Simulation Library Optimization and Parallelization of Stochastic Simulation Code for Computational Epidemiology

Parallelizing the StochPy Library

Resistance Epidemiology Modeling Initiative
Dr. Eric Lofgren

WASHINGTON STATE UNIVERSITY
Resistance Epidemiology Modeling Initiative

REMI-HPCStochPy

Michael J McNaughton
Ryan Charit
Gerald Hoff

September 28, 2022

I. Introduction

I.1 Project Introduction

The Resistance Epidemiology Modeling Initiative (REMI) conducts simulations modeling epidemics, particularly in smaller populations and focused on the beginning and end of epidemics. Such models can help with preventing and containing present and future epidemics and as such are of great utility. Most of the major tools for disease modeling in the field are deterministic, which function well for large population sizes. However, since the REMI lab focuses most of their research around smaller populations, a stochastic modeling tool is preferred, as stochastic models are superior for use in small populations [1]. Unfortunately, the tools for such modeling methods are not as highly developed as would be ideal for a research field as important as epidemiology.

As such, our project problem is to create an optimized and parallelized fork of the StochPy library. This library is used by REMI to conduct stochastic simulations in computational chemistry, physics, epidemiology, and ecology. The shortcoming of StochPy is that the stochastic simulation algorithm is slow, especially when trying to perform large computations with upwards of 10 million groups of simulations. However, this algorithm is also embarrassingly parallel, which means that if the codebase were refactored to run the simulations on many cores at once, the amount of time for a job to complete would be drastically reduced [2].

I.2 Background and Related Work

Stochastic simulations are generally performed using a process called the Gilespie direct method, making use of an algorithm popularized by Daniel Gilespie [3]. The algorithm converts the rates at which certain events take place to probabilities that that event will take place. The simulations use the probabilities generated by the algorithm to determine which of a set of potential events take place by drawing from a bag of weighted probabilities for each timestep. In the case of epidemiological simulations, events can include a new individual being infected or an infected individual recovering. An increase in the population size in such simulations also increases the frequency of events occurring and thus increases the computational intensity of the simulations which drastically reduces the speed at which simulations for higher population sizes can be completed.

The current version of StochPy is state-of-the-art at the time of the creation of this report [4]. That is, there currently are no available tools or libraries known to the REMI lab or its associates that outperform the StochPy library for their purposes. If our fork of the library were to demonstrate significant speedup, it would become an incredibly useful asset to the lab and any other labs that work with large-scale stochastic simulation for epidemiological research.

Knowledge and skills we will need to learn in order to complete this project include developing a basic understanding of the functionality of the StochPy library in its current state, as well as Python tools and libraries for increasing the speed of programs through the use of just-in-time (JIT) compilation and parallelization, such as by using the Numba and Dask libraries.

There are libraries that can do stochastic simulations similar to StochPy in parallel, Cayenne [5] being one of them. However Dr.Lofgren has specified that the improvement to

StochPy is still warranted as there are some functionalities of StochPy which Cayenne does not include, on top of it being much more efficient for the many individuals making use of StochPy if the library can be improved rather than needing to shift to another library entirely.

I.3 Project Overview

The Resistance Epidemiology Modeling Initiative is a product of collaboration between the WSU Paul G. Allen School for Global Health and the WSU School of Electrical Engineering and Computer Science. REMI aims to apply advanced computational methods to problems in public health, such as antimicrobial resistance, healthcare-associated infections, and COVID-19. In order to pursue its goals, one of REMI's most frequent activities is running simulations of epidemics centered around small, localized communities of people, such as schools and hospitals. This contrasts with other, more traditional epidemic simulations that focus on large, dense population centers like New York City.

This focus on smaller population centers necessitates the use of stochastic simulation models, as opposed to deterministic models for larger population centers. Deterministic models make use of fractional measurements, handle variance poorly, and are incapable of allowing a simulated disease to go extinct. This is problematic, as nonsensical measurements like one-quarters of an infected nurse in a hospital ward, inability to properly simulate the chances of a disease with a relatively low R_0 value spreading into a small scale outbreak, and inability for a simulated disease to die out are all detrimental when determining if the simulated disease grows into an epidemic or perishes before becoming an issue. However, as population sizes increase, stochastic simulations become more computationally intensive as the number of timesteps between events gets smaller, so running the many simulations necessary for the research with populations of moderate size sequentially even on HPC clusters can take several days.

Indeed, one of REMI's main limitations in conducting its stochastic epidemic simulations is the slow speed at which they are executed, which becomes a glaring issue when considering that upwards of tens of millions of simulations may need to be run for a single research experiment. This is a direct consequence of StochPy, the stochastic simulation Python library in use by REMI, being designed for single-threaded execution. Moreover, StochPy's authors and users have made no public attempt to optimize the library, and other stochastic simulation libraries may not meet REMI's needs or smoothly interface with existing lab codebases and digital infrastructure.

Slow simulation speeds pose a significant issue for REMI. Given that REMI is involved with research surrounding antimicrobial resistance, COVID-19, and other important public health subfields, improving the speed at which simulations are conducted is critical for accelerating potentially lifesaving research, aiding in rapid response to potential outbreaks, and assisting with formulating public policy decision recommendations. Furthermore, sufficiently improving StochPy performance would enable REMI researchers to feasibly run simulations on local devices like laptops and desktop computers, reducing dependency on the Kamiak cluster and allowing for research to be conducted more efficiently and conveniently, which would further increase research output.

The bare minimum intended outcome for the project is to have StochPy demonstrate a fair amount of speedup. This can be accomplished by optimizing the codebase. We can also create a large amount of speedup comparatively easily by running the disparate simulations on individual processors. StochPy, being a stochastic simulation model, calculates the interactions of individual elements in the simulation. This makes it slower than large scale simulation algorithms, but more accurate at smaller scales. These interactions between elements of the simulation are oftentimes isolated from other interactions between other elements, meaning that these computations are done separately from other groups of computations. These groups are essentially sub-simulations of the larger simulation. These sub-simulations can be done on different cores concurrently, allowing for significant speedup. This is the embarrassingly parallel nature of the algorithm. This could be completed using threads, and it likely would not be that hard to implement. This is the minimum work for us to do.

The parallelization of sub-simulations onto standard non-GPU cores would be the easiest and likely the most effective way to create speedup. However, doing the sub-simulations on GPUs themselves could be promising for creating further speedup. Whether or not having the sub-simulations running on GPUs would be feasible or not is dependent on the architecture of the GPUs, the nature of the algorithm, and the limitations of Python.

Another area where we could see some amount of speedup is with JIT compilation. Using CPython could be beneficial. Likely JIT compilation or another form of compilation would be the only available way to work within the limitations of Python's speed. As rewriting the entire codebase in a language such as Julia or Chapel is unfeasible.

I.4 Client and Stakeholder Identification and Preferences

The primary stakeholder of this project is REMI, which is represented by Dr. Eric Lofgren. At a minimum, REMI's primary need is a speed-up in the stochastic simulation functionality of the StochPy library. This can be achieved through the optimization of the codebase to make use of JIT compilation as well as parallelization of the StochPy library, specifically in regards to its stochastic simulation feature. This parallelized fork should still be capable of operating on existing lab machines and the Kamiak cluster. Stretch goals include adding the ability to employ GPUs to improve simulation speeds, such as through NVIDIA's CUDA framework, as well as the removal of unwanted popup messages accompanying the startup of StochPy.

Student members of the Lofgren Lab, along with other students working under the auspices of REMI, serve as a secondary stakeholder. For them, improving the speed of stochastic simulations is a critical need. An additional preference is improving the speed of StochPy to the point that stochastic simulations run in reasonable timeframes on portable devices like laptops, as opposed to just the Kamiak cluster.

Additionally, government institutions funding REMI's research, such as the National Institutes of Health (NIH), also act as secondary stakeholders for this project. Given the significance of REMI's research, speeding up StochPy would help fulfill NIH mission objectives. Also, it is probable that improving REMI's research capabilities would make it more competitive for future government grant funding.

Other labs and researchers using StochPy serve as indirect stakeholders for our project. Though our team is not in contact with other researchers using StochPy, producing a parallelized fork of the library stands to benefit their research activities as well. Communicating with other research teams may give input as to possible functionalities that a parallel version of StochPy could have. Of course, REMI's goals and requirements are paramount, considering that they are the sponsors of this project.

Finally, those populations which benefit from the research conducted by REMI and other labs using StochPy are tertiary stakeholders, as improvements in the speed of research at such labs could result in lifesaving policy changes and developments. Such populations could include the global community at large in the case of containing and preventing epidemics in their early stages. More specifically, the staff and patients of hospital wards, staff and students of schools, as well as staff and detainees of jails stand to benefit from REMI conducting faster simulations.

II. System Requirements Specification

II.1. Use Cases

It is difficult to provide new specific use cases, as the changes we are to implement are centered around improving existing functionalities in StochPy. Still, general current and anticipated use cases for StochPy are summarized as follows:

REMI researchers frequently need to run stochastic simulations as one of many computational activities needed for their experiments. The StochPy library allows for the sequential execution of these stochastic simulations, albeit inefficiently. As such, to begin with, a major need for REMI researchers is a general speedup in StochPy's baseline sequential execution speed. Even after acceleration with JIT compilation, removal of the welcome message, and other modern Python, REMI researchers should be able to continue using StochPy. This use case can be linked to functional requirements **II.2.1** and **II.2.3**.

Furthemore, building off the prior user story, REMI researchers also want to significantly reduce the time it takes to conduct stochastic simulations, which would allow for StochPy to be more practically used on consumer devices like laptops and desktop computers, along with generally improving research experiment speeds. To achieve this, significant simulation speed increases are beyond what StochPy currently offers. Dr. Lofgren has identified parallelization as an approach to pursue. As such, REMI researchers should be able to perform a parallel stochastic simulation function added to StochPy in the form of DoParallelStochSim. This use case can be linked to functional requirement **II.2.2**.

Dr. Lofgren has expressed an interest in running stochastic simulations on GPUs, taking advantage of their inherently parallel nature to accelerate StochPy simulation speeds even further and make full use of the Kamiak cluster. Consequently, REMI researchers should be able to use the StochPy library to run stochastic simulations on GPUs. This use case can be linked to functional requirement **II.2.4**.

II.2. Functional Requirements

II.2.1. General Speedup using Modern Python Libraries

General Speedup: Our first requirement is to attain a general speedup in the execution of the functions in the StochPy library, which we aim to achieve through the use of modern Python tools for JIT compilation such as Numba, as well as optimizing any code in the library which is inefficiently implemented.

Source: This requirement originated from Dr. Lofgren, who represents REMI, as the primary need of REMI is for the execution of simulations to be completed more quickly. One of the most effective means to facilitate this is to speedup the execution of individual instances in the simulation through the use of JIT compilation. This requirement is tied directly to REMI, as they are the stakeholder which has requested this improvement to the functionality of the library. However, this requirement is also tied to other researchers who could or do make use of the library for their own simulations, as well as policymakers and policy beneficiaries who would benefit from improvements in the speed of simulations conducted by REMI and similar research institutes.

Priority: Level 0 - Essential and required functionality.

II.2.2. Implement Parallel Stochastic Simulation

Parallel Stochastic Simulation: Our second requirement is to parallelize the DoStochSim function of the StochPy library in order to improve the speed with which simulations can be completed. We aim to achieve this using tools such as Dask or Ray meant for easy parallelization of Python code. The function DoStochSim from the StochPy library must be refactored to run simulations in parallel, and this new implementation must become a usable option in the library.

Source: This requirement originated from Dr. Lofgren, as the primary need of REMI is for stochastic simulations to execute faster. The best way to facilitate this is to parallelize the separate instances in the simulations rather than running them sequentially. This requirement is also tied directly to REMI, as they are the stakeholder which requested such an improvement. Once again, stakeholders like other researchers, policymakers, and policy beneficiaries are also involved, due to parallelization imparting benefits similar to that of the first requirement.

Priority: Level 0 - Essential and required functionality.

II.2.3. Remove Welcome Banner from StochPy

Removal of Welcome Banner from StochPy: Our third requirement is the removal of an unwanted welcome message popup built into StochPy which triggers whenever the StochPy tool is used. Finding and removing the code for generating the welcome banner is straightforward, though adding a configuration file so that the appearance of the message can be left up to the discretion of the users is also an option.

Source: This request ties directly back to Dr. Lofgren of REMI specifically, as he personally uses the tool a great deal and wishes to not have to see the message every time he does so. This would also benefit any members of our listed stakeholder groups, such as REMI and other affiliated researchers, who would also like to not be bothered by such a message.

Priority: Level 1 - Desirable functionality.

II.2.4. Run Stochastic Simulations on GPUs

Ability to Run Simulations on GPUs: Our fourth requirement is the potential addition of the ability to run StochPy simulations on GPUs rather than CPUs to make use of their additional computing power. Such a goal will require the research of the architecture of the GPUs in use by REMI as well as the library and language themselves to check for limitations which may prevent such a task from being completed in the first place.

Source: This request originated from Dr. Lofgren, as the increase in computing power available using GPUs would be a significant boon in the speedup of simulations beyond that achieved by previous version of StochPy and other stochastic simulation libraries. This requirement also ties directly back to REMI, as they possess the GPUs which would be utilized for the implementation and are once again the source of the request for this functionality. Other researchers are also involved with this requirement, as some may have access to GPUs and would be interested in a GPU-capable version of StochPy.

Priority: Level 2 - Stretch goal.

II.3. Non-Functional Requirements

Ensure Simulation Accuracy
The parallelization of StochPy should not compromise the accuracy of simulation results and should generally match that of the sequential version of StochPy.

Stable and Reliable
Considering that a large number of simulations may be run at a time or over the course of several days, along with the importance of the simulations being run, our StochPy rewrite should not be prone to crashing, freezing, stalling, or other types of instability.

Efficient
Based on the potential complexity and volume of stochastic simulations, our rewrite should be efficient, with CPU cores, main memory, and other computational assets being utilized to their fullest without slowdown originating from inefficient programming practices.

Maintainable
Our rewrite of StochPy should be maintainable, in that it should be well-commented, well-documented, adhere to object-oriented principles, and follow good programming practices. This will aid student programmers and REMI members that may need to modify parallelized StochPy in the future.

Fast
At a minimum, our parallelized fork of StochPy should run stochastic simulations faster than the current sequential version.

Continue to Accept .psc Files as Input
Currently, simulation setup and execution information is stored on preexisting .psc files. Our StochPy rewrite should continue to accept these files for simulation information.

Easy to Use
Ultimately, most users of the stochastic simulation features of StochPy do not have a technical background. There should be no added difficulty in using parallelized StochPy over sequential StochPy.

Compatible with the Kamiak Cluster and other WSU Servers
Since our project scope is largely limited to backend changes, our forked version of StochPy should continue to run on the Kamiak cluster and other capable WSU devices with minimal necessary disruptions.

Written in Python
Given that StochPy is written in Python, our rewrite should still be written in Python.

III. System Evolution

Hardware evolution
Changes to the hardware that will be used is unlikely. The REMI lab will be using the project primarily on the Kamiak cluster and personal machines, and Python is a language with which all the current users are already familiar with and are capable of implementing StochPy on said cluster and machines. We do not expect the Kamiak cluster to drastically alter its hardware, either. Since Python is a language capable of being run with little setup on all computer devices, changes in individual personal machine choices should not impact the ability to run StochPy and our changes to it. Improvements in processor and GPU performance would likely not change the way the project would be designed. The only concern would be if we created some sort of CuPy application and the GPUs were replaced with non-NVIDIA GPUs, which is unlikely.

Changing user needs
The current users of the software, the REMI lab, are experienced scientists who are adept at managing the workflow for their stochastic simulations. Their needs are simply to be able to create .psc files and run simulations based on them. As they are a lab focused on simulations for relatively small population sizes using a simulation method similarly specialized for such populations, their needs are not expected to change for the foreseeable future in regards to the necessary utility of this tool, unless their research focus shifts drastically.

Another user base that may emerge is doctors and healthcare professionals who are not expert researchers. Another capstone team is working on creating a frontend application so that these users can run their own simulations. They will likely not have access to the Kamiak cluster or other advanced computing resources, so any speedup they experience will likely not allow for the use of GPUs. This lends further emphasis to the importance of the general speedup and parallelization aspects of the project, as they will provide the most benefit to the greatest number of users.

Programming language limitations
Python is a commonly used language among scientists and researchers. It then makes sense that we are using Python. However, Python is limited in terms of high performance computing. We are in the process of evaluating whether these limitations can be overcome with different compilers and libraries as it relates to StochPy and stochastic simulations. Even if using Python makes the project run slower than it hypothetically could, we do not foresee switching to another

language, as this would mean rewriting the entire codebase in addition to the researchers likely not knowing the language used instead of Python.

IV. Project Solution Approach

I. Introduction

The Resistance Epidemiology Modeling Initiative (REMI) conducts simulations modeling epidemics, particularly in smaller populations and focused on the beginning and end of epidemics. There are not many highly developed tools available for the stochastic simulations that REMI does. Therefore, they use a custom built library, the StochPy library. The shortcoming of StochPy is that the stochastic simulation algorithm is slow, especially when trying to perform large computations. Our project problem is then to create an optimized and parallelized fork of the StochPy library.

II. System Overview

The purpose of this document is to create a more in-depth plan for our project. That is, we will go over the design of our system. We will look at a general overview of our proposed system, the architecture of our software, the way our data will be stored, and the user interface.

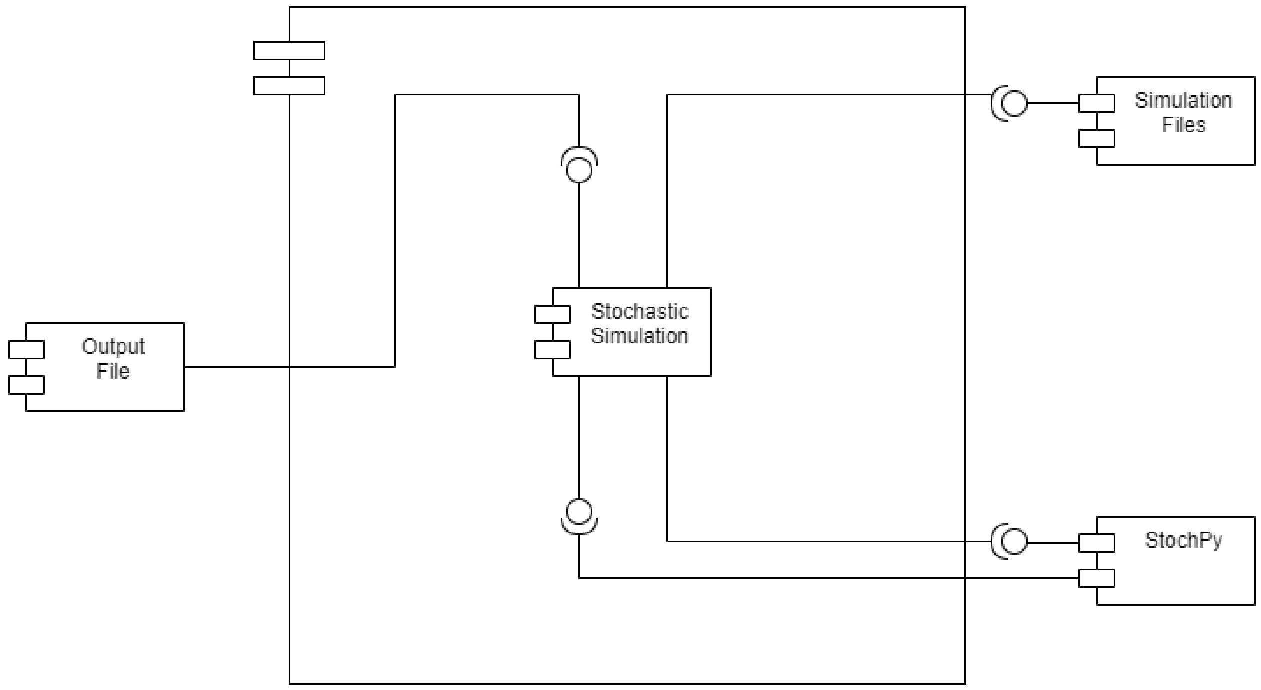
The primary functionality of the product is to upgrade the StochPy library to include a parallelized version of a function which it already has called DoStochSim, as well as implement a general speedup in the execution time of the StochPy library by refactoring code integral to stochastic simulation to make use of JIT compilation and other modern Python features. If time permits, we additionally aim to improve the StochPy library to be able to make use of the computational power of GPUs, as this functionality would further aid in achieving an overall increase in simulation speed which is sought by the client. A much more design focused additional goal of the product is to remove an unwanted pop up from the startup sequence of StochPy.

III. Architecture Design

III.1. Overview

Due to the nature of our project, it is difficult to generate an extensive architecture for the software we are creating, as we are adding new functionality to an existing library rather than building software from scratch. While a proper overview of the architecture for the library itself could be provided, to do so in the context of this report may be considered plagiarism and contrasts with the defined purpose for this section. As such, an extensive architectural overview has been omitted, as our work is much more limited in scope than what would necessitate an overview of the entire library, and to include one with that being the case may be viewed as dishonest and disingenuous. For this reason, we cannot do a decomposition of the systems we are working beyond saying that our parallel implementation of DoStochSim will have the same provided and required services as the original DoStochSim. However, expansions in the scope or changes in the nature of this project may alter our architectural design in the future. Regardless, we do not anticipate REMI or Dr. Lofgren modifying their requirements in the immediate future.

As it stands for our ad-hoc architectural design, the Stochastic Simulation subsystem is the primary one being developed at this point. It is anticipated to receive inputs from the Simulation Files subsystem, which consists of .psc files and any other other functionality that serve to provide parameters for stochastic simulation execution. The Output File subsystem represents the user-accessible “front-end” results generated by the Stochastic Simulation subsystem. Additionally, StochPy itself is represented as a greater subsystem, as to denote that our stochastic simulation improvement efforts are still but a subset of the greater library, and that our changes contribute to the greater StochPy codebase. Potential future changes in scope may alter this reality.



III.2. Subsystem Decomposition

III.2.1 Stochastic Simulation

a) Description

This subsystem encapsulates the implementation of the sped-up sequential, parallelized, and GPU-accelerated stochastic simulation functions.

b) Concepts and Algorithms Generated

Our current efforts are centered around understanding StochPy’s stochastic simulation algorithm implementation and analyzing possible methods for speedup and eventual parallelization. Dr. Lofgren has offered potential approaches for parallelization, which we are currently investigating.

c) Interface Description

Services Provided

1. Service Name*:* Improved DoStochSim
   Service Provided To: StochPy
   Description: Improved DoStochSim will implement an improved version of the current DoStochSim function by leveraging optimizations offered by modern Python libraries to accelerate sequential execution. Input values will generally consist of simulation parameters provided by .psc simulation files and any necessary auxiliary StochPy library functions. Output values will be stored in output files, and the function itself contributes to the rest of the StochPy library.

2. Service Name*:* Parallelized DoStochSim
   Service Provided To: StochPy
   Description: Parallelized DoStochSim will implement a version of DoStochSim that uses a parallel algorithm to perform stochastic simulations. Input values will generally consist of simulation parameters provided by .psc simulation files and any necessary auxiliary StochPy library functions. Output values will be stored in output files, and the function itself contributes to the rest of the StochPy library.

3. Service Name*:* GPU-Accelerated DoStochSim
   Service Provided To: StochPy
   Description: GPU-Accelerated DoStochSim will implement a version of DoStochSim that uses a parallel algorithm capable of running on GPUs. Input values will generally consist of simulation parameters provided by .psc simulation files and any necessary auxiliary StochPy library functions. Output values will be stored in output files, and the function itself contributes to the rest of the StochPy library.

Services Required

1. Service Name: Provide .psc Simulation Information
   Service Provided By: Simulation Files

2. Service Name: Provide Access to Rest of StochPy Codebase
   Service Provided By: StochPy

IV. Data Design

The initial specifications for each simulation are stored in .psc files (pronounced pisces). This data from these files is then used to do the simulation which stores output in .csv files. The StochPy library itself makes use of arrays, unassigned integers, and floats from the NumPy library for data manipulation during the simulations.

V. User Interface Design

Installation

To install StochPy, the user will need to run this command in their Python environment:

```
pip install scipy matplotlib python-libsbml jedi==0.17.2 ipython stochpy
```

For Linux/MacOS instead you can simply run this command in the directory of the cloned repository:

```
sudo python setup.py install
```

Use

The user interface will simply be whatever terminal the program is being run from. To run the application, the user will create a .psc file, storing the specifics of the simulation to be run. The user will then use the following commands to do necessary operations.
Run ipython and import stochpy:

```
import stochpy

smod = stochpy.SSA()
```

Please see Appendix A.1 for more StochPy commands.

Glossary

CUDA - Proprietary parallel programming platform created by NVIDIA.

CuPy - Open-source Python library for GPU-accelerated code. Interfaces well with NumPy and SciPy.

Graphics processing unit (GPU) - Specialized processing units that can do many small numerical operations at once. They are primarily useful for graphics processing, but can be used in some parallel programming applications.

High-performance computing (HPC) - Encompasses the subfield of computing concerned with performing advanced calculations at extremely fast speeds, especially on computer clusters.

Kamiak cluster - A high-performance computing cluster owned and operated by WSU. Used by REMI to conduct stochastic simulations and other resource-intensive computational activities.

Lofgren Lab - A WSU lab under the leadership of Dr. Eric Lofgren focused on epidemiology and public health.

Parallelization - A programming paradigm where a task is subdivided across processing units so that its subtasks can be run concurrently.

R_0 - For a communicable disease, represents how many people an infected person is expected to infect.

Resistance Epidemiology Modeling Initiative (REMI) - A research initiative established at WSU. Seeks to apply advanced computational methods to problems in public health, such as antimicrobial resistance, healthcare-associated infections, and COVID-19.

Stochastic simulation - A simulation of a system where individual variables change randomly with individual properties.

Gillespie Algorithm - A stochastic simulation algorithm. Used in the StochPy library.

Just-in-time compilation (JIT) - A compilation method that compiles the code exactly at runtime. A combination of ahead-of-time compilation and interpretation.

Appendix

A.1. Sample StochPy Commands

Basic Simulation with the Direct method

```
smod.DoStochSim(IsTrackPropensities=True)

smod.data_stochsim.simulation_endtime

smod.data_stochsim.simulation_timesteps

smod.GetWaitingtimes()

smod.PrintWaitingtimesMeans()
```

Do some Plotting

```
smod.PlotSpeciesTimeSeries()

smod.PlotWaitingtimesDistributions()

smod.PlotPropensitiesTimeSeries()
```

Write data to a text file

```
smod.Export2File()

smod.Export2File(analysis='distribution')

smod.Export2File(analysis='distribution',datatype='species')

smod.Export2File(analysis='mean',datatype='species')

smod.Export2File(analysis='std',datatype='species')

smod.Export2File(analysis='autocorrelation',datatype='species')
```

Show the means from the data of 3-th trajectory

```
smod.DoStochSim(trajectories=3) # multiple trajectories

smod.data_stochsim.simulation_trajectory

smod.PrintSpeciesMeans()

smod.PrintSpeciesStandardDeviations()
```

Switch to data from trajectory 1 and show the means of each species

```
smod.GetTrajectoryData(1)
smod.PrintSpeciesMeans()

smod.PrintSpeciesStandardDeviations()
```

Do one long simulation

```
smod.DoStochSim(trajectories=1,end=1000000,mode='steps')

smod.PrintSpeciesMeans()

smod.PrintSpeciesStandardDeviations()
```

Plot the PDF for different bin sizes

```
smod.PlotSpeciesDistributions()

smod.PlotSpeciesDistributions(bin_size=5)  # larger bin size

smod.PlotSpeciesDistributions(bin_size=10) # again a larger bin size

smod.Export2File(analysis='distribution',datatype='species')
```

Usage of the Reload Function: Ksyn = 20, kdeg = 0.2

```
smod.ChangeParameter('Ksyn',20.0)

smod.ChangeParameter('Kdeg',0.2)

smod.DoStochSim()

smod.PrintSpeciesMeans()   # should be ~Ksyn/Kdeg
```

Use another model to show the Interpolation features

```
smod.Model('dsmts-001-01.xml.psc')

smod.DoStochSim(trajectories=1000,end=50,mode='time')

smod.GetRegularGrid(npoints=51)

smod.PlotAverageSpeciesTimeSeries()

smod.PrintAverageSpeciesTimeSeries()

smod.Export2File(datatype='species',analysis='timeseries',IsAverage=True)
```

Test each method for different models:

```
smod.Model('Autoreg.psc')

smod.DoStochSim(trajectories=1,end=1000,mode='steps')

smod.Method('NextReactionMethod')
smod.DoStochSim(trajectories=1,end=1000,mode='steps')

smod.data_stochsim.species

smod.PlotWaitingtimesDistributions()

smod.Method('FirstReactionMethod')

smod.DoStochSim(trajectories=1,end=1000,mode='steps')

smod.Method('TauLeaping')

smod.DoStochSim(trajectories=1,end=1000,mode='steps')

smod.Model('DecayingDimerizing.psc')

smod.DoStochSim(method = 'Direct',trajectories=1,end=50,mode='time')

smod.DoStochSim(method = 'NextReactionMethod',trajectories=1,end=50,mode='time')

smod.DoStochSim(method = 'FirstReactionMethod',trajectories=1,end=50,mode='time')

smod.PlotWaitingtimesDistributions()

smod.DoStochSim(method = 'TauLeaping',trajectories=1,end=50,mode='time',epsilon=0.03)
# Should outperform all other implementations

smod.PlotSpeciesTimeSeries()

#smod.PlotWaitingtimesDistributions()   # Should give an error

smod.Model('chain500.psc')

smod.DoStochSim(method = 'Direct',trajectories=1,end=10000,mode='steps')

smod.DoStochSim(method =
'NextReactionMethod',trajectories=1,end=10000,mode='steps') # should outperform the
direct method and all other implementations
```

Use the Next Reaction Method to test a model with a time event

```
smod.Model('dsmts-003-03.xml.psc')

smod.DoStochSim(method = 'NextReactionMethod')

smod.DoTestsuite()
```

Use the First Reaction method to test a model with a concentration event

```
smod.Model('dsmts-003-04.xml.psc')

smod.DoStochSim(method = 'FirstReactionMethod')

smod.DoTestsuite()
```

Volume Models

```
smod.Model('dsmts-001-11.xml.psc')

smod.DoStochSim(method = 'Direct',trajectories=1000,end=50,mode ='time')

smod.PrintAverageSpeciesTimeSeries()
```

References

[1] E. T. Lofgren, "Pools versus Queues: The Variable Dynamics of Stochastic 'Steady States,'" *PLOS ONE*, vol. 10, no. 6, p. e0130574, Jun. 2015, doi: 10.1371/journal.pone.0130574.

[2] A. M. Ridwan, A. Krishnan, and P. Dhar, "A Parallel Implementation of Gillespie's Direct Method," in *Computational Science - ICCS 2004*, Berlin, Heidelberg, 2004, pp. 284–291. doi: 10.1007/978-3-540-24687-9_36.

[3] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J. Phys. Chem.*, vol. 81, no. 25, pp. 2340–2361, Dec. 1977, doi: 10.1021/j100540a008.

[4] T. R. Maarleveld, B. G. Olivier, and F. J. Bruggeman, "StochPy: A Comprehensive, User-Friendly Tool for Simulating Stochastic Biological Processes," *PLOS ONE*, vol. 8, no. 11, p. e79345, Nov. 2013, doi: 10.1371/journal.pone.0079345.

[5] D. Kishore and S. Chandrasekaran, "cayenne : Python package for stochastic simulations," *Read the Docs*, 2020. [Online]. Available: https://cayenne.readthedocs.io/en/latest/#cayenne-python-package-for-stochastic-simulations. [Accessed: Sep. 12, 2022].