

PHY604 Lecture 4

September 7, 2023

Review: Debugging tools

- Simplest debugging: print out information at intermediate points in code execution
- Running with appropriate compiler glags (e.g., `-g` for gnu compilers) can provide debugging information
 - Can make code run slower, but useful for test purposes
- Interactive debuggers let you step through your code line-by-line, inspect the values of variables as they are set, etc.
 - `gdb` is the version that works with the GNU compilers. Some graphical frontends exist.
 - Lots of examples online
 - Not very useful for parallel code.
- Particularly difficult errors to find often involve memory management
 - **Valgrind** is an automated tool for finding memory leaks. No source code modifications are necessary.

Review: Building your code with, e.g., Makefiles

- It is good style to separate your subroutines/functions into files, grouped together by purpose
 - Makes a project easier to manage (for you and version control)
 - Reduces compiler memory needs (although, can prevent inlining across files)
 - Reduces compile time—you only need to recompile the code that changed (and anything that might depend on it)
- Makefiles automate the process of building your code
 - No ambiguity of whether your executable is up-to-date with your changes
 - Only recompiles the code that changed (looks at dates)
 - Very flexible: lots of rules allow you to customize how to build, etc.
 - Written to take into account dependencies

Today's lecture:

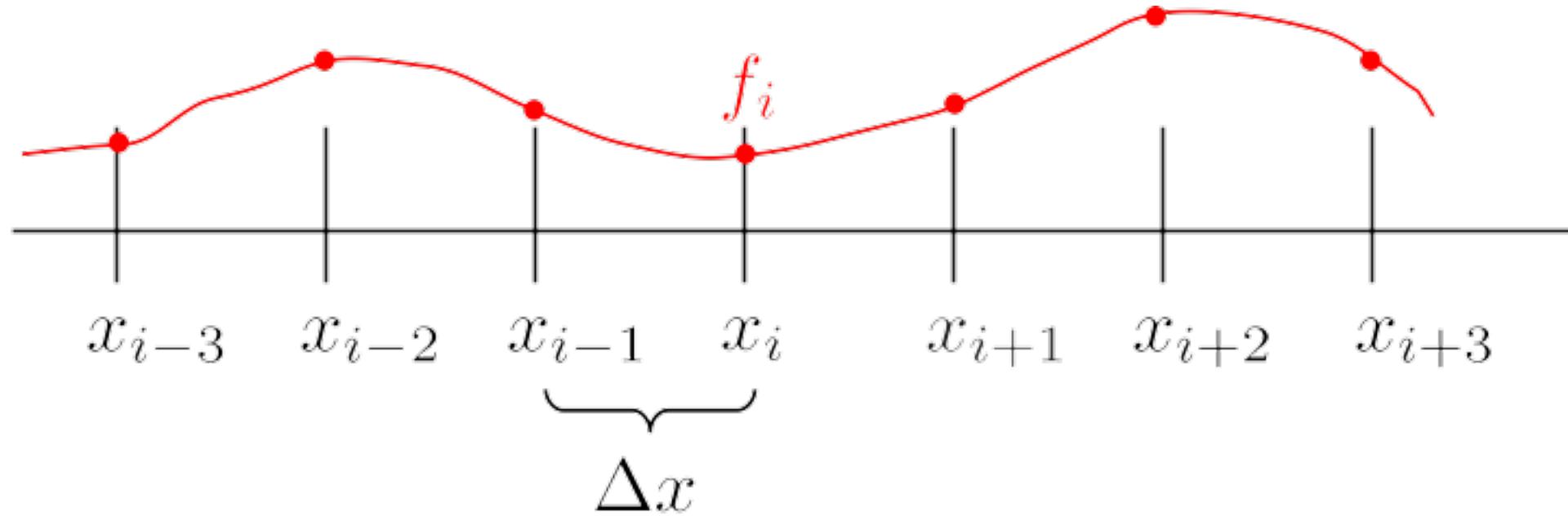
- Numerical differentiation
- Numerical integration

Numerical differentiation, Two situations:

- We have data defined only at a set of (possibly regularly spaced) points
 - Generally speaking, asking for **greater accuracy** for the derivative involves using **more of the discrete points**
- We have an analytic expression for $f(x)$ and want to compute the derivative numerically
 - If possible, it would be better to take the analytic derivative of $f(x)$, but we can learn something about error estimation in this case.
 - Used, for example, in computing the numerical Jacobian for integrating a system of ODEs (we'll see this later)

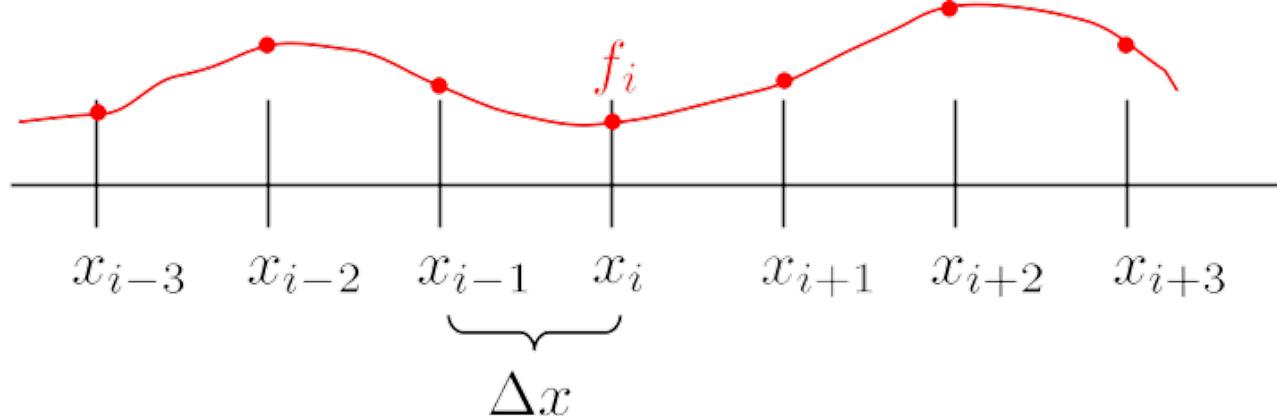
Gridded data

- Discretized data is represented at a finite number of locations
 - Integer subscripts are used to denote the position (index) on the grid
 - Structured/regular: spacing is constant



- Data is known only at the grid points: $f_i = f(x_i)$

First derivative



- Taylor expansion:

$$f_{i+1} = f(x_i + \Delta x) = f_i + \frac{df}{dx} \Bigg|_{x_i} \Delta x + \frac{1}{2} \frac{d^2 f}{dx^2} \Bigg|_{x_i} \Delta x^2 + \dots$$

- Solve for the first derivative:

$$\frac{df}{dx} \Bigg|_{x_i} = \frac{f_{i+1} - f_i}{\Delta x} - \frac{1}{2} \frac{d^2 f}{dx^2} \Bigg|_{x_i} \Delta x$$

↗ Discrete approx. of f' ↗ Leading term in the truncation error

Order of accuracy

$$\left. \frac{df}{dx} \right|_{x_i} = \frac{f_{i+1} - f_i}{\Delta x} - \left. \frac{1}{2} \frac{d^2 f}{dx^2} \right|_{x_i} \Delta x$$

- The accuracy of the finite difference approximation is determined by size of Δx
- So this finite difference expression is accurate to “order” Δx : $\mathcal{O}(\Delta x)$
- However: Making Δx small means that we are **subtracting numbers that are very close to each other**, which can result in significant rounding errors

Maximizing the accuracy

- Say we can evaluate the function to accuracy $C f(x)$ [also $C f(x+\Delta x)$]
 - For double precision: $C \simeq 10^{-16}$
- Worst-case rounding error on derivative is $2C|f(x)| / \Delta x$

- Also need to worry about associative errors: $(x + \Delta x) - x \stackrel{?}{=} \Delta x$

- So total error is: $\left| \frac{df}{dx} \Big|_{x_i} - \frac{f_{i+1} - f_i}{\Delta x} \right| \leq \frac{1}{2} \frac{d^2 f}{dx^2} \Big|_{x_i} \Delta x + \frac{2C|f_i|}{\Delta x}$

- We can minimize to find: $\Delta x = \sqrt{4C \left| \frac{f_i}{f''_i} \right|} \sim 10^{-8}$

- So “minimum” error: $\epsilon = \sqrt{4C |f_i f''_i|} \sim 10^{-8}$

Increasing accuracy with more points in the “stencil”

- First-order “forward” or “backward”:

$$f' = \frac{f_{i+1} - f_i}{\Delta x}$$

$$f' = \frac{f_i - f_{i-1}}{\Delta x}$$

2-point stencil

- Second-order “central”:

$$f' = \frac{-\frac{1}{2}f_{i-1} + 0f_i + \frac{1}{2}f_{i+1}}{\Delta x}$$

3-point stencil

Second-order central

- Consider two Taylor expansions:

$$f_{i+1} = f_i + \frac{df}{dx} \Bigg|_{x_i} \Delta x + \frac{1}{2} \frac{d^2 f}{dx^2} \Bigg|_{x_i} \Delta x^2 + \dots$$

$$f_{i-1} = f_i - \frac{df}{dx} \Bigg|_{x_i} \Delta x + \frac{1}{2} \frac{d^2 f}{dx^2} \Bigg|_{x_i} \Delta x^2 + \dots$$

- We see that:

$$\frac{df}{dx} \Bigg|_{x_i} = \frac{f_{i+1} - f_{i-1}}{2\Delta x} + \mathcal{O}(\Delta x^2) + \dots$$

Error in Second order central

$$\left| \frac{df}{dx} \Big|_{x_i} - \frac{f_{i+1} - f_{i-1}}{2\Delta x} \right| \leq \frac{1}{6} \frac{d^3 f}{dx^3} \Big|_{x_i} \Delta x^2 + \frac{C|f_i|}{\Delta x}$$

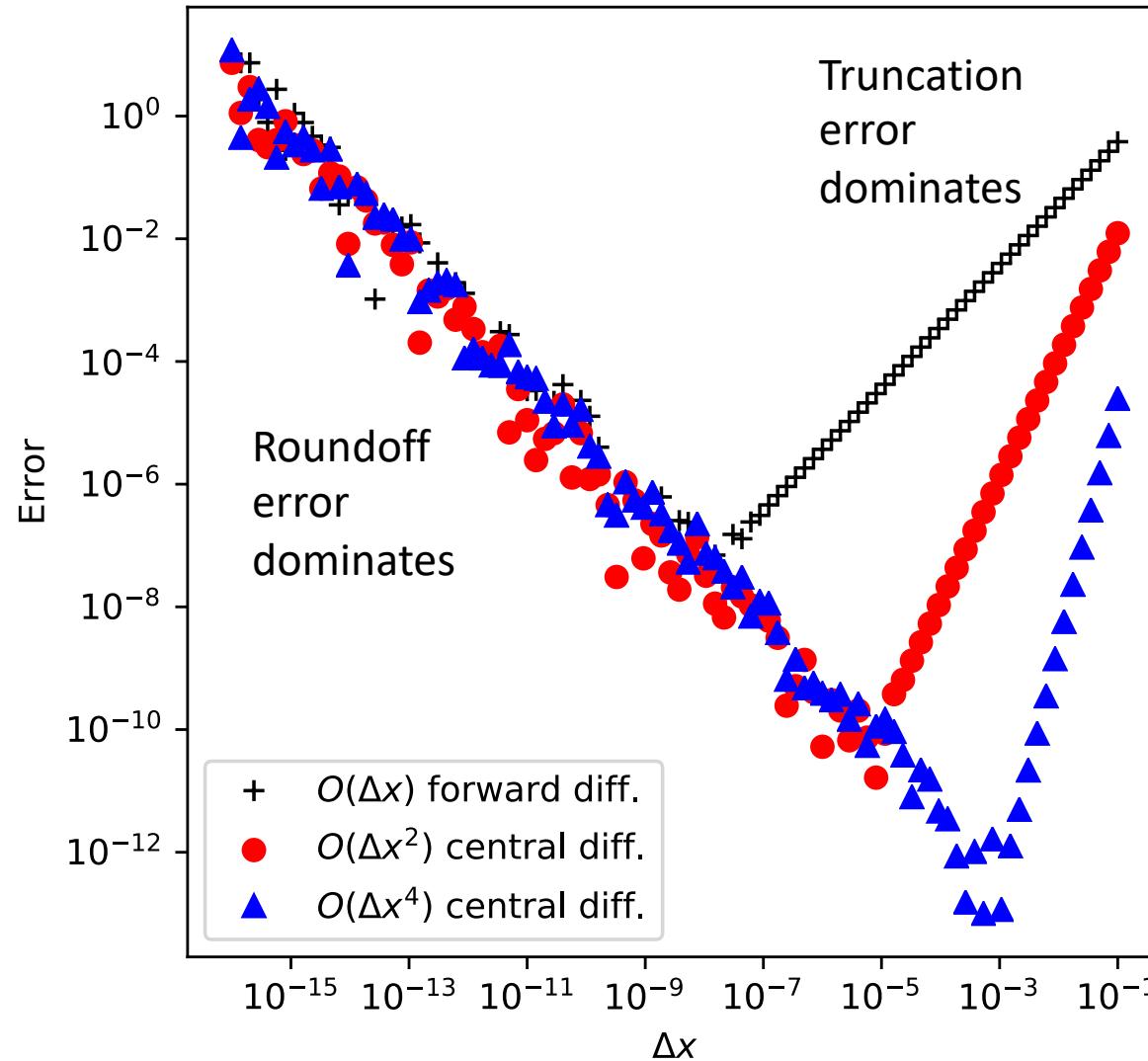
- Minimize WRT Δx : $\Delta x = \sqrt[3]{6C \left| \frac{f(x_i)}{f'''(x_i)} \right|} \sim 10^{-5}$
Assuming double prec.
- Minimum error: $\epsilon \propto \sqrt[3]{C^2 f(x_i)^2 |f'''(x_i)|} \sim 10^{-11}$
Assuming double prec.

Higher order first derivatives

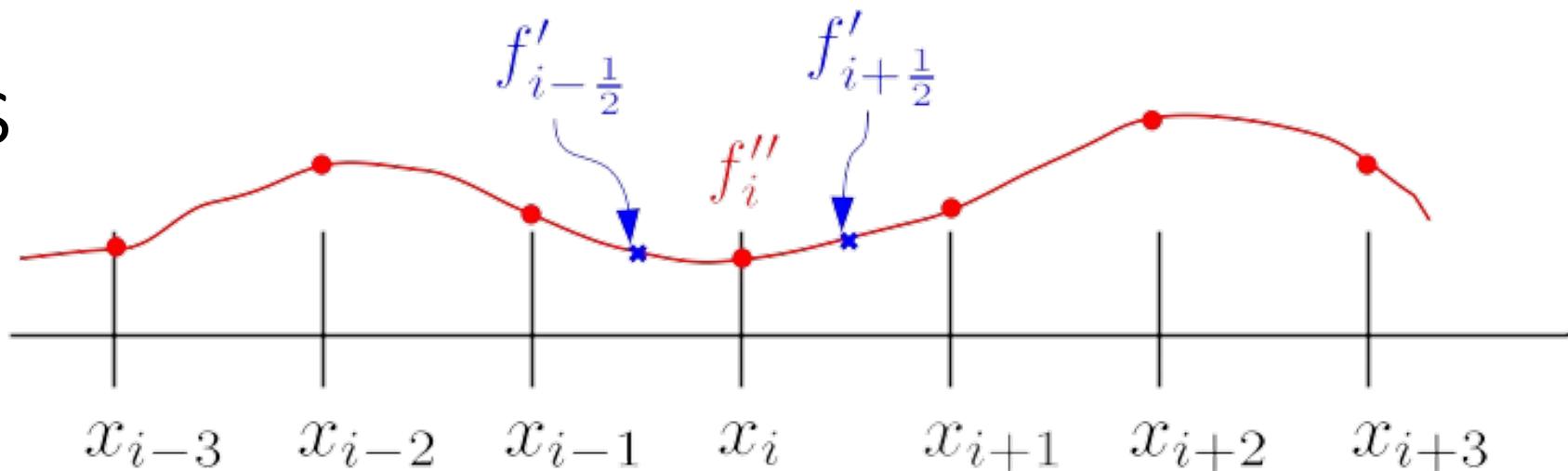
- To get accuracy to order n [i.e., $\mathcal{O}(\Delta x^n)$] follow a similar strategy:
 - 1. Write down Taylor expansion for $n+1$ finite difference points up to order $n+1$
 - 2. Solve set of polynomial equation in Δx for f'
 - 3. Obtain an expression involving weighted sum of function evaluated at $n+1$ points (some weights may be zero)
- Note: may be central, forward, or backward
- For example, for central:

Derivative	Accuracy	-5	-4	-3	-2	-1	0	1	2	3	4	5
1	2					-1/2	0	1/2				
	4				1/12	-2/3	0	2/3	-1/12			
	6			-1/60	3/20	-3/4	0	3/4	-3/20	1/60		
	8		1/280	-4/105	1/5	-4/5	0	4/5	-1/5	4/105	-1/280	

Example: Derivative of $\exp(x)$



Higher derivatives



- Write second derivative as:
$$f''_i = \frac{f'_{i+1/2} - f'_{i-1/2}}{\Delta x}$$
- Insert forward difference first derivatives, e.g.:
$$f'_i = \frac{f_{i+1} - f_i}{\Delta x}$$
- So we get:
$$f''_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}$$

Higher derivatives and error

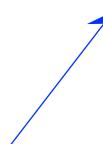
- We can also use the Taylor expansion strategy:

$$f_{i+1} = f_i + \Delta x f'_i + \frac{1}{2} \Delta x^2 f''_i + \frac{1}{6} \Delta x^3 f'''_i + \frac{1}{24} \Delta x^4 f''''_i + \dots$$

$$f_{i-1} = f_i - \Delta x f'_i + \frac{1}{2} \Delta x^2 f''_i - \frac{1}{6} \Delta x^3 f'''_i + \frac{1}{24} \Delta x^4 f''''_i + \dots$$

- Add together and rearrange: $f''_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} - \frac{1}{12} \Delta x^2 f''''_i$

$$\bullet \text{ Error: } \epsilon = \sqrt{\frac{4}{3} C |f_i f''''_i|} \sim 10^{-8}$$



Assuming double prec.

Partial and mixed derivatives

- Partial derivatives are a simple generalization
- E.g., central differences for function of two variables $f(x,y)$

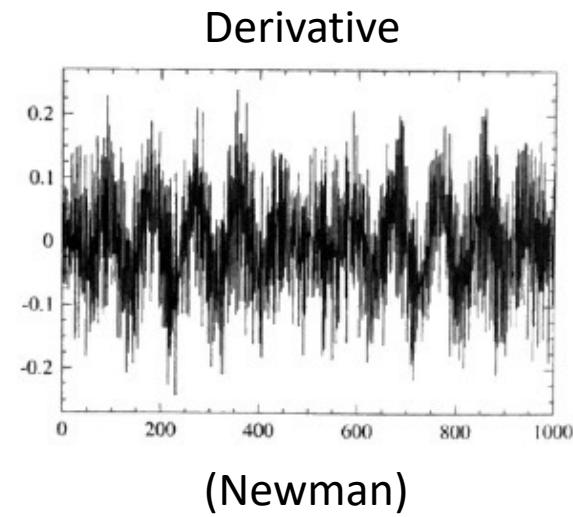
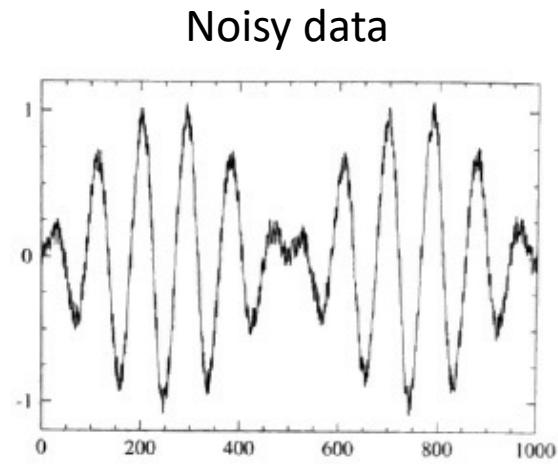
$$\frac{\partial f}{\partial x} = \frac{f(x + \Delta x, y) - f(x - \Delta x, y)}{2\Delta x} \quad \frac{\partial f}{\partial y} = \frac{f(x, y + \Delta y) - f(x, y - \Delta y)}{2\Delta y}$$

- Mixed second derivative:

$$\frac{\partial^2 f}{\partial y \partial x} = \frac{f(x + \Delta x, y + \Delta y) - f(x - \Delta x, y + \Delta y) - f(x + \Delta x, y - \Delta y) + f(x - \Delta x, y - \Delta y)}{4\Delta x \Delta y}$$

Some final comments on numerical derivation

- Taking derivatives of noisy data makes the noise much worse!
 - Fit to a smooth curve and take the derivative of that
 - Smooth the data, e.g., with a Fourier transform
- We can treat data on uneven grids with the same strategy as before, taking into account the different Δx 's between points



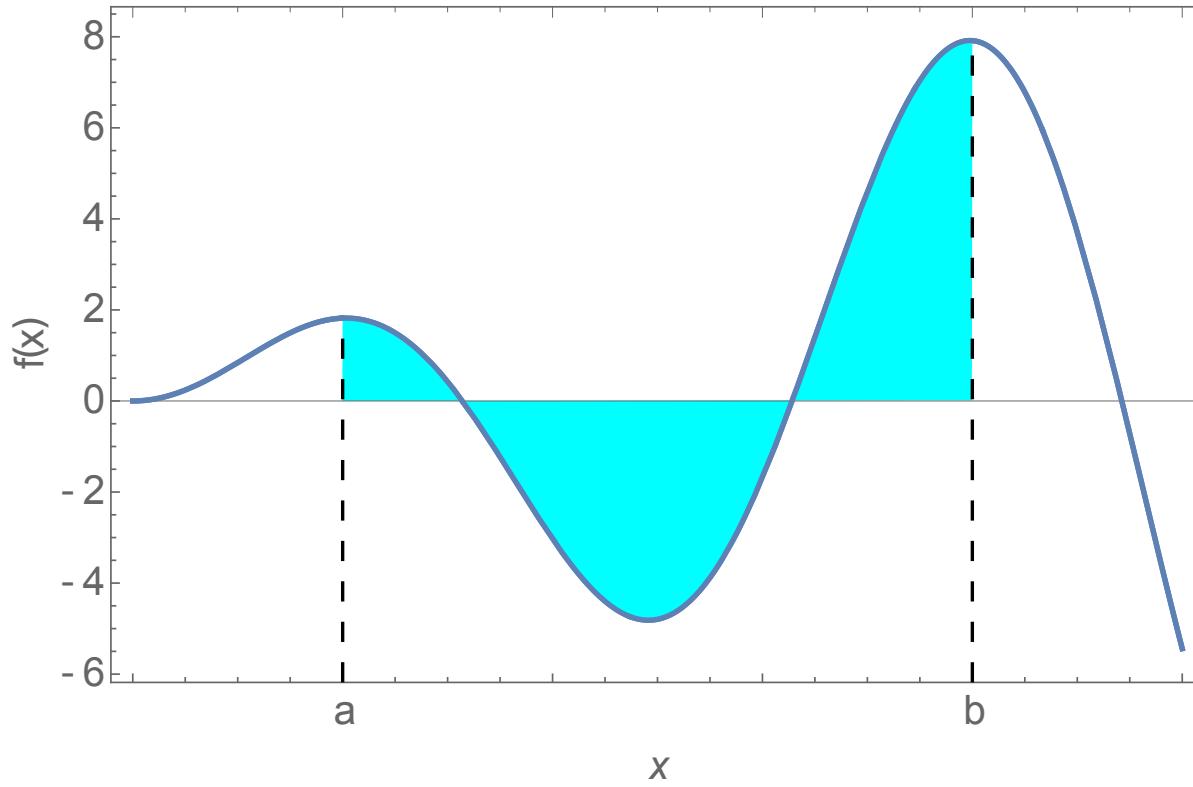
(Newman)

Today's lecture:

- Numerical differentiation
- Numerical integration

Numerical integration

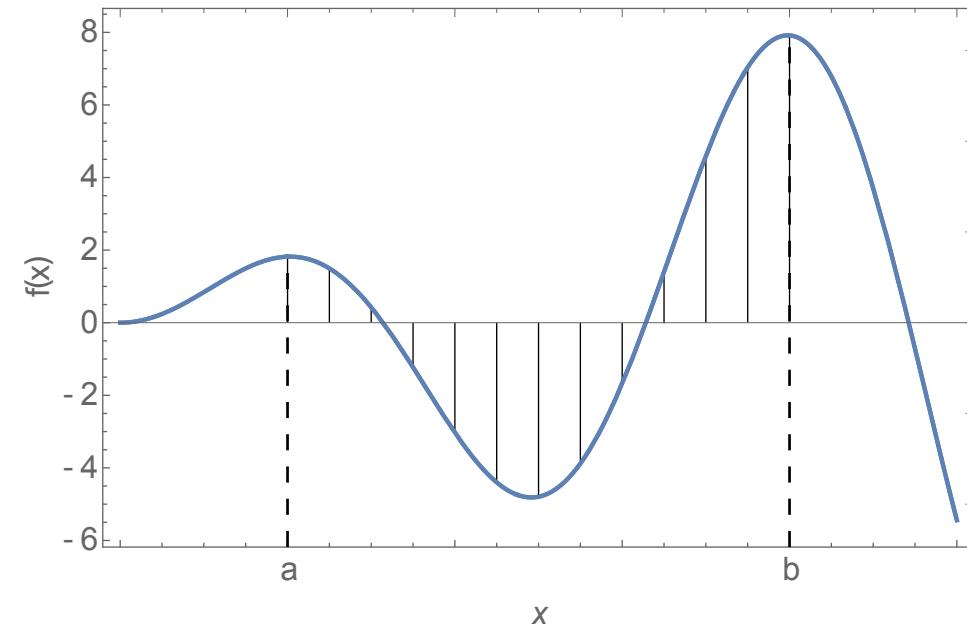
$$\int_a^b f(x)dx$$



Strategy for numerical integration:

- **Quadrature rule**: method that represents the integral as a (weighted) sum at a discrete number of points
 - **Newton-Cotes quadrature**: Fixed spacing between points
- 1. Discretize: Break up the interval into sub-intervals
- 2. Approximate the area under the curve in a subinterval by a simple polygon (rectangle, trapezoid) or a simple function (polynomial)
- 3. Sum the areas of the subintervals
- 4. Converge the integral by making more and more subintervals or using a more sophisticated weighting method

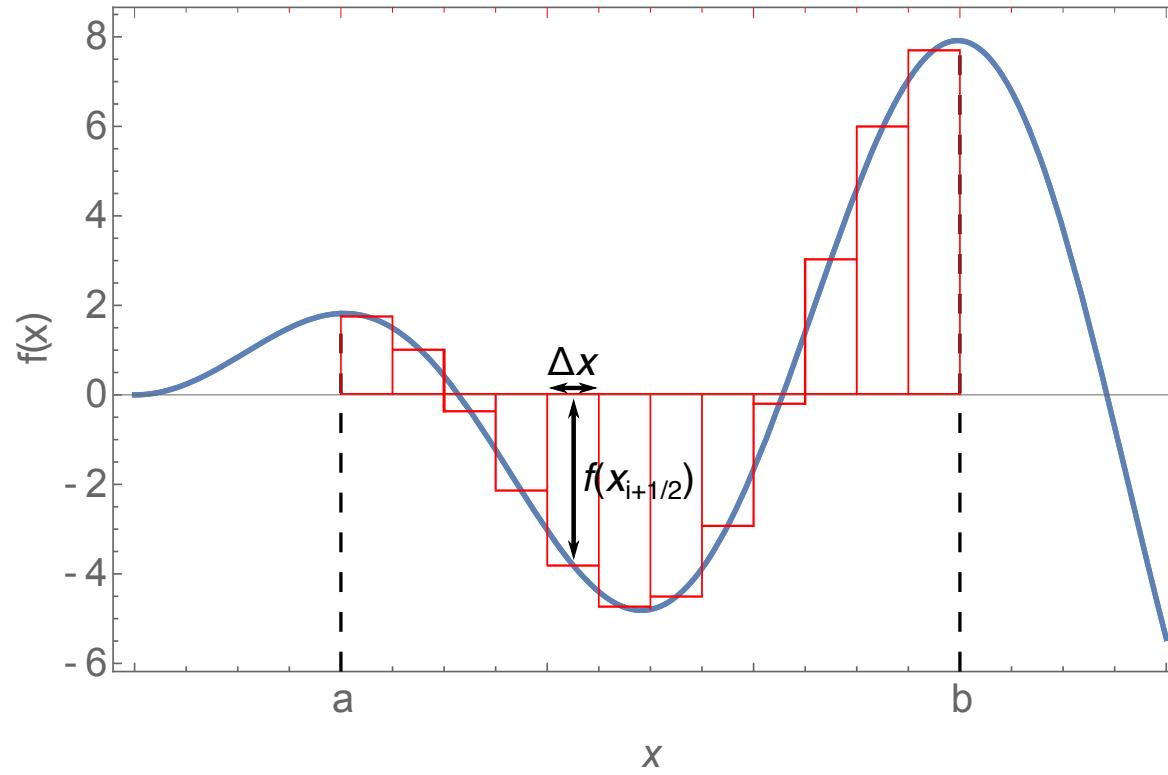
$$\int_a^b f(x)dx = \lim_{N \rightarrow \infty} \sum_{i=0}^{N-1} A_i$$



Approach 1: Midpoint rule

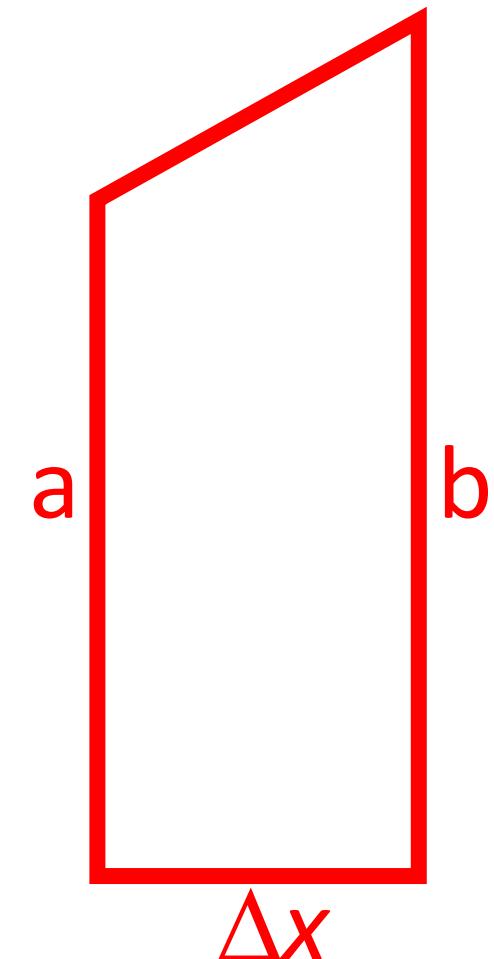
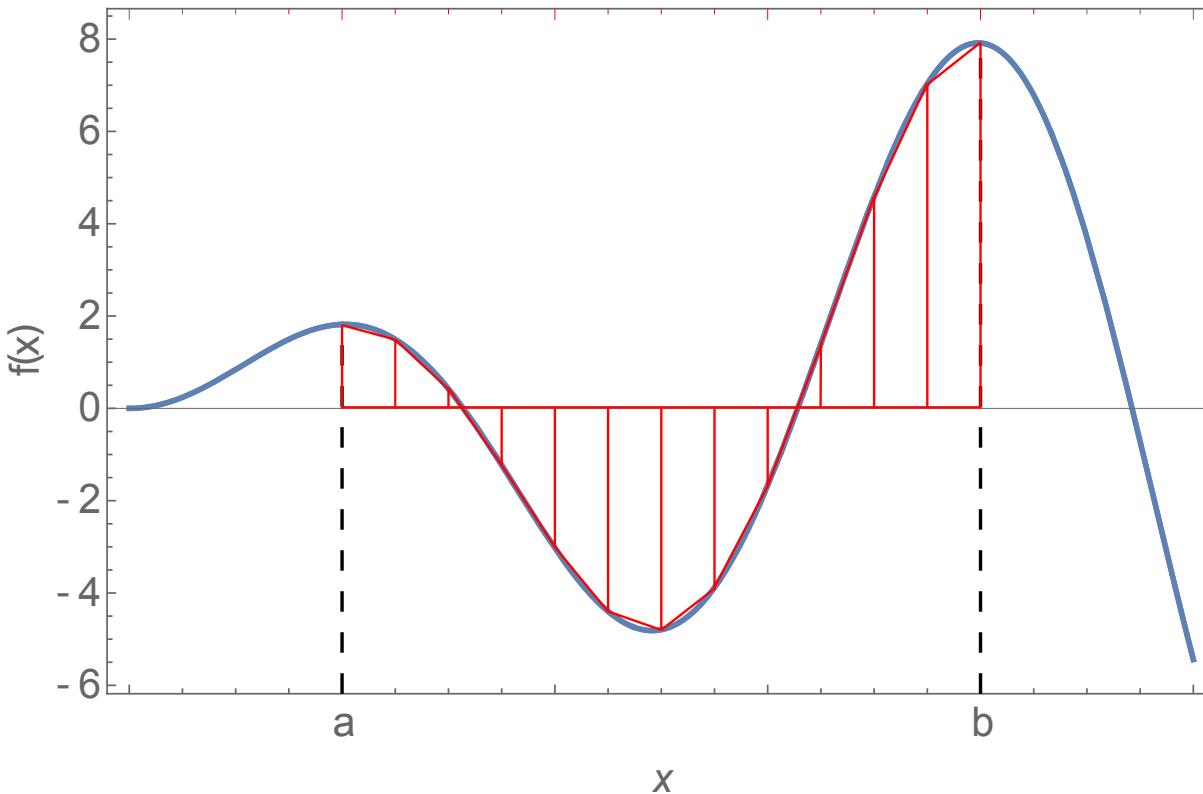
- Approximate area as rectangle with height equal to the midpoint of the subinterval $f(x_{i+1/2})$ and width Δx :

$$\int_a^b f(x)dx \simeq \lim_{N \rightarrow \infty} \sum_{i=0}^{N-1} \Delta x f(x_{i+\frac{1}{2}})$$



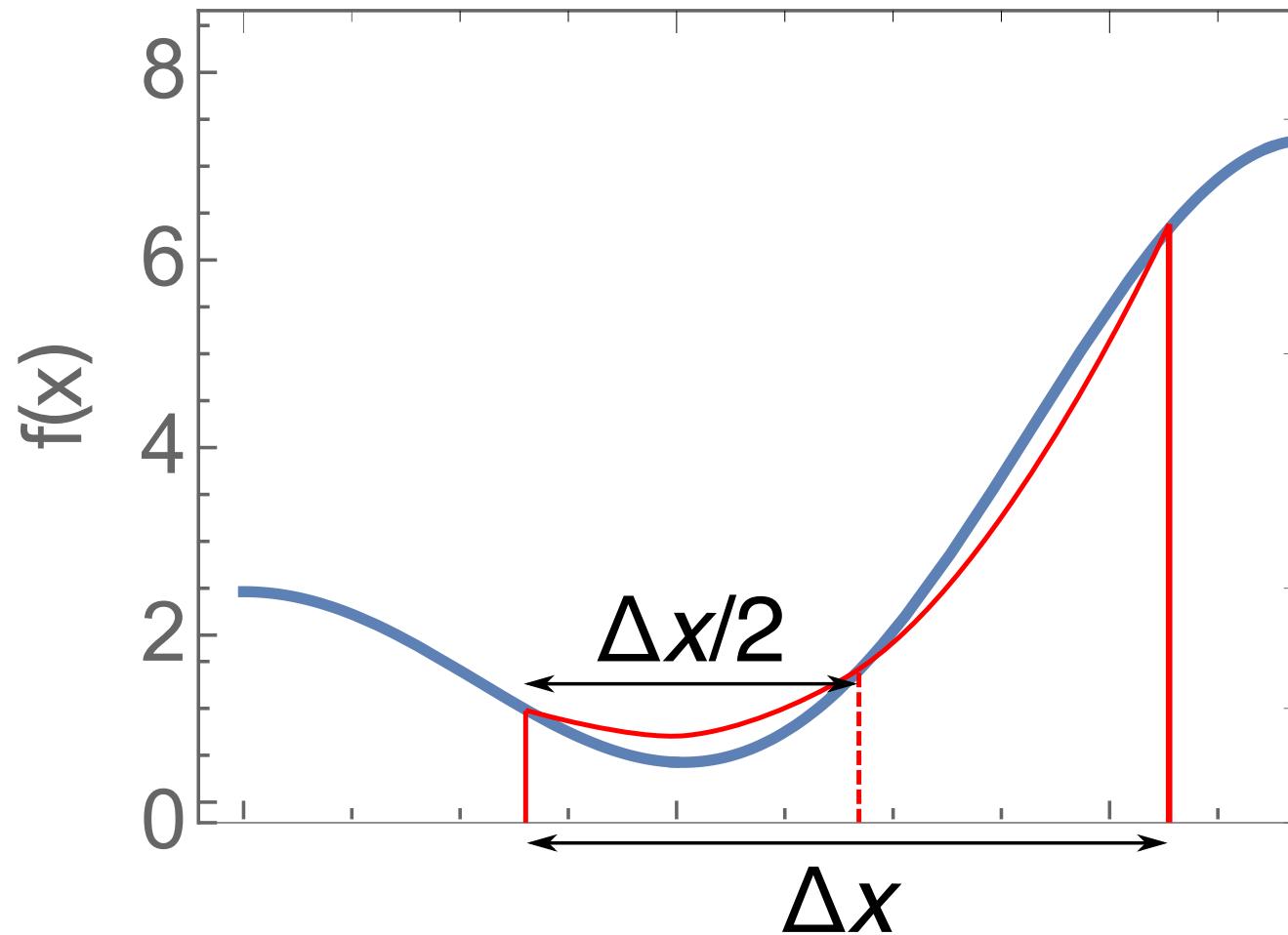
Approach 2: Trapezoid rule

- Area of subintervals approximated as a trapezoid with subinterval endpoints on the curve
- Area of trapezoid: $\Delta x(a+b)/2$



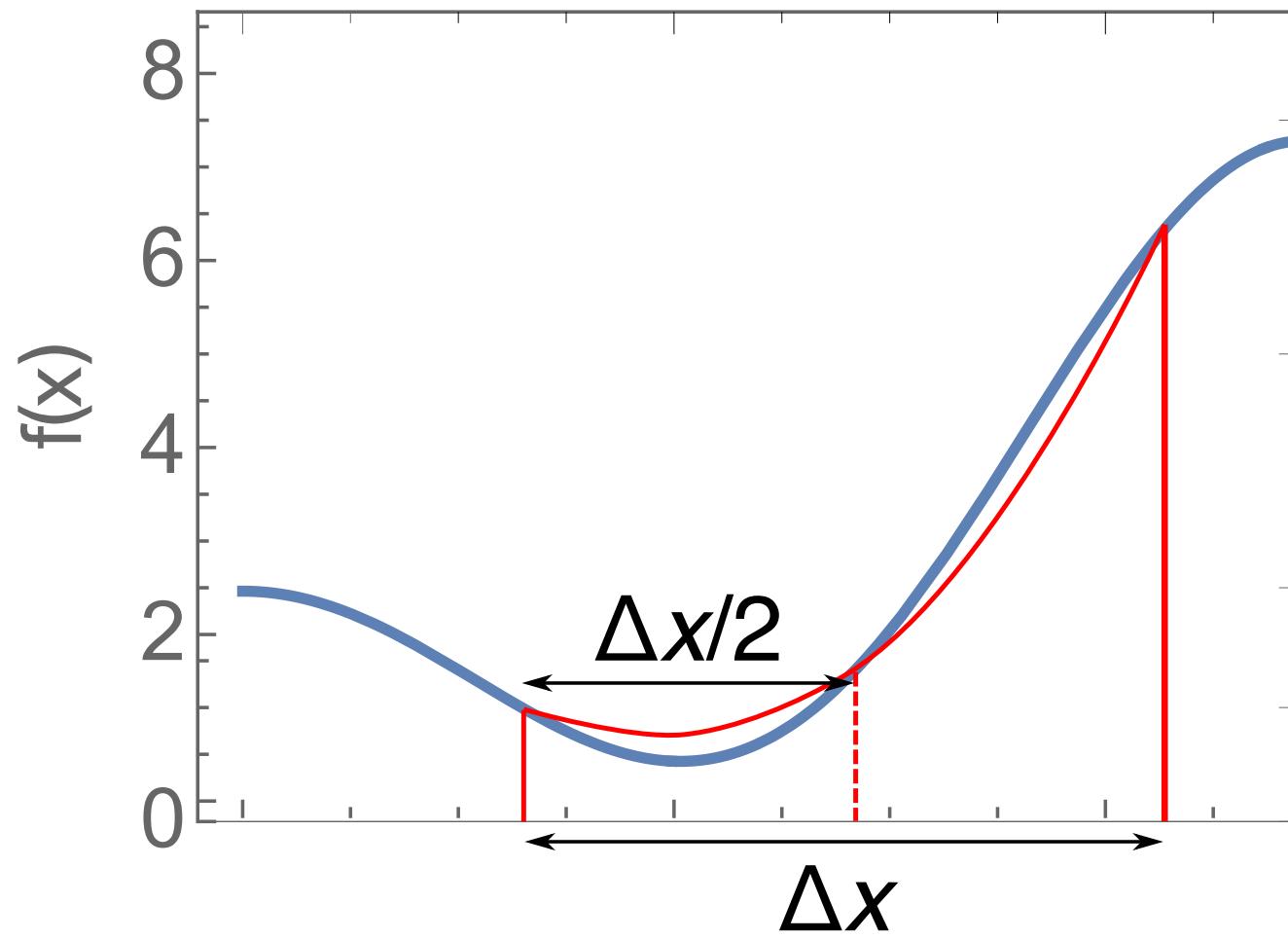
A more accurate technique: Simpson's Rule

- Approximate area of each subinterval by area under a parabola passing through points $f(x_i), f(x_{i+1/2}), f(x_{i+1})$



A more accurate technique: Simpson's Rule

$$\int_a^b f(x)dx \simeq \lim_{N \rightarrow \infty} \sum_{i=0}^{N-1} \Delta x \frac{f(x_i) + 4f(x_{i+\frac{1}{2}}) + f(x_{i+1})}{6}$$



Where does Simpson's rule come from?

- Consider the parabolic curve:

$$g(x) = Ax^2 + Bx + C$$

- We require it passes through the endpoints and midpoint of our function $f(x)$:

$$g(x_i) = Ax_i^2 + Bx_i + C = f(x_i)$$

$$g(x_{i+\frac{1}{2}}) = Ax_{i+\frac{1}{2}}^2 + Bx_{i+\frac{1}{2}} + C = f(x_{i+\frac{1}{2}})$$

$$g(x_{i+1}) = Ax_{i+1}^2 + Bx_{i+1} + C = f(x_{i+1})$$

- Solve for A, B, C

$$g(x) = f(x_i) \frac{(x - x_{i+\frac{1}{2}})(x - x_{i+1})}{(x_i - x_{i+\frac{1}{2}})(x_i - x_{i+1})} + f(x_{i+\frac{1}{2}}) \frac{(x - x_i)(x - x_{i+1})}{(x_{i+\frac{1}{2}} - x_i)(x_{i+\frac{1}{2}} - x_{i+1})} + f(x_{i+1}) \frac{(x - x_i)(x - x_{i+\frac{1}{2}})}{(x_{i+1} - x_i)(x_{i+1} - x_{i+\frac{1}{2}})}$$

Where does Simpson's rule come from?

$$g(x) = f(x_i) \frac{(x - x_{i+\frac{1}{2}})(x - x_{i+1})}{(x_i - x_{i+\frac{1}{2}})(x_i - x_{i+1})} + f(x_{i+\frac{1}{2}}) \frac{(x - x_i)(x - x_{i+1})}{(x_{i+\frac{1}{2}} - x_i)(x_{i+\frac{1}{2}} - x_{i+1})} + f(x_{i+1}) \frac{(x - x_i)(x - x_{i+\frac{1}{2}})}{(x_{i+1} - x_i)(x_{i+1} - x_{i+\frac{1}{2}})}$$

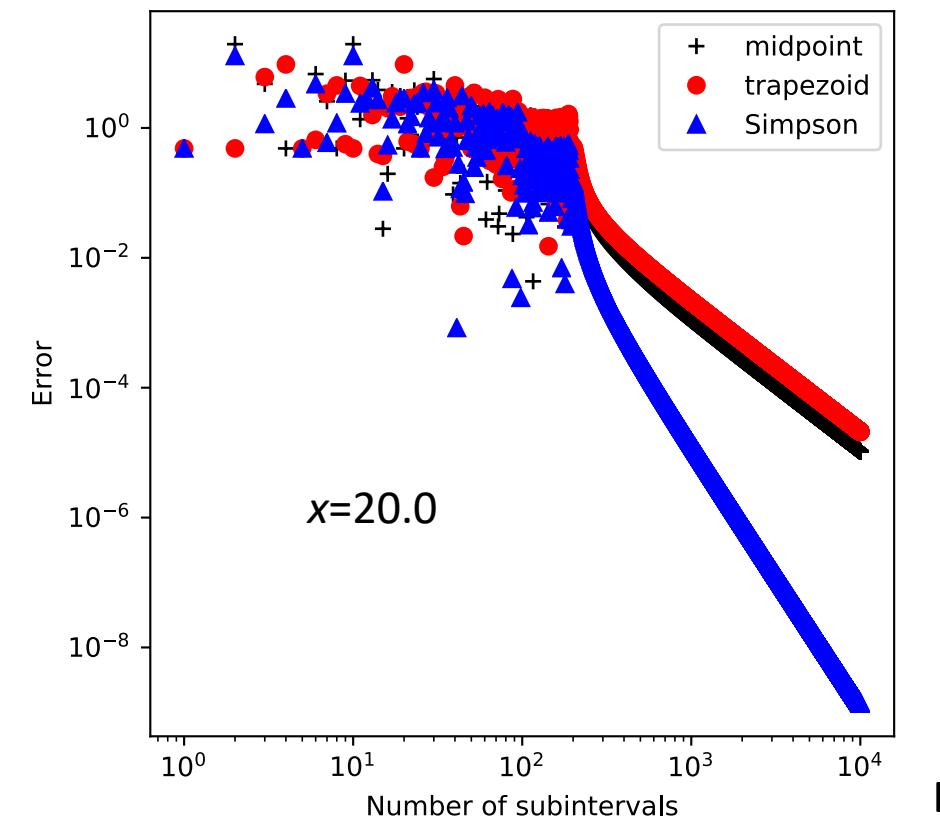
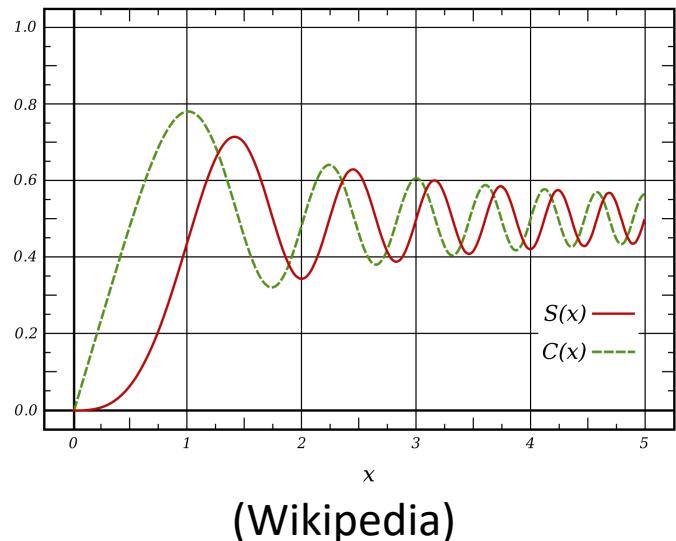
- Now we integrate over the subinterval:

$$\int_{x_i}^{x_{i+1}} g(x) dx = \frac{x_i - x_{i+1}}{6} \left[f(x_i) + 4f(x_{i+\frac{1}{2}}) + f(x_{i+1}) \right]$$

Example: Evaluating the Fresnel integral

- Fresnel functions are used in optics to describe near-field diffraction
- They can be written as an integral (or infinite sum):

$$S(x) = \int_0^x \sin(\pi t^2/2) dt$$



Errors in NC quadrature integration

- Error can be reduced by increasing the order of the polynomial or increasing the number of subintervals
- We can estimate errors in a similar way as we did for numerical differentiation (Taylor expand around points and take integrals), see, e.g., Newman Section 5.2.
 - For example, for the trapezoid rule:

$$\epsilon = \frac{1}{12} \Delta x^2 [f'(a) - f'(b)]$$

- First term in Euler-Maclaurin formula
- Simpson's rule is $O(\Delta x^4)$
- If we know the derivatives at the endpoints, we can calculate the error

Adaptive integration

- If we do not know $f'(x)$, we can still estimate the error:
 - 1. Perform the integration with N_1 and $N_2=2*N_1$ subintervals
 - 2. For, e.g., the trapezoid rule, the error using N_1 will be four times that using N_2
 - 3. The “exact” result, I is: $I = I_1 + c\Delta x_1^2 = I_2 + c\Delta x_2^2$
 - 4. Then the error on the second estimate is:
$$\epsilon_2 = c\Delta x_2^2 = \frac{1}{3}(I_2 - I_1)$$
- We can use this approach to decide when our integral is converged to our satisfaction
 - Keep doubling the number of subintervals until the error is small enough
 - Can use the results from previous function evaluations (See Newman Sec. 5.3 and 5.4 or Garcia Sec. 10.2)

Romberg Integration

- If i indicates a step in the procedure on the previous slide (i.e., doubling the number of subintervals), then we can write the integral as:

$$I = I_i + \frac{1}{3}(I_i - I_{i-1}) + \mathcal{O}(\Delta x^4)$$

- Equivalent to Simpson's rule!
- For every additional step (doubling of subintervals), we can build more and more accurate estimates
- See Newman Sec. 5.4 or Garcia Sec. 10.2 for more details

Dealing with infinity as a limit (Newman Sec. 5.8)

- Say we need to integrate over half of the number line:

$$I = \int_0^\infty f(x)dx$$

- It is impractical to simply increase the upper bound until convergence
- Instead, make a change of variables:

$$z \equiv \frac{x}{x+1} \iff x = \frac{z}{1-z}$$

$$dx = \frac{dz}{(1-z)^2}$$

- So the integral is:

$$I = \int_0^1 \frac{f\left(\frac{z}{1-z}\right)}{(1-z)^2} dz$$

Beyond Newton-Cotes: Gaussian Quadrature

- As an extra degree of freedom, lets vary the space between integration points
- We must first determine integration rules for unequal spacing
 - How do we determine weights?

$$\int_a^b f(x)dx \simeq w_1 f(x_1) + \dots + w_N f(x_N)$$

- Then, we choose a particular optimal choice of nonuniform points
- Many types of Gaussian quadrature

Theorem behind Gaussian integration

- Let $q(x)$ be a polynomial of degree N such that:

$$\int_a^b q(x)\rho(x)x^k dx = 0$$

- $k=0, \dots, N-1$ and $\rho(x)$ is a specified weight function
- Choose x_1, x_2, \dots, x_N as the roots of the polynomial $q(x)$, and use them as grid points:

$$\int_a^b f(x)\rho(x)dx \simeq w_1 f(x_1) + w_2 f(x_2) + \dots + w_N f(x_N)$$

- There exists a set of w 's where this formula is exact if $f(x)$ is a polynomial of degree $< 2N$ (!!!)
- Note that with N values, we can fit an $N-1$ degree polynomial and derive an integration formula exact for polynomials of order $< N$
 - Very accurate for curves well approximated as high-degree polynomials
- Many choices of weighting function, $\rho(x)$, leading to different q 's and x 's and w 's

Example from Garcia Sec. 10.3: Three-point Gauss-Legendre rule

- Three-point: Three grid points in the interval [-1,1]
 - $q(x)$ is cubic
- Take as the weight function $\rho(x)=1$ (Gauss-Legendre)
- We can convert an arbitrary interval $[a,b]$ to [1,-1]:

$$x = \frac{1}{2}(b+a) + \frac{1}{2}(b-a)z \iff z = \frac{x - \frac{1}{2}(b+a)}{\frac{1}{2}(b-a)}$$

$$dx = \frac{1}{2}(b-a)dz$$

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^1 f(z)dz$$

Step 1: Find polynomial $q(x)$

$$q(x) = c_0 + c_1x + c_2x^2 + c_3x^3$$

- Apply the theorem to get three equations for the coefficients:

$$\left. \begin{array}{l} \int_{-1}^1 q(x)dx = 0 \\ \int_{-1}^1 xq(x)dx = 0 \\ \int_{-1}^1 x^2q(x)dx = 0 \end{array} \right\}$$

General Solution:

$$c_0 = 0, \quad c_1 = -a, \quad c_2 = 0, \quad c_3 = 5a/3$$

- a is an arbitrary constant, if we take it to be $3/2$, we get the Legendre polynomial $P_3(x)$:

$$q(x) = \frac{5}{2}x^3 - \frac{3}{2}x$$

Step 2: Find the roots

$$q(x) = \frac{5}{2}x^3 - \frac{3}{2}x$$

- Easily factors to:

$$x = 0, \pm\sqrt{\frac{3}{5}}$$

- So our quadrature becomes:

$$\int_{-1}^1 f(x)dx \simeq w_1 f(-\sqrt{3/5}) + w_2 f(0) + w_3 f(\sqrt{3/5})$$

Step 3: Find the weights

- The theorem tells us that this quadrature is exact for polynomials up to degree $2N-1$

- Start with $f(x)=1$:
$$\int_{-1}^1 dx = 2 = w_1 + w_2 + w_3$$

- Now $f(x)=x$:
$$\int_{-1}^1 xdx = 0 = -\sqrt{3/5}w_1 + \sqrt{3/5}w_3$$

- Finally $f(x)=x^2$:
$$\int_{-1}^1 x^2dx = \frac{2}{3} = \frac{3}{5}w_1 + \frac{3}{5}w_3$$

- Solve to get: $w_1 = \frac{5}{9}, \quad w_2 = \frac{8}{9}, \quad w_3 = \frac{5}{9}$

Put it together:

3 point Gauss-Legendre quadrature

$$\int_{-1}^1 f(x)dx \simeq \frac{5}{9}f(-\sqrt{3/5}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{3/5})$$

Example: Error function

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-y^2} dy$$

- Evaluate $\operatorname{erf}(1)$:

Exact: 0.8427007929497148

3-point Trapezoid: 0.8252629555967492 , Error: -0.017437837352965557

3-point Simpsons: 0.843102830042981 , Error: 0.0004020370932662498

3-point Gauss-Legendre: 0.8426900184845107 , Error: -1.0774465204033135e-05

Example: 5th degree polynomial

$$I = \int_0^1 (1 + x^2 + x^3 + x^4 + x^5) dx$$

Exact: 2.449999999999997

3-point Trapezoid: 2.734375 , Error: 0.28437500000000027

3-point Simpsons: 2.4791666666666665 , Error: 0.0291666666666785

3-point Gauss-Legendre: 2.45 , Error: 4.440892098500626e-16

Weights and positions have been tabulated

- From Newman Sec. 5.6:

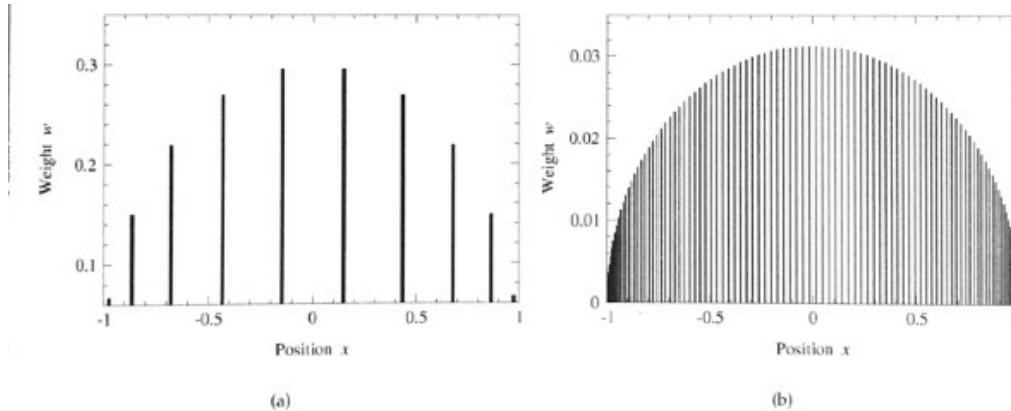


Figure 5.4: Sample points and weights for Gaussian quadrature. The positions and heights of the bars represent the sample points and their associated weights for Gaussian quadrature with (a) $N = 10$ and (b) $N = 100$.

- From Garcia 10.3:

Table 10.7: Grid points and weights for Gauss-Legendre integration.

$\pm x_i$	w_i	$\pm x_i$	w_i
$N = 2$		$N = 8$	
0.5773502692	1.0000000000	0.1834346425	0.3626837834
$N = 3$		0.5255324099	0.3137066459
0.0000000000	0.8888888889	0.7966664774	0.2223810345
0.7745966692	0.5555555556	0.9602898565	0.1012285363
$N = 4$		$N = 12$	
0.3399810436	0.6521451549	0.1252334085	0.2491470458
0.8611363116	0.3478548451	0.3678314990	0.2334925365
$N = 5$		0.5873179543	0.2031674267
0.0000000000	0.5688888889	0.7699026742	0.1600783285
0.5384693101	0.4786286705	0.9041172564	0.1069393260
0.9061798459	0.2369268850	0.9815606342	0.0471753364

Types of Gaussian Quadrature

Interval	$\omega(x)$	Orthogonal polynomials	A & S	For more information, see ...
$[-1, 1]$	1	Legendre polynomials	25.4.29	Section Gauss–Legendre quadrature , above
$(-1, 1)$	$(1 - x)^\alpha(1 + x)^\beta, \quad \alpha, \beta > -1$	Jacobi polynomials	25.4.33 ($\beta = 0$)	Gauss–Jacobi quadrature
$(-1, 1)$	$\frac{1}{\sqrt{1 - x^2}}$	Chebyshev polynomials (first kind)	25.4.38	Chebyshev–Gauss quadrature
$[-1, 1]$	$\sqrt{1 - x^2}$	Chebyshev polynomials (second kind)	25.4.40	Chebyshev–Gauss quadrature
$[0, \infty)$	e^{-x}	Laguerre polynomials	25.4.45	Gauss–Laguerre quadrature
$[0, \infty)$	$x^\alpha e^{-x}$	Generalized Laguerre polynomials		Gauss–Laguerre quadrature
$(-\infty, \infty)$	e^{-x^2}	Hermite polynomials	25.4.46	Gauss–Hermite quadrature

(Wikipedia)

- Roots and weights are tabulated, so no need to compute them

Choosing an integration method

(Newman Sec. 5.7)

- Trapezoid method:
 - Trivial to program
 - Equally spaced points, often true of experimental data
 - Good choice for poorly behaved data (noisy, singularities)
 - Adaptive method gives guaranteed accuracy level
 - Not very accurate for given number of points
- Romberg integration:
 - Equally spaced points, often true of experimental data
 - Guaranteed accuracy level
 - Potentially high accuracy for small number of points
 - Will not work well for noisy or pathological data/integrands
- Gaussian Quadrature
 - Potentially high accuracy for small number of points
 - Simple to program (weights and roots tabulated)
 - Will not work well for noisy or pathological data/integrands
 - Need to have data on specific, unequally-spaced grid

After class tasks

- If you do not already have one, make an account on github:
<https://github.com/>
- Homework and instructions for turning it in will be posted later today
- Readings:
 - [Wikipedia article on makefiles](#)
 - [Fortran best practices](#)
 - [Good Enough Practices in Scientific Computing](#)
 - [Blog on numerical differentiation](#)
 - [Wikipedia page of finite difference coefficients](#)
 - Newman Chapter 5
 - Garcia Section 10.2