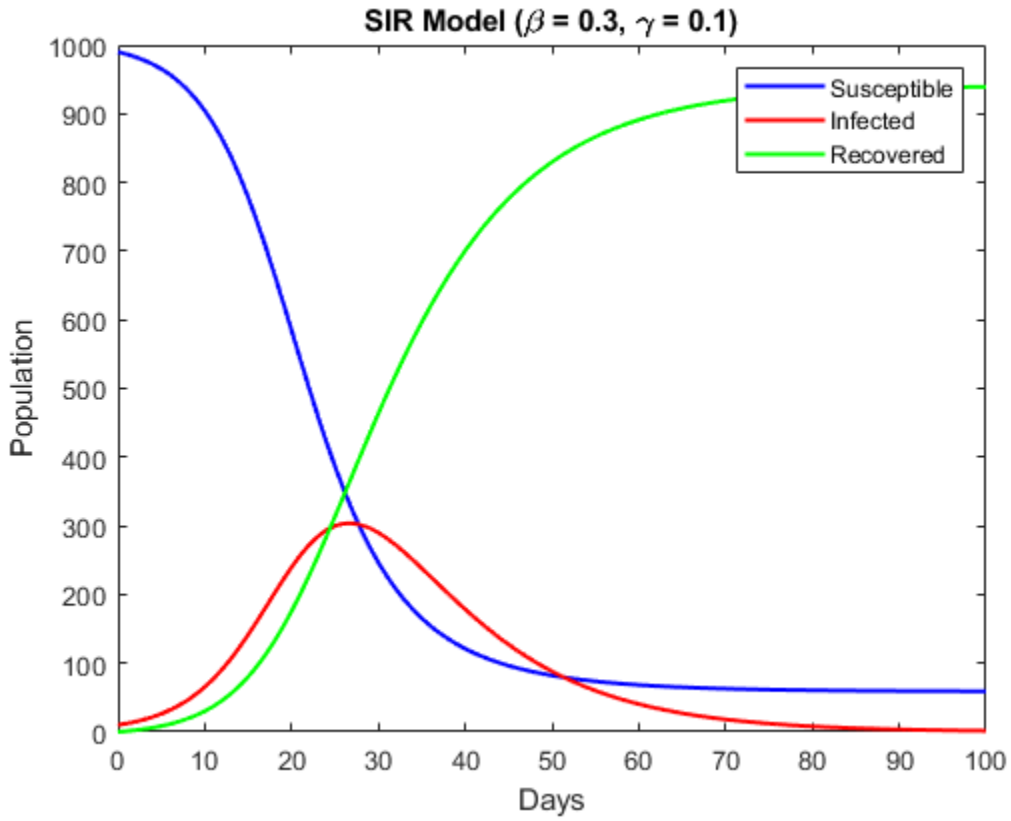
Table of Contents

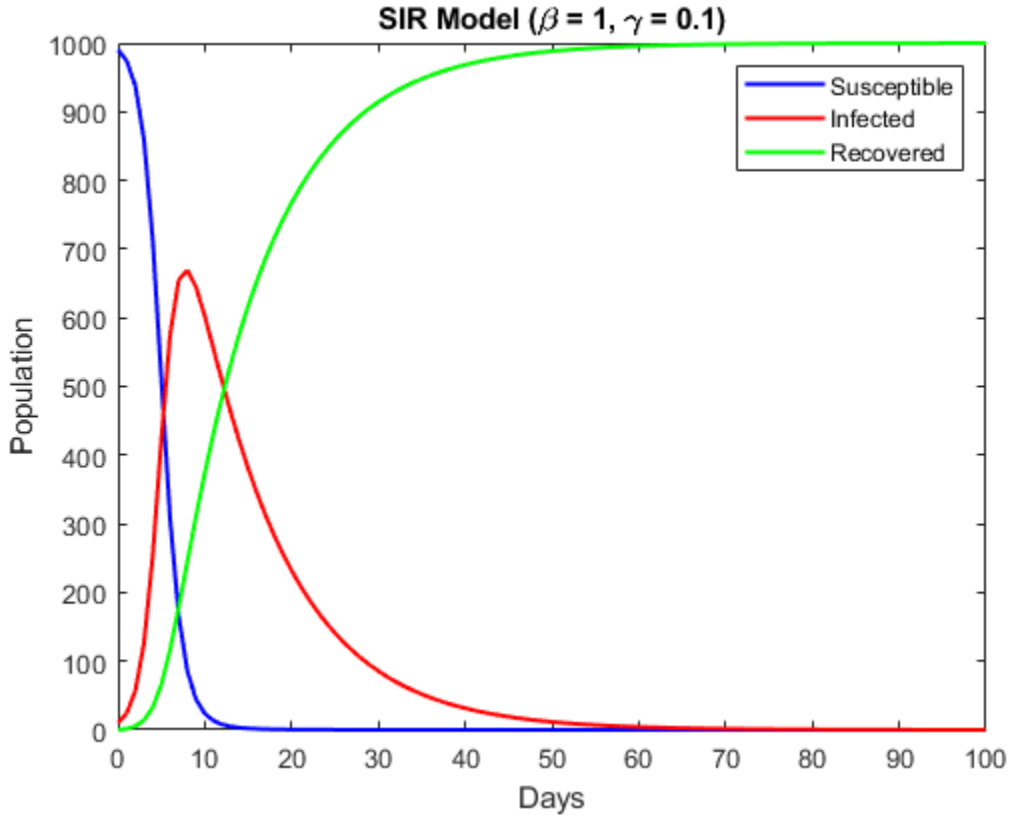
Part 1: Modeling Disease Spread

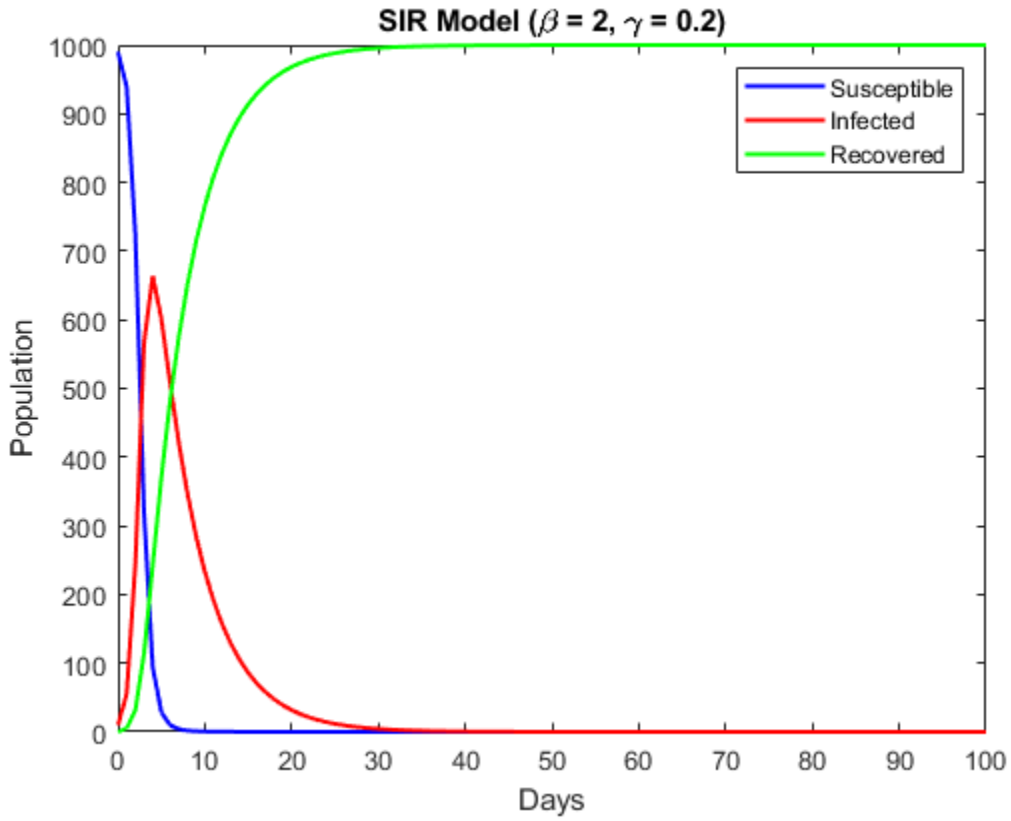
Parameters

```
clc
clear all
Total_pop = 1000; % Total population
Sim_days = 100;   % Duration of the simulation in days
T_step = 1;       % Time step in days

% Initial conditions
S0 = 990; %initial suceptible
I0 = 10;  %initial infected
R0 = 0;   %initial recovered

% Parameters for diseases
dis_params = [0.3, 0.1;  % Seasonal Influenza
              1.0, 0.1;  % COVID-19
             2.0, 0.2]; % Measles

% Loop through each disease scenario
for cdc = 1:size(dis_params, 1)

    trans_rate = dis_params(cdc, 1); %transmission rate
    rec_rate = dis_params(cdc, 2);   %recovery rate

    % Initialize arrays for S, I, R values
    susceptible = zeros(1, Sim_days + 1);
    infected = zeros(1, Sim_days + 1);
    recovered = zeros(1, Sim_days + 1);

    % Set initial values
    susceptible(1) = S0;
    infected(1) = I0;
    recovered(1) = R0;

    % Using 4th-order Runge-Kutta method
    for t = 1:Sim_days
        % Calculate k1 values
        k1_s = -trans_rate * susceptible(t) * infected(t) / Total_pop;
        k1_i = (trans_rate * susceptible(t) * infected(t) / Total_pop) -
rec_rate * infected(t);
        k1_r = rec_rate * infected(t);
```

```
        % Calculate k2 values
        k2_s = -trans_rate * (susceptible(t) + 0.5 * T_step * k1_s) *
(infected(t) + 0.5 * T_step * k1_i) / Total_pop;
        k2_i = (trans_rate * (susceptible(t) + 0.5 * T_step * k1_s) *
(infected(t) + 0.5 * T_step * k1_i) / Total_pop) ...
                - rec_rate * (infected(t) + 0.5 * T_step * k1_i);
        k2_r = rec_rate * (infected(t) + 0.5 * T_step * k1_i);

        % Calculate k3 values
        k3_s = -trans_rate * (susceptible(t) + 0.5 * T_step * k2_s) *
(infected(t) + 0.5 * T_step * k2_i) / Total_pop;
        k3_i = (trans_rate * (susceptible(t) + 0.5 * T_step * k2_s) *
(infected(t) + 0.5 * T_step * k2_i) / Total_pop) ...
                - rec_rate * (infected(t) + 0.5 * T_step * k2_i);
        k3_r = rec_rate * (infected(t) + 0.5 * T_step * k2_i);

        % Calculate k4 values
        k4_s = -trans_rate * (susceptible(t) + T_step * k3_s) * (infected(t)
+ T_step * k3_i) / Total_pop;
        k4_i = (trans_rate * (susceptible(t) + T_step * k3_s) * (infected(t)
+ T_step * k3_i) / Total_pop) ...
                - rec_rate * (infected(t) + T_step * k3_i);
        k4_r = rec_rate * (infected(t) + T_step * k3_i);

        % Update values using weighted average of k1, k2, k3, k4
        susceptible(t + 1) = susceptible(t) + (T_step / 6) * (k1_s + 2 *
k2_s + 2 * k3_s + k4_s);
        infected(t + 1) = infected(t) + (T_step / 6) * (k1_i + 2 * k2_i + 2
* k3_i + k4_i);
        recovered(t + 1) = recovered(t) + (T_step / 6) * (k1_r + 2 * k2_r +
2 * k3_r + k4_r);
    end

    % Generate plots
    figure;
    plot(0:Sim_days, susceptible, 'b-', 'LineWidth', 1.5); hold on;
    plot(0:Sim_days, infected, 'r-', 'LineWidth', 1.5);
    plot(0:Sim_days, recovered, 'g-', 'LineWidth', 1.5);
    hold off;
    xlabel('Days');
    ylabel('Population');
    legend('Susceptible', 'Infected', 'Recovered');
    title(['SIR Model (\beta = ', num2str(trans_rate), ', \gamma = ',
num2str(rec_rate), ')']);
end
```


SIR Model (β = 0.3, γ = 0.1)
Susceptible
Infected
Recovered
Population
Days
0
100
200
300
400
500
600
700
800
900
1000
0
10
20
30
40
50
60
70
80
90
100


SIR Model (β = 1, γ = 0.1)
Susceptible
Infected
Recovered
Population
Days
0
100
200
300
400
500
600
700
800
900
1000
0
10
20
30
40
50
60
70
80
90
100



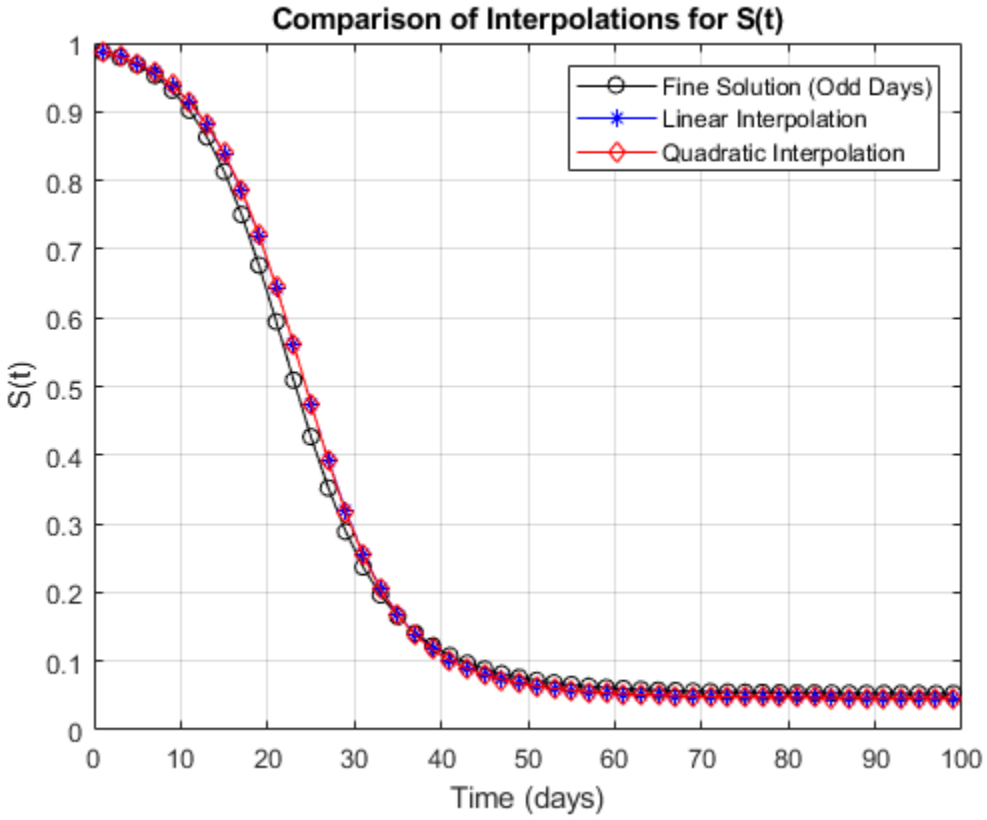
Part 2: Interpolation

```
trans_rate = 0.3;  % rate of infection
rec_rate = 0.1; % rate of recovery
Total_pop = 1;       % size of population
Sim_days = 100;     % total simulation
h1 = 1;       % finer
h2 = 2;       % coarser
S0 = 0.99; % susceptible
I0 = 0.01; % infected
R0 = 0;     % recovered
t_f = 0:h1:Sim_days; % fine steps
t_c = 0:h2:Sim_days; % coarse steps

%  time step (finer)
S_f = zeros(size(t_f));
I_f = zeros(size(t_f));
R_f = zeros(size(t_f));

% initial conditions
S_f(1) = S0;
I_f(1) = I0;
R_f(1) = R0;

% time step
```

```
for k = 1:length(t_f)-1
    dS = -(trans_rate/Total_pop) * S_f(k) * I_f(k);
    dI = (trans_rate/Total_pop) * S_f(k) * I_f(k) - rec_rate * I_f(k);
    dR = rec_rate * I_f(k);

    S_f(k+1) = S_f(k) + h1 * dS;
    I_f(k+1) = I_f(k) + h1 * dI;
    R_f(k+1) = R_f(k) + h1 * dR;
end

% coarser step
S_c = zeros(size(t_c));
I_c= zeros(size(t_c));
R_c = zeros(size(t_c));

%intial conditons
S_c(1) = S0;
I_c(1) = I0;
R_c(1) = R0;

% time step
for k = 1:length(t_c)-1
    dS = -(trans_rate/Total_pop) * S_c(k) * I_c(k);
    dI = (trans_rate/Total_pop) * S_c(k) * I_c(k) - rec_rate * I_c(k);
    dR = rec_rate * I_c(k);

    S_c(k+1) = S_c(k) + h2 * dS;
    I_c(k+1) = I_c(k) + h2 * dI;
    R_c(k+1) = R_c(k) + h2 * dR;
end

% interpolation of odd days
t_odd = 1:2:Sim_days-1;

% coaser linear interpolation
S_l = interp1(t_c, S_c, t_odd, 'linear');
I_l = interp1(t_c, I_c, t_odd, 'linear');
R_l = interp1(t_c, R_c, t_odd, 'linear');

% lagrange
S_q = interp1(t_c, S_c, t_odd, 'spline');
I_q = interp1(t_c, I_c, t_odd, 'spline');
R_q = interp1(t_c, R_c, t_odd, 'spline');

% finer odd
S_f_odd = interp1(t_f, S_f, t_odd);
I_f_odd = interp1(t_f, I_f, t_odd);
R_f_odd = interp1(t_f, R_f, t_odd);

% linear interpolation
Nint = length(t_odd);
EL2_S_l = sqrt(sum((S_l - S_f_odd).^2) / Nint);
EL2_I_l = sqrt(sum((I_l - I_f_odd).^2) / Nint);
EL2_R_l = sqrt(sum((R_l - R_f_odd).^2) / Nint);
```

```
% quad. interpolation
el2_S_q = sqrt(sum((S_q - S_f_odd).^2) / Nint);
el2_I_q = sqrt(sum((I_q - I_f_odd).^2) / Nint);
el2_R_q = sqrt(sum((R_q - R_f_odd).^2) / Nint);

% error table
ErrorTable = table(["Linear"; "Quadratic"], ...
                   [EL2_S_l; el2_S_q], ...
                   [EL2_I_l; el2_I_q], ...
                   [EL2_R_l; el2_R_q], ...
                   'VariableNames', {'Interpolation', 'S_Error', 'I_Error',
'R_Error'});

% error table
disp(ErrorTable);

% plot
figure;
plot(t_odd, S_f_odd, 'k-o', 'DisplayName', 'Fine Solution (Odd Days)');
hold on;
plot(t_odd, S_l, 'b-*', 'DisplayName', 'Linear Interpolation');
plot(t_odd, S_q, 'r-d', 'DisplayName', 'Quadratic Interpolation');
xlabel('Time (days)');
ylabel('S(t)');
legend;
title('Comparison of Interpolations for S(t)');
grid on;
```

Interpolation	*S_Error*	*I_Error*	*R_Error*
"Linear"	*0.018109*	*0.010584*	*0.016495*
"Quadratic"	*0.018253*	*0.010796*	*0.016658*



Part 3: Least Squares

```
Total_pop = 1000;
T_step = 1;
Sim_days = 30;
trans_rate = 0.3;
rec_rate = 0.1;
S0 = 990;
I0 = 10;
R0 = 0;

% Initialize arrays for susceptible, infected, and recovered individuals
susceptible = zeros(1, Sim_days + 1);
infected = zeros(1, Sim_days + 1);
recovered = zeros(1, Sim_days + 1);
susceptible(1) = S0;
infected(1) = I0;
recovered(1) = R0;

for t = 1:Sim_days
    % Calculate k1 values
    k1_s = -trans_rate * susceptible(t) * infected(t) / Total_pop;
    k1_i = (trans_rate * susceptible(t) * infected(t) / Total_pop) -
rec_rate * infected(t);
    k1_r = rec_rate * infected(t);
```

```
    % Calculate k2 values
    k2_s = -trans_rate * (susceptible(t) + 0.5 * T_step * k1_s) *
(infected(t) + 0.5 * T_step * k1_i) / Total_pop;
    k2_i = (trans_rate * (susceptible(t) + 0.5 * T_step * k1_s) *
(infected(t) + 0.5 * T_step * k1_i) / Total_pop) - ...
           rec_rate * (infected(t) + 0.5 * T_step * k1_i);
    k2_r = rec_rate * (infected(t) + 0.5 * T_step * k1_i);

    % Calculate k3 values
    k3_s = -trans_rate * (susceptible(t) + 0.5 * T_step * k2_s) *
(infected(t) + 0.5 * T_step * k2_i) / Total_pop;
    k3_i = (trans_rate * (susceptible(t) + 0.5 * T_step * k2_s) *
(infected(t) + 0.5 * T_step * k2_i) / Total_pop) - ...
           rec_rate * (infected(t) + 0.5 * T_step * k2_i);
    k3_r = rec_rate * (infected(t) + 0.5 * T_step * k2_i);

    % Calculate k4 values
    k4_s = -trans_rate * (susceptible(t) + T_step * k3_s) * (infected(t) +
T_step * k3_i) / Total_pop;
    k4_i = (trans_rate * (susceptible(t) + T_step * k3_s) * (infected(t) +
T_step * k3_i) / Total_pop) - ...
           rec_rate * (infected(t) + T_step * k3_i);
    k4_r = rec_rate * (infected(t) + T_step * k3_i);

    % Update values using weighted average of k1, k2, k3, k4
    susceptible(t + 1) = susceptible(t) + (T_step / 6) * (k1_s + 2 * k2_s +
2 * k3_s + k4_s);
    infected(t + 1) = infected(t) + (T_step / 6) * (k1_i + 2 * k2_i + 2 *
k3_i + k4_i);
    recovered(t + 1) = recovered(t) + (T_step / 6) * (k1_r + 2 * k2_r + 2 *
k3_r + k4_r);
end

% Least squares setup
t = 1:Sim_days;
Y = log(infected(1:end-1));
X = t(:);
IF = infected(end);

% estimation of k using t and I(t) arrays
k30 = sum(Y) / sum(X);
I0_est30 = exp(k30*(Sim_days) - log(IF));
trans_rate = (Total_pop/S0)*(k30 + rec_rate);

disp('Transmission Rate Beta using 30 days: ')
disp(trans_rate)
disp('estimated I0 using 30 days: ')
disp(I0_est30)

% Least squares setup
Sim_days = 10;
t = 1:Sim_days;
Y = log(infected(1:end-1));
```

```
X = t(:);
IF = infected(end);

% estimation of k using t and I(t) arrays
k10 = sum(Y) / sum(X);
trans_rate = (Total_pop/S0)*(k10 + rec_rate);
I0_est10 = exp(k10*(Sim_days) - log(IF));

disp('Transmission Rate Beta using 10 days: ')
disp(trans_rate)
disp('estimated I0 using 10 days: ')
disp(I0_est10)
```

```
Transmission Rate Beta using 30 days:
    0.3978

estimated I0 using 30 days:
   23.2251

Transmission Rate Beta using 10 days:
    2.6104

estimated I0 using 10 days:
   2.1216e+08
```

Part 4: Fourier Analysis

Parameters

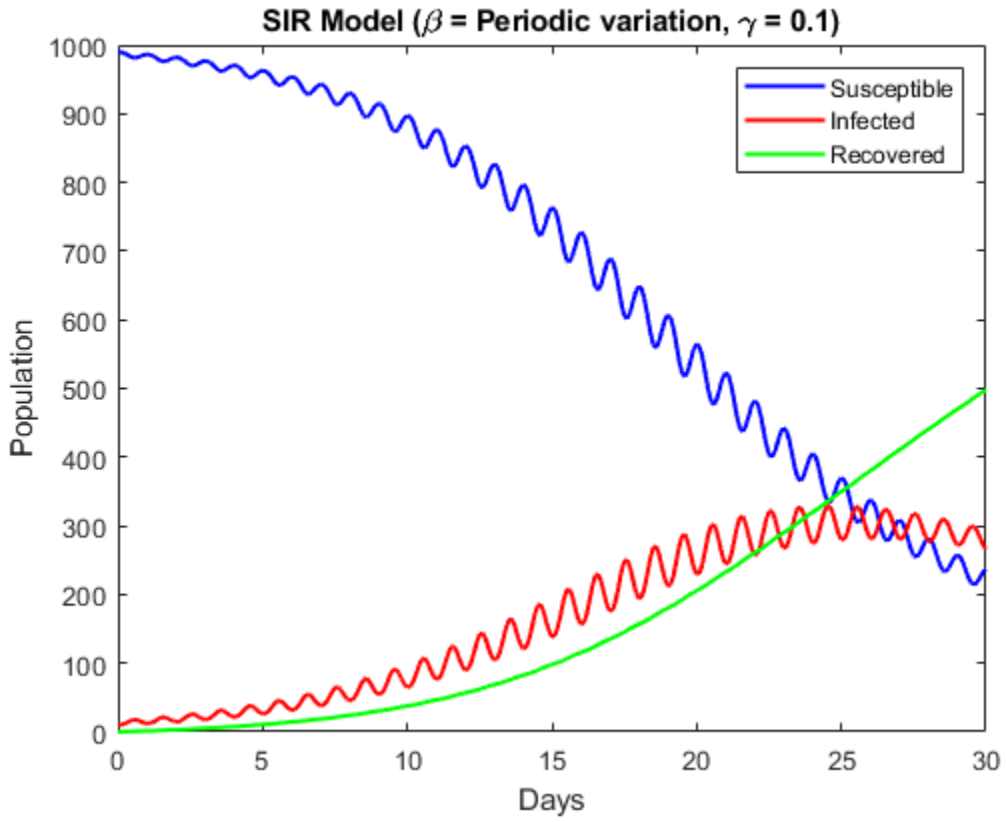
```
Total_pop = 1000; % Total population
Sim_days = 30;   % Duration of the simulation in days
T_step = 0.1;       % Time step in days

% Initial conditions
S0 = 990; %initial suceptible
I0 = 10;  %initial infected
R0 = 0;   %initial recovered

%Trans rate and recovery rate
trans_rate = @ (x) (0.3*(1+5*sin(2*pi*x))); %transmission raten1
rec_rate = 0.1;   %recovery rate

% Initialize arrays for S, I, R values
susceptible = zeros(1, Sim_days/T_step + 1);
infected = zeros(1, Sim_days/T_step + 1);
recovered = zeros(1, Sim_days/T_step + 1);

% Set initial values
susceptible(1) = S0;
infected(1) = I0;
recovered(1) = R0;
n = 1;
```

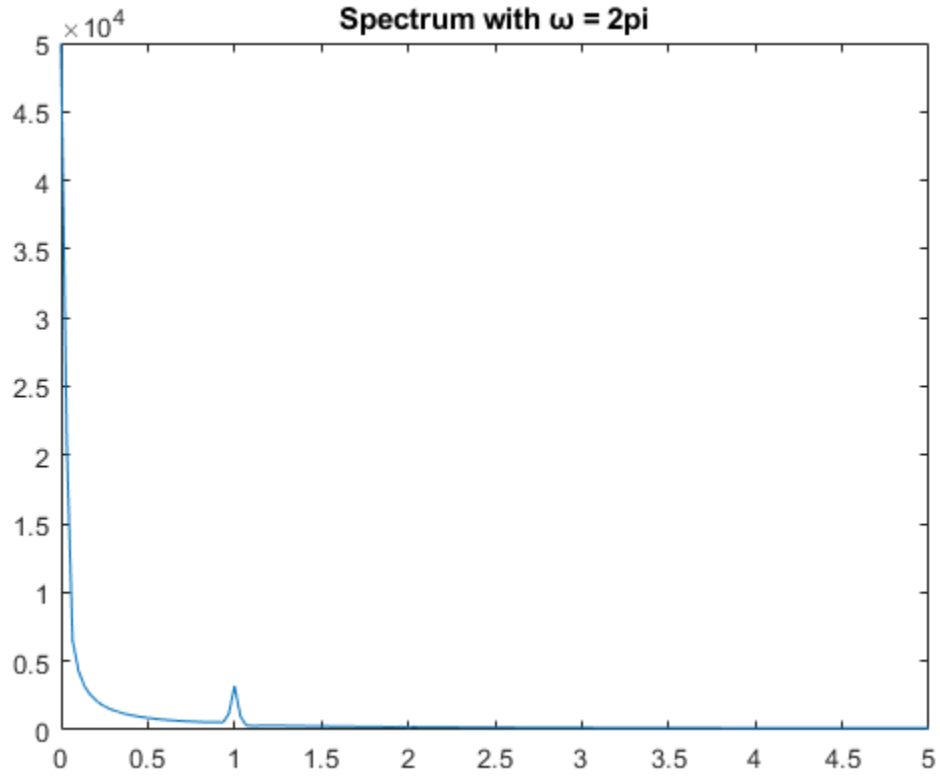
```
% Using 4th-order Runge-Kutta method
for t = 0:0.1:Sim_days-0.1
    % Calculate k1 values
    k1_s = -trans_rate(t) * susceptible(n) * infected(n) / Total_pop;
    k1_i = (trans_rate(t) * susceptible(n) * infected(n) / Total_pop) -
rec_rate * infected(n);
    k1_r = rec_rate * infected(n);

    % Calculate k2 values
    k2_s = -trans_rate(t) * (susceptible(n) + 0.5 * T_step * k1_s) *
(infected(n) + 0.5 * T_step * k1_i) / Total_pop;
    k2_i = (trans_rate(t) * (susceptible(n) + 0.5 * T_step * k1_s) *
(infected(n) + 0.5 * T_step * k1_i) / Total_pop) ...
        - rec_rate * (infected(n) + 0.5 * T_step * k1_i);
    k2_r = rec_rate * (infected(n) + 0.5 * T_step * k1_i);

    % Calculate k3 values
    k3_s = -trans_rate(t) * (susceptible(n) + 0.5 * T_step * k2_s) *
(infected(n) + 0.5 * T_step * k2_i) / Total_pop;
    k3_i = (trans_rate(t) * (susceptible(n) + 0.5 * T_step * k2_s) *
(infected(n) + 0.5 * T_step * k2_i) / Total_pop) ...
        - rec_rate * (infected(n) + 0.5 * T_step * k2_i);
    k3_r = rec_rate * (infected(n) + 0.5 * T_step * k2_i);

    % Calculate k4 values
    k4_s = -trans_rate(t) * (susceptible(n) + T_step * k3_s) * (infected(n)
+ T_step * k3_i) / Total_pop;
    k4_i = (trans_rate(t) * (susceptible(n) + T_step * k3_s) * (infected(n)
+ T_step * k3_i) / Total_pop) ...
        - rec_rate * (infected(n) + T_step * k3_i);
    k4_r = rec_rate * (infected(n) + T_step * k3_i);

    % Update values using weighted average of k1, k2, k3, k4
    susceptible(n + 1) = susceptible(n) + (T_step / 6) * (k1_s + 2 * k2_s +
2 * k3_s + k4_s);
    infected(n + 1) = infected(n) + (T_step / 6) * (k1_i + 2 * k2_i + 2 *
k3_i + k4_i);
    recovered(n + 1) = recovered(n) + (T_step / 6) * (k1_r + 2 * k2_r + 2 *
k3_r + k4_r);

    n=n+1;
end

% Generate plot
figure;
plot(0:0.1:Sim_days, susceptible, 'b-', 'LineWidth', 1.5); hold on;
plot(0:0.1:Sim_days, infected, 'r-', 'LineWidth', 1.5);
plot(0:0.1:Sim_days, recovered, 'g-', 'LineWidth', 1.5);
hold off;
xlabel('Days');
ylabel('Population');
legend('Susceptible', 'Infected', 'Recovered');
title(['SIR Model (\beta = ', 'Periodic variation', ', \gamma = ',
num2str(rec_rate), ')']);
```

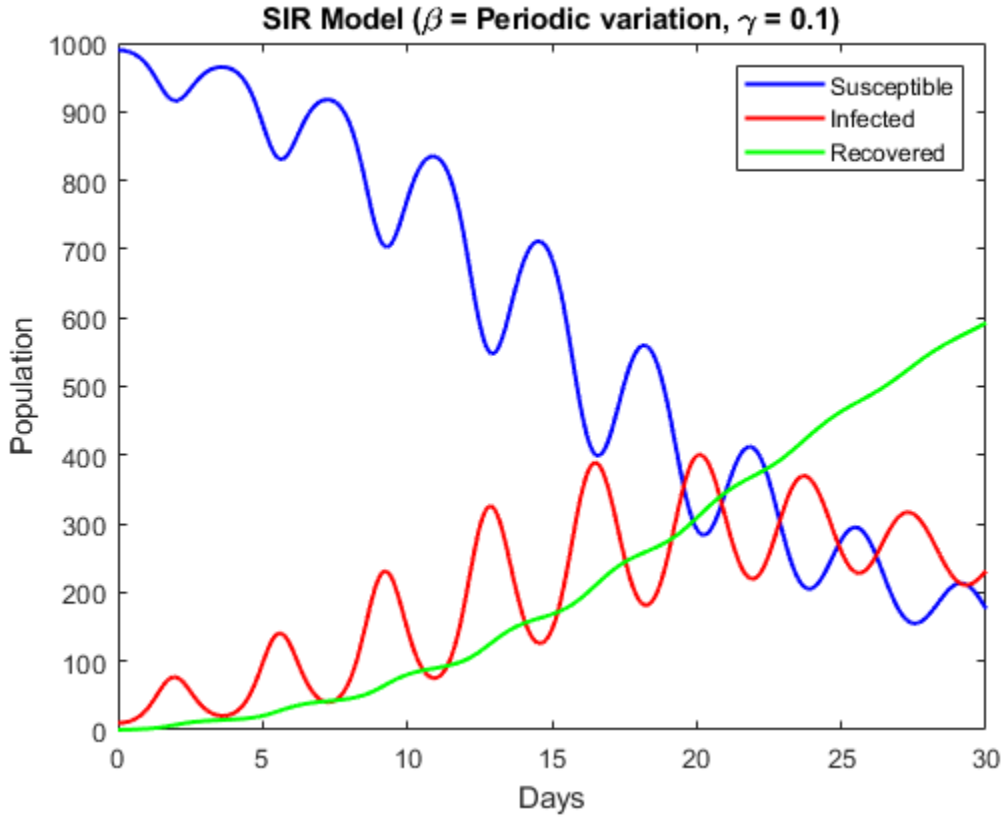
```
% Fourier transform
susceptiblefft = fft(susceptible);
infectedfft = fft(infected);
recoveredfft = fft(recovered);

% Define frequency
T = Sim_days;
N = 300;
f = 1/T.*(0:N/2);

% Plot
figure;
plot(f,abs(infectedfft(1:N/2+1)))
hold on
title('Spectrum with ω = 2pi')

% Replace ω with ω = 2π × 100/365
% Parameters
Total_pop = 1000; % Total population
Sim_days = 30;   % Duration of the simulation in days
T_step = 0.1;       % Time step in days

% Initial conditions
S0 = 990; %initial suceptible
I0 = 10;  %initial infected
R0 = 0;   %initial recovered

%Trans rate and recovery rate
trans_rate = @ (x) (0.3*(1+5*sin(2*100/365*pi*x))); %transmission raten1
rec_rate = 0.1;   %recovery rate

% Initialize arrays for S, I, R values
susceptible1 = zeros(1, Sim_days/T_step + 1);
infected1 = zeros(1, Sim_days/T_step + 1);
recovered1 = zeros(1, Sim_days/T_step + 1);

% Set initial values
susceptible1(1) = S0;
infected1(1) = I0;
recovered1(1) = R0;
n = 1;

% Using 4th-order Runge-Kutta method
for t = 0:0.1:Sim_days-0.1
    % Calculate k1 values
    k1_s = -trans_rate(t) * susceptible1(n) * infected1(n) / Total_pop;
    k1_i = (trans_rate(t) * susceptible1(n) * infected1(n) / Total_pop) -
rec_rate * infected1(n);
    k1_r = rec_rate * infected1(n);

    % Calculate k2 values
    k2_s = -trans_rate(t) * (susceptible1(n) + 0.5 * T_step * k1_s) *
```

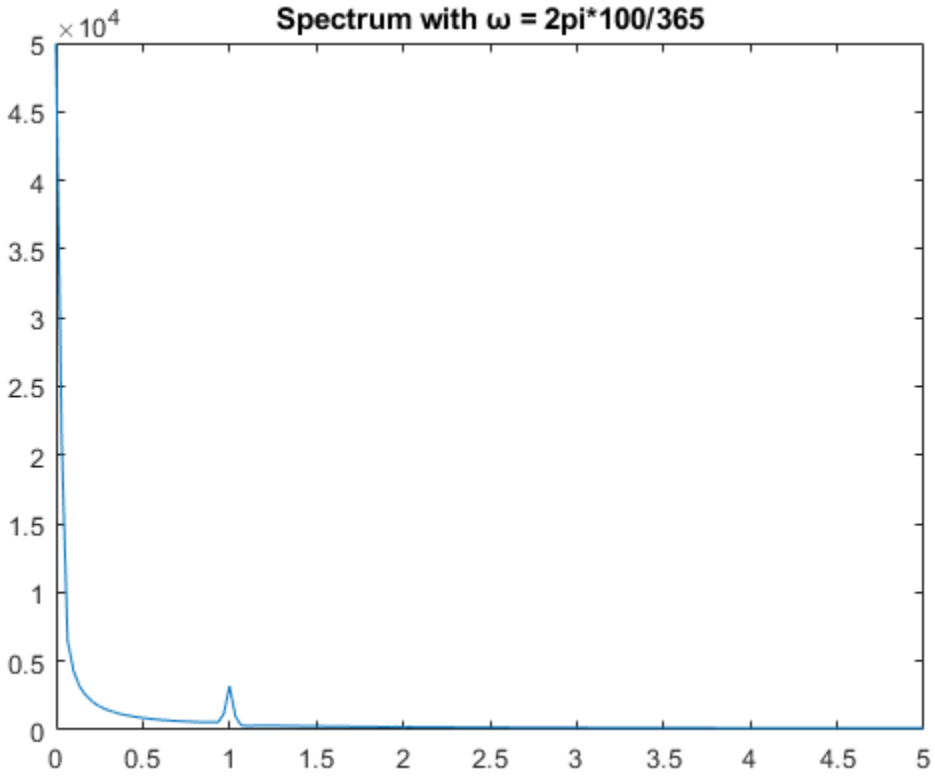
```
(infected1(n) + 0.5 * T_step * k1_i) / Total_pop;
    k2_i = (trans_rate(t) * (susceptible1(n) + 0.5 * T_step * k1_s) *
(infected1(n) + 0.5 * T_step * k1_i) / Total_pop) ...
        - rec_rate * (infected1(n) + 0.5 * T_step * k1_i);
    k2_r = rec_rate * (infected1(n) + 0.5 * T_step * k1_i);

    % Calculate k3 values
    k3_s = -trans_rate(t) * (susceptible1(n) + 0.5 * T_step * k2_s) *
(infected1(n) + 0.5 * T_step * k2_i) / Total_pop;
    k3_i = (trans_rate(t) * (susceptible1(n) + 0.5 * T_step * k2_s) *
(infected1(n) + 0.5 * T_step * k2_i) / Total_pop) ...
        - rec_rate * (infected1(n) + 0.5 * T_step * k2_i);
    k3_r = rec_rate * (infected1(n) + 0.5 * T_step * k2_i);

    % Calculate k4 values
    k4_s = -trans_rate(t) * (susceptible1(n) + T_step * k3_s) *
(infected1(n) + T_step * k3_i) / Total_pop;
    k4_i = (trans_rate(t) * (susceptible1(n) + T_step * k3_s) *
(infected1(n) + T_step * k3_i) / Total_pop) ...
        - rec_rate * (infected1(n) + T_step * k3_i);
    k4_r = rec_rate * (infected1(n) + T_step * k3_i);

    % Update values using weighted average of k1, k2, k3, k4
    susceptible1(n + 1) = susceptible1(n) + (T_step / 6) * (k1_s + 2 * k2_s
+ 2 * k3_s + k4_s);
    infected1(n + 1) = infected1(n) + (T_step / 6) * (k1_i + 2 * k2_i + 2 *
k3_i + k4_i);
    recovered1(n + 1) = recovered1(n) + (T_step / 6) * (k1_r + 2 * k2_r + 2
* k3_r + k4_r);

    n=n+1;
end

% Generate plot
figure;
plot(0:0.1:Sim_days, susceptible1, 'b-', 'LineWidth', 1.5); hold on;
plot(0:0.1:Sim_days, infected1, 'r-', 'LineWidth', 1.5);
plot(0:0.1:Sim_days, recovered1, 'g-', 'LineWidth', 1.5);
hold off;
xlabel('Days');
ylabel('Population');
legend('Susceptible', 'Infected', 'Recovered');
title(['SIR Model (\beta = ', 'Periodic variation', ', \gamma = ',
num2str(rec_rate), ')']);


% Fourier transform
susceptiblefft1 = fft(susceptible1);
infectedfft1 = fft(infected1);
recoveredfft1 = fft(recovered1);

% Define frequency
T = Sim_days;
N = 300;
```

```
f = 1/T.*(0:N/2);

% Plot
figure;
plot(f,abs(infectedfft(1:N/2+1)))
hold on
title('Spectrum with ω = 2pi*100/365')
```




Spectrum with ω = 2pi
×10^4
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


SIR Model (β = Periodic variation, γ = 0.1)
Susceptible
Infected
Recovered
Population
Days
1000
900
800
700
600
500
400
300
200
100
0
0
5
10
15
20
25
30



Published with MATLAB® R2023b

Part 1:

Provide a discussion of how the values of parameters β and γ effect the results. Explain whether your results make sense intuitively, taking into account what parameters β and γ physically represent in a disease spread model.

Ans:

The parameters β and γ play an important role in the SIR model. β which is the transmission rate shows how frequently a disease is spread. A higher β leads to faster disease spread. γ is the recovery rate, this represents the rate at which infected people recover. A higher γ reduces the time period of infection for people.

These results are consistent with the physical interpretations of β and γ, and they align themselves with real-world data for the modeled diseases in this project. For example, Measles (β=2.0,γ=0.2) exhibits rapid spread and recovery, leading to a very short but intense outbreak, whereas Seasonal Influenza (β=0.3,γ=0.1) spreads more gradually, with a longer duration and lower peak infections.

Part 2:

Comment which form of interpolation provides smaller errors.

Ans:

This part of the code used quadratic interpolation to provide smaller errors compared to linear interpolation. Quadratic interpolation uses higher orders of polynomials to give a closer approximation. Linear interpolation means the function will be linear between two coarse time steps and it performs better if the function is almost linear as well. However, larger errors can happen if the function has a significant curve. Quadratic interpolation can oscillate if the data changes abruptly.

The reason quadratic interpolation is the best answer is because it uses three points, which more easily captures the change between coarse time steps.

Part 3

Do the estimates of I(0) and Beta improve compared to true parameters?

Answer:

The values of Beta and infected at 0 days are higher when estimating using 30 and 10 days mainly because fewer people have recovered at this time. Beta being the transmission rate would be much higher if less time were given because people are more susceptible at the beginning of an epidemic. As for the initially infected individuals, the reason it is much larger using a smaller number of days is that it uses a line of best fit since the least square regression is a linear model rather than nonlinear.

Part 4:

Do you observe any periodic fluctuations in the signals due to the periodicity of β?

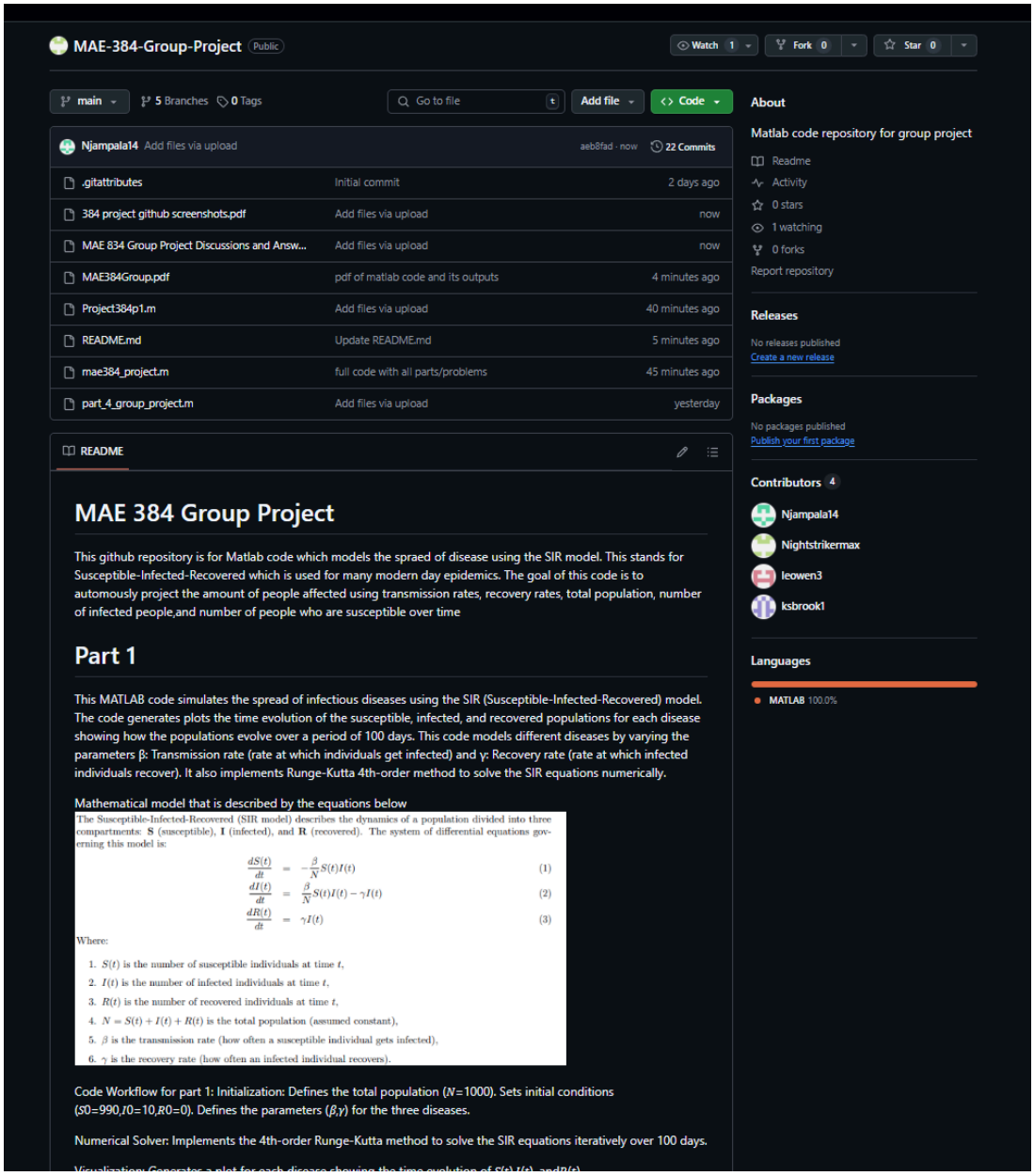
Yes, all of the signals fluctuate periodically due to the periodic transmission rate though the recovered line fluctuates far less than the other two. It is more noticeable on the graph where $\omega = 2\pi \times 100/365$.

Observe the frequency peak(s) and comment on what you see. Does it make sense physically?

The frequency peaks at 1. This does make sense physically and shows that 1 is the dominant frequency meaning it is the frequency where the infected cases fluctuate the most.

Names/ Usernames
Nischay Jampala-Njampala14
Luke Owen-leowen3
Connor Wilson-Nightstrikermax
Kenan Brooks-ksbrook1

(a) A screenshot of the main page of the project showing all uploaded files.



MAE-384-Group-Project Public

Watch 1 | Fork 0 | Star 0

main | 5 Branches | 0 Tags | Go to file | Add file | Code

Njampala14 Add files via upload aeb8fad · now 22 Commits

.gitattributes	Initial commit	2 days ago
384 project github screenshots.pdf	Add files via upload	now
MAE 834 Group Project Discussions and Answ...	Add files via upload	now
MAE384Group.pdf	pdf of matlab code and its outputs	4 minutes ago
Project384p1.m	Add files via upload	40 minutes ago
README.md	Update README.md	5 minutes ago
mae384_project.m	full code with all parts/problems	45 minutes ago
part_4_group_project.m	Add files via upload	yesterday

README

MAE 384 Group Project

This github repository is for Matlab code which models the spraed of disease using the SIR model. This stands for Susceptible-Infected-Recovered which is used for many modern day epidemics. The goal of this code is to automously project the amount of people affected using transmission rates, recovery rates, total population, number of infected people,and number of people who are susceptible over time

Part 1

This MATLAB code simulates the spread of infectious diseases using the SIR (Susceptible-Infected-Recovered) model. The code generates plots the time evolution of the susceptible, infected, and recovered populations for each disease showing how the populations evolve over a period of 100 days. This code models different diseases by varying the parameters β: Transmission rate (rate at which individuals get infected) and γ: Recovery rate (rate at which infected individuals recover). It also implements Runge-Kutta 4th-order method to solve the SIR equations numerically.

Mathematical model that is described by the equations below

The Susceptible-Infected-Recovered (SIR model) describes the dynamics of a population divided into three compartments: **S** (susceptible), **I** (infected), and **R** (recovered). The system of differential equations governing this model is:

$$\frac{dS(t)}{dt} = -\frac{\beta}{N}S(t)I(t) \quad (1)$$

$$\frac{dI(t)}{dt} = \frac{\beta}{N}S(t)I(t) - \gamma I(t) \quad (2)$$

$$\frac{dR(t)}{dt} = \gamma I(t) \quad (3)$$

Where:

1. $S(t)$ is the number of susceptible individuals at time t,
2. $I(t)$ is the number of infected individuals at time t,
3. $R(t)$ is the number of recovered individuals at time t,
4. $N = S(t) + I(t) + R(t)$ is the total population (assumed constant),
5. β is the transmission rate (how often a susceptible individual gets infected),
6. γ is the recovery rate (how often an infected individual recovers).

Code Workflow for part 1: Initialization: Defines the total population (N=1000). Sets initial conditions ($S0$=990,$I0$=10,$R0$=0). Defines the parameters (β,γ) for the three diseases.

Numerical Solver: Implements the 4th-order Runge-Kutta method to solve the SIR equations iteratively over 100 days.

Visualization: Generates a plot for each disease showing the time evolution of $S(t)$, $I(t)$, and $R(t)$.

About

Matlab code repository for group project

Readme
Activity
0 stars
1 watching
0 forks
Report repository

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 4

Njampala14
Nightstrikermax
leowen3
ksbrook1

Languages

MATLAB 100.0%

(b) A screenshot showing the content of the README file

MAE 384 Group Project

This github repository is for Matlab code which models the spraed of disease using the SIR model. This stands for Susceptible-Infected-Recovered which is used for many modern day epidemics. The goal of this code is to
automously project the amount of people affected using transmission rates, recovery rates, total population, number of infected people,and number of people who are susceptible over time

Part 1

This MATLAB code simulates the spread of infectious diseases using the SIR (Susceptible-Infected-Recovered) model. The code generates plots the time evolution of the susceptible, infected, and recovered populations for each disease showing how the populations evolve over a period of 100 days. This code models different diseases by varying the parameters β: Transmission rate (rate at which individuals get infected) and γ: Recovery rate (rate at which infected individuals recover). It also implements Runge-Kutta 4th-order method to solve the SIR equations numerically.

Mathematical model that is described by the equations below

The Susceptible-Infected-Recovered (SIR model) describes the dynamics of a population divided into three compartments: **S** (susceptible), **I** (infected), and **R** (recovered). The system of differential equations governing this model is:

$$\frac{dS(t)}{dt} = -\frac{\beta}{N}S(t)I(t) \quad (1)$$

$$\frac{dI(t)}{dt} = \frac{\beta}{N}S(t)I(t) - \gamma I(t) \quad (2)$$

$$\frac{dR(t)}{dt} = \gamma I(t) \quad (3)$$

Where:

1. $S(t)$ is the number of susceptible individuals at time t,
2. $I(t)$ is the number of infected individuals at time t,
3. $R(t)$ is the number of recovered individuals at time t,
4. $N = S(t) + I(t) + R(t)$ is the total population (assumed constant),
5. β is the transmission rate (how often a susceptible individual gets infected),
6. γ is the recovery rate (how often an infected individual recovers).

Code Workflow for part 1: Initialization: Defines the total population (N=1000). Sets initial conditions ($S0$=990,$I0$=10,$R0$=0). Defines the parameters (β,γ) for the three diseases.

Numerical Solver: Implements the 4th-order Runge-Kutta method to solve the SIR equations iteratively over 100 days.

Visualization: Generates a plot for each disease showing the time evolution of $S(t)$,$I(t)$, and$R(t)$.

Part 2

This section of the code used quadratic interpolation to provide smaller errors compared to linear interpolation. Quadratic interpolation uses higher orders of polynomials to give a closer approximation. Linear interpolation means the function will be linear between two coarse time steps and it performs better if the function is almost linear as well

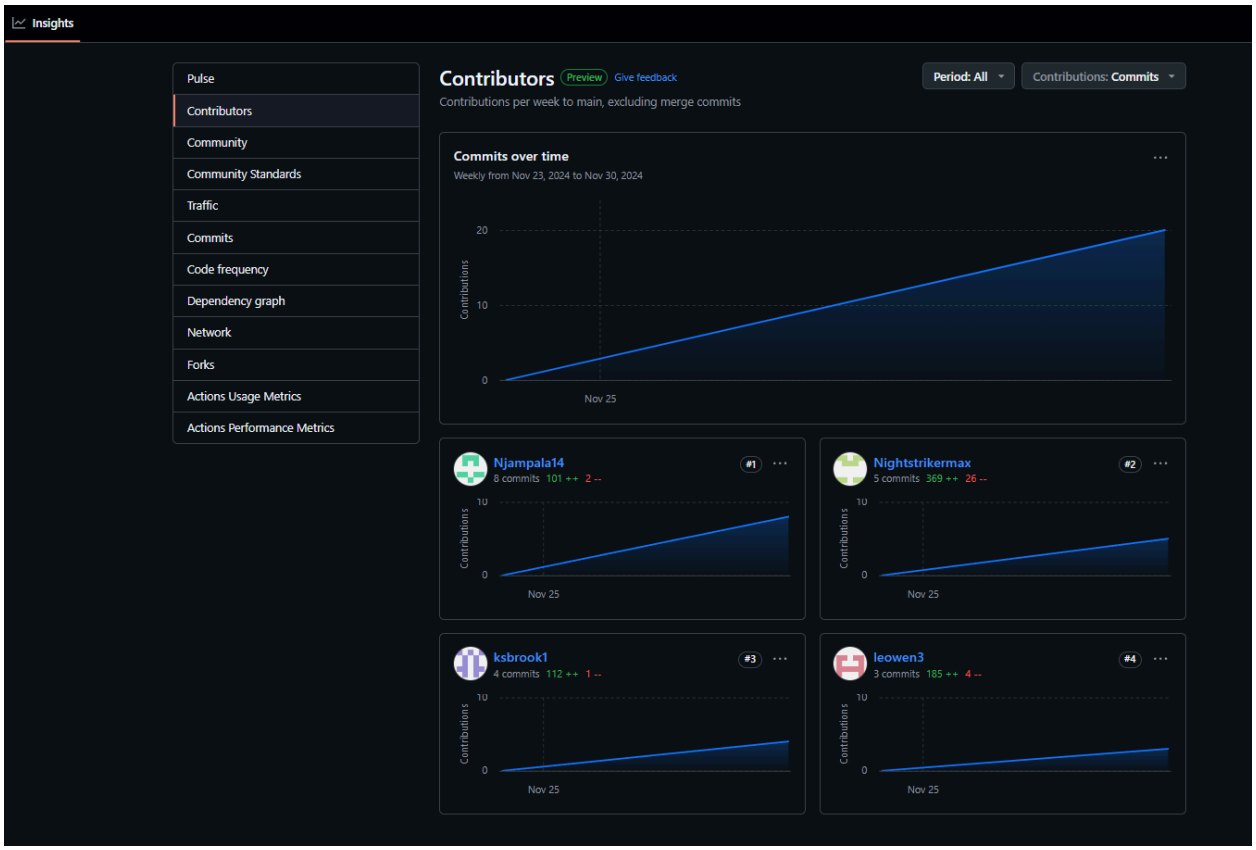
Part 3

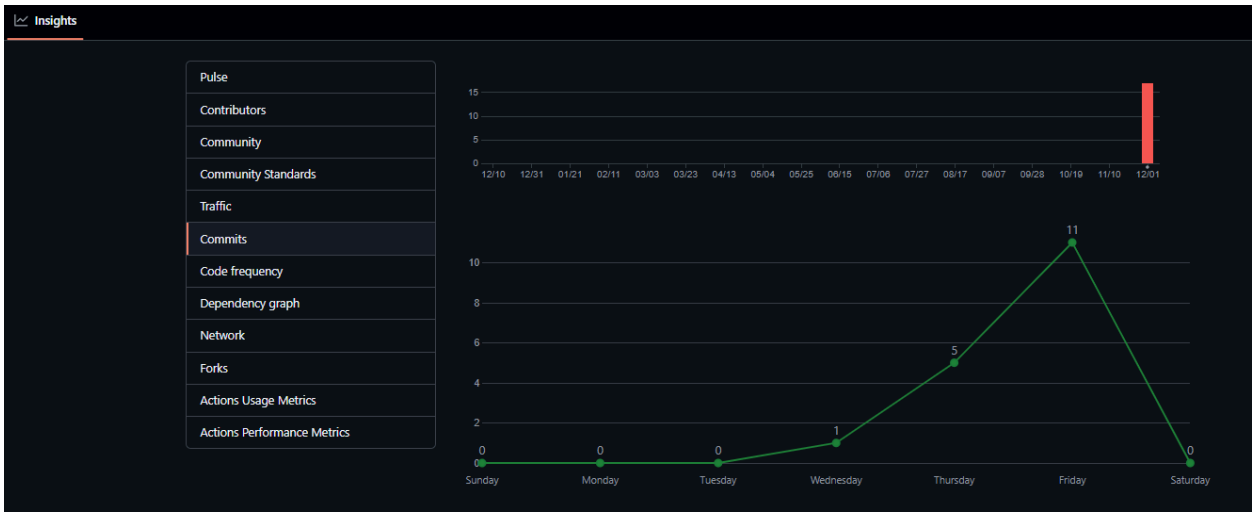
This part of the code was used to approximate the amount of initially infected individuals using the SIR model with a constant amount of susceptible individuals and the amount of infected at a certain time. This part of the code was also used to help find the transmission rate of a disease based off of the same numbers used to find the individuals who were initially infected

Part 4

Part 4 was done in mostly the same way as part 1 but instead of the transmition rate being constant a function handle was used to find and used the transmition value for each time the loop was ran. Part 4 also included MATLABS fft function to plot the spectrum. The fft (Fast Fourier Transofrm) function converts discrete signals from the time domain to the frequency domain and provides information about the frequency content, phase, and other properties of the signal.

(c) A screenshot of “Insights/Contributors”, showing the contribution activity of all members of the group. Additionally, for each contributor, click on “commits” and include screenshots of detailed history of contributions for each member of the group.





Commits

main ▾ | All users ▾ | All time ▾

Commits on Dec 6, 2024

pdf of matlab code and its outputs Nightstrikermax authored 2 minutes ago	Verified	c744c20
Update README.md Njampala14 authored 4 minutes ago	Verified	55dc5d3
Update README.md ksbrook1 authored 5 minutes ago	Verified	0e2b9bc
Update README.md Nightstrikermax authored 15 minutes ago	Verified	6b2a505
Update README.md Njampala14 authored 16 minutes ago	Verified	6ff6f04
Update README.md Njampala14 authored 18 minutes ago	Verified	b4861bd
Update README.md Njampala14 authored 19 minutes ago	Verified	b8c3061
Update README.md leowen3 authored 20 minutes ago	Verified	46a3997
Update README.md Njampala14 authored 21 minutes ago	Verified	75e4847
Update README.md Njampala14 authored 25 minutes ago	Verified	2544016
Update README.md Njampala14 authored 27 minutes ago	Verified	ac20887
Update README.md Nightstrikermax authored 28 minutes ago	Verified	e6ae63f
Add files via upload Njampala14 authored 39 minutes ago	Verified	e34eeec
full code with all parts/problems Nightstrikermax authored 44 minutes ago	Verified	e83ef32

Commits on Dec 5, 2024

Add files via upload ksbrook1 authored yesterday	Verified	8b9ca2d
Delete mae384_project.pdf ksbrook1 authored yesterday	Verified	3278811
Add files via upload ksbrook1 authored yesterday	Verified	ea7a67a
Add files via upload leowen3 authored yesterday	Verified	14b66c3
Merge pull request #1 from Nightstrikermax/part-4 leowen3 authored yesterday	Verified	a799336
Add files via upload leowen3 authored yesterday	Verified	68812f3

Commits on Dec 4, 2024

Initial commit Nightstrikermax committed 2 days ago		57f9451