Homework 2

Due: Friday 1:30pm, February 28, 2020, **in recitation**

Notes on the first (and all subsequent) assignments:

Submission: Homework assignments must be submitted in the class on the due date. If you cannot attend the class, please send your solution per email as a single PDF before class. Please hand in cleanly handwritten or typed (preferably with LaTeX) homework. Feel free to use original homework LaTeX document to write-up your homework. If you are required to hand in code or code listings, this will explicitly be stated on that homework assignment.

Collaboration: NYU's integrity policies will be enforced. You are encouraged to discuss the problems with other students. However, you must write (i.e. type) every line of code yourself and also write up your solutions independently. Copying of any portion of someone else's solution/code or allowing others to copy your solution/code is considered cheating.

Plotting and formatting: Plot figures carefully and think about what you want to illustrate with a plot. Choose proper ranges and scales (e.g. semilogx, semilogy, loglog), always label axes, and give meaningful titles. Sometimes, using a table can be useful, but never submit pages of numbers. Discuss what we can observe in, and learn from, a plot. If you do print numbers, in MATLAB for example, use `fprintf` to format the output nicely. Use `format compact` and other format commands to control how MATLAB prints things. When you create figures using MATLAB (or Python or Julia), please try to export them in a vector graphics format (.eps, .pdf, .dxf) rather than raster graphics or bitmaps (.jpg, .png, .gif, .tif). Vector graphics-based plots avoid pixelation and thus look much cleaner.

Programming: This is an essential part of this class. We will use MATLAB for demonstration purposes in class, but you are free to use other languages. The TA will give an introduction to MATLAB in the first few recitation classes. Please use meaningful variable names, try to write clean, concise and easy-to-read code. As detailed in the syllabus, in order to receive full credit, your code must be thoroughly commented.

---

1. Newton's method can be extended to *matrix-functions* as well. For example, given a square matrix $\mathbf{A}$ and real number t, the *matrix-exponential* $e^{t\mathbf{A}}$ is defined via the Taylor series for the exponential function:

$$e^{t\mathbf{A}} = 1 + t\mathbf{A} + \frac{(t\mathbf{A})^2}{2!} + \frac{(t\mathbf{A})^3}{3!} + + \frac{(t\mathbf{A})^4}{4!} + \dots \qquad (1)$$

Obviously, the matrix $\mathbf{A}$ must be square.

   (a) **[4pts]** Derive Newton's method for finding the root of an arbitrary matrix-valued function $f = f(\mathbf{X})$, where by *root* we mean that $\mathbf{X}$ is a root of f if $f(\mathbf{X}) = \mathbf{0}$, where $\mathbf{0}$ is the matrix of all zeros. Assume that the matrix argument of f is square and invertible.

   (b) **[3pts]** The square root of a matrix $\mathbf{A}$ is a matrix $\mathbf{X}$ such that $\mathbf{X}^T\mathbf{X} = \mathbf{A}$. For a symmetric positive-definite matrix $\mathbf{A}$, derive the Newton iteration for finding $\mathbf{X} = \sqrt{\mathbf{A}}$.

(c) **[3pts]** Write a program using the Newton iteration that you derived above to find the square root of the matrix

$$\mathbf{A} = \begin{pmatrix} 8 & 4 & 2 & 1 \\ 4 & 8 & 4 & 2 \\ 2 & 4 & 8 & 4 \\ 1 & 2 & 4 & 8 \end{pmatrix}. \tag{2}$$

The stopping criterion for your Newton iteration should be when the absolute difference between elements of successive iterations is at most 10^{-10}. Submit your code, as well as a *cleanly formatted* printout of the square root of the above matrix, displaying elements to 2 decimal places.

2. **[10pts]** For a yearly interest rate $0 < r < 1$ compounded over n intervals, an amount of money C grows to be

$$f(C, r, n) = C \left(1 + \frac{r}{n}\right)^n \tag{3}$$

after one year.

Let $r = 0.025$. If n is *extremely* large, say $n = 10^{16}$, in IEEE double precision arithmetic,

$$f\left(1.0, 0.025, 10^{16}\right) = 1.0, \tag{4}$$

when in fact,

$$\begin{aligned} f(1.0, 0.025, 10^{16}) &\approx \lim_{n\to\infty} \left(1 + \frac{0.025}{n}\right)^n \\ &= e^{0.025} \\ &\approx 1.025315\ldots \end{aligned} \tag{5}$$

What happened? Without using the above approximation, is there another way to evaluate f and not lose these digits of accuracy? (Assume you are only allowed to use double precision accuracy.)

3. Let the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, with $m \geq n$, have rank n and be such that when computing its $\mathbf{LU}$ decomposition $\mathbf{A} = \mathbf{LU}$ no pivoting whatsoever is needed.

(a) **[5pts]** Carefully derive the computational cost (in floating point operations, referred to as flops) of computing $\mathbf{A} = \mathbf{LU}$. Assume that addition, subtraction, multiplication, and division each count as 1 flop. For example, computing $x = a(b + c)/d - e$ requires 4 flops.

(b) **[5pts]** Now suppose that the matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is such that in computing its $\mathbf{LU}$ factorization row-pivoting is required. This means that we can write $\mathbf{PA} = \mathbf{LU}$ where $\mathbf{P}$ is a permutation matrix. Find $\mathbf{P}$, $\mathbf{L}$, and $\mathbf{U}$ for the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 \\ 6 & 6 & 0 \\ 3 & 0 & 2 \end{pmatrix}. \tag{6}$$

If you do not show your work, you will not receive any credit for the problem.

4. Consider the $n \times n$ linear system
$$\mathbf{Ax} = \mathbf{b},$$
where $\mathbf{A}$ has non-zero entries:
$$\begin{aligned} A_{ij} &= 1, && \text{if } i = j, \\ A_{ij} &= -1, && \text{if } i > j, \\ A_{in} &= 1, && \text{for all } i, \\ A_{ij} &= 0, && \text{otherwise.} \end{aligned}$$
For example, when $n = 5$, the matrix $\mathbf{A}$ is given by
$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix}.$$

   (a) **[3pts]** Using induction, show that $\mathbf{A}$ is invertible for all $n > 0$.

   (b) **[2pts]** In the case where $n = 5$ and $\mathbf{b} = \begin{pmatrix} 1 \, \frac{1}{4} \, \frac{1}{9} \, \frac{1}{16} \, \frac{1}{25} \end{pmatrix}^T$, solve $\mathbf{Ax} = \mathbf{b}$ using Gaussian elimination.

   (c) **[2pts]** For an arbitrary n, find the entries L_{ij} and U_{ij} of the $\mathbf{LU}$-factorization of $\mathbf{A}$. What is $\max_{ij} |U_{ij}|$?

   (d) **[2pts]** For large values of n, for example $n = 2000$, what problems can you see with trying to solve $\mathbf{Ax} = \mathbf{b}$ on a computer using double precision floating point arithmetic? Explain your reasoning.