MATH 437 Notes

Jordan Hoffart

February 4, 2026

Contents

1 Lecture 1

1.1 Bisection method

Let $f(x)$ be a continuous function on the interval $[a, b]$.

Proposition 1.1 (Existence of roots)**.** *If $f(a)f(b) < 0$, then there is a point $p \in (a, b)$ such that $f(p) = 0$.*

Proof. If $f(a)f(b) < 0$, then $f(a)$ and $f(b)$ have opposite signs. That is, $f(a) > 0$ and $f(b) < 0$, or $f(a) < 0$ and $f(b) > 0$. By the Intermediate Value Theorem, f attains all possible values between $f(a)$ and $f(b)$. In particular, there is a point $p \in (a, b)$ where $f(p) = 0$. □

The bisection method is an algorithm to find the point p. The algorithm is as follows for the case that $f(a) < 0 < f(b)$.

Algorithm 1 Bisection method

$x_{\text{left}} := a$, $x_{\text{right}} := b$, $n := 0$, $x := (x_{\text{left}} + x_{\text{right}})/2$
while $|x_{\text{left}} - x_{\text{right}}| >$ tol and $n \leq n_{\max}$ **do**
    if $f(x) = 0$ **then**
        return x
    else if $f(x) < 0$ **then**
        $x_{\text{left}} \leftarrow x$
    else
        $x_{\text{right}} \leftarrow x$
    end if
    $n \leftarrow n + 1$
    $x \leftarrow (x_{\text{left}} + x_{\text{right}})/2$
end while
return x

1.2 Fixed point methods

Given a continuous function $g(x)$, suppose we want to solve the equation $x = g(x)$. One possible iterative method is defined by

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

where we provide a starting value x_0. Whether or not this converges to a solution as $n \to \infty$ depends on the properties of g and the starting value x_0.

Theorem 1.2 (Existence of fixed points)**.** *If $g(x) \in [a, b]$ for all $x \in [a, b]$, then g has a fixed point in $[a, b]$.*

Proof. Let $h(x) = g(x) - x$. Then $h(a) \leq 0$, $h(b) \geq 0$ and h is continuous. If $h(a) = 0$, then a is a fixed point of g. If $h(b) = 0$, then b is a fixed point of g. If $h(a)$ and $h(b)$ are both nonzero, then $h(a) < 0 < h(b)$. By the previous proposition, there exists $x_0 \in (a, b)$ such that $h(x_0) = 0$, i.e. $g(x_0) = x_0$. □

Theorem 1.3 (Convergence of fixed-point methods)**.** *Suppose g is differentiable, and there exists k such that $|g'(x)| \leq k < 1$ for all $x \in [a, b]$. Then, g has a unique fixed point, and the iterative method* (1) *converges to this point for any initial value $x_0 \in [a, b]$.*

Proof. By the Mean Value Theorem, for distinct $x, y \in [a, b]$, there exists $z \in [a, b]$ such that

$$g(x) - g(y) = g'(z)(x - y).$$

Therefore, since $|g'(z)| \leq k < 1$, we have that

$$|g(x) - g(y)| \leq k|x - y| < |x - y|$$

for all distinct $x, y \in [a, b]$.

Now, let $x_0 \in [a, b]$, and set $x_{n+1} = g(x_n)$ for all $n \geq 0$. From above, for all $n \geq 0$,

$$|x_{n+2} - x_{n+1}| = |g(x_{n+1}) - g(x_n)| \leq k|x_{n+1} - x_n|.$$

By repeating this, we have

$$|x_{n+2} - x_{n+1}| \leq k^{n+1}|x_1 - x_0|$$

for all n. Therefore, for any $m > n \geq 0$, by writing $m = n + (m - n)$,

$$\begin{aligned} |x_m - x_n| &\leq |x_{n+(m-n)} - x_{n+(m-n-1)}| \\ &\quad + |x_{n+(m-n-1)} - x_{n+(m-n-2)}| + \cdots + |x_{n+1} - x_n| \\ &\leq (k^{m-n-1} + k^{m-n-2} + \cdots + k^n)|x_1 - x_0|. \end{aligned}$$

Since $k < 1$, the terms in the last inequality are the Cauchy tail of the convergent geometric series $\sum_i k^i$. Therefore, $|x_m - x_n| \to 0$ as $m, n \to \infty$, so the sequence $(x_n)_n$ is a Cauchy sequence of real numbers. The sequence therefore must converge to some number p.

Since $g(x_n) = x_{n+1}$ and g is continuous, taking limits of this equation yields $g(p) = p$, so p is a fixed point of g. If q is another fixed point of g, then, from above,

$$|p - q| = |g(p) - g(q)| < |p - q|,$$

which is a contradiction, so p is the only fixed point of g. □

Proposition 1.4 (Non-convergence of fixed point methods)**.** *Let g be continuously differentiable with a fixed point $g(p) = p$. If $|g'(p)| > 1$ and $x_0 \neq p$, then the fixed point iteration $x_{n+1} = g(x_n)$ will not converge to p.*

Proof. Suppose for the sake of contradiction that $x_n \to p$. Since g' is continuous, there is $\delta > 0$ such that $|g'(x)| > 1$ when $x \in (p - \delta, p + \delta)$. By the Mean Value Theorem,

$$x_{n+1} - p = g(x_n) - g(p) = g'(\xi_n)(x_n - p)$$

for some ξ_n between x_n and p. Since $x_n \to p$, the Squeeze Theorem implies that $\xi_n \to p$ as well. Therefore, there is $N > 0$ such that $\xi_n \in (p - \delta, p + \delta)$ for $n > N$, which implies that $|g'(\xi_n)| > 1$ when $n > N$. Therefore,

$$|x_{n+1} - p| = |g'(\xi_n)||x_n - p| > |x_n - p|$$

when $n > N$. In particular,

$$|x_{n+1} - p| = |g'(\xi_n)||x_n - p| > |x_{N+1} - p| =: \varepsilon_0$$

for all $n > N$. Now, if $\varepsilon_0 = 0$, then $x_{N+1} = p$, so $x_{n+1} = p$ for all $n > N$, which contradicts the inequality above. On the other hand, if $\varepsilon_0 > 0$, then, since $x_n \to p$, there is $M > N$ such that when $n > M$, $|x_{n+1} - p| < \varepsilon_0/2$, which also contradicts the inequality above. Therefore, in all cases, we reach a contradiction, so $x_n \not\to p$. □

2 Lecture 2

2.1 Newton's method

Newton's method is a fixed-point method to find the roots of a differentiable function $f(x)$. It is defined by the following algorithm:

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

If we set $g(x) = f(x)/f'(x)$, then the above equation is of the form (1), so that Newton's method is indeed a fixed-point method.

Lemma 2.1. *Suppose f is twice differentiable and $f'(p) \neq 0$. Let $g(x) = x - f(x)/f'(x)$.*

1. *p is a fixed point of g iff $f(p) = 0$.*
2. *If $f(p) = 0$, then $g'(p) = 0$.*

Proof. 1. If p is a fixed point of g, then $p = g(p) = p - f(p)/f'(p)$, so $f(p) = 0$. Conversely, if $f(p) = 0$, then $g(p) = p - f(p)/f'(p) = p$.

2.
$$g'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2}.$$
Since $f(p) = 0$, $g'(p) = 0$.

□

Theorem 2.2 (Convergence of Newton's method)**.** *Suppose f is twice differentiable, has a root at p and $f'(p) \neq 0$. For an initial value x_0 sufficiently close to p, Newton's method converges to p.*

Proof. We let $g(x) = f(x)/f'(x)$. Then, g is continuously differentiable, and, by the previous lemma, p is a fixed point of g and $g'(p) = 0$. Therefore, there exists $\delta > 0$ such that, whenever $|x - p| \leq \delta$, $|g'(x)| \leq 1/2 < 1$. By using Theorem 1.3 with $k = 1/2$, $a = p - \delta$, $b = p + \delta$, we conclude that whenever $x_0 \in [a, b]$, Newton's method converges to p. □

2.2 Quadratic convergence of Newton's method

Definition 2.3 (Order of convergence)**.** Suppose that a sequence $x_n \to p$ as $n \to \infty$. We say that the sequence converges with order $r > 0$ if there is a constant $0 \leq \lambda < \infty$ such that

$$|x_{n+1} - p| \leq \lambda |x_n - p|^r \tag{3}$$

for all n sufficiently large. For $r = 1$, we say the sequence converges linearly, and for $r = 2$, we say the sequence converges quadratically.

Theorem 2.4 (Quadratic convergence of Newton's method)**.** *Let f be a 3-times continuously differentiable function with a root at p and $f'(p) \neq 0$. Suppose that an initial value x_0 is chosen sufficiently close to p so that Newton's method converges to p. Then, the method converges quadratically.*

Proof. We let $g(x) = f(x)/f'(x)$. Then, g is a twice continuously differentiable function. Using Taylor's Theorem around p, for all n, there exists ξ_n between x_n and p such that

$$x_{n+1} = g(x_n) = g(p) + g'(p)(x_n - p) + \frac{g''(\xi_n)}{2}(x_n - p)^2.$$

From Lemma 2.1, $g(p) = p$ and $g'(p) = 0$, so

$$|x_{n+1} - p| = \frac{|g''(\xi_n)|}{2}|x_n - p|^2.$$

There exists $N > 0$ such that $|x_n - p| \leq 1$ for all $n \geq N$. Thus, for all $n \geq N$, ξ_n lies in the interval $[p-1, p+1]$. Since g'' is continuous on the closed and bounded interval $[p-1, p+1]$, we may set

$$\lambda := \max_{\xi \in [p-1, p+1]} \frac{|g''(\xi)|}{2}.$$

Then, we conclude that

$$|x_{n+1} - p| \leq \lambda |x_n - p|^2$$

when $n \geq N$, so Newton's method converges quadratically. □

2.3 Secant method

In Newton's method, we may replace $f'(x)$ by a backward difference approximation

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

Doing so gives us the secant method:

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(x_{n-1})}(x_n - x_{n-1}), \tag{4}$$

where now we must provide 2 initial conditions x_0, x_1.

3 Lecture 3

3.1 Polynomial interpolation

Definition 3.1 (Lagrange polynomials)**.** Given distinct points $x_0, \ldots, x_n$, the ith Lagrange polynomial constructed from these points is

$$L_i(x) := \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}. \tag{5}$$

Example 3.2 (Lagrange polynomial)**.** With $x_0 = 0$, $x_1 = 1$, $x_2 = 2$, we have

$$\begin{cases} L_0(x) = \dfrac{(x-1)(x-2)}{(0-1)(0-2)}, \\ L_1(x) = \dfrac{(x-0)(x-2)}{(1-0)(1-2)}, \\ L_2(x) = \dfrac{(x-0)(x-1)}{(2-0)(2-1)}. \end{cases}$$

Theorem 3.3 (Properties of Lagrange polynomials)**.** *Given distinct points x_0, $\ldots$, x_n, let L_i be the ith Lagrange polynomial constructed from these points. Then L_i is the unique polynomial of degree n such that $L_i(x_j) = 0$ if $i \neq j$ and $L_i(x_i) = 1$. Furthermore, the set $\{L_i : i = 0, \ldots, n\}$ of Lagrange polynomials is a basis for the space P_n of polynomials of degree at most n.*

Proof. From (5), we see that L_i is a product of n monomial terms $(x - x_j)/(x_i - x_j)$, so it is a polynomial of degree n. Since each monomial term evaluates to 1 at x_i, $L_i(x_i) = 1$. Since at least one monomial term vanishes at x_j, $L_i(x_j) = 0$ when $j \neq i$. Now, suppose that p is another polynomial of degree n such that $p(x_i) = 1$ and $p(x_j) = 0$ when $j \neq i$. From the Fundamental Theorem of Algebra, since p has n roots at the x_j, p can be factored as

$$p(x) = a \prod_{j \neq i} (x - x_j)$$

for some $a \in \mathbb{R}$. Since $p(x_i) = 1$, inserting this into the equation above and solving for a gives

$$a = \prod_{j \neq i} \frac{1}{x_i - x_j}$$

Therefore,

$$p(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} = L_i(x),$$

so L_i is the unique polynomial of degree n with the above properties.

Now, we show that the set of Lagrange polynomials forms a basis of P_n. Suppose that

$$\sum_{j=0}^{n} c_j L_j(x) = 0$$

for some coefficients $c_i \in \mathbb{R}$. Evaluating at $x = x_i$ and using the properties of the Lagrange polynomials gives $c_i = 0$ for all i. Therefore, the L_i are linearly independent. Now, let $p \in P_n$ and set

$$q(x) = \sum_{j=0}^{n} p(x_j) L_j(x).$$

Then $q(x_i) = p(x_i)$ for all i by construction, and $q \in P_n$. Set $r = p - q$. Then $r \in P_n$ and r vanishes at $n+1$ distinct points $x_0, \ldots, x_n$. Therefore, by the Fundamental Theorem of Algebra, $r = 0$, so $p = q$. Thus, the L_i span all of P_n, so they are a basis of P_n. □

Corollary 3.4 (Lagrange basis expansion)**.** *For the Lagrange basis $\{L_i : i = 0, \ldots, n\}$ of P_n constructed from distinct points x_0, …, x_n, any polynomial $p \in P_n$ has the following basis expansion:*

$$p(x) = \sum_{j=0}^{n} p(x_j) L_j(x). \tag{6}$$

Theorem 3.5 (Polynomial interpolation)**.** *Given finitely many samples $f(x_0)$, $f(x_1)$, …, $f(x_n)$ of a function $f(x)$ at distinct points x_0, …, x_n, there is a unique polynomial $p_n(x)$ of degree n passing through the points:*

$$p_n(x_i) = f(x_i) \text{ for all } i. \tag{7}$$

Proof. We use the Lagrange basis polynomials L_i defined by the points x_i and set

$$p_n(x) = \sum_{j=0}^{n} f(x_j) L_j(x). \tag{8}$$

From the results above, $p_n(x_i) = f(x_i)$ for all i, $p_n \in P_n$, and p_n is the only polynomial of degree n with these properties. □

Definition 3.6 (Interpolating polynomial)**.** For a function f defined at distinct points x_0, …, x_n, we call p_n defined by (8) the (Lagrange form of) the interpolating polynomial of f at the points x_i.

Lemma 3.7 (Generalization of Rolle's Theorem)**.** *Let f be an $n+1$ times differentiable function such that $f(x_0) = f(x_1) = \cdots = f(x_{n+1})$ at $n+2$ distinct points $x_0 < x_1 < \cdots < x_{n+1}$. Then, there is a point $\xi \in (x_0, x_{n+1})$ such that $f^{(n+1)}(\xi) = 0$.*

Proof. Applying Rolle's Theorem to each subinterval (x_i, x_{i+1}) gives $n+1$ distinct points $x_0^1 < x_1^1 < \cdots < x_n^1$ between x_0 and x_{n+1} such that all $f'(x_i^1) = 0$. We apply Rolle's theorem again now to the subintervals (x_i^1, x_{i+1}^1) to get n distinct points $x_0^2 < x_1^2 < \cdots < x_{n-1}^2$ between x_0 and x_{n+1} such that all $f''(x_i^2) = 0$. Proceeding inductively, we eventually conclude that there are two distinct points $x_0^n < x_1^n$ between x_0 and x_{n+1} such that $f^{(n)}(x_i^n) = 0$, and then we conclude that there is a single point $\xi \in (x_0, x_{n+1})$ such that $f^{(n+1)}(\xi) = 0$. □

Lemma 3.8 (Polynomial derivatives)**.** *For any polynomial p of degree n, $p^{(n+1)}(x) = 0$. For any polynomial q of degree $n+1$ of the form*

$$q(x) = \prod_{j=0}^{n}(x - x_j),$$

$q^{(n+1)}(x) = (n+1)!$.

Proof. Expanding p in its monomial basis gives $p(x) = \sum_{i=0}^{n} c_i x^i$ for some coefficients c_i. Taking $n+1$ derivatives of the right-hand side makes all the monomial terms vanish, so we get the first result.

For the second result, expanding the product gives us a polynomial of the form

$$q(x) = x^{n+1} + r(x)$$

where r is a polynomial of degree n. Taking $n+1$ derivatives of the right hand side and using the previous result finishes the proof. □

Theorem 3.9 (Error of polynomial interpolation)**.** *Let f be $n+1$ times differentiable and let p be the degree n Lagrange interpolating polynomial of f at the distinct points $x_0 < x_1 < \cdots < x_n$. Then, for all x, there is a point $\xi_x \in (\min(x, x_0), \max(x, x_n))$ such that*

$$f(x) = p(x) + \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\prod_{j=0}^{n}(x - x_j). \tag{9}$$

Proof. Fix $x \notin \{x_0, \ldots, x_n\}$ and let

$$g(t) = f(t) - p(t) - (f(x) - p(x))\prod_{j=0}^{n}\frac{t - x_j}{x - x_j}.$$

We observe that $g(x) = 0$ and $g(x_j) = 0$ for $j = 0, \ldots, n$. Therefore, by the previous lemmas, there is ξ_x such that

$$g^{(n+1)}(\xi_x) = f^{(n+1)}(\xi_x) - \frac{f(x) - p(x)}{\prod_{j=0}^{n} x - x_j}(n+1)! = 0.$$

Rearranging this equation finishes the proof. □

Example 3.10 (Error estimate)**.** We let $f(x) = \sin(x)$ and we estimate the error between f and the linear Lagrange interpolant at the points $x = 0$ and $x = \pi/2$:

$$p(x) = x.$$

Using the previous theorem,

$$\sin(x) = x - \frac{\sin(\xi_x)}{2}(x - 0)(x - \pi/2).$$

Therefore, for $0 \leq x \leq \pi/2$,

$$|\sin(x) - x| \leq \frac{1}{2} \max_{\xi \in [0,\pi/2]} |\sin(\xi)||x(x - \pi/2)| = \frac{1}{2} x(\pi/2 - x),$$

and thus

$$\max_{x \in [0,\pi/2]} |\sin(x) - x| \leq \max_{x \in \pi/2} \frac{1}{2} x(\pi/2 - x) = \frac{\pi^2}{32}.$$

□

4 Lecture 4

4.1 Divided differences

Definition 4.1 (Newton basis polynomials)**.** Given distinct points $x_0, x_2, \ldots, x_{n-1}$, the Newton basis polynomials are

$$\begin{cases} N_0(x) := 1, \\ N_j(x) := \prod_{i=0}^{j-1}(x - x_i), \quad j = 1, \ldots, n. \end{cases} \tag{10}$$

Proposition 4.2. *The Newton basis polynomials are a basis of the space P_n of all polynomials of degree at most n.*

Proof. Suppose that

$$\sum_{j=0}^{n} c_j N_j(x) = 0$$

for some coefficients c_j. Evaluating at $x = x_0$ makes all the terms except $c_0 N_0(x) = c_0$ vanish, so $c_0 = 0$. Then, evaluating at $x = x_1$ makes all the remaining terms vanish except $c_1 n_1(x) = c_1(x_1 - x_2)$, so $c_1 = 0$. Repeating this argument for the remaining terms, we conclude that all $c_i = 0$. Therefore, the N_j are linearly independent. Since $\dim P_n = n + 1$ and there are $n + 1$ N_j, we conclude that the set is a basis of P_n. □

Remark 4.3 (Computational cost)**.** Consider the following computations with respect to the Lagrange basis versus the Newton basis:

$$\begin{cases} p_1(x) := \sum_{j=0}^{n} c_j L_j(x), \\ p_2(x) := \sum_{j=0}^{n} c_j N_j(x). \end{cases}$$

To evaluate $p_1(x)$ using the Lagrange basis, one needs $(n + 1)(n + 2)$ multiplications. Indeed, each L_i requires $n + 1$ multiplications to evaluate at a point, assuming one precomputes and stores the denominator $(\prod_{j \neq i}(x_i - x_j))^{-1}$. Multiplying by the coefficient c_j costs another multiplication, so there are $n + 2$ multiplications required to evaluate each term in $p_1(x)$. Since there are $n + 1$ terms, we conclude that $p_1(x)$ requires $(n + 1)(n + 2)$ multiplications in total.

On the other hand, $p_2(x)$ only requires $(n + 1)(n + 2)/2$ multiplications in total. Indeed, the Newton basis requires j multiplications to evaluate $N_j(x)$. Multiplying each N_j by c_j adds 1 more multiplication. Thus, the total number of multiplications required is

$$\sum_{j=0}^{n}(j + 1) = (n + 1)(n + 2)/2.$$

Therefore, using the Newton basis to evaluate a polynomial of degree n requires half of the computational cost as with the Lagrange basis, which can provide significant increase in performance in practice. □

From the previous remark, it is computationally advantageous to use the Newton basis over the Lagrange basis. Given the values $f(x_0), \ldots, f(x_n)$ of a function at distinct points $x_0, \ldots, x_n$, the interpolating polynomial in terms of the Lagrange basis is given by (8). How do we express this polynomial in terms of the Newton basis? That is, how do we compute the coefficients c_j such that

$$p(x) = \sum_{j=0}^{n} c_j N_j(x), \qquad p(x_i) = f(x_i) \text{ for all } i?$$

The method of divided differences answers this question. We first illustrate with an example.

Example 4.4 (Divided differences)**.** Let's construct the quadratic interpolant using points x_0, x_1, x_2 and values $f(x_0)$, $f(x_1)$, $f(x_2)$ in terms of the Newton basis. We have

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1), \qquad p(x_i) = f(x_i) \text{ for all } i.$$

At $x = x_0$, this implies that $f(x_0) = c_0$. At $x = x_1$, we have the equation

$$f(x_1) = f(x_0) + c_1(x_1 - x_0),$$

so

$$c_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

For $i \neq j$, let

$$f[x_i, x_j] := \frac{f(x_j) - f(x_i)}{x_j - x_i}.$$

Then $c_1 = f[x_0, x_1]$. At $x = x_2$, we have the equation

$$f(x_2) = f(x_0) + f[x_0, x_1](x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1).$$

Now, we subtract $f(x_1)$ from both sides and divide by $x_2 - x_1$ to get

$$\begin{aligned} f[x_1, x_2] &= \frac{f(x_0) - f(x_1)}{x_2 - x_1} + f[x_0, x_1]\frac{x_2 - x_0}{x_2 - x_1} + c_2(x_2 - x_0) \\ &= -f[x_0, x_1]\frac{x_1 - x_0}{x_2 - x_1} + f[x_0, x_1]\frac{x_2 - x_0}{x_2 - x_1} + c_2(x_2 - x_0) \\ &= f[x_0, x_1] + c_2(x_2 - x_0). \end{aligned}$$

Therefore,

$$c_2 = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}.$$

Motivated by this, for distinct i, j, k, we set

$$f[x_i, x_j, x_k] := \frac{f[x_j, x_k] - f[x_i, x_j]}{x_k - x_i},$$

so that $c_2 = f[x_0, x_1, x_2]$. In conclusion, the coefficients with respect to the Newton basis are

$$p(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1).$$

□

This example generalizes to degree n as follows.

Theorem 4.5 (Newton coefficients)**.** *For distinct points $x_0, x_1, \ldots, x_n$ and a function f, we recursively define*

$$f[x_i] := f(x_i), \qquad 0 \le i \le n, \tag{11}$$

and

$$f[x_i, \ldots, x_{i+k}] := \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i},$$
$$0 \le i \le n-1,\ 1 \le k \le n-i. \tag{12}$$

Then, for the interpolating polynomial p to f at the points x_i, its coefficients in Newton form are

$$p(x) = \sum_{j=0}^{n} f[x_0, \ldots, x_j] N_j(x). \tag{13}$$

Example 4.6 (Newton coefficients)**.** For $n = 3$, we have

$$\begin{aligned}
f[x_0, x_1, x_2, x_3] &= \frac{1}{x_3 - x_0}(f[x_1, x_2, x_3] - f[x_0, x_1, x_2]),\\
f[x_0, x_1, x_2] &= \frac{1}{x_2 - x_0}(f[x_1, x_2] - f[x_0, x_1]),\\
f[x_1, x_2, x_3] &= \frac{1}{x_3 - x_1}(f[x_2, x_3] - f[x_1, x_2]),\\
f[x_0, x_1] &= \frac{1}{x_1 - x_0}(f[x_1] - f[x_0]),\\
f[x_1, x_2] &= \frac{1}{x_2 - x_1}(f[x_2] - f[x_1]),\\
f[x_2, x_3] &= \frac{1}{x_3 - x_2}(f[x_3] - f[x_2]),
\end{aligned}$$

so

$$\begin{aligned}
p(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)\\
+ f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2).
\end{aligned}$$

□

The following table let's us easily build up the divided difference coefficients starting from the function values $f(x_i)$. To be explicit, we give the table for $n = 3$.

	0	1	2	3
0	$f(x_0)$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	$f[x_0, x_1, x_2, x_3]$
1	$f(x_1)$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	
2	$f(x_2)$	$f[x_2, x_3]$		
3	$f(x_3)$			

The entries in the table are computed one column at a time from left to right. In column 1 onward, an entry in the (i, j) position of the table is computed from by using the entries in positions $(i, j-1)$ and $(i+1, j-1)$.

4.2 Hermite interpolation

Suppose that we want to find a polynomial $p(x)$ of minimal degree that interpolates a function f and its derivative f' at a finite set of points x_i:

$$p(x_i) = f(x_i), \qquad p'(x_i) = f'(x_i) \text{ for all } i.$$

Such a polynomial is called a Hermite interpolating polynomial. It can be computed using the method of divided differences. We first explain with an example.

Example 4.7 (Hermite polynomial)**.** We desire a polynomial $p(x)$ that interpolates $f(x_0)$ and $f'(x_0)$. Since we have 2 constraints, we consider a polynomial of degree 1:

$$p(x) = c_0 + c_1 x.$$

We now compute the coefficients c_0, c_1 that satisfy the constraints. First, $p'(x_0) = c_1$, so we set $c_1 = f'(x_0)$. Then, $p(x_0) = c_0 + f'(x_0)x_0$, so we set

$$c_0 = f(x_0) - f'(x_0)x_0.$$

Thus, after substituting and rearranging,

$$p(x) = f(x_0) + f'(x_0)(x - x_0).$$

□

Notice that, in the previous example, the interpolating polynomial is in Newton form with coefficients $f(x_0)$ and $f'(x_0)$:

$$p(x) = f(x_0)N_0(x) + f'(x_0)N_1(x).$$

In fact, by modifying the divided difference table, we can use the divided difference algorithm to compute the coefficients of the Hermite polynomial as follows.

For the example above, we set $z_0 = z_1 = x_0$. Then, we construct the divided difference table using the $f(z_i)$, values:

$$\begin{array}{cccc} & & 0 & 1 \\ 0 & x_0 & f(z_0) & f[z_0, z_1] \\ 1 & x_0 & f(z_1) & \end{array}$$

However, since $z_0 = z_1 = x_0$, $f(z_0) = f(z_1)$, so the difference $f[z_0, z_1]$ is undefined. In this case, we use the derivative $f'(z_0)$:

$$f[z_i, z_j] := f'(z_i) \quad \text{if } z_i = z_j. \tag{14}$$

From another viewpoint, we recall that the divided difference is formally defined as

$$f[z_i, z_j] = \frac{f(z_j) - f(z_i)}{z_j - z_i},$$

so that in the limit $z_j \to z_i$, we recover the derivative of f at z_i:

$$\lim_{z_j \to z_i} f[z_i, z_j] = f'(z_i).$$

In the general case with $n+1$ points $x_0, \ldots, x_n$ where we wish to interpolate the point values $f(x_i)$ and the derivatives $f'(x_i)$, the above procedure generalizes to the following table:

$$\begin{array}{cccccc} & & 0 & 1 & \cdots & 2n+2 \\ 0 & x_0 & f(z_0) & f[z_0, z_1] & \cdots & f[z_0, \ldots, z_{2n+1}] \\ 1 & x_0 & f(z_1) & f[z_1, z_2] & \cdots & \\ 2 & x_1 & f(z_2) & f[z_2, z_3] & \cdots & \\ 3 & x_1 & f(z_3) & f[z_3, z_4] & \cdots & \\ \vdots & \vdots & \vdots & \vdots & \cdots & \\ 2n & x_n & f(z_{2n}) & f[z_{2n}, z_{2n+1}] & & \\ 2n+1 & x_n & f(z_{2n+1}) & & & \end{array}$$

where we use (14) whenever $z_i = z_j$. The Hermite interpolating polynomial in Newton form is then

$$p(x) = \sum_{j=0}^{2n+1} f[z_0, \ldots, z_j] N_j(x). \tag{15}$$

We remark that the Hermite interpolating polynomial belongs to P_{2n+1} to satisfy the $2n+2$ constraints.

We can also compute the Hermite interpolating polynomial using the Lagrange basis polynomials L_i. The idea is to write the polynomial as

$$p(x) = \sum_{j=0}^{n} f(x_j) H_j(x) + \sum_{j=0}^{n} f'(x_j) \widehat{H}_j(x), \tag{16}$$

where H_j and $\widehat{H}_j$ are polynomials in P_{2n+1} such that

$$\begin{aligned} H_j(x_i) &= \delta_{ij}, \quad H_j'(x_i) = 0, \\ \widehat{H}_j(x_i) &= 0, \qquad \widehat{H}_j'(x_i) = \delta_{ij}, \end{aligned} \tag{17}$$

and δ_{ij} is the Kronecker delta.

Lemma 4.8. *If a polynomial $p \in P_n$ satisfies $p(x_0) = 0$ and $p'(x_0) = 0$, then there is a polynomial $q \in P_{n-2}$ such that*

$$p(x) = (x - x_0)^2 q(x).$$

Proof. From the Fundamental Theorem of Algebra, $p(x) = (x - x_0)r(x)$ for some $r \in P_{n-1}$. Then, $p'(x) = r(x) + (x - x_0)r'(x)$, so $p'(x_0) = r(x_0) = 0$. Thus, by the Fundamental Theorem of Algebra again, there is $q \in P_{n-2}$ such that $r(x) = (x - x_0)q(x)$. Therefore, $p(x) = (x - x_0)^2 q(x)$ as desired. □

Theorem 4.9 (Lagrange form of Hermite interpolating polynomial)**.** *The Lagrange form of the Hermite interpolating polynomial to the function f interpolating the values $f(x_0),\ldots,f(x_n)$ and derivatives $f'(x_0),\ldots,f'(x_n)$ is* (16) *with coefficients*

$$\begin{cases} H_j(x) = L_j(x)^2(1 - 2L_j'(x_j)(x - x_j)), \\ \widehat{H}_j(x) = L_j(x)^2(x - x_j), \end{cases} \tag{18}$$

where the L_j are the Lagrange basis polynomials constructed from the points $x_0,\ldots,x_n$.

Proof. Following the idea in (17), we seek $H_j \in P_{2n+1}$ with the aforementioned properties. We show that necessarily the H_j must be of the form (18).

From the Fundamental Theorem of Algebra, H_j must be of the form

$$H_j(x) = q_j(x) \prod_{i \neq j} (x - x_i)$$

for some $q_j \in P_{n+1}$. By multiplying and dividing by $\prod_{i \neq j}(x_j - x_i)$ and redefining q_j, H_j is of the form

$$H_j(x) = q_j(x) L_j(x).$$

We observe that $L_j'(x_i) \neq 0$ for all $i \neq j$. Therefore,

$$H_j'(x_i) = q_j'(x_i)L_j(x_i) + q_j(x_i)L_j'(x_i) = q_j(x_i)L_j'(x_i) = 0$$

for all $i \neq j$, so $q_j(x_i) = 0$ for all $i \neq j$. By repeating the previous argument, we conclude that q_j must also be of the form

$$q_j(x) = L_j(x) r_j(x)$$

for some $r_j \in P_1$. Thus,

$$H_j(x) = L_j^2(x) r_j(x).$$

We desire also that $H_j(x_j) = 1$, so $r_j(x_j) = 1$. Thus, from the Fundamental Theorem of Algebra, there is a constant c_j such that

$$r_j(x) = 1 + c_j(x - x_j).$$

We also require that $H_j'(x_j) = 0$, so

$$H_j'(x_j) = 2L_j(x_j)L_j'(x_j)r_j(x_j) + L_j^2(x_j)r_j'(x_j) = 2L_j'(x_j) + c_j = 0.$$

Thus, $c_j = -2L_j'(x_j)$, hence

$$H_j(x) = L_j^2(x)(1 - 2L_j'(x_j)(x - x_j))$$

as desired.

Now, we do a similar procedure for $\widehat{H}_j$. Since $\widehat{H}_j(x_i) = 0$ for all i, we must have that

$$\widehat{H}_j(x) = L_j(x)(x - x_j)s_j(x)$$

for some $s_j \in P_n$. Since $\widehat{H}_j'(x_i) = 0$ for $i \neq j$, we have that

$$\begin{aligned}\widehat{H}_j'(x_i) &= L_j'(x_i)(x_i - x_j)s_j(x_i) + L_j(x_i)s_j(x_i) + L_j(x_i)(x_i - x_j)s_j'(x_i) \\ &= L_j'(x_i)(x_i - x_j)s_j(x_i) \\ &= 0.\end{aligned}$$

Thus, $s_j(x_i) = 0$ for all $i \neq j$. Hence, there is a constant a_j such that

$$\widehat{H}_j(x) = a_j L_j^2(x)(x - x_j).$$

Thus,

$$\widehat{H}_j'(x) = 2a_j L_j'(x)(x - x_j) + a_j L_j^2(x).$$

Since $\widehat{H}_j'(x_j) = 1$, this implies that $a_j = 1$, which completes the proof. □

5 Lecture 5

5.1 Cubic splines

Definition 5.1 (Cubic spline)**.** Let $a = x_0 < x_1 < \cdots < x_n = b$. The cubic spline interpolating a function f at the points x_j is the piecewise cubic polynomial S with the following properties:

1. On each subinterval $[x_j, x_{j+1}]$, S is a cubic polynomial. We denote the restriction of S to this subinterval by S_j.

2. The pieces $\{S_j\}_{j=0}^{n-1}$ interpolate f at the nodes:

$$S_j(x_j) = f(x_j) \text{ and } S_j(x_{j+1}) = f(x_{j+1}) \text{ for all } j. \tag{19}$$

   This implies that S is continuous on the entire interval $[a, b]$, since $S_j(x_{j+1}) = S_{j+1}(x_{j+1})$ for all j.

3. The first and second derivatives of S are continuous on $[a, b]$, i.e.

$$\begin{cases} S_j'(x_{j+1}) = S_{j+1}'(x_{j+1}), \\ S_j''(x_{j+1}) = S_{j+1}''(x_{j+1}) \end{cases} \tag{20}$$

   for all $j = 0, \ldots, n-2$.

4. S has natural boundary conditions:

$$S''(a) = S''(b) = 0. \tag{21}$$

Example 5.2 (Cubic spline)**.** Let's construct the cubic spline passing through the points (x_0, y_0), (x_1, y_1), and (x_2, y_2) where $x_0 < x_1 < x_2$. Since there are 3 points, $n = 2$, so there are two pieces S_0 and S_1. Since S_0'' and S_1'' are degree 1 polynomials, let's expand them in terms of the Lagrange basis on their respective subintervals:

$$\begin{cases} S_0''(x) = S_0''(x_0)\dfrac{x - x_1}{x_0 - x_1} + S_0''(x_1)\dfrac{x - x_0}{x_1 - x_0}, \\ S_1''(x) = S_1''(x_1)\dfrac{x - x_2}{x_1 - x_2} + S_1''(x_2)\dfrac{x - x_1}{x_2 - x_1}. \end{cases}$$

Now, let's integrate both functions on their respective subintervals starting from x_1:

$$\begin{cases} S_0'(x) = S_0'(x_1) + S_0''(x_0)\dfrac{(x - x_1)^2}{2(x_0 - x_1)} + S_0''(x_1)\dfrac{(x - x_0)^2 - (x_1 - x_0)^2}{2(x_1 - x_0)}, \\ S_1'(x) = S_1'(x_1) + S_1''(x_1)\dfrac{(x - x_2)^2 - (x_1 - x_2)^2}{2(x_1 - x_2)} + S_1''(x_2)\dfrac{(x - x_1)^2}{2(x_2 - x_1)}. \end{cases}$$

Then, we integrate again, also from x_1:

$$\begin{cases} S_0(x) = S_0(x_1) + S_0'(x_1)(x-x_1) + S_0''(x_0)\dfrac{(x-x_1)^3}{6(x_0-x_1)} \\ \qquad\qquad + S_0''(x_1)\dfrac{(x-x_0)^3 - (x_1-x_0)^3 - 3(x_1-x_0)^2(x-x_1)}{6(x_1-x_0)}, \\ S_1(x) = S_1(x_1) + S_1'(x_1)(x-x_1) \\ \qquad\qquad + S_1''(x_1)\dfrac{(x-x_2)^3 - (x_1-x_2)^3 - 3(x_1-x_2)^2(x-x_1)}{6(x_1-x_2)} \\ \qquad\qquad + S_1''(x_2)\dfrac{(x-x_1)^3}{6(x_2-x_1)}. \end{cases}$$

Now, we require $S_0(x_0) = y_0$, $S_0(x_1) = S_1(x_1) = y_1$, and $S_1(x_2) = y_2$. We also require $S_0'(x_1) = S_1'(x_1)$, $S_0''(x_1) = S_1''(x_1)$, $S_0''(x_0) = 0$, and $S_1''(x_2) = 0$. Inserting this above and evaluating the first equation at x_0 and the second at x_2 gives

$$\begin{cases} y_0 = y_1 + S_0'(x_1)(x_0-x_1) + S_0''(x_1)\dfrac{(x_1-x_0)^2}{2}, \\ y_2 = y_1 + S_0'(x_1)(x_2-x_1) + S_0''(x_1)\dfrac{(x_1-x_2)^2}{2}. \end{cases}$$

This is a square linear system in the unknowns $S_0'(x_1)$ and $S_0''(x_1)$. Subtracting each equation by y_1 and dividing by the respective coefficient in front of $S_0'(x_1)$ gives

$$\begin{cases} [y_0, y_1] = S_0'(x_1) + S_0''(x_1)\dfrac{(x_1-x_0)}{2}, \\ [y_1, y_2] = S_0'(x_1) + S_0''(x_1)\dfrac{(x_2-x_1)}{2}, \end{cases}$$

where $[y_i, y_j] := (y_i - y_j)/(x_i - x_j)$ is the divided difference of y_i and y_j. Subtracting the first equation from the second and dividing by $(x_2 - x_0)/2$ gives

$$S_0''(x_1) = 2[y_0, y_1, y_2] = S_1''(x_1),$$

where $[y_0, y_1, y_2] := ([y_1, y_2] - [y_0, y_1])/(x_2 - x_0)$ is the divided difference of the y_i's. If we now multiply the first equation by $x_2 - x_1$, the second by $x_1 - x_0$, and subtract the second from the first, we get

$$S_0'(x_1) = \frac{[y_0, y_1](x_2-x_1) - [y_1, y_2](x_1-x_0)}{x_2 - x_0} = S_1'(x_1).$$

We have now found the unknowns $S_0'(x_1)$ and $S_0''(x_1)$ in terms of the data (x_i, y_i). We can use the Taylor expansions of the polynomials S_j from x_1 to

give their formulas in terms of these coefficients:

$$\begin{cases} S_0(x) = y_1 + \dfrac{[y_0,y_1](x_2-x_1)-[y_1,y_2](x_1-x_0)}{x_2-x_0}(x-x_1) \\ \qquad\qquad + [y_0,y_1,y_2](x-x_1)^2 + \dfrac{S_0^{(3)}(x_1)}{6}(x-x_1)^3, \\ S_1(x) = y_1 + \dfrac{[y_0,y_1](x_2-x_1)-[y_1,y_2](x_1-x_0)}{x_2-x_0}(x-x_1) \\ \qquad\qquad + [y_0,y_1,y_2](x-x_1)^2 + \dfrac{S_1^{(3)}(x_1)}{6}(x-x_1)^3, \end{cases}$$

To compute $S_0^{(3)}(x_1)$ and $S_1^{(3)}(x_1)$, we return to the very first equations in the example and take derivatives, applying the boundary conditions and the previous results:

$$\begin{cases} S_0^{(3)}(x_1) = \dfrac{S_0''(x_1)}{x_1-x_0} = \dfrac{2}{x_1-x_0}[y_0,y_1,y_2], \\ S_1^{(3)}(x_1) = \dfrac{S_1''(x_1)}{x_1-x_2} = \dfrac{2}{x_1-x_2}[y_0,y_1,y_2]. \end{cases}$$

In conclusion, the general form for a piecewise cubic spline passing through 3 points is

$$\begin{cases} S_0(x) = y_1 + \dfrac{[y_0,y_1](x_2-x_1)-[y_1,y_2](x_1-x_0)}{x_2-x_0}(x-x_1) \\ \qquad\qquad + [y_0,y_1,y_2](x-x_1)^2\left(1-\dfrac{1}{3}\dfrac{x_1-x}{x_1-x_0}\right), \\ S_1(x) = y_1 + \dfrac{[y_0,y_1](x_2-x_1)-[y_1,y_2](x_1-x_0)}{x_2-x_0}(x-x_1) \\ \qquad\qquad + [y_0,y_1,y_2](x-x_1)^2\left(1-\dfrac{1}{3}\dfrac{x-x_1}{x_2-x_1}\right), \end{cases}$$

where S_0 is defined on $[x_0,x_1]$ and $S_1$is defined on $[x_1,x_2]$. □

Remark 5.3. The procedure in the previous example does not easily generalize to more nodes. In this case, a more straightforward procedure involves simply expanding each piece S_j in some basis and setting up a large linear system imposed by the constraints. Motivated by the previous example where the final answer was expressed in terms of powers of $x-x_1$, we propose to use the following basis expansion for each S_j on $[x_j,x_{j+1}]$:

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

In fact, this is just a Taylor expansion of each S_j about x_j, so

$$\begin{cases} a_j := S_j(x_j), \\ b_j := S_j'(x_j), \\ c_j := \dfrac{S_j''(x_j)}{2}, \\ d_j := \dfrac{S_j^{(3)}(x_j)}{6}. \end{cases}$$

The compatibility conditions then become

$$\begin{cases} y_{j+1} = S_{j+1}(x_{j+1}) = S_j(x_{j+1}) \\ \qquad = a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3, \\ b_{j+1} = S_{j+1}'(x_{j+1}) = S_j'(x_{j+1}) \\ \qquad = b_j + 2c_j(x_{j+1} - x_j) + 3d_j(x_{j+1} - x_j)^2, \\ 2c_{j+1} = S_{j+1}''(x_{j+1}) = S_j''(x_{j+1}) = 2c_j + 6d_j(x_{j+1} - x_j), \\ a_j = S_j(x_j) = y_j, \\ c_0 = S_0''(x_0) = 0, \\ S_{n-1}''(x_n) = 0 = 2c_{n-1} + 6d_{n-1}(x_n - x_{n-1}). \end{cases}$$

These equations lead to a large linear system that is no longer easily solvable by hand, but the system is simple to set up for a particular n and given values (x_j, y_j). □

5.2 Numerical differentiation

We recall the limit definition of a derivative:

$$f'(x_0) := \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}. \tag{22}$$

We expect that, for h small enough, the divided difference on the right-hand side is a good approximation to $f'(x_0)$.

Theorem 5.4 (First-order numerical differentiation)**.** *If f is twice differentiable on an interval $(x_0 - \delta, x_0 + \delta)$ and f'' is bounded on that interval, then for all h small enough*

$$\left| f'(x_0) - \frac{f(x_0 + h) - f(x_0)}{h} \right| \leq \sup_{x \in (x_0 - \delta, x_0 + \delta)} \left| \frac{f''(\xi)}{2} \right| h. \tag{23}$$

Proof. We perform a Taylor expansion near x_0:

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2} f''(\xi_h)$$

where ξ_h is between x_0 and $x_0 + h$ and h is small enough so $x_0 + h \in (x_0 - \delta, x_0 + \delta)$. Rearranging, taking absolute values, and taking a supremum over $\xi_h \in (x_0 - \delta, x_0 + \delta)$ finishes the proof. □

Thus, the numerical differentiation formula above is only first-order accurate. To achieve higher-order accurate formulas, we can use more points along with Taylor expansions or Lagrange expansions.

Example 5.5 (Lagrange expansion)**.** Consider 3 points x_0, $x_1 = x_0 + h$, $x_2 = x_0 + 2h$. The Lagrange polynomial interpolating a smooth function f through these points is

$$p(x) = f(x_0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f(x_2)\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}.$$

From Theorem 3.9, for all $x \in (x_0, x_2)$, there is a point $\xi_x \in (x_0, x_2)$ such that

$$f(x) = p(x) + \frac{f^{(3)}(\xi_x)}{6}(x-x_0)(x-x_1)(x-x_2).$$

Since f and p are smooth, we deduce that the mapping $x \mapsto f^{(3)}(\xi_x)$ is smooth on (x_0, x_1) and (x_1, x_2). Then, we may differentiate both sides to get

$$f'(x) = p'(x) + \frac{\mathrm{d}}{\mathrm{d}x}\frac{f^{(3)}(\xi_x)}{6}(x-x_0)(x-x_1)(x-x_2) + \frac{f^{(3)}(\xi_x)}{6}\frac{\mathrm{d}}{\mathrm{d}x}\left\{(x-x_0)(x-x_1)(x-x_2)\right\}.$$

Evaluating at $x = x_0$ and substituting for h gives

$$f'(x_0) = f(x_0)\frac{-3h}{(-h)(-2h)} + f(x_1)\frac{-2h}{h(-h)} + f(x_2)\frac{-h}{(2h)h} + \frac{f^{(3)}(\xi_0)}{6}(-h)(-2h).$$

Rearranging, we arrive at a differentiation formula that is second-order accurate:

$$f'(x_0) = \frac{-3}{2h}f(x_0) + \frac{2}{h}f(x_0+h) - \frac{1}{2h}f(x_0+2h) + \frac{f^{(3)}(\xi_0)}{3}h^2. \tag{24}$$

If we evaluate at x_1 instead, we get another formula:

$$f'(x_1) = \frac{f(x_1+h) - f(x_1-h)}{2h} - \frac{f^{(3)}(\xi_1)}{6}h^2. \tag{25}$$

□

We can also prove that the differentiation formulas are second-order accurate by using Taylor's theorem. We prove one of them and leave the other as an exercise to the reader.

Theorem 5.6 (Second-order differentiation formula)**.** *The differentiation formula* (25) *is second-order accurate for sufficiently smooth functions.*

Proof. We take $h > 0$ and expand in Taylor series about x_1:

$$\begin{cases} f(x_1+h) = f(x_1) + hf'(x_1) + \dfrac{h^2}{2}f''(x_1) + \dfrac{f^{(3)}(\xi_{h,+})}{6}h^3, \\ f(x_1-h) = f(x_1) - hf'(x_1) + \dfrac{h^2}{2}f''(x_1) - \dfrac{f^{(3)}(\xi_{h,-})}{6}h^3, \end{cases}$$

where $\xi_{h,+} \in (x_1, x_1+h)$ and $\xi_{h,-} \in (x_1-h, x_1)$. We subtract the two equations and rearrange:

$$\frac{f(x_1+h) - f(x_1-h)}{2h} - f'(x_1) = \frac{f^{(3)}(\xi_{h,+}) + f^{(3)}(\xi_{h,-})}{6}h^2.$$

Now, assuming that f is smooth enough where its 3rd derivative is bounded near x_1, by taking absolute values, we can bound the coefficient in front of the h^2 term on the right-hand side by a constant independent of h. This completes the proof. □

To summarize, using Lagrange polynomials is an effective way to discover differentiation formulas, and using Taylor's Theorem is an efficient way to prove that the methods are indeed accurate, provided that the functions involved are sufficiently smooth.

5.3 Richardson extrapolation

Sometimes, one can use a differentiation formula evaluated at multiple values of h to improve accuracy. This process is called Richardson extrapolation.

For example, consider the first-order formula to approximate $f'(x_0)$:

$$f'(x_0) = \frac{f(x_0+h) - f(x_0)}{h} + \mathcal{O}(h).$$

Denote the left-hand side by M and the differentiation formula on the right-hand side by $N(h)$.

Then, suppose that we can expand the error $M - N(h)$ as a power series:

$$M - N(h) = \sum_{j=1}^{\infty} K_j h^j = K_1 h + K_2 h^2 + \cdots.$$

Then,

$$2(M - N(h/2)) = K_1 h + 2K_2 (h/2)^2 + \cdots.$$

Subtracting these equations eliminates the $\mathcal{O}(h)$ term:

$$M - (2N(h/2) - N(h)) = \mathcal{O}(h^2).$$

We see that, by reusing the first-order accurate formula at h and $h/2$ and combining the results in a clever way, we formally achieve a second-order accurate formula:

$$\begin{aligned} f'(x_0) &= 2\frac{f(x_0+h/2) - f(x_0)}{h/2} - \frac{f(x_0+h) - f(x_0)}{h} \\ &= \frac{-3f(x_0) + 4f(x_0+h/2) - f(x_0+h)}{h}. \end{aligned} \tag{26}$$

Notice that, in this particular case, we re-derive (24) but with $h/2$ instead of h. However, by using more evaluations with different h's and some clever algebraic manipulations, even higher-order methods can be derived.

6 Lecture 6

6.1 Numerical integration

Example 6.1 (Trapezoid rule)**.** Consider a continuous function f defined on an interval $[x_0, x_1]$. Consider the Lagrange interpolating polynomial

$$p(x) = f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0}.$$

Suppose that f is twice continuously differentiable. We recall that, for $x \in (x_0, x_1)$, there is $\xi_x \in (x_0, x_1)$ such that

$$f(x) = p(x) + \frac{f''(\xi_x)}{2}(x - x_0)(x - x_1).$$

Since f and p are smooth, we conclude that the mapping $x \mapsto f''(\xi_x)$ is continuous on (x_0, x_1). Then, we integrate both sides:

$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x = \frac{f(x_0) + f(x_1)}{2}(x_1 - x_0) + \int_{x_0}^{x_1} \frac{f''(\xi_x)}{2}(x - x_0)(x - x_1)\,\mathrm{d}x.$$

The first term on the right is a formula for approximating the integral of f called the Trapezoid rule. Such a rule is called a quadrature rule. To bound the error of the quadrature rule, we subtract it from both sides of the previous equation and take absolute values:

$$\left|\int_{x_0}^{x_1} f(x)\,\mathrm{d}x - \frac{f(x_0) + f(x_1)}{2}(x_1 - x_0)\right| \le \frac{1}{2}\int_{x_0}^{x_1} |f''(\xi_x)|(x - x_0)(x_1 - x)\,\mathrm{d}x.$$

Since f'' is continuous on the closed and bounded interval $[x_0, x_1]$,

$$\begin{aligned}\left|\int_{x_0}^{x_1} f(x)\,\mathrm{d}x - \frac{f(x_0) + f(x_1)}{2}(x_1 - x_0)\right| &\\ \le \frac{1}{2}\max_{\xi\in[x_0,x_1]} |f''(\xi)| \int_{x_0}^{x_1} (x - x_0)(x_1 - x)\,\mathrm{d}x &\\ = \frac{1}{12}\max_{\xi\in[x_0,x_1]} |f''(\xi)|(x_1 - x_0)^3.&\end{aligned}$$

Thus, if we let $h := x_1 - x_0$, we see that the Trapezoid rule is formally third order accurate at approximating the integral of f. □

Similar to numerical differentiation, we used the Lagrange interpolating polynomial to derive a quadrature formula. Just like the previous section, we can also use Taylor's theorem to prove that the quadrature rule is third order accurate.

Theorem 6.2 (Trapezoid rule)**.** *The trapezoid rule*

$$\int_{x_0}^{x_0+h} f(x)\,\mathrm{d}x \approx \frac{h}{2}(f(x_0) + f(x_0 + h)) \tag{27}$$

provides a third order accurate approximation to the integral of a smooth function.

Proof. We take a Taylor expansion of f about x_0 to $x \in (x_0, x_0 + h)$:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(\xi_x)}{2}(x - x_0)^2.$$

Then, we integrate both sides:

$$\int_{x_0}^{x_0+h} f(x)\,\mathrm{d}x = hf(x_0) + \frac{h^2}{2}f'(x_0) + \frac{1}{2}\int_{x_0}^{x_0+h} f''(\xi_x)(x - x_0)^2\,\mathrm{d}x.$$

Now, we replace $f'(x_0)$ by the following differentiation formula from the previous section:

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2}f''(\xi_0).$$

Inserting this above gives

$$\int_{x_0}^{x_0+h} f(x)\,\mathrm{d}x = \frac{h}{2}(f(x_0) + f(x_0 + h)) - \frac{h^3}{4}f''(\xi_0) + \frac{1}{2}\int_{x_0}^{x_0+h} f''(\xi_x)(x - x_0)^2\,\mathrm{d}x.$$

Since f'' is bounded on $[x_0, x_0+h]$, the integral on the right is $\mathcal{O}(h^3)$. Therefore, the error term on the right is $\mathcal{O}(h^3)$ as desired. □

Higher-order quadrature rules can be derived by using higher degree Lagrange interpolants.

Example 6.3 (Simpson's rule)**.** Let $x_i = x_0 + ih$ with $h > 0$ and $i = 0, 1, 2$. Using the Lagrange interpolating polynomial for these points:

$$f(x) = f(x_0)\frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + f(x_1)\frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + f(x_2)\frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} + \frac{f^{(3)}(\xi_x)}{6}(x - x_0)(x - x_1)(x - x_2).$$

By integrating over $[x_0, x_2]$ and substituting for h, we arrive at Simpson's rule:

$$\int_{x_0}^{x_0+2h} f(x)\,\mathrm{d}x = \frac{h}{3}\Big(f(x_0) + 4f(x_0 + h) + f(x_0 + 2h)\Big) + \mathcal{O}(h^4). \tag{28}$$

□

In general, for $n + 1$ evenly spaced points $x_i = x_0 + ih$ with $i = 0, \ldots, n$, a quadrature rule for integrating smooth functions on $[x_0, x_n]$ is given by

$$\int_{x_0}^{x_0+nh} f(x)\,\mathrm{d}x = \sum_{j=0}^{n} w_{j,h} f(x_0 + jh) + \mathcal{O}(h^{n+1}). \tag{29}$$

Here, the $\mathcal{O}(h^{n+1})$ term depends on f having a bounded $(n+1)$st derivative on $[x_0, x_n]$, and the weights $w_{j,j}$ are

$$w_{j,h} := \int_{x_0}^{x_0+nh} L_j(x)\mathrm{d}x, \tag{30}$$

where L_j is the jth Lagrange polynomial of degree n associated to the points x_j.

6.2 Composite numerical integration

The error for the quadrature rules from the previous subsection scales with the length of the interval h that we integrate over. Larger intervals $[a, b]$ lead to larger errors. To deal with this, we can first subdivide the interval $[a, b]$ that we integrate over into equal subintervals $a = z_0 < z_1 < \cdots < z_m = b$, where $z_i = a + ih$, $h = (b - a)/m$. Then, on each subinterval $[z_i, z_{i+1}]$ of length h, we use a quadrature rule with $n + 1$ points $x_{ij} = z_i + jh/n$ for $j = 0, \ldots, n$.

Example 6.4 (Composite trapezoid rule)**.** Integrating a smooth function $f(x)$ on $[a, b]$ using the trapezoid rule on 2 subintervals:

$$\begin{aligned}
\int_a^b f(x)\,\mathrm{d}x &= \int_a^{a+h} f(x)\,\mathrm{d}x + \int_{a+h}^{a+2h} f(x)\,\mathrm{d}x \\
&= \frac{h}{2}\Big(f(a) + f(a+h)\Big) + \frac{h}{2}\Big(f(a+h) + f(a+2h)\Big) \\
&\quad + \mathcal{O}(h^3) \\
&= \frac{h}{2}\Big(f(a) + 2f(a+h) + f(a+2h)\Big) + \mathcal{O}\left(h^3\right).
\end{aligned}$$

In comparison, for the standard trapezoid rule on the interval $[a, b] = [a, a+2h]$:

$$\int_a^b f(x)\,\mathrm{d}x = \; h(\, f(a) + f(a+2h)) + \mathcal{O}((2h)^3),$$

which yields an error that is 8 times larger.

In general, for m equally spaced subintervals, the composite trapezoid rule over $[a, b] = [a, a + mh]$ reads

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{2}\left(f(a) + 2\sum_{j=1}^{m-1} f(a+jh) + f(b)\right) + \mathcal{O}(h^3),$$

with an error m^3 times smaller than the standard trapezoid rule on $[a, b]$. □

7 Lecture 7

7.1 Gaussian quadrature

In the previous sections, we developed quadrature rules of the form

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} w_i f(x_i),$$

where $x_i := a + i(b-a)/n$ and the w_i are chosen such that the formula is exact for polynomials up to degree n. By using a different set of n points and weights, we can achieve a quadrature rule that is accurate for polynomials up to degree $2n+1$. This is known as Gaussian quadrature.

Example 7.1 (Gauss quadrature for $n = 1$)**.** On the interval $[-1, 1]$, we seek points x_0, x_1 and weights w_0, w_1 such that the quadrature rule above is exact for polynomials up to degree 3. Inserting $a = -1$, $b = 1$, $f(x) = 1$, $f(x) = x$, $f(x) = x^2$, and $f(x) = x^3$ above yields

$$\begin{cases} 2 = \displaystyle\int_{-1}^{1} 1\,\mathrm{d}x = w_0 + w_1, \\ 0 = \displaystyle\int_{-1}^{1} x\,\mathrm{d}x = w_0 x_0 + w_1 x_1, \\ 2/3 = \displaystyle\int_{-1}^{1} x^2\,\mathrm{d}x = w_0 x_0^2 + w_1 x_1^2, \\ 0 = \displaystyle\int_{-1}^{1} x^3\,\mathrm{d}x = w_0 x_0^3 + w_1 x_1^3. \end{cases}$$

This is a nonlinear system of equations in 4 unknowns w_0, w_1, x_0, x_1. We can solve it by hand as follows.

From the second equation, we have $w_0 x_0 = -w_1 x_1$. Inserting this in to the 4th equation implies

$$w_0 x_0 (x_0^2 - x_1^2) = 0.$$

If $w_0 = 0$, then the second equation implies that $w_1 x_1 = 0$, which then contradicts the third equation, so $w_0 \neq 0$. If $x_0 = 0$, then we reach the same contradiction, so $x_0 \neq 0$. Therefore, $x_0^2 = x_1^2$, which means that $x_0 = x_1$ or $x_0 = -x_1$. If $x_0 = x_1$, the second equation implies $w_0 = -w_1$, which contradicts the first. Therefore, $x_0 = -x_1$, so the second equation implies that $w_0 = w_1$. Then, from the first equation, we conclude that $w_0 = w_1 = 1$. The third equation then reduces to

$$2/3 = 2x_0^2,$$

so $x_0 = \pm 1/\sqrt{3}$. We choose $x_0 = -1/\sqrt{3}$, so that $x_1 = 1/\sqrt{3}$. In conclusion, the following weights and points give exact quadrature up to degree 3:

$$w_0 = w_1 = 1,\ x_0 = -1/\sqrt{3},\ x_1 = 1/\sqrt{3}.$$

□

The procedure from the previous example does not scale to $n > 1$, so we need a different way to compute the weights and points. The Legendre polynomials help with this.

Definition 7.2 (Legendre polynomials)**.** The Legendre polynomials are the family of polynomials $\{p_n\}_{n\geq 0}$ on $[-1, 1]$ characterized by the following properties:

1. For all n, p_n is a degree n polynomial such that $p_n(1) = 1$.
2. For all $m \neq n$,
$$\int_{-1}^{1} p_m(x)p_n(x)\,\mathrm{d}x = 0.$$

Example 7.3 (Legendre polynomials)**.** The first few Legendre polynomials are as follows:
$$\begin{cases} p_0(x) = 1, \\ p_1(x) = x, \\ p_2(x) = \dfrac{1}{2}(3x^2 - 1). \end{cases}$$

□

Proposition 7.4. *The first $n + 1$ Legendre polynomials are a basis for polynomials of degree at most n.*

Proof. Suppose that
$$\sum_{i=0}^{n} c_i p_i = 0.$$
Then, by multiplying by p_j and integrating over $[-1, 1]$, we conclude that
$$c_j \int_{-1}^{1} p_j(x)^2\,\mathrm{d}x = 0.$$
Since $p_j(1) = 1$, $p_j \neq 0$, so we must have that $c_j = 0$. Since j is arbitrary, $\{p_0, \ldots, p_n\}$ is a linearly independent set of $n + 1$ polynomials of degree at most n. Since the dimension of this space is equal to the number of Legendre polynomials in the set, we are done. □

Given $p_0, \ldots, p_{n-1}$, one can compute p_n by expanding it in the following basis:
$$p_n(x) = \sum_{i=0}^{n-1} a_i p_i(x) + a_n x^n$$
and obtaining an $(n + 1) \times (n + 1)$ system of linear equations by requiring
$$\begin{cases} \displaystyle\int_{-1}^{1} p_j(x)p_n(x)\,\mathrm{d}x = 0 \text{ for all } 0 \leq j \leq n - 1, \\ p_n(1) = 1. \end{cases}$$

Example 7.5. Given p_0, p_1, and p_2 above, we compute p_3 as

$$p_3(x) = a_0 p_0(x) + a_1 p_1(x) + a_2 p_x(x) + a_3 x^3.$$

Now, the constraints read

$$\begin{cases} 0 = \displaystyle\int_{-1}^{1} p_0(x) p_3(x)\,\mathrm{d}x = a_0 \int_{-1}^{1} p_0(x)^2\,\mathrm{d}x, \\ 0 = \displaystyle\int_{-1}^{1} p_1(x) p_3(x)\,\mathrm{d}x = a_1 \int_{-1}^{1} p_1(x)^2\,\mathrm{d}x + a_3 \int_{-1}^{1} p_1(x) x^3\,\mathrm{d}x, \\ 0 = \displaystyle\int_{-1}^{1} p_2(x) p_3(x)\,\mathrm{d}x = a_2 \int_{-1}^{1} p_2(x)^2\,\mathrm{d}x \\ 1 = p_3(1) = a_0 + a_1 + a_2 + a_3. \end{cases}$$

Thus, we see that $a_0 = 0$, $a_2 = 0$, so we have the following linear system for a_1 and a_3:

$$\begin{cases} 2/3 a_1 + 2/5 a_3 = 0, \\ a_1 + a_3 = 1. \end{cases}$$

This can be easily solved, so we conclude that $p_3(x) = a_1 p_1(x) + a_3 x^3$. □

There exists a general formula for p_n, but its derivation is beyond the scope of these notes. Here's how they relate to Gaussian quadrature. We will not cover the proofs.

Lemma 7.6 (Roots of Legendre polynomials)**.** *Let p_n be the nth Legendre polynomial. Then, p_n has n distinct roots in the interval $(-1, 1)$.*

Theorem 7.7 (Gaussian quadrature via Legendre polynomials)**.** *For $n \geq 0$, let $x_0, \ldots, x_n$ be the roots of the $n+1$ Legendre polynomial p_{n+1}. Let*

$$w_i := \frac{2}{(1 - x_i^2) p'_{n+1}(x_i)^2}.$$

Then, the Gauss quadrature rule

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} w_i f(x_i)$$

is exact for polynomials up to degree $2n + 1$.

There are tabulations of Gauss quadrature weights and points on $[-1, 1]$ online. To obtain the corresponding quadrature rule on an arbitrary interval $[a, b]$, we can use the following change of variables:

$$x = \frac{a+b}{2} + \frac{b-a}{2}\widehat{x}, \qquad \mathrm{d}x = \frac{b-a}{2}\,\mathrm{d}\widehat{x}.$$

That is,

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &= \int_{-1}^1 f\left(\frac{a+b}{2}+\frac{b-a}{2}\widehat{x}\right)\frac{b-a}{2}\,\mathrm{d}\widehat{x}\\ &\approx \sum_{i=0}^n \frac{b-a}{2}\widehat{w}_i f\left(\frac{a+b}{2}+\frac{b-a}{2}\widehat{x}_i\right)\\ &= \sum_{i=0}^n w_i f(x_i),\end{aligned}$$

where $\widehat{w}_i$ and $\widehat{x}_i$ are the Gauss quadrature weights and points on the interval $[-1,1]$ and the corrsponding weights and points on $[a,b]$ are

$$w_i = \frac{b-a}{2}\widehat{w}_i, \qquad x_i = \frac{a+b}{2}+\frac{b-a}{2}\widehat{x}_i.$$

7.2 Euler's method for ordinary differential equations (ODEs)

We now wish to develop numerical methods for solving ODE problems of the form

$$y'(t) = f(t,y(t)), \qquad y(0)=y_0, \tag{31}$$

where $f(t,y)$ is a given function, y_0 is a given initial condition, and the unknown is $y(t)$. We recall some ODE theory concerning existence and uniqueness of solutions to the above problem.

Definition 7.8 (Lipschitz)**.** A function of two real variables $f(t,y)$ is Lipschitz with respect to y if there is a constant $C>0$ such that for all t, y_1, y_2,

$$|f(t,y_1)-f(t,y_2)| \le C|y_1-y_2|.$$

We call C a Lipschitz constant of f.

Proposition 7.9. *If $f(t,y)$ is Lipschitz with respect to y, it is uniformly continuous with respect to y.*

Proof. Let $\varepsilon>0$, and let $C>0$ be a Lipschitz constant of f. Then, for each t, y_0, when $|y-y_0|<\delta := \varepsilon/C$,

$$|f(t,y)-f(t,y_0)| \le C|y-y_0| < \varepsilon.$$

□

Proposition 7.10. *If $f(t,y)$ is differentiable with respect to y and there is a constant C such that $|\partial_y f(t,y)| \le C$ for all t, y, then f is uniformly Lipschitz with respect to y.*

Proof. Fix t, y_1, y_2. Applying the Mean Value Theorem in y, there is some $y_3 \in (y_1, y_2)$ such that

$$|f(t,y_1)-f(t,y_2)| = |\partial_y f(t,y_3)||y_1-y_2| \le C|y_1-y_2|.$$

Since C is independent of t, y_1, and y_2, we are done. □

Example 7.11. The function $f(t, y) = t - y^2 + 1$ is Lipschitz in y, but $f(t, y) = \sqrt{y}$ is not.

Theorem 7.12 (Picard–Lindelöf)**.** *If f is continuous with respect to t and uniformly Lipschitz with respect to y, then the ODE problem* (31) *has a unique solution.*

If a problem does not have a solution at all, it makes no sense to develop numerical methods for it. If a problem has a solution, but it is not unique, then additional information must be supplied to choose a unique solution to approximate. There is one other additional property that determines how feasible it is to approximate a solution to an ODE problem. We discuss this now.

Definition 7.13 (Stability)**.** Consider the ODE problem (31) and the following slightly perturbed problem:

$$z'(t) = f(t, z(t)) + \varepsilon, \qquad z(0) = y_0 + \delta.$$

Suppose that $z(t)$ and $y(t)$ belong to a space of function with a norm $\|\cdot\|$ defined on it. We say that (31) is stable if, whenever ε and δ are small, the difference $\|y - z\|$ is small.

That is, small perturbations in the function f or the initial data y_0 do not give vastly different solutions to the ODE. If the ODE problem has a unique solution and is stable, we say the problem is well-posed. Well-posed problems are much easier to solve than ill-posed ones. We will only concern ourselves with well-posed problems in this course.

Now, assuming we have a well-posed problem, how do we numerically compute the solution to (31)? A simple idea is to replace the derivatives in (31) with finite differences that we learned from a previous section. By using forward differences, we obtain the forward Euler method, also known as the explicit Euler method.

Suppose we want to solve the ODE problem from $t = 0$ until some final time t_F. We let $h = t_F/N$ denote the timestep size, and we set $t_n = nh$. A forward difference of $y'(t)$ at $t = t_n$ reads

$$y'(t_n) \approx \frac{y(t_{n+1}) - y(t_n)}{h}.$$

Motivated by this, we consider the following algebraic procedure. We start with the initial condition y_0. Then, for $n \geq 0$, after obtaining y_n, to obtain y_{n+1}, we solve

$$\frac{y_{n+1} - y_n}{h} = f(t_n, y_n).$$

Or, after rearranging, we get the following explicit Euler update:

$$y_{n+1} = y_n + hf(t_n, y_n). \tag{32}$$

Whether the computed values y_n accurately approximate $y(t_n)$ depends on the function f and the step size h. Let

$$e_n := y(t_n) - y_n$$

denote the error at time t_n. Assuming that the exact solution y is smooth,

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(\xi_n)$$

for some $\xi_n \in (t_n, t_{n+1})$. Using the ODE:

$$y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}y''(\xi_n).$$

Subtracting (32) from this equation:

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

Now, assume that f is Lipschitz in y with constant $L > 0$. Then, by taking absolute values and applying the triangle inequality with the Lipschitz assumption,

$$|e_{n+1}| \leq (1 + hL)|e_n| + \frac{h^2}{2}|y''(\xi_n)|.$$

Suppose that y'' is bounded with constant $M > 0$. Then, by recursively applying this inequality:

$$\begin{aligned}
|e_{n+1}| &\leq (1 + hL)\left((1 + hL)|e_{n-1}| + \frac{h^2}{2}M\right) + \frac{h^2}{2}M \\
&= (1 + hL)^2|e_{n-1}| + \frac{h^2}{2}M(1 + (1 + hL)) \\
&\vdots \\
&\leq (1 + hL)^{n+1}|e_0| + \frac{h^2}{2}M\sum_{j=0}^{n}(1 + hL)^j.
\end{aligned}$$

From the initial condition, $e_0 = 0$, so

$$|e_{n+1}| \leq \frac{h^2}{2}M\sum_{j=0}^{n}(1 + hL)^j.$$

Recalling the geometric sum identity,

$$\sum_{j=0}^{n} r^j = \frac{1 - r^{n+1}}{1 - r},$$

we apply this to the inequality above:

$$|e_{n+1}| \leq \frac{h^2}{2} M \frac{(1+hL)^{n+1}-1}{hL} = \frac{h}{2}\frac{M}{L}\left((1+hL)^{n+1}-1\right).$$

We now recall the following property of the exponential function:

$$e^x = \lim_{m\to\infty}\left(1+\frac{x}{m}\right)^m,$$

and, when $x > 0$,

$$e^x \geq \left(1+\frac{x}{m}\right)^m$$

for all m. Using this above for $m = n+1$ and $x = (n+1)hL = t_{n+1}L$, we conclude the following.

Theorem 7.14 (Accuracy of the forward Euler method)**.** *Suppose $f(t,y)$ is Lipschitz in y with constant $L > 0$, and suppose that the solution y to* (31) *has a bounded second derivative with bound $M > 0$. Then, the forward Euler method* (32) *with timestep $h > 0$ and discrete time points $t_n = nh$ has the following error bound for all n:*

$$|y(t_n) - y_n| \leq h\frac{M}{2L}\left(e^{t_n L}-1\right).$$

Thus, the forward Euler method is only first-order accurate.

8 Lecture 8

8.1 Higher-order Taylor methods

In the previous section, we derived the forward Euler method by using a forward difference approximation to the derivative. We also could have derived the method by using Taylor expansions and replacing certain derivatives with the function $f(t, y)$ from the ODE. Let us do this now, since we can generalize it to higher-order accurate methods later.

Let $y(t)$ be the solution to the ODE (31), and assume that y is smooth. Then,

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(\xi_n)$$

for some $\xi_n \in (t_n, t_{n+1})$. Using the ODE (31),

$$y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}y''(\xi_n).$$

By dropping the $\mathcal{O}(h^2)$ term, we obtain the forward Euler method:

$$y_{n+1} = y_n + hf(t_n, y_n).$$

If we continue further out in the Taylor expansion, we can obtain higher-order accurate methods. For example, expanding out one more term:

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \frac{h^3}{6}y'''(\xi_n).$$

We replace $y'(t_n)$ by $f(t_n, y(t_n))$. To deal with $y''(t_n)$, we take a derivative of (31):

$$\begin{aligned} y''(t) &= \partial_t f(t, y(t)) + \partial_y f(t, y(t))y'(t) \\ &= \partial_t f(t, y(t)) + \partial_y f(t, y(t))f(t, y(t)). \end{aligned}$$

Inserting this and dropping the $\mathcal{O}(h^3)$ term, we obtain a second-order accurate explicit method:

$$y_{n+1} = y_n + hf(t_n, y_n) + \frac{h^2}{2}\Big(\partial_t f(t_n, y_n) + \partial_y f(t_n, y_n)f(t_n, y_n)\Big). \tag{33}$$

Now, we need f, $\partial_t f$, and $\partial_y f$ to be Lipschitz in y in order to prove an error bound similar to the forward Euler method.

Example 8.1 (Second-order accurate explicit method)**.** Let $f(t, y) = y^3$. Then the second-order accurate scheme above reads

$$y_{n+1} = y_n + hy_n^3 + \frac{3h^2}{2}y_n^5.$$

□

8.2 Runge–Kutta methods

Computing derivatives of $f(t, y(t))$ is tedious and restricts the methods of the previous section to only be useful for sufficiently smooth f. To circumvent this, we can seek to replace derivatives of $f(t, y(t))$ with values of the form $f(t + \alpha, y + \beta)$. This process leads to a class of methods called Runge–Kutta methods. We explain with an example.

Returning to the second-order explicit scheme (33), if we expand $f(t+\alpha, y+\beta)$ in a Taylor expansion in both variables:

$$f(t + \alpha, y + \beta) = f(t, y) + \alpha\partial_t f(t, y) + \beta\partial_y f(t, y) + \mathcal{O}(\alpha^2) + \mathcal{O}(\alpha\beta) + \mathcal{O}(\beta^2).$$

If we set $\alpha = h/2$ and $\beta = hf(t, y)/2$,

$$f(t+h/2, y+hf(t, y)/2) = f(t, y)+ \frac{h}{2}\bigg(\partial_t f(t, y) + \partial_y f(t, y)f(t, y)\bigg)+\mathcal{O}(h^2).$$

Dropping the $\mathcal{O}(h^2)$ term, we see that the term on the right matches up with the terms in (33). Therefore, by making the substitution above, we obtain the following second-order explicit Runge–Kutta scheme:

$$y_{n+1} = y_n + hf\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}f(t_n, y_n)\right). \tag{34}$$

For even higher-order methods, the procedure is similar. Perform a high-order Taylor expansion of $y(t_{n+1})$, replacing derivatives of y with partial derivatives of f. Then, for the partial derivatives of f, replace them with suitable values of the form $f(t + \alpha, y + \beta)$ where α and β come from matching coefficients with higher-order Taylor expansions of f. A popular high-order explicit Runge–Kutta scheme is the 4th-order Runge–Kutta scheme, also known as RK4:

$$\begin{cases} k_1 := f(t_n, y_n), \\ k_2 := f(t_n + h/2, y_n + h/2k_1), \\ k_3 := f(t_n + h/2, y_n + h/2k_2), \\ k_4 := f(t_n + h, y_n + hk_3), \\ y_{n+1} = y_n + \dfrac{h}{6}\bigg(k_1 + 2k_2 + 2k_3 + k_4\bigg), \end{cases} \tag{35}$$