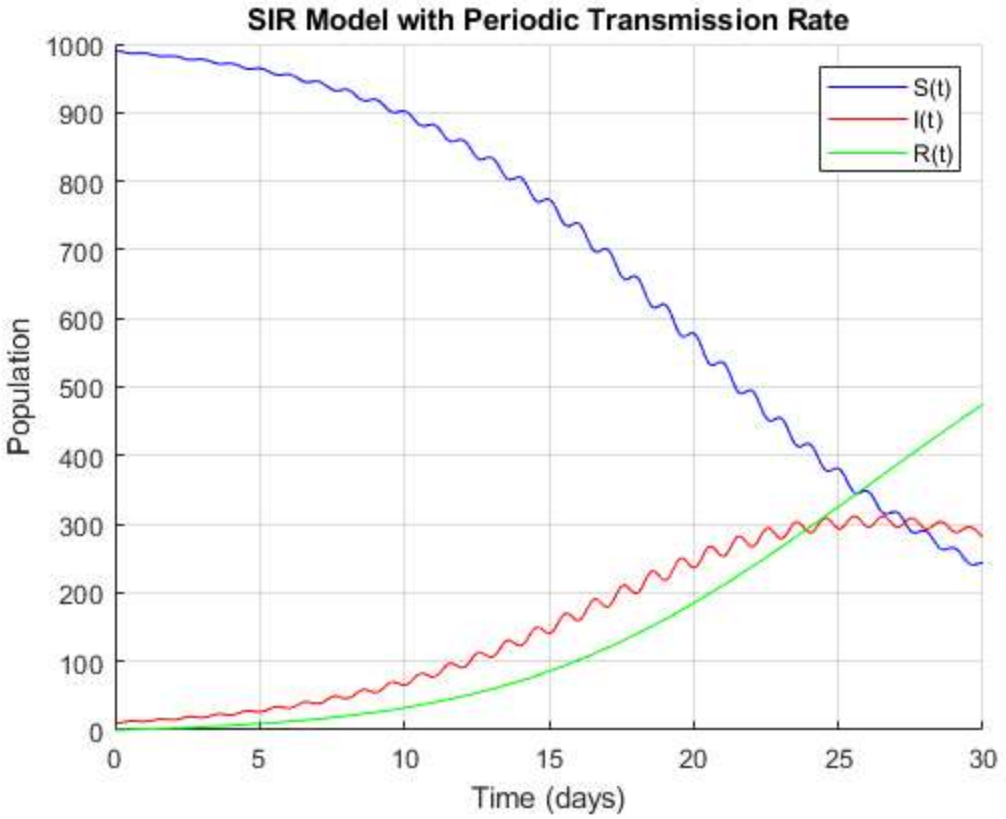
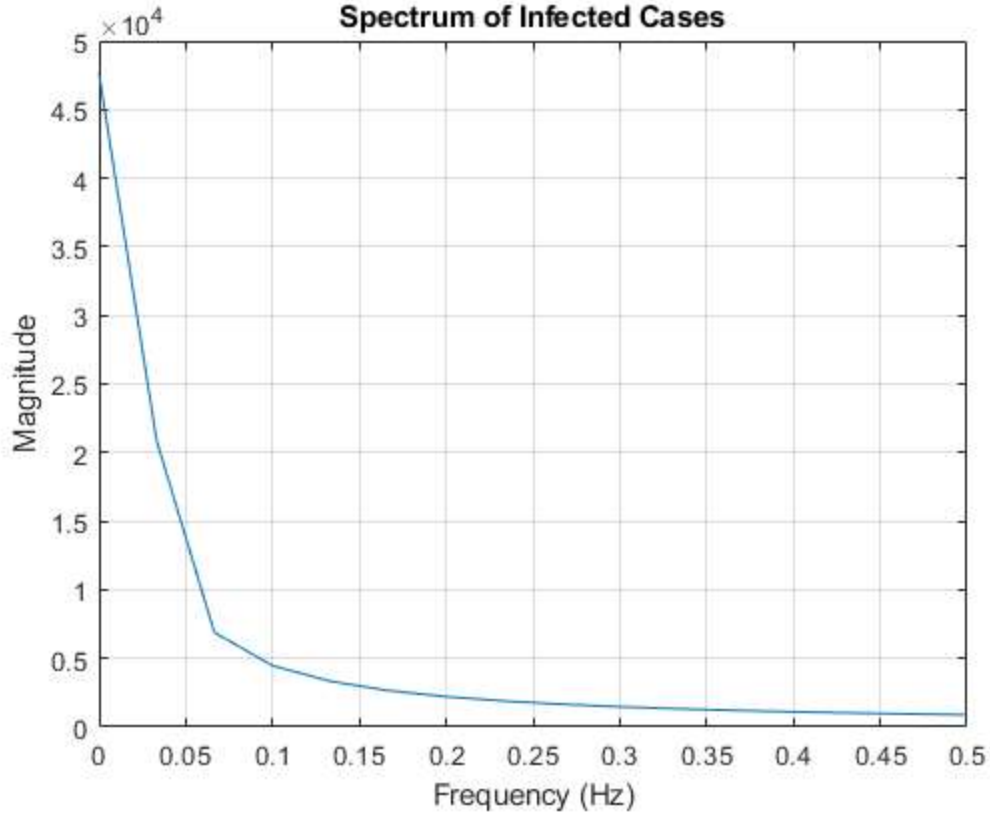
```
% Main script
% Parameters for transmission rate
beta_0 = 0.3;
Lambda = 0.5;
omega = 2*pi; % Periodic transmission rate

% Time settings
h = 0.1; % Time step (days)
t = 0:h:30; % Time vector

% Initial conditions
S0 = 990;
I0 = 10;
R0 = 0;
N = S0 + I0 + R0; % Total population

% Simulate with periodic beta
[S, I, R] = runge_kutta_SIR_periodic(beta_0, Lambda, omega, 0.1, S0, I0, R0, h, t, N);

% Plot results
figure;
hold on;
plot(t, S, 'b-', 'DisplayName', 'S(t)');
plot(t, I, 'r-', 'DisplayName', 'I(t)');
plot(t, R, 'g-', 'DisplayName', 'R(t)');
xlabel('Time (days)');
ylabel('Population');
title('SIR Model with Periodic Transmission Rate');
legend;
grid on;
hold off;

% Perform Fourier Transform
I_fft = fft(I);
f = (0:length(I_fft)-1)*(1/(h*length(I_fft)));

% Plot spectrum for infected cases
figure;
plot(f, abs(I_fft));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Spectrum of Infected Cases');
xlim([0 0.5]);
grid on;

% Function to perform Runge-Kutta 4th Order Method with periodic beta
function [S, I, R] = runge_kutta_SIR_periodic(beta_0, Lambda, omega, gamma, S0, I0, R0, h, t, N)
    % Initialize arrays
    S = zeros(1, length(t));
    I = zeros(1, length(t));
    R = zeros(1, length(t));
    S(1) = S0;
    I(1) = I0;
    R(1) = R0;

    for i = 1:(length(t) - 1)
```

```
        % Calculate beta(t)
        beta = beta_0 + Lambda * sin(omega * t(i));

        % Define ODE functions
        fS = @(S, I) -(beta/N) * S * I;
        fI = @(S, I) (beta/N) * S * I - gamma * I;
        fR = @(I) gamma * I;

        % Runge-Kutta 4th Order Method
        k1_S = fS(S(i), I(i));
        k1_I = fI(S(i), I(i));
        k1_R = fR(I(i));

        k2_S = fS(S(i) + 0.5 * k1_S * h, I(i) + 0.5 * k1_I * h);
        k2_I = fI(S(i) + 0.5 * k1_S * h, I(i) + 0.5 * k1_I * h);
        k2_R = fR(I(i) + 0.5 * k1_I * h);

        k3_S = fS(S(i) + 0.5 * k2_S * h, I(i) + 0.5 * k2_I * h);
        k3_I = fI(S(i) + 0.5 * k2_S * h, I(i) + 0.5 * k2_I * h);
        k3_R = fR(I(i) + 0.5 * k2_I * h);

        k4_S = fS(S(i) + k3_S * h, I(i) + k3_I * h);
        k4_I = fI(S(i) + k3_S * h, I(i) + k3_I * h);
        k4_R = fR(I(i) + k3_I * h);

        S(i+1) = S(i) + (1/6) * (k1_S + 2*k2_S + 2*k3_S + k4_S) * h;
        I(i+1) = I(i) + (1/6) * (k1_I + 2*k2_I + 2*k3_I + k4_I) * h;
        R(i+1) = R(i) + (1/6) * (k1_R + 2*k2_R + 2*k3_R + k4_R) * h;
    end
end
```





Published with MATLAB® R2022b