Code Overview and Notes

A loose sketch of the code structure can be found in ##############. Some key features include:

* Adaptive domain partitions which divide a rectangular input domain amongst any number of processors such that the internal boundaries are smallest (i.e. smallest total messages per iteration) with any remainders distributed amongst the sub-grids to distribute workload
* Use of an MPI datatype for row and column of each sub-grid to allow for messaging between processors without any additional copying of data
* Use of a \_config.txt file to store the input dimensions, number of partitions and time taken for the run
* The program accepts command line inputs in the order (no input validation so be cautious):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| width (int) | height (int) | Iterations (int) | periodic (bool) | results directory (string) |

* Results directory is where all data is saved. These are labelled “iteration-processor.csv”. All saves are written though a stringstream to reduce write time
* Post processing is done in python by first compiling each iteration to a single csv file, then animating that and saving as an embedded html file
* Periodic is implemented though use of self-messaging with a destination ID to copy data across without extra complication in the code (note MPI never sends a message so this process is not particularly slow), hence the only additional code is changing the “find\_neighbours” function
* #define statements are used to control code functionality
  + to\_print and to\_print\_all controls the amount of printing output
  + synch uses MPI barrier to synch every processor at each iteration and setup stage
  + CX1 adds writing run-time to a times.csv file and disables non-essential functions that are not compatible with the gcc compiler

Validation and Animations

Run-time analysis

Runtime results are collected from ######### processors on the CX1 HPC. Repeats ################.

The times are measured from after MPI has initialised to when processor zero has finished, it includes the initialisation of GOL\_grid and saving of each iteration. As this implementation uses peer to peer communications, hence the only serial part of the code is on processor zero for timing and creating the \_config.txt which is virtually negligible to the run-time so we expect good scaling with number of processors.