Christine Pho
PHYS 50733 Computational Physics
Mia Bovill
May 8, 2019

Predator-Prey Model Project

1. Introduction

The Lotka-Volterra model is the one of the earliest models of predator-prey population interaction. It demonstrates how two single species interact to change and modulate each other's populations through time. Examples of predator-prey interaction include herbivore-plant, parasite-host and other ecological population dynamics. The model has also been used in numerous and diverse fields such as epidemics to demonstrate susceptible-infective individuals and in economics to demonstrate populace-predatory institution dynamics [1].

It was discovered in the 1920s independently by American biophysicist Alfred Lotka and Italian mathematician and Vito Volterra. Volterra proposed the model to explain the increasing predator fish and decreasing prey fish in the Adriatic Sea during World War I [1]. At the same time, Lotka proposed the model to describe a chemical reaction in which the chemical concentrations oscillate. The model is based on a system of differential equations.

A commonly cited field experimental study of the Lotka-Volterra model is the study of Lynx and Snowshoe Hare Populations by Krebs et al. [2]. They followed the two populations over eight years and performed experiments such as removing lynx (predator) population and stocking supplemental food and plant fertilizer to test the effect of resource availability and predatory pressure on Snowshoe Hare (prey) population [3].

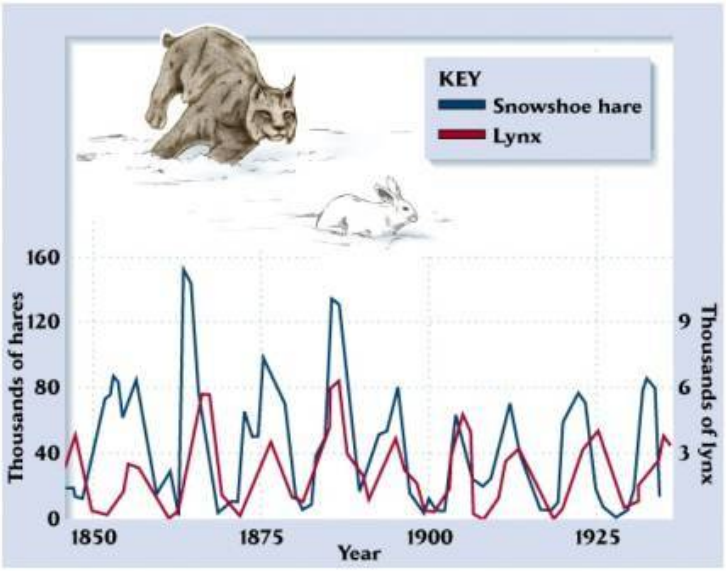

Figure 1. Snowshoe Hare and Lynx Population Study
Source: https://theglyptodon.wordpress.com/2011/05/02/the-fur-trades-records/

2. Theory

The model is written as a system of differential equations with two variables "x" and "y" where x represents the prey population of "rabbits" and y represents the predator population of "foxes." You can think of x and y as the populations in thousands, so that x=2 means that there are 2000 rabbits. This means that x and y would be limited to multiples of 0.001, since its only possible to have whole number of rabbits and foxes. But 0.001 is a pretty close spacing of values, so it is reasonable to approximate x and y as continuous real numbers as long as x and y>>0.

$$\frac{dx}{dt} = (a - \beta y)x = f(x, y)$$

Equation 1.

$$\frac{dy}{dt} = (\gamma x - \delta)y = g(x, y)$$

Equation 2.

There are four parameters a, β, γ, δ.

- The parameter a is the growth rate of species x (the prey) in the absence of interaction with species y (the predators). The growth rate decreases linearly with increasing y, by a factor β, possibly becoming negative.
- The parameter β represents the efficiency at which predators meet and kill the prey.
- The term γx is the growth rate of the predator population y in response to the size of the prey population.
- The parameter δ is the death (or emigration) rate of the predator population due to natural causes.

This model makes several assumptions that are not always realistic in nature [4].

- The prey population x grows exponentially, unless subject to predation.
- The rate of change of each population is proportional to its size.
- The food supply of predator y depends entirely on prey x and no other prey.
- The predator can consume infinite quantities of prey.
- The environment does not change to favor one species over time, and genetic adaptation does not occur.
- The populations do not approach an equilibrium through time. There are limited examples of this cyclical population dynamic in nature.

3. Method

The differential equations were integrated using a Runge-Kutta "RK4" method. It is a numerical method of solving first order differential equations with a given initial value. Each increment is the weighted average of four increments (k_1, k_2, k_3, k_4) based on four slopes at different points along the time interval.

The initial values of the populations were set to be x(0)=2.0 and y(0)=2.0. The spacing of the time steps were set to a value h, where $\mathrm{h} = (\mathrm{tf} - \mathrm{ti})/\mathrm{N}$. The final time was set $\mathrm{tf} = 30.0$ and initial time $\mathrm{ti} = 0.0$. The number of integration steps N=10000. The parameters were also fixed a, β, γ, δ for each run of the model.

Each value of x(t+h) and y(t+h), excluding the initial values, were generated by the following equations:

$$x(t+h) = x(t) + \frac{1}{6}(\, k_{1,x} + 2k_{2,x} + 2k_{3,x} + k_{4,x})$$
$$y(t+h) = y(t) + \frac{1}{6}(\, k_{1,y} + 2k_{2,y} + 2k_{3,y} + k_{4,y})$$

where

$$k_{1,x} = h\, f\big(x(t), y(t)\big) \qquad k_{1,y} = h\, g(x(t), y(t))$$

$$k_{2,x} = h\, f\left(x(t) + \frac{k_{1,x}}{2}, y(t) + \frac{k_{1,y}}{2}\right) \qquad k_{2,y} = h\, g\left(x(t) + \frac{k_{1,x}}{2}, y(t) + \frac{k_{1,y}}{2}\right)$$

$$k_{3,x} = h\, f\left(x(t) + \frac{k_{2,x}}{2}, y(t) + \frac{k_{2,y}}{2}\right) \qquad k_{3,y} = h\, g\left(x(t) + \frac{k_{2,x}}{2}, y(t) + \frac{k_{2,y}}{2}\right)$$

$$k_{4,x} = h\, f\big(x(t) + k_{3,x}, y(t) + k_{3,y}\big) \qquad k_{4,y} = h\, g\big(x(t) + k_{3,x}\,, y(t) + k_{3,y}\big)$$

4. Verification of program

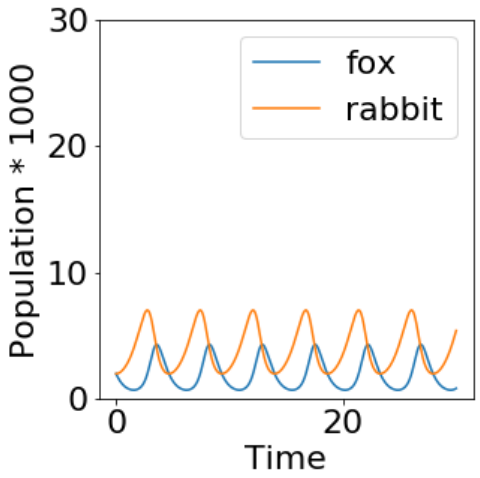

Figure 2.

This is one of the outputs of running the program and we find that the rabbit and fox populations in the output do indeed behave as expected. The populations oscillate such that maximum peaks in fox population occur lags slightly behind the maximum peaks in rabbit populations. We can see this similar behavior in Figure 1. with the Snowshoe Hare and Lynx study.

To verify the program, I wanted to see how the population behaved at if the initial conditions were at the critical points. The system has two critical points. (x,y)= (0,0) and $(\delta/\gamma, \alpha/B)$

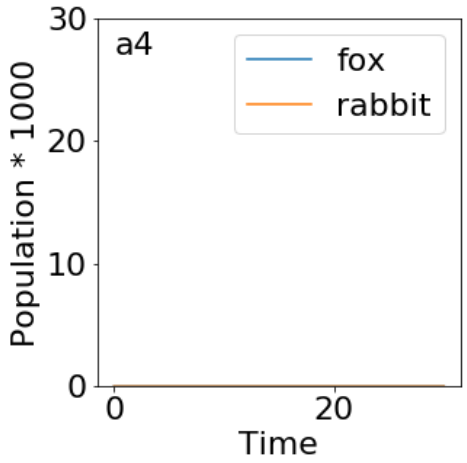

Figure 3. (x(0),y(0))= (0,0)

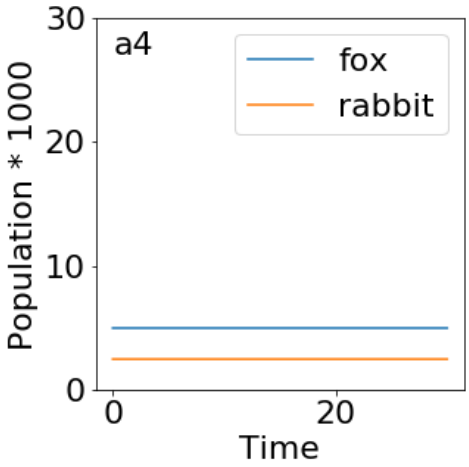

Figure 4. (x(0),y(0))= $(\delta/\gamma, \alpha/B)$

As expected, the populations did not change in these outputs since $\frac{dx}{dt}$ and $\frac{dy}{dt}$ are zero at the critical points.

5. Results

Part 1 and 3: Solve for the case $\alpha = 1, \beta = \gamma = 0.5$, and $\delta = 2$ and starting from initial x(0)=y(0)=2. Have the program make a graph showing both x and y as a function of time on the same axes from t=0 to t=20 for each case. Explain what is going on in these systems, in terms of rabbits and foxes.

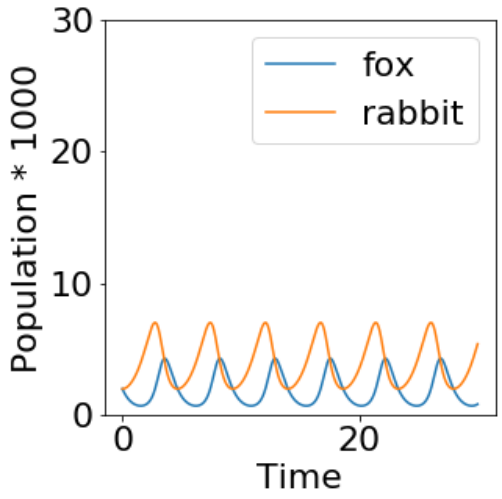

Figure 6.

Here we again have the output of a specific case of the model. One can observe that when the population of rabbits is low, there is a subsequent dip in the population of foxes due to low food sources. And as the rabbit population peaks there is a subsequent peak in the population of foxes due to increase in food sources. However, the peak in foxes quickly leads to a decrease in rabbit foxes, since they are being killed off by more predators. Both populations oscillate in this manner, without dampening.

Part 2: Explore 10 cases were β and γ were drawn from (0,1], but all other parameters remain the same.

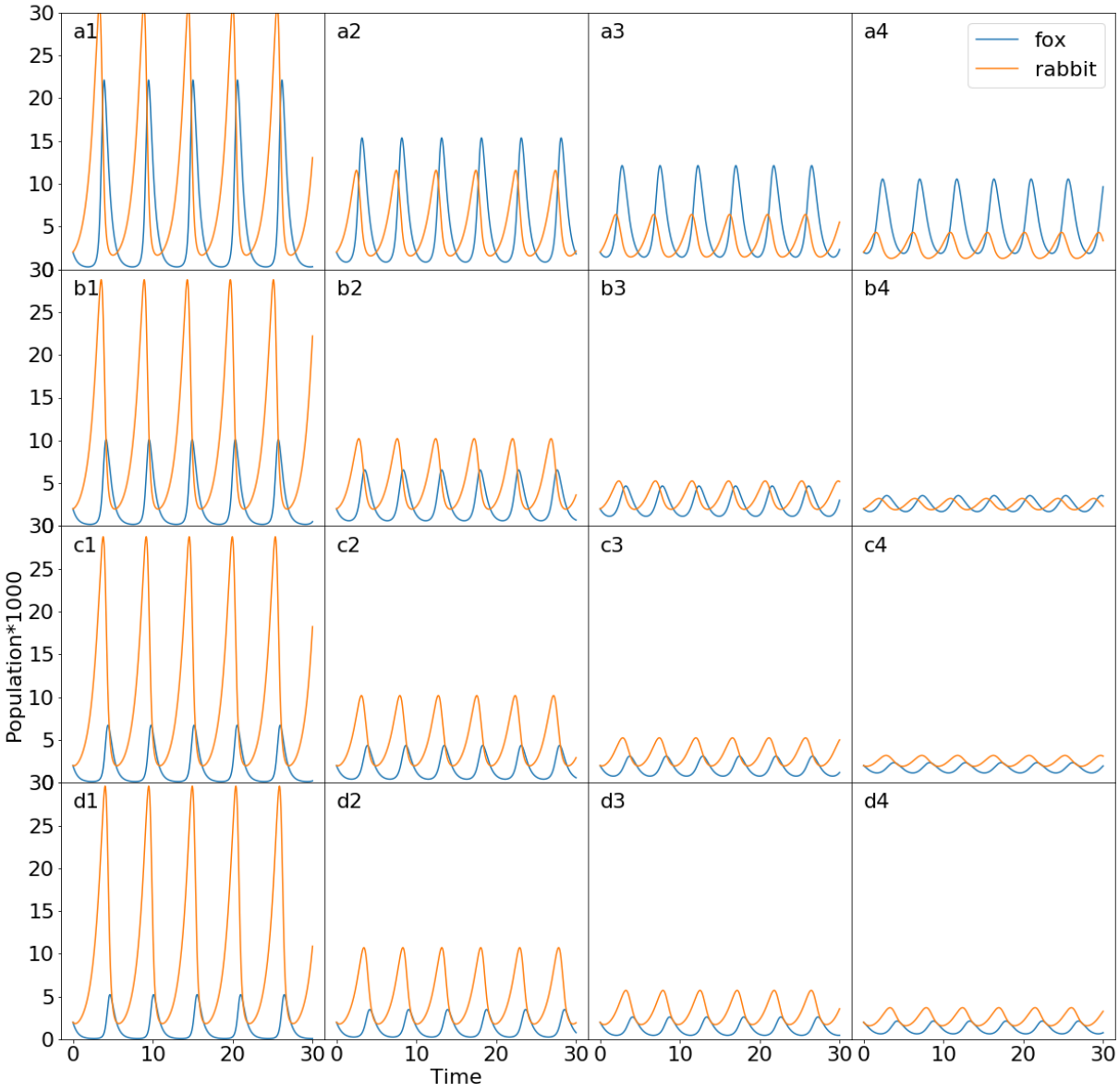

Figure 7. Varying parameters β and γ

Figure 7 shows 16 cases of the model in which β and γ are varied. The graphs are noted as a1, a2, …, d3, d4 which correspond to values of β and γ. Each row a,b,c,d has β values of 0.2, 0.4, 0.6, and 0.8. And, each column 1, 2, 3, 4 has γ values of 0.2, 0.4, 0.6, and 0.8 respectively.

All the outputs in the results show that the fox and rabbit populations oscillate without dampening. Furthermore, the fox population oscillation lags behind the rabbit oscillations in all cases.

The trends from Figure 7 show that as γ (the fox growth rate parameter) increases from left to right columns, the maximum populations of both rabbits and foxes' trend downward. Essentially, if fox populations grow too rapidly in response to rabbit population, then both populations have lower numbers. This might explain why, in nature, predator species are evolutionarily disadvantaged if they grow too rapidly in number. On the other hand, if the fox growth rate is very low (column 1 Fig. 7.) the minimum population of foxes approaches zero, which would be bad for the survival of fox species.

The trends also show that as β (the rate at which foxes meet and kill prey) increases from top to bottom rows, the fox population trends downwards. Essentially, if fox populations have a high impact on prey population, then the fox population decreases overall. Interestingly, the rabbit population curves do not seem to be affected much by the increase in β. Perhaps this strange is due to the range being explored for values of β.

There also seems to be interesting case where a high γ (the fox growth rate parameter) results in a higher number of foxes than rabbits (see Figure 8). This resembles a parasite-host model, where the parasites grow more rapidly and in larger number than hosts.

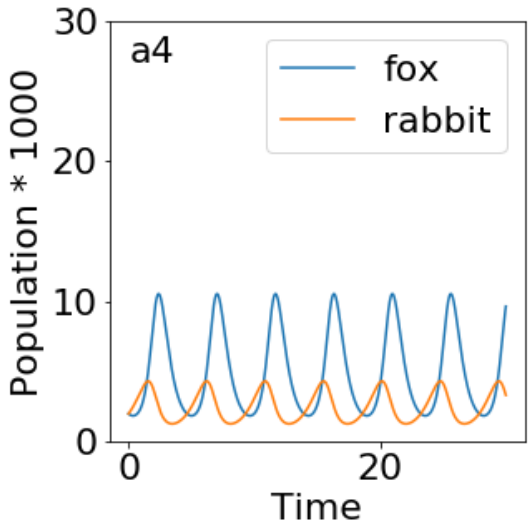

Figure 8.

6. Analysis

In summary, the results of the program show a cyclical interaction between predator and prey populations from the Lotka-Volterra model, one of the simplest and earliest predator-prey models. All results, besides the ones at the critical points, show that the population never approaches an equilibrium and oscillates forever without dampening. The predator populations lag behind the prey populations. Finally, changing the parameters affects in the populations in ways that are difficult to explain. For example, increasing the rate of encounters between predator and prey does not seem to affect the prey population curve much, and causes the prey population to actually decrease overall. Perhaps the parameters are entangled in ways that are not obvious or the range of parameters we have explored are not optimal to see the effects. The results show the complexity of a simple model, and how the parameters affect the population behaviors.

7. Critique

I really solidified my understanding of the Runge-Kutta method. And, I also found it helpful that I did a project on population modeling, which helped inform my honors research project on cancer cell population modeling. I think this project could have been improved, by introducing experimental data, modifications to the Lotka-Volterra model, and other predator-prey models.

8. Sources

[1] Hoppensteadt, F., Predator-Prey Model. (2006) *Scholarpedia, 1*(10), 1563.
[2] Krebs, C.J., Boonstra, R., Boutin, S., Sinclair, A.R.E. (2001) What Drives the 10-year Cycle of Snowshoe Hares?: The ten-year cycle of snowshoe hares—one of the most striking features of the boreal forest— is a product of the interaction between predation and food supplies, as large-scale experiments in the yukon have demonstrated, *BioScience*, *51*(1), 25–35.
[3] Stevens, A. (2010) Dynamics of Predation. *Nature Education Knowledge* 3(10):46
[4] Beals, M., Gross, L., Harrell, S. (1999). PREDATOR-PREY DYNAMICS: LOTKA-VOLTERRA. Retrieved from http://www.tiem.utk.edu/~gross/bioed/bealsmodules/predator-prey.html

9. Appendix

```
import numpy as np
from matplotlib import pyplot as plt, rcParams
import math
from scipy.integrate import odeint
from mpl_toolkits import mplot3d
%matplotlib inline

#Predator Prey Model
def dxdt(x,y,a,b):
    return a*x-b*x*y

def dydt(x,y,g,d):
    return g*x*y-d*y


#Parameters of the model

a=1.0  #alpha
b=0.1  #beta
g=0.1  #gamma
d=2.0  #delta


#Since we are looking at different cases of the model we have
#Different values for beta and gamma that we are plotting
bs=[0.2,0.4,0.6,0.8]
gs=[0.2,0.4,0.6,0.8]

N=10000 #number of steps in integration
t1=0.0  #initial time
t2=30.0 #final time
t=np.linspace(t1,t2,N)
h=(t2-t1)/N  #time step

xs=np.zeros((len(bs)*len(gs),N))
ys=np.zeros((len(bs)*len(gs),N))

####Performing Runge-Kutta RK4 Integrating
####Generating plots for different cases of beta and gamma

count=0

for i in range(len(bs)):
    b=bs[i]
    for k in range(len(gs)):
        g=gs[k]

        x0=2.0
        y0=2.0

        x=np.zeros(N)
        y=np.zeros(N)

```

```
        x[0]=x0
        y[0]=y0

        for i in range(1,N):
            xi=x[i-1]
            yi=y[i-1]

            k1x=h*dxdt(xi,yi,a,b)
            k1y=h*dydt(xi,yi,g,d)


            k2x=h*dxdt(xi+k1x/2,yi+k1y/2,a,b)
            k2y=h*dydt(xi+k1x/2,yi+k1y/2,g,d)

            k3x=h*dxdt(xi+k2x/2,yi+k2y/2,a,b)
            k3y=h*dydt(xi+k2x/2,yi+k2y/2,g,d)

            k4x=h*dxdt(xi+k3x,yi+k3y,a,b)
            k4y=h*dydt(xi+k3x,yi+k3y,g,d)

            x[i]=xi+1.0/6.0*(k1x + 2*k2x  + 2*k3x + k4x)
            y[i]=yi+1.0/6.0*(k1y + 2*k2y  + 2*k3y + k4y)

        xs[count]=x
        ys[count]=y

        count=count+1

####Plotting all 16 cases of beta and gamma


fig = plt.figure(figsize=(20,20))
rcParams.update({'font.size': 22})

# Condensing the plots
fig.subplots_adjust(hspace=0.0, wspace=0.0)

ax1 = fig.add_subplot(441)
ax1.plot(t,ys[0])
ax1.plot(t,xs[0])
ax1.text(0,27,"a1")

ax2 = fig.add_subplot(442)
ax2.plot(t,ys[1])
ax2.plot(t,xs[1])
ax2.text(0,27,"a2")

      ax3 = fig.add_subplot(443)
      ax3.plot(t,ys[2])
      ax3.plot(t,xs[2])
      ax3.text(0,27,"a3")

      ax4 = fig.add_subplot(444)
      ax4.plot(t,ys[3],label='fox')
      ax4.plot(t,xs[3],label='rabbit')
      ax4.text(0,27,"a4")
      ax4.legend()

      ax5 = fig.add_subplot(445)
      ax5.plot(t,ys[4])
      ax5.plot(t,xs[4])
```

```
115.        ax5.text(0,27,"b1")
116.
117.        ax6 = fig.add_subplot(446)
118.        ax6.plot(t,ys[5])
119.        ax6.plot(t,xs[5])
120.        ax6.text(0,27,"b2")
121.
122.        ax7 = fig.add_subplot(447)
123.        ax7.plot(t,ys[6])
124.        ax7.plot(t,xs[6])
125.        ax7.text(0,27,"b3")
126.
127.        ax8 = fig.add_subplot(448)
128.        ax8.plot(t,ys[7])
129.        ax8.plot(t,xs[7])
130.        ax8.text(0,27,"b4")
131.
132.        ax9 = fig.add_subplot(449)
133.        ax9.plot(t,ys[8])
134.        ax9.plot(t,xs[8])
135.        ax9.text(0,27,"c1")
136.        ax9.set_ylabel("Population*1000")
137.
138.        ax10 = fig.add_subplot(4,4,10)
139.        ax10.plot(t,ys[9])
140.        ax10.plot(t,xs[9])
141.        ax10.text(0,27,"c2")
142.
143.        ax11 = fig.add_subplot(4,4,11)
144.        ax11.plot(t,ys[10])
145.        ax11.plot(t,xs[10])
146.        ax11.text(0,27,"c3")
147.
148.        ax12 = fig.add_subplot(4,4,12)
149.        ax12.plot(t,ys[11])
150.        ax12.plot(t,xs[11])
151.        ax12.text(0,27,"c4")
152.
153.        ax13 = fig.add_subplot(4,4,13)
154.        ax13.plot(t,ys[12])
155.        ax13.plot(t,xs[12])
156.        ax13.text(0,27,"d1")
157.
158.        ax14 = fig.add_subplot(4,4,14)
159.        ax14.plot(t,ys[13])
160.        ax14.plot(t,xs[13])
161.        ax14.text(0,27,"d2")
162.        ax14.set_xlabel("Time")
163.
164.        ax15 = fig.add_subplot(4,4,15)
165.        ax15.plot(t,ys[14])
166.        ax15.plot(t,xs[14])
167.        ax15.text(0,27,"d3")
168.
169.        ax16 = fig.add_subplot(4,4,16)
170.        ax16.plot(t,ys[15])
171.        ax16.plot(t,xs[15])
172.        ax16.text(0,27,"d4")
173.
174.        # Removing unneeded tick axis
175.
```

```
176.        ax2.set_yticks([])
177.        ax3.set_yticks([])
178.        ax4.set_yticks([])
179.        ax6.set_yticks([])
180.        ax7.set_yticks([])
181.        ax8.set_yticks([])
182.        ax10.set_yticks([])
183.        ax11.set_yticks([])
184.        ax12.set_yticks([])
185.        ax14.set_yticks([])
186.        ax15.set_yticks([])
187.        ax16.set_yticks([])
188.
189.        ax1.set_ylim(0,30)
190.        ax2.set_ylim(0,30)
191.        ax3.set_ylim(0,30)
192.        ax4.set_ylim(0,30)
193.        ax5.set_ylim(0,30)
194.        ax6.set_ylim(0,30)
195.        ax7.set_ylim(0,30)
196.        ax8.set_ylim(0,30)
197.        ax9.set_ylim(0,30)
198.        ax10.set_ylim(0,30)
199.        ax11.set_ylim(0,30)
200.        ax12.set_ylim(0,30)
201.        ax13.set_ylim(0,30)
202.        ax14.set_ylim(0,30)
203.        ax15.set_ylim(0,30)
204.        ax16.set_ylim(0,30)
```