Dynamical characterisation of neural networks and neurophysiological time series: Parallel approaches using Python

Thomas Greg Corcoran

08 June 2011

t.corcoran@susx.ac.uk
Centre for computational neuroscience and robotics (CCNR),
University of Sussex, Brighton UK

Abstract

This work is an exploration of the use of time-delay embedding to explore the dynamical properties of neural network simulations and physiological signals. Whilst parallel approaches such as MPI and CUDA are extremely useful in the simulation of neural systems, the construction of statistics from such simulations can be even more computationally challenging.

Reconstructed attractors (via Takens embedding [1]) are naturally immutable data structures, thus making their analysis an excellent candidate for parallel and/or functional approaches. Moreover, the calculation of statistics of attractors, such as finite-time Lyapunov exponents (FTLE) and entropies [2], are typically very time-consuming. Therefore a toolbox of parallel algorithms is desirable.

This work compares and explores the use of python multiprocessing (chunk-based parallism) and Thrust/PyCuda [3,4] (fine-grained parallelism via GPGPU) approaches to the problem.

The outcome of this work is a Python package for the dynamical characterisation of neuro-physiological and simulation timeseries, bridging the dynamical and statistical properties of neural systems. These indices are useful in the multiscale characterisation of activity patterns, dimensional reduction of models, and distinguishing of chaotic from stochastic systems.

References

1. Takens F. Detecting strange attractors in turbulence. Dynamical systems and turbulence. 1981

2. Castiglione P, Falcioni M, Lesne A, Vulpiani A. Chaos and Coarse Graining in Statistical Mechanics. Cambridge University Press; 2008.

3. Jared Hoberock and Nathan Bell. Thrust: A Parallel Template Library. http://www.meganewtons.com/. 2010

4. Pinto N, Lee Y, Catanzaro B, Ivanov P, Fasih A. PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation. arXiv. 2009