DOING PHYSICS WITH PYTHON

COMPLEX SYSTEMS

DISCRETE DYNAMICAL SYSTEM WITH ONE DEGREE OF FREEDOM

Ian Cooper

matlabvisualphysics@gmail.com

DOWNLOAD DIRECTORIES FOR PYTHON CODE

Google drive

GitHub

cs_001.py

INTRODUCTION

A **dynamical system** is a system whose state is uniquely specified by a set of variables and whose behaviour is described by predefined rules.

The **degree of freedom** of a system is the number of state variables needed to uniquely specify the system's state.

In this document, I will consider discrete dynamical systems with one degree of freedom (1 dof) described by difference equations.

A 1 dof discrete dynamical system is governed by a difference equation (recurrence equation, iterative map if function independent of the time step) of the form

$$x_{t+1} = F\left(x_t, t\right)$$

A 1 dof continuous dynamical system is governed by a differential equation of the form.

$$dx / dt \equiv \dot{x} = F\left(x_t, t\right)$$

Autonomous system is a dynamical system that is independent of time where the function is $F(x_t)$ for a discrete system or $F(x)$ for a continuous system.

Non-autonomous system is a dynamical system that is time dependent where the function is $F(x_t, t)$ for a discrete system or $F(x, t)$ for a continuous system.

Linear equations are always analytically solvable, while nonlinear equations don't have analytical solutions in general. An analytical solution is of the form of $x(t) = f(t)$ and is called a **closed-form** solution. A closed-form solution is helpful because the variable x at any time t can be predict.

1 dof DISCRETE DYNAMICAL SYSTEMS

Exponential growth

Consider the very simple difference equation for the variable x

$$x_{t+1} = a\, x_t$$

where x_{t+1} is the value of x at time step t+1 and x_t is the x value at time step t, a is a model parameter and is constant, and the initial condition is given by the value of x_0.

The analytical solution is

$$x(t) = x_0\, e^{bt} \qquad b = \log\left(x_n / x_0\right) / t_n$$

where the value of the constant b is found from the value of x_n at time step n.

Thus, the variable x will increase exponentially with time as shown in figure 1 where the initial condition is $x_0 = 1$ and $a = 1.2$

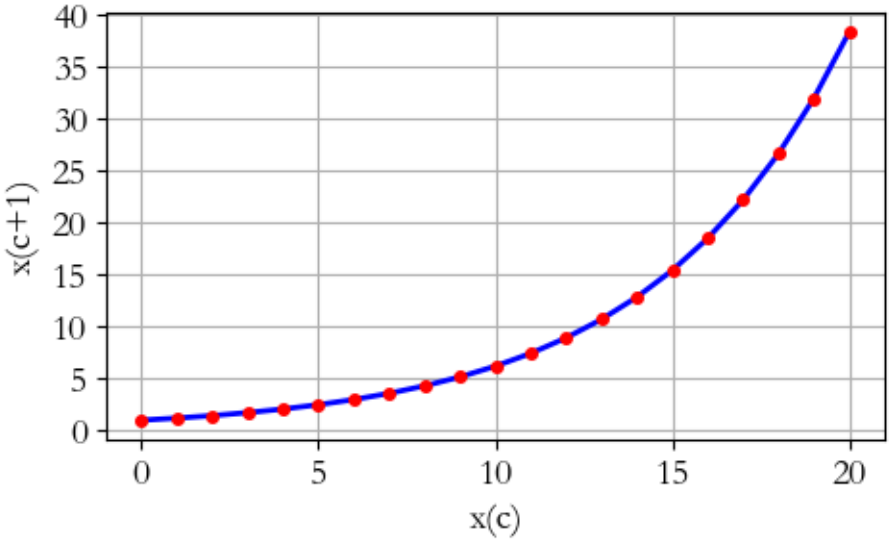

Fig. 1. Blue curve: solution of the difference equation
Red dots – exponential function
$x_0 = 1$ $a = 1.2$ **cs_001.py**

Fish population growth with constant harvesting

We can model the growth of a fish population when there is a constant removal of fish (harvesting) by the simple linear difference equation

$$x_{t+1} = a\,x_t - b$$

where x is the scaled fish population with initial condition given by x_0, and a and b are positive constants. The results of the simulation are shown in figure 2.

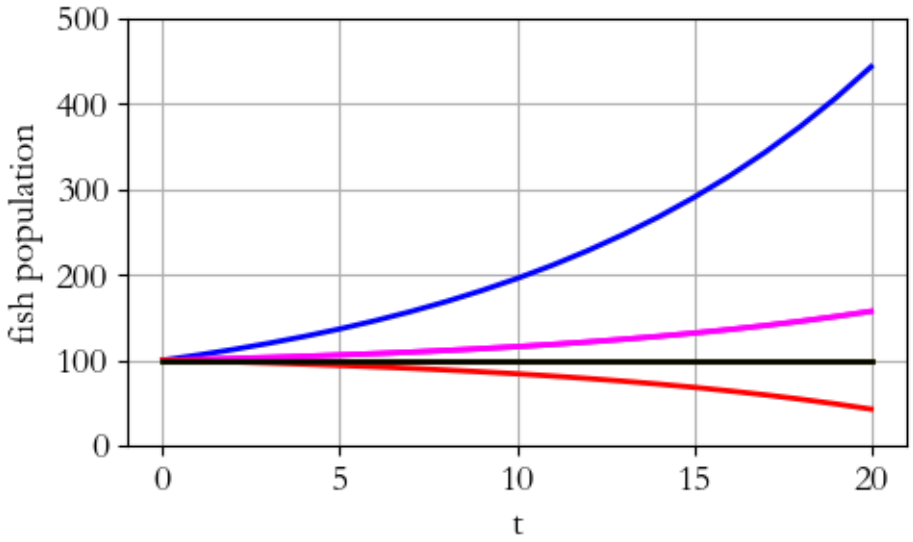

Fig. 2. Time evolution of scaled fish population: x_0 =100 and a = 1.1. **Blue curve b = 4**, **magenta curve b = 9**, **red curve b = 11**, and **black curve b = 10**.
x_0 = 1 a = 1.1 **cs_001.py**

From the plot displayed in figure 2, it is obvious why this difference equation is so useful in describing the evolution of a fish population. With x_0 = 100, for of b < 10 then the population will grow exponentially and if b > 10 then the population will become extinct. However, if b = 10, the there is a stable equilibrium and the population remains constant at 100 only if the initial population is 100 (x_0 = 100).

The steady-state solution x_{ss} occurs when $x_{t+1} = x_t$, so

$$x_{ss} = \frac{b}{a-1}$$

If $a = 1.1$

$b = 10$, then $x_{ss} = 100$

$b > 10$, then xss = 0

$b < 10$, then $x_{ss} \to \infty$ as shown in figure 2.

If $a \leq 1$, then $x_{ss} = 0$

Python

In Python it is difficult to have multiple plots on the one graph with a simulation where a parameter is assigned different values as shown in figure 2. The way I do it is to divide the code into cells, one cell for the calculation and another cell for the display of the plot. Run the code and obtain the plot. In the cell change the parameter (*b* in this case) and the variable for the color of the plotted line and run the cell. In the plot cell, highlight the code for just plotting the line and execute it by pressing F9 to add the new line to the plot. Just repeat the procedure for other lines.

```
xP = R
yP = x
plt.plot(xP,yP,linewidth=2,color = col)
```

REFERENCES

LibreTexts Mathematics