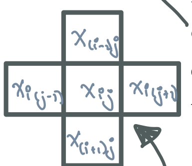
Last updated: Sunday 13th February, 2022 at 20:16.

Programming assignment #2: electrostatics on a lattice

Picture to have in mind:

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

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



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

$$u = \begin{bmatrix} x_{00} \\ x_{01} \\ x_{02} \\ \vdots \\ x_{10} \\ x_{11} \\ x_{12} \\ \vdots \\ x_{N0} \\ \vdots \\ x_{NN} \end{bmatrix}$$

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

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 $0 \le k < (N+1)^2$, we require:

what you want is:

$$0 = (x_{ij} - x_{(i-1)j}) + (x_{ij} - x_{i(j-1)}) + (x_{ij} - x_{i(j+1)}) + (x_{ij} - x_{(i+1)j})$$

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

Think: how can you identify these indices ($(i-1)j$, $i(j-1)$, $i(j+1)$, and $(i+1)j$) in u?

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 equal to zero on the boundary nodes of the grid:

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

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

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

tip: since we matched u with a block vector then the matrix A can be seen as a block matrix.

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

Tip: the u vector can be seen as a block vector. If $x^{(i)}$ is the i-th row of that initial grid then u is

$$u = \begin{bmatrix} x^{(1)T} \\ x^{(2)T} \\ \vdots \\ x^{(3)T} \end{bmatrix}$$

Problem 1. Compute $\boldsymbol{A}$ for $N = 10$ 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. *Hint: note very carefully the size of $\boldsymbol{A}$ 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 = 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.*

for example:

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad M_2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 0 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

has lower bandwidth=1   has lower bandwidth=3

Problem 3. Using `lu`, compute the LU decomposition of $\boldsymbol{A}$ for $N = 10, 20, 30, 40$, and 50. 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.

hint: in recitation you solver systems that looked like

$Lw = b$, $Uv = c$ $\}$ follow the steps you did manually

Problem 4. Write two functions:

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

which do forward substitution (solve a linear system $\boldsymbol{Lx} = \boldsymbol{b}$ where L is lower-triangular) and backwards substitution (solve a linear system $\boldsymbol{Ux} = \boldsymbol{b}$ where U is upper-triangular), respectively. For $N = 50$, use these functions and your function `lu` to solve:

this is also a block vector

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

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 a 3D plot of $\boldsymbol{\phi}_{i,j}$ as the graph of a function using `mplot3d` 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 $\boldsymbol{x}_{i,j}$).

Recall that we made the v out of the rows from our original grid. So we can see this ϕ as a block vector again:

$$\phi = \begin{bmatrix} \phi^{(1)} \\ \vdots \\ \phi^{(N)} \end{bmatrix}$$

Original grid:

$$\begin{bmatrix} x_{00} & x_{01} & \cdots & x_{0N} \\ x_{10} & x_{11} & \cdots & x_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N0} & x_{N1} & \cdots & x_{NN} \end{bmatrix}$$

Those blocks correspond to rows of a grid, that's how you rearrange them.