Numerical Analysis

Pramana

Spring 2023

The goal of this course is to provide graduate students and advanced undergraduate students with an introduction to numerical analysis, or numerical approaches to calculus. Topics to be covered include root-finding, interpolation, polynomial approximation, numerical differentiation and integration, and the numerical solution of ordinary differential equations. The course will cover mathematical theory, and also the issues concerning the implementation of the algorithms on modern computer hardware.

Professor: Chris Rycroft.

Contents

1. Introduction

January 25, 2023

For this class, we use the book *An Introduction to Numerical Analysis* by Süli and Mayers [SM03]. If I refer to "the book", I mean this one. Additionally, I will reference some python files to demonstrate our methods in action, which can be found on the Class Github Page. I do not mention them all, so feel free to explore them.

Numerical analysis is the study of the algorithms, while *Scientific Computing* emphasizes their applicaton to practical problems.

January 27, 2023

Key ideas for Numerical Analysis:

1. Approximating infintie/continuous process with finite/discrete process
2. Creating error bounds

1.1. Error analysis

1.1.1. Sources of error

There are multiple sources of error in numerical approximations:

1. Truncation/discretization: Approximations in order to compute things (finite differences, trucating infinite sequences, etc.)
2. Rounding: Finite precision arithmetic (limitations of hardware/memory storage)

Example 1.1 – Suppose we want to numerically approximate the derivative of $f(x) = e^x \sin x$. We can do it by a *finite difference approximation* to $f'(x)$:

$$f_{\text{diff}}(x;h) = \frac{f(x+h) - f(x)}{h}.$$

From Taylor series, we know

$$f(x+h) = f(x) + hf'(x) + \frac{f''(\theta)h^2}{2}, \quad \theta \in [x, x+h].$$

Thus

$$f_{\text{diff}}(x;h) = \frac{f(x+h) - f(x)}{h} = f'(x) + \underbrace{\frac{f''(\theta)h}{2}}_{\text{"error"}}.$$

Suppose that $|f''(\theta)| \leq M$. Then

$$\left|f'(x) - f_{\text{diff}}(x;h)\right| \leq \frac{Mh}{2}$$

gives the discretization error. Let $\widetilde{f}_{\text{diff}}(x;h)$ denote the finite precision approximation to $f_{\text{diff}}(x;h)$. The numerator of $\widetilde{f}_{\text{diff}}$ introduces a rounding error of size $\varepsilon\,|f(x)|$.

$\varepsilon \approx 10^{-16}$ on modern computers.

$$\begin{aligned}\left|f_{\text{diff}}(x;h) - \widetilde{f}_{\text{diff}}(x;h)\right| &\leq \left|\frac{f(x+h) - f(x)}{h} - \frac{f(x+h) - f(x) + \varepsilon\,|f(x)|}{h}\right| \\ &= \frac{\varepsilon\,|f(x)|}{h}.\end{aligned}$$

To get the total error, we use the triangle ineq.

$$\begin{aligned}\left|f'(x) - \widetilde{f}_{\text{diff}}(x;h)\right| &\leq \left|f'(x) - f_{\text{diff}}(x;h)\right| + \left|f_{\text{diff}}(x;h) - \widetilde{f}_{\text{diff}}(x;h)\right| \\ &\leq \frac{Mh}{2} + \underbrace{\frac{\varepsilon\,|f(x)|}{h}}_{\text{dominates when } h \text{ small}}\end{aligned}$$

1.1.2. Error definitions

Definition 1.1 (Absolute and relative error)

The **absolute error** is

$$\text{abs. error} = \text{true value} - \text{approximate value}.$$

The **relative error** is

$$\text{rel. error} = \frac{\text{abs. error}}{\text{true value}}$$

We can use a more accurate numerical approximation to get the "true" value.

Example in class with python code: `deriv.py`. Error was smallest in the middle (in a log plot). Our errors start with being discretization dominated, so by decreasing the step size we gain accuracy. However, when step size gets very small, our rounding error starts to matter more, and we lose accuracy.

Definition 1.2 (Algebraic convergence)

January 30, 2023

Let $y = |\text{abs. error}|$. If we have **algebraic convergence**, then

$$y \approx \alpha h^{\beta}.$$

This implies that

$$\log y \approx \log \alpha + \beta \log h,$$

so the absolute error should follow a straight line on a log plot.

2. Iterative solutions

2.1. Root-finding

We are motivated to employ root-finding algorithms because solving $f(x) = 0$ for a function f analytically may be impossible, or require too much effort.

Example 2.1 (Zeros in polynomials) – Not all functions have zeros that can be found easily (or at all).

1. $f(x) \in \mathbb{R}[x]$, where $\deg f = 2, 3, 4$ can be solved with a formula, but it gets increasingly messy.
2. But if $\deg f = 5$, then there is no general solution (Abel, 1824).
3. Does $f(x) = \cos^2 x - \frac{x \sin x}{1+x^2}$ have zeros? How many and where?
   - f is continuous
   - $f(0) = 1$, $f\left(\frac{\pi}{2}\right) < 0$.

   By IVT there exists a zero in $\left[0, \frac{\pi}{2}\right]$.

To solve the last function, consider a related problem: solving $x = g(x)$, which is the same as solving $f(x) = x - g(x) = 0$. Then

$$x - \underbrace{\left(x + \cos^2 x - \frac{x \sin x}{1 + x^2}\right)}_{g(x)} = 0$$

has the same zeros. We can solve this iteratively by starting with x_0 and defining a sequence $(x_k)_{k \in \mathbb{N}}$, where $x_{k+1} = g(x_k)$. We hope $\lim_{k \to \infty} x_k = \xi$, where $\xi = g(\xi)$, which we call a **fixed point**. The program `iter.py` does this.

Definition 2.1 (Exponential convergence)
Sometimes we encounter **exponential convergence**, where if $y = |\text{abs. error}|$,

$$y \approx \alpha e^{-\beta k} \quad (= \alpha C^k).$$

Then

$$\log y \approx \log \alpha - \beta k,$$

which is linear, and can be found with linear regression.

Theorem 2.2 (Brouwer's fixed point theorem)
If $g(x)$ is continuous on $[a, b]$ and $g(x) \in [a, b]$ on $x \in [a, b]$, then $\xi = g(\xi)$ for some $\xi \in [a, b]$.

Proof. Let $f(x) = x - g(x)$. Then $f(a) = a - g(a) \leq 0$, $f(b) = b - g(b) \geq 0$, and IVT finishes. □

2.2. Determining convergence of iterative solutions

February 1, 2023

Example 2.3 (Divergence of iterative solution) – Find the roots of $f(x) = e^x - x - 4$.

Solution. We write

$$x = e^x - 4, \quad \text{or} \quad x = \log(x+4).$$

Then run an iterative solution. For the first equation starting at $x = 2$, it blows up quickly. For the second one, it converges to our solution. □

To analyze why this happened, compare the two sequences

$$x_{k+1} = e^{x_k} - 4, \qquad x_{k+1} = \log(x_k + 4).$$

Looking at consecutive values, we see that the first sequence goes to ∞ quickly, but the small gradient of the second sequence makes it converge.

The convergence of $x_{k+1} = g(x_k)$ depends on whether $g(x)$ is a contraction.

Definition 2.2 (Contraction)
A function g on $[a, b]$ is a **contraction** if $\exists L \in [0, 1)$ such that

$$|g(x) - g(y)| \leq L\,|x - y|\,, \quad \forall x, y \in [a, b].$$

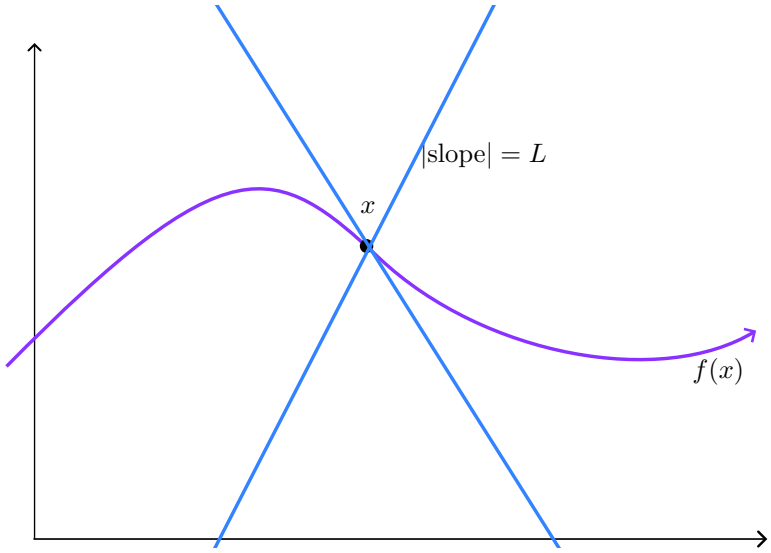

Figure 1: Example of a contraction

Figure 1 shows a graphical interpretation of a contraction. It shows that all points of f near x must lie between two blue lines with slope L.

Theorem 2.4 (Contraction mapping theorem)
Let g be continuous on $[a, b]$ such that $g(x) \in [a, b]$ for $x \in [a, b]$. If $g(x)$ is a contraction, then

- g has a unique fixed pt. $\xi \in [a, b]$,
- The sequence (x_k) given by $x_{k+1} = g(x_k)$ converges to ξ for any $x_0 \in [a, b]$.

Proof. A fixed point $\xi \in [a,b]$ exists by Theorem 2.2. To show uniqueness, assume that $\eta \in [a,b]$ is also a fixed point. Then

$$|\xi - \eta| = |g(\xi) - g(\eta)| \leq L\,|\xi - \eta| \implies |\xi - \eta| = 0.$$

To show convergence, we track the error

$$\begin{aligned} e_k &= |x_k - \xi| \\ &= |g(x_{k-1}) - g(\xi)| \\ &\leq |x_{k-1} - \xi|\, L \\ \implies e_k &\leq L e_{k-1} \leq L^2 e_{k-2} \leq \cdots \leq L^k e_0. \end{aligned}$$

This shows we have exponential convergence to ξ; smaller $L \implies$ faster convergence.

Since $\lim_{k\to\infty} L^k = 0$ if $|L| < 1$, $\lim_{k\to\infty} e_k = 0$. □

Suppose that g is differentiable on $[a,b]$ and $|g'(x)| \leq L$ for $L \in [0,1)$. Consider $x, y \in [a,b], x \neq y$. There exists $z \in [a,b]$ such that

$$g'(z) = \frac{g(x) - g(y)}{x - y}.$$

Thus differentiability and slopes less than 1 yield a contraction. But there are non-differentiable functions that are contractions: $g(x) = \frac{1}{2}|x|$ is a contraction on $[-1,1]$.

The convergence rate depends somewhat on $|g'(\xi)|$, since

$$\lim_{k\to\infty} \frac{e_{k+1}}{e_k} = |g'(\xi)|\,.$$

As $k \to \infty$, $e_k \approx \left|g'(\xi)^k\right| e_0$.

2.3. Newton's method

We try to solve $f(x) = 0$. Suppose

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

What λ gives the fastest convergence? If $\lim_{n\to\infty} x_k = \xi$, then $\xi = \xi - \lambda f(\xi) \implies f(\xi) = 0$.

$$g(x) = x\lambda f(x) \implies g'(\xi) = 1 - \lambda f'(\xi).$$

So $\lambda = \frac{1}{f'(\xi)}$ is ideal.

February 3, 2023

Use $\lambda = \frac{1}{f'(x_k)}$. Then we rewrite the recursion as

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

Newton's method is shown graphically in Figure 2.

Remark 2.5. Problems with Newton's method:

- When $f'(x_k) = 0$, there is no intersection with $y = 0$.
- Some functions diverge with Newton's method, such as $f(x) = x^{1/3}$.
- We need to compute $f'(x_k)$ for each iteration.

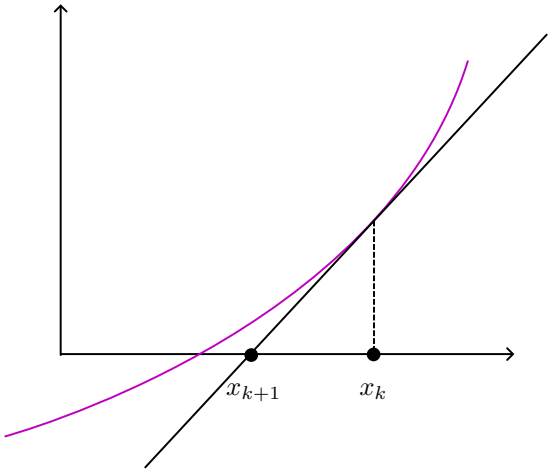

Figure 2: Newton's method

While some of these we cannot deal with, the last one may be aided by approximating f' with the **secant method**:

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

Thus

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

2.4. Bisection method

Let f be a function and a, b be values so that $f(a)f(b) < 0$ and f is continuous. IVT tells us that there is a root. Let $c = \frac{a+b}{2}$.

- If $f(c)f(b) = 0$, then $\xi = c$ is a root,
- If $f(c)f(b) < 0$, repeat with $[c, b]$,
- If $f(c)f(b) > 0$, repeat with $[a, c]$.

This is guaranteed to converge, since the interval is always chopped in half. In fact, we only need the sign of $f(c)$, $f(b)$. This is the **bisection method**.

All 3 methods have code: `newton.py`, `secant.py`, and `bisection.py`. Comparing the methods with $e^x - x - 4$, Newton's and secant have rapid (quadratic) convergence, while bisection is "linear" (both in the log plot).

2.5. Convergence rates

Definition 2.3 (Linear convergence)

Consider a sequence of errors $|x_k - \xi| \leq e_k$.

- If
$$\lim_{k\to\infty} \frac{e_{k+1}}{e_k} = \mu, \qquad \mu \in (0,1),$$
then e_k converges linearly (which is the same as exponential convergence). This means $e_k \approx \mu^k e_0$.
- The convergence rate (ρ) is the number of correct decimal digits gained per iteration, which is $\rho = -\log 10(\mu)$.
- If $\lim_{k\to\infty} \frac{e_{k+1}}{e_k} = \mu = 0$, then e_k converges super-linearly.
- If the limit converges to 1, but $e_k \to 0$, then e_k converges sub-linearly.
- If
$$\lim_{k\to\infty} \frac{e_{k+1}}{e_k^2} = \mu > 0,$$
then e_k converges quadratically.

February 6, 2023

Remark 2.6 (Determining quadratic convergence)**.** For large k in quadratic convergence,

$$e_k \approx \mu e_{k-1}^2 \approx \mu(\mu e_{k-2}^2)^2 \approx \cdots \approx \mu^{2^k-1} e_0^{2^k} = \mu^{-1}(\mu e_0)^{2^k}.$$

Thus

$$\log e_k = -\log \mu + 2^k \log(\mu e_0).$$

To check if a sequence converge quadratically, we could graph $\log(\log(e_k))$. However, this is hard to do with with computer results because of rounding error/machine precision.

2.5.1. Application: Convergence of Newton's method

Recall Newton's method uses the formula

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

Define $e_k = |x_k - \xi|$. Near x_k, we know that

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(\eta_k)(x - x_k)^2$$

for η_k between x and x_k (using Taylor's formula). Considering $f(\xi)$,

$$\begin{aligned} -f(x_k) + x_k f'(x_k) - \xi f'(x_k) &= \frac{1}{2} f''(\eta_k)(\xi - x_k)^2 \\ x_{k+1} - \xi &= \frac{1}{2} \frac{f''(\eta_k)}{f'(x_k)} (\xi - x_k)^2 \\ |x_{k+1} - \xi| &= \frac{1}{2} \left| \frac{f''(\eta_k)}{f'(x_k)} \right| |\xi - x_k|^2 . \end{aligned}$$

Suppose now that $\left|\frac{f''(x)}{f'(y)}\right| \leq A$ for all x, y and choose x_0 s.t. $|\xi - x_0| \geq \frac{1}{A}$. Then

$$|\xi - x_1| \leq \frac{1}{2} |\xi - x_0| \implies |\xi - x_k| \leq \frac{1}{2^k} |\xi - x_0| .$$

This shows that the error term e_k converges. Since $\eta_k, x_k \to \xi$ as $k \to \infty$, we also have

$$\lim_{k\to\infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|^2} = \left| \frac{f''(\xi)}{2f'(\xi)} \right| \leq \frac{A}{2}.$$

Thus Newton's method gives $e_{k+1} \approx \mu e_k^2$ for $\mu = \left| \frac{f''(\xi)}{2f'(\xi)} \right|$. So $x_k \to \xi$ *quadratically*. This motivates the following theorem.

Theorem 2.7 (Conditions for quadriatic convergence of Newton's method)

Let $f(\xi) = 0$ and $I\delta = [\xi - \delta, \xi + \delta]$. If $f \in C^2(I\delta)$ and $\left| \frac{f''(x)}{f'(y)} \right| \leq A$ for some $A > 0$ and all $x, y \in I\delta$, then Newton's method converges quadratically for an initial guess x_0 given that $|x_0 - \xi| \leq \min\left\{\frac{1}{A}, \delta\right\}$.

Remark 2.8.

- The above theorem applies to many cases, but may have problems at multiple roots where $f'(\xi) = 0$.
- Iterative equations like Newton's method can diverge or have other complicated behavior. For example, consider $x_{k+1} = ax_k(1 - x_k)$ Computer simulation show that sometimes it can converge, but also oscillates, or unpredictable behavior depending on our choice of a.
- Newton's method has the ability to generate fractals that show which root the method converges to (see Newton fractals).

3. Polynomial interpolation

3.1. Condition number

February 8, 2023

Many polynomial operation boil down to $y = f(x)$, where x is an input and y is an output (both may be scalars, vectors, or something else). We may wonder how a change Δx in input will affect the output:

$$y + \Delta y = f(x + \Delta x).$$

Definition 3.1 (Condition number)
Given a function f such that

$$y + \Delta y = f(x + \Delta x),$$

the **condition number** is

$$\kappa = \frac{|\Delta y / y|}{|\Delta x / x|}.$$

κ is unitless, and often reported as the largest such κ over a range of Δx.

3.1.1. Scalar functions

Let $f : \mathbb{R} \to \mathbb{R}$ be a scalar, differentiable function such that $f(x) = y$.

$$\begin{aligned} y + \Delta y &= f(x + \Delta x) \\ \frac{\Delta y}{y} &= \frac{f(x + \Delta x) - f(x)}{f(x)} \\ &= \frac{f(x + \Delta x) - f(x)}{\Delta x} \frac{\Delta x}{f(x)} \\ &\approx \frac{f'(x)\Delta x}{f(x)} \\ \kappa &\approx \left| \frac{f'(x)x}{f(x)} \right|. \end{aligned}$$

3.1.2. Matrix multiplication

To get a general measure of error we introduce norms, which will replace our absolute value bars for non-scalar values.

Definition 3.2 (Norm)
A **norm** $\|\cdot\| : V \to \mathbb{R}$ is defined on a real or complex vector space, and measures distance to the origin. It must be *positive definite*, have *scalar multiplication*, and satisfy the *triangle inequality*.

For $A \in \mathbb{R}^{n \times n}, x, b \in \mathbb{R}^n$, let $Ax = b$ (where A is invertible). Then $A(x + \Delta x) = b + \Delta b$. Since A is a linear map, $A\Delta x = \Delta b$. The condition number in this case is

$$\kappa = \frac{\|\Delta b\| \,/\, \|b\|}{\|\Delta x\| \,/\, \|x\|} = \frac{\|\Delta b\|}{\|b\|} \cdot \frac{\|x\|}{\|\Delta x\|} = \frac{\|A\Delta x\|}{\|\Delta x\|} \cdot \frac{\|x\|}{\|Ax\|}.$$

To "measure" A, we define the *matrix norm induced by the vector norm* on A as

$$\|A\| = \max_{v \neq 0} \frac{\|Av\|}{\|v\|}.$$

Thus

$$\kappa \leq \|A\| \cdot \frac{\|x\|}{\|Ax\|}.$$

Using the fact that $x = A^{-1}b$,

$$\kappa \leq \|A\| \frac{\left\|A^{-1}b\right\|}{\|v\|} \leq \|A\| \cdot \left\|A^{-1}\right\|.$$

Definition 3.3 (Condition number of a matrix)
We may define the condition number of a matrix A as

$$\kappa(A) \coloneqq \|A\| \cdot \left\|A^{-1}\right\|.$$

The function `numpy.linalg.cond` numerically calculates this value.

Solving a linear system has the same condition number as matrix multiplication: Suppose $C \in \mathbb{R}^{n \times n}$ and $y, f \in \mathbb{R}^n$. We try to solve

$$Cy = f,$$

where f is the input and y is the output. If C is invertible, we can rewrite to

$$y = C^{-1}f,$$

so, by the last section,

$$\kappa = \|C\| \cdot \left\|C^{-1}\right\|.$$

Example 3.1 (Calculating matrix norm) – Let

$$A = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}.$$

$$\begin{aligned} \|A\| &= \max_{v \neq 0} \frac{\|Av\|}{\|v\|} \\ &= \max_{\|v\|=1} \|Av\| \\ &= \max_{x^2+y^2=1} \|(3x, y)\| \\ &= \max_{x^2+y^2=1} \sqrt{3x^2 + y^2} = 3. \end{aligned}$$

Remark 3.2. The condition number characterizes the spread of eigenvalues. In general, it is the ratio of the singular values.

3.2. Floating-point arithmetic

February 10, 2023

To obtain an accurate answer we need to apply a *stable* numerical method to a *well-conditioned* mathematical problem. A "stable numerical method" is a method that doesn't accumulate error.

To investigate a source of error, we look at how computers work with floating-point numbers. Computers use the "scientific notation" of numbers in base 2.

$$\underbrace{\pm}_{\text{sign}} \quad \underbrace{d_1, \ldots, d_p}_{\text{mantissa bits}} \quad \underbrace{E}_{\text{exponent bits}} .$$

d_0 is assumed to be 1 (except for 0). E lies in an interval $L \leq E \leq U$. For example,

- IEEE single point precision has $p = 23$, $L = -126$, and $U = 127$,
- IEEE double point precision has $p = 52$, $L = -1022$, and $U = 1023$.

There are also exception values: Inf, NaN, 0.

How to represent $x \in \mathbb{R}$ that is not a floating point number?

- **Case 1:** Too small ($\approx 10^{-323}$) $\to 0$ or too large ($\approx 10^{308}$) $\to$ Inf
- **Case 2:** Too many mantissa bits. ε is the difference between 1 and the next floating point number after 1. This is machine precision. In IEEE double precision, $\varepsilon \approx 2.22 \times 10^{-16}$.

We have

$$\left| \frac{\texttt{round}(x) - x}{x} \right| < \varepsilon,$$

so we can bound relative error by machine error.

We want numerical methods to have backward stability. For example, $Ax = b$ should give the same solution for $(A + \Delta A)x = b + \Delta b$ for small $\Delta A, \Delta b$.

3.3. Polynomial interpolation with matrices

Definition 3.4 (Polynomial notation)

Let $\mathbb{P}_n$ denote the set of all polynomials of degree n on $\mathbb{R}$. If $p(\,\cdot\,; b) \in \mathbb{P}_n$, then

$$P(x; b) = \sum_{k=0}^{n} b_k x^k,$$

for $b \in \mathbb{R}^{n+1}$.

For data points $S = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, we want to find a polynomial that passes through every point. $\{x_0, \ldots, x_n\}$ are the interpolation points. We require our desired polynomial p to satisfy

$$p(x_i; b) = y_i.$$

This gives $n + 1$ equations to solve, so we look for $p \in \mathrm{P}_n$. These equations can become a matrix

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

or

$$V\mathbf{b} = \mathbf{y}.$$

V is a Vandermonde matrix.

Is there a unique solution to this system?

Lemma 3.3
If the $n+1$ interpolation points are distinct, then $V\mathbf{b} = \mathbf{y}$ has a unique solution.

A result in linear algebra is that if $A\mathbf{z} = 0 \implies \mathbf{z} = 0$ (over a finite-dimensional real vector space), then $A\mathbf{b} = \mathbf{y}$ has a solution.

In our case, if $V\mathbf{b} = 0$, then $p(\,\cdot\,; b) \in \mathbb{P}_n$ vanishes at $n+1$ distinct points. A polynomial of degree n has $n+1$ roots only when it is the 0 polynomial, so $b = 0$, and a unique solution exists.

Remark 3.4 (Disadvantages of Vandermonde matrices)**.** The Vandermonde matrix is ill-conditioned, which means that $\kappa(V)$ gets large.

Horner's method lets us evaluate polynomials faster: $a + bx + cx^2 \to a + x(b + cx)$ etc...

Even with smaller degree polynomials, there are hints of blow-up. When graphing an interpolation, the largest errors tend to be near the endpoints. A small change in $\mathbf{y}$ changes $\mathbf{b}$ significantly.

3.3.1. Changing the basis

We try constructing a basis such that the interpolation matrix (the analog of V) is I. We can do this using Lagrange polynomials. Let $L_k \in \mathbb{P}_n$, $k = 0, \dots, n$ where

$$L_k(x_i) = \begin{cases} 0 & \text{if } i \neq k, \\ 1 & \text{if } i = k. \end{cases}$$

This polynomial exists:

$$L_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}. \tag{3.1}$$

The Lagrange polynomial interpolates discrete points exactly. *Can we use interpolation to accurately approximate continuous functions?*

Theorem 3.5 (Cauchy approximation theorem)
Suppose that some data points come from a function f on $[a, b]$. Then

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0) \cdots (x - x_n),$$

where $\xi \in (a, b)$ and p_n is the interpolated polynomial.

February 15, 2023

Proof for the case $n = 1$. Let $p_1 \in \mathbb{P}[x_0, x_1]$. Interpolate $f \in C^2[a, b]$ at $\{x_0, x_1\}$. For some $\lambda \in \mathbb{R}$, we may write

$$q(x) = p_1(x) + \lambda(x - x_0)(x - x_1).$$

Fix $\widehat{x} \in (x_0, x_1)$ and set $q(\widehat{x}) = f(\widehat{x})$. These functions look something like Figure 3. Therefore

$$\lambda = \frac{f(\widehat{x}) - p_1(\widehat{x})}{(\widehat{x} - x_0)(\widehat{x} - x_1)}.$$

Define $e(x) = f(x) - q(x)$.

- e has three zeros in $[x_0, x_1]$.
- e' has two roots in (x_0, x_1) by Rolle's theorem.

- e'' has one root in (x_0, x_1).

Now define $\xi \in (x_0, x_1)$ s.t. $e''(\xi) = 0$. Thus

$$\begin{aligned} 0 &= f''(\xi) - p_1''(\xi) - \lambda \frac{d^2}{d\xi^2}(\xi - x_0)(\xi - x_1) \\ &= f''(\xi) - 2\lambda. \end{aligned}$$

This means $\lambda = \frac{1}{2} f''(\xi)$.

$$f(\widehat{x}) - p_1(\widehat{x}) = \lambda(\widehat{x} - x_0)(\widehat{x} - x_1) = \frac{f''(\xi)}{2}(\widehat{x} - x_0)(\widehat{x} - x_1). \qquad \square$$

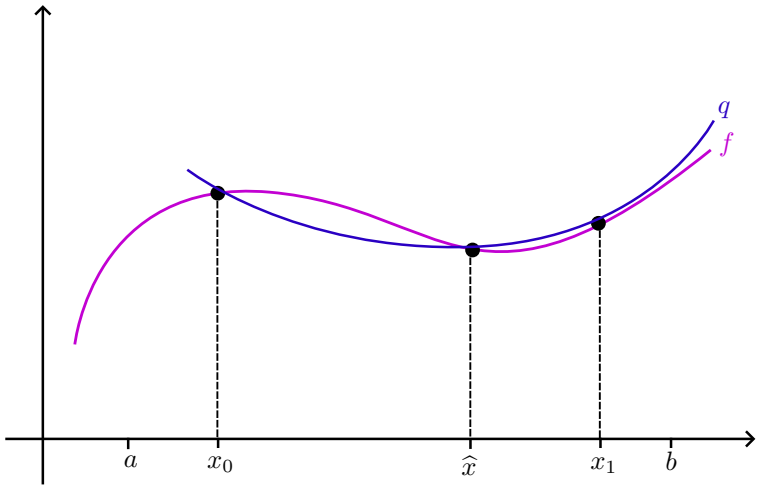

Figure 3: Cauchy approximation theorem diagram

We can generalize this proof to $[a, b]$ and to the stated equation in the theorem.

Corollary 3.6

We may bound

$$|f(x) - p_n(x)| \leq \frac{M_{n+1}}{(n+1)!} \max_{x \in [a,b]} |(x - x_0) \cdots (x - x_0)|$$

where

$$M_{n+1} = \max_{\xi \in (a,b)} \left| f^{(n+1)}(\xi) \right|.$$

To get a good approximation, we want $|(x - x_0) \cdots (x - x_0)|$ to be small. Finding a way to make this small motivates a future discussion about *Chebyshev nodes/polynomials*.

3.4. Hermite interpolation

Hermite interpolation uses facts about $f(x)$ and $f'(x)$ at certain points to create our polynomial and interpolate f. Our goal is to find a polynomial that matches $f(x)$ and $f'(x)$ at ponints $x_0, \ldots, x_n$. Since there are $2n + 2$ constraints we try to fit polynomial p of degree $2n + 1$ with the conditions

$$p_{2n+1}(x_i) = y_i, \quad p'_{2n+1}(x_i) = z_i.$$

Recall the *Lagrange polynomials* $L_k(x)$ Equation 3.1. Define

$$H_k(x) = [L_k(x)]^2 \left(1 - 2L_k'(x_k)(x - x_k)\right), \tag{3.2}$$

$$K_k(x) = [L_k(x)]^2 (x - x_k). \tag{3.3}$$

We can show

$$H_k(x_i) = \begin{cases} 0 & \text{if } i \neq k, \\ 1 & \text{if } i = k. \end{cases} \quad K_k(x_i) = 0, \qquad H_k'(x_i) = 0 \quad K_k's(x_i) = \begin{cases} 0 & \text{if } i \neq k, \\ 1 & \text{if } i = k. \end{cases}$$

Hence

$$p_{2n+1}(x) = \sum_{k=0}^{n} (y_k H_k(x) + z_k K_k(x)). \tag{3.4}$$

Equation 3.4 is (1) unique and (2) has an approximation error:

$$f(x) - p_{2n+1}(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!}((x - x_0) \cdots (x - x_n))^2$$

3.5. Linear least squares

February 17, 2023

If some data has an element of noise, an exact Lagrange interpolation may be overfitted, and a low order polynomial fit may be better suited for it.

Recall that before, the number of data points m *equaled* the number of polynomial coefficients n. For least squares, we reduce the order of the polynomial n, so that $m > n$. Now our equation

$$A\mathbf{b} = \mathbf{y}$$

has A as a "tall thin rectangular matrix". While we cannot solve it exactly, we minimize the residual:

$$r(\mathbf{b}) \coloneqq \mathbf{y} - A\mathbf{b} \in \mathbb{R}^m,$$

specifically, we want to minimize the 2-norm of r:

$$\|r(\mathbf{b})\|_2 = \left(\sum_{i=1}^{m} r_i(\mathbf{b})^2 \right)^{\frac{1}{2}},$$

since it gives us a differentiable function.

Define $\phi(\mathbf{b}) = \|r(\mathbf{b})\|_2^2$.

$$\begin{aligned} \phi(\mathbf{b}) &= \|r\|_2^2 \\ &= r^T r \\ &= (\mathbf{y} - A\mathbf{b})^T (\mathbf{y} - A\mathbf{b}) \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T A\mathbf{b} - \mathbf{b}^T A^T \mathbf{y} + \mathbf{b}^T A^T A\mathbf{b} && ((\mathbf{y}^T A\mathbf{b})^T = \mathbf{b}^T A^T \mathbf{y}) \\ &= \mathbf{y}^T\mathbf{y} - 2\mathbf{b}^T A^T \mathbf{y} + \mathbf{b}^T A^T A\mathbf{b}. \end{aligned}$$

A minimum must exist, but it need not be unique. To find the minimum, differentiate with respect to $\mathbf{b}$. Define $\mathbf{c} \coloneqq A^T\mathbf{y}$. Since

$$\frac{\partial}{\partial b_i}(\mathbf{b}^T\mathbf{c}) = c_i,$$

$$\nabla_{\mathbf{b}}(\mathbf{b}^T A^T \mathbf{y}) = A^T\mathbf{y}.$$

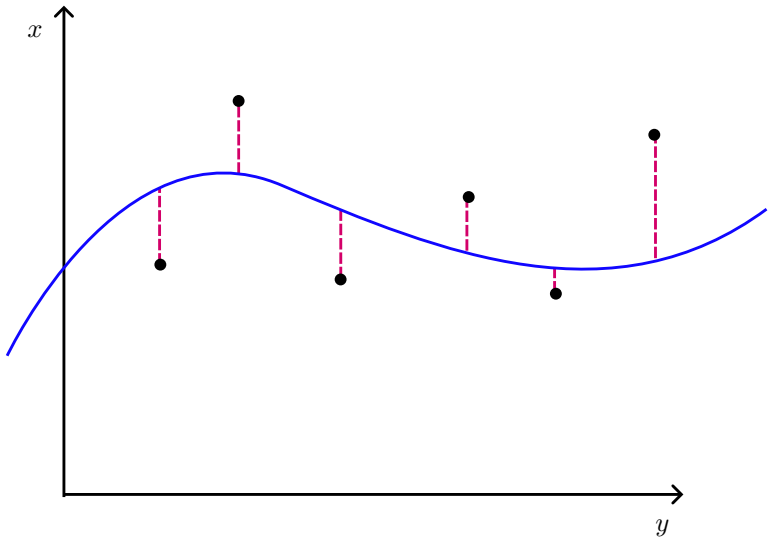

Figure 4: Linear least squares attempts to create a function that minimizes the squares of the red dashed lines.

Now we consider $\nabla_{\mathbf{b}}(\mathbf{b}^T A^T A\mathbf{b})$, and define $M = A^T A$. Note that M is symmetric.

$$\mathbf{b}^T M\mathbf{b} = \mathbf{b}^T \left(\sum_{j=1}^{n} \underbrace{M_{\bullet,j}}_{j\text{th column of } M} b_j \right).$$

Then

$$\frac{\partial}{\partial b_k}(\mathbf{b}^T M\mathbf{b}) = e_k^T \left(\sum_{j=1}^{n} M_{\bullet,j} b_j \right) + \mathbf{b}^T \left(\sum_{j=1}^{n} M_{\bullet,j} \underbrace{\frac{\partial b_j}{\partial b_k}}_{\delta_{j,k}} \right) = \sum_{j=1}^{n} M_{k,j} b_j + \mathbf{b}^T M_{\bullet,k} = 2M_{k,\bullet}\mathbf{b}.$$

where e_k is the kth unit vector. Thus

$$\nabla_{\mathbf{b}}(\mathbf{b}^T A^T A b) = 2A^T A\mathbf{b} \implies \nabla\phi(\mathbf{b}) = -2A^T\mathbf{y} + 2A^T A\mathbf{b}.$$

Solving for 0, we end up with the **normal equations**:

$$A^T A\mathbf{b} = A^T\mathbf{y}. \tag{3.5}$$

We can solve the normal equations directly, but it is not numerically stable as other methods. For exmaple, the `lstsq` function in numpy is more stable.

3.5.1. Non-polynomial least squares fitting

February 20, 2023

Example 3.7 – Approximate $e^{-x}\cos 4x$ using

$$f_n(x; b) = \sum_{k=-n}^{n} b_k e^{kx}.$$

Note that f_n is linear in b. In this case, the matrix looks like

$$\begin{bmatrix} e^{-2x_0} & e^{-x_0} & 1 & e^{x_0} & x^{2x_0} \\ e^{-2x_1} & e^{-x_1} & 1 & e^{x_1} & x^{2x_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{-2x_{n-1}} & e^{-x_{n-1}} & 1 & e^{x_{n-1}} & e^{2x_{n-1}} \end{bmatrix} \begin{bmatrix} b_{-2} \\ b_{-1} \\ b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1}. \end{bmatrix}$$

We cannot use the same method if the function is not linear in the coefficients b, but for functions that behave as follows,

$$f_n(x; \gamma a + \lambda b) = \gamma f_n(x; a) + \lambda f_n(x; b),$$

we can run a least-squares fit on them.

3.5.2. Pseudoinverse matrix

We want to generalize the inverse

Definition 3.5 (Pseudoinverse matirx)
Let $A \in \mathbb{R}^{m \times n}$. Then the **pseudoinverse** is defined as

$$A^+ = (A^T A)^{-1} A^T \in \mathbb{R}^{n \times m}.$$

Then $\mathbf{b} = A^+ \mathbf{y}$ is another way of finding the least squares coefficients with Equation 3.5. $A^+ A = I$, but $AA^+ \neq I$ in general. The pseudoinverse may be called the *left inverse*.

3.5.3. Undetermined least squares

We consider the final case where our matrix is short and wide. We can follow the same derivation as the normal equations.

$$A^T A \mathbf{b} = A^T \mathbf{y}.$$

But in this case, $A^T A \in \mathbb{R}^{n \times n}$ has rank at most m ($m < n$). Hence $A^T A$ is singular (a square matrix without an inverse), and we have infinitely many choices for $\mathbf{b}$.

Because of this, we may formulate this as an optimization problem, so let's try to minimize $\mathbf{b}^T \mathbf{b}$. This is solved by $\mathbf{b} = A^T (A^T)^{-1} \mathbf{y}$. We can redefine the pseudoinverse,

$$A^+ = A^T (A A^T)^{-1},$$

but this is a *right inverse*.

Remark 3.8 (Regularization)**.** Modify $\phi(\mathbf{b})$ to obtain a unique maximum by letting

$$\phi(\mathbf{b}) = \|r(\mathbf{b})\|_2^2 + \|S\mathbf{b}\|_2^2, \tag{3.6}$$

where $S \in \mathbb{R}^{n \times m}$ is a *scaling matrix*. This is called **regularization**. The equivalent normal equations for this is

$$(A^T A + S^T S)\mathbf{b} = A^T \mathbf{y}. \tag{3.7}$$

If $S^T S$ is positive definition, then $(A^T A + S^T S)$ is invertible. We can use regularization to prioritize different conditions.

4. Calculus

4.1. Differentiation: Finite difference approximation

February 22, 2023

We can approximate using nearby points:

forward difference	backwards difference	centered difference
$f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h}$	$f'(x_0) \approx \frac{f(x_0)-f(x_0-h)}{h}$	$f'(x_0) \approx \frac{f(x_0+h)-f(x_0-h)}{2h}$

Table 1: Difference approximations

From Taylor series, we find

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

This is first order accurate, so abs. error is $O(h)$. The centered difference results in

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

So the abs. error $O(h^2)$.

4.1.1. General procedures: Taylor series

Consider the Taylor series of f centered at x at $x, x+h, x+2h$:

$$f(x) = f(x) + 0f'(x) + 0f''(x) + O(h^3),$$

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + O(h^3),$$

$$f(x+2h) = f(x) + 2hf'(x) + 2h^2f''(x) + O(h^3).$$

We want to find a finite difference approximation s.t.

$$f'_{\text{tay}}(x) = \alpha f(x) + \beta f(x+h) + \gamma f(x+2h)$$

and

$$f'_{\text{tay}}(x) = f'(x) + O(h^2).$$

Solving for α, β, γ, we find a new finite difference approx.:

$$f'_{\text{tay}}(x) = \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h}.$$

4.1.2. General procedures: Lagrange interpolant

Instead consider the Lagrange polynomial $p_2(x)$ through $x, x+h, x+2h$ and find $p_2'(x)$. To simplify, change variables to $x \mapsto z + x$, so the three basis functions are at $z = 0, h, 2h$: L_0, L_1, L_2. The Lagrange interpolant is

$$L = f(x)L_0 + f(x+h)L_1 + f(x+2h)L_2.$$

$L'(0)$ is the same as with the Taylor series.

Remark 4.1. In general, the Lagrange interpolant method can get you an answer directly, which may be easier to calculate by hand, whereas the Taylor series method gives you a linear system, which computers can solve very quickly.

4.2. Integration: Quadrature schemes

February 24, 2023

The idea of numerical integration is to get a polynomial approximation of our function and then integrate that polynomial (which we know how to do). For example, we can integrate a Lagrange interpolation of the desired function.

$$\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b p_n(x)\,\mathrm{d}x = \sum_{k=0}^{n} w_k f(x_k), \tag{4.1}$$

where

$$w_k \coloneqq \int_a^b L_k(x)\,\mathrm{d}x \tag{4.2}$$

are the **quadrature weights**, and x_k are the **quadrature points (abscissae)**. The set of all $\{x_k, w_k\}$ are the **quadrature scheme**.

This means that we don't even have to integrate our polynomials, we simply have to evaluate our function at several points and assign our quadrature weights accordingly.

If x_i's are equally spaced, this is the *Newton-Cotes formula of order* n.

Example 4.2 (Newton-Cotes in the case $n = 1$) – Let $x_0 = a$, $x_0 = b$. Then

$$p_1(x) = L_0(x)f(a) + L_1(x)f(b) = \frac{1}{b-a}[(b-x)f(a) + (x-a)f(b)].$$

$$\int_a^b p_1(x)\,\mathrm{d}x = \frac{b-a}{2}[f(a) + f(b)].$$

This is the *trapezoid approximation*.

4.2.1. Composite quadrature schemes

Divide $[a, b]$ into n intervals $[x_i, x_{i+1}]$, $0 \le i \le n-1$. Then

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x$$

If we let $h = \frac{b-a}{n}, x_i = a + ih$,

$$\approx \frac{h}{2}(f_0 + f_1) + \cdots + \frac{h}{2}(f_{m-1} + f_m) = \frac{h}{2}(f_0 + f_m) + h\sum_{i=1}^{m-1} f_i.$$

Example 4.3 (Newton-Cotes in the case $n = 2$) –

$$\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b f(a)L_0(x) + f\left(\frac{a+b}{2}\right)L_1(x) + f(b)L_2(x)\,\mathrm{d}x.$$

Examining the quadrature weights:

$$w_1 = \int_a^b L_1(x)\,\mathrm{d}x = \frac{4}{6}(b-a).$$

Instead of calculating the other two, we note

$$w_0 + w_1 + w_2 = \int_a^b 1\,\mathrm{d}x = b - a$$

By symmetry, $w_0 = w_2$, so

$$w_0 = w_2 = \frac{b-a}{6}.$$

Thus

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]. \tag{4.3}$$

This is also called *Simpson's rule*.

Example 4.4 (Composite Simpson's rule) –

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{i=0}^{m-1}\int_{x_i}^{x_i+1} f(x)\,\mathrm{d}x$$

Let $h \coloneqq \frac{b-a}{6m}$. Then

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{m-1}\frac{h}{6}\left[f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right]$$

Define $M \coloneqq 2m$, $h \coloneqq \frac{b-a}{M}$, and $x_i \coloneqq a + ih$. Then we may rewrite the formula as

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{3}\sum_{i=0}^{\frac{M}{2}-1}\left[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i}+2)\right]. \tag{4.4}$$

4.2.2. Error analysis of integration techniques

Consider a general quadrature scheme on $[a, b]$:

$$p_n(x) = \sum_{k=0}^{n} f(x_k)L_k(x), \quad x_k \coloneqq a + kh, \quad h \coloneqq \frac{b-a}{n}.$$

We have

$$\int_a^b f(x)\,\mathrm{d}x = \left(\sum_{k=0}^{n} w_k f(x_k)\right) + E_n(f),$$

where

$$E_n(f) \coloneqq \int_a^b f(x) - p_n(x)\,\mathrm{d}x.$$

By the Cauchy approximation theorem,

$$\begin{aligned}
|E_n(f)| &= \left|\int_a^b f(x) - p_n(x)\,\mathrm{d}x\right| \\
&\leq \int_a^b |f(x) - p_n(x)|\,\mathrm{d}x \\
&\leq \frac{M_{n+1}}{(n+1)!}\int_a^b |\pi_{n+1}(x)|\,\mathrm{d}x
\end{aligned}$$

Example 4.5 (Error analysis of trapezoid rule) –

$$\begin{aligned}|E_1(f)| &\leq \frac{M_2}{2}\int_a^b |(x-a)(x-b)|\,\mathrm{d}x \\ &= \frac{M_2}{2}\int_a^b (x-a)(b-x)\,\mathrm{d}x \\ &= \frac{M_2}{12}\cdot h^3.\end{aligned}$$

February 27, 2023

Example 4.6 (Composite trapezoid rule error) – Let $f_i \coloneqq f(x_i)$, $x_i = a + ih$, $h = \frac{b-a}{m}$. The error can be written as

$$\begin{aligned}E_1(f) &= \int_a^b f(x)\,\mathrm{d}x - h\left(\frac{1}{2}f_0 + f_1 + \cdots + f_{m-1} + \frac{1}{2}f_m\right) \\ &= \sum_{i=1}^{m}\left(\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x - \frac{h}{2}(f_{i-1}+f_i)\right).\end{aligned}$$

The term inside the sum is just the trapezoid method error. Thus

$$\begin{aligned}|E_1(f)| &\leq \sum_{i=1}^{m}\frac{h^3}{12}\max_{\xi\in[x_{i-1},x_i]}|f''(\xi)| \\ &\leq \sum_{i=1}^{m}\frac{h^3}{12}M_2 \\ &= \frac{b-a}{12}M_2h^2.\end{aligned}$$

Then the composite trapezoid rule is second-order accurate, $\text{Error} = O(h^2)$.

Remark 4.7. We note that $|E_1(f)|$ is large when $|f''(\xi)|$ is large. The adaptive quadrature puts more sample points where $|f''(\xi)|$ is large to minimize the error there.

Example 4.8 (Composite Simpson's rule error) –

$$\begin{aligned}|E_2(f)| &\leq \frac{M_3}{6}\int_a^b \left|(x-a)\left(x-\frac{a+b}{2}\right)(x-b)\right|\mathrm{d}x \\ &= \frac{(b-a)^4}{196}M_3\end{aligned}$$

for a single interval. Following a similar proof to the previous example, for the composite rule, we find

$$|E_2(f)| \leq \frac{b-a}{196}M_3h^3.$$

Remark 4.9. However, when we test `quadrat.py` on $\int_1^5 \sin x\,\mathrm{d}x$, we find that error scales more like $O(h^4)$. This is because there is a sharper bound:

$$|E_2(f)| \leq \frac{b-a}{2880}M_4h^4,$$

which is fourth-order. For a proof of this, see Süli-Mayers. This means that when h is halved, the error is divided by 16(!).

5. Polynomial approximation

5.1. Function space norms

Consider $C[a,b]$, the set of continuous functions on $[a,b]$. We may define norms on this space:

We may write max instead of inf because f is continuous on a compact set, so it achieves its maximum

$$\|f\|_p := \left(\int_a^b |f(x)|^p \,\mathrm{d}x\right)^{\frac{1}{p}}, \qquad \|f\|_\infty := \max_{x\in[a,b]} |f(x)|.$$

Remark 5.1. Depending on the norm we focus on, there is a large difference in what we find. Consider $f(x) = \frac{10^{-6}}{x^{1/4}}$. We can calculate that

$$\|f\|_\infty = \infty, \quad \|f\|_2 = \left(\int_0^1 \frac{(10^{-6})^2}{x^{1/2}} \,\mathrm{d}x\right) = 10^{-6}\cdot\sqrt{2}.$$

Remark 5.2 (Weighted norms)**.** We may define a *weighted norm* as

$$\|f\|_2 = \left(\int_a^b w(x) f(x)^2 \,\mathrm{d}x\right)^{\frac{1}{2}}, \quad w(x) > 0.$$

For example, if we let $a = -1$, $b = 1$, and $w(x) = \frac{1}{\sqrt{1-x^2}}$, the norm represents the function more at -1 and 1.

Theorem 5.3 (Weierstass approximation theorem)
Given $f \in C[a,b]$, for all $\varepsilon > 0$, we can find a polynomial p such that

$$\|f - p\|_\infty < \varepsilon.$$

The proof that this construction works is not trivial, but we will not prove it here. It is in my Analysis I notes.

Construction. Consider $C[0,1]$. We can define

$$p_n(x) := \sum_{k=0}^{n} p_{nk}(x) f\left(\frac{k}{n}\right), \quad p_{nk} := \binom{n}{k} x^k (1-x)^{n-k}. \qquad \square$$

These are called *Berstein polynomials*.

The Weierstrass approximation is usually only good in theory though, as it can get very slow to coverge within a certain distance of a function.

5.2. Minimizing the infinity norm for polynomials

March 1, 2023

Is there a degree n polynomial p_n such that $\|f - p_n\|_\infty$ is minimized? We define the function

$$\rho_n(f) := \inf_{q\in\mathcal{P}_n} \|f - q\|_\infty = \|f - p_n\|_\infty.$$

This is the minimax error.

Theorem 5.4 (Polynomial minimizing infinity norm of each degree exists)
Given $f \in C[a,b]$ there exists $p_n \in \mathcal{P}_n$ such that

Since we write min here, we imply that p_n acutally exists

$$\|f - p_n\|_\infty = \min_{q\in\mathcal{P}_n} \|f - q\|_\infty.$$

Proof sketch. Define $E\colon \mathbb{R}^{n+1} \to \mathbb{R}$ such that

$$E(c_0, \dots, c_n) \coloneqq \|f - q_n\|_\infty, \qquad q_n \coloneqq c_0 + \cdots + c_n x^n.$$

We require two claims:

Claim 5.1. E is continuous.

Claim 5.2. There is a nonempty, bounded, closed set $\mathcal{S}$ such that the lower bound of E on $\mathcal{S}$ is the same as its lower bound on $\mathbb{R}^{n+1}$.

Combining these two, since $\mathcal{S}$ is compact, E hits its minimum on $\mathcal{S}$, proving the theorem. □

There is a problem with Theorem 5.4: it is not *constructive*; we have no way of finding the polynomial p_n.

Example 5.5 – Let's try and construct them for small values of n.

- **Case 1** ($n = 0$): Consider $f \in C[a, b]$ for simplicity. Define $p_n(x) = c_0$. Since f is continuous on a compact set, it hits its maximum and minimum. Let

  $$f(\xi) = \max f, \qquad f(\eta) = \min f.$$

  Then

  $$c_0 = \frac{1}{2}\left(f(\xi) + f(\eta)\right).$$

- **Case 2** ($n = 1$): We will consider $f(x) = x^2$ on $[0, 1]$. Let $p_1 = c_0, c_1 x$. Then the minimax error is

  $$\rho_1(f) = \left\|x^2 - (c_0 + c_1 x)\right\|_\infty.$$

  By analyzing the derivative or the vertex of this parabola, it has maxima at $x = 0, \frac{c_1}{2}, 1$. Thus

  $$\begin{aligned}\|f - p_1\|_\infty &= \max\left\{|f(0) - p(0)|, \left|f\left(\frac{c_1}{2}\right) - p - 1\left(\frac{c_1}{2}\right)\right|, |f(1) - p_1(1)|\right\} \\ &= \max\left\{|c_0|, |1 - c_0 - c_1|, \left|\frac{-c_1^2}{4} - c_0\right|\right\}.\end{aligned}$$

  ρ_1 is minimized when the maxima have the same value. Let $\rho_1(f) = a$. To remove absolute values, we square everything:

  $$c_0^2 = a^2, \quad (1 - c_0 - c_1)^2 = a^2, \quad \left(\frac{c_1^2}{4} + c_0\right)^2 = a^2.$$

  This has a solution, $p_1(x) = -\frac{1}{8} + x \implies \rho_1(f) = \frac{1}{8}$.

While we could continue for higher n, it will get much more complicated, so we seek more general approaches.

Theorem 5.6 (De la Valleé Poussin)
Let $f \in C[a,b]$, $r \in \mathcal{P}_n$. Consider $n+2$ points such that

$$a \leq x_0 < x_1 < \cdots < x_{n+1} \leq b.$$

such that $f(x_i) - r(x_i)$ and $f(x_{i+1}) - r(x_{i+1})$ have opposite signs for $0 \leq i \leq n$. Then

$$\rho_n(f) = \min_{q \in \mathcal{P}_n} \|f - q\|_\infty \geq \min_i |f(x_i) - r(x_i)|.$$

Proof. Suppose the conclusion is false for contradiciton. Thus

$$|q(x_i) - f(x_i)| < |r(x_i) - f(x_i)|, \qquad 0 \leq i \leq n+1. \tag{5.1}$$

$r(x_i) - q(x_i) = (r(x_i) - f(x_i)) - (q(x_i) - f(x-i))$. However, from 5.1, we have that the second term is smaller in magnitude than the first. Thus $r - q$ also changes signs $n+1$ times. So $r - q$ has $n+1$ roots. However, $r - q \in \mathcal{P}_n$, so $r - q = 0 \implies r = q$. This means that 5.1 is an equality, which is a contradiction. □

Definition 5.1 (Minimax polynomial)
The polynomial q that minimizes $\|f - q\|_\infty$ is called a **minimax polynomial**.

This is theorem 8.4 in Süli-Mayers

Theorem 5.7 (Oscillation theorem)
Given $f \in C[a,b]$, $r \in \mathcal{P}_n$ is a minimax polynomial for f if and only if there exists a sequence of $n+2$ points $x_0, \ldots, x_{n+1}$

$$a \leq x_0 < x_1 < \cdots < x_{n+1} \leq b$$

such that

$$|f(x_i) - r(x_i)| = \|f - r\|_\infty, \quad 0 \leq i \leq n+1$$

and

$$f(x_i) - r(x_i) = -\left(f(x_{i+1}) - r(x_{i+1})\right).$$

Theorem 5.8 (Uniqueness theorem)
Each $f \in C[a,b]$ has a unique minimax polynomial $p_n \in \mathcal{P}_n$.

Proof. Suppose that $q_n \neq p_n$ is also a minimax polynomial. Then

$$\|f - p_n\|_\infty = \|f - q_n\|_\infty = E_n(f).$$

Consider $\frac{1}{2}(p_n + q_n)$. We can show

$$\left\| f - \frac{1}{2}(p_n + q_n) \right\| \leq E_n(f),$$

so this is a minimax polynomial as well. By Theorem 5.7, we can find $\{x_0, \ldots, x_{n+1}\} \subseteq$

$[a, b]$ so that

$$\left| f(x_i) - \frac{1}{2}(p_n(x_i) - q_n(x_i)) \right| = E_n(f),$$

$$\implies |(f(x_i) - p_n(x_i)) + (f(x_i) - q_n(x_i))| = 2E_n(f). \tag{5.2}$$

We also know

$$|f(x_i) - p_n(x_i)| \leq \max_{x\in[a,b]} |f(x) - p_n(x)| = \|f - p_n\|_\infty = E_n(f). \tag{5.3}$$

Similarly,

$$|f(x_i) - q_n(x_i)| \leq E_n(f). \tag{5.4}$$

Add 5.3, 5.4 and compare the result to 5.2.

This implies that $f(x_i) - p_n(x_i) = f(x_i) - q_n(x_i)$. Thus $p_n(x) - q_n(x) = 0$ at $n + 2$ points. Since $p_n - q_n \in \mathcal{P}_n$, $p_n - q_n = 0$. □

We have no way of finding the minimax polynomial for a broad class of functions. We therefore try lifting some restrictions.

5.3. Chevyshev polynomials

March 5, 2023

Definition 5.2 (Chebyshev polynomials)

On the interval $[-1, 1]$, the **Chebyshev polynomials** are defined as

$$T_n(x) \coloneqq \cos(n \cos^{-1} x).$$

Remark 5.9. While it does not seem like this formula will create polynomials, we can show that T_0 and T_1 are both polynomials, and one of the following facts will inductively tell us that, inductively, all T_n are polynomials.

Proposition 5.10 (Properties of Chebyshev polynomials)

The Chebyshev polynomials satisfy the following properties.

- $T_{n+1}(x) = 2xT_n(x) + T_{n-1}(x)$
- T_n is even if and only if n is even (same for odd)
- $|T_n(x)| \leq 1$, $x \in [-1, 1]$
- $T_n(x) = 0 \iff x = x_j = \cos\left(\frac{(2j-1)\pi}{2n}\right)$, $j = 1, \ldots, n$.
- $T_n(x) = \pm 1$ at $n + 1$ points: $x = y_k = \cos\frac{k\pi}{n}$
- The leading coefficient of $T_n(x)$ is 2^{n-1} for $n \geq 1$.

We care about this polynomials because they may give a good approximation of the minimax, or the best!

Example 5.11 (Chebyshev gives best approximation) – We approximate $f(x) = x^3$ by $p_2 \in \mathcal{P}_2$ on $[-1, 1]$.

$$x^3 - \frac{1}{2^2}T_3(x) = \frac{3}{4}x \in \mathcal{P}_2 \implies f(x) - \frac{3}{4}x = \frac{1}{4}T_3(x).$$

By examining the graph, the error is $\frac{1}{4}$ at $n+2 = 4$ points. Thus $\frac{3}{4}x$ *is* the minimax polynomial by Theorem 5.7.

Generally, the minimax polynomial of $f(x) = x^{n+1}$ is

$$p_n(x) \coloneqq x^{n+1} - \frac{1}{2^n} T_{n+1}(x).$$

We gain something from this example, but first we need to define a specific space.

Definition 5.3 (Monic polynomials)
Monic polynomials have leading coefficient 1. Let the set of all polynomials less than or equal to degree n that are monic be $\mathcal{P}_n^1$.

Corollary 5.12
Suppose $n \geq 0$. Then among all $\mathcal{P}_{n+1}^1$, the polynomial $2^{-n}T_{n+1}$ has the smallest ∞-norm on $[-1, 1]$ against x^{n+1}.

Proof. Let $r \in \mathcal{P}_{n+1}^1$. This can be written as

$$r(x) = x^{n+1} - q(x), \quad q \in \mathcal{P}_n.$$

Then

$$\begin{aligned} \min_{r \in \mathcal{P}_{n+1}^1} \|r\|_\infty &= \min_{q \in \mathcal{P}_n} \left\|x^{n+1} - q\right\|_\infty \\ &= \left\|x^{n+1} - (x^{n+1} - 2^{-n}T_{n+1})\right\|_\infty \\ &= \left\|2^{-n}T_{n+1}\right\|_\infty. \end{aligned}$$

Hence the minimum is achieved when $r(x) = 2^{-n}T_{n+1}$ □

5.3.1. Using Chebyshev polynomials in Lagrange interpolation

Recall that

$$p(x) \coloneqq \sum_{k=0}^{n} f(x_k) L_k(x) \in \mathcal{P}_n,$$

where

$$L_k(x) \coloneqq \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)}.$$

The error is given by

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \pi_{n+1}(x),$$

where

$$\pi_{n+1}(x) \coloneqq (x - x_0) \cdots (x - x_n).$$

We note that π_{n+1} is a *monic* polynomial. To minimize $\|\pi_{n+1}\|_\infty$, use Corollary 5.12. The smallest ∞-norm is achieved with $2^{-n}T_{n+1}(x)$. By choosing

$$x_j = \cos\left(\frac{(2j-1)\pi}{2(n+1)}\right), \qquad j = 1, \ldots, n+1,$$

$\pi_{n+1} = 2^{-n}T_{n+1}(x)$.

Remark 5.13. By minimizing $\|\pi_{n+1}\|_\infty$, we are not finding the best polynomial fit. However, this tends to be a very good choice to reduce error.

In general, for $f \in C^n[a,b]$,

$$p_n(x) = \sum_{k=0}^{n} f(x_k) L_k(x)$$

Of course, M_{n+1} is defined with the $n+1$th derivative, so bound only exists when $f \in C^{n+1}[a,b]$.

can have a low error by letting the x_k's be Chebyshev nodes:

$$x_k := \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left(\frac{2k-1}{2(n+1)}\right), \qquad k = 1, \dots, n+1.$$

Thus

$$\|f - p_n\|_\infty \leq \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!} M_{n+1}.$$

There is example code in `cheb_converge.py`.

5.4. Working in the 2-norm

5.4.1. Inner product spaces

March 6, 2023

Recall that a norm gives us a way to measure distance. We want an analog of angle, or orthogonality.

Definition 5.4 (Inner product)

An **inner product** $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ satisfies four axioms:

1. $\langle f+g, h\rangle = \langle f, h\rangle + \langle g, h\rangle$
2. $\langle \lambda f, g\rangle = \lambda \langle f, g\rangle$
3. $\langle f, g\rangle = \langle g, f\rangle$
4. $\langle f, f\rangle > 0$ if $f \neq 0$.

Remark 5.14. The inner product induces a norm: $\|u\| = \sqrt{\langle u, u\rangle}$.

Example 5.15 – The vector scalar product (dot product) on $\mathbb{R}^n$ is an inner product. Its induced norm is the Euclidean norm.

Theorem 5.16 (Cauchy-Schwarz inequality)

$$|\langle f, g\rangle| \leq \|f\| \|g\|, \qquad \forall f, g \in V.$$

Proof. We find

$$\|\lambda f + g\|^2 = \lambda^2 \|f\|^2 + 2\lambda \langle f, g\rangle + \|g\|^2.$$

If we let this a polynomial on λ, we know that it cannot have two distinct roots $\iff$ the discriminant is ≤ 0. By writing it out and simplifying, the inequality is proven. □

Definition 5.5 (Inner product space)
A linear space equipped with an inner product is called an **inner product space**.

Example 5.17 (Inner product spaces examples) – The following are inner product spaces:

- Let $V = C[a,b]$. Define
$$\langle f, g\rangle \coloneqq \int_a^b w(x) f(x) g(x)\,\mathrm{d}x,$$
where w is a weight function.
- Let $V = L^2(a,b) = \{f \mid f \text{ defined on } [a,b],\ \|f\|_2 < \infty\}$.

5.4.2. Best approximation in the 2-norm

Let $f \in L^2(a,b)$. We want to find $p_n \in \mathcal{P}_n$ s.t.

$$\|f - p_n\|_2 \leq \inf_{q\in\mathcal{P}_n} \|f - q\|_2 .$$

Example 5.18 ($n = 0$) – Let $f(x) = x^{-1/4}$ on $[0,1]$ with $w(x) = 1$, $n = 0$. We want

$$\begin{aligned}\|f - c\|_2^2 &= \int_0^1 (x^{-1/4} - c)^2\,\mathrm{d}x \\ &= 2 - \frac{8c}{3} + c^2.\end{aligned}$$

This is minimized at $c = \frac{4}{3}$.

To proceed for general n, the algorithms are numerically unstable. Consider

$$p_n(x) = \sum_{k=0}^{n} b_k x^k.$$

$\{1, x, \ldots, x^n\}$ are not a good bases. This is similar to the Vandermonde issue. They are not *orthogonal*. Can we construct an orthogonal basis?

5.4.3. Gram-Schmidt orthogonalization

Gram-Schmidt orthogonalization gives a way to construct an orthogonal basis $\{\phi_1, \ldots, \phi_n\}$ for a space given any basis $\{a_1, \ldots, a_n\}$. The process is as follows:

- Let $\boxed{\phi_1 = a_1}$.
- Then let $\phi_2 = a_2 - c\phi_1$, so that $\langle \phi_1, \phi_2\rangle = 0$. We calculate that
$$c = \frac{\langle \phi_1, a_2\rangle}{\langle \phi_1, \phi_1\rangle} \implies \boxed{\phi_2 = a_2 - \frac{\langle \phi_1, a_2\rangle}{\langle \phi_1, \phi_1\rangle}\phi_1}.$$
The algebra becomes simpler is ϕ_k is normalized at each step. Then $\langle \phi_k, \phi_k\rangle = 1$.
- $\phi_3 = a_3 - c_1\phi_1 - c_2\phi_2$.
$$\boxed{\phi_3 = a_3 - \frac{\langle \phi_1, a_3\rangle}{\langle \phi_1, \phi_1\rangle}\phi_1 - \frac{\langle \phi_2, a_3\rangle}{\langle \phi_2, \phi_2\rangle}\phi_2}.$$

- For general ϕ_n,

$$\boxed{\phi_n = a_n - \left(\sum_{i=1}^{n-1} \frac{\langle \phi_i, a_n \rangle}{\langle \phi_i, \phi_i \rangle} \phi_i\right)} \tag{5.5}$$

Remark 5.19. We can simplify the algebra by normalizing ϕ_i to ψ_i, so that $\langle \psi_i, \psi_i \rangle = 1$. This will form an orthonormal basis. The formula for ψ_n becomes

$$\psi_n = a_n - \left(\sum_{i=1}^{n-1} \langle \psi_i, a_n \rangle \psi_i\right).$$

March 8, 2023

Example 5.20 (Legendre polynomials) – With this method, consider the basis $\{1, \dots, x^n\}$ for $\mathcal{P}_n$ on $[-1, 1]$. Let $w(x) \equiv 1$.

$$\langle f, g \rangle = \int_{-1}^{1} f(x) g(x) \, \mathrm{d}x.$$

$\phi_1(x) = 1$
$\phi_2(x) = x - c\phi_1(x)$ such that $\int_{-1}^{1} x \, \mathrm{d}x = 0 \implies c = 0$
$\phi_3(x) = x^2 - c_1\phi_1 - c_2\phi_2 = x^2 - \frac{1}{3}$.

These are the Legendre polynomials. We can normalize these so that $\phi_m(1) = 1$:

$$1, x, \frac{1}{2}(3x^2 - 1), \dots$$

or normalize so that $\|\phi_m\| = 1$:

$$\frac{1}{\sqrt{2}}, \sqrt{\frac{3}{2}}x, \sqrt{\frac{5}{4}}(3x^2 - 1), \dots$$

Example 5.21 (Chebyshev polynomials) – We look at a related problem: Find an orthogonal basis for $\mathcal{P}_n$ on $[-1, 1]$ with weight $w(x) = \frac{1}{\sqrt{1-x^2}}$. Now we have

$$\langle f, g \rangle = \int_{-1}^{1} \frac{1}{\sqrt{1 - x^2}} f(x) g(x) \, \mathrm{d}x.$$

Motivated by seeing the derivative of $\cos^{-1}$, we try using the Chebyshev polynomials: $\phi_n(x) = T_n(x) = \cos(n \cos^{-1} x)$. Then (with the substitution $x = \cos\theta \implies dx = -\sqrt{1 - x^2} d\theta$):

$$\begin{aligned} A_{m,n} &\coloneqq \langle T_m, T_n \rangle \\ &= \int_{-1}^{1} \frac{1}{\sqrt{1 - x^2}} \cos(n \cos^{-1} x) \cos(m \cos^{-1} x) \, \mathrm{d}x \\ &= \int_{\pi}^{0} -\cos n\theta \cos m\theta \, \mathrm{d}\theta \\ &= \int_{0}^{\pi} \cos n\theta \cos m\theta \, \mathrm{d}\theta \\ &= \frac{\pi}{2} \delta_{m,n}. \end{aligned}$$

Thus, these polynomials are orthogonal w.r.t. this inner product.

Example 5.22 – Consider $w = e^{-x}$ on $x \in [0, \infty)$. Then

$$\langle f, g \rangle = \int_{0}^{\infty} e^{-x} f(x) g(x) \, \mathrm{d}x.$$

We seek an orthogonal basis $\phi_1, \phi_2, \ldots$, and an orthonormal basis $\psi_1(x), \psi_2(x), \ldots$ with the same span as $1, x, x^2, \ldots$.

- $\phi_1(x) = 1$.
$$\psi_1(x) = \frac{\phi_1(x)}{\|\phi_1\|} = 1.$$
- $\phi_2(x) = x - c\psi_1$. We want
$$0 = \langle \psi_1, x \rangle - v \langle \psi_1, \psi_1 \rangle \implies c = 1.$$
We find $\psi_2(x) = x - 1$ as well.
- $\phi_3(x) = x^2 - 4x + 2$, and $\psi_3 = \frac{1}{2}\phi_3$.

March 10, 2023

Is there a $p_n \in \mathcal{P}_n$ such that $\|f - p_n\|_2 = \inf_{q \in \mathcal{P}_n} \|f - q\|_2$? It turns out there is. It is the projection of f onto $\mathcal{P}_n$. To show this is true, we need the following theorem:

Theorem 5.23 (Orthogonal decomposition theorem)
If V is a linear space, and S is a subspace, then any $f \in V$ may be written as $f = g + g^\perp$, where $g \in S$, and $g^\perp \in S^\perp$. In addition, $\|f\|^2 = \|g\|^2 + \left\|g^\perp\right\|^2$.

Proof. Assume S has dimension n. Let $\{\psi_1, \ldots, \psi_n\}$ be an orthonormal basis for S.

$$g = \sum_{i=1}^{n} \langle f, \psi_i \rangle \psi_i$$

is the projection of f onto S. Let $g^\perp = f - g$. We need to show that $g^\perp \in S^\perp$. Recall that $S^\perp$ is the set of all vectors that are perpendicular to vectors in S. We need to show $\langle g^\perp, \psi_j \rangle = 0$ for $j = 1, \ldots, n$:

$$\begin{aligned}
\langle g^\perp, \psi_j \rangle &= \left\langle f - \sum_{i=1}^{n} \langle f, \psi_i \rangle \psi_i, \psi_j \right\rangle \\
&= \langle f, \psi_j \rangle - \sum_{i=1}^{n} \langle f, \psi_i \rangle \langle \psi_i, \psi_j \rangle \\
&= \langle f, \psi_j \rangle - \langle f, \psi_j \rangle = 0.
\end{aligned}$$

□

Theorem 5.24 (Best approximation in the 2-norm)
The projection

$$g = \sum_{i=1}^{n} \langle f, \psi_i \rangle \psi_i \tag{5.6}$$

satisfies $\|f - g\|_2 \leq \|f - h\|_2, \forall h \in S$.

Proof. For all $h \in S$, we have

$$f - h = (f - g) + (g - h).$$

Note that $f - g \in S^\perp$, $g - h \in S$.

$$\begin{aligned}\|f-h\|^2 &= \langle f-h, f-h\rangle \\ &= \langle f-g+g-h, f-g+g-h\rangle \\ &= \langle f-g, f-g\rangle + 2\langle f-g, g-h\rangle + \langle g-h, g-h\rangle \\ &= \|f-g\|^2 + \|g-h\|^2 .\end{aligned}$$

This implies

$$\|f-h\|^2 \geq \|f-g\|^2 ,$$

as desired. □

Now consider an *orthogonal* basis $\{\varphi_1, \ldots, \varphi_n\}$, such that $\langle \varphi_i, \varphi_j\rangle = A_i \delta_{i,j}$. In this case, the projection is

$$g = \sum_{i=1}^{n} \frac{\langle f, \varphi_i\rangle}{\|\varphi_i\|^2}\varphi_i. \tag{5.7}$$

Example 5.25 (Orthogonal basis best 2-norm approximation) – Let $S = \mathcal{P}_n, x \in [-1, 1], w(x) = 1$. Use the Legendre polynomials:

$$\left\{1, x, x^2 - \frac{1}{3}, \ldots\right\},$$

which are an orthogonal basis. For $f(x) = e^x$, the best approximation in $\mathcal{P}_1$ is

$$p_1(x) = \frac{\langle f, 1\rangle}{\langle 1, 1\rangle}1 + \frac{\langle f, x\rangle}{\langle x, x\rangle}x = \frac{1}{2}(e - e^{-1}) + \frac{3}{e}x.$$

Remark 5.26 (Approximation in Fourier space)**.** Another example of a linear space is Fourier space. Here,

$$V = C[0, 2\pi], \qquad \langle f, g\rangle = \int_0^{2\pi} f(x)g(x)\,\mathrm{d}x,$$

an orthonormal basis could be

$$\frac{1}{\sqrt{2\pi}}, \frac{1}{\sqrt{\pi}}\cos x, \frac{1}{\sqrt{\pi}}\sin x, \frac{1}{\sqrt{\pi}}\cos 2x, \frac{1}{\sqrt{\pi}}\sin 2x, \ldots$$

5.4.4. Quadrature, revisited

Newton-Cotes quadrature looks at equally spaced points to construct the interpolating polynomial. There will be problems when the number of points gets large.

With our knowledge of approximating polynomials in the 2-norm, we put the quadrature points at the Chebyshev nodes. This gives the *Clenshaw-Curtis* quadrature, which is very good in practice.

We look at *Gaussian quadrature*. Consider a general $n+1$-point quadrature scheme with points $\{x_0, \ldots, x_n\}$ and weights $\{w_0, \ldots, w_n\}$. By construction, quadrature will integrate polynomials in $\mathcal{P}_n$ exactly.

March 20, 2023 Recall Hermite approximation for $f(x)$ is

$$p_{2n+1}(x) = \sum_{k=0}^{n} H_k(x)f(x_k) + \sum_{k=0}^{n} K_k(x)f'(x_k),$$

where

$$H_k(x) = [L_k(x)]^2(1 - 2L_k'(x_k)(x - x_k)), \quad K_k(x) = [L_k(x)]^2(x - x_k).$$

Where L_k is a Lagrange interpolant. Thus

$$\int_a^b w(x)f(x)\,\mathrm{d}x \approx \int_a^b w(x)p_{2n+1}(x)\,\mathrm{d}x = \sum_{k=0}^{n}[w_k f(x_k) + v_k f'(x_k)],$$

where

$$w_k = \int_a^b w(x)H_k(x)\,\mathrm{d}x, \qquad v_k = \int_a^b w(x)K_k(x)\,\mathrm{d}x.$$

Theorem 5.27 (Gaussian quadrature of polynomials)
There exists a choice of $\{x_k\}$ and $\{w_k\}$ such that we can integrate polynomials up to degree $2n+1$ *exactly*.

Proof. We want to try and choose our x_k such that $v_k = 0$. Let

$$C_n \coloneqq \frac{1}{(x_k - x_0)\cdots(x_k - x_{k-1})(x_k - x_{k+1})\cdots(x_k - x_n)},$$

which we will use to simplify the equation. We have

$$\begin{aligned} v_k &= \int_a^b w(x)[L_k(x)]^2(x - x_k)\,\mathrm{d}x \\ &= \int_a^b w(x)L_k(x)\frac{(x - x_0)\cdots(x - x_{k-1})(x - x_{k+1})\cdots(x - x_n)(x - x_k)}{(x_k - x_0)\cdots(x_k - x_{k-1})(x_k - x_{k+1})\cdots(x_k - x_n)}\,\mathrm{d}x \\ &= C_n\int_a^b w(x)L_k(x)\pi_{n+1}(x)\,\mathrm{d}x. \end{aligned}$$

Let

$$\{\phi_0, \ldots, \phi_n\}$$

be an orothgonal basis of $\mathcal{P}_n$ w.r.t. $w(x)$. Take $\pi_{n+1}(x) = \phi_{n+1}(x)$. Set x_k to be the zeros of $\phi_{n+1}(x)$. Then

$$\begin{aligned} v_k &= C_n\int_a^b w(x)L_k(x)\pi_{n+1}(x)\,\mathrm{d}x \\ &= C_n\int_a^b w(x)\left[\sum_{j=0}^{n} a_{j,k}\phi_j(x)\right]\pi_{n+1}(x)\,\mathrm{d}x \\ &= 0. \end{aligned}$$

Hence

$$\int_a^b w(x)f(x)\,\mathrm{d}x \approx \int_a^b p_{2n+1}(x)\,\mathrm{d}x = \sum_{k=0}^{n} w_k f(x_k) + \underline{0}.$$ □

Example 5.28 (Gaussian quadrature with weight function) – With $x \in [-1, 1]$ and $w(x) = \frac{1}{\sqrt{1-x^2}}$ and $n = 2$, we have

$$\int_{-1}^{1}\frac{1}{\sqrt{1-x^2}}f(x)\,\mathrm{d}x = \sum_{k=0}^{2} w_k f_k.$$

The orthogonal basis is given by the Chebyshev polynomials $\phi_n(x) = \cos(n\cos^{-1}x)$. The roots

are where $\phi_3(x) = 0$. We find $x_0 = \frac{-\sqrt{3}}{2}$, $x_1 = 0$, $x_2 = \frac{\sqrt{3}}{2}$, and $w_0 = w_1 = w_2 = \frac{\pi}{3}$.

Remark 5.29. We may have also applied Theorem 5.27 by a system of equations:

$$\int_{-1}^{1} 1 \, \mathrm{d}x = w_0 + w_1 = 2,$$

$$\int_{-1}^{1} x \, \mathrm{d}x = w_0 x_0 + w_1 x_1 = 0,$$

$$\int_{-1}^{1} x^2 \, \mathrm{d}x = w_0 x_0^2 + w_1 x_1^2 = \frac{2}{3},$$

$$\int_{-1}^{1} x^3 \, \mathrm{d}x = w_0 x_0^3 + w_1 x_1^3 = 0.$$

6. Piecewise polynomial interpolation

Lagrange/Hermite polynomial interpolation is global: there is one representation over the whole interval. However, with many interpolation points, for example, we found that there were issues with evaluation and/or unwanted oscillations. We try intead to approximate with low order polynomials over many intervals. A **spline** is a degree k polynomial on each subinterval that is differentiable $k-1$ times.

6.1. Linear splines

A *linear spline* is defined by a set of **knots** $K = \{x_0, \ldots, x_n\}$. The linear spline for this set of knots is

$$s_L(x) \coloneqq \frac{x_i - x}{x_i - x_{i-1}} f(x_{i-1}) + \frac{x - x_{i-1}}{x_i - x_{i-1}} f(x_i). \tag{6.1}$$

for $x \in [x_{i-1}, x_i]$, $i = 1, \ldots, n$. We can bound the error on this spline easily when we recognize that the equation for each interval is just a first order Lagrange interpolant. Let $h_i \coloneqq x_i - x_{i-1}$ and $h = \max_i h_i$. If $f \in C^2[a,b]$, and we suppose $x \in [x_{i-1}, x_i]$, there is a $\xi \in [x_{i-1}, x_i]$ so that:

$$f(x) - s_L(x) = \frac{f''(\xi)}{2}\pi_2(x) \implies \|f(x) - s_L(x)\|_\infty \leq \frac{1}{8}h_i^2 f''(\xi).$$

So in general,

$$\|f(x) - s_L(x)\|_\infty \leq \frac{1}{8}h^2 \|f''\|_\infty .$$

Therfore, our linear splines give us an order 2 approximation to functions.

6.2. Cubic splines

Next we look at *cubic splines* using the same set of knots. Since each cubic requires 4 parameters, we have $4n$ constraints total. Suppose that we require the spline be in $C^2[a,b]$.

- The endpoints must match x_i, x_{i-1}. This adds $2n$ constraints.
- First derivatives must match at each interval boundary, so this adds $n-1$ constraints.
- Second derivatives must match at each interval boundary, so This adds $n-1$ constraints.

We are missing 2 constraints. We have several options to find those 2 missing constraints. The *natural spline* requires

$$s''(x_0) = 0, \qquad s''(x_n) = 0. \tag{6.2}$$

The *not-a-knot* spline requires

$$s_-^{(3)}(x_1) = s_+^{(3)}(x_1), \qquad s_-^{(3)}(x_{n-1}) = s_+^{(3)}(x_{n-1}). \tag{6.3}$$

Anything that adds 2 constraints works.

Example 6.1 (Constructing a cubic spline) – Suppose we try and fit the points $(0,0), (1,0), (2,1)$. We use a basis:
$c_0(x) = x^2(3-2x)$
$c_1(x) = x^2(1-x)$
$c_2(x) = (x-1)^2 x$
$c_3(x) = 2x^3 - 3x^2 + 1.$

These have the property $c_1^{(k)}(c_k) = 1$ and $c_1^{(j)}(c_k) = 0$ otherwise. We find that

$$s(x) = \begin{cases} 0c_0(x) + \alpha c_1(x) + \beta c_2(x) + 0c_3(x), & x \in [0,1) \\ c_0(x-1) + \gamma c_1(x-1) + \eta c_2(x-1) + 0c_3(x). & x \in [1,2] \end{cases}$$

We can verify $\eta = \alpha$. With the natural spline constraints:

$$\begin{cases} s''(0) = 0: & 0 = -2\alpha - 4\beta \\ s''(2) = 0: & 0 = -6 + 4\gamma + 2\alpha \\ s''(1) \text{ agrees from } - \text{ and } +: & 4\alpha + 2\beta = 6 - 4\alpha^{-2}\gamma \end{cases} \implies \alpha = \frac{1}{2}, \gamma = \frac{5}{4}, \beta = -\frac{1}{4}.$$

6.3. B-splines

March 24, 2023

B-splines are basis functions for splines. We could have a basis $\{S_1, S_2, \ldots\}$ and model a function as $\sum_i \gamma_i S_i$. In particular, these are convenient for computers to calculate.

Assume the knots are evenly spaced: $x_k = kh$. Define the **positive part** of the function $(x-a)^n$ as

$$(x-a)_+^n := \begin{cases} (x-a)^n & x \geq a \\ 0 & x < a. \end{cases}$$

This is a spline of degree n.

Lemma 6.2

Suppose that $p(x)$ is a polynomial of degree $n \geq 1$. Then for each $1 \leq r \leq n$, the function

$$Q_{(r)}(x) = \sum_{k=0}^{r} (-1)^k \binom{r}{k} p(x - kh)$$

is a polynomial of degree $n - r$ and $Q_{(n+1)}(x) \equiv 0$.

Proof. We induct on r. For $r = 1$, $Q_{(1)}(x) = p(x) - p(x-h)$. Therefore $\deg(Q_{(1)}) = n-1$. Suppose $\deg(Q_{(r)}(x))$ is a polynomial of degree $n - r$. Then

$$\deg(Q_{(r)}(x) - Q_{(r)}(x-h)) = n - r - 1.$$

$$\begin{aligned} Q_{(r)}(x) - Q_{(r)}(x-h) &= \sum_{k=0}^{r} (-1)^k \left[p(x-kh) - p(x-(k+1)h)\right] \\ &= p(x) + (-1)^{r+1} p(x-(r+1)h) \\ &\quad + \sum_{k=1}^{r} (-1)^k \left(\binom{r}{k} + \binom{r}{k-1} \right) p(x-kh) \\ &= \sum_{k=0}^{r+1} (-1)^k \binom{r+1}{k} p(x-kh) \\ &= Q_{(r+1)}(x). \end{aligned}$$

□

Theorem 6.3

For each $n \geq 1$, the function $S_{(n)}$ defined by

$$S_{(n)}(x) := \sum_{k=0}^{n+1} (-1)^k \binom{n+1}{k} (x - kh)_+^n$$

is a spline of degree n with equally spaced knots at kh, $0 \leq k \leq n$. It is identically zero outside of $(0, (n+1)h)$.

Proof. If $x < 0$, the positive parts are all 0, so $S_{(n)}(x) = 0$. If $x > (n+1)h$,

$$S_{(n)}(x) = \sum_{k=0}^{n+1} (-1)^k \binom{n+1}{k} (x - kh)^n = Q_{(n+1)}(x) = 0. \qquad \square$$

Example 6.4 ($S_{(n)}$ function) – For $n = 1$,

$$S_{(1)}(x) = \begin{cases} x & x \in (0, 1] \\ 2 - x & x \in [1, 2) \\ 0 & \text{otherwise} \end{cases}$$

For $n = 3$, we can simplify the equation by centering it at 0:

$$S_{(3)}(x + 2) = \begin{cases} \frac{1}{6}(|x| - 2)^3 & 1 < |x| < 2 \\ \frac{1}{6}(4 - 6|x|^2 + 3|x|^3) & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

7. Numerical ODEs

7.1. Ordinary differential equations

A first-order ODE for $y(t)$ is

$$\begin{cases} y' = f(t, y), \\ y(t_0) = y_0. \end{cases} \tag{7.1}$$

We call Equation 7.1 an *initial value problem*.

Example 7.1 (Examples of ODEs) – Many situations can be modeled by first-order ODEs:

- **Population growth:** $y' = ay$, $y(0) = y_0 \implies y(t) = t_0 e^{at}$.
- **Predator-prey/Lotka-Volterra model:** Let $\mathbf{y} = (\underbrace{y_1}_{\text{prey}}, \underbrace{y_2}_{\text{predator}})$ and

$$\mathbf{y}' = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} \alpha_1 y_1 - \beta y_1 y_2 \\ -\alpha_2 y_2 + \beta_2 y_1 y_2 \end{bmatrix}$$

March 31, 2023

Definition 7.1 (Autonomous ODEs)

If an ODE is only dependent on y, i.e. in the form

$$y' = f(y),$$

it is called an **autonomous ODE**.

Why do we work a lot with only first-order ODEs? Consider Newton's second law. For a particle experiencing force

$$my'' = F(t, y, y').$$

Convert this into a system of first order ODEs. By introducing $v = y'$,

$$\begin{cases} y' = v, \\ v' = \frac{F(t,y,v)}{m}. \end{cases}$$

Therefore, higher order differential equations can be turned into first order systems. Recall some typical methods of getting an analytic solution:

1. If the ODE is linear, $y' = a(t)y + b(t)$, we can use the *integrating factor method*. We place it in the form $y' - a(t)y = b(t)$. The *integrating factor (IF)* is $I = e^{\int -a(t)\mathrm{d}t}$. By multipliying the IF on both sides, we have

$$Iy' - Ia(t)y = \frac{\mathrm{d}}{\mathrm{d}x}(Iy) = Ib(t).$$

   Therefore, by integrating on both sides,

$$y = \frac{\int I \cdot b(t)\mathrm{d}t}{I}.$$

2. If $y'(t) = f(t, y) = g(y)h(t)$, we can use *separation of variables*:

$$\frac{dy}{dt}\frac{1}{g(y)} = g(t) \implies \int \frac{dy}{g(y)} = \int h(t)dt.$$

Example 7.2 (Toricelli's law) – Suppose that $y(t)$ is the height of a liquid in a container and has an outflow at the bottom of said container. *Toricelli's law* says that the rate of fluid output is proportional to $\sqrt{y}$. We have

$$\begin{cases} y' = -k\sqrt{y} \\ y0 = H, \end{cases}$$

for $t \in [0, \infty)$. By separation of variables,

$$y(t) = \frac{1}{4}(2\sqrt{H} - kt)^2.$$

Let $T = 2\sqrt{H}/k$.

It turns out there are multiple solutions on the range $t \in (-\infty, T]$ which are dependent on a constant c we choose:

$$y(t) = \begin{cases} \frac{k^2}{4}(c-t)^2 & t < c \\ 0 & t \geq c. \end{cases}$$

Theorem 7.3 (Picard's theorem)

For a proof of a similar statement to this, see Appendix A

Suppose that $f(x, y)$ is continuous in the rectangular region

$$D = [x_0, X_m] \times [y_0 - c, y_0 + c].$$

Suppose that $|f(x, y_0)| \leq K$ when $x \in [x_0, X_m]$. Suppose also that f satisfies the a *Lipschitz condition* with parameter $L > 0$:

$$|f(x, u) - f(x, v)| \leq L|u - v|, \qquad (x, u), (x, v) \in D.$$

Assume that $C \geq \frac{K}{L}(e^{L(x_M - x_0)} - 1)$. Then there exists a *unique* function $y \in C^1[x_0, X_m]$ so that

$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0. \end{cases}$$

In addition, $|y(x) - y_0| \leq C$ for $x_0 \leq x \leq X_m$.

April 3, 2023

For numerical solutions to our ODEs, we define *mesh points* $x_n = x_0 + nh$ for $0 \leq n \leq N$ with $h = \frac{X_M - x_0}{N}$. We want to find approximations to the solution so that $y_n \approx y(x_n)$. We will use this notation as the standard for the rest of the section.

7.2. One-step methods

7.2.1. Euler's method

Start with $y(x_0) = y_0$. Suppose we have approximated y_n. Then we define

$$y_{n+1} = y_n + hf(x_n, y_n). \tag{7.2}$$

We call this the **(forward) Euler method**. To test its accuracy, we take the Taylor series:

$$\begin{aligned} y(x_n + h) &= y(x_n) + hy'(x_n) + O(h^2) \\ &= y(x_n) + hf(x_n, y_n) + O(h^2). \end{aligned}$$

By taking $y(x_n) \to y_n$ and $y(x_{n+1}) \to y_{n+1}$, we have Euler's method.

7.2.2. General cases

General *one-step methods* have the form

$$y_{n+1} = y_n + h\Phi(x_n, y_n; h).$$

Definition 7.2 (Global and truncation error)

There are two ways to track the error in these type of methods. The **global error** is defined as

$$e_n \coloneqq y(x_n) - y_n.$$

The **truncation error** is defined as

$$T_n \coloneqq \frac{y(x_{n+1}) - y(x_n)}{h} - \Phi(x_n, y_n; h).$$

Definition 7.3 (Order of accuracy)

An ODE integration method has **order of accuracy** p if $|T_n| = O(h^p)$, where p is as large as possible.

In Figure 5, we can see that the global error e_2 not only has contributions from the truncation error, but also from the error e_1 from the first step. This may be a problem if we want to converge to a solution to an ODE.

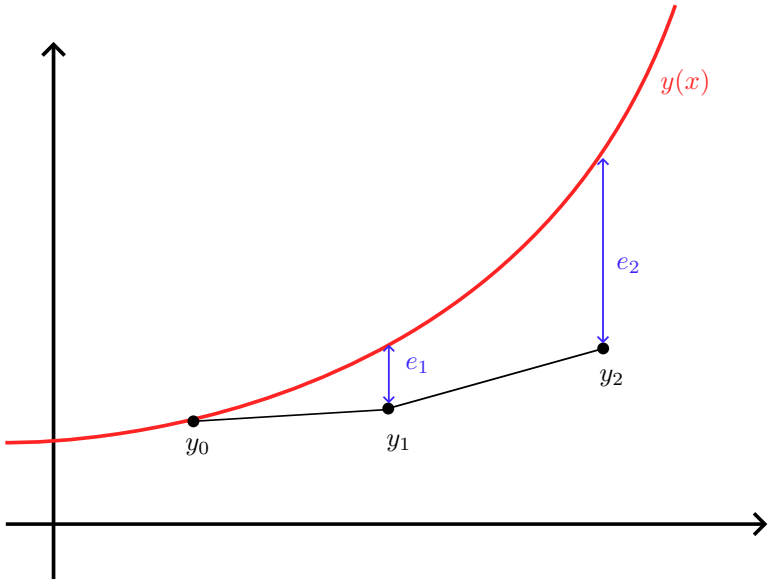

Figure 5: Accumulating error with Euler's method.

Theorem 7.4

Given a one-step method so that there exists $L_\Phi > 0$ so that for $0 \leq h \leq h_0$, and $(x, u), (x, v) \in D$, where

$$D = [x_0, X_M] \times [y_0 - c, y_0 + c],$$

we have

$$|\Phi(x, u; h) - \Phi(x, v; h)| \leq L_\Phi |u - v|.$$

In other words, Φ satisfies a *Lipschitz property* in its second term. Assuming that $|y_n - y_0| \leq C$, we have for $n = 1, \ldots, N$,

$$|e_n| \leq \frac{T}{L_\Phi}(e^{L_\Phi(x_n - x_0)} - 1).$$

Remark 7.5. This is **exponential** divergence! However, this bound is pessimistic. In practice, our bounds may diverge much slower than exponentially.

I will give a sketch of the proof. The details should be easy to fill in.

1. Solve T_n for $y(x_{n+1})$.
2. Subtract the equation $y_{n+1} = y_n + h\Phi(x_n, y_n; h)$ to get e_{n+1} in terms of e_n (and some more terms).
3. Apply the Lipschitz condition to get a bound on $|e_{n+1}|$ in terms of $|e_n|$, and show by induction that you can get a bound on $|e_n|$ in terms of T, L_Φ, h, and n.
4. Use the bound $1 + hL_\Phi \leq \exp(hL_\phi)$ (by Taylor expansion) to get the theorem statement.

7.3. Implicit 1-step methods/Runge-Kutta methods

April 7, 2023

For this topic, read Sections 12.4 (on implicit one-step methods) and 12.5 (on Runge-Kutta methods).

Runge-Kutta methods rely on a *predictor-corrector* model with their calculations. While using the derivative only to estimate the function is good, Runge-Kutta can keep in mind more curvature to the function, which will likely end up in a better estimate.

Example 7.6 (Deriving a second-order RK method) – This is largely taken from the book, but has some extra steps explained for clarity. We walk through a derviation of a second-order Runge-Kutta method and give the necessary details to derive any higher-order method.

We first note the ways of expanding our terms with a Taylor series. For y, we have

$$y(x + h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \cdots.$$

We note that $y'(x_n) = f(x_n, y_n)$. By taking the derivative of f, we may write our y derivatives purely in terms of f and its partial derivatives:

$$y''(x_n) = f_x + f_y y' = f_x + f_y f$$
$$y'''(x_n) = f_{xx} + f_{xy} f + (f_{xy} + f_{yy} f)f + f_y(f_x + f_y f).$$

For notation, we will assume all f's (and partial derivatives) are evaluated at $(x_n, y(x_n))$. For

f, we have to account for both directions,

$$f(x+h,y+k) = f(x,y) + hf_x(x,y) + kf_y(x,y) + \frac{h^2}{2}f_{xx}(x,y) + \frac{hk}{2}f_{xy}(x,y) + \frac{hk}{2}f_{yx}(x,y) + \frac{k^2}{2}f_{yy}(x,y) + \cdots .$$

If we assume that $f_{xy} = f_{yx}$ (which is the case most of the time), we can write

$$f(x+h,y+k) = f(x,y) + hf_x(x,y) + kf_y(x,y) + \frac{h^2}{2}f_{xx}(x,y) + hkf_{xy}(x,y) + \frac{k^2}{2}f_{yy}(x,y) + \cdots .$$

Let's consider the method

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n + \alpha h, y_n + \beta h k_1) \\ y_{n+1} &= y_n + h(ak_1 + bk_2). \end{aligned}$$

In this case, our Φ function is

$$\Phi(x_n, y_n; h) = af(x_n, y_n) + bf(x_n + \alpha h, y_n + \beta h k_1).$$

Recall that our method is consistent if and only if $\Phi(x,y;0) = f(x,y)$. This puts the constraint $a+b=1$. By expanding our f's in Φ, we have

$$\Phi(x_n, y(x_n); h) = af + b(f + \alpha h f_x + \beta h f f_y + \frac{1}{2}(\alpha h)^2 f_{xx} + \alpha\beta h^2 f f_{xy} + \frac{1}{2}(\beta h)^2 f^2 f_{yy} + O(h^3)).$$

Finally, we may consider the truncation error:

$$\begin{aligned} T_n = f &+ \frac{1}{2}h(f_x + ff_y) \\ &+ \frac{1}{6}h^2[f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y(f_x + f_y f)] \\ &- [af + b(f + \alpha h f_x + \beta h f f_y + \frac{1}{2}(\alpha h)^2) \\ &+ \alpha\beta h^2 f f_{xy} + \frac{1}{2}(\beta h)^2 f^2 f_{yy}] + O(h^3) \end{aligned}$$

By observing the h coefficient, we find that we need

$$b\alpha = b\beta = \frac{1}{2}$$

for second-order accuracy, or

$$\beta = a, \quad a = 1 - \frac{1}{2\alpha}, \quad b = \frac{1}{2\alpha}, \quad \alpha \neq 0.$$

α is a free parameter. The book goes over several ways that we can choose α.

April 10, 2023

An example not covered in the book of a second-order accurate RK method is *Ralston's method*. This uses $\alpha = \frac{2}{3}$. We are motivated by trying to make $\left(\frac{1}{6} - \frac{\alpha}{4}\right)$, $\left(\frac{1}{3} - \frac{\alpha}{2}\right)$ vanish. We have the formula

$$y_{n+1} = y_n + \frac{h}{4}\left[f(x_n + y_n) + 3f(x_n + \frac{2}{3}h, y_n + \frac{2}{3}hf(x_n, y_n))\right]. \tag{7.3}$$

The truncation error is

$$T_n = \frac{h^2}{6}f_y(f_x + ff_y) + O(h^3).$$

Therefore, when $f(x, y)$ is only dependent on x, f_y is zero, and this method becomes third-order accurate.

7.3.1. Butcher tableau

April 13, 2023 For a general RK method given by

$$\begin{aligned} k_i &= f\left(x_n + hc_i, y_n + h\sum_{j=1}^{i-1} a_{i,j}k_j\right) \\ y_{n+1} &= y_n + h\sum_{j=1}^{s} b_j k_j \end{aligned} \tag{7.4}$$

we can represent it by a **Butcher tableau**.

$$\begin{array}{c|cccc} c_1 & & & & \\ c_2 & a_{2,1} & & & \\ \vdots & \vdots & \ddots & & \\ c_s & a_{s,1} & \cdots & a_{s,s-1} & \\ \hline & b_1 & \cdots & b_{s-1} & b_s \end{array}$$

This gives us a concise way to write RK methods.

Example 7.7 – For the classical 4th order RK method, the corresponding Butcher tableau is

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

7.4. Linear multistep methods

The problem with RK methods is the number of evalutions of f we need to make. If $y' = f$ is a complicated system, then we would like to make those evaluations minimal while still having accuracy. A **linear multistep method** aims to do this.

Example 7.8 – We will introduce several linear multistep methods:

1. To find y_{n+1}, we may use the values of $f(x_{n-1})$ and $f(x_n)$ to interpolate a polynomial, and integrate that polynomial to get the future value:

$$\begin{aligned} y_{n+1} &= y_n + \int_{x_{n-1}}^{x_{n+1}} p(x)\,\mathrm{d}x \\ &= y_n + h\int_0^1 f_n + s(f_n - f_{n-1})\,\mathrm{d}s \\ &= y_n + h\left[\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right], \end{aligned}$$

   where f_n is f evaluated at x_n, y_n. We call this a *Adams-Bashforth formula*.

2. The *Adams-Bashforth method* is

$$y_{n+4} = y_{n+3} + \frac{h}{24}\left[55f_{n+3} - 59f_{n+2} + 37f_{n+1} - 9f_n\right].$$

3. The *Implicit (backwards) Euler method* is

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

April 14, 2023

A general linear multistep method is of the form

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f(x_{n+j}, y_{n+j}). \tag{7.5}$$

An issue we see with linear multistep methods is that we need to start with some evaluations of y_n to begin to use it. For this we can use an RK method.

Definition 7.4 (Truncation error for linear multistep methods)

The **truncation error** of a linear multistep method given by Equation 7.5 is

$$T_n := \frac{\sum_{j=0}^{k} [\alpha_j y(x_{n+j}) - h\beta_j f(x_{n+j}, y(x_{n+j}))]}{h \sum_{j=0}^{k} \beta_j}. \tag{7.6}$$

Note that, by definition, we need $\sum_{j=0}^{k} \beta_j \neq 0$.

Proposition 7.9 (Order of accuracy shortcut)

Let

$$\begin{aligned}
C_0 &= \sum_{j=0}^{k} \alpha_j \\
C_1 &= \sum_{j=0}^{k} j\alpha_j - \sum_{j=0}^{k} \beta_j \\
C_2 &= \sum_{j=0}^{k} \frac{j^2}{2}\alpha_j - \sum_{j=0}^{k} j\beta_j \\
&\vdots \\
C_n &= \sum_{j=0}^{k} \frac{j^n}{n!}\alpha_j - \sum_{j=0}^{k} \frac{j^{n-1}}{(n-1)!}\beta_j
\end{aligned}$$

A linear multistep method has order accuracy p if and only if $C_0 = C_1 = \cdots = C_p = 0$ and $C_p \neq 0$.

There are several important conditions that we need to make sure a linear multistep method satisfies for it to be reliable.

7.4.1. Zero-stability

Zero-stability is a measure of how much initial numerical errors will affect future ones. In particular, we can find these by testing the method against $y' \equiv 0$, hence the name *zero*-stability.

Definition 7.5 (Zero-stability)

A linear k-step method is **zero-stable** if there is a constant K so that for any two solution estimates (written as sequences) (y_n), (z_n) for initial values $y_0, \dots, y_{k-1}$ and $z_0, \dots, z_{k-1}$ respectively satisfies

$$|y_n - z_n| \leq K \max_{0 \leq i \leq k-1} |y_i - z_i|.$$

Definition 7.6 (Characteristic polynomials)

Define the **first and second characteristic polynomials** of a linear multistep method as

$$\rho(z) \coloneqq \sum_{j=0}^{k} \alpha_j z^j, \tag{7.7}$$

$$\sigma(z) \coloneqq \sum_{j=0}^{k} \beta_j z^j. \tag{7.8}$$

April 17, 2023

Instead of working with the zero-stability definition directly, we instead look at an equivalent (and easier to verify) formulation.

Remark 7.10 (Motivation for the root condition)**.** All we need to consider for zero-stability is the differential equation $y' = f \equiv 0$. Therefore, our multistep method will turn into

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = 0.$$

If we fix our initial values $y_0, \dots, y_{k-1}$, this becomes a homogeneous recurrence relation, and the zero condition is essentially asking when this blows up.

It turns out we have a general way of solving homogeneous recurrence relations where we end up finding the roots of the first characteristic polynomial. For example, *Binet's* formula for the Fibonacci numbers is

$$F_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n.$$

Indeed, when we look at the characteristic polynomial of its recurrence relation, $F_{n+1} = F_n + F_{n-1}$, we find that the roots are

$$\frac{1 \pm \sqrt{5}}{2},$$

which are the exponential terms in Binet's formula. To solve homogeneous recurrence relations in general, see Appendix C.

Theorem 7.11 (Root condition)

A linear multistep method is zero-stable if and only if all the roots of the first characteristic polynomial have magnitude less than 1 or have magnitude 1 and have multiplicity 1.

Example 7.12 – Consider the linear multistep method

$$y_{n+3} - y_{n+2} = \frac{h}{24}(9f_{n+3} + 19f_{n+2} - 5f_{n+1} + f_n).$$

The first characteristic polynomial becomes

$$\rho(z) = z^2(z-1).$$

The root $z = 0$ has multiplicity 2, which is fine because it has magnitude < 1. The root $z = 1$ has multiplicity 1 and magnitude 1. Thus, this method is zero-stable.

7.4.2. Consistency

Definition 7.7

A numerical method is **consistent** if the trunction error is so that for $\varepsilon > 0$, there exists $h_0 = h_0(\varepsilon)$ for which $|T_n| < \varepsilon$ for $0 < h < h_0$, when plugging in the $k+1$ points $(x_n, y(x_n)), \ldots, (x_{n+k}, y(x_{n+k}))$ that are solutions to the differential equation.

Therefore, consistency assumes that the endpoints are exactly solutions to the differential equation.

Proposition 7.13

A linear multistep method is consistent if and only if $\rho(1) = 0$ and $\rho'(1) = \sigma(1) \neq 0$.

Theorem 7.14 (Dahlquist's equivalence theorem)

For a linear k-step method that is consistent with $y' = f(x, y)$ and $y(x_0) = y_0$, where f satisfies the assumptions of Picard's theorem, zero-stability is necessary and sufficient for convergence. If y has continuous derivative of order $p+1$ and trunction error $O(h^p)$, then the global error is also $O(h^p)$.

7.4.3. Stiff systems

April 21, 2023

The definition is *stiffness* is not entirely consistent in the literature, but we can say that it happens when there are large variances in time steps of a differential equation.

Example 7.15 (Stiff system) – When a ball bounces on a table, the ball bounces much slower in comparison to how the table bounces/shifts. We cannot even see the table "bounce" because the time step is so small.

Stability problems may not show up for a while on a numerical solution, but when they do, they can cause a lot of unpredictable behavior. In general, we prefer to use implicit methods to explicit methods when it comes to stiff systems.

7.4.4. Stability

To work with stability, we look at testing our method against $y' = \lambda y$ for different values of λ. To test against $y' = \lambda y$, we use the **stability polynomial**, defined as

$$\pi(z; \lambda h) \coloneqq \sum_{j=0}^{k} (\alpha_j - \lambda h \beta_j) z^j = \rho(z) - \lambda h \sigma(z).$$

Definition

Consider a linear multistep method and its associated stability polynomial $\pi(z; \lambda h)$.

Definition 7.8 (Absolutely stable)

The method is **absolutely stable** for a given value of $\lambda h \in \mathbb{C}$ if each root $z_r = z_r(\lambda h)$ of the associated stability polynomial satisfies $|z_r(\lambda h)| < 1$.

Definition 7.9 (Region of absolute stability)

The **region of absolute stability** is the set of all points λh in $\mathbb{C}$ so that the method is absolutely stable.

Definition 7.10 (A-stable)

A linear multistep is **A-stable** is the region of absolute stability contains the region $\{\lambda h : \mathrm{Re}(\lambda h) < 0\} \subseteq \mathbb{C}$, or the left half of the complex plane.

A-stability is very hard condition to satisfy with linear multistep methods in practice, as shown through the following theorem.

Theorem 7.16 (Dahlquist's second barrier theorem)

- No explicit linear multistep method is A-stable,
- No A-stable linear multistep method has order > 2,
- The second-order A-stable linear multistep method with the smallest error is the trapezoid method.

7.5. Implicit Runge-Kutta methods

Recall that the Butcher Tableau of a classical RK method is triangular. We can extend the tableau to be square. In this case,

$$k_i = f(x_n, hc_i, y_n + h\sum_{j=1}^{s} a_{ij}k_j).$$

Since the sum ranges from 1 to s, this is an **implicit Runge-Kutta method (IRK)**.

$$\begin{array}{c|ccc} c_1 & a_{1,1} & \cdots & a_{1,s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & \cdots & a_{s,s} \\ \hline & b_1 & \cdots & b_s \end{array}$$

We may instead use a simpler version, called the **diagonally implicit Runge-Kutta method (DIRK)**, where we only have to solve for each k_i in terms of itself and previous k_j's. It has the word diagonal because its Butcher Tableau includes the diagonal.

$$\begin{array}{c|ccc} c_1 & a_{1,1} & & \\ \vdots & \vdots & \ddots & \\ c_s & a_{s,1} & \cdots & a_{s,s} \\ \hline & b_1 & \cdots & b_s \end{array}$$

I'll be using this appendix for results not covered in class.

A. Proof of (a variant of) Picard's theorem

Since this is taken from my analysis class, the exact wording of the theorem is not the same.

Theorem A.1 (Picard's theorem)
Suppose F is continuous on the rectangle

$$R \coloneqq \{(x,y) : |x - x_0| \leq a, |y - y_0| \leq b\},$$

and

$$|F(x,y) - F(x,\widetilde{y})| \leq C|y - \widetilde{y}|,$$

such that

$$\sup_{x,y \in R} |F(x,y)| = M < \infty.$$

We will prove that δ has a better bound later.

For $x \in [x_0 - \delta, x_0 + \delta]$, such that

$$\delta < \min\left\{a, \frac{b}{M}, \frac{1}{C}\right\}.$$

a solution *exists* and is *unique*.

Proof. Assume that F is defined and continuous on

$$R = \{(x,y) : x_0 - a \leq x \leq x_0 + a, y_0 - b \leq y \leq y_0 + b\}.$$

We want to apply the contraction principle. We define the operator

$$Ty(x) \coloneqq y_0 + \int_{x_0}^{x} F(t, y(t))\,\mathrm{d}t.$$

Thus the integral equation is $y = Ty$. Our metric space will be the set of all continuous functions on $[x_0 - \delta, x_0 + \delta]$ which have values in $[y_0 - b, y_0 + b]$. Let $\mathcal{M}$ be this metric space, and equip it with the sup norm.

Claim A.1. $T : \mathcal{M} \to \mathcal{M}$.

Let $y \in \mathcal{M}$. Consider

$$\begin{aligned} |Ty(x) - y_0| &= \left| \int_{x_0}^{x} F(t, y(t))\,\mathrm{d}t \right| \\ &\leq M|x - x_0| \\ &\leq M\delta. \end{aligned}$$

By requiring $\delta < \frac{b}{M}$, we are done. ■

Claim A.2. $\mathcal{M}$ is complete.

Let

$$\mathcal{M}_x \coloneqq \{f \in C[x_0 - \delta, x_0 + \delta] : |f(x) - y_0| \leq \delta\}.$$

Each of these sets are clearly closed. Since

$$\mathcal{M} = \bigcap_{x \in [x_0 - \delta, x_0 + \delta]} \mathcal{M}_x,$$

$\mathcal{M}$ is closed, and therefore it is complete. ■

Claim A.3. T is a contraction.

We will bound T to shown contraction.

$$\begin{aligned} d(Ty_1, Ty_2) &= \sup_{|x - x_0| < \delta} |Ty_1(x) - Ty_2(x)| \\ &= \sup_{|x - x_0| < \delta} \left| \int_{x_0}^{x} F(t, y_1(t)) - F(t, y_2(t)) \, \mathrm{d}t \right| \end{aligned}$$

By the Lipschitz condition, we have

$$|F(t, y_1(t)) - F(t, y_2(t))| \leq C|y_1(t) - y_2(t)| \leq C d(y_1, y_2).$$

Thus

$$\begin{aligned} d(Ty_1, Ty_2) &\leq \sup_{|x - x_0| < \delta} C d(y_1, y_2) \\ &\leq C\delta d(y_1, y_2). \end{aligned}$$

If we assume that $\delta < \frac{1}{C}$, then this is a contraction. □

Proposition A.2

We can improve the bounds of δ in Theorem A.1 to

$$\delta < \min\left\{a, \frac{b}{M}\right\}.$$

Proof. Let's choose a different metric. For a large constant L, let

$$d_L(f, g) \coloneqq \sup_{|x - x_0| < \delta} |f(x) - g(x)| e^{-L|x - x_0|}.$$

Claim A.4. $d_L(Ty_1, Ty_2) \leq \frac{C}{L} d_L(y_1, y_2)$.

$$\begin{aligned} d_L(Ty_1, Ty_2) &= \sup_{|x - x_0| < \delta} |Ty_1(x) - Ty_2(x)| e^{-L|x - x_0|} \\ &= e^{-L|x - x_0|} \sup_{|x - x_0| < \delta} \left| \int_{x_0}^{x} F(t, y_1(t)) - F(t, y_2(t)) \, \mathrm{d}t \right| \\ &\leq e^{-L|x - x_0|} \int_{x_0}^{x} C d_L(y_1, y_2) e^{L|t - x_0|} \, \mathrm{d}t \\ &= \frac{C}{L} d_L(y_1, y_2). \end{aligned}$$

□

B. Singular value decomposition

The **singular value decomposition (SVD)** is a very useful matrix factorization with many applications in numerical linear algebra.

Consider the $m \times n$ matrix $A\colon \mathbb{R}^n \to \mathbb{R}^m$. The SVD gives information that can be geometrically interpreted as the image of the unit $n-1$-sphere $\mathbb{S}^{n-1}$ in $\mathbb{R}^n$. We find that A ends up "stretching" the unit sphere in $\mathbb{R}^m$ by $\sigma_1, \dots, \sigma_m$ in orthogonal directions $\mathbf{u}_1, \dots, \mathbf{u}_m$.

Definition B.1 (SVD terms)

The **singular values** are the "stretching" factors $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. The **left singular values** $\{\mathbf{u}_1, \dots, \mathbf{u}_m\}$ in the direction of the principal semiaxes of $A^{\text{im}}(\mathbb{S}^{n-1})$. The **right singular values** $\{\mathbf{v}_1, \dots, \mathbf{v}_m\}$ are the preimages of the left singular values so that $A\mathbf{v}_i = \sigma_i \mathbf{u}_i$.

Given our matrix $A \in \mathbb{R}^{m \times n}$, we write

$$AV = \hat{U}\hat{\Sigma},$$

where $V \in \mathbb{R}^{n \times n}$ is an orthogonal matrix, $\hat{U} \in \mathbb{R}^{m \times n}$ has orthonormal columns, and $\hat{\Sigma} \in \mathbb{R}^{n \times n}$ is diagonal with non-negative real entries, and contains our singular values.

Definition B.2 (Reduced and full SVD)

Using the fact that V is orthogonal, we have $V^T V = VV^T = I$. Thus

$$A = \hat{U}\hat{\Sigma}V^T \tag{B.1}$$

is called the **reduced SVD**.

We may use an alternative form of the SVD called the **full SVD**. We add $m-n$ more orthonormal vectors as columns to $\hat{U}$ to form U, a square matrix. To negate the effects of this, we add $m-n$ rows of zeros to $\hat{\Sigma}$ to form Σ, a $m \times n$ matrix. Our equation becomes

$$A = U\Sigma V^T. \tag{B.2}$$

For a geometric intuition, consider how $U\Sigma V^T$ maps $\mathbb{S}^{n-1}$. V^T preserves the sphere, but can rotate it. Σ stretches the sphere, and U does the final rotation for the sphere.

B.1. Properties of the SVD

For the SVD to be useful, we want to know when a SVD can be formed, and if decompositions are unique to the matrix. The next theorem covers this.

Theorem B.1

$A \in \mathbb{R}^{m \times n}$ has a full singular value decomposition. Additionally,

- The $\{\sigma_j\}$ are uniquely determined by A.
- If A is square, and the $\{\sigma_j\}$ are distinct, then $\{u_j\}$ and $\{v_j\}$ are uniquely determined up to sign.

Proposition B.2 (Additional properties of the SVD)

- $\operatorname{rank} A = r$ is the number of nonzero singular values.
- $\operatorname{range} A = \operatorname{span}\{\mathbf{u}_1, \ldots, \mathbf{u}_r\}$.
- $\operatorname{null} A = \operatorname{span}\{\mathbf{v}_{r+1}, \ldots, \mathbf{v}_n\}$.
- The matrix norm induced by the euclidean norm satisfies $\|A\|_2 = \sigma_1$. As a result, $\kappa(A) = \sigma_1/\sigma_n$.
- The singular values of A are the square roots of the eigenvalues of $A^T A$.

Proposition B.3 (Using SVD to find pseudoinverse)

Define the pseudoinverse of a scalar σ as $\frac{1}{\sigma}$, or 0 if $\sigma = 0$. By defining the pseudoinverse of a diagonal matrix as the transpose and pseudoinverse of each entry, we have the formula

$$A^+ := V\Sigma^+ U^T. \tag{B.3}$$

This can be used in the same ways that the pseudoinverse has been used previously.

C. Solving homogeneous recurrence relations

Definition C.1

We are trying to solve **linear recurrence relations**. These are of the form

$$\sum_{j=0}^{k} a_j h_{n+j} + b_n = 0 \iff h_{n+k} = -\frac{1}{a_k}\left(\sum_{j=0}^{k-1} a_j h_{n+j} + b_n\right)$$

If a_i independent of n, then we have **constant coefficients**. When $b_n = 0$, we are working with a **homogeneous** recurrence.

Consider a sequence that has both constant coefficients and is homogeneous. When solving these, we need the initial conditions $h_0, \ldots, h_{k-1}$.

Definition C.2

The **characteristic polynomial** of a recurrence relation is what happens when we plug in $h_n = z^n$. In other words,

$$p(z) = \sum_{j=0}^{k} a_j z^j.$$

Example C.1 (Solving with characteristic polynomial) – Solve

$$h_n = 3h_{n-1} + 4h_{n-2} - 12h_{n-3}, \quad h_0 = 5, h_1 = -11, h_2 = 15$$

The characteristic polynomial is

$$p(z) = z^3 - 3z^2 - 4z + 12 = (z-2)(z+2)(z-3).$$

Therefore the roots are $z = -2, 2, 3$. Now for $a, b, c \in \mathbb{R}$,

$$h_n = a2^n + b(-2)^n + c3^n.$$

Solving this we get that $a = 1, b = 5, c = -1$, so

$$h_n = 2^n + 5(-2)^n - 3^n.$$

The above method always works as long as the roots of the characteristic polynomial are distinct. Otherwise,

Example C.2 (Root multiplicity in characteristic polynomial) – Solve

$$h_n = 6h_{n-1} - 12h_{n-2} + 8h_{n-3}.$$

$h_0 = 5, h_1 = 0, h_2 = -12$.

$$p(z) = (z-2)^3.$$

Here, the roots are $2, 2, 2$. There are 3 linearly independent solutions, $h_n = 2^n$, $h_n = n2^n$,$h_n = n^2 2^n$. We can check that these satisfy the recurrence. For $a, b, c \in \mathbb{R}$,

$$h_n = a2^n + bn2^n + cn^2 2^n.$$

We find that $a = 5$, $b = -6$, $c = 1$.

References

[SM03] Endre Süli and David F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003. DOI: 10.1017/CBO9780511801181.