NEUROSCIENCE GATEWAY

A Portal for Computational Neuroscience

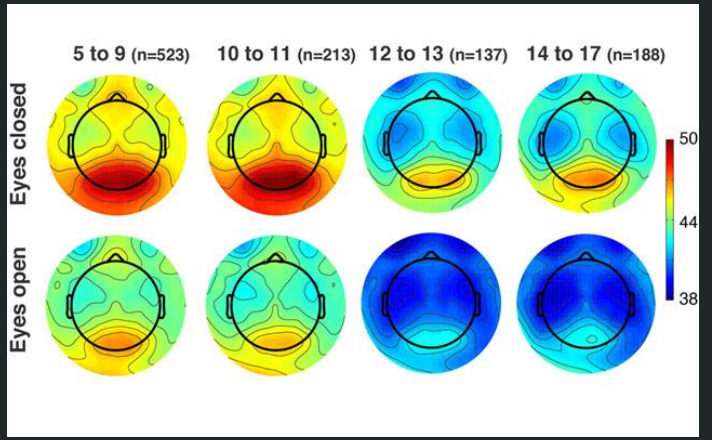
Supercomputing in Neuroscience

By: Raghav Alliyalamangalam (Westview High School), Mia Gover (The Bishop's School), Nika Kamachee (Mira Costa High School), Justin Zhang (Westview High School)

Mentors: Amitava Majumdar (UCSD), Dung Truong (UCSD), Ted Carnevale (Yale University)

Introduction

Since the mid-1960's, supercomputing has grown and developed into a formidable tool used by researchers in order to take theory to the next technological level. These extensive computational power and parallel processing capabilities have allowed for a greater understanding of the brain's intricacies – all the way down to the microscopic activity of individual neurons and the firing dynamics of neural networks as a whole. Simply using supercomputers, we can operate multiple computational processes at once, record the electrical activity of the brain, and construct lifelike digital replicas of actual neuronal behavior.



1. Parallel Computing

Expanse Supercomputer

- Where we ran our code for matrix multiplication
- Vi editor - Uses the Unix operating system to allow users to edit their program
- Different Types of basic commands
  - **Sbatch** - This will submit the job to the Expanse machine's queue where most probably it will wait a bit in the queue
  - **SLURM** - This is a script that can be used to run the compiled code on the Expanse supercomputer
  - **ls -ltr -** This will list (in chronological order) all the files you have in your directory and the last file will be the output file.
  - **Cat** - reads each File parameter in sequence and writes it to standard output.

Notice from the `SLURM_matrixmultiply_MPI2` file that it is set to run on 2 cores for the parallel MPI matrix multiply code i.e. it has the following commands in it:

```
#SBATCH --ntasks-per-node=2

srun --mpi=pmi2 -n 2 ./matrixmultiply_MPI
```

Then you will submit the job of running the MPI parallel matrix multiply code to the Expanse supercomputer.

First submit the MPI matrix multiply parallel code to the Expanse machine by submitting the SLURM script to the Expanse machine by issuing:

```
sbatch SLURM_matrixmultiply_MPI2
```

This will submit the job to the Expanse machine's queue where most probably it will wait a bit in the queue. Then you can issue the following command to see how your job is waiting in the queue.

```
squeue -a | grep "your_userID"
```

When the job is done, that same command will not show anything i.e. you don't have a job that is either waiting or running, meaning it has completed.

Then you can do the following command in your directory:

```
ls -ltr
```

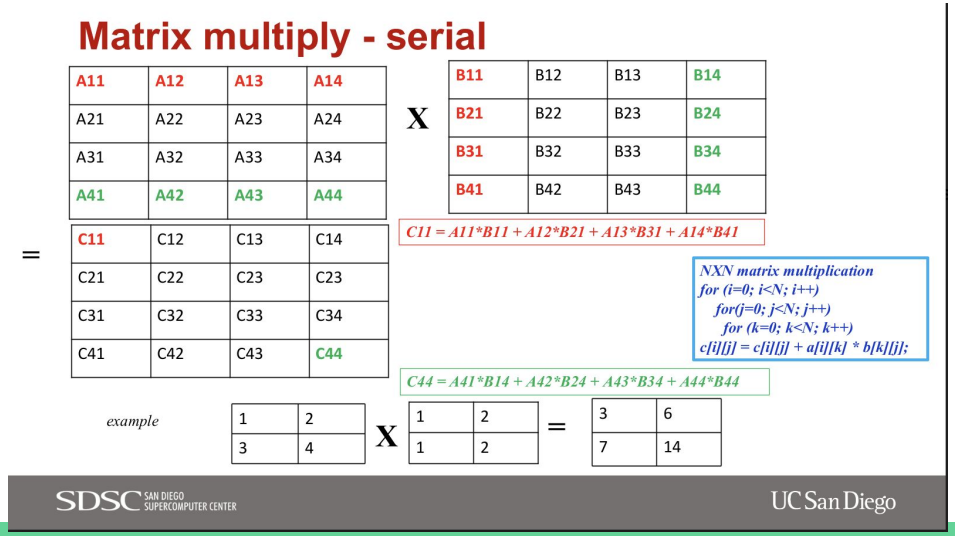
which will list (in chronological order) all the files you have in your directory and the last file will be the output file. Do a 'cat' on that file to see the output and the time it took to run the MPI matrix multiple code on 2 cores.

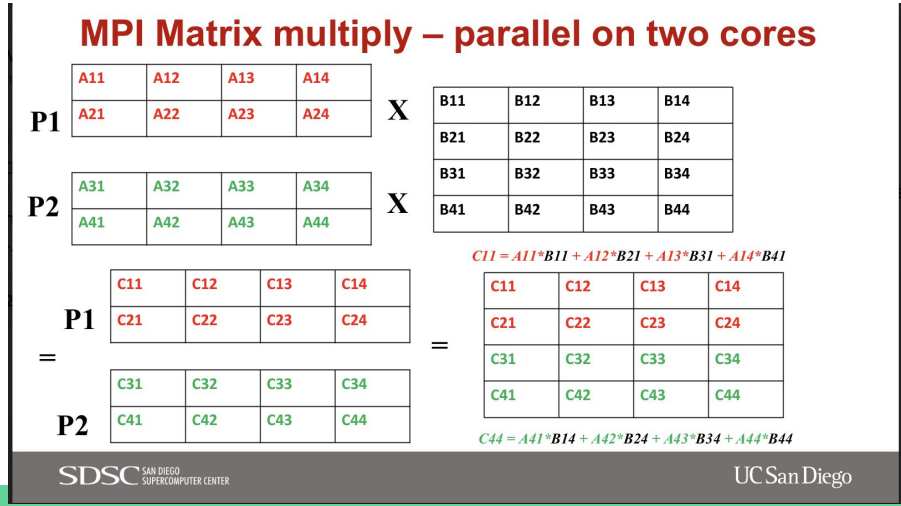
Repeat the above steps to create SLURM files for 4, 8 and 16 cores' MPI runs and submit them and when the jobs are done look at the output timings.

Plot the timings in a two dimensional graph where the X axis is "number of cores (MPI tasks)" and Y axis is "time in sec".

What is Matrix Multiplication?

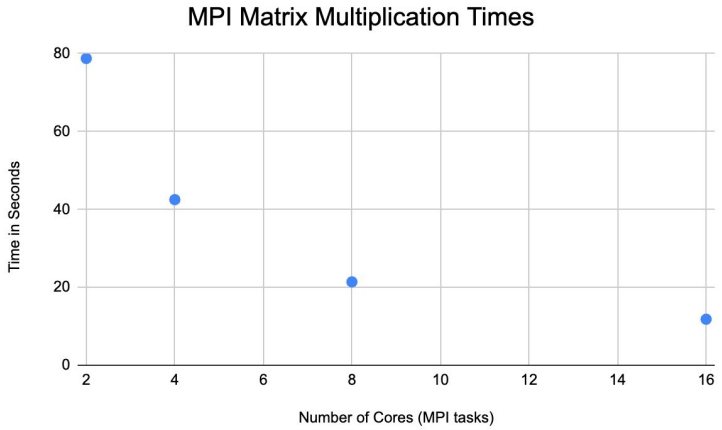
- A type of binary operation
- The product of two matrices is the dot product of the two matrices
- Code for a matrix multiplier can be written for both MPI and OpenMP





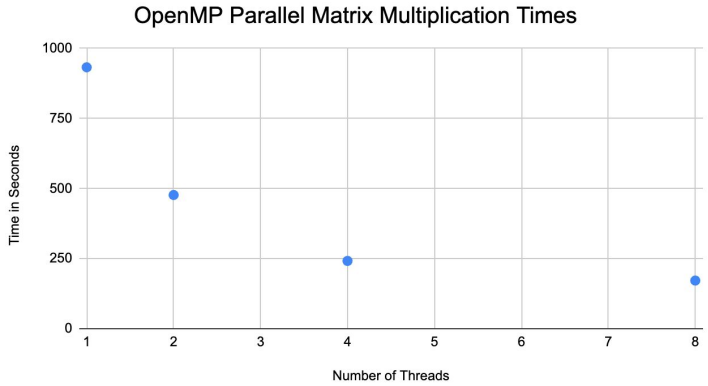
MPI Matrix Multiplication

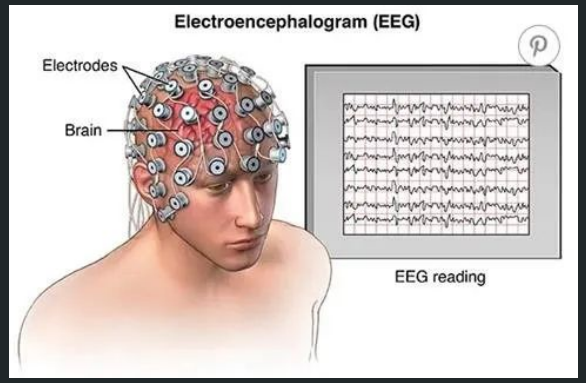
- MPI is a way to program on distributed memory devices
- MPI allows you to run your program on any number of processes
- Ran a matrix multiplication model on Expanse supercomputer using MPI to see the speedup of the number of cores over time.
- Ran the model on 2, 4, 8, and 16 cores
- More reliable and flexible than OpenMP
- Most used in High-performance computing



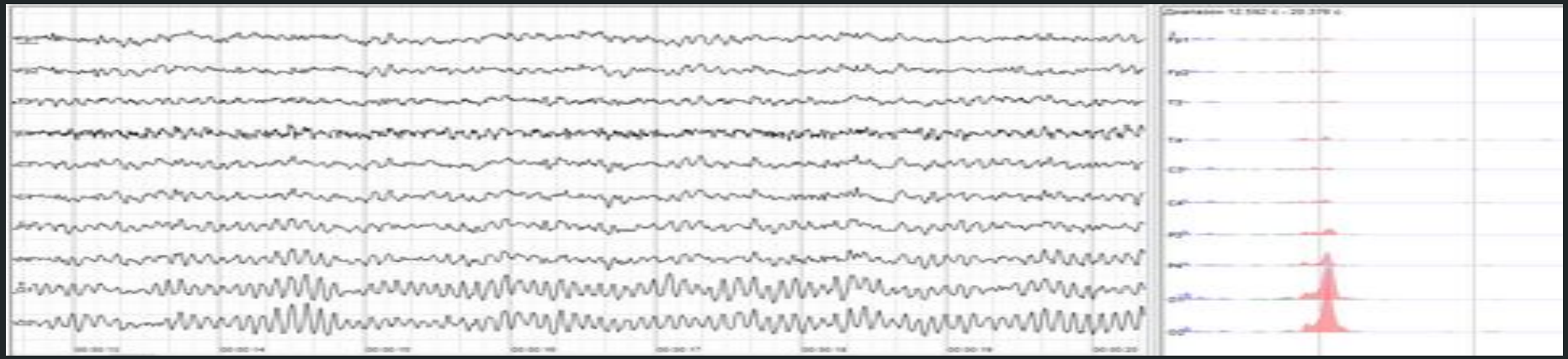
OpenMP Matrix Multiplication

- OpenMP is a way to program on shared memory devices
- The number of threads is constrained by the number of cores or processors available
- Processes exchange data by passing messages to each other.
- Ran a matrix multiplication program on OpenMP to see the speedup of number of threads over time
- Ran the model with 1, 2, 4, and 8 threads
- Benefits includes the simplicity

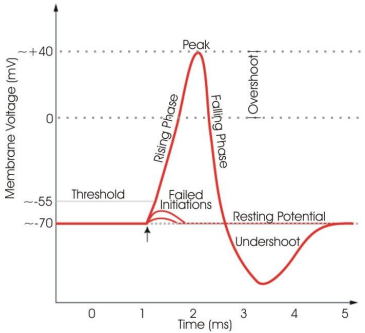


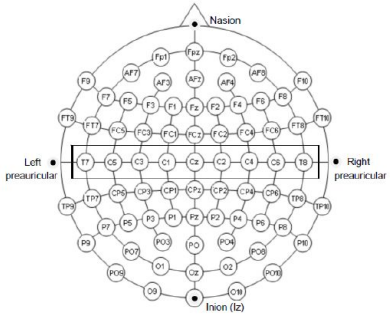


2. EEG Data + Analysis

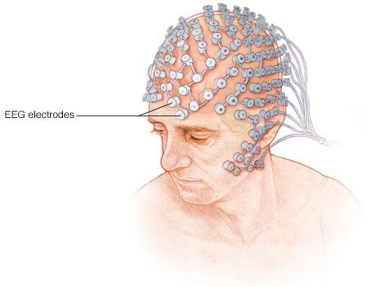


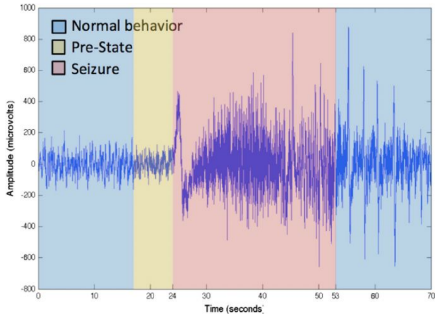
EEG Overview



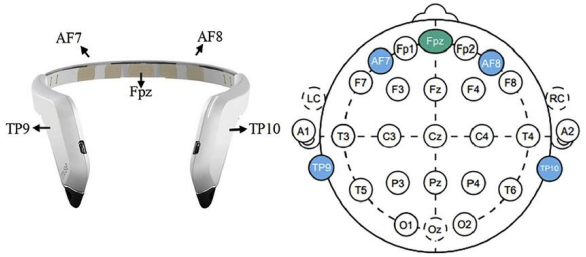


- Neurons firing (from electrical or chemical synapses) generates electricity (so measuring the electrical activity at the scalp can give an idea of what's going on within the brain)
  - Resting state voltage is -70 mV, but can spike up to 40 mV when firing
  - Creates electrical activity in the brain
- EEG(Electroencephalogram) is a method of measuring brain activity
- Electrodes on the scalp continuously measure the voltage at certain points of the scalp
  - Used for sleep monitoring and seizure detection
  - EEG data are linked to phenomenon through experimental data





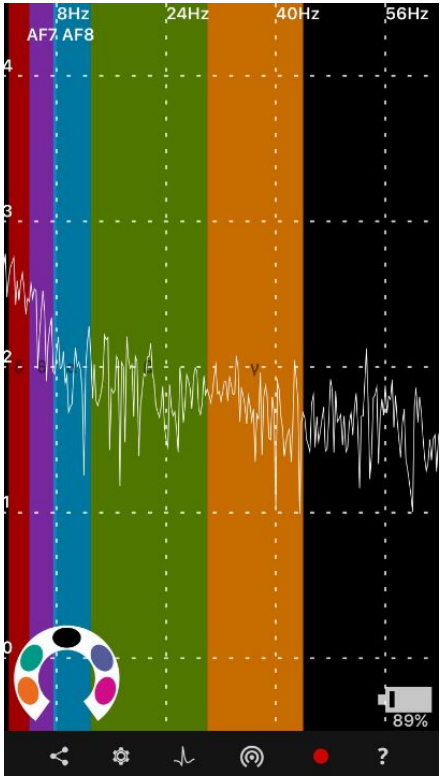
Mind Monitor and EEGLAB



- Muse headset: a device used to measure electrical activity at 4 electrodes in the I 10-20 system
  - Typically used for meditation: not enough channels for actual experiments
- EEGLAB: software used to process EEG data
  - Import the data points for the channels
  - EEGLAB can process the data by filtering the bad portions of data, running ICA decomposition, and plotting graphs
- Experiment
  - Used mind monitor to measure brain activity with eyes open and eyes closed
  - Imported the data into EEGLAB, processed, and analyzed the data

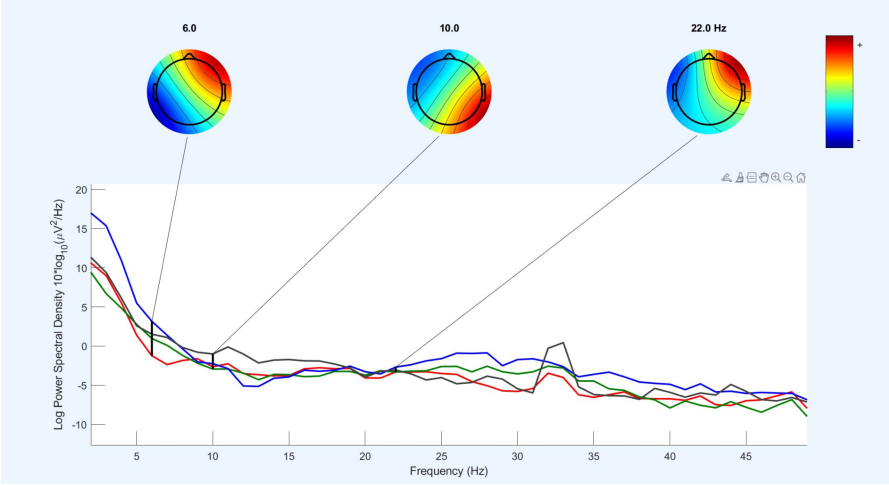
Methods

- Used Muse headset to record data for one minute with eyes open and eyes closed (data was collected by the mind monitor app on my phone)
- Imported the two datasets into EEGLAB
- Processing
  1. Remove DC Offset
  2. Reference the data
  3. High pass filter: remove line noise
  4. Remove artifacts (bad portions of data, such as blink or other muscle movements)
- Plot the power spectra

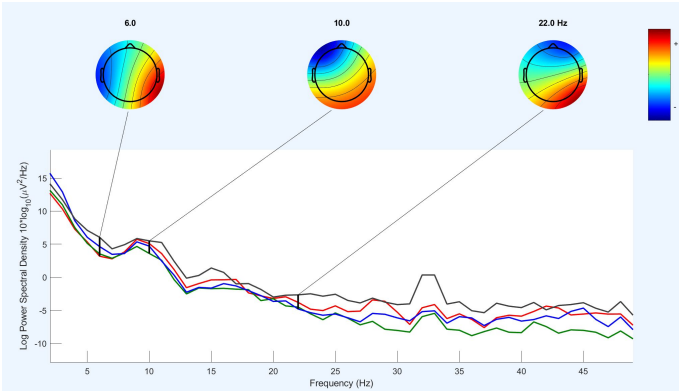


Results

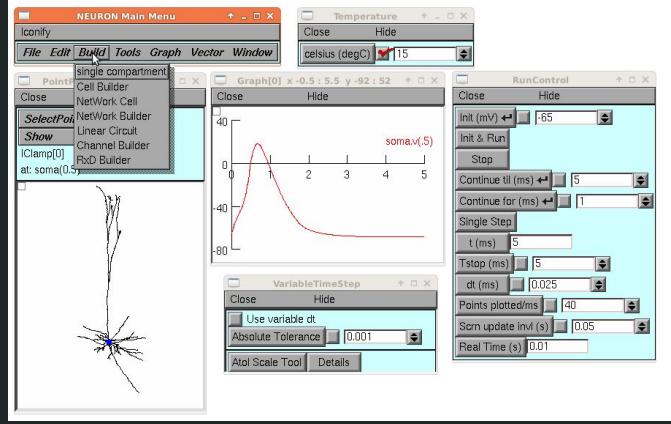
- Eyes closed power spectra has a stronger alpha (8-12hz) activity
    - Consistent with prior experimental results: alpha activity is associated with sleepiness
- The gamma activity (30-35hz) is rather unusual: present in both datasets, but not much prior experimental evidence indicates that strong activity should be there
    - Likely due to inaccuracies in the 4 electrode Muse headset
    - Potentially open for further exploration



Eyes Open



Eyes Closed



3. Modeling with NEURON

Computational Modeling + NEURON

Uses:

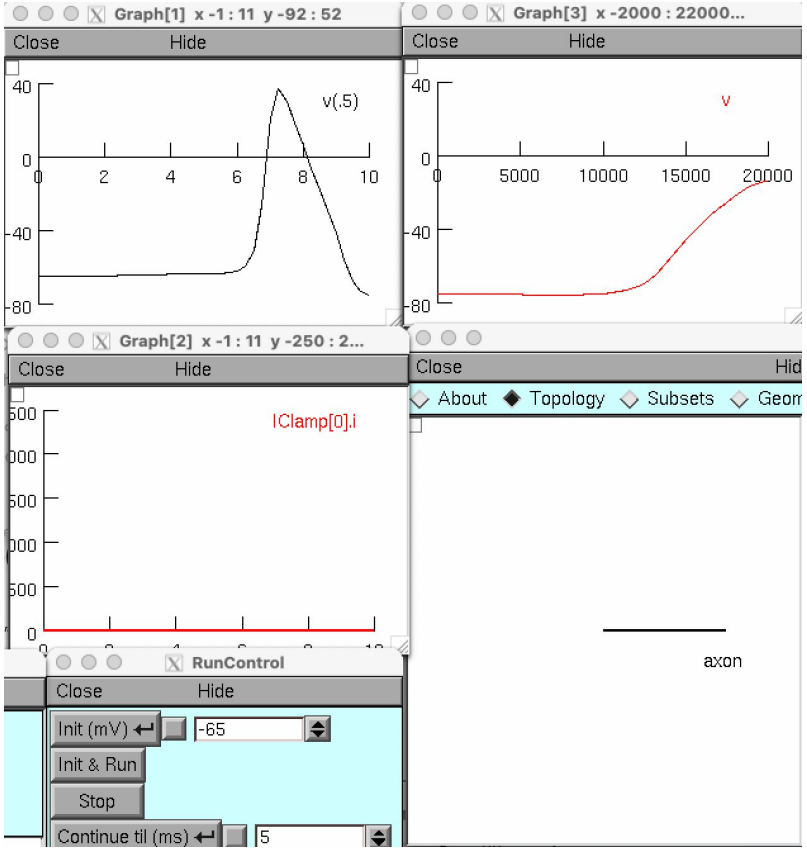
- Predict experimental results
- Test the plausibility of a theory
- Improve understanding of certain phenomena

NEURON:

- Models of neurons + neural circuits
- Linked to experimental observations, complex anatomical properties, electrical/chemical signaling
- Created our own models and looked at + tested other people’s from papers

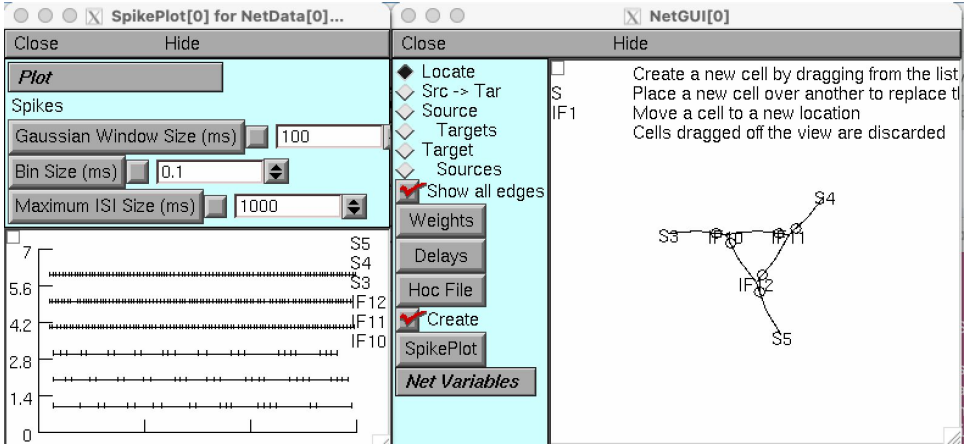
Squid Axon

- Injected a current into the axon, opening ion channels
- Positive ions flowing through these channels create a spike in the membrane potential (difference in charge between inside and outside of cell)



Friesen Model

- IF (integrate and fire) cells inhibit their neighbors
- This causes the pattern seen in the graph, where each cell takes turns spiking
- Changed parameters, such as interval and noise, to see how they affected the spiking patterns
    - For example, noise changed the randomness of the spike pattern, and interval changed how far apart each group was



Cortical Column Model

- Parallel simulations of large network models on HPC (high performance computing) resources
  - In particular, the source code for a thalamocortical column model described by Traub et al. 2005
  - Simulations were run on a Linux cluster so that the computation could be broken down into pieces that each ran on their own CPUs (parallel computing allowed for reasonable run times)
- Explored how run time varied with the number of processors used to execute simulation
  - Run time reflected how quickly the CPUs were able share data (such as firing times of all axons) and maintain the synchrony amongst themselves (even though some CPUs handled more work than others), all while running independently