Independent Study : Implementation of Non-linear Workbook in Python

Prajwal Padmanabha under supervision of Dr. Ananda Dasgupta

Introduction

The book 'The Nonlinear Workbook' by Willi Hans Steeb is a toolbox of various algorithms and methods used in the field of non-linear dynamics. The book contains the tools and brief overview of the math behind the tools, ranging from chaos to genetic algorithms to neural networks. In addition to this, the book also lists implementation of the algorithm in C++ and Java. This book is of immense use for anyone foraying into the field of scientific computation.

One of the widely used languages in computation in science is Python. The main reasons behind this are the ease of coding and the availability of numerous packages for specific purposes. The goal of the independent study is to study the algorithms and tools listed in the book and to recreate them in Python (using plain python and at places possible, the available additional packages).

1 Nonlinear and Chaotic Maps

A map is a function $f : S \to S$ where $S \subset \mathbb{R}$.

For continuous time, a differential equation would be $\dot{x} = f(x)$ while a discrete time evolution would be written as $x_{t+1} = f(x_t)$ and this is called a difference equation. In both cases, $x_0 \in S$.

The sequence $x_0, f(x_0), f(f(x_0))....$ is called the trajectory of x_0. In this trajectory, a point is a fixed point of the map if $x^* = f(x^*)$ and periodic if $x^* = f^{(n)}(x^*)$. We consider some examples of maps and check for periodicity and fixed points in them.

1.1 Trajectories and Periodicity in One Dimension

Example 1 : A Simple map

$$f(x) = \begin{cases} \frac{x}{2} & \text{x is even} \\ 3x + 1 & \text{x is odd} \end{cases}$$

Example 2 : Logistic Map

$$f(x) = 4x(1 - x)$$

Example 3 : Bernoulli Map

$$f(x) = \begin{cases} 2x & x \leq 0.5 \\ 2x - 1 & \text{otherwise} \end{cases}$$

Example 4 : Bungalow Tent Map

$$f_a(x) = \begin{cases} \dfrac{1-a}{a}x & x \in [0, a) \\[2ex] \dfrac{2a}{1-2a}x + \dfrac{1-3a}{1-2a} & x \in [a, 0.5) \\[2ex] \dfrac{2a}{1-2a}(1-x) + \dfrac{1-3a}{1-2a} & x \in [0.5, 1-a) \\[2ex] \dfrac{1-a}{a}(1-x) & x \in [1-a, 1] \end{cases}$$

Example 5 : Gauss Map

$$f(x) = \begin{cases} 0 & x = 0 \\ [1/x] & \text{otherwise} \end{cases}$$

where $[n]$ denotes fractional part of n .
The codes to print the trajectories of these maps can be found in **Appendix A**.

Certain points to be noted with regard to code:

1. Unlike C++ or Java, no new *long int* variable needs to be declared in Python to handle large values since there is no upper bound on the value of memory assigned to a variable. The disadvantage of this is, if a large number is given, the program will not terminate with an overflow error and instead will continue drawing more memory.
2. For maps like Bernoulli Map and Tent Map, taking floating point numbers may not be accurate representation of fractional values since these maps are chaotic and sensitive to initial conditions. Taking this into account, the code has been written with a provision to do computation in fractional mode. The caveat here is that the numerator and denominator can quickly become large values (in Bernoulli map), which slows down computation significantly.

1.2 Invariant Density

Consider a fully chaotic, one hump map, $f(x)$. Then, invariant density can be defined as

$$\rho(x) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \delta\big(x - f^{(t)}(x_0)\big)$$

This function acts as a measure of how many times a point has been visited, i.e, like a probability density. We shall see that this is equivalent to a probability density. Due to this, there is a need for the function to be fully chaotic. If the function is not fully chaotic, i.e, there exists an attractor, or a periodicity, the function ends up at a finite set of points making the invariant density exist only at those points. Using the logistic map, we shall see examples of this.

The reason for function to be one hump is not clearly understood. An example of double hump map also shows invariant density. Further investigation needs to be done in this matter.

Some properties of Invariant Density

Consider an integrable function $g(x)$. Then,

$$\langle g(x)\rangle = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} g(x_t) = \int_0^1 \rho(x)g(x)dx$$

Setting $g(x)=1$, we obtain, just like a probability density,

$$\int_0^1 \rho(x)dx = 1$$

Let us define

$$\sigma(y) := \int_0^1 \delta\big(y - f^k(x)\big)\rho(x)dx$$

Then,

$$\begin{aligned}
\int_0^1 \sigma(y)g(y)dy &= \int_0^1\int_0^1 \delta\big(y-f^k(x)\big)\rho(x)g(x)dxdy = \int_0^1 \rho(x)g(x_k)dx \\
&= \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\int_0^1 \delta\big(x-f^t(x_0)\big)g(f^k(x_0))dx \\
&= \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} g\big(f^{t+k}(x_0)\big) = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\int_0^1 \delta\big(y-f^{t+k}(x_0)\big)g(y)dy \\
&= \int_0^1 \rho(y)g(y)dy
\end{aligned}$$

Since σ was arbitrary, we can state $\sigma(y)=\rho(y)$. Hence, setting $k=1$ in definition of σ, we get

$$\begin{aligned}
\rho(y) &= \int_0^1 \rho(x)\delta\big(y-f(x)\big)dx \\
\implies \rho_{t+1}(y) &= \int_0^1 \rho_t(x)\delta\big(y-f(x)\big)dx
\end{aligned}$$

And this allows us to find out the invariant density as $T\to\infty$ by time evolution of $\rho_0(x)$

Examples of Invariant Densities

Example 1 : Bungalow Tent Map
Example 2 : Double Hump Map
Example 3 : Logistic Map
Example 4 : Sine Map

1.3 Discrete Fourier Transform

Let $f(t)$ be a continuous time signal, sampled at N points, $f[0], f[1], ..., f[N-1]$. Fourier transform of $f(t)$ is given by (barring normalization)

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

The samples points $f[n]$ is equivalent to an impulse in Δt having area $f[n]$. Then,

$$\begin{aligned} F[\omega] &= \int_0^{T(N-1)} f(t)e^{-i\omega t}dt \\ &= f[0]e^{-i\omega\times 0} + f[1]e^{-i\omega\times 1} + ... + f[N-1]e^{-i\omega\times T(N-1)} \\ &= \sum_{n=0}^{N-1} f[n]e^{-i\omega nT} \end{aligned}$$

Analogous to the fundamental time period after which the signal repeats in Fourier expansion, since the sampling is done only from 0 to N-1, we can assume that the signal repeats after the N-1 time sampling. Hence, we can write

$$\omega = 0, \frac{2\pi}{NT}, 2\times\frac{2\pi}{NT}, ..., (N-1)\times\frac{2\pi}{NT}$$

This gives us the expression for discrete fourier transform:

$$F[k] = \sum_{n=0}^{N-1} f[n]e^{-i\frac{2\pi nk}{N}}$$

and with normalization,

$$F[k] = \frac{1}{N}\sum_{n=0}^{N-1} f[n]e^{-i\frac{2\pi nk}{N}}$$

This can be rewritten as a matrix equation:

$$\vec{F} = D\vec{f}$$

$$[D]_{ij} = \frac{1}{N}W^{ij} \qquad \text{where } W = e^{-i\frac{2\pi}{N}}$$

Since this is a matrix multiplication with a vector, the order complexity of this operation is $\mathcal{O}(n^2)$

DFT of cosine map

The map considered is $x(t) = cos\left(\frac{2\pi t}{N}\right)$ sampled at $t = 0, 1, ..., N-1$. The DFT for this is given analytically by

$$X[k] = \begin{cases} \frac{1}{2} & k = 1 \\ \frac{1}{2} & k = N-1 \\ 0 & \text{otherwise} \end{cases}$$

DFT of Logistic Map

DFT of a logistic map shows interesting properties. We assume that we have N samples from the logistic map. Depending on the value of r we see different results. $r = 4$ shows chaos in the DFT. $r = 3.2$ shows two period and $r = 3.5$ shows four period which is consistent with the bifurcation diagram of logistic map.

1.4 Fast Fourier Transform

The DFT explained in the previous subsection is of the order of $\mathcal{O}(n^2)$ which can become very slow of large values for n (as is obtained in time series data). To make this more efficient, some properties in the DFT must be noted.

Ignoring normalization constant, the DFT is given by

$$F[k] = \sum_{n=0}^{N-1} f[n] e^{-i\frac{2\pi nk}{N}} = \sum_{n=0}^{N-1} f[n]\, W_N^{nk}$$

Certain values of W_N^{nk} repeat multiple times due to:

1. The value of product nk repeats multiple times.
2. W_N^{nk} represent the roots of unity. These have repeated absolute values (assuming N is even).

To give an example, $N = 8$. Then,

$$\begin{aligned} W_8^1 &= e^{-i\frac{2\pi}{8}} = \frac{1-i}{\sqrt{2}} := a \\ W_8^2 &= a^2 = -i \\ W_8^3 &= a^3 = -ia = a^* \\ W_8^4 &= a^4 = -1 \\ W_8^5 &= a^5 = -a \\ W_8^6 &= a^6 = i \\ W_8^7 &= a^7 = ia = -a^* \\ W_8^8 &= a^8 = 1 \end{aligned}$$

Since any nk falling outside 0 to 7 can be written as a modulo 8 value falling within 0 to 7. Hence, only four values are needed, $a, a^*, i, 1$, which simplifies the computation speeding up DFT. This is the core idea used in Fast Fourier Transform.

One algorithm for performing FFT is called Decimation in Time algorithm. This splits the summation over k into two parts. (Note: let us assume that $N = 2^p$ for some value p)

$$\begin{aligned} m &= \frac{k}{2} \text{ if k is even} \\ m &= \frac{k-1}{2} \text{ if k is odd} \end{aligned}$$

Then,

$$F[k] = \sum_{m=0}^{\frac{N}{2}-1} f[2m]\, W_N^{2mk} + \sum_{m=0}^{\frac{N}{2}-1} f[2m+1]\, W_N^{(2m+1)k}$$

$$\text{Note that } W_N^{2mk} = e^{-i\frac{2\pi}{N}2mn} = e^{-i\frac{2\pi}{N/2}mn} = W_{\frac{N}{2}}^{mn}$$

$$\implies F[k] = \sum_{m=0}^{\frac{N}{2}-1} f[2m]\, W_{N/2}^{mk} + \sum_{m=0}^{\frac{N}{2}-1} f[2m+1]\, W_{N/2}^{mk}\, W_N^k$$

$$\implies F[k] = G[k] + W_N^k\, H[k]$$

which is a two $\frac{N}{2}$ order DFT. This process can be iterated until two point DFT is reached (hence the assumption of N as a power of 2).

1.5 Lyapunov Exponent

Lyaponov exponent characterizes how infinitesimally close trajectories change with time.

Let $x^*(t)$ and $x(t)$ be two trajectories of the map $x_{t+1} = f(x_t)$ with $x^*(0)$ and $x(0)$ being infinitesimally close. Then,

$$y_t := x_t^* - x_t$$
$$\implies y_{t+1} = x_{t+1}^* - x_{t+1} = f(x_t^*) - f(x_t)$$

Doing Taylor expansion around x_t, we get $y_{t+1} = (x_t^* - x_t)\dfrac{df(x_t)}{dx} = y_t\dfrac{df(x_t)}{dx}$

In a continuous system, this would give the solution $y(t) = y(0)e^{\lambda t}$. λ is called the Lyapunov exponent. In a n dimensional system, there is a spectrum of Lyapnunov exponents $\lambda_1, \lambda_2 ... \lambda_n$. Maximal Lyapunov exponent is defined as

$$\lambda = \lim_{t\to\infty} \lim_{y_0 \to 0} \frac{1}{t} log \frac{|y(t)|}{|y_0|}$$

In a discrete time system, this translates to

$$\lambda(x_0, y_0) := \lim_{T\to\infty} \frac{1}{T} log|\frac{y_T}{y_0}|$$

Since $f'(x_t) = \dfrac{y_{t+1}}{y_t}$,

$$\lambda(x_0, y_0) = \lim_{T\to\infty} \sum_{t=0}^{T} \frac{1}{T} log(|f'(x_t)|)$$

If $\lambda > 0$, this means that infinitesimally close trajectories exponentially grow apart in time. This is a measure of chaos of the system in the sense that it is very sensitive to initial conditions and any small perturbation will lead to a butterfly effect over time.
Note that λ depends on the inital point x_0. Hence, if x_0 is a fixed point or a periodic point or eventually periodic, the Lyapunov exponent will be the sum of constant terms. If the slope at that point is less than 1, then λ is negative (0 if slope is less than e and diverges otherwise). Hence, calculating Lyaponov exponents for those points is pointless.

Lyapunov exponent for Logistic map with r=4

1.6 Autocorrelation Function

Let $x_{t+1} = f(x_t)$. Then, the time average of x is, $\langle x \rangle = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} x_t$. Autocorrelation function can then be defined as:

$$C_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} (x_t - \langle x \rangle)(x_{t+\tau} - \langle x \rangle)$$

Autocorrelation function is a measure of the correlation in time series data. Hence, it can be used as a check for randomness in the system (i.e, $C_{xx}(\tau) \to 0$ as $\tau \to \infty$). In the event of correlation existing i.e, non randomness, it can give clues about the underlying model of the time series data.

Autocorrelation Function for Logistic Map

Appendix A : Codes