Table of Contents

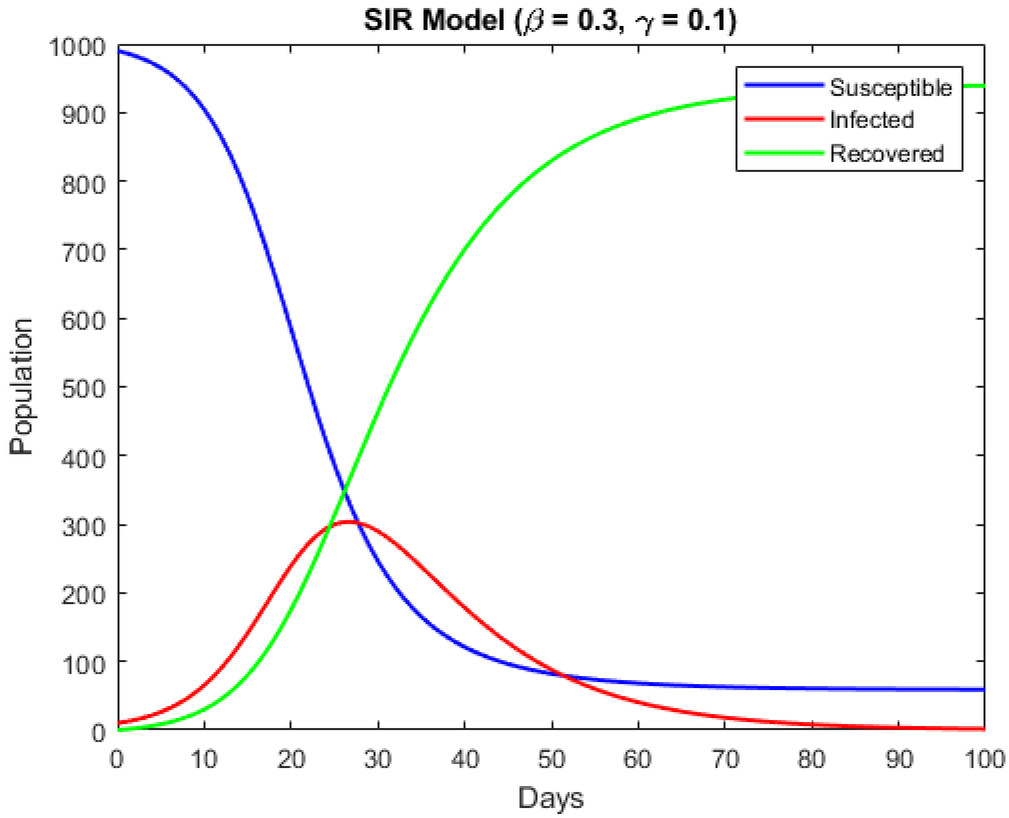
Part 1: Modeling Disease Spread 1
Part 2: Interpolation 4
Part 3: Least Squares 7
Part 4: Fourier Analysis 9

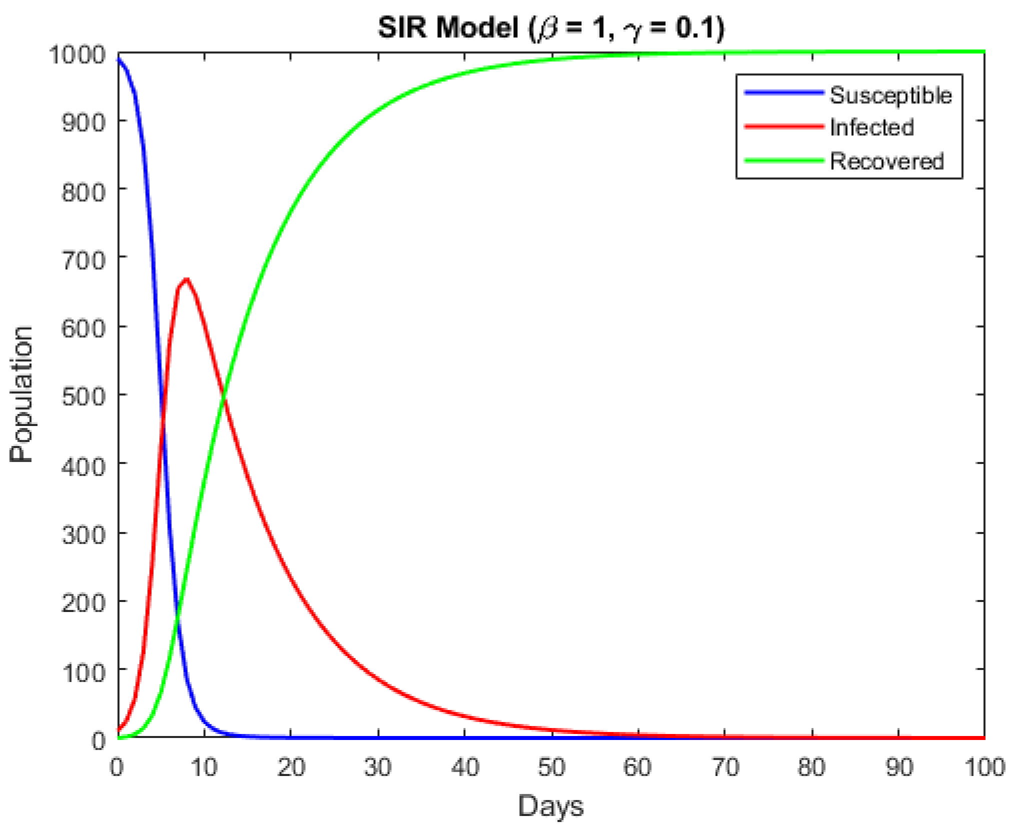
Part 1: Modeling Disease Spread

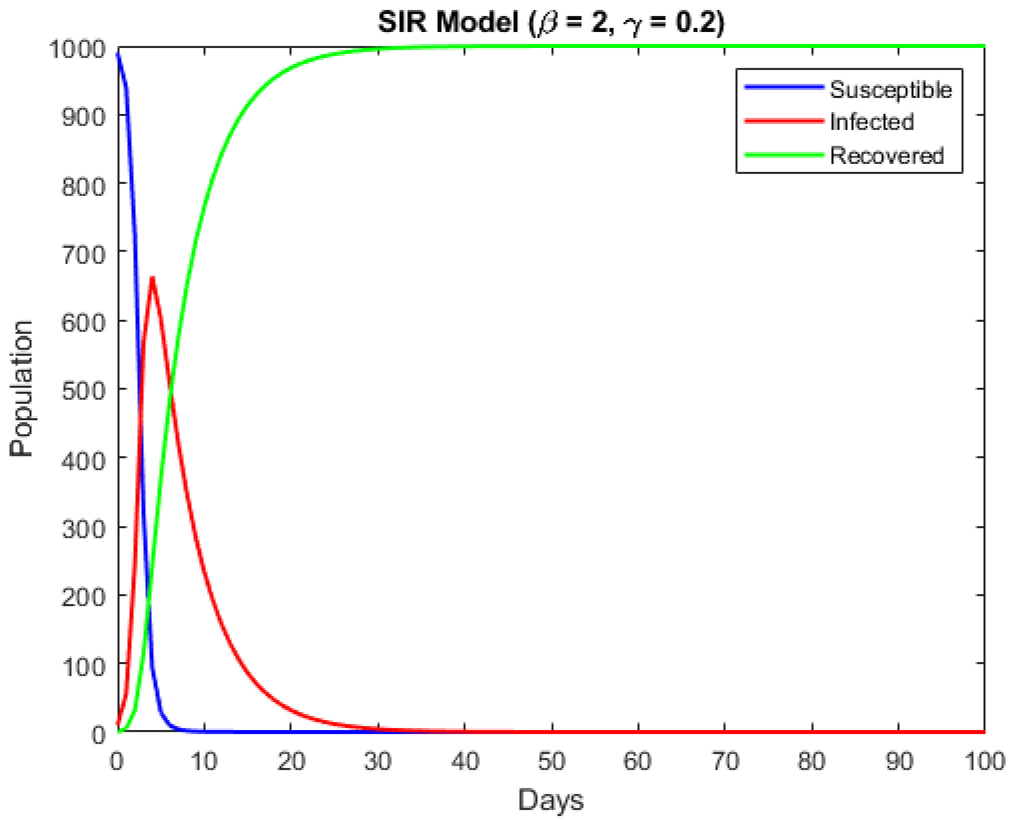
Parameters

```
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)
Population
Days
Susceptible
Infected
Recovered


SIR Model (β = 1, γ = 0.1)
Population
Days
Susceptible
Infected
Recovered



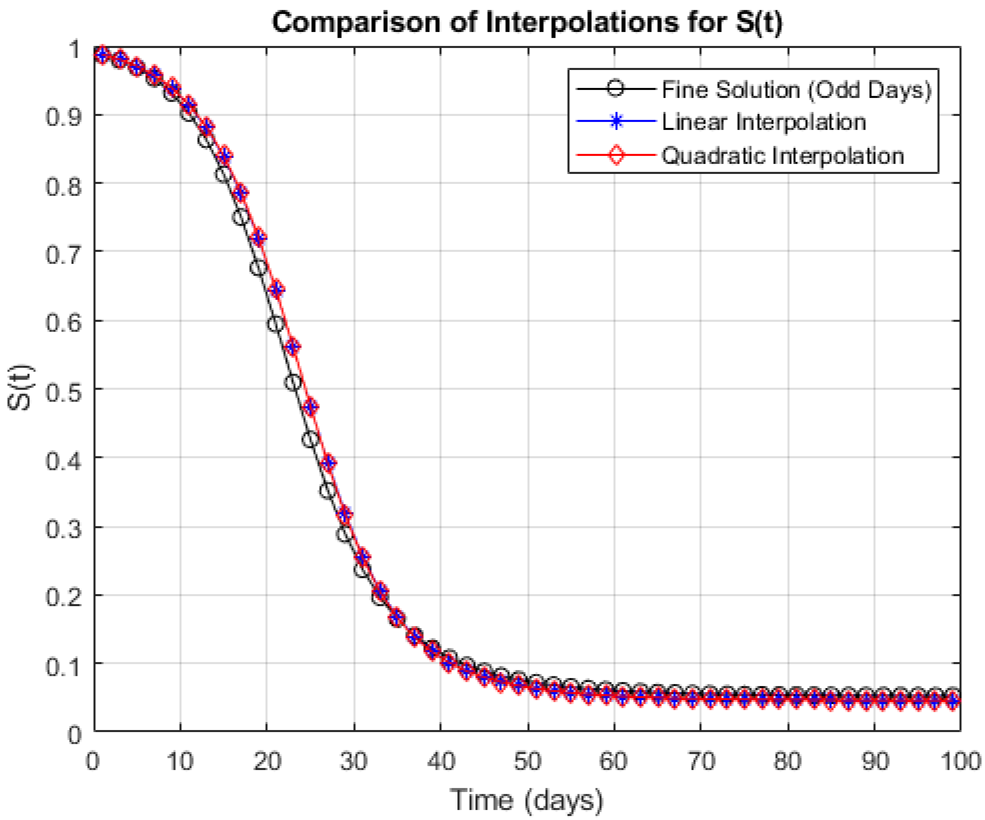
Part 2: Interpolation

```
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

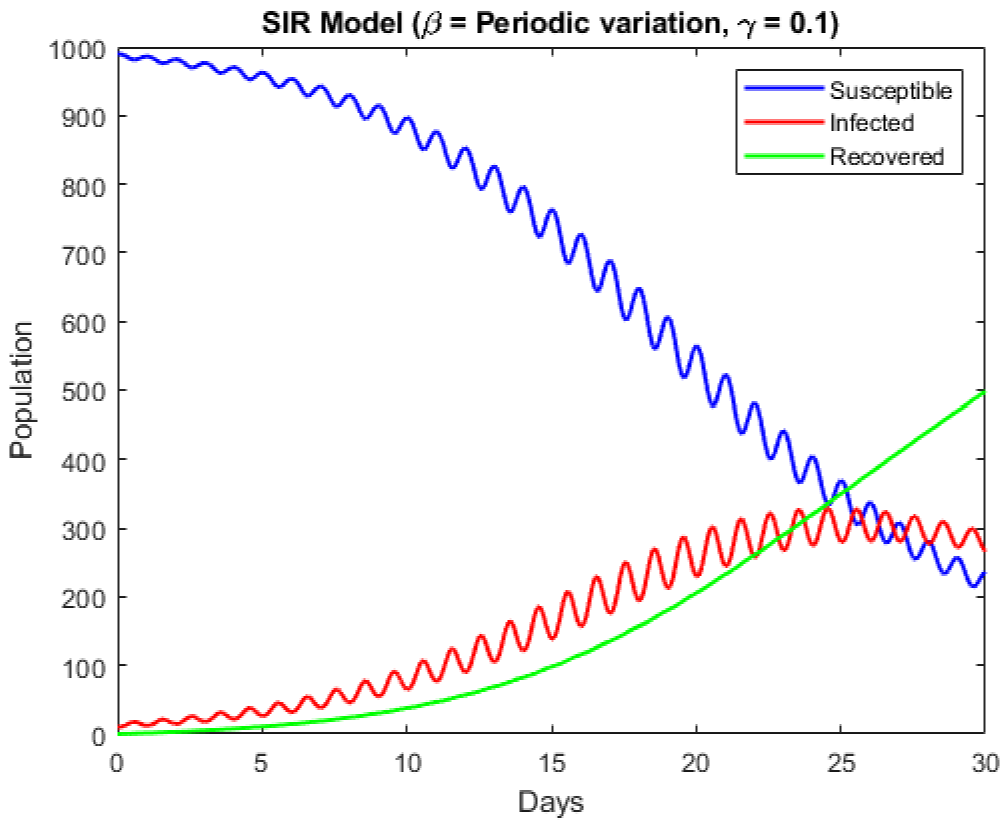
```
Total_pop = 1000; % Total population
Sim_days = 30;   % Duration of the simulation in days
T_step = 0.1;       % Time step in days

% 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;
```

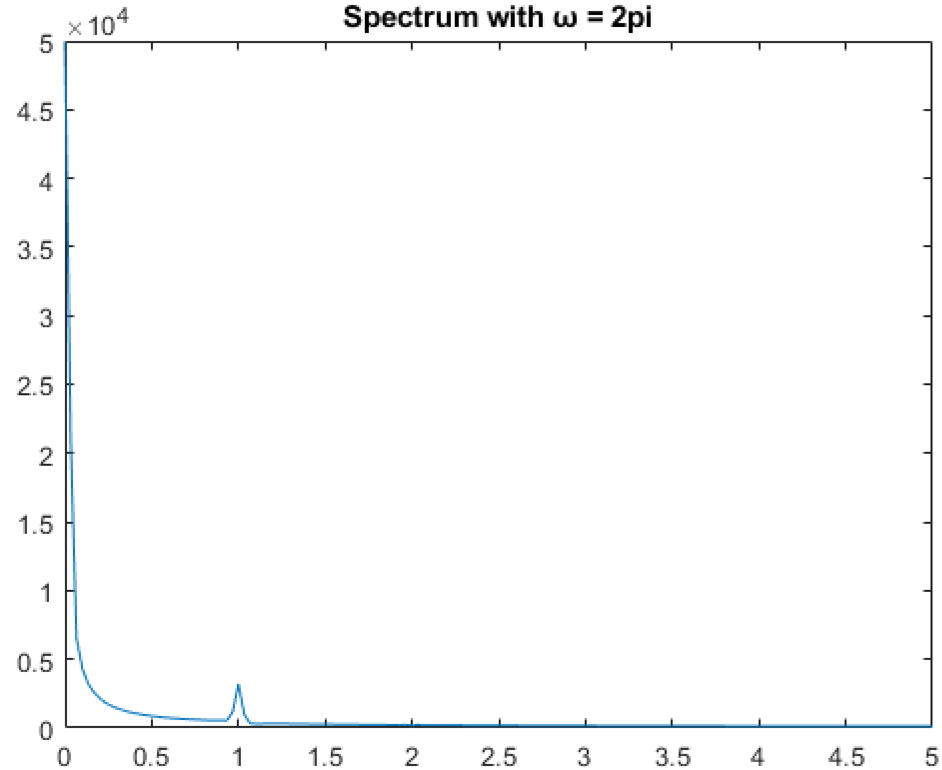
```
% Using 4th-order Runge-Kutta method
for t = 0:0.1:Sim_days-0.1
    % Calculate k1 values
    k1_s = -trans_rate(t) * susceptible(n) * infected(n) / Total_pop;
    k1_i = (trans_rate(t) * susceptible(n) * infected(n) / Total_pop) -
rec_rate * infected(n);
    k1_r = rec_rate * infected(n);

    % 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), ')']);
```

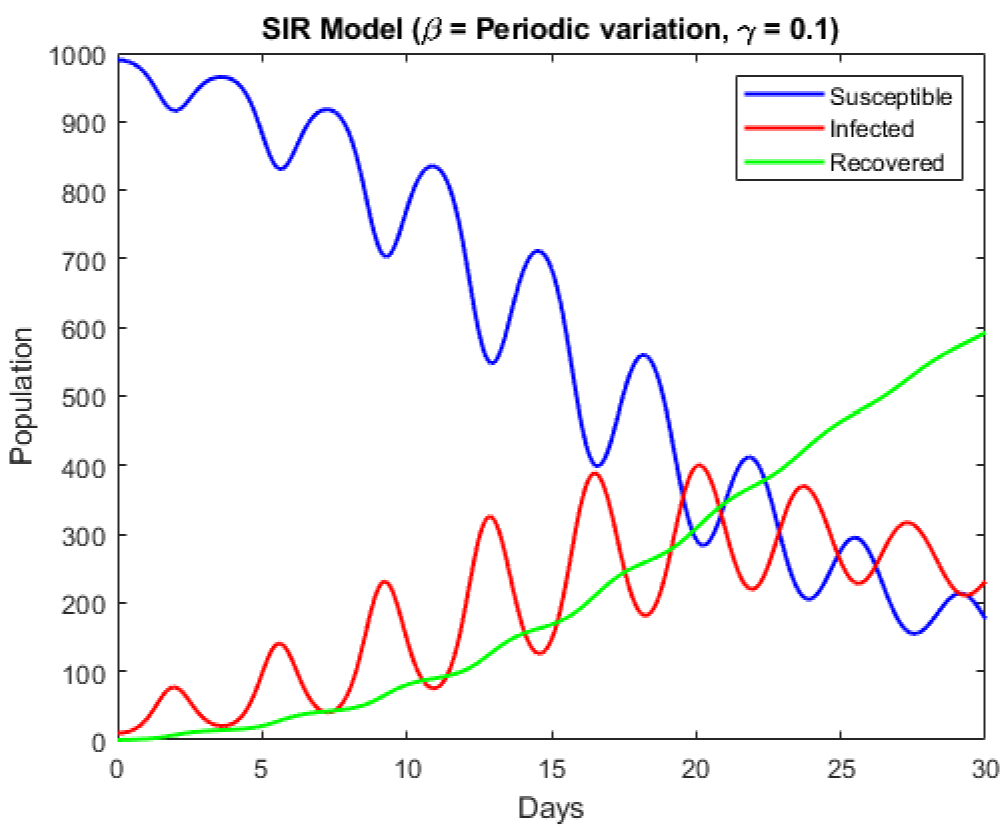
```
% Fourier transform
susceptiblefft = fft(susceptible);
infectedfft = fft(infected);
recoveredfft = fft(recovered);

% 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) *
```

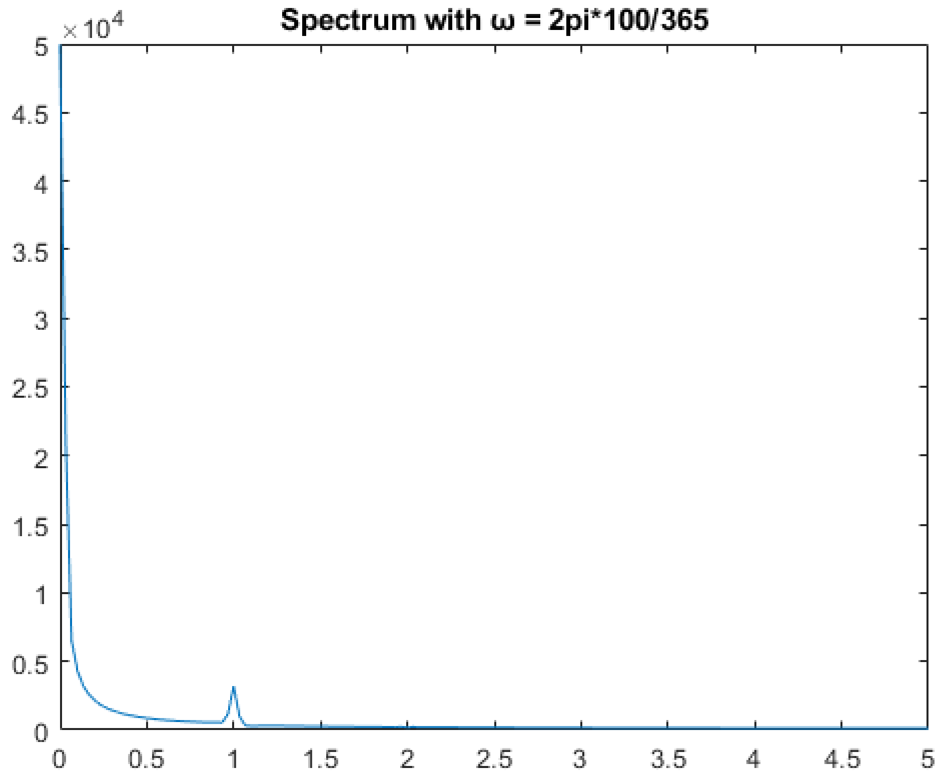
```
(infected1(n) + 0.5 * T_step * k1_i) / Total_pop;
    k2_i = (trans_rate(t) * (susceptible1(n) + 0.5 * T_step * k1_s) *
(infected1(n) + 0.5 * T_step * k1_i) / Total_pop) ...
        - rec_rate * (infected1(n) + 0.5 * T_step * k1_i);
    k2_r = rec_rate * (infected1(n) + 0.5 * T_step * k1_i);

    % 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