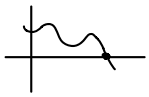
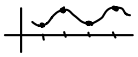
9/1

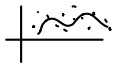
Outline

1) Nonlinear equations 

2) Machine Arithmetic (floating point)

3) Linear Systems ($A\vec{x} = \vec{b}$)

4) Interpolation 

5) Approximation  (least Squares)

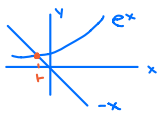
6) Differentiation / Integration

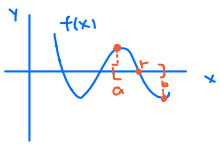
7) ODEs

Root Finding : Goal to find $r \in \mathbb{R}$ s.t $f(r) = 0$

ex. solve $e^x = -x$

Rewrite $g(x) := e^x + x = 0$



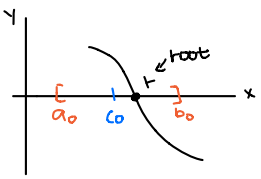
ex. 

Intermediate value Thm (IVT)

let $f \in C[a,b]$ (i.e. f is continuous on $[a,b]$)

If $f(a)f(b) < 0$, then $\exists r \in (a,b)$ s.t $f(r) = 0$.

Bisection Method



Know $r \in (a_0, b_0)$ by IVT

Estimate $r \approx c_0 = (a_0 + b_0)/2$

Error : $e_0 = |r - c_0| \leq (a_0 + b_0)/2$

① If $f(c_0) = 0$, then $r = c_0$.

② If $f(a_0)f(c_0) > 0$, then $r \in (c_0, b_0)$

set $[a_1, b_1] = [c_0, b_0]$

③ If $f(a_0)f(c_0) < 0$, then $r \in (a_0, c_0)$

by IVT and set $[a_1, b_1] = [a_0, c_0]$

choose midpoint $c_1 = (a_1 + b_1)/2$ and repeat this entire process

Error Estimate :

$e_0 = |r - c_0| \leq (b_0 - a_0)/2$

$e_1 = |r - c_1| \leq (b_1 - a_1)/2 \leq (b_0 - a_0)/2^2$

$\vdots$

$e_n = |r - c_n| \leq \cdots \leq (b_0 - a_0)/2^{n+1}$

So $e_n \leq (b_0 - a_0)/2^{n+1}$ and $r \approx c_n = (a_n + b_n)/2$

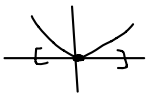
Remarks :

① With $e_n \leq (b_0 - a_0)/2^{n+1} < TOL$ tolerance (10^{-b}) can solve for n to ensure tolerance is met.

$$n > \frac{\ln\left(\frac{a_0 - b_0}{TOL}\right)}{\ln(2)} - 1$$

② Method always converges if you have a starting bracket.

③ $f(x)$ plays no role in computing n for TOL satisfaction (for error)

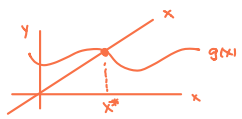
④  double root, no bracket possible.
method can fail!

⑤ Convergence in slow:
Have $e_{n+1} \approx \frac{1}{2} e_n$ (linear convergence)

Fixed Point Iteration (FPI)

Consider solving $\cos(x) = 0$, can use bisection...

OR. $g(x) := \cos(x) + x = x$. can instead solve $g(x) = x$ A point x^* s.t. $g(x^*) = x^*$ is called a fixed point of g.



9/3

Iterative Method: Try to solve $g(x) = x$ using

$$\begin{cases} x_{i+1} = g(x_i) \\ x_0 = \text{starting value} \end{cases}$$

ex: 1) $\begin{cases} x_{i+1} = x_i(\frac{1}{10}) + 1 \\ x_0 = 0 \end{cases}$ $[\, g(x) = 1 + \frac{x}{10} \,]$

$x_1 = 1$, $x_2 = 1.1$, $x_3 = 1.11$ .... $x^* = 1.1111\ldots = \frac{10}{9}$

Have $\frac{10}{9} = g(\frac{10}{9})$

2) $\begin{cases} x_{i+1} = 3x_i + 1 \\ x_0 = 0 \end{cases}$

$x_1 = 1$, $x_2 = 4$, $x_3 = 13$, $x_4 = 40$ .... FPI blows up!

3) $\begin{cases} \vec{x}_{i+1} = A\vec{x}_i \\ \vec{x}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \mathbb{R}^2 \end{cases}$

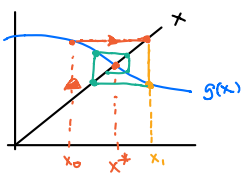
4) $\begin{cases} y' = f(t,y) \\ y(0) = y_0 \end{cases}$ $\Leftrightarrow$ $\begin{cases} y_{i+1}(t) = y_0 + \int_0^t f(s, y_i(s))\, ds \\ y_0(t) = 0 \end{cases}$

5) $\begin{cases} x_{i+1} = x_i + \cos x_i \\ x_0 = 0 \end{cases}$ $x_1 = 1$, $x_2 = 1.5403\ldots$

$x_3 = 1.57079\ldots$ correct up to 5 decimal places

Geometric Interpretation.

$x_{i+1} = g(x_i)$



cobweb diagrams

Convergence of FPI

existence of fixed points in Levy's note.

contraction mappings.

Let $g \in [a,b]$, g is a contraction on $[a,b]$

if $\exists L$, $0 \le L < 1$, s.t. $|g(x) - g(y)| \le L|x-y| \quad \forall x,y \in [a,b]$

contraction Mapping THM

Let $g : [a,b] \to [a,b]$ be a contraction.

Then 1) $\exists$ unique fixed point x^* of g. $g(x^*) = x^*$

2) FPI converges to x^* starting from any $x_0 \in [a,b]$

FPI: $x_{i+1} = g(x_i)$

Remarks:

1) If $g \in C^1(a,b)$, g is a contraction provided $|g'(x)| \le L \le 1$ for $\forall x \in (a,b)$

$$|g(x) - g(y)| = |g'(\eta)(x-y)| \le |g'(\eta)||x-y| \le L|x-y|$$

(η: by mean-value theorem)

2) If fixed point x^* is known, can check $|g'(x^*)| < 1$ for g to be a contraction on a small interval around x^*

(have contraction locally around x^*)

3) To stop iteration, need stopping criteria.

(i) Run FPI $N < \infty$ times,

(ii) Or stop when $|x_{i+1} - x_i| < TOL$ (absolute error)

ex: $\cos x = 0$

$g(x) = \cos x + x = x$

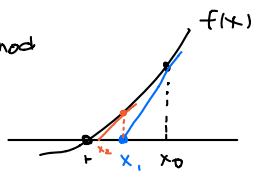
Have $g'(x) = -\sin x + 1$, At least on $[\frac{\pi}{4}, \frac{3\pi}{4}]$ have $|g'(x)| \le L < 1$

For $x_0 \in [\frac{\pi}{4}, \frac{3\pi}{4}]$, $x_{i+1} = g(x_i)$ will converge.

Can also note $x^* = \frac{\pi}{2}$ is a root / fixed point

and $|g'(\frac{\pi}{2})| = 0 < 1$ so FPI converges locally near $x^* = \frac{\pi}{2}$

Newton's Method



Taylor Expansion: $f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2}f''(x_0)(x-x_0)^2 + O(x-x_0)^3$

$L(x)$ (Linear)

$f(x) \approx L(x) = f(x_0) + f'(x_0)(x-x_0)$

$\Rightarrow \quad 0 = L(x_1) = f(x_0) + f'(x_0)(x_1 - x_0)$

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

Method: $x_{n+1} = x_n - f(x_n)/f'(x_n)$

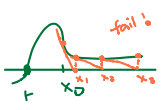
x_0 = starting guess

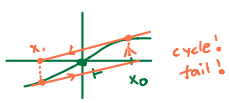
Rmk: Can approximate $f'(x) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$ (secant method)

Remarks: 1) Newton is an FPI, with $g(x) = x - f(x)/f'(x)$

$$g'(r) = 1 - \frac{f(r)f'(r) - f(r)f''(r)}{(f'(r))^2} = 0 < 1$$

2) Newton can fail!





9/8 Error Analysis + Convergence

$f(r)=0$. want root r

let $e_n = |x_n - r|$

consider Newton's method : $x_{n+1} = \underbrace{x_n - f(x_n)/f'(x_n)}_{=: g(x_n)}$

Remark . $f(x) = f(x_0) + (x-x_0)f'(x_0) + \frac{1}{2}(x-x_0)^2 f''(x_0) + \frac{1}{6}(x-x_0)^3 f'''(c)$ ← b/t x, x_0

Taylor expand :

$f(r) = f(x_n) + (r-x_n)f'(x_n) + \frac{1}{2}(r-x_n)^2 f''(\xi_n)$ → is between x_n and r

root, so $f(r)=0$

$-f(x_n)/f'(x_n) = (r-x_n) + \frac{1}{2}(r-x_n)^2 f''(\xi_n)/f'(x_n)$

$x_n - f(x_n)/f'(x_n) = r + \frac{1}{2}(r-x_n)^2 f''(\xi_n)/f'(x_n)$

$= x_{n+1}$

$x_{n+1} - r = \frac{1}{2}(r-x_n)^2 f''(\xi_n)/f'(x_n)$

OR $e_{n+1} = \frac{1}{2} e_n^2 \left|f''(\xi_n)/f'(x_n)\right|$ $e_n = |x_n - r|$

this roughly says $e_{n+1} \approx C e_n^2$ (quadratic convergence)

$\lim_{n\to\infty} \frac{e_{n+1}}{e_n^2} = \lim_{n\to\infty} \frac{1}{2}\left|\frac{f''(\xi_n)}{f'(x_n)}\right| = \frac{1}{2}\left|\frac{f''(r)}{f'(r)}\right|$

↑ this provided 1) have convergence
2) $f \in C^2$ near r
3) $f' \neq 0$

Definition :

Let $e_n = |x_n - r|$ be the error of an iterative method.

If $\lim_{n\to\infty} \frac{e_{n+1}}{e_n^p} = s < \infty$ $(p \geq 1)$

we say the convergence is of order p.

(can think $e_{n+1} \approx s e_n^p$)

- $p=2$ (quadratic convergence)
  Ex: Newton's method (provided $f' \neq 0$ near the root)

- $p=1$ (linear convergence)
  Ex: Bisection method is linearly convergent $e_{n+1} \approx \frac{1}{2} e_n^1$ ($s = \frac{1}{2}$)
  note: we need $s<1$ for linear convergence.
  otherwise may get blow up

Notions of Error and Accuracy.

Want r s.t. $f(r)=0$ r = real root

r_{approx} = approximate root

Have $r \approx r_{approx}$ and $f(r_{approx}) = \varepsilon << 1$

Definition :

forward error := $|r - r_{approx}|$ need to know r beforehand.

backward error := $|\varepsilon| = |f(r_{approx})|$

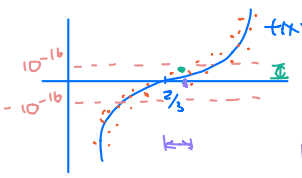
Motivation : data $[f(x)]$ → Eqn solver → solution $[r]$

" backward error " " forward error "

Ex. $f(x) = (x - \frac{2}{3})^3 = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27} = 0$

Actual root $= r = 2/3$

lots of machine (floating point) numbers nearby

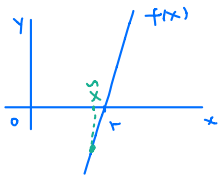


point ↓ • will have zero backward error, and is a "root".

as far as computer is concerned.

but has large forward error and doesn't solve our problem.

Ex:



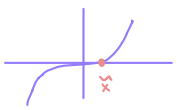
↔ forward error small $|r - \hat{x}| << 1$

↕ backward error $|f(\hat{x})| >> 1$ large

Ex. $f(x) = 0$, $f(x) = x^{15}$. Suppose $r = 0$. $r_{approx} = 10^{-1}$

bkwd. error $= (10^{-1})^{15} = 10^{-15} << 1$

fwd. error $= |0 - 10^{-1}| = 0.1$ which is fairly large



Convergence of FPI: $x_{n+1} = g(x_n)$

Have local convergence by contraction mapping if $|g'(r)| \le L < 1$

By Mean-value theorem, $\exists\ \xi_n$ between x_n and r.

s.t. $\underbrace{g(x_n)}_{=x_{n+1}} - \underbrace{g(r)}_{=r} = g'(\xi_n)(x_n - r)$

$\Rightarrow \frac{x_{n+1} - r}{x_n - r} = g'(\xi_n) \Rightarrow \lim_{n\to\infty} \frac{e_{n+1}}{e_n} = |g'(r)|$

So FPI is (at least) linearly convergent, provided $|g'(r)| < 1$

Remark: If $g'(r) = 0$ (this is the case for Newton's method)

then $g(x_n) - g(r) = \cancel{g'(r)}^{0}(x_n - r) + \frac{1}{2}g''(\xi_n)(x_n - r)^2$

then $\lim_{n\to\infty} \frac{e_{n+1}}{e_n^2} = \frac{1}{2}|g''(r)|$ (quadratic convergence)

9/10

Binary Numbers

$b_i = 0$ or 1 (bits)

$\ldots b_2 b_1 b_0 \underset{\text{radix}}{\bullet} b_{-1} b_{-2} b_{-3} \ldots = \ldots b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0 + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} \ldots$

ex: (Terminating Binary)

$(100.1)_2 = 0 \times 2^0 + 0 \times 2^1 + \underbrace{1 \times 2^2}_{4} + \underbrace{1 \times 2^{-1}}_{0.5} = (4.5)_{10}$

↑ denotes binary

ex: (Nonterminating)

$(0.\overline{10})_2 = (0.10101010\ldots)_2$

$x = 0.\overline{10}$

$2^2 x = 10.\overline{10}$ ← multiplication by 2 shifts radix

$(2^2 - 1)x = (10)_2$

$= (2)_{10}$

$x = 0.\overline{9}$

$10x = 9.\overline{9}$

subtract $9x = 9$

$x = 1$

$\Rightarrow \quad x = \frac{2}{3}$ $\qquad 0.\overline{97} \times 100 = 97.\overline{97}$

How to convert decimal to binary?

ex. $(13.6)_{10} = (13)_{10} + (0.6)_{10}$

integer part: $\frac{13}{2} = 6$ R1; $\frac{6}{2} = 3$ R0; $\frac{3}{2} = 1$ R1; $\frac{1}{2} = 0$ R1 (stop when zero)

read backward

$(1101)_2 = (13)_{10}$

note $1 + 2 + 8 = 13$

fractional part:

$0.6 \times 2 = 1.2 = 0.2 + 1$

$0.2 \times 2 = 0.4 = 0.4 + 0$

$0.4 \times 2 = 0.8 = 0.8 + 0$

$0.8 \times 2 = 1.6 = 0.6 + 1$

$0.6 \times 2 = 1.2 = 0.2 + 1$ starts to repeat

read the integer part forward.

$(0.\overline{1001})_2 = (0.6)_{10}$

SO $(13.6)_{10} = (1101.\overline{1001})_2$

Floating point numbers:

- finite precision arithmetic
- IEEE-754 standard (1985 - Institute of electrical and Electronic Engineers)

Normalized (base 2) floating point.

$[\pm] 1.[\text{mantissa}] \times 2^{[\text{exponent}]}$

sign

mantissa: string of bits (0 or 1)

leading 1 not stored (it is implied)

this is the normalized part

ex. $(13.6)_{10} = (1101.1)_2 = +1.1011 \times 2^3$

Levels of position:

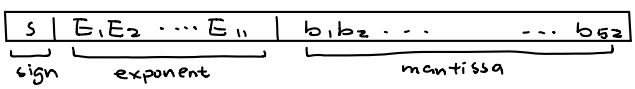
precision	sign	exponent	mantissa	Total bits
single	1	8	23	32
double	1	11	52	64

$x \in \mathbb{R} \longrightarrow fl(x)$ (floating point) $\longrightarrow$ computer word

$\pm 1.\square \times 2^{\square}$

how floats are stored on a computer

Double precision word:



(1 bit) (11 bits) (52 bits)

$s = 0$ if positive

$s = 1$ if negative

Remark: For the exponent, we don't store the exponent E exactly, but rather we store $E + 1023$ (exponent bias)

Remark : The range of exponents we are allowing is $-1022 \le E \le 1023$ for the float $\pm 1.\square \times 2^E$

Rounding :

ex: $(9.4)_{10} = (1001.\overline{0110})_2$

$= + 1.\boxed{001\ldots\ldots1100}\ \underbrace{110\ldots}\ \times 2^3$

remainder of the mantissa needs to be cut somehow

Option 1 : (chopping)

- throw away the remainder
- bad since always rounding down!

Option 2: IEEE Round to Nearest Rule

Rules: (the point of this rule is to try to cancel out equally likely errors.)

1) $\pm 1.\boxed{b_1 b_2 \cdots b_{52}}\ \underbrace{00\ldots}\ \times 2^E$

If all zeros, then chop bits beyond 52^{nd} bit

2) $\pm 1.\boxed{b_1 b_2 \cdots b_{52}}\ \underbrace{1\ldots\ldots}\ \times 2^E$

If $b_{53} = 1$ and rest of bits are not all zero, then round up by adding 1 to the 52^{nd} bit and carring if necessary.

3) $\pm 1.\boxed{b_1 b_2 \cdots b_{52}}\ \underbrace{1\ldots\ldots}\ \times 2^E$

If $b_{53} = 1$ and rest of bits are all 0's

then { If $b_{52} = 1$, round up by adding 1 to b_{52}
else, if $b_{52} = 0$, round down by truncating the remainder

9/15

Floating Point : Part II

$\pm\ 1.\boxed{b_1 b_2 \cdots b_{52}} \times 2^E$ ← $-1022 \le E < 1023$

$\pm$: sign

1. : leading 1

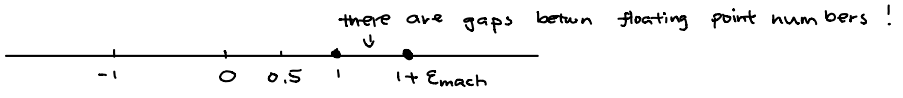
$b_1 b_2 \cdots b_{52}$: mantissa $b_i = 0, 1$

11 bits of precision

$E = E_1 E_2 \cdots E_{11}$

$E_i = 0$ or 1

Machine ε



- Defn: ε_{mach} is the distance between 1 and the next smallest repreientable number bigger than 1.

- In double precision

$\varepsilon_{mach} = 2^{-52} \approx 2.22 \times 10^{-16}$ → can expect accuracy of about 16 decimal places in double precision

- Relative Rounding Error

For IEEE-754 with rounding to Nearest Rule,

$|fl(x) - x| < \frac{1}{2}\varepsilon$

$\frac{|x-fl(x)|}{|x|} \leq \frac{1}{2}\varepsilon_{mach}$

where $x \in \mathbb{R}$, $fl(x)$ = normalized floating point number

$x \in \mathbb{R}$. $fl(x)$ = float form

$fl(1) = +\ 1.\boxed{0\cdots01} \times 2^0$

↑ replace $b_{52} = 1$ to get next highest float

$fl(1+\varepsilon_{mach}) = +\ 1.\boxed{000\cdots01} \times 2^0 = 1 + \underbrace{2^{-52}}_{\varepsilon_{mach}}$

numbers between 1, $1+\varepsilon_{mach}$ get rounded to either 1 or $1+\varepsilon_{mach}$

ex: $fl(x) \neq x$ for $x = 9.4$

$(9.4)_{10} = (1001.\overline{0110})_2$ remainder R to cut off

$fl(9.4) = +\ 1.\boxed{0010\ 1100\ \cdots\ 1100}\ \underbrace{1100}_{R} \times 2^3$

(last) By truncating, we lose $R = (0.\overline{1100})_2 \times 2^{-52} \times 2^3 = (0.\overline{1100})_2 \times 2^{-49} = 0.8 \times 2^{-49}$ in base 10

(add) By rounding to nearest rule 2), we add 1 to the b_{52} bit. to get an addition of $2^{-52} \times 2^3 = 2^{-49}$

So $fl(9.4) = 9.4 + 2^{-49} - (0.8 \times 2^{-49}) = 9.4 + \underbrace{0.2 \times 10^{-49}}$ rounding error

not in float form, just the magnitude.

Remark:

1) 3 rules for arithmetic: $a \oplus b$, $a \ominus b$, $a \otimes b$, $a \oslash b$.
see Michael Overton notes for details.

2) Computations are performed in higher precision, but results are stored in double precision.

3) VPA (varable precisian arithmetic)
>> help (vpa) in Matlab.
>> vpa (pi, 100) want 100 digits

Loss of significance

>> $\frac{1}{1-(1+\varepsilon_{mach}/2)}$

>> -Inf (note $1+\frac{\varepsilon}{2}$ rounded to 1

- the answer should be $-\frac{2}{\varepsilon_{mach}} = -2^{53}$
- the problem here is due to cancellation of nearly equal numbers, called catastrophic cancellation.

ex: $x^2 + 9^{12}x = 3$

solution should be $\frac{-9^{12} \pm \sqrt{(9^{12})^2 - 4(1)(-3)}}{2(1)}$

$x_\pm = \frac{-9^{12} \pm \sqrt{9^{24}+12}}{2}$ in Matlab $x_1 = 0$ (+) ↓ $x_2 \approx -2.82 \times 10^{11}$ (−)

$x_2^2 + 9^{12}x_2 - 3 = -3$ (matlab)

using matlab $x_+ = 0$. definitely not a solution

$x_- \approx -2.824 \times 10^{11}$

Two options to get positive root x_+?

option 1: use VPA or symbolic arithmetic

option 2: stay in double precision but modify computation.

$ax^2+bx+c=0$ $x_+ = \frac{-b+\sqrt{b^2-4ac}}{2a}\left(\frac{b+\sqrt{b^2-4ac}}{b+\sqrt{b^2-4ac}}\right) = \frac{-2c}{b+\sqrt{b^2-4ac}}$ (no more catastrophs cancellation

$\frac{-b + \sqrt{b^2-4ac}}{2a}$ $\left(\frac{2c}{b + \sqrt{b^2-4ac}}\right)$ $\frac{2c}{b+\sqrt{b^2-4ac}}$ if $b \gg 1$

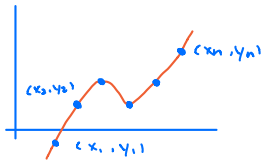
By hand $x_+ = \frac{6}{q^{12} + \sqrt{q^{24}+12}} = 1.0622 \times 10^{-11}$

$x_+^2 + q^{12} x_+ - 3 = 0$

9/17

System of Equations

ex:



suppose want to fit a polynomial

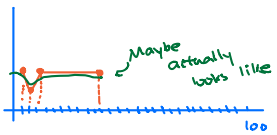
$p(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} \in \mathbb{P}_{n-1}$

n unknown coefficients

Get $p(x_1) = y_1$, $p(x_2) = y_2$, $\vdots$, $p(x_n) = y_n$ } system of eqns for n unknowns. $\{a_i\}_{i=0}^{n-1}$

ex: $\begin{cases} -u''(x) = f(x) & [0 \le x \le 100] \\ u(0) = u_0, \quad u'(0) = u_1 \end{cases}$

IVP



After discretizing, get linear system $A\vec{x} = \vec{b}$ to solve.

usually very large.

How to solve $A\vec{x} = \vec{b}$?

$A = A_{n\times n} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$ $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$

$(A^{-1})_{n\times n}$ is the matrix s.t $AA^{-1} = A^{-1}A = I_{n\times n} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1_n \end{pmatrix}_{n\times n}$

Have $Ax = b$

$A^{-1}Ax = A^{-1}b$

$x = A^{-1}b$ taking matrix inverse is computationally bad.

Componentwise Computations.

① $\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i =: x_i y_i$ called Einstein summation convention.

② $A\vec{x} = \vec{b}$

$(\vec{b})_i = (A\vec{x})_i = \sum_{j=1}^{n} a_{ij} x_j = a_{ij} x_j$ two repeated indices imply a summation

③ $(AB)_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{ik} b_{kj}$

Big O Notation

Def: $f(n) = O(g(n))$ as $n \to \infty$ if $\exists M, N$ s.t $|f(n)| \le M|g(n)| \quad \forall n \ge N$

ex: $n^3 + 2n^2 - n = O(n^3) = n^3 + O(n^2)$

Apply triangle inequality $|n^3 + 2n^2 - n| \le n^3 + 2n^2 + n \le 4n^3$ for $M=4$ $N=1$

$n^2 \le n^3$, $n \le n^3$, provided $n \ge 1$

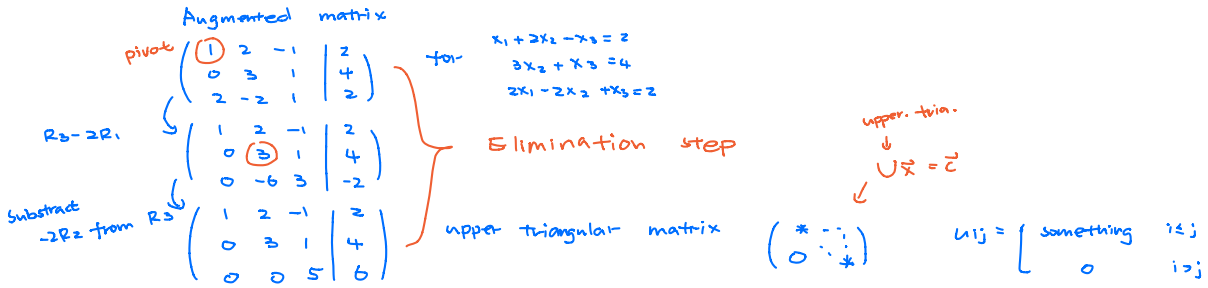
ex: $5n^4 + n^3 + 1 = O(6n^4) = O(n^4) = 5n^4 + O(n^3)$

suppose $n = 10^6$

then $5n^4 = 5 \cdot 10^{24}$ ← dominating term

$n^3+1 = 10^{18}+1 = O(n^3)$

(Naive) Gaussian Elimination ($A\vec{x}=\vec{b}$)



Back solve step.

$x_1+2x_2-x_3=2$ $\quad x_1 = 2-2x_2+x_3$

$3x_2+x_3=4$ $\quad x_2 = 1/3(4-x_3)$

$5x_3=6$ $\quad x_3 = 6/5$

(Naive) Gaussian Elimination ($A\vec{x}=\vec{b}$)

Operation counts: $A = A_{n\times n}$

① Elimination step (to get $U\vec{x}=\vec{c}$)

Requires $\frac{n^3}{3}+\frac{n^2}{2}-\frac{5n}{6}$ multiplications / divisions.

$= O(\frac{n^3}{3})$ (with a similar number of additions/subtractions)

+, −, /, · counts are called flops.

ex: For $A_{100\times 100}$, need $\approx \frac{1}{3}100^3 = \frac{1}{3}10^6$ multiplications / divisions

② Backsolve step (to get $\vec{x}$)

Requires $\frac{n^2}{2}+\frac{n}{2}$ multiplications / divisions.

$= O(\frac{n^2}{2})$

ex: For $A_{100\times 100}$, need $\approx \frac{100^2}{2} = \frac{1}{2}10^4$ flops

Remarks: ① Elimination is much more expensive

② Operation counts can be used to predict computation times.

ex: if know $A_{10\times 10}$ takes 10^{-2} seconds to solve. can use operation count to give an estimate for $A_{10000000\times 10000000}$

③ Counts can be used to compare algorithms.

Operation count example:

$(A\vec{x})_i = \sum_{j=1}^{n} a_{ij}x_j \qquad (1 \le i \le n)$

For just the i^{th} entry, need n multiplications, one for each $1 \le j \le n$

$n-1$ additions, since adding n terms

$= n+(n-1) = 2n-1$ flops

For entire vector $A\vec{x}$, need $n(2n-1) = 2n^2-n = O(n^2)$ flops

ex: $(AB)_{ij} = \sum_{k=1}^{n} A_{ik}B_{kj} \qquad AB \in \mathbb{R}^{n\times n}$

$AB = A[\vec{b}_1, \vec{b}_2, \dots \vec{b}_n]$

Using previous result from matrix-vector product, we need $n(2n^2-n) = 2n^3-n^2 = O(n^3)$ flops.

9/22

LU Decomposion (matrix form of elimination)

Goal: $A = LU = \begin{pmatrix} 1 & 0 \\ * & 1 \end{pmatrix}\begin{pmatrix} 1 & * \\ 0 & 1 \end{pmatrix}$ = Lower triangular × Upper triangular

- If 1's on main diagonal of L this is Doolittle decomposition.
- Idea: Getting LU encodes the elimination step of Gaussian elimination (this is $O(n^3)$)

If $Ax = b$, then $LUx = b$

① solve $Ly = b$ for y by forward substitution
② solve $Ux = y$ for x by back substitution
] both are $O(n^2)$ in complexity

Remark: Don't solve ①, ② by inverting L or U.

ex: $A = \begin{pmatrix} 1 & 2 & -1 \\ 0 & 3 & 1 \\ 2 & -2 & 1 \end{pmatrix}$ = subtract $2R_1$ from R_3, $0R_1$ from R_2 $\begin{pmatrix} 1 & 2 & -1 \\ (0) & 3 & 1 \\ (2) & -6 & 3 \end{pmatrix}$ ← storing multipliers for convenience, these will form L, there is really zeros in these slots. Subtract $-2R_2$ from R_3 $\begin{pmatrix} 1 & 2 & -1 \\ (0) & 3 & 1 \\ (2) & (-2) & 5 \end{pmatrix}$

claim: $A = LU = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & -2 & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 & -1 \\ 0 & 3 & 1 \\ 0 & 0 & 5 \end{pmatrix}$ Doolittle since 1's on main diagonal of L

in Matlab.

```
>> [L,U] = lu(A);
>> [L,U,P] = lu(A);
```
↑ pivot matrix

Remark: Suppose want to solve $Ax_1 = b_1$, $Ax_2 = b_2$, ⋮, $Ax_r = b_r$ with same A each time.

If separately solving, cost is $O(rn^3 + rn^2) \sim O(n^4)$ if $r \approx n$

But if eliminating once (via LU decomposition) then cost is $O(n^3 + rn^2) \sim O(n^3)$ if $r \approx n$.

⇒ cheaper.

Sources of error in Gaussian elimination (LU)

① $A = \begin{pmatrix} \boxed{0}^{\text{pivot}} & 1 \\ 1 & 0 \end{pmatrix}$ Have 0 pivot and can't eliminate A_{21} entry.

solution is to swap rows (will get us a $PA = LU$ decomposition)
↳ permutation matrix

② swamping: suppose $\varepsilon < \varepsilon_{mach} \approx 10^{-16}$ (say $\varepsilon = 10^{-20}$)

$A = \begin{pmatrix} \boxed{\varepsilon}^{\text{pivot}} & 1 \\ 1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{pmatrix}}_{L}\underbrace{\begin{pmatrix} \varepsilon & 1 \\ 0 & \boxed{1 - 1/\varepsilon} \end{pmatrix}}_{U}$ ← problem term, in double precision, stored as $-\frac{1}{\varepsilon}$

stored as → $\begin{pmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{pmatrix}\begin{pmatrix} \varepsilon & 1 \\ 0 & -1/\varepsilon \end{pmatrix} = \begin{pmatrix} \varepsilon & 1 \\ 1 & 0 \end{pmatrix} \neq A$

$\tilde{L}$ $\tilde{U}$

- So finite precision can lead to catastrophic cancellation (called swamping)
- solution again is to swap rows.

$\begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 0 & \boxed{1-\varepsilon} \end{pmatrix}$ ← stored as 1 on computer

$= \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \varepsilon & \boxed{\varepsilon+1} \end{pmatrix}$ stored as $\begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix}$

$PA = LU$ Decomposition

big pivot to avoid swamping

$A = \begin{pmatrix} \boxed{\varepsilon} & 1 \\ 1 & 1 \end{pmatrix} \xrightarrow[R_1 \text{ with } R_2]{\text{switch}} \begin{pmatrix} \boxed{1} & 1 \\ \varepsilon & 1 \end{pmatrix}$

search 1st column for larger entry and swapping that row to the top (called partial pivoting by rows)

In matrix form, use $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ permutation matrix (an nxn matrix with a single 1 in each column / row)

to get $PA = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix}$ row swapped

suppose have $PA = LU$ to solve $Ax = b$. 1) $PAx = Pb$

2) $LUx = Pb$

$\underbrace{\quad}_{L}$ solve $Lc = Pb$ for c using forward substitution

then solve $Ux = c$ for x by back substitution.

>> x = A \ b ;

choose most optimal method (Matlab \ command)

ex: $\begin{pmatrix} 3 & 7 \\ 6 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -11 \end{pmatrix}$ solve using $PA = LU$

$A = \begin{pmatrix} 3 & 7 \\ 6 & 1 \end{pmatrix} \xrightarrow[P=\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ to swap } R_1, R_2 \text{ of } A]{} \begin{pmatrix} 6 & 1 \\ 3 & 7 \end{pmatrix} \xrightarrow[-\frac{1}{2}R_1 \text{ from } R_2]{\text{subtract}} \begin{pmatrix} 6 & 1 \\ (\frac{1}{2}) & \frac{13}{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{pmatrix}\begin{pmatrix} 6 & 1 \\ 0 & \frac{13}{2} \end{pmatrix} = LU$

Have $PA = LU$ $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}A = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{pmatrix}\begin{pmatrix} 6 & 1 \\ 0 & \frac{13}{2} \end{pmatrix}$

Now ① Solve $Lc = Pb$ for c to get $c = \begin{pmatrix} -11 \\ \frac{13}{2} \end{pmatrix}$

② Solve $Ux = c$ for x to get $\vec{x} = \begin{pmatrix} -7 \\ 1 \end{pmatrix}$

Algorithm for back-substitution

$U\vec{x} = \vec{b}$ / upper trig.

$u_{11}x_1 + u_{12}x_2 + \cdots + u_{1n}x_n = b_1$

$\vdots$

$u_{nn}x_n = b_n$

Have $x_n = \frac{b_n}{u_{nn}}$ in general, $x_i = \frac{1}{u_{ii}}\left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j\right)$

For HW 3, do an operation count to construct $\vec{x} = (x_1, \ldots x_n)$

9/24

Error for Systems $Ax = b$

Error: suppose x_a is an approximate solution

Definition: $r = b - Ax_a$ is the residual vector

backward error $= \|b - Ax_a\|$ (think $|f(x_a)|$ for root finding)

forward error $= \|x - x_a\|$

vector / matrix norms: Suppose $x \in \mathbb{R}^n$

A norm satisfies (i) $\|x\| \geq 0$ and $\|x\| = 0$ iff $x = 0$

(ii) $\|kx\| = |k|\|x\|$ (scaling)

(iii) $\|x+y\| \leq \|x\| + \|y\|$ (triangle inequality)

- can also define matrix norms (for $\mathbb{R}^{n\times m}$)

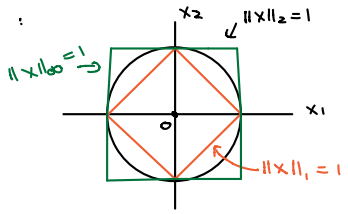
ex: $\|x\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{1/2}$ Euclidean or 2-norm

$\|x\|_1 = \sum_{i=1}^n |x_i|$ taxicab norm

$\|x\|_\infty = \max_{i=1,\dots,n} |x_i|$

there are others, for instance, $\|x\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$

Unit circles:



Matrix Norms: $A \in \mathbb{R}^{n\times n}$, $\|A\|$ (or mxn matrix)

need $\|\cdot\| : \mathbb{R}^{n\times n} \to \mathbb{R}$ to satisfy norm properties

① $\|A\|_F := \left(\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2\right)^{1/2}$ Frobenius norm

② Operator p-norms:

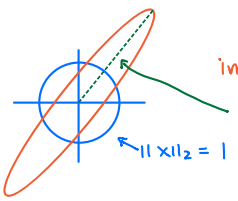
$$\|A\|_p = \max_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p = 1} \|Ax\|_p$$

vector of length 1 ↓

since $\frac{\|Ax\|_p}{\|x\|_p} = \|A\left(\frac{x}{\|x\|_p}\right)\|_p$

- measures maximum stretching of unit sphere $\|x\|_p = 1$ by A
- matrix norm is inherited from vector norm

ex:



Common p-norms:

① $\|A\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^n |a_{ij}|$ = maximum absolute column sum of A

② $\|A\|_\infty = \max_{i=1,\dots,n} \sum_{j=1}^n |a_{ij}|$ = maximum absolute row sum of A

③ $\|A\|_2 = \sqrt{\sigma(A^TA)}$ where $\sigma(B) = \max_i |\lambda_i|$ (the "spectral radius")

λ_i are eigenvalues of B

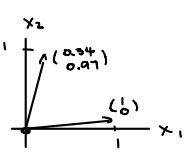
In matlab, would use >> norm(A, "1, 2, inf, ...")

Error Magnification

$A = \begin{pmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{pmatrix}$ $b = A(:,1) = \begin{pmatrix} 4.1 \\ 9.7 \end{pmatrix}$ $x = A\backslash b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

suppose now b is $b_{++} = \begin{pmatrix} 4.11 \\ 9.7 \end{pmatrix}$ ← error of 0.01

then $x = A\backslash b_{++} = \begin{pmatrix} 0.34 \\ 0.97 \end{pmatrix}$



Approximate solution

$Ax = b$. On a computer $A\tilde{x} = \tilde{b}$ ← b stored with error (so is A but we ignore for now)

	bkwd. error	fwd. error
absolute	$\|b - A\tilde{x}\|_p$	$\|x - \tilde{x}\|_p$
relative	$\frac{\|b - A\tilde{x}\|_p}{\|b\|_p}$	$\frac{\|x - \tilde{x}\|_p}{\|x\|_p}$

How does $\frac{\|b - \tilde{b}\|}{\|b\|}$ affect $\frac{\|x - \tilde{x}\|}{\|x\|}$?

error in problem data — error in problem solution

$A(x - \tilde{x}) = \tilde{b} - b$

$\|\tilde{x} - x\| = \|A^{-1}(\tilde{b} - b)\| \leq \|A^{-1}\| \cdot \|\tilde{b} - b\|$

Also have $b = Ax$ and so $\|b\| = \|Ax\| \leq \|A\| \|x\|$

then we have $\frac{\|x - \tilde{x}\|}{\|x\|} \leq \|A\|\|A^{-1}\| \frac{\|\tilde{b} - b\|}{\|b\|}$

$\frac{\|x - \tilde{x}\|}{\|x\|}$: error in soln. $\|A\|\|A^{-1}\|$: this factor is called the condition number of A : $\frac{\|\tilde{b} - b\|}{\|b\|}$: error in data

$\text{cond}_p(A) := \|A\|_p \|A^{-1}\|_p$

Remark: ① $\text{Cond}_p(A)$ is the max. error magnification in solving $Ax = b$

$$\frac{\frac{\|x - \tilde{x}\|}{\|x\|}}{\frac{\|\tilde{b} - b\|}{\|b\|}} \leq \text{cond}_p(A)$$

② If cond(A) is "small", this is good. If cond(A) is "large" (10^9, 10^{20}...) is bad.
↳ called these ill-conditioned problems

③ data $\frac{\|b - \tilde{b}\|}{\|b\|}$ error is on order of $\varepsilon_{mach} \approx 10^{-16}$ in double presicion

④ If $\text{cond}_p(A) \sim \frac{1}{\varepsilon_{mach}}$, can expect good accuracy.
If $\text{cond}_p(A) \sim 10^k$, expect to lose k digits of accuracy in computing x.

9/29

Iterative Method for systems

Until now, solve $Ax = b$ via direct method
↳ solve in finite # of steps in extra arithemtic.
(i.e. $PA = LU$)

Can try iteration:

$\underbrace{Ax - b + x}_{g(x)} = x$ with $x_{k+1} = g(x_k)$ it will turn out to not guarantee convergance

Jacobi Method:

Split A as $A = D + R$ (D = diagonal of A, R = everything else)

ex: $\underset{A}{\begin{pmatrix} 2 & 0 \\ -1 & 4 \end{pmatrix}} = \underset{D}{\begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}} + \underset{R}{\begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix}}$

Now $Ax = b \Rightarrow (D+R)x = b \Rightarrow Dx = b - Rx \Rightarrow x = D^{-1}(b - Rx)$

Jacobi Method: $x_{k+1} = D^{-1}(b - Rx_k)$

x_0 = starting guess

$D = \begin{pmatrix} a_1 & & \\ & a_2 & \\ & & \ddots \\ & & & a_n \end{pmatrix}$ $\quad D^{-1} = \begin{pmatrix} a_1^{-1} & & \\ & a_2^{-1} & \\ & & \ddots \\ & & & a_n^{-1} \end{pmatrix}$

Remark : ① Need stopping criteria.

i) Iterate finite (say N) number of times, then stop.

ii) Can use forward or backward "error" : $\|x_{k+1} - x_k\|_p < \text{TOL}$ (10^{-6})

or $\|Ax_k - b\|_p < \text{TOL}$

② Convergence guaranteed if A is strictly diagonally dominant : $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$

ex: $A = \begin{pmatrix} 2 & 0 & 1 \\ -1 & 4 & 1 \\ 0 & 3 & 5 \end{pmatrix}$ A is SPD since $|2| > |0| + |1|$, $|5| > |0| + |3|$

$|4| > |-1| + |1|$

note if A is not SPD, can try swapping rows or columns to make it SPD

③ Iterative method often preffered if A is sparse (has a lot of zeros)

- can sometime get $O(n^2)$ cost in solving sparse system.

④ Two other methods are Gauss - Seidel and SOR (successive over-relaxation)

Nonlinear Systems :

How to solve $\vec{x} = \vec{g}(\vec{x})$ if $\vec{g}$ is not linear? (can't write $\vec{g}$ as matrix A)

ex: $\begin{cases} f_1(x,y) = e^x - y = 0 \\ f_2(x,y) = xy - e^x = 0 \end{cases}$ can write as $\vec{F}(\vec{x}) = \vec{0}$

$\begin{pmatrix} e^x - y \\ xy - e^y \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Recall 1d Newton :

$0 = f(x_1) = f(x_0) + f'(x_0)(x_1 - x_0) + O((x_1 - x_0)^2)$

new guess, initial guess

$\Rightarrow x_1 = x_0 - f(x_0)/f'(x_0)$ and then iterate ...

Multivariable Newton method :

$0 = \vec{F}(\vec{x}_1) = \vec{F}(\vec{x}_0) + D\vec{F}(\vec{x}_0)(\vec{x}_1 - \vec{x}_0) + O(\|\vec{x}_1 - \vec{x}_0\|^2)$ vector-valued taylor expansion

$= \begin{pmatrix} \partial_{x_1} f_1 & \partial_{x_2} f_1 & \cdots & \partial_{x_n} f_1 \\ \vdots & & & \\ \partial_{x_1} f_n & \partial_{x_2} f_n & \cdots & \partial_{x_n} f_n \end{pmatrix}$ Jacobian matrix

$-\vec{F}(\vec{x}_0) = D\vec{F}(\vec{x}_0)(\vec{x}_1 - \vec{x}_0)$

$-[D\vec{F}(\vec{x}_0)]^{-1} \vec{F}(\vec{x}_0) = \vec{x}_1 - \vec{x}_0 \quad \Longrightarrow \quad \vec{x}_1 = \vec{x}_0 - [D\vec{F}(\vec{x}_0)]^{-1} \vec{F}(\vec{x}_0)$

Method : $\begin{cases} \vec{x}_{n+1} = \vec{x}_n - [D\vec{F}(\vec{x}_n)]^{-1} F(\vec{x}_n) \\ \vec{x}_0 = \text{starting guess} \end{cases}$

To avoid computing inverses, method is to get

$\vec{s} = [D\vec{F}(\vec{x}_n)]^{-1} \vec{F}(\vec{x}_n)$

$D\vec{F}(\vec{x}_n) \cdot \vec{s} = \vec{F}(\vec{x}_n)$ ← linear system

① Fix $\vec{x}_0$ starting guess

② Solve for $\vec{s}$ in the linear system $D\vec{F}(\vec{x}_0)\vec{s} = \vec{F}(\vec{x}_0)$

with $PA = LU$ or Matlab \ command

③ Update $\vec{x}_1 = \vec{x}_0 - \vec{s}$

④ Repeat the iteration

Convergence of Multivariable Newton

Suppose $\vec{x}_{n+1} = \vec{g}(\vec{x}_n)$

In 1d, FPI converges locally if $|g'(r)| < 1$ (since then have contraction mapping)

In the vector case, FPI converges locally if $\rho(D\vec{g}(r)) < 1$

where $\rho(A) = \max_{i=1,\dots,n} |\lambda_i|$ and λ_i are eigenvalues of A

Thm: $\rho(A) \le \|A\|$ for any matrix operator norm

Remark: For Newton, $\vec{g}(\vec{x}) = \vec{x} - [D\vec{F}(\vec{x})]^{-1}\vec{F}(\vec{x})$ so $D\vec{g}(\vec{x})$ is very complicated.

Ex: $\begin{cases} f_1(x,y) = e^x - y = 0 \\ f_2(x,y) = xy - e^x = 0 \end{cases}$ $(\vec{F}(\vec{x}) = \vec{0})$

Solution is $(x, y) = (1, e)$

Jacobian: $D\vec{F}(\vec{x}) = \begin{pmatrix} \partial_x f_1 & \partial_y f_1 \\ \partial_x f_2 & \partial_y f_2 \end{pmatrix} = \begin{pmatrix} e^x & -1 \\ y - e^x & x \end{pmatrix}$

① Fix $\vec{x}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ (just need close to root)

② Solve $D\vec{F}(\vec{x}_0)\vec{s} = \vec{F}(\vec{x}_0)$ for $\vec{s}$

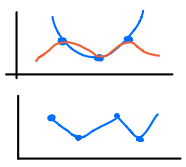
or $\begin{pmatrix} 1 & -1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ $\Rightarrow$ $s_1 = 1$, $s_2 = 0$ so $\vec{s} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

③ $\vec{x}_1 = \vec{x}_0 - \vec{s} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$

④ Repeat

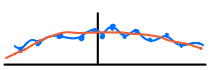
10/6

Interpolation

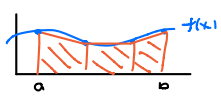


- polynomial $p \in \mathbb{P}_n$ degree n
- trig
- piecewise (local) interpolation
- pw linear

Uses: i) To approximate functions



ii) Riemann sum



pw linear interpolant

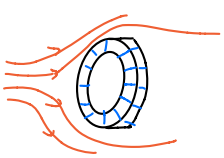
iii) ODE: $y' = f(t, y(t))$

$\to y(t) = y_0 + \int_0^t f(s, y(s))\,ds$

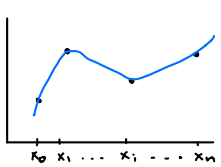
can approximate using interpolation

iv) PDE: finite element methods.

use 2D linear interpolation



Polynomial Interpolation



(x_i, y_i) $i = 0, 1, \dots n$ data points

Want to fit degree n (or less) polynomial $p \in \mathbb{P}_n$ set of n degree polynomials (or less)

Theorem: If the $n+1$ points x_i are distinct, then $\exists!\ p \in \mathbb{P}_n$ interpolating $\{(x_i, y_i)\}_{i=0}^n$.

To construct $p \in \mathbb{P}_n$, need a basis of $\mathbb{P}_n$.

(a linearly independent spanning set)

Def: ① A set $\{v_i\} \subset V$ is linearly independent.

if $c_1 v_1 + \cdots + c_n v_n = 0 \Rightarrow c_i = 0\ \forall i$

② $\{V_i\}$ span V if any $v \in V$ can be written as a linear combination of the $\{V_i\}$.

Monomial basis of $\mathbb{P}_n = \{1, x, x^2, \cdots, x^n\}$

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \in \mathbb{P}_n$$

to interpolate, we want $p_n(x_i) = y_i \quad \forall i = 0, 1, \ldots, n$

get system to solve for $\{a_i\}_{i=0}^n$.

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$$

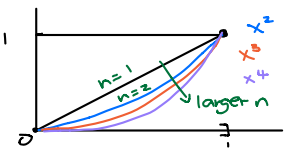
$\vec{y}$ $\qquad$ A $\qquad$ $\vec{a}$

$\vec{y} = A\vec{a}$ – can solve system in Matlab $a = A \backslash y$

A – a "Vandermonde matrix" (very ill-conditioned)

For $n \geq 10$, $\text{cond}(A) = 10^{12}$ in general

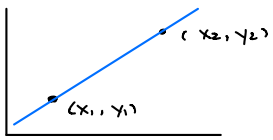
$\{1, x, x^2, \cdots, x^n\}$ is not "sufficiently" linearly independent



so for large n the basis functions look less and less linearly independent.

So for n large, monomial basis is bad.

Lagrange Interpolation – basis for $\mathbb{P}_n$.



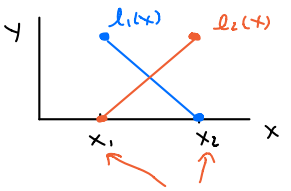
linear interpolation of 2 points

monomial basis = $p(x) = a_0 + a_1 x$

Lagrange basis = $p(x) = y_1 \underbrace{\frac{x - x_2}{x_1 - x_2}}_{l_1(x)} + y_2 \underbrace{\frac{x - x_1}{x_2 - x_1}}_{l_2(x)}$ linear Lagrange basis functions

$$p(x) = y_1 \frac{x - x_2}{x_1 - x_2} + y_2 \frac{x - x_1}{x_2 - x_1} = y_1 l_1(x) + y_2 l_2(x)$$

linear depen. / linear indep.



$l_1(x)$ = plug x_1 into $p(x)$ get $\frac{x_1 - x_2}{x_1 - x_2} = 1$

x_2 into $p(x)$ get $x_2 - x_2 / \cdots = 0$

l_1, l_2 "sufficiently" linearly independent

at the nodes, $l_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} = \delta_{ij}$ (Kronecker delta)

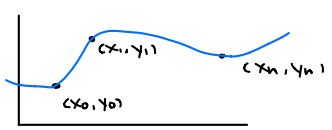
matrix system looks like

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{A_{ij} = l_i(x_j) = \delta_{ij}} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

note that $\text{cond}(I) = 1$

$\text{cond}(I) = \|I\| \|I^{-1}\| = \|I\|^2 = 1$ well-conditioned.

$n+1$ point cases



Lagrange interpolant is $p_n(x) = \sum_{i=0}^n y_i l_i(x)$

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} = \frac{(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0)(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}$$

skipped $j = i$ term

Remarks: ① note $l_i \in \mathbb{P}_n$, $\forall i = 0, 1 \ldots n$ since have $(n+1) - 1$ terms in the product

② we have interpolation:

$$p_n(x_k) = \sum_{i=0}^n y_i \underbrace{l_i(x_k)}_{= \delta_{ik}} = y_{k \cdot 1} = y_k$$

$$= \delta_{ik} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$$

10/8

Newton's Divided Differences (interpolation)

Recall: ① Monomial basis: $p(x) = \sum_{i=0}^{n} a_i x^i$ ← basis functions

- simple form but ill-conditioned system (Vandermonde) to obtain.

② Lagrange basis: $P_n(x) = \sum_{i=0}^{n} y_i \left(\prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \right)$ ← basis functions

└ easy to find, well-conditioned, but more complicated form.

ex: (1,1), (2,5), (3,4) use Lagrange.

$$P_2(x) = 1 \cdot \frac{(x-2)(x-3)}{(1-2)(1-3)} + 5 \cdot \frac{(x-1)(x-3)}{(2-1)(2-3)} + 4 \cdot \frac{(x-1)(x-2)}{(3-1)(3-2)}$$

Newton basis for interpolation: $\{(x_i, y_i)\}_{i=0}^{n}$

$$p(x) = a_0 \cdot 1 + a_1 \cdot (x - x_0) + a_2 (x - x_0)(x - x_1) + \cdots + a_n (x - x_0)(x - x_1) \cdots (x - x_{n-1})$$ ← basis functions

basis functions: $N_0(x) = 1$, $N_i(x) = \prod_{k=0}^{i-1} (x - x_k)$ for $i = 1, 2, \ldots, n$

ex:
$$y_0 = p(x_0) = a_0 \cdot 1 + 0 + 0$$
$$y_1 = p(x_1) = a_0 \cdot 1 + a_1 (x_1 - x_0) + 0$$
$$y_2 = p(x_2) = a_0 \cdot 1 + a_1 (x_2 - x_0) + a_2 (x_2 - x_0)(x_2 - x_1)$$

get lower triangular system $O(n^2)$

For Newton, easy to add new points:

Let $P_n \in \mathbb{P}_n$ interpolate $n+1$ points

set $P_{n+1}(x) = P_n(x) + C_{n+1} N_{n+1}(x)$ ← $= 0$ for $x = x_0, \ldots, x_n$

some constant

P_{n+1} interpolates original $n+1$ points.

now can choose C_{n+1} so P_{n+1} interpolates new point

$$y_{n+1} = P_{n+1}(x_{n+1}) = P_n(x_{n+1}) + C_{n+1} N_{n+1}(x_{n+1})$$

$$C_{n+1} = \frac{y_{n+1} - P_n(x_{n+1})}{N_{n+1}(x_{n+1})}$$ ← known as the $(n+1)^{st}$ "divided difference"

Notation: $C_{n+1} = f[x_0, \ldots, x_n, x_{n+1}]$

Method: Let $y_i = f(x_i)$

x_0	$f(x_0)$	$\frac{f(x_1) - f(x_0)}{x_1 - x_0} =: f[x_0, x_1]$	$\frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} =: f[x_0, x_1, x_2]$	$f[x_0, x_1, x_2, x_3]$
x_1	$f(x_1)$	$\frac{f(x_2) - f(x_1)}{x_2 - x_1} =: f[x_1, x_2]$	$=: f[x_1, x_2, x_3]$	top row gives coefficients $\{a_n\}$
x_2	$f(x_2)$	$\frac{f(x_3) - f(x_2)}{x_3 - x_2} =: f[x_2, x_3]$		
x_3	$f(x_3)$			

definition of divided differences:

$$f[x_k] := f(x_k)$$

$$f[x_k, x_{k+1}] = f[x_{k+1}] - f[x_k] \;/\; x_{k+1} - x_k$$

$$f[x_k, x_{k+1}, x_{k+2}] = f[x_{k+1}, x_{k+2}] - f[x_k, x_{k+1}] \;/\; x_{k+2} - x_k$$

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots$$

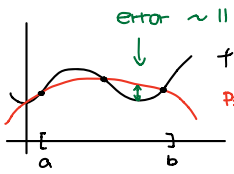
ex: (1,1), (2,5), (3,4)

1	1	$\frac{5-1}{2-1} = 4$	$\frac{-1-4}{3-1} = -\frac{5}{2}$	$\frac{4}{3}$
2	5	$\frac{4-5}{3-2} = -1$	$\frac{2-(-1)}{4-2} = \frac{3}{2}$	
3	4	$\frac{6-4}{4-3} = 2$		
4	6			

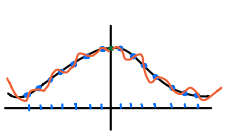
$$P(x) = 1 + 4(x-1) + \left(-\frac{5}{2}\right)(x-1)(x-2) + \left(\frac{4}{3}\right)(x-1)(x-2)(x-3)$$

10/13

Interpolation Error (Approximation Theorm)



Runge Phenomenon



$f(x) = \frac{1}{1+12x^2}$

$p(x)$ polynomial interpolant "wiggle" at endpoints, called Rung phenomenon

evenly-spaced nodes

- Moral: High degree interpolation does not implies accuracy (in general)
- solution: Move some nodes towards endpoints for higher accuracy

Thm: Let $f \in C^{n+1}[a,b]$, and $p_n \in \mathbb{P}_n$ be a polynomial interpolant at nodes $x_0 \dots x_n \in [a,b]$

then $\forall x \in [a,b]$, $\exists\ \xi_n \in (a,b)$

s.t $f(x) - p_n(x) = \frac{f^{(n+1)}(\xi_n)}{(n+1)!} \prod_{j=0}^{n} (x - x_j)$ basically Taylor series remeinder.

note error $\equiv 0$ at the nodes

Ex: let $f(x) = e^x$ and let $p(x)$ interpolate through $x = -1, -\frac{1}{2}, 0, \frac{1}{2}, 1$

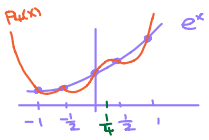
Find $|f(1/4) - p_4(1/4)|$ error bound

Have

$|f(1/4) - p_4(1/4)| \le \frac{1}{5!} |f^{(5)}(\xi)| |(\frac{1}{4}+1)(\frac{1}{4}+\frac{1}{2})(\frac{1}{4})(\frac{1}{4}-\frac{1}{2})(\frac{1}{4}-1)|$

$f^{(5)}(\xi) = e^{\xi} \le e$ on the interval $[-1,1]$

≤ 0.000995



Optimal Node Spacing (for polynomial interpolation)

$f(x) - p_{n-1}(x) = \frac{f^{(n)}(\xi_n)}{n!} \prod_{j=1}^{n} (x - x_j)$ ← want to minimize error by choosing nodes optimally

- suppose f and n are given
- Define $w(x) = \prod_{j=1}^{n} (x - x_j)$ for convenience (the "node polynomial")

Chebyshev's Theorem: The choice of nodes $x_1 \dots x_n$ in the interval $[-1,1]$ that minimizes $\max_{-1 \le x \le 1} |w(x)|$ is $x_i = \cos\frac{(2i-1)\pi}{2n}$ $(i = 1, \dots, n)$

in which case the minimum value of it is $\frac{1}{2^{n-1}}$

x_i called the Chebyshev nodes

ex: Find worse case error for approximating $f(x) = e^x$ by $p_4 \in \mathbb{P}_n$ on $[-1,1]$.

error $= |f(x) - p_4(x)| \le \frac{1}{5!} |f^{(5)}(\xi)| |w(x)|$

$x_i = \cos\frac{(2i-1)\pi}{2n}$, $\prod_{j=1}^{5} (x - x_j) = (x - \cos\frac{\pi}{10})(x - \cos\frac{3\pi}{10}) \cdots (x - \cos\frac{9\pi}{10})$

Have

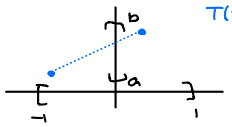
$|w(x)| \le \frac{1}{2^{n-1}} = \frac{1}{2^{5-1}} = \frac{1}{16}$

so error $\le \frac{1}{5!} e \cdot \frac{1}{16} = 0.00142$ on the interval $[-1,1]$

best possible upper bound obtained when Chebyshev nodes used

any other choice of nodes yields a potentially larger error

Change of Interval $T: [-1,1] \to [a,b]$



$T(x) = \alpha x + \beta$ $\quad T(-1) = a$ $\quad T(1) = b$

get $T(x) = \frac{b-a}{2} x + \frac{b+a}{2}$

So on the interval $[a,b]$, the Chebyshev nodes are given by $x_i = \frac{b+a}{2} + \frac{b-a}{2} \cos\frac{(2i-1)\pi}{2n}$ $\quad i = 1,\dots,n$

How does the $|w(x)|$ error transform?

Let Chebyshev nodes $x_i \in [-1,1]$

and new Chebyshev nodes $y_i = \frac{b+a}{2} + \frac{b-a}{2} x_i \in [a,b]$

let $y = \frac{b+a}{2} + \frac{b-a}{2} x \in [a,b]$

Have $|w(y)| = |\prod_{i=1}^{n} (y - y_i)| = |\prod_{i=1}^{n} (\frac{b-a}{2})(x - x_i)| \le (\frac{b-a}{2})^n |\prod_{i=1}^{n} (x - x_i)| \le \frac{1}{2^{n-1}} (\frac{b-a}{2})^n$

$\le \frac{1}{2^{n-1}}$ by Chebyshev Thm

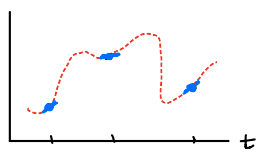
So a bound on polynomial interpolation error on $[a,b]$ with Chebyshev points is given by $|f(x) - p_{n-1}(x)| \le \frac{|f^{(n)}(\xi)|}{n!} \cdot \frac{1}{2^{n-1}} (\frac{b-a}{2})^n$

$\Rightarrow \|f - p_{n-1}\|_\infty \le \frac{\|f^{(n)}\|_\infty}{n!} \frac{1}{2^{n-1}} (\frac{b-a}{2})^n$ where $\|f\|_\infty = \max_{a \le x \le b} |f(x)|$

the error bound with Chebyshev nodes ☺

10/19

Hermite Interpolation

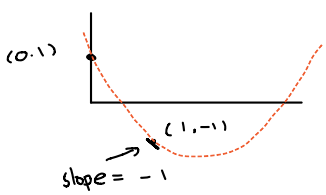


position of a car

want to match slope + values

Goal: Given data points, or $f(x)$, want to find $p \in \mathbb{P}$ interpolating $\{(x_i, f(x_i))\}_i$ and possibly derivatives $f^{(\ell)}(x_j)$ at some x_j.

Global Hermite



can let $P_2(x) = a_0 + a_1 x + a_2 x^2$

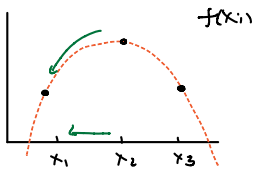
the 3 constrains give system for $\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$.

$\Rightarrow P_2(x) = 1 - 3x + x^2$

consider $f(x)$ and a polynomial interpolant for (x_0, y_0) and also $f^{(j)}(x_0)$ $\quad 0 \le j \le n$

$\Rightarrow p_n(x) = \sum_{j=0}^{n} \frac{f^{(j)}(x_0)}{j!} (x - x_0)^j$ taylor polynomial

Newton Approach (repeated nodes)



x_1	$f(x_1)$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$
x_2	$f(x_2)$	$f[x_2, x_3]$	
x_3	$f(x_3)$		

Newton polynomial

$p(x) = f(x_1) + f[x_1, x_2](x - x_1) + f[x_1, x_2, x_3](x - x_1)(x - x_2)$

what happens as $x_2 \to x_1$?

$\lim_{x_2 \to x_1} p(x) = f(x_1) + \lim_{x_2 \to x_1} f[x_1, x_2] \cdot (x - x_1) + \lim_{x_2 \to x_1} f[x_1, x_2, x_3](x - x_1)(x - x_1)$

let $\lim_{x_2 \to x_1} f[x_1, x_2] = \lim_{x_2 \to x_1} \frac{f(x_2) - f(x_1)}{x_2 - x_1} =: f'(x_1)$

So repeated nodes in Newton divide differences are used to interpolate derivate conditions:

$$\begin{array}{c|ccc} x_1 & f(x_1) & & \\ & & \searrow f'(x_1) = f[x_1, x_1] & \\ x_1 & f(x_1) & \nearrow & \searrow f[x_1, x_1, x_3] = \frac{f[x_1, x_3] - f[x_1, x_1]}{x_3 - x_1} =: f[x_1, x_1, x_3] \\ & & \searrow f[x_1, x_3] & \nearrow \\ x_3 & f(x_3) & & \end{array}$$

Ex: Find $p \in \mathbb{P}_2$ interpolating $(1, -1)$, $(0, 1)$ and $p'(1) = -1$

$$\begin{array}{c|ccc} 1 & -1 & & \\ & & \searrow -1 & \\ 1 & -1 & \nearrow & \searrow \frac{-2-(-1)}{0-1} = 1 \\ & & \searrow \frac{1-(-1)}{0-1} = -2 & \nearrow \\ 0 & 1 & & \end{array}$$

so $P_2(x) = -1 - (x-1) + (x-1)^2$

$= 1 - 3x + x^2$

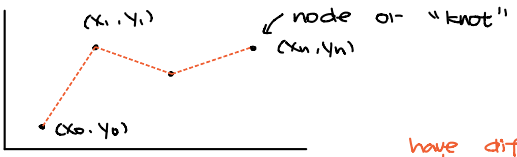
Remarks:

① For multiple nodes x_j of multiplicity m (m repetitions) we suppose we know $f(x_j), f'(x_j), \dots, f^{(m-1)}(x_j)$ [for existence / uniqueness]

② For higher derivatives, we define

$$\underbrace{f[x_j, x_j, \dots, x_j]}_{m \text{ repetitions}} := \frac{f^{(m-1)}(x_j)}{(m-1)!}$$ note rescaling

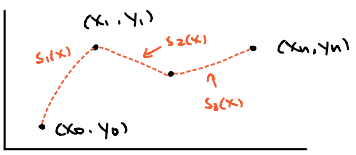
Piecewise Polynomial Interpolation



have different linear function on each intval

linear piecewise: $L(x) = \begin{cases} L_0(x) & x_0 \le x \le x_1 \\ \vdots & \\ L_{n-1}(x) & x_{n-1} \le x \le x_n \end{cases}$

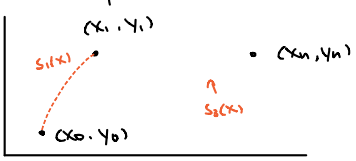
Spline interpolation



a spline is a piecewise polynomial of degree k, that is of class $C^{k-1}(a, b)$

each $S_i(x)$ is cubic and want a curve that is $C^2(a, b)$

Cubic splines



$$\begin{cases} S_1(x) = y_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 \\ \vdots \\ S_{n-1}(x) = y_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3 \end{cases}$$

Have unknows $(b_i, c_i, d_i)_{i=1}^{n-1}$ $\Rightarrow$ $3(n-1)$ unknows

# of equations:

Each $S_i(x)$ already interpolates (x_i, y_i)

But want S_i to interpolate (x_{i+1}, y_{i+1}) too.

Get $n-1$ equations by forcing $y_{i+1} = S_i(x_{i+1})$ $(i = 1, \dots, n-1)$

Also need $S'_{i-1}(x_i) = S'_i(x_i)$ for $i = 2 \dots n-1$ inorder to match slopes at interior pts.

$\Rightarrow$ get $n-2$ equations.

Also want $S''_{i-1}(x_i) = S''_i(x_i)$ for $i = 2 \dots n-1$, to match curvature at interior nodes.

$\Rightarrow$ get $n-2$ equations.

Have $3n-5$ equations to solve for $3n-3$ unknowns.

so have 2 more unknowns than equations, can't solve for them uniquely.

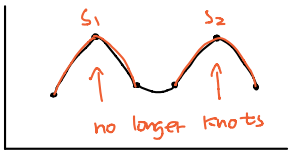
Choices of two equations to close system:

① natural spline: $S_1''(x_1) = S_{n-1}''(x_n) = 0$

② clamped spline: Fix $S_1'(x_1)$ and $S_{n-1}'(x_n)$ arbitrarily at endpoints x_1, x_n.

③ curvature-adjusted: Fix $S_1''(x_1)$ and $S_{n-1}''(x_n)$ arbitrarily

④ not-a-knot spline:

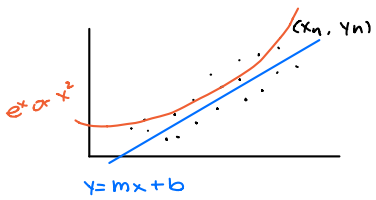


set $S_1'''(x_2) = S_2'''(x_2)$ and $S_{n-2}'''(x_{n-1}) = S_{n-1}'''(x_{n-1})$

$\Rightarrow d_1 = d_2$ and $d_{n-2} = d_{n-1}$

10/22

Least Squares Approximation



- Try to fit data to a model (user's choice)
- Consider $y = mx + b$ (model), Force data to fit model

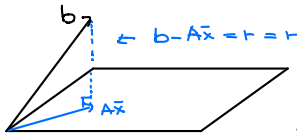
$$\begin{aligned} mx_1 + b &= y_1 \\ mx_2 + b &= y_2 \\ &\vdots \\ mx_n + b &= y_n \end{aligned} \quad \Rightarrow \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}_{n\times 2} \begin{pmatrix} m \\ b \end{pmatrix}_{2\times 1} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Get overdetermined system to solve for m, b

General problem: Find $\bar{x}$ that best approximates a "solution" of $Ax = b$

Geometric Approach. $Ax = b$, $\bar{x}$ = approximate solution, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$

suppose $m > n$, so more eqn than unknowns.



$b - A\bar{x} = r$ = residual (error)

$W := \text{span}(A) = \{Ax \mid x \in \mathbb{R}^n\}$

$A\bar{x} = \text{proj}_W b$ is the best approximation to b in $\text{Span}(A) = W$.

Since $r \perp W \Rightarrow \forall x \in \mathbb{R}^n$, $\langle Ax, r\rangle = 0$

$\Rightarrow (Ax)^T(b - A\bar{x}) = 0$ (recall $\langle x, y\rangle = x^T y$)

$\Rightarrow x^T A^T (b - A\bar{x}) = 0$

$\Rightarrow x^T (A^T b - A^T A\bar{x}) = 0 \quad \forall x \in \mathbb{R}^n$

Since $\forall x \in \mathbb{R}^n$, $\Rightarrow A^T b - A^T A\bar{x} = 0$

$A^T A\bar{x} = A^T b$ ← normal equations that we can solve for $\bar{x}$

note: ① If $\langle x, y\rangle = 0 \quad \forall x \in \mathbb{R}^n$, then $y = 0$.

pf: let $x = y$ then $\langle x, y\rangle = \langle y, y\rangle = \|y\|^2 = 0 \Rightarrow y = 0$

② In matlab, >> $\bar{x} = (A^T A) \backslash (A^T b)$

Why called least-squares?

Residual Minimization:

$Ax = b$ want to minimize $r = b - Ax$ (residual) over all possible $x \in \mathbb{R}^n$

For least-squares, want $\|r\|_2^2 = r_1^2 + r_2^2 + \cdots + r_n^2$ minimized.

2 norm (hence least squares)

For calc 3, find x s.t $\nabla_x \|r\|_2^2 = 0$ (critical points)

we'll compute this componentwise:

$(A)_{ij} = a_{ij}$, $(\vec{x})_j = x_j$, $\vec{x}\cdot\vec{x} = \sum_{i=1}^n x_i x_i = x_i x_i$ (Einstein notation)

$(\nabla\phi)_i = \partial_{x_i}\phi$, $(Ax)_{ij} = a_{ij}x_j$

$$\nabla_x \|r\|_2^2 = 0 = (\nabla_x \|r\|_2^2)_i = \partial_{x_i}(\|r\|_2^2) = \partial_{x_i}(r_j r_j) = r_j \partial_{x_i} r_j + r_j \partial_{x_i} r_j = 2 r_j \partial_{x_i} r_j$$

To compute $\partial x_i r_j$, recall $r = b - Ax$, $r_j = b_j - a_{jk} x_k$

$\partial x_i r_j = \cancel{\partial x_i b_j}^{=0} - a_{jk} \partial x_i x_k = -a_{jk} \delta_{ik} = -a_{ji}$

$= \begin{cases} 1 & i=k \\ 0 & i \neq k \end{cases} = \delta_{ik}$

So $0 = 2 r_j \partial x_i r_j = 2 r_j (-a_{ji}) = 2(\underbrace{b_j - a_{jk} x_k}_{b - Ax})(\underbrace{-a_{ji}}_{-A^T})$

$0 = 2(-A^T)(b - Ax)$ switch order so dimensions work out correctly

$A^T A x = A^T b$ (normal equations)

Normal Eqn Pitfalls.

$A = \begin{pmatrix} 1 & 1 \\ \varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix}$, $0 < \varepsilon < \varepsilon_{mach} \approx 10^{-16}$

$A^T A = \begin{pmatrix} 1 & \varepsilon & 0 \\ 1 & 0 & \varepsilon \end{pmatrix} \begin{pmatrix} 1 & 1 \\ \varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix} = \begin{pmatrix} 1+\varepsilon^2 & 1 \\ 1 & 1+\varepsilon^2 \end{pmatrix}$ stored as $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ in Matlab

$\det(A^T A) = 0$. not invertible, new method or higher precision

Based on QR decomposition.

$A \in \mathbb{R}^{m \times n}$, $A = QR$ Q: orthogonal $\mathbb{R}^{m \times m}$ matrix $Q^{-1} = Q^T$

R: upper triangular matrix in $\mathbb{R}^{m \times n}$, $R = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix}_{m \times n}$

Thm: QR always exists

>> [Q, R] = qr(A)

Thm: If Q is orthogonal, then $\|Qr\|_2 = \|r\|_2$.

Any minimizer x of $\|b - Ax\|_2$ also minimizes $\|Q^T A x - Q^T b\|_2$

note: $\|r\|_2^2 = \|Q^T r\|_2^2 \Leftrightarrow \|Qr\|_2^2 = \|r\|_2^2$

with $A = QR$, $Q^T A = Q^T Q R = R$

Ex: let $A \in \mathbb{R}^{4 \times 2}$, $x \in \mathbb{R}^2$, $b \in \mathbb{R}^4$, let $A = QR$.

$\min_{x \in \mathbb{R}^2} \|Ax - b\|_2 = \min_{x \in \mathbb{R}^2} \|\underbrace{Q^T A}_{=R} x - Q^T b\|_2 = \min_{x \in \mathbb{R}^2} \left\| \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} \right\|_2$ where $\vec{c} = Q^T \vec{b}$

- have c_3, c_4 constant for all $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, so ignore c_3, c_4, choose $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ to solve.

$\begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$ solution is $\bar{x}$ and this easily solved by back-substitution

- have then $\min_{x \in \mathbb{R}^2} \|Ax - b\|_2 = \|A\bar{x} - b\|_2 = \sqrt{c_3^2 + c_4^2}$

least-squares by QR

let $Ax = b$, and $A = QR$, $A \in \mathbb{R}^{m \times n}$

set $\hat{R}$ = upper $n \times n$ submatrix of R

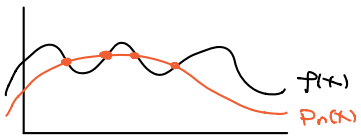
$\hat{c}$ = upper n entries of $\vec{c} = Q^T \vec{b}$

solve $\hat{R}\bar{x} = \hat{c}$ for the least-squares solution $\bar{x}$.

prefered over $A^T A x = A^T$ if done in finite precision.

10/27

Least squares for functions



want "best" approximation of f out of $\mathbb{P}_n$ or some other finite dimensional set

need norm to quantify "best"

Def: let V be a vector space, $\|\cdot\| : V \to \mathbb{R}$ is a norm if

1) $\|f\| \geq 0$ and $\|f\| = 0 \iff f = 0$
2) $\|\alpha f\| = |\alpha| \|f\|$ $\forall \alpha \in \mathbb{R}$
3) $\|f+g\| \leq \|f\| + \|g\|$

- Distance between f, g is $\|f-g\|$

- Choice of norm: (suppose f is $C[a,b]$ so everything is well-defined

$$\|f\|_{L^\infty(a,b)} = \|f\|_\infty := \max_{a \leq x \leq b} |f(x)|$$

$$\|f\|_{L^2(a,b)} = \|f\|_2 := \left(\int_a^b |f(x)|^2 dx \right)^{1/2}$$ use for L^2 or least-squares minimization

$$\|f\|_{L^1(a,b)} = \|f\|_1 := \int_a^b |f(x)| dx$$

- For 2-norm, can define inner-product $\langle f, g \rangle = \int_a^b f(x) g(x) dx$ think $\sum_{i=1}^n x_i y_i$ in $\mathbb{R}^n$
  with $\|f\|_2 = \sqrt{\langle f, f \rangle} = \left(\int_a^b |f(x)|^2 dx \right)^{1/2}$

- Useful result: $\|f+g\|_2^2 = \|f\|_2^2 + 2\langle f, g \rangle + \|g\|_2^2$

- If $\langle f, g \rangle = 0$, we say f and g are orthogonal, and if also $\|f\|_2 = \|g\|_2 = 1$. we say they are orthonormal.

Least-squares problem:

Goal: Let $V_n = \text{span}(g_1, \dots, g_n)$ be a finite dimensional space (think $\mathbb{P}_n$) with $\{g_i\}_{i=1}^n$ forming a basis of V_n.

Find $f_n \in V_n$ closest to f in the L^2-sense (least-squares)

i.e. find f_n s.t $\|f - f_n\|_{L^2}^2 = \min_{v \in V_n} \|f - v\|_2^2 = \min_{v \in V_n} \left(\int_a^b |f(x) - v(x)|^2 dx \right)$

Expand $v = \sum_{i=1}^n c_i g_i \in V_n$ c_i: unknown coefficients $\{c_i\}_{i=1}^n$

g_i: basis vectors

then $\|f - v\|_2^2 = \|f\|_2^2 - 2\langle f, v \rangle + \|v\|_2^2$

$= \|f\|_2^2 - 2\langle f, \sum c_i g_i \rangle + \langle \sum c_i g_i, \sum c_j g_j \rangle$

$= \|f\|_2^2 - 2 \sum c_i \langle f, g_i \rangle + \sum_i \sum_j c_i c_j \langle g_i, g_j \rangle$

$= \phi(\vec{c})$

want to minimize over all possible $\vec{c}$, so set $\nabla_c \phi(\vec{c}) = \vec{0}$

$$0 = \frac{\partial}{\partial c_k} \phi(\vec{c}) = 0 - 2 \sum \frac{\partial c_i}{\partial c_k} \langle f, g_i \rangle + \sum_i \sum_j \frac{\partial}{\partial c_k} c_i c_j \langle g_i, g_j \rangle$$

$\frac{\partial c_i}{\partial c_k} = \delta_{ik} = \begin{cases} 1 & i=k \\ 0 & \text{else} \end{cases}$ $\qquad = c_i \delta_{ik} + c_j \delta_{jk}$

$$= -2 \langle f, g_k \rangle + \sum_i c_i \langle g_i, g_k \rangle + \sum_j c_j \langle g_k, g_j \rangle$$

$$= -2 \langle f, g_k \rangle + 2 \sum_i c_i \langle g_i, g_k \rangle$$

$\Rightarrow$ Get $\sum_{i=1}^n c_i \langle g_i, g_k \rangle = \langle f, g_k \rangle$

This is a linear system $A\vec{c} = \vec{b}$ with $A_{ik} = \langle g_i, g_k \rangle$ and $b_k = \langle f, g_k \rangle$

note: $A_{ik} = A_{ki}$ since $\langle g_i, g_k \rangle$ is symmetric.

If $\{g_i\}_{i=1}^n$ are orthonormal, then $\langle g_i, g_k \rangle = \delta_{ik} = \begin{cases} 1 & i=k \\ 0 & \text{else} \end{cases}$

the we get $c_k = \langle f, g_k \rangle$

$\Rightarrow v = \sum_{k=1}^n c_k g_k = \sum_{k=1}^n \langle f, g_k \rangle g_k$ is our L^2-minimizing vector.

from linear algebra, this is nothing other than orthogonal projection of f onto V_n.

Generalized Least-Squares

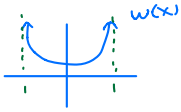
$\langle f, g \rangle_w := \int_a^b f(x) g(x) w(x)\, dx$ $w(x)$: weight function, need $w(x) \geq 0$, and $w(x) \not\equiv 0$ on any subinterval of $[a,b]$.

and

$\|f\|_{2,w} = \left(\int_a^b |f(x)|^2 w(x)\, dx \right)^{1/2}$

All pervious theory applies with $\|f\|_{2,w}$ and $\langle f, g \rangle_w$.

Ex: $w(x) = \frac{1}{\sqrt{1-x^2}}$ on $[-1,1]$



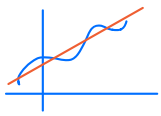
$\min_{v \in V_n} \|f - v\|_{2,w}^2 = \min_{v \in V_n} \left(\int_{-1}^1 |f(x) - v(x)|^2 w(x) \right)^{1/2}$

want $|f-v|$ small where w is large.

so this $w(x)$ places emphasis near $x = \pm 1$

How to find orthonormal basis?

Example: $V_n = \text{span}(1, x) = \mathbb{P}_1$ and $w(x) \equiv 1$ on $[0,1]$



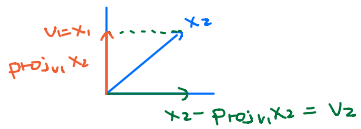
so $\langle f, g \rangle_w = \int_0^1 f(x) g(x)\, dx$

Note $\int_0^1 1 \cdot x\, dx = \frac{1}{2} x^2 \Big|_0^1 = \frac{1}{2} \neq 0$, so $\{1, x\}$ is not orthogonal.

Use Gram-Schmidt to orthgonalize:

let $x_1 = 1$

$x_2 = x$



Define $v_1 = x_1 = 1$

$v_2 = x_2 - \text{proj}_{v_1} x_2 = x_2 - \frac{\langle x_2, v_1 \rangle}{\|v_1\|_2^2} v_1$

So now $v_2 = x - \frac{\langle x, 1 \rangle}{\|1\|_2^2} \cdot 1 = x - \frac{1/2}{1} \cdot 1 = x - 1/2$

$\|1\|_2^2 = \int_0^1 1^2 dx = 1$, $\langle x, 1 \rangle = \int_0^1 1 \cdot x\, dx = \frac{1}{2} x^2 \Big|_0^1 = \frac{1}{2}$

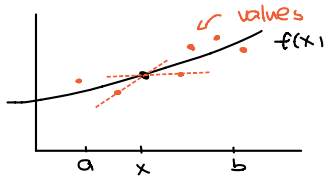
Then $v_1 = 1$, v_1 already normalized b/c $\|1\|_2^2 = 1$

and $v_2 = x - 1/2$, not normalized $\|v_2\|_2 = \left(\int_0^1 |x - 1/2|^2 dx \right)^{1/2} = \sqrt{\frac{1}{12}}$

then $\{v_1, v_2\} = \left\{ 1, \frac{x - 1/2}{1/\sqrt{12}} \right\}$ is an orthonormal basis of $\mathbb{P}_1$ on $[0,1]$ with $w(x) = 1$.

10/29.

Numerical Differentiation



values stored on computer due to finite precision

- Derivatives are sensitive to error when computed numerically
- Integration is not as sensitive to error

$f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$, $f'(x) \approx \frac{f(x+h)-f(x)}{h}$ (forward 2-point difference formula)

Error: $D_h f(x) := \frac{f(x+h)-f(x)}{h}$

$= \frac{1}{h}[(f(x) + hf'(x) + \frac{h^2}{2}f''(\xi)) - f(x)]$ used Taylor expansion cut at the $O(h^2)$ term where ξ btw $x, x+h$.

$= f'(x) + \underbrace{\frac{h}{2}f''(\xi)}_{O(h)}$ truncation error

$|D_h f(x) - f'(x)| \le \frac{h}{2}|f''(\xi)| \le \frac{h}{2}\|f''\|_\infty$

$\Rightarrow \|D_h f - f'\|_\infty \le \frac{h}{2}\|f''\|_\infty$ note as $h\to 0$, $D_h f \to f'$

but in practice there is a limitation on the h due to rounding error.

Backward 2-point difference

$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(\xi)$ stop Taylor here with ξ btw $x-h, x$.

$f'(x) = \underbrace{\frac{f(x)-f(x-h)}{h}}_{D_h f(x)} + \underbrace{\frac{h^2}{2}f''(\xi)}_{O(h) \text{ error term}}$

$\|f' - D_h f\|_\infty \le \frac{h}{2}\|h''\|_\infty$

Centered Differencing

Taylor $\begin{cases} f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{3!}f'''(\xi_1) & \xi_1 \text{ btw } x, x+h \\ f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{3!}f'''(\xi_2) & \xi_2 \text{ btw } x-h, x \end{cases}$

$f(x+h) - f(x-h) = 2hf'(x) + \frac{h^3}{6}[f'''(\xi_1) + f'''(\xi_2)]$

$f'(x) = \frac{f(x+h)-f(x-h)}{2h} - \frac{h^2}{12}[f'''(\xi_1) + f'''(\xi_2)]$

$= 2f'''(\xi)$ by IVT with ξ btw ξ_1 and ξ_2

$f'(x) = \frac{f(x+h)-f(x-h)}{2h} - \frac{h^2}{6}f'''(\xi)$ error is now $O(h^2)$

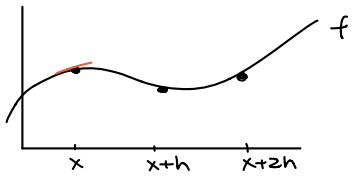
Theorem: (Generalized Intermediate Value Theorem)

let $f \in C[a,b]$, and let $x_1, \dots x_n \in [a,b]$.

let $a_i > 0$ for $i = 1, \dots, n$.

Then $\exists\, \xi \in [a,b]$ s.t. $(a_1 + \dots + a_n) f(\xi) = a_1 f(x_1) + \dots + a_n f(x_n)$.

Method of Undetermined Coefficients



How to get difference quotient for $f'(x)$ using only these points?

Write $f'(x) = Af(x) + Bf(x+h) + Cf(x+2h)$ — A, B, C are unknow coeef

$= Af(x)$

$+ B\left[f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(\xi_1)\right]$

$+ C\left[f(x) + 2hf'(x) + \frac{4h^2}{2}f''(x) + \frac{8h^3}{6}f'''(\xi_2)\right]$

$= (A+B+C)f(x) + (Bh + 2Ch)f'(x) + \left(B\frac{h^2}{2} + C2h^2\right)f''(x) + O(h^3)$

Request $A+B+C=0$

$h^2\left(\frac{B}{2} + 2C\right) = 0$

$h(B+2C) = 1$

solve for A, B, C Get $f'(x) \approx \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h}$

Numerical Differentiation and step size h

$f'(x) = \frac{f(x+h) - f(x)}{h} + \frac{1}{2}f''(\xi) =: D_h f(x) + \frac{h}{2}f''(\xi)$

$\Rightarrow |f'(x) - D_h f| \le \frac{h}{2}\|f''\|_\infty = \frac{Mh}{2}$

In practice, f is stored with error : $\tilde{f}(x) = f(x) + e(x)$

$\tilde{f}(x)$ ↑ computer – stored version

↳ error $\sim \varepsilon_{mach} \approx 10^{-16}$

$\Rightarrow D_h\tilde{f}(x) - D_h f(x) = D_h e(x)$ note that D_h is linear

so $|D_h\tilde{f}(x) - D_h f(x)| = \left|\frac{e(x+h) - e(x)}{h}\right| \le \frac{|e(x+h)| + |e(x)|}{h} \le \frac{2\varepsilon}{h}$

↑ tri-inequality

$|f'(x) - D_h\tilde{f}(x)| \le |f'(x) - D_h f(x)| + |D_h\tilde{f}(x) - D_h f(x)|$

$\pm D_h f(x)$

$|f'(x) - D_h f(x)| \le \frac{Mh}{2}$ truncation error

$|D_h\tilde{f}(x) - D_h f(x)| \le \frac{2\varepsilon}{h}$ rounding error

$\Rightarrow |f'(x) - D_h\tilde{f}(x)| \le \frac{Mh}{2} + \frac{2\varepsilon}{h}$

To minimize upper bound, set $\frac{d}{dh}\left(\frac{Mh}{2} + \frac{2\varepsilon}{h}\right) = 0 \Rightarrow h = 2\sqrt{\frac{\varepsilon}{M}}$ (optimal value of h)

In this case,

error upper bound $= \frac{Mh}{2} + \frac{2\varepsilon}{h} = 2\sqrt{\varepsilon M} \approx \sqrt{\varepsilon} \approx 10^{-8}$

Richardson Extrapolation

centered differences for $f'(x)$: $f'(x) = \frac{f(x+h)-f(x-h)}{2h} - \frac{h^2}{6}f'''(\xi_h)$ ← error term $O(h^2)$

↓ ↳ $F(h)$: approximation formula

Q: the thing we want to approximate

Goal: want to find more accurate approximation.

let $F(h)$ approximate Q

then suppose $Q \approx F(h) + Kh^n$ suppose K a constant

note $\lim_{h\to 0} F(h) = Q$

Idea: If we replace $h \mapsto \frac{1}{2}h$, then error $\approx Kh^n = K(\frac{1}{2}h)^n = Kh^n \cdot \frac{1}{2^n}$ is reduced by a factor of 2^n.

$$\frac{Q-F(h)}{2} \approx \frac{Kh^n}{2^n} \approx Q - F(\tfrac{h}{2}) \quad \Rightarrow \quad Q = \frac{2^n F(\frac{h}{2}) - F(h)}{2^n - 1}$$ Richardson Extrapolation Formula

will show the extrapolation is at least $O(h^{n+1})$

The Big-O properties

① $O(h^p) = O(ch^p)$ as $h \to 0$ for any c = constant, $c \neq 0$

② $O(h^p) \pm O(h^q) = O(h^r)$ as $h \to 0$ where $r = \min(p, q)$

suppose $Q = F_n(h) + Kh^n + O(h^{n+1})$

↑ denotes that $F(h)$ is $O(h^n)$

$Q = F_n(\frac{h}{2}) + Kh^n\frac{1}{2^n} + O(h^{n+1})$

↑ used Big-O Property ①

Extrapolation formula is

$$\frac{2^n F_n(\frac{h}{2}) - F_n(h)}{2^n - 1} = \frac{1}{2^n-1}\left[2^n\left(Q - \cancel{Kh^n}\frac{1}{2^n} - O(h^{n+1}) - \left(Q - \cancel{Kh^n} - O(h^{n+1})\right)\right)\right]$$

$$= Q + O(h^{n+1})$$

So $Q = \underbrace{\frac{2^n F_n(\frac{h}{2}) - F_n(h)}{2^n - 1}}_{F_{n+1}(h)} + O(h^{n+1}) = F_{n+1}(h) + O(h^{n+1})$

the error is now $O(h^{n+1})$ or possibly higher

Extrapolating Centered Differences for f'

$Q = F_2(h) + Kh^2$

$f'(x) = \frac{f(x+h)-f(x-h)}{2h} - \frac{h^2}{6}f'''(\xi_h)$

we get $F_3(h) = \frac{2^2 F_2(\frac{h}{2}) - F_2(h)}{2^2 - 1}$

$$= \frac{1}{3}\left[4\left(\frac{f(x+\frac{h}{2}) - f(x-\frac{h}{2})}{2(\frac{h}{2})}\right) - \frac{f(x+h)-f(x-h)}{2h}\right]$$

$$= \frac{1}{6h}\left[f(x-h) - 8f(x-\tfrac{h}{2}) + 8f(x+\tfrac{h}{2}) - f(x+h)\right] + O(h^3)$$

↳ our new higher order formula for $f'(x)$

Remarks: ① To get $O(h^3)$ term explicitly, need to Taylor expand.

② In fact the error above is not $O(h^3)$, it is $O(h^4)$!

because $F_3(h) = F_3(-h)$

error term must be exactly the same for both h and $-h$, so the error can only be $O(h^p)$ for even p and so is at least $O(h^4)$

③ In general, $h \to \frac{h}{q}$ $(q \in \mathbb{N},\ q \geq 2)$

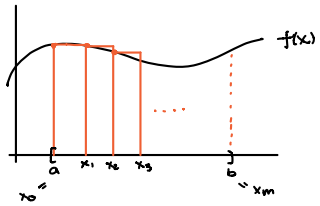
we get

$$Q = \frac{q^p(F(h) - F(\frac{h}{q}))}{1 - q^p} + O(h^r)$$

with $r > p$ and where $F(h) = Q + Kh^p + O(h^r)$

11/10

Quadrature (numerical integration)



$\int_a^b f(x)\,dx \approx \sum_{i=1}^{m} f(x_i^*)\,\Delta x_i$ (Riemann sum approximation)

$x_i^* \in [x_{i-1}, x_i]$, $\Delta x_i = x_i - x_{i-1}$

In general,

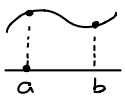
$\int_a^b f(x)\,dx = \sum_{i=0}^{m} w_i f(x_i) + E(f)$ ← error

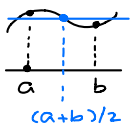
$= Q(f)$ quadrature formula

where $\{w_i\}_{i=0}^{\infty}$ are weights and $\{x_i\}_{i=0}^{\infty}$ are nodes.

Newton-Cotes quadrature

- will approximate f by polynomials at equally-spaced points, and polynomials are easy to integrate.
- closed Newton-Cotes: use endpoints / open Newton-Cotes: don't use endpoints.





midpoint

Lagrange basis for interpolation:

$l_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$ $(n+1)$ nodes $\{x_i\}_{i=0}^{n}$

$f(x) \approx P_n(x) = \sum_{i=0}^{n} f(x_i) l_i(x)$ is the n^{th} degree interpolating polynomial.

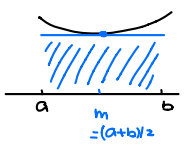
So $\int_a^b f(x)\,dx = \int_a^b P_n(x)\,dx = \sum_{i=0}^{n} f(x_i) \int_a^b l_i(x)\,dx = Q(f)$

($f(x_i)$: nodes; $\int_a^b l_i(x)\,dx$: weights w_i)

notes: ① Weights only depend on $\{x_i\}_{i=0}^{n}$, not on f.

② If $f \in \mathbb{P}_n$, then $Q(f)$ integrates f exactly.

Midpoint Rule: (open Newton-Cotes)



← interpolant $P_0(x)$

$\int_a^b f(x)\,dx \approx \int_a^b f(m)\,dx = (b-a) f(m)$

Error Analysis (using Taylor expansions)

$\int_a^b f(x)\,dx = \int_a^b \left[f(m) + (x-m) f'(m) + \frac{(x-m)^2}{2} f''(\xi_x)\right] dx$

$= f(m)(b-a) + f'(m)\underbrace{\left[\tfrac{1}{2}x^2 - mx\right]\Big|_a^b}_{=0} + \frac{1}{2}\int_a^b f''(\xi_x)(x-m)^2\,dx$

$m = (a+b)/2$

$= f(m)(b-a) + \frac{1}{2}\int_a^b f''(\xi_x)(x-m)^2\,dx$

$= f(m)(b-a) + \frac{1}{2} f(\xi) \int_a^b (x-m)^2\,dx$ where $\xi \in (a,b)$

$= f(m)(b-a) + \frac{1}{24}(b-a)^3 f''(\xi)$

Integral Mean-Value theorem

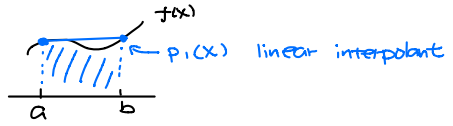
let $f, g \in [a,b]$ and $g \geq 0$. then $\exists\, \xi \in (a,b)$ s.t. $\int_a^b f(x) g(x)\,dx = f(\xi) \int_a^b g(x)\,dx$

note: thm also true if $g \leq 0$

Definition: A quadrature rule has degree of precision k if it integrates $p \in \mathbb{P}_k$ exactly.

ex: midpoint Rule has degree of precision 1.

Trapezoid Rule: C closed Newton-Cotes



Have $f(x) = \underbrace{f(a)\frac{x-b}{a-b} + f(b)\frac{x-a}{b-a}}_{= p_1(x) \text{ in lagrange basis}} + \underbrace{\frac{f''(\xi)}{2!}(x-a)(x-b)}_{= \text{interpolation error } E(x)}$

Recall: $f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=0}^{n}(x-x_j)$ ← node polynomial

$$\int_a^b f(x)\,dx = \int_a^b p_1(x)\,dx + \int_a^b E(x)\,dx = (b-a)\frac{f(a)+f(b)}{2} + \frac{1}{2}\int_a^b f''(\xi_x)(x-a)(x-b)\,dx$$

$g(x) = (x-a)(x-b) \le 0$ on $[a,b]$

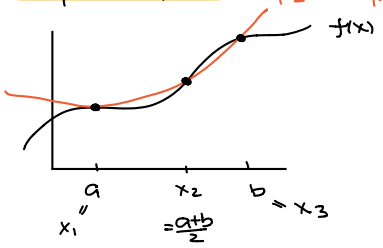
$$= (b-a)\frac{f(a)+f(b)}{2} + \frac{1}{2}f''(\xi)\int_a^b (x-a)(x-b)\,dx$$

$$= -(b-a)^3/6$$

$$= (b-a)\frac{f(a)+f(b)}{2} - \frac{1}{12}f''(\xi)(b-a)^3$$

degree of precision is 1

Simpson's Rule $P_2(x)$ quadratic interpolant



write $f(x) = P_2(x) + E(x)$

↳ use Lagrange basis

$$\int_a^b f(x)\,dx = \int_a^b P_2(x)\,dx + \int_a^b E(x)\,dx$$

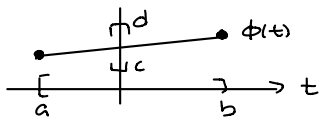
$$\int_a^b f(x)\,dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] - \frac{(b-a)^5}{90\cdot 32}f^{(4)}(\xi)$$

the degree of precision is 3.

Change of interval:

Suppose $\int_a^b f(t)\,dt \approx \sum_{i=0}^{n} w_i f(t_i) = Q(f)$

and we want to approximate $\int_c^d f(x)\,dx$ use a linear map from $[a,b] \to [c,d]$



$$\phi(t) = c + \frac{d-c}{b-a}(t-a)$$

$$\int_c^d f(x)\,dx = \int_a^b f(\phi(t))\cdot\frac{d-c}{b-a}\,dt \approx \frac{d-c}{b-a}\sum_{i=0}^{n} w_i f(\phi(t_i))$$

↑ let $x = \phi(t)$ $dx = \phi'(t)\,dt$

Example: $\int_0^1 f(t)\,dt \approx \underbrace{\frac{1}{2}}_{w_i}(f(\underbrace{0}_{t_0}) + f(\underbrace{1}_{t_1}))$ convert to $[c,d]$

Have $\phi(t) = c + (d-c)t$ $(a=0,\ b=1)$

$$\int_c^d f(x)\,dx = (d-c)\sum_{i=0}^{1} \underbrace{w_i}_{w_i=\frac{1}{2}} f(\phi(t_i))$$

↳ use $\phi(0)=c$, $\phi(1)=d$ for $i=0,1$

$$= \frac{d-c}{2}[f(c) + f(d)]$$

Method of Undetermined Coefficients

$\int_a^b f(x)\,dx \approx Q(f) := w_1 f(x_1) + w_2 f(x_2)$ pick two points x_1, x_2 as the nodes

choose weights so that $Q(f)$ integrates polynomials exactly for heightest possible degree.

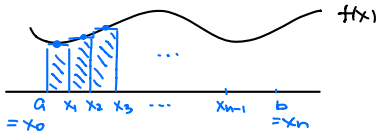
Example: $\int_0^1 f(x)\,dx \approx w_0 f(0) + w_1 f(1)$ choose endpoints for example.

Want quadrature to be exact for $f(x)=1$ and $f(x)=x$

$$\Rightarrow \int_0^1 1\,dx = 1 = w_0 \cdot 1 + w_1 \cdot 1 = Q(1)$$
$$\int_0^1 x\,dx = \tfrac{1}{2} = w_0 \cdot 0 + w_1 \cdot 1$$
$$\Rightarrow w_1 = \tfrac{1}{2},\ w_0 = \tfrac{1}{2}$$

Get $\int_0^1 f(x)\,dx \approx \frac{1}{2}[f(0)+f(1)]$ (trapezoid rule)

Composite Midpoint Rule



$n+1$ nodes $\{x_i\}$ n subintervals

let $h = \frac{b-a}{n}$ = constant subinterval width

let $m_i = \frac{x_i + x_{i+1}}{2}$ (midpoint)

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = h f(m_i) + \frac{1}{24} h^3 f''(\xi_i) \quad \text{where } \xi_i \in [x_i, x_{i+1}]$$

so

$$\int_a^b f(x)\,dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)\,dx = h \sum_{i=0}^{n-1} f(m_i) + \frac{1}{24} h^3 \sum_{i=0}^{n-1} f''(\xi_i)$$

By generalized IVT, $\sum_{i=0}^{n-1} f''(\xi_i) = n f''(\xi)$ since have n terms in the sum will all coefficients equal to 1.

error terms is:

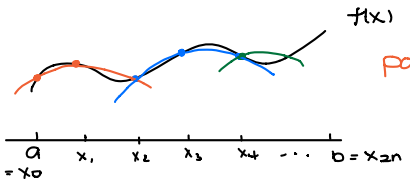
$$\frac{1}{24} h^3 n f''(\xi) = \frac{1}{24} h^2 (b-a) f''(\xi)$$

So $\int_a^b f(x)\,dx = h \sum_{i=0}^{n-1} f(m_i) + \frac{1}{24} h^2 (b-a) f''(\xi)$

Remarks: ① order of method is $O(h^2)$

② This integrates $p \in \mathbb{P}_1$ exactly

Composite Simpson's Rule



parabola interpolants

- need odd # of nodes
- need even # of subintervals
- $2n+1$ nodes, $2n$ subintervals.
- $h = \frac{b-a}{2n}$ = constant width

Apply Simpson's Rule on $[x_{2i}, x_{2i+2}]$ for $i = 0, \dots, n-1$:

$$\int_{x_{2i}}^{x_{2i+2}} f(x)\,dx = \frac{2h}{6}[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})] - \frac{(2h)^5}{90 \cdot 32} f^{(4)}(\xi_i) \quad \text{with } \xi_i \in [x_{2i}, x_{2i+2}]$$

Now sum:

$$\int_a^b f(x)\,dx = \sum_{i=0}^{n-1} \int_{x_{2i}}^{x_{2i+2}} f(x)\,dx$$

$$= \frac{h}{3} \sum_{i=0}^{n-1} [f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})] - \frac{h^5}{90} \underbrace{\sum_{i=0}^{n-1} f^{(4)}(\xi_i)}_{= n f^{(4)}(\xi) \text{ by generalized IVT}}$$

$$= \frac{h}{3}\left[f(a) + f(b) + 2\sum_{i=1}^{n-1} f(x_{2i}) + 4\sum_{i=0}^{n-1} f(x_{2i+1})\right] - \frac{1}{90} h^5 n f^{(4)}(\xi)$$

$= h^4 h n = h^4 \frac{b-a}{2}$

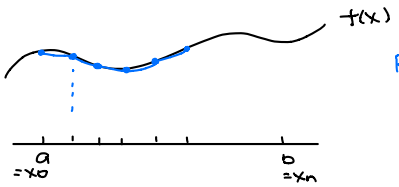
$$= \frac{h}{3}\left[f(a) + f(b) + 2\sum_{i=1}^{n-1} f(x_{2i}) + 4\sum_{i=0}^{n-1} f(x_{2i+1})\right] - \frac{1}{180} h^4 (b-a) f^{(4)}(\xi)$$

Remarks: ① This method is $O(h^4)$

② Method integrates $p \in \mathbb{P}_3$ exactly.

③ Some authors use n subintervals with n even. Here we used $2n$ subintervals.

Composite Trapezoid Rule



piecewise linear interpolation $\quad h = \frac{b-a}{n}$, n = # of intervals.

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{n-1} f(x_i)\right] - \frac{h^2}{12}(b-a) f''(\xi) \qquad x_i = a + ih,\ i = 0, 1, \dots, n.$$

11/17

Weighted Quadrature

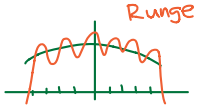
quadrature weights, nodes

$$\int_a^b f(x) w(x)\,dx \approx \sum_{k=1}^{n} A_k f(x_k) = Q(f)$$

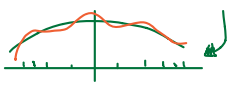
↳ in general can consider integrals with fixed weights.

- Can we choose the nodes and weights so that $Q(f)$ is exact for the highest degree possible polynomial.
- We have now $2n$ degrees of freedom and will be able to integrate $p \in \mathbb{P}_{2n-1}$ exactly. This is acheived by Graussion Quadrature.

Remark: Runge phenomenon.



chebyshev points



Example: $\int_{-1}^{1} f(x)\,dx \approx Q(f) = w_1 f(x_1) + w_2 f(x_2)$ think $\int_{-1}^{1} f(x)\cdot 1\,dx$, weight $w(x) \equiv 1$

Request $Q(f)$ is exact for $\{1, x, x^2, x^3\}$: 4 unknows, 4 equations.

$$w_1 + w_1 = \int_{-1}^{1} 1 = 2$$
$$w_1 x_1 + w_2 x_2 = \int_{-1}^{1} x\,dx = 0 \qquad \Rightarrow \quad w_1 = w_2 = 1$$
$$w_1 x_1^2 + w_2 x_2^2 = \int_{-1}^{1} x^2\,dx = 2/3 \qquad x_1, x_2 = \pm\sqrt{1/3}$$
$$w_1 x_1^3 + w_2 x_2^3 = \int_{-1}^{1} x^3\,dx = 0$$

So $Q(f) = f(\sqrt{\tfrac{1}{3}}) + f(-\sqrt{\tfrac{1}{3}})$

Orthogonal Polynomials

$\{P_n\} \subset \mathbb{P}_n$ is orthogonal with respect to the inner-product $\langle \cdot, \cdot \rangle_w$ on (a, b) if $m \neq n$, have $\langle P_n, P_m \rangle_w = \int_a^b P_n(x) P_m(x) w(x)\,dx = 0$

Theorem: If $\{P_0, P_1, \dots, P_n\}$ is an orthogonal on $[a, b]$, and $\deg(P_i) = i$,

then P_i has exactly i distinct root on (a,b).

Example:

let $\langle f,g\rangle_w = \int_{-1}^{1} f(x)\cdot g(x)\cdot 1\,dx$, weight $w(x)\equiv 1$

An orthogonal set in $\mathbb{P}_2$ is $\{1, x, \frac{3}{2}(x^2-\frac{1}{3})\}$ on $[-1,1]$ with $w(x)\equiv 1$.

these are called the first 3 Legendre polynomials

note
- 0 has 0 roots
- x has 1 roots on $(-1,1)$
- $x^2-\frac{1}{3}$ has two roots $\pm\sqrt{1/3}$ on $(-1,1)$

Gauss-Legendre quadrature (weight $w(x)\equiv 1$ and interval $[-1,1]$)

Fix n. let $\{x_i\}_{i=1}^n$ be the n roots of the n^{th} legendre polynomial.

Interpolate f at x_i:

$f(x) \approx \sum_{i=1}^n l_i(x) f(x_i)$ where $l_i(x) = \prod_{\substack{j=1\\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}$ lagrange basis for interpolation

$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^n f(x_i) \underbrace{\int_{-1}^{1} l_i(x)\,dx}_{=w_i}$ weights for quadrature

$= \sum_{i=1}^n w_i f(x_i)$

$= Q(f)$

Theorem: $Q(f)$ obtained is exact for all $p\in\mathbb{P}_{2n-1}$

Example: $I = \int_{-1}^{1} e^{-x^2}\cdot 1\,dx$, $w(x)\equiv 1$

know degree 2 legendre polynomial is $P_2(x) = \frac{3}{2}(x^2-\frac{1}{3})$

By previous theorem, optimal nodes are the roots of $P_2(x)$ (for 2-node quadrature)

$\Rightarrow x_1, x_2 = \pm\sqrt{1/3}$

So $Q(f) = w_1 f(\sqrt{1/3}) + w_2 f(-\sqrt{1/3})$ and use undetermined coefficients.

Want $Q(f)$ to evaluate $\int_{-1}^{1} f(x)\,dx$ exactly for $f(x) = 1, x$

[two unknowns w_1, w_2, need two functions]

So $\int_{-1}^{1} 1\,dx = 2 = w_1 + w_2$

$\int_{-1}^{1} x\,dx = 0 = w_1\sqrt{1/3} + w_2(-\sqrt{1/3})$

$\Rightarrow w_1 = w_2 = 1$

$Q(f) = f(\sqrt{1/3}) + f(-\sqrt{1/3})$ and this is exact for any $p\in\mathbb{P}_{2n-1} = \mathbb{P}_3$ by Gaussian quad. theorem. ($n=2$)

So $\int_{-1}^{1} e^{-x^2}\,dx \approx e^{-(\sqrt{1/3})^2} + e^{-(-\sqrt{1/3})^2} \approx 1.6929..$ Exact integral is $1.71124...$

Change of interval:

$\int_a^b f(x)\,dx = \int_{-1}^{1} \underbrace{f\left(\frac{a+b}{2} + \frac{b-a}{2}t\right)}_{=g(t)} dt \cdot \frac{b-a}{2}$

$= \int_{-1}^{1} g(t)\,dt \cdot \frac{b-a}{2}$

then can apply Gauss-Legendre quadrature to g.

Quick way to obtain some common orthogonal polynomials.

without resorting to Gram-Schmidt.

Case: Gauss-Legendre for $\int_{-1}^{1} f(x)\cdot 1\,dx$, $w(x)\equiv 1$ on $[-1,1]$

Recursion relation to generate orthogonal polynomials on $[-1,1]$ with $w(x)\equiv 1$

- $P_{-1}(x) \equiv 0$
- $P_0(x) \equiv 1$
- :

$(j+1)P_{j+1} = (2j+1)xP_j - jP_{j-1}$

$j=0$: $P_1 = xP_0 = x\cdot 1 = x$

$j=1$: $2P_2 = 3xP_1 - P_0 = 3x^2 - 1$

$P_2(x) = \frac{1}{2}(3x^2-1)$

So $\{1, x, \frac{1}{2}(3x^2-1)\}$ are the first 3 Legendre polynomial.

In general, the Gaussian quadrature optimal node theorem remains valid for other orthogonal polynomials on different intervals or with different weights.

$\int_a^b f(x)w(x)\,dx \approx \sum_{k=1}^{n} A_k f(x_k)$

$w(x)$: fixed weight

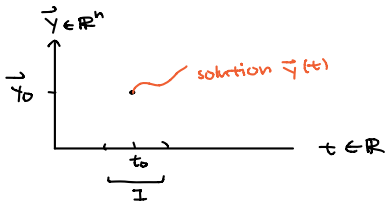
and to find x_k, would use orthogonal polynomials on $[a,b]$ w.r.t $w(x)$.

11/19 **Numerical Method for ODE**

Initial Value Problem (IVP)

- 1st order system $\begin{cases} \vec{y}'(t) = \vec{f}(t, y(t)) \\ \vec{y}(t_0) = \vec{y}_0 \end{cases}$ $\forall\, t \in I \subseteq \mathbb{R}$

$\vec{y}: I \to \mathbb{R}^n$, $\vec{f}: D \subset \mathbb{R}^{n+1} \to \mathbb{R}^n$ $(n \geq 1)$



- Higher order ODE can always be reduced to a 1st order system.

Example: $y'' = (y')^2 + y$

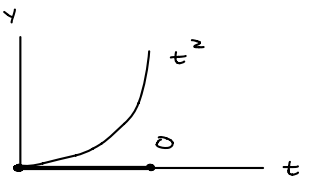
let $y_1 = y \Rightarrow y_1' = y' = y_2$

$y_2 = y' \quad y_2' = y'' = (y')^2 + y = (y_2)^2 + y_1$

So get the 1st order system for $\vec{y}(t) = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}(t)$, $\frac{d}{dt}\vec{y}(t) = \vec{f}(t, \vec{y}(t)) = \begin{pmatrix} y_2 \\ (y_2)^2 + y_1 \end{pmatrix}(t)$

Potential Issues.

1. non-uniqueness of solutions: $\begin{cases} y'(t) = 2\sqrt{|y(t)|} \\ y(0) = 0 \end{cases}$

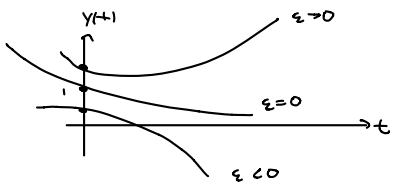


∃ two solutions: $y \equiv 0$ $(t>0)$

$y(t) = t^2$

2. non-existence: $|y'(t)| = -1$ no solutions to this ODE.

3. Instability: $\begin{cases} y'(t) = 100y - 101e^{-t} \\ y(0) = 1 + \varepsilon \end{cases}$ think ε mach size.



solution is $y(t) = e^{-t} + \varepsilon e^{100t}$

- If we make a small ε error in $y(0)$, our new solution no longer decays toward 0!
- We say the ODE is unstable with respect to perturbations in the IC.

Existence & Uniqueness of IVP

Recall $\vec{f}$ is Lipschitz w.r.t $\vec{y}$ if $\exists\, k \in \mathbb{R}$ s.t $\forall t \in I$

$\|\vec{f}(t, \vec{y}_1) - \vec{f}(t, \vec{y}_2)\| \leq k\|\vec{y}_1 - \vec{y}_2\|$ Can replace $\|\cdot\|$ with $|\cdot|$

Remark: If $\vec{f} \in C^1$, then $\vec{f}$ is Lipschitz provided $\|\frac{\partial \vec{f}}{\partial \vec{y}}\| \leq k$

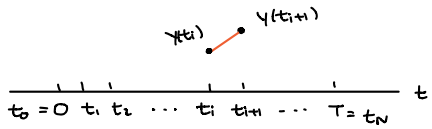
where $\frac{\partial \vec{f}}{\partial \vec{y}}$ is the Jacobian matrix $\left[\frac{\partial \vec{f}}{\partial \vec{y}}\right]_{ij} = \frac{\partial f_i}{\partial y_j}$

Theorem: let $\vec{f}$ be Lipschitz in $\vec{y}$ near the point $(t_0, \vec{y}_0)$ and suppose $\vec{f}$ is continuous, w.r.t t, then $\exists \varepsilon > 0$ and $\exists! \vec{y}(t)$ s.t $\begin{cases} \vec{y}'(t) = \vec{f}(t, \vec{y}(t)) \\ \vec{y}(t_0) = \vec{y}_0 \end{cases}$ for $t_0 - \varepsilon < t < t_0 + \varepsilon$

One-step method

$\begin{cases} y' = f(t, y(t)) \\ y(0) = y_0 \end{cases}$ $t \in (0, T)$

$\begin{cases} N+1 & \text{nodes (mesh points)} \\ N & \text{intervals} \end{cases}$



Only one step is used to approximate $y(t_{i+1})$

in general mesh not uniform: $h_i = t_i - t_{i-1}$ (step size) $i = 1, 2, \dots, N$

Forward Euler Method:

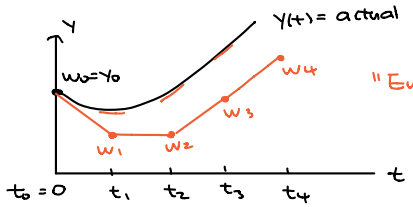
suppose $h_i = h =$ constant $\forall i$, let $\{w_i\}_{i=0}^{N}$ where $w_i \approx y(t_i)$ (approximate value; exact solution at $t = t_i$)

For small h, $y'(t) \approx \frac{y(t+h) - y(t)}{h}$

$y(t+h) \approx y(t) + \underbrace{h y'(t)}_{= f} = y(t) + h f(t, y(t))$

so $\underbrace{y(t_i+h)}_{\approx w_{i+1}} \approx \underbrace{y(t_i)}_{\approx w_i} + h f(t_i, \underbrace{y(t_i)}_{\approx w_i}) \Rightarrow \begin{cases} w_{i+1} = w_i + h f(t_i, w_i) \\ w_0 = y_0 \end{cases}$ $\forall i = 0, \dots, N-1$

Geometrically,



"Euler polygonal" our approximate solution connected with stright lines.

this method is explicit, w_{i+1} is given explicitly in terms of other values (w_i)

Numerical Stability

large h small h

Consider $\begin{cases} y' = \eta y \\ y(0) = y_0 \end{cases}$ ("Dahlquist test equation")

Solution is $y(t) = y_0 e^{\eta t}$ and for $\eta < 0$ expect decay as $t \to \infty$

so request that $\{w_i\} \to 0$ as $i \to \infty$

Check Forward Euler:

$w_{i+1} = w_i + \underbrace{h f(t_i, w_i)}_{\eta w_i} = w_i + h\eta w_i = (1 + h\eta) w_i = (1+h\eta)^2 w_{i-1} = \dots = (1+h\eta)^{i+1} w_0 = \alpha^{i+1} w_0$

magnification factor $\alpha^{i+1} := (1 + h\eta)$

For $\{w_i\} \to 0$ as $i \to \infty$ need $\alpha^{i+1} \to 0$ as $i \to \infty$ iff $|\alpha| < 1$

So need $|1 + h\eta| < 1$ for stability

$-1 < 1 + h\eta < 1 \Rightarrow -2/\eta > h > 0$ $(\eta < 0)$

$\Rightarrow$ stability for forward Euler requires $0 < h < -2/\eta$

if $h > -2/\eta$ blow up.

if $h = -2/\eta$ WWW

Note: ① If $\eta = -10$, $y' = -10y$, $0 < h < 1/5$ for stability.

② If need size restriction on h, we call the method conditionally stable.

11/24

Euler Methods : Part II.

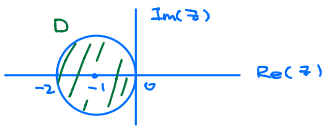
Forward Euler $\begin{cases} w_{i+1} = w_i + hf(t_i, w_i) \\ w_0 = y_0 \end{cases}$ for solving $\begin{cases} y' = f(t,y) \\ y(0) = y_0 \end{cases}$ with $w_i \approx y(t_i)$

this method requires small h for numerical stability: $\begin{cases} y' = \lambda y \\ y(0) = y_0 \end{cases}$ (test equation, $\lambda < 0$)

need $|1 + \lambda h| < 1$ for stability $\Rightarrow$ $0 < h < -\frac{2}{\lambda}$

Definition: let $\lambda < 0$ and $y' = \lambda y$ be the "test equation" (complex numbers)
The linear stability domain of a numerical method is the set $z := \lambda h \in D \subseteq \mathbb{C}$
s.t the numerical solution of $y' = \lambda y$ decays.

Ex. For Forward Euler, $D = \{z \in \mathbb{C} \mid |1+z| < 1\}$ ($|z-(-1)| < 1$)



Definition: If $D = \{z \in \mathbb{C} \mid Re(z) < 0\}$ we call the method absolutely stable, or A-stable.
the point is A-stable method have no restriction on h.

Forward Euler for systems. $\begin{cases} \vec{y}'(t) = \vec{f}(t, \vec{y}(t)) \\ \vec{y}(0) = \vec{y}_0 \end{cases}$

can apply Forward Euler componentwise $\begin{cases} w_{i+1}^{(n)} = w_i^{(n)} + h f^{(n)}(t_i, w_i^{(n)}) \\ w_0^{(n)} = y_0^{(n)} \end{cases}$

denote n^{th} components of a vector; i denotes the i^{th} iteration.

Error Analysis and Convergence.
- rounding error : due to computer finite precision. - very difficult to analyze in ODE.
- truncation error : due to numerical discretization $y' = f(t,y)$ Euler method.
  - local truncation error : Error made in one-step of the method.
  - global truncation error : the accumulation of all local errors on a time interval $[0, T]$

Local error for Forward Euler. $w_{i+1} = w_i + hf(t_i, w_i)$
replace w_i with $y(t_i)$ to get $y(t_{i+1}) \stackrel{?}{=} y(t_i) + hf(t_i, y(t_i))$
have $t_{i+1} = t_i + h$ $\rightarrow$ $y(t_i + h) - [y(t_i) + hf(t_i, y(t_i))]$
$= [y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(\xi)] - [y(t_i) + hf(t_i, y(t_i))]$
$y'(t_i) = f(t_i, y(t_i))$
$= \frac{h^2}{2} y''(\xi) = O(h^2)$ local truncation error for forward Euler

Global error for Forward Euler $\begin{cases} y' = f(t, y(t)) \\ y(a) = y_a \end{cases}$ $t \in (a, b)$

Theorem: $g_i := |w_i - y(t_i)| \leq \frac{Mh}{2L}(e^{L(t_i - a)} - 1)$
where $|y''(t)| \leq M$ on $[a,b]$ and a is a Lipschitz constant for f in the y-variable.

Remarks: ① As $h \to 0$, have $g_i \to 0$ $\forall i$, so forward Euler converges.
② This is an $O(h)$ method (1st order method)
③ Note local error is $O(h^2)$ but global error is $O(h)$ let $h = \frac{t_{final} - t_0}{N}$
Have N steps and each time get $O(h^2)$ error:
global error $\approx O(h^2)N = O(h^2)\frac{t_{final} - t_0}{h} = O(h)(t_{final} - t_0)$

Backward Euler Method $\begin{cases} y' = f(t,y) \\ y(0) = y_0 \end{cases}$

$y'(t) \approx \underbrace{\frac{y(t) - y(t-h)}{h}}_{\text{bkwd difference quotient}} \Rightarrow y(t) \approx y(t-h) + hy'(t) = y(t-h) + hf(t, y(t))$

let $w_i \approx y(t_i)$ to get $\begin{cases} w_{i+1} = w_i + hf(t_{i+1}, w_{i+1}) \quad \text{(replaced } t \text{ with } t_{i+1}) \\ w_0 = y_0 \end{cases}$

- this is an implicit method, need to solve $F(z) := z - [w_i + hf(t_{i+1}, z)] = 0$ for z.
  then update $w_i \to z = w_{i+1}$
- can use bisection, Newton, fzero ...
  preferred ☺ slow ☹

Local truncation error.

$y(t_{i+1}) - [y(t_i) + h\boxed{f(t_{i+1}, y(t_{i+1}))}]$ $= y'(t_{i+1})$ from the ODE

$= y(t_{i+1}) - y(t_i) - hy'(t_{i+1})$

$= y(t_{i+1}) - y(t_{i+1} - h) - hy'(t_{i+1})$

$= \cancel{y(t_{i+1})} - [\cancel{y(t_{i+1})} - \cancel{hy'(t_{i+1})} + \frac{h^2}{2}y''(\xi)] - \cancel{hy'(t_{i+1})}$

$= -\frac{h^2}{2}y''(\xi)$

- local error is $O(h^2)$ $\Rightarrow$ global error is $O(h)$, 1st order method

Stability of backward Euler

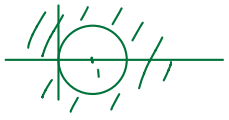
test equation $\begin{cases} y' = \lambda y = f(t,y) \quad \text{with } \lambda < 1 \\ y(0) = 1 \end{cases}$

Apply bkwd Euler: $\begin{cases} w_{i+1} = w_i + hf(t_{i+1}, w_{i+1}) = w_i + h\lambda w_{i+1} \\ w_0 = 1 \end{cases}$

$\Rightarrow w_{i+1} = \frac{1}{1-\lambda h} w_i = \cdots = \left(\frac{1}{1-\lambda h}\right)^i w_0$

require $\left|\frac{1}{1-\lambda h}\right| < 1$ for stability, but this holds for any $h > 0$ (since $\lambda < 0$ too)
so no restriction on h! Backward Euler is a A-stable method.

Remarks: ① Bkwd Euler is A-stable, forward Euler is not!
② Both method are $O(h)$, 1st order methods.
③ For bkwd Euler, gain the stability in exchange for solving nonlinear equation.
④ Forward Euler is explicit (easy to get w_{i+1})
⑤ Stability region for bkwd Euler:



$z = \lambda h \in \mathbb{C}$
← the set $|1 - z| > 1$
contains entire left plane $\Rightarrow$ A-stable.

More ODE methods.

ODE $\begin{cases} y' = f(t, y(t)) \\ y(0) = y_0 \end{cases}$

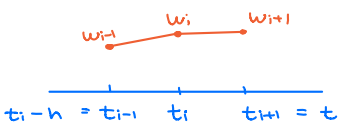
so far we seen - forward Euler (explicit , not A-stable)
- Backward Euler (implicit , A-stable)

Centered differencing $y'(t) = \frac{y(t+h) - y(t-h)}{2h} + O(h^2)$ 2nd order accurate

- Fwd / Bkwd Euler , would have $O(h)$ error

plug into ODE : $[y(t+h) - y(t-h)]/2h \approx f(t, y(t))$

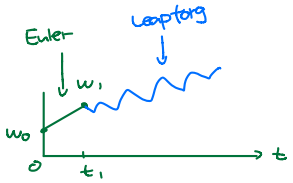
$y(t+h) \approx y(t-h) + 2h f(t, y(t))$



$w_i \approx y(t_i)$, h = constant stepsize in time

Leapfrog method → $\begin{cases} w_{i+1} = w_{i-1} + 2h f(t_i, w_i) \\ w_0 = y_0 \ , \ w_1 = y_1 \end{cases}$

Usually obtain a second point (in order to get Leapfrog started) by using a single forward Euler step (or something else)



Remarks : ① Method is explicit , two-step . (multistep methods)
② Local truncation error is $O(h^3)$
⇒ Global error is $O(h^2)$ so the method is 2nd order .

Stability : test eqn $\begin{cases} y' = \lambda y = f(t, y) \\ y(0) = y_0 \end{cases}$ $(\lambda < 0)$ actual solution is $y(t) = y_0 e^{\lambda t}$ which decays.

$w_{i+1} = w_{i-1} + 2h f(t_i, w_i)$
$= w_{i-1} + 2h(\lambda w_i)$

to solve , assume $w_i = r^i$ for some $r \in \mathbb{R}$ (or $\mathbb{C}$)

$r^{i+1} = r^{i-1} + 2h\lambda r^i$

$r = r^{-1} + 2h\lambda$ rewrite use quadratic formula

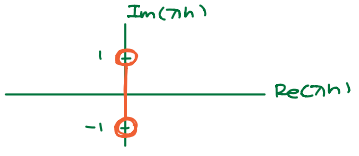
$r = r_\pm = -h\lambda \pm \sqrt{1 + (h\lambda)^2}$

Get the general solution is $w_i = c_1 r_+^i + c_2 r_-^i$ took a linear combination,
need to check that w_i decays as $i \to \infty$ c_1 , c_2 arbitrary constant.

Have $r_+ = \underbrace{-h\lambda}_{>0} + \sqrt{\underbrace{1 + (h\lambda)^2}_{>1}} > 0 + 1 = 1$, $r_+^i \to \infty$ as $i \to \infty$ so w_i blows up !

so Leapfrog is unstable for all $h > 0$ when solving $y' = f(t, y)$

Remark : ④ For systems , $\vec{y}' = A\vec{y}$.
when A has complex eigenvalues λ.
Leapfrog can be useful .



Stability region for Leapfrog

Method based on Quadrature $\begin{cases} y' = f(t, y) \\ y(0) = y_0 \end{cases}$

$h = t_{i+1} - t_i$ = constant

$\int_{t_i}^{t_{i+1}} y(t)\,dt = \int_{t_i}^{t_{i+1}} f(t, y(t))\,dt$

$y(t_{i+1}) - y(t_i)$ (by FTC) can apply any quadrature rule,

here we use Trapezoid Rule: $\frac{1}{2}h\left[f(t_i, y(t_i)) + f(t_{i+1}, y(t_{i+1}))\right]$

$$y(t_{i+1}) \approx y(t_i) + \frac{1}{2}h\left[f(t_i, y(t_i)) + f(t_{i+1}, y(t_{i+1}))\right]$$

$\Rightarrow$ Trapezoid method $\quad w_{i+1} = w_i + \frac{1}{2}h\left[f(t_i, w_i) + f(t_{i+1}, w_{i+1})\right]$

$$w_0 = y_0$$

Remarks : ① Method is implicit, one-step

② Local truncation error is $O(h^3)$, global error is $O(h^2)$

Stability : $y' = \lambda y$, $\lambda < 0$

$$w_{i+1} = \frac{1 + \lambda h/2}{1 - \lambda h/2} w_i \qquad \text{want } \left|\frac{1 + \lambda h/2}{1 - \lambda h/2}\right| < 1, \text{ holds } \forall\, h > 0.$$

the method is A-stable. no h restriction

Explicit Trapezoid (Heun's method, Improved Euler)

$$\begin{cases} w_{i+1} = w_i + \frac{1}{2}h\left[f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i))\right] \\ w_0 = y_0 \end{cases}$$

replaced w_{i+1} in Trapezoid w/ a Forward Euler-

Remarks : ① Method is $O(h^2)$ in global error, explicit, one-step.

② For stability, need $0 < h < -2/\lambda$

from checking the test equation $y' = \lambda y$ ($\lambda < 0$)

$\Rightarrow$ conditionally stable

③ This is a "predictor-corrector" method.

Euler predicts w_{i+1}, then trapezoid corrects it to a new w_{i+1}.

1 2 | 3

ode45 : 4^{th} order method, 5^{th} order method

↳ "Runge-Kutta methods"

$e_i \approx |z_{i+1} - y_{i+1}|$ ↳ obtained from 5^{th} order method

↳ obtained from 4^{th} order method

If e_i is small enough. ode45 accepts the time stepsize and continues onto next y-value.

ode15s is a stiff ODE solver

stiff ODE

$y' = \lambda y \quad (\lambda < 0)$

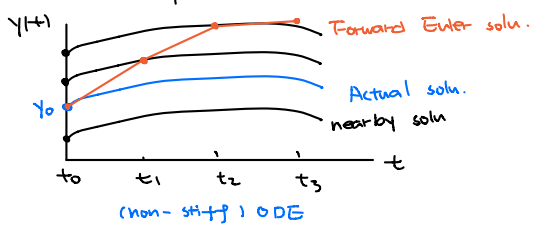
For forward Euler, need $0 < h < -2/\lambda$ for stability.

$\ll 1$ if λ large and negative

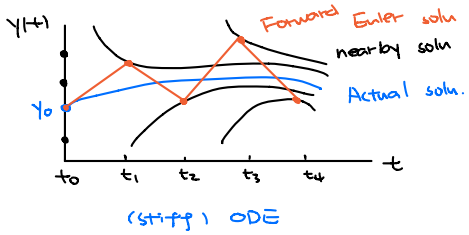
Definition : An ODE is stiff if explicit ODE solver require very small time steps for stability.

- For $y' = y^2 - y^3$; $y(0) = 10^{-6}$, ode45 requires 1,000,000 time steps while ode15s only requires $\approx$ 100 timesteps.

Graphically : $y' = f(t, y)$



(non-stiff) ODE



(stiff) ODE

note: $y' = \lambda y$ ($\lambda < 0$) implicit methods typically are A-stable methods.
no restriction on h

Runge - Kutta methods

1- stage RK method (explicit, 1- step, 1st order)

$w_{i+1} = w_i + a s_1$ we assume the iteration has this form (ansatz)

$s_1 = h f(t_i, w_i)$ "stage" of the RK method

Q: Can we find $a \in \mathbb{R}$ so the method has the best possible (highest order) truncation error?

Have $w_{i+1} = w_i + ahf(t_i, w_i)$

$y(t_{i+1}) - [y(t_i) + ah f(t_i, y(t_i))]$

$y(t_i) + hy'(t_i) + O(h^2) - y(t_i) - ahy'(t_i) = O(h^2)$

↑ cancel if $a = 1$ ↑

$y(t_{i+1}) = y(t_i) + hy'(t_i) + O(h^2)$ taylor exp.

$f(t_i, y(t_i)) = y'(t_i)$ from the ODE

$t_{i+1} = t_i + h$, h constant

so $a = 1$ and we get

$w_{i+1} = w_i + hf(t_i, w_i)$ Forward Euler.

2- stage Runge - Kutta method (explicit, 1- step method, 2nd order)

Assumed form of the iteration scheme:

$w_{i+1} = w_i + a s_1 + b s_2$

$s_1 = h f(t_i, w_i)$

$s_2 = h f(t_i + \alpha h, w_i + \beta s_1)$

] "stages"

- unknowns are a, b, α, β
- can we choose these so the truncation error order is as high as possible?
  - Yes. Get $O(h^3)$ truncation error.
- stripping the details we compute

$y(t_{i+1}) - [y(t_i) + a s_1 + b s_2]$

and Taylor expand a lot to find

$1 - a - b = 0$

$\frac{1}{2} - \alpha b = 0$

$\frac{1}{2} - b\beta = 0$

- ∞ solutions since 3 eqns, 4 unknowns.

A natural choice is $a = b = \frac{1}{2}$, $\alpha = \beta = 1$ (for symmetry)

Get

$w_{i+1} = w_i + \frac{1}{2}(s_1 + s_2)$

$s_1 = hf(t_i, w_i)$

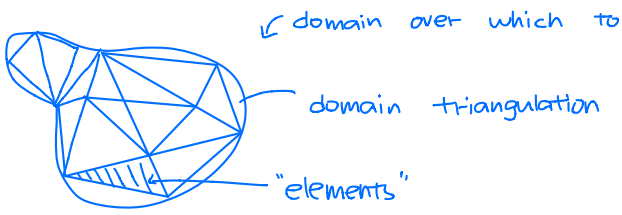
$s_2 = hf(t_i + h, w_i + s_1)$

this is explicit trapzoid or improved Euler.

12/8

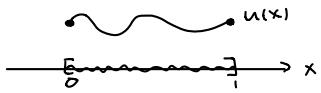
Crash Course in Finite Elements

2D example



FEM for ODE: ① form the variational (weak) formulation of ODE
② project the infinite-dimensional problem to a finite-dim subspace.
③ From the finite-dimensional problem obtain a system of equations that can be solved to get an approximate solution

ODE (boundary-value problem):
$$\begin{cases} (-pu')' + qu = f & \text{on } (0,1) \subset \mathbb{R} \\ u(0) = \alpha, \quad (pu')(1) = \beta \end{cases}$$
↑ the classical or strong form of ODE



p.q.f functions
α, β constants

Let's convert to weak formulation:

Multiply ODE by $v \in C^\infty([0,1])$ and intergate by parts

v is arbitrary called a test function

$$\int_0^1 (-pu')'v + \int_0^1 quv = \int_0^1 fv$$

$$\int_0^1 (-pu')'v = -pu'v\Big|_0^1 - \int_0^1 (-pu')v' \quad \text{(integrated by parts)}$$
$$= -\underbrace{p(1)u'(1)}_{=\beta}v(1) + p(0)\underbrace{u'(0)}_{\text{not known but can choose } v(0)=0}v(0) + \int_0^1 (pu')v'\,dx.$$

from the boundary condition

so this term disappears.

$$\Rightarrow \int_0^1 pu'v' + quv = \int_0^1 fv\,dx + \beta v(1)$$

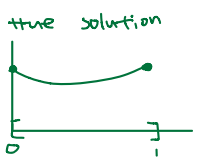
this is the variational (weak) form of the ODE

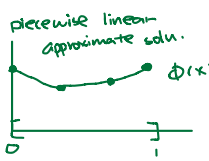
- variational since v can vary, it's arbitrary.
- weak since we can accept weaker conditions on u, no longer need u'' to exist.

New problem asks to find $u \in V$ (V: some function space)
s.t the weak form holds for all $v \in C^\infty([0,1])$. s.t. $v(0)=0$

Remarks: ① If u is a classical solution, it is also a weak solution.
And if u is a weak solution, s.t also $u \in C^2(0,1)$, then it is also a classical solution that satisfies the original ODE.

② For classical ODE, need u to be twice-differentiable.
For the weak form, one derivative is enough.
Even more so, we can allow u' to be undefined at a finite number of points. (Since we're integrating and pointwise values don't affect the integral)

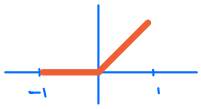




note derivative is defined everywhere except a finite # of points.
this function $\phi(x)$ won't satisfy the classical form but will satisfy the weak form.

Example: $y'(t) = H(t)$ on $(-1, 1)$, Heaviside function $H(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}$

- Suppose $y(t)$ is continuous
- we expect our solution to be



$y(t) = \begin{cases} 0 & t < 0 \\ t & t \geq 0 \end{cases}$ this is the solution satisfying $y(-1) = 0$.

but $y(t)$ not differentiable at the origin.

- So $y(t)$ does not satisfy the classical form of the ODE but will instead satisfy the integral (weak) form.

check it: weak form:

let v be a smooth test function s.t $v(-1) = v(1) = 0$.

Multiply ODE by v and integrate by parts.

$\int_{-1}^{1} y'v = \int_{-1}^{1} Hv$

$\int_{-1}^{1} y'v = yv\Big|_{-1}^{1} - \int_{-1}^{1} yv'$

$= y(1)v(1)^{0} - y(-1)v(-1)^{0} - \int_{-1}^{1} yv'$

$-\int_{-1}^{1} yv' = \int_{-1}^{1} Hv$

to check $y(t) = \begin{cases} 0 & t < 0 \\ t & t \geq 0 \end{cases}$ satisfies $-\int_{-1}^{1} yv' = \int_{-1}^{1} Hv$ the weak form.

$LHS = -\int_{-1}^{1} yv' = -\int_{0}^{1} tv'(t)\,dt = -tv(t)\Big|_{0}^{1} + \int_{0}^{1} 1 \cdot v(t)\,dt = \int_{0}^{1} v(t)\,dt = \int_{-1}^{1} H(t)v(t)\,dt = RHS$

FEM.

ODE: $\begin{cases} -(pu')' + qu = f & \text{on } (0,1) \\ u(0) = \alpha, \quad (pu')(1) = \beta \end{cases}$

weak form: $\underbrace{\int_0^1 pu'v' + quv}_{=: B(u,v)} = \underbrace{\int_0^1 fv\,dx + \beta v(1)}_{=: L(v)}$

$B(u,v)$ since LHS is bilinear (linear in u, v separately)

$L(v)$ since RHS is linear in v

Problem:

find $u \in V$ s.t. $B(u,v) = L(v)$ for all $v \in C^{\infty}$ s.t $v(0) = 0$

V is infinite dimensional function space (think continuous functions or polynomials)

Idea for getting to FEM:

Replace V with V_n a finite-dimensional subspace, and try to solve the following problem:

Find $u \in V_n$ s.t $B(u,v) = L(v)$ holds for all $v \in V_n$ s.t $v(0) = 0$.

the choice of V_n dictates the type of approximation for u.

$V_n = \mathbb{P}_n$ → polynomial u

V_n = trig functions. → trig solution u (spectral method)

For finite elements, there is another choice for V_n