RRTMGP: topography support package

Igor Polonsky
Atmospheric and Environmental Research, Inc.

1. Introduction

The mountain covers ~ 25 % of land suggesting that the resolved (grid-scale) surface features have to be included into include into radiative transfer through parameterization. In this package such a parameterization has been implemented as an extension to a two-stream radiative transfer where the following surface fluxes are available: up and down diffuse SW and LW as well as direct (unscattered) SW flux.

Here the list of terrain features which effects are implemented in the code:

Feature	Relevant to	Required data
surface slope angle variability	SW	Slope statistics per computational grid
Terrain shadowing	SW	Horizon angle pre-calculated on an azimuth grid for all relevant DEM grid points
Sky-view factor	LW	Integral over horizon angles (see above)

A brief discussion of the effect parameterization can be found in Section 6 of the document and in more details in the papers enlisted in Section 8.

The code was implemented using Python and was tested using digital elevation map (DEM) of 6' resolution.

2. Package installation

The package can be acquired from git repository
https://github.com/inpolonsky/rrtmgp_topography.git
and requires python 3 to execute. The package includes the following files:

README.txt - an instruction how to acquire ETOPO1_Ice_c_gdal.grd dataset;
LICENSE – software license;
geodesy.py - auxiliary geophysical functions;
topography.py - main module that performs parameter computation
prepareTopographyExample.py - sample code that creates parameter dataset for RRTMGP test. It shows the use of the package and relies on DEM dataset
ETOPO1_Ice_c_gdal.grd
to be in the working directory

ETOPO1_Ice_c_gdal.grd represents ETOPO1 1 Arc-Minute Global Relief Model (http://www.ngdc.noaa.gov/mgg/global/global.html) and can be acquired using link

https://www.ngdc.noaa.gov/mgg/global/relief/ETOPO1/data/ice_surface/cell_registered/netcdf/ETOPO1_Ice_c_gdal.grd.gz

3. Run the RRTMGP test case

To prepare topography parameters a user may start with topography parameters dataset preparation by running

python PrepareTopographyExample.py -grid -72 -22 -63 –12

that creates files:

rrtmgp-lw-topography-sample.nc
rrtmgp-sw-topography-sample.nc

which RRTMGP test *flux_compute_topography* can read. To test how these parameters can influence fluxes and heating rates, the user has to acquire RRTMGP

https://github.com/RobertPincus/rte-rrtmgp

branch *topography*
as well as RRTMGP test suite

https://github.com/RobertPincus/rte-rrtmgp-test-val.git

also branch *topography*.

The user should follow the RRTMGP as well as RRTMGP test suite instructions how to install, compile and run the code on the system.

To compile and run *flux_compute_topography* test the user has to update Makefile in test directory to ensure successful compiling, then to compile the test case

make -f Makefile test_flux_compute_top

The included python script simplifies running the test case and may be run as

python run_tests.py --test flux_compute_topography.ini

The git repository already contains files

test/flux_compute_topography/ref/rrtmgp-lw-topography-sample.nc
test/flux_compute_topography/ref/rrtmgp-sw-topography-sample.nc

that include the topography related parameters while files

test/flux_compute_topography/ref/rrtmgp-lw-inputs-outputs-clear-top.nc
test/flux_compute_topography/ref/rrtmgp-sw-inputs-outputs-clear-top.nc

contain other variables required to run RT computation in SW or LW spectral regions as well as storing the reference values for fluxes and heating rates.

The use may either regenerate rrtmgp-lw[sw]-topography-sample.nc or use ones that are cloned from git repository.

4. How to update the user code

To include topography effect into user code the following actions should be performed.

For SW region.

Firstly, the user code must allow to enter the surface albedo for the direct solar and diffuse radiation components independently since the topography affects only the direct solar radiance reflection from surface. Secondly, the user must read the total correction factor and apply it to modify the direct surface albedo.

For LW region.

A few lines of the code that modify the downward, upward and net fluxes at surface level must be added that use the view factor parameters.

5. prepareTopographyExample.py modification

PrepareTopographyExample.py was developed as a sample code that creates parameter dataset for RRTMGP test. However, the user is encouraged to modify it to meet their requirements as they see fit. Its main functionality consists of two major routines:

1. ***readDEM****(ncFile, numLat, numLon)* that reads DEM dataset and reshapes it into a matrix with longitude dimension corresponding to rows and columns to latitude dimension, respectfully. Generally, this function implementation depends on the user DEM dataset structure. *Important note, the user DEM may include the ocean floor depth as negative numbers. It's the user responsibility to ensure that DEM over ocean provides valid elevation map.*

2. ***topCalc****(inGrid, lat1, lat2, lon1, lon2, delX=1, delY=1, saveFig=False, toPlot=False, nBlocks=100, add_aux=False, nZen=19, nHz=16)* performs all computations and generates output as a single netCDF per computational grid cell. Later, these files can be used to generate a dataset that can be organized to accommodate user specific requirements.

3. The auxiliary routines (**prepareSW** and **prepareLW)** generate datasets that are specific to the RRTMGP test *flux_compute_topography*.

topCalc parameters:

inGrid is the matrix that contains the DEM dataset for the whole Earth.

lat1, lat2 are the southern and northern latitudes of the computational grid box with resolution of delLat (default is 1 degree);

lon1, lon2 are the western and eastern longitudes of the computational grid box with resolution of delLon (default is 1 degree);

toPlot is the flag controlling whether the plot showing the computational progress and visualizing some parameters will be generated (default is not to generate);

saveFig is the flag indicating whether the produced figures will be saved (default is not to save);

nBlocks is the amount of DEM cells on which the computational area is extended in all directions (the default is 100);

add_aux is the flag controlling whether to put extra information into the output netCDF file. The implemented code can either create a simplified output that provides the just necessary parameters that quantify the topography effect or add extra data that allows the user to look into details of the

parameter generations (slope distributions, horizon angle distributions as a function of solar azimuth, etc)

nZen is the number of solar zenith angles the code uses to create tables for (the default value is 19);

nHz is the number of solar azimuth angles the code uses to create tables for (the default value is 16);

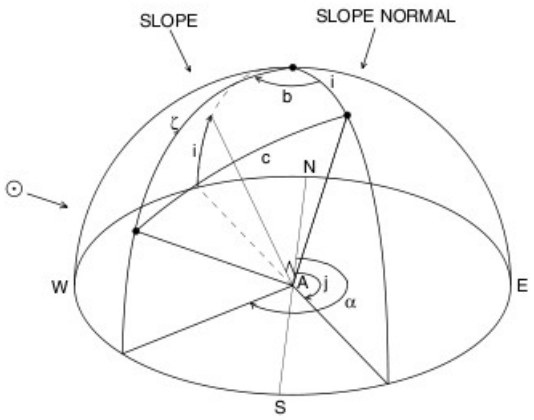

Figure 1 Calculation of solar incidence angle on a sloping surface. Symbols denote solar zenith angle (ζ), slope angle (i), slope aspect (j), solar azimuth angle (α), and b = α − j

6. A very short overview of terrain topography effect parameterization

6.1 Terrain effect on SW fluxes

We start with effect of the sloping surface on the shortwave (SW) radiation as some quantities will be reused for the longwave (LW) effect (horizon angle estimate).

The amount of direct solar radiation falling on a given surface is proportional to the area presented normal to the direction of the incoming solar beam. In a nut-shell the topography effect includes:

1. the effect of slope orientation on this area;
2. possibility for some slopes to be shaded by the neighbors from the direct solar radiation thus effectively excluding them from process of the reflection.

To characterize the problem the following parameters (see Figure 1) are required:

1. the solar zenith angle (ζ), solar azimuth angle (α);
2. slope angle (i), slope aspect (j).

If the effect of slope orientation can be computed quite straightforward (see Refs [1, 2]) and fully characterized by a factor

$$O_x = 1 + tan(i)\ tan(\zeta) cos(b)$$

However, this equation doesn't include a possibility for a cell to be shadowed by the neighbors that requires more elaborate computations. The general approach used in [1-4] is for a given grid point to estimate a horizon angle for a set of solar azimuth angles, the number of angles used may differ. If Manners et al [1] uses 16 angles whereas Muller and Scherer [2] uses 24 while Dozier et al [3] shows that 32 seems to be more adequate number for their test area (the southern Sierra Nevada).
General procedure used by all authors are very similar. For every azimuthal direction for the set of locations the horizon angle is calculated along the line of sight up to a specified maximum distance (e.g, 10 km [3]) where the horizon angle, H, (measured from local zenith) is estimated as

$$tan(H) = \Delta h / D$$

where Δh is the difference between orographic heights at the far and local points and D is the distance between these two points. As result the smallest horizon angle is selected to be a representative for this azimuthal direction.

The horizon angle computation is the most time-consuming procedure (some discussion about an efficient implementation can be found in [3]) since the horizon angle angular distribution has to be computed for number of azimuthal direction (in our case default 16) for every grid point at DEM resolution.

The computed horizon angles allow us to estimate whether a given grid cell will be illuminated for a given solar position by introducing a binary shadow mask

$$mask_{shadow}(\zeta,\alpha) = \begin{cases} 0, & \zeta < H_\alpha \\ 1, & otherwise \end{cases}$$

The total correction factor combines binary shadow mask and factor O_x and has a form:

$$f_{cor}(\zeta,\alpha) = O_x(\zeta,\alpha)\ mask_{shadow}(\zeta,\alpha)$$

Since the correction factor for a computational grid point just a weighted average of the all correction factors of all contributing DEM grid cells at preprocessing stage these average factors are computed for a set of solar zenith and azimuth angles. Then during the RRTMGP run a correction factor for a given solar position is found by 2D interpolation.

6.2 Terrain effect on LW fluxes

For SW the terrain shadowing results in blocking solar radiation and thus reduction in reflected radiation which affects radiance balance up to TOA. However, for LW the situation is somewhat simpler affecting only the surface level energy balance as downward radiance includes two components: flux from the sky and flux from the surface

$$F_d = \frac{1}{\cos(i)}\left[V\,F_{d,sky} + (1-V)\,F_{d,surf}\right]$$

where factor, V, is the sky-view factor and $cos(i)$ is used to correct for the extra area of the sloped surface of the grid-box. The sky-view factor has a form

$$V = \frac{1}{N}\sum_{n=1}^{N}\sin^2\left[H(\phi_k) + \frac{\pi}{2} + \frac{1}{\tan(i)\cos(\phi_k - j)}\right].$$

More detailed discussion on the sky-view factor derivation can be found in [1, 3]. Note the difference between the definition used in [1, 4] and [2] (the surface area correction is often omitted from parameterizations of diffuse surface radiation) that are discussed in more details in [1]
Here, the correct formulas that include effect of non-unit emissivity:

$$F_{net} = \frac{V}{\cos(i)}\frac{F_{net,0}}{1-(1-\varepsilon)(1-V)}$$

$$F_d = \frac{1}{\cos(i)}F_{d,0} - \frac{1-V}{\cos(i)}\frac{F_{net,0}}{1-(1-\varepsilon)(1-V)}$$

$$F_u = \frac{1}{\cos(i)}\left[F_{d,0} - \frac{F_{net,0}}{1-(1-\varepsilon)(1-V)}\right]$$

that allow to create a simple algorithm to include the topography effect for the surface level. Independence of the correction factor on solar angles allows us to precompute three quantities:

$$\left\langle\frac{1}{\cos(i)}\right\rangle,\ \langle V\rangle,\ \left\langle\frac{V}{\cos(i)}\right\rangle$$

for every computational grid point.

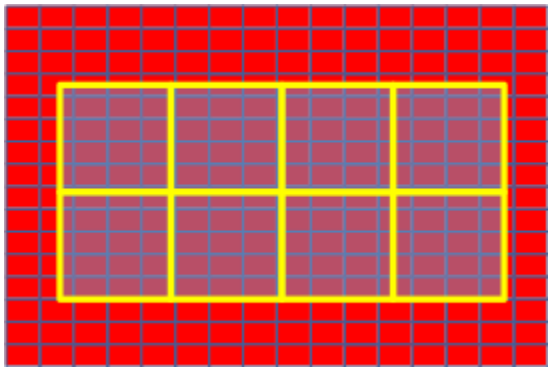

Figure 2 Assumed orientation of computational grid (yellow lines) with respect to the DEM grid (blue lines)

6.3 Implementation details

The code uses two grids:

computational grid is the grid used to run radiative transfer computation. It is defined by the most northern and southern latitudes and the most eastern and western longitudes. The user may also specify

different resolutions for latitude and longitude directions. According to the user input the code will construct a homogeneous rectangular grid and computes parameters for each cell.

DEM grid is the grid on which DEM is defined. The portion of the DEM dataset that processed is determined by the the user defined computational area which extends in all directions by the user defined number of DEM cells (the default value is 100) to ensure the correctness the horizon angles computations for DEM cells that are located close to the boundary of the computational area.

At current stage of the development both grids are assumed to be aligned and following the Earth geographical latitudes and longitude grid (Figure 2).

Any DEM cell may contribute to up to 4 different computational grid cell. In parameter estimation all DEM cells contributions take into the account the weight which defines as the potion of the DEM cell that makes a contribution to the computational cell. The weight varies between 0 and 1.

Currently, the above parameterization has been implemented as a Python preprocessor which for the user defined computational domain reads the DEM database and then compute:

- the total correction factor $f_{cor}(\zeta,\alpha)$ for a set of solar zenith and azimuth angles;
- $\left\langle \frac{1}{\cos(i)} \right\rangle, \langle V \rangle, \left\langle \frac{V}{\cos(i)} \right\rangle$.

Used algorithms are a combination of algorithms discussed in [1,2,3]

The raw digital terrain elevation data (DEM) is assumed to be aligned with the model grid. For each cell of the DEM, an x and y gradient is calculated by finite differences at the four corner points:

$$\frac{\partial h}{\partial x} = \frac{h_{i+1,j} - h_{i,j} + h_{i+1,j+1} - h_{i,j+1}}{2\Delta x}$$

$$\frac{\partial h}{\partial y} = \frac{h_{i,j+1} - h_{i,j} + h_{i+1,j+1} - h_{i+1,j}}{2\Delta y}$$

then slope angle (i), slope aspect (j) defined as:

$$\tan(i) = \sqrt{\left(\frac{\partial h}{\partial x}\right)^2 + \left(\frac{\partial h}{\partial y}\right)^2}$$

$$\tan(j) = \frac{\partial h}{\partial y} \Big/ \frac{\partial h}{\partial x}$$

The horizon angles is computed for every DEM cell for N azimuthal directions (default 16) (the maximum distance traced along each direction is default to 20 km).
Finally, the contribution of a given DEM cell to computational grid cell is computed as a portion of the cell area located within computational grid cell area.

7. Support

Igor Polonsky
Atmospheric and Environmental Research, Inc.
131 Hartwell Avenue
Lexington, MA 02421-3126
office: 781-761-2228
fax: 781-761-2499
e-mail: ipolonsky@aer.com

8. References

1. Manners, J., Vosper, S.B. and Roberts, N. (2012), Radiative transfer over resolved topographic features for high - resolution weather prediction. Q.J.R. Meteorol. Soc., 138: 720-733. doi:10.1002/qj.956

2. Müller, M.D. and D. Scherer, 2005: A Grid- and Subgrid-Scale Radiation Parameterization of Topographic Effects for Mesoscale Weather Forecast Models. Mon. Wea. Rev., 133, 1431–1442, https://doi.org/10.1175/MWR2927.1

3. J. Dozier, J. Bruno, P. Downey, A faster solution to the horizon problem, Computers & Geosciences, Volume 7, Issue 2,1981,Pages 145-151,https://doi.org/10.1016/0098-3004(81)90026-1.

4. Dozier, Jeff, Marks, Danny. (1987). Snow mapping and classification from LANDSAT Thematic Mapper data. Annals of Glaciology. 9. 97-103. 10.1017/S026030550000046X.