CS 111: Homework 2: Due by 11:59 pm Sunday, October 10, 2021

Submit your paper as one PDF file, and tell GradeScope which page(s) each problem is on. If you worked with a partner, you must each turn in your own homework paper, and report the name and perm number of your partner. No groups of more than two allowed.

1. Consider the code for the temperature problem in the file `cs111/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 $k = 100$.

2.1. How many elements are there in b?

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

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

3. Consider the permutation matrix

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

3.1. Find a 4-permutation `v = [ something ]` such that `A[v,:]  == P @ A` holds for *every* 4-by-4 matrix A. Test your answer by running a few lines of Python, and turn in the result.

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

4. Write `Usolve()`, analogous to `Lsolve()` in the file `cs111/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)):
```