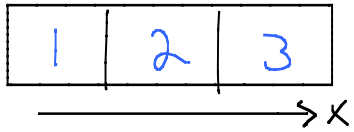
Laplacian matrix in 2 dimensions

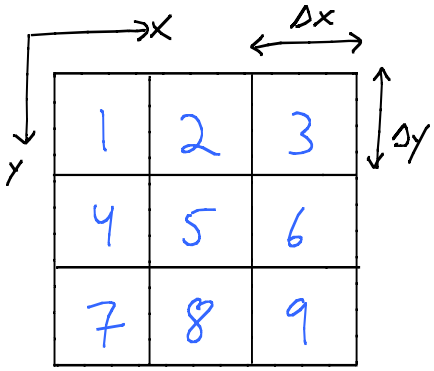
In 1d, for $N_x = 3$:



$$\frac{d^2}{dx^2} \longrightarrow L^1_{N_x=3} = \frac{1}{\Delta x^2} \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix} \end{array}$$

(Note tridiagonal symmetry)

Now in 2d, $N_x = 3$ and $N_y = 3$:



Assuming $\Delta x = \Delta y = 1$, we have:

$$\frac{d^2}{dx^2} + \frac{d^2}{dy^2} \longrightarrow L^2_{\substack{N_x=3 \\ N_y=3}} =$$

	1	2	3	4	5	6	7	8	9
1	-2	1	0	1	0	0	0	0	0
2	1	-2	1	0	1	0	0	0	0
3	0	1	-2	0	0	1	0	0	0
4	1	0	0	-2	1	0	1	0	0
5	0	1	0	1	-2	1	0	1	0
6	0	0	1	0	1	-2	0	0	1
7	0	0	0	1	0	0	-2	1	0
8	0	0	0	0	1	0	1	-2	1
9	0	0	0	0	0	1	0	1	-2

Note block tridiagonal symmetry!

Kronecker product

Notice that we can decompose $L^2_{N_x,N_y}$ into expressions involving 1-d Laplacian matrix:

$$\begin{bmatrix} 1\cdot L^1_{N_x} & 0 & 0 \\ 0 & 1\cdot L^1_{N_x} & 0 \\ 0 & 0 & 1\cdot L^1_{N_x} \end{bmatrix} + \begin{bmatrix} -2\cdot I_{N_x} & 1\cdot I_{N_x} & 0 \\ 1\cdot I_{N_x} & -2\cdot I_{N_x} & 1\cdot I_{N_x} \\ 0 & 1\cdot I_{N_x} & -2\cdot I_{N_x} \end{bmatrix}$$

(# of blocks along diagonal equal to N_y)

Using the "Kronecker" product: $\begin{bmatrix} A & B \\ C & D \end{bmatrix} \otimes \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} A\begin{bmatrix} E & F \\ G & H \end{bmatrix} & B\begin{bmatrix} E & F \\ G & H \end{bmatrix} \\ C\begin{bmatrix} E & F \\ G & H \end{bmatrix} & D\begin{bmatrix} E & F \\ G & H \end{bmatrix} \end{bmatrix} = \begin{bmatrix} AE & AF & BE & BF \\ AG & AH & \cdots & \\ CE & & \ddots & \\ CG & & & \end{bmatrix}$

This becomes $L^2_{N_x,N_y} = I_{N_y} \otimes L^1_{N_x} + L^1_{N_y} \otimes I_{N_x}$

(I_{N_x} is identity matrix w/ N_x rows and columns)

Matlab implementation: full vs sparse

```
clear

Nx=80;
Ny=80;

dx=1; dy=1; %grid spacing

%full matrices:
L1x=1/dx^2*diag(-2*ones(Nx,1))+diag(ones(Nx-1,1),1)+diag(ones(Nx-1,1),-1);
L1y=1/dy^2*diag(-2*ones(Ny,1))+diag(ones(Ny-1,1),1)+diag(ones(Ny-1,1),-1);
L2=kron(eye(Ny),L1x)+kron(L1y,eye(Nx));

%sparse matrices:
sL1x=1/dx^2*spdiags(-2*ones(Nx,1),0,sparse(Nx,Nx))+spdiags(ones(Nx,1),1,sparse(Nx,Nx))+spdiags(ones(Nx,1),-1,sparse(Nx,Nx));
sL1y=1/dy^2*spdiags(-2*ones(Ny,1),0,sparse(Ny,Ny))+spdiags(ones(Ny,1),1,sparse(Ny,Ny))+spdiags(ones(Ny,1),-1,sparse(Ny,Ny));
sL2=kron(speye(Ny),sL1x)+kron(sL1y,speye(Nx));

B=zeros(Nx,Ny);
B(round(Nx/2),round(Ny/2))=1;%boundary condition in center

tic
Y= L2\reshape(-B,Nx*Ny,1);
disp(['with full it takes ' num2str(toc) 's'])

tic
Y= sL2\reshape(-B,Nx*Ny,1);
disp(['with sparse it takes ' num2str(toc) 's'])

surf(reshape(Y,Nx,Ny))
```

When $N_x = N_y = 3$, this Kronecker expression for L^2 gives us exactly what we constructed by hand:

$$\begin{bmatrix}
-4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4
\end{bmatrix}$$

Takes the $N_x \times N_y$ matrix $-B$ and concatenates the columns to create a column vector

Takes the column vector Y and "folds up" the columns to create a $N_x \times N_y$ matrix

Output:

```
with full it takes 13.5819s
with sparse it takes 0.021787s
```