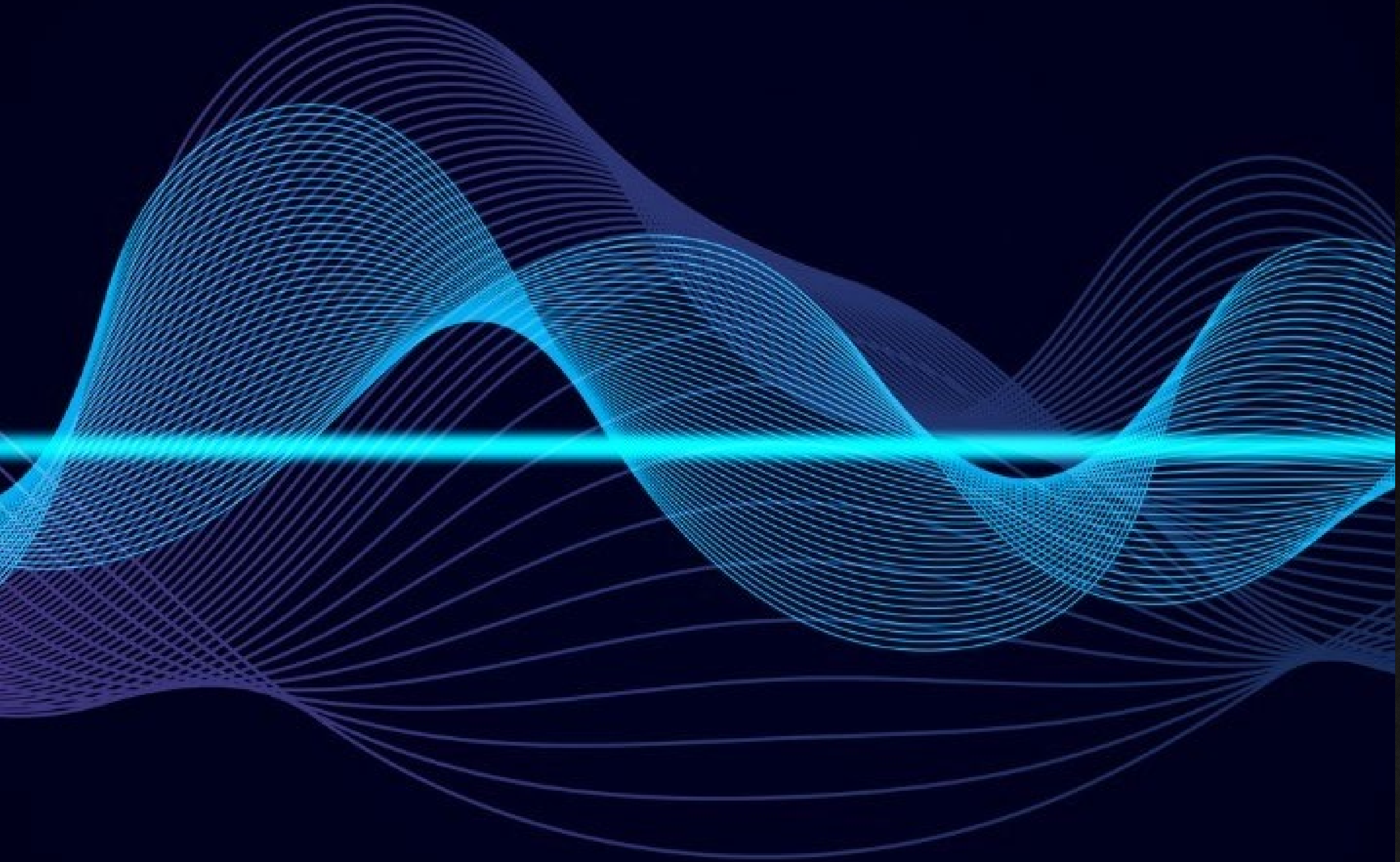


Computational Physics

PHYS 6260

Numerical Integration & Differentiation

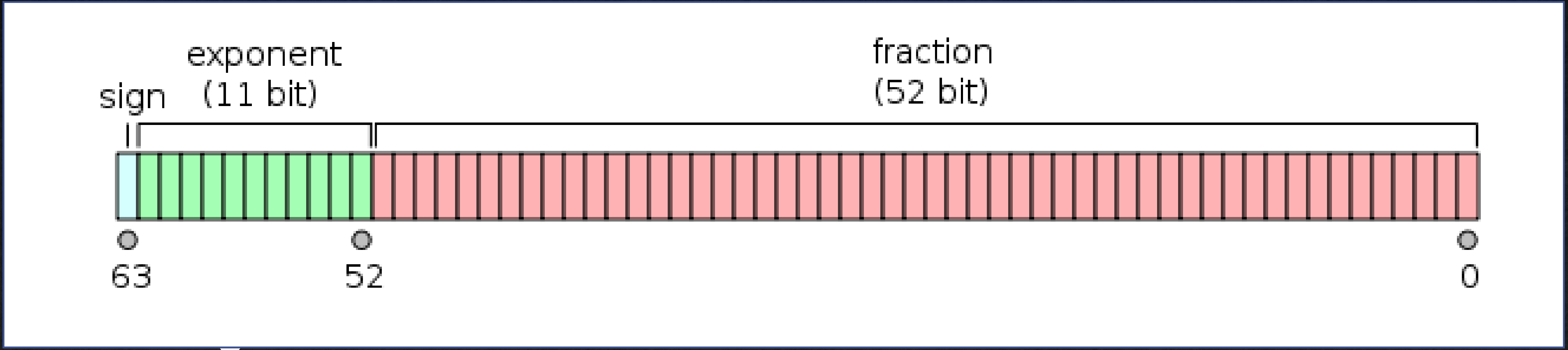
Announcements:

- HW1: Due Friday 1/10
- No class next week
- One recorded lecture next week since we're already ahead of schedule

Accuracy & Speed

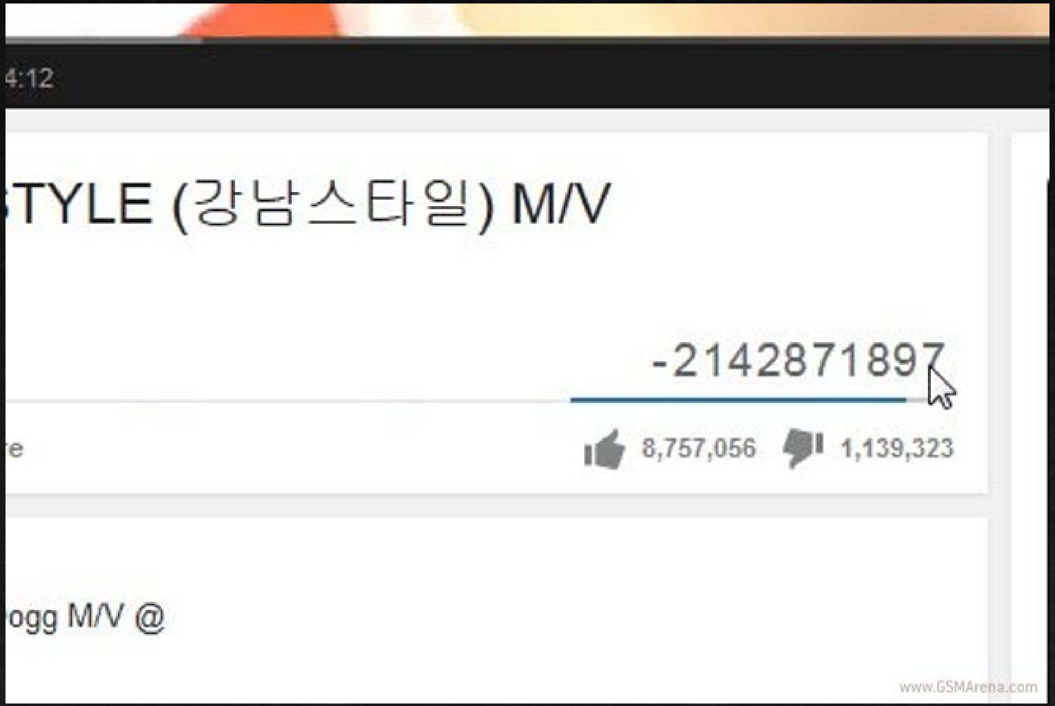
Variables and ranges

- Computers have limitations in accuracy and speed
- They can only store so much precision and perform so many operations per second
- Variables have a set number of bits to represent them, placing limations on them
- For example, floats (64-bit by default in Python) have a min/max of 10^{-308} and 10^{308}
- Why is this? Let's look how it's stored

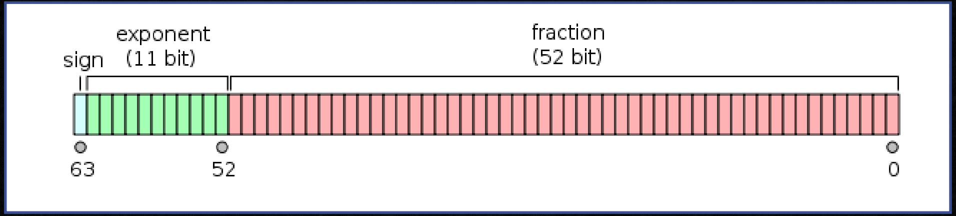


Variables and ranges

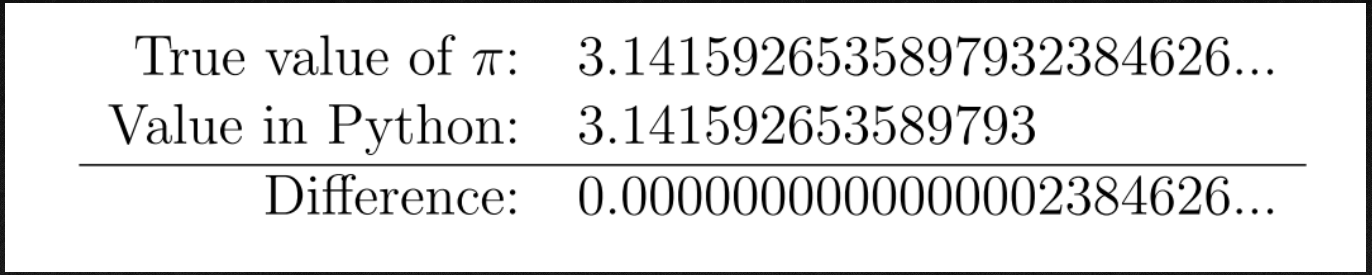
- For signed integers, one bit stores the sign, and the rest stores the number
- For example, the max signed 32-bit integer is $2^{31} - 1$
- If the variable exceeds this max, it's known as an overflow
- Python does some clever memory management to represent integers with *arbitrary precision*



Numerical Error



- There are only 51 bits for the fraction that translates into 16 decimal places ($2^{51} \sim 10^{16}$)
- Irrational numbers can only be represented approximately
- The difference between the stored and actual number is called ***rounding error***



True value of π:	3.1415926535897932384626...
Value in Python:	3.141592653589793
Difference:	0.0000000000000002384626...

- ***Machine precision***: refers to the maximum amount of precision (1 part in 10^{16} for 64-bit numbers)

Program speed

Operation speed

- The speed of computation, measured in cycles and depends on the clock speed (GHz) of the processor, depends on the type of operation

Cycles	Operations
3-5	Addition, subtraction, absolute values, multiplication
~10	Division
10-50	Square roots, exponentials, hyperbolic functions
50-1000	Trigonometric, logarithmic, and power functions

- Multi-part operations, such as matrix operations and special functions like Bessel function or Gamma functions take much longer

Numerical integration

Section Outline

We will cover these topics

- Basic integration methods
  - Trapezoidal rule
  - Simpson’s rule
- Errors in numerical integration
- Adaptive Integration
- Gaussian Quadrature

Basic integration methods

- Consider some function f(x) that we want to integrate from x = a → b

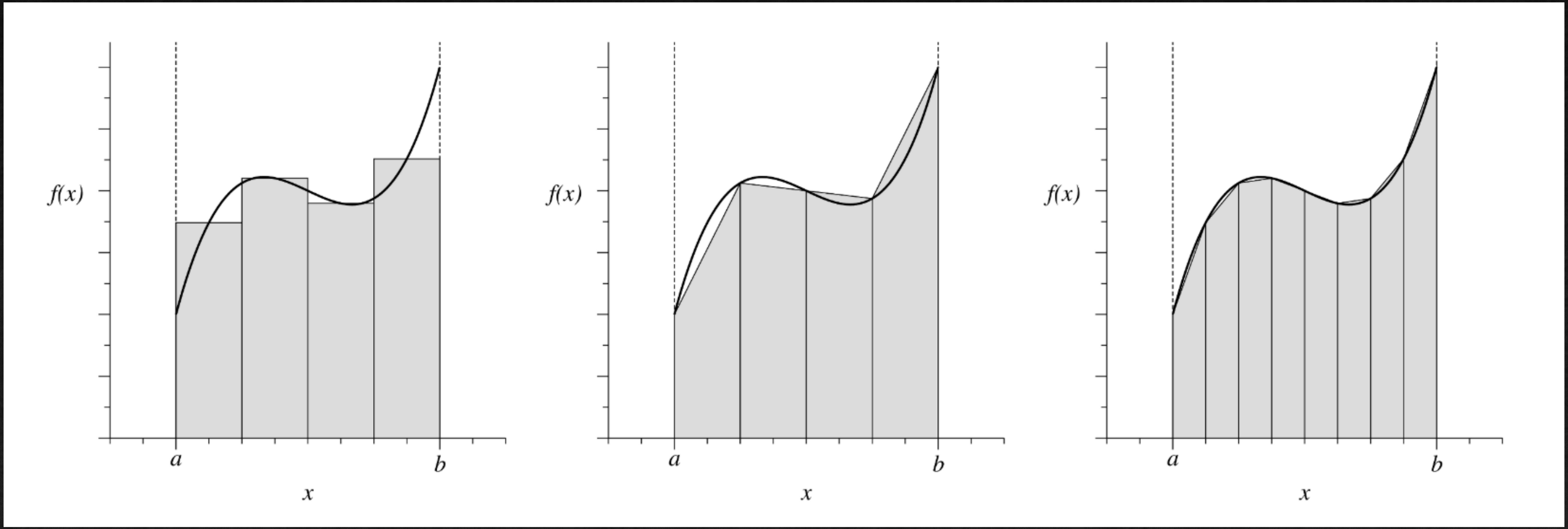
$$I(a,b) = \int_a^b f(x)dx$$

- It is impossible to compute an integrate **exactly** but there are several approximate methods
- Let’s start with the simplest method, the **Trapezoidal rule**

Trapezoidal rule

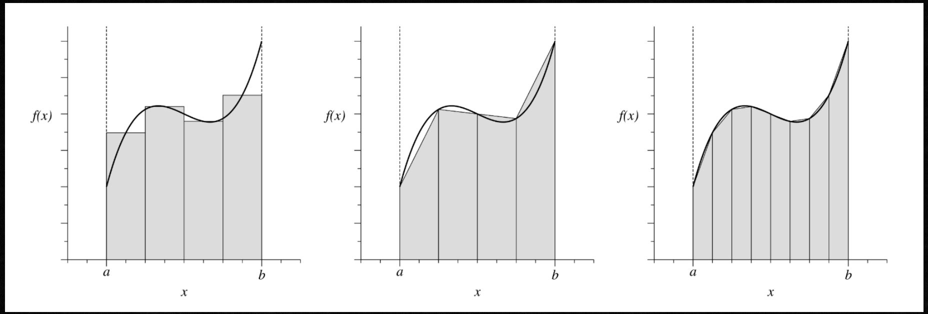
$$I(a,b) = \int_a^b f(x)dx$$

- Let's first do a visual inspection of accuracy
  - Left: rectangular slices to approximate integral
  - Middle: trapezoidal slices to approximate integral
  - Right: smaller trapezoids



Trapezoidal rule

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



- Now let's express this method mathematically
  - Consider N slices
- Slice width: h = (b – a)/N
- Left and right edges of the kth slice:

  x = a + (k-1)h x = a + kh

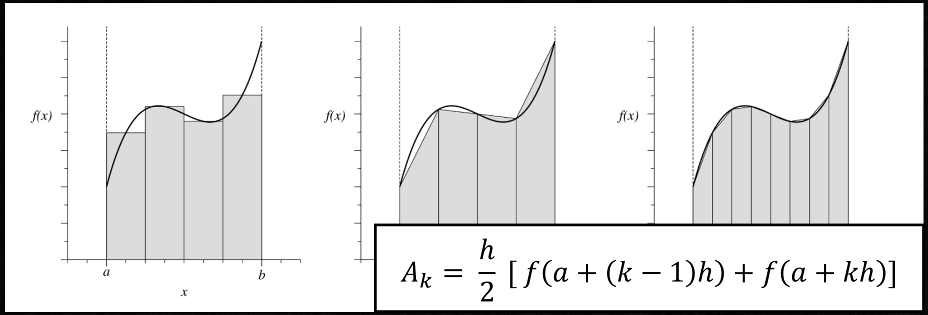
- This results in an area for the kth slice of

$$A_k = \frac{h}{2}\,[\,f(a+(k-1)h) + f(a+kh)]$$

- This is the basis of the trapezoidal rule

Trapezoidal rule

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



$$A_k = \frac{h}{2}\left[f(a+(k-1)h) + f(a+kh)\right]$$

- Extending this to the entire integral, summing the slices

$$\begin{aligned} I(a,b) \simeq \sum_{k=1}^{N} A_k &= \frac{h}{2}\sum_{k=1}^{N}[f(a+(k-1)h)+f(a+kh)] \\ &= h\left[\frac{1}{2}f(a)+f(a+h)+f(a+2h)+\ldots+\frac{1}{2}f(b)\right] \end{aligned}$$

$$\boxed{I(a,b) \simeq h\left[\frac{1}{2}(f(a)+f(b))+\sum_{k=1}^{N-1}f(a+kh)\right]}$$

- *Notice that the value is just the summation of all interior slices and half of the boundary slices*

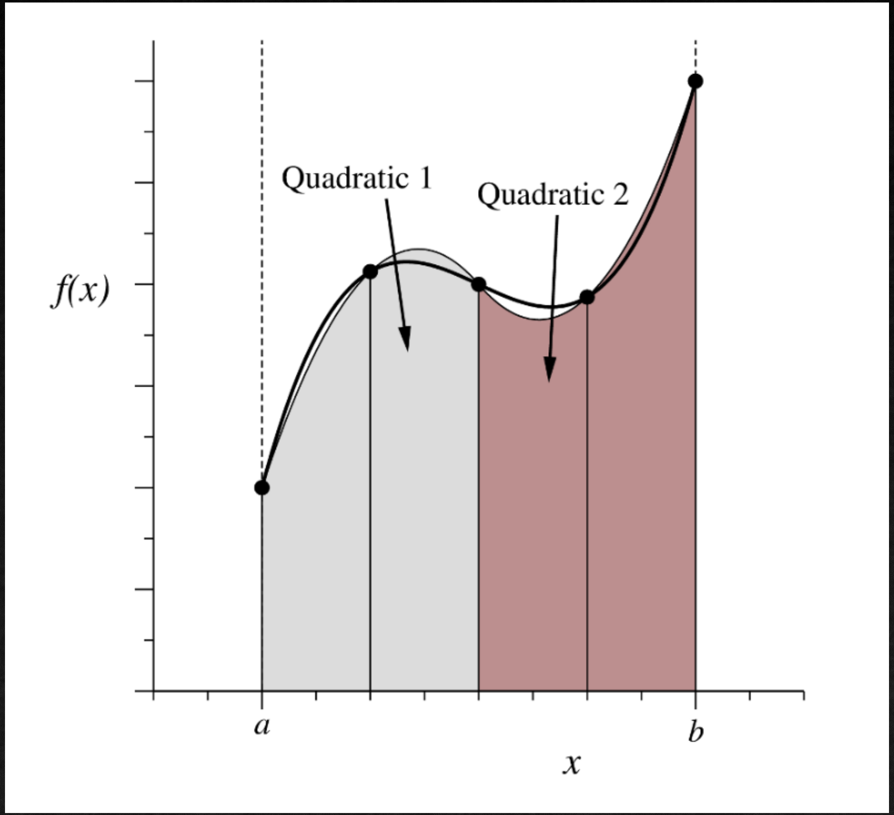
In-class problem: Trapezoidal rule

- Consider the function: $f(x) = x^4 - 2x + 1$
- Integrate f(x) from x = [0,2] with the trapezoidal rule
- The exact answer is 4.4
- Use a function to define f(x) and compare your results for various numbers of slices
  - N = 10, 100, 1000
- How does the error decrease with increasing resolution?

$$\boxed{I(a,b) \simeq h\left[\frac{1}{2}(f(a)+f(b)) + \sum_{k=1}^{N-1} f(a+kh)\right]}$$

Simpson’s rule

- In cases were accuracy isn’t needed, using the trapezoidal rule is perfectly fine
- As we will see in this class, we can increase the accuracy by increasing the order of the polynomial approximator
- Simpson’s rule uses a parabolic fit instead of linear (trapezoidal)
- This is especially useful for rapidly varying functions
- Much more accurate, especially at a small number of slices

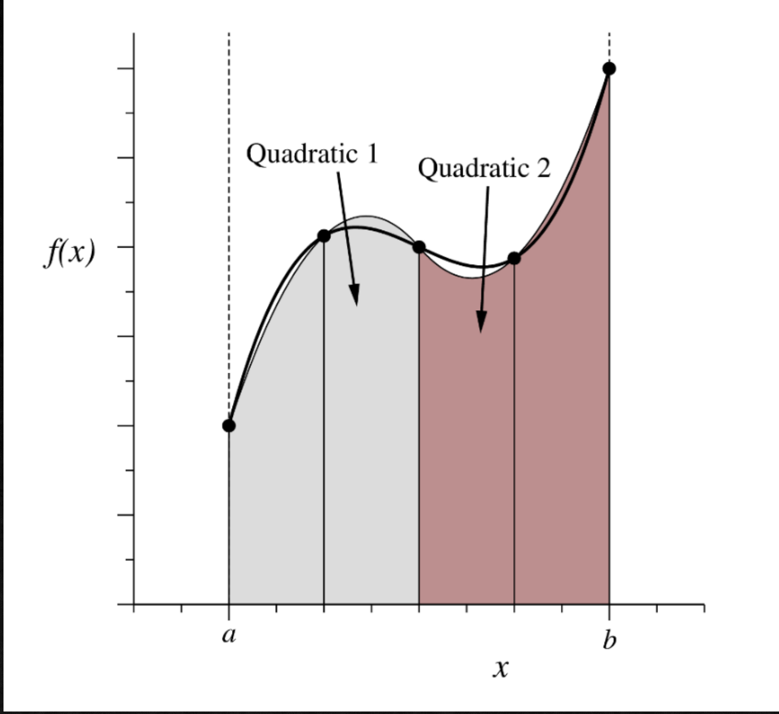


Simpson’s rule

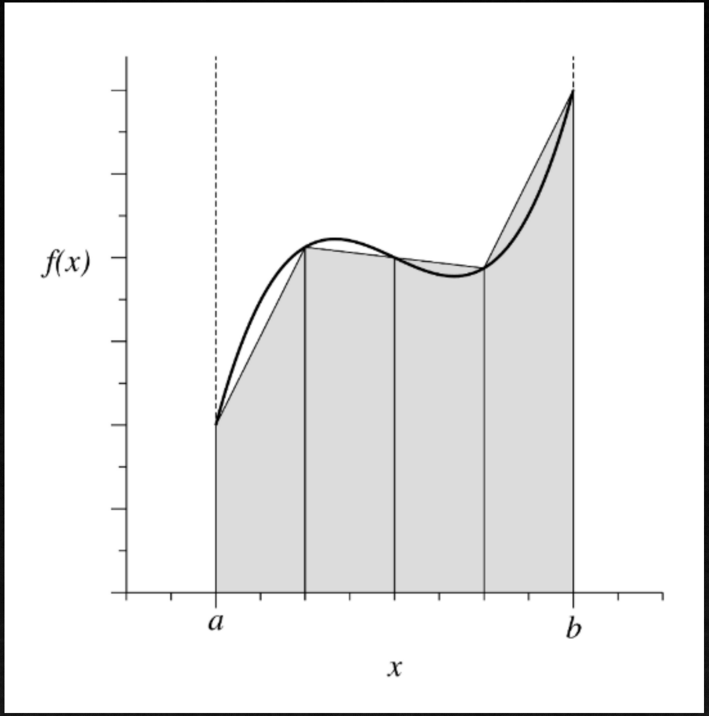
- One restriction is that Simpson’s rule requires an even number of slices
- See notes for derivation.
- Final expression:

$$I(a,b) \simeq \frac{h}{3}[f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) + \ldots + f(b)]$$

$$= \boxed{\frac{h}{3}\left[f(a) + f(b) + 4\sum_{\substack{\text{odd } k\\ 1\ldots N-1}} f(a+kh) + 2\sum_{\substack{\text{even } k\\ 2\ldots N-2}} f(a+kh)\right]}$$



Numerical integration errors



- ***Approximation errors:*** come from numerical integration
  - Not rounding error
- We can estimate the approximation error by comparing the slices and a Taylor expansion
- The Taylor expansion around the kth point at $x_k = a+kh$ is

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

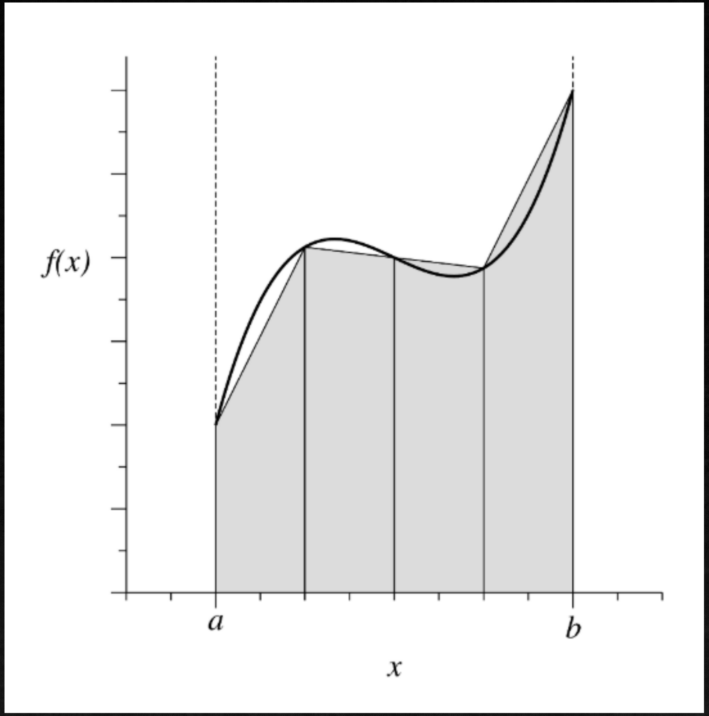
  where f' and f'' are the 1st and 2nd spatial derivatives
- We can integrate this Taylor expansion from x_{k-1} to x_k

Numerical integration errors

- After some calculus and algebra (see lecture notes), we arrive at

$$\begin{aligned}
\int_a^b f(x)dx &= \sum_{k=1}^{N}\int_{x_{k-1}}^{x_k} f(x)dx \\
&= \frac{1}{2}h\sum_{k-1}^{N}[f(x_{k-1})+f(x_k)] + \frac{1}{4}h^2[f'(a)-f'(b)] + \\
&\quad \frac{1}{12}h^3\sum_{k=1}^{N}[f''(x_{k-1})+f''(x_k)] + O(h^4)
\end{aligned}$$

where
  - the 1st term is the trapezoidal rule
  - the 2nd term doesn’t depend on the interior values
  - the 3rd term is the approximation error



Numerical integration errors

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{k=1}^{N}\int_{x_{k-1}}^{x_k} f(x)dx \\ &= \frac{1}{2}h\sum_{k-1}^{N}[f(x_{k-1})+f(x_k)] + \frac{1}{4}h^2[f'(a)-f'(b)] + \\ &\quad \frac{1}{12}h^3\sum_{k=1}^{N}[f''(x_{k-1})+f''(x_k)] + O(h^4)\end{aligned}$$

- We recognize that the 3rd term is the trapezoidal rule for f’’(x)

$$\int_a^b f''(x)dx = \frac{1}{2}h\sum_{k-1}^{N}[f''(x_{k-1})+f''(x_k)] + O(h^2)$$

- So that we can express this error in terms of f’(x)

$$\begin{aligned}\frac{1}{12}h^3\sum_{k=1}^{N}[f''(x_{k-1})+f''(x_k)] &= \frac{1}{6}h^2\int_a^b f''(x)dx + O(h^4) \\ &= \frac{1}{6}h^2[f'(b)-f'(a)] + O(h^4),\end{aligned}$$

Numerical integration errors

- Substituting this result into the integral, we combine the original 2nd term and the error term

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{k=1}^{N}\int_{x_{k-1}}^{x_k} f(x)dx \\ &= \frac{1}{2}h\sum_{k-1}^{N}[f(x_{k-1})+f(x_k)] + \frac{1}{4}h^2[f'(a)-f'(b)] + \\ &\quad \frac{1}{12}h^3\sum_{k=1}^{N}[f''(x_{k-1})+f''(x_k)] + O(h^4)\end{aligned}$$

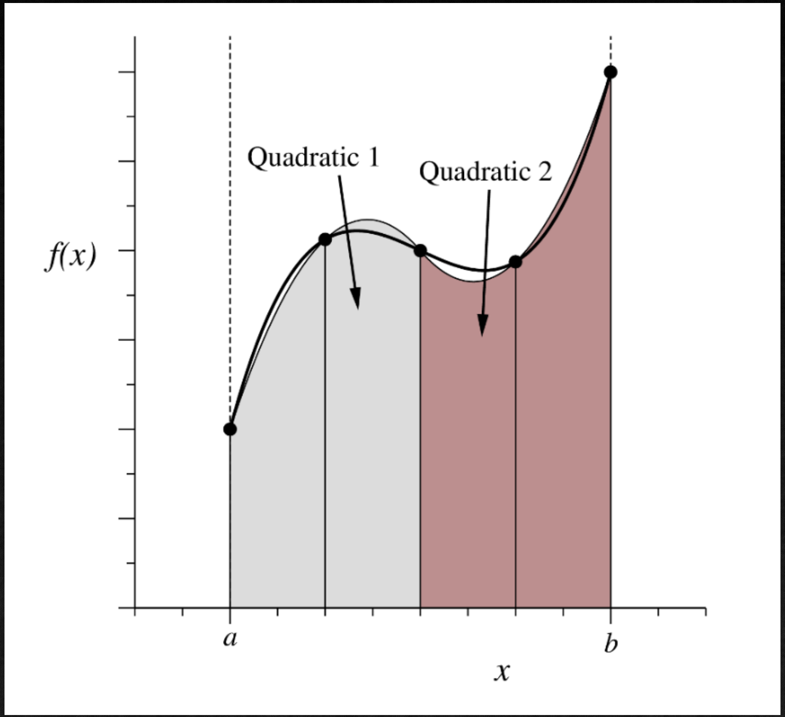
$$\begin{aligned}\frac{1}{12}h^3\sum_{k=1}^{N}[f''(x_{k-1})+f''(x_k)] &= \frac{1}{6}h^2\int_a^b f''(x)dx + O(h^4) \\ &= \frac{1}{6}h^2[f'(b)-f'(a)] + O(h^4),\end{aligned}$$

$$\boxed{\int_a^b f(x)dx = \frac{1}{2}h\sum_{k=1}^{N}[f(x_{k-1}+f(x_k)] + \frac{1}{12}h^2[f'(a)-f'(b)] + O(h^4).}$$

- The approximation error ϵ is the first term for the ***Euler-Maclaurin formula*** that quantifies the error.
- The trapezoidal rule is ***first-order accurate*** O(h) and has second-order errors O(h^2)

Numerical integration errors

- The trapezoidal rule is ***first-order accurate*** O(h) and has second-order errors O(h^2)
- This means that the numerical error will decrease as the square of the slice size
- If one performs the same analysis on Simpson's rule
  - ***Third-order accurate*** O(h^3) with fourth-order errors O(h^4)
  - If the trapezoidal rule needed N slices for some level of accuracy, Simpson's rule will only need $\sqrt{N}$
- Errors for both methods depend on the derivative at the integration bounds (unfortunately)



Numerical integration errors

Practical estimation of errors

- The major disadvantage to this approach
  - f(x) needs to be known in a closed-form solution
- In nearly all scientific calculations, the closed-form solution is not known
- However, we can use the fact that the trapezoidal rule errors are O(h^2)
  - If we half the slice size, we cut the error by one-quarter
- The integral has a true value “I” that can be expressed as

$$\boxed{I = I_1 + ch_1^2}$$

where I_1 is the numerical integral with N_1 steps and a stepsize of h_1

Numerical integration errors

$$\boxed{I = I_1 + ch_1^2}$$

- We now consider the case with double the steps and half the stepsize $h_2 = h_1/2$

$$\boxed{I = I_2 + ch_2^2}$$

- We equate "I" and find that

$$\boxed{I_2 - I_1 = ch_1^2 - ch_2^2 = 3ch_2^2}$$

- Rearrange the expression in terms of the error $\epsilon_2 = ch_2^2$ to arrive at

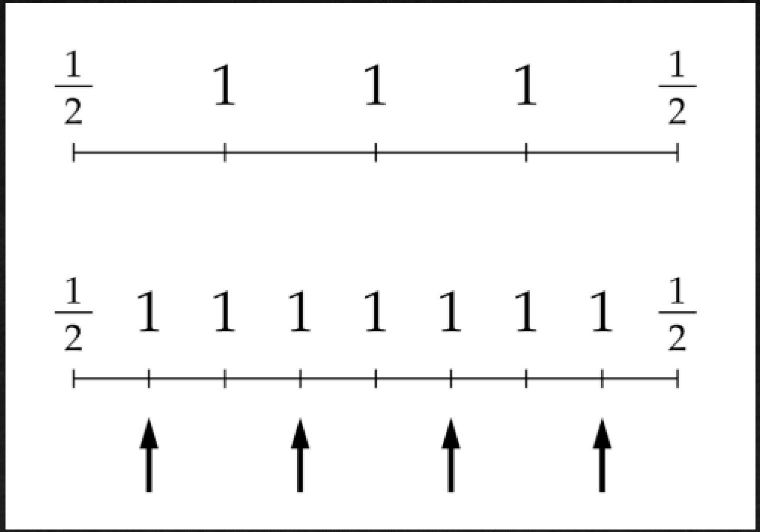
$$\boxed{\epsilon_2 = \frac{1}{3}(I_2 - I_1)}$$

- For Simpson's rule, the coefficient is 1/15

Adaptive integration

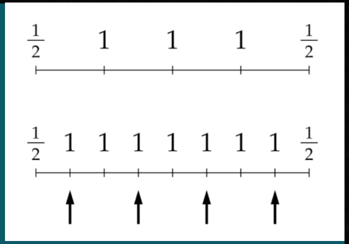
Choosing the number of steps

- Using the approximation error decreases as $(I_2 - I_1)/3$ when the steps are doubled
- We can double the steps until a specified limit is reached
- Advantage of doubling the resolution is that we've already calculated half of the values in the new sum



Adaptive integration

Choosing the number of steps



$\frac{1}{2}$ 1 1 1 $\frac{1}{2}$
$\frac{1}{2}$ 1 1 1 1 1 1 1 $\frac{1}{2}$

$$I_i = \frac{1}{2} I_{i-1} + h_i \sum_{\substack{\text{odd } k \\ 1\ldots N_i - 1}} f(a + kh_i)$$

Procedure

1. Choose an initial number of steps N_1 and decide on a target accuracy
2. Calculate the first integral I_1 with the standard trapezoidal rule
3. Double the number of steps and calculate the improved integral estimate
4. If the absolute magnitude of the error is less than the target accuracy, halt the calculation. Otherwise, repeat from step #2

See notes / book for details about adaptive Simpson's rule.

Romberg Integration

- This approach takes the error estimate into account for the integration estimate
- For example with the trapezoidal rule, this makes the result **third-order accurate**

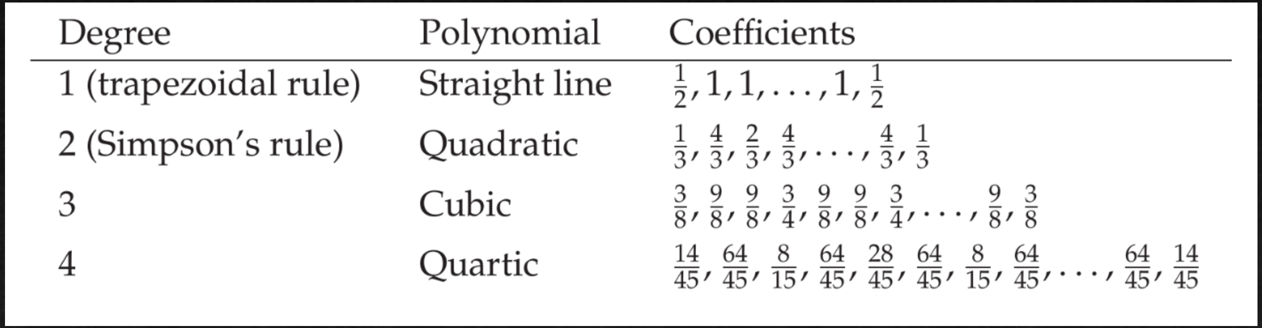
$$I = I_i + \frac{1}{3}(I_i - I_{i-1}) + O(h_i^4)$$

- We can generalize this idea to an arbitrary number of iterations, correcting for errors
- This is known as ***Romberg Integration. See notes for details.***

Higher-order integration

- In general, integration with polymonial fitting functions is given by $$\int_a^b f(x)dx \simeq \sum_{k=1}^{N} w_k f(x_k),$$

- The coefficients depend on the order and are known as the **Newton-Cotes** formulae



Degree	Polynomial	Coefficients
1 (trapezoidal rule)	Straight line	$\frac{1}{2}, 1, 1, \ldots, 1, \frac{1}{2}$
2 (Simpson's rule)	Quadratic	$\frac{1}{3}, \frac{4}{3}, \frac{2}{3}, \frac{4}{3}, \ldots, \frac{4}{3}, \frac{1}{3}$
3	Cubic	$\frac{3}{8}, \frac{9}{8}, \frac{9}{8}, \frac{3}{4}, \frac{9}{8}, \frac{9}{8}, \frac{3}{4}, \ldots, \frac{9}{8}, \frac{3}{8}$
4	Quartic	$\frac{14}{45}, \frac{64}{45}, \frac{8}{15}, \frac{64}{45}, \frac{28}{45}, \frac{64}{45}, \frac{8}{15}, \frac{64}{45}, \ldots, \frac{64}{45}, \frac{14}{45}$

- One improvement we can make on these methods is variable step sizes
- For example, we would want to sample the function (1) more frequently if the slope is steep, or (2) less often if the slope is shallow

Gaussian quadrature

Overview

- Gaussian quadrature is best used for smoothly varying functions
- The key difference between this method and the previous ones is
  - **We use N non-uniformly spaced points**

- General approach:
  - Approximate the function with a linear combination of polynomials
  - Use this approximation to calculate weights for the function evaluation
  - *Note: The evaluation points are non-uniformly distributed*
  - Calculate the weighted sum

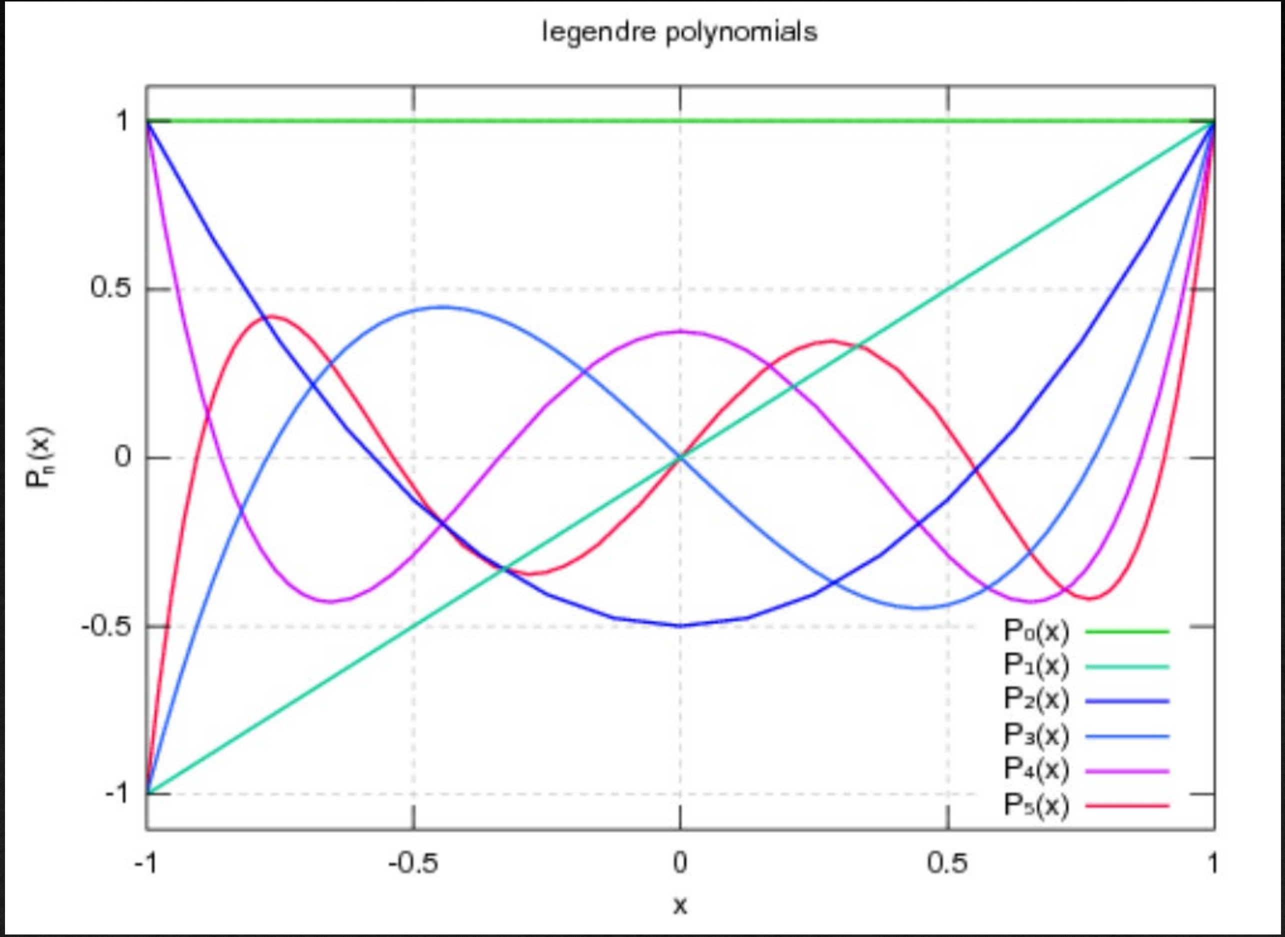
- Goal: Fit a $(N-1)^{th}$ degree polynomial through N points

Gaussian quadrature

Sample points

$$\int_a^b f(x)dx \simeq \sum_{k=1}^{N} w'_k f(x'_k)$$

- See notes for derivation
- Choose Legendre polynomials as the basis
- The evaluation points are their roots



Gaussian quadrature

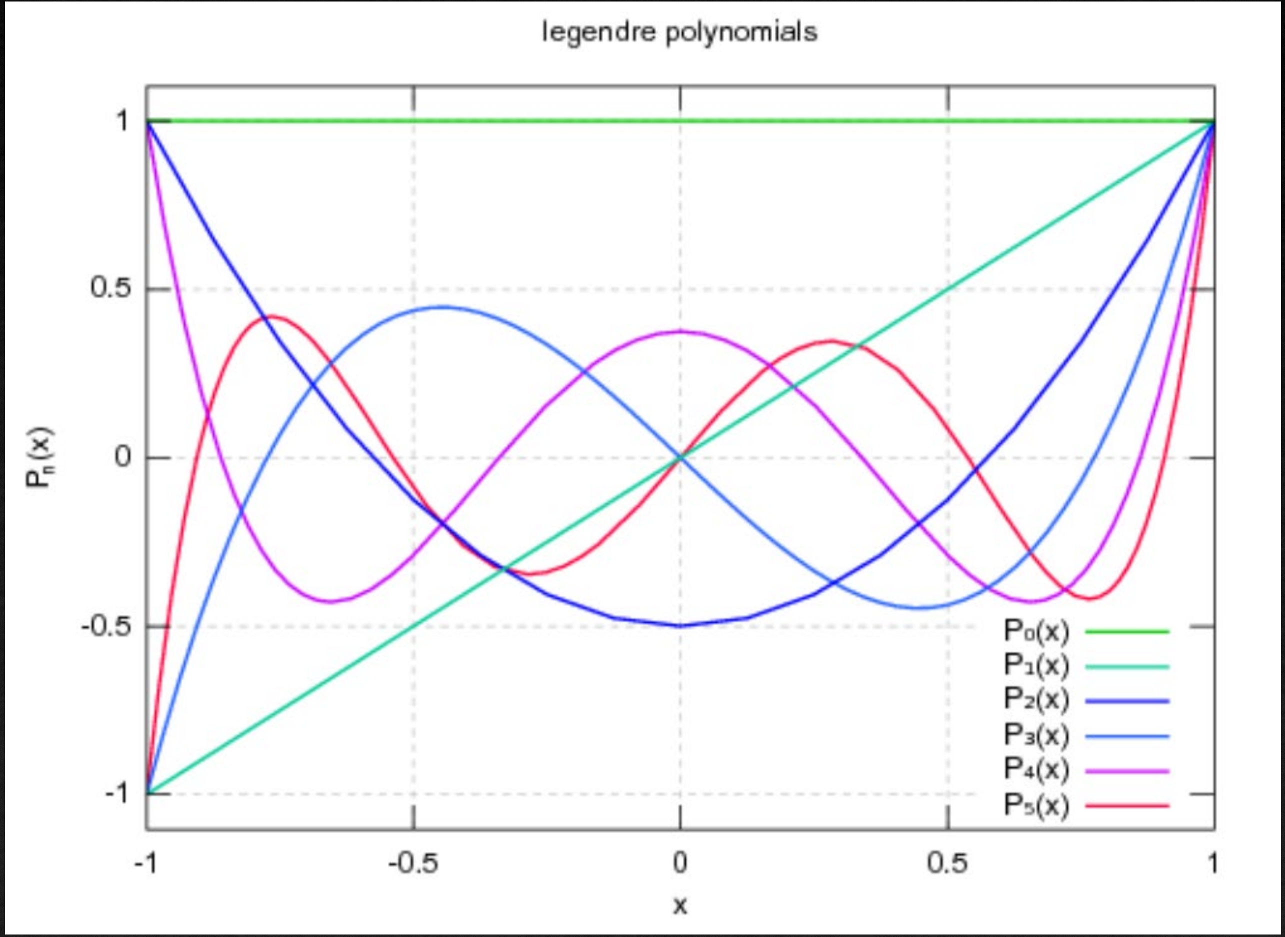
Sample points

$$\int_a^b f(x)dx \simeq \sum_{k=1}^{N} w'_k f(x'_k)$$

- The weights are determined by the derivative of the polynomial at x_k

$$w_k = \left[\frac{2}{1-x^2}\left(\frac{dP_N}{dx}\right)^{-1}\right]_{x=x_k}$$

- These weights are somewhat computationally intensive but *only need to be calculated once for a given N*
  - Very convenient before consumer computers



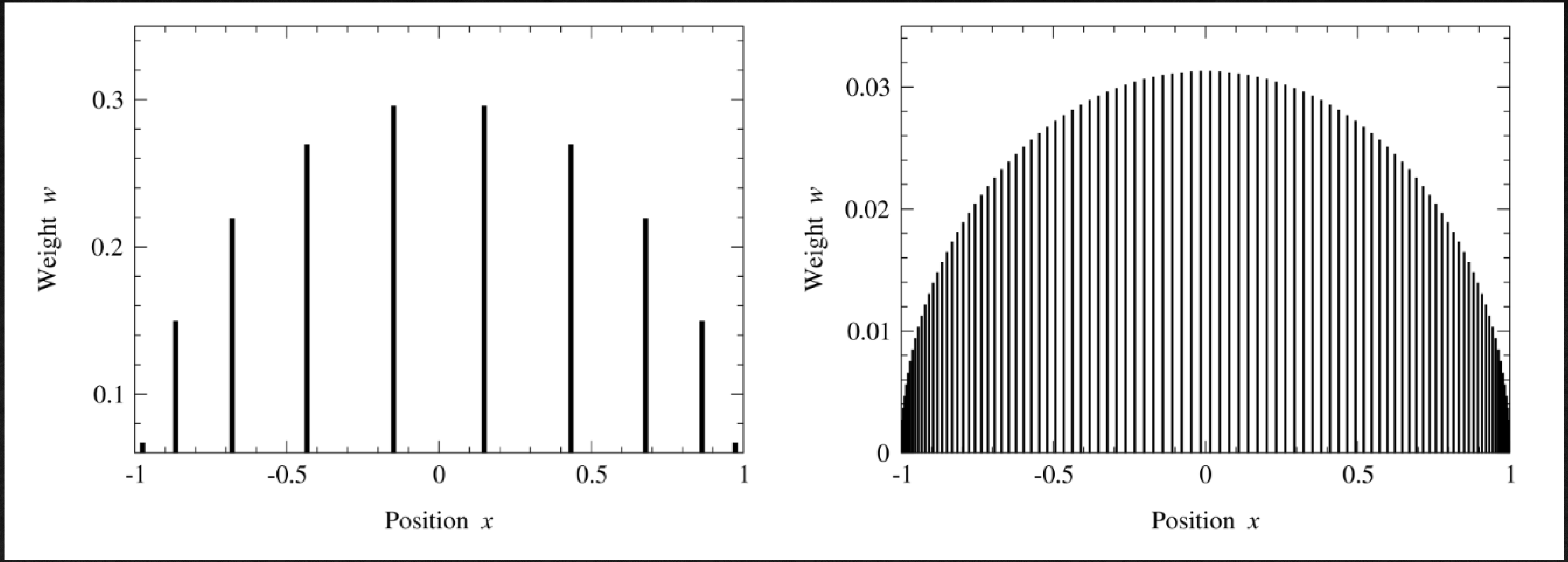
Gaussian quadrature

Sample points

$$\int_a^b f(x)dx \simeq \sum_{k=1}^{N} w'_k f(x'_k)$$

- See example code `gaussxw.py` on Canvas for weight calculation

$$w_k = \left[\frac{2}{1-x^2}\left(\frac{dP_N}{dx}\right)^{-2}\right]_{x=x_k}$$



In-class problem

Gaussian quadrature for a simple function

- Consider the integral we evaluated earlier that has a value of 4.4

$$\int_0^2 (x^4 - 2x + 1)dx$$

  - Need to remap function to [-1,+1]: $x'_k = \frac{1}{2}(b-a)x_k + \frac{1}{2}(b+a)$
  - And the re-weighting: $w'_k = \frac{1}{2}(b-a)w_k$
- Using Gaussian quadrature (with `gaussxw.py`) with N = 3, calculate the integral

$$\boxed{\int_a^b f(x)dx \simeq \sum_{k=1}^{N} w'_k f(x'_k)}$$

- Usage:

```
from gaussxw import gaussxw
x, w = gaussxw(N)   # returns sample points (x) and weights (w) for N points
```

Choosing an integration method

- We've covered four integration methods: trapezoidal, Simpson's, Romberg, and Gaussian quadrature
- Which one is best?
  - There is no right answer to such a general question, and the choice depends on the integral being evaluated
- General rule-of-thumb:
  - Use Romberg or Gaussian integration for smooth functions
  - Use trapezoidal or Simpson's integration for noisy or rapidly varying functions

Choosing an integration method

Method	Advantages & when to use
Trapezoidal	Easy to implement; uniformly spaced data (lab data); noisy or poorly behaved data (discontinuities); Good accuracy in its adaptive form but requires more computing time
Simpson’s	Same benefits as trapezoidal rule but with better accuracy
Romberg	Excellent accuracy with limited data points for smoothly varying functions; more difficult to implement
Gaussian Quad	Same benefits as Romberg. Very easy to program; Non-uniform spacing of data; Very fast are weights are calculated (usually stored in a table); Most accurate for smoothly varying data

Multi-variable integrals

- Physics problems are usually multi-dimensional, barring some symmetry or simplication
- We can generalize the integration methods covered in this chapter to 2D or 3D

- As an example, let's inspect the 2D integral

$$I = \int_0^1 \int_0^1 f(x,y)dxdy,$$

- That can be rewritten as

$$I = \int_0^1 F(y)\,dy, \quad \text{where} \quad F(y) = \int_0^1 f(x,y)\,dx$$

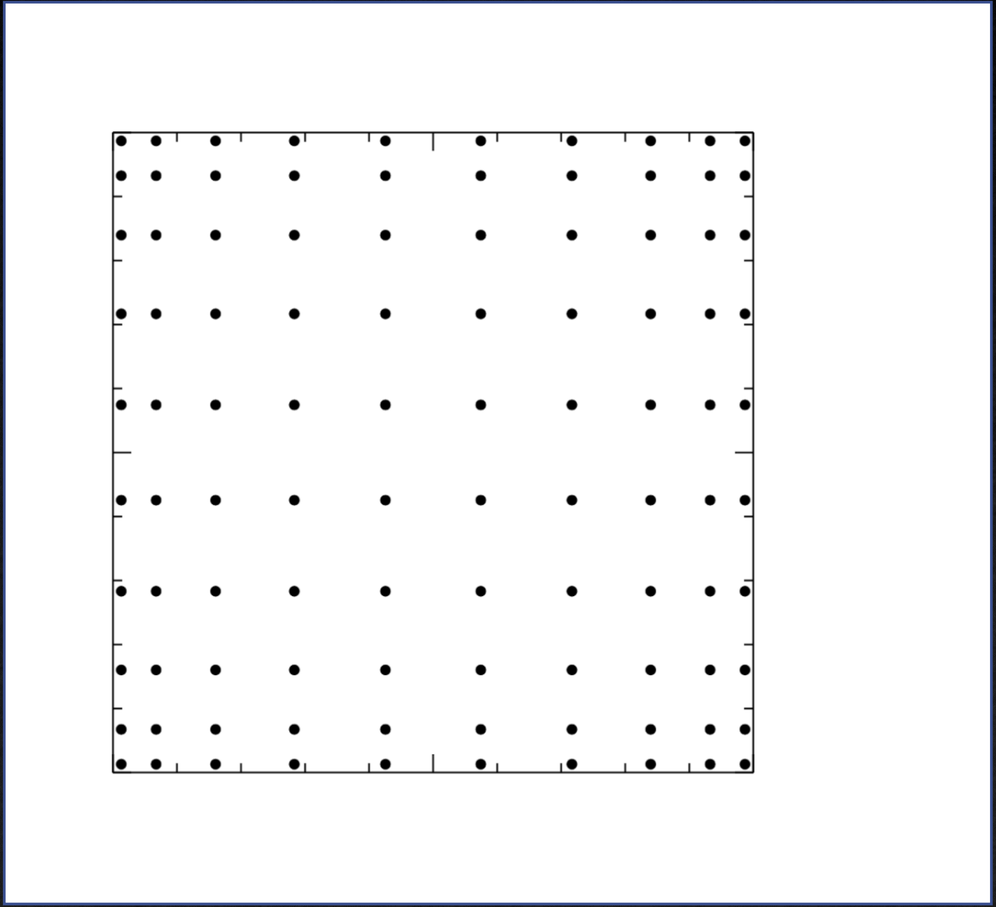
Multi-variable integrals

$$I = \int_0^1 F(y)\,dy, \quad \text{where} \quad F(y) = \int_0^1 f(x,y)\,dx$$

- We can split this into two integrals
  - First evalutate F(y) at a suitable set of y values
  - Use these F(y) values in the integral over x
- Think about it as reducing the 2D data to 1D data to a scalar (0D)

- For example if we use Gaussian quadrature with N points in each dimension

$$F(y) \simeq \sum_{i=1}^{N} w_i f(x_i, y) \quad \text{and} \quad I \simeq \sum_{j=1}^{N} w_j F(y_j).$$



Numerical differentiation

Section Outline

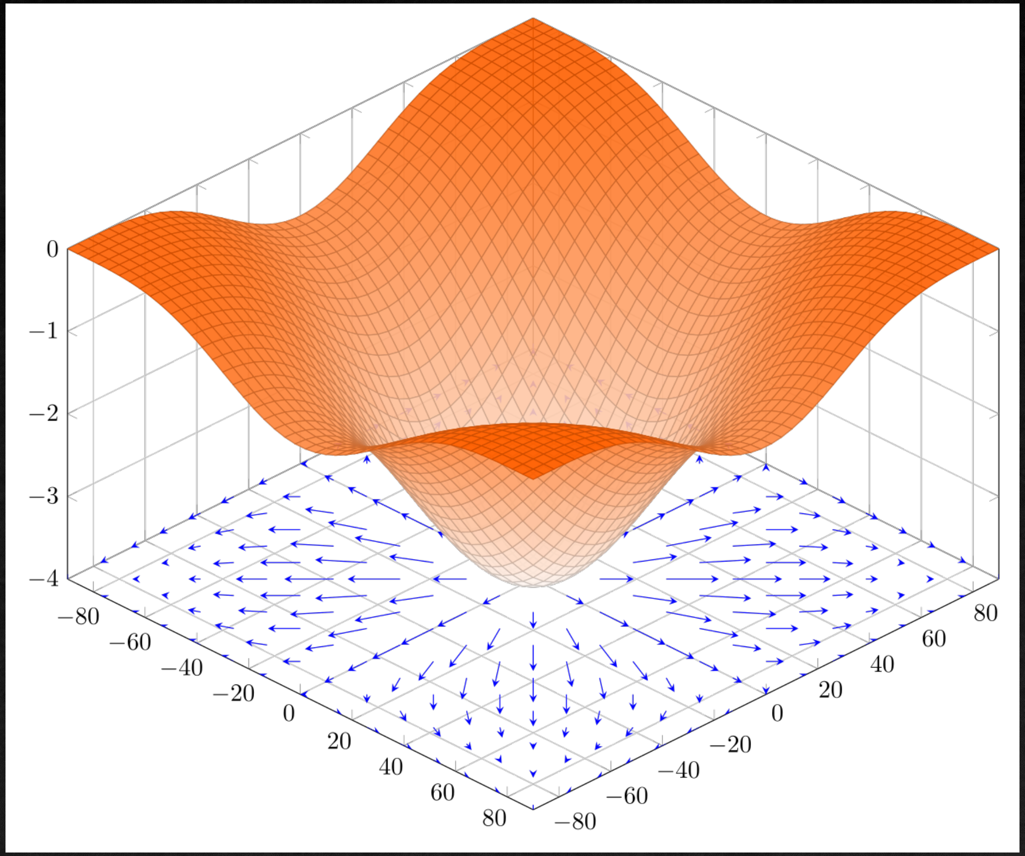
We will cover these topics

- Derivatives
  - Forward and backward diff
  - Errors
  - Central differencing
  - Noisy data
- Interpolation

Derivatives

Forward and backwards differences

- Numerical differentiation is far easier than integration
- Numerical derivatives are used less often because they can be calculated analytically for functions
- There are some practical issues with numerical derivatives and associated round-off errors, so they are not used as often
- However that being said, they are extremely important when solving partial differential equations, which we will cover in Chapter 9 (mid-March)



Derivatives

Forward and backwards differences

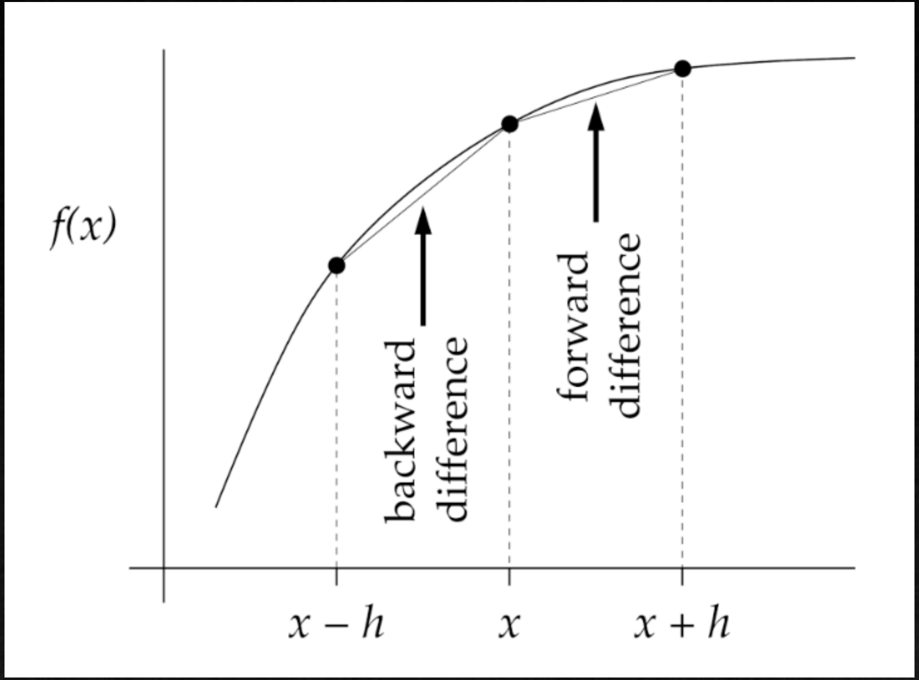
- Standard derivative definition:

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

- Instead of letting $h \to 0$, allow h to be small

$$\frac{df}{dx} \simeq \frac{f(x+h) - f(x)}{h}.$$

- This formulation is known as *forward differencing* because we're using the forward data point



- *Backwards differencing* calculates the slope in the opposite direction

$$\frac{df}{dx} \simeq \frac{f(x) - f(x-h)}{h}.$$

- Most cases use FD but BD is preferred to avoid boundaries or discontinuities

Derivatives

Numerical errors

- Using the same approach as estimating integration errors, we use a Taylor expansion of f(x) around x:

$$f(x+h) = f(x) + hf'(x) + \tfrac{1}{2}h^2 f''(x) + \ldots$$

- Solving for f'(x), we arrive at

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \tfrac{1}{2}hf''(x) + \ldots$$

- Showing that the *approximation error* is $\frac{1}{2}h|f''(x)|$ and scales linearly with step size

Derivatives

Numerical errors

$$\frac{df}{dx} \simeq \frac{f(x+h) - f(x)}{h}.$$

- Rounding error becomes an issue as the step size becomes small
  - Because we're taking the difference between two similar numbers

- Let's estimate the smallest step size *h* before the approximation errors are dominated by rounding errors
- Take the rounding error to be C = 10^{-16}

- In the worst case scenario, the difference $f(x+h) - f(x) \simeq 2C|f(x)|$
- Putting this into the difference formula, we have an overall error of $2C|f(x)|/h$

Derivatives

Numerical errors

$$\frac{df}{dx} \simeq \frac{f(x+h) - f(x)}{h}.$$

- Putting this into the difference formula, we have an overall error of $2C|f(x)|/h$
- Let's compare this with the approximation error. The total error is $\epsilon = \frac{2C|f(x)|}{h} + \frac{1}{2}h|f''(x)|.$
- We want to minimize this error, so we differentiate w.r.t. the step size h. It's minimized at

$$h = \sqrt{4C\left|\frac{f(x)}{f''(x)}\right|}.$$

- Substituting this step size back into the total error, we arrive at

$$\epsilon = h|f''(x)| = \sqrt{4C|f(x)f''(x)|}$$

where the minimal error occurs when $h \approx \sqrt{C} = 10^{-8}$ if f(x) and f''(x) are around one

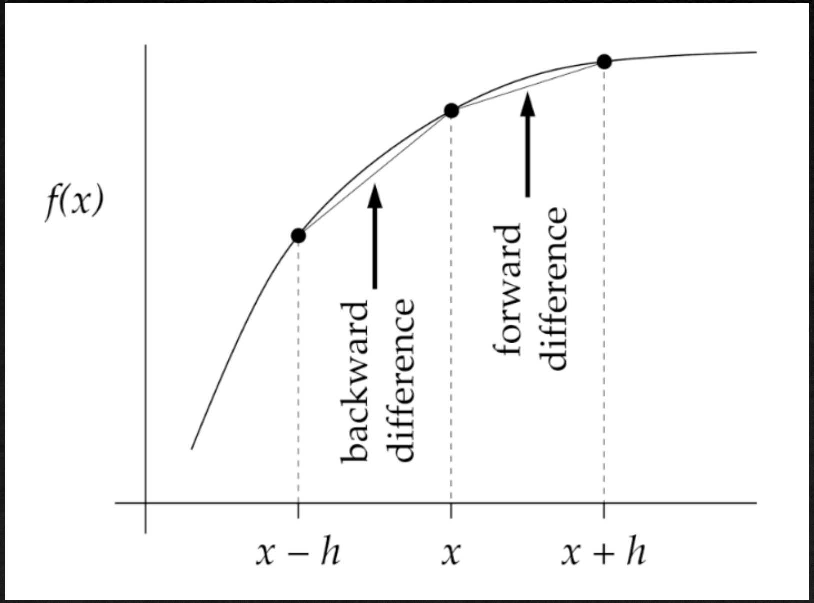
Derivatives

Central differencing

- Central differencing uses points on both the positive and negative sides

$$\frac{df}{dx} \simeq \frac{f(x+h/2) - f(x-h/2)}{h}$$

- Much more accurate than forward and backward differencing
  - Minimal error occurs at $h \simeq C^{-\frac{1}{3}} \simeq 10^{-5}$
  - Error of ~10^{-10}, a factor of 100 better than FD/BD
- There are some practical difficulties in evaluating the function at points within the slice



Derivatives

Central differencing

- There are some practical difficulties in evaluating the function at points within the slice
- This is avoided by taking the difference between the two adjacent points

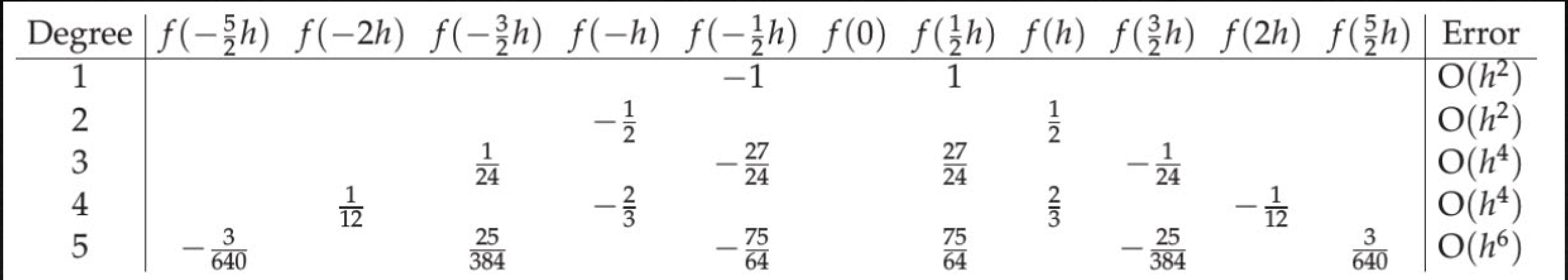
$$\frac{df}{dx} \simeq \frac{f(x+h/2) - f(x-h/2)}{h}$$

- Higher-ordered polynomials can be fit through the sample points, further improving the accuracy
- *Downside: one either needs "interior" points and/or many points*

Derivatives

Central differencing

- Higher-ordered polynomials can be fit through the sample points



Degree	$f(-\frac{5}{2}h)$	$f(-2h)$	$f(-\frac{3}{2}h)$	$f(-h)$	$f(-\frac{1}{2}h)$	$f(0)$	$f(\frac{1}{2}h)$	$f(h)$	$f(\frac{3}{2}h)$	$f(2h)$	$f(\frac{5}{2}h)$	Error
1					-1		1					$O(h^2)$
2				$-\frac{1}{2}$				$\frac{1}{2}$				$O(h^2)$
3			$\frac{1}{24}$		$-\frac{27}{24}$		$\frac{27}{24}$		$-\frac{1}{24}$			$O(h^4)$
4		$\frac{1}{12}$		$-\frac{2}{3}$				$\frac{2}{3}$		$-\frac{1}{12}$		$O(h^4)$
5	$-\frac{3}{640}$		$\frac{25}{384}$		$-\frac{75}{64}$		$\frac{75}{64}$		$-\frac{25}{384}$		$\frac{3}{640}$	$O(h^6)$

Table 5.1: Coefficients for numerical derivatives. The coefficients for central approximations to the first derivative of $f(x)$ at $x = 0$. To derive the full expression for an approximation, multiply the samples listed in the top row of the table by the coefficients in one of the other rows, then divide by h. For instance, the cubic approximation would be $[\frac{1}{24}f(-\frac{3}{2}h) - \frac{27}{24}f(-\frac{1}{2}h) + \frac{27}{24}f(\frac{1}{2}h) - \frac{1}{24}f(\frac{3}{2}h)]/h$. For derivatives at points other than $x = 0$ the same coefficients apply—one just uses the appropriate sample points around the value x of interest. The final column of the table gives the order of the approximation error on the derivative.

Derivatives

Second-order derivatives

- We can estimate the second derivative by taking the central difference of our numerical approximation of f'(x)
- Consider the first derivative at the halfway points

$$f'(x+h/2) \simeq \frac{f(x+h)-f(x)}{h}, \qquad f'(x-h/2) \simeq \frac{f(x)-f(x-h)}{h},$$

which we can difference to find the second derivative

$$\boxed{\begin{aligned} f''(x) &\simeq \frac{f'(x+h/2)-f'(x-h/2)}{h} \\ &= \frac{[f(x+h)-f(x)]/h-[f(x)-f(x-h)]/h}{h} \\ &= \frac{f(x+h)-2f(x)+f(x-h)}{h^2}. \end{aligned}}$$

Derivatives

Second-order derivatives

$$\begin{aligned} f''(x) &\simeq \frac{f'(x+h/2) - f'(x-h/2)}{h} \\ &= \frac{[f(x+h) - f(x)]/h - [f(x) - f(x-h)]/h}{h} \\ &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \end{aligned}$$

- The approximation error is the same as forward/backward differences: $\sqrt{C} \simeq 10^{-8}$
- This is the simplest expression for a second derivative. We won't be exploring higher-order approximations in this class

Derivatives

Partial derivatives

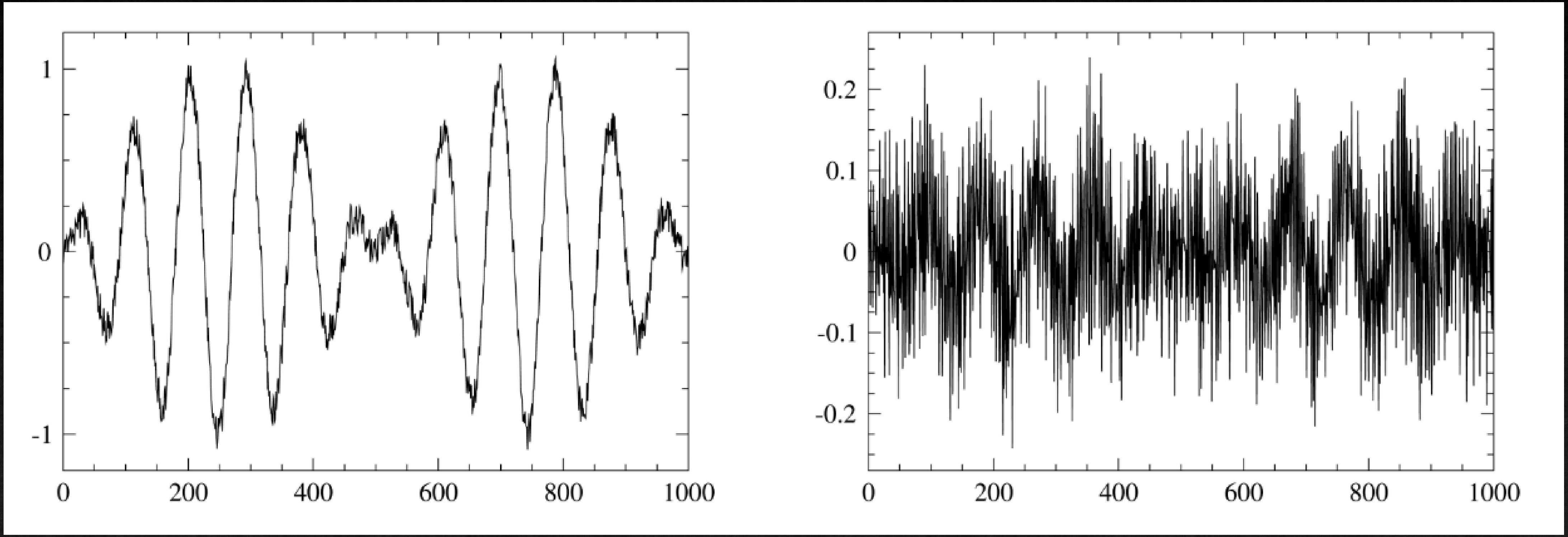
- Many physics equations have partial derivatives
- The same methods for ordinary derivatives can be used for partial derivatives
- For example, the central differences of f(x,y) are

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

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

Derivatives

Noisy data

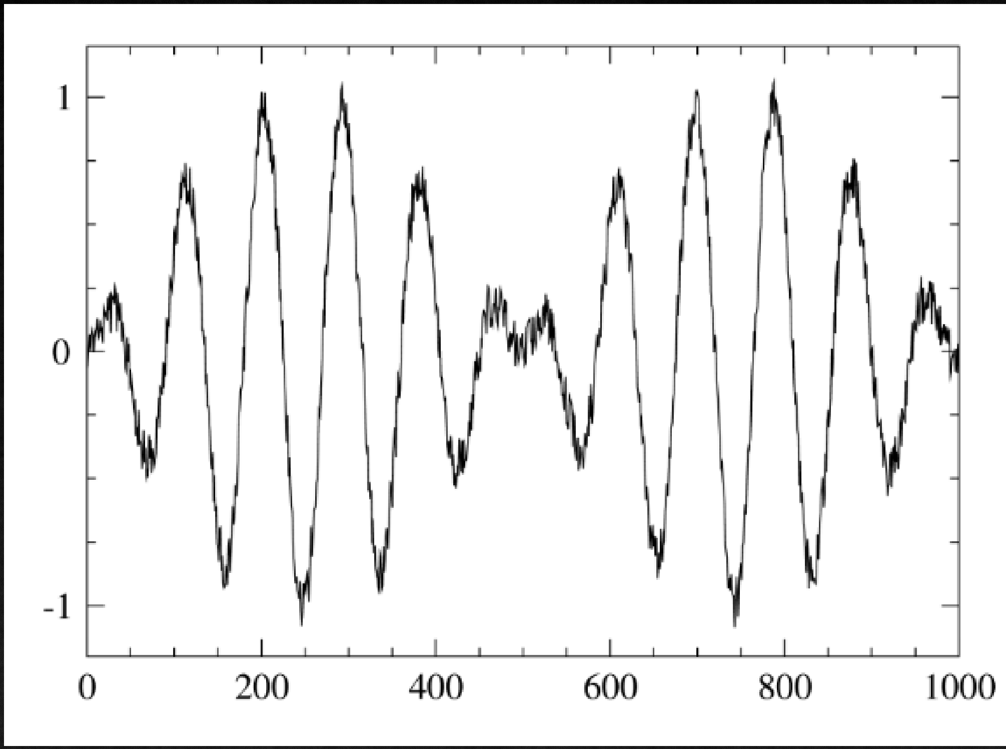


f(x)

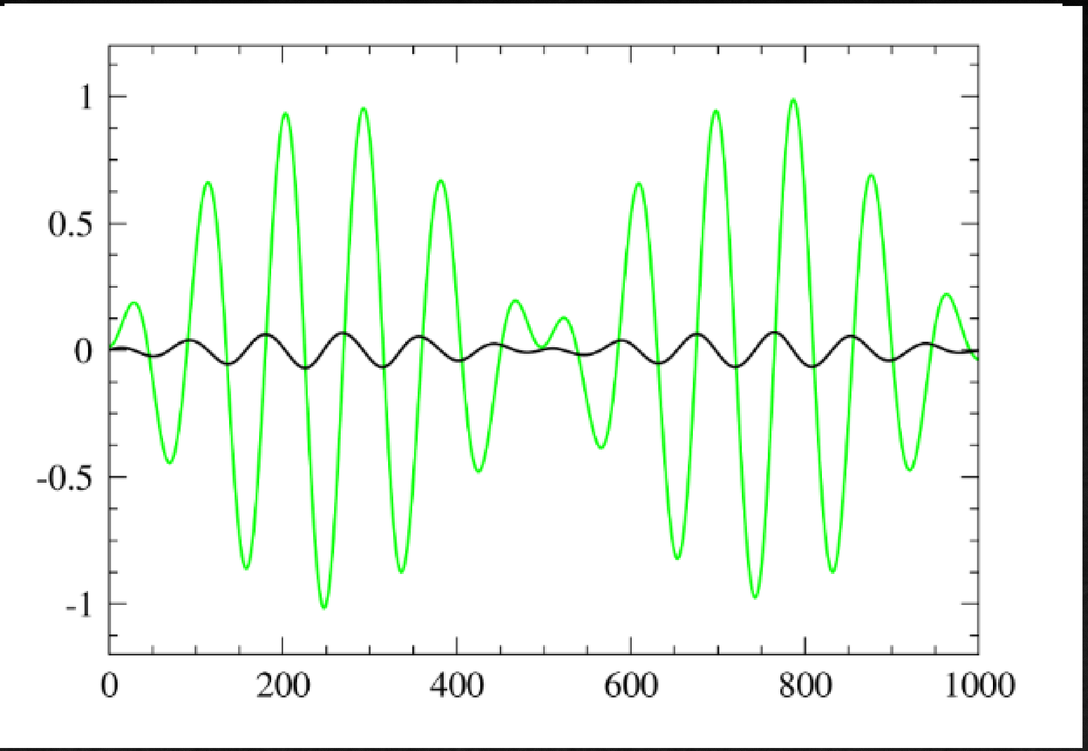
df/dx

Derivatives

Noisy data



f(x)



Smooth then difference

df/dx

Derivatives

Noisy data

Strategies

1. Increase the value of *h* so that the total error (take C to be the noise) is minimized
2. Using the least-squares method, fit an analytical solution to the data. Take the derivative of that function
3. Smooth the data without a fitting function. Take the numerical derivative of those data
   - Smoothing can be done with a running average or Fourier transforms (Newman Chapter 7)
   - The figure in the previous slide was smoothed with a Fourier transform