ESSI21 Moon Polar Ice and Ejecta Stratigraphy (MoonPIES)

Welcome to the `MoonPIES` model documentation. This document will outline how to install, configure, and run the model.

Motivation

The *Moon Polar Ice and Ejecta Stratigraphy* model investigates ice and ejecta with depth at the lunar poles. A recent model of ice delivery and loss to and from lunar cold traps suggested that "gigaton ice deposits" may be preserved at depth (Cannon et al. 2020).

Here, we reproduce the Cannon et al. 2020 model (hereafter *Cannon model*) and extend it to:

- Include the effects of *ballistic sedimentation*
- Update the treatment of *impact gardening*
- Improve constraints on *large icy impactors*

The model is intended to be easy to run,modify and extend as our understanding of lunar polar ice improves. See below for step-by-step instructions for running the model. For help getting started, please contact somebody@domain.com.

Model prerequisites

The model requires Python 3 and is tested on all main operating systems (Linux/Mac/Windows). We recommend installing python using Anaconda or Miniconda.

The model also requires the following python packages to be installed:

- numpy
- pandas

These are installed by default with *Anaconda*, but can also be installed with:

```
conda install numpy pandas
```

For more advanced environment configuration with `conda`, refer to the following conda environment tutorial.

Note: While the model will run on Windows, we recommend Windows users install the Windows Subsystem for Linux (WSL), particularly to simplify running the model in parallel. See WSL installation guide here.

Model Installation

The model is packaged as a `.zip` file which can be downloaded from *public repository link*. Once downloaded, extracting all contents of the `.zip` to a

directory of your choice. The unzipped package has the following structure:

```
moonpies_package/
|- data/
|  |- costello_etal_2018_t1.csv
|  |- needham_kring_2017.csv
|  |- crater_list.csv
|- figs/
|- moonpies/
|  |- config.py
|  |- moonpies.py
|  |- my_config.py
|- README.md
```

Running the model

Default configuration

To run the model, open up a terminal (Mac/Linux/WSL) or command prompt (Windows) and navigate to the `moonpies/` directory:

```
cd /path/to/moonpies/
```

For help navigating the terminal, see the following cheatsheet.

Now you can run the model in its default configuration using the following command:

```
python moonpies.py
```

The model will run and write outputs to a new folder in the `data/` directory. Ouputs are saved by today's date and random seed (e.g. `data/yymmdd/mpies_#####`, where `#####` is the 5-digit random seed used. The full directory structure will now look like:

```
moonpies_package/
|- data/
|  |- 210706/
|  |  |- mpies_85637/
|  |  |  |- run_metadata_mpies.csv
|  |  |  |- run_metadata_mpies.csv
|  |  |  |- crater_list.csv
|  |- costello_etal_2018_t1.csv
|  |- needham_kring_2017.csv
|  |- crater_list.csv
...
```

Custom random seed

A particular random seed can be specified as an argument to `moonpies.py` from the terminal like so:

```
python moonpies.py 12345
```

The seed can be any integer from 1 to 99999 (5-digits max). If no seed is specified, it is randomly chosen. Specifying the random seed allows you to reproduce prior model runs as long as an identical configuration is also used.

Custom configuration

All parameters to `moonpies.py` can be modified in a configuration file and passed as an argument. The `my_config.py` file in the `moonpies/` directory is an template which you may edit or copy to produce custom `config.py` files.

Warning: The `default_config.py` file contains the default parameters and should never be modified directly. However, `default_config.py` may be a useful reference for descriptions and units of all model parameters.

To run the model with a custom configuration, specify the `--cfg` or `-c` flag followed by the path to a custom `config.py` file:

```
python moonpies.py --cfg my_config.py
```

Note: If the seed and config file are both specified, the cmd-line seed will always take precedence. E.g., `12345` will be the random seed even if `seed` is defined in my_config.py.

```
python moonpies.py 12345 my_config.py
```

If no random seed is specified, the model will first look for a `seed` field in `my_config.py` and then fall back on randomly choosing a seed as in the default case.

Note: Since model runs with the same `run_name` and `seed` will be written to the same directory, you may overwrite outputs if you change configurations without changing the name. We recommend that each new configuration has its own `config.py` file defined and a unique `run_name` selected to avoid output conflicts.

Running in parallel (Linux/Mac/WSL only)

The `moonpies.py` model is compatible with GNU parallel which allows the model to be run in parallel (Tange, 2011).

GNU parallel is available from UNIX package managers. E.g.:

```
apt install parallel  # Ubuntu / WSL
brew install parallel  # MacOS
```

Note: On MacOS, you may first need to install homebrew (see brew.sh).

Now, many iterations of the model may now be run in parallel with a simple command:

```
seq 1000 | parallel -P-1 python moonpies.py
```

This example will start 1000 runs of `moonpies.py`, each with a unique random seed and output directory so that no data is overwritten. To configure your `parallel` runs:

- The number of runs is given by the `seq N` parameter.
- By default, `parallel` will use all available cores on your system but this can be configured with the `-P` flag. Ex. Use `-P2` to use 2 cores; use `-P-1` to use all cores except one.

Model details

Here, we will give an overview of the main functions in the `MoonPIES` model.

The model is divided into two primary sections: **Setup** and **Main loop**. The Setup phase prepares all stratigraphy columns, while the Main Loop steps through time to add and remove ice from each column.

Setup

The setup phase initializes all stratigraphy columns and also pre-computes any data that is unchanged in the Main Loop.

1. `read_crater_list()`: Import list of large craters near the pole.
2. `randomize_crater_ages()`: Randomly vary age of craters within their error bars.
3. `get_ejecta_thickness_matrix()`: Pre-compute ejecta thickness at each point on the model grid vs. time (3D array: $GridX \times GridY \times Time$).
4. `get_volcanic_ice()`: Pre-compute ice-delivery by volcanic outgassing at each time (1D array: Time).
5. `get_ballistic_sed_matrix()`: Coming soon!
6. `init_strat_columns()`: Initialize empty ice columns and their associated ejecta thickness columns over time (1D arrays: Time)

Main Loop

The Main Loop steps through model time from the past to the present and accumulates ice in each strat column.

1. `new_ice_mass()`: Compute the total mass of ice delivered to the polar region in this timestep.
2. `get_ice_thickness()`: Convert mass of ice to 1D thickness added to each strat column.

3. `update_ice_cols()`: Apply ice addition and removal processes for this timestep (see below)

The treatment of ice in the main loop depends on the mode of the model.

ESSI mode: If `mode == 'essi'`, then `update_ice_cols()` will do the following:

1. *Coming soon*: If a crater was formed at this timestep, apply ballistic sedimentation effects to all ice columns based on their distances from the impact.
2. Add new ice thickness to each strat column.
3. Remove ice to a gardening depth determined by the current model time using the Costello et al. (2020) model scaled by the historical impact flux (Ivanov et al. 2000).

Cannon mode: If `mode == 'cannon'`, then `update_ice_cols()` will do the following:

1. Add new ice thickness to each strat column.
2. Remove ice to a constant depth of 10 cm at all times using the method of Cannon et al. (2020)

When run in *Cannon mode*, our model outputs yield similar ice thicknesses over time as those published in Cannon et al. (2020).

Module documentation

Coming soon! Here we describe each module in more detail.

Authors

This model was produced by CJ Tai Udovicic, K Frizzell, K Luchsinger, A Madera, T Paladino, M Kopp, M Meier, R Patterson, F Wroblewski, G Kodikara, and D Kring. It is adapted and updated from the model by Cannon et al. (2020).

License and Citations

Coming soon! This code is made available under the ____ license which allows use with attribution. The can be cited as:

> Authors et al. (2021). Title.

Acknowledgements

This model was produced during the 2021 LPI Exploration Science Summer Intern Program which was supported by funding from the Lunar and Planetary Institute and the NASA Solar System Exploration Research Virtual Institute.