Software Documentation

iGEM SynShine

Aim

The software component is designed to complement the hardware and the modeling sections of the project while linking it with the wet-lab component. It provides a framework which aims to test the experimental data against the mathematical model obtained and helps to refine and choose parameters which make the model more robust for the given bacterial system. It can then be extended to suit the needs of any such system provided that the differential equations governing the mathematical model isn't drastically altered. Thus, it acts as a supplementary framework for the entire project.

1 Modeling Software

The modeling software consists of the following scripts which are used to initialize bacterial conditions, run simulations, attain data points across the simulation trajectory and plotting of data. The modelling software works best on a Linux terminal, or a WSL2 terminal on a Windows machine*. * Each individual component of the modelling script works independently irrespective of operating system. The software is hassle-free on a Linux machine, post-installation of numpy and matplotlib. On Windows, WSL2 is needed to run the bash script, and the graphing script requires an additional Xorg server client to work on Windows if the bash script is used.

1.1 Initialization Script

A bash script which also acts as the driver for the program works via the following way: a parameters file contains user inputs for setting the initial parameters of the system, for each of the interested variables and, with the required range and precision. A Python3 script parses the input and generates the required input files for the binary executable, thus generating all the required inputs for simulation. The driver code also proceeds to run the simulation with the co-culture model.

1.2 Simulation of the bacterial co-culture model

The binary executable is used to simulate the bacterial system over time. This is done by intergrating the differential equations that govern the system, using a Runge-Kutta 1 integrator a.k.a, Euler Integrator. The error introduced during consecutive timesteps is of the $O(\Delta t^2)$. The differential equations which model the bacterial system can be found on the modeling part of the Wiki. Euler's method for solving first order, first degree differential equations is a well known simplistic system which can be used to approximate the solution to a system of differential equations. In this method, we assume a small time-step within which we approximate the variable(s) to change by the amount equal to the first degree derivative multiplied with the time-step. In case of higher Runge-Kutta methods, we employ similar techniques to change the variables using higher degree derivatives. Given the first degree nature of the differential equations employed to model the bacterial system, Euler's method was used to model the change in variables.

1.3 Graph Plotting

A Python3 script is used to generate graphs for the data obtained from each of the simulations. Additional packages needed: numpy, matplotlib.

2 InFORM: Interface For OptoMatic Response Management

Our hardware's interface to our software us through an Arduino, which collects the various experimental parametres from the sensors. Here is where InFORM comes in:its aim is to convert responses produced by the Arduino module into a well organised format, so that it can be used for data visualisation. This allows the module to communicate efficiently with the user and provides a way to obtain the values of the various experimental parameters.

3 References

- Ascher, Uri M.; Petzold, Linda R. (1998). Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. Philadelphia: Society for Industrial and Applied Mathematics. ISBN 978-0-89871-412-8.
- Owen, A.B. (1992). "Orthogonal arrays for computer experiments, integration and visualization". Statistica Sinica. 2: 439–452
- Salsa, Sandro. Partial Differential Equations in Action: from Modelling to Theory. Springer, 2016.