Math 260, Fall 2020
Last updated: 9/11/20

HOMEWORK 4

DUE FRI. SEP. 18

Exercises. (Nothing to submit here).

E1 (operation count for GE). A few details about Gaussian elimination you may find useful when implementing. Refer to the Gaussian elimination algorithm.

a) When reducing column k, which entries of the matrix A are updated?

b) Show that the number of multiplications required for the LU factorization (with Gaussian elimination) is $n^3/3 + O(n^2)$. A useful fact is that

$$\sum_{k=1}^{n} k^p = \frac{1}{(p+1)} k^{p+1} + O(k^p)$$

for ayn integer $p \geq 1$ (in particular, for $p = 1$ and $p = 2$).

Programming problems.

Q1 (Gaussian elimination). Some template code is provided. Note: You may want to implement the version without pivoting first, get that to work, then add pivoting. I've included both versions of the template (pivoting/no pivoting). Your sbumission only needs to have one version of the algorithm.

a) Modify the 'forward solve' code (from class) to take in the combined L and U matrix from Gaussian elimination and can solve $Ly = b$ (so it will ignore the U part).

Update your backwards solve code as needed, again so that it takes the combined matrix and can solve $Ux = y$ (note that there may not be anything to do).

b) Implement Gaussian elimination with partial pivoting to find the LU factorization of a matrix A. Store L and U in a combined matrix.

Notes on partial pivoting: Gaussian elimination should update and return the permutation vector p. Then you have two options: either modify the forward solver to solve $Ly = Pb$ or just compute Pb directly (less ideal but valid).

c) Now complete the `linsolve` function. Optional: write the backward solve $Ux = y$ 'in-place', so that you don't need to create separate lists to store x and y.

d) Write a test that uses (c) to solve a system with a 4×4 matrix. Have it then check the size of the residual to verify the solution makes sense (optionally, you could also check the error by computing the exact solution, but not required here). You can use any 'typical' matrix A (that doesn't have any special properties like being the identity matrix).

Q2 (A Matrix class). In this problem you will implement a Matrix class (from scratch), similar to numpy's (2d) `array`. The point is to illustrate how you would use classes to give mathematical structure to your objects. I've marked some items as (E). For full credit, do at least one; for extra credit, do more.

Read the requirements carefully; you are expected to implement the class as requested *or* if you really want, to do something you think is better (include a brief comment).

Your Matrix class, representing an $m \times n$ matrix of real numbers, should have the following:

- **Member variables:** a variable for the matrix data; you may also want the number of rows/cols to be variables (up to you)
- **Initialization:** The constructor can accept a list of rows as an input.

  If the constructor is passed a tuple (m, n), it creates an $m \times n$ zero matrix.
- **printing:** Calling `print(a)` will print the matrix (E: format it nicely so it looks like an actual matrix!)
- **Operations:** (all done with operator overloading)

  - `a[(j,k)]` returns the (j, k) entry of A (the input here is a tuple (j, k))[1]

  - `a[(j,k)]=v` sets the (j, k) entry of A to v (use `__setitem__`).

  - `a+b` and `a-b` do what you expect (and returns a new matrix)

  - `a==b` if and only if all their elements are equal

  - (E) `a+=b` also does what you expect
- **Methods:**

  - `a.size()` that returns a tuple with the dimensions (rows/cols)

  - `a.deepcopy()` returns a new Matrix with the same values as A, but totally distinct from A (so no shared references)

  - (E) `a.dot(x)` computes Ax (and returns the result)

Testing: Write a function that illustrates your code works (it should be simple like the array example - just enough to show that the implementation is correct).

[1] Note that `a[j,k]` then also works since `j,k` creates a tuple. You need the input to be a tuple since `__getitem__(self,k)` must be written with one input 'key' `k`.