module 07
numerical integration and differentiation

ahmad f. taha

ce 2989 — numerical methods in civil & env. engineering

email: ahmad.taha@vanderbilt.edu

webpage: http://lab.vanderbilt.edu/taha

April 1, 2025

motivation

- what is numerical differentiation/integration?
- why is it needed instead of analytically computing derivatives $f'(x)$ or integrals $\int f(x)dx$?
- in many applications, we don't have the closed-form expression of the function so we cannot integrate or differentiate
- sometimes, we do have access the function $f(x)$ and we still can't really evaluate integration
- so we need numerical method to approximate values of integration (areas under the curve) or derivatives (rate of change of functions)
- numerical differentiation: calculating rate of change of a function based on data points
- applications: signal processing, control systems, structural design, ...
- machine learning: most common ML algorithms are based on *gradient computation*
- gradient-based optimization heavily relies on numerical differentiation to calculate the gradients of loss functions, guiding the learning process
- numerical integration/differentiation: very sensitive to small errors in data, leading to inaccurate results given noisy data
- **this module**: present most common numerical diff + integration techniques + error bounds + analysis

we start with the basic definition

- **objective of num diff:** approximate $f'(x), f''(x), f'''(x)$, etc..only from data points $f(x_i)$
- **definition of derivative:**

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h} \approx \frac{f(x_0 + h) - f(x_0)}{h} \quad \text{for a small h}$$

- practically, use small $h = 0.01$ or $h = 0.0001$ to approximate the derivative
- the smaller the better? hmmm not really actually
- **example:** given $f(x) = \sin(x)$, approximate $f'(1)$
- **results:**

h	relative error
0.100000000000000	0.079471349402736
0.010000000000000	0.007803640314835
0.001000000000000	0.000778870464261
0.000100000000000	0.000077872052482
0.000010000000000	0.000007787052229
0.000001000000000	0.000000778724808
0.000000100000000	0.000000077415348
0.000000010000000	0.000000005496710
0.000000001000000	0.000000097244201
0.000000000100000	0.000000108237621
0.000000000010000	0.000002163055846
0.000000000001000	0.000080029673119
0.000000000000100	0.001358343083757
0.000000000000010	0.006860929812673
0.000000000000001	0.027409112053748

- what's going on? is there an optimal h^*??

lagrange interpolation + num diff

- **main idea of num diff:** approximate $f(x)$ with an interpolation $p(x)$ from module 6 then use $p'(x)$ to approximate $f'(x)$ at x
- recall the lagrange polys, corresponding errors, and derivative:

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{k=0}^{n}(x - x_k), \quad p_n(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x)$$

$$\Rightarrow \quad f'(x_i) = \sum_{k=0}^{n} f(x_k)L'_{n,k}(x_i) + \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{k=0,k\neq i}^{n}(x_i - x_k)$$

- **example**: consider first order interpolation $(n = 1)$ using lagrange and approximate $f'(x_0)$
- we need two data points: x_0 and $x_0 + h = x_1$, then we can write

$$L_{1,0}(x) = \frac{x - x_1}{-h}, \quad L_{1,1}(x) = \frac{x - x_0}{h}$$

- hence $p_1(x) = -f(x_0)\dfrac{x - x_1}{h} + f(x_0 + h)\dfrac{x - x_0}{h}$
- computing $p'_1(x) = \dfrac{f(x_0 + h) - f(x_0)}{h} \approx f'(x_0)$
- this means that lagrange interpolation can approximate derivatives via forward divided diff

error analysis

- recall the error analysis of lagrange polys:

lagrange poly truncation error

the relationship between the function and its nth order lagrange poly approximation can be written as

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\dots(x - x_n)$$

where $x_0, x_1, \dots, x_n$, and ξ are all in interval $[a, b]$

- then for $p_1(x)$ in the previous slide, we can write

$$f(x) - p_1(x) = \frac{f''(\xi(x))}{2}(x - x_0)(x - x_1)$$

$$\Rightarrow f'(x) - p_1'(x) = \frac{1}{2}(2x - x_0 - x_1)f''(\xi) + \frac{(x - x_0)(x - x_1)}{2}\frac{df''(\xi)}{dx}$$

- to compute error of approximating $f'(x_0)$, we get

$$|f'(x_0) - p_1'(x_0)| = |\frac{h}{2}f''(\xi)|$$

forward/backward divided diff

- in general, if you have two data points x_0 and $x_1 = x_0 + h$, you can approximate the derivative via forward (+ve h) and backward (-ve h) divided diff:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

- and in general, we can write the error as

$$\left| f'(x_0) - \frac{f(x_0 + h) - f(x_0)}{h} \right| \leq \frac{M|h|}{2}$$

  where M is the upper bound for $f''(x)$ for $x \in [x_0, x_0 + h]$
- error here is *of order* $O(h)$
- how do we extend this to more data points to approximate $f'(x)$?
- let's use $n + 1$ data points and lagrange polys then we can write:

$$(n+1)\textbf{-point formula} \quad f'(x) \approx p_n'(x) = \sum_{k=0}^{n} f(x_k) L_{n,k}'(x)$$

- as outlined before, the error at a specific point $x = x_j$ is

$$\boxed{f'(x_j) - p_n'(x_j) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{k=0, k \neq j}^{n} (x_j - x_k)}$$

examples

- usually **3-point formula** is used to approximate $f'(x)$:

$$f'(x_j) = \sum_{k=0}^{2} f(x_k) L'_{2,k}(x) + \frac{f^{(3)}(\xi)}{3!} \prod_{k=0, k\neq j}^{2} (x_j - x_k)$$

$$\Rightarrow f'(x_j) = f(x_0)\frac{2x_j - x_1 - x_2}{(x_0 - x_1)(x_0 - x_2)} + f(x_1)\frac{2x_j - x_0 - x_2}{(x_1 - x_0)(x_1 - x_2)} +$$

$$f(x_2)\frac{2x_j - x_0 - x_1}{(x_2 - x_0)(x_2 - x_1)} + \frac{f'''(\xi)}{3!} \prod_{k=0, k\neq j}^{2} (x_j - x_k)$$

- **special case:** equidistant nodes $x_{i+1} = x_i + h$, you get a nicer form of derivative approximation:

$$f'(x_0) = \frac{-3f(x_0) + 4f(x_1) - f(x_2)}{2h} + \frac{h^2}{3} f^{(3)}(\xi_0)$$

$$f'(x_1) = \frac{f(x_2) - f(x_0)}{2h} - \frac{h^2}{6} f^{(3)}(\xi_1), \; f'(x_2) = \frac{f(x_0) - 4f(x_1) + 3f(x_2)}{2h} + \frac{h^2}{3} f^{(3)}(\xi_2)$$

- **note:** although error for midpoint and endpoint formulas are both order of h, the midpoint is twice as good because it considers points from either side of the point x_j

5-point formula to approximate $f'(x)$

- **exercise:** derive 5-point formula for $f'(x_0)$: $n = 4$ with $x_0, x_1 = x_0 + h, x_2 = x_0 + 2h, x_3 = x_0 + 3h, x_4 = x_0 + 4h$
- five-point formula to approximate $f'(x)$ can be written as

$$f'(x_j) = \sum_{k=0}^{4} f(x_k) L'_{4,k}(x) + \frac{f^{(5)}(\xi)}{5!} \prod_{k=0, k\neq j}^{4} (x_j - x_k)$$

- if the points are equally spaced, you arrive at the **midpoint 5-point rule**

$$f'(x_0) = \frac{-f(x_0+2h) + 8f(x_0+h) - 8f(x_0-h) - f(x_0-2h)}{12h} + \frac{h^4}{30} f^{(5)}(\xi), \xi \in [x_0-2h, x_0+2h$$

- could also define end point for $f'(x_j)$ which is useful in derivation clamped splines where the derivative evaluations aren't given
- can also compute **5-point end point formulae** (where $\xi \in [x_0, x_0 + 4h]$):

$$f'(x_0) = \frac{-25f(x_0) + 48f(x_0+h) - 36f(x_0+2h) + 16f(x_0+3h) - 3f(x_0+4h)}{12h} + \frac{h^4}{5} f^{(5)}(\xi)$$

- in general, and given that $f(x) = p_n(x) + R_n(x)$, we can write

$$f'(x) = p'_n(x) + \text{order}(h^n), \quad f''(x) = p''_n(x) + \text{order}(h^{n-1}), \quad f'''(x) = \ldots$$

example: implement the 3- and 5-point methods to approximate $f'(2)$

x	1.8	1.9	2.0	2.1	2.2
$f(x) = xe^x$	10.889365	12.703199	14.778112	17.148957	19.855030

- you can see $h = 0.1$...can use 3 data points to approximate $f'(2)$:

$$f'(x_0) \approx \frac{-3f(x_0) + 4f(x_1) - f(x_2)}{2h} = \frac{-3f(2) + 4f(2.1) - f(2.2)}{0.2} = 22.032$$

- alternatively, you could have used the backwards data points with $h = -0.1$ to obtain:

$$f'(x_0) \approx \frac{-3f(x_0) + 4f(x_1) - f(x_2)}{2h} = \frac{-3f(2) + 4f(1.9) - f(1.8)}{-0.2} = 22.054$$

- could use $x_1 = 2$ with $h = 0.1$ and use $f'(x_1) = \dfrac{f(x_2) - f(x_0)}{2h}$ to obtain

$$f'(x_1) \approx \frac{f(x_2) - f(x_0)}{2h} = \frac{f(2.1) - f(1.9)}{0.2} = 22.228$$

- orrrrr the same formula but with $h = 0.2$:

$$f'(x_1) \approx \frac{f(x_2) - f(x_0)}{0.4} = \frac{f(2.2) - f(1.8)}{0.2} = 22.414$$

- true derivative value: $f'(2) = 22.167$
- 5-point rule: $f'(2) = \dfrac{1}{1.2}(f(1.8) - 8f(1.9) + 8f(2.1) - f(2.2)) = 22.166$

higher order derivatives

- can we similarly approximate higher order derivatives?
- consider taylor series approximations $f(x_0 + h)$ and $f(x_0 - h)$
- can end up deriving the following relationship ($\xi_1 \in [x_0, x_0 + h]$, $\xi_{-1} \in [x_0 - h, x_0]$):

$$f''(x_0) = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} - \frac{h^2}{24}\left(f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\right)$$

- apply the ivt, we can get $f^{(4)}(\xi) = \frac{1}{2}\left(f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\right)$ resulting in

central div. diff. $f''(x_0) = \dfrac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} - \dfrac{h^2}{12} f^{(4)}(\xi),\ \xi \in [x_0 - h, x_0 + h]$

- **example:** following the previous example for $f(x) = xe^x$ we can approximate $f''(2)$:

$$f''(2) \approx \frac{1}{0.1^2}(f(1.9) - 2f(2.0) + f(2.1)) = 29.593$$

- for $h = 0.2$, we obtain $f''(2) \approx 29.704$ while the real value $f''(2) = 29.556$
- relative errors for $h = 0.1$ and 0.2 are 0.0012 and 0.005

there's an optimal h

- how to choose an optimal h: the smaller the better?
- not really. here's an example comparing $f'(0.9)$ for $f(x) = \sin(x)$:

Approximation of derivative of sin(x) at 0.9 using three-point midpoint

i	h	Approx	cos(0.9)	error
1	0.100000000000	0.620574469542	0.621609968271	0.001035498729
2	0.010000000000	0.621599608156	0.621609968271	0.000010360114
3	0.001000000000	0.621609864669	0.621609968271	0.000000103602
4	0.000100000000	0.621609967235	0.621609968271	0.000000001036
5	0.000010000000	0.621609968254	0.621609968271	0.000000000016
6	0.000001000000	0.621609968277	0.621609968271	-0.000000000006
7	0.000000100000	0.621609967943	0.621609968271	0.000000000327
8	0.000000010000	0.621609969054	0.621609968271	-0.000000000783
9	0.000000001000	0.621609985707	0.621609968271	-0.000000017436
10	0.000000000100	0.621609985707	0.621609968271	-0.000000017436
11	0.000000000010	0.621608320373	0.621609968271	0.000001647898
12	0.000000000001	0.621613871488	0.621609968271	-0.000003903217

- why does that happen? the derivative limit definition requires $h \to 0$???
- from the table we see the optimal h is between 10^{-6} and 10^{-5}
- consider that we are computing $f'(x)$ via

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h} - \frac{h^2}{6} f'''(\xi)$$

- define $\epsilon_m = 2.22 \times 10^{-16}$, to be the machine precision[1]

[1] an upper bound on the relative approximation error due to rounding in floating point number systems

how errors propagate

- when computing $f(x_0 + h)$ and $f(x_0 - h)$, both quantities have errors $e(x_0 + h)$ and $e(x_0 - h)$
- you can assume the following:

$$|e(x_0 \pm h)| \leq \epsilon_m, \quad |f'''(x)| \leq M$$

- then you can prove the following result on the error

$$|f'(x_0) - f_c'(x_0)| \leq \underbrace{\frac{Mh^2}{6}}_{\text{truncation error}} + \underbrace{\frac{\epsilon_m}{h}}_{\text{round-off error}}$$

- what do you notice? if you ignore the second term, you see that with a smaller h, you get smaller error
- but second term starts to dominate as h becomes so much smaller
- is there an optimal h^*? yessss because $e(h) = \frac{Mh^2}{6} + \frac{\epsilon_m}{h}$ so we can find h^* given a fixed M using high-school calculus: $h^* = \sqrt[3]{\frac{3\epsilon_m}{M}}$
- **previous example:** $f(x) = \sin(x)$ hence $M = \max|\cos(x)| = \cos(0.8) = 0.696$ for $x \in [0.8, 1]$ hence $h^* = 9.85 \times 10^{-6}$, consistent with results
- this formula is only valid for this specific derivative, 3-point formula
- this formula is in practice not really useful..why?

summary of results

forward finite-divided-difference formulas: two versions are presented for each derivative (the latter is more accurate):

First Derivative	Error
$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$	$O(h)$
$f'(x_i) = \frac{-f(x_{i+2}) + 4f(x_{i+1}) - 3f(x_i)}{2h}$	$O(h^2)$
Second Derivative	
$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2}$	$O(h)$
$f''(x_i) = \frac{-f(x_{i+3}) + 4f(x_{i+2}) - 5f(x_{i+1}) + 2f(x_i)}{h^2}$	$O(h^2)$
Third Derivative	
$f'''(x_i) = \frac{f(x_{i+3}) - 3f(x_{i+2}) + 3f(x_{i+1}) - f(x_i)}{h^3}$	$O(h)$
$f'''(x_i) = \frac{-3f(x_{i+4}) + 14f(x_{i+3}) - 24f(x_{i+2}) + 18f(x_{i+1}) - 5f(x_i)}{2h^3}$	$O(h^2)$
Fourth Derivative	
$f''''(x_i) = \frac{f(x_{i+4}) - 4f(x_{i+3}) + 6f(x_{i+2}) - 4f(x_{i+1}) + f(x_i)}{h^4}$	$O(h)$
$f''''(x_i) = \frac{-2f(x_{i+5}) + 11f(x_{i+4}) - 24f(x_{i+3}) + 26f(x_{i+2}) - 14f(x_{i+1}) + 3f(x_i)}{h^4}$	$O(h^2)$

summary of results (cont'd)

centered finite-divided difference formulas: two versions are presented for each derivative (the latter is more accurate):

	Error
First Derivative	
$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$	$O(h^2)$
$f'(x_i) = \frac{-f(x_{i+2}) + 8f(x_{i+1}) - 8f(x_{i-1}) + f(x_{i-2})}{12h}$	$O(h^4)$
Second Derivative	
$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}$	$O(h^2)$
$f''(x_i) = \frac{-f(x_{i+2}) + 16f(x_{i+1}) - 30f(x_i) + 16f(x_{i-1}) - f(x_{i-2})}{12h^2}$	$O(h^4)$
Third Derivative	
$f'''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + 2f(x_{i-1}) - f(x_{i-2})}{2h^3}$	$O(h^2)$
$f'''(x_i) = \frac{-f(x_{i+3}) + 8f(x_{i+2}) - 13f(x_{i+1}) + 13f(x_{i-1}) - 8f(x_{i-2}) + f(x_{i-3})}{8h^3}$	$O(h^4)$
Fourth Derivative	
$f''''(x_i) = \frac{f(x_{i+2}) - 4f(x_{i+1}) + 6f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{h^4}$	$O(h^2)$
$f''''(x_i) = \frac{-f(x_{i+3}) + 12f(x_{i+2}) - 39f(x_{i+1}) + 56f(x_i) - 39f(x_{i-1}) + 12f(x_{i-2}) - f(x_{i-3})}{6h^4}$	$O(h^4)$

richardson's method

- we have already seen that there are two approaches to improve derivative estimates with finite divided differences
- first approach is through making h (step-size) smaller
- second approach using more data-points or higher-order formulas
- a third approach that's different is based on *richardson extrapolation*
- this method uses two derivative estimates to compute a third, more accurate approximation

richardson's method

- is a general method for generating high-accuracy results using low-order equations
- method results in $M - N_j(h) = \sum_{k=j}^{\infty} K_k h^k$
- recall the following (from writing taylor series approx. for $f(x+h)$ and $f(x-h)$):

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \left(\frac{h^2}{3!} f^{(3)}(x) + \frac{h^4}{5!} f^{(5)}(x) + \frac{h^6}{7!} f^{(7)}(x) + \dots \right)$$

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+2)}{h^2} - \left(2\frac{h^2}{4!} f^{(4)}(x) + 2\frac{h^4}{6!} f^{(6)}(x) + 2\frac{h^6}{8!} f^{(8)}(x) + \dots \right)$$

- you can see that the results above can be written as

$$M = N(h) + K_2 h^2 + K_4 h^4 + \dots$$

  where M is the actual value, $N(h)$ is the derivative approximation as a function of h, and constants K_i are not related to h

- **main idea of richy's extrapolation:** use two approximations of the same order, but with different h ($N_j(h)$ and $N_j(h/2)$) at iteration j then combine the two approximations so that the error term of order h^j is eliminated

richardson – cont'd

$$M = N(h) + K_2 h^2 + K_4 h^4 + K_6 h^6 + \dots \quad (a)$$

- notice that you can rewrite (a) as

$$M = N(h/2) + K_2/4 \cdot h^2 + K_4/16 \cdot h^4 + K_6/64 \cdot h^6 + \dots$$

$$\text{or } 4M = 4N(h/2) + K_2 h^2 + K_4/4 \cdot h^4 + K_6/16 \cdot h^6 + \dots \quad (b)$$

- computing $(b) - (a)$, we get

$$M = \underbrace{\frac{1}{3}\left(4N(h/2) - N(h)\right)}_{N_2(h)} - 1/4 \cdot K_4 h^4 - 5/16 \cdot K_6 h^6 - \dots \quad (c)$$

- this derivation entails that a simple combination of two second-order approximations for the second derivative using central difference (via $N(h)$ and $N(h/2)$) produces accurate results with error of the order h^4
- but you can do so recursively by writing

$$M = N_2(h) - 1/4 \cdot K_4 h^4 - 5/16 \cdot K_6 h^6 + \dots \quad (a)'$$

$$M = N_2(h/2) - 1/64 \cdot K_4 h^4 - 5/1024 \cdot K_6 h^6 + \dots \quad (b)'$$

- by computing $(c) - 16(b)'$ we can now write

$$M = \underbrace{\frac{1}{15}\left(16N_2(h/2) - N_2(h)\right)}_{N_3(h)} + 1/64 \cdot K_6 h^6 + 1/1024 \cdot K_8 h^8$$

richardson – example

- this procedure can be repeated again and again, recursively
- to help us obtain better and better approximations, and smaller errors
- the algorithm can be written as
  1. step 1: select h and compute $D(i,0) = N(\frac{h}{2^i}),\ i = 0, 1, \ldots, n$
  2. step 2: for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, i$, compute

$$D(i,j) = \frac{1}{4^j - 1}\left(4^j D(i, j-1) - D(i-1, j-1)\right)$$

- these iterations lead to $D(n,n) = M + \text{order}(h^{2(n+1)})$
- **example:** using richy's method, approximate $f'(1) = 1$ for $f(x) = \ln(x)$ using $h = 0.2, 0.1, 0.05$ (so $n = 2$)
- step 1: compute $D(0,0), D(1,0), D(2,0)$

$$D(0,0) = N(h) = \frac{f(1.2) - f(0.8)}{0.4} = 1.0136,\ D(1,0) = N(h/2) = \frac{f(1.1) - f(0.9)}{0.2} = 1.0033$$

$$D(2,0) = N(h/4) = \frac{f(1.05) - f(0.95)}{0.1} = 1.00083$$

- step 2: compute the following

$$D(1,1) = \frac{1}{3}(4D(1,0) - D(0,0)) = 0.99991,\ D(2,1) = \frac{1}{3}(4D(2,0) - D(1,0)) = 0.99994$$

$$D(2,2) = \frac{1}{15}(16D(2,1) - D(1,1)) = 1.00000015$$

integrals are harder

- although integrals of $\int f(x)dx$ are harder to compute (specifically indefinite ones)
- we will see how numerically they can be relatively straightforward
- in this module, we show how to perform numerical integration via two steps:
  1. approximating $f(x)$ by an nth degree polynomial p_n then
  2. integrating the polynomial over a defined interval
- we know how to easily integrate any nth degree poly
- starting with $\{x_0, x_1, \ldots, x_n\}$ be distinct points in interval $[a, b]$
- lagrange interpolating poly can be written as $p_n(x) = \sum_{i=0}^{n} f(x_i)L_{n,i}(x)$
- then we can write

$$\int_a^b f(x)dx \approx \int_a^b p_n(x) = \int_a^b \sum_{i=0}^{n} f(x_i)L_{n,i}(x)dx = \sum_{i=0}^{n} f(x_i) \int_a^b L_{n,i}(x)dx = \sum_{i=0}^{n} A_i f(x_i)$$

- this formula is called the newton-coates formula—it's actually very general and encapsulates any finite interpolation
- error is given as $e = \frac{1}{(n+1)!} \int_a^b f^{(n+1)}(\xi(x)) \prod_{i=0}^{n}(x - x_i)dx$
- we will now see how this works for $n = 1, 2, \ldots$

the trapezoidal rule ($n = 1$)

- for $n = 1$ with $x_0 = a$ and $x_1 = b$, we obtain this approximating poly:

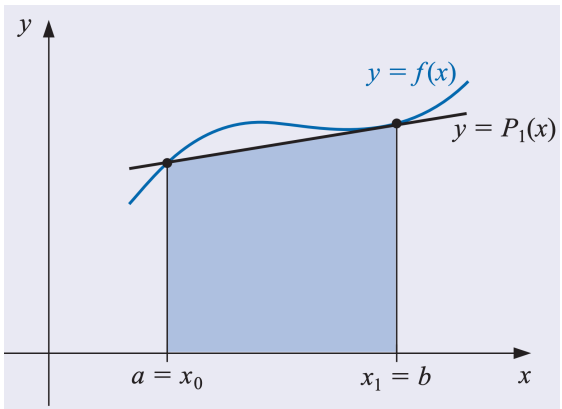
$$\int_{x_0}^{x_1} f(x)dx \approx A_0 f(x_0) + A_1 f(x_1) = \frac{1}{2}(x_1 - x_0)f(x_0) + \frac{1}{2}(x_1 - x_0)f(x_1)$$

- the error we get is:

$$\int_{x_0}^{x_1} \frac{f''(\xi(x))}{2!}(x - x_0)(x - x_1)dx = -\frac{f''(\xi)}{12}(x_1 - x_0)^3$$

- note that here we had to use the weighted mvt
- which leads us to this definition: the **quadrature formula for** $n = 1$ is (also known as the trapezoid rule for $h = x_1 - x_0$):

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}(f(x_0) + f(x_1)) - \frac{h^3}{12}f''(\xi)$$



the composite trapezoidal rule

- we can apply the trapezoidal rule for smaller lengths of segments
- or for far more data points
- considering $n+1$ data points $a = x_0, x_1, \ldots, x_n = b$, then we can write assuming **no uniform spacing**

$$\int_a^b f(x)dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx \approx \sum_{i=1}^{n} \frac{h_i}{2}(f(x_{i-1}) + f(x_i)), \quad h_i = x_i - x_{i-1}$$

- **special case:** uniform spacing or equidistant data points with $h = (b-a)/n$:

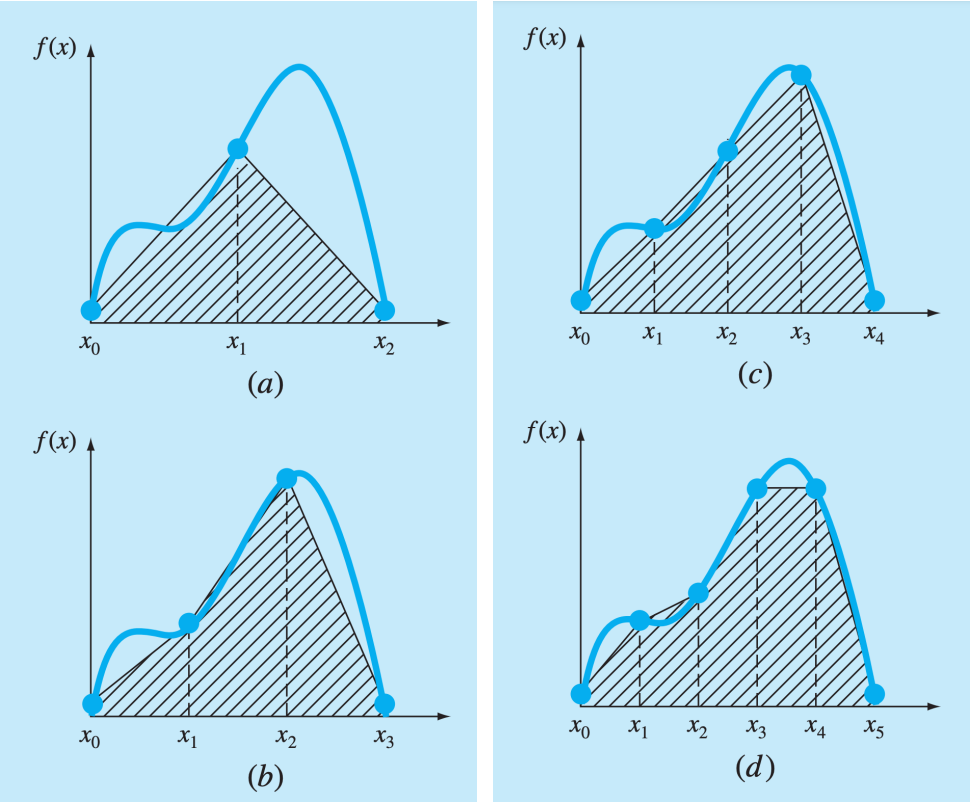
$$\int_a^b f(x)dx \approx h\left(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right) = \underbrace{(b-a)}_{\text{width}} \underbrace{\frac{f(x_0)+f(x_n)+2\sum_{i=1}^{n-1} f(x_i)}{2n}}_{\text{avg. height}}$$

- as for the error, we can perform this approximation $f''(\xi) \approx \frac{1}{n}\sum_{i=1}^{n} f''(\xi_i)$ leading to

$$\sum_{i=1}^{n} -\frac{h^3}{12} f''(\xi_i) \approx -f''(\xi)\frac{(b-a)^3}{12n^2}$$

illustration

- illustration of the composite trapezoidal rule with: (a) two segments, (b) three segments, (c) four segments, and (d) five segments:



- doubling segments (or n) leads to quartering the error

example

- use the two-segment trapezoidal rule on $[0, 0.8]$ to estimate the integral and error of integral of this function

$$f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$$

- **solution:** for $n = 2$, we compute $f(0) = 0.2$, $f(0.4) = 2.456$, and $f(0.8) = 0.232$ hence

$$\int_0^{0.8} f(x)dx \approx h\left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right) = 0.4\left(\frac{f(0) + f(0.8)}{2} + f(0.4)\right) = 1.0688$$

- the analytical solution yields $\int_0^{0.8} f(x) = 1.6405$
- btw, using $n = 1$ we get 0.1728 which is quite inaccurate
- for $n = 2$, error is still large (around 35%)
- note that $f''(x) = 25 - 400x + 3 \cdot 675x^2 + \ldots$
- to compute $f''(\xi)$, we think of (or approximate) $f''(\xi)$ as the average as the average value of $f''(x)$ over $[0, 0.8]$ which is -60 in this case
- can also compute the truncation error as

$$-f''(\xi)\frac{(b-a)^3}{12n^2} = -\frac{0.8^3}{12 \cdot 2^2}(-60) = 0.64$$

extending the example to larger n

TABLE 21.1 Results for multiple-application trapezoidal rule to estimate the integral of $f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$ from $x = 0$ to 0.8. The exact value is 1.640533.

n	h	I	ε_t (%)
2	0.4	1.0688	34.9
3	0.2667	1.3695	16.5
4	0.2	1.4848	9.5
5	0.16	1.5399	6.1
6	0.1333	1.5703	4.3
7	0.1143	1.5887	3.2
8	0.1	1.6008	2.4
9	0.0889	1.6091	1.9
10	0.08	1.6150	1.6

- first, note that this method is so easy to implement (3 lines of code)
- second, error progressively becomes better but as you increase n, you will see that the error starts to increase again due to round off errors. example:

Segments	Segment Size	Estimated d, m	ε_t (%)
10	1.0	288.7491	0.237
20	0.5	289.2636	0.0593
50	0.2	289.4076	9.5×10^{-3}
100	0.1	289.4282	2.4×10^{-3}
200	0.05	289.4336	5.4×10^{-4}
500	0.02	289.4348	1.2×10^{-4}
1000	0.01	289.4360	-3.0×10^{-4}
2000	0.005	289.4369	-5.9×10^{-4}
5000	0.002	289.4337	5.2×10^{-4}
10,000	0.001	289.4317	1.2×10^{-3}

quadrature formula for $n = 2$, simpson's 1/3 rule

- we can similarly derive or approximate areas under curves via second order lagrange polys (or newton divided diff too)
- so for three data points ($n = 2$), we can similarly derive both the area and the error assuming equidistant data points
- the approximate area under $f(x)$ for $x \in [a, b] = [x_0, x_2]$ is given by

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2)) - \underbrace{\frac{h^5}{90}}_{=\frac{(b-a)^5}{2880}} f^{(4)}(\xi)$$

- how did we obtain this equation? lagrange polys
- but alternatively, you could write taylor series approx for $f(x)$ around x_1 then integrate over $x \in [x_0, x_2]$
- this above equation is called **simpson's 1/3 rule**
- what does it really entail? here we see that the trap rule uses locally linear interpolating polys
- but with simpson, we utilize locally quadratic interpolating polys
- what happens if your function is a third degree polynomial?

simpson's 3/8 rule, quadrature formula for $n = 3$

- similarly, we can use $n = 3$ or cubic lagrange polys to approximate the integration
- this results in

$$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{8}(f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)) - \underbrace{\frac{3h^5}{80}}_{=\frac{(b-a)^5}{6480}} f^{(4)}(\xi)$$

- note that we can approximate the integral now with

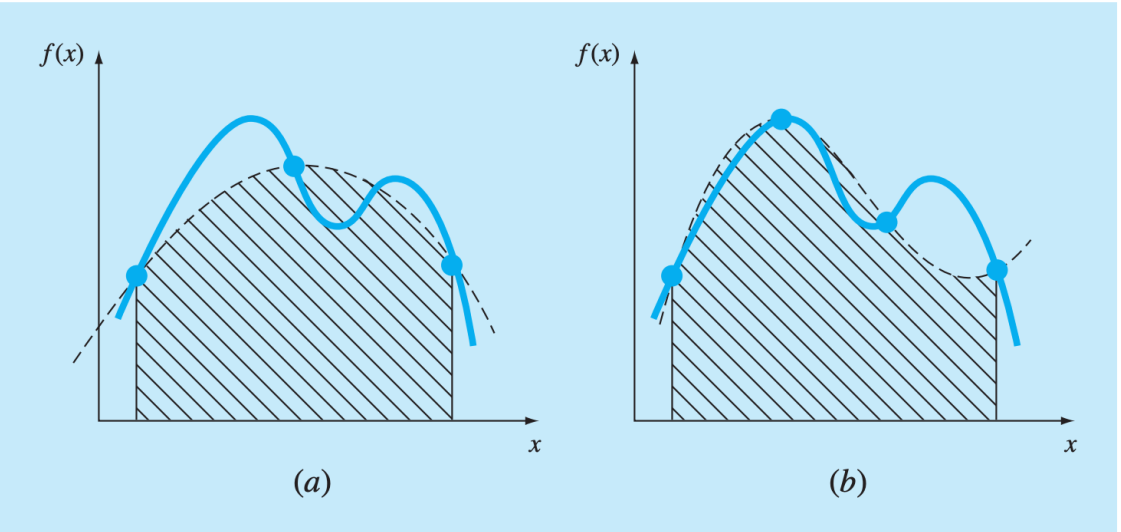
$$\int_{x_0}^{x_3} f(x)dx \approx \underbrace{(b-a)}_{\text{width}} \underbrace{\frac{1}{8}(f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3))}_{\text{avg. height}}$$

- the two interior points are given weights of three-eighths, whereas the end points are weighted with one-eight
- notice how the denominator in the error for 3/8 rule is larger than the denominator for 1/3 rule, so it yields accurate results

3/8 versus 1/3: discussions and insights

- simpson's 1/3 rule is usually the preference because it attains good accuracy with only 3 points
- with that in mind, when you have odd number of points, 3/8 rule is used
- now suppose you have 6 number of data points available to integrate $f(x)$
- to perform such an integration, you can use the 1/3 rule for the first 3 data points and 3/8 rule for the last 3 and simply add the approximated areas

FIGURE 21.10
(*a*) Graphical depiction of Simpson's 1/3 rule: It consists of taking the area under a parabola connecting three points. (*b*) Graphical depiction of Simpson's 3/8 rule: It consists of taking the area under a cubic equation connecting four points.



trap versus simpson

- compare trap versus simpson 1/3 for these functions over $[0, 2]$

$$f(x) = \{x, x^2, x^4, \frac{1}{x+1}, \sin(x)\}$$

- recall that for trap:
  $\int_{x_0}^{x_1} f(x)dx \approx \frac{1}{2}(x_1 - x_0)f(x_0) + \frac{1}{2}(x_0 - x_1)f(x_1) = f(0) + f(2)$
- and for simpson 1/3rd:

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2)) = \frac{1}{3}(f(0) + 4f(1) + f(2))$$

- we obtain the following results

function	x	x^2	x^4	$\frac{1}{x+1}$	$\sin(x)$
trap	2.0000	4.0000	16.0000	1.3333	0.9093
simp	2.0000	2.6667	6.6667	1.1111	1.4251
exact	2.0000	2.6667	6.4000	1.0986	1.4161

investigating impact of composition

- approximate $\int_0^4 e^x dx = 53.59815$ via simpson's formula and simpson's formula with 2 and 4 segments
- **solution:** for simpson 1/3rd rule (no composition):

$$\int_0^4 e^x dx = \frac{2}{3}(e^0 + 4e^2 + e^4) = 56.7695$$

- using 2 segments, we obtain

$$\int_0^4 e^x dx = \int_0^2 e^x dx + \int_2^4 e^x dx = \frac{1}{3}(e^0 + 4e^1 + e^2) + \frac{1}{3}(e^2 + 4e^3 + e^4) = 53.8638$$

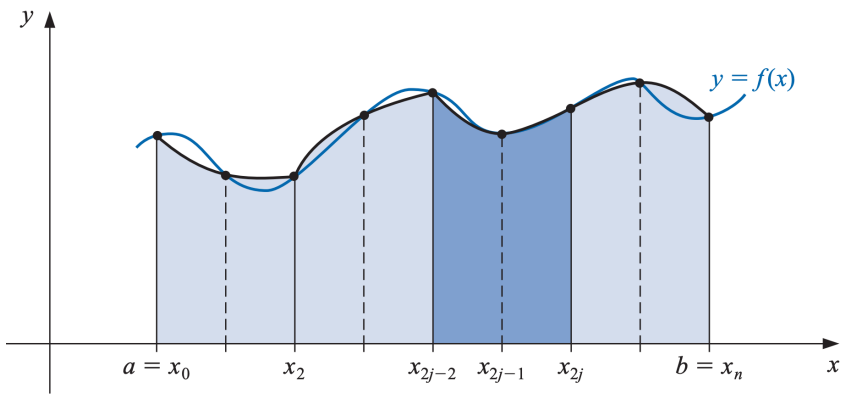
- using 4 segments, we obtain

$$\int_0^4 e^x dx = \int_0^1 e^x dx + \int_1^2 e^x dx + \int_2^3 e^x dx + \int_3^4 e^x dx$$

$$= \frac{1}{6}(e^0 + e^4 + 2e^1 + 2e^2 + 2e^3 + 4e^{0.5} + 4e^{1.5} + 4e^{2.5} + 4e^{3.5}) = 53.61622$$

- relative error improves with more compositions

generalization of simpson's 1/3 composition



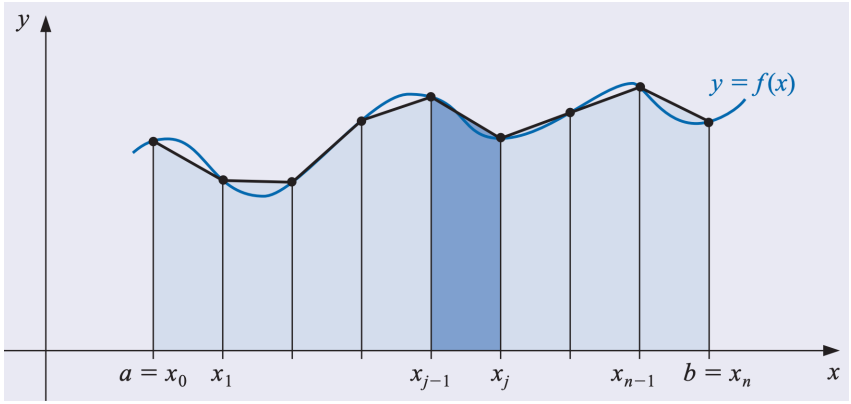
- to approximate $\int_a^b f(x)dx$, and using simpson's 1/3 rule, choose an even number n, divide $[a, b]$ into n equal intervals
- then apply simpson's 1/3 rule on each interval
- then sum the values to approximate the integral
- **generalization**:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(f(a) + f(b) + 2\sum_{j=1}^{\frac{n}{2}-1} f(x_{2j}) + 4\sum_{j=1}^{\frac{n}{2}} f(x_{2j-1})\right)$$

with n being even and $h = (b-a)/n$ and $x_j = a + jh$

- the error from this composition can be written as $e(f) = \frac{a-b}{180}h^4 f^{(4)}(\xi)$ where $\xi \in (a, b)$

composite trapezoidal rule



- composite trapezoidal rule requires only one interval for each application
- so n can be even or odd
- we can also generalize the composite trapezoidal rule as follows

$$\int_a^b f(x)dx \approx \frac{h}{2}\left(f(a) + f(b) + 2\sum_{j=1}^{n-1} f(x_j)\right)$$

with n being even or odd and $h = (b-a)/n$

- the error from this composition can be written as $e(f) = \frac{a-b}{12}h^2 f''(\xi)$ where $\xi \in (a, b)$

example on error bounds and composition

- for $I = \int_0^{\pi} \sin(x)dx$, obtain optimal h that ensures approximation error less than 0.00002 via composite trap and simpson's rules then determine minimum number of compositions/segments n
- for the composite simpson:

$$|e(f)| = \left| \frac{\pi}{180} h^4 \sin(\xi) \right| \leq \frac{\pi h^4}{180} < 0.0002 \quad \Rightarrow \quad h < 0.184$$

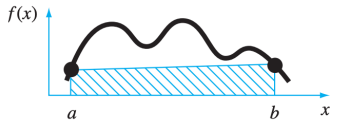
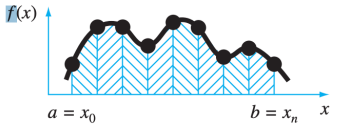
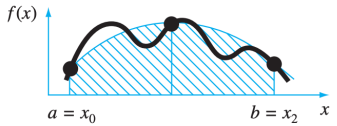
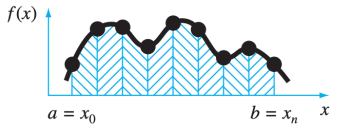
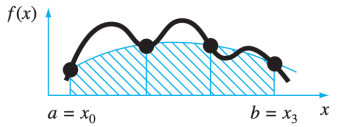
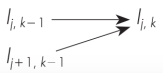
- because we segment $\frac{\pi}{n} < 0.814$ then $n > 17.08$
- similarly, for the trap composite method, we obtain

$$|e(f)| \leq \frac{\pi h^2}{12} < 0.0002 \quad \Rightarrow \quad h < 0.00874 \quad \Rightarrow \quad n > 359.4$$

- trap needs 360 subintervals whereas simpson only requires 18

summary of the integration methods

Method	Data Points Required for One Application	Data Points Required for n Applications	Truncation Error	Application	Programming Effort
Trapezoidal rule	2	$n + 1$	$\simeq h^3 f''(\xi)$	Wide	Easy
Simpson's 1/3 rule	3	$2n + 1$	$\simeq h^5 f^{(4)}(\xi)$	Wide	Easy
Simpson's rule (1/3 and 3/8)	3 or 4	≥ 3	$\simeq h^5 f^{(4)}(\xi)$	Wide	Easy
Higher-order Newton-Cotes	≥ 5	N/A	$\simeq h^7 f^{(6)}(\xi)$	Rare	Easy
Romberg integration	3			Requires $f(x)$ be known	Moderate

Method	Formulation	Graphic Interpretations	Error
Trapezoidal rule	$I \simeq (b-a)\frac{f(a)+f(b)}{2}$		$-\frac{(b-a)^3}{12}f''(\xi)$
Multiple-application trapezoidal rule	$I \simeq (b-a)\frac{f(x_0)+2\sum_{i=1}^{n-1} f(x_i)+f(x_n)}{2n}$		$-\frac{(b-a)^3}{12n^2}\bar{f}''$
Simpson's 1/3 rule	$I \simeq (b-a)\frac{f(x_0)+4f(x_1)+f(x_2)}{6}$		$-\frac{(b-a)^5}{2880}f^{(4)}(\xi)$
Multiple-application Simpson's 1/3 rule	$I \simeq (b-a)\frac{f(x_0)+4\sum_{i=1,3}^{n-1} f(x_i)+2\sum_{j=2,4}^{n-2} f(x_j)+f(x_n)}{3n}$		$-\frac{(b-a)^5}{180n^4}f^{(4)}$
Simpson's 3/8 rule	$I \simeq (b-a)\frac{f(x_0)+3f(x_1)+3f(x_2)+f(x_3)}{8}$		$-\frac{(b-a)^5}{6480}f^{(4)}(\xi)$
Romberg integration	$I_j, k = \frac{4^{k-1} I_{j+1,k-1} - I_{j,k-1}}{4^{k-1}-1}$		$O(h^{2k})$