Last updated: Sunday 13th February, 2022 at 13:10.

Programming assignment #2: electrostatics on a lattice

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

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

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 $\boldsymbol{u} \in \mathbb{R}^{(N+1)^2}$ 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:

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

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

$$\sum_{l \sim k} (\boldsymbol{u}_k - \boldsymbol{u}_l) = d_k \boldsymbol{u}_k - \sum_{l \sim k} \boldsymbol{u}_k = 0, \tag{3}$$

where "$l \sim k$" means that the grid points indexed by the l and k are neighbors, and where d_k is the "degree" of node k (the numbers of neighbors it has). To make this condition easier to work with, we will "stack" (3) for each k into the rows of a matrix $\boldsymbol{A}$. The entries of $\boldsymbol{A}$ are given by:

$$\boldsymbol{A}_{k,l} = \begin{cases} d_k & \text{if } k = l, \\ -1 & \text{if } k \sim l, \\ 0 & \text{otherwise .} \end{cases} \tag{4}$$

So that (3) can be rewritten as the matrix equation:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{0}. \tag{5}$$

Note that $\boldsymbol{u} = \boldsymbol{0}$ trivially satisfies (5). It is possible to find more interesting solutions by searching for eigenpairs $\boldsymbol{A}\boldsymbol{u} = \lambda \boldsymbol{u}$, where $\lambda \neq 0$.

Problem 1. Compute $\boldsymbol{A}$ 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 $\boldsymbol{A}$ are colored in white.

Problem 2. Write a function with the signature:

```
L, U, P = lu(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 $\boldsymbol{A}$ for $N = 10, 30, 100, 300$, and 1000. Plot $\boldsymbol{L}$ in the same way you plotted $\boldsymbol{A}$ 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 $\boldsymbol{L}$ and the lower bandwidth of $\boldsymbol{L}$, each with N on the horizontal axis.

Problem 4. Write two functions:

```
x = fsolve(L, b)     x = bsolve(U, b)
```

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

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

where $\boldsymbol{e}_{i,j}$ is the $(k, l)$the standard basis vector—i.e., it has a 1 in the position corresponding to $\boldsymbol{x}_{i,j}$, and 0s everywhere else. Make plots of $\boldsymbol{\phi}_{i,j}$ using `imshow` 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 $\boldsymbol{x}_{i,j}$).