PHY604 Lecture 3

September 2, 2025

Today’s lecture:

- Good programming practices:
  - Debugging
  - Misc. good practices
- Numerical differentiation

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.

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

We have not really discussed general coding style

- Depends very much on the language, and is often a matter of opinion (google it)
- Some general rules:
  - 1. Use a consistent programming style
  - 2. Use brief but descriptive variable and function names
  - 3. Avoid "magic numbers"
    - Name your constants, specify your flags
  - 4. Use functions and/or subroutines for repetitive tasks
  - 5. Check return values for errors before proceeding
  - 6. Share information effectively (e.g., using modules or namespaces)
  - 7. Limit the scope of your variables, methods, etc.
  - 8. Think carefully about the most effective way to input and output data
  - 9. Be careful about memory, i.e., allocating and deallocating
  - 10. Make your code readable and portable, you will thank yourself (or your collaborators will thank you) later.

Today's lecture:

- Good programming practices:
  - Debugging
  - Misc. good practices
- Numerical differentiation

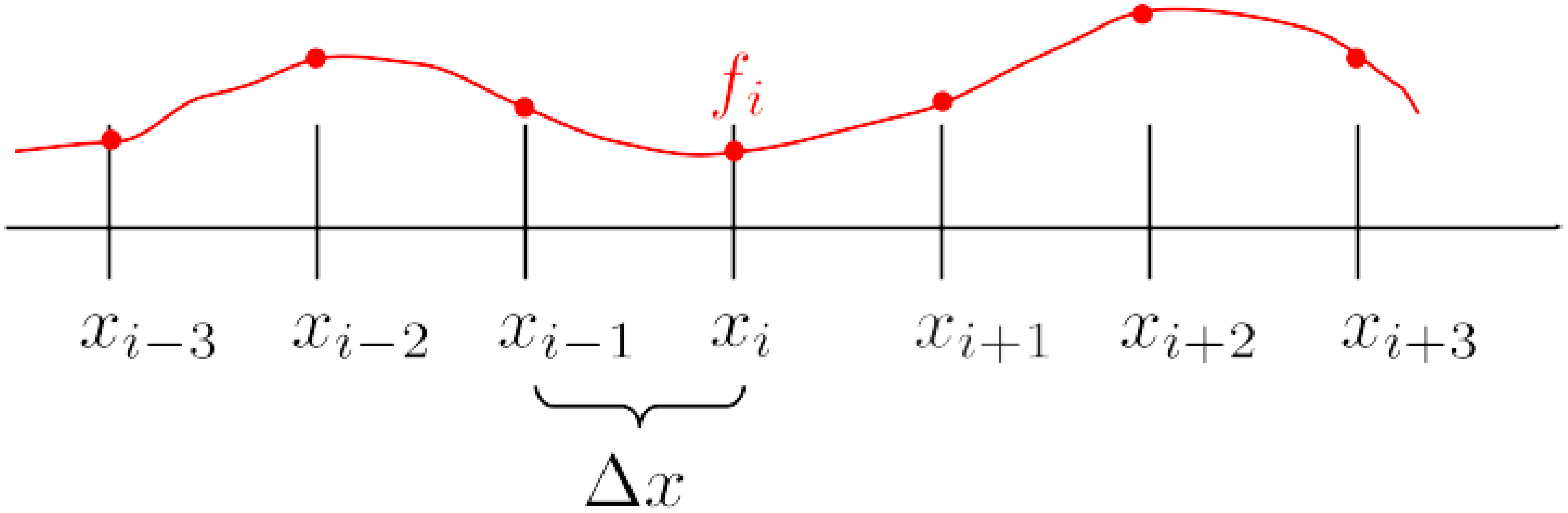
Numerical differentiation, Two situations:

- 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)

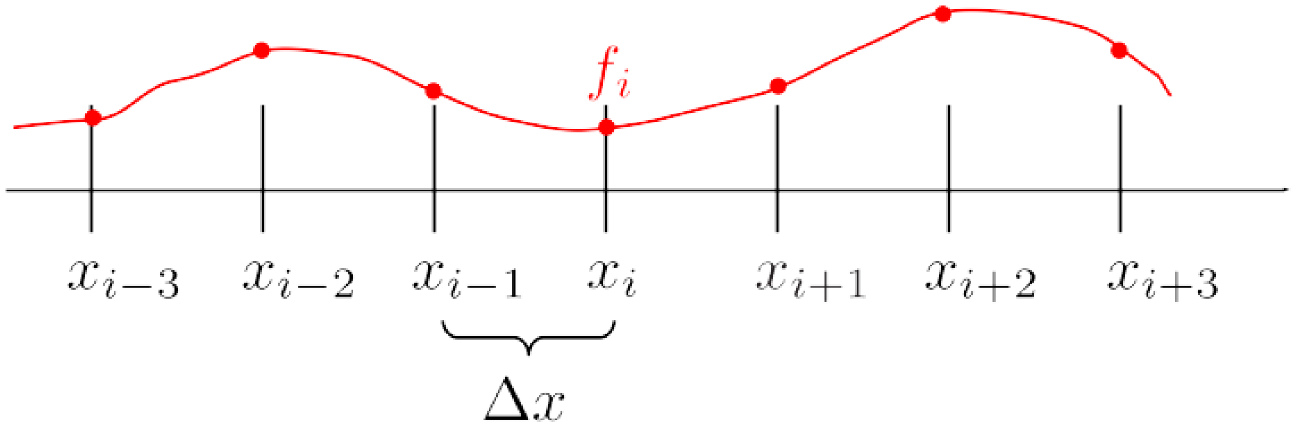
Gridded data

- 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:

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

- 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

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

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

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

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

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

- Minimize WRT 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

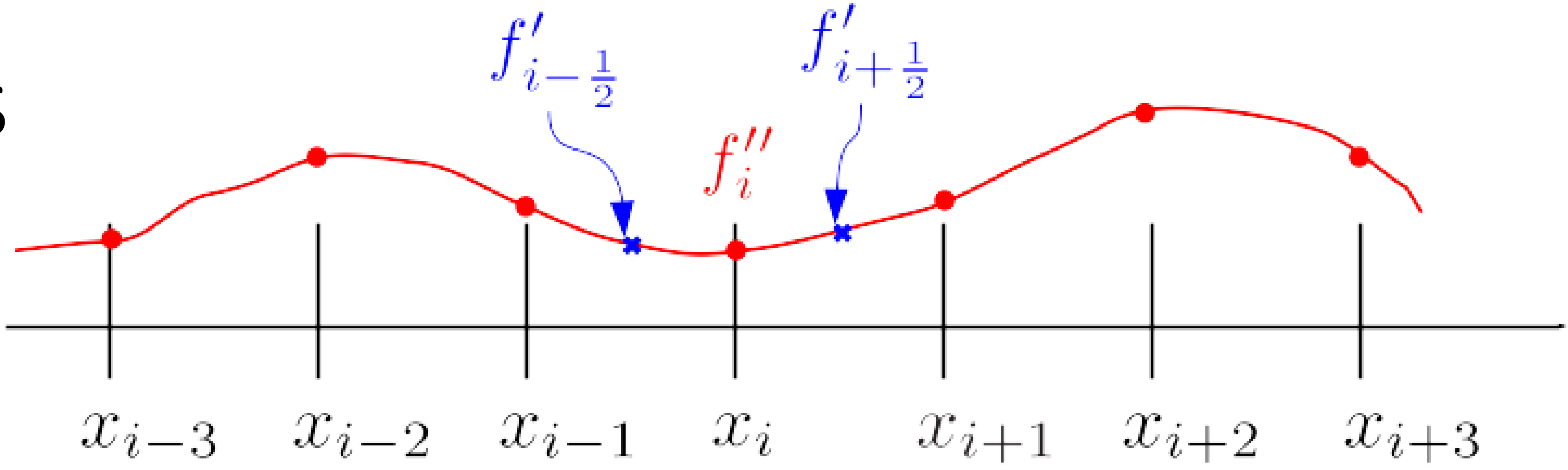
- To get accuracy to order *n* [i.e.,$\mathcal{O}(\Delta x^n)$] follow a similar strategy:
  - 1. Write down Taylor expansion for *n+1* finite difference points up to order *n*+1
  - 2. Solve set of polynomial equation in Δx for *f'*
  - 3. Obtain an expression involving weighted sum of function evaluated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

D

Higher derivatives



- Write second derivative as: $$f''_i = \frac{f'_{i+1/2} - f'_{i-1/2}}{\Delta x}$$

- Insert central 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:

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

Key takeaways from numerical differentiation

- There is a minimum error you can achieve, given by the balance between roundoff and truncation errors

- Higher-order approximations have better truncation errors, but (usually) require more function evaluations

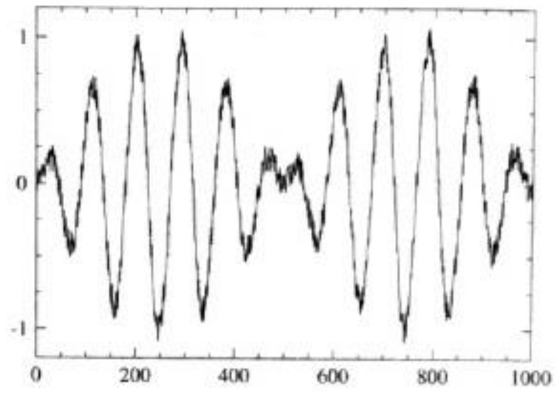
- Increasing the precision helps with roundoff errors (as usual)

Some final comments on numerical derivation

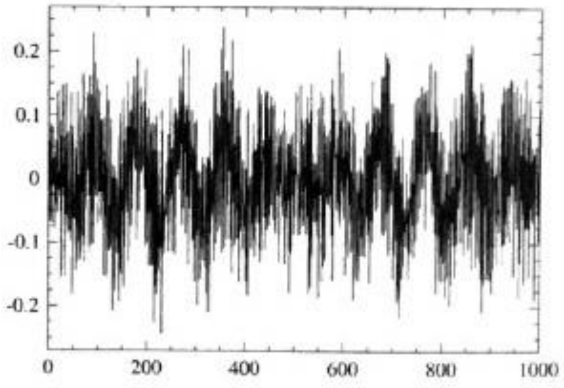
- Taking derivatives of noisy data makes the noise much worse!
  - Fit to a smooth curve and take the derivative of that
  - Smooth the data, e.g., with a Fourier transform

- 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)

After class tasks

- If you do not already have one, make an account on github: https://github.com/
- Readings:
  - Wikipedia artical on makefiles
  - Blog on numerical differentiation
  - Wikipedia page of finite difference coefficients
  - Newman Section 5.10
  - Garcia Section 10.2