RWTH Aachen

Science Proposal

GreenSloth: A Curated Web Resource for Validating and Comparing Peer-Reviewed Computational Models of Photosynthesis

---

Author:
Elouën Corvest

Supervisor:
Prof. Anna Matuszyńska

January 28, 2026

Contents

1 Introduction

2 Material and Methods

2.1 Model Packaging

2.2 Model Demonstrations

2.2.1 Daylight Simulation

2.2.2 FvCB Addon

2.2.3 Standard PAM Simulation

2.2.4 MCA of Photosynthesis

2.2.5 Fitting of Flourescence and NPQ

2.3 GreenSlothUtils

The **GreenSlothUtils** package, written in Python, contains several utility functions to assist in packaging an already written kinetic Ordinary Differential Equation (ODE) model into a more code-readable and ready for uploading to the **GreenSloth** website format. The functions included in this package vary in their complexity, from simple directory creation to rewriting model files. While **GreenSlothUtils** has been written in a packge format, there is no intention to upload to package sharing services such as PyPI. However, the package is intended to be uses as a local package to assist in contribution to the **GreenSloth** website, therefore the GitHub repository is publically available at `https://github.com/ElouenCorvest/GreenSlothUtils.git`.

2.3.1 Documentation

All the functions included in the package are documented in the GitHub repository, however some key functions are also documented below to showcase their utility in this project. This biggest upside of this package is the ability to quickly create a new model directory with all the necessary files and format for uploading to **GreenSloth**. For ease of access, the functions required for this are combined inside a Command-line Interface (CLI).

Easily the most important part of the CLI of **GreenSlothUtils** is the `--help` command (see Code-Block 1), which gives an overview of all the possible commands and options available in the package. Each of the other commands also include a `--help` option to give more specific information about the specific command.

Using these commands following a specific order allows for a quick and easy creation of a new model directory.

1. Create the directory

First, the `initialize` command is used to create a new and complete model directory with the correct name and structure (see Code-Block 2). One required argument is the name of the model, of which the directory should be named after. In the **GreenSloth** ecosystem, the name of the model is the first author's last name followed by the year of publication, e.g., `Corvest2000`. This command also includes one optional argument, `--path`, which allows the user to specify where the new model directory should be created. If no path is given, the directory is created in the current working directory. The directory created contains various different sub-directories and files, all preformatted to the **GreenSloth** website standards (see Directory Tree 1).

```
Corvest2000/
├── figures/
│   ├── demonstrations.ipynb
│   └── paper_figures.ipynb
├── model/
│   ├── basic_funcs.py
│   ├── derived_quantities.py
│   ├── __init__.py
│   └── rates.py
├── model_info/
│   ├── comps.csv
│   ├── derived_comps.csv
│   ├── derived_params.csv
│   ├── params.csv
│   └── rates.csv
└── README_script.py
```

Directory Tree 1: Example directory structure created by the `initialize` command of GreenSlothUtils.

The `figures` directory consists of two Jupyter Notebooks. The `demonstrations.ipynb` includes all Model Demonstrations for the model, already written out, with each demonstration having its own cell. The creator of the model simply has to replace the `None`

```
$ GreenSloth-init --help

Usage: GreenSloth-init [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  compare-gloss-to-model    Compares glosses to model
  from-model-to-gloss       Generate temporary Glosses from model info.
  initialize                Create '<model-name>' directory.
  latex-from-model          Write LaTex from Model
  python-from-gloss         Write Python Variables from Glossaries
  update-glosses-from-main  Update glosses from main
```

Code-Block 1: The help command of the CLI of GreenSlothUtils

```
$ GreenSloth-init initialize --help

Usage: GreenSloth-init initialize [OPTIONS] <model-name>

  Create '<model-name>' directory.

Options:
  -p, --path TEXT  Path to create model directory. Defaults to path here.
  --help           Show this message and exit.
```

Code-Block 2: The initialize command of the CLI of GreenSlothUtils

values of each string representation of model parts needed for the demonstration. For more information on what the demonstrations entail, please refer to Section 2.2. The `paper_figures.ipynb` is a blank notebook intended to be used to recreate the figures of the model's publication. Both notebooks already include necessary imports and helper functions to quickly get started. If the model is correctly implemented inside the `model` directory, the importing of the model should work.

The `model` directory consists of the actual ODE model implementation in Python. For ease of use, **GreenSloth** ecosystem sets using the `mxlpy` [1] package as a standard for writing kinetic ODE models in Python. This has been chosen due to its ease of use, speed, and continuous support from the maintainers. The most important file of the `model` directory is the `__init__.py` file, which contains the main model initialization. As a strong recommendation, the actual parts of the model, such as the rates, basic functions, and derived quantities, should be written in separate files, as shown in (see Directory Tree 1). This allows for better code readability and easier debugging. To help with formatting a model written using `mxlpy`, the **GreenSlothUtils** package contains a subpackage, `greenslothutils.mxlpy_formatter`, which contains several functions to assist in rewriting a model into the correct format for **GreenSloth**.

The `model_info` directory contains all the necessary information about the model in a `.csv` format. This includes the variables, derived varaibles and parameters, parameters, and rates of the model. Each of these files follow a specific format, where a short description, the publication depiction, the **GreenSloth** depiction, and the variable used in Python code. Additionally, the variables and rates also include a column of KEGG IDs and Glossary Indeces to assist in linking the model to existing databases and the **GreenSloth** overarching glossary. The parameters, on the other hand, also include a column for units, values and sources given in the publication. These `.csv` files are used to automatically generate the glossary entries of the model

when uploading to the **GreenSloth** website, therefore it is important that they are correctly filled out. To assist in this process, functions have been created, that extract the valuable information from an existing model implementation in `mxlpy` into the tables, already seperating them into the correct columns. But more on those functions later.

Finally, the `README_script.py` file is a template script intended to be used to write the model's README file for the **GreenSloth** website and in general the model's documentation. This script includes several prefilled sections, such as tables extracted from the prior mentioned `model_info` directory, a generic installation guide, and the assumptions and brief description of the model demonstrations. The user simply has to fill in the specific details of the model, such as a short summary, LaTeXversion of the equations used, the recreation of the publication figures, and notes to the demonstrations. Writing the README in this Python script format allows for easier accessing of repeating information, such as specific variables or parameters, without having to manually find them and replace them in a written Markdown file. Make a few changes in the script, run it, and the README file is ready to go.

2. Extract Information from the Model

Once the directory is created and the model is completedly implemented, the next step is to extract necessary information from the model into the `model_info` tables. This is done using several commands of the **GreenSlothUtils** CLI.

2.4 Website

2.5 Additional Methods

2.5.1 Usage of AI

To stay completely transparent during this thesis, it has to be noted that several Artificial Intelligence (AI) tools have been used to assist in the code writing, documentation, and text writing process. Several Large Language Models (LLMs) have been used, including Gemini (Google) to help with finding models, ideas and writing code snippets for visualisation. Additionally, GitHub Copilot has been used to assist in code writing, mainly to streamline the writing and debugging process. The exact parts of this thesis that have been assisted by AI tools cannot be exactly determined, as the suggestions are mainly used to inspire and speed up the writing process. However, every part of this thesis has been critically evaluated and edited by the author to ensure scientific and real accuracy and quality.

3 Results

4 Discussion

References

[1] Marvin Van Aalst et al. *MxlPy - Python Package for Mechanistic Learning in Life Science*. 2025. DOI: 10.1101/2025.05.06.652335.