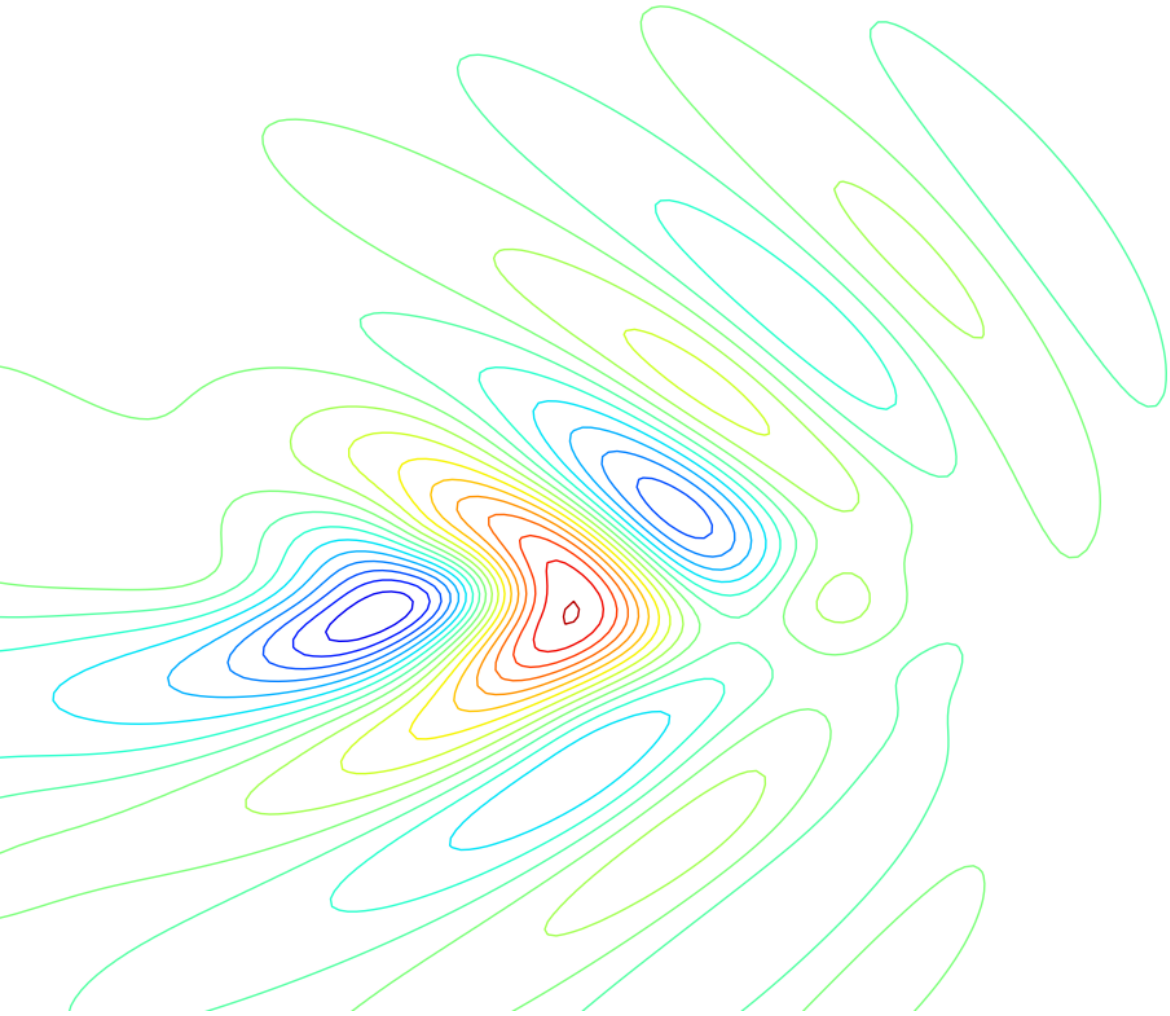

Draft version 1-d5

Clemens Spensberger

December 21, 2012

Bedymo: Concept and Realisation

Technical Report

Geophysical Institute
Faculty of Mathematics and Natural Sciences
University of Bergen

Draft version 1-d5
Bergen, December 21, 2012

Contents

Todo list

I. Introduction

1. Motivation and Background

The hemispheric asymmetry of the atmospheric general circulation is introduced by inhomogeneities of surface properties. Orography and localised heating are two examples for such inhomogeneities. In order to disentangle the observed combined effect of all of the earth's inhomogeneities, it is crucial to understand how the effect of a single idealised inhomogeneity affects the global circulation (*Held*, 2005). This understanding also permits predictions of how the circulation on the northern hemisphere will react, e.g. to a reduction of Arctic sea–ice cover (*Stroeve et al.*, 2007; *Serreze et al.*, 2007). The observed and projected loss of sea–ice due to climate change constitutes a new large–scale source of heat which presumably impacts the large–scale circulation.

The effects of orography, that is usually idealised as an isolated Gaussian mountain, have been examined by dynamical meteorologists for the last decades. A linearised version of this system has received special attention, as it is easy enough to be solved analytically but still is in reasonable agreement with observations (*Smith*, 1980; *Hoskins and Karoly*, 1981). In physical terms, by linearising the system one assumes that the approaching air masses flow over the mountain without deviation from the background flow. The limitations of the linear model and the transition from linear towards non–linear regimes was subject to several studies (*Cook and Held*, 1992; *Ting and Yu*, 1998; *Wang and Kushner*, 2010). Although they argue for a more gradual transition between the linear and the non–linear regimes *Ringler and Cook* (1997) maintain the concept of a critical mountain height that separates the two regimes. Additionally *Ringler and Cook* (1997) examine the dependence of the critical height on pertinent meteorological parameters. Focusing more on the non–linear impacts, *Valdes and Hoskins* (1991) found, that the inclusion of non–linear terms in the lower boundary conditions can significantly alter the results.

The non–linearity enters the model system via the advection term. It allows the eddies to influence themselves by their respective wind field deviations. This self–advection gives rise to phenomena like a deviation of the streamlines around the mountain, total blocking of some air masses or the emergence of a secondary circulation around the mountain. It was suggested, that the Himalayas also act as a divide between air masses of very different properties (*Boos and Kuang*, 2010). This would also be a non–linear effect that is very closely related to orographic blocking.

All studies referred to above focus on the steady–state response of the atmospheric circulation. However, the transient circulation will also adapt to a newly included surface inhomogeneity, changing for example the location and intensity of storm tracks. The location of these tracks will in turn drastically influence the local climates in the mid–latitudes through the prevailing wind and precipitation patterns. Much less work has been

conducted on this topic, and for example the mid–winter depression of the intensity of the Pacific storm track still remains to be explained (*Chang et al.*, 2002). Recent studies on this topic investigate the influence of the jet stream's structure (*Son et al.*, 2009) and the influence of different mountain ranges and plateaus in the vicinity of the Himalayas (*Park et al.*, 2010).

This result of the topographic blocking on both the mean and the transient circulation has not been accounted for in any study the applicant is aware of. It is the main reason behind the development of Bedymo. In particular the main open questions that were kept in mind when drafting the model's basic concepts that are laid out in the following section are:

1. What criterion (if any) separates the linear from the non–linear response to an isolated surface feature or inhomogeneity?
2. How important is topographic blocking compared to other non-linear effects? Can the role of mountains as air–mass divides be accounted for in linear theories?
3. How can changes in the transient circulation due to named inhomogeneity be described? How does the transient circulation influence the mean circulation in these cases?
4. What are the governing processes causing the mid–winter depression of the intensity of the Pacific storm track?

2. Basic concepts

- Solution of the governing equations in grid–point space to allow for explicit blocking.
- Model of the free atmosphere: Lower boundary is set to the top of the planetary boundary layer.
- Two conceptual parts:
  - Lower part: Maximum one model layer, where explicit blocking is allowed
  - Upper part: One or more model layers in terrain–following coordinates
- Configurable set of simplifications around quasi–geostrophy
- Application in the mid- and high latitudes

II. Model physics

3. Fundamental equations

Starting with the primitive equations

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -2\boldsymbol{\Omega} \times \boldsymbol{u} - \frac{1}{\rho}\nabla p + \boldsymbol{g} + \boldsymbol{F}_r \tag{3.1}$$

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \boldsymbol{u}) = 0 \tag{3.2}$$

$$c_v \left(\frac{\partial}{\partial t} + \boldsymbol{u} \cdot \nabla \right) T + p \left(\frac{\partial}{\partial t} + \boldsymbol{u} \cdot \nabla \right) \alpha = J \tag{3.3}$$

the so-called metric terms appear when changing to a local Cartesian coordinate system on a sphere. Using the transformation

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{i}\frac{du}{dt} + \boldsymbol{j}\frac{dv}{dt} + \boldsymbol{k}\frac{dw}{dt} + u\frac{d\boldsymbol{i}}{dt} + v\frac{d\boldsymbol{j}}{dt} + w\frac{d\boldsymbol{k}}{dt} \tag{3.4}$$

along with the following expressions for the total derivatives of the unit vectors $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$

$$u\frac{d\boldsymbol{i}}{dt} = u \cdot u\frac{\partial \boldsymbol{i}}{\partial x} = \frac{u^2}{a\cos\varphi}\frac{d\boldsymbol{i}}{d\lambda} = \frac{u^2}{a\cos\varphi}(\boldsymbol{j}\sin\varphi + \boldsymbol{k}\cos\varphi) \tag{3.5}$$

$$v\frac{d\boldsymbol{j}}{dt} = v\left(u\frac{\partial \boldsymbol{j}}{\partial x} + v\frac{\partial \boldsymbol{j}}{\partial y} \right) = \frac{uv}{a\cos\varphi}\frac{\partial \boldsymbol{j}}{\partial \lambda} + \frac{v^2}{a}\frac{\partial \boldsymbol{j}}{\partial \varphi} = \frac{uv}{a\cos\varphi}(\boldsymbol{i}\sin\varphi) + \frac{v^2}{a}\boldsymbol{k} \tag{3.6}$$

$$w\frac{d\boldsymbol{k}}{dt} = w\left(u\frac{\partial \boldsymbol{k}}{\partial x} + v\frac{\partial \boldsymbol{k}}{\partial y} \right) = \frac{uw}{a\cos\varphi}\frac{\partial \boldsymbol{k}}{\partial \lambda} + \frac{vw}{a}\frac{\partial \boldsymbol{k}}{\partial \varphi} = \frac{uw}{a\cos\varphi}(\boldsymbol{i}\cos\varphi) + \frac{vw}{a}\boldsymbol{j} \tag{3.7}$$

yields the equations of motion:

$$\frac{\partial u}{\partial t} + \boldsymbol{u} \cdot \nabla u + \frac{uv}{a}\tan\varphi + \frac{uw}{a} = fv - f'w - \frac{1}{\rho}\frac{\partial p}{\partial x} + F_{rx} \tag{3.8}$$

$$\frac{\partial v}{\partial t} + \boldsymbol{u} \cdot \nabla v + \frac{u^2}{a}\tan\varphi + \frac{vw}{a} = -fu - \frac{1}{\rho}\frac{\partial p}{\partial y} + F_{ry} \tag{3.9}$$

$$\frac{\partial w}{\partial t} + \boldsymbol{u} \cdot \nabla w + \frac{u^2 + v^2}{a} = f'w - \frac{1}{\rho}\frac{\partial p}{\partial z} + F_{rz} - g \tag{3.10}$$

3.1. Momentum equations in z and p-coordinates

Scale analysis of the vertical momentum equation for synoptic scale flow in the mid-latitudes leads to the well–known hydrostatic approximation. In the horizontal, only terms that are $\mathcal{O}(Ro)$ or larger have been kept. The scale analysis showst that for the geometry of the earth, the metric terms are generally negligible.

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{v} + f\boldsymbol{k} \times \boldsymbol{v} = -\frac{1}{\rho}\nabla p + D\nabla^2 \boldsymbol{v} \tag{3.11}$$

$$\frac{\partial p}{\partial z} = -\rho g \tag{3.12}$$

Exploiting the hydrostatic approximation allows the derivation of some relations

$$\phi = gz \tag{3.13}$$

$$\frac{\partial}{\partial x}\Big|_p = \frac{\partial}{\partial x}\Big|_z + \frac{\partial z}{\partial x}\Big|_p \frac{\partial}{\partial z} \tag{3.14}$$

$$\longrightarrow 0 = \frac{\partial p}{\partial x}\Big|_z + \frac{\partial z}{\partial x}\Big|_p \frac{\partial p}{\partial z} = \frac{1}{g}\frac{\partial \phi}{\partial x}\Big|_p (-\rho g) \tag{3.15}$$

$$\longrightarrow -\frac{1}{\rho}\nabla_z p = -\nabla_p \phi \tag{3.16}$$

that allow the transformation of the vertical coordinate from z to p. Using those relations, the momentum equations in p-coordinates results to

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{v} + f\boldsymbol{k} \times \boldsymbol{v} = \nabla_p \phi + D\nabla_p^2 \boldsymbol{v} \tag{3.17}$$

$$\frac{\partial \phi}{\partial p} = -\frac{RT}{p} \tag{3.18}$$

A similar form of the momentum equations can be derived also in z coordinates, using the Boussinesq approximation. In this approximation the density field is partitioned into a stationary basic state ρ_0 and a pertubation ρ'. Both the corresponding basic state pressure p_0 and the pressure pertubations p' are assumed to be in hydrostatic balance.

$$\rho = \rho_0(z) + \rho'(x, y, z, t) \tag{3.19}$$

$$p = p_0(z) + p'(x, y, z, t) \tag{3.20}$$

The geopotential pertubation corresponding to p' can be defined as $\phi' = \frac{1}{\rho_0}p'$. With these relations

$$\frac{1}{\rho}\nabla_z p \approx \frac{1}{\rho_0}\nabla_z p' = \nabla_z \phi' \quad , \tag{3.21}$$

$$\longrightarrow \frac{\partial \phi'}{\partial z} = \frac{1}{\rho_0}\frac{\partial p'}{\partial z} - \frac{p'}{\rho_0^2}\frac{\partial \rho_0}{\partial z} \approx g\frac{\rho'}{\rho_0} = b' \quad . \tag{3.22}$$

Why neglect the second term?

The b' is an expression of the buoyancy pertubation. The approximated momentum equa-

tions in z coordinates hence read

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{u} \cdot \nabla_z \boldsymbol{v} + f\boldsymbol{k} \times \boldsymbol{v} = \nabla_z \phi' + D\nabla_z^2 \boldsymbol{v} \quad , \tag{3.23}$$

$$\frac{\partial \phi'}{\partial z} = b' \quad . \tag{3.24}$$

$$\tag{3.25}$$

3.2. Continuity equation

In pressure coordinates the continuity equations is just

$$\nabla \cdot \boldsymbol{u} = \nabla_p \cdot \boldsymbol{v} + \frac{\partial \omega}{\partial p} = 0 \quad . \tag{3.26}$$

Following again the Boussinesq approxmation in z-coordinates simplifies the continuity by replacing density by the basic state density $\rho_0 = \rho_0(z)$.

$$\nabla \cdot \boldsymbol{u} + \frac{w}{\rho_0}\frac{\partial \rho_0}{\partial z} = \nabla_z \cdot \boldsymbol{v} + \frac{1}{\rho_0}\frac{\partial}{\partial z}(\rho_0 w) = 0 \tag{3.27}$$

3.3. Thermodynamic equation

The conversion of the thermodynamic equation needs a little more work:

$$c_p \frac{dT}{dt} - \alpha \frac{dp}{dt} = J \tag{3.28}$$

$$\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla_h\right) T + \left(\frac{\partial T}{\partial p} - \frac{RT}{c_p p}\right)\omega = \frac{J}{c_p} \tag{3.29}$$

$$\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla_h\right) T + \frac{\partial \theta}{\partial p}\frac{T}{\theta}\omega = \frac{J}{c_p} \quad \text{with } \theta \equiv T\left(\frac{p_0}{p}\right)^{R/c_p} \tag{3.30}$$

$$\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla_h\right) T - N^2 \frac{RT^2}{pg^2}\omega = \frac{J}{c_p} \quad \text{with } N^2 \equiv \frac{g}{\theta}\frac{\partial \theta}{\partial z} \tag{3.31}$$

$$-\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla_h\right)\frac{\partial \phi}{\partial p} - \frac{N^2}{g^2}\left(\frac{\partial \phi}{\partial p}\right)^2 \omega = \frac{RJ}{c_p p} \quad \text{with } T = -\frac{p}{R}\frac{\partial \phi}{\partial p} \tag{3.32}$$

The derivation in z-coordinates works much along the same lines. The main difference is that pressure and Temperature are deviations from a basic state:

$$c_p \frac{dT'}{dt} - \alpha_0 \frac{dp'}{dt} = J \tag{3.33}$$

$$\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla_h\right) T' + \left(\frac{\partial T'}{\partial z} - \frac{g}{c_p}\right) w = \frac{J}{c_p} \quad \text{with } \omega \approx -\rho_0 g w \tag{3.34}$$

$$\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla_h\right) T' - N^2 \frac{\theta_0}{g}\omega = \frac{J}{c_p} \tag{3.35}$$

$$\left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla_h\right)\frac{\partial \phi'}{\partial z} - N^2 \omega = \frac{gJ}{c_p \theta_0} \quad \text{with } b' = g\frac{\rho'}{\rho_0} \approx g\frac{T'}{\theta_0} \tag{3.36}$$

3.4. Vorticity equation

Cross–differentiating the momentum equations along the definition of the vorticity $\zeta = \partial v/\partial x - \partial u/\partial y$

$$\begin{aligned}\frac{\partial}{\partial x}\left(\frac{dv}{dt}\right) - \frac{\partial}{\partial y}\left(\frac{du}{dt}\right) =& \frac{\partial \zeta}{\partial t} + \boldsymbol{u}\cdot\nabla\zeta \\ &+ \left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} - \frac{\partial u}{\partial x}\frac{\partial u}{\partial y}\right) + \left(\frac{\partial v}{\partial x}\frac{\partial v}{\partial y} - \frac{\partial v}{\partial y}\frac{\partial u}{\partial y}\right) \\ &+ \left(\frac{\partial \dot\eta}{\partial x}\frac{\partial v}{\partial \eta} - \frac{\partial \dot\eta}{\partial y}\frac{\partial u}{\partial \eta}\right) + \beta v + f\frac{\partial \dot\eta}{\partial \eta} + D(\eta)\nabla_\eta^2\zeta = 0\end{aligned} \tag{3.37}$$

yields a prognostic equation for the vorticity in both the $p-$ and $z-$system. Since the resulting equations in $p-$ and $z-$coordinates only differ by a constant, the following equations are written with a generalised vertical coordinate η which can be interpreted as both p or z. Vertical velocity is denoted $\dot\eta$. In the following, the constants differing between the coordinate systems are denoted as a function of η, here only $D(\eta)$.

Using vector notation and neglecting vertical advection, this formula can be shortened to

$$\frac{\partial \zeta}{\partial t} + \boldsymbol{v}\cdot\nabla_h\zeta + (f+\zeta)\frac{\partial \dot\eta}{\partial \eta} + \boldsymbol{k}\cdot\left(\frac{\partial \boldsymbol{v}}{\partial \eta}\times\nabla_h\dot\eta\right) + \beta v + D(\eta)\nabla_\eta^2\zeta = 0 \quad . \tag{3.38}$$

The numerical precision can be improved by casting this equation from its advective into its flux-divergence form.

$$\frac{\partial \zeta}{\partial t} + \nabla_h\cdot(\boldsymbol{v}\zeta) + f\frac{\partial \dot\eta}{\partial \eta} + \boldsymbol{k}\cdot\left(\frac{\partial \boldsymbol{v}}{\partial \eta}\times\nabla_h\dot\eta\right) + \beta v + D(\eta)\nabla_\eta^2\zeta = 0 \quad . \tag{3.39}$$

4. A hierarchy of simplifications for the free atmosphere

The above equations may be written in terms of the stream function ψ and the velocity potential χ (keeping only the vertical velocity ω) by using the relations

$$u \equiv \frac{\partial \chi}{\partial x} - \frac{\partial \psi}{\partial y} \quad , \tag{4.1}$$

$$v \equiv \frac{\partial \chi}{\partial y} + \frac{\partial \psi}{\partial x} \quad , \tag{4.2}$$

$$\zeta = \nabla_h^2\psi \quad \text{and} \tag{4.3}$$

$$\frac{\partial \phi}{\partial \eta} = f_0\frac{\partial \psi}{\partial \eta} \quad \text{for } \eta \in \{p, z\} \quad . \tag{4.4}$$

The same generalised notation using η and $\dot\eta$ for the vertical coordinate is used as in the derivation of the vorticity tendency.

The vorticity equation, the thermodynamic equation and continuity then read

$$\left(r + \frac{\partial}{\partial t}\right) \nabla^2\psi + \nabla_\eta(\boldsymbol{v}\nabla^2_\eta\psi) + f_0\mathcal{D} + \boldsymbol{k}\cdot\left(\frac{\partial \boldsymbol{v}}{\partial \eta}\times\nabla\dot{\eta}\right) + \beta v + D(\eta)\nabla^4\psi = 0 \tag{4.5}$$

$$\left(r + \frac{\partial}{\partial t} + \boldsymbol{v}\cdot\nabla_\eta\right)\frac{\partial\psi}{\partial\eta} + N^2C(\eta)\dot{\eta} = Q(\eta) \tag{4.6}$$

$$-\frac{\partial\dot{\eta}}{\partial\eta} = \nabla^2\chi = \mathcal{D} \tag{4.7}$$

What about $\frac{\partial p_s}{\partial t}$? Why zero? Prognostic equation?

The only difference between the coordinate systems is in C, D and Q:

$$C(\eta = p) = \frac{f_0}{g^2}\left(\frac{\partial\psi}{\partial p}\right)^2 \quad , \qquad Q(\eta = p) = -\frac{RJ}{f_0 c_p p} \tag{4.8}$$

$$C(\eta = z) = -\frac{1}{f_0} \quad , \qquad Q(\eta = z) = \frac{gJ}{f_0 c_p \theta_0} \tag{4.9}$$

This system is the semi–geostrophic (SG) system. In comparison to the standard QG–system, the full advection terms have been kept, resulting in the vorticity equation containing a tilting term and the absolute vorticity $f_0 + \nabla^2\psi$ instead of only the planetary vorticity f_0 in the streching term.

The simplified standard quasi–geostrophic equation system is obtained by omitting ageostrophic advection. As a consequence $\chi = 0$ and everything can be expressed in terms of the stream function.

$$\left(r + \frac{\partial}{\partial t}\right) \nabla^2_\eta\psi + \nabla_\eta(\boldsymbol{v}_g\nabla^2_\eta\psi) + f\mathcal{D} + \beta\frac{\partial\psi}{\partial x} + D(\eta)\nabla^4_\eta\psi = 0 \tag{4.10}$$

$$\left(r + \frac{\partial}{\partial t}\right)\frac{\partial\psi}{\partial\eta} + \nabla_\eta(\boldsymbol{v}_g\frac{\partial\psi}{\partial\eta}) + N^2C(\eta)\dot{\eta} = Q(\eta) \tag{4.11}$$

$$\chi = 0, \quad -\frac{\partial\dot{\eta}}{\partial\eta} = \mathcal{D} \tag{4.12}$$

To linearise the system, only the deviations from a stationary basic state Ψ_0 and the corresponding basic state winds $\boldsymbol{v}_{g0}$ are considered:

$$\left(r + \frac{\partial}{\partial t}\right) \nabla^2_\eta\psi + \nabla_\eta(\boldsymbol{v}_{g0}\nabla^2_\eta\psi) + f\mathcal{D} + \beta\frac{\partial\psi}{\partial x} + D(\eta)\nabla^4_\eta\psi = 0 \tag{4.13}$$

$$\left(r + \frac{\partial}{\partial t}\right)\frac{\partial\psi}{\partial\eta} + \nabla_\eta(\boldsymbol{v}_{g0}\frac{\partial\psi}{\partial\eta}) + N^2C(\eta)\dot{\eta} = Q(\eta) \tag{4.14}$$

$$\chi = 0, \quad -\frac{\partial\omega}{\partial p} = \mathcal{D} \tag{4.15}$$

5. Modifications in the blocking layer

What are the modifications?

Not determined yet.

III. Model numerics

6. Grid organisation

The vertical grid is organised as a generalised version of the 2–layer model used to study baroclinic instability. In this setup the vertical velocity components $\dot{\eta}$ are evaluated in between the standard model levels, yielding the staggered grid in figure 6.1. Additionally, the vertical velocities at the bottom and at the top of the model domain are set by boundary conditions (sec. 11). The vertical grid indexing starts at the top with the number `1`. $\dot{\eta}$–levels bear the same index as the standard model level below.

The vertical grid does not depend on the orography and potentially cuts through the land surface. The problems arising from this cut are discussed physically in section 5.

In the horizontal, a rectangular Cartesian Arakawa-C grid is employed. Analogous to the vertical dimension, the locations where each horizontal velocity component is defined are moved by $\Delta x/2$ against coordinate direction relative to the scalar grid point.

The grid point indexes for all scalars run from `1` to `nx` (or `ny`, respectively). The necessary number of boundary values depends on the model configuration. For the standard advection scheme three values are needed. Their indexes are added outside the model domain (e.g. indexes `-2` until `0` and `nx+1` (`ny+1`) until `nx+3` (`ny+3`)). WIP

In contrast to many other non–global meteorological models, the model domain does not need to be rectangular. The model boundary may be set (almost) freely along grid cell boundaries. The only limitation is that every exclusion from the model domain must be at least two grid cells "thick". This limitation is necessary to avoid that a grid cell outside the model domain is connected to the model domain via two opposing boundaries. In this case it would be impossible to set correct boundary conditions for the opposing boundaries.

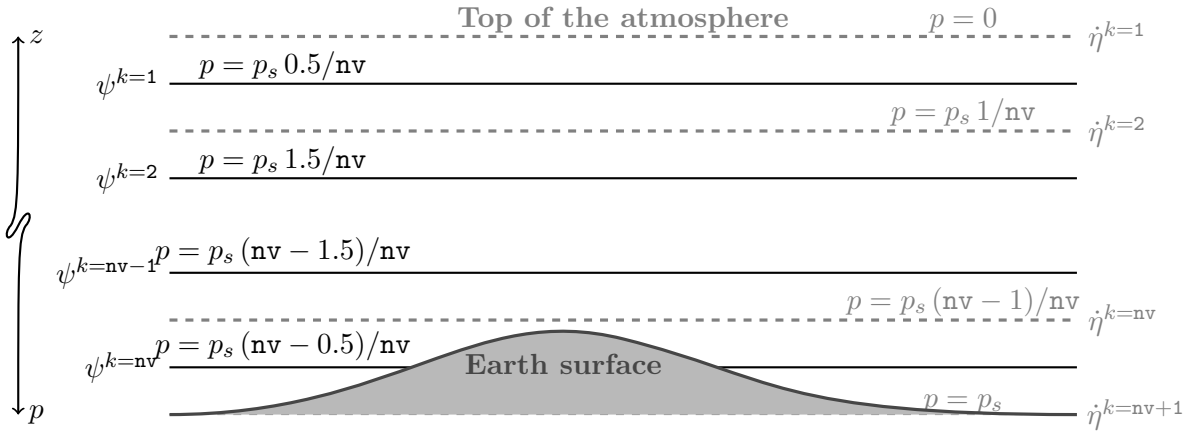

Figure 6.1.: Vertical grid staggering and indexing. `nv` denotes the number of vertical levels. The pressure levels are equally spaced between the standard pressure p_s and the Top of the Atmosphere, as indicated by the formulae. The calculation of the height-levels works analogously, with z increasing from zero to the height of the rigid lid at `ztop`.

7. Time integration scheme

There are several time integration schemes implemented in Bedymo. All of them can be formulated as a series of the basic explicit Euler steps. Hence, they perform one or more of these basic integration steps for advancing one model time step. The available schemes are:

ID= 1 4^{th}-order Runge-Kutta

ID= 2 Miller-Pearce

ID= 3 Purely quasi-implicit

ID= 4 Leap-frog with RAW filter (recommended)

ID= 5 Purely explicit Euler

These schemes are all implemented by one generic Eulerian–integration method, which uses the three time levels setup by the different schemes. These time levels are (1) the old state of the prognostic variable t_o used in the time discretisation (2) the intermediate state of all variables t_i used everywhere but in the time discretisation and (3) the new state of the prognostic variable t_n that is now being calculated. As visible from table 7.1, the old and intermediate time level is always identical for the purely explicit scheme (accordingly for a implicit scheme the intermediate level would always be equal to the new time level). The recommended Miller–Pearce scheme increases accuracy by integrating quasi–implicitly every second time step. In these cases a new intermediate state is calculated by a normal explicit time step. Subsequently this result is used to recalculate the same time step approximately implicit.

The Leap frog is stabilised by the Robert-Asselin filter modified as suggested by *Williams* (2010). The filter is applied to the vorticity, to the stream function and to all tracers.

Table 7.1.: Indices t_o, t_i and t_n for a) explicit, b) the Miller–Pearce, c) the Leap Frog and d) 4^{th}-order Runge Kutta time integration scheme. Letters denote a temporary storage for a sub–time step.

a) Time step #	t_o	t_i	t_n
1	0	0	1
2	1	1	2
3	2	2	3
4	3	3	4
5	4	4	5
6	5	5	6
7	6	6	7
8	7	7	8
9	8	8	9

b) Time step #	t_o	t_i	t_n
1	0	0	1
	1	1	x
2	1	x	2
3	2	2	3
	3	3	x
4	3	x	4
5	4	4	5
	5	5	x
6	5	x	6

c) Time step #	t_o	t_i	t_n
1	0	0	1
2	0	1	2
3	1	2	3
4	2	3	4
5	3	4	5
6	4	5	6
7	5	6	7
8	6	7	8
9	7	8	9

d) Time step #	t_o	t_i	t_n
	0	0	x
	0	x	y
	0	y	z
1	0	x-z	1
	1	1	x
	1	x	y
	1	y	z
2	1	x-z	2
	2	2	x

8. Advection scheme

In BEDYMO the 1^{st}-order to 6^{th}-order upstream advection schemes are implemented. With increasing order, the numerical diffusion and phase errors are reduced. All schemes can be expressed by the generic expression

$$\frac{\partial}{\partial x}(u\chi) \approx \frac{1}{\Delta x}\Big(\max(u,0)\chi^+\big|_{i+0.5} + \min(u,0)\chi^-\big|_{i+0.5} - \max(u,0)\chi^+\big|_{i-0.5} - \min(u,0)\chi^-\big|_{i-0.5}\Big) \quad . \tag{8.1}$$

The discretised version of the derivatives χ^+ and χ^- depends on the order of the scheme. The separation between positive and negative wind speeds which reflects the upwind-bias, is only effective for odd-numbered schemes. For even-numbered schemes $\chi^+ = \chi^-$. Equivalent expressions as presented here for the x-direction also hold for advection in y-direction. The ID of the respective scheme is equivalent to the order of the scheme.

The odd-numbered schemes are summarised in table 8.1.

In multi–dimensional advection, the Taylor expansion on which the derivation of these schemes is based would contain many cross derivatives along several dimensions. As the amount of data points used in the computational stencil would increase with the order of the scheme to the power of the flow dimension, this technique is soon getting unfeasibly complex to compute and derive. An elegant way to circumvent this problem in multi–dimensional advection is using the time–splitting technique (*Smolarkiewicz*, 1982; *Tremback et al.*, 1987). Following this technique, the one–dimensional advection is applied separately for every dimension.

The derivation of the schemes is given in the appendices A and B.

Table 8.1.: Overview over the first four upwind–biased spatial discretisation schemes

Order	$\chi^+\big\vert_{i+0.5}$	$\chi^-\big\vert_{i+0.5}$
1^{st}	χ_i	χ_{i+1}
2^{nd}	$\frac{\chi_i + \chi_{i+1}}{2}$	$\frac{\chi_i + \chi_{i+1}}{2}$
3^{rd}	$\frac{2\chi_{i+1} + 5\chi_i - \chi_{i-1}}{6}$	$\frac{-\chi_{i+2} + 5\chi_{i+1} + 2\chi_i}{6}$
4^{th}	$\frac{-1\chi_{i+2} + 7\chi_{i+1} + 7\chi_i - \chi_{i-1}}{12}$	$\frac{-1\chi_{i+2} + 7\chi_{i+1} + 7\chi_i - \chi_{i-1}}{12}$

9. Laplace–inversion scheme

To retrieve the stream function from a given vorticity field, the equation $\nabla^2\psi = \zeta$ must be solved for ψ. The Laplacian must accordingly be inverted to allow this. There are several algorithms proposed that accomplish this for different sets of assumptions. For BEDYMO the stabilised version of the Bi-Conjugate Gradient (Bi-CGstab) method (*van der Vorst*, 1992; *Munksgaard*, 1980) has proven to yield good and fast results.

To accelerate the convergence process further, a preconditioning technique may be applied. In mathematical terms, the all preconditioning methods aim to level out the magnitudes of the eigenvalues of the matrix that is being inverted. With all eigenvalues exactly equal, the convergence would be exact after only one iteration. In BEDYMO the only ILU(0)-preconditioning is implemented and used.

$x_0 \leftarrow 0$
$r \leftarrow b - Ax_0$
$r_0 \leftarrow r$
$\rho_o \leftarrow \alpha \leftarrow \omega_0 \leftarrow 1$
$v_0 \leftarrow p_0 \leftarrow 0$
for $i \leftarrow 1, 2, 3, \ldots$ **do**
  $\rho_n \leftarrow < r_0, r >$
  if $\rho_n < \rho_{max}$ **then**
    return
  end if
  $\beta \leftarrow \frac{\alpha}{\omega}\frac{\rho_n}{\rho_o}$
  $p \leftarrow r + \beta(p - \omega v)$
  $v \leftarrow Ap$
  $\alpha \leftarrow \frac{\rho_n}{<r_0, v>}$
  $s \leftarrow r - \alpha v$
  $t \leftarrow As$
  $\omega \leftarrow \frac{<t,s>}{<t,t>}$
  $x \leftarrow x + \alpha p + \omega s$
  $r \leftarrow s - \omega t$
  $\rho_o \leftarrow \rho_n$
end for

Figure 9.1.: The unpreconditioned Bi-CGstab algorithm as implemented in BEDYMO.

10. Lateral boundary conditions

There are currently four boundary types implemented in BEDYMO. They can be set independently for the eastern/western boundaries and the southern/northern boundaries. Their respective mathematical formulation is summarised in table 10.1. In physical terms they might be characterised as follows:

ID= −1 Allows eddies and waves to propagate through the boundary, reentering the model domain on the opposite side. Is most useful to represent shape of a circumglobal latitude band as a model domain.

ID= 0 As the stream function is set to zero within the complete boundary, both the boundary–parallel and the boundary–normal wind component vanish. However, there might be a strong gradient of the stream function near the boundary, resulting in strong boundary-parallel winds close to the boundary. This condition represents a wall with a "no-slip"-condition placed directly at the first boundary grid point.

ID= 1 While the gradient of the stream function normal to the boundary still vanishes with this condition, the boundary–parallel gradient is constant. Consequently, the boundary–parallel are implicitly set to zero, while the boundary–normal winds stay constant. This allows eddies and waves to propagate outside the model domain. Waves might however also (at least partially) be reflected on the boundary.

ID= 2 The constant gradient condition for the stream function yields a constant wind at the boundary.

How do waves behave with this condition? Explanation?

As not all combinations of boundary types yield physically meaningful results, the following list gives a short explanation to a few recommended configurations:

All boundaries= −1 Double–periodic domain for idealised model setups.

All boundaries= 0 A closed box setup for idealised model studies.

East–West= −1 / North–South= 1 Resembles a periodic latitude channel, where waves may propagate freely through the northern and southern boundaries.

The temperature $\frac{\partial \psi}{\partial \eta}$ and vertical wind velocity $\dot{\eta}$ is calculated directly from the stream function and vorticity fields. For this reason, no explicit boundary condition is needed for these variables.

Table 10.1.: Implemented boundary conditions and their ID

ID	Condition for ζ	Condition for ψ	Condition for $\frac{\partial \psi}{\partial T}$
−1	by ψ (periodic)	periodic	by ψ (periodic)
0	by ψ (zero)	zero	by ψ (zero)
1	by ψ	constant	by ψ
2	by ψ	constant gradient	by ψ

11. Lower and upper boundary conditions

The upper boundary is considered a rigid lid, such that $\dot{\eta}^{k=1} = 0$. At the lower boundary, the wind follows the orography, which can be expressed as

$$\dot{\eta}^{k=\mathtt{nv}+1} = \frac{1}{2}\left(u^{i+0.5} + u^{i-0.5}\right)\frac{\partial \eta_s}{\partial x} + \frac{1}{2}\left(v^{j+0.5} + v^{j-0.5}\right)\frac{\partial \eta_s}{\partial y} \tag{11.1}$$

Settle modifications for the blocking layer and adapt as necessary.

The slopes of the orography η_s is determined with high accuracy during the initialisation of the model.

For the vorticity ζ and the stream function ψ no explicit boundary conditions are required.

12. Integration sequence

1. Calculate all terms contributing to the vorticity tendency and integrate the vorticity to the new time level
2. Set boundary conditions for the vorticity
3. Infer the new stream function from the new vorticity by inverting the Laplacian
4. Diagnostically calculate new vertical wind
5. Diagnose the divergence $\partial\dot{\eta}/\partial\eta$, and in Semi-Geostrophy infer the velocity potential
6. Diagnostically calculate new horizontal winds

13. Discretised equations

Following the integration sequence presented above, all bits and pieces laid out in the previous sections of this chapter will be combined with the model equations to construct the discretised equations as they are solved in BEDYMO. The calculations for a time step begin with the prediction of the new vorticity field.

13.1. Prognosticating vorticity

The solution technique is explained here for the most complex version of the vorticity equation (4.5), the one derived in the semi-geostrophic framework. For convenience it is repeated here:

$$\left(r + \frac{\partial}{\partial t}\right)\nabla^2\psi + \nabla_\eta(\boldsymbol{v}\nabla^2_\eta\psi) + f_0\mathcal{D} + \boldsymbol{k}\cdot\left(\frac{\partial \boldsymbol{v}}{\partial \eta}\times\nabla\dot{\eta}\right) + \beta v + D(\eta)\nabla^4\psi = 0$$

The terms of this equations are in order of appearance (1) Ekman friction E_ζ, (2) the local vorticity tendency, (3) vorticity flux divergence A_ζ, (4) the streching term S, (5) the

tilting term T, (6) the β–effect B and (7) a damping term D. Using these symbols, the local vorticity tendency may be expressed as:

$$\frac{\partial \zeta}{\partial t} = -(A_\zeta + S + T + B + E + D) \tag{13.1}$$

This equation can easily be discretised for all Euler–like schemes as:

$$\zeta_{t_n} = \zeta_{t_o} - \Delta t \, (A_\zeta + R + T + B + E + D)|_{t_i} \tag{13.2}$$

For a description of the time levels t_n, t_i and t_o as well as the discretisation for other time integration schemes see chapter 7.

The individual terms contributing to the vorticity tendency are given in their discretised version by:

$$\begin{aligned} A_\zeta|_{t_i} = {} & \frac{1}{m_x \Delta x}\Big(\max(u,0)\zeta^+\big|_{i+0.5} + \min(u,0)\zeta^-\big|_{i+0.5} - \max(u,0)\zeta^+\big|_{i-0.5} - \min(u,0)\zeta^-\big|_{i-0.5}\Big) \\ & + \frac{1}{m_y \Delta y}\Big(\max(v,0)\zeta^+\big|_{j+0.5} + \min(v,0)\zeta^-\big|_{j+0.5} - \max(v,0)\zeta^+\big|_{j-0.5} - \min(v,0)\zeta^-\big|_{j-0.5}\Big) \end{aligned} \tag{13.3}$$

$$R|_{t_i} = \begin{cases} (f + \zeta_{t_i})\mathcal{D}_{t_i} & \text{if } \texttt{ladv\_ageo} \\ f\mathcal{D}_{t_i} & \text{if } \neg\texttt{ladv\_ageo} \end{cases} \tag{13.4}$$

$$\begin{aligned} T|_{t_i} = {} & \frac{\Delta^k u_{t_i}^{i+0.5} + \Delta^k u_{t_i}^{i-0.5}}{2\Delta\eta} \frac{\dot\eta_{t_i}^{j+1} + \dot\eta_{t_i}^{j+1,k+1} - \dot\eta_{t_i}^{j-1} - \dot\eta_{t_i}^{j-1,k+1}}{2m_y\Delta y} \\ & - \frac{\Delta^k v_{t_i}^{j+0.5} + \Delta^k v_{t_i}^{j-0.5}}{2\Delta\eta} \frac{\dot\eta_{t_i}^{i+1} + \dot\eta_{t_i}^{i+1,k+1} - \dot\eta_{t_i}^{i-1} - \dot\eta_{t_i}^{i-1,k+1}}{2m_x\Delta x} \end{aligned} \tag{13.5}$$

$$B|_{t_i} = \beta v_{t_i} \tag{13.6}$$

$$E_\zeta|_{t_i} = r\zeta_{t_i} \tag{13.7}$$

$$D|_{t_i} = D_\sigma \nabla^2 \zeta_{t_i} \tag{13.8}$$

The horizontal wind components u and v, the vertical wind ω and the divergence $\mathcal{D}$ are diagnosed as detailed in the following sections. The Laplace operator ∇^2 will be defined in the following section, the upwind biased expressions for the advected gradients are listed in table 8.1. The operators $\Delta^k u_t$ and $\Delta^k v_t$ express a standard centered difference as far an upper and a lower layer are present. Otherwise the discretisation is biased towards the model domain:

$$\left(\Delta^k u_t, \Delta^k v_t\right) = \begin{cases} \left(u_t^{k+1} - u_t^k, v^{k+1} - v_t^k\right) & \text{if } \exists k+1 \text{ and } \nexists k-1 \\ \left(u_t^k - u_t^{k-1}, v^k - v_t^{k-1}\right) & \text{if } \nexists k+1 \text{ and } \exists k-1 \\ \frac{1}{2}\left(u_t^{k+1} - u_t^{k-1}, v^{k+1} - v_t^{k-1}\right) & \text{if } \exists k+1 \text{ and } \exists k-1 \end{cases} \tag{13.9}$$

The terms in (13.2) can individually be (de–)activated in the configuration namelist. The control variables are called (in order of the appearance of the terms): `ladv`, `lrotation`, `ltilting`, `lcoriolis`, `lfriction` and `ldamp`.

lrotation does not exist yet

13.2. Diagnosing stream function

Once the new vorticity field is calculated, the stream function needs to be established to allow diagnosing all remaining variables. While the inverse operation can be formulated easily by the usual discretisation of the Laplacian

$$\zeta_{t_i} = \nabla^2 \psi_{t_i} \approx \frac{\psi_{t_i}^{i+1} - 2\psi_{t_i} + \psi_{t_i}^{i-1}}{m_x^2 \Delta x^2} + \frac{\psi_{t_i}^{j+1} - 2\psi_{t_i} + \psi_{t_i}^{j-1}}{m_y^2 \Delta y^2} \quad , \tag{13.10}$$

it is not straightforward to calculate ψ from ζ. As the Laplacian is a linear operator, the equation can be written in terms of a matrix multiplication as $\zeta_{t_i} = \mathcal{L}\psi_{t_i}$. Each row of the matrix $\mathcal{L}$ contains the information, how the vorticity in the corresponding grid cell depends on the stream function in all other grid cells. As one can see from 13.10, the vorticity depends one the stream function of the grid cell itself, plus the four directly neighbouring grid cells. This fact makes the matrix $\mathcal{L}$ sparse (the vast majority of the grid cells do not contribute to a specific vorticity value) and diagonally–dominant (the grid cell depends mostly on itself).

Matrixes with these properties can be inverted by several algorithms (section 9), allowing to calculate the stream function by $\psi_{t_i} = \mathcal{L}^{-1}\zeta_{t_i}$. It is however important to note that these algorithms do not require either $\mathcal{L}$ or $\mathcal{L}^{-1}$ to be stored in memory in their entirety. This fact eases the memory requirement of the algorithms drastically for larger model domains by avoiding the storage of myriads of zeros.

13.3. Diagnosing temperature

The measure $T_\psi = \frac{\partial \psi}{\partial \eta}$ is called "temperature" within the model. It is defined in the same places as the vertical velocity. As derived for both coordinate systems in section 3.3, the relation to the real temperature can be established via the relation between the (geostrophic) stream function and the (geostrophic) geopotential and the thermal wind balance. Combining these equation yields

$$\begin{aligned} \frac{\partial \psi}{\partial p} &= -\frac{RT}{f_0 p} \quad \text{in } p\text{-coordinates and} \\ \frac{\partial \psi'}{\partial z} &= -g\frac{T'}{\theta_0} \quad \text{in } z\text{-coordinates.} \end{aligned}$$

Thus, in both coordinate systems, the variations of $\frac{\partial \psi}{\partial \eta}$ are entirely due to variations in temperature.

Due to the vertical staggering of the grid (section 6, figure 6.1), the following discretisation is also a centered difference:

$$T_\psi = \frac{\partial \psi}{\partial \eta}\Big|_{t_i} \approx \frac{\psi_{t_i}^k - \psi_{t_i}^{k-1}}{\Delta \eta} \quad . \tag{13.11}$$

13.4. Diagnosing vertical winds

The vertical velocity component is diagnosed using the thermodynamic equation, here repeated in the most complex semi-geostrophic version (4.6).

$$\left(r + \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla_\eta\right) \frac{\partial \psi}{\partial \eta} + N^2 C(\eta)\dot{\eta} = Q(\eta)$$

Solving this equation for $\dot{\eta}$ yields

$$\dot{\eta} = -\frac{1}{N^2 C(\eta)} \left(\left(r + \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla \right) \frac{\partial \psi}{\partial \eta} - Q(\eta) \right) = -\frac{1}{N^2}(L + A_T + E_T + F) \quad . \tag{13.12}$$

The three contributing terms are, in order, (1) Ekman friction E_T, (2) the local temperature tendency L, (3) temperature advection A_T and (4) diabatic forcing F. In their discretised version, they are given by:

$$L|_{t_i} = \frac{(\psi^k_{t_i} - \psi^k_{t_o}) - (\psi^{k-1}_{t_i} - \psi^{k-1}_{t_o})}{\Delta t \Delta p} \tag{13.13}$$

$$\begin{aligned} A_T|_{t_i} = {} & \frac{1}{m_x \Delta x}\Big(\max(u,0)T^+\big|_{i+0.5} + \min(u,0)T^-\big|_{i+0.5} - \max(u,0)T^+\big|_{i-0.5} - \min(u,0)T^-\big|_{i-0.5}\Big) \\ & + \frac{1}{m_y \Delta y}\Big(\max(v,0)T^+\big|_{j+0.5} + \min(v,0)T^-\big|_{j+0.5} - \max(v,0)T^+\big|_{j-0.5} - \min(v,0)T^-\big|_{j-0.5}\Big) \end{aligned} \tag{13.14}$$

$$E_T|_{t_i} = rT_{t_i} \tag{13.15}$$

$$F|_{t_i} = -\frac{g}{\theta_0} Q_{t_i} \tag{13.16}$$

Again, the advection terms can be disabled by setting `ladv` to false. The diabatic heating Q may be formulated freely in the model source code and activated by setting `lforce`.

13.5. Diagnosing divergence and velocity potential

The last diagnostic that is needed in the vorticity equation is the divergence $\mathcal{D}$. It can be determined by the continuity equation

$$\frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y} + \frac{\partial \dot{\eta}}{\partial \eta} = 0 \quad .$$

From the definition

$$\mathcal{D} \equiv \frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y}$$

follows

$$\mathcal{D} = \frac{\partial \dot{\eta}}{\partial \eta} \quad . \tag{13.17}$$

This equation may be discretised as following centered difference:

$$\mathcal{D}_{t_i} = \frac{\dot{\eta}^{k+1}_{t_i} - \dot{\eta}^k_{t_i}}{\Delta \eta} \tag{13.18}$$

The velocity potential is calculated from the divergence in the same way as the stream function is calculated from vorticity, in this case by $\chi_{t_i} = \mathcal{L}^{-1}\mathcal{D}_{t_i}$.

13.6. Diagnosing horizontal winds

By the definition of the stream function and the velocity potential

$$u = \frac{\partial \chi}{\partial x} - \frac{\partial \psi}{\partial y} \quad \text{and} \quad v = \frac{\partial \chi}{\partial y} + \frac{\partial \psi}{\partial x} \quad . \tag{13.19}$$

Using centered differences these equations read in discretised form

$$u_{t_i} \approx \frac{\chi_{t_i}^{i+0.5} - \chi_{t_i}^{i-0.5}}{m_x \Delta x} - \frac{\psi_{t_i}^{j+0.5} - \psi_{t_i}^{j-0.5}}{m_y \Delta y} \quad \text{and} \quad v_{t_i} \approx \frac{\chi_{t_i}^{i+0.5} - \chi_{t_i}^{i-0.5}}{m_y \Delta y} + \frac{\psi_{t_i}^{j+0.5} - \psi_{t_i}^{j-0.5}}{m_x \Delta x} \quad . \tag{13.20}$$

The centered differences can be applied everywhere within the model domain, as the stream function is defined on at least one row of boundary values outside the actual domain.

With this equation the model is closed and can be integrated.

IV. Model application

14. Installation and Update

Please report errors during the installation procedure to `csp001@uib.no`. The model was tested on a Linux machine maintained by the IT department of the University of Bergen. The necessary libraries are installed on `gfi-storm`, `skd-cyclone` and the computers in the G-Lab.

14.1. Prerequisites

- A Fortran 2003 compiler, e.g. gfortran-4.5 or newer.
- The netCDF library, version 4.0 or newer.
- NumPy & f2py for the python bindings and the python runscript
- SciPy, matplotlib & PyQt for BedymoGUI

14.2. Installation procedure

The below commands assume that you are within the UiB network.

1. Get the code from the public git–repository.

   ```
   $ git clone /Data/gfi/users/tsp065/lib/bedymo.git
   ```

   This creates a new directory called `bedymo` in the current directory. Use

   ```
   $ cd bedymo
   ```

   to get there.

2. Choose the branch of the model that you want to use. To check which branches are available use

   ```
   $ git branch
   ```

   The currently active branch is indicated by an asterisk. If unsure which branch to choose, take `zflux`. Switch to this branch by

   ```
   $ git checkout zflux
   ```

3. Run the compile script in the `bedymo` directory.

```
$ ./compile
```

If your host is not known to the compile script it will issue a warning, but tries to compile BEDYMO assuming that the NetCDF library is within the default library search path.

4. Start a short model run, using the default configuration provided via `bedymo_std.nml` to test the compilation result.

```
$ ./bedymo.x
```

You might also use the shared object `bedymo.so`, to run the model via the python runscript or BEDYMOGUI.

14.3. Update and development procedures

To update the source code, use the command

```
$ git pull
```

from within the `bedymo` directory. This command fetches all the updates from the git-repository that you cloned. If you followed the above installation procedure this is `/Data/gfi/users/tsp065/lib/bedymo.git`.

The command will fail with a message about uncommitted changes if any of the files in the repository have been edited. That includes the standard model configuration file `bedymo_std.nml`. In case only this configuration file is changed, you can restore the standard model configuration by executing

```
$ git checkout bedymo_std.nml
```

Note: Local changes to the file are being overriden by this command without further checks!

If you edited the model source code and want to keep the changes commit your changes by

```
$ git commit -a
```

and provide a meaningful but brief description of your changes. Once your changes are committed, you can retry the `git pull`. This time it will try to merge your version with the updated version from the public repository. If git is not able to merge the file automatically, it will insert marks into the file to point out the conflicts. When resolved, commit the merged version with `git commit -a`.

15. Configuration namelist

15.1. Numerics

The constants defined in the Numerics section define many aspects of how exactly the governing equations are to be solved. The most relevant part of this section is likely the grid configuration. In the standard configuration this section looks like:

```
&numerics
  nx = 180,
  ny = 180,
  np = 3,
  nt = 5,
  dx = 120000.,
  dy = 120000.,
  dt = 1800,
  ldt_dyn = F,
  latbdrtyp_ew = -1,
  latbdrtyp_sn = 0,
  alpha = 0.1,
  adv_order_max = 3,
  int_scheme_id = 2,
  bdr_damp = 0.2, /
```

The parameters `nx`, `ny` and `np` define the size of the grid (without boundaries) in grid points in the x-, y- and vertical dimension and `nt` is the number of time levels that are stored for the prognostic variables. Different schemes have different minimum requirements for `np` in the range 2–5. Above the minimum requirement will an increase of this number increase memory consumption without changing the solution.

The parameters `dx` and `dy` define the grid spacing in meters, `dt` the initial time step in seconds. Be aware that the time step is dynamically determined from a stability criterion if `ldt_dyn` is set to true.

The parameters `latbdrtyp_ew` and `latbdrtyp_sn` determine the lateral boundaries for the East–West direction and the South–North direction, respectively. Refer to chapter 10 for valid values and their meaning.

The parameter `alpha` is the coupling parameter for the Robert-Asselin time filter that is used for the Leap Frog time integration scheme. For `alpha`=0 there is no coupling.

The parameters `adv_order_max` and `int_scheme_id` determine which advection scheme and time integration scheme is used. Refer to chapters 8 and 7 for a description of valid values and their meaning.

The parameter `bdr_damp` controls the damping within the boundaries for condition 10. Refer to chapter 10 for details.

15.2. Physics

The physics section determines which variant of the equations outlined in chapter 4 are solved. Furthermore some few physical constants are to be defined here.

```
&physics
  lforce = F,
  ladv   = T,
  lconst_adv = F,
  ladv_ageo = F,
  lcoriolis  = T,
  lconst_coriolis = F,
  ltilting = F,
  lfriction = F,
  ldamp = T,
  lgeo_inhomo = F,
  fcor = 1.0e-4,
  betacor = 1.67e-11,
  bruntvf = 1.0e-2,
  ug = 10.0,
  vg = 0.0,
  ekman_coeff = 2.0e-6,
  damp_coeff = 1.0e+2, /
```

Currently it is not possible to have deviations from the β-plane.

The Boolean parameter `lforce` enable (if set to true) or disable the additional forcing terms both for vorticity and temperature. `ladv`, `lconst_adv`, `ladv_geo` determine the type of the advection term: is it generally enabled, is it linearised, and is the ageostrophic advection taken into account? `lcoriolis` and `lconst_coriolis` define if the coriolis force is enabled, and if deviations from the β-plane are considered. Setting `ltilting`, `lfriction` and `ldamp` enables the tilting term, scale-independent ekman friction and scale-selective damping $\propto \nabla^4\psi$, respectively. Setting `lgeo_inhomo` allows the specification of inhomogeneous basic state winds components u_g and v_g. They are read from the setup netCDF-file.

The parameters `fcor` and `betacor` define the coriolis parameter f, and the change of the of the coriolis parameter with y,which is usually called β. The units for both values are s^{-1}.

The parameter `bruntvf` (called N^2 in the equations) represents the square of the Brunt-Väisälä frequency as a measure of static stability. The unit is s^{-2}.

The parameters `ug` and `vg` define the basic state, geostrophically balanced wind components in x and y direction. Units: $\mathrm{m\,s}^{-1}$.

The parameters `ekman_coeff` and `damp_coeff` define the strength of the scale-independent ekman friction and the scale selective damping. The units for the friction coefficient is s^{-1} and for the damping coefficient $\mathrm{m}^2\,\mathrm{s}^{-1}$.

15.3. Initialisation for the tracer module

The tracer section determines, if tracer variables are included in the model run, and if they are, how they are initialised.

```
&tracer
  lhumidity = F,
  lpassive = T,
  npas = 1,
  tra_surftyp = 0,
  tra_amp = 1e-3,
  tra_lx  = 4.0,
  tra_ly  = 4.0,
  tra_cx  = 50,
  tra_cy  = 100, /
```

Currently, there are no parameterisations for moist processes in the model. Hence, this parameter currently just adds three more passive tracers with suggestive names.

The Boolean parameters `lhumidity` and `lpassive` enable parts of the tracer module, if set to true. `lhumidity` includes three tracer variables with named "water vapour", "cloud water" and "precip water" in the calculations. `lpassive` adds `npas` passive tracers to the calculations, which are just named "tracer 1" to "tracer `npas`".

The following parameters determine how to initialise *all* tracer variables: The first parameter `tra_surftyp` defines the type of the surface function. Currently there are five shape functions, and two additional shape IDs (-1 and 0) bear special meaning: 0 specifies an all-zero initial field, -1 an arbitrary initial field that is read from the setup netCDF file. Consequently, valid values for this parameter range from -1–5. They are summarized in table 15.1.

The other parameters determine the generic parameters for an initialisation by an idealised shape: First the amplitude `tra_amp` (in units of the tracer) of the shape, followed by the length scales of the structure in x and y, called `vor_lx` and `vor_ly`, and the centre of the structure `vor_cx` and `vor_cy`. The length scale and centre are given in grid points and may be floating point numbers. If either the centre or the length scale for one axis is zero, then a ridge-like structure is generated, as the dependence of shape function on this axis is omitted. By this means, one could for example generate a triangular ridge in the y-direction by setting `vor_cy` or `vor_ly` to zero, and choosing `vor_surftyp`$= 2$.

15.4. Initialisation for the vorticity

The vorticity field may optionally be initialised via some idealized shape, analogously to the tracer initialisation.

```
&init
  vor_surftyp = 0,
  vor_amp = 5e-5,
  vor_lx  = 3.0,
```

Table 15.1.: Description of the implemented surface types

ID	Description
−1	Read initial field from setup netCDF file
0	No surface, whole field set to zero
1	Cuboid with sharp edges of length $2l_x$ and $2l_y$.
2	Cone with elliptic base area
3	Gaussian or bell-shape
4	Witch of Agnesi function (similar to Gaussian but with x^{-2}-falloff)
5	Dipole structure with exponential falloff

```
  vor_ly  = 3.0,
  vor_cx  = 100,
  vor_cy  = 60, /
```

The parameters in this section are defined in the same way as in the previous section. The only difference is that the amplitude must now be given in the units of vorticity, s^{-1}. For `top_surftyp`=−1, a variable called "vorticity" is read from the setup netCDF-file.

15.5. Topography

The topography can be defined analogously to the initialisation field for the vorticity and the tracers in the previous sections.

```
&topo
  top_surftyp = 3,
  top_amp = 2000,
  top_lx  = 5.0,
  top_ly  = 15.0,
  top_cx  = 45,
  top_cy  = 90, /
```

The parameters in this section are defined in the same way as in the previous sections. The only difference is that the amplitude must now be given in the units of the orography, either $\mathrm{Pa} = \mathrm{kg\,m^{-1}s^{-2}}$ or m. For `top_surftyp`=−1, a variable called "surface pressure" for BedymoP or "zsur" for BedymoZ is read from the setup netCDF-file.

15.6. Time and output control

The final section of the configuration file defines all parameters that are related to model time and output of the meteorological results.

```
&timeout
  lout     = T,
  loutbdr  = T,
  loutinit = T,
  loutend  = T,
  lcoards_compatible = T,
  tstop    = 0,
  tsstop   = 604800,
  dtout    = 43200, /
```

The Boolean parameters `lout`, `loutbdr`, `loutinit`, `loutend` and `lcoards_compatible` define specifics of if and how the results are written to a netCDF file. The output of results can be suppressed completely by disabling `lout`. The following parameters determine if (1) the boundaries shall be part of the output, (2) the initialisation field shall be part of the output, (3) the last time step shall be part of the output and (4) if the output shall be compatible to the COARDS [1] convention. The latter is strongly encouraged, but involves the transposition of all fields that are part of the output.

The parameters `tstop` and `tsstop` both define the end of the model run: For `tstop` it can must given in time steps, for `tsstop` it must be given in seconds. There is no precedence for these parameters, the model will stop as soon as one of the end conditions is met. It is therefore strongly recommended to set the unused parameter to zero.

The parameter `dtout` defines the output interval in seconds. Please note, that the output likely is not directly valid for the specified intervals, as the first time step *after* completing the interval is written. Depending on the time step, this might be some hundred or thousand seconds after the specified output times.

16. Running via Fortran executable

Running `$ ./bedymo.x -v` shows some relevant information about the binary and for running the model:

```
$ ./bedymo.x -v
bedymo Z beta-0.18.3-3-g045111a (2012-11-06 13:43:51 +0100)
(c) 2011-2012 Clemens Spensberger <csp001@uib.no>
              Geophysical Institute @ University of Bergen

          Compiler: gfortran -pedantic -O2
      Compile host: gfi063203.klientdrift.uib.no
      Compile time: 2012-12-19 13:34:18 +0100
```

[1] COARDS is short for "Cooperative Ocean/Atmosphere Research Data Service". More information: `http://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html`.

```
Known arguments:
        -f, --file: Manually specify input file as the following argument.
                    The file is expected to be a bedymo namelist for a
                    standard run or a bedymo netcdf-file for a restart run.
        -h, --help: Print this information and exit.
        -i, --info: Print this information and exit.
       -q, --quiet: Suppress any output on stdout.
     -r, --restart: Initiate restart run. Expects new end of model run
                    as following argument. Unit: Seconds.
       -s, --setup: Manually specify a setup netcdf-file. It should contain
                    basic state winds and/or topography, as specified in
                    the configuration namelist.
     -v, --version: Print this information and exit.
```

The version information is drawn from the git repository at compile time. It shows the latest tagged (git slang for "named") version, which is in this case *beta-0.18.3*. The appendix *3-g0451111a* to the version reveals that three revisions have been committed into the repository since the original version, and that the latest commit has the (shortened) git hash *g0451111a*. Beneath the copyright information some compiler info is given to be able to trace the binaries. The known arguments section follows the standard UNIX formatting; the possible arguments and their alternative version are explained after the colon.

In most cases BEDYMO will be called without any arguments. In this case it uses the standard input file `bedymo_std.nml` as the model configuration. For the restart case the standard input file – containing both the meteorological fields and the model configuration – is `bedymo_restart.nc`.

17. Running via python runscript

18. Running via BedymoGUI

19. Model output

BEDYMO produces three different kinds of output: (1) Progress information on stdout, (2) status information, (3) the model configuration and other incidences in a report file and last but not least (4) the actual meteorological results in a netCDF file. The output of progress information can be suppressed by providing the `-q` or `--quiet` parameter. This mechanism is also used by BEDYMOGUI and the python runscript to avoid that the model clutters the python shell. The meteorology output can also be disabled completely by setting `lout` to false in the model configuration (see chapter 15.6). This might be useful, if the model results shall only be ad-hoc and live visualised, e.g. by BEDYMOGUI.

In the following sections, the model status and report files and the netCDF output are described in detail.

19.1. Report file

The head section of the report contains practically the same information as one could obtain by running `./bedymo.x -v`, which is described in detail in section 16. The only additional information is the starting time for the model run in the third line of the report.

```
# ---------------------------<| Bedymo Report |>--------------------------
#
#             Start time: 08.11.2011 11:43:19 +0100
#              Host name: gfi063203.klientdrift.uib.no
#                Version: bedymo alpha-0.9.6-3-g70b391b (2011-11-02 14:12:50 +0100)
#               Compiler: gfortran
#       Compiler options: -pedantic -O2
#           Compile time: 2011-11-08 11:42:54 +0100
#           Compile host: gfi063203.klientdrift.uib.no
#
# ---------------------------<| Configuration |>--------------------------
```

Here, the configuration namelist follows. It is identical to the configuration namelist, that was used as input. For this reason, this part is omitted in the description of the report file. The configuration namelist is described in section 15.

Please note that all lines in the report, except for the reprint of the namelist are commented out by the hash sign "#" in the beginning of each line. For this reason, each report file is a valid input namelist in its own right, which could be used to re-start a model run with an identical configuration.

```
# ---------<|               Starting integration               |>---------
#
# --------------------------<| Report Summary |>--------------------------
#
#               End time: 11.11.2011 13:53:12 +0100
#                Version: bedymo alpha-0.9.6-3-g70b391b
#       Number of errors:     0
#     Number of warnings:     0
#      Number of notices:     0
#
# -------------------------<| Bedymo Report End |>-------------------------
```

After the reprint of the model configuration, the actual integration is starting. In the above example no errors, warnings or notable events occurred. Thus, the start of the model run is directly followed by footer section of the report. In addition to the end time of the model run and the (repeated) version information the number of errors, warnings and notable events is given. In the example nothing bad happened, such that all these counters are zero.

As an error causes an immediate termination of the model run, this counter will never exceed 1.

19.2. Status file

The second output file permits to get a short glance of the model status, e.g. to check how far the integration has progressed or if the model fields are still physically meaningful. The Fortran format for each line is `(a8,2i6,i10,3i3,9e15.7)`. An example output (in which the last seven columns are omitted) for the first five time steps is given below:

```
Status      1   300       300  1  1  4  0.0000000E+00  0.9720000E+07 [...]
Status      2  5428      5728  2  2  3  0.0000000E+00  0.9720012E+07 [...]
Status      3  4796     10524  3  3  4 -0.7599594E-17  0.9724631E+07 [...]
Status      4  3966     14490  0  0  1  0.1795092E-19  0.9736113E+07 [...]
Status      5  3354     17844  1  1  4  0.3100361E-07  0.9752037E+07 [...]
[...]
```

The meaning of the first seven columns is

1. the string “Status”,
2. the time step number,
3. the integration time step Δt,
4. the model time in seconds since the model start,
5. the time index `tlo`,
6. the time index `tli`,
7. the time index `tln`.

They are followed by nine floating point values, showing

8. the domain integrated vorticity,
9. the domain integrated kinetic energy,
10. the domain averaged x-wind,
11. the domain averaged y-wind,
12. the domain averaged stream function,
13. the domain averaged vorticity tendency due to the beta effect,
14. the domain averaged vorticity tendency due to x-advection,
15. the domain averaged vorticity tendency due to y-advection,
16. the total domain averaged vorticity tendency.

A first assessment of how meaningful the solution is may be obtained by checking the conservation of vorticity, energy and momentum based on columns 8–11. Also, a very small dynamic time step in column 3 indicates a blown-up solution with very high wind speeds.

19.3. Meteorology output

The meteorology is output as a NetCDF file, that complies to the *NetCDF Climate and Forecast (CF) Metadata Conventions*[2], version 1.5. In addition, if the parameter `lcoards_compatible` is set (which is highly recommended, see section 15.6) the file also complies to the *COARDS*[3] standard.

The NetCDF file stores all relevant information to repeat an the model run, or to extend the modelled period by restarting and using the latest output as an initialisation. The information about the model version, compilation and configuration are all stored as global attributes (see table 19.1).

Currently the variables that are written cannot be configured. In principle all fields can be diagnosed directly from the stream function as this is the only prognostic variable. For convenience, however, also vorticity, the wind velocity components and a temperature-like quantity are written. This information, along with the coordinate axis in which the variables are defined are summarised in table 19.2.

The main attributes for these variables and the coordinate variables in the NetCDF file are compiled in table 19.3.

[2]More information: `http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.5/cf-conventions.html`

[3]COARDS is short for "Cooperative Ocean/Atmosphere Research Data Service". More information: `http://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html`.

Table 19.1.: Global attributes for the BEDYMO meteorology output.

Name	Description
`title`	The string "Bedymo model output".
`institution`	The string "Geophysical Institute @ University of Bergen".
`Conventions`	Which conventions does this NetCDF file subscribe to? Always "CF-1.5", often also "COARDS".
`model_version`	The model version. Refer to chapter 16 for a detailed description.
`model_version_date`	When was the above model version created?
`model_config`	A list of values for all variables in the BEDYMO namelist. It is used for restart, but may also be used for documentary purposes.
`model_config_version`	An integer value, that is incremented whenever the above list changes.
`compile_host`	On which host was the binary compiled, that produced this output?
`compile_time`	When was the binary compiled, that produced this output?
`compile_cmd`	Which compiler was used with which options to compile the binary that produced this output?

Table 19.2.: Variables in the BEDYMO meteorology output.

Name	Axes	Description
`time_step`	(T)	Time step that was output.
`stream function`	(T, P_f, Y_b, X_b)	Stream function.
`vorticity`	(T, P_f, Y_b, X_b)	Relative vorticity.
`x_wind`	(T, P_f, Y, X)	Wind velocity component along the x-axis.
`y_wind`	(T, P_f, Y, X)	Wind velocity component along the y-axis.
`p_wind`	(T, P_h, Y_b, X_b)	Wind velocity component along the p-axis.
`temperature`	(T, P_h, Y, X)	Change of stream function with height $\frac{\partial \psi}{\partial p}$.
`surface_pressure`	(Y_b, X_b)	Stationary surface pressure field representing orography.

Table 19.3.: Variable attributes for the variables in table 19.2 and coordinate variables.

Name	Description
`standard_name`	Name of the variable according to CF-1.5.
`units`	Units for the variable in a UDUNITS[a] compatible format.
`axis`	Type of axis for coordinate variables: "X", "Y", "Z" or "T".
`positive`[b]	Used for the pressure axis with the value "down" to indicate that the pressure is increasing downwards.

[a] more information: `http://www.unidata.ucar.edu/software/udunits/`.
[b] optional

V. Model development

20. Source code organisation

The source code is organised in different *modules*. The following list provides an overview on the purposes of each. They are ordered by their appearance in `bedymo.f95`, which is the order of their hierarchy of inter–dependencies. Each module may thus only depend on (and use variables from) modules higher up on that list. WIP

kind Defines the precision for integer and real numbers used throughout the model.

consts Defines several physical parameters, plus some meta–information (like version, compiler, etc.) about the model itself.

ptr Defines procedure pointers and their interfaces as a new data types.

logfile Provides a logging and error reporting mechanism.

surflib Provides functions to create idealised surfaces (e.g. for topography, initialisation for variables, etc.)

config Reads and writes the model configuration from namelists and BEDYMO netCDF files.

metin Reads BEDYMO netCDF files and extracts variables necessary for restart or non-idealised initialisation.

grid Defines the model domain and grid; provides generic subroutines for setting boundary values and for applying and inverting the Laplacian.

diag Defines diagnostic variables, such as the horizontal wind velocity components and provides functions to calculate these.

advection Provides functions to calculate the advection terms.

trace Tracer advection module, providing the time integration for tracer variables and hooks for among others water vapour parameterisations.

prog Defines prognostic variables and subroutines to calculate these. In a strict sense this would only be the vorticity, but due to their close relation also stream function and vertical wind velocity are counted here as "prognostic".

timeint Provides functions that implement the different time integration schemes

metout Writes the model results as BEDYMO netCDF files, that also serve as restart files.

runctrl Provides functions for initialising, running and terminating the model.

bedymo Is not a Fortran module, but the main program. Parses command–line arguments and subsequently invokes the runctrl module.

Table 20.1.: A non–comprehensive list of variables that are widely used in the source code

Name	Module	Symbol	Description
`advptr`	`ptr`		Procedure pointer type, pointing to an advection scheme implementation
`alpha`	`timeint`	α	Coupling constant of the two leap–frog time steps in the Robert–Asselin filter
`avail_int_schemes`	`runctrl`		Array of procedure pointers (type `intptr`) containing all available time–integration schemes
`axy`	`grid`	X_a, Y_a	First index of variable arrays including boundaries for x– and y–directions
`bdr_ew`	`grid`		Boundary condition index for the eastern and western boundaries
`bdr_sn`	`grid`		Boundary condition index for the eastern and western boundaries
`betacor(`Y`,`X`)`	`grid`	β	Change of the Coriolis parameter with y
`bruntvf(`Y`,`X`)`	`grid`	N^2	Square of the Brunt-Väisälä frequency
`config_nml`	`config`		String containing the complete model configuration; It is saved to a netCDF generic attribute to allow restarting from netCDF files.
`config_nml_version`	`config`		Version of model configuration variable set, that is necessary to restart
`dadxplus`	`advection`		Array of procedure pointers (type `advptr`) containing all available advection schemes (for positive wind speeds)
`dadxminus`	`advection`		Array of procedure pointers (type `advptr`) containing all available advection schemes (for negative wind speeds)
`damp_coeff`	`prog`	D_p	Damping coefficient
`div(`P`,`Y`,`X`)`	`diag`	$-\frac{\partial\omega}{\partial p} - \frac{d \ln p_s}{dt}$	Divergence of the wind field
`dp(`Y_b`,`X_b`)`	`grid`	Δp	Vertical grid spacing
`dt`	`timeint`	Δt	Time step
`dx(`Y_b`,`X_b`)`	`grid`	Δx	Horizontal grid spacing (x–direction)

Name	Module	Symbol	Description
`dy`(Y_b,X_b)	`grid`	Δy	Horizontal grid spacing (y–direction)
`ekman_coeff`	`prog`	r	Ekman friction coefficient
`fcor`(Y,X)	`grid`	f, f_0	Coriolis parameter
`gen_surface`	`surflib`		Array of procedure pointers (type `surfptr`) containing all available functions
`infile`	`bedymo`		Overrides the standard input file names. Namelist file for standard model run or BEDYMO netCDF file for restarted run
`intptr`	`ptr`		Procedure pointer type, pointing to a time–integration scheme implementation
`int_scheme`	`runctrl`		Procedure pointer (type `intptr`) pointing to the employed time–integration scheme
`int_scheme_id`	`runctrl`		Index of the employed time–integration scheme
`ioerr`	`metin/out`		NetCDF error code for any netCDF read or write operation
`ladv`	`config`		Enable advection?
`ladv_ageo`	`config`		Enable advection by ageostrophic wind components?
`lcoards_compatible`	`config`		Make the output compatible to the COARDS standardisation project? (involves transposing all output fields)
`lconst_adv`	`config`		Enable advection only by stationary basic state?
`lconst_coriolis`	`config`		Enable constant Coriolis parameters (f, β) within the model domain?
`lcoriolis`	`config`		Enable Coriolis force?
`ldamp`	`config`		Enable scale–selective damping? ($\propto \nabla^4\psi$)
`lforce`	`config`		Enable forcing of the vorticity and vertical velocity fields?
`lfriction`	`config`		Enable ekman friction?
`llpsi`(P,Y_b,X_b)	`prog`	$\nabla^2\zeta$	Laplacian of the vorticity (used in damping term)
`lout`	`runctrl`		Output meteorological fields during the model run?
`loutbdr`	`config`		Include boundaries of prognostic variables in the output?

Name	Module	Symbol	Description
loutinit	runctrl		Output meteorological fields directly after the initialisation?
loutend	config		Output meteorological fields at the end of the model run?
lquiet	runctrl		Suppress any output (e.g. progress info) on stdout?
lpsi(P,Y_b,X_b,T)	prog	ζ	Vorticity (or the Laplacian applied to the stream function psi)
ltilting	config		Enable the tilting term?
nerror	logfile		Number of errors encountered during model run
ncid	metin/out		NetCDF opened file index
ni	kind		Precision (byte length) of integers, set to 4
nnote	logfile		Number of notifications encountered during model run
np	grid	P	Number of grid points in the vertical
nr	kind		Precision (byte length) of reals, set to 8
nt	timeint	T	Number of time levels available for each prognostic variable
nwarn	logfile		Number of warnings encountered during model run
nx	grid	X	Number of grid points in the model domain in x–direction
ny	grid	Y	Number of grid points in the model domain in y–direction
order_max	advection		The maximum advection scheme order used
p0	const	p_0	Reference surface pressure, set to 1013.25 hPa
ps(Y_b,X_b)	grid	p_s	Surface pressure
psdx(Y_b,X_b)	grid	$\frac{\partial p_s}{\partial x}$	Surface pressure gradient from topography in x–direction
psdy(Y_b,X_b)	grid	$\frac{\partial p_s}{\partial y}$	Surface pressure gradient from topography in y–direction
psi(P,Y_b,X_b,T)	prog	ψ	Stream function
restart	bedymo		If greater than zero: New end of the restarted model run
surfptr	ptr		Procedure pointer type, pointing to a surface generation function
t	timeint	n	Time step number

Name	Module	Symbol	Description
`temp(`P`,`Y`,`X`)`	`diag`	$\frac{\partial\psi}{\partial p}$	Change of stream function with pressure, may be interpreted as a temperature
`tli`	`timeint`	t_i	Index of the intermediate time level in the integration
`tln`	`timeint`	t_n	Index of the new time level in the integration
`tlo`	`timeint`	t_o	Index of the old time level in the integration
`ts`	`timeint`	t	Time since model start
`u(`P`,`Y`,`X`)`	`diag`	u	Wind velocity component in x–direction
`v(`P`,`Y`,`X`)`	`diag`	v	Wind velocity component in y–direction
`vor_tend(`P`,`Y`,`X`)`	`timeint`	$\frac{\partial\zeta}{\partial t}$	Vorticity tendency due to all contributing terms
`w(`P`,`Y_b`,`X_b`)`	`diag`	ω	Vertical (pressure coordinate) wind speed
`zx`	`grid`	$X_b + X_a$	Last index for variable arrays including boundaries for x–direction
`zy`	`grid`	$Y_b + Y_a$	Last index for variable arrays including boundaries for y–direction

21. Call tree

Table 21.1.: BEDYMO call tree for a standard model run

Subroutine	Module	Remarks
main program	`bedymo`	
parse_cmdline	`bedymo`	
init_bedymo	`runctrl`	
`init_logfile`	`logfile`	
`init_surflib`	`surflib`	
`config_by_namel`	`config`	
`init_grid`	`grid`	
`ilaplace_precond_init`	`grid`	
`init_diag`	`diag`	
`init_advection`	`advection`	
`init_trace`	`trace`	
`gen_surface`	`surflib`	

Subroutine	Module	Remarks
init_prog	prog	
gen_surface	surflib	
set_latbdr_l	grid	
ilaplace	grid	
laplace	grid	several times per iteration
ilaplace_precond_LU	grid	several times per iteration
set_latbdr	grid	several times per iteration
ilaplace_precond_L	grid	several times per iteration
set_latbdr	grid	
cal_hor_wind	diag	for the temperature advection
cal_ver_wind	prog	
cal_advx	advection	
cal_advy	advection	
cal_temp_force	diag	
cal_temp	diag	
set_latbdr	grid	
cal_div	diag	
cal_hor_wind	diag	possibly including ageo. winds
init_timeint	timeint	
init_metout	metout	
run	runctrl	
output	metout	if loutinit
entering integration loop		
int_scheme	timeint	
cal_vor_tendency	prog	
cal_advx	advection	
cal_advy	advection	
cal_rotation	prog	
cal_betaeffect	prog	
cal_tilting	prog	
cal_vor_friction	prog	
cal_vor_force	prog	
cal_trace_tendency	trace	
cal_trace_force	trace	
cal_advx	advection	for each tracer
cal_advy	advection	for each tracer
step_apply	timeint	
set_latbdr_l	grid	
ilaplace	grid	
laplace	grid	several times per iteration
ilaplace_precond_LU	grid	several times per iteration
set_latbdr	grid	several times per iteration
ilaplace_precond_L	grid	several times per iteration

Subroutine	Module	Remarks
set_latbdr	grid	for vorticity and each tracer
step_prepare	timeint	
cal_ver_wind	prog	
cal_advx	advection	
cal_advy	advection	
cal_temp_force	diag	
cal_temp	diag	
set_latbdr	grid	
cal_div	diag	
set_latbdr_l	grid	
ilaplace	grid	if ladv_ageo
laplace	grid	several times per iteration
ilaplace_precond_LU	grid	several times per iteration
set_latbdr	grid	several times per iteration
ilaplace_precond_L	grid	several times per iteration
set_latbdr	grid	if ladv_ageo
cal_hor_wind	diag	
output	metout	if output time step
end of integration loop		
output	metout	if loutend
term_bedymo	runctrl	
term_metout	metout	
term_timeint	timeint	
term_prog	prog	
term_trace	trace	
term_diag	diag	
term_grid	grid	
term_logfile	logfile	

22. Change log

Versions of BedymoZ

beta 0.18.3 Introduced temperature friction to be consistent with vorticity and reduce numerical instabilities.

beta 0.18.2 Bugfix in the calculation of the vertical wind, introduced ldt_dyn.

beta 0.18.1 Checks and fixes for the 4^{th}-order Runge-Kutta scheme.

beta 0.18.0 Major revision of the time integration handling: Moved prog before timeint, implemented RAW-filter and introduced a maximum time step of 3 h. Changed recommended time stepping to Leap-Frog.

beta 0.16.2 Adapted to new UiB infrastructure.

beta 0.16.1 Several bugfixes making the 3D and linear modes of operation usable again. Adapted compile script for `skd-cyclone` and MacOS.

beta 0.16.0 Introduced a tracer transport module and the possibility for non-idealised topography and basic-state winds by reading those fields from a setup netCDF-file.

beta 0.14.1 Adapted to new UiB infrastructure.

beta 0.14.0 Introduced Semi-Geostrophy mode, taking into account the ageostrophic advection.

beta 0.12.4 Adapted to new UiB infrastructure.

beta 0.12.3 Consistent boundary conditions for the new grid and the flux form of the advection.

beta 0.12.2 Maintenance version, further cleaned-up code and improved BedymoGUI.

beta 0.12.1 Maintenance version, cleaned-up code and improved compile script.

beta 0.12.0 Introduction of ILU(0) preconditioning. Adapted the advection terms to flux form, consequently changed to a staggered grid (Arakawa C) in the horizontal.

beta 0.10.1 Adapted to new UiB infrastructure.

beta 0.10.0 Adaption to z-coordinate system, rename to BedymoZ.

Versions of BedymoP

beta 0.11.1 Adapted to new UiB infrastructure.

beta 0.11.0 Consistent use of the p-coordinate system, rename to BedymoP.

Versions prior to the split-up

alpha 0.9.8 Adapted to new UiB infrastructure.

alpha 0.9.7 Improvements to BedymoGUI and to the compile script.

alpha 0.9.6 Bug-fix: Sign error in continuity equation.

alpha 0.9.5 Technical improvements: git version tag in precompiler, Bedymo executable independent of `LD_LIBRARY_PATH`, compile script for `gfi-storm`.

alpha 0.9.4 Moved basic state winds `ug`, `vg` to namelist. Consequently introduced `config_nml_version` 2.

alpha 0.9.3 Introduced `lquiet` for BedymoGUI compatibility.

alpha 0.9.2 Proper cleanup (among others finishing netCDF output) in error case.

alpha 0.9.1 Introduced boundary condition "determined by ψ".

alpha 0.9.0 Introduced topography effect and added the ability to restart from netCDF-files.

alpha 0.8.0 Bug-fix in subroutine grid/ilaplace: to improve energy conservation. Introduced error handling + logfile + status file mechanisms.

alpha 0.7.0 First numbered version.

23. Licence

To be determined.

Appendix

A. Construction of higher-order upwind-biased advection schemes

The derivation will be demonstrated using the one–dimensional advection equation of any quantity χ by the wind field $u > 0$:

$$\frac{\partial \chi}{\partial t} + u\frac{\partial \chi}{\partial x} = \mathcal{F}\,. \tag{A.1}$$

Using Taylor–expansion of the advected quantity χ

$$\chi(x+\Delta x) = \chi(x) + \frac{\partial \chi}{\partial x}\Delta x + \frac{\partial^2 \chi}{\partial x^2}\frac{\Delta x^2}{2} + \frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} + \frac{\partial^4 \chi}{\partial x^4}\frac{\Delta x^4}{24} + \frac{\partial^5 \chi}{\partial x^5}\frac{\Delta x^5}{120} + \mathcal{O}(\Delta x^6) \tag{A.2}$$

and the shorthand notation $\chi_{i+m} \equiv \chi(x+m\Delta x)$ one can easily evaluate the left-hand side of the following equations.

$$\begin{aligned}
\chi_i - \chi_{i-1} &= 1\,\frac{\partial \chi}{\partial x}\Delta x - 1\,\frac{\partial^2 \chi}{\partial x^2}\frac{\Delta x^2}{2} + 1\,\frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} - 1\,\frac{\partial^4 \chi}{\partial x^4}\frac{\Delta x^4}{24} + 1\,\frac{\partial^5 \chi}{\partial x^5}\frac{\Delta x^5}{120} + \mathcal{O}(\Delta x^6) && \text{(a)}\\
\chi_{i+1} - \chi_{i-1} &= 2\,\frac{\partial \chi}{\partial x}\Delta x + 2\,\frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} + 2\,\frac{\partial^5 \chi}{\partial x^5}\frac{\Delta x^5}{120} + \mathcal{O}(\Delta x^7) && \text{(b)}\\
\chi_{i+1} - \chi_{i-2} &= 3\,\frac{\partial \chi}{\partial x}\Delta x - 3\,\frac{\partial^2 \chi}{\partial x^2}\frac{\Delta x^2}{2} + 9\,\frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} - 15\,\frac{\partial^4 \chi}{\partial x^4}\frac{\Delta x^4}{24} + 33\,\frac{\partial^5 \chi}{\partial x^5}\frac{\Delta x^5}{120} + \mathcal{O}(\Delta x^6) && \text{(c)}\\
\chi_{i+2} - \chi_{i-2} &= 4\,\frac{\partial \chi}{\partial x}\Delta x + 16\,\frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} + 64\,\frac{\partial^5 \chi}{\partial x^5}\frac{\Delta x^5}{120} + \mathcal{O}(\Delta x^7) && \text{(d)}\\
\chi_{i+2} - \chi_{i-3} &= 5\,\frac{\partial \chi}{\partial x}\Delta x - 5\,\frac{\partial^2 \chi}{\partial x^2}\frac{\Delta x^2}{2} + 35\,\frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} - 65\,\frac{\partial^4 \chi}{\partial x^4}\frac{\Delta x^4}{24} + 275\,\frac{\partial^5 \chi}{\partial x^5}\frac{\Delta x^5}{120} + \mathcal{O}(\Delta x^6) && \text{(e)}
\end{aligned} \tag{A.3}$$

For odd–ordered schemes these equations suffice. They can be constructed taking into account the first (first three, all five) equation(s), neglecting all terms of order $\mathcal{O}(\Delta x^2)$ ($\mathcal{O}(\Delta x^4)$, $\mathcal{O}(\Delta x^6)$) and higher, and solving for $\frac{\partial \chi}{\partial x}$. For the even–ordered schemes one needs to replace the second and fourth equation of the above system by either the first two or the second two equations of (A.4). The difference between the two sets lies in the computational stencil used: for the above two equations its the three (five) points centred about the current grid point with index i, while the stencil is centred about one

point upwind ($i-1$ for $u>0$) in the lower two equations. All four equations (A.4) may be truncated after $\mathcal{O}(\Delta x^4)$, as they won't be used in the construction of the fifth–order scheme.

$$\begin{aligned}
\chi_{i+1}+\chi_{i-1}-2\chi_i = & +2\,\frac{\partial^2\chi}{\partial x^2}\frac{\Delta x^2}{2} & & +2\,\frac{\partial^4\chi}{\partial x^4}\frac{\Delta x^4}{24}+\mathcal{O}(\Delta x^6) & \text{(a)}\\
\chi_{i+2}+\chi_{i-2}-2\chi_i = & +8\,\frac{\partial^2\chi}{\partial x^2}\frac{\Delta x^2}{2} & & +32\,\frac{\partial^4\chi}{\partial x^4}\frac{\Delta x^4}{24}+\mathcal{O}(\Delta x^6) & \text{(b)}\\
\chi_{i}+\chi_{i-2}-2\chi_i = -2\,\frac{\partial\chi}{\partial x}\Delta x & +4\,\frac{\partial^2\chi}{\partial x^2}\frac{\Delta x^2}{2} & -8\,\frac{\partial^3\chi}{\partial x^3}\frac{\Delta x^3}{6} & +16\,\frac{\partial^4\chi}{\partial x^4}\frac{\Delta x^4}{24}+\mathcal{O}(\Delta x^5) & \text{(c)}\\
\chi_{i+1}+\chi_{i-3}-2\chi_i = -2\,\frac{\partial\chi}{\partial x}\Delta x & +10\,\frac{\partial^2\chi}{\partial x^2}\frac{\Delta x^2}{2} & -26\,\frac{\partial^3\chi}{\partial x^3}\frac{\Delta x^3}{6} & +82\,\frac{\partial^4\chi}{\partial x^4}\frac{\Delta x^4}{24}+\mathcal{O}(\Delta x^5) & \text{(d)}
\end{aligned} \tag{A.4}$$

A.1. The first–order upwind scheme

Taking into account only (A.3a), and neglecting terms of order $\mathcal{O}(\Delta x^2)$ and higher, one readily obtains the standard first–order upwind scheme

$$\frac{\partial\chi}{\partial x}\approx\frac{\chi_i-\chi_{i-1}}{\Delta x}\quad . \tag{A.5}$$

A.2. Two variants for second–order upwind schemes

Taking $2\cdot(A.3a)+(A.4a)$, such that higher–order spatial derivations of χ cancel each other out, yields the well–known centered difference scheme

$$\frac{\partial\chi}{\partial x}\approx\frac{\chi_{i+1}-\chi_{i-1}}{2\Delta x}\,. \tag{A.6}$$

This scheme is no longer upwind–biased and is can be proven to be unconditionally unstable when used as a spatial discretisation in the advection term. This scheme and the equations (A.4a, A.4b) are only provided in this derivation to show the connection to the advection schemes published by *Tremback et al.* (1987). In this study, the odd–ordered schemes derived in this section and the even–ordered schemes derived with (A.4a, A.4b) appear as the terms linear in the Courant number (called α in *Tremback et al.* (1987), p.541).

Using the upwind shifted stencil in (A.4c, A.4d) the second–order upwind scheme is constructed by $4\cdot(A.3a)+(A.4c)$:

$$\frac{\partial\chi}{\partial x}\approx\frac{3\chi_i-4\chi_{i-1}+\chi_{j-2}}{6\Delta x}\quad . \tag{A.7}$$

A.3. The third–order upwind scheme

Using (A.3a-c), neglecting terms of higher order than $\mathcal{O}(\Delta x^3)$ and calculating $3 \cdot (A.3a) - (A.3c) + 3 \cdot (A.3b)$, the resulting equation can be solved for $\frac{\partial \chi}{\partial x}$, yielding:

$$\frac{\partial \chi}{\partial x} \approx \frac{2\chi_{i+1} + 3\chi_i - 6\chi_{i-1} + \chi_{i-2}}{6\Delta x} \tag{A.8}$$

A.4. Two variants for fourth–order upwind schemes

First, again using the centered stencil corresponding to *Tremback et al.* (1987), the approximation

$$\frac{\partial \chi}{\partial x} \approx \frac{-\chi_{i+2} + 8\chi_{i+1} - 8\chi_{i-1} + \chi_{i-2}}{12\Delta x} \tag{A.9}$$

is obtained by combining the equations (A.3a,c) and (A.4a,b) by taking $18 \cdot (A.3a) + 10 \cdot (A.4a) - 2 \cdot (A.3c) - (A.4b)$. As expected the resulting scheme is no longer upwind–biased.

As for the second–order scheme, an upwind–biased variant can be constructed by moving the computational stencil one point upwind, as done in (A.4c,d). Adding $18 \cdot (A.3a) + 10 \cdot (A.4c) + 4 \cdot (A.3c) - (A.4d)$ and solving for $\frac{\partial \chi}{\partial x}$ yields

$$\frac{\partial \chi}{\partial x} \approx \frac{3\chi_{i+1} + 10\chi_i - 18\chi_{i-1} + 6\chi_{i-2} - \chi_{i-3}}{12\Delta x} \quad . \tag{A.10}$$

A.5. The fifth–order upwind scheme

Last but not least, by using all equations in (A.3) one can construct a fifth–order upwind scheme. As pointed out in section A.2, the available equations have to be combined such, that higher order derivatives of χ vanish. With this constraint the following system of equations for the weight factors a_1 to a_5 of the five individual equations in (A.3) emerges:

$$\begin{array}{rrrrrrr} 0 = & -a_1 & & -3a_3 & & -5a_5 \\ 0 = & a_1 & +2a_2 & +9a_3 & +16a_4 & +35a_5 \\ 0 = & -a_1 & & +15a_3 & & -65a_5 \\ 0 = & a_1 & +2a_2 & +33a_3 & +64a_4 & +275a_5 \end{array} \quad .$$

The remaining degree of freedom (which corresponds to the freedom of expanding or reducing the resulting fraction in (A.11)) is used to restrict the solution set of this system of equations to the smallest natural numbers solving it. The solution respecting this constraint is given by $a_1 = 20$, $a_2 = 40$, $a_3 = -10$, $a_4 = -5$, $a_5 = 2$. Hence, calculating $20 \cdot (A.3a) + 40 \cdot (A.3b) - 10 \cdot (A.3c) - 5 \cdot (A.3d) + 2 \cdot (A.3e)$ and solving for $\frac{\partial \chi}{\partial x}$ yields the fifth–order upwind–scheme:

$$\frac{\partial \chi}{\partial x} \approx \frac{-3\chi_{i+2} + 30\chi_{i+1} + 20\chi_i - 60\chi_{i-1} + 15\chi_{i-2} - 2\chi_{i-3}}{60\Delta x} \tag{A.11}$$

A.6. Extension for negative wind speeds and more dimensions

The same derivations as outlined in the sections above can be carried out for $u < 0$. In this case, moving the stencil upwind means moving its centre from i towards $i + 1/2$ for odd–ordered schemes and towards $i + 1$ for even–ordered schemes. It soon becomes obvious, that one can obtain the corresponding scheme for wind speeds of the opposite sign by changing both the sign of the weighting coefficients and of the index offsets in the numerator of the respective schemes.

The computationally expensive use of an `if`–structure in the loop over all grid cells can be avoided by using the `min` and `max` functions every higher–level programming language provides:

$$u_i \frac{\partial \chi}{\partial x}\Big|_i = \max(u_i, 0) \cdot \frac{\partial \chi}{\partial x^+}\Big|_i + \min(u_i, 0) \cdot \frac{\partial \chi}{\partial x^-}\Big|_i \quad . \tag{A.12}$$

The approximations for $\frac{\partial \chi}{\partial x^+}|_i$ and $\frac{\partial \chi}{\partial x^-}|_i$ by the different schemes are summarised in table A.1.

In multi–dimensional advection, the Taylor expansion (A.2) used to derive the advection scheme would contain many cross derivatives along several dimensions. As the amount of data points used in the computational stencil would increase with the order of the scheme to the power of the flow dimension, this technique is soon getting unfeasibly complex to compute and derive. An elegant way to circumvent this problem in multi–dimensional advection is using the time–splitting technique (*Smolarkiewicz*, 1982; *Tremback et al.*, 1987). Following this technique, the one–dimensional advection is applied separately for every dimension.

Table A.1.: Overview over upwind–biased spatial discretisation schemes

Order	$\left.\frac{\partial \chi}{\partial x^+}\right\|_i$	$\left.\frac{\partial \chi}{\partial x^-}\right\|_i$
1^{st}	$\frac{\chi_i - \chi_{i-1}}{\Delta x}$	$\frac{\chi_{i+1} - \chi_i}{\Delta x}$
2^{nd}	$\frac{3\chi_i - 4\chi_{i-1} + \chi_{j-2}}{6\Delta x}$	$\frac{-\chi_{j+2} + 4\chi_{i+1} - 3\chi_i}{6\Delta x}$
3^{rd}	$\frac{2\chi_{i+1} + 3\chi_i - 6\chi_{i-1} + \chi_{i-2}}{6\Delta x}$	$\frac{-\chi_{i+2} + 6\chi_{i+1} - 3\chi_i - 2\chi_{i-1}}{6\Delta x}$
4^{th}	$\frac{3\chi_{i+1} + 10\chi_i - 18\chi_{i-1} + 6\chi_{i-2} - \chi_{i-3}}{12\Delta x}$	$\frac{\chi_{i+3} - 6\chi_{i+2} + 18\chi_{i+1} - 10\chi_i - 3\chi_{i-1}}{12\Delta x}$
5^{th}	$\frac{-3\chi_{i+2} + 30\chi_{i+1} + 20\chi_i - 60\chi_{i-1} + 15\chi_{i-2} - 2\chi_{i-3}}{60\Delta x}$	$\frac{2\chi_{i+3} - 15\chi_{i+2} + 60\chi_{i+1} - 20\chi_i - 30\chi_{i-1} + 3\chi_{i-2}}{60\Delta x}$

B. Construction of higher-order upwind-biased flux estimator schemes

The derivation will be demonstrated using the one–dimensional flux divergence equation of any quantity χ by the wind field $u > 0$:

$$\frac{\partial \chi}{\partial t} + \frac{\partial u\chi}{\partial x} = \mathcal{F}\,. \tag{B.1}$$

This equation is simply discretised by by the centered difference

$$\frac{\Delta \chi_i}{\Delta t} = -\frac{u_{i+0.5}\chi_{i+0.5} - u_{i-0.5}\chi_{i-0.5}}{\Delta x} + \mathcal{F}_i\,. \tag{B.2}$$

The horizontal winds are defined on a staggered grid, such that $u_{i+0.5}$ is directly available. χ however is only defined on the unstaggered grid, and hence must be interpolated on to the staggered grid.

Using the same Taylor expansion (A.2) as above, the following set of equations follows:

$$\begin{aligned}
\chi_{i-0.5} - \chi_i &= -\frac{1}{2}\frac{\partial \chi}{\partial x}\Delta x + \frac{1}{4}\frac{\partial^2 \chi}{\partial x^2}\frac{\Delta x^2}{2} - \frac{1}{8}\frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} + \mathcal{O}(\Delta x^4) \quad \text{(a)}\\
\chi_{i+0.5} - \chi_i &= +\frac{1}{2}\frac{\partial \chi}{\partial x}\Delta x + \frac{1}{4}\frac{\partial^2 \chi}{\partial x^2}\frac{\Delta x^2}{2} + \frac{1}{8}\frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} + \mathcal{O}(\Delta x^4) \quad \text{(b)}\\
\chi_{i-1.5} - \chi_i &= -\frac{3}{2}\frac{\partial \chi}{\partial x}\Delta x + \frac{9}{4}\frac{\partial^2 \chi}{\partial x^2}\frac{\Delta x^2}{2} - \frac{27}{8}\frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} + \mathcal{O}(\Delta x^4) \quad \text{(c)}\\
\chi_{i+1.5} - \chi_i &= +\frac{3}{2}\frac{\partial \chi}{\partial x}\Delta x + \frac{9}{4}\frac{\partial^2 \chi}{\partial x^2}\frac{\Delta x^2}{2} + \frac{27}{8}\frac{\partial^3 \chi}{\partial x^3}\frac{\Delta x^3}{6} + \mathcal{O}(\Delta x^4) \quad \text{(d)}
\end{aligned} \tag{B.3}$$

Analogous to the construction of the advection schemes, the flux estimators can be con-

structed by linearly superposing those equations.

$$\begin{aligned}
(a) \qquad & \chi_i = \chi_{i-0.5} + \mathcal{O}(\Delta x) \\
(a) + (b) \qquad & \chi_i = \tfrac{1}{2}\big(\chi_{i+0.5} + \chi_{i-0.5}\big) + \mathcal{O}(\Delta x^2) \\
(c) - 3(b) - 6(a) \qquad & \chi_i = \tfrac{1}{8}\big(- \chi_{i-1.5} + 6\chi_{i-0.5} + 3\chi_{i+0.5}\big) + \mathcal{O}(\Delta x^3) \\
(d) + (c) - 9(b) - 9(a) \qquad & \chi_i = \tfrac{1}{16}\big(- \chi_{i-1.5} + 9\chi_{i-0.5} + 9\chi_{i+0.5} - \chi_{i+1.5}\big) + \mathcal{O}(\Delta x^4)
\end{aligned} \tag{B.4}$$

How to derive the other flux divergence variants?

These flux estimatators correspond to the ones derived by *Tremback et al.* (1987). However, as they note, those flux estimators do not reduce to the advection schemes derived in appendix A.

C. List of symbols and hints on notation

Table C.1.: List of mathematical symbols as they are used in the documentation.

Symbol	Unit	Description
a	m	Radius of the earth
α	$\mathrm{m^3\,kg^{-1}}$ or 1	Specific volume or coupling constant for Robert-Asselin filter
β	$\mathrm{s^{-1}}$	Secondary Coriolis parameter $= f_y = 2a^{-1}\Omega\cos\varphi$
c_p, c_v	$\mathrm{J\,kg^{-1}\,K^{-1}}$	Specific heat capacity of air for constant pressure and constant volume
γ	rad	Deformation angle: the angle between the x-axis and the axis of dilatation
D	$\mathrm{m^2\,s^{-1}}$	Damping coefficient for scale-selective damping
$\mathcal{D}$	$\mathrm{s^{-1}}$	Divergence $= u_x + v_y$
δ	$\mathrm{s^{-1}}$	Total deformation $= (\delta_+^2 + \delta_\times^2)^{1/2}$
δ_+, $\delta_\times$	$\mathrm{s^{-1}}$	Streching deformation $= u_x - v_y$ and shear deformation $= u_y + v_x$
η	m or Pa	General vertical coordinate, stands for z or p
$\dot{\eta}$	$\mathrm{m\,s^{-1}}$ or $\mathrm{Pa\,s^{-1}}$	General vertical velocity, stands for w or ω
f	$\mathrm{s^{-1}}$	Primary Coriolis parameter $= 2\Omega\sin\varphi$
f'	$\mathrm{s^{-1}}$	Secondary Coriolis parameter $= 2\Omega\cos\varphi$
$\boldsymbol{F}_r$	$\mathrm{m\,s^{-2}}$	General forcing terms in for the momentum equation, e.g. frictional terms
$\boldsymbol{g}$, g	$\mathrm{m\,s^{-2}}$	Graviational acceleration of the Earth, $\boldsymbol{g} = (0, 0, g)$
$\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$	1	Unit vectors in x, y and z (or equivalent for different coordinate systems) directions
J	$\mathrm{J\,kg^{-1}\,s^{-1}}$	Diabatic heating in the temperature tendency equation
λ	rad	Longitude
N	$\mathrm{s^{-1}}$	Brunt-Väisälä frequency
ω	$\mathrm{Pa\,s^{-1}}$	Vertical velocity in pressure coordinates
$\boldsymbol{\omega}$	$\mathrm{s^{-1}}$	Three-dimensional vorticity vector

Symbol	Unit	Description
Ω	s^{-1}	Rotation frequency of the earth
p	Pa	Pressure
p_s	Pa	Surface pressure
φ	rad	Latitude
Φ	$\mathrm{m}^2\,\mathrm{s}^{-2}$	Geopotential
q	s^{-1}	Quasi-geostrophic potential vorticity
$\mathcal{Q}$, $\mathcal{Q}_p$	$\mathrm{m\,s}^{-2}$, $\mathrm{m}^2\,\mathrm{s}^{-2}\,\mathrm{Pa}^{-1}$	Diabatic heating in z- and p-coordinates
ψ	$\mathrm{m}^2\,\mathrm{s}^{-1}$	Horizontal streamfunction
r	s^{-1}	Damping coefficient for Ekman scale-independent damping
R	$\mathrm{J\,kg}^{-1}\,\mathrm{K}^{-1}$	Ideal gas constant for air
ρ	$\mathrm{kg\,m}^{-3}$	Density
σ	1	Terrain-following vertical coordinate defined by p/p_s
t	s	Time
T	K	Temperature
θ	K	Potential temperature
χ	$\mathrm{m}^2\,\mathrm{s}^{-1}$ or any	Velocity potential or meta-symbol symbolising other symbols
$\boldsymbol{u}$, $\boldsymbol{v}$	$\mathrm{m\,s}^{-1}$	Three-dimensional and two-dimensional velocity vectors
u, v, w	$\mathrm{m\,s}^{-1}$	Velocity vector components in x, y and z directions
ζ	s^{-1}	Vertical component of the vorticity vector $= \boldsymbol{\omega}^{(3)}$

Table C.2.: List of mathematical and numerical operators, using χ as the meta-symbol.

Operator	Description
$\dfrac{d\chi_1}{d\chi_2}$	Total derivative of χ_1 with χ_2
$\dfrac{\partial\chi_1}{\partial\chi_2}$	Partial derivative of χ_1 with χ_2
χ_t, χ_x, χ_y, χ_z, χ_p	Partial derivatives of χ along the coordinate axes t, x, y, z and p
∇	Nabla-Operator
∇^2	Laplacian operator
$J(\chi_1, \chi_2)$	Jacobian of χ_1 and χ_2
$\Delta\chi$	A small variation of χ
$\boldsymbol{\chi}^{(i)}$	The i'th component of the vector $\boldsymbol{\chi}$
χ_i	The discretised χ at the grid cell i in x-direction
χ_j	The discretised χ at the grid cell j in y-direction
χ_k	The discretised χ at the grid cell k in the vertical

Bibliography

Boos, W. R., and Z. Kuang, Dominant control of the south asian monsoon by orographic insulation versus plateau heating, *Nature*, *463*, 218–222, doi:doi:10.1038/nature08707, 2010.

Chang, E. K. M., S. Lee, and K. L. Swanson, Storm track dynamics, *Journal of Climate*, *15*(16), 2163–2183, doi:10.1175/1520-0442(2002)015⟨02163:STD⟩2.0.CO;2, 2002.

Cook, K. H., and I. M. Held, The stationary response to large-scale orography in a general circulation model and a linear model, *Journal of the Atmospheric Sciences*, *49*(6), 525–539, doi:10.1175/1520-0469(1992)049⟨0525:TSRTLS⟩2.0.CO;2, 1992.

Held, I. M., The gap between simulation and understanding in climate modeling, *Bull. Amer. Meteor. Soc.*, *86*(11), 1609–1614, 2005.

Hoskins, B. J., and D. J. Karoly, The steady linear response of a spherical atmosphere to thermal and orographic forcing, *J. Atmos. Sci.*, *38*(6), 1179–1196, 1981.

Munksgaard, N., Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients, *ACM Trans. Math. Softw.*, *6*, 206–219, doi:http://doi.acm.org/10.1145/355887.355893, 1980.

Park, H.-S., J. C. H. Chiang, and S.-W. Son, The role of the central asian mountains on the midwinter suppression of north pacific storminess, *Journal of the Atmospheric Sciences*, *67*(11), 3706–3720, doi:10.1175/2010JAS3349.1, 2010.

Ringler, T. D., and K. H. Cook, Factors controlling nonlinearity in mechanically forced stationary waves over orography, *Journal of the Atmospheric Sciences*, *54*(22), 2612–2629, doi:10.1175/1520-0469(1997)054⟨2612:FCNIMF⟩2.0.CO;2, 1997.

Serreze, M. C., M. M. Holland, and J. Stroeve, Perspectives on the arctic's shrinking sea-ice cover, *Science*, *315*(5818), 1533–1536, doi:10.1126/science.1139426, 2007.

Smith, R. B., Linear theory of stratified hydrostatic flow past an isolated mountain, *Tellus*, *32*(4), 348–364, doi:10.1111/j.2153-3490.1980.tb00962.x, 1980.

Smolarkiewicz, P. K., The multi-dimensional crowley advection scheme, *Monthly Weather Review*, *110*(12), 1968–1983, doi:10.1175/1520-0493(1982)110⟨1968:TMDCAS⟩2.0.CO;2, 1982.

Son, S.-W., M. Ting, and L. M. Polvani, The effect of topography on storm-track intensity in a relatively simple general circulation model, *Journal of the Atmospheric Sciences*, *66*(2), 393–411, doi:10.1175/2008JAS2742.1, 2009.

Stroeve, J., M. M. Holland, W. Meier, T. Scambos, and M. Serreze, Arctic sea ice decline: Faster than forecast, *Geophys. Res. Lett.*, *34*(9), L09,501–, 2007.

Ting, M., and L. Yu, Steady response to tropical heating in wavy linear and nonlinear baroclinic models, *Journal of the Atmospheric Sciences*, *55*(24), 3565–3582, doi:10.1175/1520-0469(1998)055⟨3565:SRTTHI⟩2.0.CO;2, 1998.

Tremback, C. J., J. Powell, W. R. Cotton, and R. A. Pielke, The forward–in–time upstream advection scheme: Extension to higher orders, *Monthly Weather Review*, *115*(2), 540–555, doi:10.1175/1520-0493(1987)115⟨0540:TFTUAS⟩2.0.CO;2, 1987.

Valdes, P. J., and B. J. Hoskins, Nonlinear orographically forced planetary waves, *Journal of the Atmospheric Sciences*, *48*(18), 2089–2106, doi:10.1175/1520-0469(1991)048⟨2089:NOFPW⟩2.0.CO;2, 1991.

van der Vorst, H. A., Bi-CGstab: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM Journal on Scientific Computing*, *13*(2), 631–644, doi:DOI:10.1137/0913035, 1992.

Wang, L., and P. J. Kushner, Interpreting stationary wave nonlinearity in barotropic dynamics, *Journal of the Atmospheric Sciences*, *67*(7), 2240–2250, doi:10.1175/2010JAS3332.1, 2010.

Williams, P. D., The raw filter: An improvement to the robert-asselin filter in semi-implicit integrations, *Mon. Wea. Rev.*, *139*(6), 1996–2007, 2010.