Reaction Simulator:

A BIOEN 437 project

By: Matthew Van Ginneken & Olivia Walsh

Background

- Create a web based app to visualize reaction kinetics
  - Easy to use for beginners
  - Accessible
  - Simple
- Similar Packages
  - COPASI
  - JigCell

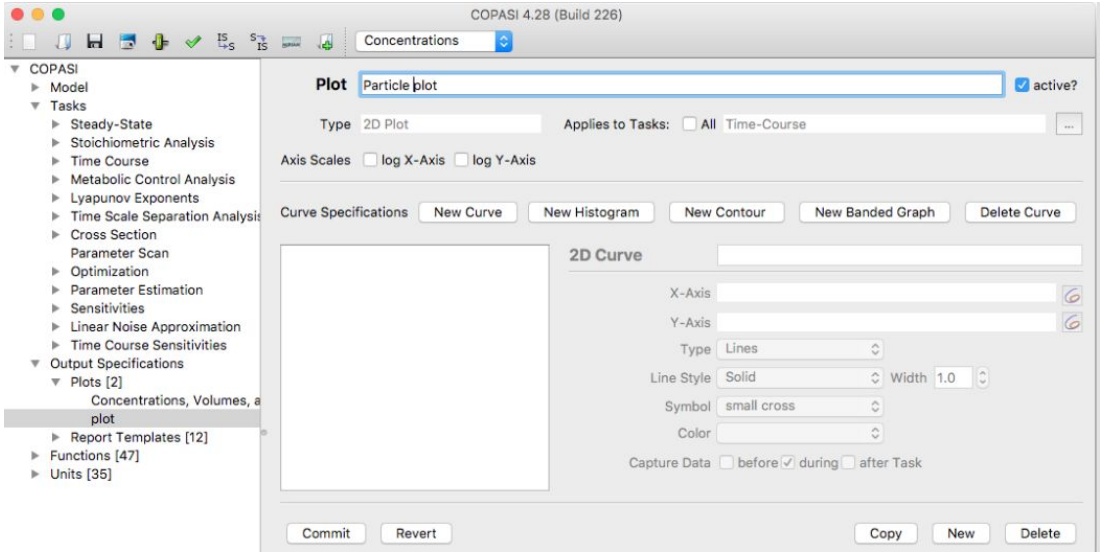
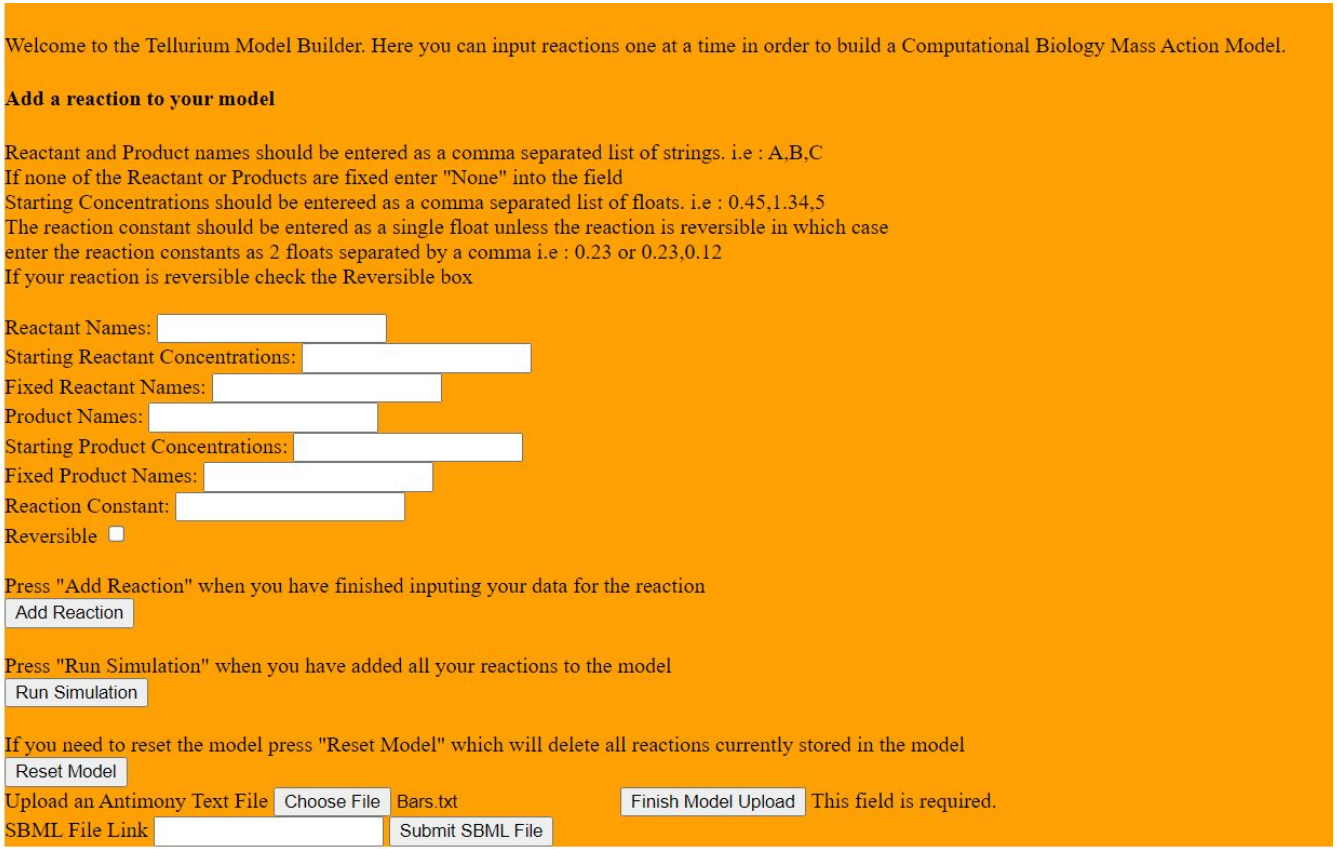


Figure 1: A clip of COPASI graph creator

Use Cases

1) The user can iteratively submit reactions and constants and then view the steady state graph
2) The user can submit an antimony txt file and then view the steady state graph and steady state values
3) The user can submit an SBML file and then view the steady state graph and steady state values

Demo


Welcome to the Tellurium Model Builder. Here you can input reactions one at a time in order to build a Computational Biology Mass Action Model.
Add a reaction to your model
Reactant and Product names should be entered as a comma separated list of strings. i.e : A,B,C
If none of the Reactant or Products are fixed enter "None" into the field
Starting Concentrations should be entereed as a comma separated list of floats. i.e : 0.45,1.34,5
The reaction constant should be entered as a single float unless the reaction is reversible in which case
enter the reaction constants as 2 floats separated by a comma i.e : 0.23 or 0.23,0.12
If your reaction is reversible check the Reversible box
Reactant Names:
Starting Reactant Concentrations:
Fixed Reactant Names:
Product Names:
Starting Product Concentrations:
Fixed Product Names:
Reaction Constant:
Reversible
Press "Add Reaction" when you have finished inputing your data for the reaction
Add Reaction
Press "Run Simulation" when you have added all your reactions to the model
Run Simulation
If you need to reset the model press "Reset Model" which will delete all reactions currently stored in the model
Reset Model
Upload an Antimony Text File
Choose File
Bars.txt
Finish Model Upload
This field is required.
SBML File Link
Submit SBML File

Design - Flask Webpage

User Runs Python Files

↓

python Initializes Flask App And hosts it on the local web port.

↓

user navigates to web route http://127.0.0.1:5000

↓

System Restg Directs user to Builder

Uses Python Func() to reset model

Reset Model

Add Reaction

Sends user input to Save2Dict

Model Upload

Runs loadModel()

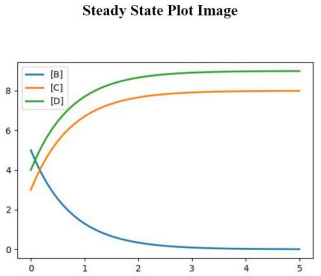
Submit SBML File

Runs loadSBMLO

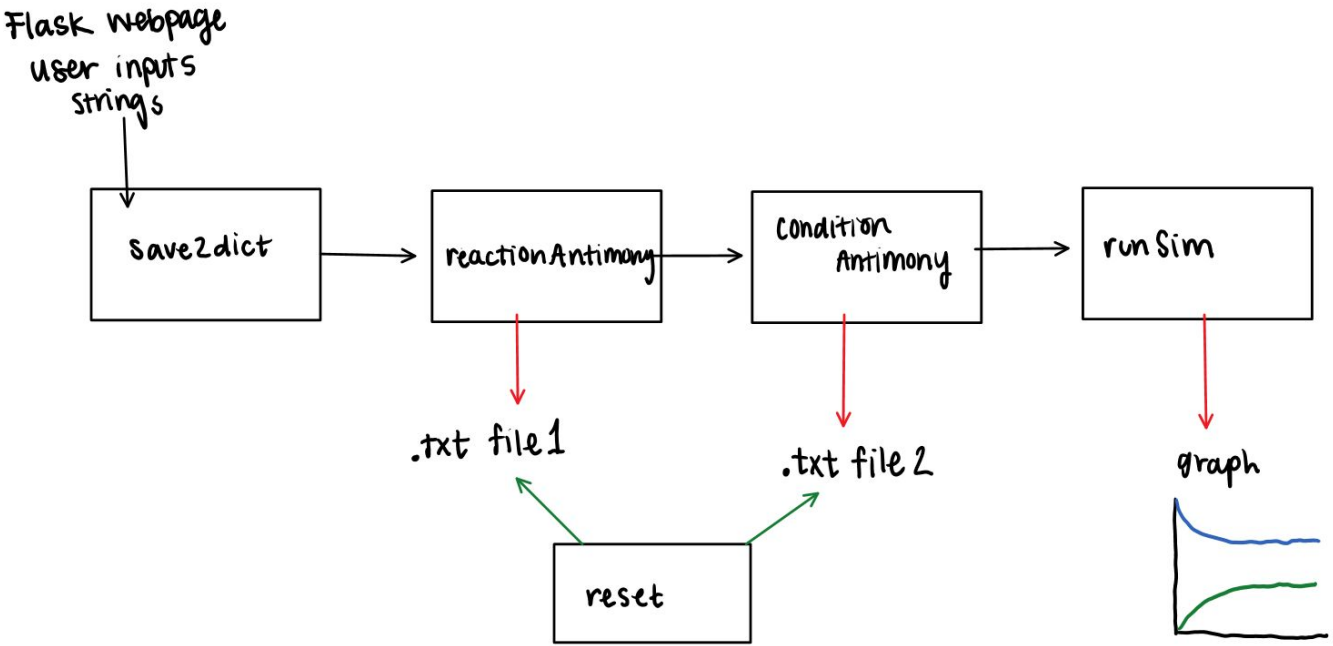
Run Simulation

Runs runSim()

Tellurium is Run and SS Graph is Produced



Design - Running the Simulations Through Python



Design - Python Functions

- Save2dict
- Antimony Builder
  - 2 separate functions, iteratively add to two separate txt files
- runSim
  - JPG file of the graph

```
dict = {
    'Reactants' = ['A', 'B'],
    'fixedReactants' = ['A'],
    'ReactantIC' = [10,2],
    'Products' = ['C'],
    'fixedProducts' = ['C'],
    'ProductIC' = [0],
    'Reversibility' = False,
    'RxnConstant' = [3]
```

```
$A+B+C->E ; k0*A*B*C
E->$A+B+C ; k1*E
```

```
A=10;B=0;C=0
E=3
k0=10;k1=2
```

Project Structure

AMCowmvg.egg-info	Working Pip Installable Version
AMCowmvg	Working Pip Installable Version
__pycache__	Testing2
dist	Working Pip Installable Version
docs	Final Final Version
examples	Working Pip Installable Version
LICENSE	Testing2
MANIFEST.in	Working Pip Installable Version
README.md	Final Final Version
pyproject.toml	Working Pip Installable Version
setup.cfg	Working Pip Installable Version
test_unit.py	Testing2

```
D:.
|   LICENSE
|   MANIFEST.in
|   pyproject.toml
|   README.md
|   setup.cfg
|   test_unit.py
|
+---AMCowmvg
|   |   antimonyTools.py
|   |   forms.py
|   |   routes.py
|   |   testing.py
|   |   __init__.py
|   |
|   +---static
|   |       antimony1.txt
|   |       antimony2.txt
|   |       output.jpg
|   |       test.txt
|   |
|   +---templates
|           buildSystem.html
|           home.html
|           results.html
|           testing.html
|
+---docs
|       Component Specifications.pdf
|       Final Presentation.pdf
|       Functional Specs_updated.pdf
|
+---examples
        Antimony_Text_Upload.mp4
        Base_Case.mp4
        reset.mp4
        SBML_Upload.mp4
```

Lessons Learned

- Working on different parts of the code then combining
- Creating tests using Python's *unittest*
- Documentation and process of creating working code and submitting as a python package

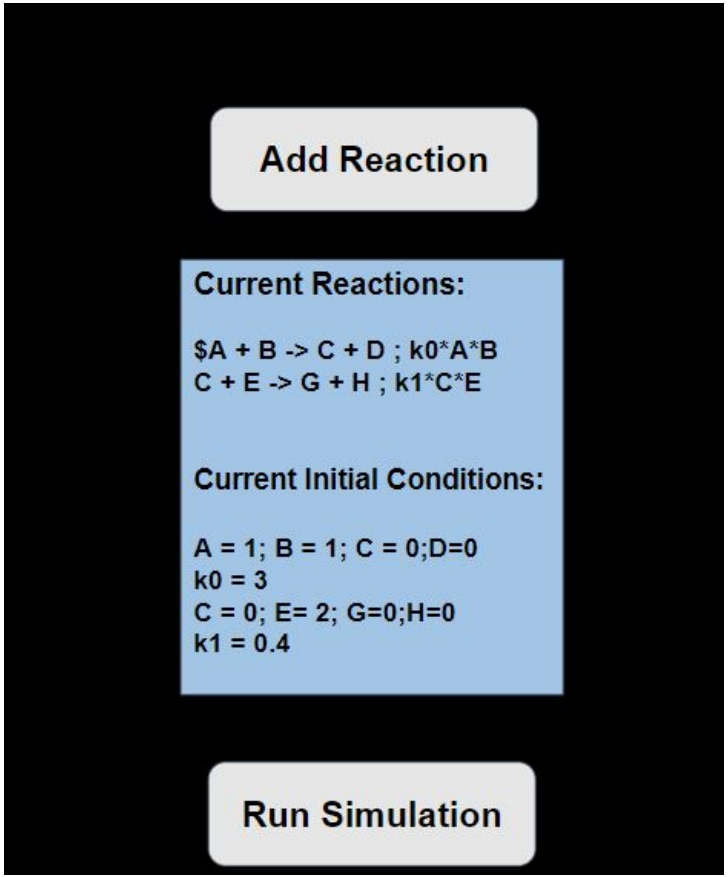
Future Directions

- Adding in other functions from tellurium
  - Showing different features of the simulation such as steady state values and system fluxes
- User can export their antimony txt file and then resume working with it on our webpage
- Being able to remove a reaction after it has been submitted

Future Directions

Listing the reactions that are submitted

- Help the user with organization
- User can check if the reaction was inputted correctly



Future Directions

Allowing the user to simulate more than one type of reaction kinetics

- Implemented with options given in a home page