Documentation for the ABC model

Ross Bannister, University of Reading, UK

March 10, 2017

The "ABC model" is a non-hydrostatic toy model – designed by Ruth Petrie, Ross Bannister, and Mike Cullen – for use in convective-scale data assimilation investigations. Source code for model and diagnostics are freely available from git-hub. This is a very brief guide to the model and how to use it.

1 The model equations

The equations solved by the model are the following

$$\frac{\partial u}{\partial t} + B\mathbf{u}\cdot\nabla u + C\frac{\partial \tilde{\rho}'}{\partial x} - fv = 0, \tag{1a}$$

$$\frac{\partial v}{\partial t} + B\mathbf{u}\cdot\nabla v + fu = 0, \tag{1b}$$

$$\frac{\partial w}{\partial t} + B\mathbf{u}\cdot\nabla w + C\frac{\partial \tilde{\rho}'}{\partial z} - b' = 0, \tag{1c}$$

$$\frac{\partial \tilde{\rho}'}{\partial t} + B\nabla\cdot(\tilde{\rho}\mathbf{u}) = 0, \tag{1d}$$

$$\frac{\partial b'}{\partial t} + B\mathbf{u}\cdot\nabla b' + A^2 w = 0. \tag{1e}$$

The prognostic variables are as follows: u is the zonal wind, v is the meridional wind, w is the vertical wind, $\mathbf{u} = (u, v, w)$ is the wind vector, $\tilde{\rho}$ is a density-like variable (where $\tilde{\rho}'$ is the perturbation, $\tilde{\rho} = \tilde{\rho}_0 + \tilde{\rho}'$, where in this model, $\tilde{\rho}_0 = 1$), and b' is a buoyancy-like variable (for meteorologists, b' is related to potential temperature, θ', by $b' = g\theta'/\theta_{\rm R}$, where g is the acceleration due to gravity and $\theta_{\rm R}$ is the reference potential temperature of 273K). The dimension variables are as follows: x is longitudinal distance, z is vertical distance, and t is time. Constant parameters to be chosen by the user are as follows: A (units s^{-1}) is the static stability (equivalent to the pure gravity wave frequency), B (dimensionless) multiplies the advection and divergence terms, and C (units $\mathrm{m}^2\mathrm{s}^{-2}$) relates density perturbations to pressure perturbations, $p' = C\rho_0\tilde{\rho}'$, where ρ_0 is a reference density. The value of $\sqrt{BC}$ is the pure acoustic wave speed). These parameters give the model its "ABC" name. The remaining constant is f, which is the Coriolis parameter.

There is also a tracer transport equation, which advects a tracer, q, with the wind vector $\mathbf{u}$, and not by the modified winds, $B\mathbf{u}$:

$$\frac{\partial q}{\partial t} + \mathbf{u}\cdot\nabla q = 0. \tag{2}$$

The model is run in a 2D slice (longitude/height) geometry. All variables are considered constant in the meridional direction. The model grid is an Arakawa-C grid in the horizontal

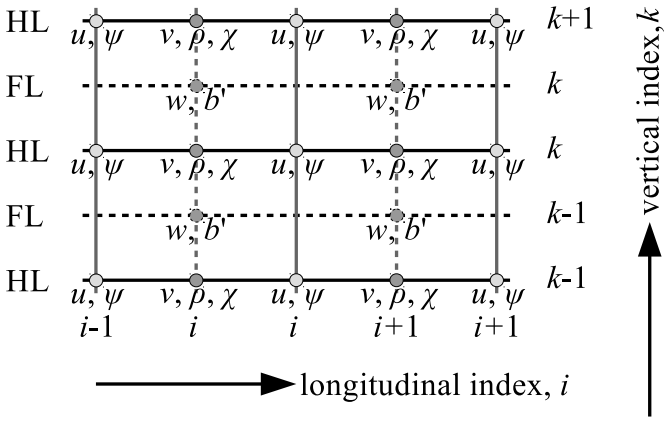

Figure 1: The arrangement of variables on the toy model's grid: an Arakawa-C grid in the horizontal and a Charney-Phillips grid in the vertical. Note the abbreviations: FL=Full Level and HL=Half Level.

and a Charney-Phillips grid in the vertical (Fig. 1). The horizontal resolution of the model is 1.5km, there are 360 grid-points in the horizontal, and 60 vertical levels.

The scientific rational for this model, together with more details, are in a paper[1], which is currently under peer review with the journal Geoscientific Model Development.

2 The files in the git-hub repository

The model is written in fortran-90, and is contained in the following files:

```
Boundaries.f90
BoundaryMod.f90
DefConsTypes.f90
Diagnostics.f90
Forcings.f90
Functions.f90
Initialise.f90
Linear_Analysis.f90
Main.f90
Model.f90
ReadWrite_data.f90
UM_data_proc.f90
```

Also available is the make file for compilation with a Linux-based operating system:

```
makefile
```

The following files are also included:

```
UserOptions.nl
Processed_Initial_Data.nc
```

[1]The "ABC model": a non-hydrostatic toy model for use in convective-scale data assimilation investigations by Petrie, Bannister and Cullen.

(an example namelist file and sample initial conditions for the model). Diagnostics output from the model can be viewed by running python codes as follows:

```
BalanceScale_1.py
PlotEnergy.py
PlotWaveSpeeds.py
PlotMany.py
```

3 Compiling the code

The code is compiled in the standard way in Linux by typing *make* in the directory containing the source code and the make file. Users should check that the make file is correctly configured to his or her system (some example configurations are given in the make file). Users should also check that the include files in the Fortran code are correct for his or her system. The code requires a Fortran-90/95 compiler, and the netcdf and nag libraries.[2]

4 Running the software

Control of the mode of operation, and the values of parameters is made with the namelist file *UserOptions.nl*. There are more modes of operation that the ones described below, but it is assumed here that users will be most interested in the run modes documented.

4.1 Linear analysis

The model's linear modes (frequencies, and horizontal and vertical speeds of gravity and acoustic modes) are investigated by using the following example namelist.

```
! Define namelist
&UserOptions
! Linear analysis
   make_ics_from_um          = .FALSE.
   run_toy_model             = .FALSE.
   do_linear_analysis        = .TRUE.
   linear_analysis_odir      = '.'
   wavespeed_experiment      = 'Waves'
   A                         = 0.02
   B                         = 0.01
   C                         = 1.0E4
   f                         = 1.0E-4
/
```

The *run_toy_model = .FALSE.* and *do_linear_analysis = .TRUE.* ensure that the model itself is not run, but the linear analysis routine is run. Files are output to directory *linear_analysis_odir*, and output filenames will contain a part with the string *wavespeed_experiment*. The values of A, B, C, and f are set here. The output of this mode can be analysed using the program *PlotWaveSpeeds.py*, which generates various plots (in encapsulated postscript format).

[2] The nag library is used to compute the eigenstates of the linearised model when looking at the normal modes of the model (this is the "linear analysis" run mode in Sect. 4.1). Users not wishing to do these calculations and who do not have access to the nag library should 'comment-out' the call to *nag_sym_eig_all* in *Linear_Analysis.f90*, and remove references to nag software in the compile and link options in the make file.

4.2 Running the model

The model is run from a file of initial conditions by setting the following example namelist.

```
! Define namelist
&UserOptions
! Reading and processing UM data
  make_ics_from_um          = .FALSE.
  datadirUM                 = '.'
  init_um_file              = ''
  model_ics_data_out_file   = 'Processed_Initial_Data.nc'
  latitude                  = 144
  Regular_vert_grid         = .TRUE.
  gravity_wave_switch       = .FALSE.
!
  A                         = 0.02
  B                         = 0.01
  C                         = 1.0E4
  f                         = 1.0E-4
  Tracer_level              = 20
!
! Run the forward model
  run_toy_model             = .TRUE.
  datadirMODEL              = '.'
  model_ics_data_read_file  = 'Processed_Initial_Data.nc'
  model_output_file         = 'Fields.nc'
  diagnostics_file          = 'Diagnostics.dat'
  dt                        = 1.0
  runlength                 = 10800.0
  ndumps                    = 6
  convection_switch         = .FALSE.
  pressure_pert             = .FALSE.
  press_source_i            = 180
  press_source_k            = 30
  x_scale                   = 60
  z_scale                   = 3
  press_amp                 = 0.01
  Adv_tracer                = .TRUE.
  Lengthscale_diagnostics   = .TRUE.
!
  do_linear_analysis        = .FALSE.
  linear_analysis_odir      = '.'
  wavespeed_experiment      = '.'
/
```

Not all of the variables shown above are needed to run the model. The important ones are: *run_toy_model*, which should be switched on, *datadirMODEL* is the directory containing the initial conditions, *model_ics_data_read_file* is the name of the initial condition file (a netcdf file), *model_output_file* is the filename of the output fields (a netcdf file), *diagnostics_file* contains some diagnostics data (text file), *dt* is the time step (seconds), *runlength* is the model integration length (in seconds), *ndumps* is the number of times the model will dump its state over the integration, *Adv_tracer* switches on/off tracer advection (the q field above), and *Lengthscale_diagnostics* switches on/off calculation and output of numerically derived lengthscales of each variable at the end of the integration. The values of the parameters A, B, C, and f are

also set in this namelist.

Note that the initial conditions file provided, *Processed_Initial_Data.nc*, has been prepared to be a nearly balanced state according to parameters $C = 10^4 \mathrm{m}^2\mathrm{s}^{-2}$ and $f = 10^{-4}\mathrm{s}^{-1}$ (other parameters do not play a role in the generation of this file). Running the model with this initial state, but with C and f parameters different to these may produce highly unbalanced results. The authors advise that C and f are not changed without first generating a new consistent set of initial conditions (this is done from a specific Met Office Unified Model (UM) dump, and running with *make_ics_from_um* switched on to generate a new set of initial conditions[3]).[4]

The output of this code can be analysed with the programs *BalanceScale_1.py*, *PlotEnergy.py*, and *PlotMany.py*. *BalanceScale_1.py* plots the geostrophic and hydrostatic imbalance as a function of time and horizontal scale, *PlotEnergy.py* plots the total energy in the model as a function of time (to check conservation), and *PlotMany.py* plots each field in the main output file.

[3]This step is not yet documented here as it is assumed that the user will not have access to a suitable UM dump. Please contact the authors if you wish to use this option.

[4]It is possible to use an approximate equivalence of parameters to *effectively* run with a different value of C to that used above (but actually keep the same value of C as above, and hence use the set of initial conditions provided). Two model runs are approximately equivalent if the product BC is the same (some model fields will be scaled accordingly, but the underlying physics will be approximately the same for two model runs with a common value of BC). Thus instead of modifying C to a required value C_{req} (and hence the need to generate a consistent set of initial conditions), you may wish instead to run the model with the original C as above (call C_{orig}), but instead modify B accordingly to a new value. In the above $B_{\mathrm{orig}} = 0.01$, and $C_{\mathrm{orig}} = 10^4 \mathrm{m}^2\mathrm{s}^{-2}$. Let B_{req} and C_{req} be the required new values. The parameters that you could actually run with, B_{new} and C_{new}, which are approximately equivalent to B_{req} and C_{req}, could then be set to $B_{\mathrm{new}} = B_{\mathrm{req}} C_{\mathrm{req}}/C_{\mathrm{orig}}$, and $C_{\mathrm{new}} = C_{\mathrm{orig}}$.