PHYS 331 - Numerical Techniques for the Sciences I.

Homework 5: Matrix Multiplication and Triangular Systems

Posted October 2, 2023.
Due October 18, 2023.

Problem 1 – Matrix-Vector Multiplication (15 points)

(a) Use the template `problem1.ipynb` to develop your own matrix-vector multiplication function. Make sure it checks for the size of the input matrix and vector and returns an appropriate error when the multiplication is not possible.

The function that you should implement is `multiplyMatrixVector(M, v)`, where `M` is the matrix in the product represented as a two-dimensional `np.array` object, and `v` is the vector represented as a one-dimensional `np.array`. Note that you can construct a 2D NumPy array by passing in a 2D Python list; for instance, to construct the matrix **A** shown in (b) below, you may write

```
M = np.array([[7, 0, 2],[-5, 1, 3],[3, 0, 1]])
```

and access the element (i,j) using, for instance,

```
M[i, j]  # Provided i and j are assigned values prior
         # to this line, of course.
```

Note that for a NumPy array `M`, `M.size` will return the *total number of elements* in the N-dimensional array. The field `M.shape` will, however, provide the dimensions of the array as a tuple; this may be needed in your implementation for this problem, depending on how you choose to write it.

Do *not* use `numpy.matrix` multiplication methods, or any other built-in solution - the point of this exercise is to define matrix-vector multiplication operations yourself!

(b) Use the function `main` in the template `problem1.ipynb` to test your answer by carrying out the multiplication $\mathbf{A}\vec{v}$ where

$$\mathbf{A} = \begin{pmatrix} 7 & 0 & 2 \\ -5 & 1 & 3 \\ 3 & 0 & 1 \end{pmatrix}, \quad \vec{v} = \begin{pmatrix} 1 \\ 5 \\ 7 \end{pmatrix} \tag{1}$$

and comparing with the result you obtain by doing the multiplication by hand. Include a second test case of your own. *Print out the results from both tests in your notebook when submitting your solution.*

Problem 2 – Triangular System Solver (15 points):

(a) Use the template `problem2.ipynb` to develop a function `triSolve(M, b, upperOrLower)` that solves triangular systems of equations, e.g. $\mathbf{M}\vec{x} = \vec{b}$ (where the matrix $\mathbf{M}$ is upper or lower triangular), via a backsubstitution method. The input variable `upperOrLower` should inform the function whether the input system is upper or lower triangular, and the function may otherwise ignore components in the lower or upper triangular sector of the matrix, respectively; implement your function such that `upperOrLower = 0` corresponds to an upper triangular, and `upperOrLower = 1` corresponds to a lower triangular. Have your function return the solution column vector $\vec{x}$ as a 1D NumPy array.

(b) Use the function `main` in the template `problem2.ipynb` to check your answer by solving $\mathbf{M}\vec{x} = \vec{b}_1$ with

$$\mathbf{M} = \begin{pmatrix} 9 & 0 & 0 \\ -4 & 2 & 0 \\ 1 & 0 & 5 \end{pmatrix}, \quad \vec{b}_1 = \begin{pmatrix} 8 \\ 1 \\ 4 \end{pmatrix} \tag{2}$$

(c) Add further code to the function `main` in template `problem2.ipynb` that defines a random triangular matrix of size 100×100 (using uniform or gaussian distributions), solves for a random right-hand side, and checks the result.

Problem 3 – Triangular Matrix Inverter (10 points):

(a) Use the routine of the previous exercise to build a matrix inverter for triangular matrices, which should be encapsulated in the function `triangleInverse(M)`. Your solution should return the solution $\mathbf{M}^{-1}$ as a 2D NumPy array.

(b) Check your answer to the previous item by solving linear systems with the same matrix $\mathbf{M}$ as in problem 2, but with new right-hand sides defined below. For this exercise you must use your result the inverse matrix you obtained in (a) in combination with your matrix-vector multiplication routine of Problem 1. Use `problem3.ipny` as a template.

$$\vec{b}_2 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \quad \vec{b}_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \vec{b}_4 = \begin{pmatrix} 7 \\ 2 \\ 8 \end{pmatrix} \tag{3}$$