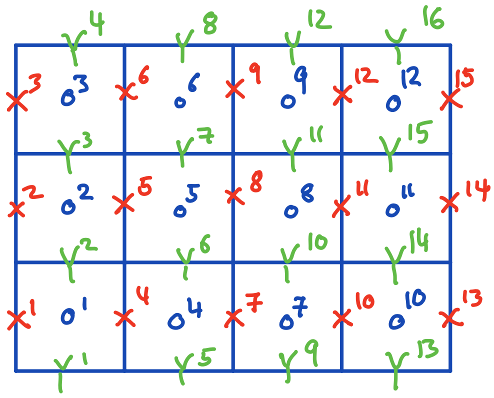
Discrete operators in 2D

Staggered grid in 2D



$N_x = 4$, $N_y = 3$, $\Rightarrow N = N_x N_y = 12$

faces in x-dir.: $N_{fx} = (N_x+1) N_y = 15$

faces in y-dir.: $N_{fy} = N_x (N_y+1) = 16$

Total faces : $N_f = N_{fx} + N_{fy} = 31$

Discrete gradient in 2D:

Continuous gradient: $\nabla h = \begin{pmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \end{pmatrix}$

approximate $\frac{\partial h}{\partial x} \sim \underline{dhx}$ on x-faces

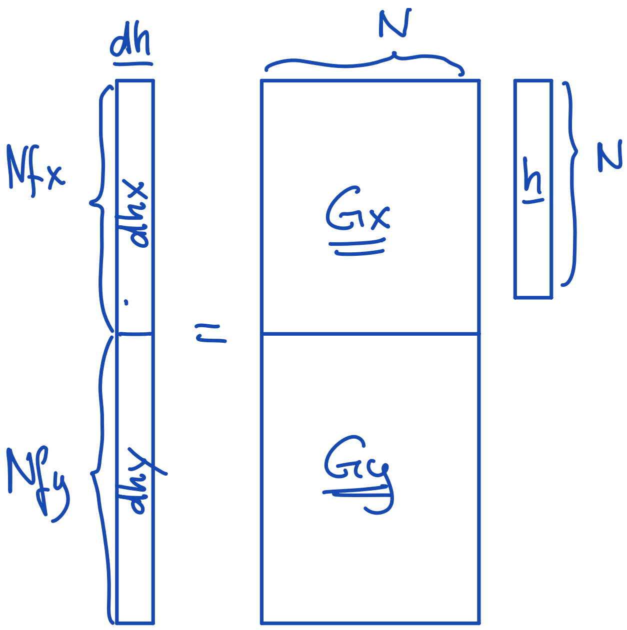
approximate $\frac{\partial h}{\partial y} \sim dh_y$ on y-faces

Choose to build $\underline{\underline{G}}$ such that the resulting gradient vector is ordered as $\underline{dh} = \begin{bmatrix} \underline{dhx} \\ \underline{dhy} \end{bmatrix}$

$\Rightarrow$ 2D Gradient can be decomposed as

$$\underline{\underline{G}} = \begin{bmatrix} \underline{\underline{Gx}} \\ \underline{\underline{Gy}} \end{bmatrix} \quad \text{where} \quad \underline{dhx} = \underline{\underline{Gx}}\ \underline{h}$$

$$\underline{dhy} = \underline{\underline{Gy}}\ \underline{h}$$



Matrix dimensions:

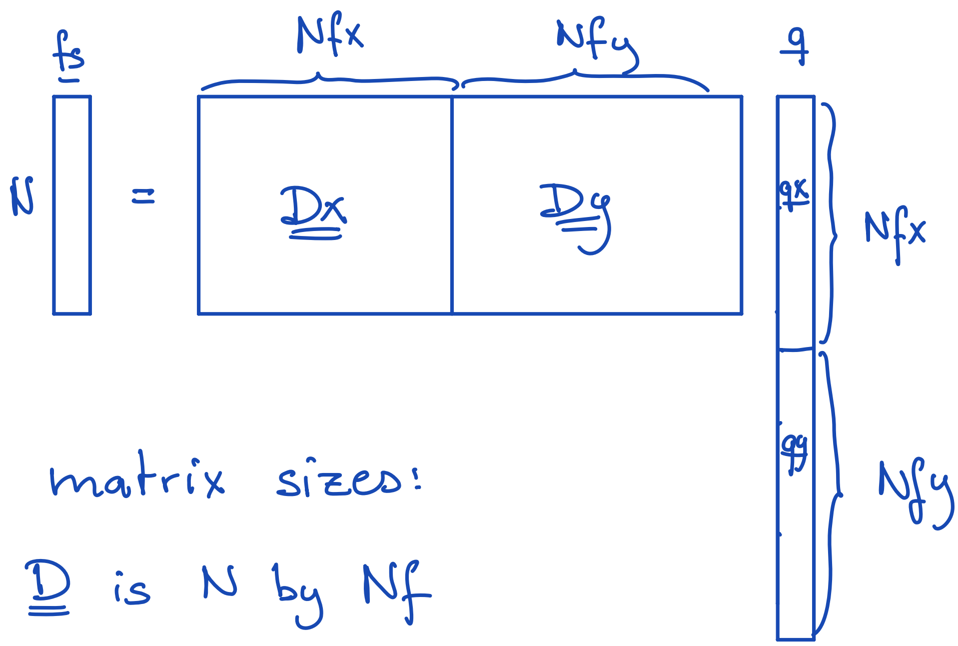
$\underline{\underline{G}}$ is $N_f \times N$

$\underline{\underline{G_x}}$ is $N_{fx} \times N$

$\underline{\underline{G_y}}$ is $N_{fy} \times N$

Discrete divergence in 2D

$$\nabla \cdot \underline{q} = \frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} \approx \underline{\underline{D}}\,\underline{q} = \underline{\underline{D_x}}\,\underline{q_x} + \underline{\underline{D_y}}\,\underline{q_y}$$



matrix sizes:

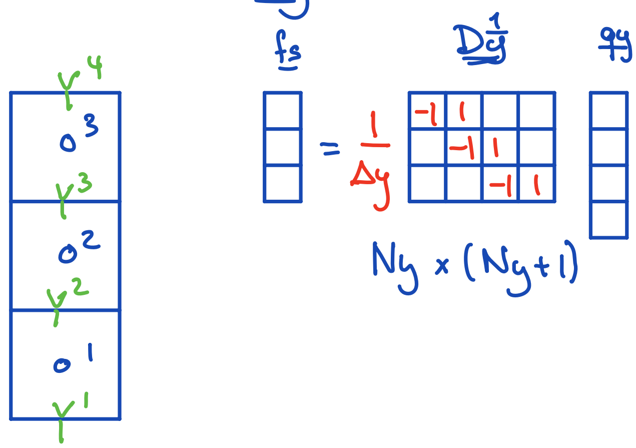
$\underline{\underline{D}}$ is N by N_f

$\underline{\underline{D_x}}$ is N by N_{fx}

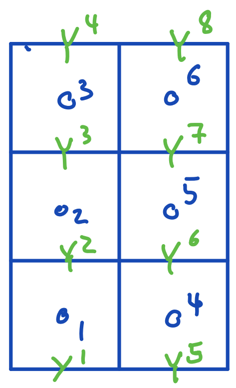
$\underline{\underline{D_y}}$ is N by N_{fy}

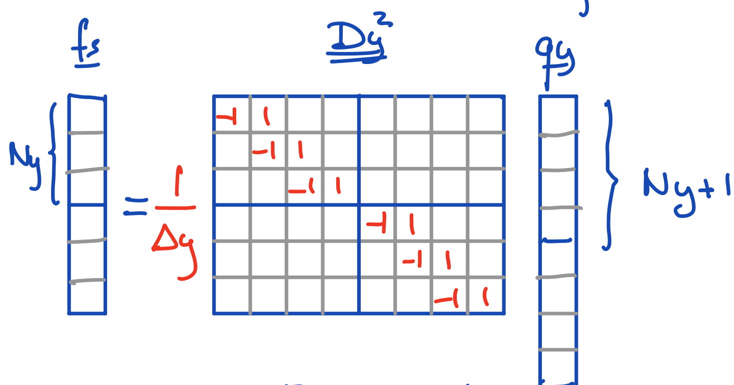
Building the 2D discrete divergence matrix

Start with D_y in 1D:



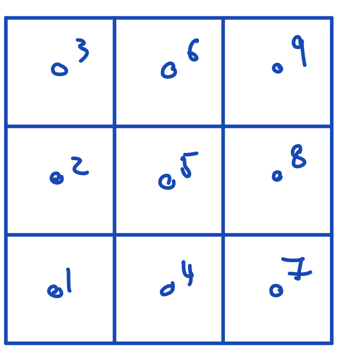
Suppose we add a second column of cells





$$D_y^2 = \begin{bmatrix} D_y^1 & \\ & D_y^1 \end{bmatrix}$$

2 by 2 block matrix with D_y^1 on diagonal



$$D_y^2 = \begin{bmatrix} D_y^1 & & \\ & D_y^1 & \\ & & D_y^1 \end{bmatrix}$$

In general:

$\underline{\underline{D_y^2}}$ is a block matrix with N_x by N_x blocks of size N_y by (N_y+1). Diagonal blocks are $\underline{\underline{D_y^1}}$ and all others are zero.

Tensor product construction of $\underline{\underline{D_y^2}}$

The discrete 2D operator can easily and efficiently be assembled using Konecker/tensor products.

Definition:

If $\underline{\underline{A}}$ is a $m \times n$ matrix and $\underline{\underline{B}}$ is a $p \times q$ matrix, then the Kronecker product $\underline{\underline{A}} \otimes \underline{\underline{B}}$ is the $mp \times nq$ block matrix:

$$\underline{\underline{A}} \otimes \underline{\underline{B}} = \begin{bmatrix} a_{11}\underline{\underline{B}} & \cdots & a_{1n}\underline{\underline{B}} \\ \vdots & \ddots & \vdots \\ a_{m1}\underline{\underline{B}} & \cdots & a_{mn}\underline{\underline{B}} \end{bmatrix}$$

Hence we can construct $\underline{\underline{D_y}}^2$ as

$$\boxed{\underline{\underline{D_y}}^2 = \underline{\underline{I_x}} \otimes \underline{\underline{D_y}}^1} = \begin{bmatrix} \underline{\underline{D_y}}^1 & & & & \\ & \underline{\underline{D_y}}^1 & & & \\ & & \underline{\underline{D_y}}^1 & & \\ & & & \ddots & \\ & & & & \underline{\underline{D_y}}^1 \end{bmatrix}$$

where $\underline{\underline{I_x}}$ is a N_x by N_x identity matrix.

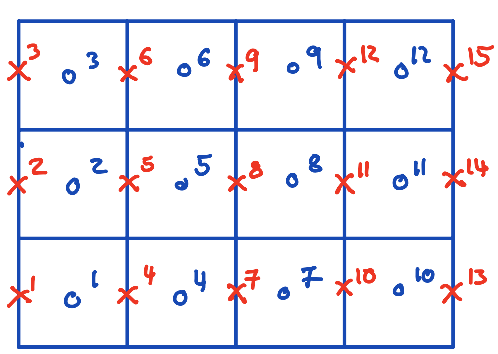
In Matlab the tensor product is obtained as

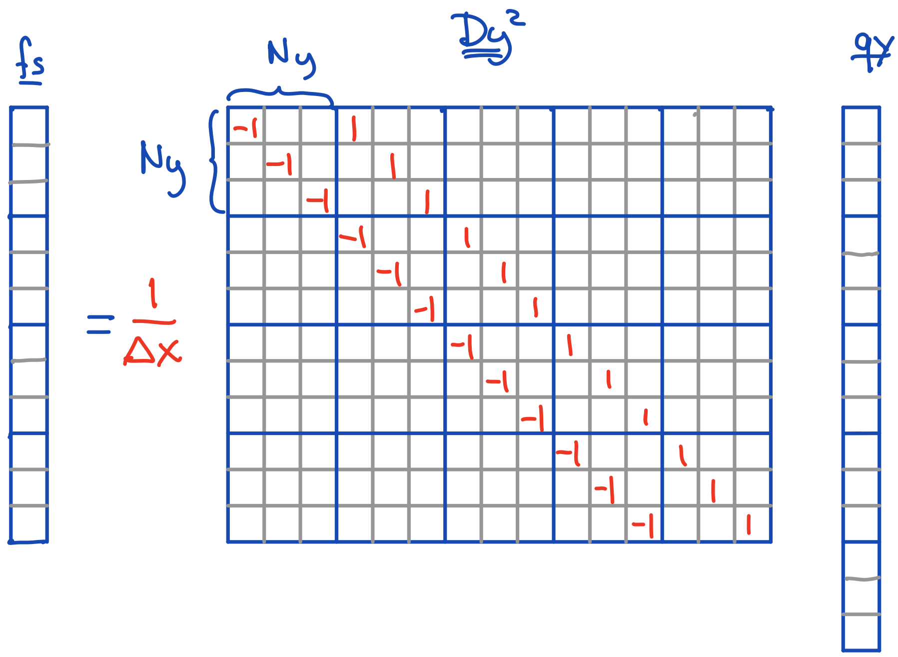
$$\underset{\substack{\uparrow \\ \text{2D op.}}}{\underline{\underline{D_y}}} = \text{kron}(\underline{\underline{I_x}}, \underset{\substack{\uparrow \\ \text{1D op}}}{\underline{\underline{D_y}}});$$

So how do we build $\underline{\underline{D_x}}^2$?

On a x-first grid $\underline{\underline{D_x}}^2 = \underline{\underline{I_y}} \otimes \underline{\underline{D_x}}^1$!

But what does $\underline{\underline{D_x}}^2$ look like on a y-first grid?





$\Rightarrow$ $\underline{\underline{Dx}}^2$ is a sparse diagonal matrix

(this could be assembled with spdiags)

$\underline{\underline{Dx}}^2$ is also a block matrix built from

Ny by Ny Identities matrices.

$$\underline{\underline{Dx}}^2 = \begin{bmatrix} -\underline{\underline{I_y}} & \underline{\underline{I_y}} & & & & \\ & -\underline{\underline{I_y}} & \underline{\underline{I_y}} & & & \\ & & -\underline{\underline{I_y}} & \underline{\underline{I_y}} & & \\ & & & -\underline{\underline{I_y}} & \underline{\underline{I_y}} & \\ & & & & -\underline{\underline{I_y}} & \underline{\underline{I_y}} \end{bmatrix} = \underline{\underline{Dx}}' \otimes \underline{\underline{I_y}}$$

In Matlab: $\underline{\underline{Dx}} = \mathrm{kron}(\underline{\underline{Dx}}, \underline{\underline{I_y}})$

Discrete gradient matrix

The $\underline{\underline{G_x}}$ and $\underline{\underline{G_y}}$ matrices could be built using 1D matrices and Kronecker products. Instead, we use the fact that the $\underline{\underline{D}}$ and $\underline{\underline{G}}$ matrices are adjoints:

$$\boxed{\underline{\underline{G}} = -\underline{\underline{D}}^T} \qquad \text{true in interior}$$

Need to impose natural BC's. $\Rightarrow$ set $\underline{\underline{G}} = 0$ on all boundary faces.

Make vector containing all bnd faces:

```
dof_f_bnd = [dof_f_xmin; dof_f_xmax; ...
             dof_f_ymin; dof_f_ymax];
```

Zero out corresponding rows in $\underline{\underline{G}}$:

```
G(dof_f_bnd,:) = 0;
```

2D mean operator

$\underline{\underline{M}}$ has same structure as $\underline{\underline{G}}$

$$\underline{\underline{G}} = \begin{bmatrix} \underline{\underline{G_x}} \\ \underline{\underline{G_y}} \end{bmatrix} \quad \Rightarrow \quad \underline{\underline{M}} = \begin{bmatrix} \underline{\underline{M_x}} \\ \underline{\underline{M_y}} \end{bmatrix}$$

but it is better to assemble $\underline{\underline{M}}$ from 1D operators with Kronecker product. We have simply

$$\underbrace{\underline{\underline{M_x}}}_{2D} = \underbrace{\underline{\underline{M_x}}}_{1D} \otimes \underline{\underline{I_y}} \qquad \underbrace{\underline{\underline{M_y}}}_{2D} = \underline{\underline{I_x}} \otimes \underbrace{\underline{\underline{M_y}}}_{1D}$$

We won't deal with the Curl in this class