# PRECURSOR KNOWLEDGE

- Precision Error
  ↳ computers have a limit to digits stored
- Algorithmic Error
  ↳ approximations made in algorithm
- Round Off Error
  ↳

# NOTATION

★ root: $f(x)=0$

minimizer: $f'(x)=0$

- $x \in (a,b) \rightarrow a < x < b$

  $x \in [a,b] \rightarrow a \leq x \leq b$

- "Big Oh" Notation summerizes error terms

↳ $F(h) = L + O(G(h))$

$G(h)$: core error eq.

$O$: collects +/- & constants

ex: $\alpha_n = 42 - \frac{12}{n^3} - \frac{3}{n^5} + \frac{6}{n^7}$

$= 42 + O\left(\frac{1}{n^3}\right)$

# MEAN VALUE THEOREM

- let $f \in C^n[a,b]$

$$\frac{d^{n-1}}{dx^{n-1}} f(a) = A, \quad \frac{d^{n}}{dx^{n-1}} f(b) = B$$

then there exists $c \in [a,b]$ s.t.

$$\frac{d}{dx^n} f(c) = \frac{B-A}{b-a}$$

# RIEMANN SUM OF INTEGRALS

$$\int_a^b f(x)\,dx = \lim_{n\to\infty} \frac{b-a}{n} \sum_{i=1}^{n} f(x_i) = a + \frac{i(b-a)}{n}$$

## TAYLOR'S THM

- let $f \in C^{n+1}[a,b]$ & $x \in (a,b)$

$$P_n(x) = f(\bar{x}) + \sum_{k=1}^{n} \frac{1}{k!} f^k(\bar{x})(x-\bar{x})^k$$

then there exists for any $c \in (a,b)$ s.t.

$$|f(x) - P_n(x)| = \frac{1}{n+1} f^{(n+1)}(c)(x-\bar{x})^{(n+1)}$$

- COROLLARY:

let $M = \max\{|f^{n+1}(c)| : x \in (a,b)\}$

then $|f(x) - P_n(x)| = \frac{1}{(n+1)!} M (b-a)^{n+1} \to 0$ as $n \to \infty$

## ABSOLUTE (AE) & RELATIVE ERROR

$$AE = |p - p^*| \quad , \quad RE = \frac{|p-p^*|}{|p^*|}$$

- Used to determine quality of algorithm
  - ↳ AE < Tolerance (used if $p^* = 0$)
  - ↳ RE < 5E-t for t significant digits

# RATES OF CONVERGENCE

- $\lim_{n\to\infty} |\alpha_n - \alpha| = 0$ must be true

- $p^k$ is:

  counter NOT power

| | |
|---|---|
| linearly converg. | $\lvert p^{k+1} - p \rvert \le \rho \lvert p^k - p^* \rvert$ |
| super linear " " | $\lvert p^{k+1} - p \rvert \le \rho^k \lvert p^k - p^* \rvert$ |
| quadratically " " | $\lvert p^{k+1} - p \rvert \le M \lvert p^k - p^* \rvert^2$ |

- testing for convergence

① $\lim_{k\to\infty} \dfrac{|p^{k+1} - p^*|}{|p^k - p^*|}$ → $\geq 1$, sublinear; → $< 1$, linear; → $0$, ②

② $\lim_{k\to\infty} \dfrac{|p^{k+1} - p^*|}{|p^k - p^*|^2}$ → exists, quad.; → $\infty$, super-lin.

- don't forget:

  # of iterations is not necessarily # of evaluations

  ↳ depends on each method

# SOLVING NONLINEAR EQ.

## I. BISECTION:

- Algorithm checks $f(a^k), f(b^k)$ with desired $f(x^*)$ answer to slowly converge to it using

$$c = \frac{a^k + b^k}{2} \approx x^*$$

where if $f(a^k) < 0$ & $f(b^k) > 0$ and $f(c) < 0$ then $a^{k+1} = c$

- CONVERG:

$$|p_n - p^*| \le \frac{(b-a)}{2^n}, \text{ lin.}$$

Ensure MVT

## IA. REGULA-FALSI

$$\hat{x} = a - \frac{f(a)(b-a)}{f(b)-f(a)}$$

- same solving philosophy of bisection w/ a faster method of determining c
  - ↳ $f(a)f(b) < 0 \rightarrow b = c$
  - " " $> 0 \rightarrow a = c$
- fast if intervals are tight and good
  - ↳ linearly converg.

# II. FIXED Pt METHOD

- guess a form of $f(x)$ w/ $g(x)$ s.t.:
  - ↳ $g(x^k) = x^{k+1} \rightarrow g(x^*) = x^*$
  - *ex: $f(x) = x^2 - x - 2 = 0$, $g(x) = x = x^2 - 2$ OR $\sqrt{x+2}$
  - ↳ $|g'(x)| < 1 \rightarrow$ attractor, converg. possible
    - $\geq 1 \rightarrow$ repellor, likely to fail
  - ↳ Test: 1 confirm if fixed pt exists
    - 2 check convergence

- CONVERG: (linear)

$$|p - p_n| \leq \frac{k^n}{1-k} |p_{n+1} - p_n|,$$

↳ smaller $k \rightarrow$ faster converg.

$$k = \max(|g'(x)| : x \in [a,b])$$

# IIA. NEWTON-RAPHSON

- extension of II, follows same restrictions
  iterate until $f(p_k) = p_k$
- $0 = f(p_0) + f'(p_0)(x - p_0) \Rightarrow p^k = p^{k-1} - \dfrac{f(p^{k-1})}{f'(p^{k-1})}$

  where $f(p^k) = p^{k+1}$

check this to make sure it works

- CONVERG. PROOF: (Quadratic)

let $g(x) = x - \frac{f(x)}{f'(x)}$

if $|g'(x)| < 1$ then

$$g'(x) = 1 - \left( \frac{f'(x)f'(x) - f(x)f''(x)}{(f'(x))^2} \right)$$

for $f \in C^2$, $f'(p^*) \neq 0$, $f(p^*) = 0$

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

$$= 1 - 1 + 0$$

$$= 0$$

## II B. SECANT

- uses an approximation of

$$f'(x) \approx \frac{f(p^k) - f(p^{k-1})}{p^k - p^{k-1}}$$

↳ super linear converg.

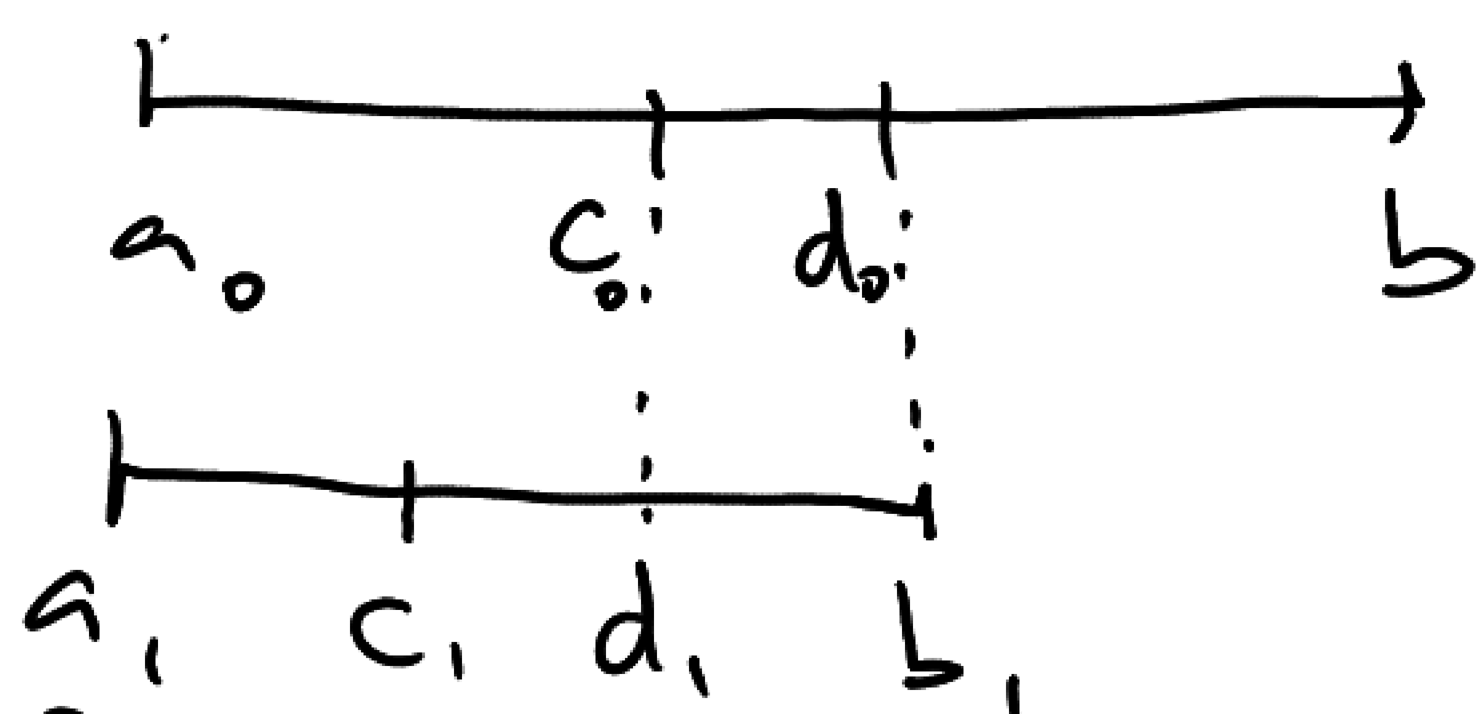
# III. Derivative Free Methods

## III A. Dichotomous

- similar to bisection, finds $c = c \pm \varepsilon$ where $c \in (a, b)$

- $|p_k - p^*| \le \frac{1}{2^k}|b-a| + \varepsilon$ ← can exclude $\varepsilon$ if $\varepsilon < tol$

## III B. Golden Ratio




- $k+1$ eval.
- searches for 2 'c' values instead of 1 like bisection where now $c = (c \pm d)$
  ↑ small diff.

PROOF:

Interval $[x_0, x_1]$, $I_k = b_k - a_k$

$$I_{k+1} = b_{k+1} - a_{k+1} = d_k - a_k$$

$$I_{k+2} = b_{k+2} - a_{k+2} = d_{k+1} - a_k$$

Assuming all intervals are equally spaced:

$$I_k = I_{k+1} + I_{k+2}, \quad \frac{I_{k+1}}{I_k} = \frac{I_{k+2}}{I_k} = K$$

then:

$$1 = \frac{I_{k+1}}{I_k} + \frac{I_{k+2}}{I_k} = \frac{I_{k+1}}{I_k} + \frac{I_{k+2}}{I_{k+1}} \cdot \frac{I_{k+1}}{I_k}$$

$$= K + K^2$$

$$\therefore K = \frac{-1 \pm \sqrt{5}}{2} = \frac{-1 + \sqrt{5}}{2}$$

# POLYNOMIAL INTERPOLATION

## I. LAGRANGE

- $P_n(x) = \sum_{i=0}^{n} L_{n,i}(x) \cdot y_i$

$$L_{n,i}(x) = \prod_{j=0}^{j=n} \frac{(X - X_n)}{(X_i - X_n)}$$

Polynomial extends as far as n-1 for n pts

↳ in essence, every polynomial part contains every "x contribution" except that of focus, $X_i$

↳ the "length" of $L_{n,i}(x)$ depends on the # of pts provided

i.e. 4 pts → 3 deg. poly. produced

- ex: 2nd Order $L(x)$

(-1, -6) (-2, -13), (1, 2)

$$L_{n,0} = \frac{(x+2)(x-1)}{(-1+2)(-1-1)}$$

$$L_{n,1} = \frac{(x+1)(x-1)}{(-2+1)(-2-1)}$$

$$L_{n,2} = \frac{(x+1)(x+2)}{(2+1)(2+2)}$$

$$P_2(x) = L_{n,0}\, y_0 + L_{n,1}\, y_1 + L_{n,2}\, y_2$$

# II. Newton's Divided Difference

- name is quite literal but best explained w/ an example:

$$f(x) = \sum p_k(x)$$

$$P_0(x) = a_0 = f(x_0)$$

$$p_1(x) = a_0 + a_1(x - x_0)$$

$$p_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) \ldots$$

How to solve for $a_i$:

$$a_i = f[x_i, x_j] = \frac{y_j - y_i}{x_j - x_i}, \quad i < j$$

$$a_{i+1} = f[x_i, x_j, x_k] = \frac{f[x_j, x_k] - f[x_i, x_j]}{x_k - x_i}$$

↳ notes: though only the 1st instances of each calculated $a_i$ will be used all will be needed to calculate $a_k$ for $k$ deg. poly.; easiest to do w/ a table

*ex:

| $x$ | $f(x)$ | 1st | 2nd | 3rd |
|---|---|---|---|---|
| $z_1$ | $z_2$ | | | |
| $b_1$ | $b_2$ | $f$ | $i$ | $k$ |
| $c_1$ | $c_2$ | $g$ | $j$ | |
| $d_1$ | $d_2$ | $h$ | | |

$a_0 = z_2$

$a_1 = f$

$a_2 = i$

$a_3 = k$

# APPROXIMATING DIFFERENTIATION

- approximations are made by differentiating the $p(x)$ interpolation
- poly. interpolation best works near the center of data
  - ↳ that's why its good for getting 1 approx. & checking multiple pts
  - ↳ $|f'(x) - p'(x)| \leq \frac{M}{(N!)} \Delta^N$ for $x \in [x_0, x_1]$

    $\Delta = (x_1 - x_0)$

    $M = \max(f^{N+1}(x))$

    $N$ = # of iterations

- Newton's divided difference method is something more familiar...

$$DD_{f'}(h) = \frac{f(\hat{x}+h) - f(\hat{x})}{h} \quad \text{(forward diff.)}$$

$$DD_{f'}(-h) = \frac{f(\hat{x}) - f(\hat{x}-h)}{h} \quad \text{(backward diff.)}$$

$$\downarrow$$

$$CD_{f'}(h) = \frac{DD_{f'}(h) + DD_{f'}(-h)}{2}$$

↳ better suited for individual form because method centers around data pts rather than finding a general form

↳ approx. based on Taylor expansion centered @ $\hat{x}$ w/ $x = \hat{x}+h, \hat{x}-h$

$$f(x) = f(\hat{x}+h) = f(\hat{x}) + \frac{f'(\hat{x})}{1!}(\hat{x}+h-\hat{x}) + \frac{f''(x)}{2!}(\hat{x}+h-\hat{x})^2 + O(h^3)$$

$$f(\hat{x}-h) = f(\hat{x}) + \frac{f'(\hat{x})}{1!}(\hat{x}-h-\hat{x}) + \frac{f''(x)}{2!}(\hat{x}-h-\hat{x})^2 + O(h^3)$$

$$CD_f = \frac{DD_{f'}(h) - DD_{f'}(-h)}{2} = \frac{f(\hat{x}+h) - f(\hat{x}-h)}{2h}$$

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

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

↳ higher order approximations are an expansion of the 1st order approx.

$$f''(x) = (f')'$$

$$\approx \left(\frac{f(x+h) - f(x)}{h} - \frac{f(x) - f(x-h)}{h}\right)\left(\frac{1}{h}\right)$$

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

↳ based on derivation of Lagrange polynomials for n given pts

# Numerical Integration

## I. Riemann Sums

- $\int_a^b f(x)dx \underset{N\to\infty}{\approx} \sum f(x_*)\Delta \rightarrow \frac{b-a}{n}$

$x_* = a + n\Delta$ LHS

$= a + (n+\frac{1}{2})\Delta$ MID

$= a + (n+1)\Delta$ RHS

- Error for $\underline{1 \text{ RS}}$

$a = a + i\Delta \rightarrow x_k$ $\quad M = \max\{f'(x) \mid x \in [x_k, x_{k+1}]\}$

$b = a + (i+1)\Delta$ $\quad$ * exists due to MVT

$\rightarrow x_{k+1}$

$c \in [x_k, x_{k+1}]$

$$\left|\int_a^b f(x)dx - f(c)\Delta\right|$$

$$= \left|\int_{x_k}^{x_{k+1}} f(x)dx - f(c)\Delta\right| = \int_{x_k}^{x_{k+1}} |f(x) - f(c)|dx$$

$$\leq \int_{x_k}^{x_{k+1}} M|c-x|dx$$

$$= M \int_{x_k}^{c} (c-x)dx + \int_c^{x_{k+1}} (x-c)dx$$

$$= M\left[\frac{1}{2}(c-x_k)^2 + \frac{1}{2}(x_{k+1}-c)^2\right] \leq M\Delta^2$$

$$\boxed{\therefore \left|\int_a^b f(x)dx - f(c)\Delta\right| \leq M\Delta^2}$$

* error will vary based on "rule" used

$$\Delta \begin{matrix} LHR \\ RHR \end{matrix} \rightarrow M\left(\frac{1}{2}\Delta^2\right)$$

$$\Delta \text{ midpt} \rightarrow M\left(\left(\frac{1}{2}\Delta\right)^2\right) = M\left(\frac{1}{4}\Delta^2\right)$$

- ERROR for <u>FULL</u> RS

$$\Delta = \frac{b-a}{n}, \quad x_k = a + k\Delta, \quad a = x_0$$

$$|M| = \max\{|f'(x)| : x \in [a,b]\},$$

$$\left|\int_a^b f(x)\,dx - \sum_{k=0}^{N-1} f(c_k)\Delta\right| \le \sum_{k=0}^{N-1} |M|\Delta^2$$

$$= N|M|\Delta^2$$

$$= |M|(b-a)\Delta$$

* $|M|$ is where greatest change occurs regardless of sign

II. QUADRATURE

- applies a weight to RS

$$\int_a^b f(x)\,dx \approx \sum_{k=0}^{n-1} a_k f(x_k)$$

what the weight varies by rule . . . $\xrightarrow[\text{NEXT}]{}$

$\hookrightarrow a_k = \frac{b-a}{N}$ for RS

# IIA. Trapezoid Rule

- $\boxed{h = x_{k+1} - x_k}$

$$\int_{x_k}^{x_{k+1}} f(x)\,dx \approx \sum_{k=0}^{N-1} \int_{x_k}^{x_{k+1}} p_k(x)\,dx$$

$$P_1(x) = f(x_k) + \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}$$

$$\int P_1(x)\,dx = \underbrace{f(x_k)\,x}_{\star} + \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}\left(\frac{(x - x_k)^2}{2}\right)\Bigg|_{x_k}^{x_{k+1}}$$

$$= \frac{f(x_{k+1}) - f(x_k)}{2}\,(x_{k+1} - x_k)$$

$$= \frac{1}{2}\big(f(x_{k+1}) - f(x_k)\big)\,h$$

$$\therefore \int_a^b f(x)\,dx \approx \sum_{k=0}^{N-1}\left(\frac{f(x_{k+1}) - f(x_k)}{2}\right) h$$

$$= \left(\frac{f(a)}{2} + \sum_{k=1}^{N-1} f(a+kh) + \frac{f(b)}{2}\right) h$$

$\star$

- ERROR for 1 TRAPEZOID

$$\left| \int_{x_k}^{x_{k+1}} f(x)\,dx - \left( \frac{f(x_{k+1}) - f(x_k)}{2} \right) h \right|$$

$$= | ① - ② |$$

① $x_k = x$ , $x_{k+1} = x + h$

$$\int_{x_k}^{x_{k+1}} f(y)\,dy = \int_0^h f(x+t)\,dt$$

$$= \int_0^h f(x) + f'(x)t + O(t^2)\,dt$$

$$= f(x)t + f'(x)\frac{t^2}{2} + O(t^3) \Big|_0^h$$

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

② $$\left( \frac{f(x+h) - f(x)}{2} \right) h = h\left( \frac{f(x) + f'(x)h + f(x) + O(h^2)}{2} \right)$$

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

$$| ① - ② | = O(h^3) = \frac{M}{2} h^3 ,$$

$$M = \max \{ f''(x) : x \in [a,b] \}$$

- ERROR for <u>FULL</u> TRAPEZOIDS

same proof as before

$$\left| \int_a^b f(x)\,dx - \sum_{k=0}^{N-1} \left( \frac{f(x_{k+1}) + f(x_k)}{2} \right) h \right| \le \frac{M}{2}(b-a)h^2$$

- Comparison of:

| | TRAPEZOID | & | MIDPT RS |
|---|---|---|---|
| eval: | $N+1$ | | $N$ |
| error: | $O\left(\left\|\frac{b-a}{n}\right\|^2\right)$ | | $O\left(\frac{b-a}{n}\right)$ |

# III. ADVANCED QUADRATURE RULE

## III A. SIMPSON'S RULE

- do quadratic interpolation over $\{x_k, x_{k+1}, x_{k+2}\}$

$P_k(x) = f(x_0)\, L_{N,0}$ ← start w/ 1 Lagrange poly.

$$L_{N,0} = \frac{(x-(x_0+h))(x-(x_0+2h))}{(x_0-(x+h))(x_0-(x+2h))} = \frac{\text{"} \quad \text{"}}{2h^2}$$

$$\int_{x_k}^{x_{k+2}} L_{N,0}\,dx \to y = x - x_0 - h \to \int_{-h}^{h} \frac{(y)(y-h)}{2h^2}\,dy$$

$$= \frac{1}{2h^2}\left( \frac{y^3}{3} - \frac{y^2}{2}h \right)\Big|_{-h}^{h} = \frac{1}{3}h$$

$$\therefore \int_{x_k}^{x_{k+2}} f(x_0)\, L_{N,0}\,dx = \frac{1}{3} h\, f(x_0)$$

- this process can be summed over w/ various
# of pts to form ...

## III B. COMPOSITE RULE

- $\int_a^b f(x)\,dx = \sum_{k=0}^{N-1} \int_{x_{N-2}}^{x_N} f(x)\,dx$

$$\approx \frac{h}{3}\left(f(x_0) + 4f(x_1) + f(x_2)\right)$$

(SIMPSON'S 3PT RULE)

$$\approx \frac{3h}{8}\left(f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)\right)$$

(SIMPSON'S 4PT RULE)

$$\approx \frac{2h}{45}\left(7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)\right)$$

(BOOLE'S RULE)

↳ consider the 3pt form as the focus
the reason why the summation works
is because either 3 pts can be supplied
from the data, or the creation of 3
from 2 given a specific interval such
that $\int f(x)\,dx = \sum \int f(x)\,dx$

* if 3 pts are supplied → single term
* if intervals are supplied → extended sum

this method is what makes the COMPOSITE

- Error Summary:

| | Midpt | Trape. | 3-pt | 4-pt | 5-pt |
|---|---|---|---|---|---|
| # of eval | 1 | 2 | 3 | 4 | 5 |
| $\Sigma$ (single) | $O(h^2)$ | $O(h^3)$ | $O(h^5)$ | $O(h^5)$ | $O(h^7)$ |
| (extended) | | | $O(h^4)$ | $O(h^4)$ | $O(h^6)$ |

- Observations:
  - ↳ if the provided $f(x)$ is linear, then the trapezoid rule will best approx. because it is <u>linear</u>
  - ↳ n-pt methods are better for higher order $f(x)$
    ex: 3-pt is good for quadratic relations

# Adaptive Rules for Integration

- repeating a method n-times can help provide a "feel" for convergence while also reusing past iterations for confirmation of the final iteration's output

## I. Recursive Trapezoid Rule

- $$T_{2k} = \frac{1}{2} T_k + \sum_{i \text{ is odd}} f(a + i\hat{h})\,\hat{h}, \quad \hat{h} = \frac{h}{2} = \frac{b-a}{2k}$$

output of composite Trapezoid rule over 2k intervals

## II. Recursive Simpson's Rule

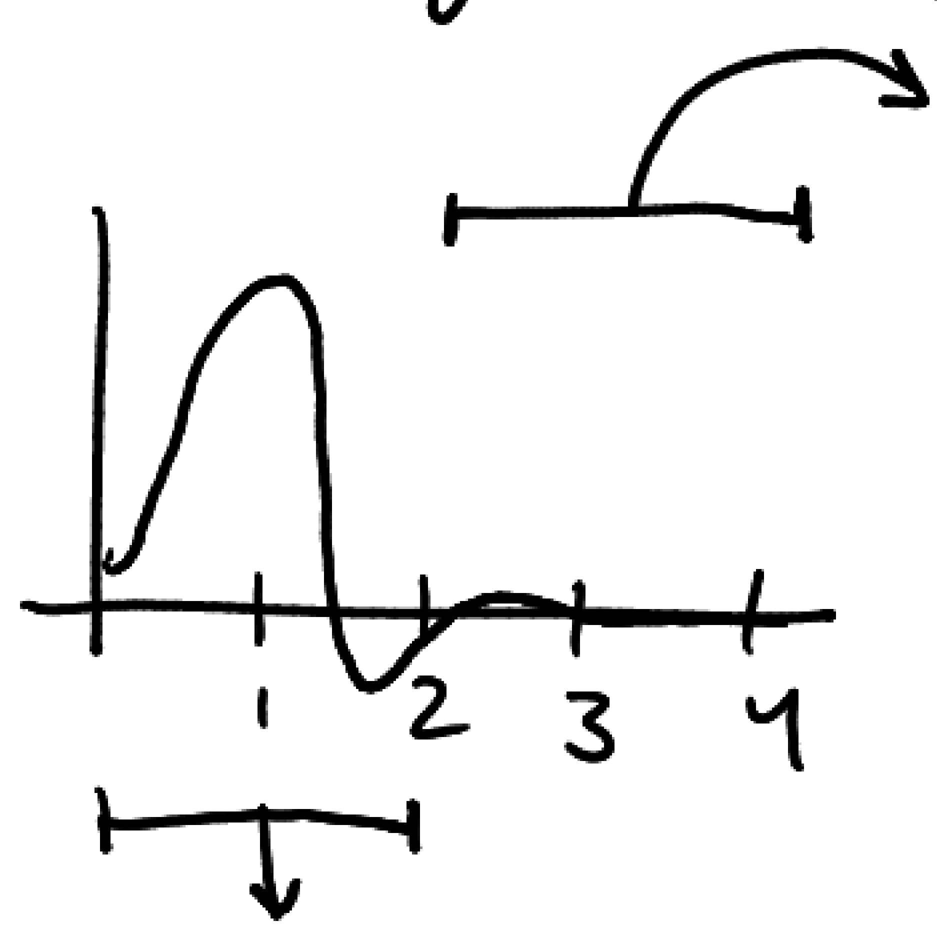
- can generate composite Simpson's Rule from $T_k$

$$S_k = \frac{4T_{2k} - T_k}{3}$$

## III. Adaptive Quadrature

- composite rules require equally spaced intervals

consider:




low order methods better

higher order methods better

NEXT →

→ in order to determine which sections to use smaller intervals, the error can be approximated:

$$\left|\int_a^b f(x)\,dx - S_k\right| \le \frac{M}{k^4}$$

← assumed to be the same; exact are diff due to diff. k

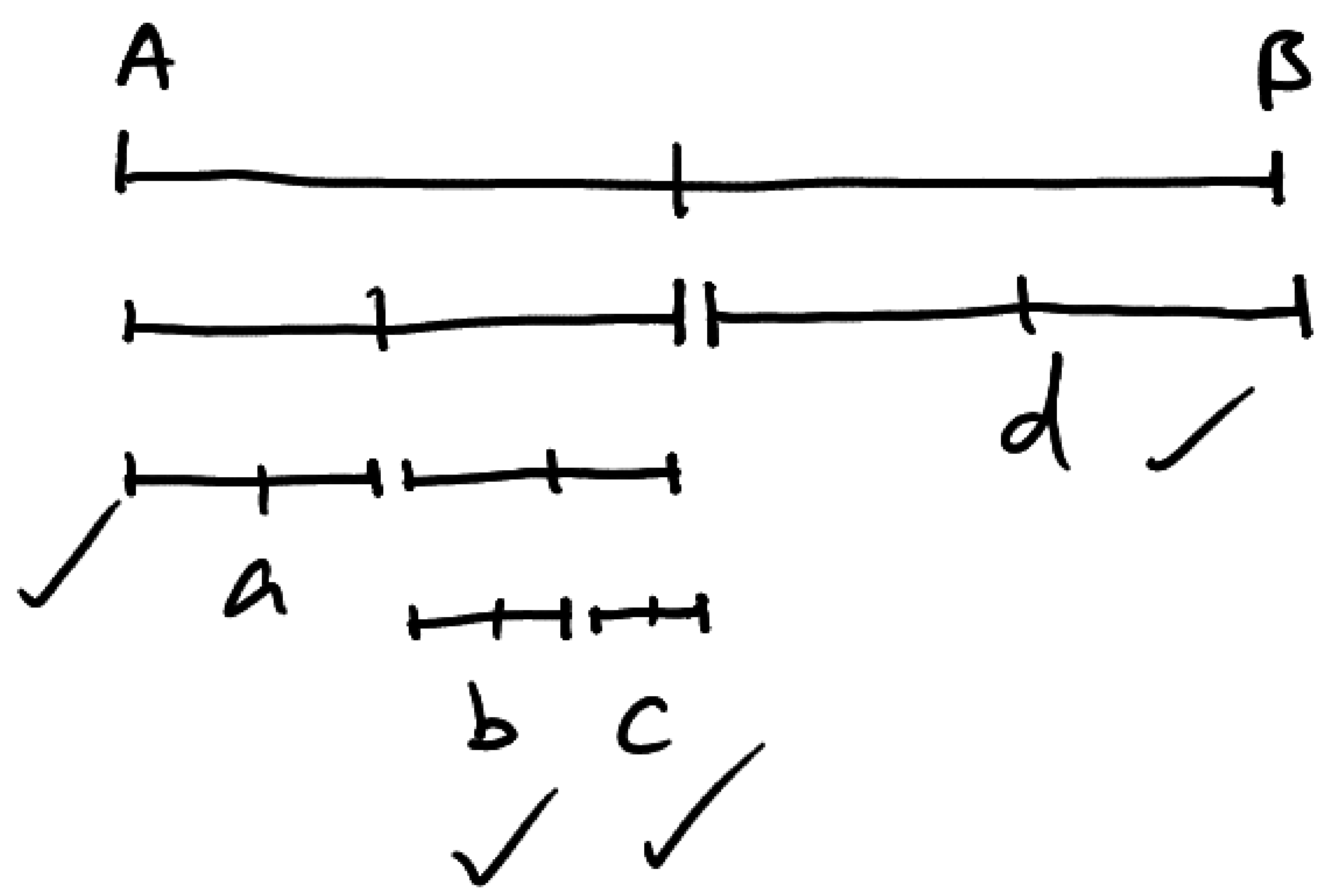
↓

$$\left|\int_a^b f(x)\,dx - S_{2k}\right| \le \frac{M}{(2k)^4}$$

$$\le \frac{1}{15}\left(\frac{15}{2^4}\right)\frac{M}{k^4}$$

$$\le \frac{1}{15}\left|S_{2k} - S_k\right| \approx \frac{1}{10}\varepsilon$$

- How the method works for $\int_a^b f(x)\,dx$ within $\varepsilon$ tol.:




$$ADQ(A, B, \varepsilon)$$

$$= ADQ\left(A, \frac{A+B}{4}, \frac{\varepsilon}{4}\right)$$

$$+ ADQ\left(\frac{A+B}{4}, \frac{A+B}{4} + j, \frac{\varepsilon}{8}\right)$$

$$+ ADQ\left(\frac{A+B}{4}, \frac{A+B}{2}, \frac{\varepsilon}{8}\right)$$

$$+ ADQ\left(\frac{A+B}{2}, B, \frac{\varepsilon}{2}\right)$$

any <u>method</u> can be used

# MONTE CARLO METHOD

- the adaptive quadrature begins to fail for multivariate integrals eq. due to the immense # of eval. needed
  - ↳ requires $O\left(\frac{1}{h^n}\right)$ for size $h$ in $\mathbb{R}^n$

⇓

this can be circumvented w/ inference:

$$\langle f \rangle = \left(\frac{1}{b-a}\right)\left(\frac{1}{d-c}\right)\int_a^b\int_c^d f(x,y)\,dy\,dx$$

&

$$(b-a)(d-c)\underbrace{\langle f \rangle}_{\text{can be approximated...}} = \int_a^b\int_c^d f(x,y)\,dy\,dx$$

$$\langle f \rangle \approx \frac{1}{Z}\sum f(x_i, y_i)$$

↳ $Z = N^n$ for $N$ eval. @ $\mathbb{R}^n$

⇓

$$\text{integral} = \left(\begin{array}{c}\text{area of}\\ \text{region}\end{array}\right)\frac{1}{Z}\sum_{i=1}^{Z} f + O\left(\frac{1}{\sqrt[n]{Z}}\right)$$

for $\mathbb{R}^2$, area $= (b-a)(d-c)$

for $\mathbb{R}^n$ where $Z = N^n$

- ex:

for $\mathbb{R}^1$ & accuracy $= 10^{-3}$:

$$|\text{int} - \text{simpson's rule}| \le \frac{0.02}{N^4}$$

which method is more efficient for $\mathbb{R}^1, \mathbb{R}^2, \mathbb{R}^5, \mathbb{R}^{10}$?

Simpson's Rule calls $(2N+1)^n$

$$\frac{0.02}{N^4} \le 10^{-3} \rightsquigarrow N = 3,\ n = 1$$

MC. calls $\frac{0.1}{\sqrt{z}} \le 10^{-3} \rightsquigarrow (10^2)^2 \le z$

$\therefore$ MC shows improvement over SR from $\mathbb{R}^5$

# SOLVING ODE IVP NUMERICALLY

- a WELL-POSED IVP is one that:
  - ↳ has a unique soln
  - ↳ the gradient is continuous

## I. EULER'S METHOD

- $\frac{dy}{dt} = f(t, y)$ , $t \in [a, b]$

  $y(a) = y_0$

  ↓

  $y(t_1) \approx y_0 + f(t_0, y_0)h$

  $y(t_2) \approx y(t_1) + f(t_1, y(t_1))h$

  $\vdots$

  $y_{k+1} = y_k + h \underbrace{\phi(t_k, y_k)}_{\text{increment fnc.}}$

  ↳ method is considered a DISCRETE VARIABLE METHOD because approx. is based on adding an increment

- Error Analysis

Local $O(h^2)$

Global $O(h)$

method only works if a 2nd derivative exists

intuitively makes sense because steps depend on h

↳ Global error tends to be overestimated
△ local step $= h$, global step $= \frac{b-a}{N}$

↳ method considered 1st order

## II. Taylor's Series Method

- $y(x) \approx y(x_0) + y'(x_0)(x-x_0)$
$$+ \frac{y''(x_0)}{2!}(x-x_0)^2 + \frac{y'''(x_0)}{3!}(x-x_0)^3$$

for an ODE: $y(x_0)$, $x_0$, $y'(x)$ are provided

used in tandem to find $y''(x_0)$ & $y'''(x_0)$

- ERROR ANALYSIS

For $y \in C^{p+1}$, the $p^{th}$ order Taylor:

LOCAL | GLOBAL
--- | ---
$O(h^{p+1})$ | $O(h^{p})$

↳ only useful if $y''$ & $y'''$ easy to solve for

↳ Euler = 1st Order Taylor

# III. HEUN'S METHOD

- ALGORITHM:

$y' = f(t, y)$, $y_a = y_0$

LOOP

$\tilde{y} = y_0 + f(t_0, y_0)h$

$t_1 = t_0 + h$

$y_1 = y_0 + \frac{1}{2}(f(t_0, y_0) + f(t_1, \tilde{y}))h$

- ERROR ANALYSIS

LOCAL | GLOBAL
--- | ---
$O(h^3)$ | $O(h^2)$

## IV. Runge-Kutta Method

- expand on Heun Method & weigh <u>multiple</u> slopes to find approx.
  - ↳ Euler is <u>1</u> pt approx.
  - ↳ Heun is <u>2</u> pt approx.

↓

4 pt aka RK4:

$$F_1 = f(t_k, y_k)$$

$$\tilde{y}_{k+a_1} = y_k + \frac{1}{2}(F_1)h$$

$$F_2 = f\left(t_k + \frac{1}{2}h, \tilde{y}_{k+a_1}\right)$$

$$\tilde{y}_{k+a_2} = y_k + \frac{1}{2}(F_2)h$$

$$F_3 = f\left(t_k + \frac{1}{2}h, \tilde{y}_{k+a_2}\right)$$

$$\tilde{y}_{k+a_3} = y_k + F_3 h$$

$$F_4 = f(t_k + h, \tilde{y}_{k+a_3})$$

$$t_{k+1} = t_k + h$$

$$y_{k+1} = y_k + \frac{h}{6}(F_1 + 2F_2 + 2F_3 + F_4)$$

- Error Analysis

$y \in C^5$

| Local | Global |
| --- | --- |
| $o(h^5)$ | $o(h^4)$ |

# Method Stability

- all methods will fail if the stepsize is too small

$\downarrow$

$$\frac{dy}{dt} = \lambda y, \quad y(0) = y_0$$

$\downarrow$

$$y = y_0 \exp(\lambda t)$$

use the test eq. to determine stability

$\lambda$ is any real #;

problems arise when $\lambda < 0 \ldots$

- STABILITY RELATION:

$$|y_{k+1}| \leq |y_k|$$

↳ for Euler & Heun:

$$0 < h < \left|\frac{2}{\lambda}\right|$$

↳ for RK4

$$\left|1 + \frac{(\lambda h)^4 + 4(\lambda h)^3 + 12(\lambda h)^2 + 24\lambda h}{24}\right| \leq 1$$

# FINAL EXAM STRUCTURE

- Exam format:
  - ↳ 2 quizzes:
    - 1 w/ the real exam (3 hr)
    - 1 to submit exam answers (15 min)
  - ↳ make sure to write agreement @ pg 1 of notes
- handwritten notes have higher priority than online

## Review Format

1. code w/ eq.
2. explanation
3. Error
4. Convergence & Complexity if provided