Ch 1: Simple Iteration Method

Wednesday, September 5, 2018   12:52 PM

Motivation

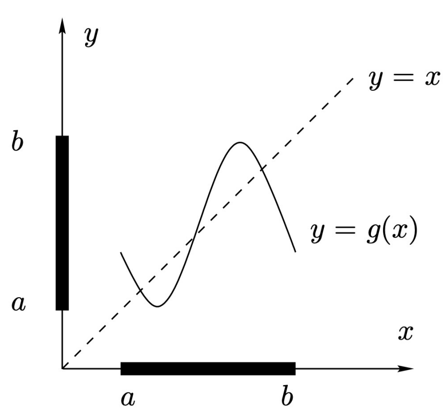
- Goal: given $f(x) = 0$, find x
- Motivation for numerical methods
  - $ax + b = 0 \Rightarrow x = -\frac{b}{a}$
  - $ax^2 + bx + c = 0 \Rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
  - $\vdots$
  - $ax^5 + bx^4 + cx^3 + dx^2 + ex + f = 0 \Rightarrow$ No formula!
- If the order of polynomial is ≥ 5, there is **no explicit zero formula**

Bolzano's Theorem (Theorem 1.1)

- Statement
  - Let f be a real-valued continuous function on the interval $[a, b]$
  - If $f(a)f(b) \leq 0$, then $\exists \xi \in [a, b]$ s.t. $f(\xi) = 0$
- Explanation
  - If a continuous function has values of **opposite sign** inside an interval
  - Then it has a **root** in that interval
- Proof
  - By the Intermediate Value Theorem
- Note
  - This theorem does not guarantee the uniqueness of solution

Brouwer's Fixed Point Theorem (Theorem 1.2)

- Statement
  - If $g \in \mathcal{C}$, and $g(x) \in [a, b]$ for $x \in [a, b]$, then $\exists \xi \in [a, b]$ s.t. $g(\xi) = \xi$
  - Here, the real number ξ is called the **fixed point** of g



- Proof
  - Let $f(x) \coloneqq x - g(x)$
  - Then, $f(a)f(b) = \underbrace{\big(a - g(a)\big)}_{\leq 0}\underbrace{\big(b - g(b)\big)}_{\geq 0} \leq 0$
  - By the Intermediate Value Theorem, $\exists \xi \in [a, b]$ s.t. $f(\xi) = 0$
  - Therefore $\xi - g(\xi) = 0 \Leftrightarrow g(\xi) = \xi$
- Why care about fixed point?
  - Finding fixed point is **numerically easier** in the sense of iteration

Simple Iteration

- Algorithm
  - **Initial guess**: $x_0 \in [a, b]$
  - **Iterate**: $x_{k+1} \coloneqq g(x_k)$
  - Stop when $|x_{n+1} - x_n| < \varepsilon$, where ε is a small number
- Example
  - Given $g(x) = \frac{1}{2}\left(x^2 + \frac{1}{2}\right)$, the fixed point of g should satisfy
    - $x = \frac{1}{2}\left(x^2 + \frac{1}{2}\right) \Leftrightarrow x^2 - 2x + \frac{1}{2} = 0$
  - Let $f(x) \coloneqq x^2 - 2x + \frac{1}{2}$, then we need to find the roots of f
  - Analytical method
    - $x = 1 \pm \frac{\sqrt{2}}{2} \approx 1.7$ or 0.3
  - Numerical method
    - $x_0 = 1$
    - $x_1 = g(x_0) = g(1) = \frac{3}{4} = 0.75$
    - $x_2 = g(x_1) = g\left(\frac{3}{4}\right) = \frac{17}{32} \approx 0.53$
    - $x_3 = g(x_2) = g\left(\frac{17}{32}\right) \approx 0.39$
    - $\vdots$
- Counter-example
  - Suppose $f(x) = x^2 - 2$
  - Then the roots of f should satisfy $f(x) = 0 \Leftrightarrow x^2 = 2 \Leftrightarrow x = \frac{2}{x}$
  - Let $x_{k+1} = g(x_k) \coloneqq \frac{2}{x_k}$ and $x_0 = 1$, then $x_1 = 2, x_2 = 1, x_3 = 2 \cdots$
  - Here, the sequence $\{x_k\}$ diverges for $g(x) = \frac{2}{x}$

Two Main Questions Over This Chapter

- When does $x_{k+1} = g(x_k)$ converge?
  - If the iteration is **unstable**
    - $x_{k+1} = g(x_k)$ **diverges**
  - If the iteration is **stable**
    - The **contraction argument** guarantees convergence
    - And the **convergence rate** is **linear**
- Given $f(x)$, how to find $g(x)$?
  - There are infinitely many g for a given f as long as $f(x) = 0 \Leftrightarrow g(x) = x$
  - Possible choice for $g(x)$
    - $g(x) = x + f(x)$, or
    - $g(x) = x + \ln(f(x) + 1)$
  - Newton's method (and secant method) will guarantee a contracting $g(x)$

Contractions

- Definition
  - Let g be a real-valued continuous function on the interval $[a, b]$
  - Then g is a **contraction** on $[a, b]$ if $\exists L \in (0,1)$ s.t.
  - $|g(x) - g(y)| \le L|x - y|, \forall x, y \in [a, b]$ (**Lipschitz condition**)
  - Here, L is called **Lipschitz constant**
- Remark on Lipschitz condition
  - $|g(x) - g(y)| \le L|x - y|, \forall x, y \in [a, b]$
  - $\Rightarrow \dfrac{|g(x) - g(y)|}{|x - y|} \le L$
  - $\Rightarrow \lim\limits_{y \to x} \dfrac{|g(x) - g(y)|}{|x - y|} \le L$
  - $\Rightarrow |g'(x)| \le L < 1$ (assume g is differentiable)

Contraction Mapping Theorem (Theorem 1.3 & 1.4 & 1.5)

- Statements
  - Let g be a contraction on $[a, b]$
  - Suppose $g(x) \in [a, b], \forall x \in [a, b]$. Then

    (1) $\exists \xi \in [a, b]$ s.t. $g(\xi) = \xi$

    (i.e. There exists a fixed point)

    (2) $\{x_{k+1} = g(x_k)\}$ converges to ξ, $\forall x_0 \in [a, b]$

    (i.e. The iterative **algorithm works**)

    (3) If the iteration stop at $|x_k - \xi| \le \varepsilon$, then

$$k \leq 1 + \left[\frac{\ln|x_1 - x_0| - \ln\big(\varepsilon(1-L)\big)}{\ln(1/L)}\right]$$

where $[x]$ is the largest integer less than or equal to x

- Proof for (1)
  - See Theorem 1.2
- Proof for (2)
  - $\underbrace{|x_{k+1} - \xi|}_{E_{k+1}} = |g(x_k) - g(\xi)|$
  - $\leq L \underbrace{|x_k - \xi|}_{E_k}$ by Lipschitz condition
  - $\leq L^2 \underbrace{|x_{k-1} - \xi|}_{E_{k-1}}$, since $\underbrace{|x_k - \xi|}_{E_k} \leq L \underbrace{|x_{k-1} - \xi|}_{E_{k-1}}$ by induction
  - $\vdots$
  - $\leq L^{k+1} \underbrace{|x_0 - \xi|}_{E_0} \to 0$ as $k \to \infty$
- Proof for (3)
  - From the proof for (2), we know that
    - $E_k \leq L^k E_0 \leq \varepsilon$
  - Taking log on both side, we obtain
    - $k \leq \log_L \frac{\varepsilon}{E_0}$
  - Calculate E_0
    - $E_0 = |x_0 - \xi| = |x_0 - x_1 + x_1 - \xi|$
      $\leq |x_0 - x_1| + |x_1 - \xi| \leq |x_0 - x_1| + L|x_0 - \xi|$
    - $\Rightarrow E_0 \leq |x_0 - x_1| + L \cdot E_0$
    - $\Rightarrow E_0 \leq \frac{|x_1 - x_0|}{1 - L}$
  - Therefore
    - $k \geq \log_L \frac{\varepsilon}{\frac{|x_1 - x_0|}{1-L}} = \log_L \frac{\varepsilon(1-L)}{|x_1 - x_0|} = \frac{\ln|x_1 - x_0| - \ln\big(\varepsilon(1-L)\big)}{\ln(1/L)}$
- Corollary
  - Given $g: [a, b] \to [a, b]$, and $g \in \mathcal{C}^1[a, b]$
  - If $|g'(x)| \leq L < 1$, then the sequence $\{x_k = g(x_{k-1})\}$ converges to ξ
- Remark on Corollary
  - If we relax $|g'(x)| < 1$ to be just $|g'(\xi)| < 1$
  - Then when $\boldsymbol{x_0}$ **is close to** $\boldsymbol{\xi}$, $\{x_k\}$ will **converge** to ξ
  - Since in a small neighborhood of ξ, $g'(x) \sim |g'(\xi)| < 1$

Stability of Fixed Point (Theorem 1.3)

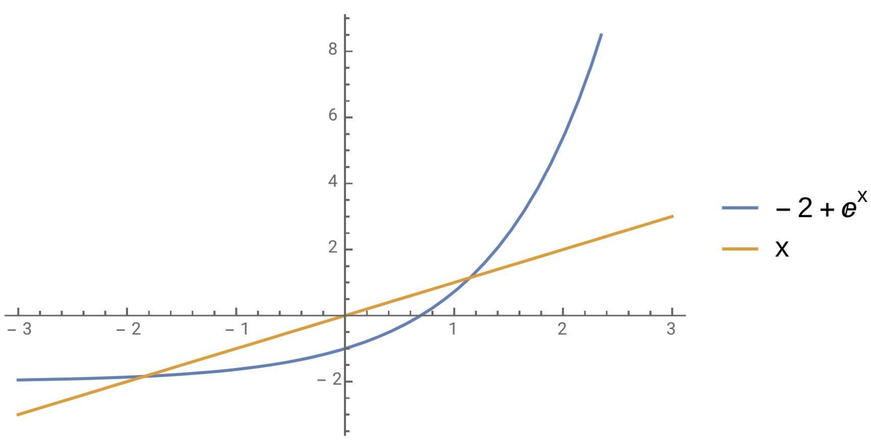
- Stable Fixed Point
  - If $\xi = g(\xi)$, and $|g'(\xi)| < 1$, then ξ is a **stable fixed point**
  - A stable fixed point can be found via $\{x_{k+1} = g(x_k)\}$
- Unstable Fixed Point
  - If $\xi = g(\xi)$, and $|g'(\xi)| > 1$, then ξ is a **unstable fixed point**
  - If ξ is an unstable fixed point, then $\{x_{k+1} = g(x_k)\}$ won't converge to ξ

Rate of Convergence (Definition 1.4 & 1.7)

- Suppose $\xi = \lim_{k\to\infty} x_k$, and define $E_k \coloneqq |x_k - \xi|$
- An algorithm is said to converge **linearly** if
  - $\lim_{k\to\infty} \frac{E_{k+1}}{E_k} = \mu$, for some constant $\mu \in (0,1)$
- An algorithm is said to converge **superlinearly** if
  - $\lim_{k\to\infty} \frac{E_{k+1}}{E_k} = 0$
- An algorithm is said to converge **quadratically** if
  - $\lim_{k\to\infty} \frac{E_{k+1}}{E_k^2} = \mu$, for some constant $\mu > 0$
- An algorithm is said to converge **with order q** if
  - $\lim_{k\to\infty} \frac{E_{k+1}}{E_k^q} = \mu$, for some constant $\mu > 0$

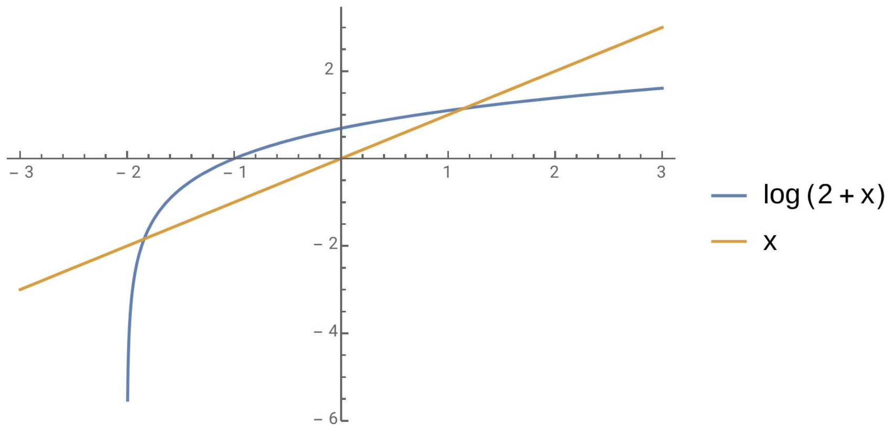
$f(x) = e^x - x - 2$ (Example 1.7)

- Define $g(x) = e^x - 2$



  - We observed that $g(x)$ maps $[1,2]$ to $[1,2]$
    - By the Fixed Point Theorem, $\exists \xi \in [1,2]$ s.t. $g(\xi) = \xi$
    - We need to check whether $g(x)$ satisfies the Lipschitz condition
    - $g'(\xi) = e^{\xi} \in [e^1, e^2]$
    - $\Rightarrow \boldsymbol{|g'(\xi)| > 1}$
    - $\Rightarrow$ unstable fixed point

        - $\Rightarrow$ the algorithm won't work
    - And $g(x)$ also maps $[-2,-1]$ to $[-2,-1]$
        - By the Fixed Point Theorem, $\exists\xi \in [1,2]$ s.t. $g(\xi) = \xi$
        - $g'(\xi) = e^{\xi} \in [e^{-2}, e^{-1}]$
        - $\Rightarrow \boldsymbol{|g'(\xi)| < 1}$
        - $\Rightarrow$ stable fixed point
        - $\Rightarrow$ run $\{x_{k+1} = g(x_k)\}$ for ξ
- Define $g(x) = \ln(x+2)$



    - We observed that $g(x)$ maps $[1,2]$ to $[1,2]$
        - $g'(\xi) = \dfrac{1}{\xi+2} \in \left[\dfrac{1}{4}, \dfrac{1}{3}\right]$
        - $\Rightarrow |g'(\xi)| < 1$
        - $\Rightarrow$ stable fixed point
        - $\Rightarrow$ run $\{x_{k+1} = g(x_k)\}$ for ξ
    - And $g(x)$ also maps $(-2,-1)$ to $(-2,-1)$
        - $g'(\xi) = \dfrac{1}{\xi+2} \in (1, +\infty)$
        - $\Rightarrow |g'(\xi)| > 1$
        - $\Rightarrow$ unstable fixed point
        - $\Rightarrow$ the algorithm won't work
- Remark
    - $x = e^x - 2 \Rightarrow f(x) = e^x - x - 2$
        - We have a stable fixed point $\xi \in [-2,-1]$, and a unstable $\xi \in [1,2]$
    - $x = \ln(x+2) \Rightarrow e^x = x + 2 \Rightarrow f(x) = e^x - x - 2$
        - We have a stable fixed point $\xi \in [1,2]$, and a unstable $\xi \in [-2,-1]$
    - Therefore the choice of g will affect the convergence behavior
    - So how can we design a function $g(x)$ s.t. every fixed point is stable?

Newton's Method (Definition 1.6)

- In Newton's method, $g(x)$ is defined as

- $g(x) = x - \frac{f(x)}{f'(x)}$
- It's obvious that $f(\xi) = 0 \Leftrightarrow g(\xi) = \xi$
- Why the **fixed points of g is stable**
  - We want to show that $|g'(\xi)| < 1$
  - $g(x) = x - \frac{f(x)}{f'(x)} \Rightarrow g'(x) = 1 - \left(\frac{f(x)}{f'(x)}\right)'$
  - $\left(\frac{f}{f'}\right)' = \frac{f' \cdot f' - f \cdot f''}{(f')^2} = 1 - \frac{f \cdot f''}{(f')^2}$
  - $\Rightarrow g'(x) = 1 - \left(1 - \frac{f(x) \cdot f''(x)}{[f'(x)]^2}\right) = \frac{f(x) \cdot f''(x)}{[f'(x)]^2}$
  - $\Rightarrow |g'(\xi)| = \frac{f(\xi) \cdot f''(\xi)}{[f'(\xi)]^2} = 0 < 1$

Convergence of Newton's Method (Theorem 1.8)

- Statement
  - Newton′s method **converges quadratically** $i.e.\ \lim_{k \to \infty} \frac{E_{k+1}}{E_k^2} \le \mu < 1$
- Assumption
  - $f(\xi) = 0$
  - $f \in C^2$ in $[\xi - \delta, \xi + \delta] = I_\delta$, since we need to use f' and f''
  - $\boldsymbol{f'(\xi) \neq 0}$, since it will appear at the denominator
  - $\left|\frac{f''(x)}{f'(y)}\right| \le \boldsymbol{A}, (\forall x, y \in I_\delta)$
  - $|x_0 - \xi| \le \frac{1}{A}$ ($i.e.$ The initial guess is not too far away from ξ)
- Proof
  - **Expand $f(\xi)$** at x_k to obtain $f(x_k)$ and $f'(x_k)$
    - $f(\xi) = f(x_k + \xi - x_k)$
    - $= f(x_k) + f'(x_k)(\xi - x_k) + \frac{1}{2} f''(x_k)(x_k - \xi)^2 + \cdots$

      (by Taylor expansion of f)
    - $= f(x_k) + f'(x_k)(\xi - x_k) + \frac{1}{2} f''(\theta_k)(x_k - \xi)^2$

      (for some constant $\theta_k \in (x_k, \xi)$, by the Mean Value Theorem)
  - **Express $\frac{f(x_k)}{f'(x_k)}$** in x_{k+1} **using $\frac{f''}{f'}$**, since we already know $\left|\frac{f''(x)}{f'(y)}\right| \le A$
    - By assumption, $f(\xi) = 0$
    - $\Rightarrow f(x_k) + f'(x_k)(\xi - x_k) + \frac{1}{2} f''(\theta_k)(x_k - \xi)^2 = 0$

″

- $\Rightarrow f(x_k) = -\frac{1}{2} f''(\theta_k)(x_k - \xi)^2 - f'(x_k)(\xi - x_k)$
- $\Rightarrow \frac{f(x_k)}{f'(x_k)} = -\frac{1}{2} \frac{f''(\theta_k)}{f'(x_k)} (\xi - x_k)^2 - (\xi - x_k)$

- **Compute E_{k+1}**, and express it with E_k
  - $$\begin{aligned} E_{k+1} &= |x_{k+1} - \xi| = |g(x_k) - \xi| \\ &= \left| \left(x_k - \frac{f(x_k)}{f'(x_k)} \right) - \xi \right| \\ &= \left| \left\{ x_k - \left[-\frac{1}{2} \frac{f''(\theta_k)}{f'(x_k)} (\xi - x_k)^2 - (\xi - x_k) \right] \right\} - \xi \right| \\ &= \left| x_k + \frac{1}{2} \frac{f''(\theta_k)}{f'(x_k)} (\xi - x_k)^2 + \xi - x_k - \xi \right| \\ &= \frac{1}{2} \left| \frac{f''(\theta_k)}{f'(x_k)} \right| (\xi - x_k)^2 \\ &= \frac{1}{2} \left| \frac{f''(\theta_k)}{f'(x_k)} \right| E_k^2 \end{aligned}$$
- Show the algorithm **converges**
  - By assumption, $|x_k - \xi| \leq \frac{1}{A}$, and $\left| \frac{f''(x)}{f'(y)} \right| \leq A, (\forall x, y \in I_\delta)$
  - So, $E_{k+1} = \frac{1}{2} \underbrace{\left| \frac{f''(\theta_k)}{f'(x_k)} \right|}_{\leq A} \underbrace{E_k}_{\leq \frac{1}{A}} E_k \leq \frac{1}{2} \cdot A \cdot \frac{1}{A} \cdot E_k = \frac{1}{2} E_k \to 0$ as $k \to +\infty$
  - Therefore x_k converges to ξ
- Show the algorithm **converges quadratically**
  - $\frac{E_{k+1}}{E_k^2} = \frac{1}{2} \left| \frac{f''(\theta_k)}{f'(x_k)} \right| \leq \frac{1}{2} A$
  - As $k \to +\infty$, both x_k and θ_k converge to ξ
  - Thus, $\lim_{k \to +\infty} \frac{E_{k+1}}{E_k^2} = \frac{1}{2} \left| \frac{f''(\xi)}{f'(\xi)} \right| = \mu$, where $\mu \in \left(0, \frac{A}{2}\right]$ is a constant

Secant Method (Definition 1.8)

- Motivation
  - Sometimes f' can be hard to find in Newton's method
  - But we can **approximate f'** using a difference quotient
  - $i.e. f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$
- Definition
  - $x_{k+1} = x_k - f(x_k) / \left(\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \right) = x_k - f(x_k) \left(\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \right)$
- Note
  - The secant method requires **two initial values** x_0 and x_1

Convergence of Secant Method (Theorem 1.10)

- Statement
  - Let $f \in C^1[\xi-\delta, \xi+\delta]$ s.t. $f(\xi)=0$ and $f'(\xi) \neq 0$
  - If $\boldsymbol{x_0, x_1}$ **is close to** $\boldsymbol{\xi}$, then $\{x_{k+1} = g(x_k)\}$ converges **at least linearly**
- Proof
  - WLOG, assume $\alpha := f'(\xi) > 0$ in a small neighborhood of ξ
  - Choose I be a neighborhood of ξ such that
    - $0 < \frac{3}{4}\alpha < f'(x) < \frac{5}{4}\alpha, \forall x \in I$
  - Compute x_{k+1}
    - $x_{k+1} = x_k - f(x_k) / \left(\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \right)$
    - By the Mean Value Theorem
      - $\frac{f(x_k) - \overbrace{f(\xi)}^{0}}{x_k - \xi} = f'(\eta_k) \Rightarrow f(x_k) = f'(\eta_k)(x_k - \xi)$, and
      - $\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} = f'(\theta_k)$
      - for some $\theta_k \in [x_k, x_{k-1}], \eta_k \in (x_k, \xi)$
    - Therefore, $\boldsymbol{x_{k+1} = x_k - \frac{f'(\eta_k)(x_k - \xi)}{f'(\theta_k)}}$
  - Check $\frac{E_{k+1}}{E_k} < 1$
    - $E_{k+1} = x_{k+1} - \xi = E_k - \frac{f'(\eta_k)}{f'(\theta_k)} E_k = \left[1 - \frac{f'(\eta_k)}{f'(\theta_k)}\right] E_k$
    - $\Rightarrow \frac{E_{k+1}}{E_k} = \left[1 - \frac{f'(\eta_k)}{f'(\theta_k)}\right] < \left(1 - \frac{5\alpha/4}{3\alpha/4}\right) = \frac{2}{3} < 1$
  - Therefore secant method converges at least linearly

Ch 2: Solution of Systems of Linear Equations

Friday, December 7, 2018 10:52 PM

LU Decomposition

Monday, September 17, 2018 9:56 AM

What is Matrix

- A matrix is **a list of numbers**
  - $A_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$
- A matrix is **a list of column vectors**
  - $A = [\overrightarrow{a_1}, \overrightarrow{a_2}, \cdots, \overrightarrow{a_n}]$
- A matrix is **a list of row vectors**
  - $A = \begin{bmatrix} \overrightarrow{b_1} \\ \overrightarrow{b_2} \\ \vdots \\ \overrightarrow{b_m} \end{bmatrix}$
- A matrix is **a function**
  - Given $A_{m\times n}: \mathbb{R}^n \to \mathbb{R}^m$ and $\vec{y}_{m\times 1} = A_{m\times n}\vec{x}_{n\times 1}$. Then
  - $\vec{y} = A\vec{x} = [\overrightarrow{a_1}, \dots, \overrightarrow{a_n}] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} x_i \overrightarrow{a_i}$
  - $\vec{y} = A\vec{x} = \begin{bmatrix} \overrightarrow{b_1} \\ \overrightarrow{b_2} \\ \vdots \\ \overrightarrow{b_m} \end{bmatrix} \vec{x} = \begin{bmatrix} \overrightarrow{b_1} \cdot \vec{x} \\ \overrightarrow{b_2} \cdot \vec{x} \\ \vdots \\ \overrightarrow{b_m} \cdot \vec{x} \end{bmatrix}$

Gaussian Elimination (Section 2.2)

- Introduction
  - Gaussian elimination is an algorithm for solving systems of linear equations
  - A sequence of **elementary row operations** is performed to modify the matrix into the **upper-triangular** form
- Example
  - Given $A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 2 \\ -1 & 5 & -4 \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} 6 \\ 16 \\ -3 \end{bmatrix}$, find $\vec{x}$ s.t. $A\vec{x} = \vec{b}$
  - We want to **generate** as many **zeros** as possible **below the diagonal**
    - $A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 0 \\ 0 & 6 & -3 \end{bmatrix}, \vec{b} = \begin{bmatrix} 6 \\ 4 \\ 3 \end{bmatrix}$
    - $A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & -3 \end{bmatrix}, \vec{b} = \begin{bmatrix} 6 \\ 4 \\ -9 \end{bmatrix}$

- Therefore, $\begin{cases} x_1 + x_2 + x_3 = 6 \\ x_2 = 2 \\ x_3 = 3 \end{cases} \Rightarrow \begin{cases} x_1 = 1 \\ x_2 = 2 \\ x_3 = 3 \end{cases}$

- Remark
  - **Digging holes downwards** is the same as **multiplying by lower-triangular matrices**
  - $A = \begin{bmatrix} \overrightarrow{b_1} \\ \vdots \\ \overrightarrow{b_s} \\ \vdots \\ \overrightarrow{b_r} \\ \vdots \\ \overrightarrow{b_m} \end{bmatrix} \xrightarrow{\text{row operation}} \begin{bmatrix} \overrightarrow{b_1} \\ \vdots \\ \overrightarrow{b_s} \\ \vdots \\ \overrightarrow{b_r} + c\overrightarrow{b_s} \\ \vdots \\ \overrightarrow{b_m} \end{bmatrix} = \begin{bmatrix} \overrightarrow{b_1} \\ \vdots \\ \overrightarrow{b_s} \\ \vdots \\ \overrightarrow{b_r} \\ \vdots \\ \overrightarrow{b_m} \end{bmatrix} + c \begin{bmatrix} \vec{0} \\ \vdots \\ \vec{0} \\ \vdots \\ \overrightarrow{b_s} \\ \vdots \\ \vec{0} \end{bmatrix} = A + cE_{rs} \cdot A = (I + cE_{rs})A$
  - where $[E_{rs}]_{ij} = \begin{cases} 1 & \text{if } i = r, s = j \\ 0 & o.w. \end{cases}$, and $r > s$
  - Note that $(\boldsymbol{I} + \boldsymbol{cE_{rs}})$ **is a lower-triangular matrix**
  - In the example above, we are indeed multiplying lower-triangular matrices

$$A\vec{x} = \vec{b}$$
$$(I - 2E_{21})A\vec{x} = (I - 2E_{21})\vec{b}$$
$$(I + E_{31})(I - 2E_{21})A\vec{x} = (I + E_{31})(I - 2E_{21})\vec{b}$$
$$(I - 3E_{32})(I + E_{31})(I - 2E_{21})A\vec{x} = (I - 3E_{32})(I + E_{31})(I - 2E_{21})\vec{b}$$

- Proposition 1: The **product of lower-triangular-matrices** is also lower-triangular
  - Statement
    - Given two lower-triangular matrices L_{ij} and $L_{pq}(i > j, \text{and } p > q)$
    - Their product $L_{ij}L_{pq}$ is also lower-triangular
  - Proof
    - $L_{ij}L_{pq} = \left(I + cE_{ij}\right)\left(I + dE_{pq}\right) = I + cE_{ij} + dE_{pq} + cdE_{ij}E_{pq}$
    - where $E_{ij}E_{pq} = \begin{cases} 0_{m\times n} & j \neq p \\ E_{iq} & j = p \end{cases}$ is also lower-triangular
  - Corollary
    - Given a list of lower-triangular matrix $L_{i_1 j_1}, \dots, L_{i_k j_k}$
    - Their product $L_{i_1 j_1} \times \cdots \times L_{i_k j_k}$ is also lower-triangular

- Proposition 2: The **inverse of lower-triangular-matrix** is also lower-triangular
  - If L_{ij} is a lower-triangular matrix, then L_{ij}^{-1} is also lower-triangular
  - Claim: If $L_{ij} = I + cE_{ij}$, then $L_{ij}^{-1} = \left(I - cE_{ij}\right)$
  - Proof: $L_{ij}L_{ij}^{-1} = \left(I + cE_{ij}\right)\left(I - cE_{ij}\right) = I - c^2 \underbrace{E_{ij}^2}_{0} = I$

LU Decomposition (Section 2.3)

- Goal
  - We want to decompose A into $\boldsymbol{L} \times \boldsymbol{U}$, where

    - L is a **lower-triangular** matrix, and
    - U is an **upper-triangular** matrix
- General Idea
    - The elimination process for A can be written as follows
        - $L_{n,n-1}L_{n-1,n-2}\cdots L_{31}L_{21}A = U$
    - Moving L's to the other side, we obtain
        - $A = L_{21}^{-1}L_{31}^{-1}\cdots L_{n-1,n-2}^{-1}L_{n,n-1}^{-1}U$
    - Hence
        - $A = LU$, where $L = L_{21}^{-1}L_{31}^{-1}\cdots L_{n-1,n-2}^{-1}L_{n,n-1}^{-1}$ is a lower-triangular matrix
- Motivation for LU decomposition
    - Given $Ax = b$, find $x = A^{-1}b$
    - Approach 1
        - $A^{-1} = \frac{1}{\det A}\begin{bmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{bmatrix}$, where $A_{ij} = (-1)^{i+j}\text{Cof}(a_{ij})$, and $a_{ij} = [A]_{ij}$
        - $\boldsymbol{\mathcal{O}(n!)}$ **operations** needed to find x
    - Approach 2
        - Given the equation $A\vec{x} = \vec{b}$, we want to solve for $\vec{x}$
        - Suppose we have already obtained the **LU decomposition** $\boldsymbol{A = LU}$
        - Then $A\vec{x} = \vec{b}$ becomes $LU\vec{x} = \vec{b}$
        - Let $\vec{y} = U\vec{x}$, then $L\vec{y} = \vec{b}$
        - We first **solve the equation** $\boldsymbol{L\vec{y} = \vec{b}}$ **for** $\boldsymbol{y}$ in time $\mathcal{O}(n^2)$
            - $\begin{bmatrix} L_{11} & & & \\ L_{12} & L_{22} & & \\ \vdots & \vdots & \ddots & \\ L_{1n} & L_{2n} & \cdots & L_{nn} \end{bmatrix}\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$
        - Then **solve the equation** $\boldsymbol{\vec{y} = U\vec{x}}$ **for** $\boldsymbol{\vec{x}}$ in time $\mathcal{O}(n^2)$
            - $\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} U_{11} & U_{21} & \cdots & U_{n1} \\ & U_{22} & \cdots & U_{n2} \\ & & \ddots & \vdots \\ & & & U_{nn} \end{bmatrix}\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$
        - The time complexity for LU decomposition is $\mathcal{O}(n^3)$
        - After that, $\mathcal{O}(n^2)$ operations are needed to get the final result

QR Factorization

Friday, December 7, 2018 10:51 PM

Least-Square Fitting (Section 2.9)

- Example
  - Find the solution for $A\vec{x} = \vec{b}$, where $A = \begin{bmatrix} 3 & 1 \\ 1 & 1 \\ 4 & 2 \end{bmatrix}, \vec{b} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$
  - There are 3 constraints for 2 variables, so this system is **over-determined**
  - In general, such a system will have **no solution**
    - $\begin{cases} 3x_1 + x_2 = 1 \\ x_1 + x_2 = 0 \end{cases} \Rightarrow \begin{cases} x_1 = 1/2 \\ x_2 = -1/2 \end{cases}$, but it won't satisfy $4x_1 + 2x_2 = 2$
  - We can instead find a solution that best fit the equation
    - *i.e.* Find a vector $\vec{x}$ such that $A\vec{x} - \vec{b}$ is as small as possible
- 2-Norm
  - If $\vec{z} = \begin{bmatrix} z_1 \\ \cdots \\ z_n \end{bmatrix}$, then $\|\vec{z}\|_2 = \sqrt{z_1^2 + \cdots + z_n^2}$ is called the **2-norm** of a vector
- Least-Square Fitting
  - We want to **find** $\min\limits_{\vec{x} \in \mathbb{R}^2} \left\| \boldsymbol{A\vec{x} - \vec{b}} \right\|_{\mathbf{2}}$
  - Square $\left\| A\vec{x} - \vec{b} \right\|_2$, and expand the result
    - $$\begin{aligned} \left\| A\vec{x} - \vec{b} \right\|_2^2 &= \left\langle A\vec{x} - \vec{b}, A\vec{x} - \vec{b} \right\rangle, \qquad \text{since } \|\vec{z}\|_2^2 = \langle \vec{z}, \vec{z} \rangle \\ &= \left(A\vec{x} - \vec{b}\right)^T \cdot \left(A\vec{x} - \vec{b}\right) \\ &= \left(\vec{x}^T A^T - \vec{b}^T\right) \cdot \left(A\vec{x} - \vec{b}\right), \qquad \text{note that } (AB)^T = B^T A^T \\ &= \vec{x}^T A^T A\vec{x} - \vec{b}^T A\vec{x} - \vec{x}^T A^T \vec{b} + \vec{b}^T \vec{b} \\ &= \vec{x}^T A^T A\vec{x} - 2\vec{x}^T A^T \vec{b} + \vec{b}^T \vec{b}, \qquad \text{since } \vec{b}^T A\vec{x} = \vec{x}^T A^T \vec{b} \end{aligned}$$
  - Hypothetically, suppose $x, a, b \in \mathbb{R}$
    - Suppose we want to minimize $f(x) = a^2x^2 - 2abx + b^2$
    - Then we need to solve the root of f'
    - $f'(x) = 2a^2x - 2ab \equiv 0 \Rightarrow x = \dfrac{ab}{a^2} = \dfrac{b}{a}$
  - Now, back to the problem
    - Let $\vec{F}(\vec{x}) = \vec{x}^T A^T A\vec{x} - 2\vec{x}^T A^T \vec{b} + \vec{b}^T \vec{b}$
    - Set $\nabla_{\vec{x}} F = 2A^T A\vec{x} - 2A^T \vec{b} \equiv \vec{0}$
    - Then $(A^T A)\vec{x} = A^T \vec{b} \Rightarrow \boldsymbol{\vec{x}} = \left(\boldsymbol{A^T A}\right)^{\mathbf{-1}} \boldsymbol{A^T \vec{b}}$

- Summary
  - If $A_{m\times m}$ is a square matrix, then
    - $A\vec{x} = \vec{b} \Rightarrow \vec{x} = A^{-1}\vec{b}$
  - If $A_{m\times n}$ is over-determined (*i.e.* $m > n$), then
    - $A\vec{x} \approx \vec{b} \Rightarrow \vec{x} = \left(A^TA\right)^{-1}A^T\vec{b}$
    - Here, $\left(\boldsymbol{A^TA}\right)^{-1}\boldsymbol{A^T}$ is called the **pseudo-inverse** of A

Gram-Schmidt Orthogonalization

- Motivation
  - $A = [\overrightarrow{a_1}, \dots, \overrightarrow{a_n}]$ maps the i-th standard basis $\overrightarrow{e_i}$ to $\overrightarrow{a_i}$
  - But the resulting vectors $\{\overrightarrow{a_1}, \dots, \overrightarrow{a_n}\}$ may not be orthonormal
  - Gram–Schmidt process is a method to **orthonormalize the vectors**
  - Later on, we can use this method to **compute the QR Factorization** of A
- Gram-Schmidt Process

Orthogonalization	Normalization
$\overrightarrow{q_1} = \overrightarrow{a_1}$	$\overrightarrow{q_1} = \frac{\overrightarrow{q_1}}{\lVert\overrightarrow{q_1}\rVert_2}$
$\overrightarrow{q_2} = \overrightarrow{a_2} - \langle\overrightarrow{a_2}, \overrightarrow{q_1}\rangle\overrightarrow{q_1}$	$\overrightarrow{q_2} = \frac{\overrightarrow{q_2}}{\lVert\overrightarrow{q_2}\rVert_2}$
$\overrightarrow{q_3} = \overrightarrow{a_3} - \langle\overrightarrow{a_3}, \overrightarrow{q_1}\rangle\overrightarrow{q_1} - \langle\overrightarrow{a_3}, \overrightarrow{q_2}\rangle\overrightarrow{q_2}$	$\overrightarrow{q_3} = \frac{\overrightarrow{q_3}}{\lVert\overrightarrow{q_3}\rVert_2}$
$\vdots$	$\vdots$
$\overrightarrow{q_k} = \overrightarrow{a_k} - \sum_{i=1}^{k-1}\langle\overrightarrow{a_k}, \overrightarrow{q_i}\rangle\overrightarrow{q_i}$	$\overrightarrow{q_k} = \frac{\overrightarrow{q_k}}{\lVert\overrightarrow{q_k}\rVert_2}$

- Remark
  - $\langle\overrightarrow{a_k}, \overrightarrow{q_i}\rangle\overrightarrow{q_i}$ is the projection of $\overrightarrow{a_k}$ onto $\overrightarrow{q_i}$ (assuming $\overrightarrow{q_i}$ is normalized)
- Proof: $\lVert\overrightarrow{\boldsymbol{q_i}}\rVert_2 = \mathbf{1}$
  - This is obviously true by the normalization process
- Proof: $\mathbf{span}\{\overrightarrow{\boldsymbol{a_1}}, \dots, \overrightarrow{\boldsymbol{a_k}}\} = \mathbf{span}\{\overrightarrow{\boldsymbol{q_1}}, \dots, \overrightarrow{\boldsymbol{q_k}}\}$
  - $\overrightarrow{q_k} \in \text{span}\{\overrightarrow{a_1}, \dots, \overrightarrow{a_k}\}$
  - $\overrightarrow{a_k} \in \text{span}\{\overrightarrow{q_1}, \dots, \overrightarrow{q_k}\}$
- Proof: $\overrightarrow{\boldsymbol{q_i}} \perp \overrightarrow{\boldsymbol{q_j}}$
  - For $i = 2, j = 1$, we need to show that $\langle\overrightarrow{q_2}, \overrightarrow{q_1}\rangle = 0$
    - $\langle\overrightarrow{q_2}, \overrightarrow{q_1}\rangle = c\langle\overrightarrow{a_2} - \langle\overrightarrow{a_2}, \overrightarrow{q_1}\rangle, \overrightarrow{q_1}\rangle$
      $= c\langle\overrightarrow{a_2}, \overrightarrow{q_1}\rangle - c\langle\overrightarrow{a_2}, \overrightarrow{q_1}\rangle\langle\overrightarrow{q_1}, \overrightarrow{q_1}\rangle$
      $= c\langle\overrightarrow{a_2}, \overrightarrow{q_1}\rangle - c\langle\overrightarrow{a_2}, \overrightarrow{q_1}\rangle = 0$
  - More generally, we can show that $\langle\overrightarrow{q_k}, \overrightarrow{q_j}\rangle = 0$, for $k \neq j$

- $\langle \vec{q_k}, \vec{q_j} \rangle = c\left\langle \left(\vec{a_k} - \sum_{i=1}^{k-1} \langle \vec{a_k}, \vec{q_i} \rangle \vec{q_i} \right), \vec{q_j} \right\rangle$

$$= c\langle \vec{a_k}, \vec{q_j} \rangle - c \sum_{i=1}^{k-1} \langle \vec{a_k}, \vec{q_i} \rangle \underbrace{\langle \vec{q_i}, \vec{q_j} \rangle}_{\delta_{ij}}$$

$$= c\langle \vec{a_k}, \vec{q_j} \rangle - c\langle \vec{a_k}, \vec{q_j} \rangle = 0$$

QR Factorization (Theorem 2.12 & 2.13)

- Goal
  - We want to factorize A into $Q \times R$, where
  - Q is a **unitary matrix**, and
  - R is an **upper-triangular matrix**
- Unitary Matrix
  - Matrix $Q_{m\times n} = [\vec{q_1}, \dots, \vec{q_n}]$ is said to be **unitary** if $\langle \vec{\boldsymbol{q_i}}, \vec{\boldsymbol{q_j}} \rangle = \boldsymbol{\delta_{ij}} = \begin{cases} \mathbf{1} & \boldsymbol{i = j} \\ \mathbf{0} & \boldsymbol{i \neq j} \end{cases}$
  - If Q is a unitary matrix, then $\boldsymbol{Q^T Q} = \begin{bmatrix} \vec{\boldsymbol{q_1}}^T \\ \vdots \\ \vec{\boldsymbol{q_n}}^T \end{bmatrix} [\vec{\boldsymbol{q_1}}, \dots, \vec{\boldsymbol{q_n}}] = \boldsymbol{I_{n\times n}}$
  - Note: δ_{ij} is called Kronecker delta function
- How to Use **Gram-Schmidt Process** to Compute QR Factorization
  - Perform the Gram-Schmidt process to matrix $A = [\vec{a_1}, \dots, \vec{a_n}]$

$\vec{q_1} = \vec{a_1}$	$\vec{q_1} = \dfrac{\vec{q_1}}{\lVert\vec{q_1}\rVert_2}$
$\vec{q_2} = \vec{a_2} - \langle \vec{a_2}, \vec{q_1} \rangle \vec{q_1}$	$\vec{q_2} = \dfrac{\vec{q_2}}{\lVert\vec{q_2}\rVert_2}$
$\vec{q_3} = \vec{a_3} - \langle \vec{a_3}, \vec{q_1} \rangle \vec{q_1} - \langle \vec{a_3}, \vec{q_2} \rangle \vec{q_2}$	$\vec{q_3} = \dfrac{\vec{q_3}}{\lVert\vec{q_3}\rVert_2}$
$\vdots$	$\vdots$
$\vec{q_n} = \vec{a_n} - \sum_{i=1}^{n-1} \langle \vec{a_n}, \vec{q_i} \rangle \vec{q_i}$	$\vec{q_n} = \dfrac{\vec{q_n}}{\lVert\vec{q_n}\rVert_2}$

  - We can **express $\vec{a_i}$ in terms of** our newly computed **orthonormal basis $\vec{q_i}$**
    - $\vec{a_1} = \langle \vec{a_1}, \vec{q_1} \rangle \vec{q_1}$
    - $\vec{a_2} = \langle \vec{a_2}, \vec{q_1} \rangle \vec{q_1} + \langle \vec{a_2}, \vec{q_2} \rangle \vec{q_2}$
    - $\vec{a_3} = \langle \vec{a_3}, \vec{q_1} \rangle \vec{q_1} + \langle \vec{a_3}, \vec{q_2} \rangle \vec{q_2} + \langle \vec{a_3}, \vec{q_3} \rangle \vec{q_3}$
    - $\vdots$
    - $\vec{a_n} = \sum_{i=1}^{n} \langle \vec{a_n}, \vec{q_i} \rangle \vec{q_i}$
  - This can be written in matrix form

- $\underbrace{[\overrightarrow{a_1}, \dots, \overrightarrow{a_n}]}_{A} = \underbrace{[\overrightarrow{q_1}, \dots, \overrightarrow{q_n}]}_{Q} \underbrace{\begin{bmatrix} \langle \overrightarrow{a_1}, \overrightarrow{q_1} \rangle & \langle \overrightarrow{a_2}, \overrightarrow{q_1} \rangle & \langle \overrightarrow{a_3}, \overrightarrow{q_1} \rangle & \cdots & \langle \overrightarrow{a_n}, \overrightarrow{q_1} \rangle \\ & \langle \overrightarrow{a_2}, \overrightarrow{q_2} \rangle & \langle \overrightarrow{a_3}, \overrightarrow{q_2} \rangle & \cdots & \langle \overrightarrow{a_n}, \overrightarrow{q_2} \rangle \\ & & \langle \overrightarrow{a_3}, \overrightarrow{q_3} \rangle & \cdots & \langle \overrightarrow{a_n}, \overrightarrow{q_3} \rangle \\ & & & \ddots & \vdots \\ & & & & \langle \overrightarrow{a_n}, \overrightarrow{q_n} \rangle \end{bmatrix}}_{R}$

- Motivation for QR Factorization
  - Recall in least-square fitting, we obtain $\boldsymbol{\vec{x}} = \left(\boldsymbol{A^T A}\right)^{\boldsymbol{-1}} \boldsymbol{A^T \vec{b}}$
  - If we have factorized for A into QR, then
    - $\vec{x} = \left(A^T A\right)^{-1} A^T \vec{b}$
    - $A^T A \vec{x} = A^T \vec{b}$, by multiplying $A^T A$ on both sides
    - $R^T Q^T Q R \vec{x} = R^T Q^T \vec{b}$, by substituting $A = QR$
    - $R^T R \vec{x} = R^T Q^T \vec{b}$, since $Q^T Q = I$
    - $\boldsymbol{R\vec{x} = Q^T \vec{b}}$, if we assume R is not singular
  - Here, R is an upper-triangular matrix, and $Q^T \vec{b}$ is a column vector
  - It's **easy to solve for $\vec{x}$**, once we are **given the QR Factorization of A**

Norm & Condition Number

Wednesday, September 26, 2018 10:49 AM

Norm (Definition 2.6)

- Let $\mathcal{V}$ be a linear space, and $\|\cdot\|: \mathcal{V} \to \mathbb{R}_{\geq 0}$
- $\|\cdot\|$ is said to be a **norm** if
  - $\|\vec{\boldsymbol{v}}\| = \mathbf{0} \Leftrightarrow \vec{\boldsymbol{v}} = \vec{\mathbf{0}}$ (positive definite)
  - $\|\boldsymbol{\alpha}\vec{\boldsymbol{v}}\| = |\boldsymbol{\alpha}|\|\vec{\boldsymbol{v}}\|$
  - $\|\vec{\boldsymbol{v}} + \vec{\boldsymbol{w}}\| \leq \|\vec{\boldsymbol{v}}\| + \|\vec{\boldsymbol{w}}\|$ (triangle inequality)

Vector Norm (Definition 2.7 & 2.8 & 2.9)

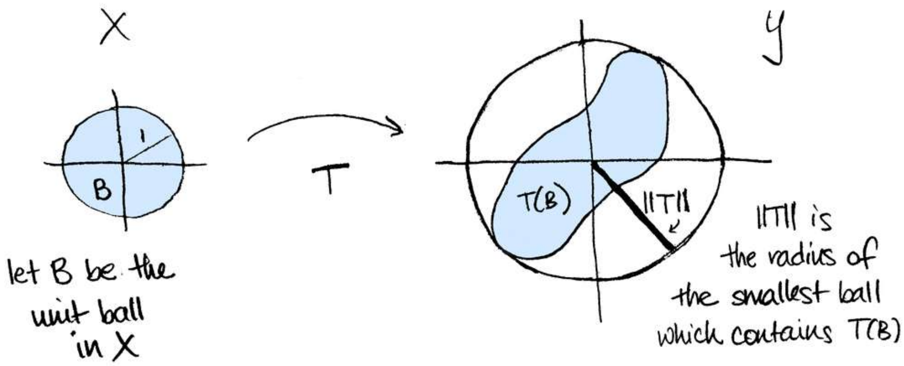
- Vector norms

Name	Formula
2-norm Euclidean norm	$\|\vec{v}\|_2 \coloneqq \sqrt{v_1^2 + \cdots + v_n^2} = \left[\sum v_i^2\right]^{1/2}$
1-norm Taxicab norm Manhattan norm	$\|\vec{v}\|_1 \coloneqq \lvert v_1\rvert + \cdots + \lvert v_n\rvert = \sum \lvert v_i\rvert$
∞-norm maximum norm	$\|\vec{v}\|_\infty \coloneqq \max_{i\in\{1,\dots,n\}} \lvert v_i\rvert$
p-norm	$\|\vec{v}\|_p \coloneqq \left[\sum \lvert v_i^p\rvert\right]^{1/p}$

- Minkowski's inequality
  - $\|\vec{u} + \vec{v}\|_p \leq \|\vec{u}\|_p + \|\vec{v}\|_p$
  - This proves the triangle inequality for p-norm

Matrix Norm (Definition 2.10)

- **Frobenius norm**
  - We can view matrix as a **list of number**, and define $\|A\|_F \coloneqq \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{i,j}^2}$
- **Operator norm / induced norm**
  - $\|A\|_{p,q} \coloneqq \sup_{\vec{x}\in\mathbb{R}^n\setminus\{\vec{0}\}} \frac{\|A\vec{x}\|_q}{\|\vec{x}\|_p}$
  - The matrix is viewed as a **linear transformation**
  - Operator norm is a means to **measure the "size"** of linear operators
  - Note that the operator norm has two parameter p and q



- In particular, if both parameters are equal to p, we simply call it $\boldsymbol{p}$**-norm**
  - $\|A\|_p := \sup_{\vec{x}\in\mathbb{R}^n\setminus\{\vec{0}\}} \frac{\|A\vec{x}\|_p}{\|\vec{x}\|_p}$
  - 1-norm, 2-norm and ∞-norm are defined similarly
- **Triangle inequality** for p-norm
  - Without loss of generality, suppose $\|\vec{x}\|_p = 1$
  - By triangle inequality of vector, $\|(A+B)\vec{x}\|_p \leq \|A\vec{x}\|_p + \|B\vec{x}\|_p$
  - Thus, $\frac{\|(A+B)\vec{x}\|_p}{\|\vec{x}\|_p} \leq \frac{\|A\vec{x}\|_p}{\|\vec{x}\|_p} + \frac{\|B\vec{x}\|_p}{\|\vec{x}\|_p} \overset{\sup}{\Longrightarrow} \|\boldsymbol{A}+\boldsymbol{B}\|_{\boldsymbol{p}} \leq \|\boldsymbol{A}\|_{\boldsymbol{p}} + \|\boldsymbol{B}\|_{\boldsymbol{p}}$
- $\|\boldsymbol{AB}\|_{\boldsymbol{p}} \leq \|\boldsymbol{A}\|_{\boldsymbol{p}}\|\boldsymbol{B}\|_{\boldsymbol{p}}$
  - By triangle inequality of vector,$\|AB\vec{x}\|_p \leq \|A\|_p\|B\|_p\|\vec{x}\|_p$
  - Thus, $\frac{\|AB\vec{x}\|_p}{\|\vec{x}\|_p} \leq \|A\|_p\|B\|_p \overset{\sup}{\Longrightarrow} \|AB\|_p \leq \|A\|_p\|B\|_p$

$\|A\|_1$ = Maximum Absolute Column Sum (Theorem 2.8)

- Statement
  - Given $A_{n\times m} = [\overrightarrow{a_1}, \dots, \overrightarrow{a_n}]$, then $\|A\|_1 = \max_{j\in\{1,\dots,n\}} \left\|\overrightarrow{a_j}\right\|_1 = \max_{\boldsymbol{j}\in\{\boldsymbol{1},\dots,\boldsymbol{n}\}} \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{m}} |\boldsymbol{a_{ij}}|$
- Note
  - The **1-norm** of matrix is also called **maximum absolute column sum**
- Proof
  - Let $C := \max_{j\in\{1,\dots,n\}} \left\|\overrightarrow{a_j}\right\|_1 = \max_{j\in\{1,\dots,n\}} \sum_{i=1}^{m} |a_{ij}|$
  - Show that $\|A\vec{x}\|_1 \leq C\|\vec{x}\|_1, \forall x \in \mathbb{R}^n \setminus \{\vec{0}\}$
    - $\|A\vec{x}\|_1 = \sum_{i=1}^{m} |(A\vec{x})_i|$, by definition of 1-norm of vector

      $= \sum_{i=1}^{m} \left|\sum_{j=1}^{n} a_{ij}x_j\right|$, by definition of $A\vec{x}$

ij

$$\leq \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}||x_j| \text{, by triangle inequality}$$

$$= \sum_{j=1}^{n} \left(\sum_{i=1}^{m} |a_{ij}| \right) |x_j| \text{, since only } |a_{ij}| \text{ depends on } i$$

$$\leq C \sum_{j=1}^{n} |x_j| \text{, by maximality of } C$$

$$= C\|\vec{x}\|_1 \text{, by definition of 1-norm of vector}$$

- Look for an $\vec{x}$ s.t. $\|A\vec{x}\|_1 = C\|\vec{x}\|_1$
  - Let $J \coloneqq \arg\max_{j \in \{1,\dots,n\}} \|\vec{a_j}\|_1$, then $\|\vec{a_J}\|_1 = C$
  - Let $\vec{x} \in \mathbb{R}^n$ s.t. $[\vec{x}]_k = \begin{cases} 1 & k = J \\ 0 & k \neq J \end{cases}$, then $\|\vec{x}\|_1 = 1$
  - Therefore $\|A\vec{x}\|_1 = \|\vec{a_J}\|_1 = C = C\|\vec{x}\|_1$

$\|A\|_\infty$ = Maximum Absolute Row Sum (Theorem 2.7)

- Statement
  - Given $A_{n \times m} = \begin{bmatrix} \vec{b_1} \\ \vdots \\ \vec{b_m} \end{bmatrix}$, then $\|A\|_\infty = \max_{i \in \{1,\dots,m\}} \|\vec{b_i}\|_1 = \boldsymbol{\max_{i \in \{1,\dots,m\}} \sum_{i=1}^{m} |a_{ij}|}$
- Note
  - The ∞**-norm** of matrix is also called **maximum absolute row sum**
- Proof
  - Let $C \coloneqq \max_{i \in \{1,\dots,m\}} \|\vec{b_i}\|_\infty = \max_{i \in \{1,\dots,m\}} \sum_{j=1}^{n} |a_{ij}|$
  - Show that $\|A\vec{x}\|_\infty \leq C\|\vec{x}\|_\infty, \forall x \in \mathbb{R}^n \setminus \{\vec{0}\}$
    - $\|A\vec{x}\|_\infty = \max_{i \in \{1,\dots,m\}} \left| \sum_{j=1}^{n} a_{ij}x_j \right|$, by definition of ∞-norm

$$\leq \max_{i \in \{1,\dots,m\}} \sum_{j=1}^{n} |a_{ij}||x_j| \text{, by the triangle inequality}$$

$$\leq \left[\max_{i \in \{1,\dots,m\}} \sum_{j=1}^{n} |a_{ij}| \right] \|\vec{x}\|_\infty \text{, by definition of } \infty\text{-norm}$$

$$= C\|\vec{x}\|_\infty$$

  - Look for an $\vec{x}$ s.t. $\|A\vec{x}\|_\infty = C\|\vec{x}\|_\infty$
    - Let $I = \operatorname{argmax}_{i \in \{1,\dots,m\}} \|\vec{b_i}\|_1$, then $\|\vec{b_I}\|_1 = \sum_{j=1}^{n} |a_{Ij}| = C$

Ij

- Let $\vec{x} \in \mathbb{R}^n$ s.t. $[\vec{x}]_j = \begin{cases} 1 & b_{Ij} > 0 \\ -1 & b_{Ij} < 0 \end{cases}$, then $\|\vec{x}\|_\infty = 1$
- Then $[A\vec{x}]_I = \overrightarrow{b_I}^T \vec{x} = \left|\sum_{j=1}^{n} a_{Ij} x_j\right| = \sum_{j=1}^{n} |a_{Ij}| = C = C\|\vec{x}\|_\infty$

$\|A\|_2$ = Largest Singular Value (Theorem 2.9)

- Positive-definite
  - A matrix A is said to be positive-definite if
  - All its **eigenvalues** are **positive**, and all the **eigenvector** is **orthonormal**
  - *i.e.* $\lambda_i \in \mathbb{R}^+$ and $\begin{cases} \|\overrightarrow{x_i}\|_2 = 1 \\ x_i \perp \overrightarrow{x_j} \end{cases} \Rightarrow \langle \overrightarrow{x_i}, \overrightarrow{x_j} \rangle = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$ for $A\overrightarrow{x_i} = \lambda_i \overrightarrow{x_i}$
- Symmetric
  - A matrix A is said to be **symmetric** if $\boldsymbol{A^T = A}$
- Statement (special case)
  - Assume $A_{n\times n}$ is a positive-definite symmetric matrix
  - Then $\|A\|_2 = \sup_{\vec{x} \in \mathbb{R}^n \setminus \{\vec{0}\}} \frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2} = \boldsymbol{\max_{i \in \{1,\dots,n\}} |\lambda_i|}$
- Proof (for special case)
  - Let $C = \sup_{\vec{x} \in \mathbb{R}^n \setminus \{\vec{0}\}} \frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2}$
  - Show $C \leq \max_{i \in \{1,\dots,n\}} |\lambda_i|$
    - Express $\vec{x}$ as a linear combination of the orthonormal vectors $\{x_i\}$
      - □ $\vec{x} = \sum c_i \vec{x}_i \Rightarrow \|\vec{x}\|_2 = \sqrt{\sum c_i^2}$
    - Similarly, express $A\vec{x}$ using $\{x_i\}$
      - □ $A\vec{x} = \sum c_i A\vec{x}_i = \sum c_i \lambda_i \vec{x}_i \Rightarrow \|A\vec{x}\|_2 = \sqrt{\sum c_i^2 \lambda_i^2}$
    - Thus, $\frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2} \leq \sqrt{\frac{\sum c_i^2 \lambda_i^2}{\sum c_i^2}} \leq \max_{i \in \{1,\dots,n\}} |\lambda_i|, \forall \vec{x} \in \mathbb{R}^n \setminus \{\vec{0}\}$
  - Look for an $\vec{x}$ s.t. $\|A\vec{x}\|_2 = C\|\vec{x}\|_2$
    - Let $I = \arg\max_{i \in \{1,\dots,n\}} |\lambda_i|$, then $|\lambda_I| = C$
    - Let $\vec{x} = \overrightarrow{x_I}$, then $\frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2} = \frac{\|A\overrightarrow{x_I}\|_2}{\|\overrightarrow{x_I}\|_2} = \sqrt{\frac{c_I^2 \lambda_I^2}{c_I^2}} = |\lambda_I| = C = C\|\vec{x}\|_2$
- Statement (General Case)
  - Define $B_{n\times n} = A_{n\times m}^T A_{m\times n}$, then B is a positive-definite symmetric matrix

- Let $S_i = \sqrt{\lambda_i}$, then S_i is called the **singular values** of A
- The previous statement can be generalized to $\|\boldsymbol{A}\|_2 = \max\limits_{i\in\{1,\ldots,n\}} |\boldsymbol{S_i}|$

Conditioning of Function (Example 2.5 & 2.6)

- Motivation
  - Suppose the input x has a **perturbation** of τ (because of machine percision)
  - We'd like to know how the output f will be affected by τ
  - **Condition** measures **the sensitivity** of the output **to perturbations** in the input
- **Absolute conditioning**
  - $\text{Cond}(f) = \sup\limits_{\substack{x,y\in\mathcal{D}\\ x\neq y}} \frac{|f(x)-f(y)|}{|x-y|}$
  - If f is differentiable, then $\text{Cond}(f) = \sup\limits_{x\in\mathcal{D}} |f'(x)|$
- **Absolute local conditioning**
  - $\text{Cond}_x(f) = \sup\limits_{\substack{|\delta x|\to 0\\ x+\delta x\in\mathcal{D}}} \frac{|f(x+\delta x)-f(x)|}{|\delta x|}$
  - If f is differentiable, then $\text{Cond}_x(f) = \begin{cases} |f'(x)| & \text{if } f \text{ is a scalar function} \\ |\nabla f(x)| & \text{if } f \text{ is a vector function} \end{cases}$
- **Relative local conditioning**
  - $\text{Cond}_x(f) = \sup\limits_{\substack{|\delta x|\to 0\\ x+\delta x\in\mathcal{D}}} \frac{|f(x+\delta x)-f(x)|/|f(x)|}{|\delta x|/|x|} = \sup\limits_{\substack{|\delta x|\to 0\\ x+\delta x\in\mathcal{D}}} \frac{|f(x+\delta x)-f(x)|}{|\delta x|}\frac{|x|}{|f(x)|}$
  - In particular, If f is differentiable, then $\text{Cond}_x(f) = \frac{|f'(x)|}{|f(x)|}|x|$
  - Motivation
    - $f(x) = 1, f(x+\delta x) = 2$
    - $g(x) = 100, g(x+\delta x) = 101$
    - Both f and g increased 1, but the effects are different!
- Example: $f(x) = \sqrt{x}$
  - Absolute
    - If $\mathcal{D} = [0,1]$, then $\text{Cond}(f) = +\infty$
    - If $\mathcal{D} = [1,2]$, then $\text{Cond}(f) = \frac{1}{2}$
  - Absolute local
    - $\text{Cond}_x(f) = f'(x) = \frac{1}{2\sqrt{x}} \to \begin{cases} \infty \text{ (ill-conditioned)} & \text{as } x\to 0 \\ 0 \text{ (well-conditioned)} & \text{as } x\to +\infty \end{cases}$
  - Relative local
    - $\text{Cond}_x(f) = \frac{|f'(x)|}{|f(x)|}|x| = \frac{1/(2\sqrt{x})}{\sqrt{x}}|x| = \frac{1}{2}, \forall x\in\mathcal{D}$

Condition Number of Matrix (Definition 2.12)

- Definition
  - $\boldsymbol{\kappa(A)} = \|\boldsymbol{A}\|\|\boldsymbol{A^{-1}}\|$ is called the **condition number** of A
  - If $\boldsymbol{\kappa(A)} \gg \mathbf{1}$, then we say A is **ill-conditioned**
- Note: $\kappa(A) = \kappa(A^{-1})$ and $\kappa(A) \geq 1$
- $\boxed{x(+\delta x)} \xrightarrow{A} \boxed{b(+\delta b)}$
  - $\begin{cases} Ax = b \\ A(x + \delta x) = b + \delta(b) \end{cases} \Rightarrow \delta b = A\delta x$
  - $\text{Cond}_x(A) = \dfrac{\|\delta b\|/\|b\|}{\|\delta x\|/\|x\|}$, by definition

    $= \dfrac{\|\delta b\|}{\|b\|} \cdot \dfrac{\|x\|}{\|\delta x\|}$

    $= \dfrac{\|A\delta x\|}{\|Ax\|} \cdot \dfrac{\|x\|}{\|\delta x\|}$, since $\delta b = A\delta x$ and $b = Ax$

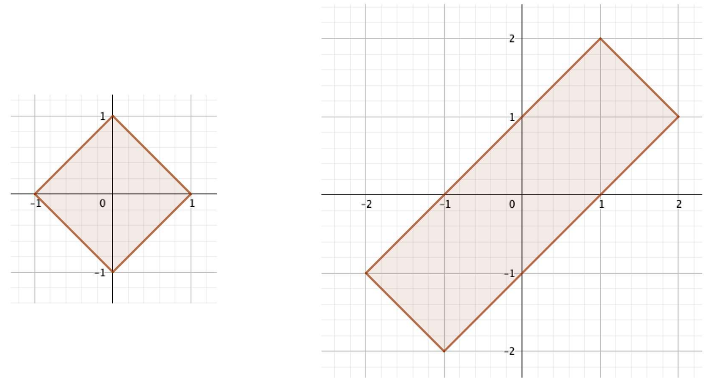
    $= \dfrac{\|A\delta x\|}{\|\delta x\|} \cdot \dfrac{\|A^{-1}b\|}{\|b\|}$, assuming A is not singular

    $\leq \|A\|\|A^{-1}\|$ by definition of matrix norm
- $\boxed{A(+\delta A)} \xrightarrow{b} \boxed{x(+\delta x)}$
  - $Ax = b$
  - $Ax = (A + \delta A)(x + \delta x)$, since b is viewed as the function here
  - $Ax = Ax + \delta Ax + A\delta x + \underbrace{\delta A\delta x}_{\approx 0}$, since $\delta A\delta x$ is a second order turbulence
  - $\delta Ax + A\delta x = 0$
  - $\delta x = -A^{-1} \cdot \delta A \cdot x$
  - $\|\delta x\| \leq \|A^{-1}\|\|\delta A\|\|x\|$, since $\|PQ\| \leq \|P\|\|Q\|$ for any matrix P, Q
  - $\dfrac{\|\delta x\|/\|x\|}{\|\delta A\|/\|A\|} = \dfrac{\|\delta x\|}{\|x\|} \cdot \dfrac{\|A\|}{\|\delta A\|} \leq \dfrac{\|A^{-1}\|\|\delta A\|\|x\|}{\|x\|} \dfrac{\|A\|}{\|\delta A\|} = \|A^{-1}\|\|A\|$
- We can similarly analyze $\boxed{b(+\delta b)} \xrightarrow{A} \boxed{x(+\delta x)}$ and $\boxed{A(+\delta A)} \xrightarrow{x} \boxed{b(+\delta b)}$
- Note: The **choice of norm** will affect the condition number
  - $A = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & & \ddots & \\ 1 & & & 1 \end{bmatrix} \Rightarrow A^{-1} = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ \vdots & & \ddots & \\ -1 & & & 1 \end{bmatrix}$
  - $\begin{cases} \|A\|_\infty = \|A^{-1}\|_\infty = 2 \\ \|A\|_1 = \|A^{-1}\|_1 = n \end{cases} \Rightarrow \begin{cases} \text{Cond}_{L_1}(A) = \|A\|_1\|A^{-1}\|_1 = n^2 \\ \text{Cond}_{L_\infty}(A) = \|A\|_\infty\|A^{-1}\|_\infty = 4 \end{cases}$

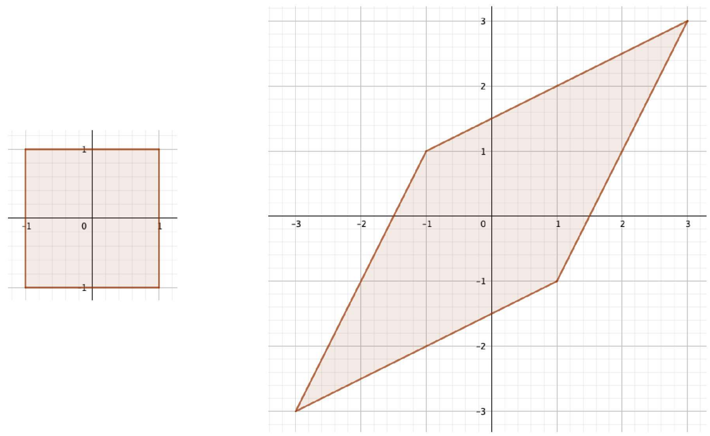
Example for Condition Number

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

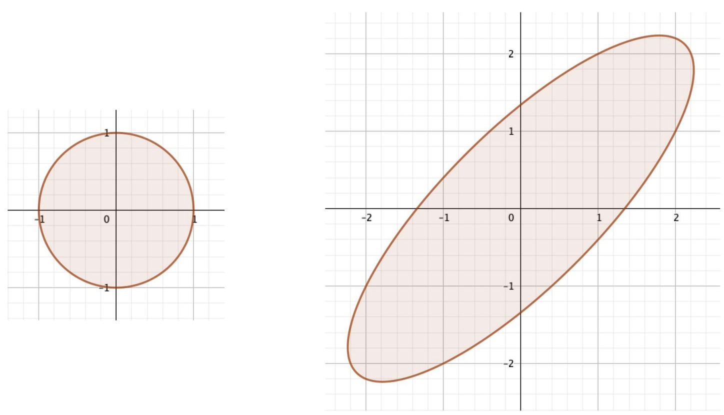
- 1 norm



- ∞ norm



- 2 norm



  - $S = \left\{ \begin{bmatrix} c \\ d \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} \middle| x^2 + y^2 = 1 \right\}$
  - Let $A^{-1} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$, then $\begin{bmatrix} x \\ y \end{bmatrix} = A^{-1} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} b_{11}c + b_{12}d \\ b_{21}c + b_{22}d \end{bmatrix}$
  - Since $x^2 + y^2 = 1$, we have $\alpha c^2 + \beta cd + \gamma d^2 = 1$, where
    - $\alpha = b_{11}^2 + b_{21}^2$
    - $\beta = 2b_{11}b_{12} + 2b_{21}b_{22}$
    - $\gamma = b_{12}^2 + b_{22}^2$
  - Since discriminant $= \beta^2 - 4\alpha\gamma < 0$, the graph S is an ellipse
  - $\kappa(A) = \dfrac{\text{length of major axis}}{\text{length of minor axis}} = \dfrac{S_{max}}{S_{min}}$, where S is the singular value

Ch 3: Special Matrices

Friday, September 28, 2018 10:37 AM

Symmetric Positive Definite Matrix

- Definition
  - Matrix A is called **symmetric positive definite** (s.p.d) if
  - $\boldsymbol{A = A^T}$ (symmetric)
  - $\boldsymbol{x^T A x > 0}, \forall x \in \mathbb{R}^{n\times n} \setminus \{0\}$ (positive definite)
- Proof: $a_{ii} > 0$
  - $\boldsymbol{a_{ii} = e_i^T A e_i} > 0$, where $[e_i]_k = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{if } k \neq i \end{cases}$
- Proof: $\lambda_i \in \mathbb{R}^+$ for $Ax_i = \lambda_i x_i$
  - Use $\overline{\lambda_i}$ to denote the conjugate of λ_i, we first need to show that $\overline{\lambda_i} = \lambda_i$
  - Taking conjugate on both sides of $Ax_i = \lambda_i x_i$, we obtain $A\overline{x_i} = \overline{\lambda_i}\overline{x_i}$ (note: $A = \bar{A}$)
  - $\begin{cases} x_i^T A \overline{x_i} = x_i^T(\overline{\lambda_i}\overline{x_i}) = \overline{\lambda_i} x_i^T \overline{x_i} \\ x_i^T A \overline{x_i} \overset{\text{sym}}{=} x_i^T A^T \overline{x_i} = (Ax_i)^T \overline{x_i} = \lambda_i x_i^T \overline{x_i} \end{cases} \Rightarrow \overline{\lambda_i} x_i^T \overline{x_i} = \lambda_i x_i^T \overline{x_i} \Rightarrow \boldsymbol{\overline{\lambda_i} = \lambda_i} \Rightarrow \boldsymbol{\lambda_i} \in \mathbb{R}$
  - $x_i^T A x_i = \lambda_i x_i^T x_i \Rightarrow \boldsymbol{\lambda_i} = \dfrac{\boldsymbol{x_i^T A x_i}}{\boldsymbol{x_i^T x_i}} > \boldsymbol{0}$, since $x_i^T A x_i > 0$ and $x_i^T x_i > 0$
  - Note: $\lambda_i \in \mathbb{R}$ holds for all symmetric matrices
- Proof: $\langle x_i, x_j \rangle = 0$ for $\lambda_i \neq \lambda_j$
  - $\begin{cases} x_i^T A x_j = x_i^T(\lambda_j x_j) = \lambda_j x_i^T x_j \\ x_i^T A^T x_j = (Ax_i)^T x_j = \lambda_i x_i^T x_j \end{cases} \Rightarrow \boldsymbol{(\lambda_j - \lambda_i) x_i^T x_j = 0}$
  - If $\lambda_i \neq \lambda_j$, then $\boldsymbol{x_i^T x_j = \langle x_i, x_j \rangle = 0}$
- Proof: $\det(A) > 0$
  - $A[\overrightarrow{x_1}, \overrightarrow{x_2}, \dots, \overrightarrow{x_n}] = [\lambda_1\overrightarrow{x_1}, \lambda_2\overrightarrow{x_2}, \dots, \lambda_n\overrightarrow{x_n}] = [\overrightarrow{x_1}, \overrightarrow{x_2}, \dots, \overrightarrow{x_n}]\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$
  - Let $X = [\overrightarrow{x_1}, \overrightarrow{x_2}, \dots, \overrightarrow{x_n}], \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$, then $AX = X\Lambda \Rightarrow \boldsymbol{A = X\Lambda X^{-1}}$
  - Therefore, $|A| = |X\Lambda X^{-1}| = |X||\Lambda||X|^{-1} = |\boldsymbol{\Lambda}| = \displaystyle\prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{\lambda_i > 0}$
- Proof: Let $I \subsetneq \{1,2,\dots,n\}$, then $B = A_{II}$ is also s.p.d.
  - $A = A^T \Rightarrow A_{II} = A_{II}^T \Rightarrow B = B^T$
  - Define $y \in \mathbb{R}^n$ s.t. $[\boldsymbol{y}]_{\boldsymbol{i}} = \begin{cases} [\boldsymbol{x}]_{\boldsymbol{i}} & \boldsymbol{i \in I} \\ \boldsymbol{0} & \boldsymbol{o.w.} \end{cases}$, then $\boldsymbol{x^T B x = y^T A y > 0}, \forall \boldsymbol{x} \in \mathbb{R}^{|\boldsymbol{I}|} \setminus \{\boldsymbol{0}\}$

- Cholesky Decomposition
  - If A is s.p.d., then $\exists L$ lower diagonal s.t. $A = LL^T$
  - Note: This is saying that after LU decomposition, $U = L^T$

Ordinary Differential Equation (Boundary Value Problem)

- Suppose $u(x) \in \mathcal{C}^2[0,1]$, find the solution for $\begin{cases} u'' + 2u' = -1 \\ u(x=0) = 0 \\ u(x=1) = 0 \end{cases}$
- We can evenly **sample N points** on $[0,1]$: $\Delta x = \dfrac{1}{N+1}, x_i = i\Delta x$
- Compute first derivative $u'(x_j)$ using $u(x_{j+1})$ and $u(x_{j-1})$
  - $u'(x_j) = \lim_{\delta \to 0} \dfrac{u(x_j + \delta) - u(x_j - \delta)}{2\delta} \approx \dfrac{u(x_{j+1}) - u(x_{j-1})}{2\Delta x}$
  - Note: $\boldsymbol{Du}\Big|_j = \dfrac{\boldsymbol{u(x_{j+1}) - u(x_{j-1})}}{\boldsymbol{2\Delta x}}$ is called the **discrete derivative** of u at j
- Compute second derivative $u''(x_j)$ using $u(x_{j+2}), u(x_j)$ and $u(x_{j-2})$

$$\begin{aligned} u''(x_j) &= \lim_{\delta \to 0} \frac{u'(x_j + \delta) - u'(x_j - \delta)}{2\delta} \approx \frac{u'(x_{j+1}) - u'(x_{j-1})}{2\Delta x} \\ &\approx \frac{\left(\frac{u(x_{j+2}) - u(x_j)}{2\Delta x}\right) - \left(\frac{u(x_j) - u(x_{j-2})}{2\Delta x}\right)}{2\Delta x}, \text{ by substituting } u' \\ &= \frac{u(x_{j+2}) - 2u(x_j) + u(x_{j-2})}{4\Delta x^2} \end{aligned}$$

- Compute second derivative $u''(x_j)$ using $u(x_{j+1}), u(x_j)$ and $u(x_{j-1})$
  - In practice, we want to only use neighboring points to have a local approximation
  - Thus, $u''(x_j) \approx \dfrac{u'(x_{j+1/2}) - u'(x_{j-1/2})}{\Delta x} = \dfrac{\boldsymbol{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}}{\boldsymbol{\Delta x^2}}$
- Substitute u', u'' into the ODE
  - $\dfrac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{\Delta x^2} + 2\left(\dfrac{u(x_{j+1}) - u(x_{j-1})}{2\Delta x}\right) \approx -1$
  - Define $U := \begin{bmatrix} u(x_1) \\ \vdots \\ u(x_n) \end{bmatrix}$, then $\dfrac{U_{j+1} - 2U_j + U_{j-1}}{\Delta x^2} + \dfrac{U_{j+1} - U_{j-1}}{\Delta x} = -1$
  - Let $A \in \mathbb{R}^{n \times n}\ s.t.\ [A]_{ij} = \begin{cases} (\Delta x^2)^{-1} - (\Delta x)^{-1} & i = j-1 \\ -2(\Delta x^2)^{-1} & i = j \\ (\Delta x^2)^{-1} + (\Delta x)^{-1} & i = j+1 \\ 0 & o.w. \end{cases}$, then $\boldsymbol{AU = -1}$
  - Note: A is said to be a **tri-diagonal** matrix $\begin{bmatrix} * & * & & & \\ * & * & * & & \\ & * & * & * & \\ & & * & * & * \\ & & & * & * \end{bmatrix}$

Ch 4: Simultaneous Iteration

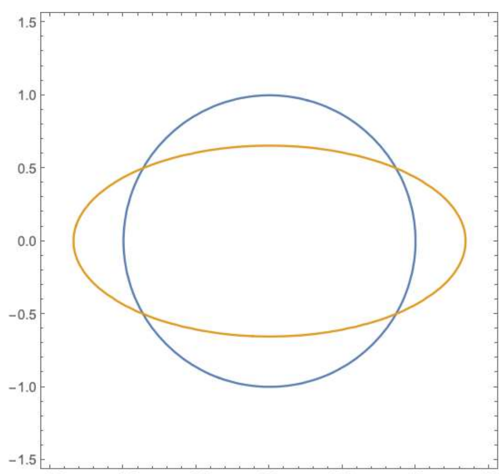
Wednesday, October 3, 2018 10:24 AM

Continuous Functions Preserve Convergence For Cauchy Sequence

- Cauchy sequence
  - In $D \subseteq \mathbb{R}^n$, $\{\vec{x}^{(k)}\}_{k=0}^{+\infty}$ is called a **Cauchy sequence** if
  - $\forall \varepsilon > 0, \exists k_\varepsilon > 0$ s.t. $\left\|\vec{\boldsymbol{x}}^{(\boldsymbol{m})} - \vec{\boldsymbol{x}}^{(\boldsymbol{n})}\right\|_\infty < \boldsymbol{\varepsilon}$ $(\forall m, n > k_\varepsilon)$
  - Note: $\mathbb{R}^n$ is complete since every Cauchy sequence converges to some point in $\mathbb{R}^n$
- Continuity
  - Given $\xi \in D \subseteq \mathbb{R}^n$, $f: D \to \mathbb{R}^n$ is said to be **continuous** if
  - $\forall \varepsilon > 0, \exists \delta_\varepsilon > 0$ s.t. $\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{\xi})\|_\infty < \boldsymbol{\varepsilon}$ $\left(\forall x \in B(\xi; \delta_\varepsilon)\right)$
  - Here, $B(\xi; \delta_\varepsilon)$ is an open ball at ξ with radius δ_ε
- Lemma
  - If $\vec{f}: D(\subseteq \mathbb{R}^n) \to \mathbb{R}^n$ is **continuous**, and $\{\vec{x}^{(k)}\} \to \vec{\xi} \in D$ is a **Cauchy sequence**
  - Then $\vec{f}(\vec{x}^{(k)})$ also **converges** to $\vec{f}\left(\vec{\xi}\right)$

Introduction to Simultaneous Nonlinear Equations

- Given $\vec{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$, where $f_i: \mathbb{R}^n \to \mathbb{R}$, we want to look for $\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ s.t. $\vec{f}(\vec{x}) = \vec{0}$
- In general, we don't know whether such root exists, but we can solve for some special cases
  - If $\vec{f}$ is linear $\left(\text{i.e. } \vec{f}(\vec{x}) = A\vec{x} - \vec{b}\right)$, we can use the knowledge from Chapter 2
  - For $\vec{f}(\vec{x}) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2^2 - 1 \\ 5x_1^2 + 21x_2^2 - 9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \pm\sqrt{3}/2 \\ \pm 1/2 \end{bmatrix}$



- Solving $\vec{f}(\vec{x}) = \vec{0}$ is the extension of solving $f(x) = 0$ from Chapter 1
- In Chapter 1, we are given $\boldsymbol{f}(\boldsymbol{x}): \boldsymbol{D}(\subseteq \mathbb{R}) \to \mathbb{R}$, and asked to find a $x \in \mathbb{R}$ s.t. $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$
  - We transformed this problem to a fixed point finding problem

- Define g s.t. $g(x) = x \Leftrightarrow f(x) = 0$ $\left(\text{e.g. } g(x) = x - f(x) \text{ or } g(x) = x - \frac{f(x)}{f'(x)}\right)$
- Start with initial guess x_0 and iterate $x_k \coloneqq g(x_{k-1})$
- We used contraction mapping theorem to show the iterative method converges to ξ

- In order to solve for $\vec{f}(\vec{x}) = \vec{0}$, we need to
  - **Design a function $\vec{g}$** s.t. $\vec{f}(\vec{x}) = \vec{0} \Leftrightarrow \vec{g}(\vec{x}) = \vec{x}$
  - Start with $\vec{x}^{(0)} \in D \subseteq \mathbb{R}^n$, and **iterate $\vec{x}^{(k)} = \vec{g}\left(\vec{x}^{(k-1)}\right)$**
  - Show the sequence $\left\{\vec{x}^{(k)}\right\}$ **converge to $\vec{\xi}$**

Simultaneous Iteration

- Given $\boldsymbol{g}: \boldsymbol{D}(\subseteq \mathbb{R}^{\boldsymbol{n}}) \to \mathbb{R}^{\boldsymbol{n}}$ s.t. $\boldsymbol{g}(\boldsymbol{D}) \subseteq \boldsymbol{D}$, and let $\vec{x}^0 \in D$
- The recursion defined by $\vec{x}^{(k)} = \vec{g}\left(\vec{x}^{(k-1)}\right)$ is called a **simultaneous iteration**
- For $n = 1$, this is just simple iteration in Chapter 1

Example of Simultaneous Iteration

- Given $\vec{g}: (0,1)^2 \to \mathbb{R}^n$ defined as $\vec{g}(\vec{x}) = \frac{1}{2}(\vec{x} + \vec{u})$, where $\vec{u} = \begin{pmatrix}1\\1\end{pmatrix}$
- We'd like to find a fixed point $\vec{\xi} \in D \coloneqq [0,1]^2$
- Algebraic method: $\vec{g}\left(\vec{\xi}\right) = \frac{1}{2}\left(\vec{\xi} + \vec{u}\right) = \vec{\xi} \Rightarrow \vec{\xi} = \vec{u}$
- Numeric method: Do **simultaneous iteration** with initial value of $\vec{x}^{(0)} = \begin{pmatrix}0\\0\end{pmatrix}$
- Check $\vec{x}^{(k-1)} \in D \Rightarrow \vec{x}^{(k)} \in D$
  - $\vec{x}^{(k)} = \begin{pmatrix}x^k\\y^k\end{pmatrix} = \frac{1}{2}\left(\vec{x}^{(k-1)} + \vec{u}\right) = \frac{1}{2}\left(\begin{pmatrix}x^{k-1}\\y^{k-1}\end{pmatrix} + \begin{pmatrix}1\\1\end{pmatrix}\right) = \begin{pmatrix}x^{k-1}/2 + 1\\y^{k-1}/2 + 1\end{pmatrix}$
  - $\vec{x}^{(k-1)} \in D \Rightarrow \begin{cases}x^{k-1} \in (-1,1)\\y^{k-1} \in (-1,1)\end{cases} \Rightarrow \begin{cases}x^{k-1}/2 + 1 \in (0,1)\\y^{k-1}/2 + 1 \in (0,1)\end{cases} \Rightarrow \begin{cases}x^k \in (-1,1)\\y^k \in (-1,1)\end{cases} \Rightarrow \vec{\boldsymbol{x}}^{(\boldsymbol{k})} \in \boldsymbol{D}$
- Check sequence converge to the fixed point
  - $\underbrace{\left\|\vec{x}^{(k)} - \vec{u}\right\|}_{E_k} = \left\|g\left(\vec{x}^{(k-1)} - \vec{u}\right)\right\| = \left\|\frac{1}{2}\left(\vec{x}^{(k-1)} + \vec{u}\right) - \vec{u}\right\| = \frac{1}{2}\underbrace{\left\|\vec{x}^{(k-1)} - \vec{u}\right\|}_{E^{k-1}}$
  - $\boldsymbol{E^k} = \frac{1}{2}E^{k-1} = \left(\frac{1}{2}\right)^k E^0 = \left(\frac{1}{2}\right)^k \to \boldsymbol{0}$ as $k \to \infty$

Contraction Mapping Theorem

- Lipschitz continuity
  - Given $g: D(\subseteq \mathbb{R}^n) \to D(\subseteq \mathbb{R}^n)$
  - We say g is **Lipschitz continuous** if $\|\vec{\boldsymbol{g}}(\vec{\boldsymbol{x}}) - \vec{\boldsymbol{g}}(\vec{\boldsymbol{y}})\|_\infty \le \boldsymbol{L}\|\vec{\boldsymbol{x}} - \vec{\boldsymbol{y}}\|_\infty, \forall \vec{\boldsymbol{x}}, \vec{\boldsymbol{y}} \in \boldsymbol{D}$
  - Here L is called **Lipschitz constant**
  - If $L < 1$, then we say g is a **contraction map**

- Contraction mapping theorem
  - Suppose $\boldsymbol{D} \subseteq \mathbb{R}^{\boldsymbol{n}}$ **closed**, $\vec{g}: D \to \mathbb{R}^n$ is a **contraction** map in ∞-norm and $\boldsymbol{g}(\boldsymbol{D}) \subseteq \boldsymbol{D}$
  - Then $\exists!\, \vec{\boldsymbol{\xi}} \in \boldsymbol{D}$ **s.t.** $\vec{\boldsymbol{g}}\left(\vec{\boldsymbol{\xi}}\right) = \vec{\boldsymbol{\xi}}$, and $\left\{\vec{\boldsymbol{x}}^{(\boldsymbol{k})} = \vec{\boldsymbol{g}}\left(\vec{\boldsymbol{x}}^{(\boldsymbol{k}-\mathbf{1})}\right)\right\} \to \vec{\boldsymbol{\xi}}, \forall \boldsymbol{x}^{(\mathbf{0})} \in \boldsymbol{D}$
- Proof
  - Note: In the proof below, we assume the existence of ξ for the first two parts.
  - Uniqueness of fixed points
    - Suppose $\vec{\eta}, \vec{\xi}$ are both fixed points of $\vec{g}$ (*i.e.* $\vec{g}(\vec{\eta}) = \vec{\eta}$ and $\vec{g}\left(\vec{\xi}\right) = \vec{\xi}$)
    - Then $\left\|\vec{\eta} - \vec{\xi}\right\|_\infty = \left\|\vec{g}(\vec{\eta}) - \vec{g}\left(\vec{\xi}\right)\right\|_\infty < L\left\|\vec{\eta} - \vec{\xi}\right\|_\infty$ by Lipschitz condition
    - Therefore, $\underbrace{(1-L)}_{>0} \underbrace{\left\|\vec{\eta} - \vec{\xi}\right\|_\infty}_{\geq 0} \leq 0 \Rightarrow \left\|\vec{\eta} - \vec{\xi}\right\|_\infty = 0 \Rightarrow \vec{\boldsymbol{\eta}} = \vec{\boldsymbol{\xi}}$
  - **If $\vec{\boldsymbol{\xi}}$ exists**, then $\{\vec{x}^{(k)}\}$ converges to $\vec{\xi}$
    - $\underbrace{\left\|\vec{x}^{(k+1)} - \vec{\xi}\right\|_\infty}_{E^{k+1}} = \left\|\vec{g}(\vec{x}^{(k)}) - \vec{g}\left(\vec{\xi}\right)\right\|_\infty \leq L \underbrace{\left\|\vec{x}^{(k)} - \vec{\xi}\right\|_\infty}_{E^k}$
    - Expand the inequality, we have $E^{k+1} \leq LE^k \leq \cdots \leq L^{k+1}E^0$
    - Compute E^0 (optional)
      - $\underbrace{\left\|\vec{x}^{(0)} - \vec{\xi}\right\|_\infty}_{E^0} = \left\|\vec{x}^{(0)} - \vec{x}^{(1)} + \vec{x}^{(1)} - \vec{\xi}\right\|_\infty$

        $\leq \left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty + \left\|\vec{x}^{(1)} - \xi\right\|_\infty$

        $\leq \left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty + L\underbrace{\left\|\vec{x}^{(0)} - \vec{\xi}\right\|_\infty}_{E^0}$
      - $E_0 \leq \left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty + LE_0$
      - $E_0 \leq \dfrac{1}{1-L}\left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty$
    - Therefore $E^{k+1} \leq L^{k+1}\dfrac{1}{1-L}\left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty$
    - Since $L \in (0,1)$, as $k \to \infty$, we have $E^k \to 0 \Leftrightarrow x^{(k)} \to \xi$
  - Existence of $\vec{\xi}$ (by showing $\{\vec{x}^{(k)}\}$ is a Cauchy sequence)
    - Assume $m > n$
    - $\left\|\vec{x}^{(m)} - \vec{x}^{(n)}\right\|_\infty = \left\|\vec{x}^{(m)} - \vec{x}^{(m-1)} + \vec{x}^{(m-1)} - \vec{x}^{(m-2)} + \vec{x}^{(m-2)} + \cdots - \vec{x}^{(n)}\right\|_\infty$

      $= \underbrace{\left\|\vec{x}^{(m)} - \vec{x}^{(m-1)}\right\|_\infty}_{\leq L^{m-1}\left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty} + \underbrace{\left\|\vec{x}^{(m-1)} - \vec{x}^{(m-2)}\right\|_\infty}_{\leq L^{m-2}\left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty} + \cdots + \underbrace{\left\|\vec{x}^{(n+1)} - \vec{x}^{(n)}\right\|_\infty}_{\leq L^{n}\left\|\vec{x}^{(0)} - \vec{x}^{(1)}\right\|_\infty}$

      $\leq \left(L^{m-1} + L^{m-2} + \cdots + L^n\right)\left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty$

      $= L^n\left(L^{m-n-1} + L^{m-n-2} + \cdots + 1\right)\left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty$

      $\leq L^n\dfrac{1}{1-L}\left\|\vec{x}^{(1)} - \vec{x}^{(0)}\right\|_\infty$
    - Therefore $\left\|\vec{x}^{(m)} - \vec{x}^{(n)}\right\|_\infty \to 0$ as $n \to +\infty$ *i.e.* $\{\vec{\boldsymbol{x}}^{(\boldsymbol{k})}\}$ **is a Cauchy sequence**

- Due to **completeness** of D, it **converges** to some point; call it $\vec{\xi}$
  - Note: In 1D, the existence of $\vec{\xi}$ is guaranteed by the Intermediate Value Theorem

Jacobian Matrix

- Definition
  - Suppose $\vec{g} = [g_1, \dots, g_n]^T : \mathbb{R}^n \to \mathbb{R}^n, \vec{g} \in C^1$, and $\frac{\partial g_i}{\partial x_j}$ exists at $\vec{\xi}, \forall i, j \in \{1, \dots, n\}$
  - Then **Jacobian matrix** $J_{\vec{g}}\left(\vec{\xi}\right)$ of $\vec{g}$ is defined as $\left[\boldsymbol{J_{\vec{g}}\left(\vec{\xi}\right)}\right]_{i,j} = \boldsymbol{\frac{\partial g_i}{\partial x_j}\left(\vec{\xi}\right)}$
- Theorem
  - Suppose $\vec{g}: D(\subseteq \mathbb{R}^n) \to \mathbb{R}^n$ and $\vec{g} \in C^1$. Let $\vec{\xi} \in D$ be a fixed point of $\vec{g}$
  - If $\left\|\boldsymbol{J_{\vec{g}}\left(\vec{\xi}\right)}\right\|_\infty < \mathbf{1}$ (in a small neighborhood of ξ, $\vec{g}$ is a contraction map)
  - then $\{\boldsymbol{\vec{x}^{(k+1)} = \vec{g}(\vec{x}^{(k)})}\}$ **converges to** $\boldsymbol{\vec{\xi}}$ given $\vec{x}^{(0)}$ is close enough to ξ
- Example
  - $\vec{g}(\vec{x}) = \begin{bmatrix} g_1(x_1, x_2) \\ g_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2^2 - 1 \\ 5x_1^2 + 21x_2^2 - 9 \end{bmatrix} \Longrightarrow J_{\vec{g}} = \begin{bmatrix} 2x_1 & 2x_2 \\ 10x_1 & 42x_2 \end{bmatrix}$
  - $\|J_{\vec{g}}\|_\infty = \max\{2|x_1| + 2|x_2|, 10|x_1| + 42|x_2|\} = 10|x_1| + 42|x_2|$

Newton's method

- Definition
  - $\vec{g}(\vec{x}) = \vec{x} - \left[J_{\vec{f}}(\vec{x})\right]^{-1} \vec{f}(\vec{x})$
- Example
  - $\vec{f}(\vec{x}) = \begin{bmatrix} x^2 + y^2 + z^2 - 1 \\ 2x^2 + y^2 - 4z \\ 3x^2 - 4y + z^2 \end{bmatrix} \Longrightarrow J_{\vec{f}} = \begin{bmatrix} 2x & 2y & 2z \\ 4x & 2y & -4 \\ 6x & -4 & 2z \end{bmatrix}$
  - $\vec{g}(\vec{x}) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} 2x & 2y & 2z \\ 4x & 2y & -4 \\ 6x & -4 & 2z \end{bmatrix}^{-1} \begin{bmatrix} x^2 + y^2 + z^2 - 1 \\ 2x^2 + y^2 - 4z \\ 3x^2 - 4y + z^2 \end{bmatrix}$
- Theorem
  - Suppose $\vec{g}: D(\subseteq \mathbb{R}^n) \to \mathbb{R}^n$ and $\vec{g} \in C^1$
  - Let $\vec{\xi} \in D$ be a fixed point of $\vec{g}$
  - If all $\partial_i \partial_j \vec{f}$ is continuous, and $\boldsymbol{J_{\vec{g}}\left(\vec{\xi}\right)}$ is non-singular
  - Then $\{\boldsymbol{\vec{x}^{(k+1)} = \vec{g}(\vec{x}^{(k)})}\}$ **converges to** $\boldsymbol{\vec{\xi}}$ given $\boldsymbol{\vec{x}^{(0)}}$ **is close enough to** $\boldsymbol{\xi}$

Ch 5: Eigenvalue Decomposition

Wednesday, October 10, 2018 9:58 AM

Eigenvalue Decomposition

- Introduction
  - If $A_{n\times n}$ has eigenvectors $\overrightarrow{x_1}, \dots, \overrightarrow{x_n}$ with corresponding eigenvalues $\lambda_1, \dots, \lambda_n$
  - Then $A\overrightarrow{x_i} = \lambda_i \overrightarrow{x_i} \Leftrightarrow \boldsymbol{A} \underbrace{[\overrightarrow{\boldsymbol{x_1}}, \dots, \overrightarrow{\boldsymbol{x_n}}]}_{\boldsymbol{X}} = \underbrace{[\overrightarrow{\boldsymbol{x_1}}, \dots, \overrightarrow{\boldsymbol{x_n}}]}_{\boldsymbol{X}} \underbrace{\begin{bmatrix} \boldsymbol{\lambda_1} & & \\ & \ddots & \\ & & \boldsymbol{\lambda_n} \end{bmatrix}}_{\boldsymbol{\Lambda}}$
  - This gives the **eigenvalue decomposition** $\boldsymbol{A = X\Lambda X^{-1}}$ (assuming X is not singular)
  - In this chapter, let's further assume that $\boldsymbol{A}$ **is symmetric**, then $\boldsymbol{x_i} \perp \boldsymbol{x_j}$, and $\boldsymbol{\lambda_i} \in \mathbb{R}$
- List of Matrix Decompositions

Name	Formula	Procedure
LU decomposition	$A = LU$	Gauss-elimination
QR decomposition	$A = QR$	Gram-Schmidt process
Eigenvalue decomposition	$A = X\Lambda X^{-1}$	???

- "No-Go Theorem" (Abel Theorem)
  - There is **no finite procedure** that provides eigenvalue decomposition
  - Finding the eigenvalues is equivalent to **solving the characteristic equation**
    - $A\vec{x} = \lambda\vec{x} \Leftrightarrow (A - \lambda I)\vec{x} = 0 \Leftrightarrow \vec{x} \in \text{Null}(A - \lambda I) \Leftrightarrow p(\lambda) \coloneqq \det(A - \lambda I) = 0$
  - Abel-Ruffini Theorem: **No explicit root formula** for polynomial of **degree 5 or higher**

Power Iteration

- General Idea
  - Suppose $A\overrightarrow{x_i} = \lambda_i \overrightarrow{x_i}$ for $i \in \{1 \dots n\}$, and $\underbrace{|\lambda_1| > |\lambda_2|}_{\text{strictly larger}} \geq \cdots \geq |\lambda_n|$
  - Choose arbitrary $\vec{v} \in \mathbb{R}^n$, then $\vec{v} = c_1\overrightarrow{x_1} + \cdots + c_n\overrightarrow{x_n}$ for some coefficients $c_1, \dots, c_n$
  - $A^k\vec{v} = A^k \sum_{i=1}^{n} c_i\overrightarrow{x_i} = \sum_{i=1}^{n} c_i\left(A^k\overrightarrow{x_i}\right) = \sum_{i=1}^{n} c_i\lambda_i^k\overrightarrow{x_i} = \underbrace{c_1\lambda_1^k\overrightarrow{x_1}}_{\gg \text{others}} + \cdots + c_n\lambda_n^k\overrightarrow{x_n}$
  - Since $|\lambda_1|$ the the largest eigenvalue, $\left|\boldsymbol{c_1\lambda_1^k}\right|$ **is signficiently larger** than the rest
- Algorithm
  - Choose $v^{(0)} \in \mathbb{R}^n$ s.t. $\left\|v^{(0)}\right\|_2 = 1$
  - For $k = 1,2, \dots$
    - $w \leftarrow Av^{(k-1)}$ Apply A
    - $v^{(k)} \leftarrow w/\|w\|_2$ Normalization
    - $\lambda^{(k)} \leftarrow \left\langle v^{(k)}, Av^{(k)}\right\rangle$ Compute Rayleigh quotient
- Convergence rate for $v^{(k)}$

- Claim: $\left\|\boldsymbol{v}^{(\boldsymbol{k})} - (\pm\overrightarrow{\boldsymbol{x_1}})\right\| = \boldsymbol{O}\left(\left|\frac{\boldsymbol{\lambda_2}}{\boldsymbol{\lambda_1}}\right|^{\boldsymbol{k}}\right)$
- $v^{(k)} = \alpha_k A^k \vec{v}^{(0)}$, for some normalization constant α_k

  $= \alpha_k\left(c_1\lambda_1^k\overrightarrow{x_1} + c_2\lambda_2^k\overrightarrow{x_2} + \cdots + c_n\lambda_n^k\overrightarrow{x_n}\right)$ for some stretching coefficients $c_1, \ldots, c_n$

  $= \alpha_k\lambda_1^k\left[c_1\overrightarrow{x_1} + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\overrightarrow{x_2} + \cdots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\overrightarrow{x_n}\right]$
- Therefore the error term is approximately $O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$ given that $c_1 \neq 0$

- Convergence rate for $\lambda^{(k)}$
  - Claim: $\left|\boldsymbol{\lambda}^{(\boldsymbol{k})} - \boldsymbol{\lambda_1}\right| = \boldsymbol{O}\left(\left|\frac{\boldsymbol{\lambda_2}}{\boldsymbol{\lambda_1}}\right|^{\boldsymbol{2k}}\right)$
  - If $\|\vec{x} - \overrightarrow{x_1}\| = O(\varepsilon)$, then $\left|\frac{\langle\vec{x}, A\vec{x}\rangle}{\langle\vec{x}, \vec{x}\rangle} - \lambda_1\right| = O(\varepsilon^2)$
  - Here, $\frac{\langle\vec{x}, A\vec{x}\rangle}{\langle\vec{x}, \vec{x}\rangle}$ is called Rayleigh quotient

Variations of Power Iteration

- Power iteration only picks the **largest eigenvalue**. What if we want to find other ones?
- If we want to find the **smallest eigenvalue**, then can use **inverse power iteration**
- For finding a eigenvalue **closest to some number**, we can use **shifted power iteration**

Simultaneous Iteration and QR Iteration

- Goal
  - Obtain the **full set of eigenvalues and eigenvectors** simultaneously
- General idea for simultaneous iteration
  - Suppose $A\overrightarrow{x_i} = \lambda_i\overrightarrow{x_i}$ for $i \in \{1 \ldots n\}$, and $\underbrace{|\lambda_1| > |\lambda_2|}_{\text{strictly larger}} \geq \cdots \geq |\lambda_n|$
  - Arbitrarily choose $V = [\overrightarrow{v_1}, \ldots, \overrightarrow{v_n}] \in \mathbb{R}^{n\times n}$, then
  - $\overrightarrow{v_i} = c_{1i}\overrightarrow{x_1} + c_{2i}\overrightarrow{x_1} + \cdots + c_{ni}\overrightarrow{x_n}$ for some stretching coefficients $c_{1i}, \ldots, c_{ni}$
  - $A^kV = \left[\sum_{i=1}^{n}\lambda_i^k c_{1i}\overrightarrow{x_i}, \sum_{i=1}^{n}\lambda_i^k c_{2i}\overrightarrow{x_i}, \ldots, \sum_{i=1}^{n}\lambda_i^k c_{ni}\overrightarrow{x_i}\right]$
  - If we use $\left[A^kV\right]_i$ to denote the i-th column of A^kV, then
    - $\left[A^kV\right]_1 \to \overrightarrow{x_1}$
    - $\left[A^kV\right]_2 \to \overrightarrow{x_1} + O\left[\left(\frac{\lambda_2}{\lambda_1}\right)^k\right]\overrightarrow{x_2}$
    - $\left[A^kV\right]_3 \to \overrightarrow{x_1} + O\left[\left(\frac{\lambda_2}{\lambda_1}\right)^k\right]\overrightarrow{x_2} + O\left[\left(\frac{\lambda_3}{\lambda_1}\right)^k\right]\overrightarrow{x_2}$

- ⋮
  - We can use QR factorization to obtain $\overrightarrow{x_1}, \dots, \overrightarrow{x_n}$
- Algorithm: Simultaneous iteration
  - Let $\underline{Q}^{(0)} \leftarrow I$
  - For $k \leftarrow 1,2, \dots$
    - $Z \leftarrow A\underline{Q}^{(k-1)}$ Apply A
    - $Z \rightarrow \underline{Q}^{(k)} R^{(k)}$ Normalization by QR factorization
    - $A^{(k)} \leftarrow \left[\underline{Q}^{(k)}\right]^T A\underline{Q}^{(k)}$ Compute Rayleigh quotient
- Algorithm: QR iteration
  - Let $A^{(0)} \leftarrow A$
  - For $k \leftarrow 1,2, \dots$
    - $A^{(k-1)} \rightarrow Q^{(k)} R^{(k)}$ QR factorization
    - $A^{(k)} \leftarrow R^{(k)} Q^{(k)}$ Recombine factors in reverse order
  - $\underline{Q}^{(k)} \leftarrow Q^{(1)} Q^{(2)} \cdots Q^{(k)}$
- Convergence rate
  - $\left\| q_i^{(k)} - (\pm x_i) \right\| = O\left(C^k\right)$ and $\left| A_{ii}^{(k)} - \lambda_i \right| = O\left(C^{2k}\right)$ where $C = \max_{k \in \{1,\dots,n-1\}} \left| \frac{\lambda_{k+1}}{\lambda_k} \right|$
- Note
  - For the two algorithms above, $\underline{Q}^{(k)}$ converges to X, and $A^{(k)}$ converges to Λ
  - In practice we often prefer QR iteration

Equivalence of Simultaneous Iteration and QR Iteration

- **QR iteration** is **equivalent** to **simultaneous iteration**, in the sense that both generates
  - The QR factorization: $A^{(k)} = \left[\underline{Q}^{(k)}\right]^T A\underline{Q}^{(k)}$
  - The projection: $A^k = \underline{Q}^{(k)} \underline{R}^{(k)}$, where $\underline{R}^{(k)} := R^{(k)} R^{(k-1)} \cdots R^{(1)}$
- Note: I added additional parentheses in the proof below for clarification
- Proof: QR iteration gives $A^{(k)} = \left[\underline{Q}^{(k)}\right]^T A\underline{Q}^{(k)}$
  - Using induction, assume $A^{(k-1)} = \left[\underline{Q}^{(k-1)}\right]^T A\underline{Q}^{(k-1)}$
  - $A^{(k)} = R^{(k)} Q^{(k)}$, by the algorithm of QR iteration

    $= \left(\left[Q^{(k)}\right]^T A^{(k-1)}\right) \cdot Q^{(k)}$, since $A^{(k-1)} = Q^{(k)} R^{(k)} \Rightarrow R^{(k)} = \left[Q^{(k)}\right]^T A^{(k-1)}$

    $= \left[Q^{(k)}\right]^T \cdot A^{(k-1)} \cdot Q^{(k)}$

    $= \left[Q^{(k)}\right]^T \cdot \left(\left[\underline{Q}^{(k-1)}\right]^T A\underline{Q}^{(k-1)}\right) \cdot Q^{(k)}$, by IH $A^{(k-1)} = \left[\underline{Q}^{(k-1)}\right]^T A\underline{Q}^{(k-1)}$

    $= \left(\left[Q^{(k)}\right]^T \left[\underline{Q}^{(k-1)}\right]^T\right) A \left(\underline{Q}^{(k-1)} Q^{(k)}\right)$

$= \left[\underline{Q}^{(k)}\right]^T A\underline{Q}^{(k)}$, by definition of $\underline{Q}$

- Proof: QR iteration gives $A^k = \underline{Q}^{(k)} \cdot \underline{R}^{(k)}$
  - Using induction, assume $A^{k-1} = \underline{Q}^{(k-1)} \cdot \underline{R}^{(k-1)}$
  - $A^k = A \cdot A^{k-1}$

    $= A \cdot \left(\underline{Q}^{(k-1)}\underline{R}^{(k-1)}\right)$, by inductive hypothesis $A^{k-1} = \underline{Q}^{(k-1)} \cdot \underline{R}^{(k-1)}$

    $= \left(A\underline{Q}^{(k-1)}\right) \cdot \underline{R}^{(k-1)}$

    $= \left(\underline{Q}^{(k-1)}A^{(k-1)}\right) \cdot \underline{R}^{(k-1)}$, since $A^{(k-1)} = \left[\underline{Q}^{(k-1)}\right]^T A\underline{Q}^{(k-1)} \Rightarrow A\underline{Q}^{(k-1)} = \underline{Q}^{(k-1)}A^{(k-1)}$

    $= \underline{Q}^{(k-1)} \cdot A^{(k-1)} \cdot \underline{R}^{(k-1)}$

    $= \underline{Q}^{(k-1)} \cdot \left(Q^{(k)}R^{(k)}\right) \cdot \underline{R}^{(k-1)}$, by the algorithm $A^{(k-1)} = Q^{(k)}R^{(k)}$

    $= \left(\underline{Q}^{(k-1)}Q^{(k)}\right) \cdot \left(R^{(k)}\underline{R}^{(k-1)}\right)$

    $= \underline{Q}^{(k)} \cdot \underline{R}^{(k)}$, by definition of $\underline{Q}^{(k)}$ and $\underline{R}^{(k)}$

Midterm Review

Monday, October 22, 2018 9:56 AM

Chapter Summary

- Ch1: $f(x)=0$
- Ch2: LU, QR, norm, conditioning
- Ch3: symmetric positive definite
- Ch4: $\vec{f}(\vec{x})=\vec{0}$
- Ch5: Eigenvalues: power/simultaneous/QR iteration

Zero-Finding Problem

- Iterative method
  - $f(x)=0 \xRightarrow{\text{look for } g} g(x)=x \xRightarrow{\text{simple iteration}} \begin{cases} \text{initial guess } x_0 \\ x_{k+1}=g(x_k) \end{cases}$
  - $\vec{f}(\vec{x})=\vec{0} \xRightarrow{\text{look for } \vec{g}} \vec{g}(\vec{x})=\vec{x} \xRightarrow{\text{simultaneous iteration}} \begin{cases} \text{initial guess } \vec{x}^{(0)} \\ \vec{x}^{(k+1)}=g\left(\vec{x}^{(k)}\right) \end{cases}$
- Contraction Mapping Theorem in $\mathbb{R}$
  - $\underbrace{|x_{k+1}-\xi|}_{\exists\xi \text{ by IVT}} = \underbrace{|g(x_k)-g(\xi)| \le L|x_k-\xi|}_{g \text{ contraction}} \le L^k|x_0-\xi| \to 0 \text{ as } k\to\infty$
- Contraction Mapping Theorem in $\mathbb{R}^n$
  - $\left\|\vec{x}^{(k+1)}-\vec{x}^{(k)}\right\|_\infty = \underbrace{\left\|\vec{g}\left(\vec{x}^{(k)}\right)-\vec{g}\left(\vec{x}^{(k-1)}\right)\right\|_\infty \le L\left\|\vec{x}^{(k)}-\vec{x}^{(k-1)}\right\|_\infty}_{g \text{ contraction}}$
  - $\left\{\vec{x}^{(k)}\right\}$ is a Cauchy sequence, so it converges to ξ
- Relaxation
  - If $g\in\mathcal{C}^1$ and $|g'(\xi)|<1$, then $\{x_k\}$ converges to ξ if x_0 is close to ξ
  - If $\vec{g}\in\mathcal{C}^1$ and $\left\|J_{\vec{g}}\left(\vec{\xi}\right)\right\|_\infty<1$, then $\left\{\vec{x}^{(k)}\right\}$ converges to $\vec{\xi}$ if $\vec{x}^{(0)}$ is close to $\vec{\xi}$
- Newton's method
  - $g(x)=x-\dfrac{f(x)}{f'(x)} \Rightarrow g'(\xi)=0 \Rightarrow g$ is contracting at ξ
  - $\vec{g}(\vec{x})=\vec{x}-\left[J_{\vec{f}}(\vec{x})\right]^{-1}\vec{f}(\vec{x}) \Rightarrow \left\|J_{\vec{g}}\left(\vec{\xi}\right)\right\|_\infty=0 \Rightarrow \vec{g}$ is contracting at $\vec{\xi}$
- Example
  - Given $\vec{f}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right)=\begin{bmatrix}x_1^2+x_2^2-2\\x_1-x_2\end{bmatrix}$
  - Prove $\vec{x}=\pm\begin{bmatrix}1\\1\end{bmatrix}$ is a zero
    - $\vec{f}\left(\begin{bmatrix}1\\1\end{bmatrix}\right)=\vec{f}\left(\begin{bmatrix}-1\\-1\end{bmatrix}\right)=\vec{0}$

- Find $\vec{g}(\vec{x})$ defined by Newton's method
  - $J_{\vec{f}}(\vec{x}) = \begin{bmatrix} 2x_1 & 2x_2 \\ 1 & -1 \end{bmatrix}$
  - $\left[J_{\vec{f}}(\vec{x})\right]^{-1} = \begin{bmatrix} \dfrac{1}{2(x_1+x_2)} & \dfrac{x_2}{x_1+x_2} \\ \dfrac{1}{2(x_1+x_2)} & -\dfrac{x_1}{x_1+x_2} \end{bmatrix}$
  - $\vec{g}(\vec{x}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} \dfrac{1}{2(x_1+x_2)} & \dfrac{x_2}{x_1+x_2} \\ \dfrac{1}{2(x_1+x_2)} & -\dfrac{x_1}{x_1+x_2} \end{bmatrix} \begin{bmatrix} x_1^2+x_2^2-2 \\ x_1-x_2 \end{bmatrix} = \begin{bmatrix} \dfrac{x_1^2+x_2^2}{2(x_1+x_2)} \\ \dfrac{x_1^2+x_2^2}{2(x_1+x_2)} \end{bmatrix}$
- If $x_1, x_2 \in \left(\frac{1}{2}, 1\right)$, show that $\left\|\vec{g}(\vec{x}) - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\|_\infty \le C \left\|\vec{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\|_\infty$ for some $C < 1$
  - $\left\|\vec{g}(\vec{x}) - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\|_\infty = \dfrac{(x_1-1)^2+(x_2-1)^2}{2|x_1+x_2|} \le \dfrac{1}{2}\left(|x_1-1|^2+|x_2-1|^2\right)$

    $\le \max\{|x_1-1|^2, |x_2-1|^2\} = \left\|\vec{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\|_\infty^2 \le \dfrac{1}{2}\left\|\vec{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\|_\infty$

Norm, Condition Number, and QR Factorization

- Definition of $\|A\|_p$
  - $\|A\|_p = \sup\limits_{\vec{x}\neq 0} \dfrac{\|A\cdot\vec{x}\|_p}{\|\vec{x}\|_p}$
- Explicit formula for $\|A\|_\infty$ and $\|A\|_1$
  - $\|A\|_1 = \max\limits_j \|\vec{a_j}\|_1 = \max\limits_j \sum\limits_{i=1}^{m} |a_{ij}|$
  - $\|A\|_\infty = \max\limits_i \left\|\vec{b_i}\right\|_1 = \max\limits_i \sum\limits_{j=1}^{n} |a_{ij}|$
  - Given $A = \begin{bmatrix} 2 & 1 \\ -3 & 1 \end{bmatrix}$, then $\|A\|_1 = 5$, and $\|A\|_\infty = 4$
- Computing $\|A\|_2$
  - Define $B = A^T A = \begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 13 & 5 \\ 5 & 2 \end{bmatrix}$
  - Then B is a s.p.d matrix, so $\lambda_i \in \mathbb{R}^+$ and $\vec{x_i} \perp \vec{x_j}$
  - $B\vec{x} = \lambda\vec{x} \Longrightarrow \det(B-\lambda I) = 0 \Longrightarrow \lambda = \dfrac{1}{2}\left(15 \pm \sqrt{221}\right)$
  - $\|A\|_2 = \max\limits_i \sqrt{\lambda_i} = \sqrt{\dfrac{1}{2}\left(15+\sqrt{221}\right)}$
- Show $\text{cond}_x(A) \le \kappa(A)$
  - Suppose $Ax = b \Leftrightarrow x = A^{-1}b$
  - $\text{cond}_x(A) = \dfrac{\|\delta b\|/\|b\|}{\|\delta x\|/\|x\|} = \dfrac{\|\delta b\|}{\|b\|}\cdot\dfrac{\|x\|}{\|\delta x\|} = \dfrac{\|A\cdot\delta x\|}{\|b\|}\cdot\dfrac{\|A^{-1}b\|}{\|\delta x\|} \le \|A\|\|A^{-1}\| = \kappa(A)$

- Find the QR factorization of A

$\vec{q_1} = \vec{a_1} = \begin{bmatrix}2\\3\end{bmatrix}$	$\vec{q_1} = \frac{\vec{q_1}}{\\|\vec{q_1}\\|_2} = \frac{1}{\sqrt{13}}\begin{bmatrix}2\\3\end{bmatrix}$
$\vec{q_2} = \vec{a_2} - \langle\vec{a_2}, \vec{q_1}\rangle\vec{q_1} = \begin{bmatrix}1\\1\end{bmatrix} - \frac{1}{13}\left\langle\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}2\\3\end{bmatrix}\right\rangle\begin{bmatrix}2\\3\end{bmatrix} = \frac{1}{13}\begin{bmatrix}3\\-2\end{bmatrix}$	$\vec{q_2} = \frac{\vec{q_2}}{\\|\vec{q_2}\\|_2} = \frac{1}{\sqrt{13}}\begin{bmatrix}3\\-2\end{bmatrix}$

  - $Q = [\vec{q_1}, \vec{q_2}] = \frac{1}{\sqrt{13}}\begin{bmatrix}2 & 3\\3 & -2\end{bmatrix}$, and $R = \begin{bmatrix}\langle\vec{a_1}, \vec{q_1}\rangle & \langle\vec{a_2}, \vec{q_1}\rangle\\0 & \langle\vec{a_2}, \vec{q_2}\rangle\end{bmatrix} = \frac{1}{\sqrt{13}}\begin{bmatrix}13 & 5\\0 & 1\end{bmatrix}$

Eigen-Decomposition

- Power iteration
  - Initialize $v^{(0)} \in \mathbb{R}^n$ *s.t.* $\left\|v^{(0)}\right\|_2 = 1$
  - For $k = 1,2, \ldots$
    - $w \leftarrow Av^{(k-1)}$
    - $v^{(k)} \leftarrow \frac{w}{\|w\|}$
    - $\lambda^{(k)} \leftarrow \left\langle v^{(k)}, Av^{(k)}\right\rangle$
  - $v^{(k)}$ converges the eigenvector with the largest eigenvalue
  - $\lambda^{(k)}$ converges to the largest eigenvalue
  - $\left\|v^{(k)} - (\pm\vec{x_1})\right\| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$ and $\left|\lambda^{(k)} - \lambda_1\right| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right)$
- Simultaneous iteration
  - Initialize $\underline{Q}^{(0)} \leftarrow I$
  - For $k = 1,2, \ldots$
    - $Z \leftarrow A\underline{Q}^{(k-1)}$
    - $Z \rightarrow \underline{Q}^{(k)}R^{(k)}$
    - $A^{(k)} \leftarrow \left[\underline{Q}^{(k)}\right]^T A\underline{Q}^{(k)}$
  - Then $\underline{Q}^{(k)}$ converges to X with rate $O\left(C^k\right)$, and $A^{(k)}$ converges to Λ with rate $O\left(C^{2k}\right)$
  - $\left\|q_i^{(k)} - (\pm\vec{x_i})\right\| = O\left(C^k\right)$ and $\left|A_{ii}^{(k)} - \lambda_i\right| = O\left(C^{2k}\right)$ where $C = \max_{k\in\{1,\ldots,n-1\}} \frac{|\lambda_{k+1}|}{|\lambda_k|}$
- QR iteration
  - Initialize $A^{(0)} \leftarrow A$
  - For $k = 1,2, \ldots$
    - $A^{(k-1)} \rightarrow Q^{(k)}R^{(k)}$
    - $A^{(k)} \leftarrow R^{(k)}Q^{(k)}$
  - Then $\underline{Q}^{(k)} \coloneqq Q^{(k)} \cdots Q^{(1)}$ converges to X and $A^{(k)}$ converges to Λ
- Simultaneous iteration and QR iteration are equivalent

Ch 6-10: Approximation & Integration

Friday, December 7, 2018 10:50 PM

Polynomial Approximation Theory

Monday, October 15, 2018 9:57 AM

Approximation Theory

- Goal
  - Given $f(x)$, we want to find a **numerical representation** $\boldsymbol{p(x)}$ s.t. $\boldsymbol{f(x) - p(x)}$ **is small**
  - In this case we can store the function f using **finite number of coefficients**
- How to find $p(x)$
  - Local methods
    - Spline interpolation (using piecewise polynomial)
      - Finite difference method
      - Finite element method
    - Padé approximation (using rational function)
  - Global methods
    - **Orthogonal polynomial**
    - Fourier approximation (using sin and cos)
- How to quantize the approximation
  - L_∞: $\sup_{x\in[a,b]} |f(x) - p(x)|$
  - L_2: $\sqrt{\int_a^b |f(x) - p(x)|^2 dx}$ (numerically easier to compute)

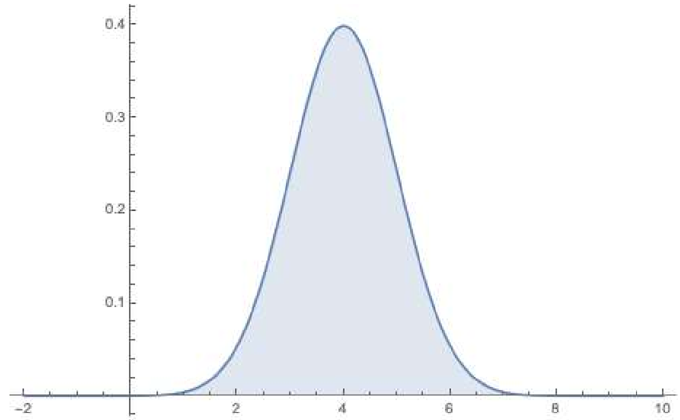
Main Questions about Polynomial Approximation

- Why can we use a polynomial $g(x)$ to approximate a complex function $f(x)$?
  - Weierstrass Approximation Theorem
  - Best approximation theory
- How to do the polynomial approximation
  - Projection
  - Interpolation
- How to analyze the approximation error

Weierstrass Approximation Theorem

- General idea
  - We can **use a polynomial** $\boldsymbol{p(x)}$ **to approximate** $f \in \mathcal{C}[a,b]$ with $|f| \le M$
- Theorem
  - Suppose $f \in \mathcal{C}[a,b]$ and $|f| \le M$
  - $\forall \varepsilon > 0, \exists p \in \mathbb{P}$ s.t. $\|f(x) - p(x)\|_\infty = \sup_{x\in[a,b]} |f(x) - p(x)| < \varepsilon$
- Review: Gaussian distribution

- $x \mapsto \int_{-\infty}^{\infty} \frac{1}{h\sqrt{\pi}} \exp\left(-\frac{(u-x)^2}{h^2}\right) du$



- The standard deviation $\sigma = \frac{h}{\sqrt{2}}$ is controlled by h
- The function **decays very fast when x is far from u**

- Define a new function $S_h f(x)$
    - $\boldsymbol{S_h f(x) \coloneqq \frac{1}{h\sqrt{\pi}} \int_{-\infty}^{\infty} f(u) \exp\left(-\frac{(u-x)^2}{h^2}\right) du}$
    - Note: Each point of $S_h f(x)$ is a local approximation of $f(x)$ with Gaussian weight
- $S_h f(x)$ is a good approximation
    - Since $\frac{1}{h\sqrt{\pi}} \int_{-\infty}^{\infty} \exp\left(-\frac{(u-x)^2}{h^2}\right) du = 1$, we can write f as
        - $f(x) = \frac{1}{h\sqrt{\pi}} \int_{-\infty}^{\infty} f(x) \exp\left(-\frac{(u-x)^2}{h^2}\right) du$
    - So $|S_h f(x) - f(x)| = \frac{1}{h\sqrt{\pi}} \int_{-\infty}^{\infty} |f(u) - f(x)| \exp\left(-\frac{(u-x)^2}{h^2}\right) du = A + B$, where
        - $$\begin{aligned} A &\coloneqq \frac{1}{h\sqrt{\pi}} \int_{|x-u|<\delta} \underbrace{|f(u) - f(x)|}_{\leq 2M} \exp\left(-\frac{(u-x)^2}{h^2}\right) du \\ &\leq \frac{1}{h\sqrt{\pi}} 2M \int_{|x-u|<\delta} \underbrace{\exp\left(-\frac{(u-x)^2}{h^2}\right)}_{<1} du \\ &< \frac{1}{h\sqrt{\pi}} 2M \underbrace{\int_{|x-u|<\delta} du}_{2\delta} \\ &= \frac{1}{h\sqrt{\pi}} 2M 2\delta = \frac{4M}{\sqrt{\pi}} \frac{\delta}{h} \end{aligned}$$
            - We can choose δ small enough such that $A \leq \frac{\varepsilon}{2}$
        - $B \coloneqq \frac{1}{h\sqrt{\pi}} \int_{|x-u|\geq\delta} |f(u) - f(x)| \exp\left(-\frac{(u-x)^2}{h^2}\right) du$
            - $\exp\left(-\frac{(u-x)^2}{h^2}\right)$ decays very fast when $\frac{(u-x)^2}{h^2} \geq \frac{\delta^2}{h^2} \gg 1$

- We can choose h small enough such that $B \leq \frac{\varepsilon}{2}$
- Therefore $|S_h f(x) - f(x)| \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$

- **Apply Taylor expansion** to $S_h f(x)$
  - Then $S_h f(x) = \frac{1}{h\sqrt{\pi}} \int_{-\infty}^{\infty} f(u) \underbrace{\sum_{n=0}^{N} (-1)^n \frac{(u-x)^{2n}}{n!\, h^{2n}}}_{\text{taylor expan. of exp term}} du + \varepsilon$
  - Define $p(x) \coloneqq \frac{1}{h\sqrt{\pi}} \int_a^b f(u) \sum_{n=0}^{N} (-1)^n \frac{(u-x)^{2n}}{n!\, h^{2n}} du$, then $|p(x) - f(x)| \leq 2\varepsilon$
  - *i.e.* The approximation error can be arbitrarily small

Best Approximation Theory

- General idea
  - There exists an N-th degree polynomial p^* that leads to an error curve $p^* - f$ oscillating back and forth between ε and $-\varepsilon$, a total of $N+2$ times, giving a worse-case error ε
- Theorem
  - Suppose $f \in \mathcal{C}[a, b]$ and $|f| \leq M$
  - $\forall \mathbb{P}_N \coloneqq \{p \in \mathbb{P} | \deg p \leq N\}, \exists!\, p^* \in \mathbb{P}_N$ s.t.
  - $\|f(x) - p^*(x)\|_\infty < \|f(x) - q(x)\|_\infty, \forall q \in \mathbb{P}_N$
- Property of p^*
  - $E(x) = f(x) - p^*(x)$ has $\boldsymbol{N+2}$ **extremas** $(x_1, \ldots, x_{N+2})$
  - $|\boldsymbol{E(x_i)}|$ **are equal** $\forall i \in \{1, \ldots, N+2\}$
  - $E(x)$ has $\boldsymbol{N+1}$ **roots** $(y_1, \ldots, y_{N+1}) \Leftrightarrow f(y_i) = p^*(y_i) \Leftrightarrow \boldsymbol{p^*(x)}$ **iterpolates** $\boldsymbol{f(x)}$ **at** $\boldsymbol{y_i}$
- Remarks
  - The existence is **not numerically tractable**
  - $p^*(x)$ interpolates $f(x)$ at $N+1$ points (unknowns)

Lagrange Interpolation & Chebyshev Nodes

Wednesday, October 31, 2018 9:59 AM

Polynomial Interpolation

- Assume we have information at N points of f: $f(x_1), \dots, f(x_N)$
- We look for a k-th order polynomial $p_k(x) = a_0 + a_1 x + \cdots + a_k x^k$ to interpolate $f(x)$
    - $\begin{cases} a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_k x_1^k = f(x_1) \\ \vdots \\ a_0 + a_1 x_N + a_2 x_N^2 + \cdots + a_k x_N^k = f(x_N) \end{cases}$
    - $\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^k \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} a_0 \\ \vdots \\ a_k \end{bmatrix}}_{\vec{a}} = \underbrace{\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix}}_{\vec{f}}$
- Relation between N and k
    - If $N = k + 1$, then the equation is uniquely solvable: $\vec{a} = X^{-1}\vec{f}$
    - If $N < k + 1$, then the equation has infinite solutions: $\min\|\vec{a}\|_1$ s.t. $X\vec{a} = \vec{f}$
    - If $N > k + 1$, then the equation has no exact solution: use least square fitting
- Property of **Vandermonde matrix $\boldsymbol{X}$**
    - $\text{cond}(X) \gg 1 \Leftrightarrow X$ is ill-conditioned, so **$\boldsymbol{X^{-1}}$ is inaccurate numerically**
    - *i.e.* If there exists a small error in $\vec{f}$, it is magnified in $\vec{a} = X^{-1}\vec{f}$

Lagrange Interpolation

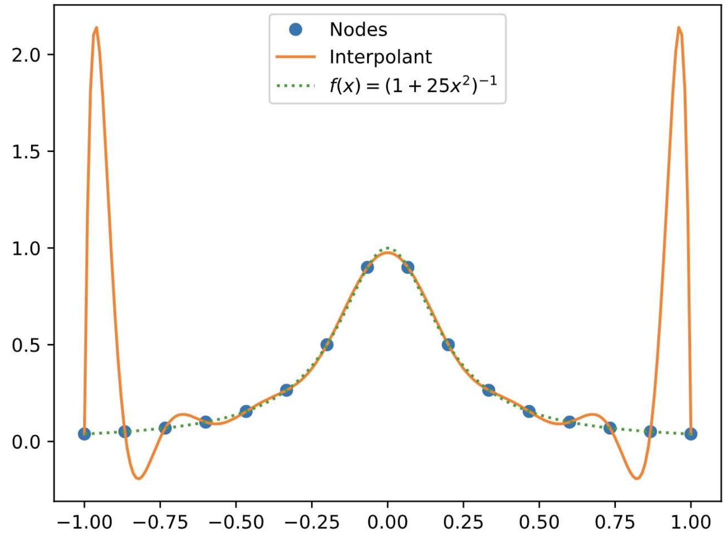
- Lagrange polynomial
    - Define $(N-1)$-th *order* polynomial $\boldsymbol{l_j(x)} \coloneqq \dfrac{\prod_{i \neq j}(\boldsymbol{x} - \boldsymbol{x_i})}{\prod_{i \neq j}(\boldsymbol{x_j} - \boldsymbol{x_i})}$. Then
        - $l_j(x_i) = \begin{cases} \dfrac{\prod_{i \neq j}(x_i - x_i)}{\prod_{i \neq j}(x_j - x_i)} = 0 & \text{for } i \neq j \\ \dfrac{\prod_{i \neq j}(x_j - x_i)}{\prod_{i \neq j}(x_j - x_i)} = 1 & \text{for } i = j \end{cases} \Rightarrow l_j(x_i) = \delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$
    - Define $\boldsymbol{p(x)} \coloneqq \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{f(x_i) l_i(x)}$. Then
        - $p(x_j) = \sum_{i=1}^{N} f(x_i) l_i(x_j) = \sum_{i=1}^{N} f(x_i)\delta_{ij} = f(x_j), \forall j \in \{1, \dots, N\}$
        - Therefore, p **interpolate $\boldsymbol{f}$ at $\boldsymbol{x_1, \dots, x_N}$**
- Error analysis for Lagrange interpolation
    - Define the error function $E(x) \coloneqq f(x) - p(x)$

- Define the axillary function $G_t(x) := E(x) - \frac{\prod_{i=1}^{N}(x - x_i)}{\prod_{i=1}^{N}(t - x_i)} E(t)$
- $G_t(x)$ has (at least) $N + 1$ zeros, since
  - $G_t(x_j) = E(x_j) - \frac{\prod_{i=1}^{N}(x_j - x_i)}{\prod_{i=1}^{N}(t - x_i)} E(t) = 0 - 0 \cdot E(t) = 0, \forall j \in \{1, \dots, N\}$
  - $G_t(t) = E(t) - \frac{\prod_{i=1}^{N}(t - x_i)}{\prod_{i=1}^{N}(t - x_i)} E(t) = E(t) - E(t) = 0$
- Taking N-th derivaties of G_t, we have $G_t^{(N)}(x) = E^{(N)}(x) - \frac{N!}{\prod_{i=1}^{N}(t - x_i)} E(t)$
- By Rolle's Theorem, $G_t^{(N)}$ has (at least) one zero
  - If $G_t^{(k)}(a) = G_t^{(k)}(b) = 0$, then $\exists x \in (a, b)$ s.t. $G_t^{(k+1)}(x) = 0$
  - *i.e.* The number of zeros decrease by 1 each time we take the derivative
- Choose $\xi \in \mathbb{R}\ s.t.\ G_t^{(N)}(\xi) = 0 \Leftrightarrow E^{(N)}(\xi) = \frac{N!}{\prod_{i=1}^{N}(t - x_i)} E(t)$
- Then $\frac{N!}{\prod_{i=1}^{N}(t - x_i)} E(t) = E^{(N)}(\xi) = f^{(N)}(\xi) - \underbrace{p^{(N)}(\xi)}_{\deg(p) < N} = f^{(N)}(\xi)$
- Therefore, $\boldsymbol{E(t) = f^{(N)}(\xi) \frac{\prod_{i=1}^{N}(t - x_i)}{N!}}$

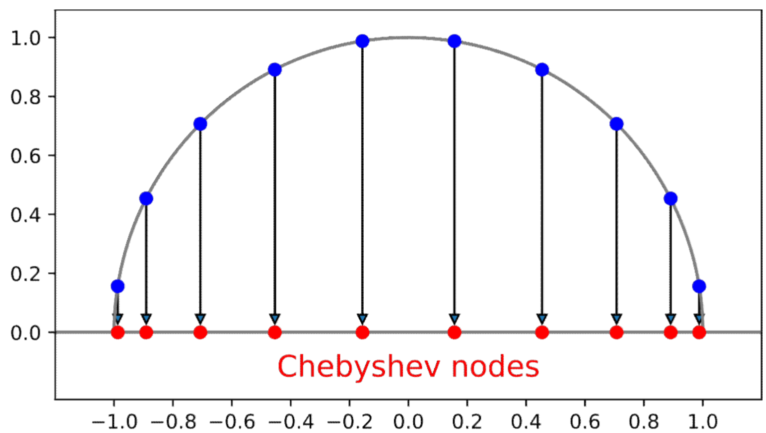
* Remark
  - **If $\boldsymbol{f(x) \in \mathbb{P}_{N-1}}$, then $\boldsymbol{f^{(N)}(\xi) = 0}$**
  - So, $E(t) = \underbrace{f^{(N)}(\xi)}_{0} \frac{\prod_{i=1}^{N}(t - x_i)}{N!} = 0, \forall t \in [a, b]$
  - *i.e.* $\boldsymbol{p(x) = f(x)}, \forall x \in [a, b]$

Runge's Phenomenon and Chebyshev Nodes

* Motivation
  - From the previous analysis, we know that $E(t) = \underbrace{\frac{f^N(\xi)}{N!}}_{\text{const}} \prod_{i=1}^{N}(t - x_i)$
  - In order to have a good approximation, we want $\boldsymbol{\min_{\{x_i\}} \sup_{t \in [a,b]} \left| \prod_{i=1}^{N}(t - x_i) \right|}$
  - But if we sample $\{x_i\}$ evenly in $[a, b]$, then $\sup_{t \in [a,b]} \left| \prod_{i=1}^{N}(t - x_i) \right|$ could be large
* Runge Phenomenon
  - **Equispaced** interpolation with **high degree polynomial** may result in **oscillation at the edges of interval**



- Chebyshev grids
    - For interval $[-1,1]$, we can pick the Chebyshev grids $\left\{ \boldsymbol{x_i = \cos\theta_i} \,\middle|\, \boldsymbol{\theta_i = \frac{i}{N}\pi} \right\}$
    - So the distribution of x_i is **concentrated at the ending points** of the interval



Polynomial Projection & Quadrature Method

Wednesday, October 31, 2018 9:59 AM

Polynomial Projection

- Goal
  - Let $f \in \mathcal{C}^\infty[-1,1]$ be fixed, we look for $p \in \mathbb{P}_N$ s.t. **p is "closest" to f**
- Relation with interpolation
  - Recall the equation we want to solve in polynomial interpolation
  - $$\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^k \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} a_0 \\ \vdots \\ a_k \end{bmatrix}}_{\vec{a}} = \underbrace{\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix}}_{\vec{f}}$$
  - Each column of X is determined by the sample points $x_1, x_2, \ldots, x_N$
  - Each row of X is defined by the monomials $1, x_i, x_i^2, \ldots, x_i^k$
  - In projection, we **replace the monomial polynomials** by **orthogonal polynomials**
- Analogy of projection in $\mathbb{R}^3$
  - Let $\vec{v} = v_1\vec{i} + v_2\vec{j} + v_3\vec{k}$, then $\boldsymbol{\vec{p} = v_1\vec{i} + v_2\vec{j}}$ is the **closest point to $\vec{v}$ in the x-y plane**
  - Let $\vec{q} \in x$-y plane be arbitrary, then
  - $\|\vec{v} - \vec{q}\|_2^2 = \langle \vec{v} - \vec{q}, \vec{v} - \vec{q} \rangle = \langle (\vec{v} - \vec{p}) + (\vec{p} - \vec{q}), (\vec{v} - \vec{p}) + (\vec{p} - \vec{q}) \rangle$
    $$= \underbrace{\langle \vec{v} - \vec{p}, \vec{v} - \vec{p} \rangle}_{\|\vec{v} - \vec{p}\|_2^2} + 2 \underbrace{\langle \vec{v} - \vec{p}, \vec{p} - \vec{q} \rangle}_{0} + \underbrace{\langle \vec{p} - \vec{q}, \vec{p} - \vec{q} \rangle}_{\geq 0} \geq \|\vec{v} - \vec{p}\|_2^2$$
  - Therefore $\|\vec{v} - \vec{p}\|_2 \leq \|\vec{v} - \vec{q}\|, \forall \vec{q} \in x$-$y$ plane
- Polynomial projection
  1. $\mathbb{P}_N$ is a subspace of $\mathcal{C}^\infty$
     - Let $p, q \in \mathbb{P}_N$, then $\alpha \cdot p(x) + \beta \cdot q(x) \in \mathbb{P}_N$
  2. Build a list of orthogonal polynomials $\phi_0, \phi_1, \ldots$
     - See definition below
  3. $\forall f \in \mathcal{C}^\infty[a, b], f(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots$ for some constant $c_0, c_1, \ldots$
     - This is guaranteed by Weierstrass Approximation Theorem
  4. Then the **best approximation of f in** $\mathbb{P}_N$ is $\boldsymbol{p(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots c_N\phi_N(x)}$
     - Let $q \in \mathbb{P}_N$ be arbitrary, then
     - $\|f - q\|_2 = \langle f - q, f - q \rangle = \langle (f - p) + (p - q), (f - p) + (p - q) \rangle$
       $$= \langle f - p, f - p \rangle + 2 \underbrace{\langle f - p, p - q \rangle}_{0} + \underbrace{\langle p - q, p - q \rangle}_{\geq 0}$$
       $$\geq \langle f - p, f - p \rangle = \|f - p\|_2$$
     - This proves the optimality of $p(x)$

- Note: $\|g\|_2 = \sqrt{\langle g, g \rangle} = \sqrt{\int_a^b g^2(x) w(x) dx}$

Orthogonal Polynomials

- Definition
  - Given an interval $[a, b]$ and a weight function $w(x)$ (used in function dot product)
  - **Orthogonal polynomials** sequence is a list of polynomials $\{\phi_0, \phi_1, \dots, \phi_N, \dots\}$ *s.t.*
    - $\mathbf{deg}\, \boldsymbol{\phi_i = i}$
    - $\boldsymbol{\langle \phi_i, \phi_j \rangle_w = \int_a^b \phi_i(x) \phi_j(x) w(x) dx \begin{cases} = 0 & \textbf{if } i \neq j \\ \neq 0 & \textbf{if } i = j \end{cases}}$
  - Moreover, if $\langle \boldsymbol{\phi_i, \phi_j} \rangle_{\boldsymbol{w}} = \boldsymbol{\delta_{ij}}$, then $\{\phi_0, \phi_1, \dots, \phi_N, \dots\}$ is said to be **orthonormal**
- Recurrence relation of **orthonormal polynomials**
  - $\boldsymbol{\phi_{m+1} = (\alpha_m x + \beta_m) \phi_m + \gamma_m \phi_{m-1}}$ where $\alpha_m, \beta_m, \gamma_m \in \mathbb{R}$
  - On the LHS, ϕ_{m+1} has $(m + 2)$ degrees of freedom, so we need $(m + 2)$ contraints
    - $\langle \phi_{m+1}, \phi_{m+1} \rangle = 1$
    - $\langle \phi_{m+1}, \phi_i \rangle = 0, \forall i \in \{0, \dots, n - 1\}$
  - However on the RHS, $(\alpha_m x + \beta_m)\phi_m + \gamma_m \phi_{m-1}$ has only 3 degrees of freedom
  - Only the first 3 constraints will be used, and the rest will be automatically satisfied
    - For $i \leq m - 2$, $\langle \phi_{m+1}, \phi_i \rangle = \alpha_m \underbrace{\langle x\phi_m, \phi_i \rangle}_{0} + \beta_m \underbrace{\langle \phi_m, \phi_i \rangle}_{0} + \gamma_m \underbrace{\langle \phi_{m-1}, \phi_i \rangle}_{0} = 0$
    - Note: $\langle x\phi_m, \phi_i \rangle = \langle \phi_m, x\phi_i \rangle$ where $x\phi_i \in \text{span}\{\phi_0, \dots, \phi_{m-1}\} \perp \phi_m$
  - In order to determine $\alpha_m, \beta_m, \gamma_m$, we only need to solve $\boldsymbol{\begin{cases} \langle \phi_{m+1}, \phi_{m+1} \rangle = 1 \\ \langle \phi_{m+1}, \phi_m \rangle = 0 \\ \langle \phi_{m+1}, \phi_{m-1} \rangle = 0 \end{cases}}$
- Examples of orthogonal polynomials

Name	Domain	Weight Function	Recurrence Relation
Legendre	$[-1,1]$	$w(x) = \frac{1}{2}$	$\phi_{n+1} = \frac{2n+1}{n+1} x\phi_n - \frac{n}{n+1}\phi_{n-1}$
Chebyshev	$[-1,1]$	$w(x) = \frac{1}{\sqrt{1-x^2}}$	$T_{n+1} = 2xT_n - T_{n-1}$
Hermite	$(-\infty, \infty)$	$w(x) = e^{-x^2}$	$H_{n+1} = xH_n - nH_{n-1}$

Gauss Quadratures

- Definition
  - The **roots of $\boldsymbol{\phi_m}$** are called **Gauss quadratures** for ϕ_m
- **$\boldsymbol{\phi_m}$ has $\boldsymbol{m}$ Gauss quadratures**
  - ϕ_0 is a constant not equal to 0, so it has no root
  - ϕ_1 has a zero in $[a, b]$
    - Assume ϕ_1 has no zero in $[a, b]$. WLOG, assume $\phi_1(x) > 0, \forall x \in [a, b]$. Then

        - $\langle \phi_1, \phi_0 \rangle = \int_a^b \underbrace{\phi_1(x)}_{>0} \underbrace{\phi_0(x)}_{>0} \underbrace{w(x)}_{>0} dx > 0$, which contradicts $\langle \phi_1, \phi_0 \rangle = \delta_{0,1}$
    - ϕ_2 has two roots in $[a, b]$
        - Assume ϕ_2 has only one root ξ, then $(x - \xi)\phi_2(x)$ is either all > 0 or < 0
        - WLOG, assume $(x - \xi)\phi_2(x) > 0, \forall x \in [a, b]$. Then
        - $\langle (x - \xi)\phi_2, \phi_0 \rangle = \int_a^b \underbrace{\phi_2(x)(x - \xi)}_{>0} \underbrace{\phi_0(x)}_{>0} \underbrace{w(x)}_{>0} dx > 0$
        - But $\langle (x - \xi)\phi_2, \phi_0 \rangle = \langle \phi_2, (x - \xi)\phi_0 \rangle = 0$, since $(x - \xi)\phi_0 \in \text{span}\{\phi_0, \phi_1\} \perp \phi_2$
        - Therefore ϕ_2 has at least two roots
- Computing Gauss quadratures using recurrence relation
    - Given the recurrence relation
        - $\phi_{n+1} = (\alpha_n x + \beta_n)\phi_n + \gamma_n \phi_{n-1}$
    - Define $a_n = -\frac{\gamma_n}{\alpha_n}, b_n = -\frac{\beta_n}{\alpha_n}, c_n = \frac{1}{\alpha_n}$, then we can rewrite the recurrence as
        - $x\phi_n = a_n \phi_{n-1} + b_n \phi_n + c_n \phi_{n+1}$
    - Written in matrix form,
        - $x \underbrace{\begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_{n-1}(x) \\ \phi_n(x) \end{bmatrix}}_{\vec{\phi}} = \underbrace{\begin{bmatrix} b_0 & c_0 & & & \\ a_1 & b_1 & c_1 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_{n-1}(x) \\ \phi_n(x) \end{bmatrix}}_{\vec{\phi}} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ c_n \phi_{n+1}(x) \end{bmatrix}$
    - For $\phi_{n+1}(x) = 0$, we have $x\vec{\phi} = A\vec{\phi}$
    - Therefore, **Gauss quadratures are the eigenvalues of A**

Chebyshev Polynomials

- **Chebyshev polynomials** can be defiend using either one of the equations below
    - $\boldsymbol{T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), T_0 = 1, T_1 = x}$
    - $\boldsymbol{T_n(x) = \cos[n \cdot \arccos(x)]}$
- Prove the equivalence
    - Define $C_n(x) = \cos[n \cdot \arccos(x)]$
    - Base case
        - $C_0 = \cos(0) = 1 = T_0$
        - $C_1 = \cos(\arccos(x)) = x = T_1$
    - Inductive step
        - Assume $C_{n-1} = T_{n-1}$ and $C_n = T_n$, then we want to show that $C_{n+1} = T_{n+1}$
        - $C_{n+1}(x) = \cos[(n + 1)\arccos(x)]$

          $= \cos[(n + 1)\theta]$, where $\theta \coloneqq \arccos(x) \Leftrightarrow x = \cos\theta$

          $= \cos\theta \cos(n\theta) - \sin(n\theta) \sin\theta$

$$= 2\cos\theta\cos(n\theta) - (\cos\theta\cos(n\theta) + \sin(n\theta)\sin\theta)$$
$$= 2\cos\theta\cos(n\theta) - \cos[(n-1)\theta]$$
$$= 2x\cos(n\theta) - \cos[(n-1)\theta]$$
$$= 2x\cos(n\arccos x) - \cos[(n-1)\arccos(x)]$$
$$= 2xC_n(x) - C_{n-1}(x)$$
$$= 2xT_n(x) - T_{n-1}(x)$$
$$= T_{n+1}(x)$$

- By induction, $C_n = T_n, \forall n \in \mathbb{N}$
- Thus two definitions are equivalent

- Find the recurrence matrix and zeros
  - $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \Leftrightarrow xT_n(x) = \frac{1}{2}\big(T_{n+1}(x) + T_{n-1}(x)\big)$
  - Written in matrix form, we have
  - $$x\underbrace{\begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{n-1} \\ T_n \end{bmatrix}}_{\vec{T}} = \underbrace{\begin{bmatrix} 0 & 1 & & & \\ 1/2 & 0 & 1/2 & & \\ & \ddots & \ddots & \ddots & \\ & & 1/2 & 0 & 1/2 \\ & & & 1/2 & 0 \end{bmatrix}}_{A}\underbrace{\begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{n-1} \\ T_n \end{bmatrix}}_{\vec{T}} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1/2\, T_{n+1} \end{bmatrix}$$
  - $T_{n+1} = 0 \Leftrightarrow x\vec{T} = A\vec{T} \Leftrightarrow$ zeros of $T_{n+1}(x) = \mathrm{eig}(A)$

Using Numerical Integration to Compute the Projection Coefficients

- Motivation
  - In $\mathbb{R}^n$, given $\vec{v} \in \mathbb{R}^n$, $\vec{v} = \sum_{i=1}^{n} v_i\vec{e_i} = \sum_{i=1}^{n} \langle \vec{v}, \vec{e_i}\rangle \vec{e_i}$
  - Similarly, for $f \in C^\infty[a,b]$, $f = \sum_{i=0}^{+\infty} c_i\phi_i = \sum_{i=0}^{+\infty} \langle f, \phi_i\rangle \phi_i = \sum_{i=0}^{+\infty}\left[\int_a^b f(x)\phi_i(x)w(x)dx\right]\phi_i$
  - But the integration $c_i = \int_a^b f(x)\phi_i(x)w(x)dx$ is sometimes hard to perform
  - We can **compute** $\boldsymbol{\alpha_i} = \sum_{i=0}^{J} \boldsymbol{f(x_k)\phi_i(x_k)w(x_k)}$ for samples $\{x_0, \dots, x_J\}$ to **approximate** $\boldsymbol{c_i}$
- Theorem
  - If $\boldsymbol{f} \in \mathbb{P}_{\boldsymbol{2N+1}}$, then $\int_a^b \boldsymbol{f(x)w(x)dx} = \sum_{m=0}^{N} \boldsymbol{f(x_m)w_m}$, where
    - $\{\boldsymbol{x_0}, \dots, \boldsymbol{x_N}\}$ are the **Gauss quadratures** of ϕ_{N+1}, determined by $w(x)$ and $[a,b]$
    - $w_m = \int_a^b l_m(x)w(x)dx$
- Proof
  - Case 1: When $f \in \mathbb{P}_N$

- f can be written as $f(x) = \sum_{m=0}^{N} f(x_m) l_m(x)$
- Then $\int_a^b f(x)w(x)dx = \int_a^b \left[\sum_{m=0}^{N} f(x_m) l_m(x)\right] w(x)dx$

$$= \sum_{m=0}^{N} f(x_m) \left[\int_a^b l_m(x)w(x)dx\right] = \sum_{m=0}^{N} f(x_m)\, w_m$$

- Case 2: When $f \in \mathbb{P}_{2N+1} \setminus \mathbb{P}_N$
  - Define $p(x) \coloneqq \sum_{m=0}^{N} f(x_m) l_m(x)$, then $p \in \mathbb{P}_N$
  - Define $r(x) \coloneqq f(x) - p(x) \in \mathbb{P}_{2N+1}$, then $r(x_m) = 0, \forall m \in \{0, \dots, N\}$
  - Therefore, we can write r as $r(x) = q(x) \prod_{m=0}^{N} (x - x_m)$, for some $q \in \mathbb{P}_N$
  - $\int_a^b \big(f(x) - p(x)\big) w(x)dx = \int_a^b r(x)w(x)dx$

$$= \int_a^b \left[\prod_{m=0}^{N} (x - x_m)\right] q(x)w(x)dx = \left\langle \prod_{m=0}^{N} (x - x_m), q \right\rangle = 0$$

since $\prod_{m=0}^{N} (x - x_m) = c\phi_{N+1}$ for some $c \in \mathbb{R}$, and $q \in \mathbb{P}_N \perp \phi_{N+1}$

  - Thus, $\int_a^b f(x)w(x)dx = \int_a^b p(x)w(x)dx = \int_a^b \left[\sum_{m=0}^{N} f(x_m) l_m(x)\right] w(x)dx$

$$= \sum_{m=0}^{N} f(x_m) \left[\int_a^b l_m(x)w(x)dx\right] = \sum_{m=0}^{N} f(x_m)\, w_m$$

- Corollary
  - For $\boldsymbol{f} \in \mathbb{P}_{\boldsymbol{N+1}}$, the projection **coefficients $\boldsymbol{c_i}$** is **equal** to the numerical **apprixmation $\boldsymbol{\alpha_i}$**
  - Since $c_i = \langle f, \phi_i \rangle = \int_a^b \underbrace{f(x)\phi_i(x)}_{\in \mathbb{P}_{2N+1}} w(x)dx = \sum_{k=0}^{N} f(x_k)\phi_i(x) w_k = \alpha_i, \forall i \in \{0, \dots, n\}$

Summary and Error Analysis for Polynomial Projection

- Let $\{\phi_0, \phi_1, \phi_3, \dots\}$ be a list of orthogonal polynomials
- Given $f \in C^\infty[a, b]$, it can be written as $f(x) = \sum_{n=0}^{+\infty} c_n \phi_n(x)$, for some constants $c_n \in \mathbb{R}$
- We first **project $\boldsymbol{f}$ to** $\mathbb{P}_N$ by **truncating** the summation to N
  - $f(x) \approx p(x) = \sum_{n=0}^{N} c_n \phi_n(x)$ with error $= \sum_{n=N+1}^{+\infty} c_n \phi_n(x)$

- By regularization theory, if $f \in C^{\nu}[a,b]$, then $c_n = O(n^{-\nu})$
- Therefore for $\{c_{N+1}, c_{N+2}, \dots\}, c_i = O(n^{-\nu}) \lesssim c_N = O(N^{-\nu})$ is small

- Since c_n is hard to obtain directly, we use α_n to **approximate** them by numerical integration
  - $p(x) \approx \tilde{p}(x) = \sum_{n=0}^{N} \alpha_n \phi_n(x)$ with error$\sim(\alpha_n - c_n)$ for $f \notin \mathbb{P}_{N+1}$
  - $\alpha_n = \sum_{k=0}^{N} f(x_k)\phi_n(x_k)w_k$, by numerical integration

$$= \sum_{k=0}^{N}\left[\left(\sum_{m=0}^{+\infty} c_m\phi_m(x_k)\right)\phi_n(x_k)w_k\right], \text{ by substituting } f$$

$$= \sum_{m=0}^{+\infty}\left[c_m \sum_{k=0}^{N} \phi_n(x_k)\phi_m(x_k)w_k\right]$$

$$= \sum_{m=0}^{N}\left[c_m \sum_{k=0}^{N} \underbrace{\phi_n(x_k)\phi_m(x_k)}_{\in\mathbb{P}_{2N}\subseteq\mathbb{P}_{2N+1}} w_k\right] + \sum_{m=N+1}^{+\infty}\left[c_m \sum_{k=0}^{N} \phi_n(x_k)\phi_m(x_k)w_k\right]$$

$$= \sum_{m=0}^{N}\left[c_m \int_a^b \phi_n(x)\phi_m(x)w(x)dx\right] + \sum_{m=N+1}^{+\infty}\left[c_m \sum_{k=0}^{N} \phi_n(x_k)\phi_m(x_k)w_k\right]$$

$$= \sum_{m=0}^{N} c_m\delta_{mn} + \sum_{m=N+1}^{+\infty}\left[c_m \sum_{k=0}^{N} \phi_n(x_k)\phi_m(x_k)w_k\right]$$

$$= c_n + \underbrace{\sum_{m=N+1}^{+\infty} k_m c_m}_{O(N^{-\nu})\ll 1}, \text{ where } k_m = \sum_{k=0}^{N} \phi_n(x_k)\phi_m(x_k)w_k \text{ is a constant}$$

  - Therefore $\alpha_n - c_n = O(N^{-\nu})$ is small

Relation with Polynomial Interpolation

- Recall polynomial interpolation
  - Given f, we want to find $p(x) = \sum_{n=0}^{N} a_n x^n$ such that $p(x_k) = f(x_k), \forall k \in \{0, \dots, N\}$
  - Then we want to solve $\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^N \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^N \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix}}_{\vec{a}} = \underbrace{\begin{bmatrix} f(x_0) \\ \vdots \\ f(x_N) \end{bmatrix}}_{\vec{f}}$ for $\vec{a}$
  - Since the X is ill-conditioned, computing $\vec{a} = X^{-1}\vec{f}$ will result in large numeric error
- How to design the matrix X so its condition number is minimized
  - We can replace the monomials by orthonormal polynomials $\{\phi_0, \dots, \phi_N\}$
  - And choose the Gauss quadratures of ϕ_{N+1} to be the sample points $\{x_0, \dots, x_N\}$

- Then we want to solve $\underbrace{\begin{bmatrix} \phi_0(x_0) & \cdots & \phi_N(x_0) \\ \vdots & \ddots & \vdots \\ \phi_0(x_N) & \cdots & \phi_N(x_N) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix}}_{\vec{a}} = \underbrace{\begin{bmatrix} f(x_0) \\ \vdots \\ f(x_N) \end{bmatrix}}_{\vec{f}}$ for $\vec{a}$
- In this case, A is almost unitary and $\text{cond}(A) \approx 1$

- Proof
  - Let $W = \text{diag}(w_0, \dots, w_N)$, where $w_k = \int_a^b l_k(x) w(x) dx$. Then $A^T W A = I$, since
  - $[A^T W A]_{mn} = \sum_{k=0}^{N} \underbrace{\phi_m(x_k)\phi_n(x_k)}_{\in \mathbb{P}_{m+n} \subseteq \mathbb{P}_{2N+1}} w_k = \int_a^b \phi_m(x)\phi_n(x)w(x)dx = \langle \phi_m, \phi \rangle = \delta_{mn}$
  - Therefore A is almost unitary and $\text{cond}(A) \approx \frac{\max\{w_i\}}{\min\{w_i\}} \approx 1$

Integration Rules & Undetermined Coefficients

Wednesday, November 7, 2018 4:12 AM

Composite Integration Rules

- Introduction
  - We **divide** $[\boldsymbol{a},\boldsymbol{b}]$ **into** $\boldsymbol{N}$ **intervals** $\{\boldsymbol{x_0},\dots,\boldsymbol{x_N}\}$, where $x_k = a + k\Delta x$ and $\Delta x = \dfrac{b-a}{N}$
  - Then $\int_a^b f(x) = \sum_{k=0}^{N-1}\int_{x_k}^{x_{k+1}} f(x)dx$, and we can use polynomials to approximate $\int_{x_k}^{x_{k+1}} f(x)dx$
- Methods

Rule	Type of Function	Polynomial used in $[x_k, x_{k+1}]$	Error
Midpoint	piecewise constant	$p(x) = f\left(\dfrac{x_k + x_{k+1}}{2}\right)$	$O(\Delta x^2)$
Trapezoidal	piecewise linear	$p(x) = f(x_k) + \dfrac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}(x - x_k)$	$O(\Delta x^2)$
Simpson's	piecewise quadratic		$O(\Delta x^4)$

  - In general, for piecewise **polynomial with order $2i+1$ or $2i$**, the **error term is $O(\Delta x^{2i+2})$**
- Error analysis
  - For $f \in C^{\nu}[a,b]$, $\int_a^b f(x)\,dx - \sum_{i=0}^{N} f(x_i)w_i = O(N^{-\nu})$, so the error decrease as N increase

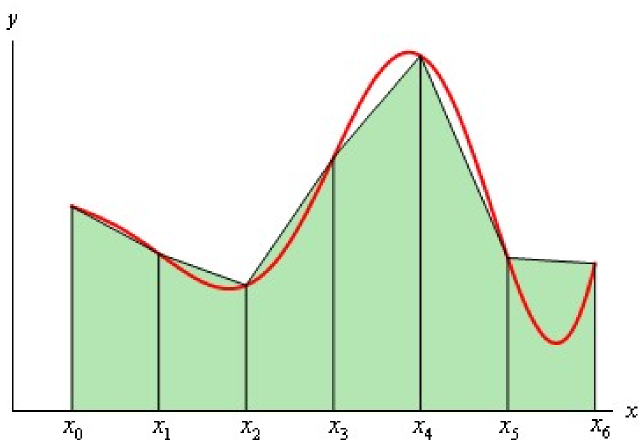
Trapezoidal Rule

- Procedure
  - For interval $[x_k, x_{k+1}]$, we look for a **linear polynomial p** that interpolates f at x_k and x_{k+1}
    - $\begin{cases} p(x_k) = f(x_k) \\ p(x_{k+1}) = f(x_{k+1}) \end{cases} \Rightarrow p(x) = f(x_k) + \dfrac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}(x - x_k)$
  - Integrate p in the interval $[x_k, x_{k+1}]$, then
    - $\int_{x_k}^{x_{k+1}} p(x)dx = f(x_k)\Delta x + \dfrac{f(x_{k+1}) - f(x_k)}{\underbrace{x_{k+1} - x_k}_{\Delta x}} \underbrace{\int_{x_k}^{x_{k+1}} (x - x_k)dx}_{\Delta x^2/2} = \dfrac{\Delta x}{2}\big(f(x_k) + f(x_{k+1})\big)$
  - Summing up all intervals, we have
    - $\int_a^b p(x)dx = \sum_{k=0}^{N-1}\int_{x_k}^{x_{k+1}} p(x)dx = \sum_{k=0}^{N-1}\dfrac{\Delta x}{2}\big(f(x_k) + f(x_{k+1})\big) = \Delta x\left[\dfrac{f(a)}{2} + \sum_{k=1}^{N-1} f(x_k) + \dfrac{f(b)}{2}\right]$
  - Note that $f(a)$ and $f(b)$ have weight $\dfrac{1}{2}$, and $f(x_1),\dots,f(x_{N-1})$ have weight 1
- Error analysis
  - Recall the error analysis in interpolation

- If $p \in \mathbb{P}_N$ interpolates f at $\{x_0, \dots, x_N\}$, then $f(x) - p(x) = \dfrac{f^{(N+1)}(\xi)}{(N+1)!}\prod_{i=0}^{N}(x - x_i)$

- In the interval $[x_k, x_{k+1}]$:
  - Since $p \in \mathbb{P}_1$ interpolates f at $\{x_k, x_{k+1}\}$, $f(x) - p(x) = \dfrac{1}{2}f''(\xi_x)(x - x_k)(x - x_{k+1})$
  - After integration, $\int_{x_k}^{x_{k+1}} f(x)dx - \int_{x_k}^{x_{k+1}} p(x)dx = \dfrac{1}{2}\int_{x_k}^{x_{k+1}} f''(\xi_x)(x - x_k)(x - x_{k+1})dx$
  - Recall the Mean Value Theorem for Integrals
    - If $f \in C^\infty[a, b]$, and g is an integrable function that **does not change sign** on $[a, b]$
    - Then $\int_a^b f(x)g(x)dx = f(\eta)\int_a^b g(x)dx$ for some $\eta \in [a, b]$
  - Define $E_k \coloneqq \int_{x_k}^{x_{k+1}} f(x)dx - \int_{x_k}^{x_{k+1}} p(x)dx = \underbrace{\dfrac{f''(\eta)}{2}}_{\text{const.}}\underbrace{\int_{x_k}^{x_{k+1}}(x - x_k)(x - x_{k+1})dx}_{O(\Delta x^3)} = O(\Delta x^3)$
- Therefore, the error over the entire interval is $\int_a^b f(x)dx - \int_a^b p(x)dx = \sum_{k=0}^{N-1} E_k = O((b - a)\Delta x^2)$
- When N increases, Δx decreases, so does the error

- Summary
  - We divide $[a, b]$ into $\{x_0, \dots, x_N\}$, and approximate f by a linear function p in each $[x_k, x_{k+1}]$
  - In each interval $[x_k, x_{k+1}]$, $\boldsymbol{\int_{x_k}^{x_{k+1}} f(x)dx \overset{O(\Delta x^3)}{\approx} \int_{x_k}^{x_{k+1}} p(x)dx = \dfrac{\Delta x}{2}\left(f(x_k) + f(x_{k+1})\right)}$
  - For the entire interval, $\boldsymbol{\int_a^b f(x)dx \overset{O(\Delta x^2)}{\approx} \int_a^b p(x)dx = \dfrac{\Delta x}{2}\left[f(a) + 2\sum_{k=1}^{N-1} f(x_k) + f(b)\right]}$



Midpoint Rule

- In $[x_k, x_{k+1}]$, we use the **constant function** $\boldsymbol{p(x) = f(x_{k+1/2}) \coloneqq f\left(\dfrac{x_k + x_{k+1}}{2}\right)}$ to approximate f
  - $\int_{x_k}^{x_{k+1}} f(x)dx \approx \int_{x_k}^{x_{k+1}} p(x)dx = \boldsymbol{f(x_{k+1/2})\Delta x}$
- Then the approximation for the entire interval $[a, b]$ is

- $\int_a^b f(x)dx = \sum_{k=0}^{N-1} \int_{x_k}^{x_{k+1}} f(x)dx \approx \sum_{k=0}^{N-1} f(x_{k+1/2})\Delta x = \boldsymbol{\Delta x\left(f(x_{1/2}) + f(x_{3/2}) + \cdots + f(x_{N-1/2})\right)}$

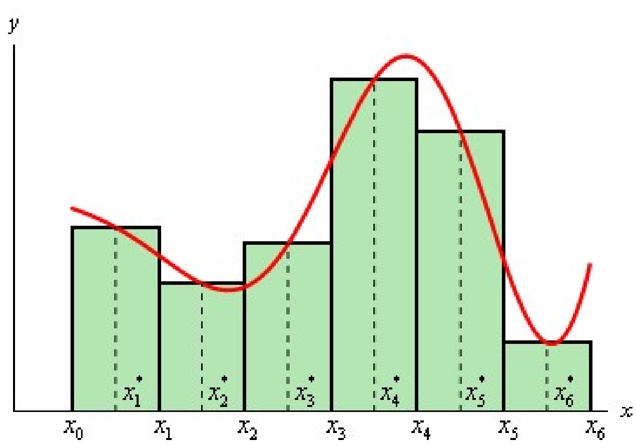
- (Invalid) Error analysis using Mean Value Theorem
  - Recall the error formula in interpolation: $f(x) - p(x) = \frac{f^{(N+1)}(\xi)}{N!}\prod_{i=0}^{N}(x - x_i)$
  - In the case of midpoint rule, we have $N = 1$, so $f(x) - p(x) = f'(\xi)(x - x_{k+1/2})$
  - After integration, $\int_{x_k}^{x_{k+1}} f(x)dx - \int_{x_k}^{x_{k+1}} p(x)dx = \int_{x_k}^{x_{k+1}} f'(\xi)(x - x_{k+1/2})dx$
  - By the Mean Value Theorem, $\int_{x_k}^{x_{k+1}} f'(\xi)(x - x_{k+1/2})dx \overset{?}{=} f'(\eta)\underbrace{\int_{x_k}^{x_{k+1}}(x - x_{k+1/2})dx}_{0} = 0$
  - But the equal sign does not hold, since MVT requires $g(x) > 0$ or $g(x) < 0, \forall x \in [a, b]$
- Error analysis using Taylor expansion
  - By Taylor expansion, $f(x) = f(x_{k+1/2}) + f'(x_{k+1/2})(x - x_{k+1/2}) + \frac{1}{2}f''(x_{k+1/2})(x - x_{k+1/2})^2 \cdots$
  - $\int_{x_k}^{x_{k+1}} f(x)dx - \int_{x_k}^{x_{k+1}} p(x)dx = \int_{x_k}^{x_{k+1}} [f(x) - p(x)]dx$

$$= \int_{x_k}^{x_{k+1}} \left[\underbrace{f(x_{k+1/2})}_{p(x)} + f'(x_{k+1/2})(x - x_{k+1/2}) + \frac{1}{2}f''(x_{k+1/2})(x - x_{k+1/2})^2 + \cdots - p(x)\right]dx$$

$$= \int_{x_k}^{x_{k+1}} \left[f'(x_{k+1/2})(x - x_{k+1/2}) + \frac{1}{2}f''(x_{k+1/2})(x - x_{k+1/2})^2 \cdots\right]dx$$

$$= f'(x_{k+1/2})\underbrace{\int_{x_k}^{x_{k+1}}(x - x_{k+1/2})dx}_{0} + \underbrace{\frac{1}{2}f''(x_{k+1/2})\int_{x_k}^{x_{k+1}}(x - x_{k+1/2})^2dx + \cdots}_{O(\Delta x^3)}$$

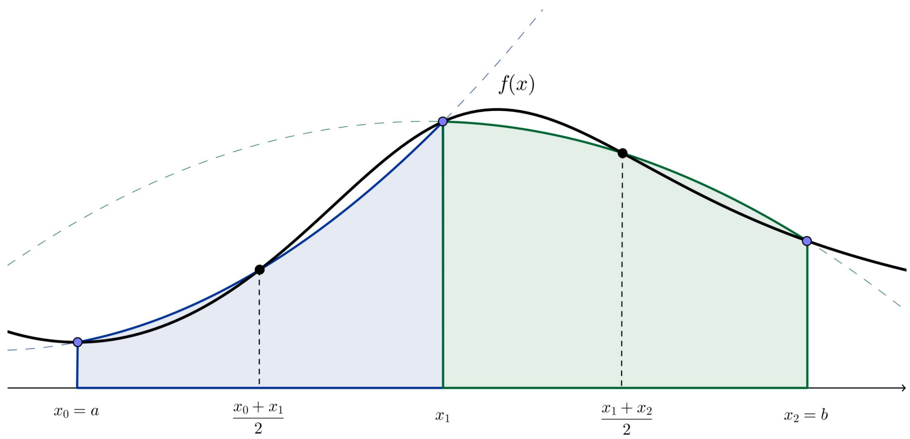
  - So in each interval $[x_k, x_{k+1}]$, $\boldsymbol{\int_{x_k}^{x_{k+1}} f(x)dx \overset{O(\Delta x^3)}{\approx} \int_{x_k}^{x_{k+1}} p(x)dx}$
  - $\int_a^b f(x)dx = \sum_{k=0}^{N-1} \int_{x_k}^{x_{k+1}} f(x)dx \overset{O(\Delta x^2)}{\approx} \sum_{k=0}^{N-1} \int_{x_k}^{x_{k+1}} p(x)dx = \Delta x[f(x_{1/2}) + f(x_{3/2}) + \cdots + f(x_{N-1/2})]$



Simpson's Rule

$2N$

- We divide $[a, b]$ to even number of intervals $\{x_0, \dots, x_{2N}\}$, where $x_k = a + k\Delta x$ and $\Delta x = \frac{b-a}{2N}$
- For each $[x_{2i}, x_{2i+2}]$, we look for a **quadratic polynomial** p that interpolates f at $x_{2i}, x_{2i+1}, x_{2i+2}$
  - $\int_{x_{2i}}^{x_{2i+2}} f(x)dx \overset{O(\Delta x^5)}{\approx} \int_{x_{2i}}^{x_{2i+1}} p(x)dx \underbrace{\left(= \sum_k f(x_k) l_k(x)\right)}_{\text{Lagrange interpolation}} = \frac{\Delta x}{3}[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})]$
- Then the approximation over the entire domain $[a, b]$ is
  - $\int_a^b f(x)dx \overset{O(\Delta x^4)}{\approx} \frac{\Delta x}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + \dots + 2f(x_{2N-2}) + 4f(x_{2N-1}) + f(x_{2N})]$
  - Note: $\boldsymbol{f(a), f(b)}$ **get weight** $\boldsymbol{\frac{1}{3}}$**, even terms get weight** $\boldsymbol{\frac{2}{3}}$**, and odd terms get weight** $\boldsymbol{\frac{4}{3}}$



Richardson Extrapolation

- Introduction
  - Let $\text{Tr}(f; N) = \Delta x \left[\frac{f(a)}{2} + \sum_{k=1}^{N-1} f(x_k) + \frac{f(b)}{2}\right]$ be the trapezoidal integration of f with N intervals
  - The error of $\text{Tr}(f; N)$ is
    - $E^{(N)} = \int_a^b f(x)dx - \text{Tr}(f; N) = \sum_{k=0}^{N-1} E_k = c \sum_{k=0}^{N-1} f''(\eta_i)\Delta x^3 = c \underbrace{\left[\sum_{k=0}^{N-1} f''(\eta_i)\Delta x\right]}_{\approx \sum f'(x_{i+1}) - f'(x_i)} \Delta x^2$

      $= c\Delta x^2[f'(x_1) - f'(x_0) + f'(x_2) - f'(x_1) + \cdots f(x_N) - f(x_{N-1})] + O(\Delta x^4)$

      $= c\Delta x^2[f'(b) - f'(a)] + O(\Delta x^4)$
  - The error of $\text{Tr}(f; 2N)$ is
    - $E^{(2N)} = c\left(\frac{\Delta x}{2}\right)^2 [f'(b) - f'(a)] + O(\Delta x^4) = \frac{c}{4}\Delta x^2[f'(b) - f'(a)] + O(\Delta x^4)$
  - Comparing $E^{(N)}$ and $E^{(2N)}$, we have
    - $E^{(2N)} = \frac{1}{4} E^{(N)} + O(\Delta x^4)$

- $\left(\int_a^b f(x)dx - \mathrm{Tr}(f;2N)\right) = \frac{1}{4}\left(\int_a^b f(x)dx - \mathrm{Tr}(f;N)\right) + O(\Delta x^4)$
- $\boldsymbol{\int_a^b f(x)dx = \frac{1}{3}\big(4\mathrm{Tr}(f;2N) - \mathrm{Tr}(f;N)\big) + O\big(\Delta x^4\big)}$
- Summary
  - Do trapezoidal for $N+1$ points
  - Do trapezoidal for $2N+1$ points
  - **Compute** $\boldsymbol{\frac{1}{3}\big(4\mathrm{Tr}(f;2N) - \mathrm{Tr}(f;N)\big)}$
- Note: Richardson extrapolation also extends the accuracy to higher orders

Undetermined Coefficients Method

- Introduction
  - For $f \in \mathbb{P}_{2N+1}$, if we know $f(x_0), \dots, f(x_N)$, then $\int_a^b f(x) = \sum_{i=0}^{N} f(x_i)w_i$
  - For f of higher degrees, how can we choose $\{x_0, \dots, x_N\}$ to have the best accuracy?
  - On the right hand side, x_i and w_i each contributes for $N+1$ degress of freedom
  - We can **solve for the coefficients $\boldsymbol{x_i}$ and $\boldsymbol{w_i}$** to obtain approximation with error $O\big(x^{2N}\big)$
- Example
  - Compute $\int_{-1}^{1} f(x)dx = w_1 f(x_1) + w_2 f(x_2)$ for the following f
    - If $f(x) = 1$, then $2 = w_0 + w_1$
    - If $f(x) = x$, then $0 = x_0 w_0 + x_1 w_1$
    - If $f(x) = x^2$, then $\frac{2}{3} = x_0^2 w_0 + x_1^2 w_1$
    - If $f(x) = x^3$, then $0 = x_0^3 w_0 + x_1^3 w_1$
  - Solving the system of equations above, we have
    - $\begin{cases} w_0 + w_1 = 2 \\ x_0 w_0 + x_1 w_1 = 0 \\ x_0^2 w_0 + x_1^2 w_1 = \frac{2}{3} \\ x_0^3 w_0 + x_1^3 w_1 = 0 \end{cases} \Rightarrow \begin{cases} w_0 = 1 \\ x_0 = -\frac{1}{\sqrt{3}} \end{cases}, \begin{cases} w_1 = 1 \\ x_1 = \frac{1}{\sqrt{3}} \end{cases}$
  - Therefore, $\int_{-1}^{1} f(x)\,dx = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$ for $f \in \mathbb{P}_3$

Review for Approximation & Integration

Monday, November 12, 2018 10:52 AM

Polynomial Interpolation

- Goal
  - Given $N+1$ grid points $\{x_0, \dots, x_N\}$ and their evaluation $\{f(x_0), \dots, f(x_N)\}$
  - We look for $\boldsymbol{p(x) = a_0 + a_1 \dots + a_N x^N} \in \mathbb{P}_N$ s.t. $\boldsymbol{p(x_i) = f(x_i)}, \forall i \in \{0, \dots, N\}$
- Method 1: Directly compute the coefficients (numerically inaccurate)
  - $$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^N \\ 1 & x_1 & x_1^2 & \cdots & x_1^N \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^N \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}}_{\vec{a}} = \underbrace{\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix}}_{\vec{f}}$$
  - If matrix A is not singular, then we have a **unique solution for $\vec{a}$**
  - But since A is Vandermonde matrix, it is **ill-conditioned**
  - So solving $A\vec{a} = \vec{f}$ directly will result in large numeric error in $\vec{a}$
- Method 2: **Lagrange interpolation**
  - For each sample point x_k, define $\boldsymbol{l_i(x) := \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_i - x_j}}$, then $\boldsymbol{l_i(x_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}}$
  - Define $\boldsymbol{p(x) := \sum_{i=1}^{N} f(x_i) l_i(x)}$, then $p(x_i) = f(x_i), \forall i \in \{0, \dots, N\}$
- Example of Lagrange interpolation
  - Given $\{x_0 = -1.5, x_1 = -0.5, x_2 = 0.5, x_3 = 1.5\}$, find the explicit formula for l_0
  - $l_0 = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} \cdot \frac{x - x_3}{x_0 - x_3} = -\frac{1}{6}(x + 0.5)(x - 0.5)(x - 1.5)$
- Error analysis
  - $E(x) = f(x) - p(x) = \frac{f^{(N+1)}(\xi)}{(N+1)!} \prod_{i=0}^{N} (x - x_i)$ for some $\xi \in \mathbb{R}$
  - For $f \in \mathbb{P}_N$
    - $f^{(N+1)}(\xi) = 0 \Rightarrow E(x) = 0 \Rightarrow f(x) = p(x), \forall x$
  - Example
    - Let $f(x) = 1$, then $1 = f(x) = p(x) = \sum_{i=0}^{N} f(x_i) l_i(x) = \sum_{i=0}^{N} l_i(x)$
  - For $f(x) = x^{N+1} + c_N x^N + \cdots + c_0 \in \mathbb{P}_{N+1}$

- $f^{(N+1)}(\xi) = (n+1)! \Rightarrow E(x) = \prod_{i=0}^{N}(x - x_i)$
- $\Rightarrow f(x) = E(x) + p(x) = \prod_{i=0}^{N}(x - x_i) + \sum_{i=0}^{n} f(x_i) l_i(x)$

- Example
  - Let $f(x) = (x-1)^{N+1}$, then $\sum_{i=0}^{N} f(x_i) l_i(x) = f(x) - \prod_{i=0}^{N}(x - x_i)$
  - Evaluate both sides at $x = 0$, then
  - $\sum_{i=0}^{N}(x-1)^{N+1} l_i(0) = (0-1)^{N+1} - \prod_{i=0}^{N}(0 - x_i) = (-1)^{N+1}\left[1 - \prod_{i=0}^{N} x_i\right]$

- Chebyshev nodes
  - $\min_{\{x_i\}} \sup_{x\in[a,b]} \left|\prod_{i=1}^{N}(x - x_i)\right| \Rightarrow x_k = \cos\theta_k$, where $\theta_k = k\frac{\pi}{N}$

Polynomial Projection

- Goal
  - Given $f \in C^\infty$, we look for $p \in \mathbb{P}_N$ s.t. $\boldsymbol{p}$ **best approximate** $\boldsymbol{f}$ **in** $\boldsymbol{L^2}$
- Orthonormal polynomials
  - Given domain $[a, b]$ and weight function $w(x) > 0$
  - $\{\phi_k\}_{k=0}^{+\infty}$ is said to be a sequence of **orthonormal polynomials** if
  - $\boldsymbol{\deg \phi_i = i}$ and $\boldsymbol{\langle \phi_m, \phi_n \rangle = \int_a^b \phi_m(x)\phi_n(x)w(x)dx = \delta_{mn}} = \begin{cases} 1 & m = n \\ 0 & m \neq n \end{cases}$
- Properties of orthogonal polynomials
  - Recurrence relation
    - $\boldsymbol{\phi_{N+1} = (\alpha_N x + \beta_N)\phi_N + \gamma_N \phi_{N-1}}$ for some $\alpha_N, \beta_N, \gamma_N \in \mathbb{R}$
    - The coefficients are determined by $\begin{cases} \langle \phi_{N+1}, \phi_{N+1} \rangle = 1 \\ \langle \phi_{N+1}, \phi_N \rangle = 0 \\ \langle \phi_{N+1}, \phi_{N-1} \rangle = 0 \end{cases}$
    - Then $\phi_{N+1} \perp \phi_{N-2}, \dots, \phi_0$ is satisfied automatically
  - ϕ_N has N zeros called **Gauss quadratures**
  - $\mathbf{eig}(\boldsymbol{A})$ **= Gauss quadratures of** $\boldsymbol{\phi_{N+1}}$
    - $\phi_{k+1} = (\alpha_k x + \beta_k)\phi_k + \gamma_k \phi_{k-1} \Rightarrow x\phi_k = \frac{1}{\alpha_k}(\phi_{k+1} - \beta_k \phi_k - \gamma_k \phi_{k-1})$

- $$x\underbrace{\begin{bmatrix}\phi_0(x)\\ \phi_1(x)\\ \vdots\\ \phi_{n-1}(x)\\ \phi_n(x)\end{bmatrix}}_{\vec{\phi}}=\underbrace{\begin{bmatrix}b_0 & c_0 & & & \\ a_1 & b_1 & c_1 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1}\\ & & & a_n & b_n\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}\phi_0(x)\\ \phi_1(x)\\ \vdots\\ \phi_{n-1}(x)\\ \phi_n(x)\end{bmatrix}}_{\vec{\phi}}+\begin{bmatrix}0\\ 0\\ \vdots\\ 0\\ c_n\phi_{n+1}(x)\end{bmatrix}$$
- $\boldsymbol{\phi_{n+1}(x)=0 \Leftrightarrow x\vec{\phi}=A\vec{\phi}}$, so $\mathbf{eig}(\boldsymbol{A})=\mathbf{GQ}(\boldsymbol{\phi_{n+1}})$

- Example: Hermite polynomials
  - Hermite polynomials $\{H_0, H_1, \ldots\}$ are defined on $(-\infty, \infty)$ with weight $w=\frac{1}{\sqrt{\pi}}e^{-x^2}$
  - Given $H_0=1, H_1=\sqrt{2}x, H_2=\frac{1}{\sqrt{8}}(4x^2-2)$
  - $\int_{-\infty}^{\infty}2x^2e^{-x^2}dx=\sqrt{\pi}\int H_1^2(x)\frac{1}{\sqrt{\pi}}e^{-x^2}dx=\sqrt{\pi}\langle H_1, H_1\rangle=\sqrt{\pi}$
  - $\int_{-\infty}^{\infty}(4x^2-2x-2)e^{-x^2}dx=\int_{-\infty}^{\infty}\left(\sqrt{8}H_2H_0-\sqrt{2}H_1H_0\right)e^{-x^2}dx=0$
- Best approximation
  - Given a function $f(x)=\sum_{k=0}^{+\infty}\alpha_k\phi_k(x)$, define the projection $\boldsymbol{p(x)=\sum_{k=0}^{N}\alpha_k\phi_k(x)}$
  - Then p is the best approximation in L^2 norm. Let $q\in\mathbb{P}_N$ be arbitrary, then
  - $\|f-q\|_2=\langle f-q, f-q\rangle=\langle(f-p)+(p-q),(f-p)+(p-q)\rangle$
    $=\langle f-p, f-p\rangle+2\underbrace{\langle f-p, p-q\rangle}_{0}+\underbrace{\langle p-q, p-q\rangle}_{\geq 0}$
    $\geq\langle f-p, f-p\rangle=\|f-p\|_2$
  - Note: $\|g\|_2=\sqrt{\langle g, g\rangle}=\sqrt{\int_a^b g^2(x)w(x)dx}$
- Coefficients approximation
  - $p(x)=\sum_{k=0}^{N}\alpha_k\phi_k(x)$, but α_k is hard to compute, so we use c_k to approximate it
  - $\alpha_k=\langle f, \phi_k\rangle=\int_a^b f(x)\phi_i(x)w(x)dx\approx\sum_{i=0}^{N}f(x_i)\phi_k(x_i)w_i=c_k$, where
    - $\boldsymbol{w_i=\int_a^b l_i(x)w(x)\,dx}$ and $\{\boldsymbol{x_0}, \ldots, \boldsymbol{x_N}\}$ are the **Gauss Quadratures** of ϕ_{N+1}
  - Theorem: **If** $\boldsymbol{f\in\mathbb{P}_{2N+1}}$**, then** $\boldsymbol{\int_a^b f(x)w(x)dx=\sum_{i=0}^{N}f(x_i)w_i}$, where
  - Corollary: If $f\in\mathbb{P}_{N+1}$, then $\alpha_k=c_k, \forall k\in\{0, \ldots, N\}$
- Error analysis

truncation

- $f(x) = \sum_{k=0}^{+\infty} \alpha_k \phi_k(x) \xrightarrow[f \in C^\nu \Rightarrow \alpha_k = O(k^{-\nu})]{\text{truncation}} p(x) = \sum_{k=0}^{N} \alpha_k \phi_k(x)$
- $p(x) = \sum_{k=0}^{N} \alpha_k \phi_k(x) \xrightarrow[c_k - a_k \sim \{\alpha_{n+1}, \dots\} \text{ small}]{\text{numerical integration}} \tilde{p}(x) = \sum_{k=0}^{N} c_k \phi_k(x)$

Numerical Integration

- Trapezoidal rule
  - $\int_a^b f(x)dx \approx h\left[\frac{1}{2}f(a) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(b)\right]$
- Simpson's rule
  - $\int_a^b f(x)dx \approx h\left[\frac{f(a)}{6} + \frac{4}{6}f(x_1) + \frac{2}{6}f(x_2) + \cdots + \frac{f(b)}{6}\right]$
- Undetermined coefficients
  - $\int_{-1}^1 f(x)dx \approx Af(x_0) + Bf(x_1)$
  - How can we choose A, B, x_0, x_1 s.t. the approximation is the best?
  - $\begin{cases} 2 = A + B & \text{if } f(x) = 1 \\ 0 = Ax_0 + Bx_1 & \text{if } f(x) = x \\ 2/3 = Ax_0^2 + Bx_1^2 & \text{if } f(x) = x^2 \\ 0 = Ax_0^3 + Bx_1^3 & \text{if } f(x) = x^3 \end{cases} \Rightarrow \begin{cases} A = 1 \\ B = 1 \\ x_0 = -1/\sqrt{3} \\ x_1 = -1/\sqrt{3} \end{cases}$

Ch 12: Numerical ODE

Friday, December 7, 2018 10:50 PM

Introduction to Numerical ODE

Monday, November 12, 2018 10:36 AM

Numerical ODE

- Introduction
  - Given $\begin{cases} y' = f(x) \\ y(a) = A \end{cases}$, we can compute $y(b) = A + \int_a^b f(x)dx$ by numerical integration
  - In numerical ODE, we are **given** $\begin{cases} \boldsymbol{y' = f(x,y)} \\ \boldsymbol{y(a) = A} \end{cases}$ and **want** $\boldsymbol{y(b) = y(a) + \int_a^b f(x,y)dx}$
  - But for $f(x,y)$, we don't know the exact value for parameter y
  - Instead, we can **take small steps in $\boldsymbol{x}$ and approximate $\boldsymbol{y_n \approx y(x_n)}$** in each step
- **Initial value problem**
  - Given $\boldsymbol{u'(t) = f(t,u)}$ and initial condition $\boldsymbol{u(t=0) = u_0}$, we want to find $u(t=T)$ for some T

Reducing N^{th} Order Non-Autonomous ODE

- Autonomous
  - If the force f has **no explicit dependence on** t, then we call the ODE **autonomous**
- System of first order autonomous ODE
  - $\underbrace{\begin{bmatrix} \boldsymbol{u_1'} \\ \vdots \\ \boldsymbol{u_n'} \end{bmatrix}}_{\vec{u}'} = \underbrace{\begin{bmatrix} \boldsymbol{f_1(u_1,\dots,u_n)} \\ \vdots \\ \boldsymbol{f_n(u_1,\dots,u_n)} \end{bmatrix}}_{\vec{f}(\vec{u})}$ with initial condition $\boldsymbol{\vec{u}(t=0) = \vec{u}_0} \coloneqq \begin{bmatrix} \boldsymbol{u_1(t=0)} \\ \vdots \\ \boldsymbol{u_n(t=0)} \end{bmatrix}$
- Reducing to first order autonomous ODE
  - Given any **higher order, non-autonomous ODE** $\vec{u}^{(n)} = f\left(t,u,u',\dots,u^{(n-1)}\right)$
  - We can **reduce it to first order autonomous ODE system** $\begin{cases} \boldsymbol{\vec{u}' = \vec{f}(\vec{u})} \\ \boldsymbol{\vec{u}(t=0) = \vec{u}_{in}} \end{cases}$ and find $\vec{u}(t)$
  - Therefore numerically, we only study first order autonomous ODEs
- Example
  - We want to solve $u''' = u'u - 2t(u')^2$ with initial conditions $\begin{cases} u(t=0) = u_0 \\ u'(t=0) = u_1 \\ u''(t=0) = u_2 \end{cases}$
  - Define $\begin{cases} y_0(t) = u(t) \\ y_1(t) = u'(t) \\ y_2(t) = u''(t) \end{cases}$, then we have $y_2' = y_0 y_1 - 2t y_1^2$ with $\begin{cases} y_0(t=0) = u_0 \\ y_1(t=0) = u_1 \\ y_2(t=0) = u_2 \end{cases}$
  - To reduce to an autonomous ODE system, define $y_3(t) = t$
  - So we only need to solve $\underbrace{\begin{bmatrix} y_0' \\ y_1' \\ y_2' \\ y_3' \end{bmatrix}}_{\vec{y}'} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_0 y_1 - 2 y_3 y_1^2 \\ 1 \end{bmatrix}}_{\vec{f}(\vec{y})}$ with initial condition $\vec{y}(t=0) = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ 1 \end{bmatrix}$

Existence and Uniqueness of First Order ODE (Picard's Theorem)

- Different types of continuous

Continuous at x^*	If $x \to x^*$, then $\lvert f(x) - f(x^*)\rvert \to 0$
Lipschitz continuous at x^*	$\exists L_{x^*} \in \mathbb{R}$ s.t. $\lvert f(x) - f(x^*)\rvert \le L_{x^*}\lvert x - x^*\rvert$ for $x \in B_\varepsilon(x^*)$
Uniformly Lipschitz	$\boldsymbol{L_{x^*}}$ **is bounded** $\forall \boldsymbol{x^*}$

  - Note: In the case of Lipschitz continuity, if $f'(x^*)$ exists, then $L_{x^*} = f'(x^*)$
- Picard's Theorem
  - If $f(u)$ is **uniformly Lipschitz**, then the equation has a **unique solution**
- Example 1
  - Given $u' = u^2$ with initial condition $u(t = 0) = \eta$
  - Here $f(u) = u^2$ is not uniformly Lipschitz since $L_u = \lvert f'(u)\rvert = 2u$ is not bounded
  - u has an explicit solution $u(t) = \dfrac{\eta}{1 - \eta t}$, but it blows up at $t = \dfrac{1}{\eta}$
- Example 2
  - Given $u' = \sqrt{u}$ with initial condition $u(t = 0) = 0$
  - Since $L_u = \lvert f'(u)\rvert = \dfrac{1}{2\sqrt{u}}$ does not exist at $u = 0$, $f(u)$ is not uniformly Lipschitz
  - It turns out that there exist multiple solutions, such as $u(t) = \dfrac{1}{4}t^2$ or $u(t) = 0$

Three Key Concepts

- Consistency
  - **Local truncation error** measures **how much the true solution fail to satisfy the scheme**
  - If local truncation error $\boldsymbol{\tau_n} \to \mathbf{0}$ **as** $\boldsymbol{h} \to \mathbf{0}$, then we say the method is **consistent**
- Stability
  - The **error** $\boldsymbol{E_n} = \boldsymbol{u}(\boldsymbol{t_n}) - \boldsymbol{U_n}$ in each step should **not be amplified in the future**
- Convergence
  - We say a method **converges** if the numerical solution $\boldsymbol{U_T} \to \boldsymbol{u}(\boldsymbol{t} = \boldsymbol{T})$ **as** $\boldsymbol{h} \to \mathbf{0}$
  - Lax theorem says that for **linear ODE**, **consistency and stability imply convergence**

General Methodology

- First approximate the differential operator by a difference operator using
  - Finite difference method (Euler method, Trapezoidal rule, etc.)
  - Interpolation + differentiation
    - $p(x) := f(x_0)l_0(x) + f(x_0 + h)l_1(x) + f(x_0 + 2h)l_2(x)$, then $p'(x_0) \approx f'(x_0)$
- Then solve the resulting discrete system using
  - LU decomposition or QR decomposition (for linear system)
  - Newton's method (for non-linear system)

Euler & Trapezoidal & Runge-Kutta

Friday, December 7, 2018 10:39 PM

Euler's Method

- Forward Euler method for $f'(x_0)$
  - We look for a, b, c s.t. $\boldsymbol{f'(x_0) \approx af(x_0) + bf(x_0+h) + cf(x_0+2h)}$ **is best approximated**
  - By Taylor expansion, $\begin{cases} f(x_0+h) = f(x_0) + f'(x_0)h + \dfrac{h^2}{2}f''(x_0) + \cdots \\ f(x_0+2h) = f(x_0) + f'(x_0)2h + \dfrac{(2h)^2}{2}f''(x_0) + \cdots \end{cases}$
  - $f'(x_0) \approx af(x_0) + b\underbrace{\left[f(x_0) + f'(x_0)h + \frac{h^2}{2}f''(x_0)\right]}_{f(x_0+h)} + c\underbrace{\left[f(x_0) + f'(x_0)2h + \frac{(2h)^2}{2}f''(x_0)\right]}_{f(x_0+2h)}$
  - Collecting the coefficients of f, f', f'', we have $\begin{cases} 0 = a + b + c \\ 1 = bh + 2ch \\ 0 = \dfrac{bh^2}{2} + \dfrac{c(2h)^2}{2} \end{cases} \Rightarrow \begin{cases} a = -3/2\,h^{-1} \\ b = 2h^{-1} \\ c = -1/2\,h^{-1} \end{cases}$
  - The error term is $\boldsymbol{f'(x_0) - \underbrace{\left[-\frac{3}{2h}f(x_0) + \frac{2}{h}f(x_0+h) - \frac{1}{2h}f(x_0+2h)\right]}_{\text{difference operator}} = \mathcal{O}(h^2)}$
  - Similarly, If we only take one step, then $\boldsymbol{f'(x_0) - \underbrace{\left[-\frac{1}{h}f(x_0) + \frac{1}{h}f(x_0+h)\right]}_{\text{difference operator}} = \mathcal{O}(h)}$
- Central Euler method for $f''(x_0)$
  - We look for a, b, c s.t. $\boldsymbol{f''(x_0) \approx af(x_0-h) + bf(x_0) + cf(x_0+h)}$ **is best approximated**
  - Collecting the coefficients of f, f', f'', we have $\begin{cases} 0 = a + b + c \\ 0 = h(a-c) \\ 1 = \frac{1}{2}h^2(a+c) \end{cases} \Rightarrow \begin{cases} a = h^{-2} \\ b = -2h^{-2} \\ c = h^{-2} \end{cases}$
  - Due to symmetry, **the fourth equation** $\boldsymbol{0 = \frac{h^2}{6}(a-c)}$ **is automatically satisfied**
  - Therefore $\boldsymbol{f''(x_0) - \left[\frac{1}{h^2}f(x_0-h) - \frac{2}{h^2}f(x_0) + \frac{1}{h^2}f(x_0+h)\right] = \mathcal{O}(h^2)}$
- Example of one-step forward Euler method
  - We want to solve $u' = f(u) = \lambda u$ with initial condition $u(t=0) = u_0$
  - Denote $u_n = u(t_n)$ to be the true solution at t_n and U_n the numerical solution at t_n
  - Then $u'(t_n) \approx \frac{1}{\Delta t}(U_{n+1} - U_n) = f(U_n) = \lambda U_n \Rightarrow \frac{1}{\Delta t}U_{n+1} - \left(\lambda + \frac{1}{\Delta t}\right)U_n = 0$
  - Define $\mu = \lambda \Delta t + 1$, then $\underbrace{\frac{1}{\Delta t}\begin{bmatrix} 1 & & & \\ -\mu & 1 & & \\ & \ddots & \ddots & \\ & & -\mu & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix}}_{\vec{U}} = \underbrace{\begin{bmatrix} (\Delta t^{-1} + \lambda)U_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{S}$

- Computing the inverse of A, we have $A^{-1} = \begin{bmatrix} 1 & & & & & \\ \mu & 1 & & & & \\ \mu^2 & \mu & \ddots & & & \\ \mu^3 & \ddots & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \mu & 1 & \\ \mu^{n-1} & \cdots & \mu^3 & \mu^2 & \mu & 1 \end{bmatrix}$
- Thus, $U_n = [A^{-1}S]_N = \Delta t(1+\lambda\Delta t)^{N-1}(\Delta t^{-1}+\lambda)u_0 = u_0(1+\lambda\Delta t)^N$
- Let $T = N\Delta t \Leftrightarrow \Delta t = \frac{T}{N}$, then $\boldsymbol{U_N = u_0\left(1+\frac{\lambda T}{N}\right)^N \to u_0 e^{\lambda T}}$ **as** $\boldsymbol{N \to \infty}$
- This is the same as the analytical solution $u(T) = u_0 e^{\lambda T}$

Analysis for Euler Method

- Consistency
  - We want to show that the **local truncation error** $\boldsymbol{\tau_n \to 0}$ **as** $\boldsymbol{\Delta t \to 0}$
  - $\tau_n = \frac{u_{n+1}-u_n}{\Delta t} - f(u_n) = \frac{1}{\Delta t}\left(u_n + u'(t_n)\Delta t + \frac{1}{2}u''(t_n)\Delta t^2 + \cdots - u_n\right) - f(u_n)$

    $= u'(t_n) + \frac{1}{2}u''(t_n)\Delta t + \cdots - \underbrace{f(u_n)}_{u'(t_n)} = \frac{1}{2}u''(t_n)\Delta t + \cdots = \mathcal{O}(\Delta t)$
  - Therefore one-step forward Euler method is consistent
- Stability
  - We want to show that the error $\boldsymbol{E_n = u(t_n) - U_n}$ is **not amplified** in the future
  - In the example above, $\begin{cases} A\vec{U} = S \\ \vec{\tau} = A\vec{u} - S \end{cases} \Rightarrow A\underbrace{\left(\vec{u}-\vec{U}\right)}_{\vec{E}} = \vec{\tau} \Rightarrow \vec{E} = A^{-1}\vec{\tau}$
  - $\left\|\vec{E}\right\|_\infty \le \left\|A^{-1}\right\|_\infty \|\vec{\tau}\|_\infty$, where $\left\|A^{-1}\right\|_\infty = \Delta t \sum_{k=0}^{n-1} u^k \le \frac{1}{\lambda}\left(e^{\lambda\Delta t}\right)^N = \frac{1}{\lambda}e^{\lambda T}$ and $\|\vec{\tau}\|_\infty = \mathcal{O}(\Delta t)$
  - Since $\left\|\vec{\boldsymbol{E}}\right\|_\infty \le \boldsymbol{\mathcal{O}(\Delta t)} \Rightarrow \lim_{\Delta t \to 0} \vec{\boldsymbol{E}} = \vec{\boldsymbol{0}}$, **one-step forward Euler method is stable**
- Convergence (for non-linear case)
  - If $\boldsymbol{f(u)}$ is **Lipschitz** (with constant λ), then **Euler method is linearly convergent**
  - The numerical solution at each step is
    - $\boldsymbol{U_{n+1} = U_n + h \cdot f(U_n)}$
  - The true solution at each step is
    - $u_{n+1} = u(t_{n+1}) = u(t_n + h) = u(t_n) + u'(t_n)h + \mathcal{O}(h^2) = \boldsymbol{u(t_n) + hf\big(u(t_n)\big) + \mathcal{O}\big(h^2\big)}$
  - $E_{n+1} = |U_{n+1} - u_{n+1}| = \left|\big(U_n + h\cdot f(U_n)\big) - \Big(u(t_n) + h\cdot f\big(u(t_n)\big) + \mathcal{O}\big(h^2\big)\Big)\right|$

    $= E_n + h|f(U_n) - f(u_n)| + \mathcal{O}\big(h^2\big)$

    $\le E_n + h\lambda \underbrace{|U_n - u_n|}_{E_n} + \mathcal{O}\big(h^2\big)$

    $= (1+\lambda h)E_n + \mathcal{O}\big(h^2\big)$
  - $\boldsymbol{E_n \le (1+\lambda h)E_{n-1} + \mathcal{O}\big(h^2\big)}$

$$\le (1+\lambda h)\Big((1+\lambda h)E_{n-2} + \mathcal{O}(h^2)\Big) + \mathcal{O}(h^2)$$

$$\le (1+\lambda h)^2 E_{n-2} + \mathcal{O}(h^2) + \mathcal{O}(h^2)$$

$$\le \cdots \le (1+\lambda h)^{n-1}E_1 + \underbrace{\boldsymbol{\mathcal{O}(h^2)} + \cdots + \boldsymbol{\mathcal{O}(h^2)}}_{\boldsymbol{n}\ \textbf{copies}}$$

$$\le (1+\lambda h)^{n-1}E_1 + \mathcal{O}(h), \text{since } \mathcal{O}(nh^2) = \mathcal{O}(Th) = \mathcal{O}(h)$$

- Let $nh = T \Leftrightarrow h = \frac{T}{n}$, then $(1+\lambda h)^{n-1} = \left(1+\frac{\lambda T}{n}\right)^{n-1} \le e^{\lambda T}$
- $E_1 = \mathcal{O}(h^2)$ since $\begin{cases} \frac{U_1 - U_0}{h} = f(U_0) \\ \frac{u_1 - u_0}{h} = f(u_0) + \mathcal{O}(h) \end{cases} \Rightarrow \begin{cases} U_1 = hf(U_0) + U_0 \\ u_1 = hf(u_0) + u_0 + \mathcal{O}(h^2) \end{cases}$
- Since $E_n \le (1+\lambda h)^{n-1}E_1 + \mathcal{O}(h) = \mathcal{O}(h)$, **Euler method is linearly convergent**

Trapezoidal Rule

- Scheme
  - Approximate $u' = f(u)$ using $\boldsymbol{\frac{U_{n+1} - U_n}{h} = \frac{1}{2}\big(f(U_n) + f(U_{n+1})\big)}$
- Example: $f(u) = u^2$
  - $\frac{U_{n+1} - U_n}{h} = \frac{1}{2}\left(U_n^2 + U_{n+1}^2\right) \Rightarrow \underbrace{\boldsymbol{U_{n+1} + \frac{1}{2}hU_{n+1}^2}}_{\textbf{unknown}} = \underbrace{\boldsymbol{U_n + \frac{1}{2}hU_n^2}}_{\textbf{known}}$
  - Since the relation is implicit, we need to use Newton's method to solve for U_{n+1} at each step
- Consistency
  - $\tau_n = \frac{u_{n+1} - u_n}{h} - \frac{1}{2}\big(f(u_n) + f(u_{n+1})\big)$, where
  - $\frac{u_{n+1} - u_n}{h} = \frac{1}{h}\left(u_n + u_n'h + \frac{1}{2}u_n''h^2 + \cdots - u_n\right) = u_n' + \frac{1}{2}u_n''h + \mathcal{O}(h^2)$
  - $\frac{1}{2}\big(f(u_n) + f(u_{n+1})\big) = \frac{1}{2}\left[f(u_n) + f\left(u_n + u_n'h + \frac{1}{2}u_n''h^2 + \cdots\right)\right]$

    $$= \frac{1}{2}\left[f(u_n) + f(u_n) + f'(u_n)(u_n'h + \cdots) + \frac{1}{2}f''(u_n)(u_n'h + \cdots)^2 + \cdots\right]$$

    $$= f(u_n) + \frac{1}{2}f'(u_n)u_n'h + \mathcal{O}(h^2)$$

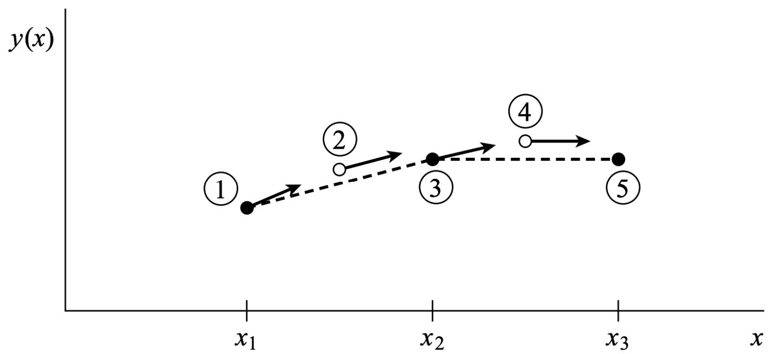
    $$= u_n' + \frac{1}{2}u_n''h + \mathcal{O}(h^2), \text{since } u_n' = f(u_n) \Leftrightarrow u_n'' = f'(u_n)u_n'$$
  - Thus, $\tau_n = \frac{u_{n+1} - u_n}{h} - \frac{1}{2}\big(f(u_n) + f(u_{n+1})\big) = \mathcal{O}(h^2)$
- Convergence
  - If $f(u)$ is Lipschitz (with constant λ), then trapezoidal rule is **2^{nd} order convergent**
  - The numerical solution and true solution at each time step is
    - $U_{n+1} = U_n + \frac{h}{2}\big(f(U_n) + f(U_{n+1})\big)$

- $u_{n+1} = u_n + \frac{h}{2}\left(f(u_n) + f(u_{n+1})\right) + \mathcal{O}(h^3)$

  - Subtract two equations and apply Lipschitz condition, we have
    - $E_{n+1} \le E_n + \frac{h}{2}(\lambda E_n + \lambda E_{n+1}) + \mathcal{O}(h^3)$
    - $E_{n+1} \le \dfrac{1+\frac{1}{2}\lambda h}{1-\frac{1}{2}\lambda h} E_n + \mathcal{O}(h^3) \le \left[\dfrac{1+\frac{1}{2}\lambda h}{1-\frac{1}{2}\lambda h}\right]^{n-1} E_1 + \mathcal{O}(nh^3) = \mathcal{O}(h^2)$
  - Therefore, **trapezoidal rule converges quadratically**

Runge-Kutta Method

- Introduction
  - The key idea of IVP is to find $\boldsymbol{U_{n+1} = U_n + \int_{t_n}^{t_{n+1}} \underbrace{f(u(t))}_{\text{unknown}} dt}$
  - In forward Euler method, we replace the integral by $hf(U_n)$
  - In trapezoidal rule, we replace the integral by $\frac{h}{2}\left(f(U_n) + f(U_{n+1})\right)$
  - In **Runge-Kutta method**, the integral is **replaced by summation** $\sum_{i=1}^{N} f\left(U_{n+\alpha_i h}\right) w_i$
- Example of RK-2: **midpoint method / modified Euler method**
  - General idea



    - Use Euler method to **calculate midpoint location** U^* (open dots ② ④)
    - **Evaluate slope $\boldsymbol{f(U^*)}$** at the midpoint (arrow after ② ④)
    - Use the slope the **calculate full step location** (filled dots ③ ⑤)
  - Formula
    - $\begin{cases} U^* = U_n + \frac{h}{2} f(U_n) \\ U_{n+1} = U_n + hf(U^*) \end{cases} \Rightarrow U_{n+1} = U_n + hf\left(U_n + \frac{h}{2} f(U_n)\right)$
  - Example
    - Suppose $u' = u$, then the analytical solution is $u(t) = Ce^t$
    - $U_{n+1} = U_n + h\left(U_n + \frac{h}{2} U_n\right) = \left(1 + h + \frac{h^2}{2}\right) U_n = e^h U_n + \mathcal{O}(h^3)$
- Common Runge-Kutta methods

Name	Butcher tableau	Scheme
Classical RK-4	$\begin{array}{c\|cccc} 0 & 0 & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$	$\begin{cases} y_1 = U_n \\ y_2 = U_n + \frac{1}{2}hf(y_1) \\ y_3 = U_n + \frac{1}{2}hf(y_2) \\ y_4 = U_n + hf(y_3) \\ U_{n+1} = U_n + h\left[\frac{f(y_1)}{6} + \frac{f(y_2)}{3} + \frac{f(y_3)}{3} + \frac{f(y_4)}{6}\right] \end{cases}$
Midpoint (RK-2)	$\begin{array}{c\|cc} 0 & 0 & \\ 1/2 & 1/2 & \\ \hline & 0 & 1 \end{array}$	$\begin{cases} y_1 = U_n \\ y_2 = U_n + \frac{1}{2}hf(y_1) \\ U_{n+1} = U_n + hf(y_2) \end{cases}$
Heun (RK-2)	$\begin{array}{c\|cc} 0 & 0 & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$	$\begin{cases} y_1 = U_n \\ y_2 = U_n + hf(y_1) \\ U_{n+1} = U_n + h\left[\frac{1}{2}f(y_1) + \frac{1}{2}f(y_2)\right] \end{cases}$
Ralston (RK-2)	$\begin{array}{c\|cc} 0 & 0 & \\ 2/3 & 2/3 & \\ \hline & 1/4 & 3/4 \end{array}$	$\begin{cases} y_1 = U_n \\ y_2 = U_n + \frac{2}{3}hf(y_1) \\ U_{n+1} = U_n + h\left[\frac{1}{4}f(y_1) + \frac{3}{4}f(y_2)\right] \end{cases}$
Generic RK-2	$\begin{array}{c\|cc} 0 & 0 & \\ \alpha & \alpha & \\ \hline & \beta & 1-\beta \end{array}$ for $\alpha(1-\beta) = 1/2$	$\begin{cases} y_1 = U_n \\ y_2 = U_n + \alpha hf(y_1) \\ U_{n+1} = U_n + h[\beta f(y_1) + (1-\beta)f(y_2)] \end{cases}$, or $\frac{U_{n+1} - U_n}{h} = \beta f(U_n) + (1-\beta)f\big(U_n + \alpha hf(U_n)\big)$

- Consistency for generic RK-2
    - $\tau_n = \frac{u_{n+1} - u_n}{h} - \beta f(u_n) + (1-\beta)f\big(u_n + \alpha hf(u_n)\big)$, where
    - $\frac{u_{n+1} - u_n}{h} = \frac{1}{h}\left[\left(u_n + hu_n' + \frac{1}{2}h^2u_n'' + \mathcal{O}(h^3)\right) - u_n\right] = u_n' + \frac{1}{2}hu_n'' + \mathcal{O}(h^2)$
    - $\beta f(u_n) + (1-\beta)f\big(u_n + \alpha hf(u_n)\big)$

      $$= \beta f(u_n) + (1-\beta)\underbrace{\left[f(u_n) + \alpha hf'(u_n)f(u_n) + \mathcal{O}(h^2)\right]}_{\text{Taylor expan. of } f(u_n + \alpha hf(u_n))}$$

      $$= \beta u_n' + (1-\beta)[u_n' + \alpha hu_n''] + \mathcal{O}(h^2)$$

      $$= u_n'(1-\beta) + \beta u_n' + \alpha(1-\beta)hu_n'' + \mathcal{O}(h^2)$$

      $$= u_n' + \frac{1}{2}hu_n'' + \mathcal{O}(h^2)$$
    - Thus, $\tau_n = \mathcal{O}(h^2)$
- Remark
    - **4-th order** Runge-Kutta is the **highest** order where the **stage number = order of accuracy**

Linear Multistep Methods & Stability

Friday, November 30, 2018 11:20 AM

Linear Multistep Methods (LMM)

- Adams–Bashforth methods
  - We can **approximate** $\boldsymbol{f(u(t))}$ by **polynomial interpolation at** $\boldsymbol{(t_n, f(U_n)), (t_{n-1}, f(U_{n-1})), \cdots}$
  - *i.e.* $f(u(t)) \approx p(t) = f(U_n)l_n(t) + f(U_{n-1})l_{n-1}(t) + f(U_{n-2})l_{n-2}(t) + \cdots$
  - Interpolation at $(t_n, f(U_n))$ and $(t_{n-1}, f(U_{n-1}))$ gives $p(t) = \frac{t - t_{n-1}}{h} f(U_n) + \frac{t_n - t}{h} f(U_{n-1})$
  - $$U_{n+1} = U_n + \int_{t_n}^{t_{n+1}} f(u(t))dt \approx U_n + \int_{t_n}^{t_{n+1}} p(t)dt$$
    $$= U_n + \left[\int_{t_n}^{t_{n+1}} \frac{t - t_{n-1}}{h} dt\right] f(U_n) + \left[\int_{t_n}^{t_{n+1}} \frac{t_n - t}{h} dt\right] f(U_{n-1})$$
    $$= U_n + \left[\frac{(t - t_{n-1})^2}{2h}\right]_{t_n}^{t_{n+1}} f(U_n) + \left[-\frac{(t_n - t)^2}{2h}\right]_{t_n}^{t_{n+1}} f(U_{n-1})$$
    $$= U_n + \left[\frac{(t_{n+1} - t_{n-1})^2}{2h} - \frac{(t_n - t_{n-1})^2}{2h}\right] f(U_n) + \left[\frac{(t_n - t_n)^2}{2h} - \frac{(t_n - t_{n+1})^2}{2h}\right] f(U_{n-1})$$
    $$= U_n + \left[\frac{(2h)^2}{2h} - \frac{h^2}{2h}\right] f(U_n) + \left[0 - \frac{h^2}{2h}\right] f(U_{n-1})$$
    $$= \boldsymbol{U_n + \frac{3h}{2} f(U_n) - \frac{h}{2} f(U_{n-1})}$$
  - Interpolation at 3 sample points gives $\boldsymbol{U_{n+1} = U_n + \frac{h}{12}(23f(U_n) - 16f(U_{n-1}) + 5f(U_{n-2}))}$
- General LMM
  - A general LMM has the form $\displaystyle \boldsymbol{\sum_{i=0}^{r} \alpha_i U_{n+i} = h \sum_{i=0}^{r} \beta_i f(U_{n+i})}$
  - If we **know** $\boldsymbol{U_n, U_{n+1}, \dots, U_{n+r-1}}$, then we can use the formula above to **estimate** $\boldsymbol{U_{n+r}}$
- Explicit vs. implicit method
  - $$\sum_{i=0}^{r} \alpha_i U_{n+i} = h \sum_{i=0}^{r} \beta_i f(U_{n+i}) = h \sum_{i=0}^{r-1} \beta_i f(U_{n+i}) + \boldsymbol{h\beta_r f(U_{n+r})}$$
  - If $\beta_r = 0$, then we have a explicit method
  - If $\beta_r \neq 0$, then the method is implicit, and we need to use Newton's method to solve for U_{n+r}
  - **Adams–Bashforth methods** are examples of **explicit linear multistep methods**

Consistency of LMM

- Local truncation error

- $\tau_n = \frac{1}{h}\sum_{j=0}^{r}\alpha_j u_{n+j} - \sum_{j=0}^{r}\beta_j f(u_{n+j})$

$$= \frac{1}{h}\sum_{j=0}^{r}\alpha_j\left[u_n + u_n' jh + \frac{1}{2}u_n''(jh)^2 + \cdots\right] - \sum_{j=0}^{r}\beta_j\left[u_n' + u_n'' jh + \frac{1}{2}u_n'''(jh)^2 + \cdots\right]$$

$$= \underbrace{\frac{1}{h}\left[\sum_{j=0}^{r}\alpha_j\right]u_n + \left[\sum_{j=0}^{r}j\alpha_j - \sum_{j=0}^{r}\beta_j\right]u_n'}_{\text{need to be 0 so that } \tau_n \to 0 \text{ as } h \to 0} + h\left[\frac{1}{2}\sum_{j=0}^{r}j^2\alpha_j - \sum_{j=0}^{r}j\beta_j\right]u_n'' + \cdots$$

- For consistency, we **require** $\boldsymbol{\sum_{j=0}^{r}\alpha_j u_n = 0}$ and $\boldsymbol{\sum_{j=0}^{r}j\alpha_j = \sum_{j=0}^{r}\beta_j}$

- Characteristic polynomial
  - Define $\boldsymbol{\rho(\xi) = \sum_{i=0}^{r}\alpha_i\xi^i}$ to be the characteristic polynomial for $\sum_{i=0}^{r}\alpha_i U_{n+i}$
  - Define $\boldsymbol{\sigma(\xi) = \sum_{i=0}^{r}\beta_i\xi^i}$ to be the characteristic polynomial for $\sum_{i=0}^{r}\beta_i f(U_{n+i})$
  - Then the requirement for **consistency** is $\begin{cases}\boldsymbol{\rho(1) = 0}\\ \boldsymbol{\rho'(1) = \sigma(1)}\end{cases}$
- Example 1: $U_{n+2} - U_{n+1} = h\left[\frac{3}{2}f(U_{n+1}) - \frac{1}{2}f(U_n)\right]$
  - $\begin{cases}\alpha_0 = 0\\ \alpha_1 = -1\\ \alpha_2 = 1\end{cases}, \begin{cases}\beta_0 = -1/2\\ \beta_1 = 3/2\\ \beta_2 = 0\end{cases} \Rightarrow \begin{cases}\rho(\xi) = 0 - \xi + \xi^2\\ \sigma(\xi) = -\frac{1}{2} + \frac{3}{2}\xi\end{cases} \Rightarrow \begin{cases}\rho(1) = 0\\ \rho'(1) = \sigma(1) = 1\end{cases} \Rightarrow$ Consistent method
- Example 2: $U_{n+2} - 3U_{n+1} + 2U_n = -hf(U_n)$
  - $\begin{cases}\alpha_0 = 2\\ \alpha_1 = -3\\ \alpha_2 = 1\end{cases}, \begin{cases}\beta_0 = -1\\ \beta_1 = 0\\ \beta_2 = 0\end{cases} \Rightarrow \begin{cases}\rho(\xi) = 2 - 3\xi + \xi^2\\ \sigma(\xi) = -1\end{cases} \Rightarrow \begin{cases}\rho(1) = 0\\ \rho'(1) = \sigma(1) = -1\end{cases} \Rightarrow$ Consistent method

Zero Stability of LMM

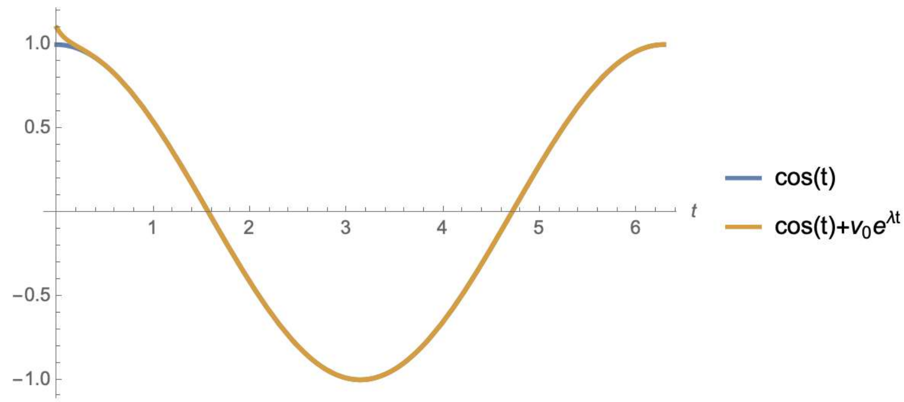
- Zero stability (informal)
  - A scheme is said to be **zero stable** if **perturbations remain bounded as $\boldsymbol{n}$** $\to \infty$
- Test problem for zero stability
  - We consider the **test problem** $\begin{cases}\boldsymbol{u' = 0}\\ \boldsymbol{u(0) = 0}\end{cases}$, where the analytical solution $u(t) \equiv 0$
  - For zero stability, we need the numerical solution to be bounded (by some constant)
  - Note that zero stability is called so since we assume **the force term $\boldsymbol{f}$ to be zero**
- Motivating example
  - Compute the test problem $\begin{cases}u' = 0\\ u(0) = 0\end{cases}$ using the scheme $U_{n+2} - 3U_{n+1} + 2U_n = -hf(U_n)$
  - Substitute in $f(u) = u' = 0$, then the scheme becomes $U_{n+2} - 3U_{n+1} + 2U_n = 0$

- With $U_0 = 0, U_1 = h$, we have $U_5 = 4.2, U_{10} = 258.4, \ldots$, so this **scheme is not zero-stable**
- To better study zero stability, we want to get an analytical solution for this method
- Write $U_{n+2} - 3U_{n+1} + 2U_n = 0$ in matrix form, we have $\begin{bmatrix} \boldsymbol{U_{n+2}} \\ \boldsymbol{U_{n+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{3} & \mathbf{-2} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{U_{n+1}} \\ \boldsymbol{U_n} \end{bmatrix}$
- The **characteristic polynomial** for this recurrenc relation is $\boldsymbol{\rho(\xi)} = \xi^2 - 3\xi + 2$
- $\xi^2 - 3\xi + 2 = 0 \Rightarrow \begin{cases} \xi_1 = 1 \\ \xi_2 = 2 \end{cases} \Rightarrow U_n = c_1\xi_1^n + c_2\xi_2^n = c_1 + c_2 2^n$ for some constant $c_1, c_2 \in \mathbb{R}$
- To determine the constant, we can use the initial values $\begin{cases} U_0 = c_1 + c_2 \\ U_1 = c_2 + 2c_2 \end{cases} \Rightarrow \begin{cases} c_1 = 2U_0 - U_1 \\ c_2 = U_1 - U_0 \end{cases}$
- Therefore, $\boldsymbol{U_n = (2U_0 - U_1) + (U_1 - U_0)2^n}$, which **blows up to** ∞ as $n \to +\infty$ if $U_1 \neq U_0$

- Root condition
  - A linear multistep method is **zero-stable if and only if** $\begin{cases} \boldsymbol{|\xi| \leq 1} & \boldsymbol{\forall \text{single root } \xi \text{ of } \rho} \\ \boldsymbol{|\xi^*| < 1} & \boldsymbol{\forall \text{repeated root } \xi^* \text{ of } \rho} \end{cases}$
- Example: $U_{n+2} - 2U_{n+1} + U_n = h\left[\frac{1}{2}f(U_{n+2}) - \frac{1}{2}f(U_n)\right]$
  - $\begin{cases} \rho(\xi) = \xi^2 - 2\xi + 1 \\ \sigma(\xi) = \frac{1}{2}\xi^2 - \frac{1}{2} \end{cases} \Rightarrow \begin{cases} \rho(1) = 0 \\ \rho'(1) = \sigma(1) = 0 \end{cases} \Rightarrow$ This method is consistent
  - Since $\xi_1 = \xi_2 = 1$ is a **double root** of ρ, this method is **not zero-stable**
  - In particular, the analytical solution is $\boldsymbol{U_n = U_0 + (U_1 - U_0)n}$, which blows up for $U_1 \neq U_0$
- Dahlquist Equivalence Theorem
  - A multistep method is **convergent** if and only if it is **consistent and zero-stable**

Absolute Stability

- Motivation
  - So far we only considered the convergence of method **as the grid is refined** ($\boldsymbol{h \to 0}$)
  - *e.g.* Trapezoidal method is a second-order method ⇔ As $h \to 0, E_n \to 0$ at h^2 rate
  - In practice, we want to choose the time step h as large as possible to reduce the computation
  - Absolute stability is used to answer **how big $\boldsymbol{h}$ can be** to produce reasonable results
- Motivating example
  - We want to solve $\begin{cases} u'(t) = \lambda(u - \cos t) - \sin t \\ u_0 = 1.0001 \end{cases}$, where $\lambda = -2100$
  - Analytical solution
    - $u' = \lambda(u - \cos t) - \sin t \Rightarrow \underbrace{u' + \sin t}_{v'} = \lambda \underbrace{(u - \cos t)}_{v}$
    - Let $v(t) = u(t) - \cos t$, then $\begin{cases} v' = \lambda v \\ v_0 = 0.0001 \end{cases} \Rightarrow v(t) = v_0 e^{\lambda t} \Rightarrow u(t) = \cos t + v_0 e^{\lambda t}$
    - Since λ is very negative, $u(t)$ goes to zero quickly

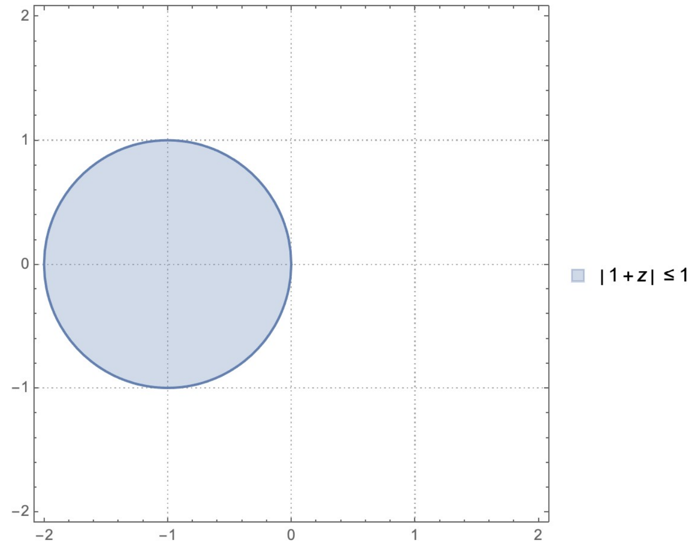


- Numerical method
  - Using forward Euler method with different step size h, we have

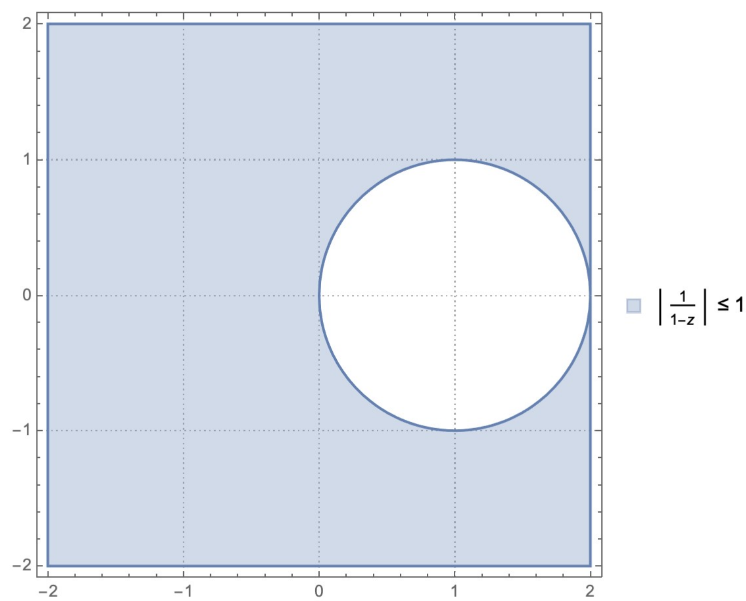
h	Error at $T = 2$
0.000400	0.396033×10^{-7}
0.000800	0.792298×10^{-7}
0.000950	0.321089×10^{-6}
0.000976	0.588105×10^{36}
0.001000	0.145252×10^{77}

  - The error increases dramatically when we change h from 0.00095 to 0.000976
  - Recall the error for forward Euler method, $E_n = (1 + \lambda h)E_{n-1} + h\tau_n$
  - If h cannot balance λ ($i.e.\, |1 + \lambda h| > 1$),the error would be propagated
  - For $h = 0.00095$, $|1 + \lambda h| = |-0.995| \leq 1$
  - For $h = 0.000976$, $|1 + \lambda h| = |-1.0496| > 1$

- Absolute stability (informal)
  - Error introduced at each step does not grow in future steps
- Linear test problem
  - The **test problem for absolute stability** is $\boldsymbol{u' = \lambda u}$ with $\lambda \in \mathbb{C}$
  - When $\text{Re}(\lambda) < 0$, the exact solution $u = u_0 e^{\lambda t} \to 0$ as $t \to 0$, so we **want $\boldsymbol{U_n \to 0}$** as well
  - Apply a numerical method to it, we will obtain $\boldsymbol{U_{n+1} = \phi(\lambda h) U_n}$ for some function ϕ
  - Here $\boldsymbol{\phi(z)}$ is called the **stability function** for the secheme
  - A method is said to be **absolutely stable** for a given step size h if $\boldsymbol{|\phi(z)| \leq 1}$
  - The region of absolute stable (or simply the **stability region**) is $\boldsymbol{\{z \,|\, |\phi(z)| \leq 1\}}$
  - *i.e.* The set of all $z \in \mathbb{C}$ for which the method is absolutely stable
- Example: forward Euler method
  - Apply the forward Euler method to $u' = \lambda u$, we have $\dfrac{U_{n+1} - U_n}{h} = \lambda U_n \Rightarrow U_{n+1} = \underbrace{(1 + \lambda h)}_{\phi(\lambda h)} U_n$
  - The **stability function is $\boldsymbol{\phi(z) = 1 + z}$**, and the **stability region is $\boldsymbol{\{z \,|\, |1 + z| \leq 1\}}$**
  - For $\lambda \in \mathbb{R}$, we **require $\boldsymbol{|1 + \lambda h| \leq 1 \Rightarrow h \leq \dfrac{2}{\lambda}}$** for forward Euler method to be absolutely stable



- Example: backward Euler method
  - Apply the forward Euler method to $u' = \lambda u$, we have $\dfrac{U_{n+1} - U_n}{h} = \lambda U_{n+1} \Rightarrow U_{n+1} = \dfrac{1}{1 - \lambda h} U_n$
  - The stability function is $\phi(z) = \dfrac{1}{1 - z}$, and the stability region is $\left\{ z \middle| \left| \dfrac{1}{1 - z} \right| \le 1 \right\}$



Review for Numerical ODE

Monday, December 10, 2018 9:58 AM

Reducing Higher Order Non-Autonomous ODE

- For $u'' = 2u'u - 3u'$, let $\begin{cases} y_0 = u \\ y_1 = u' \end{cases}$, then $\begin{cases} y_0' = y_1 \\ y_1' = 2y_0y_1 - 3y_1 \end{cases}$

Existence and Uniqueness of Solution

- Unique solution $\Leftrightarrow \vec{f}(\vec{u})$ is uniform Lipschitz $\Leftrightarrow$ Norm of $J_{\vec{f}}(\vec{u}) = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_1} \end{bmatrix}$ is bounded

Common Schemes

- Forward Euler : $\frac{U_{n+1} - U_n}{h} = f(U_n)$
- Trapezoidal: $\frac{U_{n+1} - U_n}{h} = \frac{1}{2}\left(f(U_n) + f(U_{n+1})\right)$
- Explicit midpoint: $\frac{U_{n+1} - U_n}{h} = f\left(U_{n+1/2}\right)$, where $U_{n+1/2} = U_n + \frac{h}{2}f(U_n)$
- Implicit midpoint: $\frac{U_{n+1} - U_n}{h} = f\left(\frac{U_n + U_{n+1}}{2}\right)$

Three Concepts in Numerical ODE

- Consistency
  - If $\tau_n \to 0$ as $h \to 0$, then we say the method is consistent
  - Forward Euler : $\tau_n = \frac{u_{n+1} - u_n}{h} - f(u_n) = \mathcal{O}(h)$
- Convergence
  - The error $E_n = |U_n - u_n| \to 0$ as $h \to 0$
  - We did this for FE and Trapezoidal
- Stability (zero stability and absolute stability)

Runge-Kutta

- We want to march from U_n to U_{n+1} with some stages in between
- RK-4
  - $$\begin{array}{c|cccc} 0 & 0 & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \Rightarrow \begin{cases} y_1 = U_n \\ y_2 = U_n + \frac{1}{2}hf(y_1) \\ y_3 = U_n + \frac{1}{2}hf(y_2) \\ y_4 = U_n + hf(y_3) \\ U_{n+1} = U_n + h\left[\frac{f(y_1)}{6} + \frac{f(y_2)}{3} + \frac{f(y_3)}{3} + \frac{f(y_4)}{6}\right] \end{cases}$$

- RK-2
  - $\begin{array}{c|cc} 0 & 0 & \\ \alpha & \alpha & \\ \hline & \beta & 1-\beta \end{array} \Rightarrow \begin{cases} y_1 = U_n \\ y_2 = U_n + \alpha h f(y_1) \\ U_{n+1} = U_n + h[\beta f(y_1) + (1-\beta) f(y_2)] \end{cases}$, for $\boldsymbol{\alpha(1-\beta) = \frac{1}{2}}$
  - Suppose $\alpha = \frac{1}{2}, \beta = 0 \Rightarrow \begin{cases} y_0 = U_n \\ y_1 = U_n + \frac{h}{2} f(U_n) \\ U_{n+1} = U_n + hf(y_1) = U_n + hf\left(U_n + \frac{h}{2} f(U_n)\right) \end{cases}$
  - $\tau_n = \frac{u_{n+1} - u_n}{h} - f\left(u_n + \frac{h}{2} f(u_n)\right)$

    $= \frac{1}{h}\left[u_n + u_n' h + \frac{u_n'' h^2}{2} + \mathcal{O}(h^3) - u_n\right] - \left[f(u_n) + f'(u_n)\frac{h}{2} f(u_m) + \mathcal{O}(h^2)\right] = \mathcal{O}(h^2)$

LMM and Zero Stability

- A general LLM has the form $\sum_{i=0}^{r} \alpha_i U_{n+i} = h \sum_{i=0}^{r} \beta_i f(U_{n+i})$
- Characteristic polynomials are $\rho(\xi) = \sum_{i=0}^{r} \alpha_i \xi^i$; $\sigma(\xi) = \sum_{i=0}^{r} \beta_i \xi^i$
- For consistency, we need $\begin{cases} \rho(1) = 0 \\ \rho'(1) = \sigma(1) \end{cases}$
- Rood condition for zero stability: $\begin{cases} |\xi_i| \le 1 & \xi_i \text{ is a single root} \\ |\xi_i| < 1 & \xi_i \text{ is a repeated root} \end{cases}$
- Example
  - $U_{n+1} = U_{n+1} + \frac{h}{2}\big(3f(U_{n+1}) - f(U_n)\big) \Rightarrow \begin{cases} \rho(\xi) = \xi^2 - \xi \\ \sigma(\xi) = \frac{3}{2}\xi - \frac{1}{2} \end{cases}$
  - $\begin{cases} \rho(1) = 0 \\ \rho'(1) = \sigma(1) \end{cases} \Rightarrow$ consistent; $\begin{cases} \xi_1 = 1 \\ \xi_2 = 0 \end{cases} \Rightarrow$ zero stable

Absolute Stability

- Test problem: $u' = \lambda u$ for some $\mathrm{Re}(\lambda) < 0$
- Look for the range of h, so that the numeriacl solution decays
- Forward Euler
  - $\frac{U_{n+1} - U_n}{h} = f(U_n) = \lambda U_n \Rightarrow U_{n+1} = (1 + \lambda h) U_n \Rightarrow$ stable region $\{z \,||1 + z| \le 1\}$
- RK2
  - $\begin{cases} y_1 = U_n \\ y_2 = U_n + \alpha h f(y_1) \\ U_{n+1} = U_n + h[\beta f(y_1) + \gamma f(y_2)] \end{cases} \Rightarrow \begin{cases} y_1 = (1 + \alpha\lambda h) U_m \\ U_{n+1} = \big(1 + (\beta + \gamma)\lambda h + \gamma \lambda^2 h^2\big) U_n \end{cases}$
  - Absolute stable region for RK2 is $\{z \,\big||1 + (\beta + \gamma) z + \gamma z^2\big| \le 1\}$