Two-Dimensional Hadley Model

Kerry Emanuel

Updated March, 2021

Setting up and running the model

Unzip all the files into a single subdirectory of your system. The *.h* files are needed to compile but there should be no reason to alter them. Compile the FORTRAN program Hadley.f using any standard FORTRAN compiler, such as g95 or gfortran. You can compile the program using a command such as

gfortran -o Hadley Hadley.f

Give the program a trial run using the existing *Hadley.in*, which runs the model for 5 days. An easy way to run the program is to type "runHadley *filename*", where *filename* is a new folder name you select, which will contain the output of the program. This should work on Windows and Linux operating systems. This command runs the program, deposits the output in a directory called *output*, and then copies that directory to the new folder named *output_filename*. Verify that there are new files added to the *output* directory.

Displaying the output using MATLAB

Once the model has run, start up MATLAB and run the script *HadleyPLot.m.* You should be presented with a menu of display options which are pretty much self-explanatory. There is also an option to print the last displayed plot as a .png file.

Controlling the simulation using the Hadley.in file

The *hadley.in* file in an ordinary text file that displays three columns. The first contains the value of the parameter that is actually read by the FORTRAN program. The second is the units of the parameter (if there are any), and the third is a brief description of the parameter. Herewith a more detailed description of the parameters:

Program control

1: This tells the program whether to initialize using the file *sounding.in* or to restart the integration from the end of the previous run, using *output/sounding.out*. Note that you cannot choose the latter until you have run the model at least once.

2: The length of the integration

3: The numerical time step. 5 minutes is standard, but one can get away with 10 or even 20 minutes if radiative heating is specified rather than calculated.

Domain specification

1-2: Specifies the latitudes and longitudes of the *center* of the first and last column.

3: The latitude north of which has land as a lower boundary; all columns to the right are considered land. Setting this to any number > the northern boundary will give all ocean, while setting to and number < the southern boundary will give all land.

Radiation parameters

1: Specify *y* to use the program's radiative transfer scheme. If *n*, then the model uses the radiative cooling rates specified in *sounding.in* or *output/sounding.out*, for all time. The code will run somewhat faster without the calculation of radiation.

2: Frequency of calls to the radiation scheme. To call the radiation scheme every time step would make the code very slow. For non-interactive clouds and no diurnal cycle, this can be as large as 12 hours and still give a stable integration.

3: If *y*, the temperature of the ocean (or land) surface is calculated; otherwise, it is fixed at the initial value read in from the input sounding.

4-5: Specifies whether the water vapor and cloud used by the radiative code. If either is set to no, the model looks for a binary file called *output/variables* created by a previous run *if and only if the last parameter in params.in is set to y*.

6: Solar constant

7: Carbon dioxide content

8-10: Starting month, day and hour. The last is GMT; local time is determined by the longitude of the column. Note that the time is fixed at these values unless the next parameter is set to *y*.

11: If *y*, then the month, day, and/or hour progresses through the simulation.

12: If *y*, the date progresses through the simulation. If this is *y* and the preceding parameter is *n*, then a diurnal cycle is repeated without the date changing.

13: If *y*, then the solar zenith angle is averaged over one full diurnal cycle.

14: If *y*, the solar zenith angle is averaged over one calendar year.

15: If *y*, an algorithm calculates the surface albedo of water as a function of solar zenith angle.

16: Surface albedo of ocean, if previous parameter is *n*.

17: Surface albedo of land

Surface fluxes

1: If *y*, the model-calculated wind at the lowest level is used in the calculation of surface fluxes. Setting this to *n* turns off the WISHE effect.

2: Assumed background wind used in the calculation of surface fluxes

3: A w*-type term for the surface fluxes.

Soil parameters

1: If *y*, then the model calculates soil moisture from precipitation, evaporation, and runoff.

2: Fraction of potential evaporation used over land if previous parameter is *n*.

3: A storage time scale for soil moisture and heat

4: A runoff time scale for soil moisture

5: A parameter governing the sensitivity of surface evaporation over land to soil moisture.

Ocean parameters

1: Assumed depth of the ocean mixed layer. This determines the thermal inertia of the ocean.

2: (A parameter not currently used by the model)

Miscellaneous parameters

1: A pressure depth scale over which turbulent momentum fluxes converge in the boundary layer.

2: A time scale governing the magnitude of the Fickian diffusion coefficient used by the model.

3: A damping time scale for the top sponge layer

4: Setting this to > 0 freezes the model fields at their initial values for pressure levels higher than this value.

Output

1: Time resolution of the output fields. Note that making this too small for a given integration time may overflow time-dependent arrays in the program. Specifically, the length of integration divided by this number must be < 5000. Note that the radiation in the output will look jumpy if this is set to a value smaller than the interval between calls to the radiation scheme.

2: Output for making cross-sections is averaged over this time scale at the end of the integration. If this number is greater than or equal to the integration time, then the fields are averaged over the whole length of the integration.

3: For periodic boundary conditions, the domain can move left or right at this velocity for the purposes of making composite structure of uniformly moving disturbances.

4: If *y*, then water vapor and cloud fields averaged over the whole domain and for the whole length of the integration are used if either or both the water vapor or clouds are set to noninteractive in a subsequent run.

General description of the model

This two-dimensional model integrates the primitive (hydrostatic) equations phrased in term of vorticity (in the direction orthogonal to the domain) and streamfunction. The domain is oriented north-south. Convection is parameterized following Emanuel and Živković-Rothman (1999), and this is coupled to the fractional cloud scheme of Bony and Emanuel (2001), while radiative transfer uses the scheme of Fouquart and Bonnel (1980) for shortwave radiation, and Morcrette (1991) for longwave radiation. The boundary layer is treated very crudely, with dry adiabatic adjustment applied to all quantities, including momentum, and a separate treatment of momentum damping is applied near the surface. A sponge layer at the top absorbs some of the upward propagating internal waves. A land surface may be specified in part of the domain. Very crude parameterizations of soil moisture and runoff are included. Over the ocean, conventional bulk formulae are used to calculate surface fluxes. Model variables are written to ordinary ascii files a MATLAB script is available for displaying the output.

Model Structure

The model consist of NLO columns. This number is specified in a PARAMETER statement at the beginning of the program; changing this is ordinarily the only reason one has to modify the code itself (*not recommended*). Column 1 and Column NLO are boundary columns. The model has an arbitrary number of levels in the vertical, determined by the initialization file *sounding.in* (see below).

The thermodynamic variables are defined in a 2-D grid determined by the input sounding (in the vertical) and over a horizontal grid of NLO columns. The vorticity and streamfunction are defined on a grid that is staggered a half vertical layer downward and a half column width rightward. The component of velocity orthogonal to the 2-D section is defined on the same grid as the thermodynamic variables.

The minimum allowed value of NLO is 3, in which case the model becomes a single-column model (with two boundary columns that are not, in this case, actually used, though they are retained.) Note, however, that the supplied MATLAB scripts are not ideally suited to displaying the output of the single column results.

Initialization and model control

The model reads initial fields from an ascii file called *sounding.in*, unless the model is restarted from the previous integration, in which case it reads the file *output/sounding.out*. These fields consist of a sequence of column data. There should be NLO vertical profiles, corresponding to NLO columns in the model. Note that the number of levels and vertical structure of the *sounding.in* file set the vertical structure and resolution of the model.

References

Bony, S. and K. A. Emanuel 2001: A parameterization of the cloudiness associated with cumulus convection: evaluation using TOGA COARE data. *J. Atmos. Sci.*, **58**, 3158-3183

Emanuel, K. A. and M. Živkovic-Rothman, 1999: Development and evaluation of a convection scheme for use in climate models. *J. Atmos. Sci.*, **56**, 1766-1782

Fouquart, Y. and B. Bonnel, 1980: Computation of solar heating of the Earth's atmosphere: A new parameterization. *Beitr. Phys. Atmos.*, **53**, 35-62

Morcrette, J.-J., 1991: Radiation and cloud radiative properties in the European Centre for Medium-Range Weather Forecasts forecasting system. *J. Geophys. Res.*, **96**, 9121-9132