AppliedW6

March 27, 2024

1 Question 1

Let x be the amount of fish, and y denote sharks. In the original Lotka-Volterra we had:

$$\frac{dx}{dt} = (r - f)x - \alpha xy$$
$$\frac{dy}{dt} = (s - f)y + \beta xy$$

where the constant f, corresponds to the fishing rate; r and s are the rate of growth for fish and shark respectively; α is the prop. constant of fish being eaten by shark, and β is the prop. constant of shark surviving by eating fish.

We are concerned with the rate of growth of fish in isolation, so we need only modify the expression for change in x:

$$\frac{dx}{dt} = (r - f)x - \alpha xy$$

can be re-written as:

$$\frac{dx}{dt} = \underline{rx} - fx - \alpha xy$$

The underlined quantity refers to unconstrained linear growth.

$$\frac{dx}{dt} = R(x) - fx - \alpha xy$$

with $R(x) = rx$.

For logistic growth we introduce a carrying capacity K, and

$$R(x) = rx\Big(1 - \frac{x}{K}\Big)$$

In addition, we remove the interaction term. Thus, we obtain:

$$\frac{dx}{dt} = rx\Big(1 - \frac{x}{K}\Big) - fx - \alpha xy$$

The change in y remains the same.

For the steady state, we set $\frac{dx}{dt} = 0$ and $\frac{dy}{dt} = 0$ and obtain:

$rx(1 - \frac{x}{K}) - fx - \alpha xy = 0$ and $(s - f)y + \beta xy = 0$

Let us start by solving for x using the second equation.

$(s-f)y = -\beta xy$

$x^* = \frac{f-s}{\beta}$

for the sake of convenience when solving the first equation let us call $\gamma = \frac{f-s}{\beta}$

substituting this into the equation from $\frac{dx}{dt} = 0$ we get:

$r\gamma(1-\frac{\gamma}{K}) - f\gamma - \alpha\gamma y = 0$

cancelling the γs and solving for y we get

$r(1-\frac{\gamma}{K}) - f - \alpha y = 0$

$r(1-\frac{\gamma}{K}) - f = \alpha y$

$y = \frac{r(1-\frac{\gamma}{K})-f}{\alpha}$

$y^* = \frac{r(1-\frac{f-s}{\beta K})-f}{\alpha}$

for a steady state of $(x^*, y^*) = (\frac{f-s}{\beta}, \frac{r(1-\frac{f-s}{\beta K})-f}{\alpha})$

[1]:
```
import numpy as np
from matplotlib import pyplot as plt
```

1.1 Modified Lotka Volterra

[69]:
```
def generate_LV(r, s, f, K, alpha, beta):
    dxdt = lambda x: r*x[0]*(1 - x[0]/K) - f*x[0] - alpha*x[0]*x[1]
    dydt = lambda x: (s-f)*x[1] + beta*x[0]*x[1]

    ddt = lambda t, x: np.array([dxdt(x), dydt(x)])

    return ddt
```

[70]:
```
def steady_state(r, s, f, K, alpha, beta):
    x = (f-s)/beta
    y = (r*(1-(f-s)/(beta*K))/alpha)
    return x, y
```

[71]:
```
r = 0.8
s = -0.8
f = 0.1
alpha = 0.045
beta = 0.03
K=1000
```

[72]:
```
LV = generate_LV(r=r, s=s, f=f, K=K, alpha=alpha, beta=beta)
```

[73]:
```
steady_state(r=r, s=s, f=f, K=K, alpha=alpha, beta=beta)
```

[73]:
```
(30.000000000000004, 17.244444444444447)
```

1.2 Euler's (Not necessary for this week but included for completeness)

[74]:
```
h = 0.01
x0 = np.array([50, 10])
t0 = 0
```

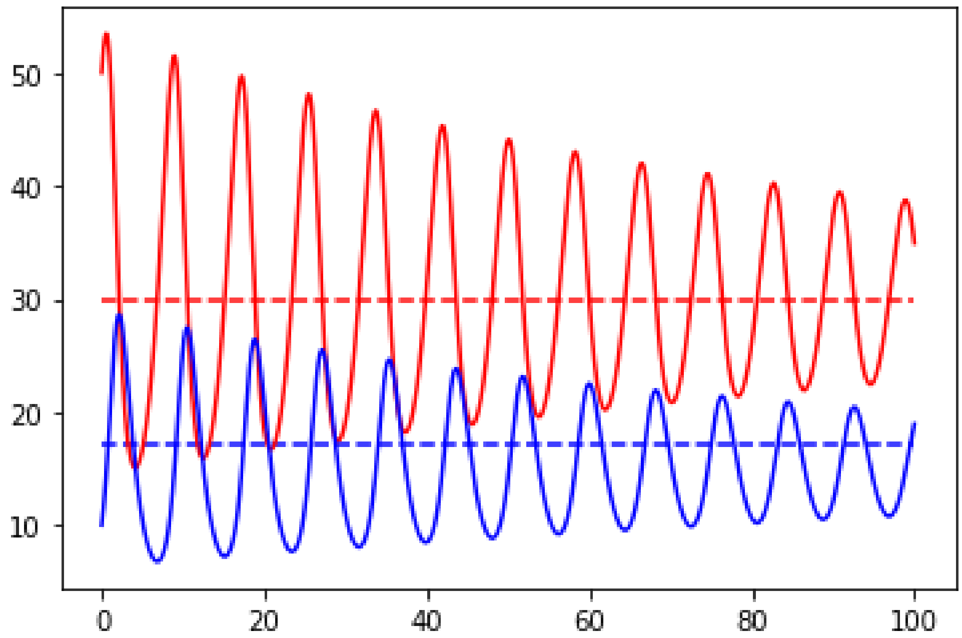
[75]:
```
def euler_next(t, x, h, df):
    return t+h, x + h*df(t, x)

def euler(df, x0, t0=0, h=0.1, max_iter=1000):
    T = np.zeros(max_iter+1)
    X = np.zeros(shape=(len(x0), max_iter+1))
    T[0] = t0
    X[:, 0] = x0.copy()
    for i in range(max_iter):
        t0, x0 = euler_next(t=t0, x=x0, h=h, df=df)
        T[i+1] = t0
        X[:, i+1] = x0
    return T, X
```

[76]:
```
t, (x, y) = euler(df=LV, x0=x0, t0=t0, h=h, max_iter=10000)
x_star, y_star = steady_state(r=r, s=s, f=f, K=K, alpha=alpha, beta=beta)
```

[77]:
```
plt.plot(t, x, label='fish', c='r')
plt.plot(t, y, label='sharks', c='b')
plt.hlines(x_star, xmin=t[0], xmax=t[-1], colors='r', ls='--')
plt.hlines(y_star, xmin=t[0], xmax=t[-1], colors='b', ls='--')
```

[77]:
```
<matplotlib.collections.LineCollection at 0x7f3e7b77a730>
```



1.3 solve_ivp

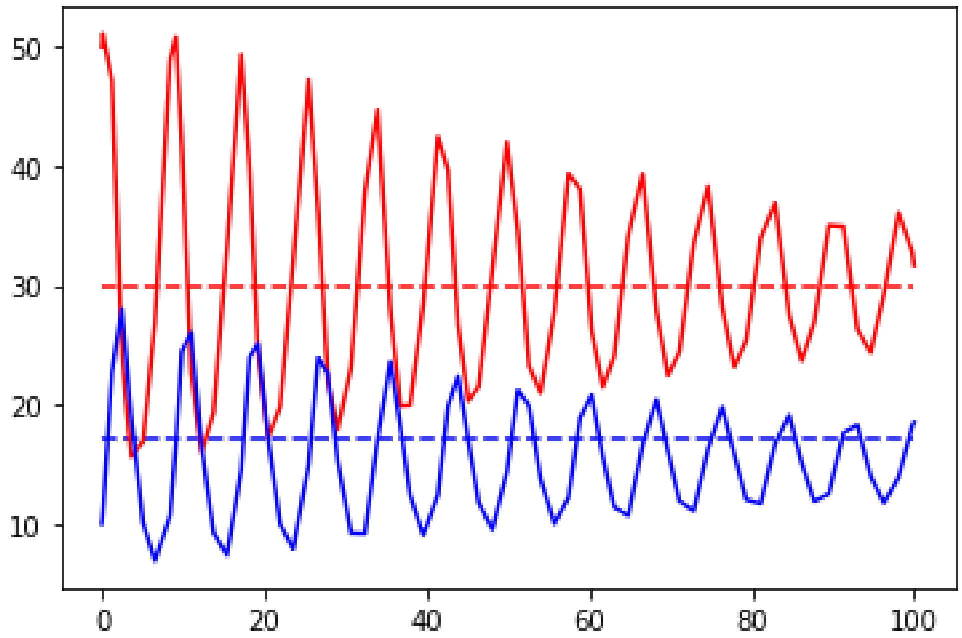
[80]:
```
from scipy.integrate import odeint, solve_ivp
```

[83]:
```
sol = solve_ivp(fun=LV, y0=x0, t_span=(0, 100), method='RK45')
```

[87]:
```
t = sol['t']
x, y = sol['y']
```

[88]:
```
plt.plot(t, x, label='fish', c='r')
plt.plot(t, y, label='sharks', c='b')
plt.hlines(x_star, xmin=t[0], xmax=t[-1], colors='r', ls='--')
plt.hlines(y_star, xmin=t[0], xmax=t[-1], colors='b', ls='--')
```

[88]:
```
<matplotlib.collections.LineCollection at 0x7f3e7b6f0d90>
```



[ ]:

2 Question 2

We use the chain rule:

$$\frac{dy}{dx}\frac{dx}{dt} = \frac{dy}{dt}$$

and obtain:

$$\frac{dy}{dx} \cdot (-y) = x$$

Doing variable separation we get:

$$-ydy = xdx$$

and we integrate on both sides to get:

$$-\frac{y^2}{2} = \frac{x^2}{2} + c$$

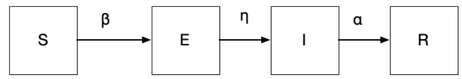
with $c' = 2c$, we get:

$$x^2 + y^2 = c'$$

Thus, the phase-plane trajectories are concentric circles with centre $(0, 0)$.

3 Question 3

Adding the Exposed compartment results in the standard SEIR model.



The diagram is as follows:

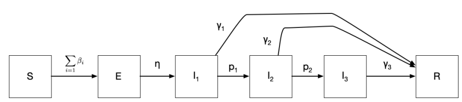
and results in the following system of ODEs:

$$\dot{S} = -\beta S I$$
$$\dot{E} = \beta S I - \eta E$$
$$\dot{I} = \eta E - \alpha I$$
$$\dot{R} = \alpha I$$

Here, η is the per-capita rate of becoming infectious, and $\frac{1}{\eta}$ is approximately the length of the so-called latent period.

Asymptomatic transmission would imply the transition to infection given by the term $\beta S I$ includes (a fraction) of the E compartment. It is generally assumed that asymptomatic individuals are less contagious than those showing symptoms, but this will be disease specific.

For clinical progression in three levels we would have a diagram as follows:



Adding deseased individuals is usually done by splitting the flow into the R compartment, at a certain mortality rate, in the case above for those in the severe stage of the disease.

The simplest way to model interventions is adjusting the parameters. For example, is often assumed to be $\beta = pc$, where p is the probability of transmission and c is the contact rate per capita. A social distancing intervention would bring down the c parameter. Other interventions like Quarantine can be modelled by adding compartments.

4 Question 4

$$\frac{dS}{dt} = \Pi S - \beta S Z - \delta S$$
$$\frac{dZ}{dt} = \beta S Z + \gamma R - \alpha S Z$$
$$\frac{dR}{dt} = \delta S + \alpha S Z - \gamma R$$

This a simple SIR model with a nice story.