PHY 604: Computational Methods in Physics and Astrophysics II

Homework #3

Due: Oct. 26, 2023

Programs can be written in any language (but python is recommended), In addition to the program, you should have a writeup that contains the plots requested in the homework questions, answers to any analytical or explanation questions, and a short description of your code and how to run it. This can be done in, e.g., LaTeX, markdown, etc. Combining the code and writeup in jupyter notebooks is highly recommended.

Code and writeup should be submitted using `git` *via github in the repo that was created from* github classroom *link.*

1. (*Matrix inverse*) (based on Garcia) An iterative method for constructing the matrix inverse can be found via Newton's method. A single step in the iteration appears as:

$$\mathbf{X}^{(k+1)} = 2\mathbf{X}^{(k)} - \mathbf{X}^{(k)}\mathbf{A}\mathbf{X}^{(k)} \tag{1}$$

where $\mathbf{A}$ is the matrix whose inverse we seek and $\mathbf{X}^{(k)}$ is our current guess for the inverse.

Find the inverse of the following matrix using this technique:

$$\mathbf{A} = \begin{pmatrix} 4 & 3 & 4 & 10 \\ 2 & -7 & 3 & 0 \\ -2 & 11 & 1 & 3 \\ 3 & -4 & 0 & 2 \end{pmatrix} \tag{2}$$

You will need to supply an initial guess and a desired tolerance. Be careful with your initial guess—some choices will diverge. A choice that works well is:

$$\mathbf{X}^{(0)} = \alpha\mathbf{A}^{\intercal} \tag{3}$$

where α is a small, positive number.

2. (*LU decomposition*)

(**a**) Write a program that solves a linear system of equations ($\mathbf{Ax} = \mathbf{b}$) via LU decomposition. The program should generate the lower-triangular matrix $\mathbf{L} = \mathbf{L}_0^{-1}\mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\ldots\mathbf{L}_{N-1}^{-1}$, where $\mathbf{L}_i$ is the matrix that performs a step of the forward substitution, and the upper-triangular matrix $\mathbf{U} = \mathbf{L}_{N-1}\ldots\mathbf{L}_2\mathbf{L}_1\mathbf{L}_0\mathbf{A}$. Then, the solution of the system is obtained with two back-substitution steps, $\mathbf{Ly} = \mathbf{b}$ and $\mathbf{Ux} = \mathbf{y}$ (see, e.g., slide 29 of Lecture 9).

(**b**) Test your program on the system:

$$\begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix} \tag{4}$$

(**c**) ***Bonus +5 pts***: Include partial pivoting in your program and test with:

$$\begin{pmatrix} 0 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix} \tag{5}$$

over...

3. (*Least squares with different basis functions*)

(**a**) In class, we discussed how to perform a general least-squares fitting using a sum of basis functions with coefficients that are fitted (e.g., see Lecture 12, slides 30-32). In the example program discussed in class (see `least_square.ipynb`), we used polynomials as our basis functions (i.e., powers of x). Write a program that performs the fitting using a similar approach, but with ***Legendre polynomials*** as basis functions.

(**b**) Consider the noisy quadratic polynomial constructed in `least_square.ipynb`. Show that you can get a good fit of this function with your program with $M = 3$ coefficients. Increase to $M = 10$. Do you get a better fit? How does the condition number of the matrix $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ change (for python use `numpy.linalg.cond`)? How does the condition number compare to using simple polynomials?

(**c**) Now repeat (**b**) for the function in the range $[-1, 1]$. How does the condition number change? Why do you think this is?

4. (*2D Fourier transform*): Recall the discussion in class that a 2D DFT Fourier transform could be performed as a sequence of two 1D FFTs. Specifically, we first perform a DFT for each row of the data:

$$F'_{ml} = \sum_{n=0}^{N-1} f_{mn} \exp\left(-i\frac{2\pi ln}{N}\right),$$

and then again for the columns of the transformed points:

$$F_{kl} = \sum_{m=0}^{M-1} F'_{ml} \exp\left(-i\frac{2\pi km}{M}\right).$$

(**a**) Write a program that performs a 2D Fourier transform using the *fast Fourier transform* (FFT) algorithm discussed in class (as opposed to the conventional DFT).

(**b**) As a first test, consider the function $f(x, y) = \sin(2\pi x/10)$ on a 64×64 grid (see Fig. 1). Plot the 2D FFT and explain why it looks the way it does. Show that your program works by performing an inverse 2D FFT and reproducing the original function

(**c**) Do the same as in (b) with the function $f(x, y) = \sin(2\pi x/10 + 4\pi y/10) + \sin(6\pi x/10 + 8\pi y/10)$.

(**d**) Generate a 2D function of your choice and take its FFT.

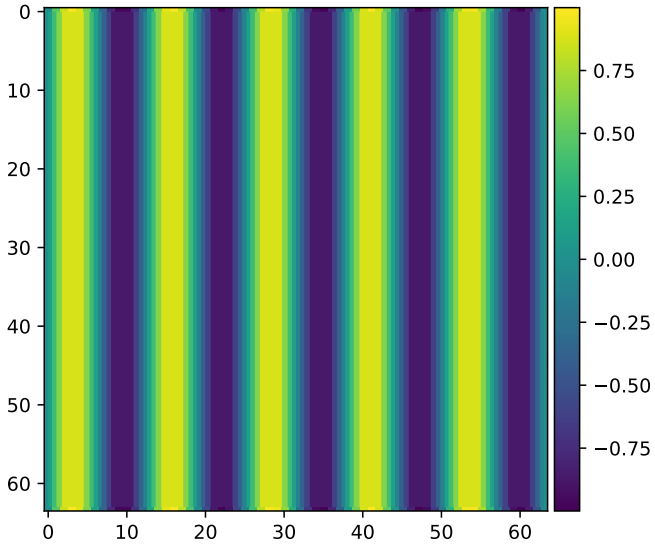

Figure 1: $f(x, y) = \sin(2\pi x/10)$ on a 64×64 grid