MTH 373/573:
Scientific Computing

Homework 2

IIIT-Delhi, Monsoon 2021
Kaushik Kalyanaraman

Due by: September 18, 2021 (Saturday) by 23:59 IST — Total: 80 points

Late submission by: ~~September 18, 2021 (Saturday) by 23:59 IST~~ — Total: ~~64~~ points

Guidelines

- Please consult with and carefully read **IIIT-Delhi's Academic Dishonesty Policy** at this link.
- *This assignment should be answered individually.* You are not allowed to discuss specifics of your solutions, solution strategies, or coding strategies with any other student in this class. There are only two exceptions:
  1. You may post queries relating to any HW question to `#homeworks` channel of the class's Discord server.
  2. You may reach out to me or one of the Ph.D. TAs – Archana (archanaa@iiitd.ac.in) or Gaurav (gauravr@iiitd.ac.in) with your clearly phrased questions about any HW problem.
- Start working on your solutions early and don't wait until close to due date.
- Each problem should have a clear and concise write-up.
- Please clearly show all steps involved in your solution.
- A late submission can turned within an additional 2 days with grading rescaled to 0.6 of total points.
- You will need to write a separate code for each computational problem and sometimes for some subproblems too. You should name each such file as `problem_`*n*`.py` where *n* is the problem number. For example, your file names could be `problem_5.py` `problem_6a.py`, `problem_6b.py`, and so on.
- *Python tip:* You can import Python modules as follows:

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import numpy.linalg as npla
import scipy.linalg as spla
```

- You can submit IPython Notebook files `*.ipynb` as well. Please use the same naming convention as above. If you choose to do this, please clean your output and also provide an exported HTML file. You can go to a Notebook's File menu options and choose `Kernel -> Restart & Run All` for a clean output. For exporting to HTML, you carry out `File -> Download as -> HTML (.html)`.

Inputs : A, b
Output : x

```
for k = 1, k ≤ n − 1, k++ do
    for i = k + 1, i ≤ n, i++ do
        a_temp ⟵ a_ik / a_kk
        a_ik ⟵ a_temp
        for j = k + 1, k ≤ n, j++ do
            a_ij ⟵ a_ij − a_temp a_kj
        end
        b_i ⟵ b_i − a_temp b_k
    end
end
x_n ⟵ b_n / a_nn
for i = n − 1, i ≥ 1, i-- do
    sum ⟵ b_i
    for j = i + 1, j ≤ n; j++ do
        sum ⟵ sum − a_ij x_j
    end
    x_i ⟵ sum / a_ii
end
```

Algorithm 1: Gaussian elimination without partial pivoting

Inputs : A, b
Output : x

for $i = 1,\ i \leq n,\ i$++ **do**
  $\ell_i \longleftarrow i, \quad s_{\max} \longleftarrow 0$
  for $j = 1,\ j \leq n,\ j$++ **do**
    $s_{\max} \longleftarrow \max\{s_{\max}, |a_{ij}|\}$
  end
  $s_i \longleftarrow s_{\max}$
end
for $k = 1,\ k \leq n-1,\ k$++ **do**
  $r_{\max} \longleftarrow 0$
  for $i = k,\ i \leq n,\ i$++ **do**
    $r \longleftarrow |a_{\ell_i,k}/s_{\ell_i}|$
    if $r > r_{max}$ **then**
      $r_{\max} \longleftarrow r, \quad j \longleftarrow i$
    end
  end
  $\ell_{\text{temp}} \longleftarrow \ell_k, \quad \ell_k \longleftarrow \ell_j, \quad \ell_j \longleftarrow \ell_{\text{temp}}$
  for $i = k+1,\ i \leq n,\ i$++ **do**
    $a_{\text{mult}} \longleftarrow a_{\ell_i,k}/a_{\ell_k,k}$
    $a_{\ell_i,k} \longleftarrow a_{\text{mult}}$
    for $j = k+1,\ j \leq n,\ j$++ **do**
      $a_{\ell_i,j} \longleftarrow a_{\ell_i,j} - a_{\text{mult}} a_{\ell_k,j}$
    end
  end
end
for $k = 1,\ k \leq n-1,\ k$++ **do**
  for $i = k+1,\ i \leq n,\ i$++ **do**
    $b_{\ell_i} \longleftarrow b_{\ell_i} - a_{\ell_i,k} b_{\ell_k}$
  end
end
$x_n \longleftarrow b_{\ell_n}/a_{\ell_n,n}$
for $i = n-1,\ i \geq 1,\ i$– **do**
  $sum \longleftarrow b_{\ell_i}$
  for $j = i+1,\ j \leq n,\ j$++ **do**
    $sum \longleftarrow sum - a_{\ell_i,j} x_j$
  end
  $x_i \longleftarrow sum/a_{\ell_i,i}$
end

Algorithm 2: Gaussian elimination with partial pivoting

Problem 1: Forward Substitution on Steroids — 8 points

Suppose we are given a linear system $Ax = b$, we say *we can solve for x without inverting A* when we mean we can use Gaussian elimination (with or without partial or full pivoting as appropriate) to compute x. For example, we can solve for $y = A^{-1}Bx$ by first computing $z = Bx$ and then *solving for y* using $Ay = z$. In case of triangular linear systems, such solution would involve use of forward and back substitutions.

Using this background, suppose you are given $L, K \in \mathbb{R}^{n\times n}$ which are both lower triangular matrices, and $B \in \mathbb{R}^{n\times n}$ any general matrix. Then, specify an algorithm for computing $X \in \mathbb{R}^{n\times n}$ so that $LXK = B$.

Problem 2: Gaussian elimination and partial pivoting — 60 points

(a) Write a function to implement Gaussian elimination with no pivoting as in algorithm 1. (15 points)

(b) Write a function to implement Gaussian elimination with partial pivoting as in algorithm 2. (15 points)

(c) Test your code. For each of the examples as stated below, report the various measurements as below in a table. Use relative measurements as necessary, and 2-norm for all appropriate norms and condition numbers. You can use functions available from `NumPy` or `SciPy` for these purposes. (30 points)

- The condition number (`np.linalg.cond`),
- the error from un-pivoted solve from your function in part (a),
- the residual from un-pivoted solve from your function in part (a),
- the error from partially-pivoted solve from your function in part (b),
- the residual from partially-pivoted solve from your function in part (b),
- the error from `np.linalg.solve`,
- the residual from `np.linalg.solve`.

Report if any of the solves fail. You should try to write your code so that it prints the above table automatically without user interaction.

For each matrix below, write 2 or 3 sentences on your observations with regard to conditioning, necessity of pivoting, and reasons for your observed results.

Use the following matrices of sizes $n = 10, 20, 30, 40$ to test your code:

1. A *random* matrix of size $n\times n$ with entries uniformly sampled from $[0, 1)$ (use `np.random.randn`).
2. The Hilbert matrix given by:

$$a_{ij} = \frac{1}{i + j - 1} \qquad (i, j = 1, \ldots, n).$$

3. The matrix given by

$$A_n = \begin{bmatrix} 1 & & & \cdots & 1 \\ -1 & 1 & & \cdots & 1 \\ -1 & -1 & 1 & \cdots & 1 \\ \vdots & & & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & 1 \end{bmatrix}$$

For each case, let $x^\star = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \in \mathbb{R}^n$ be the vector of all 1s of size n. Now, compute b as the matrix-vector product $A\,x^\star$ and solve the linear system $A\,x = b$.

Problem 3: Scaled Linear Systems **5 + 5 + 2 = 12 points**

A matrix $A = \left[a_{ij}\right]_{n \times n}$ is said to *row equilibrated* if it is scaled so that $\max\left|a_{ij}\right| = 1,\ 1 \leq i \leq n$. In solving a system of equations $Ax = b$, we can produce an equivalent system in which the matrix is row equilibrated by dividing the i^{th} equation by $\max\limits_{1 \leq j \leq n} \left|a_{ij}\right|$.

(a) Solve the system of equations:

$$\begin{bmatrix} 1 & 1 & 2 \times 10^9 \\ 2 & -1 & 10^9 \\ 1 & 2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

by using Gaussian elimination with partial pivoting and the function you wrote for it in Problem 2.

(b) Solve the same problem but now changing the linear system into a *row equilibrated* one and using Gaussian elimination *without* partial pivoting. You can reuse the function you wrote in Problem 2. Are the answers the same? Why or why not?

(c) State your reasoning for the *why or why not* part in 2 to 4 sentences.

The reasoning part is open ended and worth only 2 points. However, if you do not provide an explanation for your observations, you will not get those 2 points.

Submission Notes

1. The answer to all theoretical problems, output of Python code for computational problems including figures, tables and accompanying analyses should be provided in a single PDF file along with all your code files.
2. You can typeset (please consider using LaTeX if you choose to do so), or write by hand and scan/take photographs of solutions for theoretical questions. For scanning or photography, please put in the extra effort and provide a single PDF file of your submission.
3. For Problem 2, submit your code in one file: `problem_2.py`. You can use the Python file provided as a boilerplate.
4. For Problem 3, submit your code in one file again: `problem_3.py`. You will of course, copy-paste the functions written in `problem_2.py` for Gaussian elimination without and with partial pivoting into this file.