```
%%%% 100
% Parameters
T = 100; % Total simulation time (days)
h = 1; % Time step (days)
N = 1000; % Total population (constant)
S0 = 990; % Initial susceptible population
I0 = 10; % Initial infected population
R0 = 0; % Initial recovered population
% Parameter sets for different diseases
parameters = [
    0.3, 0.1  % Seasonal Influenza (β, γ)
              ];
for k = 1:size(parameters, 1)
    beta = parameters(k, 1);
    gamma = parameters(k, 2);
    S = zeros(1, T+1);
    I = zeros(1, T+1);
    R = zeros(1, T+1);
    S(1) = S0;
    I(1) = I0;
    R(1) = R0;
    % RK4 method
    for t = 1:T
        % Current values
        S_t = S(t); I_t = I(t); R_t = R(t);
        % Define ODEs
        dS = @(S, I) -(beta / N) * S * I;
        dI = @(S, I) (beta / N) * S * I - gamma * I;
        dR = @(I) gamma * I;
        % RK4 coefficients
        k1_S = h * dS(S_t, I_t);
        k1_I = h * dI(S_t, I_t);
        k1_R = h * dR(I_t);
        k2_S = h * dS(S_t + k1_S/2, I_t + k1_I/2);
        k2_I = h * dI(S_t + k1_S/2, I_t + k1_I/2);
        k2_R = h * dR(I_t + k1_I/2);
        k3_S = h * dS(S_t + k2_S/2, I_t + k2_I/2);
        k3_I = h * dI(S_t + k2_S/2, I_t + k2_I/2);
        k3_R = h * dR(I_t + k2_I/2);
        k4_S = h * dS(S_t + k3_S, I_t + k3_I);
        k4_I = h * dI(S_t + k3_S, I_t + k3_I);
        k4_R = h * dR(I_t + k3_I);
        S(t+1) = S_t + (k1_S + 2*k2_S + 2*k3_S + k4_S) / 6;
        I(t+1) = I_t + (k1_I + 2*k2_I + 2*k3_I + k4_I) / 6;
        R(t+1) = R_t + (k1_R + 2*k2_R + 2*k3_R + k4_R) / 6;
    end
end
h2 = 2;
t2 = 0 : h2 : 100;
%%%% This is the second part that is to interpolate the odd values that
```

```
%%%% is to be found using the Linear Newton form then using the Quadratic
%%%% Lagrange form
for z = 1 : length(t2) - 1
    SNI(z+1) = S(z) + ((S(z+1) - S(z)) / ((z+1) - (z))) * ((z+1) - (z));
    INI(z+1) = I(z) + ((I(z+1) - I(z)) / ((z+1) - (z))) * ((z+1) - (z));
    RNI(z+1) = R(z) + ((R(z+1) - R(z)) / ((z+1) - (z))) * ((z+1) - (z));
    %%%% The Quadratic Lagrange Method Form
    for j = 1 : z
        X_1(j+1) = (((j+1) - (j)) * ((j+1) - (j+3))) / ((((j) - (j+1)) * ((j) +
(j+3))));
        X_2(j+1) = (((j+1) - (j)) * ((j+1) - (j+3))) / (((j+1) - (j)) * ((j+1) -
(j+3)));
        X_3(j+1) = (((j+1) - (j)) * ((j+1) - (j+2))) / (((j+3) - (j)) * ((j+3) -
(j+2)));
        QS(j+1) = (X_1(j) * S(j)) + (X_2(j+1) * S(j+2)) + (X_3(j+1) * S(j+3));
        QI(j+1) = (X_1(j) * I(j)) + (X_2(j+1) * I(j+2)) + (X_3(j+1) * I(j+3));
        QR(j+1) = (X_1(j) * R(j)) + (X_2(j+1) * R(j+2)) + (X_3(j+1) * R(j+3));
    end
end
L2_s = sqrt((S(z+1) - SNI(z+1)/ T));
L2_i = sqrt((I(z+1) - INI(z+1)/ T));
L2_r = sqrt((R(z+1) - RNI(z+1)/ T));
L2_Qs = sqrt((QS(z+1) - SNI(z+1)/ T));
L2_Qi = sqrt((QI(z+1) - INI(z+1)/ T));
L2_Qr = sqrt((QR(z+1) - RNI(z+1)/ T));
%%%% The Errors computed from the above formulas in 2x3 matrix print
fprintf('L2 Error Table:\n');
```

```
L2 Error Table:
```

```
fprintf('               S(t)         I(t)         R(t)\n');
```

```
             S(t)        I(t)        R(t)
```

```
fprintf('Linear    : %.7f    %.7f    %.7f\n', L2_s, L2_i, L2_r);
```

```
Linear    : 9.0179185    9.3397985    28.6608672
```

```
fprintf('Quadratic : %.7f    %.7f    %.7f\n', L2_Qs, L2_Qi, L2_Qr);
```

```
Quadratic : 7.4066979    7.5802321    23.7518326
```

The values that are more accurate are the Quadratic portion due to the difference in the values being greater compared to the linear value. The linear values error are larger than the quadratic meaning that the quadratic is the one to produce less errors.