Module 4:

RENAME THE FILE TO INCLUDE YOUR COMPANY, GROUP NUMBER, AND LAST NAMES

**E.G. Elmqivst_Lee_Shee_MODULE4

Team Members:

Luke Shee, Christy Lee (Group 5, or 4 based on Canvas)

Project Title:

Utilizing differential equations to analyze the number of COVID cases in China

Project Goal:

The goal of the project is to model COVID in China utilizing a SIR model

Disease Background:

Using your assigned disease, fill in the following bullet points.

- Prevalence & incidence
  - Prevalence:
    - Across the world, around 22 thousand COVID-19 cases were reported (though not confirmed) to the World Health Organization (WHO) over the past week, not including data from the U.S. or China. Cumulatively, there has been 778.9 million COVID-19 cases reported to the WHO globally as of November 11, 2025.
      - Source: https://data.who.int/dashboards/covid19/cases
    - Up to April 13, 2024, China had a total of around 503 thousand COVID-19 cases according to worldometers.
      - Source: https://www.worldometers.info/coronavirus/country/china/
    - As of today (December 14, 2025), there has been 99.4 million COVID-19 cases from China reported to WHO.
      - https://data.who.int/dashboards/covid19/cases?n=c
  - Incidence:
    - On April 13, 2024, China had approximately 119 thousand active COVID-19 cases

        - Source: https://www.worldometers.info/coronavirus/country/china/
    - From WHO data, there's currently 0 daily reported COVID-19 cases from China, which may not reflect the actual infection rate in China given delays, limited testing, and other obstacles in confirming cases. At its peak, there was over 4000 new cases reported per million of people in China on December 27, 2022.
        - Source: https://ourworldindata.org/coronavirus/country/china
- Economic burden
    - There is an estimated cost of $671.4 per patient
    - some studies reveal that direct medical costs ranged from $1264-79315$
    - the total ecnomic burden of COVID was estimated to be between $77 billion-2.7 trillion in 2019
    - the quarantine costs of COVID exceeded 9% of the global GDP
        - Source: https://pmc.ncbi.nlm.nih.gov/articles/PMC10870589/
- Risk factors (genetic, lifestyle) & Societal determinants
    - people ages 65 and older and babies younger than 6 months have a higher risk of serious COVID infection and likelihood of requiring hospital care
    - babies younger than 6 months aren't eligible for the COVID vaccine, increasing their risk
    - People with heart disease, diabetes, chronic lung diseases, obseity, and chronic kidney diseases are more likely to have severe cases of COVID
    - The risk of having COVID is higher for people with lung diseases such as COPD and Asthma
    - Having the COVID vaccine will lower risks of severe COVID
    - regularly washing hands throughout the day, wearing a mask, avoiding close contact with anyone that is sick or is showing symptoms will lower risk of getting COVID
        - Source: https://www.mayoclinic.org/diseases-conditions/coronavirus/in-depth/coronavirus-who-is-at-risk/art-20483301
- Symptoms
    - fever or chills
    - cough
    - shortness of breath or difficulty breathing
    - sore throat
    - congestion or runny nose
    - loss of taste or smell
    - fatigue
    - muscle or body aches
    - headache
    - nausea or vomiting

    - diarrhea
        - Source: https://www.cdc.gov/covid/signs-symptoms/index.html
- Diagnosis
    - There are 2 main types of tests to help diagnose COVID
        - molecular tests: look for genetic material from the COVID virus.
            - polymerase chain reaction tests are molecular tests. PCR tests are more accurate than an antigen test
            - molecular tests can be done at home
        - Antigen tests: look for viral proteins called antigens
            - this is typically a rapid COVID test or an at-home COVID test
        - Source: https://www.mayoclinic.org/diseases-conditions/coronavirus/diagnosis-treatment/drc-20479976
- Biological mechanisms (anatomy, organ physiology, cell & molecular physiology)
    - SARS-CoV-2 entry into the cell involces the binding of the spike protein on the surface of COVID onto a receptor on the outer membrane of the cell, angiotensin-converting enzyme 2 (ACE2), causing membrane fusion
        - Source: https://www.nature.com/articles/s41580-021-00418-x

Dataset:

- The data listing confirmed COVID-19 cases per day in several countries from 2020 to 2023 was retrieved from the following GitHub page: https://github.com/CSSEGISandData/COVID-19
    - The data was gathered and managed by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE), along with the assistance of the John Hopkins Applied Physics lab and ESRI Living Atlas Team. The university center initially collected this data into the above repository to create a visual dashboard application to track and display COVID cases globally over time.
        - John Hopkins University Center gathered data on the quantity of COVID-19 cases from a great number of sources, including (but not limited to) the Chinese online health platform DXY (https://ncov.dxy.cn/ncovh5/view/pneumonia), the World Health Organization (https://www.who.int/), the US CDC (https://www.cdc.gov/coronavirus/2019-ncov/index.html), WorldoMeters (https://www.worldometers.info/coronavirus/), The Covid Tracking Project (https://covidtracking.com/data), online news sources like LA Times and Mercury News, and several public government records from U.S. State Departments and other countries worldwide.

    - The project was later announced and reported on in a journal article authored by Ensheng Dong, Hongru Du, and Lauren Gardner: https://pubmed.ncbi.nlm.nih.gov/32087114/
  - The data was later subsetted by Prof. Groves to only include entries with dates ranging from 2022-02-22 to 2022-05-10 and the number of confirmed cases each day between that time frame
- Concerning bias, the authors, who first reported the visual dashboard application and its corresponding datasets of COVID-19 cases, declared no competing interests, though the team was financially supported by the John Hopkins University, National Science Foundation (NSF), Bloomberg Philantrophies, and Stavros Niarchos Foundation. The list of data sources recorded also do not indicate any bias from the collector's perspective.

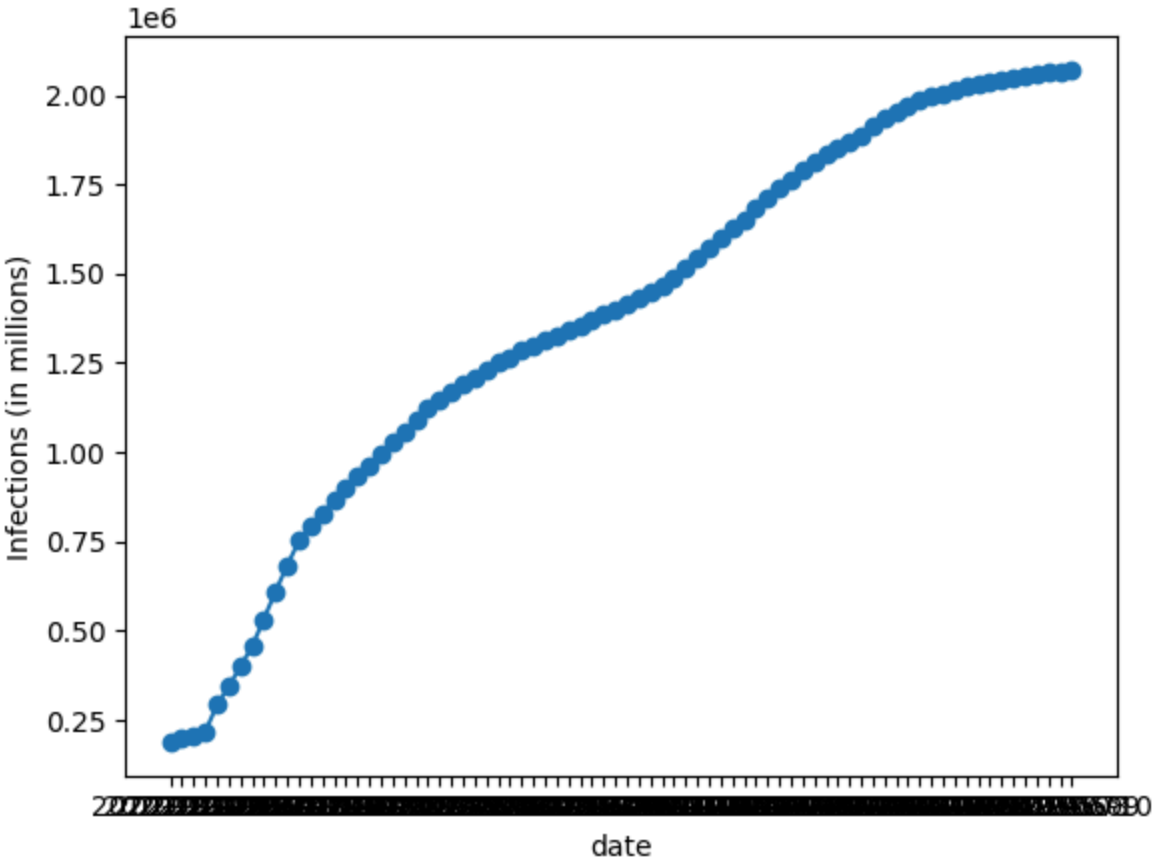
- The sources of COVID-19 confirmed cases, where the current dataset colleced its data from, are assumed to have recorded its COVID-19 data truthfully and as accurately as reasonably possible. Although, some data sources may have been limited in the efficiency in which they can accurately estimate, record, and report COVID-19 cases on a daily basis.

- The dataset utilized by our team involves the number of confirmed cases per day in China over the 2022-02-22 to 2022-05-10 period.

  - There are two columns in the dataset: "date" that holds values signifying certain dates when confirmed COVID-19 cases were measured/updated, and "confirmed_cases" that contains integer values representing the number of registered COVID-19 cases up to the date besides it.
  - Rows represent different, consecutive dates between February 22, 2022, and May 10, 2022.

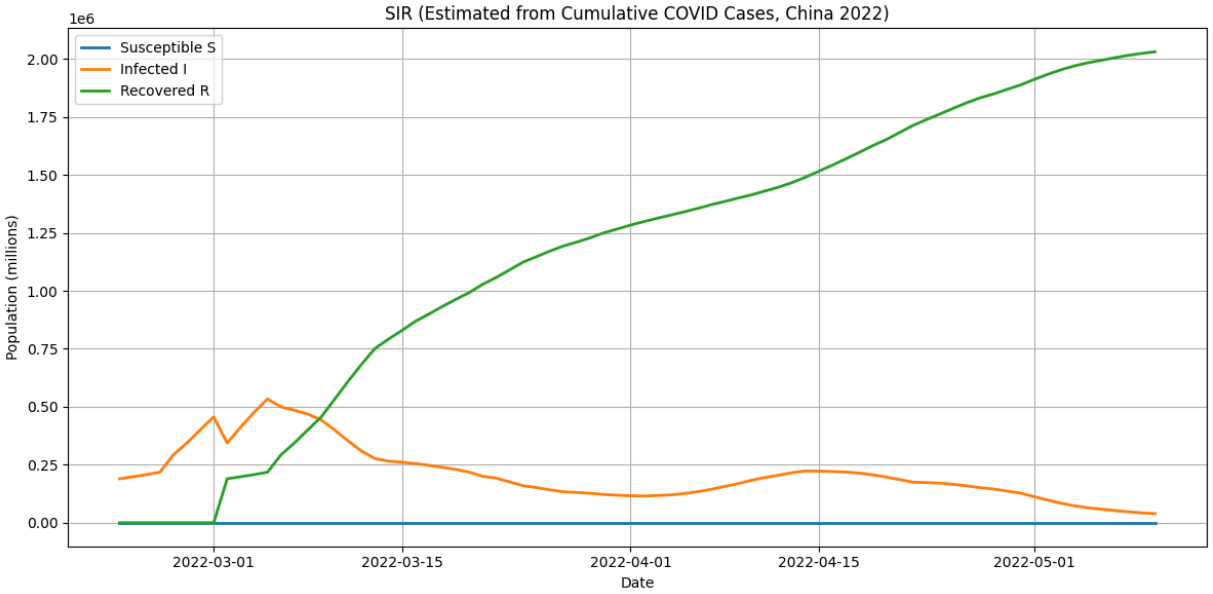
In [ ]:
```
import numpy as np, pandas as pd
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import pandas as pd

# COVID NUMBERS
N = 1410000000/1e6  # US population in millions
df_full = pd.read_csv("covid_china_data_spring_2022_cumulative (1).csv")
df_full['confirmed_cases'] = df_full['confirmed_cases']
# df_full = df_full[(df_full['date'] <= '2021-03-17')&(df_full['date'] >= '2
df_full.head()



plt.plot(df_full['date'], df_full['confirmed_cases'], 'o-')
plt.xlabel('date'); plt.ylabel('Infections (in millions)')
#plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%-m/%-y'))
```

```
#plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=4))
plt.show()
```



In [ ]:
```
from main_functions import convert_cumulative_to_SIR

sir_df = convert_cumulative_to_SIR(
    df_full,
    date_col='date',
    cumulative_col='confirmed_cases',
    population=N,              # your N = population in millions
    infectious_period=8,       # typical COVID infectious period
    new_case_col='new_cases',
    I_col='I',
    R_col='R',
    S_col='S'
)

plt.figure(figsize=(12,6))

plt.plot(sir_df['date'], sir_df['S'], label='Susceptible S', linewidth=2)
plt.plot(sir_df['date'], sir_df['I'], label='Infected I', linewidth=2)
plt.plot(sir_df['date'], sir_df['R'], label='Recovered R', linewidth=2)

plt.xlabel("Date")
plt.ylabel("Population (millions)")
plt.title("SIR (Estimated from Cumulative COVID Cases, China 2022)")
plt.legend()

#plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%-m/%-y'))
```

```
#plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=4))
plt.grid(True)
plt.tight_layout()
plt.show()
```



Data Analyis:

Methods

IN A SUMMARY, DESCRIBE THE METHODS YOU USED TO ANALYZE AND MODEL THE DATA.

Analysis

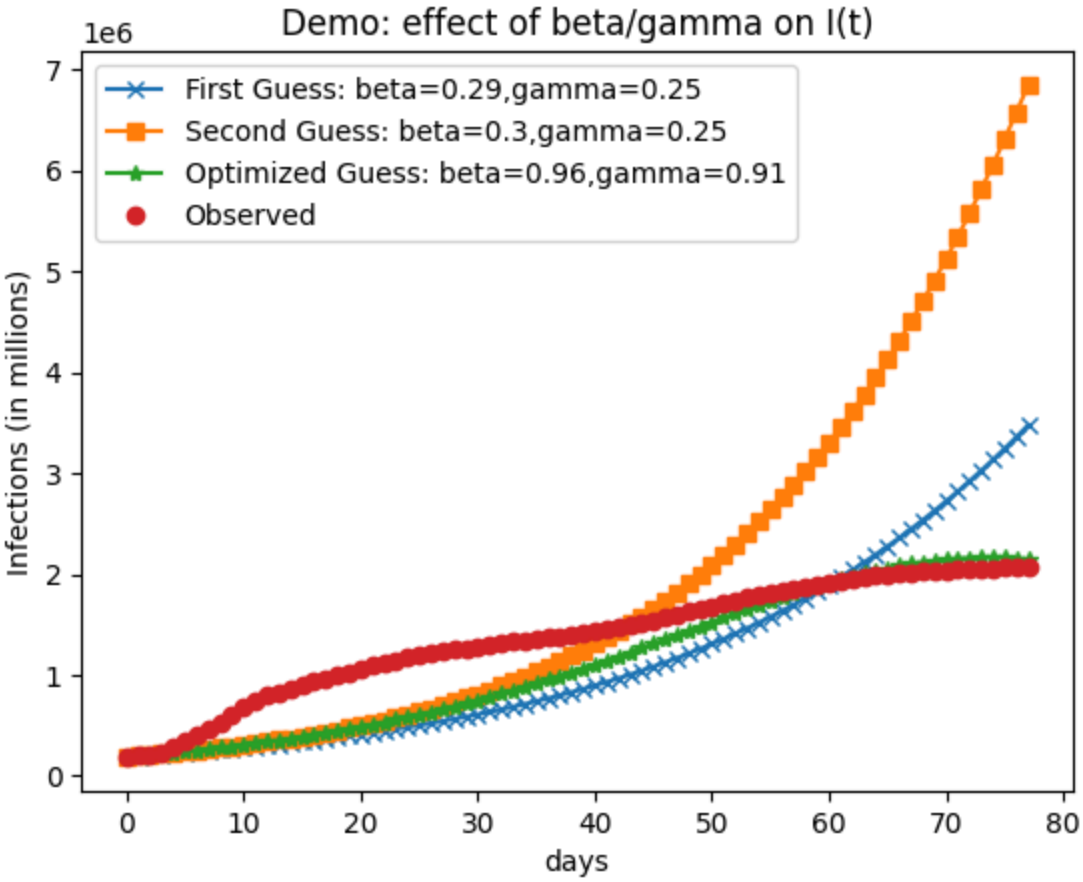
For our Module 4 Jupyter project, we aimed to model COVID in China utilizing a SIR model, attempting to answer the question of "what beta (infectiousness) and gamma (recovery) values for the SIR model give the least sum of squared error (SSE)?" We'll find these best-fitting parameters through multiple methods of training the data (full dataset vs. first half) and approximating the SIR values (Euler's method vs. RK4).

1. Fitting the SIR Model

In [ ]:
```
# Plug in guesses for gamma and beta, plot the model predictions against the
# Use an optimization routine to minimize SSE and find the best-fitting para


# Euler integrator function (already defined above but re-defining for clari
from main_functions import euler_sir
N = 1410000000 # Population number

df_full = pd.read_csv("covid_china_data_spring_2022_cumulative (1).csv")
```

```
I_obs = df_full['confirmed_cases'].values.astype(float)   # Set up I_obs arr
t_obs = np.linspace(0, len(I_obs)-1, len(I_obs)) # time array in days

# Set initial SIR values for Euler's
I0_obs = df_full.iloc[0]['confirmed_cases']
R0_obs = 0.0
S0_obs = N - I0_obs - R0_obs

beta1 = .29 #random guess for beta
gamma1 = 1/4 #random guess for gamma
beta2 = 0.30 #alternative guesses
gamma2 = 1/4 #alternative guesses

# Run Euler's method to find SIR values using guessed beta & gamma values
S1,I1,R1 = euler_sir(beta1, gamma1,S0_obs, I0_obs, R0_obs, t_obs, N)
S2,I2,R2 = euler_sir(beta2, gamma2, S0_obs, I0_obs, R0_obs, t_obs, N)

# Print out SSE's for guesses
print(f"First Guess SSE (beta = .29 and gamma = 0.25): {float(np.mean((I1 -
print(f"Second Guess SSE (beta = .30 and gamma = 0.25): {float(np.mean((I2 -


# Optimization Routine: Find best beta and gamma values, in 0.01 increments,
min_SSE = {"Best beta": 0, "Best gamma": 0, "Min SSE": float("inf")} # Place
opt_I = None # Place holder variable for I(t) list for Euler's graph with mi
for b in range(0, 100):
    for g in range(0, 100):
        S,I,R = euler_sir(b * 0.01, g * 0.01, S0_obs, I0_obs, R0_obs, t_obs,
        if np.sum((I - I_obs)**2) < min_SSE['Min SSE']:
            min_SSE['Min SSE'] = float(np.sum((I - I_obs)**2))
            min_SSE['Best beta'] = b * 0.01
            min_SSE['Best gamma'] = g * 0.01
            opt_I = I

print(f"Optimized Guess: {min_SSE}")


# Plot the I(t) list from Euler's method calculations, including first guess
plt.plot(t_obs, I1, label=f'First Guess: beta={beta1},gamma={gamma1}', marke
plt.plot(t_obs, I2, label=f'Second Guess: beta={beta2},gamma={gamma2}', mark
plt.plot(t_obs, opt_I, label=f'Optimized Guess: beta={min_SSE["Best beta"]},
plt.plot(t_obs, I_obs, 'o', label='Observed')

plt.legend()
plt.xlabel('days')
plt.ylabel('Infections (in millions)')
plt.title('Demo: effect of beta/gamma on I(t)')
plt.show()
```

```
First Guess SSE (beta = .29 and gamma = 0.25): 315054894490.2398
Second Guess SSE (beta = .30 and gamma = 0.25): 2360669047505.2554
Optimized Guess: {'Best beta': 0.96, 'Best gamma': 0.91, 'Min SSE': 88874418
30995.057}
```

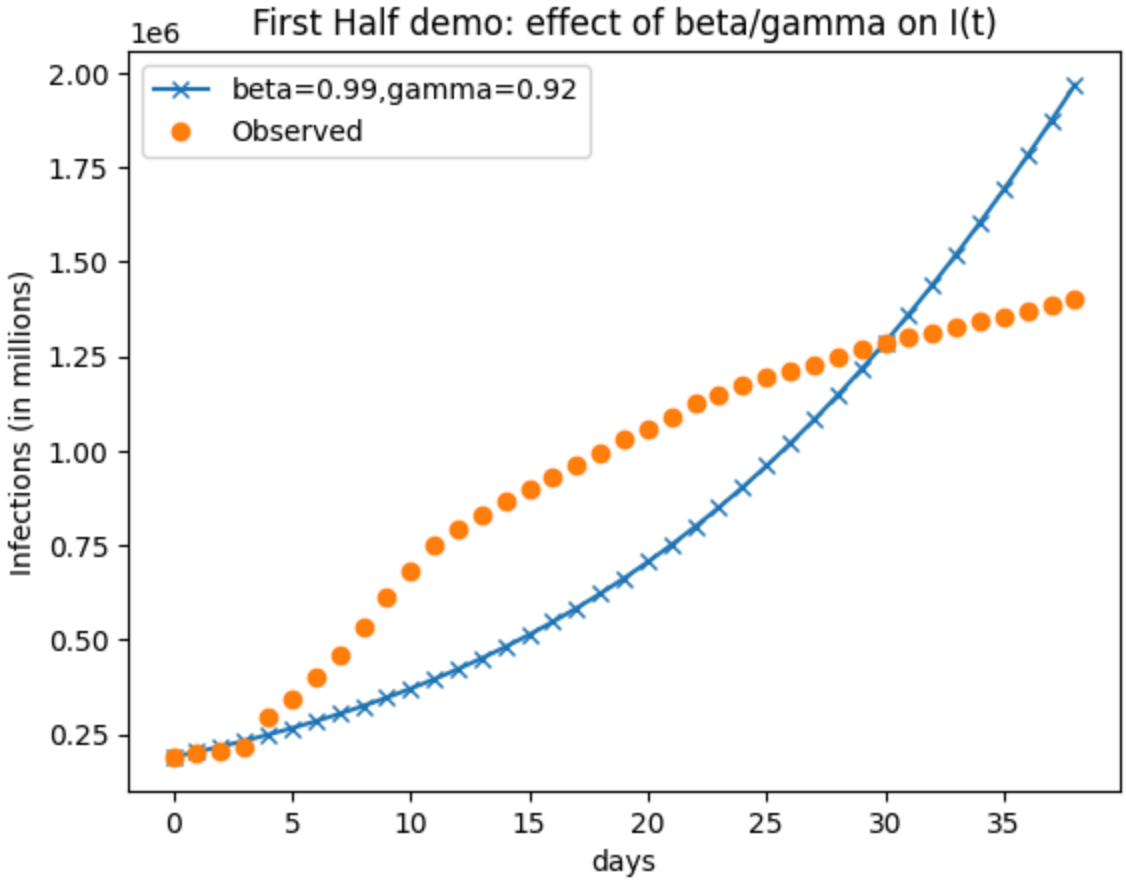


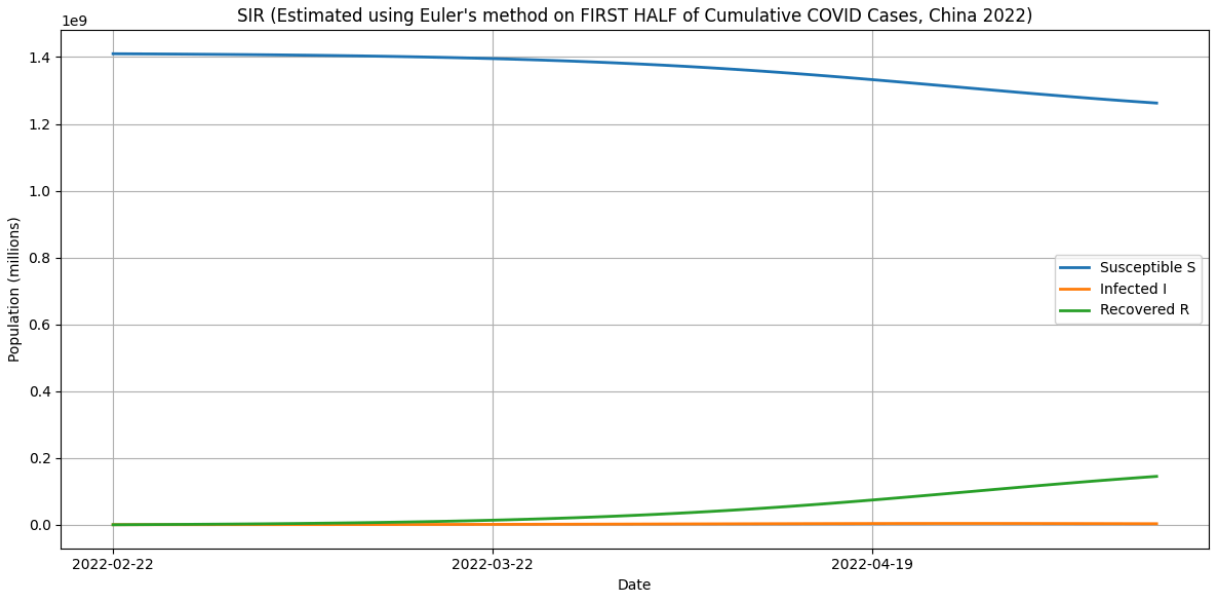
2. Predict "the future" with your fit SIR model

In [ ]:
```
# Use euler's method and your optimization routine above to find new gamma a
# FIRST HALF of the data, then simulate the SIR model forward in time using

import pandas as pd

df_full = pd.read_csv("covid_china_data_spring_2022_cumulative (1).csv")

I_obsfull = df_full['confirmed_cases'].values.astype(float)   # Set up I_obs
t_obsfull = np.linspace(0, len(I_obsfull)-1, len(I_obsfull)) # time array ir

# Split the dataset to retrieve only the first half
midpoint = int(len(df_full) / 2)
df_half = df_full.iloc[:midpoint]


# Find good beta and gamma using Euler's code, made previously, and plot I(t
from main_functions import euler_sir
N = 1410000000

I_obs = df_half['confirmed_cases'].values.astype(float) # Set up I_obs array
t_obs = np.linspace(0, len(I_obs)-1, len(I_obs)) # time array in days (only

# Set initial SIR values for Euler's
I0_obs = df_half.iloc[0]['confirmed_cases']
R0_obs = 0.0
S0_obs = N - I0_obs - R0_obs
```

```
# Optimization Routine: Find best beta and gamma values, in 0.01 increments,
min_SSE = {"Best beta": 0, "Best gamma": 0, "Min SSE": float("inf")} # Place
half_I = None # Place holder variable for I(t) list for Euler's graph with m
for b in range(0, 100):
    for g in range(0, 100):
        S,I,R = euler_sir(b * 0.01, g * 0.01, S0_obs, I0_obs, R0_obs, t_obs,
        #print(np.mean((I - I_obs)**2))
        if np.sum((I - I_obs)**2) < min_SSE['Min SSE']:
            min_SSE['Min SSE'] = float(np.sum((I - I_obs)**2))
            min_SSE['Best beta'] = b * 0.01
            min_SSE['Best gamma'] = g * 0.01
            half_I = I
print("Optimal parameters:")
print(min_SSE)

# Plotting I(t) over days with real data and predicted data from Euler's usi
plt.plot(t_obs, half_I, label=f'beta={min_SSE["Best beta"]},gamma={min_SSE["
plt.plot(t_obs, I_obs, 'o', label='Observed')

plt.legend()
plt.xlabel('days')
plt.ylabel('Infections (in millions)')
plt.title('First Half demo: effect of beta/gamma on I(t)')
plt.show()


### Plotting SIR with full time array

I_obsTest = df_full['confirmed_cases'].values.astype(float)
t_full = np.linspace(0, len(I_obsTest)-1, len(I_obsTest)) # time array in da
S,I,R = euler_sir(min_SSE['Best beta'], min_SSE['Best gamma'], S0_obs, I0_ob

plt.figure(figsize=(12,6))

plt.plot(sir_df['date'], S, label='Susceptible S', linewidth=2)
plt.plot(sir_df['date'], I, label='Infected I', linewidth=2)
plt.plot(sir_df['date'], R, label='Recovered R', linewidth=2)

plt.xlabel("Date")
plt.ylabel("Population (millions)")
plt.title("SIR (Estimated using Euler's method on FIRST HALF of Cumulative C
plt.legend()

# plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%-m/%-y'))
plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=4))
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Optimal parameters:
{'Best beta': 0.99, 'Best gamma': 0.92, 'Min SSE': 3129730712011.685}
```

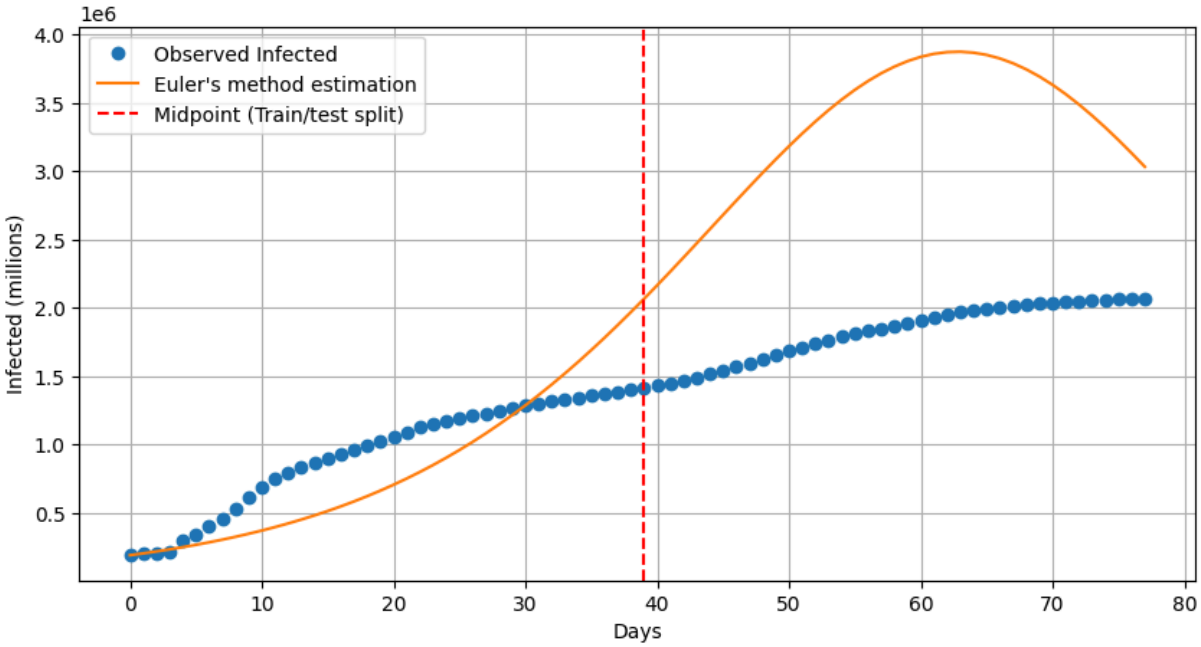




In [ ]:
```
# Calculating SSE between model predictions and data on the SECOND HALF of t

import pandas as pd

df_full = pd.read_csv("covid_china_data_spring_2022_cumulative (1).csv")

I_obsfull = df_full['confirmed_cases'].values.astype(float)   # Set up I_obs
t_obsfull = np.linspace(0, len(I_obsfull)-1, len(I_obsfull)) # time array ir

midpoint = int(len(df_full) / 2) # Midpoint index for splitting the dataset
```

```
from main_functions import euler_sir
N = 1410000000

I0_obs = df_full.iloc[0]['confirmed_cases']
R0_obs = 0.0
S0_obs = N - I0_obs - R0_obs

# Best beta and gamma values found in previous code block
firstHalf_bestBeta = 0.99
firstHalf_bestGamma = 0.92

print("SSE for First Half (Euler's Method): 3129730712011.685") # Print firs

# Run Euler's on full dataset using best beta and gamma found for the FIRST
S_full,I_full,R_full = euler_sir(firstHalf_bestBeta, firstHalf_bestGamma, S0

# Split the I(t) dataset to retrieve only the second half and calculation th
secondHalf_I = I_full[midpoint:]
secondHalf_Iobs = I_obsfull[midpoint:]
secondHalf_SSE = np.sum((secondHalf_I - secondHalf_Iobs)**2)
print(f"\nSSE for Second Half (Euler's Method): {secondHalf_SSE}")


# For visualization, plot I(t) for observed and Euler's simulated data
plt.figure(figsize=(10,5))

plt.plot(t_obsfull, I_obsfull, "o", label="Observed Infected")
plt.plot(t_obsfull, I_full, "-", label="Euler's method estimation")

#draw a line down the middle to show where the test and train cutoff is
plt.axvline(midpoint, color="red", linestyle="--", label="Midpoint (Train/te

plt.xlabel("Days")
plt.ylabel("Infected (millions)")
plt.legend()
plt.grid()
plt.show()
```

```
SSE for First Half (Euler's Method): 3129730712011.685

SSE for Second Half (Euler's Method): 91114737878387.83
```



Is the new gamma and beta close to what you found on the full dataset? Is the fit much worse? What is the SSE calculated for the second half of the data?

The new gamma and beta values were slightly close to the ones found for the full dataset, though they were notably a few tenths different (i.e. full beta, gamma = 0.96, 0.91 vs. first half beta, gama = 0.99, 0.92). The minimum SSE for the first half of data, ~3.1e12, is lower than the min SSE of the full dataset, but this lower SSE is likely because of there being half the number of data points for gathering error from compared to the full dataset. The SSE calculated for the second half of the data was 91114737878387.83, or ~9.1e13, which is notably bigger than the full dataset min SSE. Acknowledging both the SSEs from the first and second half, it's evident that the fit becomes worse than the previous full dataset one when the model starts predicting infection values in the second half.

Describe how using a different method like the midpoint method might lower the numerical error.

Different methods, espeically the midpoint method, might lower the numerical error by approximating slope values that resemble the observed data better than Euler's method. For example, the midpoint method uses the slope of the midpoint between the current point and the next step instead of the less accurate slope at the current point; the midpoint slope, thus, allow the method to lessen the amount of numerical error between its predicted values and the observed data.

> **Key Point:**
> The error you calculate is a *combination* of two sources:

1. the error associated with Euler's method (i.e. it is an imperfect numerical approximation to the true solution of the SIR model)
2. the error associated with comparing real-world data to a model with limitations.

First we will try to address the numerical error, and second we will address the limitations of the model.

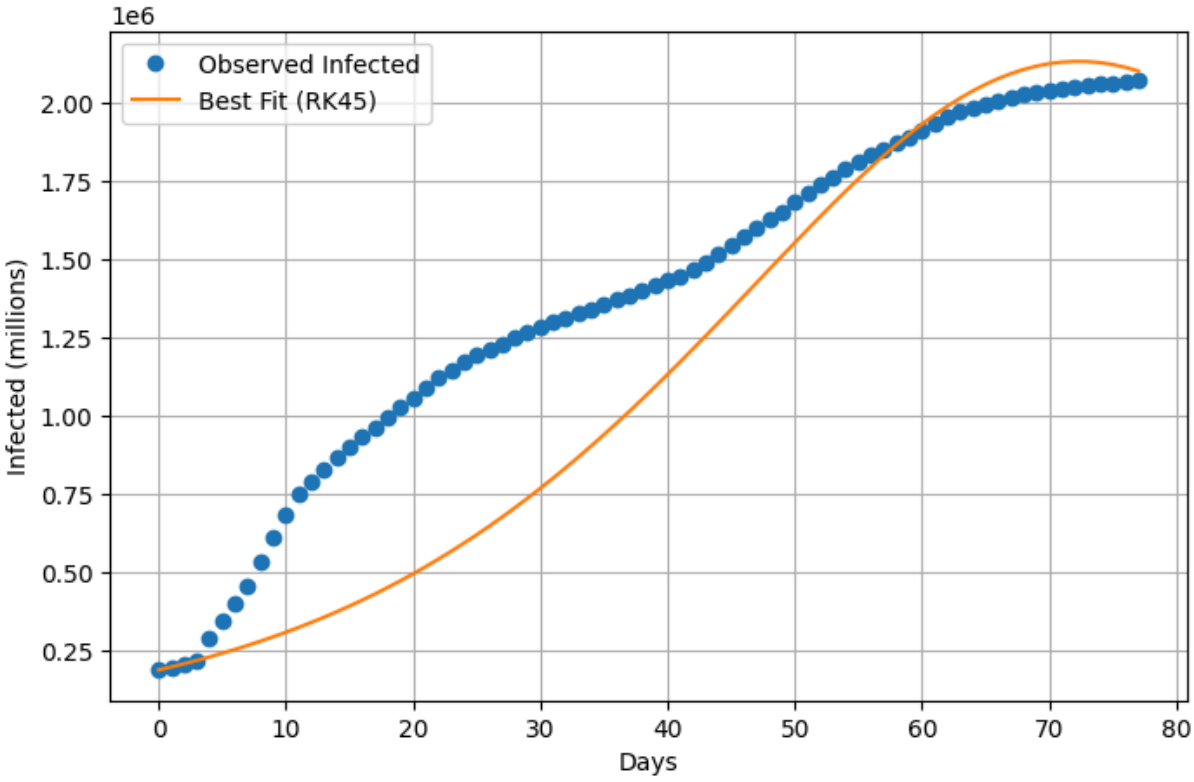
3. Decreasing numerical error with the RK4 Method

The following code calculates the ideal beta and gamma values for the entire dataset and trains the RK4 model on the entire dataset. The graph of the actual and predicted values is shown and the values of the ideal gamma, beta, and sse values when they are used for the RK4 model are printed. From this, we can see that the best beta is 0.96, the best gamma is 0.91, and the sse value when they are used is 8254889976273.559

In [ ]:
```
import numpy as np
import pandas as pd
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

df = pd.read_csv("covid_china_data_spring_2022_cumulative (1).csv")
I_obs = df["confirmed_cases"].values
t_obs = np.arange(len(I_obs))

N = 1410000000

I0_obs = I_obs[0]
R0_obs = 0.0
S0_obs = N - I0_obs - R0_obs


def sir_odes(t, y, beta, gamma):
    S, I, R = y
    dS = -beta * S * I / N
    dI = beta * S * I / N - gamma * I
    dR = gamma * I
    return [dS, dI, dR]


def solve_sir(beta, gamma, S0, I0, R0, t_eval):
    sol = solve_ivp(
        lambda t, y: sir_odes(t, y, beta, gamma),
        (t_eval[0], t_eval[-1]),
        [S0, I0, R0],
        t_eval=t_eval,
        method="RK45"
    )
    S, I, R = sol.y
    return S, I, R
```

```
min_SSE = {"beta": 0, "gamma": 0, "value": float("inf")}
best_I_curve = None
for b in range(0, 100):
    for g in range(0, 100):
        beta = b * 0.01
        gamma = g * 0.01
        S, I, R = solve_sir(beta, gamma, S0_obs, I0_obs, R0_obs, t_obs)
        sse = np.sum((I - I_obs)**2)
        if sse < min_SSE["value"]:
            min_SSE["value"] = float(sse)
            min_SSE["beta"] = beta
            min_SSE["gamma"] = gamma
            best_I_curve = I
print("Optimal parameters:")
print(min_SSE)


plt.figure(figsize=(8,5))
plt.plot(t_obs, I_obs, "o", label="Observed Infected")
plt.plot(t_obs, best_I_curve, "-", label="Best Fit (RK45)")
plt.xlabel("Days")
plt.ylabel("Infected (millions)")
plt.legend()
plt.grid()
plt.show()
```

```
Optimal parameters:
{'beta': 0.96, 'gamma': 0.91, 'value': 8254889976273.559}
```



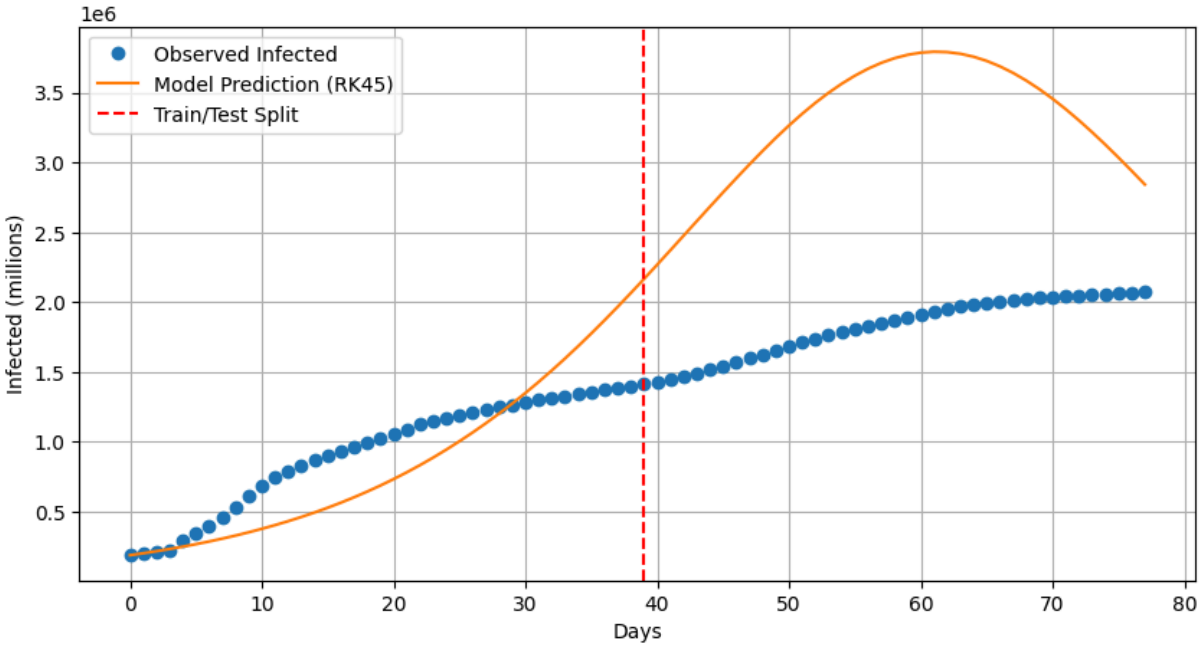
The following code utilizes the RK4 method on the first half of data to determine the ideal beta and gamma values. The same beta and gamma values obtained from the first

half of the data are used on the second half of the data to predict the number of those infected with COVID. The sse value from the beta and gamma values obtained from the first half of the data are shown below as well as the sse values obtained when those same beta and gamma values were applied to the second half the data. The graph shows the actual vs predicted utilizing the RK4 method. The red vertical line indicates the separation between the training and test data set halves.

In [ ]:
```
import numpy as np
import pandas as pd
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# --- Load data ---
df = pd.read_csv("covid_china_data_spring_2022_cumulative (1).csv")
I_obs = df["confirmed_cases"].values
t_obs = np.arange(len(I_obs))

N = 1410000000

# Initial conditions
I0_obs = I_obs[0]
R0_obs = 0.0
S0_obs = N - I0_obs - R0_obs

# ---- SIR ODE system ----
def sir_odes(t, y, beta, gamma):
    S, I, R = y
    dS = -beta * S * I / N
    dI = beta * S * I / N - gamma * I
    dR = gamma * I
    return [dS, dI, dR]

def solve_sir(beta, gamma, S0, I0, R0, t_eval):
    sol = solve_ivp(
        lambda t, y: sir_odes(t, y, beta, gamma),
        (t_eval[0], t_eval[-1]),
        [S0, I0, R0],
        t_eval=t_eval,
        method="RK45"
    )
    S, I, R = sol.y
    return S, I, R

#training on first half of data
mid = len(I_obs) // 2
t_train = t_obs[:mid]
I_train = I_obs[:mid]

min_SSE = {"beta": 0, "gamma": 0, "value": float("inf")}

# find the best beta and gamma values
for b in range(0, 100):
    for g in range(0, 100):
        beta = b * 0.01
```

```
        gamma = g * 0.01

        S, I, R = solve_sir(beta, gamma, S0_obs, I0_obs, R0_obs, t_train)
        sse = np.sum((I - I_train)**2)

        if sse < min_SSE["value"]:
            min_SSE["value"] = float(sse)
            min_SSE["beta"] = beta
            min_SSE["gamma"] = gamma

print("best beta and gamma found from the first half: ")
print(min_SSE)


beta_best = min_SSE["beta"]
gamma_best = min_SSE["gamma"]

S_full, I_full, R_full = solve_sir(beta_best, gamma_best, S0_obs, I0_obs, R0


I_pred_second_half = I_full[mid:]
I_obs_second_half = I_obs[mid:]

SSE_second_half = np.sum((I_pred_second_half - I_obs_second_half)**2)

print("\nSSE on SECOND HALF (RK45 model trained on first half):")
print(SSE_second_half)


plt.figure(figsize=(10,5))

plt.plot(t_obs, I_obs, "o", label="Observed Infected")
plt.plot(t_obs, I_full, "-", label="Model Prediction (RK45)")

#draw a line down the middle to show where the test and train cutoff is
plt.axvline(mid, color="red", linestyle="--", label="Train/Test Split")


plt.xlabel("Days")
plt.ylabel("Infected (millions)")
plt.legend()
plt.grid()
plt.show()
```

```
best beta and gamma found from the first half:
{'beta': 0.99, 'gamma': 0.92, 'value': 3336014926254.8994}

SSE on SECOND HALF (RK45 model trained on first half):
86650668492161.3
```



Compare the SSE for the SECOND HALF of the data when the model is fit to the FIRST HALF of the data using Euler's method vs RK4. Did RK4 do a better job? Why or why not?

The RK4 did a better job at predicting the second half of data (when fit to the FIRST HALF of data) than Euler's method. We know this because RK4 got a lower SSE on the second half, ~8.7e13, than Euler's method SSE on the second half, ~9.1e13.

4. Improving model fit by overcoming model limitations

Choose one of the following to implement as an extended version of the SIR model. Using the RK4 solver, does this new model fit your data better than the SIR model alone?

Options to overcome limitations (choose ONE to implement):

1. Include births in the model as described in reading.
2. Include deaths in the model as described in reading.
3. Include an exposed compartment (SEIR model).
4. Include loss of immunity (i.e. R population can go back to S population).
5. Include at least two I populations with varying degrees of infectiousness.
6. Include at least two age brackets with varying degress of infectiousness and recovery times.

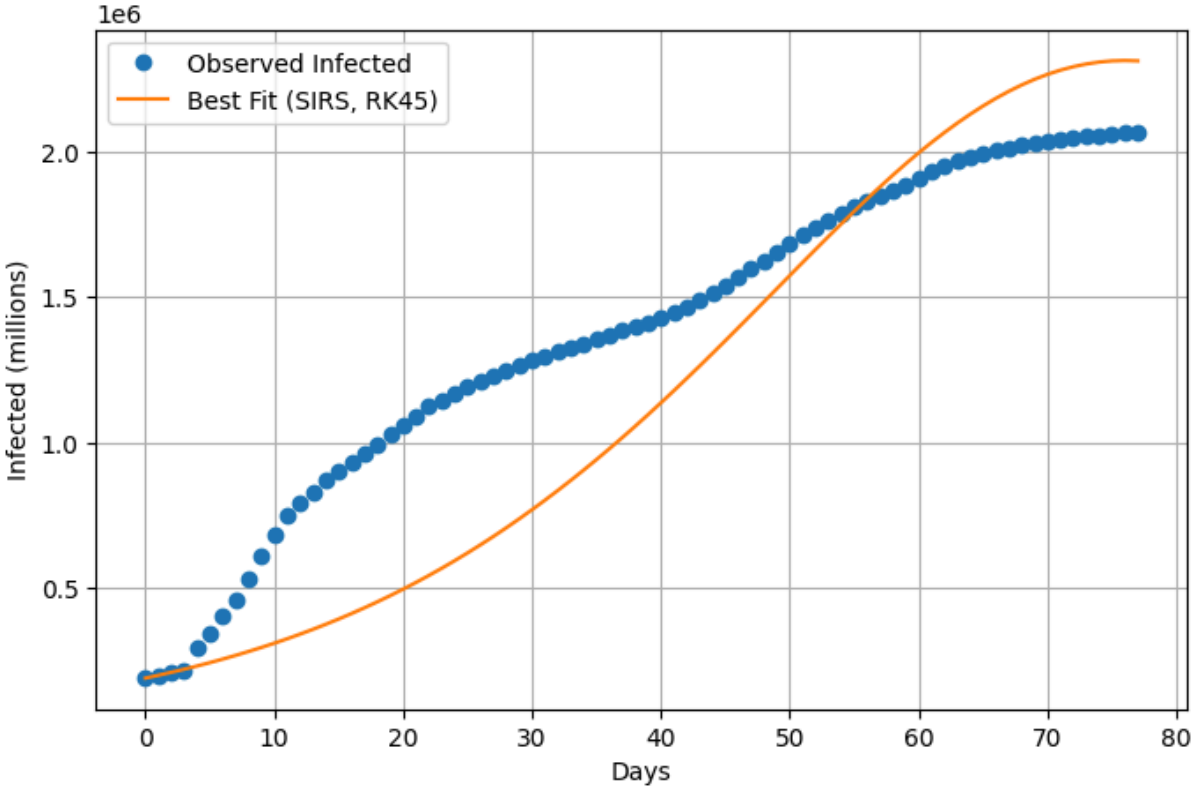
Note that if you have implemented an extended model and are having trouble fitting the parameters, document what you have tried and explain what you would change in future directions.

The following code implements number 4 of the above: include loss of immunity.

In [ ]:
```
import numpy as np
import pandas as pd
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

df = pd.read_csv("covid_china_data_spring_2022_cumulative (1).csv")
I_obs = df["confirmed_cases"].values
t_obs = np.arange(len(I_obs))

N = 1410000000

I0_obs = I_obs[0]
R0_obs = 0.0
S0_obs = N - I0_obs - R0_obs




def sirs_odes(t, y, beta, gamma, omega):
    S, I, R = y
    dS = -beta * S * I / N + omega * R
    dI = beta * S * I / N - gamma * I
    dR = gamma * I - omega * R
    return [dS, dI, dR]


def solve_sirs(beta, gamma, omega, S0, I0, R0, t_eval):
    sol = solve_ivp(
        lambda t, y: sirs_odes(t, y, beta, gamma, omega),
        (t_eval[0], t_eval[-1]),
        [S0, I0, R0],
        t_eval=t_eval,
        method="RK45"
    )
    S, I, R = sol.y
    return S, I, R



min_SSE = {"beta": 0, "gamma": 0, "omega": 0, "value": float("inf")}
best_I_curve = None

#immunity to covid lasts ~3–4 months so 4 months is used here (~120 days)
omega = 1 / 120

for b in range(0, 100):
    for g in range(0, 100):
        beta = b * 0.01
        gamma = g * 0.01

        S, I, R = solve_sirs(
            beta, gamma, omega,
            S0_obs, I0_obs, R0_obs, t_obs
```

```
        )

        sse = np.sum((I - I_obs)**2)

        if sse < min_SSE["value"]:
            min_SSE.update({
                "beta": beta,
                "gamma": gamma,
                "omega": omega,
                "value": float(sse)
            })
            best_I_curve = I

print("Optimal parameters:")
print(min_SSE)


plt.figure(figsize=(8,5))
plt.plot(t_obs, I_obs, "o", label="Observed Infected")
plt.plot(t_obs, best_I_curve, "-", label="Best Fit (SIRS, RK45)")
plt.xlabel("Days")
plt.ylabel("Infected (millions)")
plt.legend()
plt.grid()
plt.show()
```

```
Optimal parameters:
{'beta': 0.99, 'gamma': 0.9400000000000001, 'omega': 0.008333333333333333,
'value': 8843229615803.412}
```

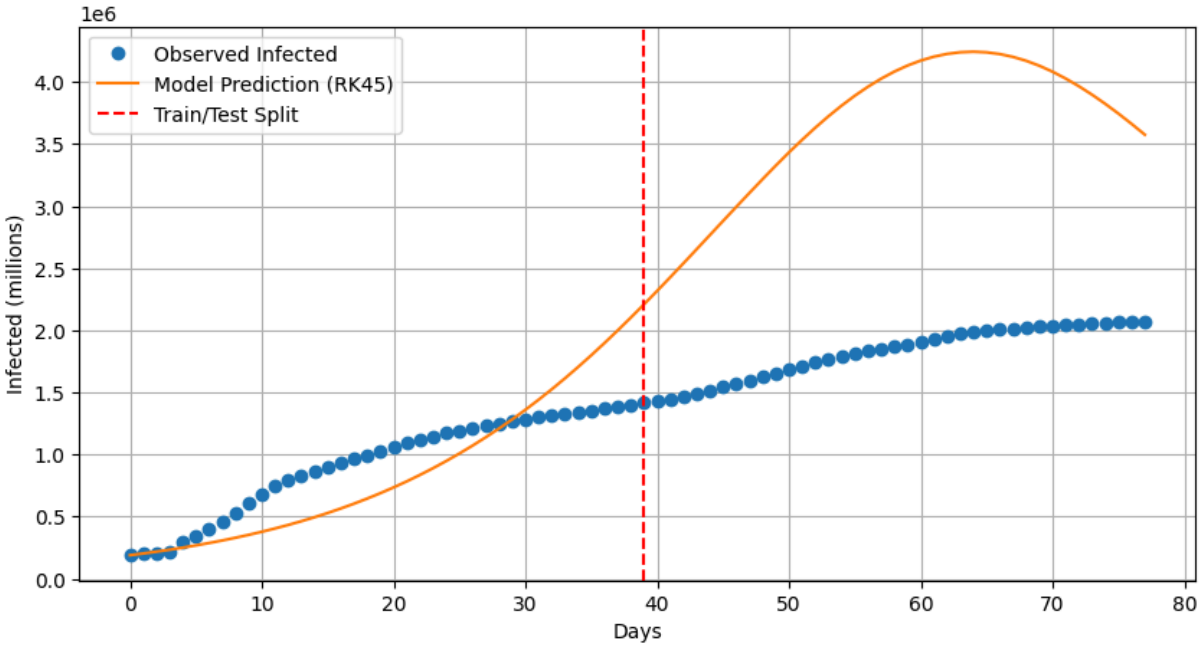


In [ ]:
```
import numpy as np
import pandas as pd
```

```
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

df = pd.read_csv("covid_china_data_spring_2022_cumulative (1).csv")
I_obs = df["confirmed_cases"].values
t_obs = np.arange(len(I_obs))

N = 1410000000

I0_obs = I_obs[0]
R0_obs = 0.0
S0_obs = N - I0_obs - R0_obs




def sirs_odes(t, y, beta, gamma, omega):
    S, I, R = y
    dS = -beta * S * I / N + omega * R
    dI = beta * S * I / N - gamma * I
    dR = gamma * I - omega * R
    return [dS, dI, dR]


def solve_sirs(beta, gamma, omega, S0, I0, R0, t_eval):
    sol = solve_ivp(
        lambda t, y: sirs_odes(t, y, beta, gamma, omega),
        (t_eval[0], t_eval[-1]),
        [S0, I0, R0],
        t_eval=t_eval,
        method="RK45"
    )
    S, I, R = sol.y
    return S, I, R



min_SSE = {"beta": 0, "gamma": 0, "omega": 0, "value": float("inf")}
best_I_curve = None

#immunity to covid lasts ~3-4 months so 4 months is used here (~120 days)
omega = 1 / 120

#training on first half of data
mid = len(I_obs) // 2
t_train = t_obs[:mid]
I_train = I_obs[:mid]

for b in range(0, 100):
    for g in range(0, 100):
        beta = b * 0.01
        gamma = g * 0.01

        S, I, R = solve_sirs(
            beta, gamma, omega,
            S0_obs, I0_obs, R0_obs, t_train
```

```
        )

        sse = np.sum((I - I_train)**2)

        if sse < min_SSE["value"]:
            min_SSE.update({
                "beta": beta,
                "gamma": gamma,
                "omega": omega,
                "value": float(sse)
            })
            best_I_curve = I

print("best beta and gamma found from the first half: ")
print(min_SSE)


beta_best = min_SSE["beta"]
gamma_best = min_SSE["gamma"]

S_full, I_full, R_full = solve_sirs(beta_best, gamma_best, omega, S0_obs, I0


I_pred_second_half = I_full[mid:]
I_obs_second_half = I_obs[mid:]

SSE_second_half = np.sum((I_pred_second_half - I_obs_second_half)**2)

print("\nSSE on SECOND HALF (RK45 model trained on first half):")
print(SSE_second_half)


plt.figure(figsize=(10,5))

plt.plot(t_obs, I_obs, "o", label="Observed Infected")
plt.plot(t_obs, I_full, "-", label="Model Prediction (RK45)")

#draw a line down the middle to show where the test and train cutoff is
plt.axvline(mid, color="red", linestyle="--", label="Train/Test Split")


plt.xlabel("Days")
plt.ylabel("Infected (millions)")
plt.legend()
plt.grid()
```

```
best beta and gamma found from the first half:
{'beta': 0.99, 'gamma': 0.92, 'omega': 0.008333333333333333, 'value': 349830
2604526.697}

SSE on SECOND HALF (RK45 model trained on first half):
134683833174107.31
```



The SSE value found after adding in an omega value representing the length of immunity was slightly higher than that of the original predictions of the entire dataset utilizing the RK4 method without the addition of the loss of immunity (SIRS model). We predict that this may be due to the estimated immunity length being inaccurate and the actual immunity time being of a different length.

This can be improved upon in future iterations by optimizing the value of omega to fall between the estimated immunity lengths of 3-4 months provided in the literature. Currently, we are unable to implement this due to the amount of computational power and the sheer amount of time required to run code that includes omega optimization in the nested for loop.

Verify and validate your analysis:

Verification

Our RK4 model produced lower SSE values than the Euler's model, both when trained on the full dataset (~8.3e12 SSE for RK4 vs. ~8.9e12 for Euler's) or only the first half (~8.7e13 second half SSE for RK4 vs. ~9.1e13 second half SSE for Euler's). These results make sense to us because the Runge-Kutta methods generally calculate more accurate slope estimations than Euler's method, allowing for predictions closer to the observed values.

As shown above, we also went about improving/verifying our original SIR model by incorporating a loss of immunity feature, represented by the added "omega" parameter into the system of ODEs, in which individuals in the recovered (R) group move back to the susceptible (S) group over time. Curiously, our SIR RK5 model produced higher SSEs

with the loss of immunity implemented (~8.8e12 SSE with full dataset, ~1.3e14 SSE predicted for second half with first half fitting) than without loss of immunity (~8.3e12 SSE with full dataset, ~8.7e13 SSE predicted for second half with first half fitting). This result implies that including the loss of immunity feature into our RK4 model may have worsened the performance. We believe this outcome for the fact that our loss of immunity parameter, omega, was not optimized due to the long run times it would require; the suboptimal omega value could have then increased error between our model prediction and observed values, explaining the higher SSEs.

Validation

To further validate our results, we calculated the basic reproductive numbers, R0, for our Euler's and RK4 models and compared them to real world R0 numbers found by outside literature estimating COVID-19 in China using SIR-based models. We're specifically comparing R0 values to external sources not only since it measures general contagiousness and transmissability of COVID-19 for each infection within a fully susceptible population, but also because its formula "R0 = β/γ" embodies the two key variables/results, beta and gamma, that our study aimed to find and optimize (Achaiah et al., 2020; Jones, 2007). Below are the calculated R0 values from our Euler's and RK4 models, either trained by the full dataset or first half of it:

- Euler's - Full Dataset: R0 = 0.96/0.91 = 1.055
- RK4 - Full Dataset: R0 = 0.96/0.91 = 1.055
- Euler's - First Half: R0 = 0.99/0.92 = 1.076
- RK4 - First Half: R0 = 0.99/0.92 = 1.076

Overall, all the R0 values from our models were slightly greater than 1, indicating that China was still in COVID-19 epidemic conditions within the time frame of late February 2022 to early May 2022 (Jones, 2007). However, our R0 values are significantly lower to the ones in other studies predicting COVID-19 infectivity in China, albeit ones using data before 2022. For instance, Jia Wangping et al. employed a standard SIR model on COVID-19 case data in Wuhan, China, from January 22nd to April 2nd of 2020, and they got a higher mean R0 value of 2.58 with a 95% confidence of 1.48-4.29 (2020). In the same study, researchers collected R0 estimates from external literature that were all higher than our R0 values, including R0 = "1.4–2.5... 2.68 (95% CI 2.47–2.68)... 3.6–3.8... and 6.47 (95% CI 5.71–7.23)" (Wangping et al., 2020). Another study by Khan et al. applied SIR and Kmeans clustering together to group regions in China into generated R0 value clusters spanning from 1.576 to 2.235, a range above our own R0s values (2021).

Despite being significantly lower, our calculated R0 values may still be acceptable given that the training data for our SIR models came from early 2022, unlike several related journal articles that frequently used COVID-19 case data from 2020 periods. With several factors like lockdown protocols and social distancing being implemented to decrease COVID-19 infectivity, it's plausible that the basic reproductive number R0

naturally reduced between 2020 and 2022, supporting why our R0 numbers are significantly lower than ones from older studies. The reduction of R0 after COVID-19 control measures is even supported by a study from Wang et al., who discovered that the reproduction number R significantly decreased in China across different stages of progressive COVID-19 emergency checkpoints (2020).

References

- Achaiah, N. C., Subbarajasetty, S. B., & Shetty, R. M. (2020). R0 and Re of COVID-19: Can We Predict When the Pandemic Outbreak will be Contained?. Indian journal of critical care medicine : peer-reviewed, official publication of Indian Society of Critical Care Medicine, 24(11), 1125–1127. https://doi.org/10.5005/jp-journals-10071-23649
- Jones, J. H. (2007). Notes on R0. California: Department of Anthropological Sciences, 323, 1-19.
- Khan, I. M., Haque, U., Zhang, W., Zafar, S., Wang, Y., He, J., Sun, H., Lubinda, J., & Rahman, M. S. (2021). COVID-19 in China: Risk Factors and R0 Revisited. Acta Tropica, 213, 105731. https://doi.org/10.1016/j.actatropica.2020.105731
- Wang, K., Zhao, S., Li, H., Song, Y., Wang, L., Wang, M. H., Peng, Z., Li, H., & He, D. (2020). Real-time estimation of the reproduction number of the novel coronavirus disease (COVID-19) in China in 2020 based on incidence data. Annals of translational medicine, 8(11), 689. https://doi.org/10.21037/atm-20-1944
- Wangping, J., Ke, H., Yang, S., Wenzhe, C., Shengshu, W., Shanshan, Y., Jianwei, W., Fuyin, K., Penggang, T., Jing, L., Miao, L., & Yao, H. (2020). Extended SIR Prediction of the Epidemics Trend of COVID-19 in Italy and Compared With Hunan, China. Frontiers in Medicine, Volume 7-2020. https://doi.org/10.3389/fmed.2020.00169

Conclusions and Ethical Implications:

To synthesize, we created code to predict COVID infections in China utilizing a SIR model in several ways, finding optimized beta (infectiousness) and gamma (recovery) values for each model iteration. When training our model on the full dataset provided, we found that the RK4 method produced a lower SSE (~8.3e12) than the Euler's method (~8.9e12), both using optimized beta and gamma values of 0.96 and 0.91 respectively. Additionally, we implemented a loss of immunity future to the RK4 method model to complement our verification, resulting in slightly different beta and gamma values (beta = 0.99, gamma = 0.94) but higher SSE due to lack of optimization of the omega (immunity loss) parameter. Training the model on only the first half of the dataset gave the same trend: RK4 had the least SSE of ~8.7e13, followed by Euler's method with ~9.1e13 second half SSE, and lastly RK4 with unoptimized loss of immunity with ~1.3e14, all of which had an optimized beta of 0.99 and gamma of 0.92. Employing those beta and gamma values, we also found that the basic reproductive number, or R0, of our model is 1.055 for full dataset training and 1.076 for first half training, both values being significantly lower than

R0s calculated in external literature but perhaps still justifiable by how our dataset collected case information from later dates. All in all, we conclude that a SIR model can work for predicting COVID infections in China, that RK4 is a more accurate method to execute the SIR model than Euler's method, and that R0 values for our model are greater than 1.

Ethical implications to consider may include heightened reactions to SIR model predictions of increasing COVID-19 infection rates in China. When we trained the SIR model on the first half of the dataset, the model mainly overestimated the millions of people infected when simulating future values in the second half, generating great SSE values. If our SIR model overestimations were to be made public during the period when the data was still being collected, Chinese residents may be misled to believe that COVID infections will rapidly rise soon, potentially giving them more anxiety as well as motivating them to stockpile resources and social distance themselves more than necessary. The Chinese government may also employ further COVID-19 control procedures in response to the SIR model overestimations, which can greatly hinder many people's jobs and livelihoods as a result. To avoid the consequences of inaccurate disease predictions, emphasis should be placed on scrutinizing SIR model predictions, paying attention to any limitations the model has, while carefully factoring in observed data trends and other model approximations to decide the best plan for future action, both in individual and government levels.

The increased accuracy from using the RK4 method may boost people's confidence in the SIR model predictions, allowing for better decision making related to the COVID-19 pandemic in China. However, this boosted confidence may wrongfully discourage researchers from improving the accuracy of the SIR model further by addressing more model limitations. Thus, researchers working on COVID SIR models in China should continue to optimize the design of their models to eliminate overconfidence.

Finally, it should be noted that our R0 values being greater than 1 implies that COVID-19 infected individuals in China are increasing, meaning that the country is still in an epidemic during the 2022 period from February 22nd to May 10th. This implication may spur Chinese regulatory bodies, at that early 2022 period, to prolong quarantine policies like face masks, social distancing, and potential lockdown protocols to ensure that the epidemic fades away. The R0 values may also be used as a means to persuade other Chinese citizens to still be cautious about COVID-19 and continue following the standard practices in reducing its spread, even if it bothers their mental wellness.

Limitations and Future Work:

- We had a limited amount of data. when training the first and second halves of the data, there were only ~39 days worth of data to train the first half and then ~39 days

worth of data to compare the predicted second half to. With a larger dataset, the training portion of the data would have been more accurate.

- The model only models one part of the pandemic. The COVID 19 pandemic featured many aspects with various fluctuations in the number of people infected dependent on the state of lockdown, masking mandates, vaccinations, etc. The model does not have a way to accurately model any changes in policy affecting the spread of the disease.

- Vaccinations were not taken into account. With vaccinations, the number of people able to become infected would have been lower with added immunity. In future iterations of this project, we hope to model immunity gained from multiple rounds of vaccinations and predict how the number of infected people changed after each round of the vaccination was introduced.

- Using smaller step sizes to find the ideal beta and gamma values would also be beneficial. Currently, we are not able to use smaller step sizes due to the extensive time it would take the code to run.

- The time for immunity to COVID was found to be 3-4 months. With more time and computational resources, we would fit the data to find out the number of days (between 90 days and 120 days) that fit the data the best.

- For future iterations, deaths would also be included in the model as deaths played an important role in the spread of the pandemic.

- The model does not take into account people coming into contact with others infected with COVID and also does not take into account the act of quarantining. In the future, we hope to expand the model to be able to replicate these patterns.

- We are interested in using our model to investigate which strand of vaccination given to the public was best able to prevent the spread of diseases and predict how the number of infected people would have changed with each strand of vaccination available to assess which is the most effective. We would do this by changing the value of omega to that of each vaccination type and predicting the model's prediction of infected individuals to compare vaccinations.

NOTES FROM YOUR TEAM:

This is where our team is taking notes and recording activity.

Progress Notes

- 11/18/2025: Started project, shared a GitHub page with each other, figured out and debugged inital in_class_COVID and main_functions python files given.

- 11/20/2025: Began filling out our Jupyter notebook in class as well as the code (Christy did most of the background and code, and Luke did the Dataset description and other miscelaneous areas).
- 11/22/2025: Finished up code and notes for first module check-in.
- 11/25/2025: Christy started making code for plotting SIR and fitting the SIR model using Euler's method (Luke not in class due to Thanksgiving break).
- 12/2/2025: Kept working on data analysis sections 1-3, with Luke working on fitting SIR Model on first half of dataset + predicting "future" values, and Christy looking into and implementing the RK4 method code.
- 12/4/2025: Did more work finishing up those data analysis sections from last class.
- 12/6/2025: Finished up the data analysis sections 1-3 and added more documentation for second module check-in.
- 12/12/2025: Christy worked on and completed section 4 of data analysis, namely by implementing a loss of immunity feature to our model, as well as the Limitations and Future Work sections.
- 12/15/2025: Luke worked on and completed the Verification and Validation section as well as the Conclusion and Ethical Implications section. Other code fixes and documentation was also added to prepare Jupyter project for final submission.

Code Progress

All code progress and commit history is recorded on the team's shared GitHub page linked here: https://github.com/christylee1/Module4_covid

QUESTIONS FOR YOUR TA:

First Module Check-in Questions

- There wasn't many online sources displaying updated prevalance and incidence rates for COVID in China and in general. Do you know any good sources for current COVID data in countries like China that we've missed, or should we stick to the prevalence and incidence background sources we have currently (which are quite outdated)?
- Are all our sections looking good? Is there any areas we need to add a bit more in?

Second Module Check-in Questions

- Did we do the predicting the "future" values for the second-half data correctly?
- When plotting the SIR calculated from convert_cumulative_to_SIR() using the full dataset, we noticed that the Susceptible S values were constantly at zero instead of starting near the max population number while slowly decreasing. Is this an issue,

and if so, how can we fix it to show more accurate S-values over time for the real dataset?

- Our optimization routine essentially boiled down to using a nested loop to find the beta and gamma values, in increments of 0.01, that give the lowest SSE. Is this optimization method good? (I'm not sure if it's too simplistic and if a more complex optimization method was intended)