_{j>i} |a_{ij}| \\
\implies |\lambda_{GS}| \left(1 - \sum_{j<i} \frac{|a_{ij}|}{|a_{ii}|}\right) &\leq \sum_{j>i} |a_{ij}| < \left(1 - \sum_{j<i} \frac{|a_{ij}|}{|a_{ii}|}\right)
\end{aligned}$$

Dividing by $\left(1 - \sum_{j<i} \frac{|a_{ij}|}{|a_{ii}|}\right)$ gives:

$$|\lambda_{GS}| < 1$$

And this completed the proof. □

6.3.2 Conjugate Gradient Method to solve Ax=b

Question 100. In this subsection, we suppose A is a **Symmetric positive definite (SPD)**, and we would like to solve $Ax = b$.
Idea: Find an A-orthogonal basis $\{p_k\}$ and expand the root x^* with respect to this basis.

Definition 101 (A-orthogonal)**.** A set of vectors $\{p_k\}$ is said to be A**-orthogonal** if:

$$\langle p_k, Ap_l \rangle = 0 \text{ if } k \neq l$$

Algorithm 102. Let $\{p_k\}$ be such a basis. And suppose x^* is a solution, that is, $Ax^* = b$. Then we can expand x^* with respect to this basis p_k and get:

$$x^* = \sum_{k=0}^{n-1} \alpha_k^* p_k$$

Then:

$$\begin{aligned} \langle p_i, b \rangle = \langle p_i, Ax^* \rangle &= \sum_{k=0}^{n-1} \alpha_k^* \langle p_i, Ap_k \rangle \\ &= \alpha_i^* \langle p_i, Ap_i \rangle \end{aligned}$$

Therefore we see the

$$\alpha_i^* = \frac{\langle p_i, b \rangle}{\langle p_i, Ap_i \rangle}$$

And the solution is

$$x^* = \sum_{k=0}^{n-1} \frac{\langle p_k, b \rangle}{\langle p_k, Ap_k \rangle} p_k$$

which is the formulae to find x^*.

Question 103. How do we find this basis $\{p_k\}$? The algorithm to find this basis is as followed:
Consider the function

$$f(x) = \frac{1}{2}\langle x, Ax \rangle - \langle b, x \rangle$$

Its gradient is:

$$\nabla f = Ax - b = 0$$

We set $\nabla f = 0$ to find the local min of f.
If we start at x_0, we take

$$p_0 = b - Ax_0 = -\nabla f(x_0)$$

Let r_k be the **residue at the k-th step**:

$$r_k = b - Ax_k$$

We will get $\{p_k\}$ by Gram-Schmidt process:

$$p_k = r_k - \sum_{i<k} \frac{\langle Ap_i, r_k \rangle}{\langle Ap_i, p_i \rangle} p_i$$

If we take $x_{k+1} = x_k + \alpha_k p_k$, and consider a function of α_k:

$$g(\alpha_k) = f(x_k + \alpha_k p_k)$$

The derivative with respect to α_k is zero will solve for α_k:

$$g'(\alpha_k) = 0 \Longrightarrow \alpha_k = \frac{\langle p_k, r_k \rangle}{\langle Ap_k, p_k \rangle}$$

Theorem 104. *Let $\{p_k\}$ be A-orthogonal, that is, $\langle p_k, Ap_l \rangle = 0$ if $k \neq l$. Starting from x_0 and let:*

$$\begin{aligned} r_k &= b - Ax_k \\ x_{k+1} &= x_k + \alpha_k p_k \\ \alpha_k &= \frac{\langle p_k, r_k \rangle}{\langle p_k, Ap_k \rangle} \end{aligned}$$

Then we will have $x_n = x^*$ where $Ax^* = b$. Exactly n steps and we can get our x^*.

Proof. Since $x_{k+1} - x_k = \alpha_k p_k$, we get (by telescoping):

$$x_n = \sum_{k=0}^{n-1} \alpha_k p_k$$

Recall for x^*, we have:

$$x^* = \sum_{k=0}^{n-1} \alpha_k^* p_k \text{ , where } \alpha_k^* = \frac{\langle p_k, b \rangle}{\langle p_k, Ap_k \rangle}$$

Now it will be done if we can check that $\langle p_k, r_k \rangle = \langle p_k, b \rangle$. This is true because:

$$\begin{aligned} \langle p_k, r_k \rangle &= \langle p_k, b - Ax_k \rangle \\ &= \langle p_k, b \rangle - \langle p_k, Ax_k \rangle \\ &= \langle p_k, b \rangle \end{aligned}$$

The $\langle p_k, Ax_k \rangle = 0$ because for any $x_k \in \text{Span}\{p_0, p_1, \ldots, p_{k-1}\}$, the inner product $\langle p_k, Ax_k \rangle$ is zero by the definition of A-orthogonality. □

Remark 105. The steps are:

$$x_0 \to r_0, p_0 \to x_1 \to r_1, p_1 \to x_2 \to r_2, p_2$$

That is, you get the p_i at each step.

Remark 106. This computation is expensive, however it is still more efficient than SD (steepest Descent) because we find local min instead of global min.

Algorithm 107. The pseudocode for Conjugate Gradient Method is found in figure (12).

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
$\alpha_k := \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
$\beta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result

Figure 12: Conjugate Gradient Pseudocode

6.4 Singular Value Decomposition for non-square matrix

The goal is for A an $m \times n$ matrix, we would like to decompose A into:

$$A_{m\times n} = U_{m\times m}S_{m\times n}V^T_{n\times n}$$

where S is a diagonal matrix, $UU^T = I_m$ and $VV^T = I_n$ (pseudo inverse).
Note that AA^T is a non-negative $n \times n$ matrix. We can order the eigenvalues of AA^T as:

$$s_1^2 \geq s_2^2 \geq \ldots s_k^2 \geq s_{k+1} = \ldots = s_n = 0$$

:

$$S = \begin{pmatrix} s_1 & 0 & \ldots & 0 \\ 0 & s_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & \ldots & 0 \\ \vdots & & & \vdots \\ 0 & \ldots & \ldots & 0 \end{pmatrix}$$

Definition 108. The s_i are called the **singular value of** A. That is, we need to find $\{v_i \in \mathbb{R}^n : \|v_i\| = 1\}$ such that $A^TAv_i = s_i^2v_i$.

Question 109. How to find the v_i? Suppose $u_i = \frac{Av_i}{s_i} \in \mathbb{R}^m$, then:

$$Av_i = s_iu_i$$
$$A^Tu_i = \frac{1}{s_i}A^TAv_i = \frac{1}{s_i}(s_i^2v_i) = s_iv_i$$

So the **Singular value decomposition SVD** will get $A_{m\times n} = U_{m\times m}S_{m\times n}V^T_{n\times n}$ where:

$$U = (u_1 \ \ldots u_m) \text{ and } V^T = \begin{pmatrix} v_1^T \\ \vdots \\ v_n^T \end{pmatrix}$$

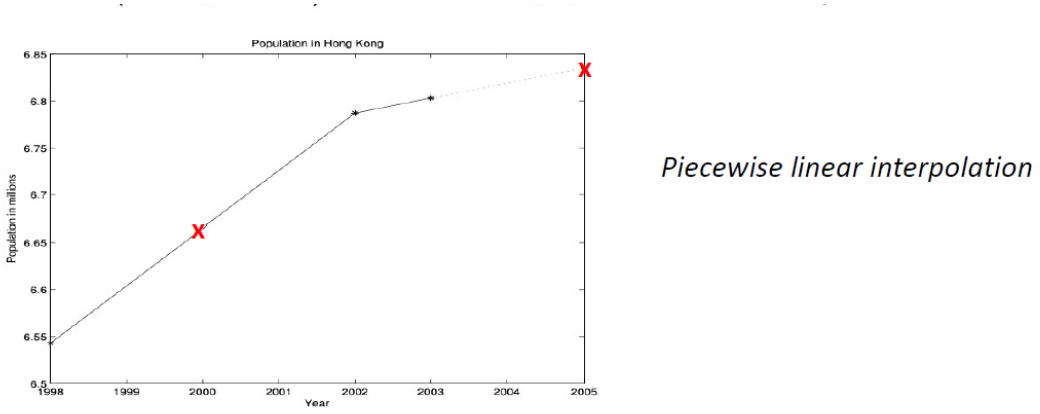


Figure 13: Piecewise linear interpolation

and

$$S = \begin{pmatrix} s_1 & 0 & \dots & 0 \\ 0 & s_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 \\ \vdots & & & \vdots \\ 0 & \dots & \dots & 0 \end{pmatrix}$$

7 Day 10: Interpolation and Extrapolation

Goal: The idea is to find functions (such as polynomials) that passes through (or "approximate") a set of data points.

1. **Interpolation**: "inside" the data.
2. **Extrapolation**: "outside" the data.

Example 110. According to the Hong Kong Census and Statistic Department, the population of Hong Kong was:

Year	1998	2002	2003
Population (in millions)	6.5437	6.787	6.8031

The questions that we would most likely be asking are:

1. (Interpolation) What was the population in the year 2000?
2. (Extrapolation) What will be the population in the year 2005?

A method of piecewise linear interpolation is in figure (13)

7.1 Polynomial Interpolation

Idea: Suppose given $n+1$ data points, we would like to find a degree n polynomial that passes through all points.

Theorem 111 (Weierstrass Approximation Theorem)**.** *Suppose $f \in C[a,b]$. For any $\varepsilon > 0$, there exists a polynomial $P(x)$ such that:*

$$|f(x) - P(x)| < \varepsilon \text{ for all } x \in [a,b].$$

Then we have the Taylor's series:

Definition 112 (Taylor's series)**.** We can find

$$T(x) = \sum_{i=0}^{n} \frac{f^{(i)}(x)}{i!} h^i$$

to approximate f

However, this method also have:

1. The good: local, as high order as possible
2. The bad: local, so it require smoothness, and we need to compute derivatives.

The algorithm is as followed:

Algorithm 113. Suppose we have $n+1$ data points $\{(x_i, f_i)\}_{i=0}^n$ where $x_i \neq x_j$. Let

$$p(x) = \alpha_0 + \alpha_1 + \alpha_2 x^2 + \ldots + \alpha_n x^n$$

such that

$$p(x_i) = f_i, \text{ for } i = 0, 1, \ldots, n$$

Then:

$$\begin{bmatrix} 1 & x_0 & x_0^2 \ldots & x_0^n \\ 1 & x_1 & x_1^2 \ldots & x_1^n \\ 1 & x_2 & x_2^2 \ldots & x_2^n \\ \vdots & \vdots & \vdots \ddots & \vdots \\ 1 & x_n & x_n^2 \ldots & x_n^n \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_n \end{bmatrix}$$

Unfortunately, the Vandermonde matrix V where

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 \ldots & x_0^n \\ 1 & x_1 & x_1^2 \ldots & x_1^n \\ 1 & x_2 & x_2^2 \ldots & x_2^n \\ \vdots & \vdots & \vdots \ddots & \vdots \\ 1 & x_n & x_n^2 \ldots & x_n^n \end{bmatrix}$$

is unstable and we should not use this.

Instead, we have the Lagrange Interpolation:

7.2 Lagrange Interpolation

Definition 114 (Lagrange Interpolation)**.** The n-th **Lagrange Interpolating polynomial** through **nodes** $\{(x_i, f_i)_{i=0}^n\}$ is:

$$L_{n,k}(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{k-1})\dots(x-x_n)}{(x_k-x_0)(x_k-x_1)\dots(x_k-x_{k-1})\dots(x_k-x_n)} = \prod_{i=0, i\neq k}^{n} \frac{x-x_i}{x_k-x_i}$$

These $L_{n,k}$ satisfy:

$$L_{n,k}(x_j) = \begin{cases} 0 & \text{if } j \neq k \\ 1 & \text{if } j = k \end{cases}$$

Theorem 115. *If $x_0, x_1, \dots, x_n$ distinct, f is a function whose values are given at these points, then a unique polynomial $P(x)$ of degree at most n exists with*

$$f(x_k) = P(x_k) \text{ for all } k = 0, 1, \dots, n$$

This polynomial is given as:

$$P(x) = f(x_0)L_{n,0}(x) + \dots + f(x_n)L_{n,n}(x_n) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x)$$

Example 116. Consider $x_0 = 2$, $x_1 = 2.75$, and $x_2 = 4$. Find $P_2(x)$ for $f(x) = \frac{1}{x}$.
First we need to find the three L:

1. The first term L_0:

$$L_0 = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-2.75)(x-4)}{(-0.75)(-2)} = \frac{2}{3}(x-2.75)(x-4)$$

2. The second term L_1:

$$L_1 = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-2)(x-4)}{(0.75)(-1.25)} = -\frac{16}{15}(x-2)(x-4)$$

3. The third term L_2:

$$L_2 = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-2)(x-2.75)}{(2)(1.25)} = -\frac{2}{5}(x-2)(x-2.75)$$

Then

$$P_2(x) = \sum_{i=0}^{2} f(x_i)L_i(x) = \frac{1}{2}L_0(x) + \frac{1}{2.75}L_1(x) + \frac{1}{4}L_2(x) = \dots = \frac{1}{22}x^2 - \frac{35}{88}x + \frac{49}{44}$$

Theorem 117. *Suppose $x_0, x_1, \dots, x_n$ are $n+1$ distinct numbers in the interval $[a, b]$ and $f \in C^{m+1}[a, b]$. Then for each $x \in [a, b]$, there exists an $\xi(x) \in [a, b]$ which depends on x continuously on each interval (x_i, x_{i+1}) such that:*

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x)1)\dots(x-x_n)$$

where $P(x)$ is the Lagrange interpolation polynomial of degree n such that $f(x_k) = P(x_k)$ for all $k = 0, 1, \dots, n$.

7.3 Newton Interpolation: Forwarded Divided Difference

Goal: Given $\{(x_i, f_i)\}_{i=0}^n$ we need to find a_i such that:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_n(x - x_0)(x - x_1)\ldots(x - x_{n-1})$$

interpolate the data points.

Definition 118 (Forward divided difference)**.** The **forward divided difference** is defined as followed:

1. The 0-th divided difference of f wrt x_i defined as $f[x_i] = F(x_i)$.

2. 1-st divided difference of f w.r.t x_i, x_{i+1} defined as

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i} = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

3. k-th divided difference of f w.r.t. $x_i, \ldots, x_{i+k}$ is defined as:

$$f[x_i, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

In particular,

$$f[x_0, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}$$

Definition 119 (Newton Interpolation)**.** The **Newton Interpolation** is the polynomial:

$$\begin{aligned} P(x) &= f[x_0] + f[x_0, x_1](x - x_0) + \ldots + f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1)\ldots(x - x_n) \\ &= f[x_0] + \sum_{j=1}^{n} f[x_0, x_1, \ldots, x_j](\prod_{i=0}^{j-1}(x - x_j)) \end{aligned}$$

Example 120. $f(x) = \frac{1}{x}$ and $x_0 = 1$, $x_1 = 2$ and $x_2 = 4$.
It might be easier to draw it out and calculate like a butterfly shape (see figure (14))
We begin with this table:

x_i	1	2	4
$f[x_i] = f(x_i)$	1	$\frac{1}{2}$	$\frac{1}{4}$

Next we calculate:

x_i	(1 and 2)	(2 and 4)
$f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - x_i)}{x_{i+1} - x_i}$	$\frac{\frac{1}{2} - 1}{2-1} = -\frac{1}{2}$	$\frac{\frac{1}{4} - \frac{1}{2}}{4-2} = -\frac{1}{8}$

And lastly,

$$f[x_1, x_2, x_3] = \frac{-\frac{1}{8} - \frac{-1}{2}}{4 - 1} = \frac{1}{8}$$

Therefore our Newton Interpolation polynomial is:

$$\begin{aligned} P(x) &= f[x_0] + f[x_0, x_1] + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &= 1 + -\frac{1}{2}(x - 1) + \frac{1}{8}(x - 1)(x - 2) \\ &= \frac{x^2}{8} - \frac{7}{8}x + \frac{7}{4} \end{aligned}$$

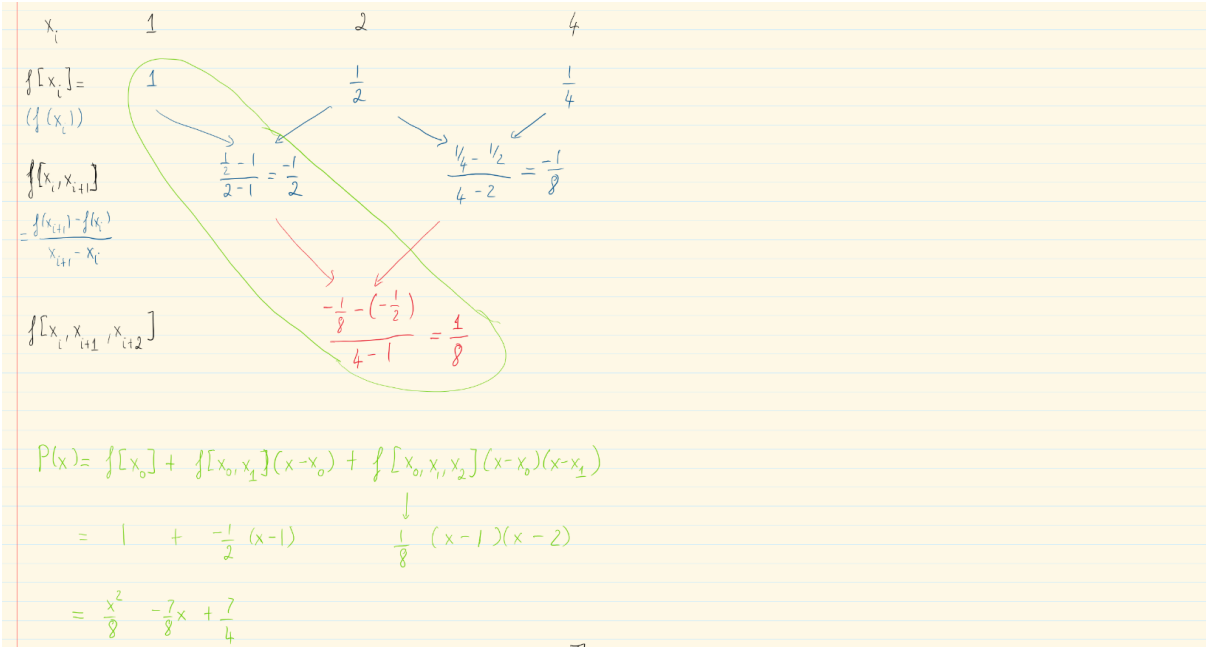


Figure 14: Newton interpolation example

7.4 Hermite Interpolation

Definition 121 (osculating polynomial)**.** We let $x_0, x_1, \ldots, x_n$ be $n+1$ distinct numbers in $[a, b]$ and also let $m_0, m_1, m_2, \ldots, m_n$ be nonnegative integers. Suppose that $f \in C^m[a, b]$ where $m = \max_{0 \le i \le n} m_i$. The **osculating polynomial** approximating f is the polynomial $P(x)$ of **least degree** satisfying:

$$\frac{d^k P(x_i)}{dx^k} = \frac{d^k f(x_i)}{dx^k} \text{ for } k = 0, 1, \ldots, m_i \text{ with } i = 0, 1, \ldots, n$$

1. If $n = 0$, then $P(x)$ is the m_0-th Taylor polynomial for f at x_0.
2. If $m_i = 0$ for each i, then $P(x)$ is the n-th Lagrange polynomial interpolating f at $x_0, x_1, \ldots, x_n$.
3. If $m_i = 1$ for each i, then $P(x)$ is the **Hermite Polynomial** satisfying:

$$P(x_i) = f(x_i), \; P'(x_i) = f'(x_i), \; i = 0, 1, 2, \ldots, n$$

Theorem 122 (Hermite Polynomial approximation)**.** *Let $f \in C^1[a, b]$, $x_0, x_1, \ldots, x_n \in [a, b]$ be distinct. Then the **unique** polynomial of least degree that agree with f and f' at $x_0, x_1, \ldots, x_n$ is the **Hermite polynomial** of degree $2n + 1$ given by:*

$$H_{2n+1} = \sum_{j=0}^{n} f(x_j) H_{n,j}(x) + \sum_{j=0}^{n} f'(x_j) \hat{H}_{n,j}(x)$$

where

$$H_{n,j} = [1 - 2(x - x_j) L'_{n,j}(x_j)] L^2_{n,j}(x)$$

and

$$\hat{H}_{n,j}(x) = (x - x_j)L_{n,j}^2(x)$$

The theorem gives a way to find the polynomial, but it's hard to implement. Therefore we will use this:

Definition 123 (Divided Difference Formula- Newton Forwarded Divided difference formula)**.** Recall the Newton Forwarded Divided difference formula:

$$P(x) = f[x_0] + \sum_{j=1}^{n} f[x_0, x_1, \ldots, x_j](x - x_0)(x - x_1)\ldots(x - x_{j-1})$$

If we defined $z_0, z_1, z_2, \ldots, z_{2n}, z_{2n+1}$ by $z_{2i} = z_{2i+1} = x_i$, i.e.:

x_0	x_0	x_1	x_1	x_2	$\ldots$	x_n	x_n
z_0	z_1	z_2	z_3	z_4	$\ldots$	z_{2n}	z_{2n+1}

We also define:

$$f[z_{2i}, z_{2i+1}] = f'(z_{2i}) = f'(x_i)$$

Then the **Hermite interpolating polynomial** will be:

$$H_{2n+1}(x) = \sum_{k=1}^{2n+1} f[z_0, z_1, \ldots, z_k](x - z_0)\ldots(x - z_{k-1})$$

Example 124. Let $f(x) = \frac{1}{x}$, $x_0 = 1$, $x_1 = 2$, and $x_2 = 4$. Find $H_5(x)$. We calculate:

$$f(x) = \frac{1}{x}: \ f(1) = 1, \ f(2) = \frac{1}{2}, \ f(4) = \frac{1}{4}$$

$$f'(x) = -\frac{1}{x^2}: \ f'(1) = -1, \ , f'(2) = -\frac{1}{4}, \ f'(4) = -\frac{1}{16}$$

We started with the following table:

x_i	1		2		4	
z_i	1	1	2	2	4	4
$f[z_i] = f(z_i)$	1	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$

Next we get:

$f[z_i]$	(1 and 1)	1 and $\frac{1}{2}$)	($\frac{1}{2}$ and $\frac{1}{2}$)	($\frac{1}{2}$ and $\frac{1}{4}$)	($\frac{1}{4}$ and $\frac{1}{4}$)
$f[z_i, z_{i+1}]$	$f'(1) = -1$	$\frac{\frac{1}{2}-1}{2-1} = -\frac{1}{2}$	$f'(2) = -\frac{1}{4}$	$-\frac{1}{8}$	$f'(4) = -\frac{1}{16}$

Next we get:

$f[z_i, z_{i+1}]$	(-1 and $-\frac{1}{2}$)	($-\frac{1}{2}$ and $-\frac{1}{4}$)	($-\frac{1}{4}$ and -$\frac{1}{8}$)	($-\frac{1}{8}$ and $-\frac{1}{16}$)
$f[z_i, z_{i+1}, z_{i+2}]$	$\frac{(-\frac{1}{2})-(-1)}{2-1} = \frac{1}{2}$	$\frac{-\frac{1}{4}-(-\frac{1}{2})}{2-1} = \frac{1}{4}$	$\frac{-(\frac{1}{8})-(-\frac{1}{4})}{4-2} = \frac{1}{16}$	$\frac{(-\frac{1}{16})-(-\frac{1}{8})}{4-2} = \frac{1}{32}$

Next we get:

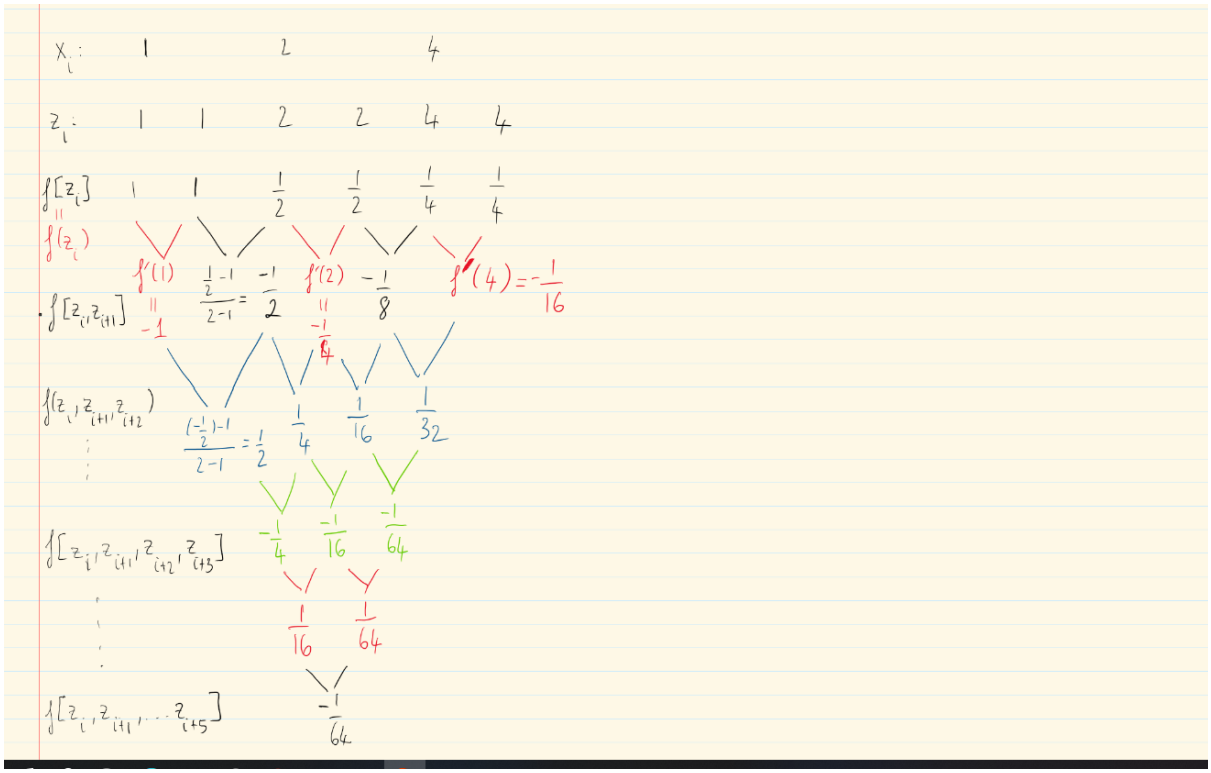


Figure 15: Hermite polynomial Forward Divided

$f[z_i, z_{i+1}, z_{i+2}]$	$(\frac{1}{2}$ and $\frac{1}{4})$	$(\frac{1}{4}$ and $\frac{1}{16})$	$(\frac{1}{16}$ and $\frac{1}{32})$
$f[z_i, z_{i+1}, z_{i+2}, z_{i+3}]$	$\frac{\frac{1}{4}-\frac{1}{2}}{2-1} = -\frac{1}{4}$	$\frac{\frac{1}{16}-\frac{1}{4}}{4-1} = -\frac{1}{16}$	$\frac{\frac{1}{32}-\frac{1}{16}}{4-2} = -\frac{1}{64}$

And next:

$f[z_i, z_{i+1}, z_{i+2}, z_{i+3}]$	$(-\frac{1}{4}$ and $-\frac{1}{16})$	$(-\frac{1}{16}$ and $-\frac{1}{64})$
$f[z_i, z_{i+1}, z_{i+2}, z_{i+3}, z_{i+4}]$	$\frac{-\frac{1}{16}-(-\frac{1}{4})}{4-1} = \frac{1}{16}$	$\frac{-\frac{1}{64}-(-\frac{1}{16})}{4-1} = \frac{1}{64}$

And finally:

$$f[z_i, z_{i+1}, z_{i+2}, z_{i+3}, z_{i+4}, z_{i+5}] = \frac{\frac{1}{64} - \frac{1}{16}}{4-1} = -\frac{1}{64}$$

Therefore:

$$\begin{aligned} H_5(x) &= f[1] + f[1,1](x-1) + f[1,1,2](x-1)^2 + f[1,1,2,2](x-1)^2(x-2) \\ &\quad + f[1,1,2,2,4](x-1)^2(x-2)^2 + f[1,1,2,2,4,4](x-1)^2(x-2)^2(x-4) \\ &= -\frac{1}{64}x^5 + \frac{7}{32}x^4 - \frac{77}{64}x^3 + \frac{53}{16}x^2 - \frac{77}{16}x + \frac{7}{2} \end{aligned}$$

See figure (15) for illustration.

7.5 Cubic Spline Interpolation

Definition 125 (cubic spline interpolation)**.** Let f be defined on $[a, b]$ and $a = x_0 < x_1 < \ldots < x_n = b$. **A cubic spline interpolant** S for f is a function satisfying:

1. $S(x)$ is a cubic polynomial on each subinterval $[x_i, x_{i+1}]$ for $i = 0, \ldots, n-1$ and that letting $S_i(x) = S(x)\Big|_{x \in [x_i, x_{i+1}]}$.
2. The $4n-2$ equations:
   (a) $S_j(x_j) = f(x_j)$, $S_j(x_{j+1}) = f(x_{j+1})$ for $j = 0, 1, \ldots, n-1$.
   (b) $S_j(x_{j+1}) = S_{j+1}(x_{j+1})$ for $j = 0, 1, \ldots, n-2$.
   (c) $S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$ for $j = 0, 1, \ldots, n-2$
   (d) $S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$ for $j = 0, 1, \ldots, n-2$
3. One of the following boundary conditions is satisfied:
   (a) Natural (free) boundary: $S''(x_0) = S''(x_n) = 0$.
   (b) Clamped boundary: $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$

Note the equation for each j:

$$S_j(x) = a_j + b_j(x - x_j) + C_j(x - x_j)^2 + d_j(x - x_j)^3$$

Remark 126. There are $4n$ equations, and $4n$ variables.

Example 127. Find a natural cubic spline passing through $(-1, 0)$, $(0, 0)$ and $(1, 1)$. Now first note that we have 2 intervals $[-1, 0]$ and $[0, 1]$. We need to find:

1. $S_0 = a_0 + b_0(x+1) + c_0(x+1)^2 + d_0(x+1)^3$ for $x \in [-1, 0]$.
2. $S_1 = a_1 + b_1(x+1) + c_1(x+1)^2 + d_1(x+1)^3$ for $x \in [0, 1]$.

We have these conditions:

1. $S_j(x_j) = S_{j+1}(x_j) = f(x_j) = 0 = f(-1)$:
   $a_0 = 0$, $a_0 + b_0 + c_0 + d_0 = 0$, $a_1 = 0$, and $a_1 + b_1 + c_1 + d_1 = 1$.
2. $S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$ gives $S_0'(0) = S_1'(0)$. So $b_0 + 2c_0 + 3d_0 = b_1$.
3. $S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$ gives $2c_0 + 6d_1 = 2c_1$.
4. Natural boundary condition: $S_0''(-1) = 0$ so $2c_0 = 0$, and $S_1''(1) = 0$ so $2c_1 + 6d_1 = 0$.

Now solve for all the a_i, b_i, c_i, d_i gives:

$$S(x) = \begin{cases} -\frac{1}{4}(x+1) + \frac{1}{4}(x+1)^3 & , x \in [-1, 0] \\ \frac{1}{2}x + \frac{3}{4}x^2 - \frac{1}{4}x^3 & , x \in [0, 1] \end{cases}$$

8 Numerical Integration and Differentiation

8.1 Quadrature Rule (Integration)

Definition 128 (Quadrature Formula)**.** Suppose $\int_a^b f(x)dx \approx \sum_{j=0}^n a_j f(x_j)$. How to define a_j? Given $n+1$ distinct points $x_0, x_1, \ldots, x_n \in [a,b]$, we consider the Lagrange interpolating polynomial:

$$\begin{aligned} f(x) &= \sum_{j=0}^n f(x_j)L_j(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\prod_{j=0}^n (x-x_j) \\ &= P_n(x) + \text{Truncation error} \end{aligned}$$

Then if we take the integration:

$$\begin{aligned} \int_a^b f(x)dx &= \sum_{j=0}^n f(x_j)\int_a^b L_j(x)dx + \frac{1}{(n+1)!}\int_a^b \prod_{j=0}^n (x-x_j) f^{(n+1)}(\xi(x))dx \\ &= \sum_{j=0}^n f(x_j)a_j + \text{Error } E(f) \end{aligned}$$

where

$$a_j = \int_a^b L_j(x)dx$$

and the Error term is:

$$E(f) = \frac{1}{(n+1)!}\int_a^b \prod_{j=0}^n (x-x_j) f^{(n+1)}(\xi(x))dx$$

Definition 129 (Trapezoidal Rule)**.** The integral is:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}[f(a)+f(b)]$$

Remark 130. Why is this approximation in Trapezoidal rule? Consider $x_0 = a$, $x_1 = b$ and then Lagrange interpolation give:

$$L(x) = \frac{x-b}{a-b}f(a) + \frac{x-a}{b-a}f(b)$$

So

$$\begin{aligned} \int_a^b f(x)dx &= f(a)\int_a^b \frac{x-b}{a-b}dx + f(b)\int_a^b \frac{x-a}{b-a}dx \\ &= \frac{b-a}{2}\Big[f(a)+f(b)\Big] \end{aligned}$$

Definition 131 (Simpson's Rule)**.** The integral is

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\Big[f(a) + 4f(\frac{a+b}{2}) + f(b)\Big]$$

Remark 132. Why is this approximation in Simpson rule? Take $x_0 = a$, $x_1 = \frac{a+b}{2}$ and $x_2 = b$. Then denote $h = \frac{b-a}{2}$, so we get $x_0 = x_1 - h$ and $x_2 = x_1 + h$. In particular, we get this Lagrange Interpolation:

$$L(x) = f(x_0)\frac{(x-x_1)(x-x_1-h)}{(-h)(-2h)} + f(x_1)\frac{(x-x_1-h)(x-x_1+h)}{(-h)h} + f(x_2)\frac{(x-x_1+h)(x-x_1)}{h(2h)}$$

If we rewriting $\frac{(x-x_1)}{(h)}$ as $\frac{x}{h}$ with a change of variable, we see that:

$$\begin{aligned}\int_a^b f &= f(x_0)h\int_{-1}^{1}\frac{x(x-1)}{2}dx + f(x_1)h\int_{-1}^{1}\frac{(x-1)(x+1)}{-1}dx + f(x_2)h\int_{-1}^{1}\frac{x(x+1)}{2}dx \\ &= \frac{b-a}{6}\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right]\end{aligned}$$

8.1.1 Degree of Precision

Definition 133 (Degree of precision (DOP))**.** The **degree of precision** of a quadrature formula

$$\int_a^b f \approx \sum_{j=0}^{n} a_j f_j$$

is **the largest** integer n usch that the formula is exact for x^k with $k = 0, 1, \ldots, n$, that is:

$$\int_a^b f = \sum_{j=0}^{n} a_j f_j = \sum_{j=0}^{n} a_j x_j^k$$

Example 134. Find A and B such that $\int_0^1 f dx \approx Af(0) + Bf(1)$ has DOP (degree of precision) ≥ 1.
The strategy is that we need to check equality for $x^0 = 1$ and $x^1 = x$:

1. For $x^0 = 1$: $\int_0^1 dx = A + B$ so $A + B = 1$.
2. For $x^1 = x$: $\int_0^1 x dx = A(0) + B(1) = B$ so $\frac{1}{2} = B$.

Therefore we get $A = B = \frac{1}{2}$. We see that this is Trapezoid rule:

$$\int_0^1 = \frac{1}{2}[f(0) + f(1)]$$

Why $n = 1$ is the smallest, because if we check for x^2 we will not get equality:

$$\frac{1}{3} = \int_0^1 x^2 dx \neq \frac{1}{2}(0) + \frac{1}{2}(1) = \frac{1}{2}$$

So it does not work for x^2, therefore 1 is the highest degree. We could also conclude that Trapzoid rule has DOP 1.

Example 135. Find A, B and C such that $\int_{-1}^1 f dx \approx Af(-1) + Bf(0) + Cf(1)$ has DOP ≥ 2.
We need to check for $1, x, x^2$:

1. Check for 1: $2 = \int_{-1}^{1} 1dx = A + B + C$
2. Check for x: $0 = \int_{-1}^{1} xdx = -A + C$
3. Check for x^2: $\frac{2}{3} = \int_{-1}^{1} x^2dx = A + C$

From these three equations we get $A = C = \frac{1}{3}$ and $B = \frac{4}{3}$. Then we get Simpson's rule:

$$\int_{-1}^{1} f \approx \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1)$$

We check that it will have equality for x^3 but not for x^4:

1. For x^3: $\int_{-1}^{1} x^3dx = 0 = \frac{1}{3}(-1) + \frac{4}{3}(0) + \frac{1}{3}(0) = 0$
2. For x^4: $\int_{-1}^{1} x^4dx = \frac{2}{5} \neq A + C = \frac{2}{3}$

Therefore Simpson rule has DOP 3.

8.2 Closed Newton-Cotes Formula (Integration)

Definition 136 (Closed Newton-Cotes Formula)**.** Let $a = x_0 < x_1 < \ldots < x_n = b$ be **equally space**, denote $h = \frac{b-a}{n}$. The $(n+1)$-point **closed Newton-Cotes formula** is:

$$\int_a^b f(x)dx = \sum_{j=0}^{n} f(x_j) \int_a^b L_j(x)dx + E(f)$$

Theorem 137. *Suppose that $\sum_{j=0}^{n} a_j f(x_j)$ is the $(n+1)$-point closed Newton-Cotes formula with $a = x_0, b = x_n$ and $h = \frac{b-a}{n}$. Then there exists $\xi \in (a, b)$ such that:*

1. *If n is even and $f \in C^{n+2}[a, b]$:*

$$\int_a^b f(x)dx = \sum_{j=0}^{n} a_j f(x_j) + \frac{h^{n+3} f^{(n+2)}(\xi)}{(n+2)!} \int_0^n t^2(t-1)\ldots(t-n)dt$$

2. *If n is odd and $f \in C^{n+1}[a, b]$:*

$$\int_a^b f(x)dx = \sum_{j=0}^{n} a_j f(x_j) + \frac{h^{n+2} f^{(n+2)}(\xi)}{(n+2)!} \int_0^n t(t-1)\ldots(t-n)dt$$

8.3 Open Newton-Cotes Formula (Integration)

Definition 138 (Open Newton-Cotes Formula)**.** Let $x_{-1} = a < \boxed{x_0 < x_1 < \ldots < x_n} < x_{n+1} = b$ be **equally space**, denote $h = \frac{b-a}{n+2}$. The $(n+1)$-point **open Newton-Cotes formula** is:

$$\int_a^b f(x)dx = \boxed{\sum_{j=0}^{n}} f(x_j) \int_a^b L_j(x)dx + E(f)$$

Note that x_0 does not have to be a and x_n does not have to be b.

Theorem 139. *Suppose that* $\sum_{j=0}^{n} a_j f(x_j)$ *is the* $(n+1)$*-point open Newton-Cotes formula with* $a = x_{-1}, b = x_{n+1}$ *and* $h = \frac{b-a}{n+2}$*. Then there exists* $\xi \in (a,b)$ *such that:*

1. *If* n *is even and* $f \in C^{n+2}[a,b]$*:*

$$\int_a^b f(x)dx = \sum_{j=0}^{n} a_j f(x_j) + \frac{h^{n+3} f^{(n+2)}(\xi)}{(n+2)!} \int_0^n t^2(t-1)\dots(t-n)dt$$

2. *If* n *is odd and* $f \in C^{n+1}[a,b]$*:*

$$\int_a^b f(x)dx = \sum_{j=0}^{n} a_j f(x_j) + \frac{h^{n+2} f^{(n+2)}(\xi)}{(n+2)!} \int_0^n t(t-1)\dots(t-n)dt$$

8.4 Composite Quadrature Rule (Integration)

The **idea:** is that we do one quadrature formula on each subinterval $[z_i, z_{i+1}]$ and then sum everything up:

Definition 140 (Composite Quadrature Rule)**.** We do:

$$\int_a^b f dx = \sum_{j=0}^{n} \int_{z_j}^{z_{i+1}} f(x)dx \approx \sum_{i=0}^{m-1} (z_{j+1} - z_j) \sum_{i=0}^{n} a_i f(z_j + t_i(z_{j+1} - z_j))$$

Definition 141 (Composite Simpson Rule)**.** We divide $[a,b]$ into n equally-spaced small subintervals: $a = x_0 < x_1 < \ldots < x_n = b$, i.e., $x_i = a + ih$ where $h = \frac{b-a}{n}$. Then:

$$\int_a^b f(x)dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx \approx \frac{h}{6} \sum_{i=1}^{n} \left[f(x_{i-1}) + 4f(\frac{x_{i-1}+x_i}{2}) + f(x_i) \right]$$

8.5 Gaussian Quadrature (Integration)

Recall the Newton-Cotes quadrature rule:

$$\int_a^b f(x)dx \approx \sum_{j=0}^{n} \underbrace{a_j}_{\text{to be found}} f(\underbrace{x_j}_{\text{known}})$$

That is, we already have fixed $(n+1)$-points $x_0, x_1, \ldots, x_n$ and try to find the $(n+1)$ coefficients $a_0, a_1, \ldots, a_n$.

Definition 142 (**Gaussian Quadrature Rule**)**.** The Gaussian Quadrature Rule is that:

$$\int_a^b f(x)dx \approx \sum_{j=0}^{n} \underbrace{a_j}_{\text{to be found}} f(\underbrace{x_j}_{\text{to be found also}})$$

That is, we need to determine **both** the $(n+1)$ points $x_0, x_1, \ldots, x_n$ and the $(n+1)$ coefficients $a_0, a_1, \ldots, a_n$. We have $(2n+2)$ unknowns, and we need to determine them from the following conditions:

Conditions: The relation:

$$\int_a^b f(x)dx = \sum_{j=0}^{n} a_j f(x_j)$$

is exact for any polynomial of degree $\leq 2n+1$.

Example 143. There are 2 equivalent ways of asking:

1. Version 1: Let $[a, b] = [-1, 1]$, then work out the Gaussian quadrature rule for $n = 1$.
2. Version 2: Find a_0, a_1, x_0, x_1 such that:

$$\int_{-1}^{1} f(x)dx \approx a_0 f(x_0) + a_1 f(x_1)$$

has DOP ≥ 3.

By Conditions, the

$$\int_{-1}^{1} f(x)dx = a_0 f(x_0) + a_1 f(x_1)$$

should be true for any polynomial of degree ≤ 3, so we will check for $1, x, x^2, x^3$:

1. For $f(x) = 1$: $2 = \int_{-1}^{1} 1dx = a_0 + a_1$
2. For $f(x) = x$: $0 = \int_{-1}^{1} xdx = a_0 x_0 + a_1 x_1$
3. For $f(x) = x^2$: $\frac{2}{3} = \int_{-1}^{1} x^2 dx = a_0 x_0^2 + a_1 x_1^2$
4. For $f(x) = x^3$: $0 = \int_{-1}^{1} x^3 dx = a_0 x_0^3 + a_1 x_1^3$

Solving for these 4 equations give $a_0 = a_1 = 1$ and $x_0 = -\frac{1}{\sqrt{3}}$ and $x_1 = \frac{1}{\sqrt{3}}$. Therefore:

$$\int_{-1}^{1} f dx \approx f(-\frac{1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}})$$

How do we construct Gaussian quadrature in general?

Theorem 144. *Let $q(x)$ be a nontrivial polynomial for degree $n + 1$ such that, for all $0 \leq k \leq n$:*

$$\int_{-1}^{1} x^k q(x)dx = 0$$

Let $x_0, x_1, \ldots, x_n$ be the zeros of q. Then the following quadrature rule:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{n} A_i f(x_i)$$

with

$$A_i = \int_{-1}^{1} L_i(x)dx = \int_{-1}^{1} \frac{\prod_{j \neq i}(t - x_j)}{\prod_{j \neq i}(x_i - x_j)} dt$$

is exact for all polynomials of degree at most $2n + 1$.

8.6 Numerical Differencing (Differentiation)

The **Aim:** find approximation of $f'(x_i)$ [or $f^{(k)}(x_i)$] via $\{f_i\}_{i=0}^{n}$

8.6.1 Forward and Backward Differences

Recall by Taylor series:

$$f(x \pm h) = f(x) \pm hf'(x) + \frac{h^2}{2} f''(\xi_{\pm}(x))$$

where $\xi_{\pm}(x) \in (x \pm h, x)$.

Definition 145 (Forward and Backward Differences)**.** The formula is:

$$f'(x) = \frac{f(x \pm h) - f(x)}{\pm h} - \underbrace{\frac{h^2}{2} f''(\xi_{\pm}(x))}_{\text{Error}}$$

1. $+$: forward difference
2. $-$: backward difference

8.6.2 Central Differences

Consider x_{i-1}, $x_i = x_{i-1} + h$, $x_{i+1} = x_{i-1} + 2h$. By Taylor series:

1. $f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2}{2} f''(x_i) + \frac{h^3}{6} f'''(\xi_1)$ where $\xi_1 \in (x_i, x_{i+1})$
2. $f(x_{i-1}) = f(x_i) - hf'(x_i) + \frac{h^2}{2} f''(x_i) - \frac{h^3}{6} f'''(\xi_2)$ where $\xi_2 \in (x_{i-1}, x_i)$

Definition 146 (Central Differences)**.** The **Central Differences** will be:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{h^2}{12}(f'''(\xi_1) + f'''(\xi_2))$$

and

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} - \frac{h^2}{24}(f^{(4)}(\xi_1) + f^{(4)}(\xi_2))$$

Where the red term are the errors.

9 Numerical ODE

The **Goal:** Approximate a solution to the following IVP: $y : [a, b] \to \mathbb{R}$:

$$\begin{cases} y'(t) & = f(t, y) \\ y(a) & = \alpha \end{cases}$$

by $\{y_i\}_{i=1}^N$ which solves some:

$$\begin{cases} y_{i+1} & = y_i + h \times \overbrace{\Phi(t_i, y_i; f, h)}^{\text{How do we find this } \Phi} \\ y_0 & = \alpha \end{cases}$$

such that $y_i \approx y(t_i)$ with some good error bounds, where: $t_0 = a$, $t_N = b$ and $t_{i+1} - t_i = h$.

Some Idea: We let:

$$y(t+h) - y(t) = \sum_{i=1}^{N} \frac{y^{(i)}(t)}{i!} h^i + O(h^{N+1}) \approx h \sum_{i=1}^{N} \frac{y^{(i)}(t)}{i!} h^{i-1} := \underbrace{h\Phi(y(t), t, h)}_{\text{Taylor series method: Runge-Kutta}}$$

9.1 Review of IVP- Initial Value Problem

Definition 147 (IVP-Initial Value Problem)**.** The problem:

$$\begin{cases} y'(t) &= f(t, y(t)),\ a \leq t \leq b \\ y(a) &= \alpha \end{cases}$$

is an IVP.

Example 148. Consider:

$$\begin{cases} y'(t) = \lambda y \\ y(0) = A \end{cases}$$

This gives $y(t) = Ae^{\lambda t}$.

Definition 149 (Lipschitz Condition)**.** Let $f : [a, b] \times D \to D$ be such that there exists an $L < \infty$ such that:

$$|f(t, x) - f(t, y)| \leq L|x - y|$$

for all $t \in [a, b]$ and $x, y \in D$. Then f is said to satisfy **Lipschitz Condition**.

Lemma 150. Suppose $f : [a, b] \times D :\to D$ is C^1, D is convex in $\mathbb{R}^d$ and

$$|\partial_y f(t, y)| < L \text{ (operator norm)}$$

Then f satisfy the **Lipschitz condition**.

Definition 151 (IVP well-posedness)**.** The IVP is **well-posed** if:

1. **Existence**: IVP has a solution
2. **Uniqueness:** The solution is unique
3. **Stability:** For all $\varepsilon < \varepsilon_0$, for all $\delta \in C^0([a, b], \mathbb{R}^d)$, $\delta_0 > 0$ such that:

   $$\|\delta\|_{C^0} + \delta_0 < \varepsilon$$

   the perturbed system:

   $$\begin{cases} z'(t) &= f(t, z(t)) + \delta(t) \\ z(a) &= \alpha + \delta_0 \end{cases}$$

   has a unique solution satisfying:

   $$\|z - y\|_{C^0[a,b]} < K\varepsilon$$

Theorem 152 (Global Well-posedness Theorem)**.** *Let $f : [a, b] \times \mathbb{R}^d \to \mathbb{R}^d$ satisfies the Lipschitz condition, then the IVP is well-posed, and:*

$$\|z - y\| < Ce^{L|t-a|}(1 + |b - a|)\varepsilon$$

9.2 (Explicit) One-step Method

In general the (Explicit) One-step method is:

$$\begin{cases} y_{i+1} &= y_i + h \times \overbrace{\Phi(t_i, y_i; f, h)}^{\text{How do we find this } \Phi} \\ y_0 &= \alpha \end{cases}$$

Definition 153 (Euler's Method)**.** The Euler's Method is when we take $\Phi(t_i, y_i; f, h) = f(t_i, y_i)$:

$$\begin{cases} y_{i+1} &= y_i + hf(t_i, y_i) \\ y + 0 &= \alpha \end{cases}$$

Example 154. The ODE $\frac{dy}{dx} = -2y + x^3 e^{-2x}$ with $y(0) = 1$. This ODE has analytical solution is

$$y = \frac{x^4 + 4}{4} e^{-2x}$$

The Euler is then:

$$y_{i+1} = y_i + h(-2y_i + x_i^3 e^{-2x_i})$$

Definition 155 (Taylor Series Method of order m)**.** Recall the Taylor series:

$$x(t_{j+1}) = x(t_j + h) = x(t_j) + hx'(t_j) + \frac{h^2}{2!}x''(t_j) + \frac{h^3}{3!}x'''(t_j) + \dots$$

A numerical method derived by truncating Taylor series up to the term involving h^m is called a Taylor Series method of order m.

Remark 156. Taylor Series method of order 1 is the Euler method.

Algorithm 157. The pseudocode for TSM of order 3:

$$\begin{aligned} &\text{For } j = 0, 1, \dots, M: \text{ do} \\ &\quad t_{j+1} = t_0 + jh, \\ &\quad a = f(t_j, x_j), \\ &\quad b = f_t(t_j, x_j) + f_x(t_j, x_j)a, \\ &\quad c = f_{tt}(t_j, x_j) + 2af_{tx}(t_j, x_j) + bf_x(t_j, x_j) + a^2 f_{xx}(t_j, x_j), \\ &\quad x_{j+1} = x_j + ah + \frac{b}{2}h^2 + \frac{c}{6}h^3 \\ &\quad \text{end} \end{aligned}$$

Example 158. Consider:

$$\begin{cases} x'(t) &= f(t, x) \\ x(t_0) &= x_0 \end{cases}$$

The TSM of order 1, which is the Euler method is as followed. By Taylor series:

$$\begin{aligned} x(t_{j+1}) &= x(t_j) + hx'(t_j) + \overbrace{O(h^2)}^{\text{error}} \\ x(t_{j+1}) &\approx x(t_j) + hf(t_j, x(t_j)) \end{aligned}$$

Denote

1. $x(t_i)$ the exact value of $x(t)$ at t_i
2. x_i the approximated value by numerical method

Then the Euler method is:

$$x_{j+1} = x_j + hf(t_j, x_j)$$

The two are different:

1. $\{x(t_j)\}$: exact/analytical
2. $\{x_j\}$: numerically approximation

That is $x_j \approx x(t_j)$ and not equal.
The TSM of order 3 is:

$$\overbrace{x(t_{j+1}) = x(t_j) + hx'(t_j) + \frac{h^2}{2}x''(t_j) + \frac{h^3}{3!}x'''(t_j)}^{\text{approximation}} + \underbrace{O(h^3)}_{\text{Error}}$$

In which we calculated the terms:

$$\begin{aligned} x'(t) &= f(t,x) \\ x''(t) &= f_t(t,x) + f_x(t,x)x'(t) \\ x'''(t) &= f_{tt}(t,x) + 2f_{tx}(t,x)x'(t) + f_x(t,x)x''(t) + f_{xx}(t,x)(x')^2 \end{aligned}$$

9.3 Error Analysis

Definition 159 (Truncation Error)**.** There are 2 types of truncation errors:

1. **Local truncation error** τ_i at ith step (time t_i) is defined to be:

$$\tau_i(h) := \frac{y(t_i) - \Big(y(t_{i-1}) + h\Phi(t_{i-1}, y(t_{i-1}, h))\Big)}{h} = \frac{y(t_i) - y(t_{i-1})}{h} - \Phi(t_{i-1}, y(t_{i-1}), h)$$

2. **Global truncation error** e_i at the ith step is defined to be:

$$e_i(h) = |y(t_i) - y_i) = \left| y(t_i) - \left(y_0 + h\sum_{k=0}^{i-1} \Phi(t_k, y_k, h) \right) \right|$$

Remark 160. Local truncation error measures the accuracy of the method at a specific step assuming that the method is **exact at the previous steps**.

Definition 161 (Consistence and Convergence)**.** A one-step difference equation method with local truncation error $\tau_i(h)$ and global truncation error $e_i(h)$ at the i-th step is called

1. **consistent** with the IVP if

$$\lim_{h\to 0} \max_{1\le i\le N} |\tau_i(h)| = 0$$

2. **convergent** with the IVP if

$$\lim_{h\to 0}\max_{1\le i\le N}|e_i(h)| = 0$$

Remark 162. Consistency determine **local** effect while convergence determines **global** effect of the method.

Definition 163 (Order of a Method)**.** A numerical method to solve IVP of order p if

$$\max_{1\le i\le N}\tau_i(h) = O(h^p)$$

Example 164. Show that Euler method is of order 1 and consistent.
Recall the Euler method is that:

$$y_{i+1} = y_i + hf(t_i, y_i),\ y_0 = \alpha$$

Then:

$$\begin{aligned}\tau_i(h) &= \frac{y(t_i) - \Big[y(t_{i-1}) + hf(t_{i-1}, y(t_{i-1}))\Big]}{h}\\ &= \frac{\frac{1}{2}h^2y''(\xi_{i-1})}{h} \text{ for } \xi_{i-1}\in(t_{i-1}, t_i)\\ &= \frac{1}{2}hy''(\xi_{i-1})\end{aligned}$$

If $\|y''\| \le M$ then $|\tau_i(h)| \le \frac{M}{2}h = O(h)$. Therefore:

$$\max_i \tau_i(h) = O(h) \implies \text{ order 1}$$

and:

$$\lim_{h\to 0}\max_i|\tau_i(h)| = 0 \implies \text{ consistent}$$

Remark 165. Similar to the derivation of Euler method, Taylor method of order n has local truncation error $O(h^n)$ and thereby consistent.

Lemma 166 (Discrete Gronwall Inequality)**.** Let $s, t > 0$ and $\{a_i\}_{i=0}^k$ satisfying:

$$\begin{cases} a_0 & \ge -\frac{t}{s}\\ a_{i+1} & \le (1+s)a_i + t,\ \text{for } i = 0, 1, \ldots, k-1\end{cases}$$

Then $a_{i+1} \le e^{(i+1)s}(a_0 + \frac{t}{s}) - \frac{t}{s}$.

Proof. We have:

$$\begin{aligned}a_{i+1} &\le (1+s)a_i + t \le (1+s)\Big[(1+s)a_{i-1} + t\Big] + t\\ &\le \ldots \le (1+s)^{i+1}a_0 + t\Big[(1+s)^i + \ldots + (1+s) + 1\Big]\\ &= (1+s)^{i+1}a_0 + t\frac{(1+s)^{i+1} - 1}{s}\\ &= (1+s)^{i+1}(a_0 + \frac{t}{s}) - \frac{t}{s}\\ &\le e^{(i+1)s}(a_0 + \frac{t}{s}) - \frac{t}{s}\end{aligned}$$

where the last inequality is true by Taylor expansion:

$$e^s = 1 + s + \frac{s^2}{2!}e^\xi \geq 1 + s \implies (1+s)^{i+1} \leq e^{(i+1)s}$$

This completed the lemma proof. □

Theorem 167. *Let $\Phi : [a,b] \times \mathbb{R}^d \times [0,h_0] \to \mathbb{R}^d$ be continuous, satisfying Lipschitz condition in variable y:*

$$|\Phi(t,p,h) - \Phi(t,q,h)| \leq L|p-q|$$

Then:

1. *Consistency $\iff$ convergence $\iff$ $\Phi(t,y,0) = f(t,y)$ for $t \in [a,b]$*
2. *If, in addition, there exists $\tau(h)$ such that $\tau_i(h) \leq \tau(h)$ for $i = 0,1,\ldots,N$ and $0 \leq h \leq h_0$, then:*

$$|e_i| \leq \frac{\tau(h)}{L}(e^{L(t_i-a)} - 1)$$

Proof. We skipped the part convergence implies consistence. We will now prove the part Consistent (local) $\implies$ convergence (global). By definition:

$$\begin{aligned} y(t_i) &= y(t_{i-1}) + h\Phi(t_{i-1}, y(t_{i-1}), h) + h\tau(h) \\ y_i &= y_{i-1} + h\Phi(t_{i-1}, y_{i-1}, h) \end{aligned}$$

Subtract the 2 equations give:

$$y(t_i) - y_i = y(t_{i-1}) - y_{i=1} + h\Big[\Phi(t_{i-1}, y(t_{i-1}, h) - \Phi(t_{i-1}, y_{i-1}, h)\Big] + h\tau(h)$$

Now note that $e_i = y(t_i) - y_i$ and $e_{i-1} = y(t_{i-1}) - y_{i-1}$, and using the Lipschitz condition

$$\|\Phi(t_{i-1}, y(t_{i-1}, h) - \Phi(t_{i-1}, y_{i-1}, h)\| \leq L|y(t_{i-1} - y_{i-1}| = L|e_{i-1}|$$

and taking absolute value we get:

$$\begin{aligned} |e_i| &\leq |e_{i-1}| + hL|e_{i-1}| + h|\tau_i| \\ |e_i| &\leq |e_{i-1}|(1+hL) + h|\tau| \end{aligned}$$

since all the $|\tau_i| \leq |\tau|$. Now if we denote $s = hL$ and $t = |\tau|$, using discrete Gronwall Inequality Lemma:

$$|e_i| \leq |e_{i-1}|(1+hL) + h|\tau| \leq e^{ihL}(|e_0| + \frac{h\tau}{hL}) - \frac{h\tau}{hL} = \frac{\tau}{L}\Big[e^{L(t_i-a)} - 1\Big]$$

Therefore when $h \to 0 \xrightarrow{\text{consistent}} \tau(h) \to 0 \xrightarrow{\text{by inequality above}} e_i \to 0 \implies \max_i |e_i| \to 0 \implies$ convergence.
Our next goal is to show that consistence $\iff \Phi(t_i, y(t_i), 0) = f(t_i, y(t_i))$:

$$\begin{aligned} \tau_i(h) &= \frac{y(t_i) - y(t_{i-1})}{h} - \Phi(t_{i-1}, y(t_{i-1}), h) \\ &\xrightarrow{h\to 0} f(t_{i-1}, y(t_{i-1})) - \Phi(t_{i-1}, y(t_{i-1}), 0)) \\ &\to 0 \end{aligned}$$

So $\Phi(t_{i-1}, y(t_{i-1}), 0) = f(t_{i-1}, y(t_{i-1}))$. □

Lemma 168 (Error bound of Euler Method)**.** Consider the IVP:

$$\begin{cases} y'(t) &= f(t, y(t)), \ a \le t \le b \\ y(a) &= \alpha \end{cases}$$

where f is Lipschitz condition with constant L over $[a, b] \times \mathbb{R}$ and $|y''(t)| \le M$ for any $t \in [a, b]$. Then the Euler method satisfying:

$$|e_i| \le \frac{Mh}{2L}\left(e^{L(t_i - a)} - 1\right)$$

Proof. By Taylor's theorem: there exists $\xi_{i-1} \in [t_{i-1}, t_i]$ such that:

$$y(t_i) = y(t_{i-1}) + hf(t_{i-1}, y(t_{i-1})) + \frac{h^2}{2}y''(\xi_{i-1})$$

and Euler:

$$y_i = y_{i-1} + hf(t_{i-1}, y_{i-1})$$

Taking the two equations subtract:

$$y(t_i) - y_i = y(t_{i-1}) - y_{i-1} + h\left[f(t_{i-1}, y(t_{i-1})) - f(t_{i-1}, y_{i-1})\right] + \frac{h^2}{2}y''(\xi_{i-1})$$

Take absolute value:

$$|e_i| \le (1 + hL)|e_{i-1}| + \frac{h^2}{2}M$$

From Gronwall Inequality, take $s = hL$ and $t = \frac{h^2 M}{2}$:

$$\begin{aligned} |e_i| &\le e^{ihL}(|e_0| + \frac{Mh}{2L}) - \frac{Mh}{2L} \\ &= \frac{Mh}{2L}(e^{L(t_i - a)} - 1) \end{aligned}$$

This completed the Lemma proof. □

9.4 Runge-Kutta Methods Introduction

Recall the Taylor series method:

$$x(t_{j+1}) = x(t_j + h) = x(t_j) + hx'(t_j) + \frac{h^2}{2!}x''(t_j) + \frac{h^3}{3!}x'''(t_j) + \dots$$

And the algorithm is:

$$\begin{aligned} &\text{For } j = 0, 1, \dots, M: \text{ do} \\ &\quad t_{j+1} = t_0 + jh, \\ &\quad a = f(t_j, x_j), \\ &\quad b = f_t(t_j, x_j) + f_x(t_j, x_j)a, \\ &\quad c = f_{tt}(t_j, x_j) + 2af_{tx}(t_j, x_j) + bf_x(t_j, x_j) + a^2 f_{xx}(t_j, x_j), \\ &\quad x_{j+1} = x_j + ah + \frac{b}{2}h^2 + \frac{c}{6}h^3 \\ &\text{end} \end{aligned}$$

You need to evaluate higher order derivatives: TROUBLE

Idea of Runge-Kutta methods: achieve high order local truncation error without computations of $f^{(k)}(t, y(t))$ for $1 \le k \le n - 1$.

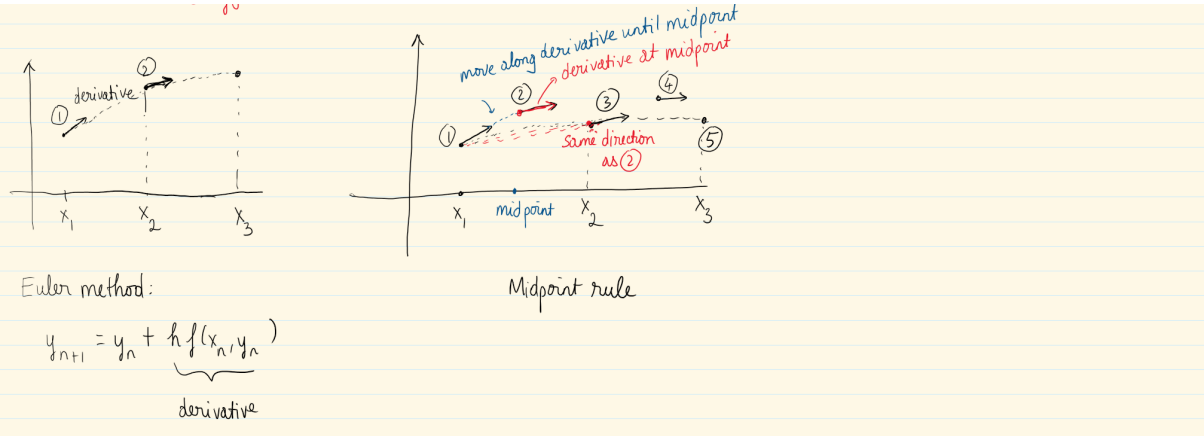


Figure 16: Euler Method vs Midpoint Rule

9.5 The Mid-point method (a Runge-Kutta Method)

Recall the IVP:

$$\begin{cases} y'(t) &= f(t,y) \\ y(a) &= \alpha \end{cases}$$

If we do Taylor method of order 2 we will get:

$$T^{(2)}(t,y) = f(t,y) + \frac{h}{2}f'(t,y) = \boxed{f(t,y) + \frac{h}{2}\frac{\partial f}{\partial t}(t,y) + \frac{h}{2}\frac{\partial f}{\partial y}(t,y).f(t,y)} \tag{1}$$

The **Idea:** Find $a_1 f(t+\alpha_1, y+\beta_1) \approx T^{(2)}(t,y)$: The derivation is:

$$a_1 f(t+\alpha_1, y+\beta_1) = \boxed{a_1 f(t,y) + a_1\alpha_1\frac{\partial f}{\partial t}(t,y) + a_1\beta_1\frac{\partial f}{\partial y}(t,y)} \tag{2}$$

$$\underbrace{+\frac{\alpha_1^2}{2}\frac{\partial^2 f}{\partial t^2}(\xi,\eta) + \alpha_1\beta_1\frac{\partial^2 f}{\partial t\partial y}(\xi,\eta) + \frac{\beta_1^2}{2}\frac{\partial^2 f}{\partial y^2}(\xi,\eta)}_{O(h^2)} \tag{3}$$

Now we match the coefficients in the two boxes and get:

$$\begin{cases} a_1 &= 1 \\ a_1\alpha_1 &= \frac{h}{2} \\ a_1\beta_1 &= \frac{h}{2}f(t,y) \end{cases} \Longrightarrow \begin{cases} a_1 &= 1 \\ \alpha_1 &= \frac{h}{2} \\ \beta_1 &= \frac{h}{2}f(t,y) \end{cases}$$

Definition 169 (Midpoint Method)**.** The approximation is now:

$$\begin{cases} y_{i+1} &= y_i + hf(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i,y_i)) \\ y_0 &= \alpha \end{cases}$$

See figure (16) for illustration:

9.6 Modified Euler's Method (a Runge-Kutta Method)

Idea: Find $a_1 f(t,y) + a_2 f(t+\alpha_2, y+\delta_2 f(t,y)) \approx T^{(3)}(t,y)$. Now the Taylor's method of order 3 is:

$$\begin{aligned} T^{(3)}(t,y) &= f(t,y) + \frac{h}{2}f'(t,y) + \frac{h^2}{6}f''(t,y) \\ &= \boxed{f(t,y) + \frac{h}{2}\left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}f\right)} \\ &\quad + \underbrace{\frac{h^2}{6}\left(\frac{\partial^2 f}{\partial t^2} + \frac{\partial^2 f}{\partial t \partial y}f + \frac{\partial^2 f}{\partial y \partial t}f + \frac{\partial^2 f}{\partial y^2}f^2 + \frac{\partial f}{\partial y}.\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}.\frac{\partial f}{\partial y}f\right)}_{\text{error}} \end{aligned}$$

And deriving $a_1 f(t,y) + a_2 f(t+\alpha_2, y+\delta_2 f(t,y))$ gives:

$$\begin{aligned} a_1 f(t,y) + a_2 f(t+\alpha_2, y+\delta_2 f(t,y)) \approx & \boxed{(a_1+a_2)f(t,y) + a_2\alpha_2\frac{\partial f}{\partial t} + a_2\delta_2 f\frac{\partial f}{\partial y}} \\ & + a_2\frac{\alpha_2^2}{2}\frac{\partial^2 f}{\partial t^2} + a_2\alpha_2\delta_2 f\frac{\partial^2 f}{\partial t\partial y} + a_2\frac{\delta_2^2 f^2}{2}\frac{\partial^2 f}{\partial y^2} \end{aligned}$$

Comparing the boxes give:

$$\begin{cases} a_1 + a_2 &= 1 \\ a_2\alpha_2 &= \frac{h}{2} \\ a_2\delta_2 &= \frac{h}{2} \end{cases}$$

The solution is **not** unique, so we choose one solution: $a_1 = a_2 = \frac{1}{2}$ and $\delta_2 = \alpha_2 = h$:

Definition 170 (Modified (Improved) Euler's Method)**.** The system of approximation in **Modified Euler Method** is:

$$\begin{cases} y_{i+1} &= y_i + \frac{h}{2}\Big[f(t_i,y_i) + f(t_i+h, y_i+hf(t_i,y_i))\Big] \\ y_0 &= \alpha \end{cases}$$

9.7 General Runge-Kutta Method

Definition 171 (**General Runge-Kutta Methods**)**.** This is a **one-step** method defined by:

$$\begin{cases} y_{i+1} &= y_i + \underbrace{\sum_{j=1}^{s} b_j k_j}_{h\Phi(t_i,y_i,h)} \\ y_0 &= \alpha \end{cases}$$

where the

$$k_j = hf\left(t_i + c_j h, y_i + \sum_{l=1}^{s} a_{jl}k_l\right)$$

is called a **Runge-Kutta Method.** In the equation we have these terms:

1. h: step size
2. s: number of sub-steps (stages) taken from t_i to t_{i+1}
3. b_j: weight of the j-th sub-step to update y_i
4. c_j: increment factor of the j-th sub-step
5. a_{jl}: the weight that the l-th sub-step contributes to the j-th sub-step.

Example 172. Recall the **Mid-point Method** the iteration is:

$$y_{i+1} = y_i + hf(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i))$$

If we rewrite this a little bit, we will see that:

$$y_{i+1} = y_i + 0.\underbrace{hf(t_i, y_i)}_{k_1} + \underbrace{hf(t_i + \frac{h}{2}, y_i + \frac{1}{2}\overbrace{hf(t_i, y_i))}^{k_1}}_{k_2}$$

Recall:

$$k_1 = hf(t_i + c_1 h, y_i + a_{11}k_1 + a_{12}k_2) = hf(t_i, y_i)$$
$$k_2 = hf(t_i + c_2 h, y_i + a_{21}k_1 + a_{22}k_2) = hf(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_1)$$

So we can see that:

1. h: step size
2. $s = 2$: number of sub-steps (stages) taken from t_i to t_{i+1}
3. $b_1 = 0$, $b_2 = 1$: weight of the j-th sub-step
4. $c_1 = 0$ and $c_2 = \frac{1}{2}$: increment factor of the j-th sub-step
5. $a_{11} = a_{12} = 0$ and $a_{21} = \frac{1}{2}$, and $a_{22} = 0$: the weight that the l-substep contributes to the j-th substep.

Example 173. Recall the **Modified Euler's Method**, the iteration is:

$$y_{i+1} = y_i + \frac{h}{2}\left[f(t_i, y_i) + f(t_i + h, y_i + hf(t_i, y_i))\right]$$

Taking a closer look at this iteration:

$$y_{i+1} = y_i + \frac{1}{2}\underbrace{hf(t_i, y_i)}_{k_1} + \frac{1}{2}\underbrace{hf(t_i + h, y_i + \overbrace{hf(t_i, y_i)}^{k_1})}_{k_2}$$

And with:

$$k_1 = hf(t_i + c_1 h, y_i + a_{11}k_1 + a_{12}k_2) = hf(t_i, y_i)$$
$$k_2 = hf(t_i + c_2 h, y_i + a_{21}k_1 + a_{22}k_2) = hf(t_i + h, y_i + k_1)$$

We see that:

1. h: step size
2. $s = 2$
3. $b_1 = \frac{1}{2}$ and $b_2 = \frac{1}{2}$
4. $c_1 = 0$ and $c_2 = 1$
5. $a_{11} = a_{12} = 0$ and $a_{21} = 1$ and $a_{22} = 0$.

Theorem 174 (Consistency and Stability)**.** *We have the following conditions:*

1. *If $\sum_{j=1}^{s} b_j = 1$, then the Runge-Kutta method is* ***consistent***
2. *If $\sum_{l=1}^{s} a_{jl} = c_j$, then the Runge-Kutta method is* ***stable***
3. *If $\sum_{j=1}^{s} b_j = 1$ and $\sum_{l=1}^{s} a_{jl} = c_j$ and $\sum_{j=1}^{s} b_j c_j = \frac{1}{2}$, then the Runge-Kutta method is of second order.*

Proof. Direct calculation. □

Example 175. The midpoint method has :

1. h: step size
2. $s = 2$: number of sub-steps (stages) taken from t_i to t_{i+1}
3. $b_1 = 0$, $b_2 = 1$: weight of the j-th sub-step
4. $c_1 = 0$ and $c_2 = \frac{1}{2}$: increment factor of the j-th sub-step
5. $a_{11} = a_{12} = 0$ and $a_{21} = \frac{1}{2}$, and $a_{22} = 0$: the weight that the l-substep contributes to the j-th substep.

So:

1. $b_1 = 0$, $b_2 = 1$ so $b_1 + b_2 = 1$: so it is consistent.
2. $a_{11} + a_{12} = 0 = c_1$ and $a_{21} + a_{22} = \frac{1}{2}$ so it is stable.
3. $b_1 c_1 + b_2 c_2 = \frac{1}{2}$ so it is second order.

Example 176. The modified Euler Method has:

1. h: step size
2. $s = 2$
3. $b_1 = \frac{1}{2}$ and $b_2 = \frac{1}{2}$
4. $c_1 = 0$ and $c_2 = 1$
5. $a_{11} = a_{12} = 0$ and $a_{21} = 1$ and $a_{22} = 0$.

So:

1. $b_1 + b_2 = 1$: so it is consistent.

2. $a_{11} + a_{12} = c_1$ and $a_{21} + a_{22} = \frac{1}{2}$ so it is stable.

3. $b_1c_1 + b_2c_2 = \frac{1}{2}$ so it is second order.

Remark 177. The Runge-Kutta method is:

1. **Explicit** if $a_{jl} = 0$ for all $l \geq j$, i.e., $k_j \leftarrow k_1, k_2, \ldots, k_{j-1}$: **no equation needs to be solved.**

2. **Semi-implicit** if $a_{jl} = 0$ for all $l > j$, i.e., $k_j \leftarrow k_1, k_2, \ldots, k_j$: **Each k_j can be solved by an equation containing k_l for $l < j$.**

3. **Implicit**: Otherwise, i.e., $k_j \rightarrow k_1, k_2, \ldots, k_j, k_{j+1}, \ldots$: **the $k_1, \ldots, k_j, k_{j+1}, \ldots$ need to be solved simultaneously.**

Example 178. For the Midpoint method, we see that $a_{11} = a_{12} = a_{22} = 0$ so this is explicit. Note that we do not care $a_{21} = \frac{1}{2}$.

Example 179. For the Modified Euler Method, we see that $a_{11} = a_{12} = a_{22} = 0$ so this is explicit.

Remark 180. An explicit/semi-explicit/implicit RK method is an **explicit** one-step method, since the y_{i+1} can be found by y_i.

Remark 181. Implicit RK method can be more stable, but it is not easy to compute.

9.7.1 Butcher Tableau

Definition 182 (Butcher Tableau)**.** The **Butcher Tableau** (matrix) is a compact way to describe a Runge-Kutta method through parameter defined as:

$$\left(\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \ldots & a_{1s} \\ c_2 & a_{21} & a_{22} & \ldots & a_{2s} \\ \vdots & \vdots & & a_{ij} & \vdots \\ c_s & a_{s1} & a_{s2} & \ldots & a_{ss} \\ \hline & b_1 & b_2 & \ldots & b_s \end{array}\right) = \left(\begin{array}{c|c} \overrightarrow{c} & A \\ \hline & \overrightarrow{b}^T \end{array}\right)$$

Remark 183. We observe that:

1. $c_1 = 0$ is used in general, and $a_{11} = 0$ holds for explicit methods.

2. An explicit RK method with s sub-steps may not have order higher than s, and thereby the matrix A is lower triangular and all its diagonal entries are zero.

Theorem 184 (Consistency and Stability in Butcher Tableau)**.** *Recall the theorem:*

1. *If $\sum_{j=1}^{s} b_j = 1$, then the Runge-Kutta method is **consistent***

2. *If $\sum_{l=1}^{s} a_{jl} = c_j$, then the Runge-Kutta method is **stable***

3. *If $\sum_{j=1}^{s} b_j = 1$ and $\sum_{l=1}^{s} a_{jl} = c_j$ and $\sum_{j=1}^{s} b_j c_j = \frac{1}{2}$, then the Runge-Kutta method is of second order.*

In Butcher Tableau:

$$\left(\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & & a_{ij} & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}\right)$$

For the c_1 a_{11} $a_{12} \dots a_{1s}$: ***stable*** *if the sum of all the* a *in each line equal to the* c_i.
For the b_1 $b_2 \dots b_s$: ***consistent*** *if the sum of all the* b *equal to* 1.

Example 185 (Butcher Tableau for Midpoint Method)**.** Recall Midpoint Method has:

- $b_1 = 0$ and $b_2 = 0$
- $c_1 = 0$ and $c_2 = \frac{1}{2}$
- $a_1 1 = a_1 2 = 0$ and $a_{21} = \frac{1}{2}$ and $a_{22} = 0$.

So we get the Butcher Tableau:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

From this Butcher Tableau we see it is consistent and stable.

Example 186 (Butcher Tableau for Modified Euler Method)**.** Recall Midpoint Method has:

- $b_1 = \frac{1}{2}$ and $b_2 = \frac{1}{2}$
- $c_1 = 0$ and $c_2 = 1$
- $a_1 1 = a_1 2 = 0$ and $a_{21} = 1$ and $a_{22} = 0$.

So we get the Butcher Tableau:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

From this Butcher Tableau we see it is consistent and stable.

Definition 187 (**Heun's Method**)**.** The iteration is:

$$y_{i+1} = y_i + \frac{1}{4}k_1 + \frac{3}{4}k_3$$

in which:

$$\begin{aligned} k_1 &= hf(t_i, y_i) \\ k_2 &= hf(t_i + \frac{1}{3}h, y_i + \frac{1}{3}k_1) \\ k_3 &= hf(t_i + \frac{2}{3}h, y_i + \frac{2}{3}k_2) \end{aligned}$$

Example 188 (Butcher Tableau for Heun's Method)**.** Rewriting the iteration:

$$y_{i+1} = y_i + \frac{1}{4}k_1 + \frac{3}{4}k_3$$
$$y_{i+1} = y_i + \frac{1}{4}k_1 + 0k_2 + \frac{3}{4}k_3$$

This gives the b: $b_1 = \frac{1}{4}$, $b_2 = 0$, and $b_3 = \frac{3}{4}$.
Now rewrite the k_1:

$$k_1 = hf(t_i, y_i)$$
$$k_1 = hf(t_i + 0h, y_i + 0k_1 + 0k_2 + 0k_3)$$

The red term is $c_1 = 0$ and the blue terms are the first row of a: $a_{11} = a_{12} = a_{13} = 0$.
The second k_2:

$$k_2 = hf(t_i + \frac{1}{3}h, y_i + \frac{1}{3}k_1)$$
$$k_2 = hf(t_i + \frac{1}{3}h, y_i + \frac{1}{3}k_1 + 0k_2 + 0k_3)$$

This again give the red term $c_2 = \frac{1}{3}$ and the blue term: $a_{21} = \frac{1}{3}, a_{22} = a_{23} = 0$.
Finally rewrite k_3:

$$k_3 = hf(t_i + \frac{2}{3}h, y_i + \frac{2}{3}k_2)$$
$$k_3 = hf(t_i + \frac{2}{3}h, y_i + \frac{2}{3}k_2 + 0k_1 + 0k_3)$$

So this gives the red term $c_3 = \frac{2}{3}$ and the blue terms: $a_{32} = \frac{2}{3}, a_{31} = a_{33} = 0$.
And now we get our Butcher Tableau for Heun's Method:

0	0	0	0
$\frac{1}{3}$	$\frac{1}{3}$	0	0
$\frac{2}{3}$	0	$\frac{2}{3}$	0
	$\frac{1}{4}$	0	$\frac{3}{4}$

From this Butcher Tableau we see this method is consistent (last row sum is 1) and stable (each row has same sum to c).

Definition 189 (**RK4**)**.** The iteration is

$$y_{i+1} = y_i + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4$$

where:

$$k_1 = hf(t_i, y_i)$$
$$k_2 = hf(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1)$$
$$k_3 = hf(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2)$$
$$k_4 = hf(t_i + h, y_i + k_3)$$

Example 190 (Butcher Tableau for RK4 method)**.** We first rewrite the iteration:

$$y_{i+1} = y_i + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4$$

And this gives the b: $b_1 = \frac{1}{6}$, $b_2 = \frac{1}{3}$, $b_3 = \frac{1}{3}$ and $b_4 = \frac{1}{6}$.
Now rewrite the k_1:

$$\begin{aligned} k_1 &= hf(t_i, y_i) \\ k_1 &= hf(t_i + 0h, y_i + 0k_1 + 0k_2 + 0k_3 + 0k_4) \end{aligned}$$

The red term is $c_1 = 0$ and the blue terms are the first row of a: $a_{11} = a_{12} = a_{13} = a_{14} = 0$.
The second k_2:

$$\begin{aligned} k_2 &= hf(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1) \\ k_2 &= hf(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 + 0k_2 + 0k_3 + 0k_4) \end{aligned}$$

This again give the red term $c_2 = \frac{1}{2}$ and the blue terms: $a_{21} = \frac{1}{2}, a_{22} = a_{23} = a_{24} = 0$.
The third k_3:

$$\begin{aligned} k_3 &= hf(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2) \\ k_3 &= hf(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 + 0k_1 + 0k_3 + 0k_4) \end{aligned}$$

So this gives the red term $c_3 = \frac{1}{2}$ and the blue terms: $a_{32} = \frac{1}{2}, a_{31} = a_{33} = a_{34} = 0$.
The last k_4:

$$\begin{aligned} k_4 &= hf(t_i + h, y_i + k_3) \\ k_4 &= hf(t_i + 1h, y_i + 1k_3 + 0k_1 + 0k_4 + 0k_2) \end{aligned}$$

So this gives the red term $c_4 = 1$ and the blue terms: $a_{43} = 1, a_{41} = a_{42} = a_{44} = 0$.
And we get the Butcher Tableau:

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

From this Butcher Tableau we see that it is consistent (last row sum is 1) and stable (each row sum is equal to c).

Remark 191. RK4 is a popular method. See figure (17) and figure (18) for illustration.

9.8 Summary of One-Step Method

A numerical method which use only 1 single previous value y_n to compute y_{n+1} is called **single-step method**.

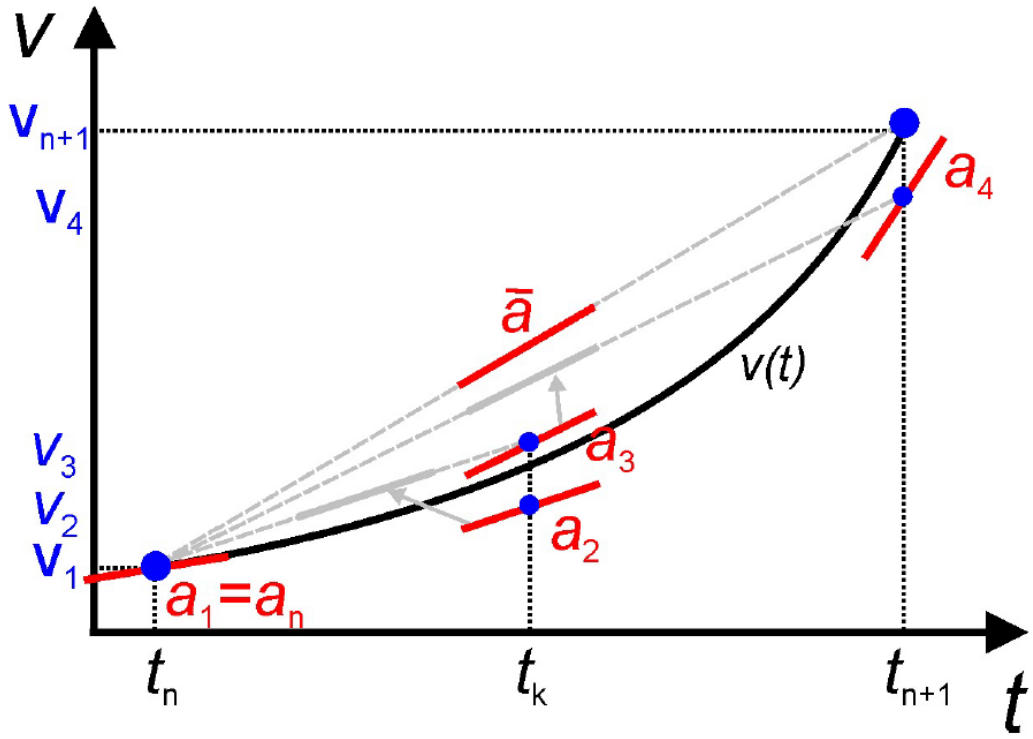


Figure 17: RK4 method illustration

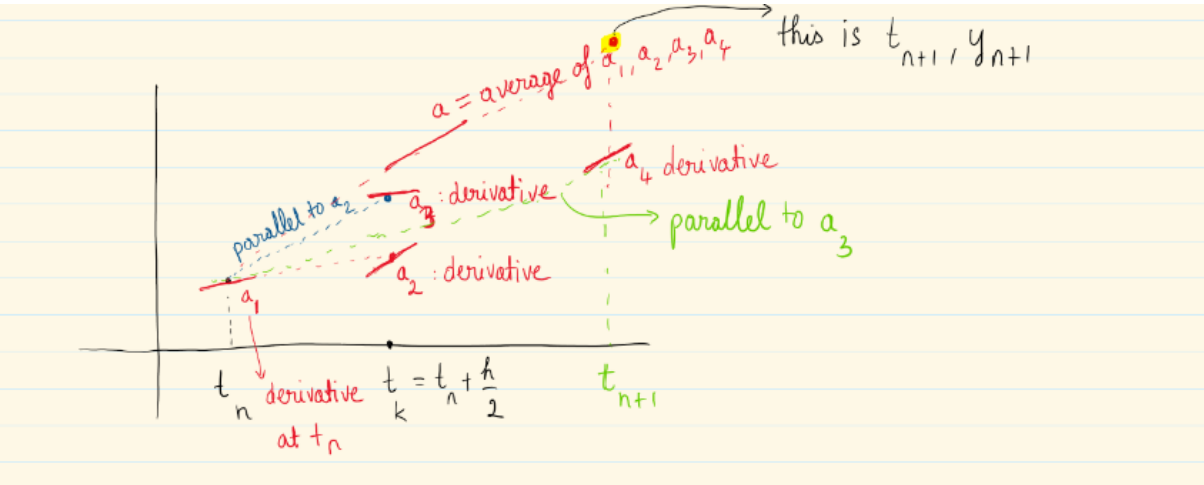


Figure 18: RK4 method illustration 2

Definition 192 (Explicit one step method)**.** The iteration is:

$$\begin{cases} y_{i+1} & = y_i + h\Phi(t_i, y_i; f, h) \\ y_0 & = \alpha \end{cases}$$

In which we have

Definition 193 (Euler Method)**.**

$$y_{i+1} = y_i + hf(t_i, y_i)$$

Definition 194 (Mid point method)**.**

$$y_{i+1} = y_i + hf(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i))$$

Definition 195 (Modified Euler Method)**.**

$$y_{i+1} = y_i + \frac{h}{2}\Big[f(t_i, y_i) + f(t_i + h, y_i + hf(t_i, y_i))\Big]$$

10 Multistep Methods to solve ODE

A numerical method which uses more than one previous value from $y_0, y_1, \ldots, y_{n-1}, y_n$ to evaluate $y_{n+1} \approx y(t_{n+1})$ is called **a multi-step method**. If m previous values are used, then we call this multistep method a **m-step multi-step method**.

Definition 196 (general m-step method)**.** A **general m-step method** is defined by:

$$\sum_{j=0}^{m} a_j y_{k+j} = h \sum_{j=0}^{m} b_j f(t_{k+j}, y_{k+j})$$

where the $y_0, y_1, \ldots, y_{k+m-1}$ are given. Without loss of generality, $a_m = 1$. The method is **explicit** if $b_m = 0$, and it is **implicit** if $b_m \neq 0$ (i.e, finding y_{k+m} involves solving an equation or not).

Example 197. All these examples are single-step, because we have:

$$y_{n+1} - y_n = h.\Big[\ldots\Big] \Longrightarrow m = 1$$

1. $y_{n+1} = y_n + hf(t_n, y_n)$: **explicit**.
2. $y_{n+1} = y_n + hf(t_n, y_{n+1})$: **implicit**.
3. $y_{n+1} = y_n + \frac{h}{2}\Big[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\Big]$: **implicit**.
4. $y_{n+1} = y_n + hf(t_{n+\frac{1}{2}}, \frac{y_n + y_{n+1}}{2})$: **implicit**.

Algorithm 198 (General derivation of multi-step methods)**.** : We have the IVP:

$$\begin{cases} y'(t) & = f(t,y) \\ y(a) = \alpha & \end{cases}$$

Integrate the ODE over each interval $[t_n, t_{n+1}]$:

$$\int_{t_n}^{t_{n+1}} y'(t)dt = \int_{t_n}^{t_{n+1}} f(t,y(t))dt$$

Therefore:

$$y(t_{n+1}) = y(t_n) + \underbrace{\boxed{\int_{t_n}^{t_{n+1}} f(t,y(t))dt}}_{\text{use some quadrature rules to approximate this integral}}$$

In approximating the integral with quadrature rules, you may use some of previous values $y_0, y_1, \ldots, y_{n-1}, y_n$.

10.1 Adams-Bashforth Methods

Approximate f by its interpolation at the following m nodal points:

$$t_n, t_{n-1}, \ldots, t_{n-(m-1)}$$

The resultant methods are called **m-step Adams-Bashforth methods**. Note that these methods are **explicit**.

Definition 199 (**2-step Adams-Bashforth Method**)**.** The iteration is:

$$y_{n+1} = y_n + h\left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right) \text{ for } n = 1, 2, 3, \ldots$$

Note that this method is **explicit**.

Algorithm 200. How to get **2-step Adams-Bashforth Method**, assuming that we are currently at step t_{n+1}:
Interpolating of f at t_n, t_{n+1}:

$$L(t) = \frac{t - t_n}{t_{n-1} - t_n} f_{n-1} + \frac{t - t_{n-1}}{t_n - t_{n-1}} f_n$$

Then:

$$\begin{aligned} \int_{t_n}^{t_{n+1}} f(t,y(t))dt &\approx \int_{t_n}^{t_{n+1}} L(t)dt : \text{ Trapezoid rule} \\ &= \frac{h}{2}\Big[L(t_n) + L(t_{n+1})\Big] \\ &= h\left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right) \text{ , plug in } t_n, t_{n+1} \text{ into } L(t) \end{aligned}$$

Therefore:

$$y_{n+1} = y_n + h\left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right)$$

Definition 201 (**3-step Adams-Bashforth Method**)**.** The iteration is:

$$y_{n+1} = y_n + h\left(\frac{32}{12}f_n - \frac{4}{3}f_{n-1} + \frac{5}{12}f_{n-2}\right) \text{ for } n = 2, 3, 4, \ldots$$

Note that this method is **explicit**.

Algorithm 202. How to get **3-step Adams-Bashforth Method**, assuming we are currently at t_{n+1}: Interpolation of f at t_n, t_{n-1}, t_{n+1}:

$$L(t) = \frac{(t-t_n)(t-t_{n-1})}{(t_{n-2}-t_n)(t_{n-2}-t_{n-1})}f_{n-2} + \frac{(t-t_n)(t-t_{n-2})}{(t_{n-1}-t_n)(t_{n-1}-t_{n-2})}f_{n-1} + \frac{(t-t_{n-1})(t-t_{n-2})}{(t_n-t_{n-1})(t_n-t_{n-2})}f_n$$

Then using Simpson Rule:

$$\begin{aligned}\int_{t_n}^{t_{n+1}} f(t, y(y))dt &\approx \int_{t_n}^{t_{n+1}} L(t)dt \\ &\approx \frac{h}{6}\left[L(t_n) + 4L(\frac{t_n + t_{n+1}}{2}) + L(t_{n+1})\right] \\ &= h\left[\frac{23}{12}f_n - \frac{4}{3}f_{n-1} + \frac{5}{12}f_{n-2}\right]\end{aligned}$$

Therefore:

$$y_{n+1} = y_n + h\left[\frac{23}{12}f_n - \frac{4}{3}f_{n-1} + \frac{5}{12}f_{n-2}\right]$$

10.2 Adams-Moulton Methods

We approximate f by its interpolating at the following $(m+1)$-nodes:

$$t_{n+1}, t_n, t_{n-1}, \ldots, t_{n-(m-2)}, t_{n-(m-1)}$$

This is called m-step **Adams-Moulton Methods**. Note that these methods are **implicit**.

Example 203. Here are some examples:

1. Implicit/Backward Euler Method:

$$y_{n+1} = y_n + hf_{n+1}$$

2. Crank-Nicolson scheme/single-step Adams-Moulton method:

$$y_{n+1} = y_n + \frac{h}{2}(f_n + f_{n+1})$$

3.

$$y_{n+1} = y_n + h(\frac{5}{12}f_{n+1} + \frac{8}{12}f_n - \frac{1}{12}f_{n-1})$$

4.

$$y_{n+1} = y_n + \frac{h}{24}(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$$

5.
$$y_{n+1} = y_n + \frac{h}{720}(251f_{n+1} + 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3})$$

Algorithm 204. How to derive the **Adams-Moulton (AM) Method**, assuming currently at t_{n+1}:
Interpolating at k points:
$$t_{n+1}, t_n, t_{n-1}, \ldots, t_{n-(k-1)}$$
And get $L_k(t)$:
Then:
$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} L_k(t)dt$$

Define backward difference::
$$\nabla^0 f_{n+1} = f_{n+1}$$
$$\nabla^i f_{n+1} = \nabla^{i-1} f_{n+1} - \nabla^{i-1} f_n, \text{ for } i = 1, 2, \ldots$$

Then the interpolation is:
$$L_k(t) = \sum_{i=0}^{k} \frac{\nabla^i f_{n+1}}{i!h^i}(t - t_{n+1})(t - t_n)\ldots(t - t_{n-i+1})$$

Then:
$$y_{n+1} = y_n + \sum_{i=0}^{k} \frac{\nabla^i f_{n+1}}{i!h^i} \int_{t_n}^{t_{n+1}} (t - t_{n+1})(t - t_n)\ldots(t - t_{n-i+1})dt$$

Writing $t = t_n + sh$:
$$\begin{aligned} y_{n+1} &= y_n + h\sum_{i=0}^{k} \frac{\nabla^i f_{n+1}}{i!h^i} \int_0^1 (s-1)s\ldots(s+i-1)ds \\ &= y_n + h\sum_{i=0}^{k} (-1)^i \nabla^i f_{n+1} \int_0^1 \binom{-s+1}{i} ds \end{aligned}$$

Starting $k = 0$, then $k = 1$, do this inductively.
At $(k+1)$-step: add the term $h(-1)^{k+1}\nabla^{k+1} f_{n+1} \int_0^1 \binom{-s+1}{k+1} ds$ to the k-th step method.

10.3 Explicit and Implicit Euler's Method

Definition 205 (**Explicit Euler's Method -EEM**)**.** The iteration is
$$y_{n+1} = y_n + hf(t_n, y_n)$$

Definition 206 (**Implicit Euler's Method -IEM**)**.** The iteration is
$$y_{n+1} = y_n + hf(t_n, y_{n+1})$$

Example 207. Consider:
$$\begin{cases} x'(t) &= \alpha x(t) \\ x(0) &= 1 \end{cases}$$
The true solution is $x(t) = e^{\alpha t}$. If $\alpha < 0$, the $x(t)$ decays to 0 as $t \to \infty$.

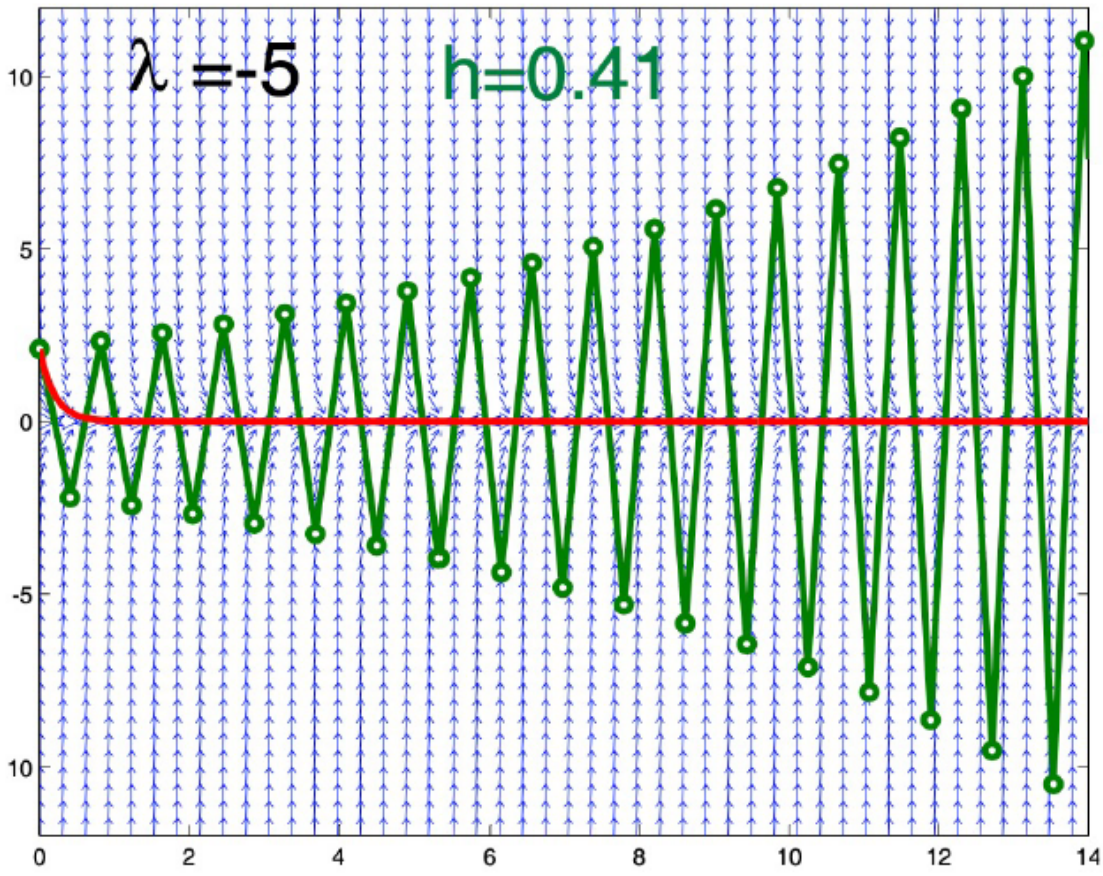


Figure 19: Explicit Euler is not stable

Remark 208. The **explicit Euler method- EEM** is **not** stable for fast decaying equations. For example with $\alpha = -5$, we need $h < -\frac{2}{\alpha} = 0.4$. When we choose $h = 0.41$, we see blow up. See figure (19) for illustration.

The iteration is

$$x_{n+1} = x_n + hf(t_n, x_n)$$

With $x_0 = 1$, $x_{n+1} = x_n + h(\alpha x_n) = (1 + h\alpha)x_n$. Therefore:

$$x_n = (1 + h\alpha)^n \approx x(t) = e^{\alpha t}$$

If $\alpha < 0$: $x(t) = e^{\alpha t} \to 0 \approx x_n = (1 + h\alpha)^n \xrightarrow{?} 0$.
If we want $x_n \to 0$, that is:

$$\begin{aligned} &|1 + h\alpha|^n \to 0 \\ \iff &|1 + h\alpha| < 1 \\ \iff &0 < h < -\frac{2}{\alpha} \end{aligned}$$

So we require the condition $h < -\frac{2}{\alpha}$, otherwise, if $h > -\frac{2}{\alpha}$, $x_n \not\to 0$ and we get blow up.

Remark 209. The **IEM- Implicit Euler Method** is better because:
Consider the iteration:

$$x_{n+1} = x_n + hf(t_{n+1}, x_{n+1})$$

With $x_0 = 1$ and $x_{n+1} = h\alpha x_{n+1}$:

$$\begin{aligned} x_{n+1} &= \frac{1}{1 - h\alpha} x_n \\ x_n &= \frac{1}{(1 - h\alpha)^n} \xrightarrow{?} 0 \end{aligned}$$

Since $\alpha < 0$, then:

$$\left|\frac{1}{(1 - h\alpha)}\right| < 1 \iff h > 0$$

And this is always true. So this method is better.

Example 210. Consider:

$$\begin{cases} x'(t) &= \lambda x(t) \\ x(o) &= 1 \end{cases}$$

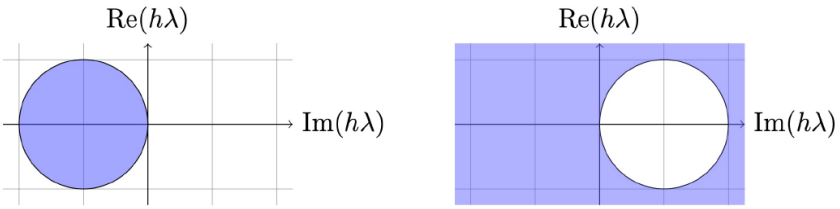
where $\lambda = \mu + i\nu \in \mathbb{C}$.
Then the solution is:

$$x = e^{\alpha t} = e^{\mu t} e^{i\nu t} = e^{\mu t}\Big(\cos(\nu t) + i\sin(\nu t)\Big)$$

If $\mathrm{Re}(\lambda) = \mu < 0$: $x(t)$ decays to zero as $t \to \infty$. See figure (20) for comparison.

From the graph we see that:

1. Implicit Euler is more desirable as their stability region contains all $h\lambda$ where $\mathrm{Re}(\lambda) < 0$.
2. **Using explicit Euler Method, we need to choose small step size h.**



Stability region for explicit Euler (left) and implicit Euler (right) methods to the test problem $\dot{x} = \lambda x$.

Figure 20: stability for explicit and implicit euler method

10.4 Crank-Nicolson Scheme (single-step Adams-Moulton Method)

The iteration is

$$y_{n+1} = y_n + \frac{h}{2}\Big[f(t_{n+1}, y_{n+1}) + f(t_n, y_n)\Big]$$

If f is **non-linear**, how to solve for y_{n+1}.
Idea: use iterative methods to compute y_{n+1}. There are **Newton's method** and **Fixed-point Iterative Method**.

Definition 211 (**Newton's Method**)**.** Introduce a nonlinear function:

$$\Phi(y) = y - \left[y_n + \frac{h}{2}\left(f(t_{n+1}, y) + f(t_n, y_n)\right)\right]$$

Then y_{n+1} is given by $\Phi(y) = 0$. From Newton method:

$$y_{n+1}^{(k+1)} = y_{n+1}^{(k)} - \frac{\Phi(y_{n+1}^{(k)})}{\Phi'(y_{n+1}^{(k)})} \text{ for } k = 0, 1, 2, \ldots$$

Newton's method converges **locally**. Find a good initial guess $y_{n+1}^{(0)}$.

Definition 212 (**Fixed-point Iterative Method**)**.** Introduce the iterative function:

$$\Psi(y) = y_n + \frac{h}{2}\left(f(t_{n+1}, y) + f(t_n, y_n)\right)$$

Find the fixed-point of $\Psi(y)$. The fixed-point iteration:

$$y_{n+1}^{(k+1)} = \Psi(y_{n+1}^{(k)}) \text{ for } k = 0, 1, 2, \ldots$$

This method converges **globally** under certain conditions on $f(t, x)$.

10.5 Predictor Corrector Method

Idea: There are 2 steps:

1. Take AB (Adams-Bashforth) as a **predictor**.
2. Take AM (Adams-Moulton) as a **corrector**.

Example 213. We can use:

1. 2-step AB Method, **predictor**:

$$\tilde{y}_{n+1} = y_n + h\left(\frac{3}{2}f(t_n, y_n) - \frac{1}{2}f(t_{n-1}, y_{n-1})\right)$$

2. 2-step AB Method, **corrector**:

$$y_{n+1} = y_n + h\left(\frac{5}{12}f(t_{n+1}, \tilde{y}_{n+1}) + \frac{8}{12}f(t_n, y_n) - \frac{1}{12}f(t_{n-1}, y_{n-1})\right)$$

We use the $\tilde{y}_{n+1}$ obtained from the predictor to use in the corrector.

10.6 Local Truncation Error of Multi-step Methods

Recall a **general m-step method** is defined by:

$$\sum_{j=0}^{m} a_j y_{k+j} = h \sum_{j=0}^{m} b_j f(t_{k+j}, y_{k+j})$$

where the $y_0, y_1, \ldots, y_{k+m-1}$ are given. Without loss of generality, $a_m = 1$. The method is **explicit** if $b_m = 0$, and it is **implicit** if $b_m \neq 0$ (i.e, finding y_{k+m} involves solving an equation or not).

Definition 214 (**local truncation error - consistent-order**)**.** The **local truncation error** of the m-step general method is:

$$\tau_{k+m}(h) = \frac{1}{h}\sum_{j=0}^{m} a_j y(t_{k+j}) - \sum_{j=0}^{m} b_j f\Big(t_{k+j}, y(t_{k+j})\Big)$$

The multistep method is **consistent** if:

$$\lim_{h\to 0} |\tau_{k+m}(h)| = 0$$

The method is **of order** p if p is the largest number such that:

$$\tau_{k+m}(h) = O(h^p)$$

Theorem 215. *A linear multistep is **consistent** if and only if:*

$$\sum_{j=0}^{m} a_j = 0, \text{ and } \sum_{j=0}^{m} j a_j = \sum_{j=0}^{m} b_j$$

The general m-step method is of order $p \geq 1$ if and only if:

$$\begin{cases} \sum_{j=0}^{m} a_j = 0 \\ \sum_{j=0}^{m} j^{\alpha} a_j = \alpha \sum_{j=0}^{m} j^{\alpha-1} b_j \ \ \alpha = 1, 2, \ldots, p \\ \sum_{j=0}^{m} j^{p+1} a_j \neq (p+1) \sum_{j=0}^{m} j^p b_j \end{cases}$$

Proof. We get the truncation error:

$$\begin{aligned}
\tau_{k+m}(h) &= \frac{1}{h}\sum_{j=0}^{m} a_j y(t_{k+j}) - \sum_{j=0}^{m} b_j f\Big(t_{k+j}, y(t_{k+j})\Big) \\
&= \sum_{j=0}^{m} a_j \frac{y(t_k + jh)}{h} - \sum_{j=0}^{m} b_j y'(t_k + jh) \\
&\overset{\text{Taylor}}{=} \sum_{\alpha=2}^{N}\left[\frac{1}{\alpha!}\sum_{j=0}^{m} a_j y^{(m)}(t_k) j^{\alpha} h^{\alpha-1}\right] + \sum_{\alpha=1}^{N}\left[\frac{1}{(\alpha-1)!}\sum_{j=0}^{m} b_j y^{(m)}(t_k) j^{\alpha-1} h^{\alpha-1}\right] + O(h^N) \\
&= \left(\sum_{j=0}^{m} a_j\right)\frac{y(t_k)}{h} + \sum_{\alpha=1}^{N}\left(\sum_{j=0}^{m} a_j j^{\alpha} - b_j j^{\alpha-1}\alpha\right) y^{(m)}(t_k) h^{\alpha-1} + O(h^N) \\
&= (\ldots)\frac{1}{h} + (\ldots)1 + (\ldots)h + \ldots + (\ldots)h^{N-1} + O(h^N)
\end{aligned}$$

In order for it to be consistent, we want $\tau \xrightarrow{h\to 0} 0$, that is, we want the two blue terms to be both zero:

$$\sum_{j=0}^{m} a_j = 0 \text{ and } \sum_{j=0}^{m} a_j.j = \sum_{j=0}^{m} b_j$$

In order for it to have order p, we want:

1. All the terms $\frac{1}{h}, 1, h, h^2, \ldots, h^{p-1}$ have coefficients $= 0$:

$$\begin{aligned}
\sum a_j &= 0 \\
\sum_{j=0}^{m} j^{\alpha} a_j &= \alpha \sum_{j=0}^{m} j^{\alpha-1} b_j \text{ ,where } \alpha = 1, 2, \ldots, p
\end{aligned}$$

2. h^p has coefficient non-zero:

$$\sum_{j=0}^{m} j^{p+1} a_j \neq (p+1)\sum_{j=0}^{m} j^{p} b_j$$

Which is precisely what we required in the theorem. □

Example 216. Show that for 2-step AB method has order 2:

$$y_{k+2} - y_{k+1} = h\left[\frac{3}{2}f(t_{k+1}, y_{k+1}) - \frac{1}{2}f(t_k, y_k)\right]$$

We use the theorem:
First note that we get $a_0 = 0, a_1 = -1, a_2 = 1$ and $b_0 = -\frac{1}{2}, b_1 = \frac{3}{2}, b_2 = 0$.
So we get:

1. $\sum a_j = 0 + (-1) + 1 = 0$
2. $\sum j a_j = 1(-1) + 2(1) = 1 = \sum b_j = -\frac{1}{2} + \frac{3}{2}$

These two conditions give consistence.
Now for order 2 we see:

1. $\alpha = 1$: $\sum ja_j = 1(-1) + 2(1) = 1 = 1\sum j^0 b_j = \sum b_j = -\frac{1}{2} + \frac{3}{2}$
2. $\alpha = 2$: $\sum j^2 a_j = 2\sum jb_j = 3$
3. $\alpha = 3$: The $\sum j^3 a_j = 1(-1) + 2^3(1) = 7 \neq \sum j^2 b_j = 3(1.\frac{3}{2}) = \frac{9}{2}$

Therefore this has order 2.

Example 217. Show from definition that 2-step AB method has order 2:

$$y_{k+2} - y_{k+1} = h\left[\frac{3}{2}f(t_{k+1}, y_{k+1}) - \frac{1}{2}f(t_k, y_k)\right]$$

Recall that $\tau_{k+m}(h) = O(h^2)$. We need to check that $\tau_{k+2}(h) = O(h^2)$:
The definition is:

$$\tau_{k+2}(h) = \frac{1}{h}y(t_{k+2}) - \frac{1}{h}y(t_{k+1}) - \frac{3}{2}y'(t_{k+1}) + \frac{1}{2}y'(t_k)$$

Let's compute each term in the definition and then add (combine) them together:

$$\begin{aligned}
\frac{1}{h}y(t_{k+2}) &\overset{\text{Taylor}}{=} \frac{1}{h}y(t_k) + 2y'(t_k) + 2y''(t_k) + \frac{4}{3}h^2y'''(t_k) + O(h^3) \\
-\frac{1}{h}y(t_{k+2}) &\overset{\text{Taylor}}{=} -\frac{1}{h}y(t_k) - y'(t_k) - \frac{h}{2}y''(t_k) - \frac{1}{6}h^2y'''(t_k) + O(h^3) \\
-\frac{3}{2}y'(t_{k+1}) &\overset{\text{Taylor}}{=} 0 + -\frac{3}{2}y'(t_k) - \frac{3}{2}hy''(t_k) - \frac{3}{4}h^2y'''(t_k) + O(h^3) \\
\frac{1}{2}y'(t_k) &\overset{\text{Taylor}}{=} 0 + \frac{1}{2}y'(t_k)
\end{aligned}$$

Add them all we get:

$$\tau_{k+2}(h) = \frac{0}{h} + 0 + 0h + \frac{5}{12}h^2y'''(t_k) + O(h^3)$$

So it is of order 2.

11 Stability

Definition 218 (stable)**.** A numerical method to solve IVP is **stable** if there is a constant K and step size $h_0 > 0$ such that the difference between 2 solutions y_i and $\tilde{y}_i$ to the IVP with initial values y_0 and $\tilde{y}_0$ satisfying:

$$|y_i - \tilde{y}_i| \leq K|y_0 - \tilde{y}_0| \text{ for all } i = 0, 1, \ldots, N$$

whenever $h \leq h_0$ and $Nh \leq b - a$.

Remark 219. Recall explicit and implicit Euler method: the Explicit method is **not** stable for fast decaying equations.

Definition 220 (**characteristic polynomial**)**.** Given a multi-step of the form:

$$y_0 = \alpha_0, y_1 = \alpha_1, \ldots, y_{m-1} = \alpha_{m-1}$$

and:

$$y_{i+1} = a_{m-1}w_i + \ldots + a_0 w_{i+1-m} + h\Big[b_m f(t_{i+1}, y_{i+1} + \ldots + b_0 f(t_{i+1-m}, y_{i+1-m}))\Big]$$

We define the **characteristic polynomial** as:

$$P(\lambda) = \lambda^m - a_{m-1}\lambda^{m-1} - a_{m-2}\lambda^{m-2} - \ldots - a_1\lambda - a_0$$

Definition 221 (**Root condition**)**.** A multi-step method satisfying the **Root condition** if the roots λ_i of the associated characteristic polynomial $P(\lambda)$ satisfying:

1. $|\lambda_i| \leq 1$ for any $i = 1, 2, \ldots, m$
2. all roots with magnitude 1 are simple roots, i.e., if $P(\lambda_i) = 0$ and $|\lambda| = 1$, then $P'(\lambda_i) \neq 0$.

Definition 222 (**stability**)**.** A multi-step method with the associated characteristic polynomial $P(\lambda)$ is called:

1. **zero-stable** if it satisfying the root condition
2. **strongly stable** if it satisfies the root condition and $P(\lambda)$ has $\lambda = 1$ as the only root with magnitude one
3. **weakly stable** if it satisfies the root condition and $P(\lambda)$ has more than one distinct roots with magnitude one
4. **unstable** if it does not satisfy the root condition.

Example 223. Show that the 4-th order Milne's method

$$y_{i+1} = y_{i-3} + \frac{4h}{3}\Big[2f(t_i, y_i) - f(t_{i-1}, y_{i-1}) + 2f(t_{i-2}, y_{i-2}) - \Big]$$

is only weakly stable. We first observe that:

$$y_{i+1} = 0y_i + 0y_{i-1} + 0y_{i-2} + 1y_{i-3}$$

So we get our Characteristic polynomial is:

$$\begin{aligned} P(\lambda) &= \lambda^4 - 1 \\ &= (\lambda - 1)(\lambda + 1)(\lambda + i)(\lambda - i) \end{aligned}$$

So we get 4 roots: $\lambda_1 = 1, \lambda_2 = -1, \lambda_3 = i, \lambda_4 = -i$.
The root condition is satisfied:

1. all $|\lambda_i| \leq 1$.
2. all roots with magnitude 1 are simple.

It has more than 1 distinct roots with magnitude 1, therefore it is weakly stable.

Example: $y'(t) = -15y$ for $0 \leq t \leq 0.5$ and $y(0) = 1$. Exact solution: $y(t) = e^{-15t}$.

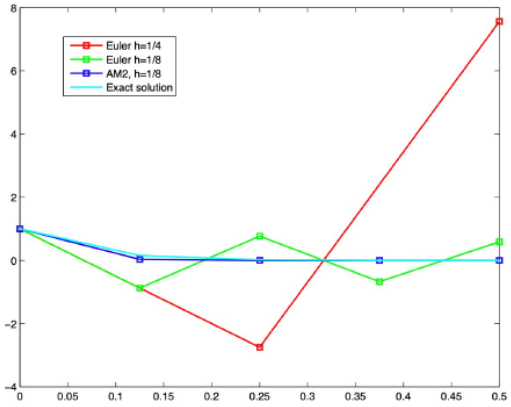


Figure 21: Sniff equation $y' = -15y$

12 Stiff Equations

Definition 224 (**Stiff Equation**)**. Stiff Equation** are differential equations for which some numerical methods are unstable, unless the step size is extremely small. They are characterized:

1. The derivative grows as $t \to \infty$ while the actual solution does not grow in magnitude.
2. The exact solution has a fast decaying term, of the form e^{-ct} where $c > 0$ is large.
3. Different components of the IVP system evolve on different time scales.

The fast decaying part of the solution is called the **transient** solution, and the rest ("more important part") of the solution is called the **steady-state** solution.

Example 225. The IVP $y'(t) = -15y$ fir $0 \leq t \leq 0.5$ and $y(0) = 1$ has exact solution $y(t) = e^{-15t}$. See figure (21) for illustration.

Example 226. The IVP is :

$$\begin{cases} y' & = -\lambda y + e^{-t} \text{ where } \lambda \ll 1 \\ y(0) & = 0 \end{cases}$$

Then the exact solution is:

$$y(t) = \frac{1}{\lambda - 1}(e^{-t} - e^{-\lambda t})$$

where the e^{-t} is steady state and the $e^{-\lambda t}$ is fast decaying-transient.

Definition 227 (**Test IVP**)**.** The Test IVP is:

$$y' = \lambda y, \ t \geq 0, \ , y(0) = \alpha, \ , \lambda < 0$$

Definition 228 (**region of absolute stability**)**.** The **region of absolute stability** R of a method is the set:

$$R = \{h\lambda \in \mathbb{C} : y_i \to 0, \text{ as } i \to \infty\}$$

where the $\{y_i\}$ is generated b applying the method to the test IVP.

Definition 229. A method is:

1. **A-stable** if its region of absolute stability R contains the entire left half plane.
2. **L-stable** if it is A-stable and at each fixed t_i, the numerical solution y_i satisfy $y_i \to 0$ as $\text{Re}(\lambda) \to -\infty$.

Example 230. The region of absolute stability of Euler method is:

$$R = \{h\lambda : |1 + h\lambda| < 1\} \Longrightarrow h < \frac{2}{|\lambda|}$$

To see this, we apply explicit Euler Method to the test IVP:

$$\begin{aligned} y_{i+1} &= y_i + h(\lambda y_i) \\ \Longrightarrow y_i &= (1 + h\lambda)^i \alpha \end{aligned}$$

The exact solution is $y(t) = \alpha e^{\lambda t}$. Therefore the global error at t_i:

$$|y(t_i) - y_i| = |e^{ih\lambda} - (1 + h\lambda)^i| . |\alpha|$$

If $\lambda < 0$ then $e^{ih\lambda} \xrightarrow{i\to\infty} 0$. So in order to ensure that $|y(t_i) - y_i| \to 0$, we need $|1 + h\lambda| < 1$, which gives $h < \frac{2}{|\lambda|}$.

Theorem 231. *A one-step method $y_{i+1} = y_i + h\Phi(t_i, y_i, h)$, which can be rewritten as $y_{i+1} = Q(h\lambda)y_i$, has the region of absolute stability:*

$$R = \{h\lambda \in \mathbb{C} : |Q(h\lambda)| < 1\}$$

Here the function $Q(z)$ is called the ***stability function****.*

Remark 232. For the L-stability of one-step methods, $y_i \to 0$ as $\text{Re}(\lambda) \to -\infty$ is equivalent to $|Q(h\lambda)| \to 0$ as $\text{Re}(\lambda) \to -\infty$.

Consider a multi-step method:

$$y_0 = \alpha_0, y_1 = \alpha_1, \ldots, y_{m-1} = \alpha_{m-1}$$

and:

$$y_{i+1} = a_{m-1}y_i + \ldots + a_0 y_{i+1-m} + h\Big[b_m f(t_{i+1}, y_{i+1} + \ldots + b_0 f(t_{i+1-m}, y_{i+1-m}))\Big]$$

Apply this method to test IVP:

$$\begin{aligned} y_{i+1} &= a_{m-1}y_i + \ldots + a_0 y_{i+1-m} + h\lambda\Big[b_m y_{i+1} + \ldots b_0 y_{i+1-m}\Big] \\ &= h\lambda y_{i+1} + (a_{m-1} + h\lambda b_{m-1})y_i + \ldots + (a_0 + h\lambda b_0)y_{i+1-m} \\ &= h\lambda b_m y_{i+1} + \sum_{j=1}^{m} (a_{m-j} + h\lambda b_{m-j}) y_{i-j+1} \end{aligned}$$

Definition 233 (Stability polynomial)**.** The **Stability polynomial** $Q(z, h\lambda)$ is:

$$Q(z, h\lambda) = (1 - h\lambda b_m)z^m - \sum_{j=1}^{m}(a_{m-j} + h\lambda b_{m-j})z^{m-j}$$

The numerical solution y_i can be expressed as a linear combination of β_i^k with $k = 1, 2, \ldots, m$ where β_k are roots of $Q(z, h\lambda)$ for a fixed λ. To ensure the condition $|y_i| \to 0$, we need the region of absolute stability:

$$R = \{h\lambda \in \mathbb{C} : |\beta_k| < 1. \text{ for all zeros } \beta_k \text{ of } Q(z, h\lambda)\}$$

Example 234. The Implicit Trapezoidal method:

$$\begin{aligned} y_0 &= \alpha \\ y_{i+1} &= y_i + \frac{h}{2}\Big[f(t_{i+1}, y_{i+1}) + f(t_i, y_i)\Big],\ 0 \le j \le N-1 \end{aligned}$$

is A-stable but not L-stable. To see this, we apply method to test IVP:

$$y_{i+1} = y_i + \frac{h\lambda}{2}\Big[y_{i+1} + y_i\Big] = \frac{h\lambda}{2}y_{i+1} + (1 + \frac{h\lambda}{2})y_i$$

The stability polynomial is then:

$$Q(z, h\lambda) = (1 - \frac{h\lambda}{2})z - (1 + \frac{h\lambda}{2})$$

The root is then:

$$z = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} = \frac{2 + h\lambda}{2 - h\lambda}$$

The region of absolute stability is then:

$$R = \{h\lambda \in \mathbb{C} : \left|\frac{2 + h\lambda}{2 - h\lambda}\right| < 1\}$$

If we denote $h\lambda = a + ib \in \mathbb{C}$, then the inequality above is:

$$\begin{aligned} \left|\frac{2 + h\lambda}{2 - h\lambda}\right| < 1 &\iff |2 + a + ib| < |2 - a - ib| \\ &\iff (2 + a)^2 + b^2 < (2 - a)^2 + b^2 \\ &\iff a - \mathrm{Re}(h\lambda) < 0 \end{aligned}$$

Therefore it is A-stable.
On the other hand:

$$\begin{aligned} y_{i+1} &= \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} y_i \\ \implies y_i &= \left(\frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}}\right)^i \alpha \xrightarrow{\lambda \to -\infty} (-1)^{2i}\alpha \neq 0 \end{aligned}$$

So it is not L-stable.

13 Systems of First-Order IVPS

Consider the system:

$$\begin{cases} y_j'(t) & = f_j(t, y_1, \ldots, y_m), \text{ for } j = 1, 2, \ldots, m, \ t \in [a, b] \\ y_j(a) & = \alpha_j, \text{ for } j = 1, 2, \ldots, m \end{cases}$$

We introduce an m-dimensional vector function $Y(t) = [y_1(t), \ldots, y_m(t)]^T : [a, b] \to \mathbb{R}^m$ and rewrite the above IVP system:

$$\begin{cases} Y'(t) & = F(t, Y) \ t \in [a, b] \\ Y(a) & = \alpha \end{cases}$$

where $F(t, Y) : [a, b] \times \mathbb{R}^m \to \mathbb{R}^m, \alpha = [\alpha_1, \ldots, \alpha_m]^T$.
All methods are the same with the case of 1 IVP.

Example 235. Let $h = 0.5$ and apply the Euler's method to solve the IVP system:

$$\begin{cases} y_1' = y_2, \ y_2' = -y_1, \ 0 \le t \le 1 \\ y(1) = 0, \ y_2(0) = 0 \end{cases}$$

We let:

$$Y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}. \ F(t, Y) = \begin{bmatrix} y_2 \\ -y_1 \end{bmatrix}, \ Y_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Then:

$$Y_1 = Y_0 + hF(t_0, Y_0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.5 \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -0.5 \end{bmatrix}$$

and:

$$Y_2 = Y_1 + hF(t_1, Y_1) = \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} + 0.5 \begin{bmatrix} -0.5 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.75 \\ -1 \end{bmatrix}$$