**Stochastic PIM Code Manual**

**How to download and run**

1. Download the current version from Github
   1. git clone <https://github.com/hplp/Stochastic-PIM.git>
2. Compile the code using the make file, there are currently two configurations
   1. make convolution: only performs the convolution function
   2. make all: all the steps of the machine learning
3. Run the program and provide the following files in order: weight file (csv or txt), input file (csv or txt). The output can be stored to a csv file by adding an extra input with “> FILENAME”
   1. Example of printing to terminal: ./a.out weights.csv inputs.csv
   2. Example of csv formatted output: ./a.out weights.csv inptus.csv > OUTPUT.csv
4. If trying to run individual files, compile using either g++ or gcc, the version of clang++ on boldrock does not work with the code as of May 2022

**Parser Information**

If an excel file is used to write weights manually, **make sure it is saved exactly as the following format: CSV (Comma delimited)**. Any other type of CSV format from excel will break the parser. An error will be outputted if the number of rows for the weight file is different than the rows in the input file.

Graphical user interface, text, application

Description automatically generated

Only the filenames are needed for the parser, the dimension will be automatically detected by the program. There is an issue for very large inputs with the parser, but this is not a problem until there are several million values to be read.

**Code Documentation**

**ASDM.h/cpp & ASC\_adder.h/cpp**

Direct conversion from the MATLAB functions

**SchmittTrigger.h/cpp**

A copy of MATLAB’s Schmitt trigger function

**DutyCycle.h/cpp**

Includes 2 functions, one to find the duty cycle of all pulses in a stream and one for the average.

* dutyCycle
  + Returns a vector, works by looking for the rising and falling edges of a stream
    - 1. A rising edge is found
    - 2. The falling edge after the rising edge is found
    - 3. The rising edge after the falling edge is found
    - 4. A ratio is calculated using the three values and the process repeats until it gets to the end of the pulse stream
* averageDutyCycle
  + Returns a float that is the average value of the vector from the dutyCycle function
* findRisingEdge
  + Helper function to find a rising edge
* findNextFallingEdge
  + Helper function to find a falling edge

**Parser.h/cpp**

Two functions, one for loading weight values and the other for loading px values

* readWeights
  + Given a csv or a txt file, it converts it to a 2D float vector that is indexed by (row, column) order.
* readInputs
  + Converts a csv or txt file of the original input into a vector of floats to be used

**Cell.h/cpp**

Very basic cell object to be used in the convolution function, only has read and write functionality at the moment

* Write
  + Converts a weight to a conductance value based on max and min conductance
* Read
  + Returns the conductance of the cell

**Activation.h/cpp**

Activation function of the system

* Activation
  + Checks the duty cycle of the incoming stream and then returns the appropriate value as needed

**StochasticPIM.h/cpp**

The final object made with all of the parts needed. Note: This is only an object and needs a main file declared elsewhere to compile

* Max
  + Helper function for the convolution stage
* Initialize
  + Sets all of the environmental variables within the object
  + Creates an array of cells with the same dimension as the input weight file from the parser
  + Performs the write operation on all of the cells using the weight file specified within the input parameter
    - Note: Since the array is in the order of (column, row) but the weight vector is (row, column), the writing process also transposes the weight vector
  + Allocates memory for the convolution function output
* **convolutionFunction**
  + Scalable model of the convolution function output
  + ORDER OF OPERATIONS:
  + 1. Memory is allocated for multiplying the px values by the unit vector and also the ASDMs
  + 2. The input vector px is multiplied by the unit vector and then fed into the ASDM
  + 3. All of the cell conductances are copied into another vector for easy access
  + 5. Cell conductances are multiplied by unit vectors
  + 6. The current stream is generated through the multiplication process
  + 7. I\_total is calculated, C\_total follows
  + All C\_total values are fed into the ASDM and then placed in the convolutionOutput object of the function
* **activationFunction**
  + Calls on the activation function previously, but it is looped to do it column by column
* **poolingFunction**
  + The window thing
* **doEverything**
  + Performs all 3 stages in succession

**StochasticConvolution.cpp**

* Main file that only calls the convolution function

**Dependency Flowchart**

Diagram

Description automatically generated