Übung zur Meteorologischen Modellierung (Numerik): Das barotrope Modell

Beside its importance for conceptual and analytical studies of the large- and synoptic-scale circulation, the barotropic model is of historical interest. It was the first (working) numerical weather prediction model (in its quasi-geostrophic non-divergent version; Charney, J. G., Fjortoft, R. and von Neumann, J. 1950: Numerical integration of the barotropic vorticity equation. *Tellus, 2*, 237–54).

The following gives the basics on developing and programming such a model (supplementing the lecture). The aim is to program, to test and to run your own barotopic model.

1. Equations

The barotropic model represents the dynamic of a homogeneous incompressible fluid which is in hydrostatic balance. This may be expressed using the horizontal equations of motion (for *u,v*) and the continuity equation. Assuming barotropic conditions (no change of velocities with height, i.e. no thermal wind, no vertical advection), no friction, and using cartesian (z-) coordinates, the equation of motions read

$$\begin{aligned}\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}-fv &= -\frac{1}{\rho}\frac{\partial P}{\partial x}\\ \frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+fu &= -\frac{1}{\rho}\frac{\partial P}{\partial y}\end{aligned}$$

With Coriolis parameter *f*, pressure *p*, and (constant) density ρ.

The continuity equation with constant ρ reads

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0$$

With vertical velocity *w*. By integrating the hydrostatic equation

$$\frac{\partial P}{\partial z}=-g\rho$$

P/ρ in the equations of motions can be replaced by the geopotential *gh*:

$$\begin{aligned}\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}-fv &= -\frac{\partial gh}{\partial x}\\ \frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+fu &= -\frac{\partial gh}{\partial y}\end{aligned}$$

Integrating the continuity equation from z=*0* (with *w(0)=0*) to *h* and using

$$w(h)=\frac{dh}{dt}=\frac{\partial h}{\partial t}+u\frac{\partial h}{\partial x}+v\frac{\partial h}{\partial y}$$

a prognostic equation for the geopotential *gh* is optained (needed in the equations of motion):

$$\frac{\partial gh}{\partial t} + u\frac{\partial gh}{\partial x} + v\frac{\partial gh}{\partial y} = -gh(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y})$$

These equations form a closed system with three unknows: the so called 'primitive equations' of the divergent barotropic model (or 'shallow water model')

Although this model may already be solved numerically, we will apply some simplifications (approximations) to make the numerical treatment much easier (e.g., we have already used the hydrostatic balance to obtain a diagnostic relation between pressure and geopotential). First, the Coriolis parameter f is linearized about a reference latitude y_0 (with $f(y_0)=f_0$):

$$f = f_0 + \frac{df}{dy}y = f_0 + \beta y$$

Which is commonly called 'beta plane approximation'.

A second approximation, valid for the large (synoptic) scale circulation in the mid-latitudes is the quasi-geostrophic approximation: Using scale analysis (see lecture Theoretical Meteorology) it can been shown that geostrophic equilibrium is a first order approximation for small Rossby numbers ($Ro=U/(Lf_0)$ << 1). by (i) splitting the flow (u,v) and the diviation of the geopotential from a mean value h_0 into a geostrophic (u_g,v_g,,h_g) and a (small; O(Ro)) ageostrophic (u_a,v_a,h_a) part ($u=u_g+u_a$,$v=v_g+v_a$,$h=h_0+h_g+h_a$), (ii) inserting this into the shallow water equations, (iii) applying scale analysis, (iv) eliminating geostrophy, (v) using the non-divergence of the geostrophic part, and (vi) neglecting all terms smaller than O(Ro), we derive a system for the temporal change of the geostrophic flow:

$$\begin{aligned}
\frac{\partial u_g}{\partial t} + u_g\frac{\partial u_g}{\partial x} + v_g\frac{\partial u_g}{\partial y} - f_0 v_a - \beta y v_g &= -\frac{\partial g h_a}{\partial x}\\
\frac{\partial v_g}{\partial t} + u_g\frac{\partial v_g}{\partial x} + v_g\frac{\partial v_g}{\partial y} + f_0 u_a + \beta y u_g &= -\frac{\partial g h_a}{\partial y}\\
\frac{\partial g h_g}{\partial t} + u_g\frac{\partial g h_g}{\partial x} + v_g\frac{\partial g h_g}{\partial y} = -g h_0(\frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y})
\end{aligned}$$

Introducing the geostrophic vorticity $\zeta_g = \frac{\partial v_g}{\partial x} - \frac{\partial u_g}{\partial y}$, the first two equations can be combined to the (quasi-geostrophic) barotropic vorticity equation, and the system reduces to

$$\frac{\partial \zeta_g}{\partial t} + u_g\frac{\partial \zeta_g}{\partial x} + v_g\frac{\partial \zeta_g}{\partial y} + \beta v_g = -f_0(\frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y})$$

$$\frac{\partial g h_g}{\partial t} + u_g\frac{\partial g h_g}{\partial x} + v_g\frac{\partial g h_g}{\partial y} = -g h_0(\frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y})$$

Using geostrophic balance, a geostrophic streamfunctin can be introduced, $\psi=gh_g/f_0$, with

$$u_g = -\frac{\partial\Psi}{\partial y}$$

$$v_g = \frac{\partial\Psi}{\partial x}$$

$$\zeta_g = \nabla^2\Psi = \frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2}$$

Eliminating the divergence of the ageostrophic flow and using the geostrophic streamfunction we derive the quasi-geostrophic divergent barotropic model (quasi-geostrophic shallow water model):

$$\frac{\partial(\nabla^2 - \lambda^{-2})\Psi}{\partial t} = -\frac{\partial\Psi}{\partial x}\frac{\partial(\nabla^2 - \lambda^{-2})\Psi}{\partial y} + \frac{\partial\Psi}{\partial y}\frac{\partial(\nabla^2 - \lambda^{-2})\Psi}{\partial x} - \beta\frac{\partial\Psi}{\partial x} = -J(\Psi,(\nabla^2 - \lambda^{-2})\Psi) - \beta\frac{\partial\Psi}{\partial x}$$

With only one prognostic variable, the streamfunction, this model completely describes the flow (within the scope of the applied approximations). Here λ is the Rossby radius of deformation ($\lambda=(gh_0)^{1/2}/f_0$), and $J(a,b)=(\partial a/\partial x\ \partial b/\partial y - \partial a/\partial y\ \partial b/\partial x)$ the Jacobi operator. Note that the term $\lambda^{-2}\psi$ in the Jacobi operator (i.e. the advection of thickness by the geostrophic flow) is only formally kept in the equation, since $J(\psi,-\lambda^{-2}\psi)=0$.

An even more drastic approximation is to totally disregard the divergence. Then, the dynamic is reduced to the non-divergent barotropic model (the non-divergent barotropic vorticity equation), which, as written above, was utilized for the first (working) numerical weather prediction:

$$\frac{\partial\zeta_g}{\partial t} + u_g\frac{\partial\zeta_g}{\partial x} + v_g\frac{\partial\zeta_g}{\partial y} + \beta v_g = 0$$

or

$$\frac{\partial\nabla^2\Psi}{\partial t} = -\frac{\partial\Psi}{\partial x}\frac{\partial\nabla^2\Psi}{\partial y} + \frac{\partial\Psi}{\partial y}\frac{\partial\nabla^2\Psi}{\partial x} - \beta\frac{\partial\Psi}{\partial x} = -J(\Psi,\nabla^2\Psi) - \beta\frac{\partial\Psi}{\partial x}$$

A third version of the barotropic model is the so called equivalent barotropic model, which has been often used in the beginning of numerical weather prediction and for theoretical studies. To relax the barotropic condition (no thermal wind), but keeping a simple system, it is assumed that strength but not the direction of the wind may change with height. The vertical profile of the flow *A(p)* is assumed to be independent in space *(x,y)* and time *(t)*. Then, the horizontal flow *(u,v)* can be written as *A(p)(<u>(x,y,t),<v>(x,y,t))*, with vertical average <.> (and <A> =1). Here (common in meteorological applications), p-coordinates are used in the vertical. Introducing the varable $\psi^*=<A^2>\psi$ we obtain:

$$\frac{\partial(\nabla^2 - A(p_s)\lambda^{-2})\Psi^*}{\partial t} = -J(\Psi^*,\nabla^2\Psi^*) - \beta\frac{\partial\Psi^*}{\partial x}$$

This model is valid for level *p** with $A(p^*)=<A^2>$, the so called equivalent-barotropic level. Typical wind profiles give a *p** of about 600-500hPa. This level is in practice identical with the level of minimum divergence.

Formaly, the difference between the three quasi-geostrophic versions lies in the factor $A(p_s)$, in the term representing the temporal change of the geopotential (generation of vorticity by stretching): $A(p_s)=0$ results in the non-divergent model where vorticity is changed by advection of absolut vorticity only. $A(p_s)=1$ results in the shallow water equations. Here, advection of absolut vorticity and change of thickness due to divergence/convergence contribute to the vorticity tendency. $A(p_s)$ determines the effect of divergence on the vorticity production within the equivalent barotropic layer. In practice, the choise of $A(p_s)$ controls the phase speed of Rossby waves which has consequences for the quality of the weather prediction. Thus, $A(p_s)$ may be used as a tuning parameter to improve the forecast scill empirically.

2. Numerical Solution

To solve the barotropc modell numerically, different methods may be applied. (grid-point, semi-Lagrange, spectral, finite elements). Here, we will use a 'classical' grid-point method. We focus on the simplest barotropic model: The non-divergent version. As might be seen from the equations, going to the quasi-geostrophic shallow water model or to the quasi-geostrophic equivalent barotropic model is easy, while solving the primitive equation version needs some more effort.

From the respective equation

$$\frac{\partial \nabla^2 \Psi}{\partial t} = -\frac{\partial \Psi}{\partial x}\frac{\partial \nabla^2 \Psi}{\partial y} + \frac{\partial \Psi}{\partial y}\frac{\partial \nabla^2 \Psi}{\partial x} - \beta \frac{\partial \Psi}{\partial x} = -J(\Psi, \nabla^2 \Psi) - \beta \frac{\partial \Psi}{\partial x}$$

We see that we need to compute the temporal evolution of the streamfunction, which only depends on the streamfunction itself. From streamfunction we are able to diagnose all variable needed, like wind, vorticity, and geopotential. Thus, the numerical aim is to compute a new streamfunction field (on a discrete time axis) from a given one (plus appropriate boundary conditions), analog to the numerical solution of the (nonlinear) advection problem (see Meteorologische Modellierung/Numerik part one).

In principle we have to do the following: Starting from initial condition for the streamfunction (or the geopotential) and using suitable boundary conditions (in x,y) we have to solve $(\partial\psi/\partial t)_{t=0}$. To to this we have (i) to compute the right hand side (forcing) from a given ψ. This is done by numerical approximations/discretizations for the derivatives in space (or the Jacobi operator). (ii) Then we have to solve the Poisson- (or Helmholtz-) equation

$$\nabla^2 \Theta = G(x, y)$$

to get $(\partial\psi/\partial t)_{t=\Delta t}$. (iii) Finally we have to do the temporal extrapolation (time step) to obtain $\psi_{t=\Delta t}$. These three steps (i-iii) are repeated until we have the final ψ at the end of the prediction (t=T).

Before solving the model numerically using a grid point scheme, we need to define the model domain and a suitable grid. in addition we need to choose adequate values for the free parameters (f_0, β, g, etc.). Since the equations are formulated in cartesian coordinates on a β-plane, a canal in zonal direction (for meteorological application) or a squared basin (ocean) may be chosen as the domain. a proper grid may be a cartesian grid with NxM grid points. The gridsize (resolution) in x- and y-direction needs to be determined from the processes under consideration and the available computer power. The resolution and the processes also set the maximum time step since we have to fulfill the Courant-Friedrich-Levy criterion. In the non-divergent (quasi-geostrophic) model, the time step is in general determined by the phase speed of the (relatively slow) Rossby-waves. Depending on the mean flow and the scale of the largest Rossby wave, a time step of about 1 hour is possible using a grid size of ca. 300km (using primitive equations, much faster (e.g. gravity) waves would require a much shorter time step).

As the domain is limited, we need to introduce appropriate boundary conditions to compute, e.g., the derivatives. For a canal, cyclic boundary conditions can be used in x-direction (i.e. $\psi(0,y) = \psi(N,y)$ und $\psi(N+1,y) = \psi(1,y)$, with 1 and N indicating the first and last grid point in x). At the meridional (y) boundaries it is commonly assumed that the normal component of the flow vanishes (no transport across the boundary). This is obtained by setting ψ constant in x (i.e. $\psi(x,0)=c$, $\psi(x,M+1)=c$).

To implement the model procedure we finally need to determine the numerical approximations for the individual parts, i.e. computing the Jacobi-operator, the Laplace-operator, the horizontal derivatives, solving the Poisson- (Helmholz-) equation, and extrapolating in time.

Jacobi-Operator
There are various methods to approximate the Jacobi-operator by finite differences, which follow from the analytical expression:

$$\begin{aligned} J(a,b) &= \frac{\partial a}{\partial x}\frac{\partial b}{\partial y} - \frac{\partial a}{\partial y}\frac{\partial b}{\partial x} \\ &= \frac{\partial}{\partial x}(a\frac{\partial b}{\partial y}) - \frac{\partial}{\partial y}(a\frac{\partial b}{\partial x}) \\ &= \frac{\partial}{\partial y}(b\frac{\partial a}{\partial x}) - \frac{\partial}{\partial x}(b\frac{\partial a}{\partial y}) \end{aligned}$$

For the discretization we will use the respective grid point itself, 0=(i,j), and the surrounding 8 points (1-8), which are located as follows:

6 = (i-1,j+1)	2 = (i,j+1)	5 = (i+1,j+1)
3 = (i-1,j)	0 = (i,j)	1 = (i+1,j)
7 = (i-1,j-1)	4 = (i,j-1)	8 = (i+1,j-1)

For the above three expressions of the Jacopi-operator we obtain the following approximations:

$$\begin{aligned} J_1 &= \frac{1}{4\Delta x\Delta y}\{(a_1-a_3)(b_2-b_4)-(a_2-a_4)(b_1-b_3)\} \\ J_2 &= \frac{1}{4\Delta x\Delta y}\{a_1(b_5-b_8)-a_3(b_6-b_7)-a_2(b_5-b_6)+a_4(b_8-b_7)\} \\ J_3 &= \frac{1}{4\Delta x\Delta y}\{b_2(a_5-a_6)-b_4(a_8-a_7)-b_1(a_5-a_8)+b_3(a_6-a_7)\} \end{aligned}$$

The actual approximation (J) results from a linear combination of these J_1, J_2 and J_3: $J = \alpha J_1 + \beta J_2 + \gamma J_3$. Arakawa (1966) shows that $\alpha=\beta=\gamma=1/3$ is the best combination as it warrants the important conservations of energy and enstrophy of the system.

Laplace-Operator

The Laplace-operator, $\nabla^2 = \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, can be approximated by centered differences:

$$\nabla^2\Psi(i,j) = \nabla^2\Psi_0 = \frac{1}{\Delta x^2}(\Psi_1 + \Psi_3 - 2\Psi_0) + \frac{1}{\Delta y^2}(\Psi_2 + \Psi_4 - 2\Psi_0)$$

using the same indices as for the Jacobi-operator.

Horizontal Derivatives

Also for the derivatives in x- and y-direction ($\frac{\partial}{\partial x}, \frac{\partial}{\partial y}$) we can use central differences:

$$\begin{aligned}\frac{\partial\Psi}{\partial x} &= \frac{1}{2\Delta x}(\Psi_1 - \Psi_3)\\ \frac{\partial\Psi}{\partial y} &= \frac{1}{2\Delta y}(\Psi_2 - \Psi_4)\end{aligned}$$

Poisson- (Helmholtz-) Equations

An essential part of the barotropic quasi-geostrophic model is to solve the Poisson- ($\nabla^2\Theta = G(x,y)$) or Helmholtz- ($(\nabla^2 + \lambda)\Theta = G(x,y)$) equation to compute Θ from a known $G(x,y)$ at each time step. A potential procedure is the Gauß-Seidel method or it improvement, the Successive Over-Relaxation (SOR), which in the following will be explained using the Poisson-equation:

Starting point is the discrete representation of the Laplace-operator (see above). Inserting it we obtain for the discrete Poisson-equation

$$\nabla^2\Theta_0 - G_0 = \frac{1}{\Delta x^2}(\Theta_1 + \Theta_3 - 2\Theta_0) + \frac{1}{\Delta y^2}(\Theta_2 + \Theta_4 - 2\Theta_0) - G_0 = 0$$

from this we may formally compute Θ_0 for every grid point. However, Θ_0 depends on the (also unknown) Θ at grid points 1,2,3 und 4 ab. Thus, a direct solution is not possible, but an approximative solution can be found by iteration: First we compute the error ε_0=($\nabla^2\Theta_0 - G_0$) for a prescribed initial field Θ (e.g. 0 at all points or, perhaps better, Θ at the previous time step. ε_0 is computed following the equation above. Next we correct Θ_0 with ε_0 to obtain a new (improved) Θ_0 (=Θ'_0):

$$\Theta'_0 = \Theta_0 + \varepsilon_0 /(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2})$$

$$= (\frac{1}{\Delta x^2}(\Theta_1 + \Theta_3) + \frac{1}{\Delta y^2}(\Theta_2 + \Theta_4) - G_0)/(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2})$$

After computing a new Θ for every grid point the procedure (computing ε_0 and correcting Θ_0) is repeated until a measure of the total error (e.g. the sum of all errors squared) is below a predefined limit, or if a given number of iterations has been made. (Note: as can be seen, we can compute a new Θ without explicitly computing ε_0).

However, this procedure is not yet the Gauß-Seidel method but the Jacobi method. Though the Jacobi method looks nice, it can, in general, not be used in practice, as it turns out to convert only very slowly against the real (analytic) solution. The Gauß-Seidel method substantially improves the convergence rate. The idea is not to use only the 'old' Θ to compute the new Θ'_0 but also to utilize the new Θ' wherever it is possible. If, for example, the domain passed line-by-line from the upper left to the lower right, we can already use the new Θ' at point 2 and 4. Thus, the new Θ'_0 is given by

$$\Theta'_0 = (\frac{1}{\Delta x^2}(\Theta_1 + \Theta'_3) + \frac{1}{\Delta y^2}(\Theta_2 + \Theta'_4) - G_0)/(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2})$$

Typically we need $N^2\ p/4$ iterationsn for a $N{\times}N$ grid to decrease the initial error by factor 10^{-p} which is an improvement of factor 2 compared to the Jacobi method. Still this is a relatively large effort. A substantial reduction can be achieved by using the Successive Over-Relaxation (SOR) method. Here we do not correct the old Θ by ε_0, but over-correct it by $\omega\ \varepsilon_0$, with $1 < \omega < 2$:

$$\Theta'_0 = \Theta_0 + \omega\varepsilon_0 /(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2})$$

$$= \Theta_0 + \omega(\frac{1}{\Delta x^2}(\Theta_1 + \Theta'_3 - 2\Theta_0) + \frac{1}{\Delta y^2}(\Theta_2 + \Theta'_4 - 2\Theta_0) - G_0)/(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2})$$

Again we use also the available new Θ. The number of necessary iterations to reduce the initial error by 10^{-p} is diminished to $N\ p/3$, using an optimal ω, that is by a factor of N in comparison to Gauß-Seidel (note that N is typically a very large). Unfortunately, the choice of ω is difficult, and, for normal applications, only possible using trial-and-error.

The solution of a Helmholtz equation follows the same strategy expanding the discrete Laplace-operator by the term $\lambda\Theta_0$.

Note that there are various other methods to solve the Poisson- (Helmholz-) problem, including the use of Fourier expansion of multi-grid methods.

Derivative in Time
To extrapolate into the future, i.e. to compute the values for the next time step, we need to approximate the derivative in time numerically. As we have ssen for the decay or advection equation (see lecture) the exist various methods. Here we may use any numerically stabe

method. Implicit techniques, however, may not be aplicable due to the complexity of the problem (non-linearities). In meteorology the leapfrog scheme (three level scheme, see lecture) together with the Robert-Asselin filter (to reduce the computational mode) is commonly used, and may also be used here:

$$\Psi(t+\Delta t) = \Psi^*(t-\Delta t) + 2\Delta t\left(\frac{\partial \Psi}{\partial t}\right)$$

$$\Psi^*(t) = \Psi(t) + F\left[\Psi(t+\Delta t) + \Psi^*(t-\Delta t) - 2\Psi(t)\right]$$

With filter constant *F* (typically F=0.1). As the leapfrog scheme needs two time levels to compute the next one, another scheme like the explicit Euler needs to be used at the beginning of the integration.

3. Implementation

The implementation of the above described barotropic model needs a relatively large amount of programming, which is naturally prone to errors (theoretical and technical). Therefor it is preferable to structure and program the model in smaller units, which can be tested separately and in combination (modular structure). Following this way, we create bit by bit subroutines of individual components, which then are combined within a main program. These components may be:

- Input and output
- Grid definition
- Horizontal derivatives
- Laplace-operators
- Jacobi-operators
- Temporal extrapolation (leap frog and Euler)
- Poisson-equation
- Boundary conditions
- Others

Building the main program may be the start of the task. It defines the work flow of the model and the interaction between the different components. Again, the structuring can be facilitated by using subroutines (and/or functions) for individual parts. In addition, we have to decide and to define the (global) variables (scalars and arrays) and to give them (self explaining) names. These variables may be defined within a (fortran) 'modul'. Calls to respective subroutines, and 'dummy' subroutines (subroutines which at the moment do nothing) may also already be included in the code. Later those dummies are replaced by the actual routines. After coding the main program it needs to be tested (compiled).

In a second step we may code the input and output of the model. Parameter controlling the model run, and, perhaps, initial conditions for the streamfunctions, etc., need to be read in. Output like the predicted streamfunction, winds, etc., need to written out for a given output interval. Additionally some control parameter (e.g. number of iterations in the Poisson solver) may be written. The choice of the output format is an important issue to make the results easy to use.

After completing the more technical work we can start with the physical part of the model. First we have to initialize the model. Parameter, which have not been set or which depend on the input need to be computed/set. For example, defining the grid (latitudes, longitudes, distances, etc.) is part of the initialization, as well as the initialization of the prognostic field.

Finally, the so far dummy routines for all components need to be replaced. Here, intensive testing is indispensable. For complicated routines, e.g. the Poisson-solver, this may be done independent from the rest of the model.

After completing the coding, the physical correctness of the model needs to be tested. That is, the models numerical solution for particular cases needs to be compared with other results. In the best case with analytic solution, or with results from other models which have been published. For the barotropic model, the Rossby (-Haurwitz-) wave is a good test case (s. lecture 'Theoretische Meteorologie') as it is an analytic solution. If the model agrees with with known solutions, we can start doing some experiments.

4. The Code

The ultimate aim is to build a non-divergent barotropic model as described above. However, as already noted, substantial time for programing is needed, which may exceed the available time for the lecture. Thus, we may use the prepared program 'baro.f90' as a starting point, which already contains some of the components (infos will be given during the lecture).

To complete baro.f90 we need to replace the following dummy subroutines:

- ***sor***: Solving the Poisson-equation (inverse Laplace) with 'Successive Over-Relaxation' (SOR)
- ***mkdfdx***: Compute derivatives in x-direction
- ***laplace***: Apply the Laplace-operator to a field
- ***jacobi***: Apply the Jacobi-operator to two fields
- ***euler***: Do explicit Euler time step
- ***leapfrog***: Do the leapfrog time step (with filter)

sor

Solving the Poisson-equation using the SOR method is the most complex part of the model. We need the dimensions of the field, the gridpoint distances in x- and y-direction, and the input filed (the vorticity tendencies). Outout is the inverse Laplace of the input (the streamfunction tendencies). This is accounted for in the dummy subroutine sor:

```
      subroutine sor(pdf,pf,pdx,pdy,kx,ky)
      implicit none
!
!     subroutine sor compute the inverse Laplacian from a given field
!     by using the Successive OverRelaxation method (SOR)
!
      integer :: kx                    ! x dimension
      integer :: ky                    ! y dimension
      real :: pdx                      ! x grid point distance
      real :: pdy                      ! y grid point distance
      real :: pdf(0:kx+1,0:ky+1)       ! input: field
      real :: pf(0:kx+1,0:ky+1)        ! output: inverse Laplacian of input
!
      return
      end
```

Note that dimensions of the fields (arrays) include the boundaries (0,N+1). To complete the sor subroutine we may add the following parts:

1. Compute the initial error (ε_0=($\nabla^2\Theta_0 - G_0$), s. sectiont SOR above)
2. (over-) correct the initial solution
3. Compute the new error

4. Repeat 2 and 3 until the measure of the error is 'small enough' (or until the error-measure is reduced by factor x)

It may be convenient to note: a) Set the boundary conditions after each iteration. b) The computer precision limits the reduction of the error. Start with a small reduction, say factor 10. c) Set a maximum of iterations not to produce a dead-loop (theoretically you need (N p/3) iterations for NxN to reduce the error by 10^p).

mkdfdx

To compute x-derivatives using central differences we need the dimension of the field, the grid distance in x-direction, and the input field. Output is the x-derivative of the input field. Therefore, the dummy subroutine mkdfdx is:

```
      subroutine mkdfdx(pf,pdfdx,pdx,kx,ky)
      implicit none
!
!     subroutine mkdfdx computes x derivation from a field
!     using central differences
!
      integer :: kx                  ! x-dimension
      integer :: ky                  ! y-dimension
      real :: pdx                    ! x grid distance
      real :: pf(0:kx+1,0:ky+1)      ! input: field
      real :: pdfdx(0:kx+1,0:ky+1)   ! output: dfield/dx
!
      return
      end
```

As we already used central differences in the last semester (advection), completing mkdfdy may be not be too dificult.

laplace

The Laplace-operator should be approximated by central differences. We need the dimension of the field, the grid distance in x- and y-direction, and the input field. Output is the Laplace of the input:

```
      subroutine laplace(pf,pdf,pdx,pdy,kx,ky)
      implicit none
!
!     subroutine laplace computes the laplacian from a field
!
      integer :: kx                  ! x-dimension
      integer :: ky                  ! y-dimension
      real :: pdx                    ! x grid distance
      real :: pdy                    ! y grid distance
      real :: pf(0:kx+1,0:ky+1)      ! input: field
      real :: pdf(0:kx+1,0:ky+1)     ! output: Laplacian
!
      return
      end
```

To complete laplace is straightforward.

jacobi

Coding the conservative Jacobi-operater (afer Arakawa, see above) needs a bit more work. In particular we have to be careful using the correct indices. We need the dimensions, the grid distance in x- and y-direction, and the input fields. Output is the Jacobi-operator applied to the input fields:

```
      subroutine jacobi(p1,p2,pj,pdx,pdy,kx,ky)
      implicit none
!
!     subroutine jacobi computes the jacobi operator according to
!     Arakawa (1966)
!
!     jacobi(1,2)=(j1+j2+j3)/3. after Arakawa 1966
!
      integer :: kx                ! x-dimension
      integer :: ky                ! y-dimension
      real :: p1(0:kx+1,0:ky+1)    ! input: field 1
      real :: p2(0:kx+1,0:ky+1)    ! input: field 2
      real :: pj(0:kx+1,0:ky+1)    ! output: jacobi(1,2)
      real :: pdx                  ! x grid distance
      real :: pdy                  ! y grid distance
!
      return
      end
```

We may use temporary arrays for the three different Jacobis, which needs to be declared too.

euler

Coding the explicit Euler is an easy task (remember the decay equation). We need the dimension of the field, the time step, the tendency, and the actual field (time=t). Output is the filed at the next time step (t+1):

```
      subroutine euler(pfm,pdfdt,pf,pdelt,kx,ky)
      implicit none
!
!     subroutine euler does an explicit Euler time step
!
      integer :: kx                     ! x dimension
      integer :: ky                     ! y dimension
      real    :: pdelt                  ! time step [s]
      real    :: pfm(0:kx+1,0:ky+1)     ! input f(t)
      real    :: pdfdt(0:kx+1,0:ky+1)   ! input tendency
      real    :: pf(0:kx+1,0:ky+1)      ! output f(t+1)
!
      return
      end
```

To complete euler, we only need one additional line of code.

leapfrog

Compared to the explicit Euler implementing the filtered leapfrog needs a bit more work (though we already did it for the advection equation last semester). We need the dimension of the fields, the time step (or two times the timestep), the tendency, actual field (time=t), and the filtered field for t-1. Output is the filed at the next time step (t+1) and the filtered field for time t, which are use as t and filtered t-1, espectively, during the next timestep. Here, it is assumed that the old fields are replaced by the new ones:

```
      subroutine leapfrog(pf,pfm,pdfdt,pdelt2,kx,ky)
      implicit none
!
!     subroutine leapfrog does a leapfrog time step
!     with Robert Asselin filter
!
      integer :: kx                  ! x dimension
      integer :: ky                  ! y dimension
      real :: pdelt2                 ! 2.* timestep
      real :: pf(0:kx+1,0:ky+1)      ! input/output f(t)
      real :: pfm(0:kx+1,0:ky+1)     ! input/output f(t-1) (filtered)
      real :: pdfdt(0:kx+1,0:ky+1) ! input tendency
!
      return
      end
```

leapfrog may be cmpleted according to the equations above (or from the lecture last semester). Remember the replacement (t) -> (t-1) und (t+1) -> (t) at the end of the routine.

5. Problems

5.1. *Coding the model:* Build the non-divergent barotropic model using the above and, perhaps, given examples. Test the model by simulating the Rossby-Haurwitz wave.

The code 'baro.f90' ('homepage') already includes (almost) all settings necessary: Module *baromod* contains the dimensions *NX* and *NY* of the channel (64 and 32, resp.), the longitudinal and meridional extent (*xchannel, ychannel;* 360° and 40°), the central latitude (*rlat;* 50°N), and the timestep (*delt;* 1200s). Subroutine *initial* initializes the streamfunction with wave number 1 in x-direction (0.5 in y-direction). You only need to set the numebr of timesteps (*nrun*). At the beginning *nrun* may set to 1 to test the initial condition and the first prediction step. If the results look ok you may run a final test with *nrun*=100.

5.2. *Barotropic instability:* It can be shown (e.g. Holton) that under certain conditions a zonally symmetric flow can get unstable, i.e. small initial disturbances extract kinetic energy from the zonal flow and grow with time. On the other hand situations exist where disturbances give kinetic energy to the zonal flow (barotropic stability, which is dominant on the large scale).

A condition for a zonal jet to be barotropically unstable is a zero of the absolute (in general, the potential) vorticity within the domain, i.e.

$$\beta - \frac{\partial^2 [U]}{\partial y^2} = 0$$

somewhere (at the critical latitude y_c). Here $[U]$ denotes the zonal averaged zonal wind (jet).

Barotropic instability can be analytically or numerically studied by investigating the behavior of perturbations in different zonal mean jets. It is relatively easy to show that a simple cosine-profile ($[U]=\cos(\pi y/B)$; with B=width of the channel) is always stable. However, the profile

$$[U] = \frac{U_{\max}}{2}(1-\cos(\frac{2\pi}{B}y))$$

may be unstable (U_{max} is the wind speed in the center of the jet). Depending on the strength of the jet and the wavenumber of the disturbance, we can find stable, neutral or unstable situations (i.e. the disturbance will decrease, stay constant of increase). In general: starting from a certain (minimum) velocity of the jet waves with wavenlengths smaller than a critical value (L_c) are stable (give energy to the zonal mean flow), waves longer than L_c but shorter that a second critical value (L_0) are unstable (gain energy from the zonal flow), and very long

waves (L > L_0) are neutral (their energy stays constant). For the three classes the following applies:
1. Stable: The phase velocity of the wave is larger than the wind speed of the jet at the critical latitute
2. Unstable: The phase velocity of the wave is positive but smaller than the windspeed of the jet at the critical latitude.
3. Neutral: The phase velocity of the wave is negative.

The velocity of the jet at the critical latitude, and the phase velocity which separates 1 and 2 (*Cn*), can be computed from the instability condition for the jet:

$$[U](y_c) = Cn = \frac{U_{\max}}{2} - \frac{\beta B^2}{4\pi^2}$$

Study/verify by using your barotropic model the stability properties of the above jet-profile. Vary the wavelength of the perturbation by keeping U_{max} constant. Analyse the temporal evolution of the kinetic energy of the disturbance and of the zonal mean flow. Which are the values for the critical wavelengths? How des the meridional structure looks like for the stable and the unstable case?

You may use the following setup:

- Channel as in the original model (5.1).
- U_{max}=100m/s
- Meridional wavenumber of the disturbance = 0.5 (as in 5.1)
- 1 day of integration with 20 minutes timestep

Notes: a) To include the jet-profile you need to change the boundary conditions (subroutine boundary). So far, the streamfunction is set to 0 at the meridional boundaries (which does not allow for a jet. The new condition (in y) may be the zonal average of the streamfunction at point y=1 (for 0) and y=NY (for NY+1).
b) In subroutine initial the jet needs to be included in the initial conditions. For streamfunction, the profile is

$$[\Psi] = -\frac{U_{\max}}{2}(y - \frac{B}{2\pi}\sin(\frac{2\pi}{B}y))$$

c) In subroutine initial you can choose the x-wave number of the perturbation (parameter zk).

d) It can be useful to include the energy diagnostics in the model (e.g. in subroutine baroout), partitioned in zonal mean and perturbation.

e) Due to the numerics (inexactness of the method and/or the numerical precision of the computer) you may not expect that neutral waves exactly appear

5.3. *Weather prediction:* The non-divergent barotropic model was the first working weather prediction model. Due to all approximations entering the model it can, of course, not compete with state-of-the-art prediction models. However, you may try a prediction using your model. For this purpose a file input.dat is provided at the 'homepage'. It contains initial and verification data for a 6-hourly prediction for geopotential on the model grid (64x34; i.e. for the complete channel incl. boundary) for the time 14.02.1962 00:00 to 24.02.1962 00:00 (Hamburg storm surge 15.-16.02.1962). In addition, file input.ctl provides a grads-control file to plot the data.

Notes: a) To read the data, you may use subroutine initial. The data are unformated and may be read using

```
open(11,file='input.dat',form='unformatted')
read(11) f(1:NX,0:NY+1)
close(11)
```

Using multiple reads you may skip time steps to choose different initial times.

b) As it is geopotential you need to divide by f0 to get streamfunction.

c) Important: To keep the model numerically stable you may keep the meridional boundary condition (points 0 and NY+1) constant using the initial values. This can be done by removing the setting of the boundary conditions for the respective points in subroutine boundary.