Shallow water model documentation

Milan Kloewer **version of March 31, 2017**

This documentation aims to summarize a methodology how the shallow water equations can be discretized and solved numerically with finite differences. The corresponding model code written in Python can be found at `www.github.com/milankl/swm`.

Contents

1 Physical model

1.1 General model

The shallow water equations of interest are (see Gill, 1982)

$$\partial_t u + u\partial_x u + v\partial_y u - fv = -g\partial_x \eta + F_x + M_x \tag{1a}$$
$$\partial_t v + u\partial_x v + v\partial_y v + fu = -g\partial_y \eta + F_y + M_y \tag{1b}$$
$$\partial_t \eta + \partial_x(uh) + \partial_y(vh) = 0 \tag{1c}$$

with

$\mathbf{u} = (u, v) = (u(x, y, t), v(x, y, t))$	horizontal velocity vector
$\eta = \eta(x, y, t)$	surface displacement
$h = h(x, y, t) = \eta + H$	layer thickness
$H = H(x, y)$	undisturbed layer thickness
$f = f(y)$	coriolis parameter
$g = \text{const}$	gravitational acceleration
$\mathbf{F} = (F_x, F_y) = (F_x(x, y, t), F_y(x, y, t))$	forcing vector
$\mathbf{M} = (M_x, M_y) = (M_x(u, v, h), M_y(u, v, h))$	lateral mixing of momentum, and friction term

and differential operators

$$\partial_x = \frac{\partial}{\partial x}, \partial_y = \frac{\partial}{\partial y}, \partial_t = \frac{\partial}{\partial t}$$

on the domain $\mathcal{D} = (0, L_x) \times (0, L_y)$ of width (or east-west extent) L_x and north-south extent L_y and with cartesian coordinates x, y and time t. The initial conditions are

$$u(t = 0) = u_0(x, y) \tag{2a}$$
$$v(t = 0) = v_0(x, y) \tag{2b}$$
$$h(t = 0) = h_0(x, y) \tag{2c}$$

The kinematic boundary condition states that there is no flow through the boundary, i.e.

$$u(x = 0) = u(x = L_x) = 0 \tag{3a}$$
$$v(y = 0) = v(y = L_y) = 0, \tag{3b}$$

additionally we require no gradient of η across the boundary, hence

$$\partial_x \eta(x = 0) = \partial_x \eta(x = L_x) = 0 \tag{4a}$$
$$\partial_y \eta(y = 0) = \partial_y \eta(y = L_y) = 0, \tag{4b}$$

and the tangential velocity at the boundary should vanish, which is usually referred to as no-slip boundary conditions

$$v(x = 0) = v(x = L_x) = 0 \tag{5a}$$
$$u(y = 0) = u(y = L_y) = 0. \tag{5b}$$

However, sometimes these equations are replaced by

$$\partial_x v(x=0) = \partial_x v(x=L_x) = 0 \tag{6a}$$
$$\partial_y u(y=0) = \partial_y u(y=L_y) = 0, \tag{6b}$$

which corresponds physically to a friction-less boundary and are therefore called *free-slip* boundary conditions.

The prognostic variables so far are u, v, η, however, as η only appears in gradients, also h can be used as prognostic variable. We can separate from the advective terms in equations 1a,b the spatial gradient of kinetic energy. In combination with the pressure gradient term we introduce the Bernoulli potential p as

$$p = \frac{1}{2}(u^2 + v^2) + gh \tag{7}$$

Furthermore, with introducing the relative vorticity $\zeta = \partial_x v - \partial_y u$ the potential vorticity can be defined as

$$q = \frac{f + \zeta}{h} \tag{8}$$

and the equations 1a,b,c then become

$$\partial_t u = qhv - \partial_x p + F_x + M_x \tag{9a}$$
$$\partial_t v = -qhu - \partial_y p + F_y + M_y \tag{9b}$$
$$\partial_t h = -\partial_x(uh) - \partial_y(vh) \tag{9c}$$

which are the equations solved by the numerical model as described in section 2.

1.2 Double gyre set up

In order to simulate mid-latitudinal dynamics we choose the physical parameters of the previous section as (Cooper and Zanna, 2015; Porta Mana and Zanna, 2014; Berloff, 2005)

$$g = 10 \text{ m/s} \tag{10a}$$
$$H = 500 \text{ m} \tag{10b}$$
$$L_x = L_y = 3840 \text{ km} \tag{10c}$$

with beta-plane approximation

$$f = f_0 + \beta(y - \frac{L_y}{2}), \quad f_0 = 2\Omega \sin(2\pi \frac{\theta_0}{360°}), \quad \beta = \frac{2\Omega}{R}\cos(2\pi\frac{\theta_0}{360°}) \tag{11}$$

at Northern hemisphere mid-latitudes, with the domain $\mathcal{D}$ being centred around

$$\theta_0 = 30° \tag{12}$$

with

$$R = 6371 \text{ km}, \quad \Omega = \frac{2\pi}{86400} \text{ s}^{-1} \tag{13}$$

The forcing is set to be

$$
\begin{aligned}
F_y &= 0 \\
F_x &= \frac{F_0}{\rho_0 H}\left[\cos\left(2\pi\left(\frac{y}{L_y}-\frac{1}{2}\right)\right)+2\sin\left(2\pi\left(\frac{y}{L_y}-\frac{1}{2}\right)\right)\right] \\
F_0 &= 0.12 \text{ Pa} \\
\rho_0 &= 1000 \text{ kgm}^{-3}
\end{aligned}
$$

and start from rest, so that the initial conditions become

$$u_0 = v_0 = 0, \quad h_0 = H \tag{14}$$

1.3 Lateral mixing of momentum and friction term

In the previous section the term

$$\mathbf{M} = (M_x, M_y) = (M_x(u,v,h), M_y(u,v,h)) \tag{15}$$

was kept general. The term is desired to represent some physical kind of friction, diffusion, lateral mixing of momentum, dissipation and/or viscosity. Once the forcing is chosen to be $\mathbf{F} \neq 0$ there is generally an energy source in the shallow water model, hence, there is some additional requirement to $\mathbf{M}$ to be an energy sink to allow for some equilibrium state the shallow water model can reach. One simple approach is to set

$$\mathbf{M} = -r\mathbf{u} \tag{16}$$

in which case a linear drag, called Rayleigh friction (Gill, 1982), is applied. r^{-1} is a time scale at which momentum decays. This approach represents an energy sink and is in the barotropic shallow water model usually one way to think of bottom friction. Another more realistic approach to bottom friction is a quadratic drag (Arbic and Scott, 2008), i.e.

$$\mathbf{M} = -\frac{c_D}{H}|\mathbf{u}|\mathbf{u}, \quad |\mathbf{u}| = \sqrt{u^2+v^2} \tag{17}$$

with c_D being a dimensionless drag coefficient. The term $\mathbf{M}$ can also be used to represent the lateral mixing of momentum, the simplest approach of this family being

$$\mathbf{M} = \nu_A \nabla^2 \mathbf{u} \tag{18}$$

with $\nabla^2 = \partial_x^2 + \partial_y^2$ being the two dimensional Laplace operator and ν_A a viscosity coefficient of unit $\text{m}^2\, s^{-1}$. With the symmetric 2x2 stress tensor $\mathbf{S}$ defined by

$$\mathbf{S} = \begin{pmatrix} u_x - v_y & v_x + u_y \\ v_x + u_y & -(u_x - v_y) \end{pmatrix} \tag{19}$$

Shchepetkin and O'Brien, (1996) define a harmonic lateral mixing of momentum as

$$\mathbf{M} = \nu_A h^{-1} \nabla \cdot h\mathbf{S}. \tag{20}$$

where the scalar product between a vector $\mathbf{a} = a_i$ and a tensor $\mathbf{B} = B_{ij}$ is defined with Einstein notation as $\mathbf{a} \cdot \mathbf{B} = a_i B_{ij} = \sum_i a_i B_{ij} = c_j$ and yields a vector $c_j = \mathbf{c}$. Note that for $h = \text{const}$ equation 20 simplifies to equation (18). Equation (20) can be extend to a biharmonic operator by applying it twice

$$\mathbf{M} = \nu_B h^{-1}\nabla \cdot (h\mathbf{S}(h^{-1}\nabla \cdot h\mathbf{S}(u,v))). \tag{21}$$

where the stress tensor is regarded as a linear map, once evaluated with (u,v) and then with $h^{-1}\nabla \cdot h\mathbf{S}(u,v)$. Both viscosity coefficients ν_A and ν_B are for simplicity taken as constants. Again for $h = \text{const.}$ equation 21 reduces to $\nu_B \nabla^4 \mathbf{u}$, with $\nabla^4 = \partial_x^4 + \partial_y^4 + 2\partial_x^2\partial_y^2$. A biharmonic operator acts especially on the small scales, where waves are rapidly damped compared to the large scales, which remain mostly unaffected (CITE!). To have an additional energy sink at the large scales equation (17) is used in combination with equation (21) to have finally

$$\mathbf{M} = \nu_B h^{-1}\nabla \cdot (h\mathbf{S}(h^{-1}\nabla \cdot h\mathbf{S}(u,v))) - \frac{c_D}{H}|\mathbf{u}|\mathbf{u} \tag{22}$$

and is used in equation (9). The choice of ν_B and c_D is discussed in section 2.6.

2 Spatial discretization

2.1 Arakawa C-Grid

The domain $\mathcal{D}$ is divided into $n_x \times n_y$ grid cells, evenly spaced, so that each grid cell has the side length $\Delta x = \frac{L_x}{n_x}$ in x-direction and $\Delta y = \frac{L_y}{n_y}$ in y-direction. The total amount of grid cells is $n_x n_y$. Let the variable h_i sit in the middle of the i-th grid cell at position (x_i, y_i), i.e. $h_i = h(x_i, y_i)$. We use only one index i and with row-first indexing (see Fig. 1), so we have for $i \in \{1, ..., n_x n_y\}$

$$x_i = \frac{\Delta x}{2}(1 + (i-1) \bmod n_x) \tag{23a}$$

$$y_i = \frac{\Delta y}{2}\left(1 + 2\,\text{floor}\left(\frac{i-1}{n_x}\right)\right) \tag{23b}$$

with $a \bmod b$ being the modulo operator and floor(a) is the largest integer not greater than a. Following the ideas of the Arakawa C-grid (**Arakawa1977**; Arakawa and Lamb, 1981), the discretization of the variables u, v and q is staggered. In general, we might use an independent indexing for u, v and q and therefore introduce $j, k, l \in \mathbb{N}$

$$u_j = u(x_i + \frac{\Delta x}{2}, y_i) \tag{24a}$$

$$v_k = u(x_i, y_i + \frac{\Delta y}{2}) \tag{24b}$$

$$q_l = q(x_i + \frac{\Delta x}{2}, y_i + \frac{\Delta y}{2}). \tag{24c}$$

Hence, we distinguish between 4 different grids: (i) the T-grid, for h or tracers, (ii) the u-grid, (iii) the v-grid, (iv) the q-grid. Not for all grid cells it is necessary to evaluate u or v, as they might vanish due to the boundary conditions. The grids therefore carry a different amount of grid points. Let N_T, N_u, N_v, N_q be the total number of grid points on the respective grids then

$$\begin{aligned} N_T &= n_x n_y, & N_u &= (n_x - 1)n_y \\ N_q &= (n_x+1)(n_y+1), & N_v &= n_x(n_y-1) \end{aligned} \tag{25}$$

The n_x-th column of u-points vanish, as does the n_y-th row of v-points. However, there is no boundary condition for q, which makes it necessary to evaluate the q-grid for all points within the domain $\mathcal{D}$.

Choosing one index for the grid points leads to the advantage that every scalar variable can be represented as a vector. Numbering the grids row-first as can be seen in Fig. 1, leads to the following vector-representation $\mathtt{u}, \mathtt{v}, \mathtt{h}, \mathtt{q}$ of u, v, h and q

$$\mathtt{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N_u} \end{pmatrix}, \quad \mathtt{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N_v} \end{pmatrix}, \quad \mathtt{h} = \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_{N_T} \end{pmatrix}, \quad \mathtt{q} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_{N_q} \end{pmatrix}. \tag{26}$$

In the special case of $n_x = n_y$ the vectors $\mathtt{u}, \mathtt{v}$ are of same size, but in general $\mathtt{u}, \mathtt{v}, \mathtt{h}, \mathtt{q}$ all differ in their sizes. Unfortunately, the vector-representation of the variables leads to the necessity to distinguish between grid nodes that are in vicinity of the boundaries and those in the middle of the domain. Hence, we have on the u-grid for $j \in \{1, ..., N_u\}$

$$x_j^u = \Delta x(1 + (j-1) \bmod (n_x - 1)) \tag{27a}$$

$$y_j^u = \frac{\Delta y}{2}\left(1 + 2\,\mathrm{floor}\left(\frac{j-1}{n_x-1}\right)\right) \tag{27b}$$

where the superscript u simply clarifies that this is the jth grid point on the u-grid. On the v-grid this is similarly for $k \in \{1, ..., N_v\}$

$$x_k^v = \frac{\Delta x}{2}(1 + (k-1) \bmod n_x) \tag{28a}$$

$$y_k^v = \Delta y\left(1 + \mathrm{floor}\left(\frac{k-1}{n_x}\right)\right) \tag{28b}$$

and on the q-grid this becomes with $l \in \{1, ..., N_q\}$

$$x_l^q = \Delta x(l \bmod (n_x+1)) \tag{29a}$$

$$y_l^q = \Delta y\,\mathrm{floor}\left(\frac{l}{n_x+1}\right). \tag{29b}$$

In the following we will evaluate the model variables on different grids and denote that with $\mathbf{x}_i$, $\mathbf{x}_j^u$, $\mathbf{x}_k^v$ and $\mathbf{x}_l^q$, which is short-hand for all grid points of the T-, u-, v- and q-grid, respectively.

Model code

Relevant scripts `.py` and functions, denoted with (), and dependencies ←

```
swm_param.py
↳ set_param() ← set_grid()
↳ set_grid()
```

2.2 Gradients

2.2.1 Matrix-Vector multiplication idea

Representing the model variables in vector-form, as discussed above, enables us to think of a gradient ∂ as a linear map between two vector spaces[1] V_1, V_2. Hence, any gradient ∂ in that sense can be written as a matrix $\mathbf{G}$ which is multiplied with a vector $\mathbf{z}$ representing one of the model's variables:

$$\partial : V_1 \to V_2, \mathbf{z} \to \mathbf{G}\mathbf{z} \tag{30}$$

Having 4 different grids, we have to deal with four different vector spaces V_u, V_v, V_T, V_q with dimensions N_u, N_v, N_T, N_q, respectively. In the following we will investigate the entries of $\mathbf{G}$ such that they describe centred finite differences on the four different grids. The same holds for any linear interpolation $\mathcal{I}$ from one grid to another as will be discussed in section 2.3. We write then

$$\mathcal{I} : V_1 \to V_2, \mathbf{z} \to \mathbf{I}\mathbf{z} \tag{31}$$

2.2.2 Notation for surrounding grid points

Writing the scalar variables on a grid as vectors comes with the disadvantage, that the exact mathematical indexing gets rather complicated and lacks readability. We will therefore discuss the gradient and interpolation operations in the following only in terms of their stencil, i.e. the linear combination of the variables at the surrounding grid points that yields the desired result. To be a bit more precise and to include the treatment of the boundaries, we will use in the following the notation that is visualized in Fig. 2.

A superscript arrow points in the direction relative to the grid point where an operation is evaluated. More clearly, regarding a variable a at $\mathbf{x} = (x_a, y_a)$ somewhere in the middle of the domain away from the boundaries. Then,

$$b^{\leftarrow} = b\big|_{\mathbf{x}=(x_a - \frac{1}{2}\Delta x, y_a)}, \quad b^{\rightarrow} = b\big|_{\mathbf{x}=(x_a + \frac{1}{2}\Delta x, y_a)} \tag{32}$$

where b is a variable that sits on the grid that is shifted by $\frac{1}{2}\Delta x$ in x-direction, and

$$c^{\downarrow} = c\big|_{\mathbf{x}=(x_a, y_a - \frac{1}{2}\Delta y)}, \quad c^{\uparrow} = c\big|_{\mathbf{x}=(x_a, y_a - \frac{1}{2}\Delta y)} \tag{33}$$

where c is a variable that sits on the grid that is shifted by $\frac{1}{2}\Delta y$ in y-direction, and

$$d^{\nearrow} = d\big|_{\mathbf{x}=(x_a + \frac{1}{2}\Delta x, y_a + \frac{1}{2}\Delta y)}, \quad d^{\searrow} = d\big|_{\mathbf{x}=(x_a + \frac{1}{2}\Delta x, y_a - \frac{1}{2}\Delta y)} \tag{34a}$$

$$d^{\swarrow} = d\big|_{\mathbf{x}=(x_a - \frac{1}{2}\Delta x, y_a - \frac{1}{2}\Delta y)}, \quad d^{\nwarrow} = d\big|_{\mathbf{x}=(x_a - \frac{1}{2}\Delta x, y_a + \frac{1}{2}\Delta y)} \tag{34b}$$

where d is a variable that sits on the grid that is shifted by both $\frac{1}{2}\Delta x$ in x-direction and $\frac{1}{2}\Delta y$ in y-direction. The notation is therefore independent on the indexing, which is for comparison still given in appendix A. For the description of the advective terms (section 2.5) we may relate grid points that are further away by

$$b^{\Leftarrow} = b\big|_{\mathbf{x}=(x_a - \Delta x, y_a)}, \quad b^{\Rightarrow} = b\big|_{\mathbf{x}=(x_a + \Delta x, y_a)} \tag{35}$$

[1]Strictly speaking, these are not vector spaces as the underlying algebraic field are not the real numbers but the set of computer representable numbers given a certain precision, e.g. 64bit. These are mathematically not an algebraic field, but for simplicity here regarded as a good approximation to it.

and similar for $\Uparrow, \Downarrow$.

The operator stencils usually change close to the boundary due to boundary conditions. In order to treat these cases separately we introduce the following, index-independent notation: Call a grid node

(i) Northern boundary (NB)
(ii) Western boundary (WB)
(iii) Southern boundary (SB)
(iv) Eastern boundary (EB)

when the evaluation of a stencil for that grid node involves unresolved variables (because they sit either outside the domain $\mathcal{D}$ or sit on the boundary but always given by the boundary condition) at

(i) $y = L_y$
(ii) $x = 0$
(iii) $y = 0$
(iv) $x = L_x$

Note that there is an overlap of two adjacent boundaries that we call accordingly

North-East corner (NE)
North-West corner (NW)
South-West corner (SW)
South-East corner (SE)

This notation is unfortunately only useful when stencils are small, but then provide a readable alternative.

2.2.3 Centred differences

Centred finite difference approximation of the gradient in x-direction of a variable h on the T-grid yields a result that sits on the u-grid (as can be seen from Fig. 1) and reads

$$\partial_x h|_{\mathbf{x}=\mathbf{x}_j^u} \approx \frac{h^{\rightarrow} - h^{\leftarrow}}{\Delta x} \tag{36}$$

This is the well-known $(-1, 1)$-stencil. The y-derivative of a variable of the T-grid sits on the v-grid:

$$\partial_y h|_{\mathbf{x}=\mathbf{x}_k^v} \approx \frac{h^{\uparrow} - h^{\downarrow}}{\Delta y} \tag{37}$$

The x-derivative of a variable on the u-grid includes the kinematic boundary conditions, hence a 0 appears for computations involving the boundary nodes. The result sits on the T-grid:

$$\partial_x u|_{\mathbf{x}=\mathbf{x}_i} \approx \frac{1}{\Delta x} \begin{cases} u^{\rightarrow} - 0 & \text{if Western boundary} \\ 0 - u^{\leftarrow} & \text{if Eastern boundary} \\ u^{\rightarrow} - u^{\leftarrow} & \text{else.} \end{cases} \tag{38}$$

Similarly for the y-derivative on the v-grid which sits again on the T-grid:

$$\partial_y v|_{\mathbf{x}=\mathbf{x}_i} \approx \frac{1}{\Delta y} \begin{cases} v^{\uparrow} - 0 & \text{if Southern boundary} \\ 0 - v^{\downarrow} & \text{if Northern boundary} \\ v^{\uparrow} - v^{\downarrow} & \text{else.} \end{cases} \tag{39}$$

The discretization of $\partial_x q$ and $\partial_y q$ are in close relation to equation 36 and 37 and do not involve any boundary conditions

$$\partial_x q|_{\mathbf{x}=\mathbf{x}_k^v} \approx \frac{q^{\rightarrow} - q^{\leftarrow}}{\Delta x}, \quad \partial_y q|_{\mathbf{x}=\mathbf{x}_j^u} \approx \frac{q^{\uparrow} - q^{\downarrow}}{\Delta y} \tag{40}$$

Note, that some grid nodes of q on the boundary are simply not evaluated in this computation.

2.2.4 Implementing the tangential boundary conditions

For $\partial_y u, \partial_x v$ the tangential boundary conditions as no-slip (equation 5) or free-slip (equation 6) come into play. For simplicity we first look at $\partial_y u|_{\mathbf{x}=\mathbf{x}_l^q}$ for $l = 2$, i.e. at $\mathbf{x} = (\Delta x, 0)$, where the derivative is

$$\partial_y u|_{\mathbf{x}=(\Delta x,0)} \approx \frac{u_1 - u_b}{\Delta y} \tag{41}$$

with $u_b = u(\Delta x, -\frac{1}{2}\Delta y)$ the velocity just outside the domain if the grid where extend in negative y direction. To match the no-slip boundary condition (equation 5) we set $u_b = -u_1$, so that at $\mathbf{x} = (\Delta x, 0)$, i.e. right on the boudary we have $u = 0$, when applying linear interpolation. This yields

$$\partial_y u|_{\mathbf{x}=(\Delta x,0)} \approx \frac{2u_1}{\Delta y} \tag{42}$$

In contrast, when choosing free-slip boundary conditions (equation 6) we set $u_b = u_1$ for similar reasoning. Then

$$\partial_y u|_{\mathbf{x}=(\Delta x,0)} \approx \frac{0u_1}{\Delta y} = 0 \tag{43}$$

as desired. With introducing the parameter $\alpha \in \{0, 2\}$ we can switch between no-slip ($\alpha = 2$) and free-slip boundary conditions ($\alpha = 0$). A choice of $0 < \alpha < 2$ corresponds to partial-slip (see Madec, 2016). Following these ideas we write the y-derivative on the u-grid as

$$\partial_y u|_{\mathbf{x}=\mathbf{x}_l^q} \approx \frac{1}{\Delta y} \begin{cases} 0 & \text{if Western or Eastern boundary} \\ \alpha u^{\uparrow} & \text{if Southern boundary without SW,SE} \\ -\alpha u^{\downarrow} & \text{if Northern boundary without NW, NE} \\ u^{\uparrow} - u^{\downarrow} & \text{else.} \end{cases} \tag{44}$$

which sits then on the q-grid. Similarly we have the discretization of $\partial_x v$ as

$$\partial_x v|_{\mathbf{x}=\mathbf{x}^q_l} \approx \frac{1}{\Delta x} \begin{cases} 0 & \text{if Northern or Western boundary} \\ \alpha v^{\rightarrow} & \text{if Western boundary without NW,SW} \\ -\alpha v^{\leftarrow} & \text{if Eastern boundary without NE, SE} \\ v^{\rightarrow} - v^{\leftarrow} & \text{else.} \end{cases} \tag{45}$$

2.2.5 Gradient operation as matrix multiplication

In the following we will use a notation where the subscript denotes the direction of the derivative, hence x or y, and the superscript u, v, T, q denotes the vector space, where ∂ is mapping from. For the x-derivative of h on the T-grid this is

$$\partial^T_x : V_T \rightarrow V_u, \mathtt{h} \rightarrow \mathbf{G}^T_x \mathtt{h} \tag{46}$$

and the result $\mathbf{G}^T_x \mathtt{h}$ sits then on the u-grid (as defined in equation 36) or equivalently is element of the vector space V_u. The matrix $\mathbf{G}^T_x$ is therefore of size $N_u \times N_T$ and so in general not a square matrix. The derivatives $\partial_y h, \partial_x u, \partial_y u, \partial_x v, \partial_y v, \partial_x q$ and $\partial_y q$ can similarly be written as

$$\partial^T_y : V_T \rightarrow V_v, \mathtt{h} \rightarrow \mathbf{G}^T_y \mathtt{h} \tag{47a}$$
$$\partial^u_x : V_u \rightarrow V_T, \mathtt{u} \rightarrow \mathbf{G}^u_x \mathtt{u} \tag{47b}$$
$$\partial^u_y : V_u \rightarrow V_q, \mathtt{u} \rightarrow \mathbf{G}^u_y \mathtt{u} \tag{47c}$$
$$\partial^v_x : V_v \rightarrow V_q, \mathtt{v} \rightarrow \mathbf{G}^v_x \mathtt{v} \tag{47d}$$
$$\partial^v_y : V_v \rightarrow V_T, \mathtt{v} \rightarrow \mathbf{G}^v_y \mathtt{v} \tag{47e}$$
$$\partial^q_x : V_q \rightarrow V_v, \mathtt{q} \rightarrow \mathbf{G}^q_x \mathtt{q} \tag{47f}$$
$$\partial^q_y : V_q \rightarrow V_u, \mathtt{q} \rightarrow \mathbf{G}^q_y \mathtt{q} \tag{47g}$$

All entries of the gradient matrices $\mathbf{G}$ follow from the equations 36, 37, 38, 39, 40, 44 and 54 and are then only a matter of indexing (one way how to index all grid nodes can be seen in appendix A). For simplicity and visualization, the entries are shown for an example grid ($n_x = n_y = 5$) in Fig 3 - 6.

Model code

Relevant scripts `.py` and functions, denoted with (), and dependencies $\leftarrow$

```
swm_operators.py
↳ set_grad_mat()
```

2.3 Interpolation

2.3.1 2-point spatial interpolation

With variables that sit on four different grids it is sometimes necessary to transform one variable from one grid onto another. Regarding the term $\partial_x(uh)$ from equation 9 we either need to find a representation of u on the T-grid or of h on the u-grid in order to multiply them. This is done via linear interpolation of the closest grid points. In the following we will investigate the interpolations from any of the four grids to any other.

From T-grid to u- or v-grid and vice versa

Let $\mathcal{I}_T^u(\mathtt{h}) = \mathtt{h}_u$ the linear interpolation of $\mathtt{h}$ from the T-grid (subscript of the interpolation function $\mathcal{I}$) onto the u-grid (superscript of $\mathcal{I}$), then

$$\mathtt{h}_u = h|_{\mathbf{x}=\mathbf{x}_j^u} \approx \frac{h^{\leftarrow} + h^{\rightarrow}}{2} \tag{48}$$

which corresponds to spatial averaging of two neighbouring grid points in the x-direction. Please note the similarity to equation (36). Similarly to the gradients, we can write this operation via a matrix multiplication with $\mathbf{I}_T^u$ (sub- and superscript meaning as above)

$$\mathcal{I}_T^u : V_T \to V_u, \mathtt{h} \to \mathbf{I}_T^u \mathtt{h} \tag{49}$$

Due to the similarity in equation (48) and (36), $\mathbf{I}_T^u$ is the same as $\mathbf{G}_x^T$ but all non-zero entries replaced by $\frac{1}{2}$. Same holds for the interpolation of $\mathtt{h}$ onto the v-grid, i.e. a spatial averaging in y-direction of two neighbouring grid points

$$\mathtt{h}_v = h|_{\mathbf{x}=\mathbf{x}_k^v} \approx \frac{h^{\uparrow} + h^{\downarrow}}{2} \tag{50}$$

which can again be written as

$$\mathcal{I}_T^v : V_T \to V_v, \mathtt{h} \to \mathbf{I}_T^v \mathtt{h}. \tag{51}$$

with $\mathbf{I}_T^v$ obtained from $\mathbf{G}_y^T$ by setting all non-zero entries to $\frac{1}{2}$. Same relations hold for $\mathbf{I}_q^v$ and $\mathbf{G}_x^q$ (2-point interpolation in x-direction), $\mathbf{I}_q^u$ and $\mathbf{G}_y^q$ (2-point interpolation in y-direction). And also, including the kinematic boundary condition (equation 3), for $\mathbf{I}_u^T$ and $\mathbf{G}_x^u$ as well as $\mathbf{I}_v^T$ and $\mathbf{G}_y^v$. Interestingly,

$$\mathbf{I}_u^T = \mathbf{I}_T^{u\,\prime} \quad , \quad \mathbf{I}_v^T = \mathbf{I}_T^{v\,\prime} \tag{52}$$

where $'$ denotes the matrix transpose.

From u-grid and v-grid to q-grid

For the interpolation matrices $\mathbf{I}_u^q, \mathbf{I}_v^q$ the lateral boundary conditions are important. In fact, following the ideas around equation (44) we obtain the 2-point interpolation from the u-grid onto the q-grid as

$$\mathtt{u}_q = u|_{\mathbf{x}=\mathbf{x}_l^q} \approx \begin{cases} 0 & \text{if Western or Eastern boundary} \\ (1-\frac{\alpha}{2})u^{\uparrow} & \text{if Southern boundary without SW,SE} \\ (1-\frac{\alpha}{2})u^{\downarrow} & \text{if Northern boundary without NW,NE} \\ \frac{1}{2}(u^{\uparrow} + u^{\downarrow}) & \text{else.} \end{cases} \tag{53}$$

with α being the tangential boundary condition parameter ($\alpha = 0$ is free-slip, $\alpha = 2$ is no-slip). Again, $\mathbf{I}_v^q$ is then straight forward

$$\mathtt{v}_q = v|_{\mathbf{x}=\mathbf{x}_l^q} \approx \begin{cases} 0 & \text{if Northern or Western boundary} \\ (1-\frac{\alpha}{2})v^{\rightarrow} & \text{if Western boundary without NW,SW} \\ (1-\frac{\alpha}{2})v^{\leftarrow} & \text{if Eastern boundary without NE, SE} \\ \frac{1}{2}(v^{\rightarrow} + v^{\leftarrow}) & \text{else.} \end{cases} \tag{54}$$

2.3.2 4-point spatial interpolation

From u-grid to v-grid and vice versa

The previous interpolations involve 2-point spatial averaging, however, the interpolations $\mathcal{I}_u^v, \mathcal{I}_v^u, \mathcal{I}_q^T, \mathcal{I}_T^q$ require averaging from the four surrounding grid points (see Fig. 1) and will be described in the following.

The interpolation $\mathcal{I}_u^v$ from the u-grid onto the v-grid is

$$\mathtt{u}_v = u|_{\mathbf{x}=\mathbf{x}_k^v} \approx \frac{1}{4}\begin{cases} (u^{\searrow} + u^{\nearrow}) & \text{if Western boundary} \\ (u^{\swarrow} + u^{\nwarrow}) & \text{if Eastern boundary} \\ (u^{\searrow} + u^{\nearrow} + u^{\swarrow} + u^{\nwarrow}) & \text{else.} \end{cases} \tag{55}$$

and similarly the interpolation $\mathcal{I}_v^u$ reads

$$\mathtt{v}_u = v|_{\mathbf{x}=\mathbf{x}_j^u} \approx \frac{1}{4}\begin{cases} (v^{\searrow} + v^{\swarrow}) & \text{if Northern boundary} \\ (v^{\nearrow} + v^{\nwarrow}) & \text{if Southern boundary} \\ (v^{\searrow} + v^{\nearrow} + v^{\swarrow} + v^{\nwarrow}) & \text{else.} \end{cases} \tag{56}$$

Note, that both $\mathcal{I}_u^v$ and also $\mathcal{I}_v^u$ include the kinematic boundary condition. Once we write this interpolation as a matrix $\mathbf{I}_u^v$, following the same arguments, we can deduce that

$$\mathbf{I}_v^u = \mathbf{I}_u^{v\prime} \tag{57}$$

The interpolation from the u-grid to the v-grid is the transpose of the interpolation from v to u.

From q-grid to T-grid and vice versa

The interpolation $\mathcal{I}_q^T$ from the q-grid to the T-grid is

$$\mathtt{q}_T = q|_{\mathbf{x}=\mathbf{x}_i} \approx \frac{1}{4}(q^{\searrow} + q^{\nearrow} + q^{\swarrow} + q^{\nwarrow}) \tag{58}$$

And finally the interpolation $\mathcal{I}_T^q$ makes use of the additional boundary condition in equation (4), which is $\nabla h = 0$ at all boundaries.

$$\mathtt{h}_q = h|_{\mathbf{x}=\mathbf{x}_l^q} \approx \begin{cases} T^{\swarrow} & \text{if North-East corner (NE)} \\ T^{\searrow} & \text{if North-West corner (NW)} \\ T^{\nwarrow} & \text{if South-East corner (SE)} \\ T^{\nearrow} & \text{if South-West corner (SW)} \\ \frac{1}{2}(T^{\searrow} + T^{\swarrow}) & \text{if Northern boundary without NW,NE} \\ \frac{1}{2}(T^{\swarrow} + T^{\nwarrow}) & \text{if Eastern boundary without NE,SE} \\ \frac{1}{2}(T^{\nearrow} + T^{\searrow}) & \text{if Western boundary without NW, SW} \\ \frac{1}{2}(T^{\nearrow} + T^{\nwarrow}) & \text{if Southern boundary without SW, SE} \\ \frac{1}{4}(T^{\searrow} + T^{\nearrow} + T^{\swarrow} + T^{\nwarrow}) & \text{else.} \end{cases} \tag{59}$$

As for the gradient matrices $\mathbf{G}$ all interpolation matrices are for simplicity and readability shown in Fig. 7-12 for a sample grid with $n_x = n_y = 5$.

Model code

Relevant scripts `.py` and functions, denoted with (), and dependencies $\leftarrow$

```
swm_operators.py
↳ set_interp_mat()
```

2.4 Linear and non-linear operations

As the equations of interest are essentially non-linear we can not describe all operations in the model in terms of matrix-vector multiplications or additions of vectors. In fact, it turns out that all non-linear operations in equation (9), once discretized, are element-wise vector-vector multiplications. Let $\mathtt{a}, \mathtt{b}$ two vectors of same length N respectively element of a vector space V

$$\mathtt{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix}, \quad \mathtt{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix} \tag{60}$$

Hence, they should sit on the same grid. We then define the element-wise vector-vector multiplication $*$ as

$$* : V \times V \to V, \mathtt{a} * \mathtt{b} \to \begin{pmatrix} a_1 b_1 \\ a_2 b_2 \\ \vdots \\ a_N b_N \end{pmatrix} \tag{61}$$

and define the order of computation as inferior to matrix-vector multiplication, i.e.

$$\mathbf{A}\mathtt{b} * \mathtt{c} = (\mathbf{A}\mathtt{b}) * \mathtt{c} \neq \mathbf{A}(\mathtt{b} * \mathtt{c}) \tag{62}$$

for all matrices $\mathbf{A}$, and vectors $\mathtt{b}, \mathtt{c}$.

2.5 Advection term

In the following two different schemes are discussed that aim at discretizing the advection terms

$$(qhv, -qhu), \quad \text{with } q = \frac{f + \partial_x v - \partial_y u}{h} \tag{63}$$

with the potential vorticity q, that appear in equation (9).

2.5.1 Sadourny (1975) enstrophy conserving scheme

Sadourny, 1975 proposed the following enstrophy conserving scheme

$$qhv|_{\mathtt{x}=\mathtt{x}^u_j} \approx \mathbf{I}^u_q \mathtt{q} * \mathbf{I}^u_v (\mathtt{v} * \mathbf{I}^v_T \mathtt{h}) \tag{64a}$$

$$-qhu|_{\mathtt{x}=\mathtt{x}^v_k} \approx -\mathbf{I}^v_q \mathtt{q} * \mathbf{I}^v_u (\mathtt{u} * \mathbf{I}^u_T \mathtt{h}) \tag{64b}$$

where the brackets () simply denote that the $*$-operation is computed first and the result is then interpolated via the matrix-vector-multiplication.

Model code

Relevant scripts `.py` and functions, denoted with (), and dependencies ←

```
swm_rhs.py
↳ rhs()
swm_integration.py
↳ time_integration() ← RK4()
↳ RK4() ← rhs()
```

2.5.2 Arakawa and Lamb (1981) energy and enstrophy conserving scheme

The energy and enstrophy conserving scheme developed by Arakawa and Lamb, 1981, called AL hereafter, has a wider stencil compared to the scheme from Sadourny, 1975 (hereafter SZ), i.e. computationally more costly, but was also found to perform better (Salmon, 2007). That means AL transports less enstrophy (which is essentially squared vorticity) to higher wavenumbers, which reduces the numerical noise on the grid scale compared to SZ. For further details see also Salmon, 2004 where the scheme is provided in a much more readable notation as in the original AL paper.

As in SZ compute the mass fluxes $U = uh$ and $V = vh$ as

$$\mathtt{U} = uh|_{\mathbf{x}=\mathbf{x}^u_j} \approx \mathtt{u} * \mathbf{I}^u_T\mathtt{h} \tag{65a}$$
$$\mathtt{V} = vh|_{\mathbf{x}=\mathbf{x}^v_k} \approx \mathtt{v} * \mathbf{I}^v_T\mathtt{h} \tag{65b}$$

The advective term in the u-component qhv is then discretized as a summation of linear combinations of the surrounding potential vorticity points q and the mass fluxes U, V. We start with computing the linear combinations of q. Let $\mathbf{A}_{L1}, \mathbf{A}_{L2}, \mathbf{A}_{L3}, \mathbf{A}_{L4}$ (without meaning of the subscripts) be four different interpolations[2] (directly written as matrix) fom the q- to the T-grid defined as

$$\mathbf{A}_{L1}\mathtt{q} = \tfrac{1}{24}(2q^{\nwarrow} + q^{\nearrow} + q^{\swarrow} + 2q^{\searrow}), \quad \mathbf{A}_{L2}\mathtt{q} = \tfrac{1}{24}(q^{\nwarrow} + 2q^{\nearrow} + 2q^{\swarrow} + q^{\searrow}) \tag{66a}$$
$$\mathbf{A}_{L3}\mathtt{q} = \tfrac{1}{24}(q^{\nwarrow} + q^{\nearrow} - q^{\swarrow} - q^{\searrow}), \quad \mathbf{A}_{L4}\mathtt{q} = \tfrac{1}{24}(q^{\nwarrow} - q^{\nearrow} + q^{\swarrow} - q^{\searrow}) \tag{66b}$$

for visualization the corresponding stencils (denoted with subscript 1, 2, 3, 4) are

$$\tfrac{1}{24}\begin{vmatrix} 2 & 1 \\ 1 & 2 \end{vmatrix}_1, \quad \tfrac{1}{24}\begin{vmatrix} 1 & 2 \\ 2 & 1 \end{vmatrix}_2 \tag{67a}$$
$$\tfrac{1}{24}\begin{vmatrix} 1 & 1 \\ -1 & -1 \end{vmatrix}_3, \quad \tfrac{1}{24}\begin{vmatrix} 1 & -1 \\ 1 & -1 \end{vmatrix}_4 \tag{67b}$$

We might use $\mathbf{A}_L$ and mean then any of matrices in equation (66). We define interpolation matrices $\mathbf{R}$ to get the variables $\mathbf{A}_L\mathtt{q}$ from the T- to the u- or v-grid. As the matrices $\mathbf{R}$ contain a maximum of one entry per row, they are rather shift matrices or correspond to nearest-point interpolation. Hence, they could also be written in terms of an index (as

[2] $\mathbf{A}_{L3}$ and $\mathbf{A}_{L4}$ should be rather regarded as potential vorticity gradient due to the minus sign in their stencil.

actually done in the model code). We first look at $\mathbf{R}_v^{\uparrow}$ which picks for all v-grid points the corresponding $\mathbf{A}_L\mathtt{q}$-value that sits $\frac{1}{2\Delta y}$ North of this point on the T-grid.

$$\mathbf{R}_v^{\uparrow}\mathbf{A}_L\mathtt{q} = (\mathbf{A}_L\mathtt{q})^{\uparrow} \tag{68}$$

Hence, the interpolation of this operator shifts the T-grid southward by $\frac{1}{2\Delta y}$ to place them on the v-grid. The grid cell row closest to $y = 0$ is therefore left-out. Similar for $\mathbf{R}_v^{\downarrow}$, $\mathbf{R}_u^{\leftarrow}$ and $\mathbf{R}_u^{\rightarrow}$, which are

$$\mathbf{R}_v^{\downarrow}\mathbf{A}_L\mathtt{q} = (\mathbf{A}_L\mathtt{q})^{\downarrow}, \quad \mathbf{R}_u^{\leftarrow}\mathbf{A}_L\mathtt{q} = (\mathbf{A}_L\mathtt{q})^{\leftarrow}, \quad \mathbf{R}_u^{\rightarrow}\mathbf{A}_L\mathtt{q} = (\mathbf{A}_L\mathtt{q})^{\rightarrow}. \tag{69}$$

Once the $\mathbf{A}_L$-interpolated potential vorticity sits on the u- and v-grid they are multiplied with the mass fluxes U, V. In order to get a discretized advection term, AL interpolate the surrounding absolute vorticity fluxes qU, qV onto each u- and v-grid point. For this final interpolation we further need another set of shift-matrices $\mathbf{T}$ (they could again be written in terms of an index) that shift a variable $\mathtt{v}$ from the v to the u-grid as follows

$$\mathbf{T}_{v\to u}^{\nearrow}\mathtt{v} = \begin{cases} 0 & \text{if Northern boundary} \\ v^{\nearrow} & \text{else.} \end{cases} \tag{70}$$

and include the kinematic boundary condition (case 1). Similarly, we have

$$\mathbf{T}_{v\to u}^{\searrow}\mathtt{v} = \begin{cases} 0 & \text{if Southern boundary} \\ v^{\searrow} & \text{else.} \end{cases} \tag{71}$$

$$\mathbf{T}_{v\to u}^{\swarrow}\mathtt{v} = \begin{cases} 0 & \text{if Southern boundary} \\ v^{\swarrow} & \text{else.} \end{cases} \tag{72}$$

$$\mathbf{T}_{v\to u}^{\nwarrow}\mathtt{v} = \begin{cases} 0 & \text{if Northern boundary} \\ v^{\nwarrow} & \text{else.} \end{cases} \tag{73}$$

and also $\mathbf{T}_{u\to v}^{\nwarrow}, \mathbf{T}_{u\to v}^{\nearrow}, \mathbf{T}_{u\to v}^{\searrow}, \mathbf{T}_{u\to v}^{\swarrow}$ as well as $\mathbf{T}_u^{\Rightarrow}, \mathbf{T}_u^{\Leftarrow}, \mathbf{T}_v^{\Uparrow}, \mathbf{T}_v^{\Downarrow}$. We are now able to write the u-component of the advection term as

$$\begin{aligned} qhv|_{\mathbf{x}=\mathbf{x}_j^u} \approx\ & \mathbf{T}_{v\to u}^{\nearrow}\left(\mathbf{R}_v^{\downarrow}\mathbf{A}_{L1}\mathtt{q} * \mathtt{V}\right) + \mathbf{T}_{v\to u}^{\searrow}\left(\mathbf{R}_v^{\uparrow}\mathbf{A}_{L2}\mathtt{q} * \mathtt{V}\right) \\ & \mathbf{T}_{v\to u}^{\nwarrow}\left(\mathbf{R}_v^{\downarrow}\mathbf{A}_{L2}\mathtt{q} * \mathtt{V}\right) + \mathbf{T}_{v\to u}^{\swarrow}\left(\mathbf{R}_v^{\uparrow}\mathbf{A}_{L1}\mathtt{q} * \mathtt{V}\right) \\ & \mathbf{T}_u^{\Leftarrow}\left(\mathbf{R}_u^{\rightarrow}\mathbf{A}_{L3}\mathtt{q} * \mathtt{U}\right) - \mathbf{T}_u^{\Rightarrow}\left(\mathbf{R}_u^{\leftarrow}\mathbf{A}_{L3}\mathtt{q} * \mathtt{U}\right) \end{aligned} \tag{74}$$

and the v-component as

$$\begin{aligned} -qhu|_{\mathbf{x}=\mathbf{x}_k^v} \approx\ & -\mathbf{T}_{u\to v}^{\nearrow}\left(\mathbf{R}_v^{\rightarrow}\mathbf{A}_{L1}\mathtt{q} * \mathtt{U}\right) - \mathbf{T}_{u\to v}^{\searrow}\left(\mathbf{R}_v^{\rightarrow}\mathbf{A}_{L2}\mathtt{q} * \mathtt{U}\right) \\ & -\mathbf{T}_{u\to v}^{\nwarrow}\left(\mathbf{R}_v^{\leftarrow}\mathbf{A}_{L2}\mathtt{q} * \mathtt{U}\right) - \mathbf{T}_{u\to v}^{\swarrow}\left(\mathbf{R}_v^{\leftarrow}\mathbf{A}_{L1}\mathtt{q} * \mathtt{U}\right) \\ & -\mathbf{T}_v^{\Uparrow}\left(\mathbf{R}_v^{\downarrow}\mathbf{A}_{L4}\mathtt{q} * \mathtt{V}\right) + \mathbf{T}_v^{\Downarrow}\left(\mathbf{R}_v^{\uparrow}\mathbf{A}_{L4}\mathtt{q} * \mathtt{V}\right) \end{aligned} \tag{75}$$

These equations correspond to equation B.2 from Salmon, 2004 and 3.5 and 3.6 together with 3.34 from Arakawa and Lamb, 1981. However, from this notation it is clear that there are essentially four costly computations that can be precomputed: $\mathbf{A}_{L1}\mathtt{q}, \mathbf{A}_{L2}\mathtt{q}, \mathbf{A}_{L3}\mathtt{q}$ and $\mathbf{A}_{L4}\mathtt{q}$. The remaining shift matrices $\mathbf{R}, \mathbf{T}$ have at maximum one entry per row and can also be implemented as index.

Model code

Relevant scripts .py and functions, denoted with (), and dependencies ←

```
swm_rhs.py
↳ rhs() ← ALadvection()
↳ ALadvection()
```

2.6 Choosing the viscosity and friction coefficients

The remaining coefficients ν_A, ν_B, c_D in equation (9) together with equation (22), can not be chosen from physical principles but its choice should arise from considerations of numerical stability (Griffies and Hallberg, 2000).

2.6.1 Harmonic and biharmonic viscosity

For the given configuration as described in section 1.2 we find the choice

$$\nu_A = 540\,\mathrm{m}^2\,\mathrm{s}^{-1} \tag{76}$$

for a resolution with $\Delta x = \Delta y = 30$ km appropriate (this is assumed in the following of this chapter). That means it is chosen as small as possible but still removing clearly numerical oscillations that occur at the grid scale. This choice also resolves the Munk boundary layer width

$$W_M = \sqrt[3]{\frac{\nu_A}{\beta}} \approx \Delta x \tag{77}$$

with approximately one grid cell. It was proposed to use this as an argument to choose ν_A dependent on the resolution (Cooper and Zanna, 2015)

$$\nu_A = \beta \Delta x^3 \tag{78}$$

Although, this might a criterion for stability, for $\Delta x < 30$ km it was not found to prevent numerical oscillations at the grid scale to occur. Instead a following scaling argument is proposed: At the grid scale $\Delta x = \Delta y$ the advective terms are desired to balance with viscosity

$$\mathcal{O}((\mathbf{u} \cdot \nabla)\mathbf{u}) = \frac{U^2}{\Delta x} \sim \nu_A \frac{U}{\Delta x^2} = \mathcal{O}(\nu_A \nabla^2 \mathbf{u}) \tag{79}$$

with a velocity scale U. It follows a linear scaling of ν_A with Δx

$$\nu_A = U \Delta x \tag{80}$$

under the assumption that the velocity scale does not change considerably. Based on the empirically found value from equation (76) it is therefore proposed to use dependend on the resolution

$$\nu_A(\Delta x) = 540\,\mathrm{m}^2\,\mathrm{s}^{-1}\,\frac{\Delta x}{30\,\mathrm{km}} \tag{81}$$

and this way also implemented in the model code. The biharmonic eddy viscosity scaling is derived from the requirement that harmonic and biharmonic viscosity should be on the same order of magnitude

$$1 = \frac{\mathcal{O}(\nu_A \nabla^2 \mathbf{u})}{\mathcal{O}(\nu_B \nabla^4 \mathbf{u})} = \frac{\nu_A}{\nu_B} \Delta x^2 \tag{82}$$

Hence, we propose a scaling for ν_B as

$$\nu_B(\Delta x) = 540\,\mathrm{m}^2\,\mathrm{s}^{-1}\,\frac{\Delta x^3}{30\,\mathrm{km}} \tag{83}$$

2.6.2 Bottom friction coefficient

2.7 Biharmonic lateral mixing of momentum

2.8 Summary

All gradients are now approximated by centred finite differences and

$$\partial_t u = \mathbf{I}_q^u \mathtt{q} * \mathbf{I}_v^u (v * \mathbf{I}_T^v \mathtt{h}) - \mathbf{G}_x^T \mathtt{p} + \mathtt{F}_x + \nu_B \mathbf{L}_u^2 \mathtt{u} \tag{84a}$$

$$\partial_t v = -\mathbf{I}_q^v \mathtt{q} * \mathbf{I}_u^v (u * \mathbf{I}_T^u \mathtt{h}) - \mathbf{G}_y^T \mathtt{p} + \nu_B \mathbf{L}_v^2 \mathtt{v} \tag{84b}$$

$$\partial_t h = -\mathbf{G}_x^u (\mathtt{u} * \mathbf{I}_T^u \mathtt{h}) - \mathbf{G}_y^v (\mathtt{v} * \mathbf{I}_T^v \mathtt{h}) \tag{84c}$$

with

$$\mathtt{p} = \frac{1}{2}\left(\mathbf{I}_u^T(\mathtt{u}^2) + \mathbf{I}_v^T(\mathtt{v}^2)\right) + g\mathtt{h} \tag{85a}$$

$$\mathtt{q} = \frac{\mathtt{f}_q + \mathbf{G}_x^v \mathtt{v} - \mathbf{G}_y^u \mathtt{u}}{\mathbf{I}_T^q \mathtt{h}} \tag{85b}$$

3 Time discretization

3.1 Runge-Kutta 4th order

The discrete shallow water model is integrated forward in time with the 4th order Runge-Kutta scheme (RK4, CITE!). Summarizing the right-hand side of equations 84 with $\mathrm{rhs}(\mathtt{u},\mathtt{v},\mathtt{h}) = (\mathtt{du},\mathtt{dv},\mathtt{dh})$ the model equations reduce to

$$\partial_t \begin{pmatrix} \mathtt{u} \\ \mathtt{v} \\ \mathtt{h} \end{pmatrix} = \begin{pmatrix} \mathtt{du} \\ \mathtt{dv} \\ \mathtt{dh} \end{pmatrix}. \tag{86}$$

Using RK4, discretizing the temporal derivative reads

$$\begin{pmatrix} \mathtt{u}^{n+1} \\ \mathtt{v}^{n+1} \\ \mathtt{h}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathtt{u}^{n} \\ \mathtt{v}^{n} \\ \mathtt{h}^{n} \end{pmatrix} + \frac{\Delta t}{6} \begin{pmatrix} \mathtt{k}_1^u + 2\mathtt{k}_2^u + 2\mathtt{k}_3^u + \mathtt{k}_4^u \\ \mathtt{k}_1^v + 2\mathtt{k}_2^v + 2\mathtt{k}_3^v + \mathtt{k}_4^h \\ \mathtt{k}_1^h + 2\mathtt{k}_2^h + 2\mathtt{k}_3^h + \mathtt{k}_4^h \end{pmatrix} \tag{87}$$

with the superscript $n, n+1$ denoting the current and next time step, respectively, that lie time Δt apart. The choice of Δt is discussed below. $(\mathtt{k}^u, \mathtt{k}^v, \mathtt{k}^h)$ are approximations for $(\partial_t \mathtt{u}, \partial_t \mathtt{v}, \partial_t \mathtt{k})$ and defined as

$$(\mathtt{k}_1^u, \mathtt{k}_1^v, \mathtt{k}_1^h) = \mathrm{rhs}(\mathtt{u}^n, \mathtt{v}^n, \mathtt{h}^n) \tag{88a}$$

$$(\mathtt{k}_2^u, \mathtt{k}_2^v, \mathtt{k}_2^h) = \mathrm{rhs}(\mathtt{u}^n + \tfrac{\Delta t}{2}\mathtt{k}_1^u, \mathtt{v}^n + \tfrac{\Delta t}{2}\mathtt{k}_1^v, \mathtt{h}^n + \tfrac{\Delta t}{2}\mathtt{k}_1^h) \tag{88b}$$

$$(\mathtt{k}_3^u, \mathtt{k}_3^v, \mathtt{k}_3^h) = \mathrm{rhs}(\mathtt{u}^n + \tfrac{\Delta t}{2}\mathtt{k}_2^u, \mathtt{v}^n + \tfrac{\Delta t}{2}\mathtt{k}_2^v, \mathtt{h}^n + \tfrac{\Delta t}{2}\mathtt{k}_2^h) \tag{88c}$$

$$(\mathtt{k}_4^u, \mathtt{k}_4^v, \mathtt{k}_4^h) = \mathrm{rhs}(\mathtt{u}^n + \Delta t\mathtt{k}_3^u, \mathtt{v}^n + \Delta t\mathtt{k}_3^v, \mathtt{h}^n + \Delta t\mathtt{k}_3^h) \tag{88d}$$

Model code

Relevant scripts `.py` and functions, denoted with (), and dependencies ←

```
swm_integration.py
↳ time_integration() ← RK3() or RK4()
↳ RK3() ← rhs()
↳ RK4() ← rhs()

swm_rhs.py
↳ rhs()
```

3.2 Adams-Bashforth

The shallow water model code also allows to use Adams-Bashforth (AB, CITE!) methods of order 1 to 5. As AB methods are multi-step methods, in this case, order 1 (i.e. Euler forward) is used for the first time step, order 2 is used for the second time step, and so on until the desired order is reached. However, this method is not discussed further here, as it was found to perform not better than RK4, as explained in the next section.

Model code

Relevant scripts `.py` and functions, denoted with (), and dependencies ←

```
swm_integration.py
↳ time_integration() ← rhs(), ABcoefficients()
↳ ABcoefficients()

swm_rhs.py
↳ rhs()
```

3.3 Choosing the time-step size Δt

In the shallow water model, the fastest propagating signals are gravity waves. The phase speed c_p of those waves is Gill, 1982

$$c_p = \sqrt{gh} \approx \sqrt{gH} \tag{89}$$

where the approximation holds in the barotropic case where $\eta \ll H$. In contrast, using a reduced gravity $g' \ll g = 10$ m/s^2 usually yields much larger variations in η and the approximation in equation (89) may become less justified, but in many cases still be useful. As a consequence, waves might propagate significantly faster in certain regions, which can affect the numerical stability. The CFL-number ϵ (named after Courant, Friedrichs and Lewy, CITE!) is then

$$\epsilon = \frac{c_p \Delta t}{\Delta x} \tag{90}$$

Hence, choosing the CFL-number ϵ, we obtain the time step Δt as

$$\Delta t = \epsilon \frac{\Delta x}{c_p} \tag{91}$$

Using RK4, a choice of $\epsilon \leq 0.9$ was found to be stable in the barotropic set up of equation (10). Multi-step schemes such as AB have the advantage, that they only require one evaluation of the right-hand side per time step (in contrast to RK4 which requires 4 evaluations of the right-hand side), which could theoretically decrease the computational time required in order to integrate the model forward. However, in practice, the 3rd order Adams-Bashforth method was found to be stable for $\epsilon \leq 0.2$ (i.e. a decrease of Δt by a factor of 4 to 5), which means that the effective computational performance is on the same order but slightly outperformed by RK4.

Please note, that in general it is possible to recompute Δt during the model integration based on $c_p = \max(\sqrt{gh})$ with equation (91). This comes unfortunately with the disadvantage of unevenly spaced time steps but might be a way to avoid instability arising from large variations of η, for example in a baroclinic set up with reduced gravity $g' \ll g$.

Model code

Relevant scripts `.py` and functions, denoted with (), and dependencies ←

`swm_param.py`

↳ `set_param()` ← `set_grid()`, `set_timestep()`

↳ `set_grid()`

↳ `set_timestep()`

4 Data output

4.1 Temporal subsampling

In order to avoid huge data sets, not every model time step n is stored, but an eligible subsample n_out, which can be chosen by specifying the output time step Δt_out. This is done as follows

$$n_\text{out} = \text{floor}\left(\frac{\Delta t_\text{out}}{\Delta t}\right) \tag{92}$$

As the desired output time step Δt_out is not necessarily an integer multiple of Δt the actual output time step is (in most applications) slightly less than desired due to rounding within the floor-operation. However, the data output is therefore evenly spaced in time and no temporal interpolation is applied.

Model code

Relevant scripts `.py` and functions, denoted with (), and dependencies ←

`swm_param.py`

↳ `set_param()` ← `set_output()`

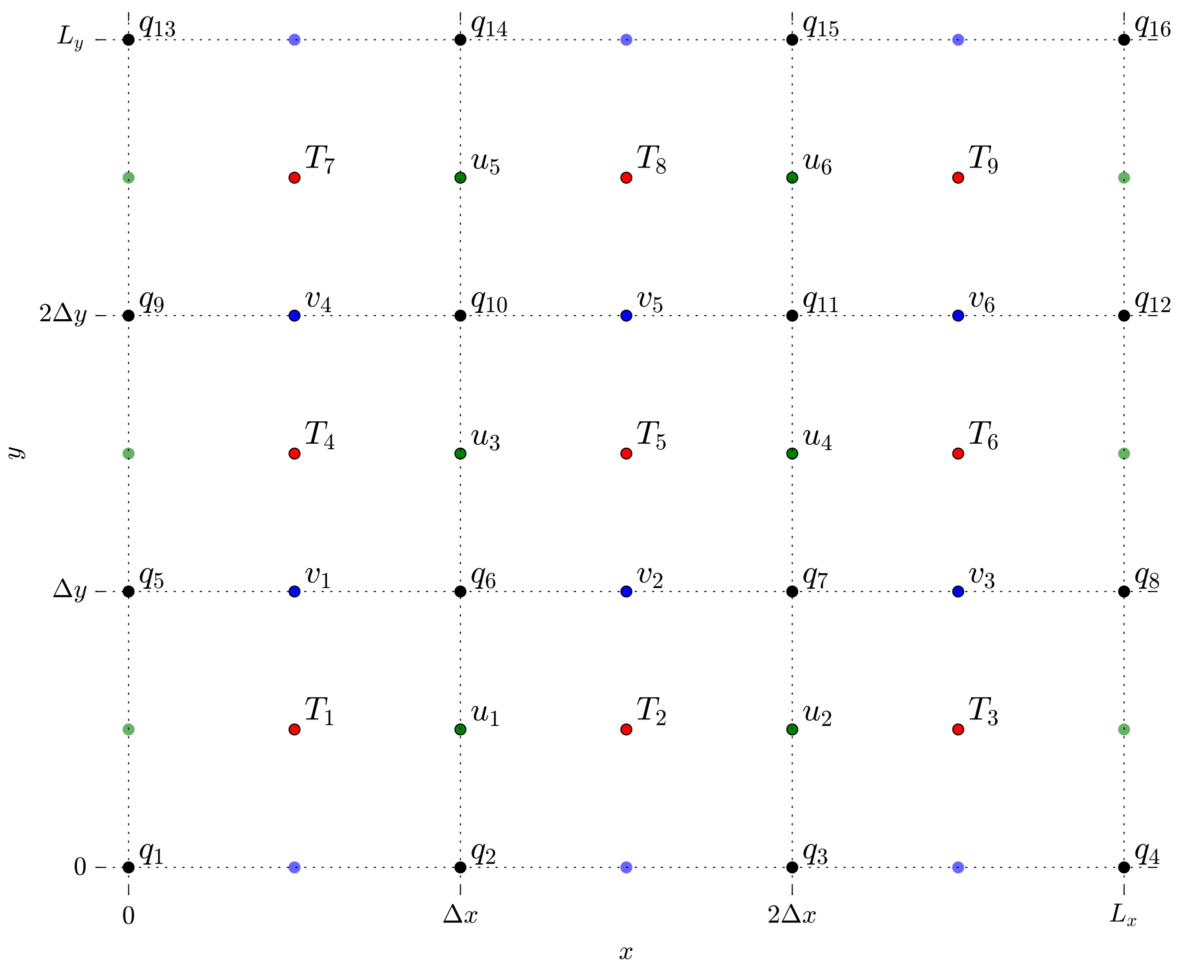
L_y
$2\Delta y$
Δy
0
y
0
Δx
$2\Delta x$
L_x
x
q_{13} q_{14} q_{15} q_{16}
T_7 u_5 T_8 u_6 T_9
q_9 v_4 q_{10} v_5 q_{11} v_6 q_{12}
T_4 u_3 T_5 u_4 T_6
q_5 v_1 q_6 v_2 q_7 v_3 q_8
T_1 u_1 T_2 u_2 T_3
q_1 q_2 q_3 q_4

Figure 1: Grid cell numbering following the ideas of an Arakawa C-grid. The values of u, v vanish at the faint grid nodes on the boundary (kinematic boundary condition, see equation 5 and 6) and therefore these grid nodes are not explicitly resolved.

↳ `set_output()` ← `set_timestep()`

↳ `set_timestep()`

`swm_integration.py`

↳ `time_integration()` ← `feedback_ini()`, `feedback()`

↳ `feedback_ini()` ← `output_nc_ini()`, `set_output()`

↳ `feedback()` ← `output_nc()`, `set_output()`

`swm_output.py`

↳ `output_nc_ini()`

↳ `output_nc()`

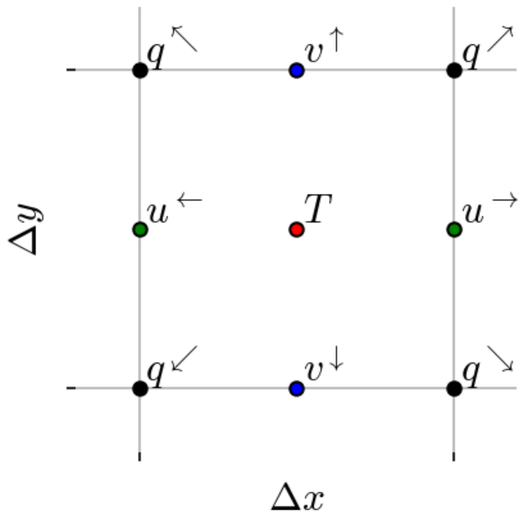

Figure 2: For readability the respective surrounding grid points are marked with a superscript arrow that points the direction. Here shown for a point on the T-grid but easily extendable for all surrounding points on any grid.

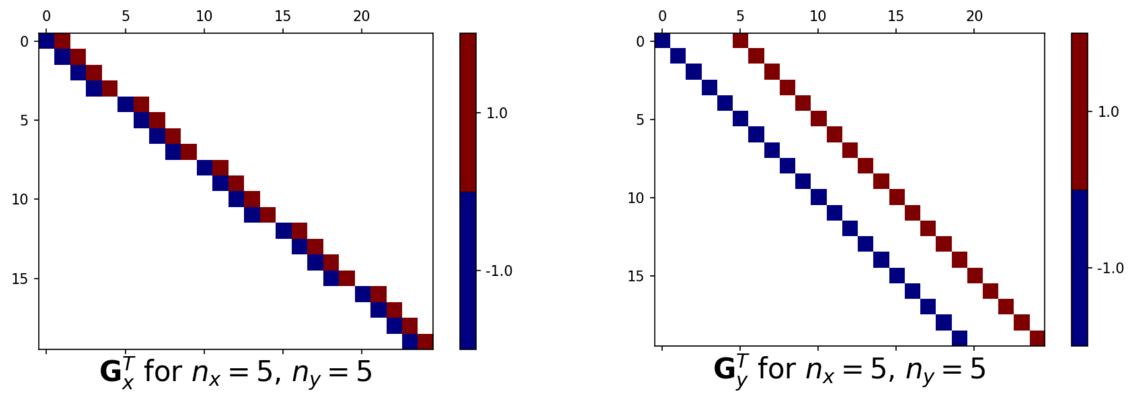

Figure 3: Spatial gradients on the T-grid for an example grid of $n_x = n_y = 5$ omitting the $\frac{1}{\Delta x}, \frac{1}{\Delta y}$ factors written as Matrix operations. (left) $\mathbf{G}_x^T$, the discretized x-derivative, (right) $\mathbf{G}_y^T$ the discretized y-derivative. Blank entries are zero. Please note, that the indexing starts here with 0 as common in many programming languages.

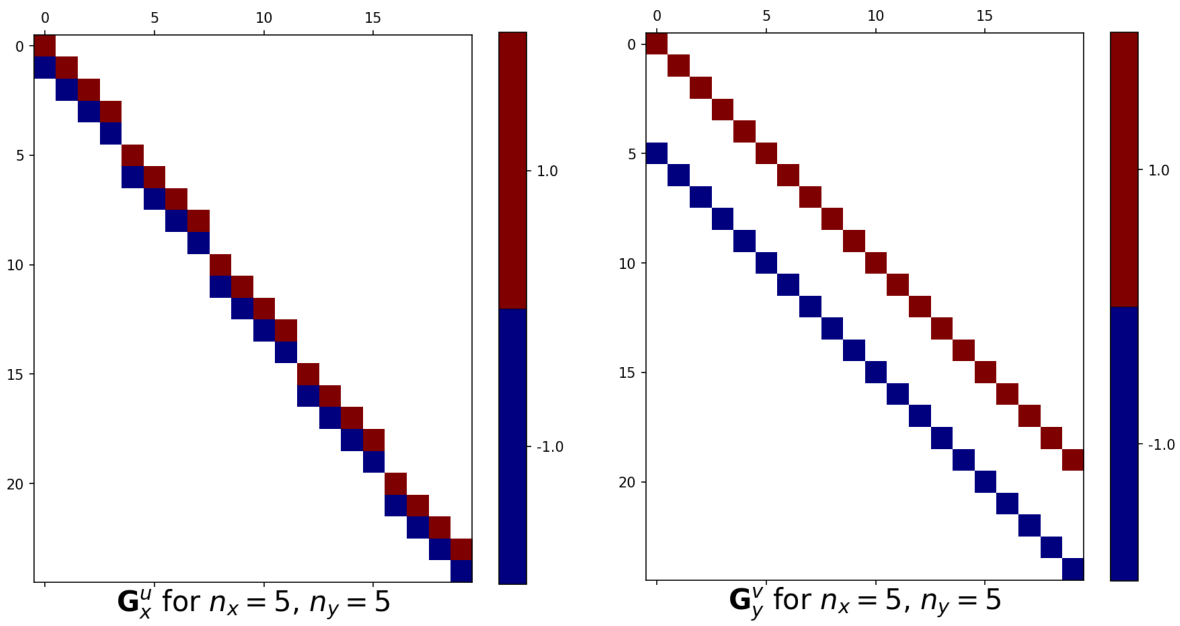

Figure 4: Same as Fig. 3 but for (left) $\mathbf{G}^u_x$ and (right) $\mathbf{G}^v_y$.

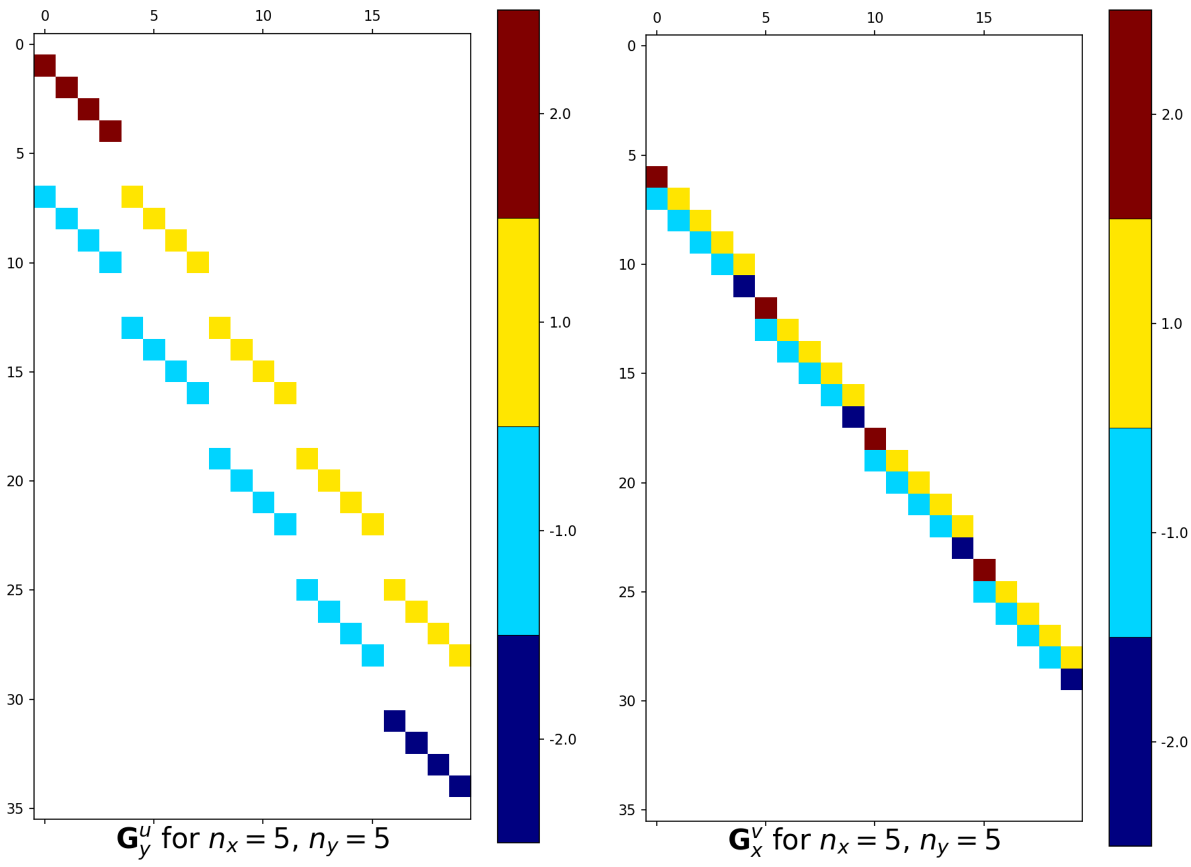

Figure 5: Same as Fig. 3 but for (left) $\mathbf{G}^u_y$ and (right) $\mathbf{G}^v_x$. These are shown here for no-slip boundary conditions, such that $\alpha = 2$ in equation 44. Replacing all entries with ± 2 by 0 yields free-slip boundary conditions.

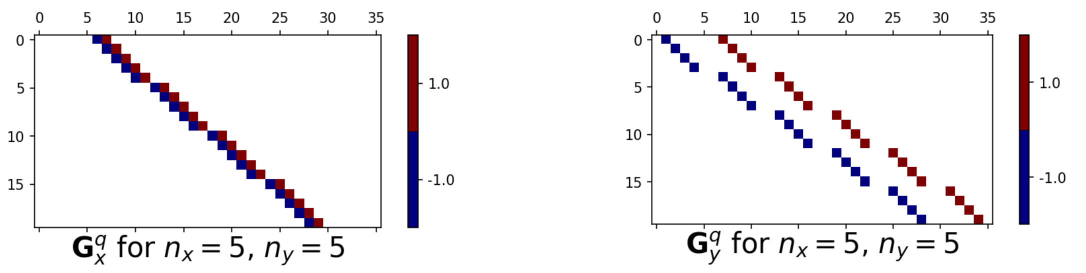

Figure 6: Same as Fig. 3 but for (left) $\mathbf{G}_x^q$ and (right) $\mathbf{G}_y^q$.

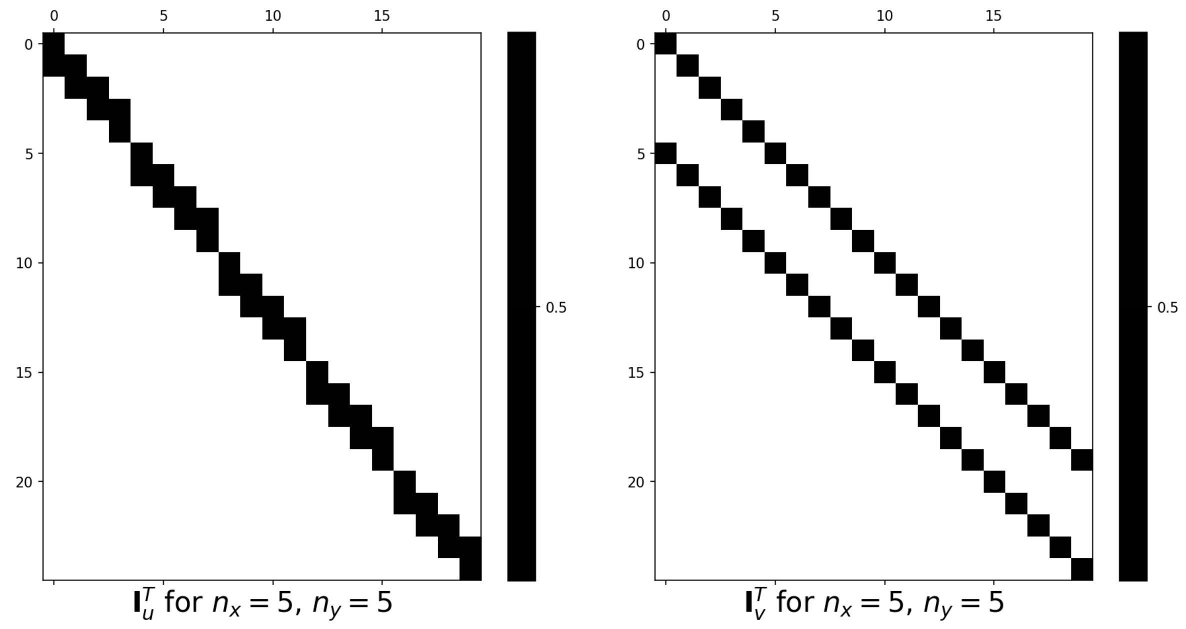

Figure 7: Spatial interpolation between different grids for an example grid of $n_x = n_y = 5$ written as Matrix operations. (left) $\mathbf{I}_u^T$, the interpolation from u- to T-grid, (right) $\mathbf{I}_v^T$, the interpolation from v- to T-grid. Blank entries are zero. Please note, that the indexing starts here with 0 as common in many programming languages.

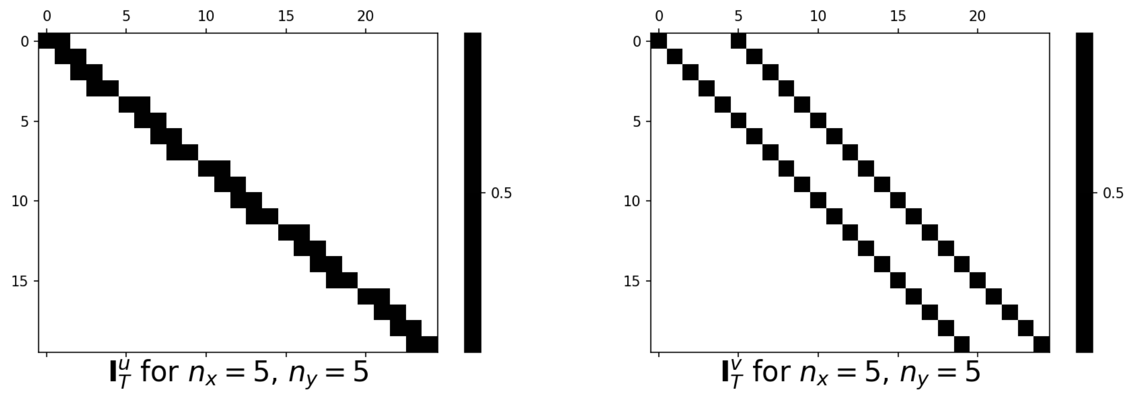

Figure 8: Same as Fig. 7, but for (left) $\mathbf{I}_T^u$ and (right) $\mathbf{I}_T^v$.

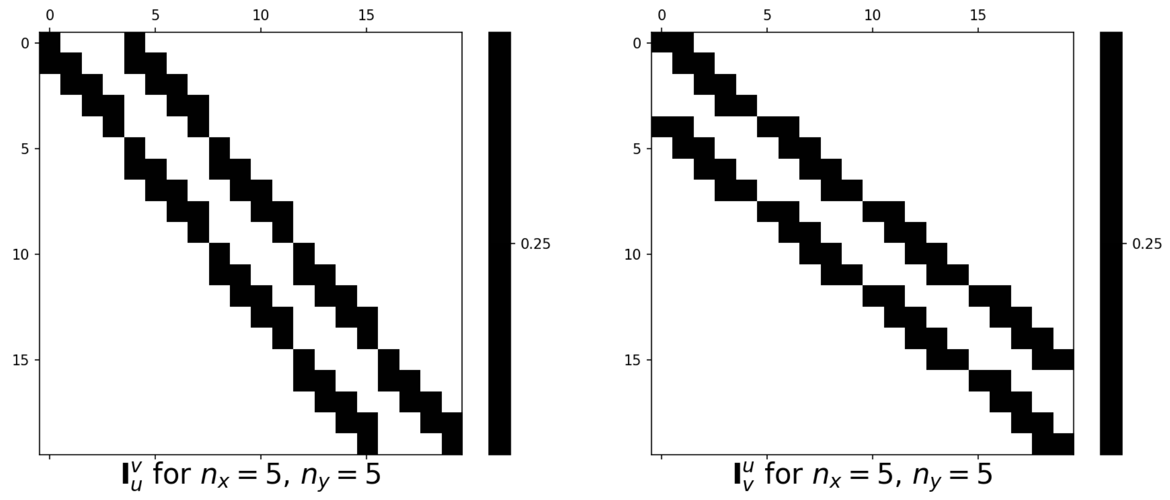

Figure 9: Same as Fig. 7, but for (left) $\mathbf{I}_u^v$ and (right) $\mathbf{I}_v^u$.

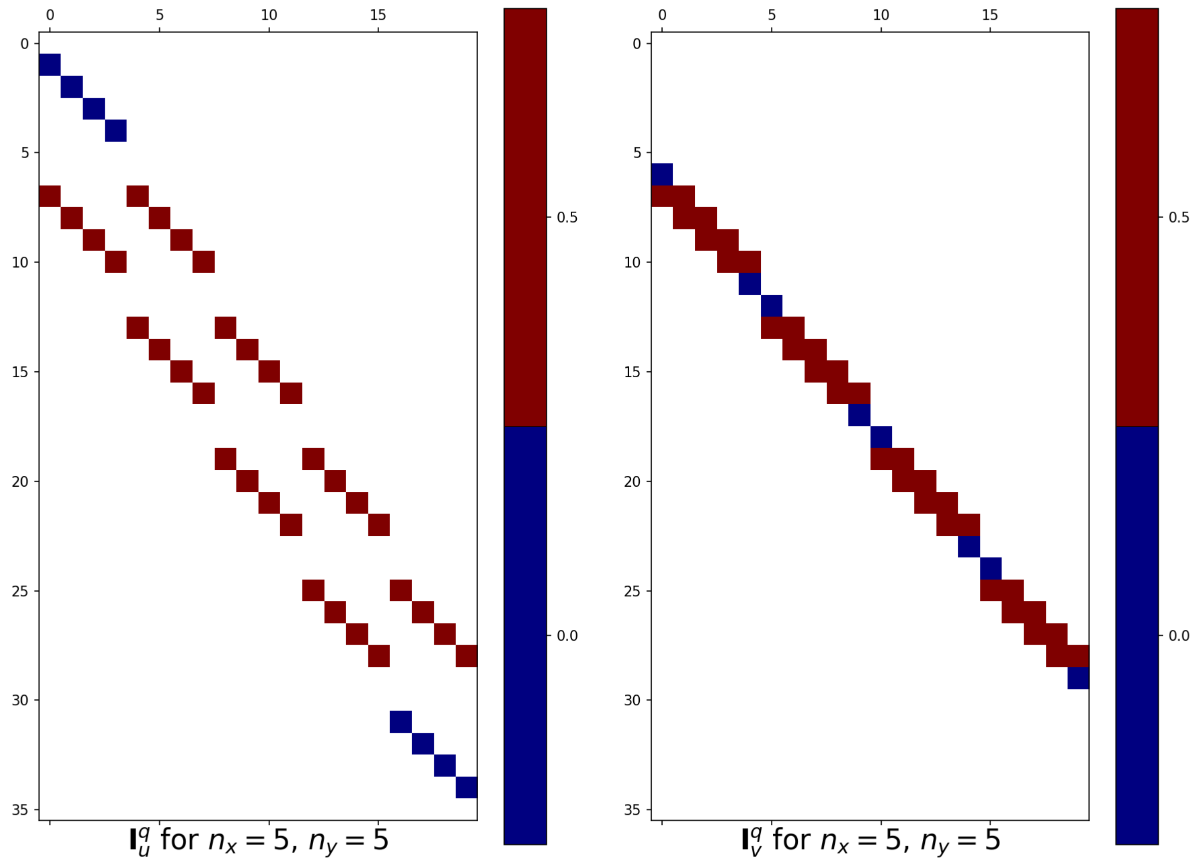

Figure 10: Same as Fig. 7, but for (left) $\mathbf{I}_u^q$ and (right) $\mathbf{I}_v^q$. The tangential boundary condition parameter is here $\alpha = 2$, which corresponds to the no-slip case. Replacing the marked entries with 0 by 1 yields free-slip boundary conditions ($\alpha = 0$).

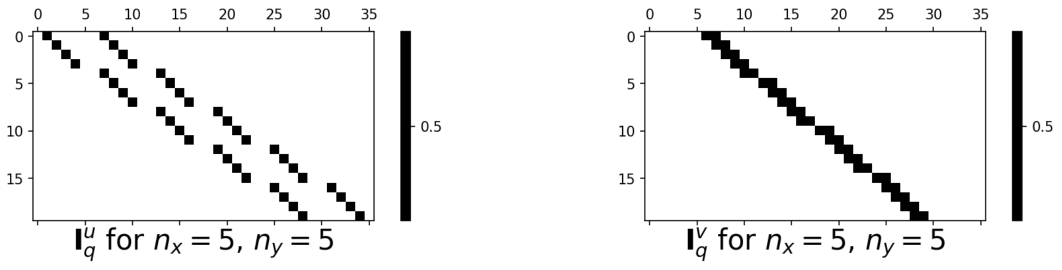

Figure 11: Same as Fig. 7, but for (left) $\mathbf{I}_q^u$ and (right) $\mathbf{I}_q^v$.

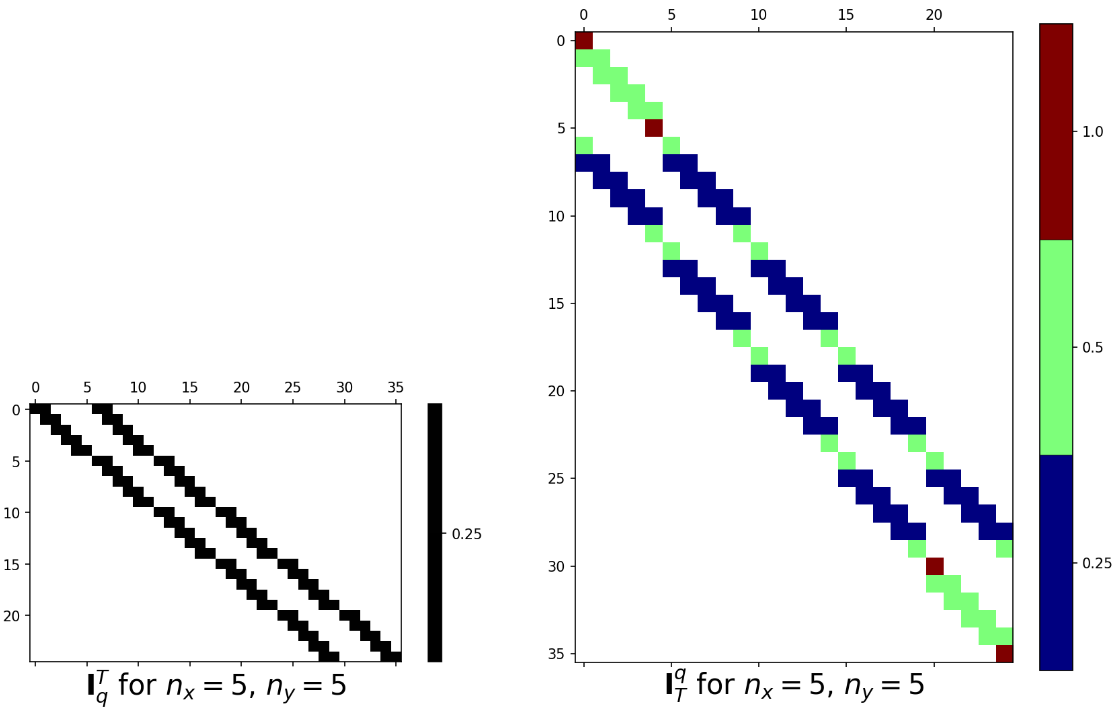

Figure 12: Same as Fig. 7, but for (left) $\mathbf{I}_q^T$ and (right) $\mathbf{I}_T^q$. For $\mathbf{I}_T^q$ the von-Neumann boundary conditions for h, i.e. $\nabla h = 0$, results in the increased values of 0.5 and 1 at the boundaries.

5 Figures

References

Arakawa, A and V R Lamb (1981). *A Potential Enstrophy and Energy Conserving Scheme for the Shallow Water Equations*. DOI: 10.1175/1520-0493(1981)109<0018:APEAEC>2.0.CO;2.

Arbic, Brian K. and Robert B. Scott (2008). "On Quadratic Bottom Drag, Geostrophic Turbulence, and Oceanic Mesoscale Eddies". In: *Journal of Physical Oceanography* 38.1, pp. 84–103. DOI: 10.1175/2007JPO3653.1.

Berloff, P S (2005). "Random-forcing model of the mesoscale oceanic eddies". In: *Journal of Fluid Mechanics* 529, pp. 71–95. DOI: 10.1017/S0022112005003393.

Cooper, F C and L Zanna (2015). "Optimisation of an idealised ocean model, stochastic parameterisation of sub-grid eddies". In: *Ocean Modelling* 88, pp. 38–53. DOI: 10.1016/j.ocemod.2014.12.014.

Gill, A E (1982). *Atmosphere-Ocean Dynamics*. Academic Press, p. 662.

Griffies, Stephen M and Robert W Hallberg (2000). "Biharmonic Friction with a Smagorinsky-Like Viscosity for Use in Large-Scale Eddy-Permitting Ocean Models". In: *Monthly Weather Review* 128.8, pp. 2935–2946. DOI: 10.1175/1520-0493(2000)128<2935:BFWASL>2.0.CO;2.

Madec, G (2016). "NEMO ocean engine". In: *Note du Pole de modelisation de l'Institut Pierre-Simon Laplace* 27.

Porta Mana, Pier Gian Luca and Laure Zanna (2014). "Toward a stochastic parameterization of ocean mesoscale eddies". In: *Ocean Modelling* 79, pp. 1–20. DOI: 10.1016/j.ocemod.2014.04.002.

Sadourny, Robert (1975). *The Dynamics of Finite-Difference Models of the Shallow-Water Equations*. DOI: 10.1175/1520-0469(1975)032<0680:TDOFDM>2.0.CO;2.

Salmon, Rick (2004). "Poisson-Bracket Approach to the Construction of Energy- and Potential-Enstrophy-Conserving Algorithms for the Shallow-Water Equations". In: *Journal of the Atmospheric Sciences* 61, pp. 2016–2036. DOI: 10.1175/1520-0469(2004)061<2016:PATTCO>2.0.CO;2.

— (2007). "A General Method for Conserving Energy and Potential Enstrophy in Shallow-Water Models". In: *Journal of the Atmospheric Sciences* 64.2, pp. 515–531. DOI: 10.1175/JAS3837.1.

Shchepetkin, Alexander F. and James J. O'Brien (1996). *A Physically Consistent Formulation of Lateral Friction in Shallow-Water Equation Ocean Models*. DOI: 10.1175/1520-0493(1996)124<1285:APCFOL>2.0.CO;2.

Appendices

A Indexed equations

$$\partial_x h|_{\mathbf{x}=\mathbf{x}^u_j} \approx \frac{h_{j+r+1} - h_{j+r}}{\Delta x}, \quad \text{with } r = \text{floor}(\frac{j-1}{n_x - 1}), \quad \forall\, j \in \{1, ..., N_u\} \tag{93}$$

$$\partial_y h|_{\mathbf{x}=\mathbf{x}_k^v} \approx \frac{h_{k+n_x} - h_k}{\Delta y}, \quad \forall\ k \in \{1, ..., N_v\} \tag{94}$$

$$\partial_x u|_{\mathbf{x}=\mathbf{x}_i} \approx \frac{1}{\Delta x} \begin{cases} u_i - 0 & \text{if } (i \bmod n_x) = 1 \\ 0 - u_{i-1} & \text{if } (i \bmod n_x) = 0 \\ u_i - u_{i-1} & \text{else.} \end{cases}, \quad \forall\ i \in \{1, ..., N_T\} \tag{95}$$

$$\partial_y v|_{\mathbf{x}=\mathbf{x}_i} \approx \frac{1}{\Delta y} \begin{cases} v_i - 0 & \text{if } i \le n_x \\ 0 - v_{i-n_x} & \text{if } i > N_y \\ v_i - v_{i-n_x} & \text{else.} \end{cases}, \quad \forall\ i \in \{1, ..., N_T\} \tag{96}$$

$$r = 3 + 2\,\text{floor}\left(\frac{l-1}{n_x+1}\right) \tag{97}$$

$$\partial_y u|_{\mathbf{x}=\mathbf{x}_l^q} \approx \frac{1}{\Delta y} \begin{cases} 0 & \text{if } l \bmod (n_x+1) \le 1 \\ \alpha u_{l-1} & \text{if } 1 < l \le n_x \\ -\alpha u_{l-r-n_x+1} & \text{if } N_q - n_x < l < N_q \\ u_{l-r} - u_{l-r-n_x+1} & \text{else.} \end{cases}, \quad \forall\ l \in \{1, ..., N_q\} \tag{98}$$

$$\mathtt{h}_u = h|_{\mathbf{x}=\mathbf{x}_j^u} \approx \frac{h_{j+r+1} + h_{j+r}}{2}, \quad \text{with } r = \text{floor}(\frac{j-1}{n_x-1}), \quad \forall\ j \in \{1, ..., N_u\} \tag{99}$$

$$\mathtt{h}_v = h|_{\mathbf{x}=\mathbf{x}_k^v} \approx \frac{h_{k+n_x} + h_k}{2}, \quad \forall\ k \in \{1, ..., N_v\} \tag{100}$$

$$\mathtt{u}_q = u|_{\mathbf{x}=\mathbf{x}_l^q} \approx \begin{cases} 0 & \text{if } l \bmod (n_x+1) \le 1 \\ (1-\frac{\alpha}{2})u_{l-1} & \text{if } 1 < l \le n_x \\ (1-\frac{\alpha}{2})u_{l-r-n_x+1} & \text{if } N_q - n_x < l < N_q \\ \frac{1}{2}(u_{l-r} + u_{l-r-n_x+1}) & \text{else.} \end{cases}, \quad \forall\ l \in \{1, ..., N_q\} \tag{101}$$

$\forall\ k \in \{1, ..., N_v\}$

$$\mathtt{u}_v = u|_{\mathbf{x}=\mathbf{x}_k^v} \approx \frac{1}{4} \begin{cases} (u_{k-r} + u_{k-r+n_x-1}) & \text{if } (k-1) \bmod n_x = 0 \\ (u_{k-r-1} + u_{k-r+n_x-2}) & \text{if } k \bmod n_x = 0 \\ (u_{k-r} + u_{k-r+n_x-1} + u_{k-r-1} + u_{k-r+n_x-2}) & \text{else.} \end{cases} \tag{102}$$

$$r = \text{floor}\left(\frac{k-1}{n_x}\right) \tag{103}$$

$$\mathtt{q}_T = q|_{\mathbf{x}=\mathbf{x}_i} \approx \frac{1}{4}(q_{i+r+1} + q_{i+r+n_x+2} + q_{i+r} + q_{i+r+n_x+1}), \quad \forall\ i \in \{1, ..., N_T\} \tag{104}$$

$$r = \operatorname{floor}\left(\frac{i-1}{n_x}\right) \tag{105}$$

$$qhv|_{\mathbf{x}=\mathbf{x}_j^u} \approx \tfrac{1}{24}(q_\alpha V^{\nearrow} + q_\beta V^{\searrow} + q_\gamma V^{\nwarrow} + q_\delta V^{\swarrow} + q_\lambda U^{\leftarrow} + q_\rho U^{\rightarrow}) \tag{106}$$

where

$$q_\alpha = 2q^{\nearrow} + 2q^{\downarrow} + q^{\uparrow} + q^{\searrow} \tag{107a}$$

$$q_\beta = 2q^{\uparrow} + 2q^{\searrow} + q^{\downarrow} + q^{\nearrow} \tag{107b}$$

$$q_\gamma = 2q^{\nwarrow} + 2q^{\downarrow} + q^{\uparrow} + q^{\swarrow} \tag{107c}$$

$$q_\delta = 2q^{\uparrow} + 2q^{\swarrow} + q^{\downarrow} + q^{\nwarrow} \tag{107d}$$

$$q_\lambda = q^{\uparrow} + q^{\nwarrow} - q^{\downarrow} - q^{\swarrow} \tag{107e}$$

$$q_\rho = q^{\downarrow} + q^{\searrow} - q^{\uparrow} - q^{\nearrow} \tag{107f}$$