CS 111: Homework 2: Due by 11:59pm Monday, January 20

Submit your homework online as a PDF file to GradeScope, and tell GradeScope which page(s) contain each problem.

1. Consider the code for the temperature problem in the lecture file `temperature.py`, especially the routines `make_A()` and `make_b()` that create the matrix A and right-hand side b. Experiment with different ways of setting the boundary conditions, which are the parameters `top`, `bottom`, `left`, and `right` to `make_b()`. Make a plot of the most interesting result that you get (in your opinion), and explain how you got it. If you want, you can also experiment with `matplotlib` to make a more interesting plot of your result. (The CS 111 logo on the course web page was obtained this way in 2010; maybe we can get a new logo this year!)

2. Again consider the routines `make_A()` and `make_b()` that create the matrix A and right-hand side b for the temperature problem. Let $k = 100$.

2a. How many elements are there in b?

2b. Considering all possible choices for the temperatures on the boundary, what is the largest number of elements of b that could possibly be nonzero?

2c. Explain why the rest of the elements of b are zero, no matter what the boundary temperatures are.

3. Write the following matrix in the form $A = LU$, where L is a unit lower triangular matrix (that is, a lower triangular matrix with ones on the diagonal) and U is an upper triangular matrix.

$$A = \begin{pmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{pmatrix}$$

4. The following three statements are all **false**. For each one, give a counterexample consisting of a 3-by-3 matrix or matrices, and show the computation that proves that the statement fails.

4a. If P is a permutation matrix and A is any matrix, then $PA = AP$.

4b. If matrix A is nonsingular, then it has a factorization $A = LU$ where L is lower triangular and U is upper triangular.

4c. The product of two symmetric matrices is a symmetric matrix.

5a. Consider the permutation matrix

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

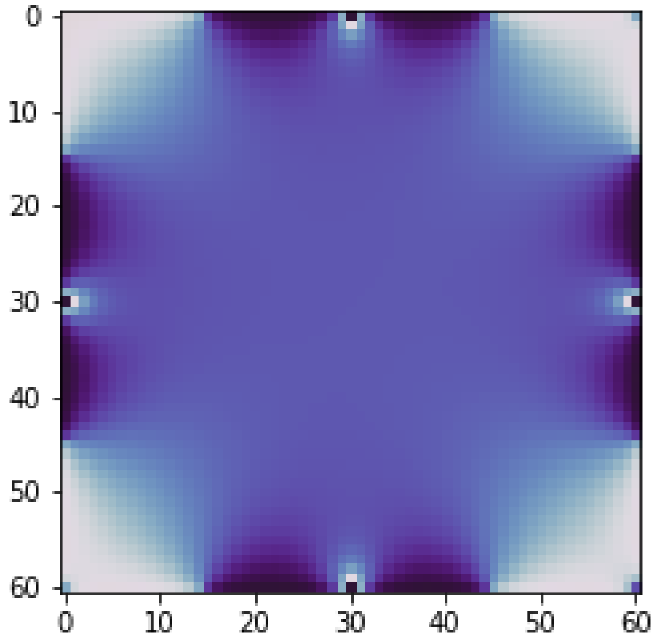
Find a 4-element permutation vector `v = np.array(something)` such that, for *every* 4-by-4 matrix A, we have `A[v,:] == P @ A`. Test your answer by running a few lines of Python, and turn in the result.

5b. For the same P, find a 4-element permutation vector `w = np.array(something)` such that, for *every* 4-by-4 matrix A, we have `A[:,w] == A @ P`. Test your answer and turn in the result.

6. Write `Usolve()`, analogous to `Lsolve()` in the lecture file `LU.py`, to solve an upper triangular system $Ux = y$. Warning: Notice that, unlike in `Lsolve()`, the diagonal elements of U don't have to be equal to one. Test your answer, both by itself and with `LUsolve()`, and turn in the result. Hint: Loops can be run backward in Python, say from $n-1$ down to 0, by writing

```
for i in reversed(range(n)):
```

1. The graph is obtained by setting the quarter portion of the end of each side to 100 degree and the midpoint of each side to 500 degree. The graph is plotted with a cyclic color mapping, which is why some of the hottest points of the map is black like they are cold.



2. **a.** 10,000
   b. 396
   c. Because the equation dictates that the right hand side is zero as the sum of the pixel and the negative average of the pixels around it should be equal

3.

$$U = \begin{pmatrix} 4 & -1 & -1 \\ 0 & \frac{15}{4} & -\frac{5}{4} \\ 0 & 0 & \frac{10}{3} \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{4} & 1 & 0 \\ -\frac{1}{4} & -\frac{1}{3} & 1 \end{pmatrix}$$

4. **a.**

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$PA = \begin{pmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix} \neq AP = \begin{pmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{pmatrix}$$

   b.

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 2 & 0 \end{pmatrix}$$

c.

$$\begin{pmatrix} 1 & 2 & 7 \\ 2 & 5 & 6 \\ 7 & 6 & 9 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 7 & 2 & 1 \\ 6 & 5 & 2 \\ 9 & 6 & 7 \end{pmatrix}$$

5. **a.**

$$\begin{pmatrix} 1 & 3 & 2 & 0 \end{pmatrix}$$

b.

$$\begin{pmatrix} 3 & 0 & 2 & 1 \end{pmatrix}$$

6.

Listing 1: Usolve

```
def Usolve(U, b):
"""Backward solve a unit upper triangular system Uy = b for y
Parameters:
U: the matrix, must be square, upper triangular
b: the right-hand side vector
Output:
y: the solution vector to U @ y == b
"""

# Check the input
m, n = U.shape
assert m == n, "matrix U must be square"
assert np.all(np.triu(U) == U), "matrix U must be upper triangular"

# Make a copy of b that we will transform into the solution
y = b.astype(np.float64).copy()

# Backwards solve
for col in reversed(range(n)):
y[col] = y[col] / U[col, col]
y[0:col] -= y[col] * U[0:col:, col]

return y
```