Numerical Computing Notes

Indian Statistical Institute, Bangalore

Bachelors of Mathematics
First Year(Batch 2021-2024)

Contents

Lecture 14
February 21, Part A

Finding a root of a function $f : \mathbb{R} \to \mathbb{R}$, i.e., to say a root of the equation

$$f(x) = 0$$

is a common problem in applied mathematics. We will discuss over how to numerically solve the equation for functions with a closed form, along with some other conditions.

Iteration Methods

All the numerical methods which we will use for approximating the root of the equation will be iteration methods. In iteration methods we compute a sequence of increasingly accurate estimate of the root of the equation $f(x) = 0$.

14.1 Bisection Method

Suppose we have a function $f : \mathbb{R} \to \mathbb{R}$, such that

- $f \in \mathscr{C}[a, b]$, i.e., f in continuous on the closed set $[a, b]$.
- $f(a)f(b) < 0$, i.e., $f(a)$ and $f(b)$ have opposite signs.

Then note that from **Intermediate Value Theorem**[1], we have, there exists at least one $\alpha \in [a, b]$ such that $f(\alpha) = 0$. In this setup we can use **bisection method** to find a root to the equation $f(x) = 0$, to our precise degree of accuracy. We will now design an algorithm for the **bisection method**.

14.1.1 Algorithm for Bisection Method

Our function will take 4 inputs:

- The continuous function f.
- Points a and b in the domain of f such that $a < b$, and we have $f(a)f(b) < 0$.
- An *error tolerance* level $\varepsilon > 0$. The *error tolerance* level will indicate our function when to stop the iteration process.

The bisection method will consist of the following steps:

Step 1. Define $c = \frac{1}{2}(a + b)$.
Step 2. If $b - c \leq \varepsilon$, then terminate the iteration process and return c as the root.
Step 3. If $\text{sgn}(f(b)) \cdot \text{sgn}(f(c)) \leq 0$, then set $a = c$, else set $b = c$, and return to **Step 1**.

[1]**Intermediate Value Theorem:** Let $f : [a, b] \to \mathbb{R}$, be a continuous function, and let η be any real number between $f(a)$ and $f(b)$, then there exists a $c \in [a, b]$ such that $f(c) = \eta$.

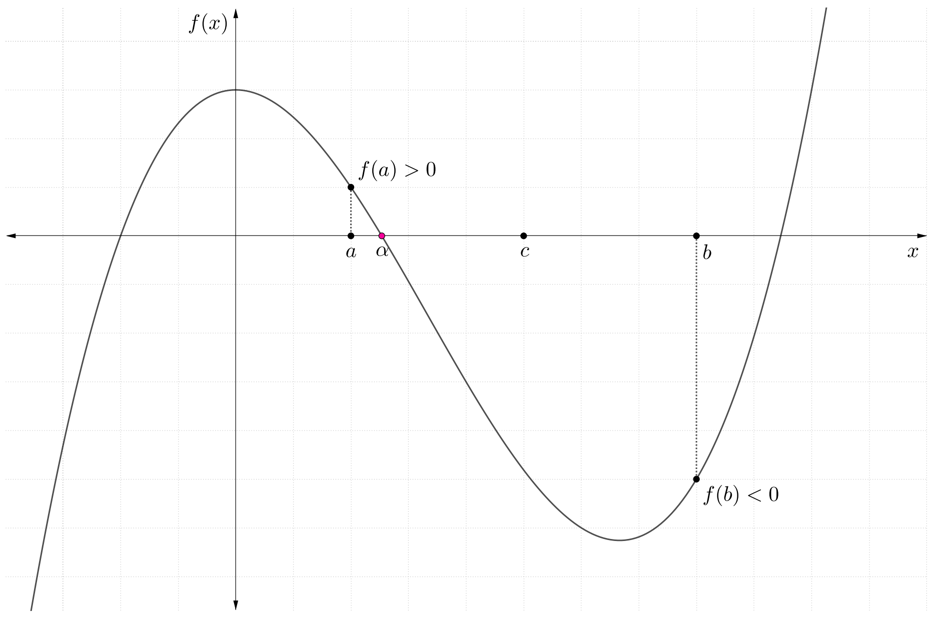

Figure 14.1.1: Bisection Method

Now, we will show that if f is a continuous function on the set $[a, b]$, where a and b are points such that $f(a)f(b) < 0$, then bisection method is guaranteed to converge to a root of f.

14.1.2 The Bisection Method is guaranteed to converge to a root

Let a_n, b_n and c_n be the n^{th} computed values of a, b and c respectively and we have $a_1 = a, b_1 = b$. And let α be the true root of the function f, i.e.,

$$f(\alpha) = 0$$

Then observe that

$$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n),\ \forall n \in \mathbb{N} \tag{14.1.1}$$

and hence from equation (14.1.1), and using induction we easily get that

$$b_n - a_n = \frac{1}{2^{n-1}}(b - a),\ \forall n \in \mathbb{N} \tag{14.1.2}$$

Now at the n^{th} iteration, we will have, $c_n = \frac{1}{2}(a_n + b_n)$, and since throughout the process we have either $f(a_n)f(c_n) \leq 0$ or $f(c_n)f(b_n) \leq 0$, thus α either lies in the interval $[a_n, c_n]$ or in the interval $[c_n, b_n]$, in either case, we since

$$c_n - a_n = b_n - c_n = \frac{1}{2}(b_n - a_n)$$

we get that

$$|\alpha - c_n| \leq \frac{1}{2}(b_n - a_n)$$

and then using equation (14.1.2), we get that

$$|\alpha - c_n| \leq \frac{1}{2^n}(b - a) \tag{14.1.3}$$

and hence since

$$\lim_{n\to\infty} \frac{1}{2^n}(b - a) = 0$$

we deduce that $c_n \to \alpha$, as $n \to \infty$, hence the **bisection method** guarantees that eventually our estimate will converge to a root of f. Now ofcourse our function can not run for enternity, so we must terminate it at a certain point this is where **Step 2** is necessary. But the next question that arises is how many iterations would we need to reach to our desired root?

14.1.3 How many iterations do we need?

Theorem 14.1.1. Let n be the number of iterations required to a root within our desired error tolerance level $\varepsilon > 0$, then we have

$$n \geq \frac{\ln\left(\frac{b-a}{\varepsilon}\right)}{\ln 2}$$

where a and b, are the endpoints of our initial interval.

The number iterations required, is equivalent to finding the n such that

$$|\alpha - c_n| \leq \varepsilon$$

But from equation (14.1.3), this is equivalent to finding n such that

$$\frac{1}{2^n}(b-a) \leq \varepsilon \Rightarrow \frac{b-a}{\varepsilon} \leq 2^n$$

taking logarithm on both sides we get that we must have

$$n \geq \frac{\ln\left(\frac{b-a}{\varepsilon}\right)}{\ln 2} \tag{14.1.4}$$

14.1.4 Pros and Cons of Bisection Method

Pros:

- The number of iterations, i.e., n can be estimated.
- Is guaranteed to converge to a root of the function.

Cons:

- The algorithm converges to a desired root more slowly than other algorithms.
- To we must find a and b such that $\alpha \in [a, b]$, which can be sometimes difficult to find.

Lecture 15
February 21, Part B

15.1 Newton's Method/Newton-Raphson Method

Unlike the Bisection Method, here we don't have to evaluate f to find the appropriate a and b, which is a good thing. However, Newton's Method, as we will see, is not guaranteed to converge, which is a bad thing. We will also see that it depends crucially on the selection of x_0. Moreover, we will need the function to be differentiable in our domain of interest.

15.1.1 Algorithm

First, we make an estimate of the root α of f, which we shall denote by x_0. Consider the equation of the line tangent to the graph of $y = f(x)$ at $(x_0, f(x_0))$

$$p_1(x) = f(x_0) + f'(x_0)\,(x - x_0)\,.$$

This is just the linear Taylor polynomial for f at x_0.

Define x_1 to be the root of $p_1(x) = 0$. We solve

$$p_1(x_1) = f(x_0) + f'(x_0)(x - x_0) = 0.$$

for x_1 to get

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Repeating this procedure, we get x_2 from x_1 as

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

Proceeding inductively, we get a sequence $\{x_n\}$ according to the following recusion formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_{n+1})}, n \geq 0 \tag{15.1.1}$$

The sequence $\{x_n\}$ is a sequence of estimates for α. This procedure of estimating the root α using the recursion formula (15.1.1) is known as the **Newton-Raphson Method**.

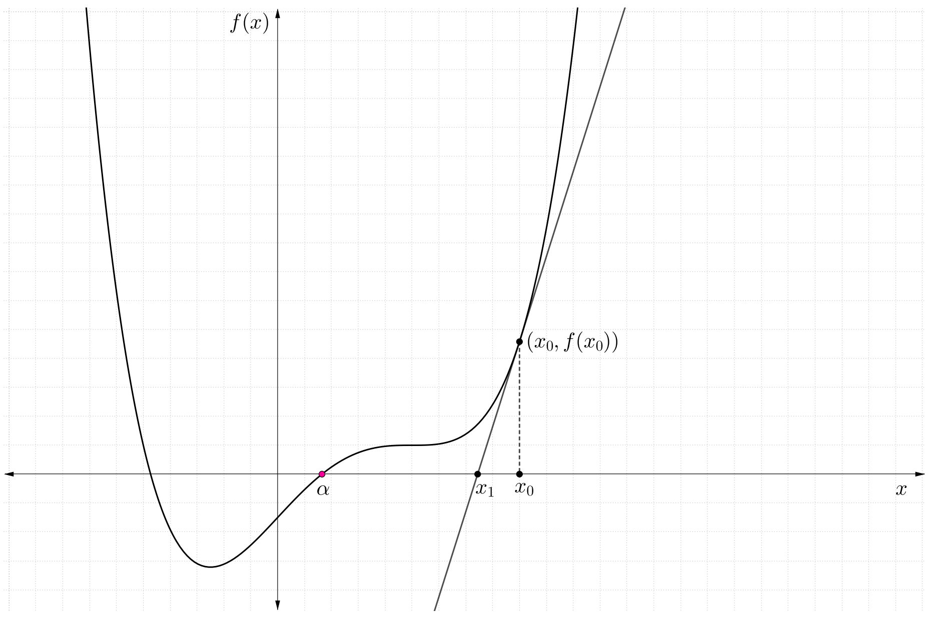

Figure 15.1.1: Newton Raphson Method

15.1.2 An example computation problem

Now, we take a look at an example computation problem that can be solved using the Newton-Raphson Method.

Given $a, b \in \mathbb{N}$, we want to compute $\frac{a}{b}$ without performing the division operation.

Early computers could support addition, subtraction and multiplication but division needed to be implemented using an algorithm as we are going to discuss. In order to solve this problem, we take the inputs a, b, following which we can get the answer by multiplying a and $\frac{1}{b}$. For this, we need to compute $\frac{1}{b}$, which can be done by solving

$$f(x) = b - \frac{1}{x} = 0 \tag{15.1.2}$$

Remember that here we are trying to estimate the root $\alpha = \frac{1}{b}$ of the function f defined in (15.1.2) . We have $f'(x) = \frac{1}{x^2}$. So, by the Newton's Method, the recursion that we get is

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

which on simplifying gives us

$$x_{n+1} = x_n \left(2 - bx_n\right), n \geq 0 \tag{15.1.3}$$

Notice that the arithmetic of (15.1.3) involves only subtraction and multiplication which were supported in the early computers.

Now, we look at the error analysis of this procedure. From $\epsilon_{x_n} = \frac{\alpha - x_n}{\alpha}$, we easily get

$$\epsilon_{x_{n+1}} = \epsilon_{x_n}^2 \implies \epsilon_{x_n} = \left(\epsilon_{x_0}\right)^{2^n} \tag{15.1.4}$$

According to (15.1.4), the relative error of x_n,i.e. ϵ_{x_n}, can rapidly decrease to 0, as n increases if we can ensure $|\epsilon_{x_0}| < 1$.

So, we want $|\frac{\alpha - x_0}{\alpha}| < 1$.

- If $x_0 > \alpha$, then $\frac{x_0 - \alpha}{\alpha} < 1 \implies x_0 < 2\alpha = \frac{2}{b}$

- If $x_0 < \alpha$, then $\frac{\alpha - x_0}{\alpha} < 1 \implies x_0 > 0$

This gives us the equivalent condition $0 < x_0 < \frac{2}{b}$ Therefore, the Newton-Raphson Method guarantees convergence to $\alpha = \frac{1}{b}$ if and only if x_0 satisfies the above condition.

At the lowest levels, this procedure is how computation is carried out by some computers even today.

Lecture 16
February 28, Part A

16.1 Error Analysis of the Newton Raphson method

We begin by assuming that we are to use the Newton Raphson method to find a root α of a function f (**which we have a closed form expression for**). We start with an initial guess of α, which we denote by x_0. The iteration used in the Newton Raphson method is

Definition 16.1.1.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \forall(n \geq 0)$$

We can calculate $f(x_n)$ and $f'(x_n)$ because we have a closed form expression for f.

Note that if any of the $f'(x_n)$'s are 0, the iteration immediately fails at that stage. This is a flaw present in the Newton Raphson method that the bisection method doesn't suffer from.

In order to carry out the error analysis, we have to make some assumptions which we list below.

- There exists a $\rho > 0$ such that f is continuously differentiable atleast twice in $[\alpha - \rho, \alpha + \rho]$. This assumption must be taken on faith.
- $f'(\alpha) \neq 0$. If this assumption is not true, it is not possible for the iteration to converge to α, as the term $\frac{f(x_n)}{f'(x_n)}$ would blow up to infinity if it did.

Note that the two assumptions listed above imply that $f' \neq 0$ in some neighbourhood of α.

Next, we assume that for some n x_n is sufficiently close to α. Sufficiently close as in sufficiently close for the manipulations that follow. Then, using a Taylor Expansion (we can do that because we have a closed form expression for f), and cutting it off after 3 terms, we can write

$$\begin{aligned}
f(\alpha) &= f(x_n) + (\alpha - x_n)f'(x_n) + \frac{1}{2}(\alpha - x_n)^2 f''(x_n) \\
\implies \quad 0 &\overset{(1)}{=} f(x_n) + (\alpha - x_n)f'(x_n) + \frac{1}{2}(\alpha - x_n)^2 f''(x_n) \\
\implies \quad 0 &\overset{(2)}{=} \frac{f(x_n)}{f'(x_n)} + (\alpha - x_n) + (\alpha - x_n)^2 \frac{f''(x_n)}{2f'(x_n)}
\end{aligned}$$

where equality (1) follows from the fact that α is a root of f, equality (2) is obtained by dividing both sides of the equation by $f'(x_n)$, which we can do since $f'(x_n)$ is close to $f'(\alpha) \neq 0$, which in turns hold because f' is continuous (by assumption) and x_n is sufficiently close to α. Using equation 16.1.1, we obtain

Corollary 16.1.0.1.

$$\alpha - x_{n+1} = (\alpha - x_n)^2 \left[-\frac{f''(x_n)}{2f'(x_n)} \right]$$

Therefore we may say that

$$\text{Error in } x_{n+1} \sim (\text{Error in } x_n)^2$$

where $\sim$ stands for proportionality. (Note that we are assuming that $f'(x_n) \neq 0$). For convergence, $\frac{f''(x_n)}{2f'(x_n)}$ must not be too big.

Next, we run into yet another problem: We cannot compute either side of equation 16.1.0.1, as we don't know the value of α. Which brings us to the next section.

16.2 Bounding some Parameters

Although we don't know the value of α, we are working with an x_n which is "sufficiently close" to it. In that case, since f'' and f' are continuous at $\alpha \in [\alpha - \rho, \alpha + \rho]$, we have

$$M := -\frac{f''(\alpha)}{2f'(\alpha)} \approx -\frac{f''(x_n)}{2f'(x_n)}$$

Rewriting 16.1.0.1, we get

$$\alpha - x_{n+1} \approx (\alpha - x_n)^2 M$$
$$\implies M(\alpha - x_{n+1}) \approx [M(\alpha - x_n)]^2$$

Now, we carried out this analysis assuming that the iteration converges. If we also assume that iterates do not again stray far from α after they get close to it (this is possible because for convergence only the behaviour in the long run matters), and if we assume that x_0 is sufficiently close to α (previously we'd let x_n be sufficiently close to α for some n whose value was unknown to us), using induction we get

Theorem 16.2.1.

$$M(\alpha - x_n) \approx [M(\alpha - x_0)]^{2^n}$$

Since the iteration converges, it cannot be the case that $|M(\alpha - x_0)| \geq 1$, which implies that

Theorem 16.2.2. Under the conditions required for 16.2.1 to hold,

$$|M(\alpha - x_0)| < 1 \implies |\alpha - x_0| < \frac{1}{|M|}$$

The above theorem implies, among other things, that if $|M|$ is very large, our initial guess x_0 will have to be very small, for the error analysis in theorem 16.2.1 to hold. Intuitively, f should not be "flat" around the root. Newton Raphson will fail often and fail miserably if $f''(\alpha)$ is finite, and $f'(\alpha) = 0$, which happens even when you're dealing with relatively ordinary functions like the trigonometric functions.

The case where $f''(\alpha)$ blows up to infinity doesn't happen as often; for examples look at functions that have exponential-like growth.

16.3 Even more computational problems

We once again ask ourselves - what are the things we can *compute*? A little reflection should reveal to you that we can compute the following and just that:

- We have a closed form expression for f, and hence
- We know the values of $x_0, x_1, \ldots$.
- We know the values of $f(x_0), f(x_1), \ldots$.
- We know the values of $f'(x_0), f'(x_1), \ldots$.
- We know the values of $f''(x_0), f''(x_1), \ldots$.

Now, using the mean value theorem and the fact that $f(\alpha) = 0$, if x_n is sufficiently close to α, we have

$$f(x_n) = f(x_n) - f(\alpha) = f'(\xi_n)(x_n - \alpha)$$

for some $\xi_n \in (\alpha, x_n)$. The last equation can be written as

$$\alpha - x_n = -\frac{f(x_n)}{f'(\xi_n)}$$

If $-\frac{f(x_n)}{f'(\xi_n)} \approx -\frac{f(x_n)}{f'(x_n)}$, then from definition 16.1.1, we obtain

$$\alpha - x_n \approx x_{n+1} - x_n$$

One way to use this is to find such a ξ_n, in which case you can guess where α lies relative to x_n, use the above equation for error analysis, etc.

We end by noting that although the bisection method always works (provided you can bracket the roots), but Newton Raphson can fail. However when it does work, Newton Raphson converges to the root much faster than the bisection method.

16.4 The Secant method

We wish to find a root α of a function f. Start with ***two*** initial guesses of α namely x_0 and x_1. Ideally, the guesses should bracket α. Then, compute subsequent guesses using the following iteration

Definition 16.4.1.

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

Note the similarity between the Newton Raphson method and the Secant method! (The derivative in the Newton Raphson method is replaced with a discrete version of it)

Unfortunately, the Secant method converges more slowly than the Newton Raphson method, and thus finds little practical use.

It is possible for the guesses to alternate between a few values, and not converge.

Also, just because the initial guesses bracket the root, that doesn't mean the subsequent guesses will too.

Normally, when we compute x_{n+1}, we discard x_{n-1}. But you can instead discard x_n if you need to, with the goal of having guesses that bracket the root. Although even an uninterrupted bracket fails when the expressions of the form $f(x_n) - f(x_{n-1})$ are too large, and thus the brackets keep getting larger and larger instead of smaller.

For an example of what can go wrong with the secant method consider the case of a vertical parabola with vertex at 0, and initial guesses ± 1.

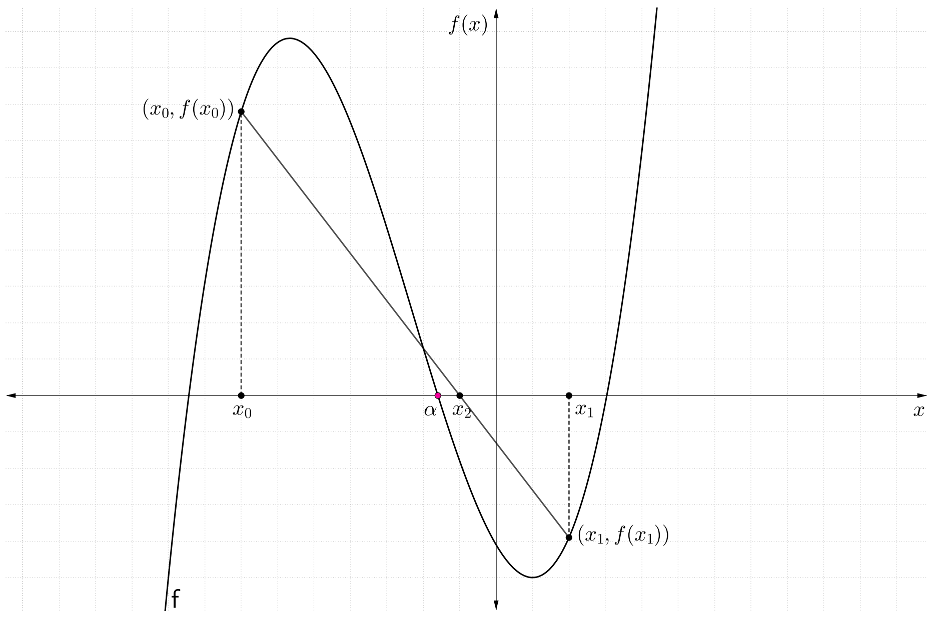

Figure 16.4.1: Secant Method

It is not clear to me whether the secant method will converge if our initial guesses bracket the root, we always choose our subsequent guesses so that they always bracket the root, and f has different signs at our guesses (including the initial guesses). Someone should probably investigate.

Lecture 17
February 28, Part B

17.1 Fixed Point Method

The **Newton Raphson method** and **secant method** are *one-point* and *two-point* iteration methods respectively (an algorithm is said to be n-point iteration method, if for the iteration to begin we need n initial guessed points). Now we will go over more general theory of *one-point* iteration methods. Conisder the simple equation

$$x^2 - 5 = 0 \tag{17.1.1}$$

then a root of the equation (17.1.1), is $\alpha = \sqrt{5}$, and consider the following iterative methods for solving the above equation:

$$x_{n+1} = 5 + x_n - x_n^2 \tag{17.1.2}$$

$$x_{n+1} = \frac{5}{x_n} \tag{17.1.3}$$

$$x_{n+1} = 1 + x_n - \frac{1}{5}x_n^2 \tag{17.1.4}$$

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{5}{x_n}\right) \tag{17.1.5}$$

Then its easy to observe that for all the above sequences whenever the sequence $\{x_n\}_{n\in\mathbb{N}}$ converges to some real number α, then $\alpha = \sqrt{5}$. This can be easily seen by assuming $\{x_n\}_{n\in\mathbb{N}}$ converges to α, and then taking limit as n tends to infinity. For example in equation (17.1.2) (assume that for some initial guess the sequence converges), then we have

$$\alpha = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty}\left(5 + x_n - x_n^2\right) = 5 + \alpha - \alpha^2 \Rightarrow \alpha = \sqrt{5}$$

Now observe that all the above iterative equations, have the general form

$$x_{n+1} = g(x_n) \tag{17.1.6}$$

for some appropriate continuous function on a suitable domain. For example in case of equation (17.1.2), the function $g(x) = 5 + x - x^2$. And assuming that x_n converges to α, we get that

$$\alpha = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} g(x_n) = g\left(\lim_{n\to\infty} x_n\right) = g(\alpha)$$

Thus α is a solution to the equation $g(x) = x$, and hence we have α is a root of g.

The above idea motivates us to see that the problem of finding a root for the equation $f(x) = 0$, can be converted into an equation $g(x) = x$, where we can take $g(x) = x - f(x)$. Thus if α is a root of f, i.e., $f(\alpha) = 0$, then we have

$$g(\alpha) = \alpha - f(\alpha) = \alpha$$

Now as we will see, if we are given some further informations about the functions we are working with, then **fixed point iteration methods** are faster than **bisection method** and are even guaranteed to converge to a root.

Remarks: *Fixed point iteration method has a lot of applications in Chaos Theory.*

Now let us look at some of the necessary conditions, we may need for the **fixed point method** to work!

The first question that arises naturally is, when does the equation $g(x) = x$ has a solution?

17.1.1 Necessary Condition for Existence of a Fixed Point

Theorem 17.1.1. Let $g : [a, b] \to \mathbb{R}$, be a continuous function, and suppose g satisfies the property

$$a \leq x \leq b \Rightarrow a \leq g(x) \leq b$$

then the equation $x = g(x)$, has at least one solution α in the interval $[a, b]$.

Proof. Define $f(x) = x - g(x)$, then we have f is a continuous function on $[a, b]$, and we further have

$$f(a) = a - g(a) \leq 0 \qquad \text{and} \qquad f(b) = b - g(b) \geq 0$$

and hence by **Intermediate Value Theorem**, we get that there exists a $\alpha \in [a, b]$, such that $f(\alpha) = 0$, but then we get $g(\alpha) = \alpha$. ∎

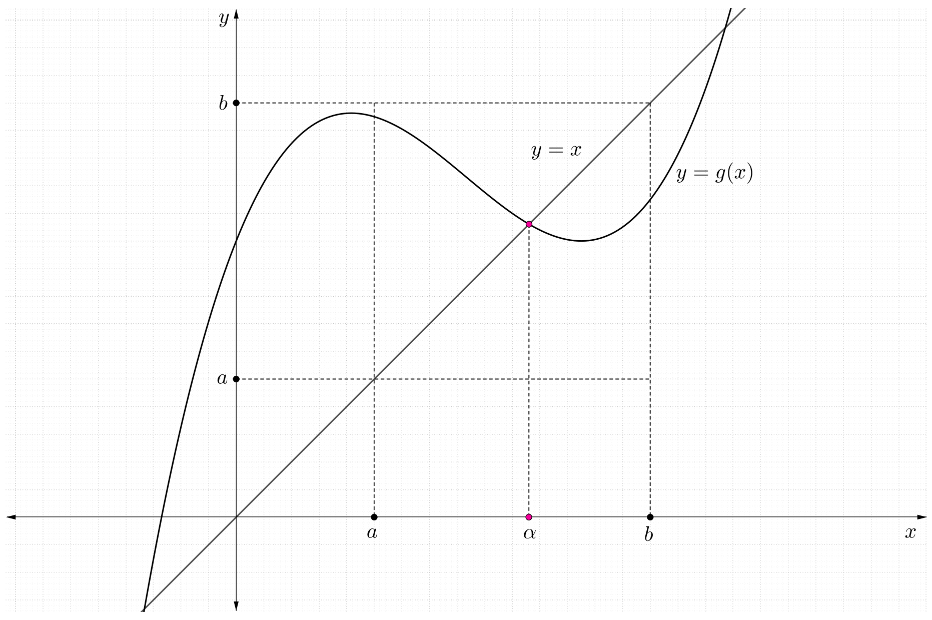

Figure 17.1.1: geometrical interpretation of **Theorem** 17.1.1

Theorem 17.1.1, can be geometrically interpretated as if we have a function $g : [a, b] \to [a, b]$, i.e., the graph of g is inside the square region $[a, b] \times [a, b]$, then the graph of g must intersect the diagonal of the square, i.e., $y = x$ line at some point, which precisely gives us that there exists a $\alpha \in [a, b]$, such that $g(\alpha) = \alpha$.

Definition 17.1.1. Now consider the following assumptions:

- g is differentiable on $[a, b]$, and further g' is continuous on $[a, b]$.
- $x \in [a, b] \Rightarrow g(x) \in [a, b]$.
- $\lambda := \max\limits_{a \leq x \leq b} |g'(x)| < 1$.

Then with the above assumptions we can guarantee, that there exists an unique root.

17.1.2 Uniqueness of the Fixed Point

Theorem 17.1.2. Assume that g satisfies all the above conditions given in **definition** 17.1.1, then g has an unique fixed point in $[a, b]$.

Proof. Now the fact that g is differentiable on $[a, b]$, tells us that g is continuous on $[a, b]$, and then second condition of our assumptions, along with **theorem** 17.1.1, gives us g has at least one fixed point in $[a, b]$.

Let $w_1, w_2 \in [a, b]$ then from **Mean Value Theorem**[1], we get that there exists a c in between w_1 and w_2 such that

$$g(w_1) - g(w_2) = g'(c)(w_1 - w_2)$$

But then we get that

$$|g(w_1) - g(w_2)| = |g'(c)||w_1 - w_2| \leq \lambda |w_1 - w_2| \tag{17.1.7}$$

Now suppose there exists $c_1, c_2 \in [a, b]$ such that $g(c_1) = c_1$ and $g(c_2) = c_2$, then we have

$$\begin{aligned} |c_1 - c_2| &= |g(c_1) - g(c_2)| \\ &\overset{(17.1.7)}{\leq} \lambda |c_1 - c_2| \end{aligned}$$

and hence we get that

$$(1 - \lambda)|c_1 - c_2| \leq 0 \tag{17.1.8}$$

But note that from the third condition in our assumptions, we have $1 - \lambda > 0$, and hence only way equation (17.1.8), can hold is

$$|c_1 - c_2| \leq 0 \Rightarrow |c_1 - c_2| = 0 \Rightarrow c_1 = c_2$$

Hence, g has an unique fixed point in $[a, b]$. ■

[1]**Mean Value Theorem:** Let $f : [a, b] \to \mathbb{R}$ be differentiable on (a, b), then there exists a $c \in (a, b)$ such that

$$f(b) - f(a) = f'(c)(b - a)$$

Lecture 18
March 4

18.1 Fixed point methods

Instead of trying to find a root to an equation of the form $f(x) = 0$, we can try to find a root to an equation of the form $x - f(x) = x$. Defining $g(x) := x - f(x)$, we see that the task of finding a root of f is equivalent to finding a **fixed point** of g (a fixed point of a function g is a real α such that $g(\alpha) = \alpha$).

We shall deal mainly with non-linear functions, as finding the roots/fixed points of linear functions can be done by employing the techniques of linear algebra, which we have already covered.

Therefore, we now set for ourselves the task of finding fixed points α of a function g. But first, we recall some theorems that we proved in previous lectures

Theorem 18.1.1.
If the following conditions obtain,

- $g \colon [a, b] \to [a, b]$.
- g is continuous on $[a, b]$.

Then g has a (not necessarily unique) fixed point in $[a, b]$. ■

Theorem 18.1.2.
If the following conditions obtain,

- $g \colon [a, b] \to [a, b]$.
- g is continuous on $[a, b]$.
- g' exists in $[a, b]$.
- g' is continuous on $[a, b]$.
- $\lambda := \max_{x \in [a,b]} |g'(x)| < 1$.

Then g has a **unique** fixed point in $[a, b]$. ■

Note that we have the same problem here as when we did the bisection method; the problem of finding a suitable interval ($[a, b]$ in this case). But let us assume that you have found such an interval, and proceed.

Under the assumptions of theorem 18.1.2, if we start with an initial guess $x_0 \in [a, b]$, and define subsequent guesses using the recursion

Definition 18.1.1.

$$x_{n+1} = g(x_n) \qquad \forall (n \geq 0)$$

then if we denote the unique fixed point of g in $[a, b]$ by α, we have

Theorem 18.1.3.

$$|\alpha - x_n| \le \lambda^n |\alpha - x_0|$$

Proof. Assuming the preconditions of theorem 18.1.2, $g\colon [a,b] \to [a,b]$. Combining that with the fact that $x_0 \in [a,b]$, it is easy to see by induction that $x_n \in [a,b]$ for all $n \ge 0$.

Now, by definition 18.1.1, and the fact that $g(\alpha) = \alpha$, we have

$$\alpha - x_{n+1} = g(\alpha) - g(x_n) = g'(c_n)(\alpha - x_n)$$

for some $c_n \in (\alpha, x_n)$, by the *mean value theorem*. Since by the preconditions of theorem 18.1.2 $\lambda \coloneqq \max_{x\in[a,b]} |g'(x)| < 1$, we obtain from the above equation

$$|\alpha - x_{n+1}| \le \lambda |\alpha - x_n|$$

By induction, we get

$$|\alpha - x_n| \le \lambda^n |\alpha - x_0|$$

■

Corollary 18.1.3.1. Under the assumptions of theorem 18.1.2,

$$\lim_{n\to\infty} x_n = \alpha$$

Proof. By theorem 18.1.3, $|\alpha - x_n| \le \lambda^n |\alpha - x_0|$. Since $\lambda < 1$, we get

$$\lim_{n\to\infty} |\alpha - x_n| = 0$$

using the squeeze theorem (the L.H.S. of the squeeze is $0 \le |\alpha - x_n|$, which follows from properties of the $|\cdot|$ function). ■

Corollary 18.1.3.2. Under the assumptions of theorem 18.1.2,

$$|\alpha - x_n| \le \frac{\lambda^n}{1-\lambda} |x_0 - x_1|$$

Proof. Note that

$$|\alpha - x_0| \overset{(1)}{\le} |\alpha - x_1| + |x_0 - x_1| \overset{(2)}{\le} \lambda|\alpha - x_0| + |x_0 - x_1|$$

$$\implies (1-\lambda)|\alpha - x_0| \le |x_0 - x_1|$$

$$\overset{(3)}{\implies} |\alpha - x_0| \le \frac{|x_0 - x_1|}{1-\lambda} \tag{4}$$

where inequality (1) is the triangle inequality, and inequality 2 follows from theorem 18.1.3. Implication (3) is justified as $\lambda < 1$ and therefore we are not dividing by 0. But then

$$|\alpha - x_n| \overset{(5)}{\le} \lambda^n |\alpha - x_0| \overset{(6)}{\le} \frac{\lambda^n}{1-\lambda} |x_0 - x_1|$$

where inequality (5) follows from theorem 18.1.3, and inequality (6) follows from inequality (4) above.

■

Now we come to what is probably the most important theorem in this lecture.

Theorem 18.1.4. Under the assumptions of theorem 18.1.2,

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = g'(\alpha)$$

Proof. Note that by the mean value theorem, for all $n \geq 0$ we have $\alpha - x_{n+1} = g(\alpha) - g(x_n) = g'(c_n)(\alpha - x_n)$ for some $c_n \in (\alpha, x_n)$. Therefore,

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = \lim_{n\to\infty} g'(c_n)$$

But since $x_n \to \alpha$, $c_n \to \alpha$ too, and then using the fact that g' is continuous in $[a, b]$, we get

$$\lim_{n\to\infty} g'(c_n) = g'(\alpha)$$

■

Definition 18.1.2. Suppose we have a sequence $\{y_n\}$ that converges to β. We say that $\{y_n\}$ converges to β **linearly**, if for all n

$$\beta - y_{n+1} \approx c(\beta - y_n)^p$$

with $p = 1$.
If $p > 1$, we say that the sequence $\{y_n\}$ converges to β **super-linearly**.

We are now ready to state the final theorem of this lecture.

Theorem 18.1.5. If we assume the preconditions of 18.1.2, and we additionally assume that $g'(\alpha) \neq 0$, x_n converges to α linearly.

Proof. Theorem 18.1.4 tells us that $\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = g'(\alpha)$. That implies, as long as $g'(\alpha) \neq 0$ and therefore higher order terms do not dominate, that for large n,

$$\alpha - x_{n+1} \approx g'(\alpha)(\alpha - x_n)$$

(The convergence is guaranteed both by theorem 18.1.1 and by the fact that $g'(\alpha) < 1$, which follows from the preconditions of 18.1.2). ■

We end by remarking that fixed point iterations are very easy to program owing to the simple nature of their iterations.

18.2 Introduction to Interpolation

18.2.1 What is Interpolation?

Let $f(x)$ be a function on an interval $[a, b]$. The goal is to construct a function $g(x)$ that approximates f to within a given error. To do so, we must specify:

- The structure of the approximating functions
- The way of measuring error

And, We have 2 problems now:

- **Problem I:** The function $f(x)$ is given (like $f(x) = e^x$), can be evaluated at any point x, and we seek a simple function $g(x)$ (like $g(x) = ax + b$) that best approximates f. The *simple* part constrains the possible accuracy.
- **Problem II:** Values are given at a set of points: $(x_0, f_0), \ldots, (x_n, f_n)$ with $f_j = f(x_j)$ but $f(x)$ is not known. The amount and type of data constrains the possible accuracy and what approximations may be constructed.

Our approach to approximation is known as **interpolation**. The process of finding the value of $f(x)$ corresponding to any value of $x = x_i$ between x_0 and x_n is called interpolation.

Definition 18.2.1 (Interpolation). The technique of estimating the value of a function for any intermediate value of the independent variable.

If the function f is known explicitly, then the value of $f(x)$ corresponding to any value of x can easily be found. Conversely, if the form of $f(x)$ is not known (as is the case in most of the applications), it is very difficult to determine the exact form of $f(x)$ with the help of tabulated set of values (x_i, f_i).

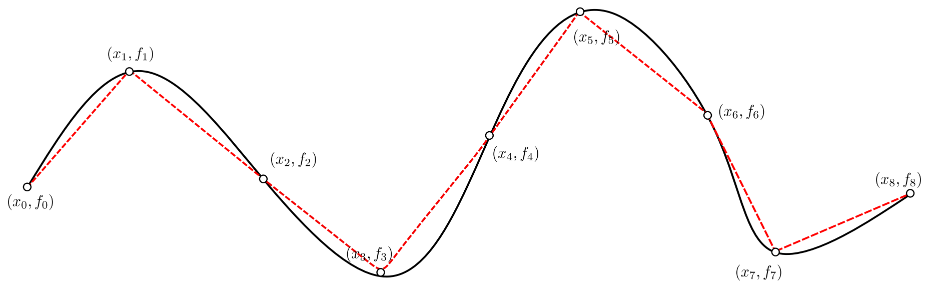

Figure 18.2.1

In such cases, $f(x)$ is replaced by a simpler function $\phi(x)$ which assumes the same values as those of $f(x)$ at the tabulated set of points. Any other value may be calculated from $\phi(x)$ which is known as the **interpolating function** or **smoothing function**.

Remark: *If $\phi(x)$ is a polynomial, then it called the* ***interpolating polynomial*** *and the process is called the* ***polynomial interpolation****. Similarly, when $\phi(x)$ is a finite trigonometric series, we have* ***trigonometric interpolation****. But we shall confine ourselves to polynomial interpolation only.*

Caution (interpolation vs. approximation): Note that *interpolation* is not exactly the same as *approximation* - it is a strategy that one hopes will approximate the function. There is no exact and unique solution. The actual function is NOT known and CANNOT be determined from the tabular data. In the case of Problem II where data is given, interpolation is natural since it uses precisely the data we are given.
For Problem I (where f is given), it is not obvious that interpolation is the right way to obtain a small max-norm (or any other error). Compare, for instance, to a best-fit line that does not have to pass through any points.

From the set $\{(x_0, f_0), \dots, (x_n, f_n)\}$ with $f_j = f(x_j)$, we would like to write,

$$f(x) \approx \phi(x) \text{ such that } \phi(x_j) = f_j \ \forall j \in \{0, 1, \dots, n\}$$

Example 18.2.1. Some examples of various kind of interpolation are following:

- Linear Interpolation (*General construction idea*)
  Here, we choose a set of $(n+1)$ functions $\{\phi_i\}_{i=0}^{n}$ and assume that ϕ is linearly dependent on ϕ_i's.

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

  (a) Polynomial Interpolation ($\phi_i(x) = x^i$)

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

  (b)

$$\phi(x) = \sum_{r=0}^{n} a_r e^{\iota r x} = \sum_{r=0}^{n} a_r \cos(rx) + \iota \sum_{r=0}^{n} a_r \sin(rx)$$

  (c) Cubic Spline interpolation
    * Here, we assume that $\phi(x)$ coincides with a cubic polynomial on every interval $[x_i, x_{i+1}], i = 0, 1, \dots, n$
    * Polynomials on different sub-intervals need not match.
- Non-Linear Interpolation
  (a) Rational function interpolation

$$\phi(x) = \frac{a_0 + a_1 x + \cdots + a_n x^n}{b_0 + b_1 x + \cdots + b_m x^m}$$

  Note that, total $n + m + 2$ parameters $a_0, a_1, \dots, a_n, b_0, b_1, \dots, b_n$ need to be determined here.

Remark: *Interpolation is a "guess" of function in a "valid domain" given to us.*

In the above case, the "valid domain" is $[\min_j x_j, \max_j x_j]$

On the other hand "guessing" a function outside the domain is known as ***Extrapolation****.*

Lecture 20
March 7, Part A

20.1 Interpolation

In numerical analysis, we are often provided with a table of values of a function $f(x)$ with respect to some argument x. Examples might be log tables, tables for various statistical distributions, non-analytic integrals etc. This approach, although extremely convenient for practical purposes, has its limitations. The most obvious one being that a table can only contain finitely many values of a function, from which we need to estimate the intermediate values. This is one of the places where interpolation becomes invaluable. It can also be used to develop polynomial approximations of transcendental or even non-analytic functions.
Interpolation basically deals with approximating a function given its value at some support points. It is used, as the name suggests, to approximate $f(x^*)$, where x^* lies between two of the given support points. This gives rise to an "approximate function" which can be used to get a reasonable guess of what $f(x^*)$ should be. The function formed must satisfy all the support points.
The subject of interpolation has also given rise to one of the most impactful algorithms of the 20^{th} century, the **Fast Fourier Transform**.

20.2 Lagrange's Interpolation Formula

Let Π_n be the set if all polynomials with degree $\leq n$. Thus, any $P(x) \in \Pi_n$ can be written in the form:

$$P(x) = \sum_{j=0}^{n} a_j x^j \qquad a_j \in \mathbb{R} \;\; \forall j = 1(1)n$$

Given (n+1) support points , the Lagrange interpolation gives a polynomial of degree n which satistfies all the points. So, if (x_i, f_i) for $i = 0(1)n$ be the support points (where f_i are the values of the function at x_i), we will have:

$$P(x_i) = f_i \quad \forall i = 0(1)n$$

Remark: *The support points must have distinct abscissae. Otherwise, one or more support points are either redundant or contradictory. So, we need:*

$$x_i \neq x_j \quad \forall\, i \neq j$$

Throughout this lecture, we will assume that this conditions is met.

We will show that given these conditions, $\exists! P \in \Pi_n$ such that the interpolation condition is staisfied, i.e, $P(x_i) = f_i \quad \forall i = 0(1)n$. We first show the existence by construction of **Lagrange Polynomials** and then uniqueness.

Definition 20.2.1. Given (n+1) support points (x_i, f_i), $i = 0(1)n$ with distinct abscissae, we define the i^{th} Lagrange Polynomials as follows:

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Then, taking $P(x) = \sum_{i=0}^{n} f_i L_i(x)$, it is easy to see that

$$L_i(x_k) = \delta_{ik} = \begin{cases} 1, & \text{if } i = k \\ 0, & \text{otherwise} \end{cases}$$

$$\therefore P(x_k) = \sum_{i=0}^{n} f_i L_i(x_k) = \sum_{i=0}^{n} f_i \delta_{ik} = f_k$$

i.e, it satisfies the interpolation condition. This proves the **existence** of P. ■

To prove the uniqueness, we assume two polynomials $P_1(x), P_2(x) \in \Pi_n$ which satisfy the interpolation conditions. Then the polynomial:

$$P_3(x) = P_1(x) - P_2(x) \in \Pi_n$$

has at least $n + 1$ roots $\{x_i : i = 0(1)n\}$, and is thus identically zero. Hence, $P_1(x) \equiv P_2(x)$, and the interpolating polynomial is **unique**. ■

This, in turn, shows that the Lagrange Polynomials are the only ones satisfying $L_i(x_k) = \delta_{ik}$, as it is a special case of the Lagrange Interpolation.

Remark: *As the polynomial is unique, we have different approaches for finding $f(x^*)$:*
1. *Finding the coefficients of $P(x)$ and then calculating $P(x^*)$.*
2. *Finding the value of $P(x^*)$ directly from the support points.*

The two interpolation methods discussed in this lecture, Lagrange Interpolation and Neville Interpolation, are best suited to approach (2).
Approach (1) is more convenient when the interpolation needs to be done multiple times. This is best done using Newton's Divided difference method, as will be discussed later.

Example 20.2.1. Say the following support points have been provided. Find the interpolated value at $x^* = 2$.

x_i	0	1	3
f_i	1	3	2

Solution. We simply compute the values of the Individual Lagrange Polynomials at x^* and then put them together.

$$L_0(x^* = 2) = \frac{(x^* - x_1)(x^* - x_2)}{(x_0 - x_1)(x_0 - x_2)} = -\frac{1}{3}$$

Similarly,

$$L_1(2) = 1 \text{ and } L_2(2) = \frac{1}{3}$$

Thus

$$P(x^*) = \sum_{i=0}^{2} f_i L_i(x^*) = -\frac{1}{3} + 3 + \frac{2}{3} = \frac{10}{3}$$

■

Lecture 21
March 7, Part B

21.1 Neville's Algorithm

An alternate approach to solving the interpolation problem would be to initially solve the problem for a smaller set of support points and then use these to recurrently arrive at the final solution.

Definition 21.1.1. Suppose we are given a set of support points (x_i, f_i) for $i = 0, 1, \ldots, n$. We define by $P_{i_0 i_1 \ldots i_k} \in \Pi_k$ the polynomial such that

$$P_{i_0 i_1 \ldots i_k}(x_{i_j}) = f_{i_j},\ \forall j = 0, 1, \ldots, k.$$

Theorem 21.1.1. The polynomials defined above obey the following recursion:

$$P_i(x) = f_i, \forall i \ (k = 0 \implies deg(P_i) = 0 \implies P_i\text{'s are constant}) \tag{21.1.1}$$

$$P_{i_0 i_1 \ldots i_k}(x) = \frac{(x - x_{i_0}) P_{i_1 i_2 \ldots i_k} - (x - x_{i_k}) P_{i_0 i_1 \ldots i_{k-1}}(x)}{x_{i_k} - x_{i_0}} \tag{21.1.2}$$

Based on the above theorem (21.1.1), we can construct a symmetric table to represent the flow of recursion as we calculate the final solution of the interpolation problem using solutions in intermediate steps.

By uniqueness of interpolating polynomials, we can say that the polynomials that we get at each intermediate step and the one that we get at the final step are all uniquely determined.

Table 21.1

$k = 0$	$k = 1$	$k = 2 \ldots$
$f_0 = P_0(x)$		
	$P_{01}(x)$	
$f_1 = P_1(x)$		$P_{012}(x) \ldots$
	$P_{12}(x)$	
$\vdots$	$\vdots$	$\vdots$
	$P_{n,n-1}(x)$	
$f_n = P_n(x)$		

Example 21.1.2. Let's verify the recursion for $k = 1$. We have

$$P_{01}(x) = \frac{(x - x_0)P_1(x) - (x - x_1)P_0(x)}{x_1 - x_0}.$$

Notice that here $i_0 = 0$ and $i_1 = 1$ with $k = 1$. So,

$$P_{01}(x_0) = \frac{-(x_0 - x_1)P_0(x_0)}{x_1 - x_0} = f_0 \; [\because P_0(x_0) = f_0].$$

and

$$P_{01}(x_1) = \frac{(x_1 - x_0)P_1(x_1)}{x_1 - x_0} = f_1 \; [\because P_1(x_1) = f_1].$$

So, we can see that the recursion works fine in this example. We can check further for $k = 2$ where we have

$$P_{012}(x) = \frac{(x - x_0)P_{12}(x) - (x - x_2)P_{01}(x)}{x_2 - x_0}.$$

Note that

$$\begin{aligned} P_{012}(x_1) &= \frac{(x_1 - x_0)P_{12}(x_1) - (x_1 - x_2)P_{01}(x_1)}{x_2 - x_0} \\ &= \frac{(x_2 - x_0)f_1}{x_2 - x_0} = f_1[\because P_{12}(x_1) = P_{01}(x_1) = f_1]. \end{aligned}$$

Lecture 21
March 7, Part B

21.2 Neville Interpolation

This is a more systematic way of implementing Lagrange Interpolation. Here, we make polynomials which satisfy a subset of the support points, and use those to construct the final polynomial in a step by step manner. As mentioned previously, we mainly use this method to find the interpolated value and not the coefficients of the polynomial.

Definition 21.2.1. For a given set of support points (x_i, f_i),$i = 0(1)n$ with distinct abscissae, we define Neville polynomials of order k as

$$P_{i_0 i_1 \ldots i_k} \in \Pi_k \text{ s.t. } P_{i_0 i_1 \ldots i_k}(x_i) = f_i \ \ \forall i \in \{i_0, i_1, \ldots, i_k\}$$

where i_α, $\alpha = 0(1)n$ as distinct elements of $\{0, 1, 2, \ldots, n\}$

We note that all these polynomials are unique, as shown in the previous section.
Thus, $P_i(x) = f_i \ \ \forall i = 0(1)n$. These constant polynomials form the zeroth level Neville Polynomials.
The key idea in Neville's Algorithm is a method to find $P_{i_0 i_1 \ldots i_k}(x)$ from $P_{i_0 i_1 \ldots i_{k-1}}(x)$ and $P_{i_1 i_2 \ldots i_k}(x)$.
Thus, knowing the zeroth level $P_0, P_1, \ldots, P_n$, we can find the first level $P_{0,1}, P_{1,2}, \ldots, P_{n-1,n}$ and so on. Once we reach the n^{th} level, we find the polynomial $P_{0,1\ldots,n}$ which satisfies:

$$P_{0,1\ldots,n} \in \Pi_n \text{ s.t. } P_{0,1\ldots,n}(x_i) = f_i \ \ \forall i = 0(1)n$$

This is precisely the interpolating polynomial $P(x)$ we are after!

Neville's Algorithm could be summarised in the following table.

$k = 0$	1	...	n-1	n
$P_0 = f_0$	$P_{0,1}$	...	$P_{0,1,\ldots,n-1}$	$P_{0,1,\ldots n}$
$P_1 = f_1$	$P_{1,2}$	...	$P_{1,2,\ldots,n}$	
$\vdots$	$\vdots$			
$P_{n-1} = f_{n-1}$	$P_{n-1,n}$			
$P_n = f_n$				

Proposition 21.2.1. The corresponding recurrence relation for Neville Interpolation is:

$$P_{i_0 i_1 \ldots i_k}(x) = \frac{(x - x_{i_0})P_{i_1 i_2 \ldots i_k}(x) - (x - x_{i_k})P_{i_0 i_1 \ldots i_{k-1}}(x)}{x_{i_k} - x_{i_0}}$$

Proof. Let

$$G(x) := \frac{(x - x_{i_0})P_{i_1 i_2 \ldots i_k}(x) - (x - x_{i_k})P_{i_0 i_1 \ldots i_{k-1}}(x)}{x_{i_k} - x_{i_0}}$$

As $P_{i_1 i_2 \ldots i_k}$, $P_{i_0 i_1 \ldots i_{k-1}}(x) \in \Pi_{k-1}$, we get $G(x) \in \Pi_k$. Now,

$$G(x_{i_0}) = \frac{-(x_{i_0} - x_{i_k})P_{i_0 i_1 \ldots i_{k-1}}(x_{i_0})}{x_{i_k} - x_{i_0}} = P_{i_0 i_1 \ldots i_{k-1}}(x_{i_0}) = f_{i_0}$$

Similarly, we have $G(x_{i_k}) = f_{i_k}$.
For $\alpha = 1(1)k-1$, we have

$$P_{i_1 i_2 \ldots i_k}(x_{i_\alpha}) = P_{i_0 i_1 \ldots i_{k-1}}(x_{i_\alpha}) = f_{i_\alpha}$$

$$\implies G(x_{i_\alpha}) = \frac{(x - x_{i_0})f_{i_\alpha} - (x - x_{i_k})f_{i_\alpha}}{x_{i_k} - x_{i_0}} = f_{i_\alpha} \quad \forall \alpha = 1(1)k-1$$

So, $G(x)$ satisfies all the conditions of $P_{i_0 i_1 \ldots i_k}(x)$.
Noting that $P_{i_0 i_1 \ldots i_k}(x)$ is unique (from previous section), this completes the proof. ■

Remark: *In theory, Neville Interpolation is just a reformulation of Lagrange Interpolation. However, it cuts down the computations required significantly and is easier to implement. For instance, the computation of the next level only requires knowledge of the previous level.*

Example 21.2.2. We try to solve the same problem as before using Neville Interpolation. Given data is

x_i	0	1	3
f_i	1	3	2

We need to find the interpolated value at $x^* = 2$.

Solution. To begin with, we have
$P_0(2) = 1$, $P_1(2) = 3$, $P_2(2) = 2$
From this, we have

$$\begin{aligned} P_{0,1}(x^* = 2) &= \frac{(x^* - x_0)P_1(x^*) - (x^* - x_1)P_0(x^*)}{x_1 - x_0} \\ &= \frac{(2-0)P_1(2) - (2-1)P_0(2)}{1-0} = 5 \end{aligned} \tag{21.2.1}$$

Similarly, $P_{1,2}(2) = \frac{5}{2}$ and thus,

$$P_{0,1,2}(x^* = 2) = \frac{(x^* - x_0)P_2(x^*) - (x^* - x_2)P_0(x^*)}{x_2 - x_0} = \frac{10}{3}$$

As expected, this gives us the same answer as before.

$k = 0$	1	2
1	5	$\frac{10}{3}$
3	$-\frac{5}{2}$	
2		

■

Lecture 22
March 11, Part A

22.1 Another View-point to the Neville's Algorithm

We will discuss slight variants of **Neville's algorithm**, we first define

$$T_{i+k,k} = P_{i,i+1....,i+k} \tag{22.1.1}$$

Theorem 22.1.1. The $T_{i,i+k}$ as defined in equation (22.1.1), satisfies the following recurrence relation:

$$\begin{aligned} T_{i,0} &:= P_i(x) = f_i, && \forall\, i = 0, 1, \ldots, n \\ T_{i,k} &:= \frac{(x - x_{i-k})T_{i,k-1} - (x - x_i)T_{i-1,k-1}}{x_i - x_{i-k}}, && 1 \le k \le i,\, i \ge 0 \end{aligned}$$

Precisely speaking the $T_{i,k}$ is just looking at the **Neville's algorithm** just from a different view-point, which as it turns out is more efficient way of evaluating the recurrence relation than the one given earlier in **Neville's algorithm**.

The process of evaluating $T_{i,k}$ is as follow:

- We evaluate $T_{i,k-1}$, i.e., we evaluate at the $(k-1)^{th}$ level, then
- Using the values we obtained at the $(k-1)^{th}$ level, we evaluate $T_{i,k}$, i.e., we evaluate at the k^{th} level using **Theorem** 22.2.1.

22.2 Newton's Interpolation Formula: Divided Differences

We already know a way how to find the $P \in \Pi_n$ satisfying $P(x_i) = f_i,\ \forall\, i \in \{0, 1, \ldots, n\}$. Now if we want to find the interpolating values for several arguments ξ_j's simulatenously, i.e., we are given a support set (say) (x_i, f_i), and we want to evaluate $P(\xi_j)$, then **Newton's method** is to be preferred.

We write the interpolation polynomial in the form:

$$P(x) = P_{01\ldots n}(x) = \sum_{j=0}^{n} a_i \left(\prod_{k=0}^{j-1} (x - x_k) \right) \tag{22.2.1}$$

where an empty product is assumed to be equal to 1.

Now, we use **Horner's scheme** to evaluate the polynomial at $x = \xi$, which basically a recursive way of evaluating the equation (22.2.1),

$$P(\xi) = a_0 + (\xi - x_0)\,(a_1 + (\xi - x_1)\,(a_2 + \cdots + (\xi - x_{n-2})\,(a_{n-1} + (\xi - x_{n-1})a_n)\cdots)) \tag{22.2.2}$$

so we just need to compute the coefficients a_i's. This can be done in many ways, but the simplest way would be to the solve the system:

$$\begin{aligned}
f_0 &= P(x_0) = a_0 \\
f_1 &= P(x_1) = a_0 + (x_1 - x_0)a_1 \\
&\vdots \\
f_n &= P(x_n) = a_0 + (x_n - x_0)a_1 + \cdots + (x_n - x_{n-1}) \cdots (x_n - x_0)a_n
\end{aligned}$$

one after the other. Though this will give us the a_i's, once we solve the whole system of linear equations, but at every step it involves one division and $2(k-1)$ many multiplications (where k indicates that we are at the k^{th} equation, $k \geq 1$)

$$k^{th} \text{ step: } a_k = \frac{f_k - \sum_{j=0}^{k-1} a_j \left(\prod_{i=0}^{j-1}(x_i - x_0)\right)}{\prod_{i=0}^{k}(x_i - x_0)}$$

so total it involves n may divisions and $n(n-1)$ many multiplications. But there's actually a more efficient way of doing this which involves only $\frac{n(n+1)}{2}$ many divisions.

22.2.1 Relation between Divided Difference Coefficients

Note that $P_{i_0 \ldots i_k} \in \Pi_k$ and $P_{i_0 \ldots i_{k-1}} \in \Pi_{k-1}$, so $P_{i_0 \ldots i_k} - P_{i_0 \ldots i_{k-1}} \in \Pi_k$, and not only so, since

$$P_{i_0 \ldots i_k}(x_{i_j}) = P_{i_0 \ldots i_{k-1}}(x_{i_j}) = f_{i_j}, \ \forall j = 0, 1, \ldots, k-1$$

hence, we get $x_{i_0}, \ldots, x_{i_{k-1}}$ are all the k roots of the polynomial $P_{i_0 \ldots i_k} - P_{i_0 \ldots i_{k-1}}$, and hence, there exists a constant say $f_{i_0 \ldots i_k} \in \mathbb{R}$, such that

$$P_{i_0 \ldots i_k}(x) = P_{i_0 \ldots i_{k-1}}(x) + f_{i_0 \ldots i_k} \prod_{j=0}^{k-1}(x - x_{i_j})$$

But then since, we know that $P_{i_0} \equiv f_{i_0}$, by induction we get that

$$P_{i_0 \ldots i_k}(x) = \sum_{j=0}^{k} f_{i_0 \ldots i_j} \left(\prod_{d=0}^{j-1}(x - x_{i_d}) \right) \tag{22.2.3}$$

Definition 22.2.1. The representation of the polynomial $P_{i_0 \ldots i_k}$ given in equation (22.2.3), is called the *Newton's representation* and the coefficients $f_{i_0 \ldots i_j}$ where $j = 0, \ldots, k$, are called the j^{th} *divided differences*.

Theorem 22.2.1. The *divided differences*, satisfies the recurrence relation given by:

$$f_{i_0 \ldots i_k} = \frac{f_{i_1 \ldots i_k} - f_{i_0 \ldots i_{k-1}}}{x_{i_k} - x_{i_0}}, \quad k \geq 1$$

Proof. From the definition of $P_{i_0 \ldots i_k}$, we already know that we can express it as:

$$P_{i_0 \ldots i_k}(x) = \frac{(x - x_{i_0})P_{i_1 \ldots i_k}(x) - (x - x_{i_k})P_{i_0 \ldots i_{k-1}}(x)}{x_{i_k} - x_{i_0}} \tag{22.2.4}$$

But then note that $f_{i_0 \ldots i_k}$ is the leading coefficient of $P_{i_0 \ldots, i_k} \in \Pi_k$ and $f_{i_1 \ldots i_k}$ and $f_{i_0 \ldots i_{k-1}}$ are the leading coefficients of the polynomials $P_{i_1 \ldots i_k} \in \Pi_{k-1}$ and $P_{i_0 \ldots i_{k-1}} \in \Pi_{k-1}$. And hence comparing the leading coefficients in either side of equation (22.2.4), we get the desired result. ■

Thus using the **Newton's interpolation formula** for *divided difference*, we can easily interpolate the polynomial $P_{i_0 \dots i_k}$, once we are given the support points (x_{i_j}, f_{i_j}).

Now let us see how does the given arguments helps us to quickly compute the interpolated polynomial. For this we consider the following example:

Example 22.2.2. Consider we want to interpolate $P_{0123} \in \Pi_3$, given the support points (x_i, f_i).

k	0	1	2	3
x_0	f_0	$\cdot$	$\cdot$	$\cdot$
x_1	f_1	f_{01}	$\cdot$	$\cdot$
x_2	f_2	f_{12}	f_{012}	$\cdot$
x_3	f_3	f_{23}	f_{123}	f_{0123}

Then note that since we know (x_i, f_i), for $i = 0, 1, 2, 3$, we can easily compute f_{01}, f_{12} and f_{23}, by using **Theorem** 22.2.1, which are in fact given by

$$f_{01} = \frac{f_1 - f_0}{x_1 - x_0}, \qquad f_{12} = \frac{f_2 - f_1}{x_2 - x_1}, \qquad f_{23} = \frac{f_3 - f_2}{x_3 - x_2}$$

Now from here we can compute f_{012} and f_{123}, which are given by

$$f_{012} = \frac{f_{12} - f_{01}}{x_2 - x_0} \qquad \text{and} \qquad f_{123} = \frac{f_{23} - f_{12}}{x_3 - x_1}$$

which finally gives us f_{0123},

$$f_{0123} = \frac{f_{123} - f_{012}}{x_3 - x_0}$$

thus, as we can see, at the k^{th} $(k \geq 1)$ level we required $n - k + 1$ many divisions (here $n = 3$), and hence, in the whole process we required $\frac{n(n+1)}{2}$ divisions, like in this case we can easily see it involved 6 divisions.

Lecture 23
March 11, Part B

23.1 Summary of Newton's divided differences

Definition 23.1.1. Given support points $(x_0, f_0), (x_1, f_1), \dots, (x_k, f_k)$, we can define the polynomial $\Phi(x) = P_{i_0 i_1 \dots i_n} \in \Pi_n$ to be the **unique** n degree polynomial that satisfies $\Phi(x_{i_j}) = f_{i_j}$ for all $0 \leq j \leq n$.

Remark: *Φ and $P_{i_0 i_1 \dots i_n}$ are merely different ways to write the same polynomial; the second form emphasizes the role of the support points in the definition of Φ.*

Now, Φ can be written using a formula called the **Newton's divided differences representation**, as the following theorem states

Theorem 23.1.1.

$$\Phi = \sum_{j=0}^{n} f_{i_0 \dots i_j} \prod_{k=0}^{j-1} (x - x_{i_k})$$

where the $f_{\dots}$ are defined as follows.
Observe that we already know what the f_{i_k}'s have to be; they are the y-coordinates of the corresponding support points. Then, given an integer k, we define

$$f_{i_0 \dots i_k} := \frac{f_{i_1 \dots i_k} - f_{i_0 \dots i_{k-1}}}{x_{i_k} - x_{i_o}}$$

The above recursive definition lets us calculate all the $f_{\dots}$ used in Newton's divided differences representation of $\Phi = P_{i_0 i_1 \dots i_n}$.

Example 23.1.2. We calculate some examples of the coefficients used in the formula in theorem 23.1.1

$$f_{01} = \frac{f_1 - f_0}{x_1 - x_0} \qquad f_{12} = \frac{f_2 - f_1}{x_2 - x_1} \qquad f_{123} = \frac{f_{23} - f_{12}}{x_3 - x_1} \qquad f_{4567} = \frac{f_{567} - f_{456}}{x_7 - x_4}$$

23.2 Error Analysis

Suppose you have a function f that you can easily calculate at some, but not all values. Or perhaps the function is only defined at some, but not all values. Instead of working with the function, you might decide to use a polynomial that coincides with it at certain values of x, where the value of f is known.

More specifically, suppose you have the set of points $x_0, x_1, \dots x_n$, and you're given a function f whose values are $f_0.f_1, \dots f_n$ at $x_0, x_1, \dots x_n$ respectively.

Then, using the results of the above section we can construct a polynomial P such that

$$P(x_i) = f_i \quad \forall (0 \leq i \leq n)$$

However unless we are exceptionally lucky, given any arbitrary $x \neq x_i \quad \forall (0 \leq i \leq n)$, we shall **not** have $P(x) = f(x)$. Our task then, is to estimate the error

$$f(x) - P(x)$$

for general $x \in \mathbb{R}$.

We can supply a formula for the error $f(x) - P(x)$ if f is differentiable sufficiently many times. But before that we need to state a definition:

Definition 23.2.1. Given reals $x_0, x_1, \dots, x_k$, we define $I_0[x_0, x_1, \dots, x_k]$ to be the smallest interval containing them

Theorem 23.2.1. Suppose that f is differentiable atleast $n+1$ times in $\operatorname{dom}(f)$. Then given a real $\overline{x} \in \operatorname{dom}(f)$, the error $\epsilon(\overline{x})$ is given by

$$\epsilon(\overline{x}) = f(\overline{x}) - P(\overline{x}) = \left(\frac{f^{(n+1)}(\xi)}{(n+1)!} \right) \prod_{k=0}^{n} (\overline{x} - x_k)$$

for some $\xi \in I_0[x_0, x_1, \dots, x_n, \overline{x}]$. Note that $\prod_{k=0}^{n} (\overline{x} - x_k)$ can also be abbreviated as $\omega(\overline{x})$. Using that abbreviation, the above identity can be written as

$$\epsilon(\overline{x}) = f(\overline{x}) - P(\overline{x}) = \omega(\overline{x}) \left(\frac{f^{(n+1)}(\xi)}{(n+1)!} \right)$$

Proof. If $\overline{x} = x_i$ for any $0 \leq i \leq n$, the result holds trivially. So we shall assume that $\overline{x} \neq x_i$ for all $0 \leq i \leq n$.

Define $F\colon \operatorname{dom}(f) \to \mathbb{R}$ by $F(x) := \epsilon(x) - \frac{\epsilon(\overline{x})}{\omega(\overline{x})}\omega(x)$. Note that

- The definition is valid because $\overline{x} \neq x_i$ for all $0 \leq i \leq n$ implies that $\omega(\overline{x}) \neq 0$.
- F is 0 at $x_0, x_1, \dots, x_n$ and $\overline{x}$. Therefore F has at least $n+2$ zeroes.
- F is differentiable atleast $n+1$ times, since $F = f - P - \frac{\epsilon(\overline{x})}{\omega(\overline{x})}\omega$, and f is differentiable at least $n+1$ times, while the remaining terms are polynomials and thus differentiable arbitrarily many times.
- The last bullet point also tells us that $F, F^{(1)}, F^{(n)}$ are all continuous (differentiability implies continuity). This bullet point, and the last are necessary to justify our appeals to Rolle's theorem below.

Now, assume W.L.O.G. that $x_0 < x_1 < \cdots < x_n$. Let b be the index such that

$$x_b < \overline{x} < x_{b+1}$$

(If $\overline{x}$ is not contained in (x_0, x_n), our analysis fails). Since F has at least $n+2$ distinct zeroes in $I_0[x_0, x_1, \dots, x_n, \overline{x}]$, by Rolle's theorem, $F^{(1)}$ has at least $n+1$ zeroes in $I_0[x_0, x_1, \dots, x_n, \overline{x}]$. Continuing this way, we see that $F^{(n+1)}$ has atleast one zero in $I_0[x_0, x_1, \ldots, x_n, \overline{x}]$. Denote the zero ξ. Then, we have

$$\begin{aligned}
&0 = F^{(n+1)}(\xi) \\
\implies &0 = \frac{\mathrm{d}^{n+1}}{\mathrm{d}x^{n+1}} \left(\epsilon(x) - \frac{\epsilon(\overline{x})}{\omega(\overline{x})}\omega(x) \right)\Bigg|_{x=\xi} \\
\implies &0 = \frac{\mathrm{d}^{n+1}}{\mathrm{d}x^{n+1}} \left(f(x) - P(x) - \frac{\epsilon(\overline{x})}{\omega(\overline{x})}\omega(x) \right)\Bigg|_{x=\xi} \\
\implies &0 = f^{(n+1)}(\xi) - \frac{\epsilon(\overline{x})}{\omega(\overline{x})}(n+1)! \qquad (1)
\end{aligned}$$

since (P has degree at most n) $\implies \frac{\mathrm{d}^{n+1}P}{\mathrm{d}x^{n+1}} = 0$ and $\frac{\mathrm{d}^{n+1}\omega}{\mathrm{d}x^{n+1}} = (n+1)!$. Rewriting equation (1), we have

$$\epsilon(\overline{x}) = \omega(\overline{x}) \left(\frac{f^{(n+1)}(\xi)}{(n+1)!} \right)$$

■

Note that if we want to use theorem 23.2.1, then we need to have some information about $\frac{f^{(n+1)}(\xi)}{(n+1)!}$. For although we know everything there is to know about ω, everything else is a bit of a mystery. So if we want to be able to say anything meaningful about the error at $\overline{x}$, we had better know something about $\frac{f^{(n+1)}(\xi)}{(n+1)!}$.

One example of a situation where we can say something meaningful about $\frac{f^{(n+1)}(\xi)}{(n+1)!}$ is when $f = \sin$. In that case, we have $\left|f^{(n+1)}(\xi)\right| \leq 1$, which helps us bound $\epsilon(\overline{x})$,

Lecture 24
March 14, Part A

24.1 Divided-Difference Scheme

We know that the interpolating polynomial $P_{i_0 i_1 \ldots i_k}$ is uniquely determined by its support points. So, it is invariant of any permutation of the indices $i_0, i_1, \ldots, i_k$. Therefore we have the following result:

Theorem 24.1.1. The coefficients/divided-differences $f_{i_0 i_1 \ldots i_k}$ are invariant of any permutations of the indices $i_0, i_1, \ldots, i_k$.

Example 24.1.2. We take up an old example and form the *divided-differences scheme* for the same:

k	0	1	2
$x_0 = 0$	$f_0 = 1$		
		$f_{01} = 2$	
$x_1 = 1$	$f_1 = 3$		$f_{012} = \frac{-5}{6}$
		$f_{12} = \frac{-1}{2}$	
$x_2 = 3$	$f_2 = 2$		

Here $f_{01} = \frac{f_1 - f_0}{x_1 - x_0} = 2$, $f_{12} = \frac{f_2 - f_1}{x_2 - x_1}$ and $f_{012} = \frac{f_{12} - f_{01}}{x_2 - x_0} = \frac{-5}{6}$.
So, we get our final interpolated polynomial as $P_{012}(x) = 1 + 2.(x-0) - \frac{5}{6}(x-0)(x-1)$.

Remark: *We will always want to choose such a permutation of the indices that satisfies*

$$|\xi - x_{i_k}| \geq |\xi - x_{i_{k-1}}|, \forall k = 1, 2, \ldots, n.$$

which dampens the error during the evaluation of the Horner's Scheme.

In accordance of the remark (24.1), we have an algorithm to choose the preferred sequence of indices. For this, we assume that the support abscissae x_i are in, say ascending, order. Then for each $k > 0$, we can choose the index i_k so that either $i_k = min\{i_l | 0 \leq l < k\} - 1$ or $i_k - max\{i_l | 0 \leq l < k\} + 1$. So, instead of the descending-diagonal sequence of indices that we get from the divided-difference scheme, here we get a zig-zag path.

Example 24.1.3. Continuing from Example 24.1.2, we can use the sequence $i_0 = 1, i_1 = 2, i_3 = 0$ for interpolating the polynomial at $\xi = 2$. Then the corresponding Newton Representation is:

$$P_{120}(x) = 3 - \frac{1}{2}(x-1) - \frac{5}{6}(x-1)(x-3).$$

24.2 An alternate notation

Sometimes, the support ordinates f_i are the function values of a given function $f(x)$, which we want to approximate using interpolation. We might want to do this because sometimes, even if we know the function, it might not have a nice closed form, making it difficult to evaluate the function at arbitrary points. We can think of the divided differences as multivariate functions of the support abscissae x_i. In general, we use the following notation for this purpose:

$$f[x_{i_0}, x_{i_1}, \dots, x_{i_k}] = f_{i_0,i_1,\dots,i_k}.$$

For instance, we have the following:

$$\begin{aligned} f[x_0] &= f(x_0) \\ f[x_0, x_1] &= \frac{f[x_1]f[x_0]}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\ &= \frac{f(x_0)(x_1 - x_2) + f(x_1)(x_2 - x_0) + f(x_2)(x_0 - x_1)}{(x_1 - x_0)(x_2 - x_1)(x_0 - x_2)}. \end{aligned}$$

By Theorem 24.1.1, we can see that the divided differences $f[x_{i_0}, x_{i_1}, \dots, x_{i_k}]$ are invariant of permutations of the indices.

Theorem 24.2.1. If the given function $f(x)$ is a polynomial of degree N, then $f[x_0, x_1, \dots, x_k] = 0$ for all $k > N$.

This is easy to see. Since, the interpolated polynomial is unique, the final polynomial must be identically equal to $f(x) \in \Pi_N$. Therefore, the coefficent of x^k in $P_{01\dots k}(x)$, which is given by $f[x_0, x_1, \dots, x_k]$, must be equal to 0 for all $k > N$.

24.3 Error Term

We have by the Newton's Interpolation formula that

$$P(x) \equiv P_{01\dots n}(x) = f[x_0] + f[x_0, x_1](x - x_0) + \dots + f[x_0, \dots, x_n](x - x_0) \dots (x - x_{n-1}).$$

Suppose we introduce a $(n+2)^{nd}$ support point (x_{n+1}, f_{n+1}) with $x_{n+1} := \overline{x}$ and $f_{n+1} := f(\overline{x})$ and $\overline{x} \neq x_i, \forall i = 0, 1, \dots, n$. Then, by Newton's Formula

$$f(\overline{x}) = P_{01\dots(n+1)}(\overline{x}) = P_{01\dots n}(x) + f[x_0, \dots, x_n, \overline{x}]\omega(\overline{x}).$$

We can conclude from the error analysis from the previous chapter, since $\omega(\overline{x}) \neq 0$, that

$$f[x_0 \dots x_n, \overline{x}] = \frac{f^{(n+1)}(\xi)}{(n+1)!} \text{ for some } \xi \in I[x_0, \dots, x_n, \overline{x}].$$

Lecture 25
March 14, Part B

25.1 Hermite Interpolation

Consider $m+1$ distinct abcissae $x_0, \dots, x_m$, now suppose we are not just given the values f_i, but we are given the first $n_i - 1$ derivatives of the function f, at the points x_i's, i.e., we are given n_i ordinate values $f_i^{(k)}$, where $k \in \{0, 1, \dots, n_i - 1\}$, for each $i \in \{0, 1, \dots, m\}$. Let

$$n + 1 = \sum_{i=0}^{m} n_i$$

We want to find a $P \in \Pi_n$, where n is defined as above just that the Interpolation conditions:

$$P^{(k)}(x_i) = f_i^{(k)}, \ \forall\, k = 0, 1, \dots, n_i - 1 \text{ and } \forall\, i = 0, 1, \dots, n \tag{25.1.1}$$

are satisfied.

Remark: *The Neville's Interpolation or Newton's Interpolation are just special case of Hermite's interpolation with $n_i = 1$, for each $i \in \{1, \dots, m\}$.*

25.1.1 The Hermite Interpolation has an Unique Solution

Theorem 25.1.1. WLOG, assume we are given that

$$x_0 < x_1 < \cdots < x_m$$

are $m+1$ abcissae, and we are further given real numbers f_i^k, where $k \in \{0, 1, \dots, n_i - 1\}$ for each $i \in \{0, 1, \dots, m\}$. Let

$$n + 1 = \sum_{i=0}^{m} n_i$$

then there exists an unique $P \in \Pi_n$ satisfying equation (25.1.1).

Proof.

Uniqueness: Suppose there exists $P_1, P_2 \in \Pi_n$ such that P_1, P_2 satisfy the equation (25.1.1). Now consider the difference polynomial

$$Q(x) = P_1(x) - P_2(x) \in \Pi_n$$

but then we clearly have

$$Q^{(k)}(x_i) = 0, \ \forall\, k \in \{0, 1, \dots, n_i - 1\} \text{ and } \forall\, i \in \{0, 1, \dots, m\}$$

thus from here we can conclude that x_i is root of Q, with multiplicity n_i, for each $i \in \{0, 1, \ldots, m\}$. But then we get that number of roots of Q is

$$\sum_{i=0}^{m} n_i = n + 1$$

but we have $Q \in \Pi_n$, hence, we must have $Q \equiv 0$, since its degree is strictly less than $n + 1$.

Existence: Suppose we assume

$$P(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$$

then note that the equation (25.1.1) can be interpreted as a system of linear equations, (say)

$$AX = B \tag{25.1.2}$$

where A is some $(n + 1) \times (n + 1)$ real valued matrix, and

$$B = \begin{pmatrix} f_0 & f_0^{(1)} & \cdots & f_0^{(n_1-1)} & \cdots & f_m & f_m^{(1)} & \cdots & f_m^{(n_m-1)} \end{pmatrix}^T$$

and $X = \begin{pmatrix} c_0 & c_1 & \cdots c_n \end{pmatrix}^T$. Now we have already shown that if there exists a solution to equation (25.1.2), then it must be unique. But then if we consider $B = \mathbf{0}$, then its obvious that $X = \mathbf{0}$ is a solution to $AX = B$, and hence its in fact the unique solution to $AX = \mathbf{0}$. This tells us that the kernel of the linear map $T_A : x \mapsto Ax$, is the zero subspace, $\ker(T_A) = \{\mathbf{0}\}$.

Hence, A is non-singular. Thus we indeed have for all $B \in \mathbb{R}^{n+1}$, there exists an unique $X \in \mathbb{R}^{n+1}$ such that $AX = B$, which completes the proof.

■

25.1.2 Generalized Lagrange Polynomials and Explicit Construction of the Hermite Interpolation Polynomial

For each $i \in \{0, 1, \ldots, m\}$ and $k \in \{0, 1, \ldots, n_i - 1\}$ construct $L_{ik} \in \Pi_n$ such that

$$L_{ik}^{(\sigma)}(x_j) = \delta_{ij}\delta_{k\sigma} \tag{25.1.3}$$

Definition 25.1.1. The polynomials $L_{ik} \in \Pi_n$ defined as in equation (25.1.3) are called the *generalized Lagrange polynomials*.

Now note that we can write $P \in \Pi_n$ satisfying equation (25.1.1) as:

$$P(x) = \sum_{i=0}^{m} \sum_{k=0}^{n_i-1} f_i^{(k)} L_{ik}(x) \tag{25.1.4}$$

This can be easily verified by computing the derivatives, we get

$$P^{(\sigma)}(x) = \sum_{i=0}^{m} \sum_{k=0}^{n_i-1} f_i^{(k)} L_{ik}^{(\sigma)}(x)$$

and hence, we get that for $\sigma \in \{0, 1, \ldots, n_d - 1\}$,

$$\begin{aligned} P^{(\sigma)}(x_d) &= \sum_{i=0}^{m} \sum_{k=0}^{n_i-1} f_i^{(k)} L_{ik}^{(\sigma)}(x_d) \\ &= \sum_{i=0}^{m} \sum_{k=0}^{n_i-1} f_i^{(k)} \delta_{id}\delta_{k\sigma} \\ &= f_d^{(\sigma)} \end{aligned}$$

thus from **Theorem** 25.1.1, we get that equation (25.1.4) is indeed true. Now lets see how we can in fact construct the polynomials L_{ik} recursively. We first construct a set of auxiliary polynomials, defined by:

$$l_{ik} := \frac{(x-x_i)^k}{k!} \prod_{\substack{j=0 \\ j\neq i}}^{m} \left(\frac{x-x_j}{x_i-x_j} \right)^{n_j}, \ \forall\, 0 \leq i \leq m,\ \forall\, 0 \leq k \leq n_i - 1 \tag{25.1.5}$$

Theorem 25.1.2. L_{ik} satisfies the following recurrence relation:

$$L_{i,n_i-1}(x) = l_{i,n_i-1}(x) \qquad \forall\, i = 0,1,\dots,m$$
$$L_{ik}(x) = l_{ik}(x) - \sum_{\nu=k+1}^{n_i-1} l_{ik}^{(\nu)}(x_i) L_{i\nu}(x) \qquad \forall\, k \leq n_i - 2$$

Proof. We first show that

$$L_{i,n_i-1}(x) := l_{i,n_i-1}(x),\ \forall\, i = 0,1,\dots,m$$

Note that from the definition of l_{ik}, we have x_j is a root of l_{i,n_i-1}, with multiplicity n_j, if $j \neq i$, and x_i is root of l_{i,n_i-1} with a multiplicity of $n_i - 1$. Thus for any $\sigma \in \{0,1,\dots,n_d-1\}$, with $d \neq i$, we have there exists at least one factor of $(x-x_d)$ in $l_{i,n_i-1}^{(\sigma)}(x)$ and hence $l_{i,n_i-1}^{(\sigma)}(x_d) = 0$. And if $d = i$, then observe that $l_{i,n_i-1}^{(\sigma)}(x_i)$ is non-zero if and only if $\sigma = n_i - 1$, and in which case we have $l_{i,n_i-1}^{(n_i-1)}(x_i) = 1$. Hence we have shown that

$$l_{i,n_i-1}^{\sigma}(x_j) = \delta_{ij}\delta_{n_i-1,\sigma}$$

But then since $l_{i,n_i-1} \in \Pi_n$, from **Theorem** 25.1.1, we get that $L_{i,n_i-1}(x) = l_{i,n_i-1}(x)$, for each $i \in \{0,\dots,m\}$.

Now to prove that other part of the recurrence relation we will use induction. Suppose $L_{i\nu}$ has the desired properties from $\nu = n_i - 1, n_i - 2, \dots, k^*$. Now fix $j \in \{0,1,\dots,m\}$ and consider the term

$$J_\sigma = \underbrace{l_{i,k^*-1}^{(\sigma)}(x_j)}_{T_1} - \underbrace{\sum_{\nu=k^*}^{n_i-1} l_{i,k^*-1}^{(\nu)}(x_i) L_{i\nu}^{(\sigma)}(x_j)}_{T_2}$$

where $\sigma \in \{0,1,\dots,n_j-1\}$, then if $j \neq i$, then both the terms T_1 and T_2 are zero, and so $J_\sigma = 0$, in this case. Now if $j = i$, then note that for $\sigma \leq k^* - 1$, T_2 does not contribute to the term J_σ, and $T_1 = 1$ if and only if $\sigma = k^* - 1$, and is 0 otherwise. So, for $\sigma \leq k^* - 1$,

$$J_\sigma = \begin{cases} 1 & \text{if } \sigma = k^* - 1 \\ 0 & \text{otherwise} \end{cases}$$

On the other hand for $\sigma > k^* - 1$, T_2 will contribute to the term J_σ, when $\nu = \sigma$, and hence in that case we get

$$J_\sigma = l_{i,k^*-1}^{(\sigma)}(x_i) - l_{i,k^*-1}^{(\sigma)}(x_i) = 0$$

thus we in fact have $J_\sigma = \delta_{ij}\delta_{k^*-1,\sigma}$, thus using **Theorem** 25.1.1 and induction, we get that

$$L_{ik}(x) = l_{ik}(x) - \sum_{\nu=k+1}^{n_i-1} l_{ik}^{(\nu)}(x_i) L_{i\nu}(x)\ \forall\, k \leq n_i - 2$$

which completes the proof. ■

Lecture 26
March 18

26.1 Algorithms for Hermite interpolation

Unfortunately Neville's algorithm and Newton's divided difference algorithm cannot be used to carry out Hermite interpolation, as they assume the x-coordinates of the support points are all different. Our task here then, is to develop algorithms that *can* be used to carry out Hermite interpolation. We begin by introducing a technical tool that'll help us with that task.

26.1.1 Virtual Abscissae

Suppose you are given m distinct reals $x_0 < x_1 < \cdots < x_m$, and for all $0 \leq i \leq m$, at real x_i you are given the values $f^{(0)}(x_i), f^{(1)}(x_i), \ldots, f^{(n_i-1)}(x_i)$, for some $n_i \in \mathbb{Z}$. Those reals and values are the support points we shall work with. Define

Definition 26.1.1.

$$n = \left(\sum_{i=0}^{m} n_i\right) - 1$$

Now list the support points as follows:

1) First, list $(x_0, f^{(0)}(x_0)), (x_0, f^{(1)}(x_0)), \ldots, (x_0, f^{(n_0-1)}(x_0))$
2) Then, append the list $(x_1, f^{(0)}(x_1)), (x_1, f^{(1)}(x_1)), \ldots, (x_1, f^{(n_1-1)}(x_1))$ in that order to the pre-existing list.
3) Then, append the list $(x_2, f^{(0)}(x_2)), (x_2, f^{(1)}(x_2)), \ldots, (x_2, f^{(n_2-1)}(x_2))$ in that order to the pre-existing list.
4) Continue this way until you...

.

.

.

m) Finally, append the list $(x_m, f^{(0)}(x_m)), (x_m, f^{(1)}(x_m)), \ldots, (x_m, f^{(n_m-1)}(x_m))$ in that order to the pre-existing list.

Since there are $n+1$ support points in total, there must be $n+1$ items in our final list. **Number the items in the list from 0 to n.** Now we introduce two more definitions.

Definition 26.1.2. Let t_i be the be the x-coordinate of the i-th element in the list of support points. Note that

$$t_0 \leq t_1 \leq t_2 \leq \cdots \leq t_n$$

Definition 26.1.3. Let s_j be the number of times t_j appears in the sequence $t_0, t_1, \ldots, t_j$.

At this point there is nothing this exposition would benefit more from than an example to illustrate the definitions we have made so far. And so we supply one. And another one too!

Example 26.1.1. Suppose you are given the list of support points

$$(-49, 63), (1, 2), (1, 3), (5, -\pi), (22, 7)$$

Then, the sequence $\{t_i\}$ is

$$-49, 1, 1, 5, 22$$

and the sequence $\{s_i\}$ is

$$1, 1, 2, 1, 1$$

Example 26.1.2. Suppose you are given list of support points

$$(1, 27), (2, 53), (2, 27), (3, 22), (4, 56), (4, 22), (4, 49), (273, 32)$$

Then, the sequence $\{t_i\}$ is

$$1, 2, 2, 3, 4, 4, 4, 273$$

and the sequence $\{s_i\}$ is

$$1, 1, 2, 1, 1, 2, 3, 1$$

26.1.2 The problem of Hermite interpolation - rephrased.

Using the language of virtual abscissae defined in the previous section, we see that the problem of finding a polynomial that interpolates the support points provided is precisely the problem of finding a polynomial $P_{01\ldots n}$ such that

$$P_{01\ldots n}^{(s_j-1)}(t_j) = f^{(s_j-1)}(t_j) \qquad \forall (0 \leq j \leq n) \tag{26.1.1}$$

We have already proved (in a previous lecture) that there exists such a polynomial $P_{01\ldots n}$ in Π_n that is **unique** in Π_n. Write

$$P_{01\ldots n}(t) = \sum_{j=0}^{n} b_j \frac{t^j}{j!}$$

Now, define

Definition 26.1.4.

$$\Pi(t) = \left[1, t, \tfrac{t^2}{2!}, \ldots, \tfrac{t^n}{n!}\right]$$

and

$$b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Note that $\Pi(t)b = P_{01\ldots n}(t)$. Therefore, condition 26.1.1 can be replaced by the condition

$$\Pi^{(s_j-1)}(t_j)b = f^{(s_j-1)}(t_j) \qquad \forall (0 \le j \le n)$$

Denote the $(n+1)\times(n+1)$ matrix that is formed by letting the j-th row be $\Pi^{(s_j-1)}(t_j)$ by V_n, then

$$V_n = \begin{bmatrix} \Pi^{(s_0-1)}(t_0) \\ \Pi^{(s_1-1)}(t_1) \\ \vdots \\ \Pi^{(s_n-1)}(t_n) \end{bmatrix}$$

Then the condition on $\Pi(t)b = P_{01\ldots n}(t)$ can be described using the language of linear algebra. More specifically, the condition on $P_{01\ldots n}(t)$ can be written

$$\begin{bmatrix} \Pi^{(s_0-1)}(t_0) \\ \Pi^{(s_1-1)}(t_1) \\ \vdots \\ \Pi^{(s_n-1)}(t_n) \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} f^{(s_0-1)}(t_0) \\ f^{(s_1-1)}(t_1) \\ f^{(s_2-1)}(t_2) \\ \vdots \\ f^{(s_n-1)}(t_n) \end{bmatrix} \tag{26.1.2}$$

Now, we prove an important theorem.

Theorem 26.1.3. The system of equations 26.1.2 is always solvable.

Proof. We proved (in an earlier lecture) that b is determined uniquely by the above condition, so if we view V_n as a linear transformation from $\mathbb{R}^{n+1}$ to $\mathbb{R}^{n+1}$, V_n is injective. Then, by the **Rank Nullity** theorem, V_n is also surjective, and thus V_n is invertible, and thus the system of equations 26.1.2 is always solvable. ■

Since V_n is invertible, as we saw in the proof immediately above, we have reduced the problem of finding $P_{01\ldots n}(t)$ to a problem in linear algebra, and since there are many kinds of algorithms to invert many kinds of matrices, we have practically obliterated the problem of Hermite interpolation.

As an aside, matrices of the form V_n are called **Generalized Vandermonde Matrices**. The **Vandermonde Matrix** (without the adjective "general") is the kind of V_n you get when $s_j = 1$ for all $0 \le j \le n$. It can be factorized as follows

$$\begin{bmatrix} 1 & t_0 & t_0^2 & \ldots & t_0^n \\ 1 & t_1 & t_1^2 & \ldots & t_1^n \\ 1 & t_2 & t_2^2 & \ldots & t_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & t_n^2 & \ldots & t_n^n \end{bmatrix} \begin{bmatrix} \frac{1}{0!} & 0 & 0 & \ldots & 0 \\ 0 & \frac{1}{1!} & 0 & \ldots & 0 \\ 0 & 0 & \frac{1}{2!} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \frac{1}{n!} \end{bmatrix}$$

(I shamelessly stole part of the latex for the above expression from Wikipedia). We conclude this section with an example.

Example 26.1.4. Suppose you're told that the sequence $\{t_i\}$ is of the form

$$t_0 = t_1 = t_2 < t_3 = t_4$$

Then, the Vandermonde Matrix V_4 of this sequence is

$$\begin{bmatrix} 1 & t_0 & \frac{t_0^2}{2!} & \frac{t_0^3}{3!} & \frac{t_0^4}{4!} \\ 0 & 1 & t_1 & \frac{t_1^2}{2!} & \frac{t_1^3}{3!} \\ 0 & 0 & 1 & t_2 & \frac{t_2^2}{2!} \\ 1 & t_3 & \frac{t_3^2}{2!} & \frac{t_3^3}{3!} & \frac{t_3^4}{4!} \\ 0 & 1 & t_4 & \frac{t_4^2}{2!} & \frac{t_4^3}{3!} \end{bmatrix}$$

26.1.3 Generalizations of the techniques used in previous sections

Note that the definition of the sequence $\{s_j\}$ depends only on the definition of $\{t_j\}$, and **as long as $\{t_j\}$ is non-decreasing**, we can define and solve the systems of equations in the previous section like we did there.

Lecture 27
March 21, Part A

Now since, we have introduced the notion of **Generalized Vandermonde Matrices**, let us see how we can prepare to use **Neville's algorithm**, or **Newton's divided difference formula**, for tackling **Hermite interpolation** problems.

27.1 Neville Type Algorithm for Hermite Interpolation

We start off by associating each sequence

$$t_i \le t_{i+1} \le \cdots \le t_{i+k}, \quad 0 \le i \le i+k \le n$$

of virtual abscissae the solution $P_{i,\dots,i+k} \in \Pi_k$ of the partial **Hermite interpolation** problem, i.e, it satisfies the equations

$$P^{(s_j-1)}_{i,i+1,\dots,i+k}(t_j) = f^{(s_j-1)}(t_j), \quad i \le j \le i+k \tag{27.1.1}$$

where s_j is the number of times t_j, occurs in the subsequence $t_i, t_{i+1}, \dots, t_j$.

Example 27.1.1. Suppose we are given that

$$x_0 = 0,\ f_0^{(0)} = -1,\ \text{and}\ f_0^{(1)} = -2$$
$$x_1 = 1,\ f_1^{(0)} = 0,\ f_1^{(1)} = 10,\ \text{and}\ f_1^{(2)} = 40$$

Then our virtual abscissae $\{t_j\}$'s are simply:

$$t_0 = t_1 := x_0 = 0 \text{ and}$$
$$t_2 = t_3 = t_4 := x_1 = 1$$

Now suppose we consider the subsequence t_1, t_2, t_3, i.e., $i = 1$ and $k = 2$, then we get

$$s_1 = 1,\ s_2 = 1,\ \text{and}\ s_3 = 2$$

and thus the corresponding partial Hermite interpolation polynomial will be $P_{123} \in \Pi_2$, satisfying

$$P_{123}^{(s_1-1)}(t_1) = P_{123}(0) = f_0^{(0)} = -1$$
$$P_{123}^{(s_2-1)}(t_2) = P_{123}(1) = f_1^{(0)} = 0$$
$$P_{123}^{(s_3-1)}(t_3) = P'_{123}(1) = f_1^{(1)} = 10$$

Using the above notations we have the following analogs to the **Neville's formulae**:

Theorem 27.1.2. The polynomials $P_{i,i+1,\dots,i+k} \in \Pi_k$ follows the following recursive relation:

- If $t_i = t_{i+1} = \cdots = t_{i+k} = x_l$ for some l, then

$$P_{i,i+1,\dots,i+k}(x) = \sum_{r=0}^{k} \frac{f_l^{(r)}}{r!}(x - x_l)^r \tag{27.1.2}$$

- And if $t_i < t_{i+k}$, then we have

$$P_{i,i+1.\dots,i+k}(x) = \frac{(x - t_i)P_{i+1,\dots,i+k}(x) - (x - t_{i+k})P_{i,\dots,i+k-1}(x)}{t_{i+k} - t_i} \tag{27.1.3}$$

27.2 Generalized Divided Differences

Analogous to the defintion of coefficients of divided differences, we now define the coefficients of generalized divided differences.

Definition 27.2.1. The coefficients of generalized divided differences, denoted by

$$f[t_i, t_{i+1}, \dots, t_{i+k}]$$

as the coefficient of x^k in the polynomial $P_{i,i+1,\dots,i+k} \in \Pi_k$ as defined above according to equation (27.1.1).

Now that we have discussed about the generalized **Neville's formulae**, we can easily derive the formulae for **Newton's generalized divided differences** as follows:

Theorem 27.2.1. The coefficients of generalized divided differences satisfies the following recurrence relations:

- If $t_i = t_{i+1} = \cdots = t_{i+k} = x_l$ for some l, then

$$f[t_i, t_{i+1}, \dots, t_{i+k}] = \frac{1}{k!} f_l^{(k)} \tag{27.2.1}$$

- And $t_i < t_{i+k}$, we have

$$f[t_i, t_{i+1}, \dots, t_{i+k}] = \frac{f[i+1, \dots, i+k] - f[i, \dots, i+k-1]}{t_{i+k} - t_i} \tag{27.2.2}$$

Proof. The proof follows directly from comparing the coefficients of x^k in the two cases in the equation (27.1.2) and equation (27.1.3) given in **Theorem** 27.1.2. ■

Lecture 28
March 21, Part B

28.1 More interpolation algorithms for virtual abscissae

We present a version of Newton's divided differences that works for virtual abscissae.

28.1.1 Newton's divided differences for virtual abscissae

Note that the previous version of Newton's divided differences we presented relies on the abscissae of the support points being pairwise different. When working with virtual abscissae, we have no such guarantee. Instead, we have to rely on alternate definitions, which we present immediately below.

Definition 28.1.1. Suppose you are given virtual abscissae $t_0 \leq t_1 \leq \cdots \leq t_n$, and the corresponding values of a function f and the derivatives of f at the virtual abscissae. Then, we define the base cases

$$f[t_i] = f(t_i)$$

and we define

$$f[t_i \ldots t_{i+k}] = \begin{cases} \dfrac{f[t_{i+1} \ldots t_{i+k}] - f[t_i \ldots t_{i+k-1}]}{t_{i+k} - t_i} & \text{if } t_{i+k} - t_i \neq 0 \\ \dfrac{f^{(k)}(x_i)}{k!} & \text{if } t_{i+k} - t_i = 0 \end{cases}$$

Now, we present an example

Example 28.1.1. Suppose you are given the support points $(0,-1),(0,-2),(1,0),(1,10),(1,40)$. Then, the virtual abscissae are $t_0 = t_1 < t_2 = t_3 = t_4$ with $t_0 = 0$ and $t_1 = 1$, and you know that

$$f^{(0)}(0) = -1, \quad f^{(1)}(0) = -2, \quad f^{(0)}(1) = 0, \quad f^{(1)}(1) = 10, \quad f^{(2)}(1) = 40$$

Then, the tableau for constructing the interpolating polynomial using Newton's divided differences is

t_i \ k	0	1	2	3	4
t_0	-1				
		-2			
t_1	-1		3		
		1		6	
t_2	0		9		5
		10		11	
t_3	0		20		
		10			
t_4	0				

Next, we present a theorem that will let you calculate the interpolation polynomial for the support points you are given.

Theorem 28.1.2. Just like in the case of the original Newton's divided differences method, the interpolating polynomial $P_{01\ldots n}$ is given by

$$P_{01\ldots n}(t) = \sum_{j=0}^{n} f[t_0 \ldots t_j] \prod_{k=0}^{j-1} (t - t_k)$$

Proof. The statement is obviously true for $n = 0$. Then, note that

$$P_{01\ldots n} - P_{01\ldots(n-1)} = f[t_0, \ldots, t_n]x^n + \text{ a degree } n-1 \text{ polynomial}$$

Also, we know that $P_{01\ldots n} - P_{01\ldots(n-1)}$ has the roots $t_0, t_1, \ldots, t_{n-1}$. Therefore,

$$P_{01\ldots n} - P_{01\ldots(n-1)} = f[t_0, \ldots, t_n] \prod_{k=0}^{j-1} (t - t_k)$$

By recursion, we are done. ■

Remark: *This proof badly needs more explaining, but until the person doing the lecture before this one pushes their notes, I can't do anything about that.*

28.2 Spline interpolation

28.2.1 Introduction

Given an interval $[a,b]$, consider an arbitrary partition

$$\Delta\colon a = x_0 < x_1 \cdots < x_n$$

on $[a,b]$. The x_i are called **knots**. Then we define spline functions as below.

Definition 28.2.1. A spline function S_Δ on $[a, b]$ with the partition Δ is a function from $[a, b]$ to $\mathbb{R}$ such that

$$S_\Delta|_{(x_{i-1},x_i)} \text{ is a polynomial } \forall(1 \leq i \leq n)$$

Definition 28.2.2. If in the above definition the restrictions $S_\Delta|_{(x_{i-1},x_i)}$ all belong to Π_3, then S_Δ is said to be a **cubic spline**.

However, the above definition of cubic splines is hard to work with, so we shall impose more stringent conditions on our cubic splines.

Definition 28.2.3. A spline function on S_Δ on $[a, b]$ with the partition Δ is a **cubic spline** if

- $S_\Delta|_{[x_{i-1},x_i]} \in \Pi_3$ for all $1 \leq i \leq n$. Note that we are now dealing with closed intervals instead of open.
- S_Δ is continuously differentiable at least twice in $[a, b]$.

We shall assume these facts of the cubic splines we work with; now, and in the future.

28.2.2 The uniqueness of splines passing through $n + 1$ support points

Now suppose you're given $n+1$ ordinates $Y = \{y_0, y_1, \ldots, y_n\}$. We can impose $n+1$ conditions on the spline function S_Δ, namely

$$S_\Delta(x_i) = y_i \qquad \forall(0 \leq i \leq n)$$

The set of conditions is denoted $S_\Delta(Y; \cdot)$, and the $n + 1$ conditions can be written

$$S_\Delta(Y; x_i) = y_i \qquad \forall(0 \leq i \leq n) \tag{28.2.1}$$

But S_Δ is not a single polynomial function; it is piecewise polynomial. So we can't use the uniqueness/existence theorems developed in previous lectures. We have to develop specialized theorems to deal with spline functions.

In order for our cubic splines satisfying conditions 28.2.1 to be unique, we need to impose additional conditions on them, but there are multiple choices of uniqueness conditions we can use. We list three of them:

1) $S''_\Delta(Y; a) = S''_\Delta(Y; b) = 0$.
2) $S^{(k)}_\Delta(Y; a) = S^{(k)}_\Delta(Y; b)$ for $k = 0, 1,$ and 2.
3) $S'_\Delta(Y; a) = y'_0$, and $S'_\Delta(Y; b) = y'_n$, where y'_0 and y'_n are fixed values given to you.

Any one of the three conditions above is sufficient to guarantee the uniqueness of the spline passing through the support points (x_i, y_i) for all $0 \leq i \leq n$.

Lecture 29
March 25, Part A

29.1 Theoretical Foundations for Cubic Spline Interpolation

Let $\Delta = \{a = x_0 < x_1 < \cdot < x_n = b\}$ be a partition of the interval $[a, b]$.

Definition 29.1.1. A cubic spline function S_Δ on Δ is real function $S_\Delta : [a, b] \to \mathbb{R}$, with the properties:

1. $S_\Delta \in \mathscr{C}^2[a, b]$, i.e., S_Δ is twice continuously differentiable on $[a, b]$.
2. $S_{\Delta|[x_i,x_{i+1}]} \in \Pi_3$, i.e., coincides on every subinterval $[x_i, x_{i+1}]$ with a polynomial of degree at most three, for all $i = 0, 1, \ldots, n-1$.

We further define an *interpolating spline function* as follows:

Definition 29.1.2. For a finite sequence $Y := (y_0, y_1, \ldots, y_n)$ of $n+1$ real numbers, we denote by

$$S_\Delta(Y; \cdot)$$

as an *interpolating spline function* S_Δ with $S_\Delta(Y; x_i) = y_i$ for all $i \in \{0, 1, \ldots, n\}$.

and we have already discussed in **Lecture 28**, that such a interpolating spline function is unique if any one of the three following condition is satisfied:

(a) $S''_\Delta(Y; a) = S''_\Delta(Y; b) = 0$.

(b) $S^{(k)}_\Delta(Y; a) = S^{(k)}_\Delta(Y; b)$ for $k = 0, 1,$ and 2.

(c) $S'_\Delta(Y; a) = y'_0$, and $S'_\Delta(Y; b) = y'_n$, where y'_0 and y'_n are any fixed real numbers.

Let $m \in \mathbb{N}$ and we shall now consider the following sets

$$\mathscr{K}^m[a, b] := \{f : [a, b] \to \mathbb{R} \mid f^{(m-1)} \text{ is absolutely continuous on } [a, b] \text{ and } f^{(m)} \in \mathscr{L}^2[a, b]\} \tag{29.1.1}$$

and we define

$$\mathscr{K}^m_p[a, b] := \{f \in \mathscr{K}^m[a, b] \mid f^{(k)}(a) = f^{(k)}(b),\ \forall\, k = 0, 1, \ldots, m-1\} \tag{29.1.2}$$

So particularly for $m = 2$, we have $f \in \mathscr{K}^2_p[a, b]$, if

- $f^{(0)}$ and $f^{(1)}$ are absolutely continuous and $f^{(2)}$ exists.
- $f^{(2)} \in \mathscr{L}^2[a, b]$, i.e., $\int_a^b |f^{(2)}(x)|^2 \,\mathrm{d}x$ exists and is finite.
- And finally $f^{(0)}(a) = f^{(0)}(b)$ and $f^{(1)}(a) = f^{(1)}(b)$.

note that the first two conditions are enough for $f \in \mathscr{K}^2[a, b]$.

29.1.1 Holladay's Identity

Then note that $S_\Delta \in \mathscr{K}^3[a,b]$, and further if $S_\Delta^{(k)}(a) = S_\Delta^{(k)}(b)$ for $k = 0, 1, 2$ then $S_\Delta \in \mathscr{K}_p^3[a,b]$. Also note that for all $f \in \mathscr{K}^2[a,b]$, we can define the following **semi-norm**

$$\|f\|^2 := \int_a^b |f(x)|^2 \, \mathrm{d}x \tag{29.1.3}$$

Now with all that definitions in mind, we are ready to define the following identity due to **Holladay**,

Theorem 29.1.1. (Holladay's Identity) Let $f \in \mathscr{K}^2[a,b]$ and let

$$\Delta = \{a = x_0 < x_1 < \cdots < x_n = b\}$$

be a partition of the interval $[a,b]$, and if S_Δ is a spline function with knots at $x_i \in \Delta$, then

$$\|f - S_\Delta\|^2 = \|f\|^2 - \|S_\Delta\|^2 - 2\left[(f'(x) - S_\Delta'(x))S_\Delta^{(2)}(x)\Big|_a^b - \sum_{i=1}^n (f(x) - S_\Delta(x))S_\Delta^{(3)}(x)\Big|_{x_{i-1}^+}^{x_i^-}\right]$$

Proof. We have

$$\begin{aligned}
\|f - S_\Delta\|^2 &= \int_a^b \left|f^{(2)}(x) - S_\Delta^{(2)}(x)\right|^2 \mathrm{d}x \\
&= \int_a^b \left|f^{(2)}(x)\right|^2 \mathrm{d}x - 2\int_a^b f^{(2)}(x)S_\Delta^{(2)}(x)\,\mathrm{d}x + \int_a^b \left|S_\Delta^{(2)}(x)\right|^2 \mathrm{d}x \\
&= \int_a^b \left|f^{(2)}(x)\right|^2 \mathrm{d}x - \int_a^b \left|S_\Delta^{(2)}(x)\right|^2 \mathrm{d}x - 2\int_a^b \left(f^{(2)}(x) - S_\Delta^{(2)}(x)\right) S_\Delta^{(2)}(x)\,\mathrm{d}x \\
&= \|f\|^2 - \|S_\Delta\|^2 - 2\int_a^b \left(f^{(2)}(x) - S_\Delta^{(2)}(x)\right) S_\Delta^{(2)}(x)\,\mathrm{d}x
\end{aligned}$$

Now observe that using **intergration by parts** we can write

$$\int_{x_{i-1}}^{x_i} \left(f^{(2)}(x) - S_\Delta^{(2)}(x)\right) S_\Delta^{(2)}(x)\,\mathrm{d}x = \left(f^{(1)}(x) - S_\Delta^{(1)}(x)\right) S_\Delta^{(2)}(x)\Big|_{x_{i-1}}^{x_i} - \int_{x_{i-1}}^{x_i} \left(f^{(1)}(x) - S_\Delta^{(1)}(x)\right) S_\Delta^{(3)}(x)\,\mathrm{d}x$$

which can be further broken down as

$$\begin{aligned}
\int_{x_{i-1}}^{x_i} \left(f^{(1)}(x) - S_\Delta^{(1)}(x)\right) S_\Delta^{(3)}(x)\,\mathrm{d}x &= \left(f(x) - S_\Delta(x)\right) S_\Delta^{(3)}(x)\Big|_{x_{i-1}^+}^{x_i^-} - \int_{x_{i-1}}^{x_i} \left(f(x) - S_\Delta(x)\right) S_\Delta^{(4)}(x)\,\mathrm{d}x \\
&\overset{(1)}{=} \left(f(x) - S_\Delta(x)\right) S_\Delta^{(3)}(x)\Big|_{x_{i-1}^+}^{x_i^-}
\end{aligned}$$

where (1), follows from the fact that $S_{\Delta|(x_{i-1},x_i)} \in \Pi_3$, and hence $S_\Delta^{(4)} \equiv 0$ on (x_{i-1}, x_i), and combining the two results we got from **intergration by parts** we get that

$$\begin{aligned}
\int_a^b \left(f^{(2)}(x) - S_\Delta^{(2)}(x)\right) S_\Delta^{(2)}(x)\,\mathrm{d}x &= \sum_{i=1}^n \int_{x_{i-1}}^{x_i} \left(f^{(2)}(x) - S_\Delta^{(2)}(x)\right) S_\Delta^{(2)}(x)\,\mathrm{d}x \\
&= \sum_{i=1}^n \left(f(x) - S_\Delta(x)\right) S_\Delta^{(3)}(x)\Big|_{x_{i-1}}^{x_i} - \sum_{i=1}^n \left(f(x) - S_\Delta(x)\right) S_\Delta^{(3)}(x)\Big|_{x_{i-1}^+}^{x_i^-} \\
&= (f(x) - S_\Delta(x))S_\Delta^{(3)}(x)\Big|_a^b - \sum_{i=1}^n \left(f(x) - S_\Delta(x)\right) S_\Delta^{(3)}(x)\Big|_{x_{i-1}^+}^{x_i^-}
\end{aligned}$$

which completes the proof of **Theorem** 29.1.1. ■

29.1.2 Minimum Semi-norm Property of Spline Functions

Theorem 29.1.2. Let $\Delta = \{a = x_0 < x_1 < \cdots < x_n = b\}$ be a partition of the interval $[a,b]$ and let $f \in \mathscr{K}^2[a,b]$. And suppose we are given the values $Y = (y_0, y_1, \ldots, y_n)$ such that $f(x_i) = y_i,\ \forall\, i \in \{0,1,\ldots,n\}$. Then

$$0 \le \|f - S_\Delta(Y;\cdot)\|^2 = \|f\|^2 - \|S_\Delta(Y;\cdot)\|^2$$

holds for every spline function $S_\Delta(Y;\cdot)$ provided one of the following conditions is true:

(a) $S_\Delta^{(2)}(a) = S_\Delta^{(2)}(b) = 0$.
(b) $f \in \mathscr{K}_p^2[a,b]$ and $S_\Delta(Y;\cdot) \in \mathscr{K}_p^2[a,b]$.
(c) $S'_\Delta(Y;a) = f'(a)$ and $S'_\Delta(Y;b) = f'(b)$.

Proof. First note that from the definition of spline interpolation function $S_\Delta(Y;\cdot)$, we have

$$\lim_{x\to x_k^-} f(x) = f(x_k) = y_k = \lim_{x\to x_k^-} S_\Delta(Y;x)$$

and

$$\lim_{x\to x_k^+} f(x) = f(x_k) = y_k = \lim_{x\to x_k^+} S_\Delta(Y;x)$$

for any $k \in \{0,1,\ldots,n\}$. Thus the term

$$\sum_{i=1}^{n} \left(f(x) - S_\Delta(Y;x)\right) S_\Delta^{(3)}(Y;x)\Big|_{x_{i-1}^+}^{x_i^-} = 0$$

and hence using **Holladay's identity** (**Theorem** 29.1.1), we get

$$\|f - S_\Delta(Y;\cdot)\|^2 = \|f\|^2 - \|S_\Delta(Y;\cdot)\|^2 - 2\left(f'(x) - S'_\Delta(Y;x)\right) S_\Delta^{(2)}(Y;x)\Big|_a^b \tag{29.1.4}$$

But then its easy to observe that for any of the three conditions (a), (b) or (c), the term

$$\left(f'(x) - S'_\Delta(Y;x)\right) S_\Delta^{(2)}(Y;x)\Big|_a^b = 0$$

Thus we have

$$\|f - S_\Delta(Y;\cdot)\|^2 = \|f\|^2 - \|S_\Delta(Y;\cdot)\|^2 \tag{29.1.5}$$

but it trivially follows from the definition of **semi-norm**, that $\|f - S_\Delta(Y;)\| \ge 0$, which completes our proof. ■

But why is the spline Interpolation, so important? We end this chapter with the following geometric interpretation of the spline interpolation.

29.1.3 Geometric Interpretation of Cubic Spline Functions

Definition 29.1.3. Let $f \in \mathscr{K}^2[a,b]$, then the curvature of f at the point $x \in [a,b]$, denoted by κ_x is defined by

$$\kappa_x = \frac{f^{(2)}(x)}{(1 + f'(x)^2)^{\frac{3}{2}}}$$

Now suppose we have $|f'(x)| \ll 1$, i.e., $|f'(x)|$ is very small compared to 1, then in that case we have

$$\kappa_x \approx f^{(2)}(x)$$

But observe that the term

$$\|f\|^2 = \int_a^b |f^{(2)}(x)|^2 \, \mathrm{d}x$$

is kind of measure for the total curvature of the function f in the interval $[a, b]$, but from **Theorem** 29.1.2, we have already shown that if the spline function satisfies certain conditions, then it minimizes the term $\|f\|^2$. So we can say that a spline function is the *"smoothest"* function to interpolate the given support points (x_i, y_i).

Lecture 30
March 25, Part B

30.1 Uniqueness of Spline Function

We now prove that under the hypothesis of **Theorem** 29.1.2, in each of the three cases there exists an unique spline function.

We restate the **Theorem** 29.1.2 as follows:

Theorem 30.1.1. Let $\Delta = \{a = x_0 < x_1 < \cdots < x_n = b\}$ be a partition of the interval $[a, b]$ and let $f \in \mathscr{K}^2[a, b]$. And suppose we are given the values $Y = (y_0, y_1, \ldots, y_n)$ such that $f(x_i) = y_i,\ \forall\, i \in \{0, 1, \ldots, n\}$. Then

$$0 \leq \|f - S_\Delta(Y; \cdot)\|^2 = \|f\|^2 - \|S_\Delta(Y; \cdot)\|^2$$

holds for every spline function $S_\Delta(Y; \cdot)$ provided one of the following conditions is true:

(a) $S_\Delta^{(2)}(a) = S_\Delta^{(2)}(b) = 0.$
(b) $f \in \mathscr{K}_p^2[a, b]$ and $S_\Delta(Y; \cdot) \in \mathscr{K}_p^2[a, b]$.
(c) $S'_\Delta(Y; a) = f'(a)$ and $S'_\Delta(Y; b) = f'(b)$.

Furthermore in each of the cases the spline function $S_\Delta(Y; \cdot)$ is uniquely determined.

Proof. In the proof of **Theorem** 29.1.2, we already shown that the spline function minimizes the semi-norm if any of the conditions (a), (b) or (c) is met. So now we only need to check the uniqueness.

Suppose there exists two spline functions $S_\Delta(Y; \cdot)$ and $\tilde{S}_\Delta(Y; \cdot)$ satisfying the hypothesis. But then since $S_\Delta(Y; \cdot), \tilde{S}_\Delta(Y; \cdot) \in \mathscr{K}^2[a, b]$, so we can use $\tilde{S}_\Delta(Y; \cdot)$ to play the role of $f \in \mathscr{K}^2[a, b]$. But then from the semi-norm minimizing property of spline function (**Theorem** 29.1.2), we get that

$$\|\tilde{S}_\Delta(Y; \cdot) - S_\Delta(Y; \cdot)\|^2 = \|\tilde{S}_\Delta(Y; \cdot)\|^2 - \|S_\Delta(Y; \cdot)\|^2 \geq 0 \tag{30.1.1}$$

But then we can switch the roles of $\tilde{S}_\Delta(Y; \cdot)$ and $S_\Delta(Y; \cdot)$, i.e., in this time we can take $S_\Delta(Y; \cdot)$ to play the role of the $f \in \mathscr{K}^2[a, b]$, then we get

$$\begin{aligned} \|\tilde{S}_\Delta(Y; \cdot) - S_\Delta(Y; \cdot)\|^2 &= \|S_\Delta(Y; \cdot) - \tilde{S}_\Delta(Y; \cdot)\|^2 \\ &= \|S_\Delta(Y; \cdot)\|^2 - \|\tilde{S}_\Delta(Y; \cdot)\|^2 \\ &\overset{(30.1.1)}{\leq} 0 \end{aligned}$$

Thus we get

$$\|\tilde{S}_\Delta(Y; \cdot) - S_\Delta(Y; \cdot)\|^2 = 0$$

which gives us

$$\int_a^b \left|\tilde{S}_\Delta^{(2)}(Y; \cdot) - S_\Delta^{(2)}(Y; \cdot)\right|^2 \mathrm{d}x = 0 \tag{30.1.2}$$

but then since $\tilde{S}_\Delta(Y;\cdot) \in \mathscr{C}^2[a,b]$ and $S_\Delta(Y;\cdot) \in \mathscr{C}^2[a,b]$, we get that equation (30.1.2) holds if and only if

$$\tilde{S}_\Delta^{(2)}(Y;\cdot) \equiv S_\Delta^{(2)}(Y;\cdot) \tag{30.1.3}$$

but then twice intergrating both sides of equation (30.1.3), we get that

$$\tilde{S}_\Delta(Y;\cdot) \equiv S_\Delta(Y;\cdot) + cx + d$$

but then we can easily compute c and d using the fact that $\tilde{S}_\Delta(Y;a) = S_\Delta(Y;a)$ and $\tilde{S}_\Delta(Y;b) = S_\Delta(Y;b)$. This gives us $c = d = 0$, and hence we have

$$\tilde{S}_\Delta(Y;\cdot) \equiv S_\Delta(Y;\cdot)$$

which completes the proof of the uniqueness. ■

We end this chapter, with some remarks on the three conditions (a), (b) and (c) in **Theorem** 30.1.1.

Remark:

*(i) The spline function arising from condition (a), is regarded as **natural spline function**, as it does not have any other dependency on the function f, which we are trying to interpolate.*

(ii) On the other hand (b), is restricted to only functions $f \in \mathscr{K}_p^2[a,b]$.

(iii) And finally, (c) kind of restricts the range of the function f, since we are fixing the values $f'(a) = y_0'$ and $f'(b) = y_n'$, where $y_0' = S_\Delta'(Y;x_0)$ and $y_n' = S_\Delta'(Y;x_n)$.

Lecture 31
March 28

31.1 Constructing an interpolating cubic spline

We want to construct an interpolating cubic spline from $[a, b]$ to $\mathbb{R}$ that takes the value y_i at x_i for $n+1$ given support points $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, with $x_0 < x_1 < \cdots < x_n$.

We begin by listing all the inputs we are given:

- The interval $[a, b]$.
- The partition $\Delta = \{a = x_0 < x_1 < \cdots < x_n = b\}$.
- The set $Y = \{y_0, y_1, \dots, y_n\}$.

We define

Definition 31.1.1.

$$I_j = [x_{j-1}, x_j] \qquad \forall(1 \leq j \leq n)$$

and

$$h_j = x_j - x_{j-1} \qquad \forall(1 \leq j \leq n)$$

31.1.1 Brief description of the following subsections, and one new definition

We have not yet constructed the interpolating cubic spline, but we shall show that it is determined completely by the values of its second derivatives at the knots (the x_i).

To that end, define

Definition 31.1.2.

$$M_i = S''_\Delta(Y; x_i) \qquad \forall(1 \leq j \leq n)$$

Next, we shall find relationships between the M_i, that almost completely determine them given Y; just a few more conditions, and we'd be done. Then, if we impose any one of the three boundary conditions from the end of the notes for lecture 28 (which are actually relations between the M_i), we shall have obtained a set of conditions that uniquely determine our interpolating cubic spline.

Then, we shall observe that the conditions on the M_i are linear equations, and we shall use linear algebra to find a set of $M_0, M_1, \dots, M_n$ that make the interpolating polynomial pass through the support points (if such a set exists).

In what follows, we denote the cubic spline we wish to find by $S_\Delta(Y; x)$. Then, as a reminder, note that the interpolating conditions on $S_\Delta(Y;x)$ are

$$S_\Delta(Y;x_i) = y_i \qquad \forall (0 \le i \le n)$$

31.1.2 The interpolating cubic spline is determined entirely by its second derivatives at the knots

Note that since we are working with the abstract form of our cubic spline (that we have not yet constructed), we cannot actually compute the M_i. Nevertheless, we can work with them without knowing their values to show that they completely characterize the cubic spline.

In what follows, fix an arbitrary interval I_{k+1}. We state a theorem.

Theorem 31.1.1.

$$S''_\Delta(Y;x)|_{I_{k+1}} = M_k\left(\frac{x_{k+1}-x}{h_{k+1}}\right) + M_{k+1}\left(\frac{x-x_k}{h_{k+1}}\right)$$

Proof. By the definition of a cubic spline, $S''_\Delta(Y;x)|_{I_{k+1}} \in \Pi_1$, is a linear function. But we also know that we must have (by the definition of the M_i, see Definition 31.1.2)

$$S''_\Delta(Y;x_k)|_{I_{k+1}} = M_k$$

and

$$S''_\Delta(Y;x_{k+1})|_{I_{k+1}} = M_{k+1}$$

Linear functions are completely determined by their values at two distinct locations, and it is easy to verify that the formula given for $S''_\Delta(Y;x)|_{I_{k+1}}$ does satisfy the two equations above. ■

Theorem 31.1.2.

$$S_\Delta(Y;x)|_{I_{k+1}} = M_k\frac{(x_{k+1}-x)^3}{6h_{k+1}} + M_{k+1}\frac{(x-x_k)^3}{6h_{k+1}} + A_k(x-x_k) + B_k$$

where

$$\begin{aligned} B_k &= y_k - \frac{M_k h_{k+1}^2}{6} \\ A_k &= \frac{y_{k+1}-y_k}{h_{k+1}} - \frac{h_{k+1}}{6}(M_{k+1}-M_k) \end{aligned}$$

Proof. Use theorem 31.1.1 and integrate twice to obtain the formula

$$S_\Delta(Y;x)|_{I_{k+1}} = M_k\frac{(x_{k+1}-x)^3}{6h_{k+1}} + M_{k+1}\frac{(x-x_k)^3}{6h_{k+1}} + c_1x + c_2$$

for some constants c_1, c_2. Now, by choosing A_k and B_k properly, we can have $c_1x+c_2 = A_k(x-x_k)+B_k$.

To see that A_k and B_k have the forms mentioned in the statement of the theorem, note that by the interpolation conditions given to us

$$\begin{aligned} S_\Delta(Y;x_k)|_{I_{k+1}} &= y_k \\ S_\Delta(Y;x_{k+1})|_{I_{k+1}} &= y_{k+1} \end{aligned} \implies \begin{aligned} M_k\frac{h_{k+1}^2}{6} + B_k &= y_k \\ M_{k+1}\frac{h_{k+1}^2}{6} + A_k h_{k+1} + B_k &= y_{k+1} \end{aligned}$$

Solving for A_k and B_k gives us the desired result. ■

Remark: *The above theorem says that the interpolating cubic spline's restrictions to the intervals I_j can be calculated in terms of the M_i. Since the spline is defined by its behaviour on the restrictions to the I_j, that completely determines the spline.*

Now, let us try to find constants $\alpha_k, \beta_k, \gamma_k, \delta_k$ such that $S_\Delta(Y;x)|_{I_{k+1}} = \alpha_k(x-x_k)^3 + \beta_k(x-x_k)^2 + \gamma_k(x-x_k) + \delta_k$

Theorem 31.1.3.

$$S_\Delta(Y;x)|_{I_{k+1}} = \alpha_k(x-x_k)^3 + \beta_k(x-x_k)^2 + \gamma_k(x-x_k) + \delta_k$$

where

$$\begin{aligned}
\delta_k &= y_k \\
\gamma_k &= \frac{y_{k+1}-y_k}{h_{k+1}} - \frac{h_{k+1}}{6}(2M_k + M_{k+1}) \\
\beta_k &= \frac{M_k}{2} \\
\alpha_k &= \frac{M_{k+1}-M_k}{6h_{k+1}}
\end{aligned}$$

Proof. Use theorem 31.1.2, and differentiate, plug in support points, solve, repeat. ∎

31.1.3 Relations between the M_i

Since we haven't constructed the interpolating cubic spline $S_\Delta(Y;x)$, we cannot compute its M_i. But there exist relations between them, relations in the form of linear equations, that we can solve to determine the M_i. Unfortunately there are $(n+1)$ M_i to determine (namely $M_0, M_1, \ldots, M_n$), and in this section we shall only be able to derive $n-1$ linear equations involving the M_i.

That is where the boundary conditions described at the end of the notes for lecture 28 come in. They reduce the degrees of freedom by an additional 2, so that we get a system of equations in the M_i that we can solve uniquely (assuming the matrix formed by them is non-singular). The boundary conditions shall be described in greater detail in the next sub-section.

Theorem 31.1.4.

$$\frac{h_k}{6}M_{k-1} + \frac{h_k + h_{k+1}}{3}M_k + \frac{h_{k+1}}{6}M_{k+1} = \frac{y_{k+1}-y_k}{h_{k+1}} - \frac{y_k - y_{k-1}}{h_k}$$

for all $1 \leq k \leq n-1$.

Proof. Recall that we had imposed the condition that our cubic splines be at least twice continuously differentiable in lecture 28.

Then, since $S_\Delta(Y;x)$ is a cubic spline, we must have

$$\lim_{x\to x^-} S'_\Delta(Y;x) = \lim_{x\to x^+} S'_\Delta(Y;x)$$

Using theorem 31.1.3, and doing a lot of boring and uninspired algebra gives us the result. ∎

Remark: *The above theorem gives us $n-1$ linear equations in $n+1$ variables. We are short 2 equations. Which brings us to the next sub-section.*

31.1.4 Boundary conditions

We recall the three boundary conditions mentioned at the end of lecture 28.

1) $S''_\Delta(Y;a) = S''_\Delta(Y;b) = 0$.
2) $S^{(k)}_\Delta(Y;a) = S^{(k)}_\Delta(Y;b)$ for $k = 0, 1,$ and 2.
3) $S'_\Delta(Y;a) = y'_0$, and $S'_\Delta(Y;b) = y'_n$, where y'_0 and y'_n are fixed values given to you.

When used in conjunction with the interpolation conditions, any one of the three conditions above is sufficient to guarantee the existence of $n+1$ linear equations in the $n+1$ variables $M_0, M_1, \ldots, M_n$. We shall prove that below.

Theorem 31.1.5. Given the interpolation conditions, and the boundary condition 1), there are a total of $n+1$ linear equations in the $n+1$ variables $M_0, M_1, \ldots, M_n$.

Proof. Theorem 31.1.4 gives us $n-1$ linear equations in $M_0, M_1, \ldots, M_n$. Boundary condition 1) says that

$$S''_\Delta(Y;a) = S''_\Delta(Y;b) = 0 \quad\Longrightarrow\quad M_0 = M_n = 0 \quad\Longrightarrow\quad M_0 = 0 \text{ and } M_n = 0$$

and therefore gives us two more linear equations in $M_0, M_1, \ldots, M_n$. We now have $n+1$ linear equations in the $n+1$ variables $M_0, M_1, \ldots, M_n$. ■

Theorem 31.1.6. Given the interpolation conditions, and the boundary condition 2), there are a total of $n+1$ linear equations in the $n+1$ variables $M_0, M_1, \ldots, M_n$.

Proof. Boundary condition 2) says that

$$S^{(k)}_\Delta(Y;a) = S^{(k)}_\Delta(Y;b) \text{ for } k = 0, 1, \text{ and } 2$$

- **Implications of $S^{(k)}_\Delta(Y;a) = S^{(k)}_\Delta(Y;b)$ for $k = 0$:**

$$S^{(0)}_\Delta(Y;a) = S^{(0)}_\Delta(Y;b) \implies y_0 = y_n$$

- **Implications of $S^{(k)}_\Delta(Y;a) = S^{(k)}_\Delta(Y;b)$ for $k = 2$:**

$$S^{(2)}_\Delta(Y;a) = S^{(2)}_\Delta(Y;b) \implies M_0 = M_n$$

- **Implications of $S^{(k)}_\Delta(Y;a) = S^{(k)}_\Delta(Y;b)$ for $k = 1$:**

$$\begin{aligned}
&S^{(1)}_\Delta(Y;a) = S^{(1)}_\Delta(Y;b) \\
\overset{(1)}{\Longrightarrow}\ & -\frac{M_0 h_1}{2} + \frac{y_1 - y_0}{h_1} - \frac{h_1(M_1 - M_0)}{6} = \frac{M_n h_n}{2} + \frac{y_n - y_{n-1}}{2} - \frac{h_n(M_n - M_{n-1})}{6} \\
\overset{(2)}{\Longrightarrow}\ & \frac{h_n}{6} M_{n-1} + \frac{h_n + h_1}{3} M_n + \frac{h_1}{6} M_1 = \frac{y_1 - y_n}{h_1} - \frac{y_n - y_{n-1}}{h_n}
\end{aligned}$$

where implication (1) follows from, e.g., theorem 31.1.3, and implication (2) follows from the facts that $y_0 = y_n$ and $M_0 = M_n$ as we proved above.

We now have $n+1$ linear equations in the $n+1$ variables $M_0, M_1, \ldots, M_n$, namely

- The $n-1$ linear equations theorem 31.1.4 give us.

- The linear equation $M_0 = M_n$ the case $k = 2$ gives us.
- The linear equation
$$\frac{h_n}{6}M_{n-1} + \frac{h_n + h_1}{3}M_n + \frac{h_1}{6}M_1 = \frac{y_1 - y_n}{h_1} - \frac{y_n - y_{n-1}}{h_n}$$
the case $k = 1$ gives us.

■

Theorem 31.1.7. Given the interpolation conditions, and the boundary condition 3), there are a total of $n+1$ linear equations in the $n+1$ variables $M_0, M_1, \ldots, M_n$.

Proof. Left as an exercise to the reader. ■

31.1.5 Using linear algebra to construct the interpolating spline

We first introduce some new notation.

Definition 31.1.3.
$$\mu_k = \frac{h_k}{h_{k+1} + h_k} \qquad \forall(1 \le k \le n-1)$$
$$\lambda_k = \frac{h_{k+1}}{h_{k+1} + h_k} \qquad \forall(1 \le k \le n-1)$$
$$d_k = \frac{6}{h_{k+1} + h_k}\left(\frac{-y_k + y_{k+1}}{h_{k+1}} - \frac{y_k - y_{k-1}}{h_k}\right) \qquad \forall(1 \le k \le n-1)$$

Now, suppose we have decided to impose boundary condition 1) on the interpolating spline; that is, the condition
$$S''_\Delta(Y; a) = S''_\Delta(Y; b) = 0$$
In addition, define $\mu_n = \lambda_0 = 0$, and $d_0 = d_n = 0$. Then the $n+1$ conditions on $M_0, M_1, \ldots, M_n$ can be written as follows after multiplying all the conditions except $M_0 = 0$ and $M_n = 0$ by the factor $\frac{6}{h_{k+1}+h_k}$
$$\underbrace{\begin{bmatrix} 2 & \lambda_0 & & & & \\ \mu_1 & 2 & \lambda_1 & & & \\ & \mu_2 & 2 & \lambda_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \lambda_{n-1} \\ & & & & \mu_n & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ \vdots \\ M_n \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_n \end{bmatrix}}_{d}$$
Empty entries denote zeroes. Matrices of the form of A are called **tridiagonal matrices**, or **banded matrices**. There exist efficient algorithms specifically designed to compute the inverses of tridiagonal matrices (if they exist). Note that A depends only on the partition Δ, not on Y. If the matrix A is invertible, then the we can precompute A^{-1} and solve for the equation
$$AM = d$$
by computing $M = A^{-1}d$. Finally, note that by their definitions,
$$0 < \mu_k, \lambda_k < 1 \qquad \forall(1 \le k \le n-1)$$
We shall discuss the invertibility of A in greater detail in the notes for the next lecture.

Lecture 32
April 1, Part A

(This part is labelled lecture 32, part B on quadrature is 33)

32.1 Definitions from Lecture 31 recapped

Recall the following definitions for the spline function $S_\Delta(Y;x)$, with $\Delta = \{x_0, \ldots, x_n\}$ a partition of $[a,b]$ and $Y = \{y_0, \ldots, y_n\}$ a set of support points and the boundary values y_0', y_n'.

Definition 32.1.1.

- $h_{k+1} = x_{k+1} - x_k \ \forall\ k \in \{0, \ldots, n-1\}$
- Moments of the spline $M_k = S_\Delta''(Y;x_k) \ \forall\ k \in \{0, \ldots, n\}$
- $\lambda_k = \dfrac{h_{k+1}}{h_k + h_{k+1}} \ \forall\ k \in \{1, \ldots, n-1\}$
- $\mu_k = 1 - \lambda_k = \dfrac{h_k}{h_k + h_{k+1}} \ \forall\ k \in \{1, \ldots, n-1\}$
- $d_k = \dfrac{6}{h_k + h_{k+1}} \left(\dfrac{y_{k+1} - y_k}{h_{k+1}} - \dfrac{y_k - y_{k-1}}{h_k} \right) \ \forall\ k \in \{1, \ldots, n-1\}$
- The three boundary conditions imposed on splines, in terms of moments:

Case (a): $S_\Delta''(Y;a) = M_0 = 0 = M_n = S_\Delta''(Y;b)$

Case (b): $S_\Delta''(Y;a) = S_\Delta''(Y;b) \implies M_0 = M_n$

$$S_\Delta'(Y;a) = S_\Delta'(Y;b) \implies \frac{h_n}{6} M_{n-1} + \frac{h_n + h_1}{3} M_n + \frac{h_1}{6} M_1 = \frac{y_1 - y_n}{h_1} - \frac{y_n - y_{n-1}}{h_n}$$

Case (c): $S_\Delta'(Y;a) = y_0' \implies \dfrac{h_1}{3} M_0 + \dfrac{h_1}{6} M_1 = \dfrac{y_1 - y_0}{h_1} - y_0'$

$$S_\Delta'(Y;b) = y_n' \implies \frac{h_n}{6} M_{n-1} + \frac{h_n}{3} M_n = y_n' - \frac{y_n - y_{n-1}}{h_n}$$

We also have from the definition of the spline function that,

$$\frac{h_k}{h_k + h_{k+1}} M_{k-1} + 2M_k + \frac{h_{k+1}}{h_k + h_{k+1}} M_{k+1} = \frac{6}{h_k + h_{k+1}} \left(\frac{y_{k+1} - y_k}{h_{k+1}} - \frac{y_k - y_{k-1}}{h_k} \right) \tag{32.1}$$

for $k \in \{1, \ldots, n-1\}$

32.2 Finding the moments of the spline

Using the definitions 32.1.1, (32.1) becomes

$$\mu_k M_{k-1} + 2M_k + \lambda_k M_{k+1} = d_k$$

for $k \in \{1, \dots, n-1\}$.

We define λ, μ, d separately for each of the three boundary conditions as follows,

Case (a): $\lambda_0 = d_0 = 0,\ \mu_n = d_n = 0$

Case (b): $\lambda_n = \dfrac{h_1}{h_n + h_1},\ \mu_n = \dfrac{h_n}{h_n + h_1},\ d_n = \dfrac{6}{h_n + h_1}\left(\dfrac{y_1 - y_n}{h_1} - \dfrac{y_n - y_{n-1}}{h_n}\right)$

Case (c): $\lambda_0 = 1,\ d_0 = \dfrac{6}{h_1}\left(\dfrac{y_1 - y_0}{h_1} - y_0'\right),\ \mu_n = 1,\ d_n = \dfrac{6}{h_n}\left(y_n' - \dfrac{y_n - y_{n-1}}{h_n}\right)$

Including these equations in (32.1) we get the following matrix equations for the moments of the spline,

$$\begin{bmatrix} 2 & \lambda_0 & 0 & \cdots & 0 & 0 \\ \mu_1 & 2 & \lambda_1 & \cdots & 0 & 0 \\ 0 & \mu_2 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & \lambda_{n-1} \\ 0 & 0 & 0 & \cdots & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{32.2}$$

$$\begin{bmatrix} 2 & \lambda_1 & 0 & \cdots & 0 & \mu_1 \\ \mu_2 & 2 & \lambda_2 & \cdots & 0 & 0 \\ 0 & \mu_3 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & \lambda_{n-1} \\ \lambda_n & 0 & 0 & \cdots & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{32.3}$$

where, (32.2) holds for cases (a) and (c), and (32.3) holds for case (b) where $M_0 = M_n$ by periodicity.

Remark: *The coefficients λ_k, μ_k, d_k are all well-defined for all k in all three cases. Further, they depend only on the x_i and not on the support points y_i and the boundary values y_0', y_n'. We also have for all k,*

$$\lambda_k \geq 0,\ \mu_k \geq 0,\ \lambda_k + \mu_k = 1$$

32.3 Existence of moments of the spline

We now prove the following existence and uniqueness theorem.

Theorem 32.3.1. The systems (32.2) and (32.3) have unique solutions for all partitions Δ of $[a, b]$.

Proof. Let A be the coefficient matrix of (32.2). If $z, w \in \mathbb{R}^{n+1}$ be such that $Az = w$ and $|z_r| = \max_k |z_k|$, then

$$\begin{aligned} w_r &= \mu_r z_{r-1} + 2z_r + \lambda_r z_{r+1} \\ \implies |w_r| &\geq 2|z_r| - \mu_r|z_{r-1}| - \lambda_r|z_{r+1}| && \text{(Triangle inequality, } \lambda_k \geq 0, \mu_k \geq 0) \\ \implies \max_k |w_k| &\geq (2 - \mu_r - \lambda_r)|z_r| && \text{(Definition of } z_r) \\ \implies \max_k |w_k| &\geq |z_r| && (\lambda_k + \mu_k = 1) \end{aligned}$$

Hence, $Az = w \implies \max_k |z_k| \leq \max_k |w_k|$

Let B be the coefficient matrix of (32.3). If $z, w \in \mathbb{R}^n$ be such that $Bz = w$ and we define $z_0 = z_n, w_0 = w_n$, we get the same equations as for A by using the periodicity.

Hence, $Bz = w \implies \max_k |z_k| \leq \max_k |w_k|$.

Now, let C denote the coefficient matrix of (32.2) or (32.3). If C is singular, there is some vector $z \neq 0$ such that $Cz = 0$. But then $\max_k |z_k| \leq 0$ which is a contradiction.

Hence, C is non-singular, and the systems both have unique solutions. ■

Lecture 34
April 1, Part B

34.1 Quadrature

We have a function f about we don't know much and we want to calculate the integral $I = \int_a^b f(x)dx$, where $a, b, f(x)$ are given. Out of the many ways of solving this problem, a series of them can be classified into a series of formulae called the **Newton-Cotes Formulae**, some of which are:

- Trapezoidal Rule
- Simpson's Rule (Most popular for "smooth" functions f and when $b - a$ is finite.)
- Milne's Rule

34.1.1 Simpson's Rule

The defining feature of a Newton-Cotes formulation is that we have uniform partitioning Δ_u of $b - a$ into (say) n sub-intervals. Let $h = \frac{b-a}{n}$ be the length of each sub-interval. Then our partition can be specified by the two numbers (n, h). Let $x_0 = a,\ x_n = b$ and $x_i = x_0 + ih,\ 0 \leq i \leq n$. Define $I_j = [x_{j-1}, x_j],\ j = 1, \dots, n$ be the jth sub-interval.
Now we create an interpolating polynomial $P_n(x) \in \Pi_n$ such that $P(x_i) = f(x_i) = f_i,\ \forall i = 0, \dots, n$. We shall use this polynomial to approximate the integral. Note that $x_k - x_i = (k - i)h$. So, we have the condition that $x_k = x_i \implies k = i$.

Recall that the **Lagrange Interpolation** gives us the polynomial $P_n(x) = \sum\limits_{i=0}^{n} f_i L_i$ such that $L_i(x) = \prod\limits_{k=0(k\neq i)}^{n}$.
Let $t = \frac{x-a}{h}$ for any $x \in [a, b]$. Here, we shall scale/convert the variable x using the dimensionless variable t. Using this, we get $x_i - x_k = h(i - k)$. So, we can write

$$\begin{aligned} L_i(x) &= \prod_{k=0(k\neq i)}^{n} \frac{(x - x_k)}{(x_i - x_k)} \\ &= \prod_{k=0(k\neq i)}^{n} \frac{t-k}{i-k} = \Phi_i(t). \end{aligned}$$

Therefore, we can finally write

$$\begin{aligned}\int_a^b P_n(x)dx &= \sum_{i=0}^{n} f_i \int_a^b L_i(x)dx \\ &= \sum_{i=0}^{n} hf_i \int_{i=0}^{n} \Phi_i(t)dt \\ &= h\sum_{i=0}^{n} f_i\alpha_i\end{aligned}$$

where $\alpha_i = \int\limits_{t=0}^{n} \Phi_i(t)dt$. Note that the weights α_i depend only on n and do depend on a, b.

Example 34.1.1. Suppose we are given $n = 2$, the support ordinates f_0, f_1, f_2 and limits $a = 0, b = 2$. What is our approximation to I?
By Lagrange Interpolation, we first get $P_2(x) \in \Pi_2$. Then, using **Simpson's Rule**, We get

$$\begin{aligned}\int_0^2 P_2(x)dx &= \frac{h}{3}[f_0 + 4f_1 + f_2] \\ &= \frac{1}{3}[f_0 + 4f_1 + f_2]\ (h = 1).\end{aligned}$$

34.1.2 Trapezoidal Rule

34.1.3 Error

The error is given by $\int\limits_a^b P_n(x)cdx - \int\limits_a^b f(x)dx$.

This error term for the **Trapezoidal Rule** is given by $\frac{h^3}{12}f^{(2)}(\xi)$ for some $\xi \in [a, b]$. The same for **Simpson's Rule** is given by $\frac{h^5}{90}f^{(4)}f(\xi)$ for some $\xi \in [a, b]$.

Apply the rules in a composite form:-
Trapezoidal Rule
$\frac{h}{2[f(x_0)+f(x_1)]}$. We apply the Trapezoidal Rule to every interval $I_i = [x_i, x_{i+1}]$ for $i = 0, \ldots, n-1$ to get $\mathscr{I}_i = \frac{h}{2}[f(x_i) + f(x_{i+1})]$. Therefore, the total integral becomes

$$\begin{aligned}T(h) &= \sum_{i=0}^{n-1} \mathscr{I}_i \\ &= h[\frac{f(a)}{2} + f(a+h) + f(a+2h) + \ldots + f(b-h) + \frac{f(b)}{2}]\end{aligned}$$

34.2 Approximation (Composite) for Simpson's Rule

Trapezoidal Rule is an $O(2)$ method while Simpson's Rule is an $O(4)$ method.

Lecture 35
April 4, Part A

We have already seen that we can find the integration $\int_a^b f(x)\,\mathrm{d}x$ using **Lagrange's interpolation**. Now we try to find additional integration rules using **Hermite interpolation**. For this we take the simple example when $P \in \Pi_3$ such that

$$P^{(i)}(a) = f^{(i)}(a) \text{ and } P^{(i)}(b) = f^{(i)}(b),\ \forall\, i = 0, 1 \tag{35.0.1}$$

For simplicity we take $a = 0,\ b = 1$. Thus recall from **Lecture 25**, that the **Hermite interpolation** gives

$$P(x) = \sum_{i=0}^{1}\sum_{k=0}^{1} f_i^{(k)} L_{ik}(x)$$

we get that

$$\begin{aligned} L_{01}(x) &= x(x-1)^2 &\quad \text{and} \quad&& L_{00}(x) &= (x-1)^2 + 2x(x-1)^2 \\ L_{11}(x) &= x^2(x-1) &\quad \text{and} \quad&& L_{00}(x) &= x^2 - 2x^2(x-1) \end{aligned}$$

and hence, we get that

$$P(x) = f(0)[(x-1)^2 + 2x(x-1)^2] + f'(0)x(x-1)^2 + f(1)[x^2 - 2x^2(x-1)] + f'(1)x^2(x-1)$$

and hence we get that

$$\int_0^1 P(x)\,\mathrm{d}x = \frac{1}{2}(f(0) + f(1)) + \frac{1}{12}(f'(0) - f'(1))$$

but then for general a and b, we can take $P(t) = f(a + ht)$ where $h := b - a$, and then

$$\begin{aligned} \int_a^b f(x)\,\mathrm{d}x &= h\int_0^1 f(a+ht)\,\mathrm{d}t \\ &\approx \int_0^1 P(t)\,\mathrm{d}t \\ &= \frac{h}{2}(f(a) + f(b)) + \frac{h^2}{12}(f'(a) - f'(b)) \end{aligned}$$

Now, when are computing the integral using different quadrature rule (integration rules) such as the above **Hermite interpolation** one, we don't apply the formula on the whole interval, instead we apply it on subintervals into which the interval $[a, b]$ has been divided. The full integration is then approximated by the sum of the approximations to the subintervals. We are here approximating the integrals locally, and then extending it, this gives rise to a composite rule. We now look at the composite rules in various cases quadrature rules.

35.1 Composition of Errors in Trapezoidal and Simpson's Rule

35.1.1 Trapezoidal Rule

In this case we assume that the quadrature rule on an interval $[x_i, x_{i+1}]$ is given by

$$I_i = \frac{h}{2}(f(x_i) + f(x_{i+1})) \tag{35.1.1}$$

where $[x_i, x_{i+1}]$ are the subintervals formed by the partition $x_i = a + ih, \ \forall\, i = 0, 1, \ldots, N$, where $h := \frac{b-a}{N}$. Thus for the entire interval we get the approximation

$$\begin{aligned} T(h) &:= \sum_{i=0}^{N-1} I_i \\ &= h\left(\frac{f(a)}{2} + f(a+h) + f(a+2h) + \cdots + f(b-h) + \frac{f(b)}{2}\right) \end{aligned}$$

Assuming that $f \in \mathscr{C}^2[a,b]$, it can be shown that (using **Peano's Error formula** which is stated in the next section) the error denoted by ε_i is

$$\varepsilon_i := I_i - \int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x = \frac{h^3}{12} f^{(2)}(\xi_i) \tag{35.1.2}$$

for some $\xi_i \in (x_i, x_{i+1})$. Thus summing all these individual error terms we get that

$$\begin{aligned} T(h) - \int_a^b f(x)\,\mathrm{d}x &= \sum_{i=0}^{N-1}\left(I_i - \int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x\right) \\ &= \frac{h^3}{12}\sum_{i=0}^{N-1} f^{(2)}(\xi_i) \\ &= \frac{h^2}{12}(b-a)\left(\frac{1}{N}\sum_{i=0}^{N-1} f^{(2)}(\xi_i)\right) \\ &\overset{(1)}{=} \frac{h^2}{12}(b-a) f^{(2)}(\xi) \end{aligned}$$

where (1) follows from the fact that

$$\min_i f^{(2)}(\xi_i) \le \frac{1}{N}\sum_{i=0}^{N-1} f^{(2)}(\xi_i) \le \max_i f^{(2)}(\xi_i)$$

and since $f \in \mathscr{C}^2[a,b]$ hence by **Intermediate Value Property**, there exists a $\xi \in [\min_i \xi_i, \max_i \xi_i]$ such that

$$f^{(2)}(\xi) = \frac{1}{N}\sum_{i=0}^{N-1} f^{(2)}(\xi_i)$$

Hence we have shown that

$$T(h) - \int_a^b f(x)\,\mathrm{d}x = \frac{h^2}{12}(b-a) f^{(2)}(\xi), \quad \text{for some } \xi \in (a,b) \tag{35.1.3}$$

Thus we get that as h tends to 0, the approximation error approaches zero as fast as h^2, so the Trapezoidal rule is a method of *order* 2. So what have shown is the following

Theorem 35.1.1. The approximation error in the **Trapezoidal rule** approaches 0 as fast as h^2, i.e., the **Trapezoidal rule** is a method of *order* 2, and in general for the partition

$$\Delta = \{x_0 = a < x_1 = a + h < \cdots < x_{N-1} = b - h < x_N = b\}$$

the error term is given by

$$\varepsilon_T := T(h) - \int_a^b f(x)\,\mathrm{d}x = \frac{h^2}{12}(b-a)f^{(2)}(\xi), \quad \text{for some } \xi \in (a,b)$$

where $h = \frac{b-a}{N}$.

35.1.2 Simpson's Rule

Recall that in Simpson's rule, the integration rule on the interval $[a,b]$ is given by the **Lagrange interpolate** $P \in \Pi_2$ with nodes at $x_0 = a, x_1 = a + h$ and $x_2 = b$, where $h = b - a$

$$\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b P(x)\,\mathrm{d}x = \frac{h}{3}(f_0 + 4f_1 + f_2)$$

where $f_i = f(x_i)$ for $i = 0, 1, 2$.

Now we assume that the number of nodes in the partition of the interval $[a,b]$ is even, i.e., let

$$\Delta = \{x_0 = a < x_1 = a + h < \cdots < x_N = b\}$$

where N is even. Now we apply Simpson's rule on each of the subintervals $[x_{2i}, x_{2i+2}]$, with nodes at x_{2i}, x_{2i+1} and x_{2i+2}, then we get that

$$I_i = \frac{h}{3}[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})], \quad \forall\, i = 0, 1, \ldots, \frac{N}{2} - 1$$

Summing all these $\frac{N}{2}$ approximations we define

$$S(h) := \frac{h}{3}\left(f(a) + 4f(a+h) + 2f(a+2h) + \cdots + 2f(b-2h) + 4f(b-h) + f(b)\right) \tag{35.1.4}$$

now we assume that $f \in \mathscr{C}^4[a,b]$, then it can be shown (using **Peano's Error formula** which is stated in the next section) the error denoted by ε_i is

$$\varepsilon_i := I_i - \int_{x_{2i}}^{x_{2i+2}} f(x)\,\mathrm{d}x = \frac{h^5}{90}f^{(4)}(\xi_i) \tag{35.1.5}$$

for some $\xi_i \in (x_{2i}, x_{2i+2})$. Thus like in the case of **Trapezoidal rule**, in we sum up all these error terms and with similar arguments we get that

$$S(h) - \int_a^b f(x)\,\mathrm{d}x = \frac{h^4}{180}(b-a)f^{(4)}(\xi), \quad \text{for some } \xi \in (a,b) \tag{35.1.6}$$

Thus we conclude this section with the following theorem:

Theorem 35.1.2. The approximation error in the **Simpson's rule** approaches 0 as fast as h^4, i.e., the **Simpson's rule** is a method of *order* 4, and in general for the partition

$$\Delta = \{x_0 = a < x_1 = a + h < \cdots < x_{N-1} = b - h < x_N = b\}$$

with N even, the error term is given by

$$\varepsilon_S := S(h) - \int_a^b f(x)\,\mathrm{d}x = \frac{h^4}{180}(b-a)f^{(4)}(\xi), \quad \text{for some } \xi \in (a,b)$$

where $h = \frac{b-a}{N}$.

35.2 Peano's Error Representations

Observe that all the integration rules (quadrature rules), we have used so far are of the form

$$\tilde{I}(f) := \sum_{j=0}^{n}\sum_{k=0}^{m_j} a_{kj} f^{(j)}(x_{kj}) \tag{35.2.1}$$

For a given integration rule $\tilde{I}$ as in equation (35.2.1), we define the integration error as

$$R(f) := \tilde{I}(f) - \int_a^b f(x)\,\mathrm{d}x \tag{35.2.2}$$

Remark: *Observe that $R : V \to \mathbb{R}$ is a **linear functional**, i.e.,*

$$R(\alpha f + \beta g) = \alpha R(f) + \beta R(g),\ \forall\, f, g \in V,\ \text{and } \alpha, \beta \in \mathbb{R}$$

where V is a suitable vector space over the field $\mathbb{R}$, for example $V = \mathbb{C}^n[a,b]$ or $V = \Pi_n$.

Now we state the following result which is due to **Peano**:

Theorem 35.2.1. (Peano's Error Formula). Suppose I be an integration rule such that the corresponding integration error satisfies the condition that $R(P) = 0$ for all $P \in \Pi_n$, i.e., every polynomial whose degree is at most n, is integrated exactly. Then for all functions $f \in \mathscr{C}^{n+1}[a,b]$, we have

$$R(f) = \int_a^b f^{(n+1)}(t) K(t)\,\mathrm{d}t$$

where

$$K(t) := \frac{1}{n!} R_x[(x-t)_+^n], \qquad (x-t)_+^n := \begin{cases} (x-t)^n & \text{if } x \geq t \\ 0 & \text{otherwise} \end{cases}$$

and $R_x[(x-t)_+^n]$ denotes the integration error of $(x-t)_+^n$ when considered as a function of x. The function $K(t)$ is called the **Peano kernel** of the linear functional R.

We first give an application of the **Peano's error formula**, in the case of **Simpson's rule**. Note that

$$\int_a^b f(x)\,\mathrm{d}x = \int_{-1}^{1} f(t)\,\mathrm{d}t = \frac{b-a}{2}\int_{-1}^{1} f\left(\left(\frac{b-a}{2}\right)t + \frac{b+a}{2}\right)\mathrm{d}t = \frac{b-a}{a}\int_{-1}^{1} g(t)\,\mathrm{d}t$$

where $g(t) = f\left(\left(\frac{b-a}{2}\right)t + \frac{b+a}{2}\right)$. Thus the task of approximating the integral $\int_a^b f(x)\,\mathrm{d}x$ is equaivalent to approximating the integral $\int_{-1}^1 g(x)\,\mathrm{d}x$, thus WLOG, we may assume that the nodes of the **Lagrange interpolate** for the **Simpson's rule** to be $-1, 0, 1$.

First of all we need to show that $R(P) = 0$ for all $P \in \Pi_3$. So we let P be any polynomial in Π_3, and consider the **Lagrange interpolate polynomial** $Q \in \Pi_2$ defined by

$$Q(-1) = P(-1), \qquad Q(0) = P(0) \qquad \text{and} \qquad Q(1) = P(1)$$

and set

$$S(x) := P(x) - Q(x)$$

Then note that $x = -1, 0, 1$ are roots of S and since $S \in \Pi_3$, we must have $S(x) = ax(x^2 - 1)$ for some constant $a \in \mathbb{R}$. Thus we get that S is an odd function and hence we get that

$$\int_{-1}^1 S(x)\,\mathrm{d}x = 0$$

and hence we get

$$R(S) = \frac{1}{3}(S(-1) + 4S(0) + S(1)) - \int_{-1}^1 S(x)\,\mathrm{d}x = 0$$

Now we have $Q \in \Pi_2$, thus we must have $Q(x) = px^2 + qx + r$ for some constants $p, q, r \in \mathbb{R}$, and hence we get that

$$\begin{aligned}
R(Q) &= \frac{1}{3}(Q(-1) + Q(0) + Q(1)) - \int_{-1}^1 Q(x)\,\mathrm{d}x \\
&= \frac{1}{3}((p - q + r) + 4r + (p + q + r)) - \int_{-1}^1 (px^2 + qx + r)\,\mathrm{d}x \\
&= \frac{1}{3}(2p + 6r) - \left(p\frac{x^3}{3} + q\frac{x^2}{2} + rx\right)\Bigg|_{-1}^1 \\
&= \frac{2}{3}(p + 3r) - \frac{2}{3}(p + 3r) \\
&= 0
\end{aligned}$$

and finally since R is a linear functional we get that

$$R(P) = R(S) + R(Q) = 0$$

but since P was arbitrary we get that $R(P) = 0$ for all $P \in \Pi_3$. The the **Simpson's rule** indeed satisfies the hypothesis of **Peano's Theorem** 35.2.1, for $n = 3$.

Lecture 36
April 4, Part B

36.1 Proof of Peano's result

Check Stoer and Bulirsch.

36.2 Gaussian Integration

36.2.1 Motivation and Introduction

The Newton-Cotes formulae we have seen so far are derived assuming that the size of each interval in the partition we are numerically integrating over is the same. This shall not be the case with Gaussian integration. In addition, Gaussian integration is deeper connections to other areas of numerical analysis, and even analysis in general.

But enough motivation! Let's get to the meat of this topic, and begin by laying out our task. Suppose you have been given a function f to integrate on an interval $[a, b]$. Then if you're using Gaussian integration to accomplish that you integrate

Definition 36.2.1.

$$\mathscr{I}(f) = \int_a^b \omega(x) f(x)\, \mathrm{d}x$$

where ω is a **weight function** that you have chosen, that satisfies

$$\omega \geq 0$$

The function ω must satisfy some conditions, which we will list after stating a definition used to frame them.

Definition 36.2.2. For all integers $k \geq 0$,

$$\mu_k = \int_a^b x^k \omega(x)\, \mathrm{d}x$$

The conditions ω must satisfy are

Definition 36.2.3.

- ω must be measurable on $[a,b]$.
- μ_k must exist and be finite for all $k \in \mathbb{Z}$, such that $k \geq 0$.
- For any polynomial s, if $s \geq 0$ on $[a,b]$, then

$$\int_a^b s(x)\omega(x)\,\mathrm{d}x = 0 \qquad \Longrightarrow \qquad s = 0 \text{ on } [a,b]$$

Remark: *By choosing $\omega = 1$, we recover the Newton-Cotes formulae we have discussed so far.*

One advantage of Gaussian integration is that it lets you "divide intervals in uneven sizes", and deal with intervals of infinite length. In particular, in the discussion above, a and b might be $\pm\infty$.

Example 36.2.1. Let c be any non-zero real. Then, the integral

$$\int_{-\infty}^{\infty} c\,\mathrm{d}x$$

does not exist, but the Gaussian integral of c over $(-\infty,\infty)$ obtained by choosing $\omega(x) = e^{-x^2}$, namely

$$\int_{-\infty}^{\infty} e^{-x^2} c\,\mathrm{d}x$$

does exist.

The above example immediately raises the question: How do we choose the weight function ω to use? Some kinds of weight functions that are commonly used are

- Legendre polynomials
- Laguerre polynomials
- Hermite polynomials
- Tchebyshev polynomials

The classes of polynomials listed above form **orthogonal** classes of polynomials, and the next subsection is dedicated to defining the basic notions required to explain what that means.

36.2.2 Orthogonal Polynomials

Definition 36.2.4. We define, for all integers $n \geq 0$,

$$\Pi_n = \{p \in R[x] \colon \deg(p) \leq n\}$$

and

$$\overline{\Pi}_n = \{p \in \Pi_n \colon \ p \text{ is monic } \}$$

Note that Π_n is a vector space.

We introduce some more definitions. In the following definitions, ω is a *weight function* on a *fixed interval* $[a,b]$ satisfying the conditions in 36.2.3.

Definition 36.2.5. Now fix an integer $n \geq 0$.
Since Π_n is a vector space, we can give it an inner product, which we do by defining the inner product $\langle \cdot, \cdot \rangle$ by

$$\langle f, g \rangle = \int_a^b \omega(x) f(x) g(x) \, \mathrm{d}x$$

for all $f, g \in \Pi_n$.

and one more definition before the definition of orthogonal polynomials:

Definition 36.2.6.

$$L^2[a,b]_\omega = \{ \langle f, f \rangle = \int_a^b \omega(x) \left(f(x) \right)^2 \mathrm{d}x \text{ exists and is finite } \}$$

for all $f \in \Pi_n$.

and finally, the definition of orthogonal polynomials

Definition 36.2.7. If $p, q \in \Pi_n$ for some integer n, we say that p and q are **orthogonal** if

$$\langle p, q \rangle = 0$$

We are done with the notes for Part B of the NC lecture on April 4.

Lecture 37
April 11, Part A

Let's recall some definition from the previous lecture.

Definition 37.0.8.

- $$\mathcal{I}(f) = \int_a^b \omega(x) f(x)\,\mathrm{d}x$$
  where ω is a **weight function** that you have chosen, that satisfying $\omega \geq 0$
- For all integers $k \geq 0$,
  $$\mu_k = \int_a^b x^k \omega(x)\,\mathrm{d}x$$
  For the above two,
  - ω must be measurable on $[a, b]$.
  - μ_k must exist and be finite for all $k \in \mathbb{Z}$, such that $k \geq 0$.
  - For any polynomial s, if $s \geq 0$ on $[a, b]$, then
    $$\int_a^b s(x)\omega(x)\,\mathrm{d}x = 0 \qquad \Longrightarrow \qquad s = 0 \text{ on } [a, b]$$
- $\forall$ integers $n \geq 0$,
  $$\Pi_n = \{p \in \mathbb{R}[x] \colon \deg(p) \leq n\} \text{ and } \overline{\Pi}_n = \{p \in \Pi_n \colon \ p \text{ is monic }\}$$
  Note that Π_n is a vector space. Also, $\overline{\Pi}_n \subset \Pi_n$
- Fix an integer $n \geq 0$. We can define an inner product $\langle \cdot, \cdot \rangle$ on Π_n by
  $$\langle f, g \rangle = \int_a^b \omega(x) f(x) g(x)\,\mathrm{d}x$$
  for all $f, g \in \Pi_n$.
- $$L^2[a, b]_\omega = \{\langle f, f \rangle = \int_a^b \omega(x)\left(f(x)\right)^2 \mathrm{d}x \text{ exists and is finite }\}$$
  for all $f \in \Pi_n$.
- If $p, q \in \Pi_n$ for some integer n, we say that p and q are **orthogonal** if $\langle p, q \rangle = 0$

37.1 Gram-Schmidt Orthogonalization on polynomials

For the vector space of all polynomials with at most degree n, Π_n, we can apply Gram-Schmidt Orthogonalization to the basis, $\{1, x, \ldots, x^n\}$ w.r.t the inner product defined above and obtain an orthogonal basis, $\{p_0(x), p_1(x), \ldots, p_n(x)\}$ with $\deg(p_r) = r \ \forall\ r$.

We can compute the orthogonal polynomials as follows:
Let $u_k(x) = x^k, \ \forall\ k \in \{0, 1, \ldots, n\}$

- $p_0 = u_0$
- $p_k = u_k - \sum_{j=0}^{k-1} \frac{\langle u_k, p_j \rangle}{\langle p_j, p_j \rangle} p_j$ for $k \in \{1, \ldots, n\}$

Recursively, we can write the above as,

- $p_0 = u_0$,i.e., $p_0(x) = 1 \ \forall\ x \in [a, b]$
- $p_k(x) = \left(x - \frac{\langle x p_{k-1}, p_{k-1} \rangle}{\langle p_{k-1}, p_{k-1} \rangle}\right) p_{k-1} - \frac{\langle x p_{k-1}, p_{k-1} \rangle}{\langle p_{k-2}, p_{k-2} \rangle} p_{k-2}(x)$ for $k \in \{1, 2, \ldots, n\}$

Note that, here we take $p_{-1} = 0$

Example 37.1.1. Let's explicitly compute p_0, p_1, p_2 with $a = -1, b = 1$. Note that, we care only about the roots of the polynomials, so we will save ourselves the trouble of normalizing the polynomials. We start with the usual basis, $u_0(x) = 1, u_1(x) = x, u_2(x) = x^2$
We take, $p_0 = u_0$. Now, taking out the p_0-component from u_1 we left with

$$p_1(x) = u_1(x) - \frac{\langle u_1, p_0 \rangle}{\langle p_0, p_0 \rangle} p_0(x) = x$$

Since,

$$\langle u_1, p_0 \rangle = \int_{-1}^{1} x \, \mathrm{d}x = 0$$

To find, p_2, we similarly take out the p_0 and p_1-components from u_2 to get,

$$p_2(x) = u_2(x) - \frac{\langle u_2, p_0 \rangle}{\langle p_0, p_0 \rangle} p_0(x) - \frac{\langle u_2, p_1 \rangle}{\langle p_1, p_1 \rangle} p_1(x) = x^2 - \frac{1}{3}$$

We now state and prove an important theorem.

Theorem 37.1.2. Each p_n has n distinct zeros inside the open interval (a, b)

Proof. Since, p_n is orthogonal to Π_{n-1}, for any polynomial q with $\deg(q) < n$, we must have

$$\langle q, p_n \rangle = \int_a^b q(x) p_n(x) \, \mathrm{d}x = 0$$

Let p_n have exactly m distinct real zeros of odd multiplicities inside (a, b). Call them, $\alpha_1, \alpha_2, \ldots, \alpha_m$. Define a polynomial,

$$q(x) := (x - \alpha_1) \cdots (x - \alpha_m)$$

Then $q(x)p_n(x)$ has all real zeros of even multiplicities, and hence does not change sign over (a, b). So,

$$\int_a^b q(x)p_n(x)\,\mathrm{d}x \neq 0$$

By the construction of $p_n(x)$ this forces $\deg(q) \geq n$. But $m \leq n$ and hence $m = n$.
So, p_n has exactly distinct roots (with odd multiplicities). Since p_n has degree n, all the zeroes are real and distinct and inside (a, b). ■

Lecture 38
April 11, Part B

38.1 Basic Properties of Orthogonal Polynomials

Now that we have shown that there exists polynomials $p_j \in \bar{\Pi}_j$, for $j = 0, 1, 2, \ldots$, such that

$$(p_i, p_j) = 0, \quad \text{for } i \neq j$$

Thus we have $\{p_1, \ldots, p_k\}$ is an orthonormal basis for the inner product space Π_k, with inner product defined by the *weight functions*

$$(f, g) := \int_a^b w(x)f(x)g(x)\,\mathrm{d}x\,, \quad f, g \in \Pi_j$$

Thus any polynomial $p \in \Pi_k$ is clearly representable as a linear combination of the orthogonal polynomials p_i, $i \leq k$, which in fact gives us the following lemma:

Lemma 38.1.1. $(p, p_n) = 0$, for all $p \in \Pi_{n-1}$.

Proof. We have $\{p_0, \ldots, p_{n-1}\}$ is an orthonormal basis for Π_{n-1}, and hence there exists scalars $a_0, a_1, \ldots, a_{n-1} \in \mathbb{R}$ such that

$$p(x) = \sum_{i=0}^{n-1} a_i p_i(x)$$

and thus by linearity of inner product we get

$$\begin{aligned}(p, p_n) &= \left(\sum_{i=0}^{n-1} a_i p_i, p\right) \\ &= \sum_{i=0}^{n-1} a_i (p_i, p) \\ &= 0\end{aligned}$$

■

Theorem 38.1.2. The roots of the polynomial p_n are real and simple.

Proof. We the consider the roots of p_n which are of odd multiplicities, i.e., we consider the roots where p_n changes sign. WLOG let them be $x_1, \ldots, x_l$, where $l \leq n$, and we define the polynomial $q(x)$ by

$$q(x) = \prod_{j=1}^{l} (x - x_j) \in \bar{\Pi}_l \tag{38.1.1}$$

But then observe that the polynomial $p_n(x)q(x)$ does not change sign (as all the roots are of even multiplicities now). Thus we must have

$$(p_n, q) = \int_a^b w(x)p_n(x)q(x)\,\mathrm{d}x \neq 0$$

but then deg$(q) = l$ cannot be strictly less than n, as otherwise we would have $(p_n, q) = 0$ (from **Lemma** 38.1.1), thus it must be the case that $l = n$. But then all the roots of p_n have multiplicity 1, and thus we get that roots of p_n are real and simple. ∎

We need this lecture by giving a glimpse of what we will do in the next lecture.

38.2 Glimpse of Exact Integration using Gaussian Integration

Consider $x_1, x_2, \dots, x_n$ be the roots of the orthogonal polynomial p_n. We consider the following matrix:

$$A := \begin{bmatrix} p_0(x_1) & p_0(x_2) & \cdots & p_0(x_n) \\ p_1(x_1) & p_1(x_2) & \cdots & p_1(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ p_{n-1}(x_1) & p_{n-1}(x_1) & \cdots & p_{n-1}(x_n) \end{bmatrix} \tag{38.2.1}$$

we will show that the following matrix is in fact nonsingular, thus we can consider the system of linear equations

$$\sum_{i=1}^{n} p_k(x_i)w_i = \begin{cases} (p_0, p_0) = \int_a^b w(x)\,\mathrm{d}x & \text{if } k = 0 \\ 0 & \text{if } k = 1, \dots, n-1 \end{cases} \tag{38.2.2}$$

then as we will see, we get $w_i > 0$ for $i = 1, \dots, n$ and in fact we have

$$\int_a^b w(x)p(x)\,\mathrm{d}x = \sum_{i=1}^{n} w_i p(x_i) \tag{38.2.3}$$

holds for all polynomial $p \in \Pi_{2n-1}$.

Lecture 39
April 18

39.1 Exactly integrating polynomials using Gaussian integration

39.1.1 Basic definitions and setup

Recall that in previous lectures we have proved that given

- an interval $[a, b]$.
- and a weight function ω on $[a, b]$.

there exist polynomials $p_0, p_1, p_2, \ldots, p_n \in \overline{\Pi}_n$, such that

- $p_i \in \overline{\Pi}_i$ for $0 \le i \le n$.
- The collection of polynomials $p_0, p_1, \ldots, p_k$ form an orthonormal basis for Π_k, for all $0 \le k \le n$.
- p_n has n simple roots.
- The p_i can be calculated using a recursive relation.

Let $x_1, x_2, \ldots, x_n$ be the n roots of p_n. Then, define the matrix $\mathbb{A}$ by

Definition 39.1.1.

$$[A]_{ki} = p_k(x_i) \qquad \forall(0 \le k \le n-1, 1 \le i \le n)$$

Then, we want to solve the system of equations given by

$$\underbrace{\begin{bmatrix} p_0(x_1) & p_0(x_2) & \ldots & p_0(x_n) \\ p_1(x_1) & p_1(x_2) & \ldots & p_1(x_n) \\ \vdots & \vdots & \vdots & \vdots \\ p_{n-1} & p_n(x_2) & \ldots & p_{n-1}(x_n) \end{bmatrix}}_{A} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \langle p_0, p_0 \rangle \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{39.1.1}$$

To clarify, the column vector on the right has $\langle p_0, p_0 \rangle$ in it's first row, and 0 in all the other rows. The reason we want to solve the above set of equations is that, if we know the values of the w_i, we can **exactly** integrate ω times any polynomial of degree at most $\boldsymbol{2n-1}$ over the interval $[a, b]$.

To solve the system of equations, we first show that

Theorem 39.1.1. The matrix A is invertible.

Proof. We shall show that for any vector c, $Ac = 0 \implies c = 0$. That shall be enough to prove that A is invertible. Let $c = (c_1, c_2, \dots, c_n)^T$ be a any vector such that $Ac = 0$. Now, $Ac = 0 \implies c^T A = 0$. Expanding the last equality into a system of equations, we obtain that

$$\sum_{i=0}^{n-1} c_i p_i(x_k) = 0 \qquad \forall (1 \leq k \leq n)$$

Therefore the polynomial Q defined by $Q(x) = \sum_{i=0}^{n-1} c_i p_i(x)$ has the roots $x_1, x_2, \dots, x_n$.

We have already proved that $u \neq v \implies x_u \neq x_v$; it is simply the statement that p_n has n simple roots.

Therefore, Q has at least n roots, namely $x_1, x_2, \dots, x_n$. But from the definition of Q, namely $Q(x) = \sum_{i=0}^{n-1} c_i p_i(x)$, and the fact that $p_i \in \overline{\Pi}_i$ for $0 \leq i \leq n$, we know that the degree of Q is at most $n-1$.

Therefore, Q must be identically 0, which implies that $c_j = 0$ for all $1 \leq j \leq k$. ■

Corollary 39.1.1.1. The solution to the system of equations 39.1.1 is unique. ■

39.1.2 The exact integration

Next, we come to the most important theorem about Gaussian integration:

Theorem 39.1.2. Let $p \in \Pi_{2n-1}$. Then,

$$\int_a^b \omega(x) p(x) \,\mathrm{d}x = \sum_{i=1}^{n} \omega_i p(x_i)$$

where the w_i are the solutions to the system of equations 39.1.1.

Proof. Use the Euclidean algorithm to write p as

$$p = p_n q + r$$

where $q \in \Pi_{n-1}$. Then it must also be the case that $r \in \Pi_{n-1}$. Write

$$q(x) = \sum_{i=0}^{n-1} \alpha_i p_i(x) \quad \text{and} \quad r(x) = \sum_{i=0}^{n-1} \beta_i p_i(x)$$

Then, note that

$$\int_a^b \omega(x) p(x) \,\mathrm{d}x = \underbrace{\int_a^b \omega(x) p_n(x) q(x) \,\mathrm{d}x}_{\text{Term 1}} + \underbrace{\int_a^b \omega(x) r(x) \,\mathrm{d}x}_{\text{Term 2}}$$

Since $q(x) = \sum_{i=0}^{n-1} \alpha_i p_i(x)$ and $p_0, p_1, \dots, p_n$ form an orthonormal basis for Π_n, it follows that Term 1 is 0. As for Term 2, note that

$$\int_a^b \omega(x) r(x) \,\mathrm{d}x = \langle p_0, p \rangle = \sum_{i=0}^{n-1} \beta_i \langle p_0, p_k \rangle = \beta_0 \langle p_0, p_0 \rangle$$

since $p_0, p_1, \dots, p_n$ form an orthonormal basis for Π_n. Therefore, $\int_a^b \omega(x) p(x) \,\mathrm{d}x = \beta_0 \langle p_0, p_0 \rangle$.

Next, we shall show that $\sum_{i=1}^n \omega_i p(x_i) = \beta_0\langle p_0, p_0\rangle$. That shall finish the proof. To that end, note that

$$\begin{aligned}\sum_{i=1}^n \omega_i p(x_i) &= \sum_{i=1}^n \omega_i (p_n(x_i)q(x_i) + r(x_i)) \\ &\overset{(1)}{=} \sum_{i=1}^n \omega_i r(x_i) \\ &= \sum_{i=1}^n \sum_{k=0}^{n-1} \omega_i \beta_k p_k(x_i) \\ &\overset{(2)}{=} \sum_{k=0}^{n-1} \sum_{i=1}^n \omega_i \beta_k p_k(x_i) \\ &= \sum_{k=0}^{n-1} \beta_k \sum_{i=1}^n \omega_i p_k(x_i) \\ &\overset{(3)}{=} \beta_0\langle p_0, p_0\rangle\end{aligned}$$

where equality (1) follows from the fact that $x_1, x_2, \ldots, x_n$ are roots of p_n, equality (2) from interchanging the sums, and equality (3) from the fact that the ω_i satisfy the system of equations 39.1.1. ■

We now state and prove another important theorem

Theorem 39.1.3. The w_i in the system of equations 39.1.1 are all greater than 0.

Proof. Define, for all integers j such that $1 \le j \le n$,

$$\overline{p}_j(x) = \prod_{\substack{k=1 \\ k\neq j}}^n (x - x_k)^2$$

Note that $\overline{p}_j \in \Pi_{2n-2} \in \Pi_{2n-1}$ for all $1 \le j \le n$, which implies, by theorem 39.1.2, that

$$\int_a^b \omega(x)\overline{p}_j(x)\,\mathrm{d}x = \sum_{i=1}^n \omega_i \overline{p}_j(x_i) = \sum_{i=1}^n \omega_i \left(\prod_{\substack{k=1 \\ k\neq j}}^n (x_i - x_k)^2\right)$$

But since $\prod_{\substack{k=1 \\ k\neq j}}^n (x_i - x_k)^2$ for all $i \neq j$, it follows that $\sum_{i=1}^n \omega_i \left(\prod_{\substack{k=1 \\ k\neq j}}^n (x_i - x_k)^2\right) = \omega_j \left(\prod_{\substack{k=1 \\ k\neq j}}^n (x_j - x_k)^2\right)$.
Therefore, we have

$$\int_a^b \omega(x)\overline{p}_j(x)\,\mathrm{d}x = \omega_j \left(\prod_{\substack{k=1 \\ k\neq j}}^n (x_j - x_k)^2\right)$$

Now, recall that any weight function on an interval $[a, b]$ must satisfy the condition that for any polynomial s, if $s \ge 0$ on $[a, b]$, then

$$\int_a^b s(x)\omega(x)\,\mathrm{d}x = 0 \qquad \Longrightarrow \qquad s = 0 \text{ on } [a, b]$$

Now, for all $1 \le j \le n$, $\overline{p}_j \ge 0$ on $[a, b]$, but $\overline{p}_j \neq 0$ on $[a, b]$, so since $\omega \ge 0$ on $[a, b]$, we have

$$\int_a^b \omega(x)\overline{p}_j(x)\,\mathrm{d}x > 0$$

Also, by elementary considerations

$$\left(\prod_{\substack{k=1\\k\neq j}}^{n}(x_j - x_k)^2\right) > 0$$

Therefore, since

$$\int_a^b \omega(x)\overline{p}_j(x)\,\mathrm{d}x = \omega_j\left(\prod_{\substack{k=1\\k\neq j}}^{n}(x_j - x_k)^2\right)$$

it follows that $\omega_j > 0$ for all $1 \leq j \leq n$. ■

Next we show that theorem 39.1.2 does not hold for some polynomials in Π_{2n}.

Theorem 39.1.4. There exists a polynomial $\overline{p} \in \Pi_{2n}$ such that there exists no collection $\omega_1, \omega_2, \ldots, \omega_n$ such that

$$\int_a^b \omega(x)\overline{p}(x)\,\mathrm{d}x = \sum_{i=1}^{n}\omega_i\overline{p}(x_i)$$

Proof. Define $\overline{p}(x) = \prod_{k=1}^{n}(x - x_k)^2$. Then recall that any weight function on an interval $[a, b]$ must satisfy the condition that for any polynomial s, if $s \geq 0$ on $[a, b]$, then

$$\int_a^b s(x)\omega(x)\,\mathrm{d}x = 0 \qquad \Longrightarrow \qquad s = 0 \text{ on } [a, b]$$

Now, for all $1 \leq j \leq n$, $\overline{p} \geq 0$ on $[a, b]$, but $\overline{p} \neq 0$ on $[a, b]$, so since $\omega \geq 0$ on $[a, b]$, we have

$$\int_a^b \omega(x)\overline{p}(x)\,\mathrm{d}x > 0$$

However, $\overline{p}(x_j) = \prod_{k=1}^{n}(x_j - x_k)^2 = 0$ for all $1 \leq j \leq n$, so $\sum_{i=1}^{n}\omega_i\overline{p}(x_i) = 0$. Therefore,

$$\int_a^b \omega(x)\overline{p}(x)\,\mathrm{d}x > 0 = \sum_{i=1}^{n}\omega_i\overline{p}(x_i)$$

and we are done. ■

Next, we state a converse to theorem 39.1.2.

Theorem 39.1.5. If there exists a collection of pairs $(\omega_1, x_1), (\omega_2, x_2), \ldots, (\omega_n, x_n)$ such that

$$\int_a^b \omega(x)p(x)\,\mathrm{d}x = \sum_{i=1}^{n}\omega_i p(x_i)$$

for all $p \in \Pi_{2n-1}$, then the x_i are the roots of p_n, and the w_i are the solutions to the system of equations 39.1.1.

Proof. See Stoer and Bulirsch. ■

39.1.3 Choosing the right weight function and orthonormal basis

Now, we list some types of intervals and the weight functions plus the systems of orthogonal bases best used with them. Through a combination of using scaling and shift transformations, most intervals you encounter can be transformed into a type of interval that is listed below:

Interval $[a, b]$	Weight function $\omega(x)$	Orthonormal basis (the p_i)
$[-1, +1]$	1	Legendre Polynomials
$[0, \infty)$	e^{-x}	Laguerre Polynomials
$(-D, +D)$	e^{-x^2}	Hermite Polynomials
$[-1, +1]$	$(1 - x^2)^{-\frac{1}{2}}$	Tchebyshev Polynomials