

Last updated: Sunday 13<sup>th</sup> February, 2022 at 17:07.

## Programming assignment #2: electrostatics on a lattice

Let  $N > 0$  be a positive integer, and consider a uniform grid of points:

$$\mathbf{x}_{i,j} = (i, j) \in \mathbb{R}^2, \quad 0 \leq i \leq N, \quad 0 \leq j \leq N. \quad (1)$$

We will consider an equilibrium electrostatics problem on this grid, thinking of it as a lattice of nodes, with each node being connected to its four nearest neighbors in the cardinal directions (north, south, east, and west). Let  $\mathbf{u} \in \mathbb{R}^{(N+1)^2}$  be a vector which contains the electric potentials at each grid node, assuming that the nodes are inserted row-by-row, from top to bottom and left to right so that:

$$\mathbf{u}_k = u(\mathbf{x}_{i,j}), \quad k = (N+1)i + j, \quad 0 \leq k < (N+1)^2. \quad (2)$$

If  $\mathbf{u}_k$  and  $\mathbf{u}_l$  give the potential for two connected grid points, the flux through the edge that connects them is  $\pm(\mathbf{u}_k - \mathbf{u}_l)$ . We require the fluxes to balance. That is, for each  $k$  such that  $0 \leq k < (N+1)^2$ , we require:

$$\sum_{l \sim k} (\mathbf{u}_k - \mathbf{u}_l) = 4\mathbf{u}_k - \sum_{l \sim k} \mathbf{u}_l = 0, \quad (3)$$

where “ $l \sim k$ ” means that the grid points indexed by the  $l$  and  $k$  are neighbors.

Next, we assume that the electric potential  $u$  is held fixed and is equal to zero on the boundary nodes of the grid:

$$u(\mathbf{x}_{i,j}) = 0 \quad \text{if} \quad i = 0, N \quad \text{or} \quad j = 0, N. \quad (4)$$

If we let the remaining values of  $u$  be variables, then we are left with a matrix equation of the form:

$$\mathbf{A}\mathbf{u} = \mathbf{0}, \quad (5)$$

where  $\mathbf{A} \in \mathbb{R}^{(N-1)^2 \times (N-1)^2}$  and  $\mathbf{u} \in \mathbb{R}^{(N-1)^2}$ .

**Problem 1.** Compute  $\mathbf{A}$  for  $N = 100$  and use `matplotlib`’s `imshow` command to make a plot of its entries. Be sure to include a `colorbar` and choose an appropriate colormap so that is easy to visualize. In particular, make sure that the zero entries of  $\mathbf{A}$  are colored in white. *Hint: note very carefully the size of  $\mathbf{A}$  and the consequence of assuming that the boundary values of  $u$  equal zero. Work from (3).*

**Problem 2.** Write a function with the signature:

$$L, U, P = \text{lu}(\mathbf{A})$$

which computes the LU decomposition of a (possibly non-symmetric!) matrix  $A$  using partial pivoting, and so that afterwards  $PA = LU$  holds. *Hint: test this on some small matrices and compare the result with `np.linalg.lu` as you go.*

**Problem 3.** Using `lu`, compute the LU decomposition of  $\mathbf{A}$  for  $N = 10, 30, 100, 300$ , and 1000. Plot  $\mathbf{L}$  in the same way you plotted  $\mathbf{A}$  in Problem 1. Count the number of nonzeros of the  $L$  factor, and find its *lower bandwidth* (the number of diagonals of the matrix that contain nonzero values). Make two plots of the number of nonzeros of  $\mathbf{L}$  and the lower bandwidth of  $\mathbf{L}$ , each with  $N$  on the horizontal axis.

**Problem 4.** Write two functions:

$$\mathbf{x} = \text{fsolve}(\mathbf{L}, \mathbf{b}) \quad \mathbf{x} = \text{bsolve}(\mathbf{U}, \mathbf{b})$$

which do forward substitution (solve a linear system  $\mathbf{L}\mathbf{x} = \mathbf{b}$  where  $L$  is lower-triangular) and backwards substitution (solve a linear system  $\mathbf{U}\mathbf{x} = \mathbf{b}$  where  $U$  is upper-triangular), respectively. For  $N = 100$ , use these functions and your function `lu` to solve:

$$\mathbf{A}\boldsymbol{\phi}_{i,j} = \mathbf{e}_{i,j}, \tag{6}$$

where  $\mathbf{e}_{i,j}$  is the  $(k, l)$ th standard basis vector—i.e., it has a 1 in the position corresponding to  $\mathbf{x}_{i,j}$ , and 0s everywhere else. Make a 3D plot of  $\boldsymbol{\phi}_{i,j}$  as the graph of a function using `matplotlib3d` for a few different choices of  $(i, j)$  after rearranging the entries of  $\boldsymbol{\phi}$  to lie on a square grid (so that they match the 2D layout of the grid nodes  $\mathbf{x}_{i,j}$ ).