```
clc;
clear;

N = 1000;
S0 = 990;
I0 = 10;
R0 = 0;

% Code from part 1
Beta = [.3 .3];
gamma = [.1 .1];

h = [1 2];
t0 = 0; tf = 100;
tiledlayout(3,1);
```

```
RValsFinal1 = zeros(1,101);
SValsFinal1 = zeros(1,101);
IValsFinal1 = zeros(1,101);
RValsFinal2 = zeros(1,51);
SValsFinal2 = zeros(1,51);
IValsFinal2 = zeros(1,51);

% Influenza Virus Loop
for j = 1:length(Beta)
    Gamma=gamma(j);
    BN1=Beta(j)/1000;
    if h(j) == 2
    time2 = t0:h(j):tf;
    Rvals2 = zeros(size(time2));
    Svals2 = zeros(size(time2));
    Ivals2 = zeros(size(time2));
    time = time2;
    else
    time1 = t0:h(j):tf;
    Rvals1 = zeros(size(time1));
    Svals1 = zeros(size(time1));
    Ivals1 = zeros(size(time1));
    Nvals1 = zeros(size(time1));
    time = time1;
    end


    Rvals1(1) = R0;
    Svals1(1) = S0;
    Ivals1(1) = I0;
    Rvals2(1) = R0;
    Svals2(1) = S0;
    Ivals2(1) = I0;


    dRdt = @(I) Gamma*I;
    dSdt = @(I,S) -BN1*S*I;
    dIdt = @(I,S) (BN1*S*I)-(Gamma*I);
    % Time loop
        for i = 1:length(time)-1
            if h(j) ==2
            R=Rvals2(i);
            S=Svals2(i);
            I=Ivals2(i);
            else
                R=Rvals1(i);
            S=Svals1(i);
            I=Ivals1(i);
            end


            rk1 = dRdt(I);           % Runge Kutta Recovered K1
            sk1 = dSdt(I,S);            % Runge Kutta Susceptible K1
            ik1 = dIdt(I,S);           % Runge Kutta Infected K1

            ik2 = dIdt((I + ik1 * (h(j)/2)) , S + sk1 * (h(j)/2));           %
Runge Kutta Infected K2
            rk2 = dRdt(I + ik1 * (h(j)/2));           % Runge Kutta Recovered K2
            sk2 = dSdt(I + ik1 * (h(j)/2)  , (S + sk1 * (h(j)/2)));           %
Runge Kutta Susceptible K2

            sk3 = dSdt(I + ik2 * (h(j)/2) ,(S + sk2 * (h(j)/2)));           %
Runge Kutta Susceptible K3
            ik3 = dIdt((I + ik2 * (h(j)/2)) ,S + sk2 * (h(j)/2));           % Runge
Kutta Infected K3
            rk3 = dRdt(I + ik2 * (h(j)/2));           % Runge Kutta Recovered K3

            ik4 = dIdt((I + ik3 * h(j)),S + sk3 * h(j));           % Runge Kutta
Infected K4
            sk4 = dSdt(I + ik3 * h(j) , (S + sk3 * h(j)));           % Runge Kutta
Susceptible K4
            rk4 = dRdt(I + ik3 * h(j));
            if h(j) == 2
            Rvals2(i+1) = R + (1/6) * (rk1 + 2*(rk2) + 2*(rk3) + rk4) * h(j);
            Svals2(i+1) = S + (1/6) * (sk1 + 2*(sk2) + 2*(sk3) + sk4) * h(j);
            Ivals2(i+1) = I + (1/6) * (ik1 + 2*(ik2) + 2*(ik3) + ik4) * h(j);
            RValsFinal2(i+1) = Rvals2(i+1);
            SValsFinal2(i+1) = Svals2(i+1);
            IValsFinal2(i+1) = Ivals2(i+1);
            else
            Rvals1(i+1) = R + (1/6) * (rk1 + 2*(rk2) + 2*(rk3) + rk4) * h(j);
            Svals1(i+1) = S + (1/6) * (sk1 + 2*(sk2) + 2*(sk3) + sk4) * h(j);
            Ivals1(i+1) = I + (1/6) * (ik1 + 2*(ik2) + 2*(ik3) + ik4) * h(j);
            RValsFinal1(i+1) = Rvals1(i+1);
            SValsFinal1(i+1) = Svals1(i+1);
            IValsFinal1(i+1) = Ivals1(i+1);
            end
        end
end
RValsFinal1(1) = R0;
SValsFinal1(1) = S0;
IValsFinal1(1) = I0;
RValsFinal2(1) = R0;
SValsFinal2(1) = S0;
IValsFinal2(1) = I0;


% Newton's Linear Interpolation
odd_days = 1:2:tf-1;
Lininterp_S = zeros(size(odd_days));
Lininterp_I = zeros(size(odd_days));
Lininterp_R = zeros(size(odd_days));

for k = 1:length(odd_days)
    t = odd_days(k);

    idx = find(time2 <= t, 1, 'last');
    x0 = time2(idx);
    x1 = time2(idx + 1);

    y0_S = SValsFinal2(idx);
    y1_S = SValsFinal2(idx + 1);

    y0_I = IValsFinal2(idx);
    y1_I = IValsFinal2(idx + 1);

    y0_R = RValsFinal2(idx);
    y1_R = RValsFinal2(idx + 1);
    Lininterp_S(k) = y0_S + (y1_S - y0_S) * (t - x0) / (x1 - x0);
    Lininterp_I(k) = y0_I + (y1_I - y0_I) * (t - x0) / (x1 - x0);
    Lininterp_R(k) = y0_R + (y1_R - y0_R) * (t - x0) / (x1 - x0);
end

% Newton's Quadratic interpolation
for k = 1:length(odd_days)
    t = odd_days(k);

    idx = find(time2 <= t, 1, 'last');
    if idx + 1 <= length(time2) && idx - 1 >= 1
        x0 = time2(idx - 1);
        x1 = time2(idx);
        x2 = time2(idx + 1);
        y0_S = SValsFinal2(idx - 1);
        y1_S = SValsFinal2(idx);
        y2_S = SValsFinal2(idx + 1);
        y0_I = IValsFinal2(idx - 1);
        y1_I = IValsFinal2(idx);
        y2_I = IValsFinal2(idx + 1);
        y0_R = RValsFinal2(idx - 1);
        y1_R = RValsFinal2(idx);
        y2_R = RValsFinal2(idx + 1);
    else

        x0 = time2(idx);
        x1 = time2(idx + 1);
        y0_S = SValsFinal2(idx);
        y1_S = SValsFinal2(idx + 1);
        y0_I = IValsFinal2(idx);
        y1_I = IValsFinal2(idx + 1);
        y0_R = RValsFinal2(idx);
        y1_R = RValsFinal2(idx + 1);

        Quadinterp_S(k) = y0_S + (y1_S - y0_S) * (t - x0) / (x1 - x0);
        Quadinterp_I(k) = y0_I + (y1_I - y0_I) * (t - x0) / (x1 - x0);
        Quadinterp_R(k) = y0_R + (y1_R - y0_R) * (t - x0) / (x1 - x0);
        continue;
    end


    Quadinterp_S(k) = quadratic_interpolation(x0, x1, x2, y0_S, y1_S, y2_S, t);
    Quadinterp_I(k) = quadratic_interpolation(x0, x1, x2, y0_I, y1_I, y2_I, t);
    Quadinterp_R(k) = quadratic_interpolation(x0, x1, x2, y0_R, y1_R, y2_R, t);
end

% Combination of Interpolated Data
S_combinedLin = zeros( 1, tf + 1);
I_combinedLin = zeros(1, tf + 1);
```

```
R_combinedLin = zeros( 1, tf + 1);
S_combinedLin(1:2:end) = SValsFinal2;
S_combinedLin(2:2:end) = Lininterp_S;
I_combinedLin(1:2:end) = IValsFinal2;
I_combinedLin(2:2:end) = Lininterp_I;
R_combinedLin(1:2:end) = RValsFinal2;
R_combinedLin(2:2:end) = Lininterp_R;

S_combinedQuad = zeros( 1, tf + 1);
I_combinedQuad = zeros(1,tf +  1);
R_combinedQuad = zeros(1,tf +  1);
S_combinedQuad(1:2:end) = SValsFinal2;
S_combinedQuad(2:2:end) = Quadinterp_S;
I_combinedQuad(1:2:end) = IValsFinal2;
I_combinedQuad(2:2:end) = Quadinterp_I;
R_combinedQuad(1:2:end) = RValsFinal2;
R_combinedQuad(2:2:end) = Quadinterp_R;

% L2 Error Calculations
E_S_Lin = compute_error(SValsFinal1, S_combinedLin);
E_I_Lin = compute_error(IValsFinal1, I_combinedLin);
E_R_Lin = compute_error(RValsFinal1, R_combinedLin);
E_S_Quad = compute_error(SValsFinal1, S_combinedQuad);
E_I_Quad = compute_error(IValsFinal1, I_combinedQuad);
E_R_Quad = compute_error(RValsFinal1, R_combinedQuad);

Susceptible = [E_S_Lin ;E_S_Quad];
Infected = [E_I_Lin ;E_I_Quad];
Recovered = [E_R_Lin;E_R_Quad];
Type = ["Linear Interpolation";"Quadratic Interpolation"];
data = table(Susceptible,Infected,Recovered,'Rownames',Type)
```

data = 2×3 table

	Susceptible	Infected	Recovered
1 Linear Interpolation	0.6517	0.5043	0.4133
2 Quadratic Interpolation	0.1273	0.1043	0.0661

6.

From the table above, we can see that quadratic interpolation provides smaller errors than linear interpolation.

```
function E_L2 = compute_error(V_h1, V_combined)
    N_int = (length(V_combined) - 1)/2;
    error_sum = sum((V_combined - V_h1).^2);
    E_L2 = sqrt(error_sum / N_int);

end
function y = quadratic_interpolation(x0, x1, x2, y0, y1, y2, x)
```

```
    b0 = y0;
    b1 = (y1 - y0) / (x1 - x0);
    b2 = ((y2 - y1) / (x2 - x1) - b1) / (x2 - x0);

    y = b0 + b1 * (x - x0) + b2 * (x - x0) * (x - x1);
end
```