Numerical integration, low-dimensional func

- Common term: "Quadrature"
- *Many* methods ...
- Newton-Coates quadrature
  - constant step size (h)
    - Trapezoidal rule
    - Simpson's rule

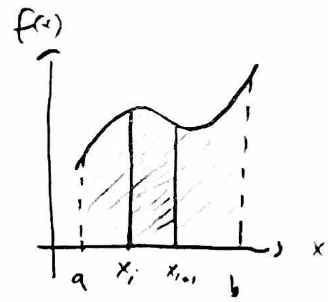
    Known methods, point out implementation approaches
- Gaussian quadrature
  - Not constant step size

History: Compute area of region $\Leftrightarrow$ construct a square with the same area

General considerations:

- How many evaluations of $f(x)$?
- How to distribute evaluations along x axis? (stepsize, ..)
- How to do interpolation at the top of the rectangle-like areas?

Just "fancy Riemann sums"



$$I = \int_a^b f(x)\,dx$$

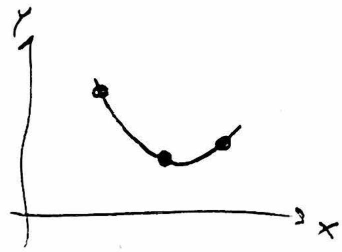
Background: Lagrange's interpolation formula

for N-th order polynomial defined by N+1 known points (x_i, y_i)

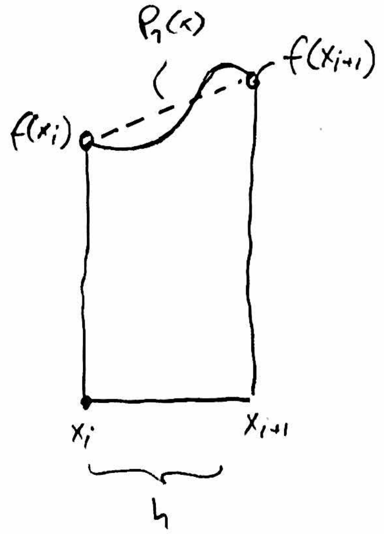
$$P_N(x) = \sum_{i=0}^{N} \left(\prod_{k \neq i} \frac{x - x_k}{x_i - x_k} \right) y_i$$

Example: Second-order polynomial going through $(x_0, y_0), (x_1, y_1), (x_2, y_2)$

$$P_2(x) = \left(\frac{x-x_1}{x_0-x_1}\right)\left(\frac{x-x_2}{x_0-x_2}\right)y_0 + \left(\frac{x-x_0}{x_1-x_0}\right)\left(\frac{x-x_2}{x_1-x_2}\right)y_1 + \left(\frac{x-x_0}{x_2-x_0}\right)\left(\frac{x-x_1}{x_2-x_1}\right)y_2$$

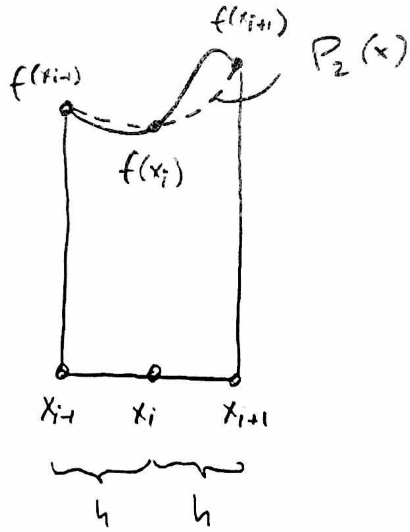


- Can use this to define polynomials to interpolate the tops of the integration rectangles



$\Downarrow$

Trapezoidal rule



$\Downarrow$

Simpson's rule

Trapezoidal rule

- Approximate $f(x)$ using first-order polynomial

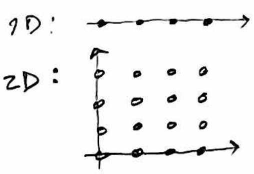
$$I = \int_{x_i}^{x_{i+1}} f(x)\,dx = \int_{x_i}^{x_{i+1}} \left[P_1(x) + \mathcal{O}\left(h^2 \frac{d^2f}{dx^2}\right) \right] dx \qquad \left[x_{i+1} = x_i + h \right]$$

$$= h\left[\tfrac{1}{2} f(x_i) + \tfrac{1}{2} f(x_{i+1}) \right] + \mathcal{O}\left(h^3 \frac{d^2 f}{dx^2} \right)$$

[Local error, i.e. error from one step]
(We'll have a global error $\mathcal{O}(h^2)$)

- Extended integration region ($a = x_0$, $b = x_{n-1}$) using n points:

$$I = \int_{x_0}^{x_{n-1}} f(x)\,dx = \underbrace{h\left[\tfrac{1}{2} f(x_0) + f(x_1) + f(x_2) + \dots + f(x_{n-2}) + \tfrac{1}{2} f(x_{n-1}) \right]}_{T_h} + \mathcal{O}(h^2)$$



- Actual error term: $I - T_h = -\frac{(b-a)}{12} h^2 f''(\xi)$

for some $\xi \in [a,b]$

- Error scales as $\mathcal{O}(h^2) = \mathcal{O}\left(\frac{1}{n_{1D}^2}\right)$ in one dimension

and $\mathcal{O}(h^2) = \mathcal{O}\left(\frac{1}{n_{dD}^{2/d}}\right)$ in d dimensions $(h_x = h_y = h_z = \dots = h)$

1D: $n = \frac{1}{h} + 1$

$n \approx \frac{1}{h}$

Number of points for stepsize h in each dim.:

1D: $n_{1D} \simeq \frac{1}{h}$

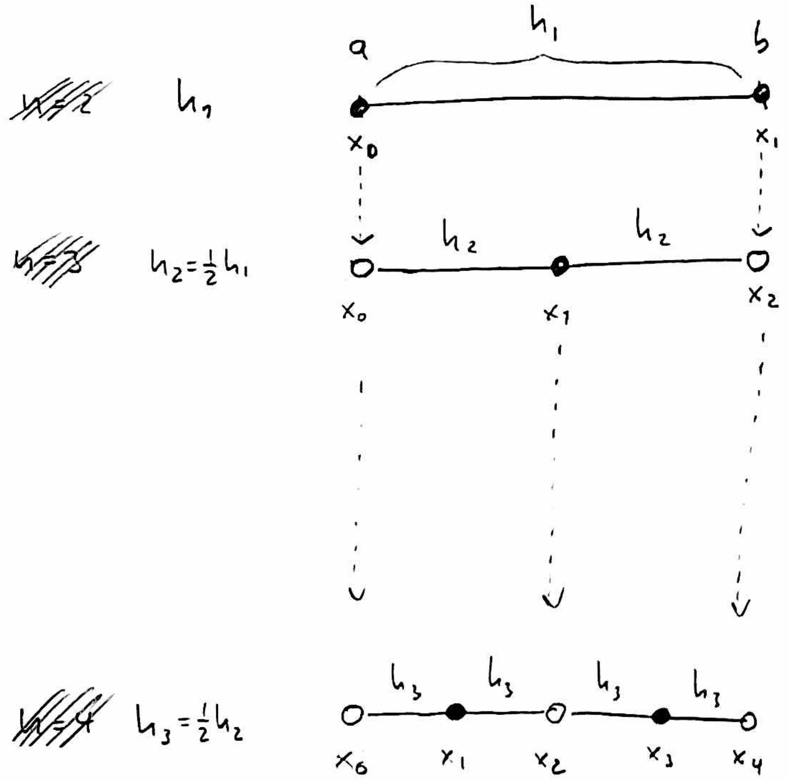
2D: $n_{2D} = n_{1D}^2 \simeq \frac{1}{h^2}$ $\quad (h_x = h_y = h)$

dD: $n_{dD} = n_{1D}^d \simeq \frac{1}{h^d} \Rightarrow h \simeq \frac{1}{n_{dD}^{1/d}}$

We don't know ξ,
but if we know that
$|f''(x)| \leq M$ (some constant),
we have a bound for the error.

$$|I - T_h| \leq \frac{(b-a)}{12} h^2 M$$

- Efficient implementation: $I = \int_a^b f(x)\,dx$



$$T_1 = h_1\left[\tfrac{1}{2} f(x_0) + \tfrac{1}{2} f(x_1)\right]$$

$$T_2 = h_2\left[\tfrac{1}{2} f(x_0) + f(x_1) + \tfrac{1}{2} f(x_2)\right]$$

$$= h_2\left[\tfrac{1}{2} f(x_0) + \tfrac{1}{2} f(x_2)\right] + h_2 f(x_1)$$

$$= \tfrac{1}{2} h_1\left[\;\dots\;\right] + h_2 f(x_1)$$

$$= \underline{\tfrac{1}{2} T_1 + h_2 f(x_1)}$$

1 new point
3 point total

$$T_3 = \tfrac{1}{2} T_2 + h_3\left[f(x_1) + f(x_3)\right]$$

2 new points
5 points total

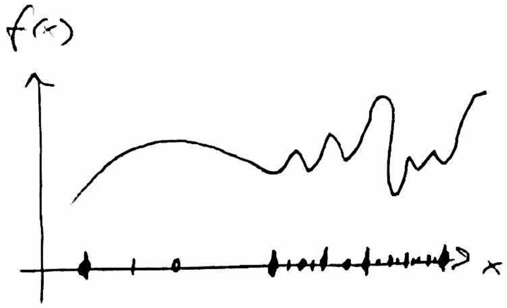
In general: $\underline{T_{j+1} = \tfrac{1}{2} T_j + h\left[f(x_1) + f(x_3) + \dots + f(x_{n-2})\right]}$

$\left[\begin{array}{l} n = \text{total number of points} \\ \quad = 2j + 1 \\ x_i : \text{midpoint in interval } i \\ \quad \text{from comp. of } T_j \end{array}\right]$

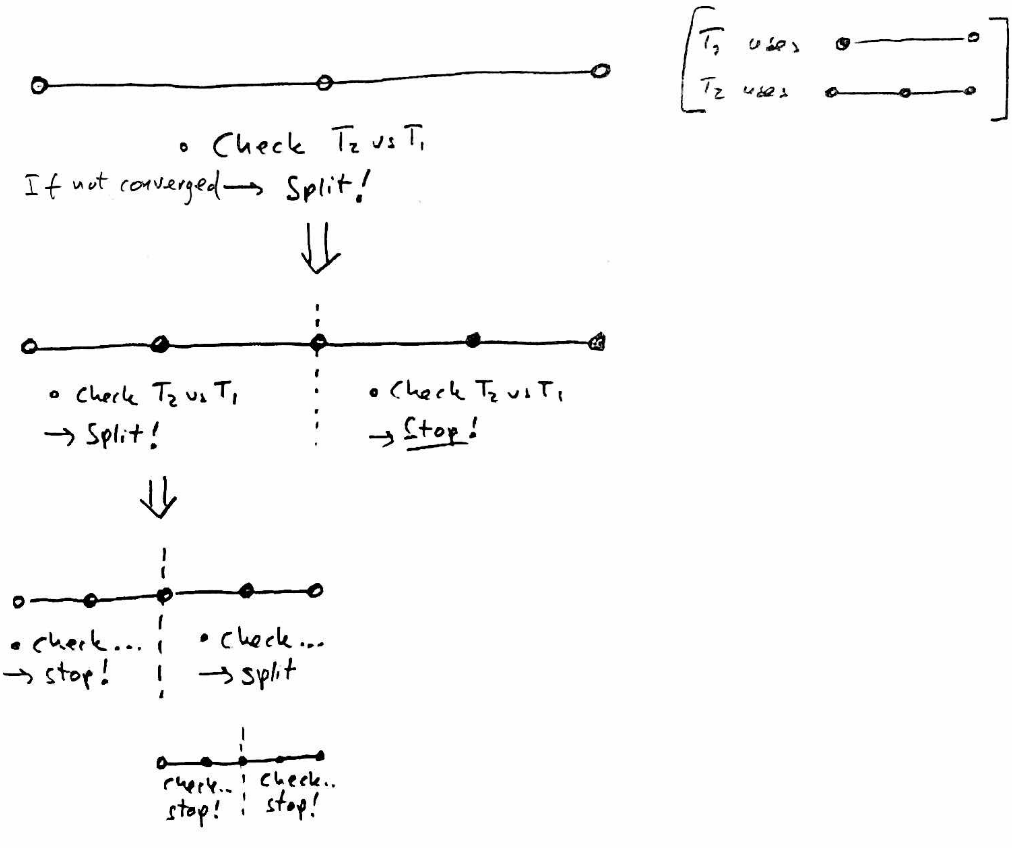
- Algorithm:
  1) Set desired relative accuracy ε
  2) Compute T_1
  3) Compute T_2
  4) while $\left|\dfrac{T_{j+1} - T_j}{T_j}\right| > \varepsilon$:

     compute the next T_{j+1} and compare with T_j

- Can also use <u>adaptive division of integration range</u>:



- Example: Recursive comparison between T_2 and T_1 $\left(\text{check } \frac{T_2 - T_1}{T_1}\right)$



Final points:

- Can implement as a <u>function calling itself</u> (recursion)

  ↳ See Morten's notes

Simpson's rule

- Approximate $f(x)$ with second-order polynomial $P_2(x)$ in each interval of three points

- $$\int_{x_i}^{x_{i+2}} f(x)\,dx = h\left[\frac{1}{3}f(x_i) + \frac{4}{3}f(x_{i+1}) + \frac{1}{3}f(x_{i+2})\right] + \mathcal{O}\left(h^5 \frac{d^4 f}{dx^4}\right)$$

  [Recall: comes from integrating Lagrange's interp. formula $P_2(x)$ across two steps.]

  [Surprise! Had naively expected $\mathcal{O}(h^4)$ local error!]

  (h^3 from leading Taylor term, integrated $\rightarrow h \times h^3 = h^4$)

- Can be seen as weighted sum of two applications of the trapezoidal rule:

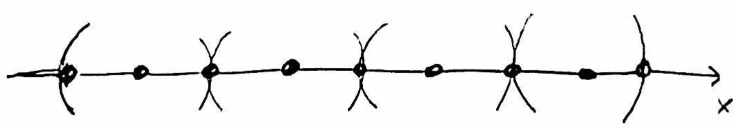
$$S_j = \frac{4}{3} T_{j+1} - \frac{1}{3} T_j$$

- Here both the $\mathcal{O}(h^3)$ and $\mathcal{O}(h^4)$ local trapezoidal errors cancel $\longrightarrow$ S has $\mathcal{O}(h^5)$ local error.

- Extended integration region using n points (n is odd)

$$I = \int_{x_0}^{x_{n-1}} f(x)\,dx = h\left[\frac{1}{3}f(x_0) + \frac{4}{3}f(x_1) + \frac{2}{3}f(x_2) + \frac{4}{3}f(x_3) + \ldots + \frac{2}{3}f(x_{n-3}) + \frac{4}{3}f(x_{n-2}) + \frac{1}{3}f(x_{n-1})\right] + \mathcal{O}(h^4)$$

Global error



- Even number of steps
- Odd number of points

(Just patch together 3-point regions side by side)

- Useful approach: 1) Use trapezoidal method that computes both T_j and T_{j+1}
  2) Improve final estimate via
  $$S = \frac{4}{3} T_{j+1} - \frac{1}{3} T_j$$

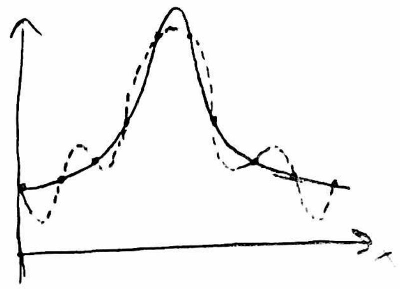
- Combine this with the smart trapezoidal implementations, e.g. the recursive one

- Can use higher-degree polynomials, but don't necessarily gain much... Example of problem:
- <u>Runge's phenomenon</u>: high-degree polynomial interpolation w/ equally spaced points → "<u>ringing</u>" (Polynomials oscillate)

- End up having to use small h to combat this

$\Downarrow$

Not gaining much from going to higher order.



Connection to I.V.P.

- Consider the initial value problem

$$\boxed{\frac{dy}{dx} = f(x)} \quad , \quad y(x_0 = a) = 0$$

Find $y(x=b)$

- Solving this is mathematically equivalent to solving the integral

$$\boxed{I = \int_a^b f(x)\,dx}$$

$$I = \int_a^b f(x)dx = \int_a^b \frac{dy}{dx}dx = \int_{y(a)}^{y(b)} dy = y(b) - y(a)$$

$$I = y(b) - y(a)$$

- We can take $y(a) = 0$ $\longrightarrow$

$$\Rightarrow \boxed{I = y(b)}$$

Need: $\frac{dy}{dx} = f(x)$

$$\Rightarrow y(x) = F(x) + C$$

can choose this

Choose $C = -F(a)$:

$$Y(a) = F(a) - F(a) = \underline{0}$$

$$y(b) = F(b) - F(a)$$

- We have discussed various stepwise methods for solving I.V.P., eg RK4:

$$\frac{dy}{dx} = f(x, y)$$

RK4: Given $Y_i = Y(x_i)$, the next step Y_{i+1} is computed as

$$\left.\begin{aligned} k_1 &= h f(x_i, Y_i) \\ k_2 &= h f\left(x_i + \tfrac{h}{2}, Y_i + \tfrac{k_1}{2}\right) \\ k_3 &= h f\left(x_i + \tfrac{h}{2}, Y_i + \tfrac{k_2}{2}\right) \\ k_4 &= h f(x_i + h, Y_i + k_3) \end{aligned}\right\} \quad Y_{i+1} = Y_i + \frac{1}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right]$$

- But for the case of an integral $\int_a^b f(x)\,dx$, f does not depend on y

$\Rightarrow$ $k_2 = k_3$ in RK4

$\Rightarrow$ Simpler version

$$k_1 = h f(x_i)$$
$$k_2 = h f(x_i + \tfrac{h}{2})$$
$$k_3 = h f(x_i + h)$$

$$Y_{i+1} = Y_i + \frac{1}{6}\left[k_1 + 4k_2 + k_3\right]$$

$$= Y_i + \frac{1}{2}\left[\frac{1}{3}k_1 + \frac{4}{3}k_2 + \frac{1}{3}k_3\right]$$

$$= Y_i + \frac{h}{2}\left[\frac{1}{3}f(x_i) + \frac{4}{3}f(x_{i+\frac{1}{2}}) + \frac{1}{3}f(x_{i+1})\right]$$

Simpson's rule with $h \to \frac{h}{2}$

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = Y_{i+1} - Y_i = \frac{h}{2}\left[\ \cdots\cdots\ \right]$$

[Integration spans two stepsizes, just like in regular Simpson's rule]

- So Simpson's rule for $I = \int_a^b f(x)\,dx$ is equivalent to RK4 for $\frac{dy}{dx} = f$ where $f = f(x)$, not $f = f(x,y)$

Gaussian quadrature

[See Morten's lecture notes]

- $I = \int_a^b f(x)dx \approx \sum_{i=0}^{n-1} \omega_i f(x_i)$

  weights   grid points

- Ex: Trapezoidal rule: $\omega : \left\{ \frac{h}{2}, h, h, \ldots, h, \frac{h}{2} \right\}$

  Simpson's rule $\omega : \left\{ \frac{h}{3}, \frac{4h}{3}, \frac{2h}{3}, \ldots, \frac{4h}{3}, \frac{h}{3} \right\}$

  Here we only adjust the n weights, the n point positions are fixed

- Result: with n points we can integrate $P_{n-1}(x)$ exact.

- Gaussian quadrature: Allow varying both weights and point positions $\Rightarrow$ $2n$ parameters to adjust

  $\Rightarrow$ [With n points we can integrate $P_{2n-1}(x)$ exact.] Works best if $f(x)$ is well approximated by $P_{2n-1}(x)$ (or lower order)

- The point positions (x_i) are found as roots of orthogonal polynomials.