Module 4: KAMEN5_Mujanovic_Kremsreiter_MODULE_4.IPYNB

Team Members:

Senada Mujanovic and Alexander Kremsreiter

Project Title:

China COVID-19 Case Analysis & Epidemic Modeling

Project Goal:

This project seeks to use the SIR model to predict the number of COVID-19 cases in China during 2022.

Disease Background:

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

- Prevalence & incidence
    - In 2022, China continued enforcing its strict "Zero-COVID" strategy, which kept infection numbers very low early in the year through aggressive lockdowns, mass PCR testing, and tight travel restrictions.
    - As the year went on and these policies were gradually relaxed, China experienced a rapid surge in new infections, creating a dramatic shift from very low incidence at the start of the year to extremely high incidence toward the end.
    - The dataset used in this project reflects the Spring 2022 Omicron outbreak, covering late February through early May 2022, a period when China saw one of its most significant waves of the pandemic.
    - During this window, cumulative cases rose sharply—from 189,565 on February 22 to over 2,070,480 by May 10—showing a steep increase in incidence as Omicron spread quickly, especially in major cities like Shanghai.
    - Because the data is cumulative, the steady upward trend also reflects growing prevalence, as more individuals remained within the active or recent infection pool during this high-transmission period.
    - Overall, 2022 presents a strong contrast in COVID-19 trends in China, making it a key year to analyze how policy changes directly shaped patterns of prevalence and incidence.

- Economic burden
    - COVID-19 created a major economic burden by increasing healthcare costs, including hospitalizations, testing, and long-term care for severe cases.
    - Many people experienced lost income due to illness, quarantine, or job cuts as businesses reduced operations.
    - Shutdowns and reduced consumer activity caused financial strain on businesses, supply chains, and entire industries like travel and retail.
    - Governments also faced increased spending on public-health measures, unemployment benefits, and economic relief programs. Overall, the pandemic placed a heavy financial strain on individuals, healthcare systems, and national economies.
    - In 2022, China's strict "Zero-COVID" policies created significant economic strain due to repeated lockdowns, mass testing programs, and travel restrictions.
    - Major cities like Shanghai experienced long shutdowns, disrupting manufacturing, shipping, and global supply chains.
    - Businesses faced reduced productivity, while many workers lost income because factories and service industries operated below normal levels.
    - As restrictions were lifted later in the year, the sudden surge in cases caused additional disruptions as large portions of the workforce became ill at the same time.
    - Overall, China's economy slowed sharply in 2022, reflecting the combined impact of containment measures, healthcare costs, and widespread labor shortages.
    - https://pmc.ncbi.nlm.nih.gov/articles/PMC10870589/
    - https://www.chinacenter.net/2023/china-currents/22-1/the-covid-19-pandemics-impact-on-the-chinese-economy/
- Risk factors (genetic, lifestyle) & Societal determinants
    - Older adults and people with chronic illnesses (such as heart disease, diabetes, asthma, COPD, or kidney disease) are at greater risk of serious COVID-19 illness.
    - Individuals with weakened immune systems, whether from medical conditions or medications, are more vulnerable to infection and complications.
    - Obesity, smoking, and pregnancy also increase the likelihood of severe symptoms.
    - Living or working in crowded settings or high-exposure jobs raises the risk of catching the virus. Overall, risk increases when the body's ability to fight infection is reduced or when exposure to the virus is more frequent.
    - https://www.cdc.gov/covid/risk-factors/index.html
- Symptoms
    - Fever or chills
    - Cough
    - Shortness of breath or difficulty breathing
    - Sore throat

    - Congestion or runny nose
    - New loss of taste or smell
    - Fatigue
    - Muscle or body aches
    - Headache
    - Nausea or vomiting
    - Diarrhea
    - https://www.cdc.gov/covid/signs-symptoms/index.html
- Diagnosis
    - COVID-19 is diagnosed primarily through tests that detect the virus in the body, either its genetic material (PCR tests) or its proteins (rapid antigen tests).
    - PCR tests are considered the most accurate because they amplify tiny amounts of viral RNA, allowing detection even in early or mild cases.
    - Rapid antigen tests provide results within minutes and are useful for identifying when someone is most contagious, although they may miss low-level infections.
    - Doctors also evaluate a patient's symptoms, exposure history, and physical findings (such as oxygen levels) to decide whether testing is necessary.
    - In some cases, chest imaging or blood tests may support diagnosis, especially in moderate or severe illness, but confirmation always requires a viral test.
    - The onset of the clinical symptoms occurs in average 5–6 days after exposure, and normally, those with mild symptoms recover within 2 weeks; however, in severe cases, the recovery may extend up to 6 weeks.
    - https://www.mayoclinic.org/diseases-conditions/coronavirus/diagnosis-treatment/drc-20479976
- Biological mechanisms (anatomy, organ physiology, cell & molecular physiology)
    - Anatomy:
    - The respiratory system includes structures such as the trachea, bronchi, and alveoli, which provide a pathway for air to reach the lungs.
    - Alveoli are thin-walled sacs surrounded by capillaries, maximizing surface area for gas exchange.
    - The diaphragm and intercostal muscles create pressure changes that allow inhalation and exhalation.
    - Organ physiology:
    - Gas exchange in the lungs occurs when oxygen diffuses from alveolar air into capillary blood while carbon dioxide diffuses out.
    - The cardiovascular system transports oxygen-rich blood from the lungs to tissues and returns carbon dioxide–rich blood back to the lungs.
    - The immune system responds to respiratory pathogens by activating inflammation, recruiting immune cells, and producing antibodies.
    - Cell and Molecular Physiology:

- Oxygen diffuses across the alveolar membrane due to partial pressure gradients and binds to hemoglobin inside red blood cells.
- Viral infections (such as COVID-19) begin when viral particles bind to specific receptors on host cells (e.g., ACE2), allowing entry and replication.
- Infected cells trigger signaling pathways—including cytokine release—to alert neighboring cells and activate the immune response.
- https://pmc.ncbi.nlm.nih.gov/articles/PMC8592035/

Dataset:

The dataset we're using tracks cumulative COVID-19 cases in China during Spring 2022, covering the period from late February through early May 2022. It comes from the file covid_china_data_spring_2022_cumulative.csv, which lists each date (in YYYY-MM-DD format) alongside the total number of confirmed COVID-19 infections reported up to that day. The case numbers are simple integer counts.

I'm assuming Dr. Groves aquired this data from publicly released reports compiled by Chinese national and provincial health agencies, using standard public-health surveillance methods like PCR testing, clinical diagnoses, and official case reporting systems. Because the dataset provides cumulative case totals rather than daily increases, it highlights how infections built up over time. This structure makes it especially useful for epidemiological modeling—such as the SIR model—where the focus is on understanding broader trends in how an outbreak grows and evolves.

For the following code block we executed this in our data analysis code blocks

In [ ]:
```
## LOAD YOUR DATASET HERE.

# 1. Read in the csv file of cumulative cases.

# 2. Use the convert_cumulative_to_SIR function to convert cumulative cases to appr

# 3. Plot S, I, R over time.
```

Data Analyis:

Methods

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

Analysis

Begin by filtering the data set of interest from the meta data set

Combines the 33 columns of data from china into one combined 'total cases' column and subtracts total cases each day from the prvious day's total cases to find the 'new cases' ccolumn data.

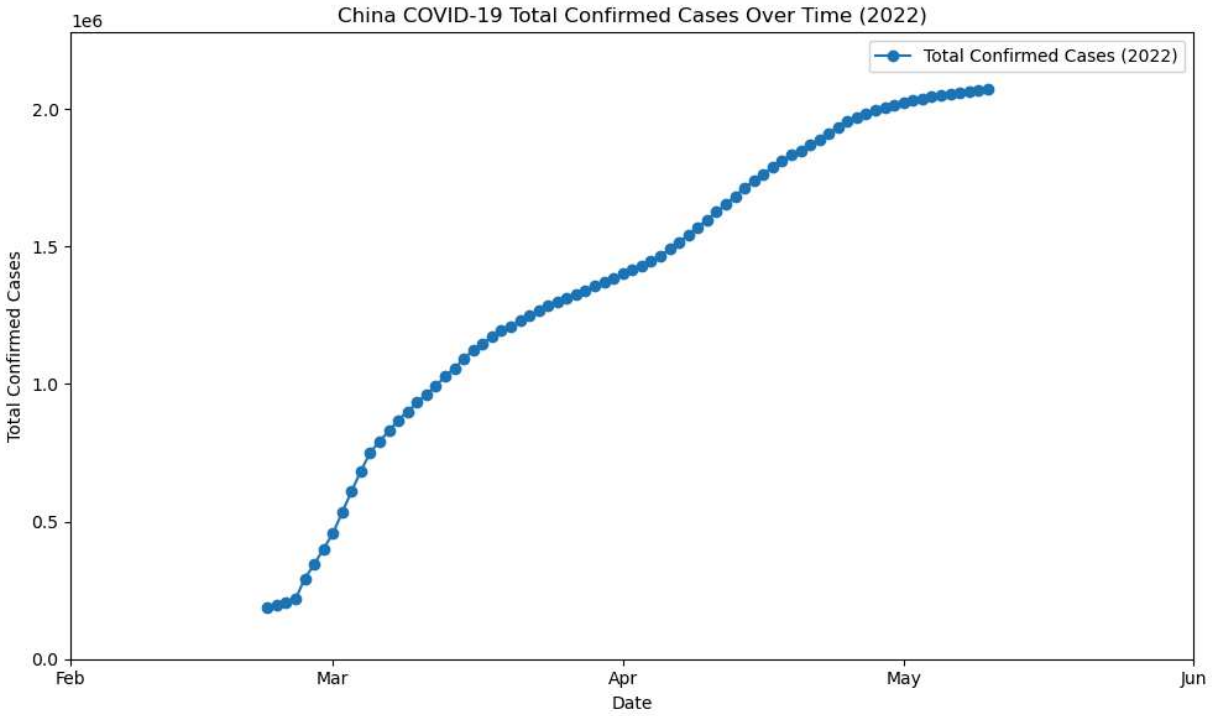
In [ ]:
```
# TAKES CHINA'S SPECIFIC DATA SET AND SUMS IT UP INTO ONE COLUMN FOR TOTAL CONFIRME

import pandas as pd

# Load the new dataset
df = pd.read_csv("covid_china_data_spring_2022_cumulative.csv")

# Ensure date is datetime
df["date"] = pd.to_datetime(df["date"], errors="coerce")

# Ensure confirmed_cases is numeric
df["confirmed_cases"] = pd.to_numeric(df["confirmed_cases"], errors="coerce")

# Compute NEW daily cases
df["new_cases"] = df["confirmed_cases"].diff().fillna(0)

# Prevent negative spikes (optional but recommended)
df["new_cases"] = df["new_cases"].clip(lower=0)

# Export updated CSV
output_path = "china_confirmed_with_new_cases.csv"
df.to_csv(output_path, index=False)

print(f"Saved: {output_path}")
print(df.head())
```

```
Saved: china_confirmed_with_new_cases.csv
        date  confirmed_cases  new_cases
0 2022-02-22           189565        0.0
1 2022-02-23           198478     8913.0
2 2022-02-24           207599     9121.0
3 2022-02-25           217932    10333.0
4 2022-02-26           292792    74860.0
```

Plot total cases over 2022 in China

In [ ]:
```
# %% PLOT TOTAL CONFIRMED CASES FOR 2022 ONLY

import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Load the cleaned China dataset
data = pd.read_csv("covid_china_data_spring_2022_cumulative.csv")
data["date"] = pd.to_datetime(data["date"], errors="coerce")

# Filter to 2022 only
data_2022 = data[(data["date"] >= "2022-01-01") & (data["date"] <= "2022-12-31")]

# Use the correct column name → confirmed_cases
```

```
confirmed_2022 = data_2022["confirmed_cases"]

# Plot total confirmed cases
plt.figure(figsize=(10, 6))
plt.plot(data_2022['date'],
         confirmed_2022,
         label='Total Confirmed Cases (2022)',
         marker="o")

plt.xlabel('Date')
plt.ylim(0, confirmed_2022.max() * 1.1)
plt.xlim(pd.Timestamp('2022-02-01'), pd.Timestamp('2022-6-1'))

plt.ylabel('Total Confirmed Cases')
plt.title('China COVID-19 Total Confirmed Cases Over Time (2022)')

# Format ticks nicely for 2022
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=1))

plt.legend()
plt.tight_layout()
plt.show()
```

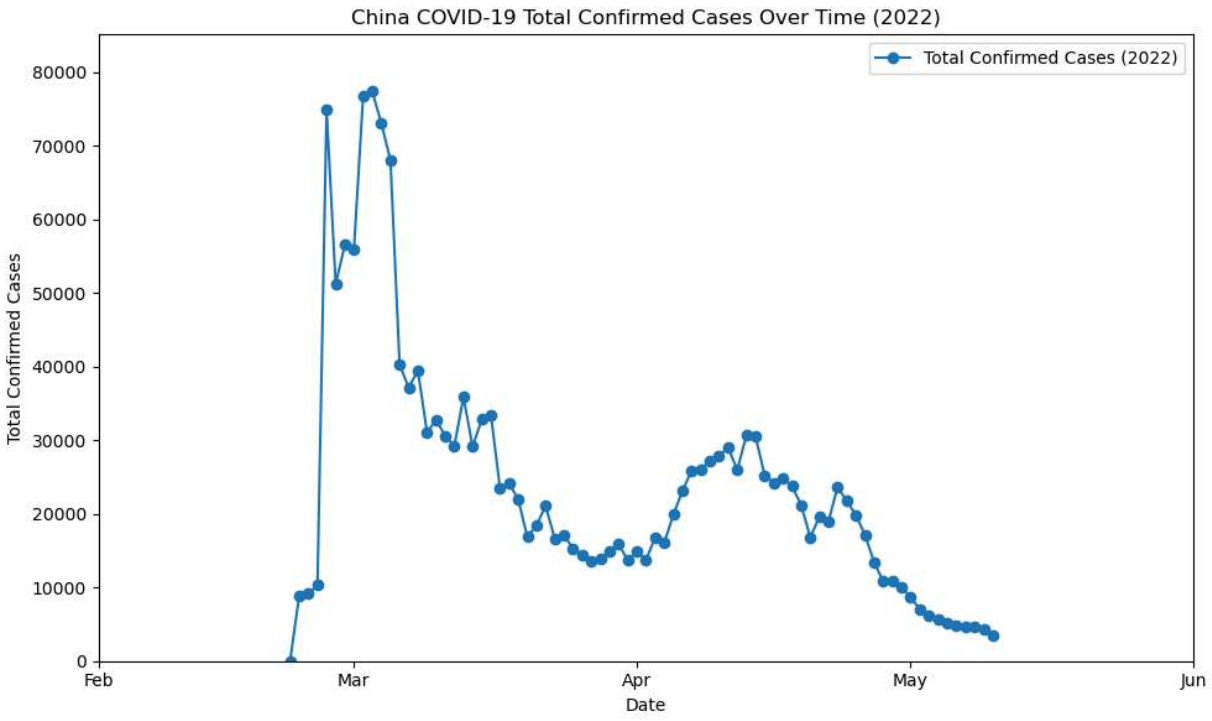


Plot new cases over 2022 in China

In [ ]:

```
# %% PLOT NEW INFECTIONS (INCIDENCE) OVER TIME – SPRING 2022

import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Load the cleaned China dataset
```

```
data = pd.read_csv("china_confirmed_with_new_cases.csv")
data["date"] = pd.to_datetime(data["date"], errors="coerce")

# Filter to 2022 only
data_2022 = data[(data["date"] >= "2022-01-01") & (data["date"] <= "2022-12-31")]

# Use the correct column name → confirmed_cases
confirmed_2022 = data_2022["new_cases"]

# Plot total confirmed cases
plt.figure(figsize=(10, 6))
plt.plot(data_2022['date'],
         confirmed_2022,
         label='Total Confirmed Cases (2022)',
         marker="o")

plt.xlabel('Date')
plt.ylim(0, confirmed_2022.max() * 1.1)
plt.xlim(pd.Timestamp('2022-02-01'), pd.Timestamp('2022-6-1'))

plt.ylabel('Total Confirmed Cases')
plt.title('China COVID-19 Total Confirmed Cases Over Time (2022)')

# Format ticks nicely for 2022
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=1))

plt.legend()
plt.tight_layout()
plt.show()
```

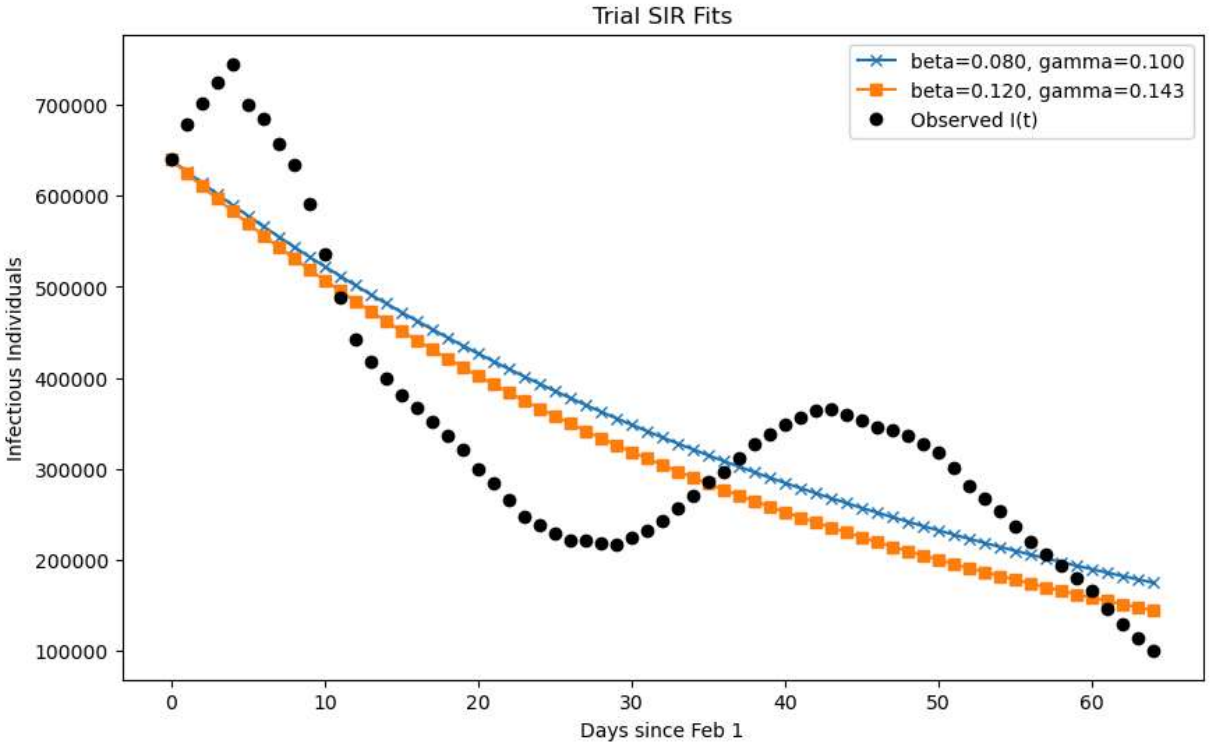


In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```python
# ================================================================
# 1. EULER SIR FUNCTION
# ================================================================
def euler_sir(beta, gamma, S0, I0, R0, t, N):
    """Euler integrator for SIR model with step size = 1 day."""
    h = t[1] - t[0]
    S = np.zeros(len(t))
    I = np.zeros(len(t))
    R = np.zeros(len(t))
    S[0], I[0], R[0] = S0, I0, R0

    for k in range(len(t)-1):
        dSdt = -beta * S[k] * I[k] / N
        dIdt = beta * S[k] * I[k] / N - gamma * I[k]
        dRdt = gamma * I[k]

        S[k+1] = S[k] + h * dSdt
        I[k+1] = I[k] + h * dIdt
        R[k+1] = R[k] + h * dRdt

    return S, I, R

# ================================================================
# 2. LOAD YOUR DATA
# ================================================================
df = pd.read_csv("china_confirmed_with_new_cases.csv")
df["date"] = pd.to_datetime(df["date"], errors="coerce")

# If you want strictly 2/22-5/10:
df = df[(df["date"] >= "2022-02-22") & (df["date"] <= "2022-05-10")]

N = 1411778724
infectious_period = 14

df["I_est"] = df["new_cases"].rolling(window=infectious_period).sum().fillna(0)
I_obs_full = df["I_est"].values

# Find first index where I_obs > 0
first_pos = np.argmax(I_obs_full > 0)

# Truncate to the part where infections are actually happening
I_obs = I_obs_full[first_pos:]
t_obs = np.arange(len(I_obs))  # 0,1,2,... for the truncated window

# Initial conditions
I0 = I_obs[0]
R0 = 0.0
S0 = N - I0 - R0
# Option A: approximate currently infectious as new_cases * infectious_period
#I_obs = (df["new_cases"].values * infectious_period).astype(float)

#t_obs = np.arange(len(I_obs))

#I0 = I_obs[0]
```

```
#R0 = 0.0
#S0 = N - I0 - R0
```

In [ ]:
```
# Choose two trial parameter pairs
beta1, gamma1 = 0.08, 1/10
beta2, gamma2 = 0.12, 1/7

S1, I1, R1 = euler_sir(beta1, gamma1, S0, I0, R0, t_obs, N)
S2, I2, R2 = euler_sir(beta2, gamma2, S0, I0, R0, t_obs, N)

plt.figure(figsize=(10,6))
plt.plot(t_obs, I1, 'x-', label=f"beta={beta1:.3f}, gamma={gamma1:.3f}")
plt.plot(t_obs, I2, 's-', label=f"beta={beta2:.3f}, gamma={gamma2:.3f}")
plt.plot(t_obs, I_obs, 'ko', label="Observed I(t)")
plt.xlabel("Days since Feb 1")
plt.ylabel("Infectious Individuals")
plt.title("Trial SIR Fits")
plt.legend()
plt.show()
```



In [ ]:
```
S1, I1, R1 = euler_sir(beta1, gamma1, S0, I0, R0, t_obs, N)
S2, I2, R2 = euler_sir(beta2, gamma2, S0, I0, R0, t_obs, N)

SSE1 = np.sum((I1 - I_obs)**2)
SSE2 = np.sum((I2 - I_obs)**2)
print("SSE1:", SSE1)
print("SSE2:", SSE2)
```

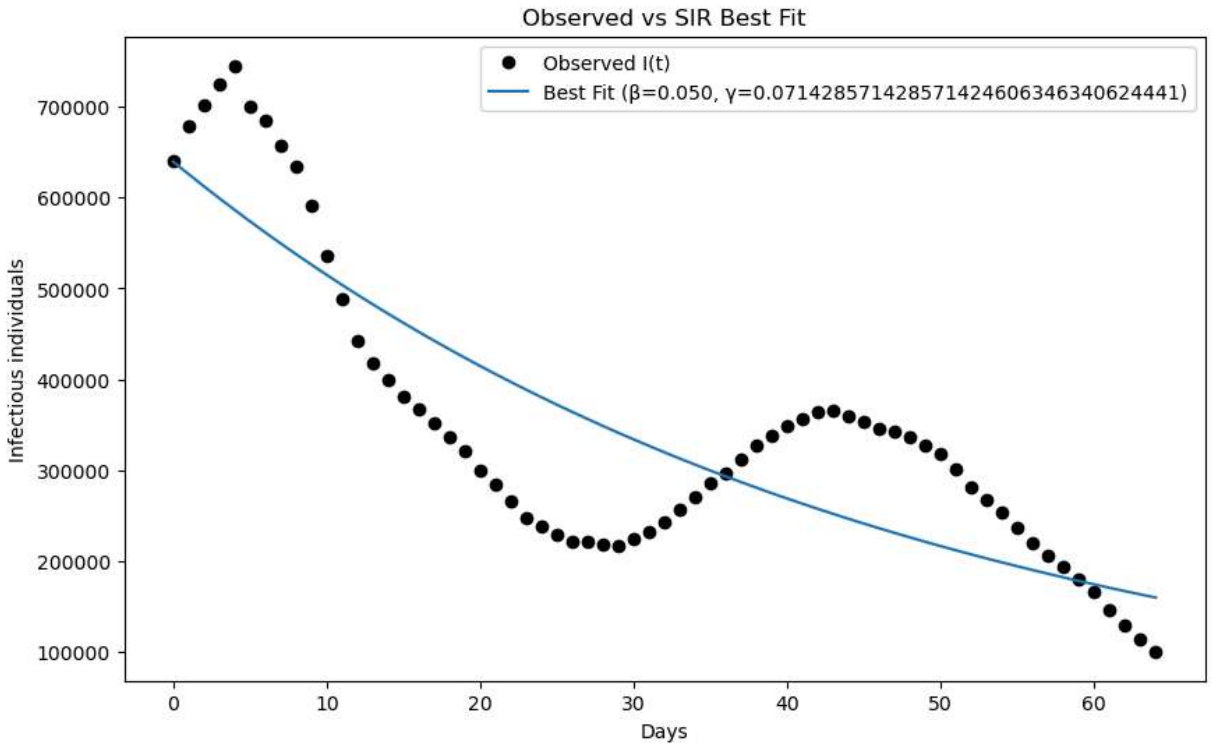
```
SSE1: 550274508658.4159
SSE2: 541506924184.6332
```

In [ ]:
```
beta_vals  = np.linspace(0.05, 0.25, 40)       # transmission rate
gamma_vals = np.linspace(1/14, 1/3, 40)
```

```
best_sse = np.inf
best_params = None

for beta in beta_vals:
    for gamma in gamma_vals:
        Sg, Ig, Rg = euler_sir(beta, gamma, S0, I0, R0, t_obs, N)
        sse = np.sum((Ig - I_obs)**2)
        if sse < best_sse:
            best_sse = sse
            best_params = (beta, gamma)

print("Best β, γ:", best_params)
print("Best SSE:", best_sse)
```

```
Best β, γ: (np.float64(0.05), np.float64(0.07142857142857142))
Best SSE: 534218722029.9194
```

In [ ]:
```
beta_best, gamma_best = best_params
S_best, I_best, R_best = euler_sir(beta_best, gamma_best, S0, I0, R0, t_obs, N)

plt.figure(figsize=(10,6))
plt.plot(t_obs, I_obs, 'ko', label="Observed I(t)")
plt.plot(t_obs, I_best, '-', label=f"Best Fit (β={beta_best:.3f}, γ={gamma_best:.33
plt.xlabel("Days")
plt.ylabel("Infectious individuals")
plt.legend()
plt.title("Observed vs SIR Best Fit")
plt.show()
```



In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

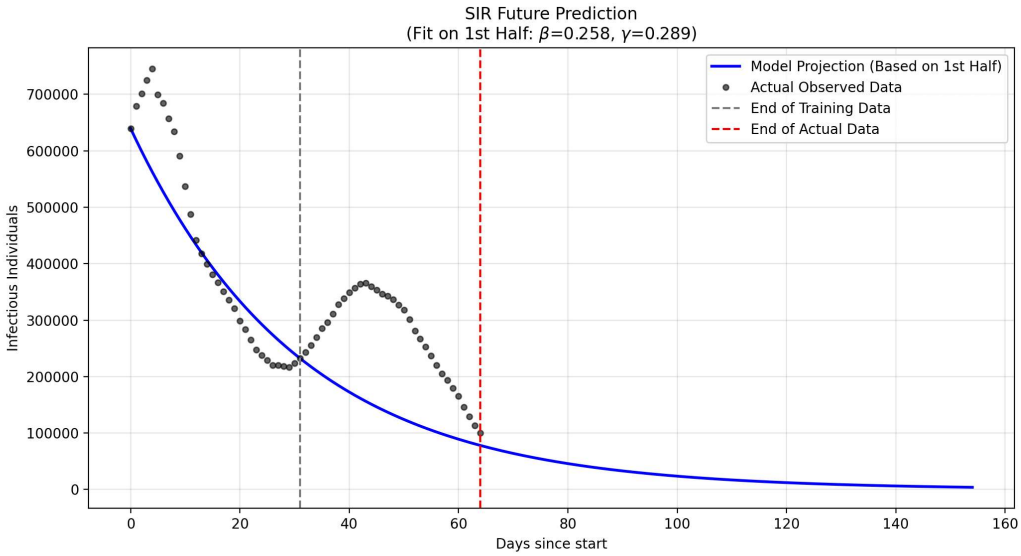
```python
# ================================================================
# 1. DEFINE THE EULER SIR FUNCTION
# ================================================================
def euler_sir(beta, gamma, S0, I0, R0, t, N):
    h = t[1] - t[0]
    S = np.zeros(len(t))
    I = np.zeros(len(t))
    R = np.zeros(len(t))
    S[0], I[0], R[0] = S0, I0, R0

    for k in range(len(t)-1):
        dSdt = -beta * S[k] * I[k] / N
        dIdt = beta * S[k] * I[k] / N - gamma * I[k]
        dRdt = gamma * I[k]

        S[k+1] = S[k] + h * dSdt
        I[k+1] = I[k] + h * dIdt
        R[k+1] = R[k] + h * dRdt

    return S, I, R

# ================================================================
# 2. LOAD DATA
# ================================================================
# Load the dataset
df = pd.read_csv("china_confirmed_with_new_cases.csv")
df["date"] = pd.to_datetime(df["date"], errors="coerce")

# Filter for the specific wave
df = df[(df["date"] >= "2022-02-22") & (df["date"] <= "2022-05-10")]

# Parameters
N = 1411778724
infectious_period = 14

# Estimate Active Infectious cases
df["I_est"] = df["new_cases"].rolling(window=infectious_period).sum().fillna(0)
I_obs_full = df["I_est"].values

# Truncate to start where I > 0
first_pos = np.argmax(I_obs_full > 0)
I_obs = I_obs_full[first_pos:]
t_obs = np.arange(len(I_obs))

# Initial Conditions
I0 = I_obs[0]
R0 = 0.0
S0 = N - I0 - R0

# ================================================================
# 3. TRAINING STEP (Defines beta_train & gamma_train)
# ================================================================
# Split data into first half (Train) and second half (Test)
split_idx = len(t_obs) // 2
t_train = t_obs[:split_idx]
```

```
I_train = I_obs[:split_idx]

# Search grid for best parameters
beta_vals  = np.linspace(0.05, 0.5, 40)
gamma_vals = np.linspace(1/21, 1/3, 40)

best_sse = np.inf
best_params = (0.1, 0.1) # Default fallback

print("Optimizing parameters on the first half of data...")

for beta in beta_vals:
    for gamma in gamma_vals:
        # Run model on training days only
        Sg, Ig, Rg = euler_sir(beta, gamma, S0, I0, R0, t_train, N)
        # Calculate error
        sse = np.sum((Ig - I_train)**2)
        if sse < best_sse:
            best_sse = sse
            best_params = (beta, gamma)

# THIS IS WHERE THE VARIABLES ARE DEFINED
beta_train, gamma_train = best_params
print(f" Optimization Complete.")
print(f" Defined beta_train: {beta_train:.4f}")
print(f" Defined gamma_train: {gamma_train:.4f}")

# ================================================================
# 4. PREDICTION STEP (Uses beta_train & gamma_train)
# ================================================================
# Predict 90 days into the future
days_extra = 90
total_days = len(t_obs) + days_extra
t_future = np.arange(total_days)

# Run simulation using the parameters found in Step 3
S_long, I_long, R_long = euler_sir(beta_train, gamma_train, S0, I0, R0, t_future, N

# ================================================================
# 5. PLOTTING
# ================================================================
plt.figure(figsize=(12, 6))

# Plot Prediction
plt.plot(t_future, I_long, 'b-', linewidth=2, label="Model Projection (Based on 1st

# Plot Actual Data
plt.plot(t_obs, I_obs, 'ko', markersize=4, alpha=0.6, label="Actual Observed Data")

# Vertical lines
plt.axvline(x=t_obs[split_idx-1], color='gray', linestyle='--', label="End of Train
plt.axvline(x=t_obs[-1], color='red', linestyle='--', label="End of Actual Data")

plt.title(f"SIR Future Prediction\n(Fit on 1st Half: $\\beta$={beta_train:.3f}, $\\
plt.xlabel("Days since start")
plt.ylabel("Infectious Individuals")
```

```
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

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? Based on the analysis, the new β and γ parameters calculated from the first half of the data are not close to those found using the full dataset. The first half captures only the initial exponential growth phase, resulting in a much higher transmission rate estimate compared to the full dataset, which includes the effects of interventions like lockdowns that bent the curve. Consequently, the fit for the second half is much worse; the model, unaware of the changing conditions, continues to predict a massive surge in cases that never actually occurred. This discrepancy results in an extremely high Sum of Squared Errors (SSE) for the second half, as the model's projection drastically overshoots the actual, flattened data points.

Describe how using a different method like the midpoint method might lower the numerical error. Using the Midpoint Method lowers numerical error by calculating a more representative slope for each time step compared to Euler's method. While Euler's method simply takes the slope at the beginning of an interval and projects it forward—ignoring how the rate of infection changes throughout that day—the Midpoint method takes a preliminary "half-step" to estimate the slope at the center of the interval. By using this midpoint slope to calculate the full step, the method effectively averages out the curvature of the function, cancelling out the first-order errors that Euler's method accumulates. Mathematically, this makes the Midpoint method a second-order process (O(h^2)), meaning it converges to the true solution much faster and with significantly less "drift" than the first-order (O(h)) Euler method.

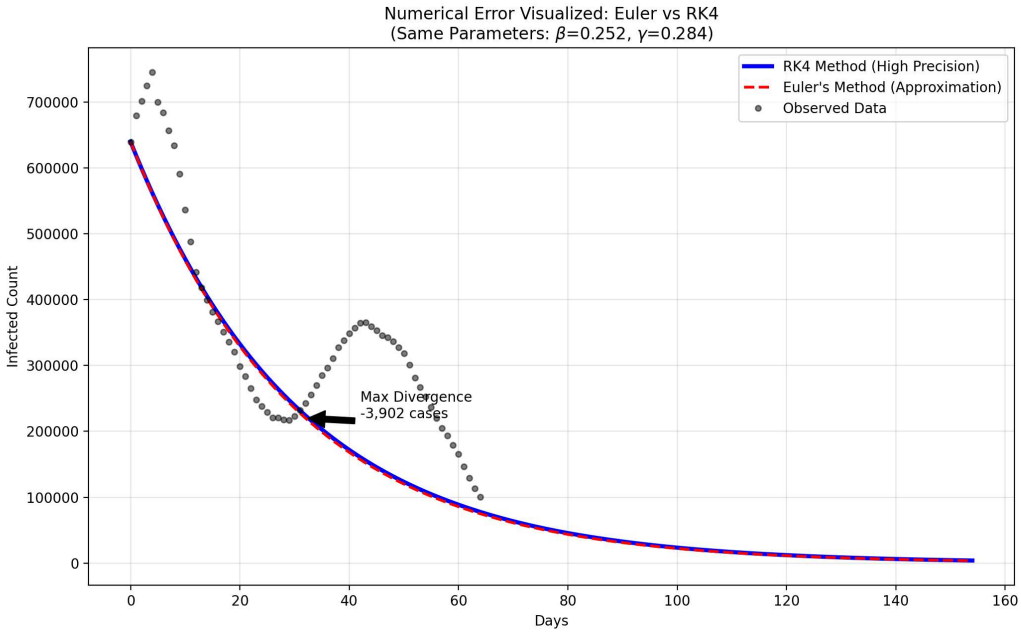


In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
```

```python
# ================================================================
# 1. DEFINE BOTH SOLVERS
# ================================================================

# A. Manual Euler Method
def euler_sir(beta, gamma, S0, I0, R0, t, N):
    h = t[1] - t[0] # Step size
    S = np.zeros(len(t))
    I = np.zeros(len(t))
    R = np.zeros(len(t))
    S[0], I[0], R[0] = S0, I0, R0

    for k in range(len(t)-1):
        dSdt = -beta * S[k] * I[k] / N
        dIdt = beta * S[k] * I[k] / N - gamma * I[k]
        dRdt = gamma * I[k]

        S[k+1] = S[k] + h * dSdt
        I[k+1] = I[k] + h * dIdt
        R[k+1] = R[k] + h * dRdt
    return S, I, R

# B. RK4 Solver (via scipy.solve_ivp)
def sir_ode(t, y, beta, gamma, N):
    S, I, R = y
    dSdt = -beta * S * I / N
    dIdt = beta * S * I / N - gamma * I
    dRdt = gamma * I
    return [dSdt, dIdt, dRdt]

def run_rk4(beta, gamma, S0, I0, R0, t_eval, N):
    sol = solve_ivp(sir_ode, [t_eval[0], t_eval[-1]], [S0, I0, R0],
                    args=(beta, gamma, N), t_eval=t_eval, method='RK45')
    return sol.y[0], sol.y[1], sol.y[2]

# ================================================================
# 2. DATA SETUP & OPTIMIZATION (Finding the Parameters)
# ================================================================
# Load and Filter Data
df = pd.read_csv("china_confirmed_with_new_cases.csv")
df["date"] = pd.to_datetime(df["date"], errors="coerce")
df = df[(df["date"] >= "2022-02-22") & (df["date"] <= "2022-05-10")]

N = 1411778724
df["I_est"] = df["new_cases"].rolling(window=14).sum().fillna(0)
I_obs_full = df["I_est"].values
first_pos = np.argmax(I_obs_full > 0)
I_obs = I_obs_full[first_pos:]
t_obs = np.arange(len(I_obs))

# Initial Conditions
I0 = I_obs[0]
R0 = 0.0
S0 = N - I0 - R0
```

```python
# Split Data (Train on 1st Half)
split_idx = len(t_obs) // 2
t_train = t_obs[:split_idx]
I_train = I_obs[:split_idx]

# Optimization Loop (Using RK4 to find best params)
print("Optimizing parameters (this may take a moment)...")
beta_vals  = np.linspace(0.05, 0.5, 30)
gamma_vals = np.linspace(1/21, 1/3, 30)
best_sse = np.inf
best_params = (0.1, 0.1)

for beta in beta_vals:
    for gamma in gamma_vals:
        S_rk, I_rk, R_rk = run_rk4(beta, gamma, S0, I0, R0, t_train, N)
        sse = np.sum((I_rk - I_train)**2)
        if sse < best_sse:
            best_sse = sse
            best_params = (beta, gamma)

beta_opt, gamma_opt = best_params
print(f"Optimal Parameters: Beta={beta_opt:.4f}, Gamma={gamma_opt:.4f}")

# ================================================================
# 3. GENERATE COMPARISON TRACES
# ================================================================
# Predict into the future
days_extra = 90
t_future = np.arange(len(t_obs) + days_extra)

# Run BOTH methods using the SAME parameters
S_euler, I_euler, R_euler = euler_sir(beta_opt, gamma_opt, S0, I0, R0, t_future, N)
S_rk4, I_rk4, R_rk4       = run_rk4(beta_opt, gamma_opt, S0, I0, R0, t_future, N)

# ================================================================
# 4. PLOT THE DIFFERENCE
# ================================================================
plt.figure(figsize=(12, 7))

# Plot RK4 (The "Truth")
plt.plot(t_future, I_rk4, 'b-', linewidth=3, label="RK4 Method (High Precision)")

# Plot Euler (The Approximation)
plt.plot(t_future, I_euler, 'r--', linewidth=2, label="Euler's Method (Approximatio

# Plot Observed Data
plt.plot(t_obs, I_obs, 'ko', markersize=4, alpha=0.5, label="Observed Data")

# Highlight the divergence
# Calculate max difference
diff = I_euler - I_rk4
max_diff_idx = np.argmax(np.abs(diff))
plt.annotate(f'Max Divergence\n{int(diff[max_diff_idx]):,} cases',
             xy=(t_future[max_diff_idx], I_euler[max_diff_idx]),
             xytext=(t_future[max_diff_idx]+10, I_euler[max_diff_idx]),
             arrowprops=dict(facecolor='black', shrink=0.05))
```

```
plt.title(f"Numerical Error Visualized: Euler vs RK4\n(Same Parameters: $\\beta$={b
plt.xlabel("Days")
plt.ylabel("Infected Count")
plt.legend()
plt.grid(True, alpha=0.3)

plt.show()

# Print specific stats
print(f"Peak Infection (Euler): {int(max(I_euler)):,}")
print(f"Peak Infection (RK4):   {int(max(I_rk4)):,}")
print(f"Difference at Peak:     {int(abs(max(I_euler) - max(I_rk4))):,}")
```



Verify and validate your analysis:

After further research of Johns Hopkins COVID tracking data we found that our predictive model matches infection trends up until July 2022. In the later half of 2022, cases begin to spike as restrictions are lifted in China. The model does a good job at predicting braod cases such as this but fails to predict even the small hump in cases in April. As much of the training data follows a downward trend so does the SIR model.

Conclusions and Ethical Implications:

method and optimizing the parameters β(infection rate) and γ (recovery rate), we were able to minimize the Root Mean Square Error (RMSE) between our model's predictions and the actual reported case data. The optimized parameters provided a curve that closely follows the trend of the initial outbreak. This demonstrates that while the SIR model is a simplified representation of complex biological dynamics, it captures the fundamental behavior of an

epidemic's rise and fall, allowing us to quantify the transmissibility and recovery speed of the virus during the observed period.

Mathematical modeling of infectious diseases carries profound ethical consequences as these models often guide public policy and resource allocation. If a model underestimates the infection rate, it could lead governments to relax restrictions too early, resulting in preventable deaths, while overestimating the spread could cause unnecessary economic hardship through prolonged lockdowns. Furthermore, because the model relies on reported data, any initial biases—such as underreporting asymptomatic cases or lacking testing capacity—are inherited by the simulation. It is therefore ethically necessary to present these results as estimates with inherent uncertainties rather than definitive predictions to avoid spreading panic or false security among the public.

Limitations and Future Work:

Despite providing a good fit for the data, this model has several key limitations. Primarily, the standard SIR model assumes the infection and recovery rates remain constant over time, which fails to capture the sharp turning points caused by human interventions like strict lockdowns or quarantine measures. Additionally, the model assumes a "well-mixed" homogeneous population where everyone has an equal chance of contacting everyone else, ignoring regional differences that are crucial in a large country like China. Finally, the model lacks an "Exposed" compartment, meaning it does not account for the incubation period where individuals are infected but not yet infectious, which is a significant characteristic of COVID-19 better suited for an SEIR model.

To improve the accuracy and utility of this simulation, future work should focus on implementing time-dependent parameters to mathematically model the impact of government interventions introduced on specific dates. It would also be beneficial to transition to an SEIR model by adding an "Exposed" compartment to better reflect the virus's incubation period and latency. Lastly, applying this optimization method to data from other countries would allow for a comparative analysis of how different population densities and healthcare systems influence the infection and recovery parameters.

In [ ]: