1 Progress

- We found a way to implement the whole process without the use of the *event-based* method in `Python`.
- The python version is written by dividing the time step into a homogeneous and an in-homogeneous part. the homogeneous is obvious integration and the in-homogeneous section is almost like event-based.
- I started implementing the system in `C++`. In this language I can make the implementation *event-based*. I created a class `Neuron` as a new level of abstraction for the system. The system will then contain Neurons and using the functions that will be written for `Neuron` and `Nervous System`, I will implement the system.
- Completely implemented the dynamics of the neural network in C++. The dynamics are event-based as they are a result of the interactions between the network and the neurons.
- Added file stream and output of data to `.csv` files in C++.
  Also added the `cleandat` .PHONY to the `Makefile` to clean all the data if used.
- Added the calculation and report of mean area intersection to `.csv` files. Now I'm running the code for 10^5 seconds.
  Ok. The simulation seems to be working well but the duration was apparently a bit short and the plots showed that the system didn't reach stability.
- I optimized the `calc_mutual_area()` method and will now run the program for a duration of $5 \times 10^5 s$ and wait for the results.

2 Problems

- The `python` version doesn't work. I assume there is something wrong with the constants defined in the system; but still, anything is possible.
- The plots for $g\tau A_i$ don't come out as what we expect and for the first run that took about 160000 seconds, the values flew into a small value of order $\mathcal{O}(10^{-2})$ and oscillated about those values for the rest of the time.

- I'm thinking if I implement the system in the *event-based* method, I might be able to have a better view of what happens inside the system. So I start with `C++`

3 Ideas