Gung-Ho Dynamics:

Mimetic, semi-implicit, forward-in-time integration of the shallow water equations on hexagonal and cubed sphere grids

Brief notes for users

John Thuburn
16 March, 2012

1 Introduction

A shallow water code based on a mimetic, semi-implicit, forward-in-time integration scheme has been developed for Gung-Ho. The code is designed for a range of unstructured grids, and has been tested for two families of grids: hexagonal-icosahedral Voronoi grids and modified equiangular cubed sphere grids. The scheme, the grids, and sample results are presented in a separate report. These brief notes are intended to allow others to get started using the code, for example to test other families of grids or to explore the code's potential for parallel scalability.

2 Generating a hexagonal Voronoi grid

Files: `gengrid_hex.f, dodecahedron.xref`.

Edit the file `gengrid_hex.f` to change the parameter `NGRIDS` to the desired value. There are eight occurrences of `NGRIDS`, so you will need to do a global search and replace. The code will generate a grid with $10 \times 4^{\mathrm{NGRIDS}-1} + 2$ faces.

If you want to change the weights given to the Heikes-Randall cost function and the centroidal cost function then modify the values of `WEIGHT1` and `WEIGHT2` at lines 459 and 460. If you want to take a different number of nudging iterations then change the range of the loop at line 464.

Compile and run `gengrid_hex.f`. The file `dodecahedron.xref` is used as input data. The code may take several minutes to run at high resolution because of the nudging iterations. In reply to the question 'Create a GRIDMAP file (0 or 1) ?' enter 1.

The code produces two formatted files `primalgrid.dat` and `dualgrid.dat` suitable for plotting. The main output is an unformatted file called `gridmap_hex_xxxxxxxxxx.dat`, where `xxxxxxxxxx` is a 10-digit integer giving the number of faces on the grid. It contains grid cell centre and vertex coordinates, edge lengths, cell areas, and cross-reference tables defining grid topology, on a hierarchy of grids suitable for a multigrid elliptic solver.

3 Building mimetic operators for a hexagonal Voronoi grid

Files: `buildop_hex.f90, gridmap_hex_xxxxxxxxxx.dat`.

In addition to this basic grid information, the integration scheme also requires tables of numbers defining the stencils and coefficients for the various mimetic operators, plus restriction/prologation operators needed for multigrid. To generate these, compile and run `buildop_hex.f90`. In reply to the request for input: 'Resolution of input file: number of faces', enter the number of faces of the desired grid.

The code reads input from the file `gridmap_hex_xxxxxxxxxx.dat` and produces a new file called `gridopermap_hex_xxxxxxxxxx.dat`. The new file contains the original grid information from `gridmap_hex_xxxxxxxxxx.dat` plus the the newly built operator information. Only the new file is needed for a model integration, so once it is built the original gridmap file can be deleted.

4 Generating a cubed sphere grid

Files: `gengrid_cube.f90`.

Edit the file `gengrid_cube.f90` to modify the parameters `ngrids` and `n0` in the first few lines of the code. The code will generate a grid with $6 \times n0^2 \times 4^{ngrids-1}$ faces.

If you wish to change the number of smoothing iterations then change the parameter `nsmooth`, but note that increasing this number makes the grid less uniform as well as smoother.

The code produces two formatted files `primalgrid.dat` and `dualgrid.dat` suitable for plotting. The main output is an unformatted file called `gridmap_cube_xxxxxxxxxx.dat`, where again `xxxxxxxxxx` is a 10-digit integer giving the number of faces on the grid.

5 Building mimetic operators for a cubed sphere grid

Files: `buildop_cube.f90, gridmap_cube_xxxxxxxxxx.dat`.

To generate the additional operators needed, compile and run `buildop_cube.f90`. In reply to the request for input: 'Resolution of input file: number of faces', enter the number of faces of the desired grid.

The code reads input from the file `gridmap_cube_xxxxxxxxxx.dat` and produces a new file called `gridopermap_cube_xxxxxxxxxx.dat`. The new file contains the original grid information from `gridmap_cube_xxxxxxxxxx.dat` plus the the newly built operator information. Only the new file is needed for a model integration, so once it is built the original gridmap file can be deleted.

6 Building your own grid

Not for the faint hearted! `MODULE grid` in `mimeticswe.f90` or in either of the buildop codes documents the information that needs to be generated. A lot of the code needed to build the mimetic operators may be similar to that in `buildop_hex.f90` and `buildop_cube.f90`, so feel free to recycle any of the code provided.

7 Running the model

Files: `mimeticswe.f90, swenml.in, gridopermap_hex_xxxxxxxxxx.dat` or `gridopermap_cube_xxxxxxxxxx.dat`.

Most model options are set via the namelist file `swenml.in`.

`ygridfile`	Name of the gridopermap file to use.
`degree`	Degree of polynomial fit to use for advection. Should work for 0, 1, 2, 3, or 4 but only tested for 2 and 4.
`dt`	Time step in seconds.
`niter`	Number of iterations of nonlinear solver. Only tested for 4 but might work with fewer.
`alpha_v`	
`alpha_pg`	Off-centring parameters for velocity and pressure gradient terms. Testing to date has used 0.5d0.
`nstop`	Length of integration in steps.
`noutput`	Number of steps between output dumps.

Other namelist switches are not used. (Caveat: some old versions of the gfortran compiler, like that on my laptop, have a compiler bug that causes failure when reading the namelist. To bypass this, remove all switches from the namelist file and set their default values in the code itself.)

Constants such as Earth's radius, acceleration due to gravity, and Earth's rotation rate can be set by changing the values in `MODULE constants`.

The orography field can be modified by editing `SUBROUTINE setorog`.

The initial condition can be set by editing `SUBROUTINE setini`. Several standard test case initial conditions are already coded and can be seleted by setting the variable `ic`.

The model can output fields on primal grid faces and dual grid faces (vertices). Edit `SUBROUTINE output` to choose the desired fields. You may need to add some code to compute the desired fields from the prognostic variables `phi2` (primal cell area integrals of ϕ) and `v1` (dual edge integrals of tangential velocity, i.e. circulations) if this is not already done.

Once all desired options have been selected, compile and run `mimeticswe.f90`. The code reads switches from the namelist file `swenml.in` and grid and operator information from the file defined by switch `ygridfile`.

8 Plotting output

Files: `join.m`, `jtaxes.m`, `jtcontour.m`, `jtplotgrid.m`, `jtrotplot.m`, `plotdiag.m` `dump1_yyyyyyyy.m`, `dump2_yyyyyyyy.m`, `fort.43`.

The model run produces files with names of the form `dump1_yyyyyyyy.m` and `dump2_yyyyyyyy.m` where `yyyyyyyy` is an eight digit integer indicating the time step of the output. `dump1...` contains primal grid fields and `dump2...` contains dual grid fields.

Some Matlab routines are supplied to plot the output. First select a plot type by setting the Matlab variable `ptype = 'latlong'` or `ptype = 'sphere '` (note the space). Then simply type the name of the output file containing the fields to be plotted (without the `.m`). This executes the Matlab commands to read in the data and call the Matlab plotting routines. Note this is slow at high resolution; be patient!

The contour values and line thicknesses may be changed by editing `jtcontour.m`. Also, for `ptype = 'sphere '`, the latitude and longitude of the viewing point can be changed by editing `jtcontour.m`.

When `ptype = 'sphere '` is chosen the plotting routines automatically use the data in `primalgrid_hex.dat`, `dualgrid_hex.dat`, `primalgrid_cube.dat`, and `dualgrid_cube.dat` to plot a coarse-resolution image of the grid in the background.

The model also writes time series of conservation-related diagnostics to file `fort.43`. These can be read in and plotted with the Matlab script `plotdiag.m`. You will need to edit `plotdiag.m` to set the correct number of time steps per day.

9 Other options

The model code contains subroutines for testing various aspects of the code and operators, including the mimetic operators, the multigrid solver, advection, and computing normal modes. These can be invoked by commenting out the call to `integrate` in the main program and uncommenting the call to the relevant routine. These may be useful for testing new grids. The Matlab script `findevals2.m` may be useful for completing the normal mode calculation and plotting the resulting eigenvalues (best done at coarse resolution!).