

# Computational Economics Assignment 1

## Ergodicity

John Stachurski

November 2025

This document begins with a short introduction to ergodic chaos

Assignment exercises are at the end

Consider the one-dimensional dynamic system

$$x_{t+1} = g(x_t) \quad \text{where} \quad g(x) := 4x(1 - x). \quad (1)$$

The law of motion in (1) provides an update rule for the state  $x_t$  and a **trajectory**  $(x_t)_{t \geq 0}$  recursively defined by  $x_{t+1} = g(x_t)$  and some fixed initial condition  $x_0$

Another way to write this trajectory is

$$x_t = g^t(x_0) \quad \text{where} \quad g^t := g \circ \cdots \circ g$$

The product on the right is  $t$  compositions of  $g$  with itself.

The map  $g$  is called either the **logistic map** or the **quadratic map**

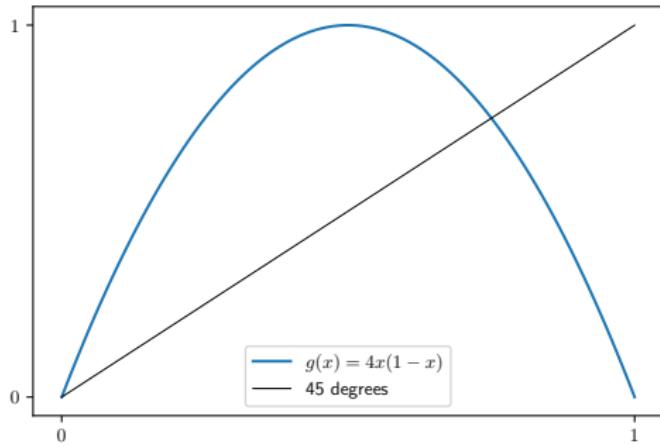


Figure: The quadratic map and 45 degree line

The 45 degree diagram on the previous slide suggests that cycles occur when we iterate with the quadratic map

This is easily confirmed by plotting time series

The next figure shows one such series when  $x_0 = 0.3$

We see that the behavior of the state is not only cyclical but also highly erratic

It shows no sign of convergence to regular behaviour

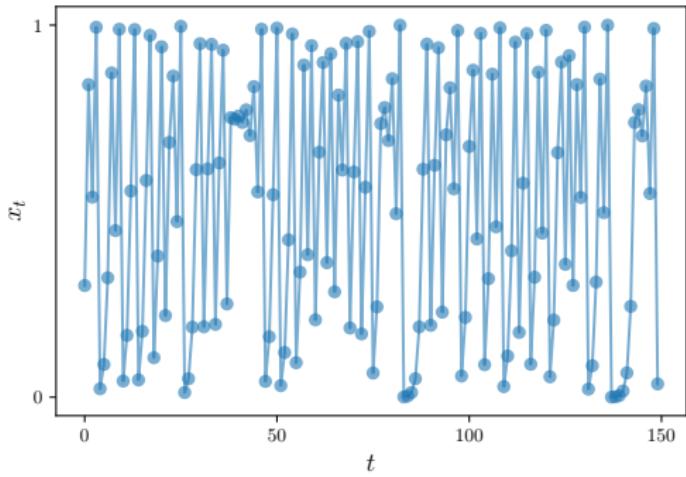


Figure: Time series from the quadratic map when  $x_0 = 0.3$

The figure on the next slide shows  $x_t = g^t(x_0)$  as a function of the initial condition  $x_0$

Specifically, we set

- $x_0$  ranges across a grid of 100 points in  $(0, 1)$
- $n = 20$

Thus, the figure shows how the forecast state 20 periods hence changes with the initial condition

Evidently there is great variability

Thus, we have sensitive dependence on initial conditions

If this is a model for some aggregate quantity, then we can say almost nothing in terms of useful predictions, since the initial condition will almost always be measured with some degree of error

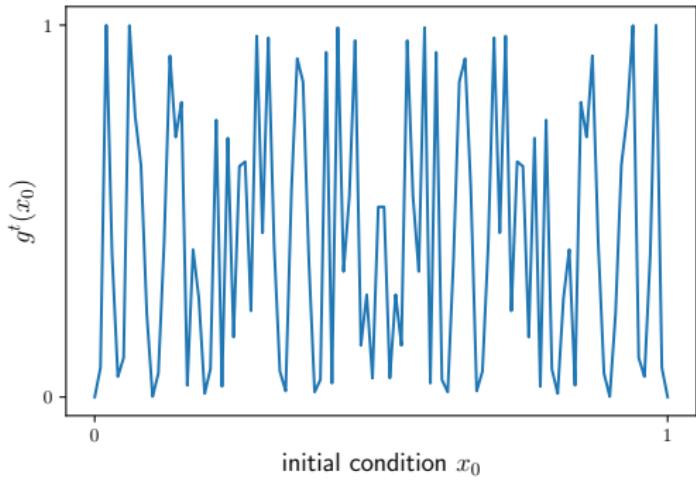


Figure: The time  $t$  state as a function of the initial condition

At the same time, the model can generate strong “statistical” predictions over long time horizons

As one way to see this, consider the histogram shown in the next figure

This is a histogram of the sequence  $(g^t(x_0))_{t=0}^n$  when  $x_0 = 0.3$  and  $n = 100,000$

Evidently there is some pattern to the apparent randomness, since the histogram is symmetric and U-shaped

Moreover, if we choose an alternative initial condition and reproduce the histogram, then, apart from a few special cases (e.g.,  $x_0 = 0$ ), exactly the same pattern arises

Hence, at the level of distributions, sensitive dependence on initial conditions disappears

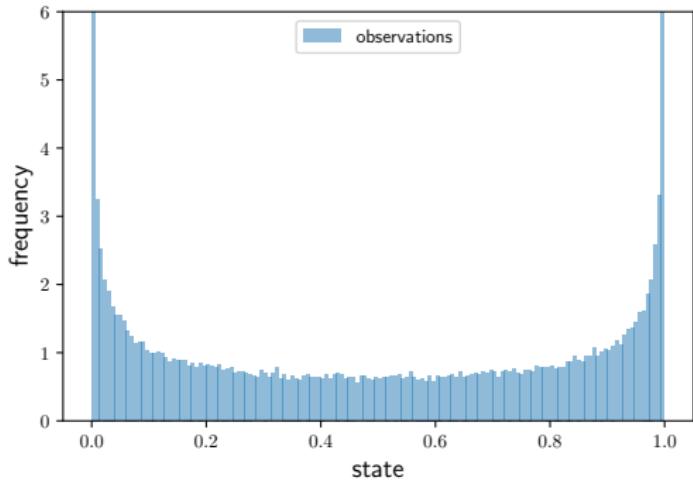


Figure: A histogram of  $(x_0, \dots, x_n)$  when  $n = 100,000$

The shape of the histogram becomes clearer as the length of the time series increases

In fact, there is a specific limit to this process: the histograms are converging on  $(0, 1)$  to the function

$$\psi^*(x) := \frac{1}{\pi \sqrt{x(1-x)}} \quad (x \in (0, 1)) \quad (2)$$

The function  $\psi^*$  is called the **arcsine distribution**

The figure on the next slide illustrates converges of the histogram to  $\psi^*$  as  $n \rightarrow \infty$  by jointly plotting the histogram of an even longer time series and the density  $\psi^*$

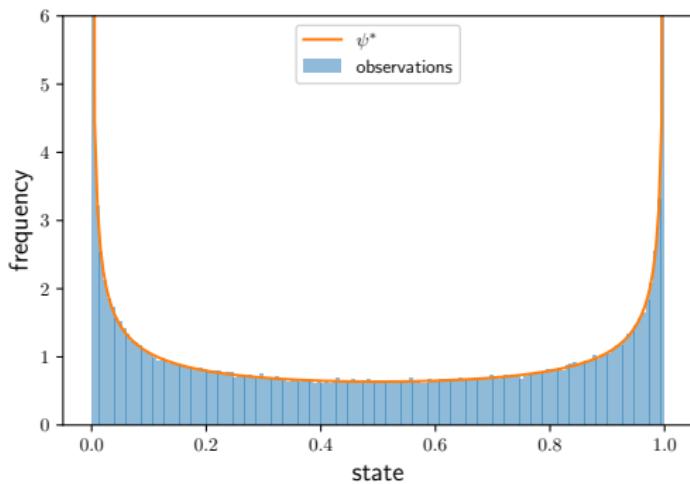


Figure: Histogram of a length 250,000 trajectory and the stationary density  $\psi^*$

In fact one can also show that, for any function  $h$  with  $\int |h(x)|\psi^*(x) dx < \infty$ , we have

$$\frac{1}{m} \sum_{t=1}^m \int h(g^t(x_0)) \rightarrow \int h(x)\psi^*(x) dx \quad (m \rightarrow \infty) \quad (3)$$

with probability one.

The result in (3) is a version of the celebrated **Birkhoff ergodic theorem**, proved by George David Birkhoff (1884–1944) in 1931, which verified fundamental claims in the field of statistical mechanics.

## Cross-sectional data

Suppose  $x_t^i$  is a measurement related to individual  $i$  in a cohort of  $n$  agents

For simplicity, let's call it "wealth"

Suppose each individual's wealth updates as  $x_{t+1}^i = g(x_t^i)$

We take as given the initial states  $(x_0^i)_{i=1}^n$ , representing the cross-sectional wealth distribution at time zero

Then we forecast the time  $t$  wealth distribution by computing  $(g^t(x_0^i))_{i=1}^n$

The next figure shows three forecasts corresponding to three different initial distributions.

Each row, containing a left and right subfigure, was constructed as follows

- an initial density  $\psi_0$  was chosen
- 100,000 observations  $(x_0^i)_{i=1}^n$  were drawn from  $\psi_0$
- the cross section was shifted forward to time  $t = 50$  by computing  $(g^t(x_0^i))_{i=1}^n$

On the left, we show the density  $\psi_0$  that was used to generate the draws and the histogram of draws

On the right we show the histogram of  $(g^t(x_0^i))_{i=1}^n$  and the density  $\psi^*$

The three initial densities are Beta(2, 2), Beta(2, 5), and Beta(5, 2)

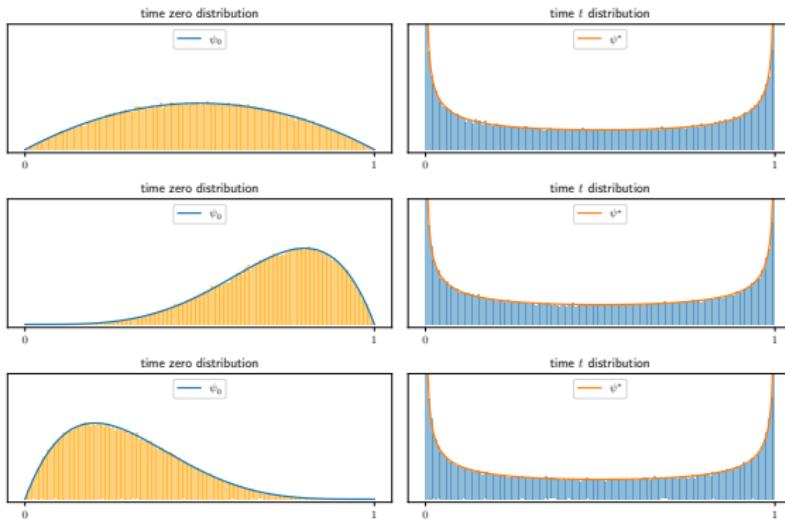


Figure: Initial (left) and predicted (right) cross-sectional distributions

The initial distributions are represented as histograms on the left hand side of the figure

It is apparent from the figure that (a) the forecast distributions display significantly more regularity than the initial conditions, and (b) the forecast cross-sectional distributions look very similar to the distribution generated from a single time series

More specifically, the simulations show that the cross-sectional histograms also converge to  $\psi^*$  as the number of countries and forecast date become large

**Exercise 1** Produce a Jupyter notebook where you replicate all figures in the graph

Easiest way:

- Write on Colab
- Download as ipynb file at the end

Pay attention to the quality of your code

- avoid cut and paste
- avoid magic numbers, etc.
- if using JAX, only use pure functions

See [https://python-programming.quantecon.org/writing\\_good\\_code.html](https://python-programming.quantecon.org/writing_good_code.html)

**Exercise 2** Adding to the same Jupyter notebook, shift a cross-section  $(x_0^i)_{i=1}^n$  forward in time  $t$  periods

1. draw  $(x_0^i)_{i=1}^n$  as IID samples from Uniform(0, 1)
2. compute the updated cross section  $(g^t(x_0^i))_{i=1}^n$

Perform step 2 as quickly as possible while still using 64-bit floats

Time your solution with the following settings

- $n = 10,000,000$
- $t = 100$

Email solution to my gmail

- `firstname_lastname_studentnumber_ass_1.ipynb`
- Check that it all runs after restarting your kernel!