pySLAMM 6.7 beta, User's Manual

July 2024

warren pinnacle consulting, inc.

PO Box 315
Waitsfield, VT 05673
(802)-496-3476

pySLAMM 6.7 beta, User's Manual

Contents

Introduction and Acknowledgements

pySLAMM 6.7 is a translation of the SLAMM 6.7 code from Delphi (object-oriented Pascal in Windows) to Python 3.11. This project was funded by the Coastal Resilience branch of NCCOS (NOAA) under contract to Consolidated Safety Services Incorporated (CSS) with oversight from Christine Addison Buckel, Ramin Familkhalili and Rebecca Atkins.

All model process code was translated to Python and tested under multiple operating systems. However, a graphical user interface (GUI) is not yet part of the Python version of SLAMM. On the other hand, pySLAMM 6.7 can read SLAMM txt files produced by SLAMM 6.7 Delphi and has been tested to machine accuracy against model results produced by SLAMM 6.7 Delphi. In this manner, models can still be produced using the Windows GUI and then production runs can be completed in Python under Windows, Linux or alternative operating systems. pySLAMM6.7 can also read inputs from and write outputs to GeoTiff format which will simplify access to model data via GIS.

The SLAMM model can be accessed in Python in several different methods:

- The model can be executed calling a Python script from the command line (**SLAMM_Run.py**) and passing the model input file as a parameter. A model run will automatically complete.
- A simplified command-line input may be utilized by running the **pySLAMM6_7.py** file. A set of commands will be shown (e.g. "Load" "Run_model") if the user enters "?"
- A Python script can be produced that imports SLAMM6.7 objects and modifies those directly in a user-produced script.

All three of these methods will be discussed in sections below.

This model was developed using Python 3.11.

Please use the SLAMM-Forum to ask SLAMM interface questions as well as other technical questions about the SLAMM Model. It is our goal for the forum to serve as a knowledgebase for model users.

Installation

Linux

1. Transfer the Installation File
   Use scp to transfer the file to the user directory
   update sudo if required "sudo apt update"
   install pip3 if required "sudo apt install python3-pip"

   note, if sudo apt is already running daily maintenance
   tasks you may need to wait a few minutes.

2. install Python 3.11 if required
   (older versions of ubuntu such as 18.0.4 may require a manual install
   https://medium.com/@elysiumceleste/how-to-install-python-3-11-3-on-ubuntu-18-04-e1cb4d404ef3)

3. install a virtual environment if required,
   install the tools: "sudo apt install python3-venv"
   create the virtual environment: "c"
   activate the virtual environment: "source ~/venv/bin/activate"

4. use pip3 to install the wheel file:
   pip3 install pySLAMM-6.7.0-py3-none-any.whl

   OR
   pip3 install pyslamm-6.7.0.tar.gz

5. navigate to the Kakahaia directory to run the test
   cd venv/Kakahaia
   python3.11 ../lib/python3.11/site-packages/Basic_Run.py Kakahaia.txt

Notes: This assumes that Basic_Run.py is in the default installation location --
   it may need to be located and the path corrected
   depending on the linux installation the test-run command may just be python without "3.11"

(This is just a test deployment the final deployment will have the user run python, import
the SLAMM library, and run commands from a python prompt)

Windows

1. Install Python 3.11, download from https://www.python.org/downloads/

2. install a virtual environment from a Windows CMD prompt
   python -m venv test_env
   test_env\Scripts\activate

3. copy the installer file to a directory on the server and navigate to that folder
   pip3 install pySLAMM-6.7.0-py3-none-any.whl

   OR
   pip3 install pyslamm-6.7.0.tar.gz

4. navigate to the Kakahaia directory to run the test

```
CD C:\Users\{username}\AppData\Local\Programs\Python\Python311\test_env\Kakahaia
python C:\{Path to Basic_Run}\Basic_Run.py Kakahaia.txt
```

(This is just a test deployment the final deployment will have the user run python, import the SLAMM library, and run commands from a python prompt)

Please send any questions to jclough@warrenpinnacle.com ll model process code was translated to Python All model process code was translated to Python

SLAMM Input Text File Format

The SLAMM input file format is a flat text file with parameter names followed by a colon and then the associated parameter value. The text file format has not changed from SLAMM 6.7 to ensure compatibility with SLAMM 6.7 Delphi. Parameter names must be precisely specified in the order expected or SLAMM will give an error message such as:

Failed to load simulation: Text Read Name Mismatch, Line 42 Expecting Variable 'IsTidal'; read variable 'IsNonTidalWetland'

Text files are organized by parameter group but they may also be split into individual files. From the pySLAMM6_7 command line, a save or saveas command can include the parameter to split files.

saveas Simulation.txt TRUE

In this case, the text file is separated into pieces

- Simulation.txt – basic parameters
- Simulation_categories.txt—a description of SLAMM categories in the simulation
- Simulation_elev_stats.txt—elevation statistics for the simulation if they have been derived
- **Simulation_file_setup.txt**—an important file that points to all of the GIS model inputs
- Simulation_infrastructure.txt – describes road and point infrastructure if relevant
- **Simulation_run_options.txt** – a selection of options for SLR scenarios and model setup options such as dates to run.
- Simulation_salinity – describes fresh-water flow data and rules for converting categories based on salinity.
- **Simulation_site_data.txt**—all of the parameters for the site and divided subsites such as accretion rates and erosion parameters.
- Simulation_uncertainty_sensitivity.txt—setup parameters for uncertainty and sensitivity analyses.
- Simulation_wind_rose.txt—data required if the wave-power erosion is to be used.

While all of these files must be present, defaults may be used in many cases for a simulation other than **file_setup, run_options,** and **site_data.**

Note that pySLAMM can read files from Delphi SLAMM 6.7 and Delphi SLAMM 6.7 can also read text files from pySLAMM. However, Delphi SLAMM 6.7 cannot yet read the split-file format. In some cases it may be efficient to use the Windows version of SLAMM 6.7 to modify parameters so that text files are properly written or to have a GUI access to model parameters.

Simple Command Line Access (pySLAMM6_7.py)

pySLAMM6_7 is a command-line interface (CLI) designed to facilitate the interaction with the TSLAMM_Simulation model. This guide provides instructions on how to use the shell, including loading and saving simulations, setting and viewing parameters, and running the model.

To start the command-line access locate pySLAMM6_7.py and type "python pySLAMM6_7.py"

To see the list of available commands a user may type "?" at any time.

Available Commands:

- **load {file_name}** Loads a simulation from the specified file. *Note that file names may need to be in quotes if they include spaces or the backslash character*

- **new** Creates a new simulation with default parameters. File locations for required raster files must be set and SLR scenarios selected before running the model.

- **save {split_files=false}** Saves the current simulation to its original file. If the file exists, prompts for overwrite permission. Type "save true" if you wish to split the file into several files based on parameter categories.

- **saveas {file_name} {split_files=false}** Saves the current simulation to a new file. If the file exists, prompts for overwrite permission. *Note that file names may need to be in quotes if they include spaces or a backslash character.* If an optional third parameter after the file name is "true," the files are split as shown on the previous page.

- **set {parameter} {value}** Sets one of a selected set of parameters to the value provided. Type "set ?" to get a complete list of parameters that can currently be edited via this interface. They are also listed below. Example command "set max_year 2050"

- **show {parameter}** Shows the value of one of the selected set of parameters. Type show ? to get a complete list of parameters that can be shown

- **show all** Lists current values for all of the parameters that can be set via this simple command-line interface.

- **run_model {cpu count}** Runs the model for all of the selected protection scenarios and sea-level-rise scenarios specified. CPU count is an optional parameter and is set to the number of CPUs available if left blank.

- **quit** Exits the SLAMM command-line interface.

Parameters
The following parameters can be set using the set command. Each parameter has a specific type as indicated in the list below:

- **file_name**: {string} the file name associated with the simulation
- **sim_name**: {string} a descriptive name for the simulation
- **time_step**: {int} time-step in years
- **max_year**: {int} last year of the simulation

- **run_specific_years**: {true/false} Instead of a time-step a user may specify years to run
- **years_string**: {"comma separated ints"} List of years to run, e.g. "2050, 2065, 2075"
- **run_uncertainty**: {true/false} Whether to run an uncertainty analysis, note this requires that uncertainty parameters have been set up.
- **run_sensitivity**: {true/false} Whether to run an uncertainty analysis, note this requires that sensitivity parameters have been set up

- **gis_years**: {"comma separated yrs"} List of years to output GIS data e.g. "2085, 2100"
- **gis_each_year**: {true/false} Output GIS data each year model is run

- **elev_file_name**: {string} DEM raster file name **(required)**
- **nwi_file_name**: {string} SLAMM land-cover classes raster **(required)**
- **slp_file_name**: {string} SLOPE raster **(required)**
- **imp_file_name**: {string} Percent Impervious raster (optional)
- **ros_file_name**: {string} Raster output site file name (optional)
- **dik_file_name**: {string} File with dike data (optional)
- **vd_file_name**: {string} Vertical Datum input raster (optional)
- **uplift_file_name**: {string} Uplift data raster (optional)
- **sal_file_name**: {string} Salinity raster input (optional)
- **d2m_file_name**: {string} Distance-to-mouth raster for SAV calculation (optional)
- **storm_file_name**: {string} Storm surge raster name (optional)
- **output_file_name**: {string} Name and file location to save output (optional)

- **Scen_A1B**: {true/false} These parameters define which IPCC Scenario is used
- **Scen_A1T**: {true/false}
- **Scen_A1F1**: {true/false}
- **Scen_A2**: {true/false}
- **Scen_B1**: {true/false}
- **Scen_B2**: {true/false}

- **Est_Min**: {true/false} These parameters define which IPCC estimate is used
- **Est_Mean**: {true/false}
- **Est_Max**: {true/false}

- **1_meter**: {true/false} These are additional fixed SLR scenarios by 2100
- **1.5_meter**: {true/false}
- **2_meter**: {true/false}

- **Protect_NoProtect**: {true/false} Run simulations in which dry land not protected
- **Protect_ProtDeveloped**: {true/false} Developed dry land is assumed protected
- **Protect_ProtAll**: {true/false} Run a sim. In which all dry land is protected

Running a model from Command Line (SLAMM_Run.py)

In addition to the interactive command line, you can run the pySLAMM model by passing the input file as a parameter to the SLAMM_Run.py script. This method provides a straightforward way to execute the model without interacting with the shell.

Usage example "python SLAMM_Run.py <filename>"

Specific steps required are:

- Open your command-line interface (CLI).
- Navigate to the directory where SLAMM_Run.py is located.
- Execute the script with the path to your simulation file.

Access SLAMM Objects via Python Scripts

The SLAMM_Run.py script provides insight into how a user may write their own script to execute the SLAMM model and change parameters as required. The basic object of the model is the TSLAMM_Simulation class that can be accessed as.

```
simulation = TSLAMM_Simulation()
```

To load a file, an open file handle is passed to the "load_store" method that is used both to read and write SLAMM text files:

```
with open(file_path, 'r') as file:
    simulation.load_store(file, file_path, VERSION_NUM, True)
```

Python scripts can then be written to execute the model and change model parameters as found in the TSLAMM_Simulation class located in SLR6.py.