

Submit your homework as a SINGLE PDF via UNM Learn by the due date.

⇒ **Only PDFs will be accepted.**

General Guidelines for Write-Up Style

- **What to hand in:** Single PDF of the entire homework with your name on each page.

This is your narrative explanation of what was done, your answers to the specific questions, and an explanation of how you arrived at your answers. If discussion is required for a question, include that. If it's only a proof, then include the theorem to be proved, followed by your concise, logical, but fully explained proof.

- **Report:** In general, your report needs to read coherently and at the level expected for a graduate course. For instance, when you answer a question, make sure that your report contains all the necessary information to understand your answer. What is the question? How did you answer it? Is some discussion required? If so, include it. If you used code, include that as well.

- **Typesetting:** If you write your answers by hand, then make sure that your handwriting is readable. Otherwise, I cannot grade it. If typesetting, please use Latex or for programming problems, annotated Jupyter notebook print-outs. Note, you can annotate Jupyter notebooks with text blocks containing Latex.

- **Plots:** All plots/figures in the report must be generated in Python and not hand drawn (unless otherwise specified in the homework question).

In general, make sure to (1) title figures, (2) label both axes, and (3) include a legend for the plotted datasets. The font-size of all text in your figures must be large and easily readable.

- **Partners:** You are encouraged (but not required) to work in pairs (group of two students) for homeworks. Hand in a single report on Learn with both collaborators cited at the top. It is expected that both of you can explain the theory and computer codes. Groups of more than two students are not allowed.

- **Tips for generating PDF:** Many of you use Latex/Overleaf, and this is the recommended way to generate PDFs. For Microsoft Word, and other word processors, you can choose PDF inside of the "File → Save As" menu. This is another way to generate a PDF of your entire write-up. Next, are some tips on how to generate and insert PDFs of your Jupyter notebook code and pencil-and-paper work into your write-up.

Jupyter code notebooks allow for easy printing to PDF. Go to "File → Download As → PDF". Or, you can go to "File → Print Preview", and then print/save to a PDF via your web browser.

For pencil and paper work, cell phones often work well enough for generating a PDF. Use a program like *genius scan* <https://thegrizzzlylabs.com/genius-scan>, or *tiny scanner* for easily converting cell phone pictures into a PDF.

Finally, you'll want to put your Jupyter PDF(s) of code, and any other PDFs of your work, together into one single PDF. For Latex/Overleaf, insert `\usepackage{pdfpages}` in your document preamble, and then type `\includepdf[page=-]{filename}` to insert a PDF inside your Latex document.

Mac Preview lets you open multiple PDFs and then do "Print → PDF" to save to a global PDF document. Microsoft Word also lets you drag and drop PDFs onto blank pages. So, you'd drag page 1 of your code PDF into Word. Then drag page 2, and so on. For merging PDFs into one, you can also use an online tool, but if you use these, make sure to never upload sensitive information. For instance, there's <https://smallpdf.com/merge-pdf> which merges multiple PDFs into one.

1. Trefethen and Bau 10.2 using Python (not Matlab). You may use code from the course website, but remember, your code should work for complex arithmetic.
2. (Based on Trefethen and Bau 10.3)

Let Z be the matrix

$$Z = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 7 \\ 4 & 2 & 3 \\ 4 & 2 & 2 \end{bmatrix}$$

Compute three reduced QR factorizations of Z . Use the "mgs()" routine from the last homework that implements modified Gram-Schmidt. Use the "house()" and "formQ()" routines from 10.2 to compute the second QR . And for the third QR , use NumPy's library routine. Note that you can call `numpy.linalg.qr(Z, mode='reduced')`.

Compare the output from these three routines, and comment on any differences you see. The differences may be large, or small.

3. (Based on Trefethen and Bau 11.3)

Take $m = 50$ and $n = 12$. Use `numpy.linspace` to define t to be an m -vector of m linearly (i.e., evenly) spaced points from 0 to 1. Using `numpy.vander`, define A (based on t) to be the $m \times n$ Vandermonde matrix associated with the least-squares fitting of a polynomial of degree $n - 1$. Take b to be the function $\cos(4t)$ evaluated on the grid t . Now, calculate and print to 16 digits the least squares polynomial coefficient vector x with the following methods

- (a) Using the normal equations and `numpy.linalg.solve`
- (b) Using QR and your “mgs()” routine from earlier.
- (c) Using QR and your “house()” routine from earlier.
- (d) Using Numpy’s library QR .
- (e) Using Numpy’s library SVD. Note that you can call
`numpy.linalg.svd(A, full_matrices=False)` to get the reduced SVD.

The calculations above will yield 5 lists of twelve coefficients (each list defines the polynomial coefficients computed using one of the methods). In each list, circle/highlight which digits appear to be wrong because of rounding error. You may want to print these lists of coefficients side-by-side in a table. You can use the SVD as a reference solution.

You do not need to explain your observations, other than please comment on the normal equations, and whether they appear to exhibit (in)stability.

- 4. Trefethen and Bau 12.1
- 5. Trefethen and Bau 12.2 parts (a), (b) and (c)