

CSCI 4/5576: Project Checkpoint

The Random Logistic Map

Amy Le (5576)
Long Tat (4576)
Emily Bertelson (4576)
Kristina Entzel (4576)

November 10, 2014

1 Review of Project Goals

The purpose of this project is to characterize the Random Logistic Map. In particular, we will be studying the stability of the map, which includes locating fixed points and generating bifurcation diagrams. The two main goals are:

1. Find the expected number of order p periodic orbits for a the random map ($p = 1, 2, 3, \dots$)
 - (a) For an initial starting value $x_0 \in [0, 1]$ and a specific random function $R_0(x)$, iterate until you find the fixed point(s), x_i^* associated with $R_0(x)$.
 - (b) Classify the fixed points in terms of a period p orbit.
 - (c) Each processor should take a different initial x_0 and report whether the initial condition led to finding a unique stable orbit (many initial conditions may converge to the same orbit or different orbits, while others may never converge). The processors should be properly load balanced.
 - (d) Repeat the above steps for a large number of different random maps $R_i(x)$, $i = 0, 1, 2, \dots, \bar{N}$ in order to find the expected number of order p periodic orbits for the random map.
2. Create a set valued bifurcation diagram [1]
 - (a) For many values of $r \in [0, 4]$, and a fixed random function $R_0(x)$, plot the locations of the periodic orbits as a function of r . A period p orbit will have p corresponding x values as its orbit locations (e.g. a period 1 orbit will have 1 fixed point, a period 2 orbit will have 2 fixed points, and so on).

As the map has an element of randomness to it, many simulations (a large \bar{N}) would be required for statistical analysis. This project is a subset of a larger work in progress and requires optimization. A serial implementation has been developed in MATLAB. The parallel implementation will be in C++.

2 Project Breakdown

The general progression of the project is outlined below:

1. Convert the Matlab code to C++ (object oriented)
2. Confirm the C++ versions of the code that we each produce work together correctly
3. Parallelize the C++ code
 - Use MPI to parallelize the mathematical computation
 - Invoke the load balancer to assign an initial condition to each fixed point iteration [2]
4. Benchmark
 - strong scaling study (speedup and efficiency)
 - weak scaling study (speedup and efficiency)

The table below summarizes how we have subdivided the project among ourselves.

Table 1: Division of Labor

Amy	make a random number distribution and use the cobweb diagram (fixed point) routine to return a list of subsequent iterations
Long	calculate period order, given a list of subsequent iterations
Kristi	calculate the average number of period p orbits, given a list of period locations, period orders, and starting positions
Emily	create the HDF5 file hierarchy to organize the simulation data; use links in the file to generate views of the data for easy plotting

The table below summarizes how we have each progressed:

Table 2: Progress

Amy	Converted Matlab code that computes the random function and does the fixed point iteration to C++. In the process of debugging the fixed point iteration; it currently does not converge.
Long	
Kristi	
Emily	

3 Description and Background

3.1 Random Logistic Map

The Random Logistic function is a special case of the Logistic function, which is not as popularly studied. In the random case, r becomes a value dependent on x . Essentially, the Random Logistic Map introduces the notion of spatial randomness to the deterministic function. The Random Logistic function and map are as follows:

$$\begin{aligned} f(x) &= R(x)x(1-x) \\ x_{n+1} &= R(x_n)x_n(1-x_n) \end{aligned} \tag{1}$$

$R(x)$ consists of a Fourier series whose coefficients are independent (but not identical) random variables drawn from a uniform distribution whose bounds depend on the Fourier mode.

$$\begin{aligned} \ln(R(x)) &= \xi(x) \\ \xi(x) &= \ln(r) + 2 \sum_{n=1}^N a_n \cos(2\pi nx) - b_n \sin(2\pi nx) \\ a_n, b_n &\sim \text{Unif}(-M_n, M_n) \\ M_n &= \sqrt{1.5 S_n} \\ S_n &= \alpha e^{-L|n|} \\ \alpha &= \sigma^2 \tanh(L/2) \\ \sigma &< \ln(4/r) \frac{\tanh(L/4)}{\sqrt{1.5 \tanh(L/2)}} \end{aligned}$$

Where $L \in (0, 1)$ represents the correlation length (and is fixed for each simulation) and $r \in [0, 4]$ is also fixed for each simulation.

An initial serial implementation has been developed for this project and has been mildly tested in MATLAB. Our project would consist of converting the existing code to C++ and implementing parallelization through MPI with proper load balancing. Other possible improvements include IO optimization and memory optimization (possible improvements through HDF5).

3.2 Testing

Even though the nature of this project is random, it is possible to compare the output of the parallel implementation to the serial implementation for accuracy. The randomness lies in the function $R(x)$, so using the same values of $R(x)$ for the serial and parallel code should output the same results.

References

- [1] Jeroen S.W. Lamb, Martin Rasmussen, and Christian S. Rodrigues. Topological bifurcations of minimal invariant sets for set-valued dynamical systems. *Proceedings of the American Mathematical Society*, 2013.

- [2] Marc H. Willibeek-LeMair and Anthony P. Reeves. Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on Parallel and Distributed Computing*, 4(4), September 1993.