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

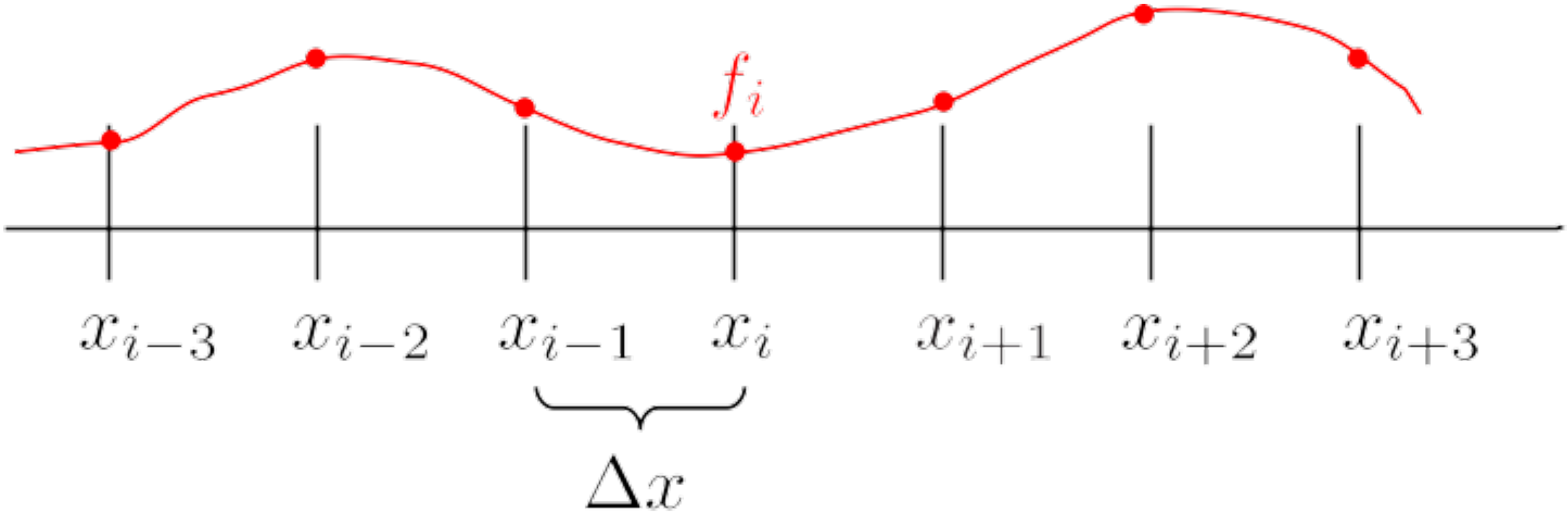
Numerical differentiation, Two situations:

- 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)

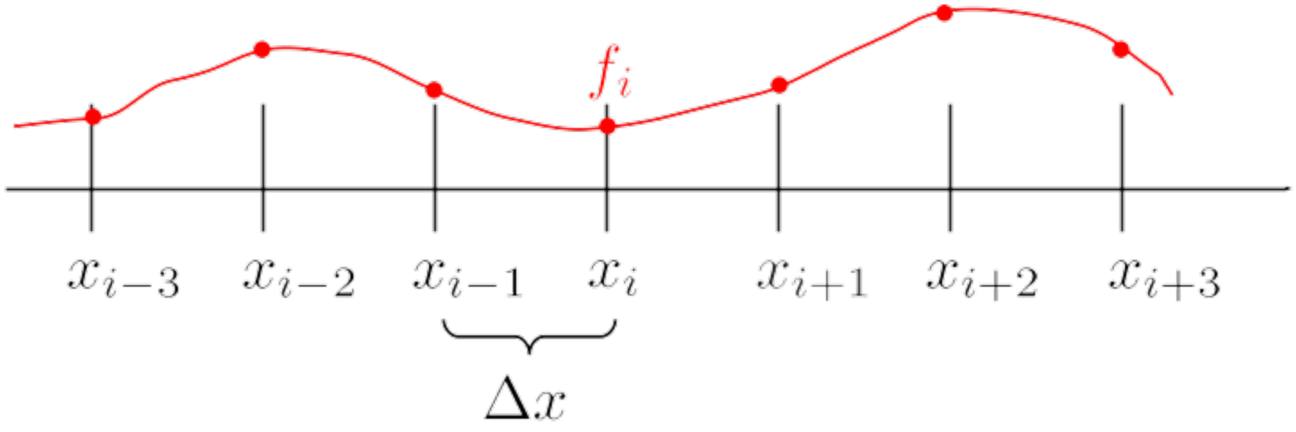
Gridded data

- 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:

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

- Solve for the first derivative:

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

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*+Δ*x*)]
  - For double precision: $C \simeq 10^{-16}$
- Worst-case rounding error on derivative is 2*C*|*f*(*x*)|/ Δ*x*
  - Also need to worry about associative errors: $(x + \Delta x) - x \stackrel{?}{=} \Delta x$
- So total error is: $\left| \frac{df}{dx}\bigg|_{x_i} - \frac{f_{i+1} - f_i}{\Delta x} \right| \leq \frac{1}{2} \frac{d^2 f}{dx^2}\bigg|_{x_i} \Delta x + \frac{2C|f_i|}{\Delta x}$
- 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":

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

2-point stencil

- Second-order "central":

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

3-point stencil

Second-order central

- Consider two Taylor expansions:

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

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

- We see that:

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

Error in Second order central

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

- Minimize WRT Δ*x*: $\Delta x = \sqrt[3]{6C \left| \frac{f(x_i)}{f'''(x_i)} \right|} \sim 10^{-5}$

Assuming double prec.

- Minimum error: $\epsilon \propto \sqrt[3]{C^2 f(x_i)^2 |f'''(x_i)|} \sim 10^{-11}$

Assuming double prec.

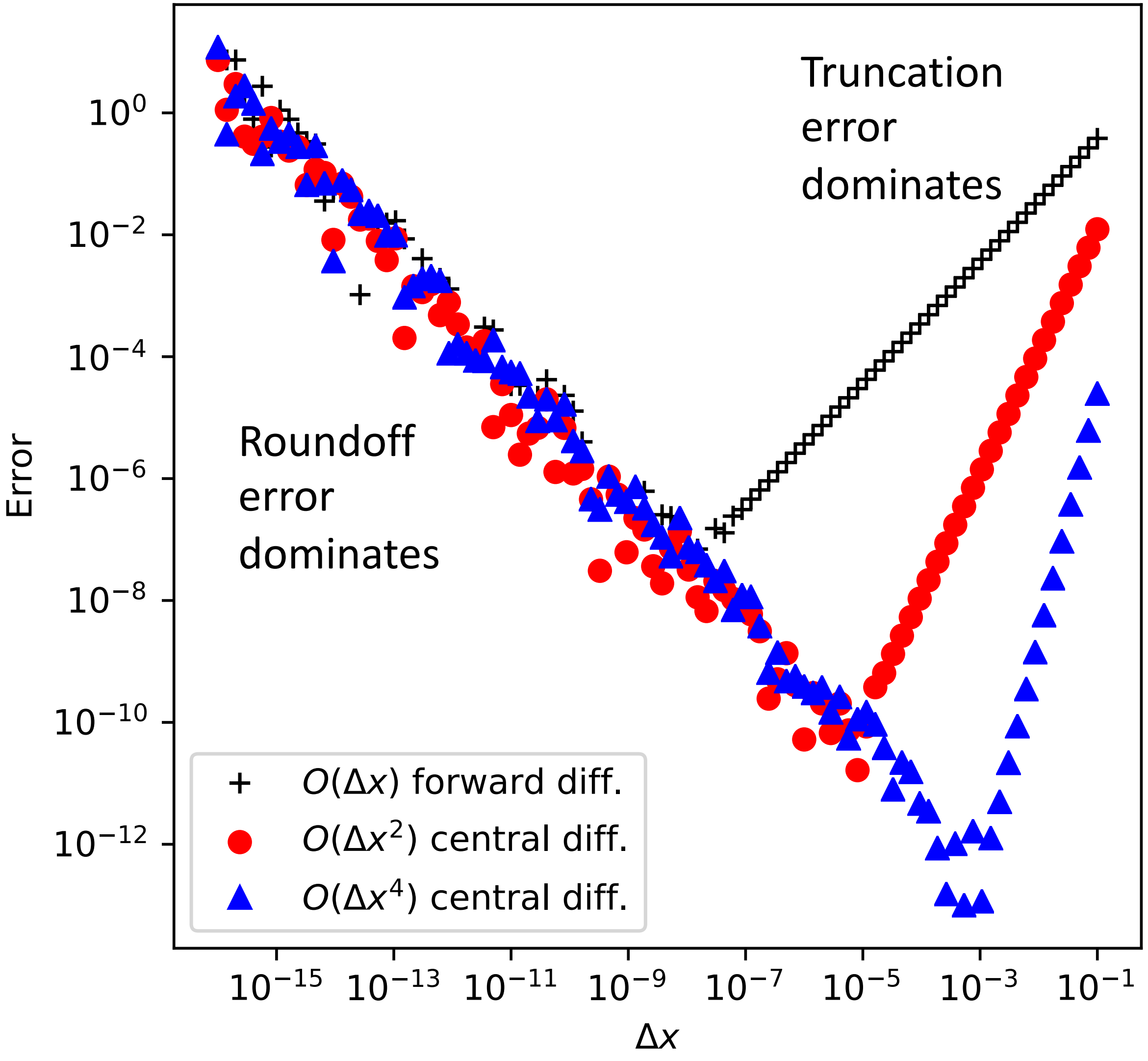
Higher order first derivatives

- 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	

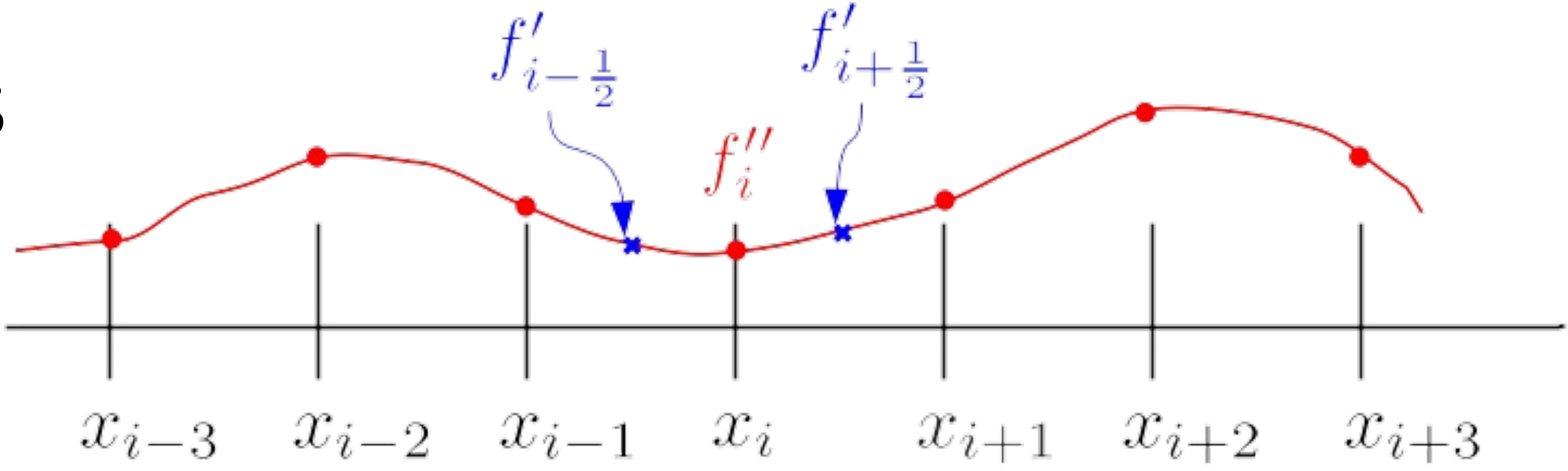
https://en.wikipedia.org/wiki/Finite_difference_coefficient

Example: Derivative of exp(x)



D

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:

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

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

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

- 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*)

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

- Mixed second derivative:

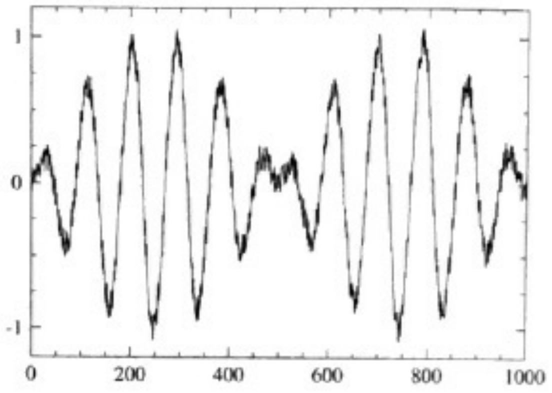
$$\frac{\partial^2 f}{\partial y \partial x} = \frac{f(x+\Delta x, y+\Delta y) - f(x-\Delta x, y+\Delta y) - f(x+\Delta x, y-\Delta y) + f(x-\Delta x, y-\Delta y)}{4\Delta x \Delta y}$$

Some final comments on numerical derivation

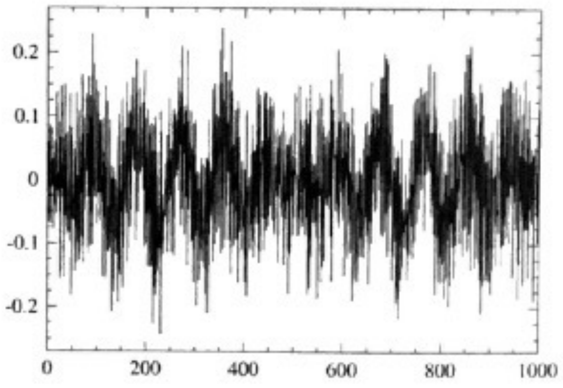
- Taking derivatives of noisy data makes the noise much worse!
  - Fit to a smooth curve and take the derivative of that
  - Smooth the data, e.g., with a Fourier transform

- We can treat data on uneven grids with the same strategy as before, taking into account the different Δx's between points

Noisy data



Derivative



(Newman)

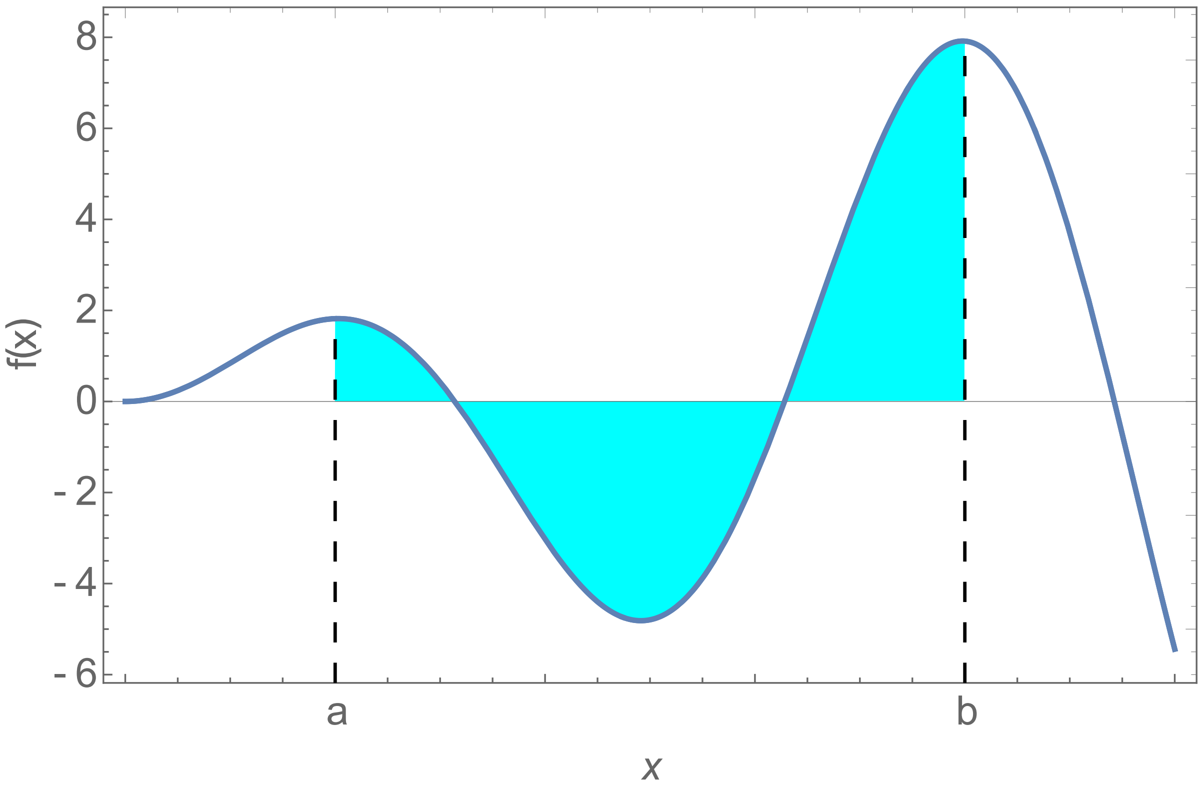
Today’s lecture:

- Numerical differentiation

- Numerical integration

Numerical integration

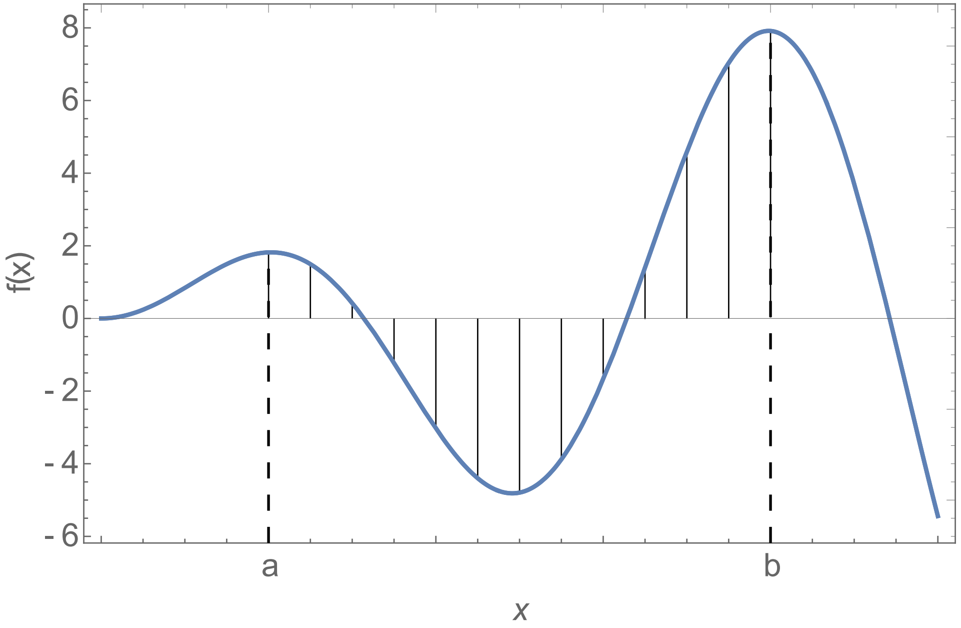
$$\int_a^b f(x)dx$$



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

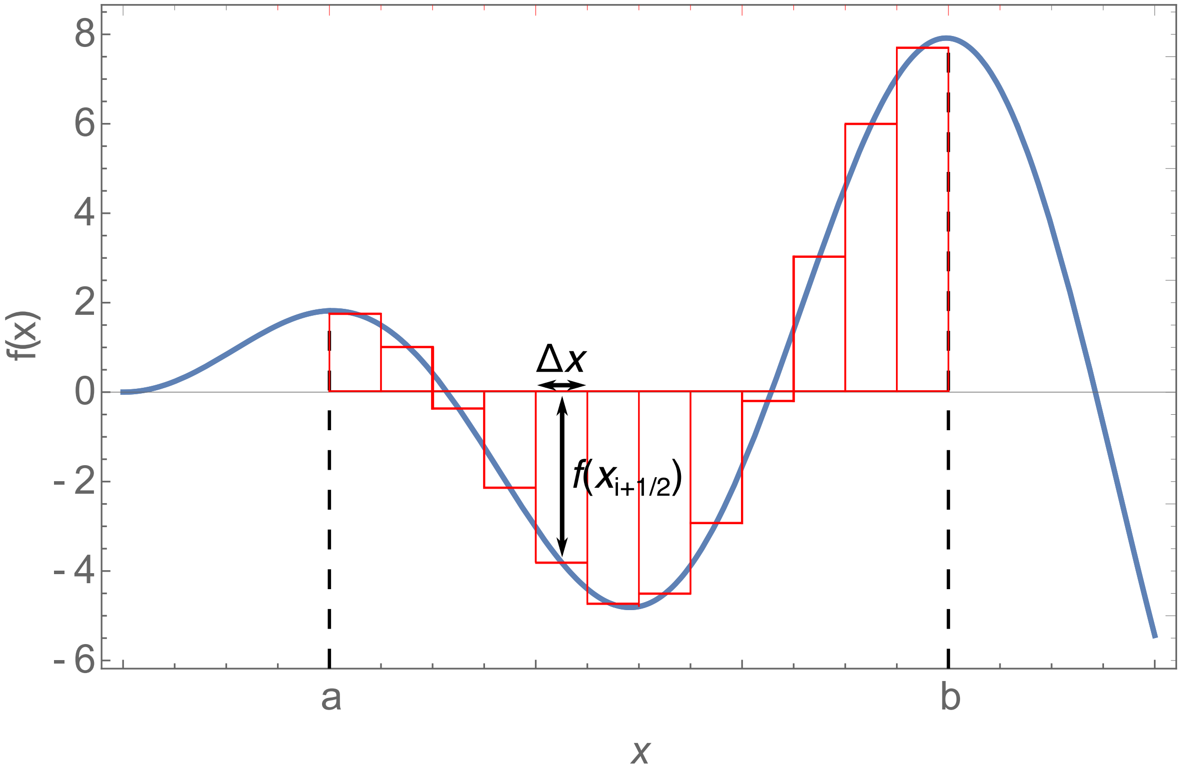
$$\int_a^b f(x)dx = \lim_{N\to\infty} \sum_{i=0}^{N-1} A_i$$



Approach 1: Midpoint rule

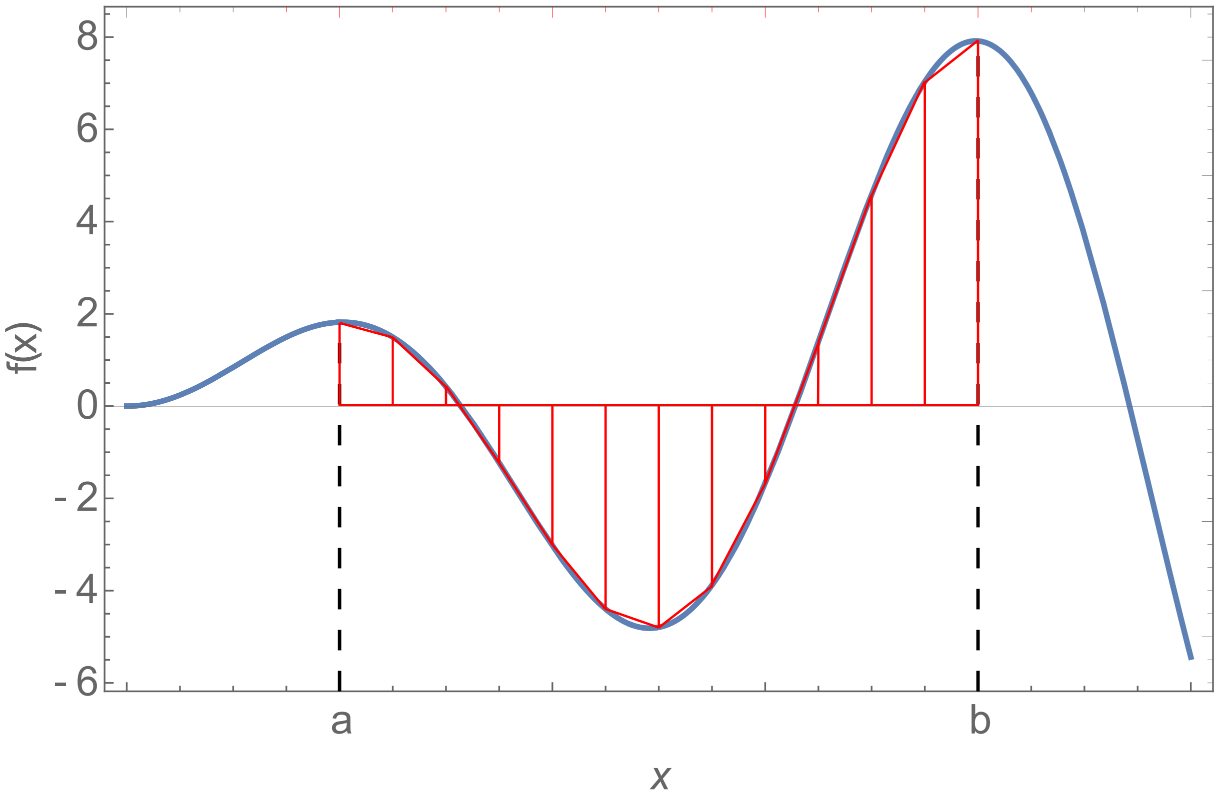
- Approximate area as rectangle with height equal to the midpoint of the subinterval $f(x_{i+1/2})$ and width Δx:

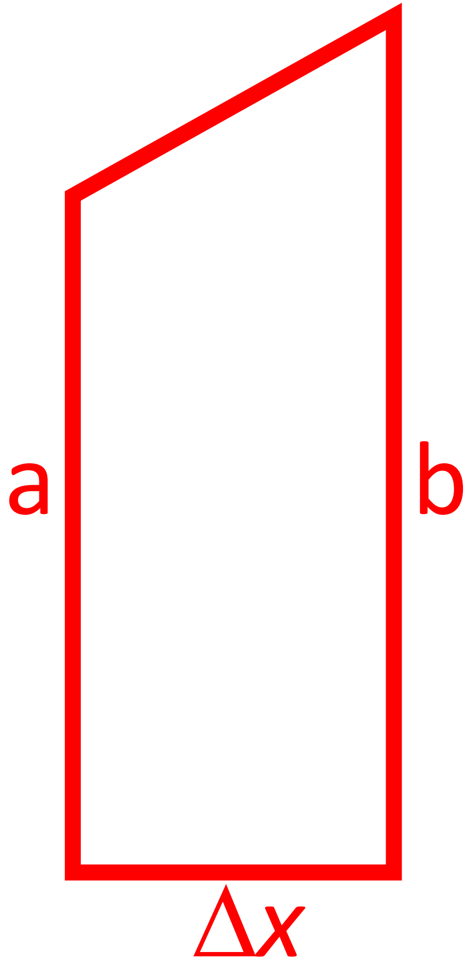
$$\int_a^b f(x)dx \simeq \lim_{N\to\infty} \sum_{i=0}^{N-1} \Delta x f(x_{i+\frac{1}{2}})$$



Approach 2: Trapezoid rule

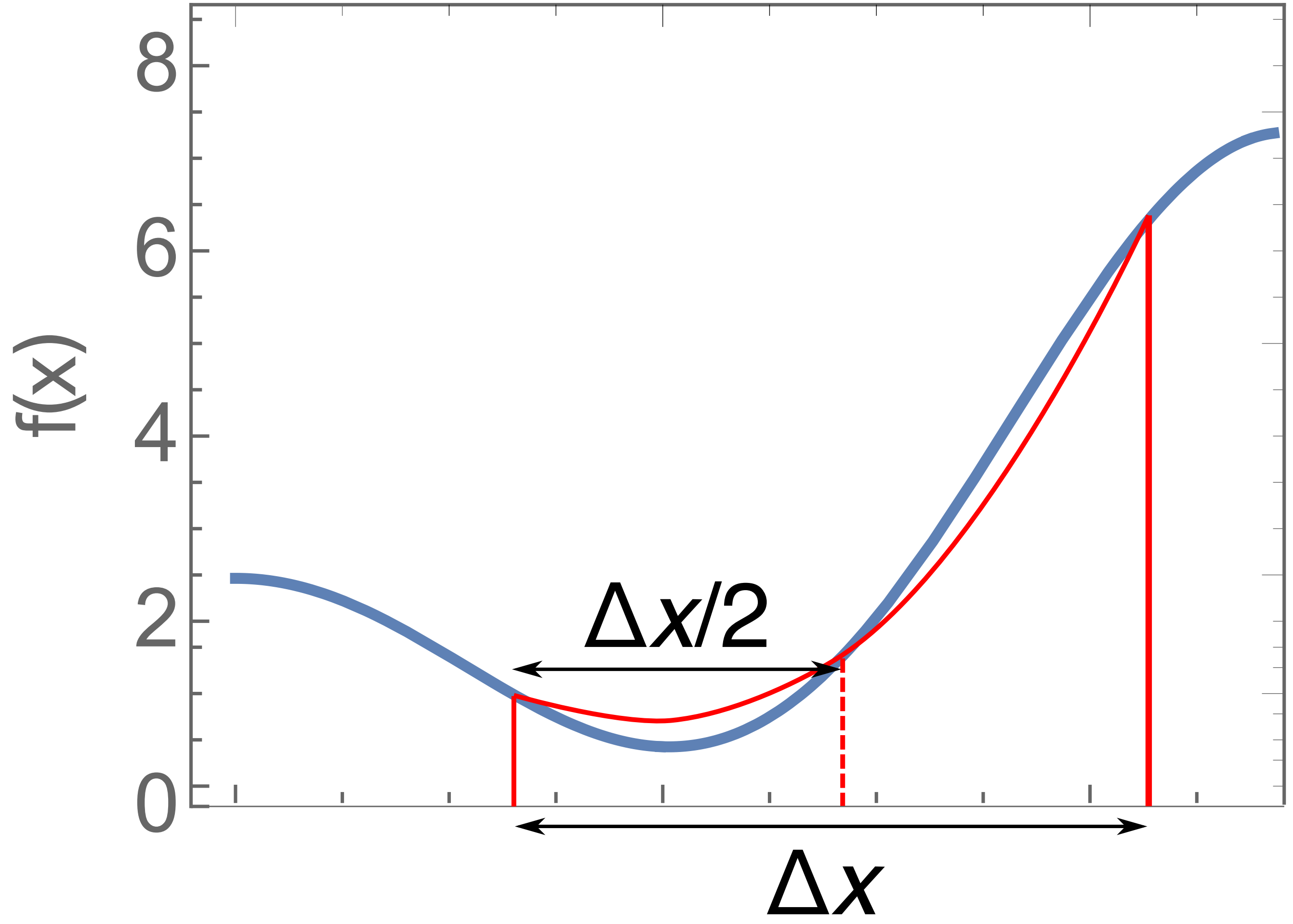
- Area of subintervals approximated as a trapezoid with subinterval endpoints on the curve
- Area of trapezoid: $\Delta x(a+b)/2$





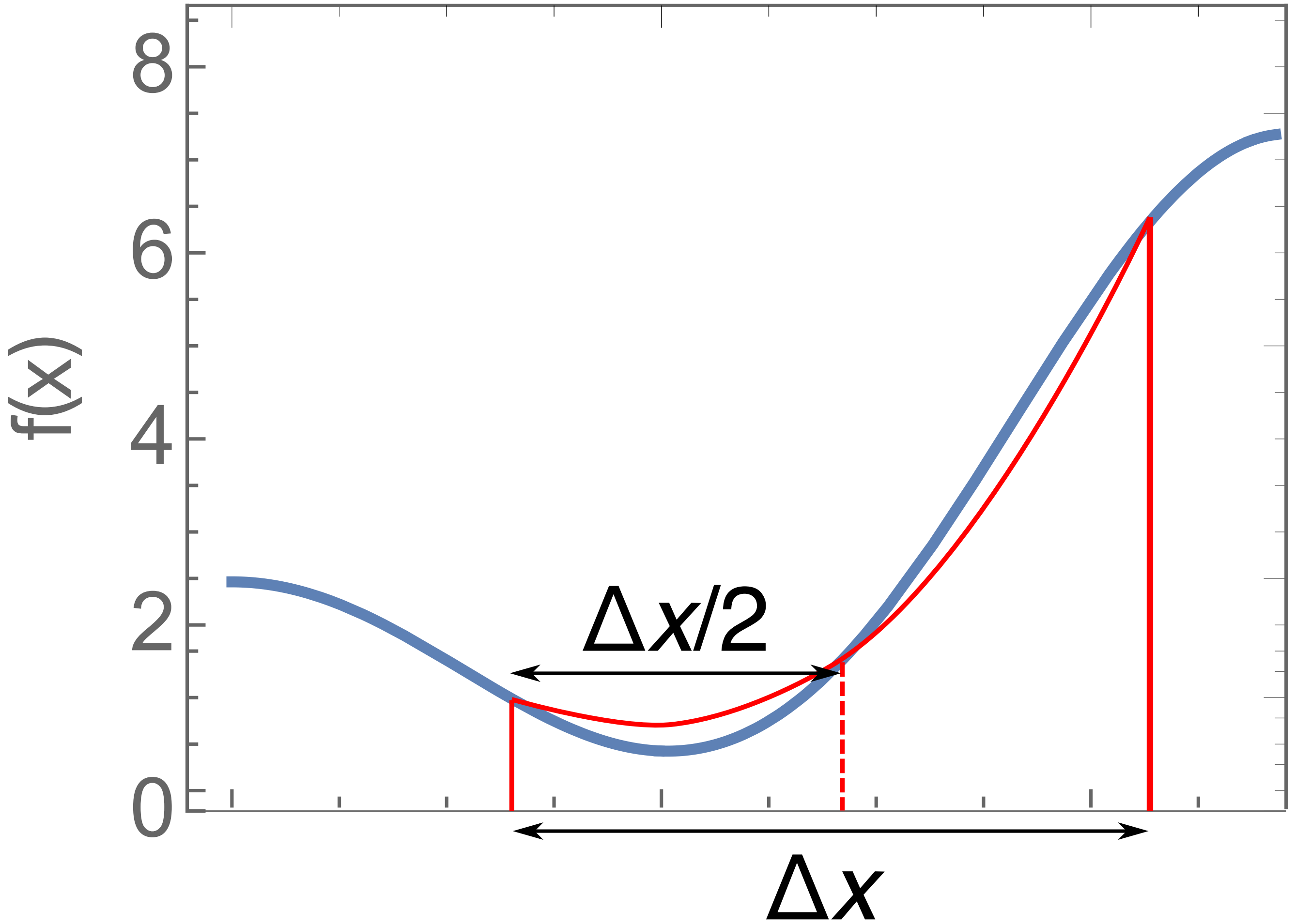
A more accurate technique: Simpson's Rule

- 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

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



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?

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

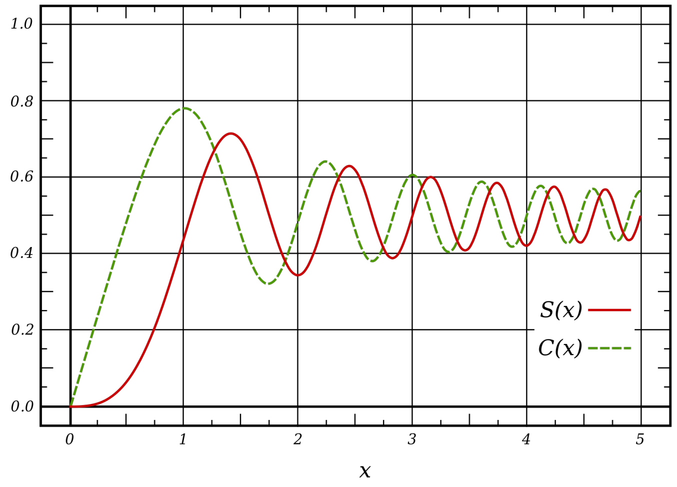
- Now we integrate over the subinterval:

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

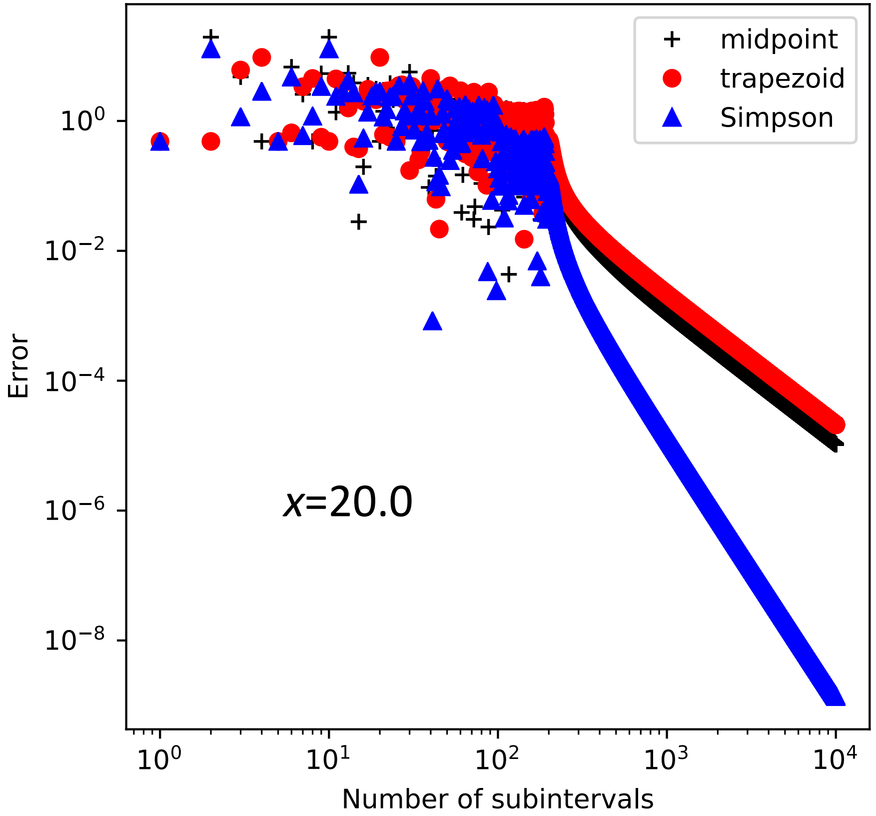
Example: Evaluating the Fresnel integral

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

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



(Wikipedia)



D

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)

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
- **NO CLASS TUESDAY SEPT. 12!!**
- Readings:
  - Blog on numerical differentiation
  - Wikipedia page of finite difference coefficients
  - Newman Chapter 5
  - Garcia Section 10.2