3 Modelling the spread of an epidemic: SIR models

Numerical Solvers

We often rely on numerical solvers to numerically integrate ordinary differential equations, ODEs. Because of the complexity of many ODE systems, these numerical solvers allow us to solve complex ODE systems that may not be solvable symbolically, or are high dimensional. In this lab we will be using `solve_ivp`, which is a part of `scipy.integrate`, to solve ODE systems related to epidemic models. You can read the documentation for `solve_ivp` at `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html`.

`solve_ivp` takes the ODE as a function, a tuple containing the start and end time, and an array with the initial conditions as arguments, and returns a bunch object containing the solution and other information. We can solve the following ODE system with the following code.

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}' = \begin{bmatrix} y_2(t) \\ \sin(t) - 5y_2(t) - y_1(t) \end{bmatrix} \qquad (3.1)$$
$$y_1(0) = 0, \quad y_2(0) = 1, \quad t \in [0, 3\pi]$$

```
import numpy as np
from scipy.integrate import solve_ivp

# define the ode system as given in the problem
def ode(t,y):
    return np.array([y[1], np.sin(t) - 5*y[1] - y[0]])

# define the t0 and tf parameters
t0 = 0
tf = 3*np.pi

# define the initial conditions
y0 = np.array([0,1])

# solve the system
sol = solve_ivp(ode, (t0,tf), y0)

# Plot the system
```

```
import matplotlib.pyplot as plt

# plot y_1 against y_2
plt.plot(sol.y[0],sol.y[1])
plt.xlabel('$y_1$')
plt.ylabel('$y_2$')
plt.show()
```

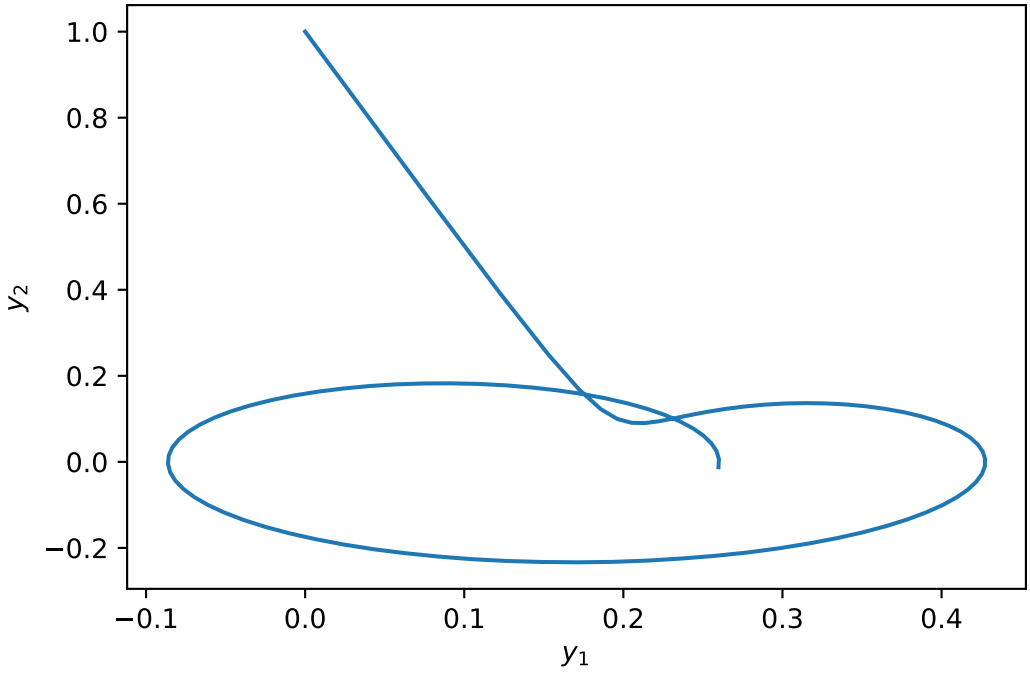

Figure 3.1: Solution to (3.1)

The SIR Model

The SIR model describes the spread of an epidemic through a large population. It does this by describing the movement of the population through three phases of the disease: those individuals who are *susceptible*, those who are *infectious*, and those who have been *removed* from the disease. Those individuals in the removed class have either died, or have recovered from the disease and are now immune to it. If the outbreak occurs over a short period of time, we may reasonably assume that the total population is fixed, so that $S'(t) + I'(t) + R'(t) = 0$. We may also assume that $S(t) + I(t) + R(t) = 1$, so that $S(t)$ represents the *fraction* of the population that is susceptible, etc.

Individuals may move from one class to another as described by the flow

$$S \to I \to R.$$

Let us consider the transition rate between S and I. Let β represent the average number of contacts made per unit time period (one day perhaps) that could spread the disease. The proportion of these contacts that are with a susceptible individual is $S(t)$. Thus, one infectious individual will on average infect $\beta S(t)$ others per day. Let N represent the total population size. Then we obtain the differential equation

$$\frac{d}{dt}(S(t)N) = -\beta S(t)(I(t)N)$$

Now consider the transition rate between I and R. We assume that there is a fixed proportion γ of the infectious group who will recover on a given day, so that

$$\frac{d}{dt}R(t) = \gamma I(t).$$

Note that γ is the reciprocal of the average length of time spent in the infectious phase.

Since the derivatives sum to 0, we have $I'(t) = -S'(t) - R'(t)$, so the differential equations are given by

$$\frac{dS}{dt} = -\beta IS, \tag{3.2}$$
$$\frac{dI}{dt} = \beta IS - \gamma I, \tag{3.3}$$
$$\frac{dR}{dt} = \gamma I. \tag{3.4}$$

Problem 1. Suppose that, in a city of approximately three million, five people who have just become infectious have recently entered the city carrying a certain disease. Each of those individuals has one contact each day that could spread the disease, and an average of three days is spent in the infectious state. Find the solution of the corresponding SIR equations using solve_ivp for fifty days, where each time period is half a day, and plot your results. Use the percentages of each state, not the actual number of people in the state.

At the peak of the infection, how many in the city will still be able to work (assume for simplicity that those who are in the infectious state either cannot go to work or are unproductive, etc.)?

Hint: Use the t-values paramter in solve_ivp to pass in an array of t-values.

Compare your plot to Figure 1.

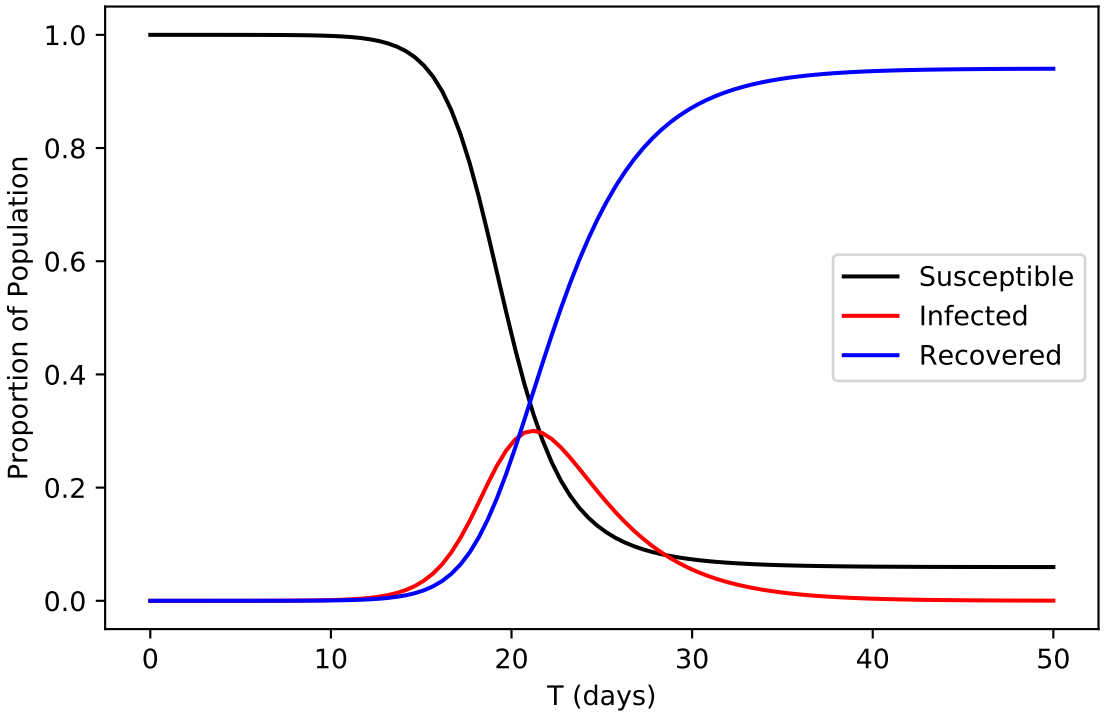

Figure 3.2: Solution to Problem (1)

SIR is an effective model for epidemic spread under certain assumptions. For example, we assume that the network is what's called "fully mixed." This implies that no group of members of a network are more likely to encounter each other than any other group. Because of this assumption, we should not use SIR to model networks we know to be poorly mixed. In fact, we should be clear in stating that almost no network is truly fully mixed; however this model is still effective for networks that are reasonably well mixed. In the next problem we will be using SIR to model data from the recent COVID-19 outbreak. To adhere to the "reasonably well mixed" criteria, we will be using only data from one county at a time.

Problem 2. On March 11, 2020, New York City had 52 confirmed cases of COVID-19. On that day New York started its lock-down measures. Using the following information, model what the spread of the virus could have been, using `solve_ivp()`, if New York did not implement any measures to curb the spread of the virus over the next 150 days:

- There are approximately 8.399 million people in New York city.
- The average case of COVID-19 lasts for 10 days.
- Each infected person can spread the virus to 2.5 people.

Plot your results for each day and compare to Figure 3.3.

1. At the projected peak, how many concurrent active cases are there?
2. Assuming that about 5% of COVID-19 cases require hospitalization, and using the fact that there are about 58,000 hospital beds in NYC, how many beds over capacity will the hospitals in NYC be at the projected peak?

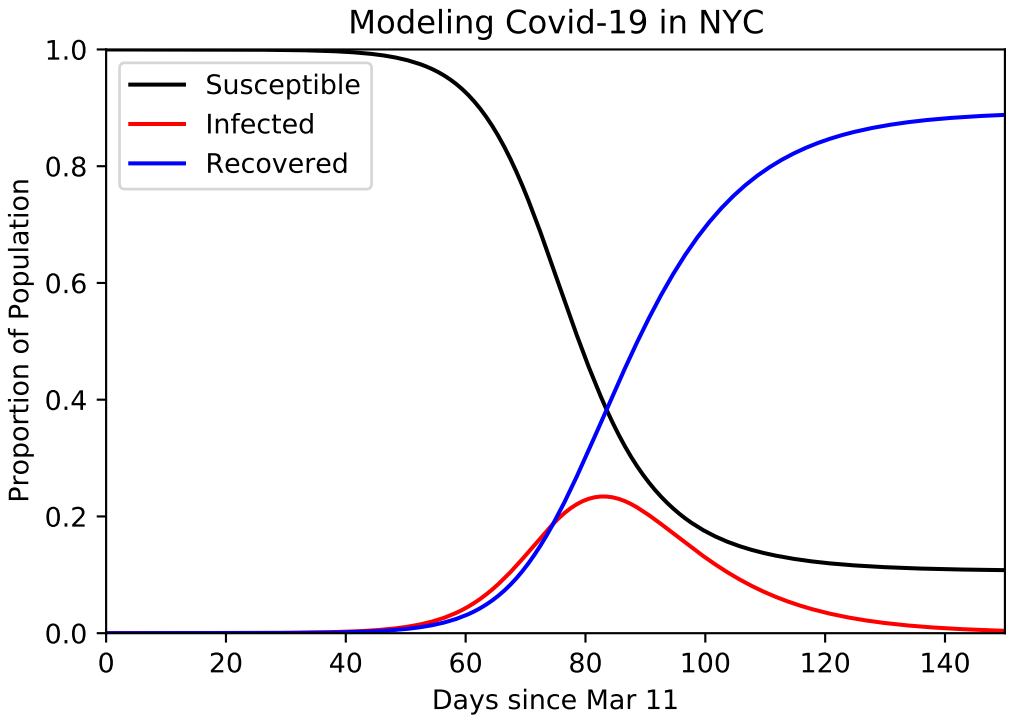

Figure 3.3: Solution to Problem (2).

Variations on the SIR Model

The SIS model is a common variation of the SIR model. SIS Models describe diseases where individuals who have recovered from the disease do not gain any lasting immunity. There are only two compartments in this model: those who are *susceptible*, and those who are *infectious*. Here, f is the rate of becoming susceptible again.

The basic equations are given by

$$\frac{dS}{dt} = -\beta IS + fI,$$
$$\frac{dI}{dt} = \beta IS - fI$$

Another alteration we can make to the SIR model is to add a birth and death rate. In the equations below we are assuming that the natural death rate together with the death rate caused by the disease is equal to the birth rate. This model is given by

$$\begin{aligned}\frac{dS}{dt} &= \mu(1-S) - \beta IS,\\ \frac{dI}{dt} &= \beta IS - (\gamma+\mu)I,\\ \frac{dR}{dt} &= \gamma I - \mu R\end{aligned}$$

where μ represents the death rate and equal birth rate, noting that any new person born is born into the susceptible population.

If we combine the last two variations we made on the SIR model we come to this formulation, which is an SIRS model. This SIRS model allows the transfer of individuals from the recovered/removed class to the susceptible class and includes modeling of the birth and death rates.

$$\frac{dS}{dt} = fR + \mu(1-S) - \beta IS, \tag{3.5}$$
$$\frac{dI}{dt} = \beta IS - (\gamma+\mu)I, \tag{3.6}$$
$$\frac{dR}{dt} = -fR + \gamma I - \mu R. \tag{3.7}$$

Problem 3. There are 7 billion people in the world. Influenza, or the flu, is one of those viruses that everyone can be susceptible to, even after recovering. The flu virus is able to change in order to evade our immune system, and we become susceptible once more, although technically it is now a different strain. Suppose the virus originates with 1000 people in Texas after Hurricane Harvey flooded Houston, and stagnant water allowed the virus to proliferate. According to WebMD, once you get the virus, adults are contagious up to a week and kids up to 2 weeks. For this lab, suppose you are contagious for 10 days before recovering. Also suppose that on average someone makes one contact every two days that could spread the flu. Since we can catch a new strain of the flu, suppose that a recovered individual becomes susceptible again with probability $f = 1/50$. The flu is also known to be deadly, killing hundreds of thousands every year on top of the normal death rate. To assure a steady population, let the birth rate balance out the death rate, and in particular let $\mu = .0001$.

Using the SIRS model above, plot the proportion of population that is Susceptible, Infected, and Recovered over a one-year span (365 days). Compare your plot to Figure 3.4.

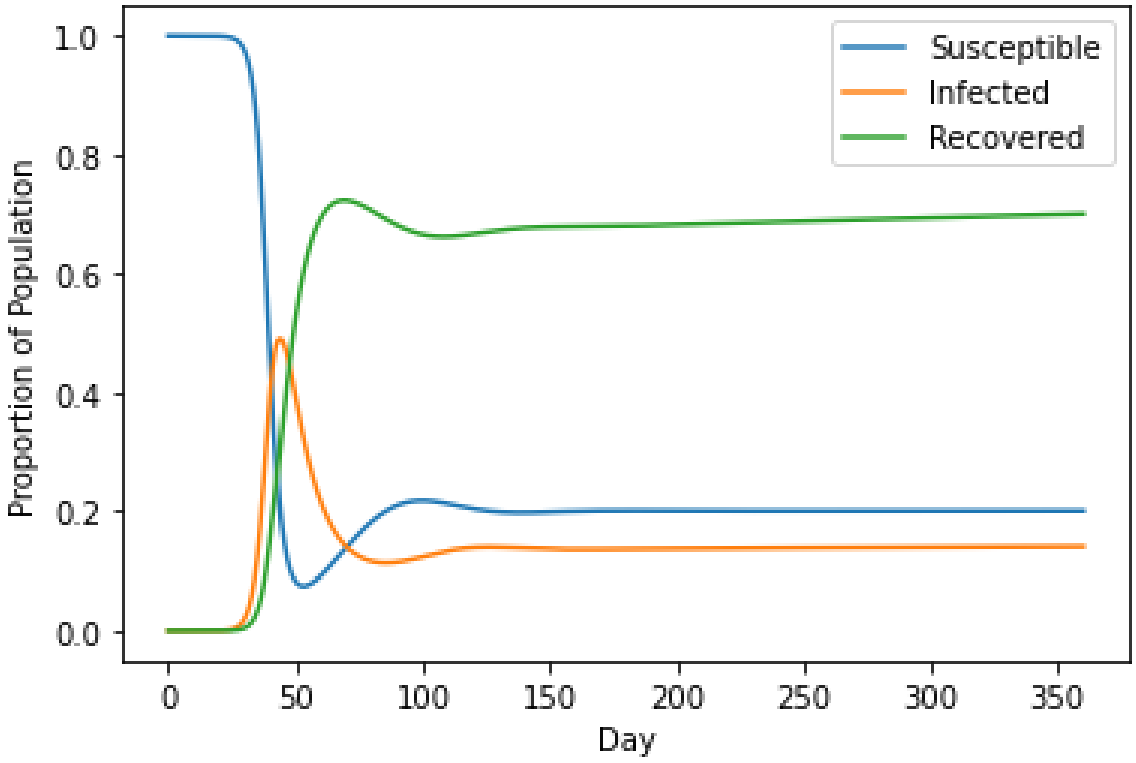

Figure 3.4: Solution to Problem (3).

Modeling COVID-19 with Social Distancing

Social distancing upsets the main assumption that is made when trying to model epidemic spread using SIR models. During the periods of lockdown instituted by governments, the interaction networks between people in a city or county were disrupted to the point that standard SIR models were no longer effective at modeling the spread of COVID-19. A paper released in May of 2020 presented some alternative models for COVID-19 that have some success in modeling its spread during periods of social distancing.

This model claims that the growth of $I(t)$ is polynomial with exponential decay (PGED). So we get the following form

$$I(t) \approx Bt^{\alpha}e^{-t/T_G},$$

which results in the following SIR type model

$$\frac{dS}{dt} = -\frac{\alpha}{t}I, \tag{3.8}$$

$$\frac{dI}{dt} = \left(\frac{\alpha}{t} - \frac{1}{T_G}\right)I, \tag{3.9}$$

$$\frac{dR}{dt} = \frac{1}{T_G}I, \tag{3.10}$$

where α and T_G are simply model parameters. In this model αT_G can be interpreted as the time of epidemic peak.

Fitting Models

Model fitting can be a frustrating task if we only use our intuition and guess and check. Thankfully, SciPy's `optimize` library has tools we can use to make these problems a lot easier. Many of the functions in this library are designed to take an arbitrary function and find whatever input makes the output close to zero. Our job is to create a function that outputs zero at the right values.

Suppose we have some data that we believe to follow a cubic trend with the following model

$$\alpha x^3 + \beta(x^2 + 2x) + \delta.$$

In order to fit the data to this model we can use `scipy.optimize.minimize` and create a function that will output zero when the correct parameters are input. `scipy.optimize.minimize` will then return an `OptimizeResult` object, which contains the optimal paramters.

```
# import the minimizer function
from scipy.optimize import minimize

# load the data and get the x and y values
data = np.load('to_fit.npy')
xs = data[:,0]
ys = data[:,1]

# define the function we want to minimize
def fun(params):
    # unpack the parameters
    a,b,d = params

    # get the model output based on the parameters
    out = a*xs**3 + b*(xs**2 + 2*xs) + d

    # find the difference between out and the data
    diff = out - ys

    # must return a float
    return np.linalg.norm(diff)

# make a guess for the parameters
p0 = (1,1,1)

# find the best parameters for this model
minimize(fun,p0)
```

Problem 4. Fit the PGED model to the COVID-19 data provided in `new_york_cases.npy`. Plot your results against $1 - S(t)$.

Hint: Set $t_0 = 1$ as the PEGD model requires to divide by t, so we must have $t \neq 0$.

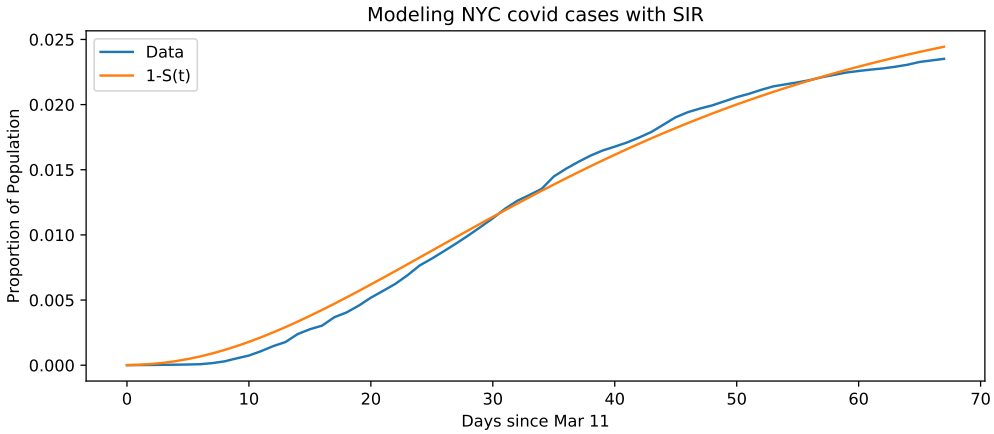

Figure 3.5: Solution to (4)

Boundary Value Problems

The next exercise uses a variation of the SIR model called an SEIR model to describe the spread of measles[1]. This new model adds another compartment, called the *exposed* or *latency* phase. It assumes that the rate at which measles is contracted depends on the season, i.e. the rate is periodic. That allows us to formulate the yearly occurrence rate for measles as a boundary value problem. The boundary value problem looks like

$$\begin{bmatrix} S \\ E \\ I \end{bmatrix}' = \begin{bmatrix} \mu - \beta(t)SI \\ \beta(t)SI - E/\lambda \\ E/\lambda - I/\eta \end{bmatrix}, \tag{3.11}$$

$$\begin{aligned} S(0) &= S(1), \\ E(0) &= E(1), \\ I(0) &= I(1) \end{aligned} \tag{3.12}$$

Parameters μ and λ represent the birth rate of the population and the latency period of measles, respectively. η represents the infectious period before an individual moves from the infectious class to the recovered class. After recovery an individual remains immune, which is why $R(t)$ is not included in the system. The set up of this problem is not normal since we are excluding $R(t)$, but it results in a nice graph.

To solve this problem we will use a full-featured BVP solver that is available in `SciPy`. The code below demonstrates how to use `solve_bvp` to solve the BVP

$$\varepsilon y'' + yy' - y = 0, \quad y(-1) = 1, \quad y(1) = -1/3, \quad \varepsilon = .1 \tag{3.13}$$

Look at figure 3.6 for the solution.

[1]Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, by Aescher, Mattheij, and Russell

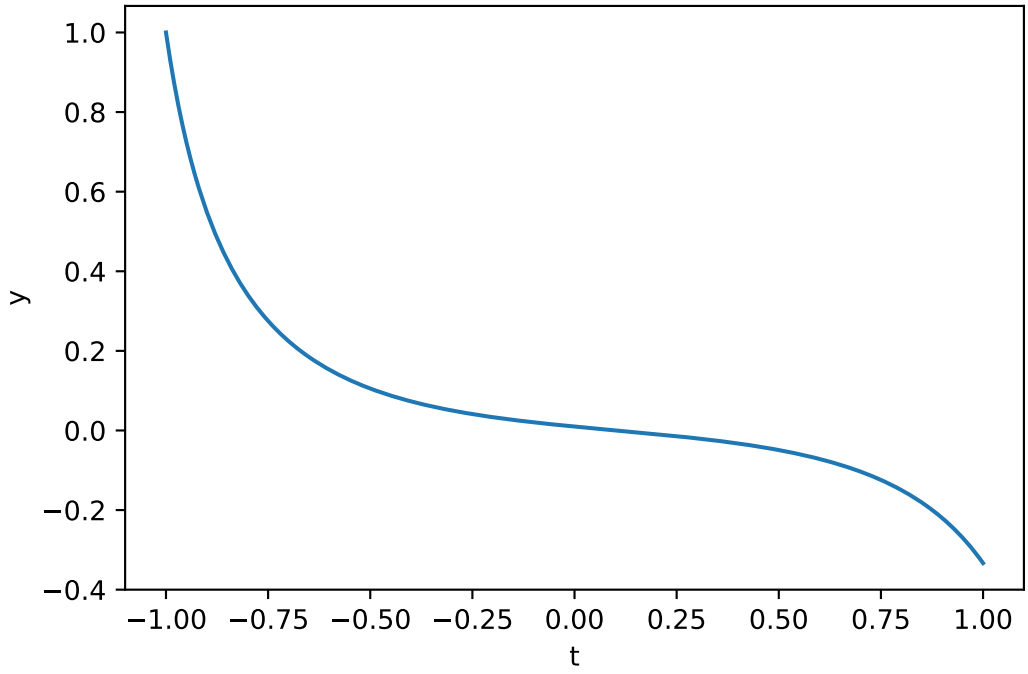

Figure 3.6: Solution to Equation (3.13)

The BVP solver expects you to pass it the boundary conditions as a callable function that computes the difference between a guess at the boundary conditions and the desired boundary conditions. When we use the BVP solver, we will tell it how many constraints there should be on each side of the domain so it knows how many entries to expect. In this case, we have one boundary condition on either side. These constraints are expected to evaluate to 0 when the boundary condition is satisfied.

```
import numpy as np
from scipy.integrate import solve_bvp
import matplotlib.pyplot as plt

epsilon, lbc, rbc = .1, 1, - 1/3

# The ode function takes the independent variable first
# It has return shape (n,)
def ode(x , y):
    return np.array([y[1] , (1/epsilon) * (y[0] - y[0] * y[1])])

# The return shape of bcs() is (n,)
def bcs(ya, yb):
    BCa = np.array([ya[0] - lbc])   # 1 Boundary condition on the left
    BCb = np.array([yb[0] - rbc])   # 1 Boundary condition on the right
    # The return values will be 0s when the boundary conditions are met exactly
    return np.hstack([BCa, BCb])
```

```
# The independent variable has size (m,) and goes from a to b with some step size
X = np.linspace(-1, 1, 200)
# The y input must have shape (n,m) and includes our initial guess for the boundaries
y = np.array([-1/3, -4/3]).reshape((-1,1))*np.ones((2, len(X)))

# There are multiple returns from solve_bvp(). We are interested in the y values which can be found in the sol field.
solution = solve_bvp(ode, bcs, X, y)
# We are interested in only y, not y', which is found in the first row of sol.
y_plot = solution.sol(X)[0]

plt.plot(X, y_plot)
plt.xlabel('t')
plt.ylabel('y')
plt.show()
```

Problem 5. Consider equations (3.11) and (3.12). Let the periodic function for our measles case be $\beta(t) = \beta_0(1 + \beta_1 \cos 2\pi t)$. Use parameters $\beta_1 = 1$, $\beta_0 = 1575$, $\eta = 0.01$, $\lambda = .0279$, and $\mu = .02$. Note: in this case, time is measured in years, so run the solution over the interval $[0, 1]$ to show a one-year cycle. The boundary conditions in (3.12) are just saying that the year will begin and end in the same state.

One issue that we encounter with this problem is that we have 6 boundary conditions but we only have 3 free variables. The 6 boundary counditions are the initial and final conditions of S, E, and I. `solve_bvp` only allows as many boundary conditions as there are free variables, so what we can do is include "dummy" variables in the ODE. This allows more boundary conditions in the BVP solver, while not changing the ODE system that we are solving. To deal with this, let $C(t) = [C_1(t), C_2(t), C_3(t)]$, and add the equation

$$C'(t) = 0$$

to the system of ODEs given above (for a total of 6 equations) resulting in this final 6 variable system

$$\begin{bmatrix} S(t) \\ E(t) \\ I(t) \\ C_1 \\ C_2 \\ C_3 \end{bmatrix}' = \begin{bmatrix} \mu - \beta(t)SI \\ \beta(t)SI - E/\lambda \\ E/\lambda - I/\eta \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

We can then apply all 6 of the boundary conditions that we need. The boundary conditions can be separated using the following trick:

$$\begin{pmatrix} C_1(0) \\ C_2(0) \\ C_3(0) \end{pmatrix} = \begin{pmatrix} S(0) \\ E(0) \\ I(0) \end{pmatrix}, \quad \begin{pmatrix} C_1(1) \\ C_2(1) \\ C_3(1) \end{pmatrix} = \begin{pmatrix} S(1) \\ E(1) \\ I(1) \end{pmatrix}.$$

Now C_1, C_2, C_3 become the 4th, 5th, and 6th rows of your solution matrix, so the 3 boundary conditions for the left are obtained by subtracting the last three entries of $y(0)$ from the first three entries, giving you $ya[0:3] - ya[3:]$. Similarly, your right boundary conditions will look like $yb[0:3] - yb[3:]$.

When you code your boundary conditions, note that `solve_bvp` changes the initial conditions to force all the entries in the return of `bcs()` to be zero. You can use the initial conditions from Fig. 3.7 as your initial guess (which will be an array of 6 elements). Remember that the initial infected proportion is small, not 0.

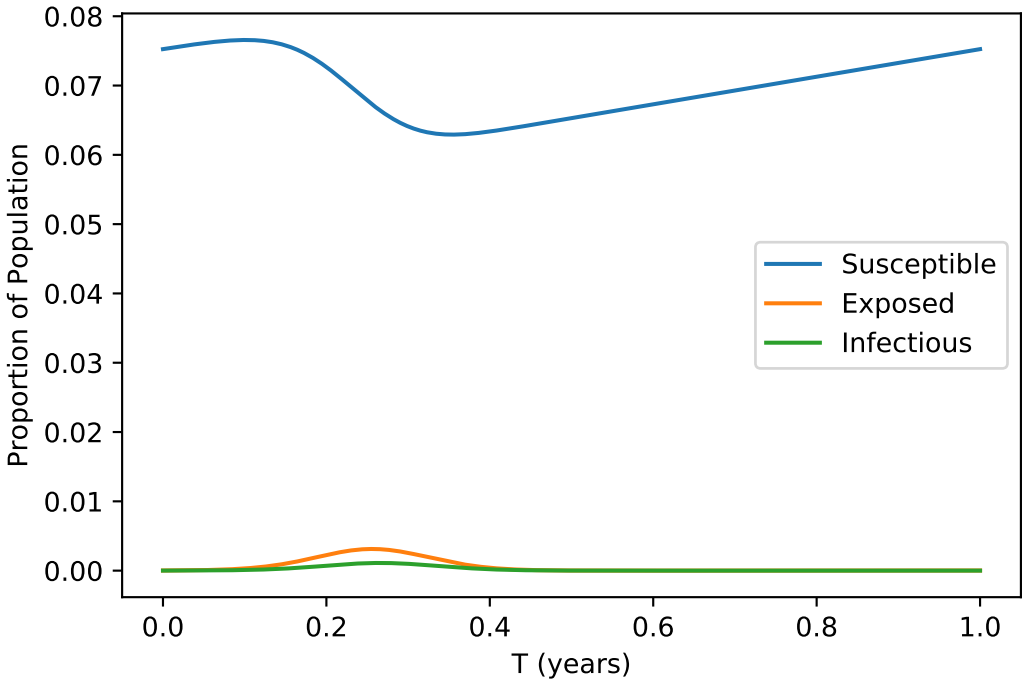

Figure 3.7: Solution to Problem (5)