# EE498
# Control System Design and Simulation

Part 2) Runge Kutta 4 Stages Method

a) The Simulink model is created using the differential equations as given below with the initial conditions as given in the journal paper, the model is first simulated using
   - Maximum step size: auto
   - Relative tolerance: 1e-5
   - Solver: ode45 (variable step size)

   and the output is obtained which correctly follows the solution described in the journal with the output values of S (0) = 0.999, E (0) = 0.001, I (0) = R (0).

The basic model can be summarized as follows

$$\frac{dS}{dt} = \mu \cdot N - \beta \cdot \frac{I \cdot S}{N} + \omega \cdot R - \mu \cdot S$$
$$\frac{dE}{dt} = \beta \cdot \frac{I \cdot S}{N} - \sigma \cdot E - \mu \cdot E$$
$$\frac{dI}{dt} = \sigma \cdot E - \gamma \cdot I - (\mu + \alpha) \cdot I$$
$$\frac{dR}{dt} = \gamma \cdot I - \omega \cdot R - \mu \cdot R.$$

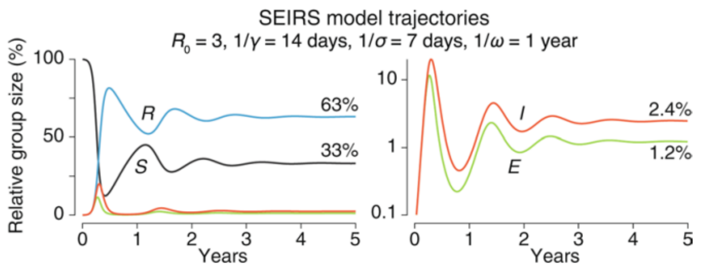

Fig 1.0 ODE's and SEIRS Output Model as given in the Journal

The Simulink Model is given as follows with its output in two separate scopes to ensure that the Infected and the Exposed group steady states could be zoomed enough to have proper visibility.

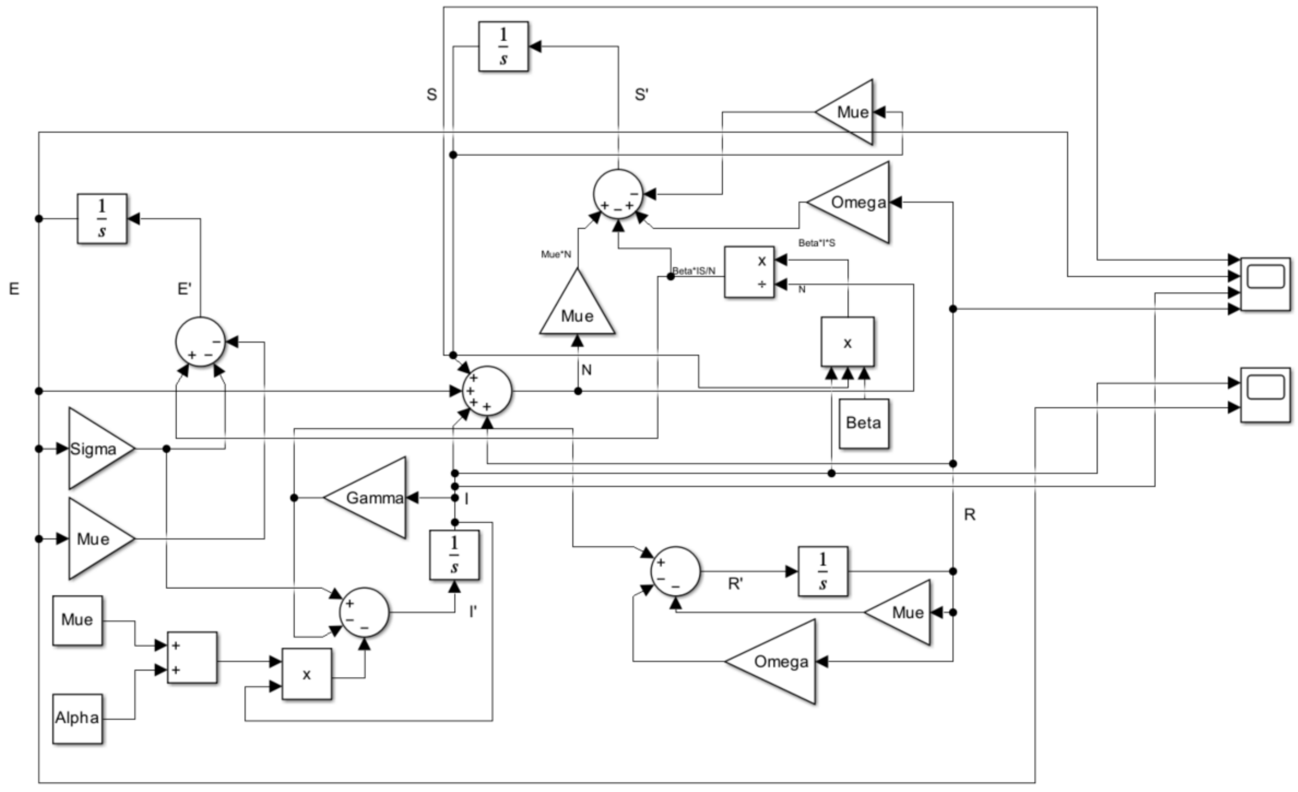

Fig 1.1 Simulink Model for the ODE's of SEIRS Model

By using the same initial condition and parameters as given below we were able to reconstruct the solution and output it on the scope.

```
% Initialisation of Variables %
R0 = 3;
Gamma = 1/14;
Sigma = 1/7;
Omega = 1/365;
Mue = 1/(76*365);
Alpha = 0;
Beta = R0 * Gamma;
```

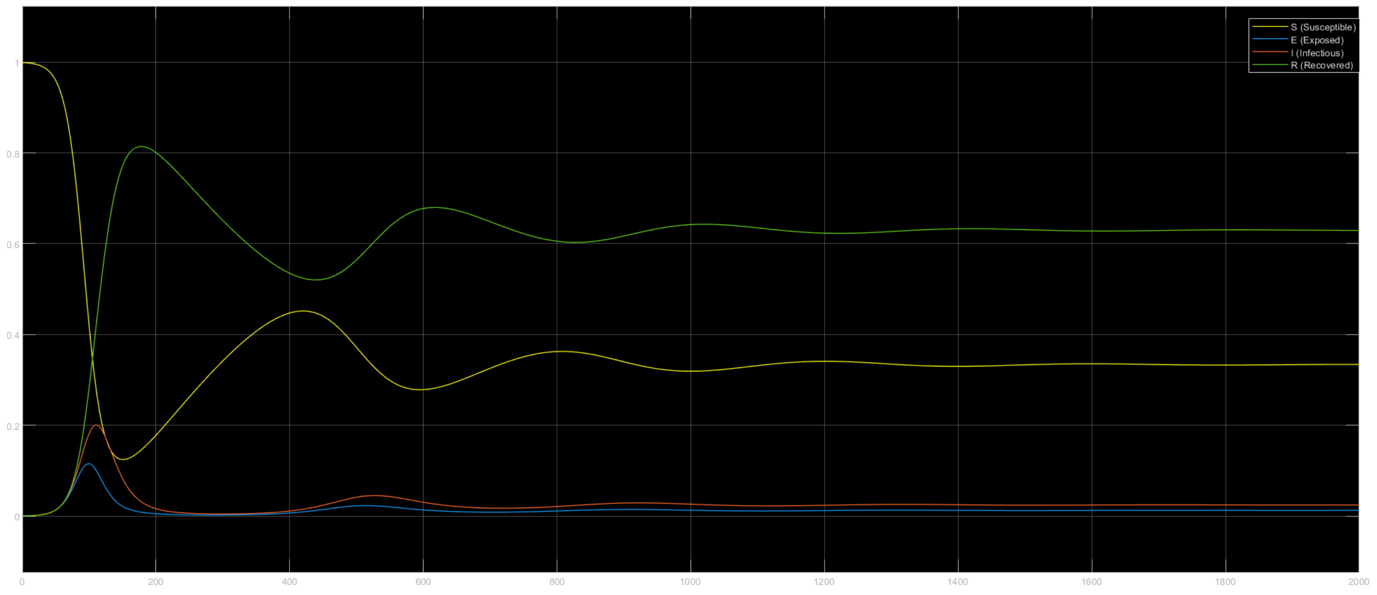

Fig 1.2 Simulink SEIRS's Model Solution

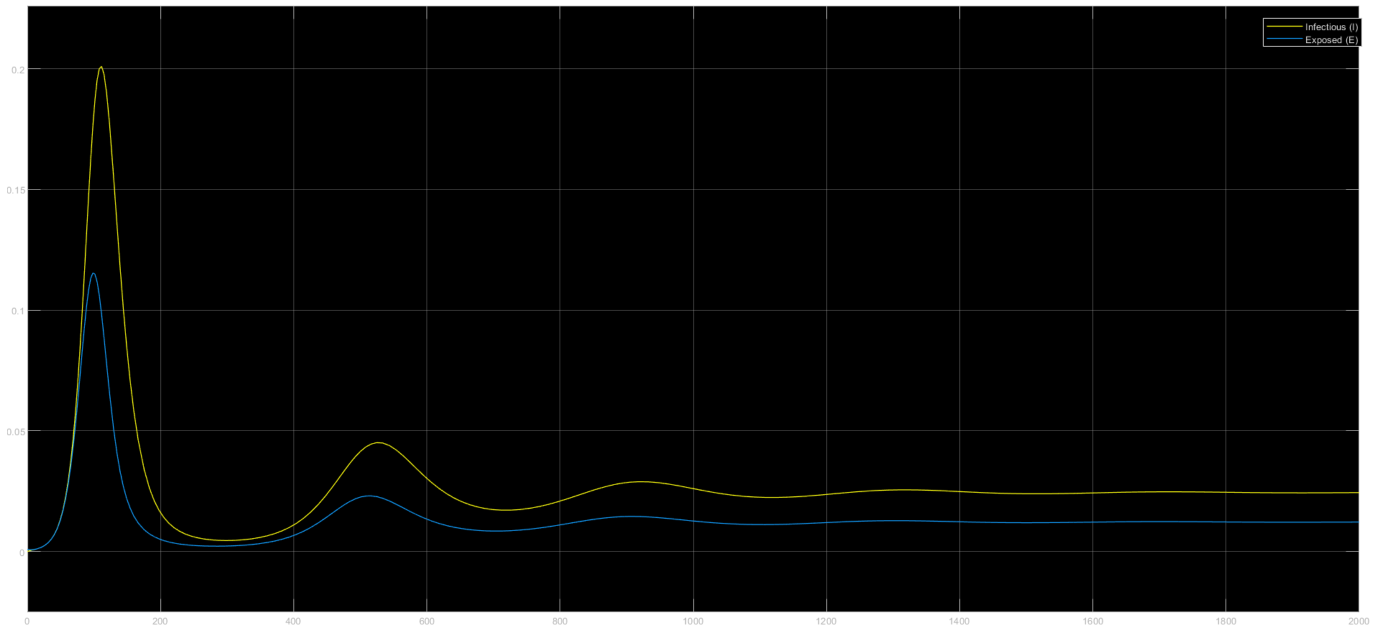

Fig 1.3 Infected and Exposed Signals Zoomed for better Visibility

Now by changing the solver settings we have generated different results which are mentioned below:

1) Changing the relative tolerance
   - Maximum step size: auto
   - Relative tolerance: 1e-2
   - Solver: ode45 (variable step size)

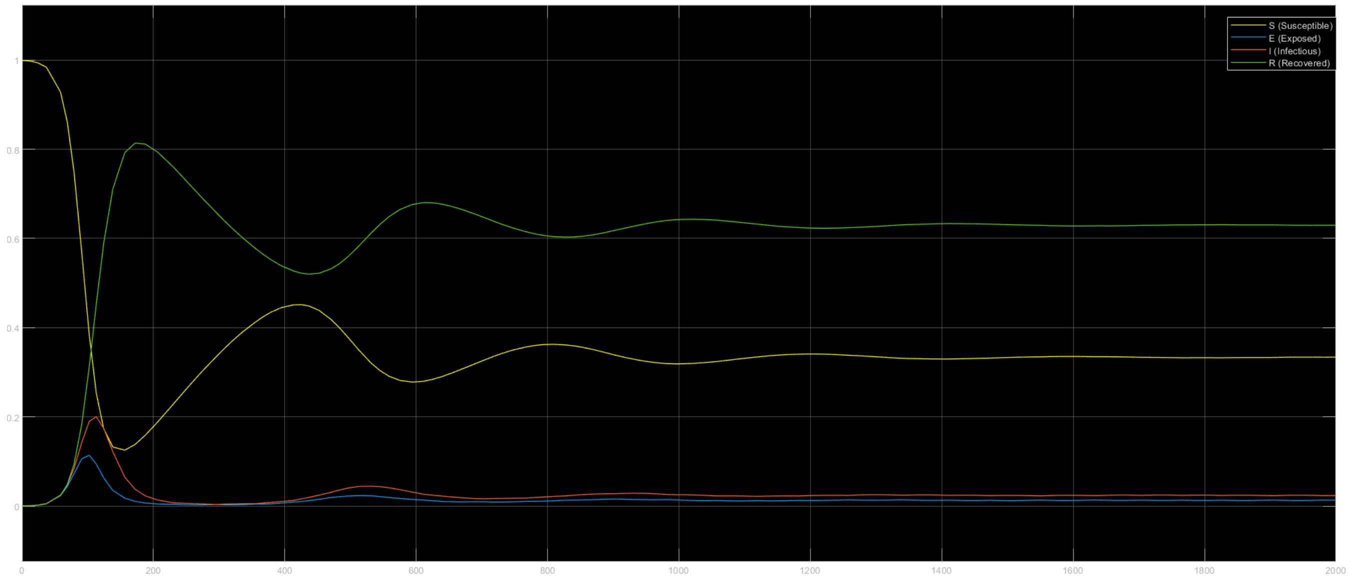

Fig 1.4 Simulink SEIRS's Model Solution

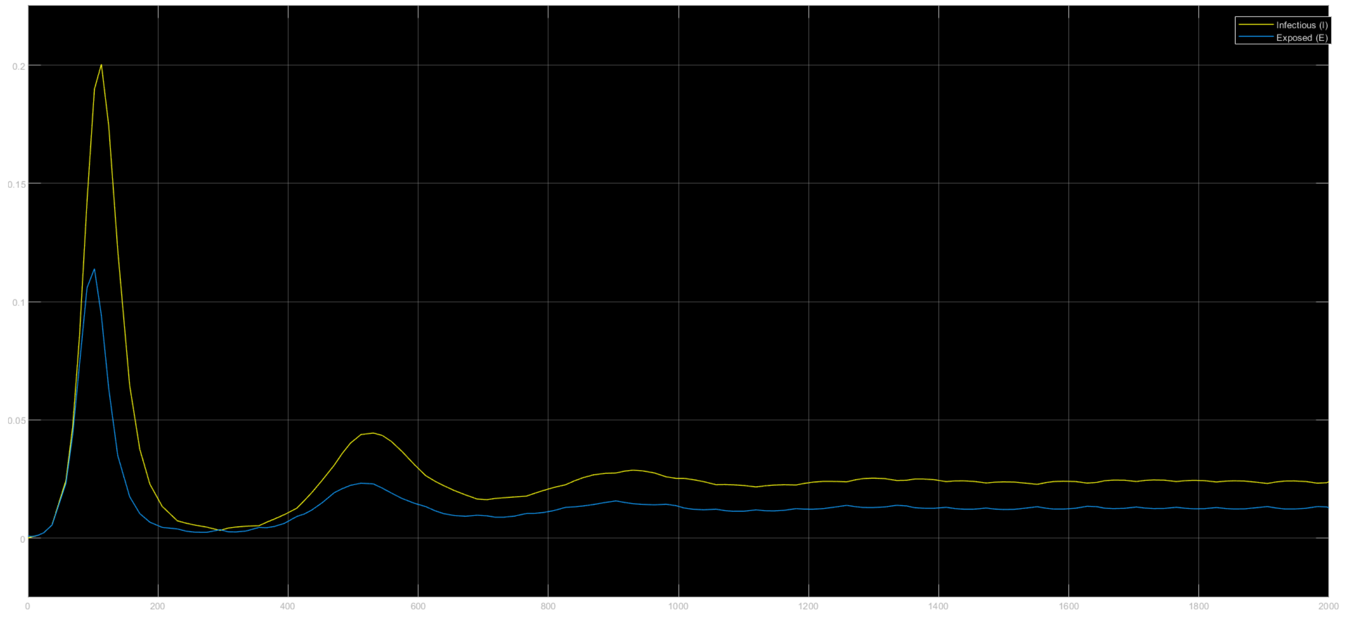

Fig 1.5 Infected and Exposed Signals Zoomed for better Visibility

2) Changing the Maximum step size
    - Maximum step size: 10
    - Relative tolerance: 1e-5
    - Solver: ode45 (variable step size)

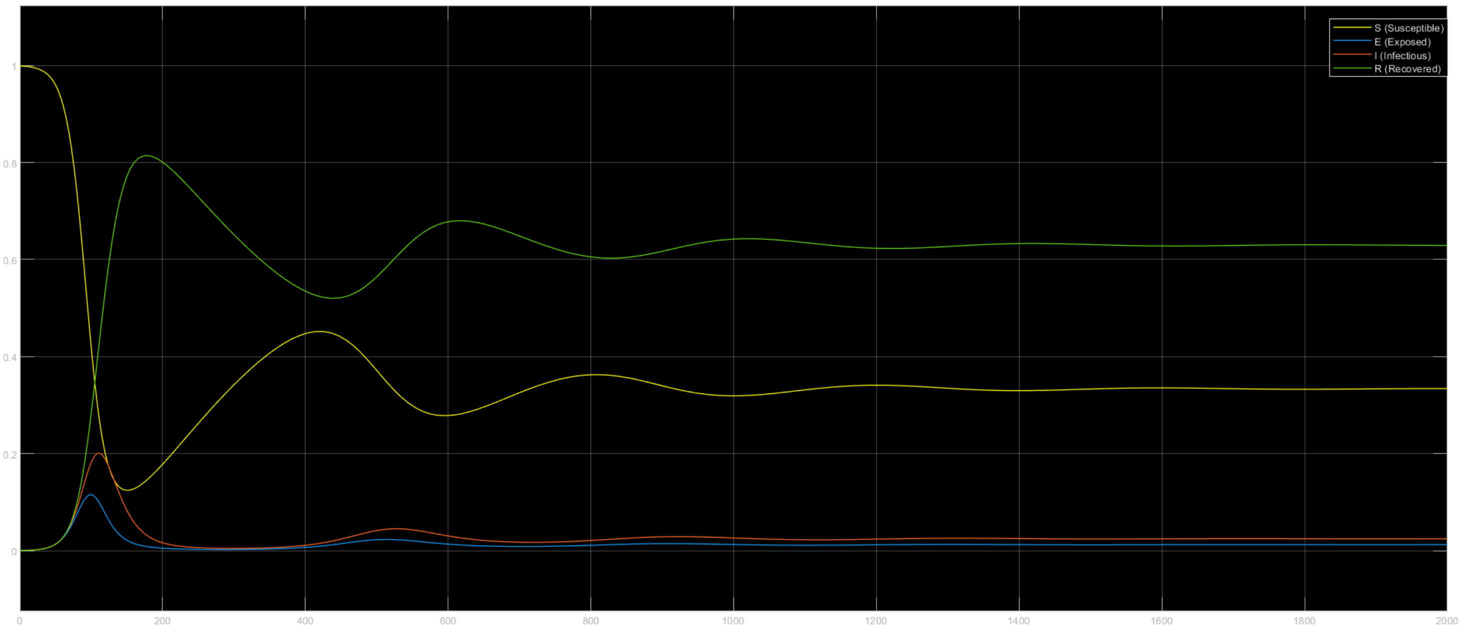

Fig 1.6 Simulink SEIRS's Model Solution

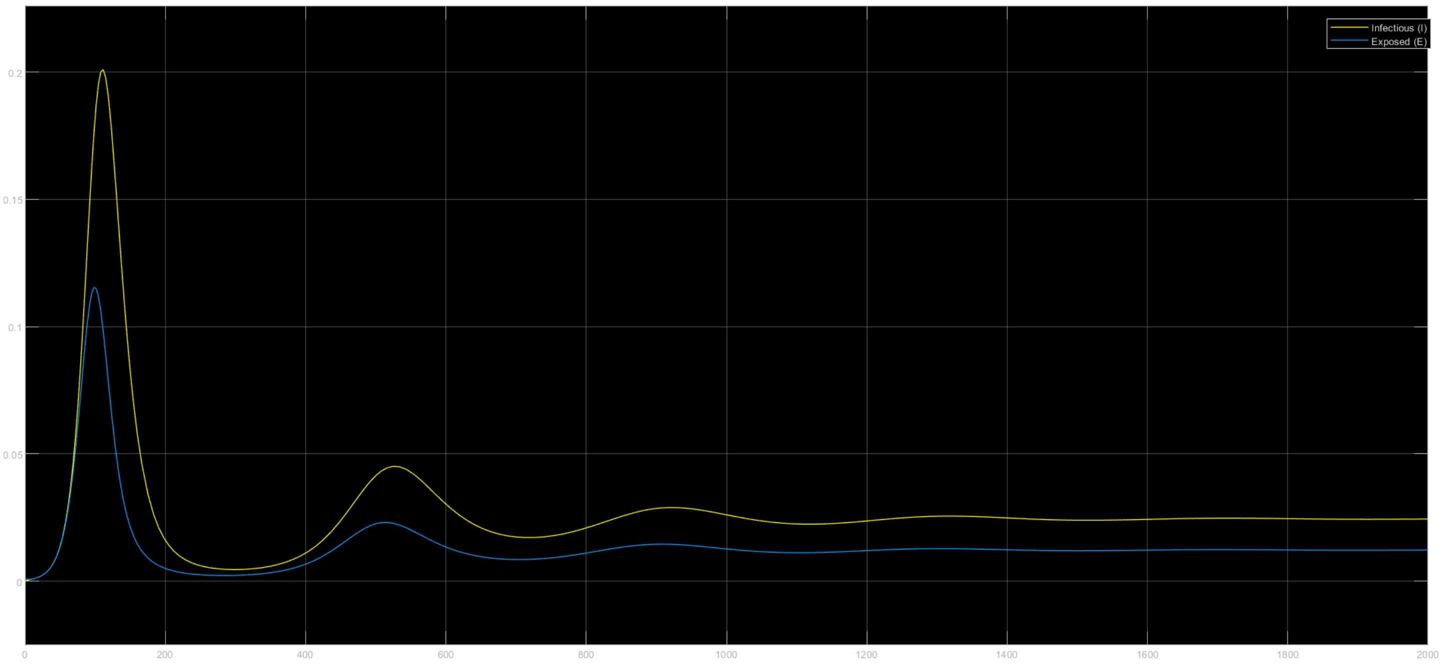

Fig 1.7 Infected and Exposed Signals Zoomed for better Visibility

3) Changing the Maximum step size
   - Maximum step size: 100
   - Relative tolerance: 1e-5
   - Solver: ode45 (variable step size)

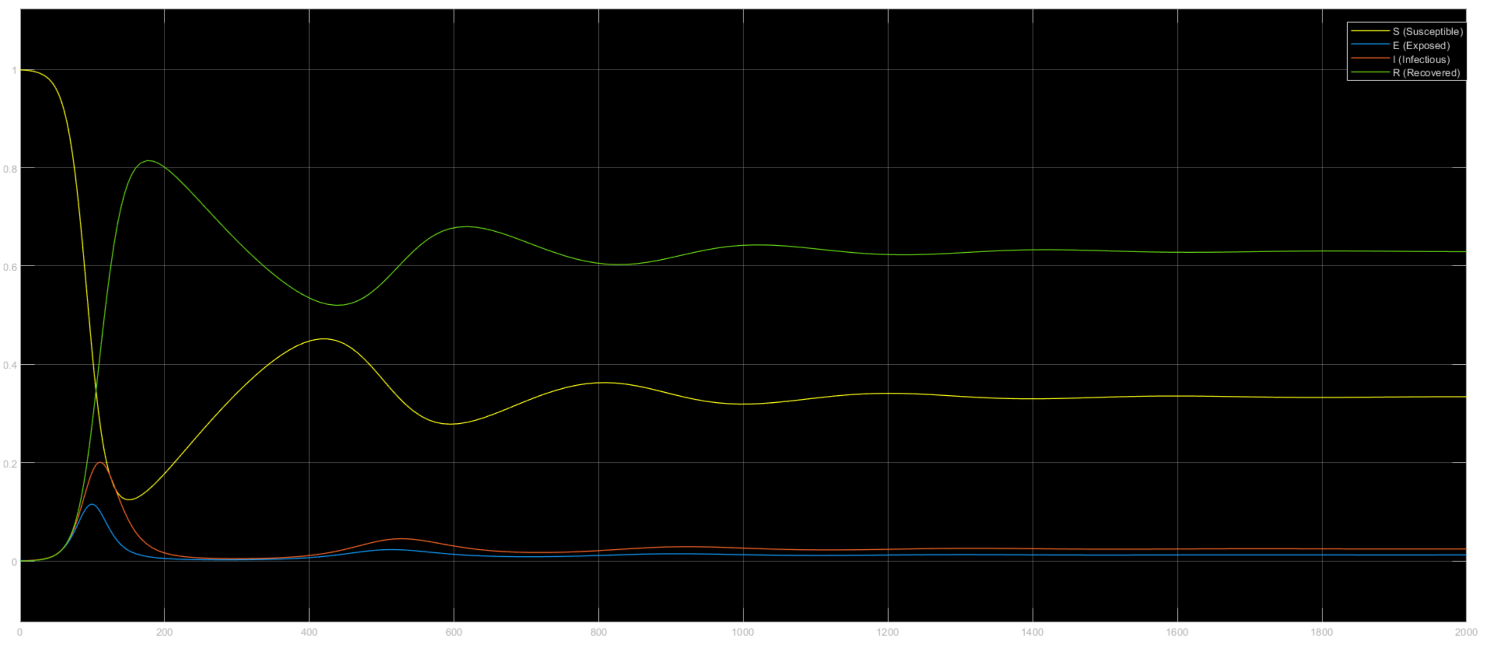

Fig 1.8 Simulink SEIRS's Model Solution

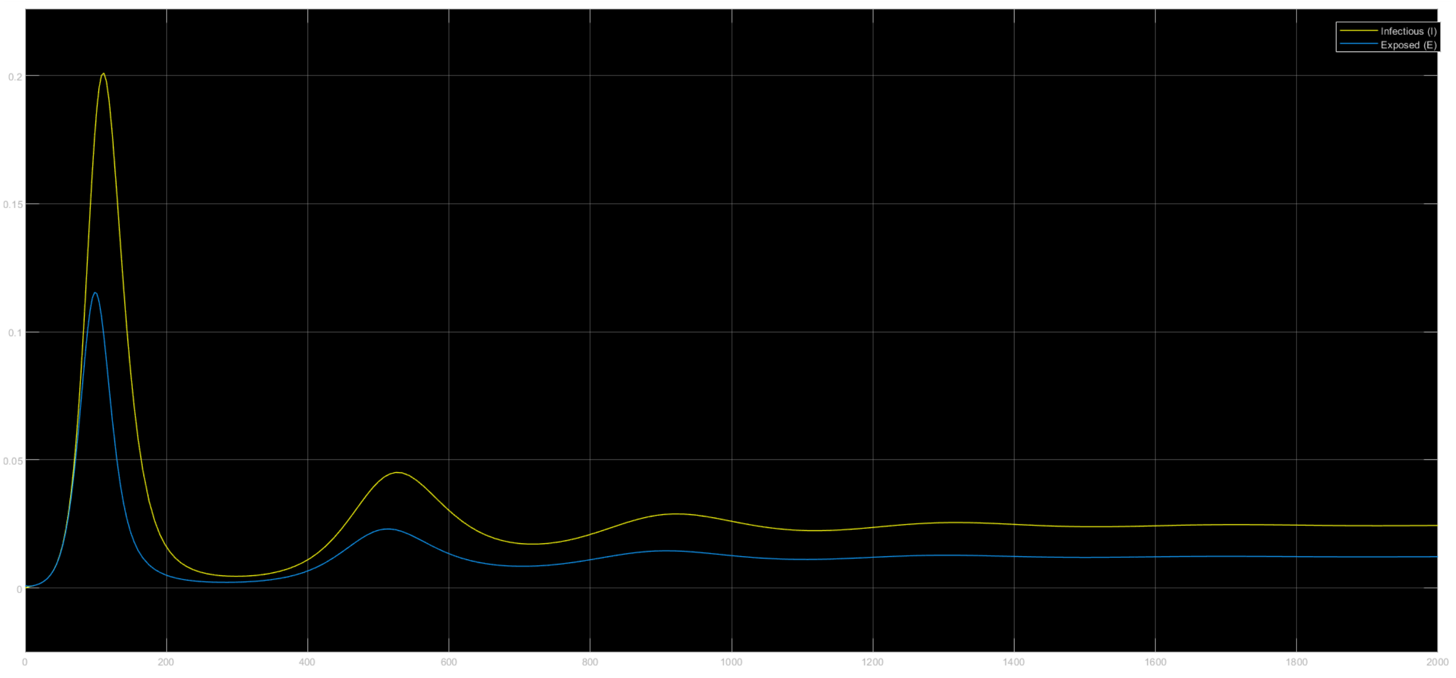

Fig 1.9 Infected and Exposed Signals Zoomed for better Visibility

4) Changing the Solver
   - Maximum step size: auto
   - Relative tolerance: 1e-5
   - Solver: ode23t (variable step size)

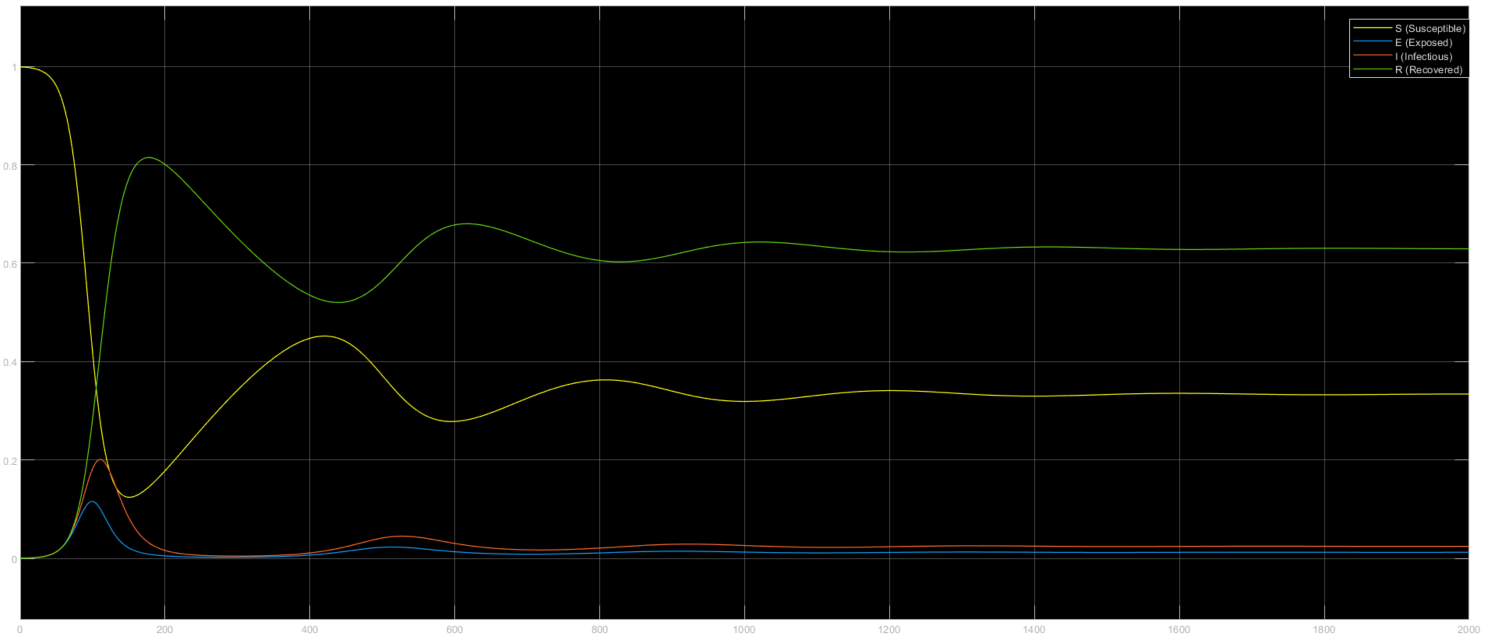

Fig 1.10 Simulink SEIRS's Model Solution

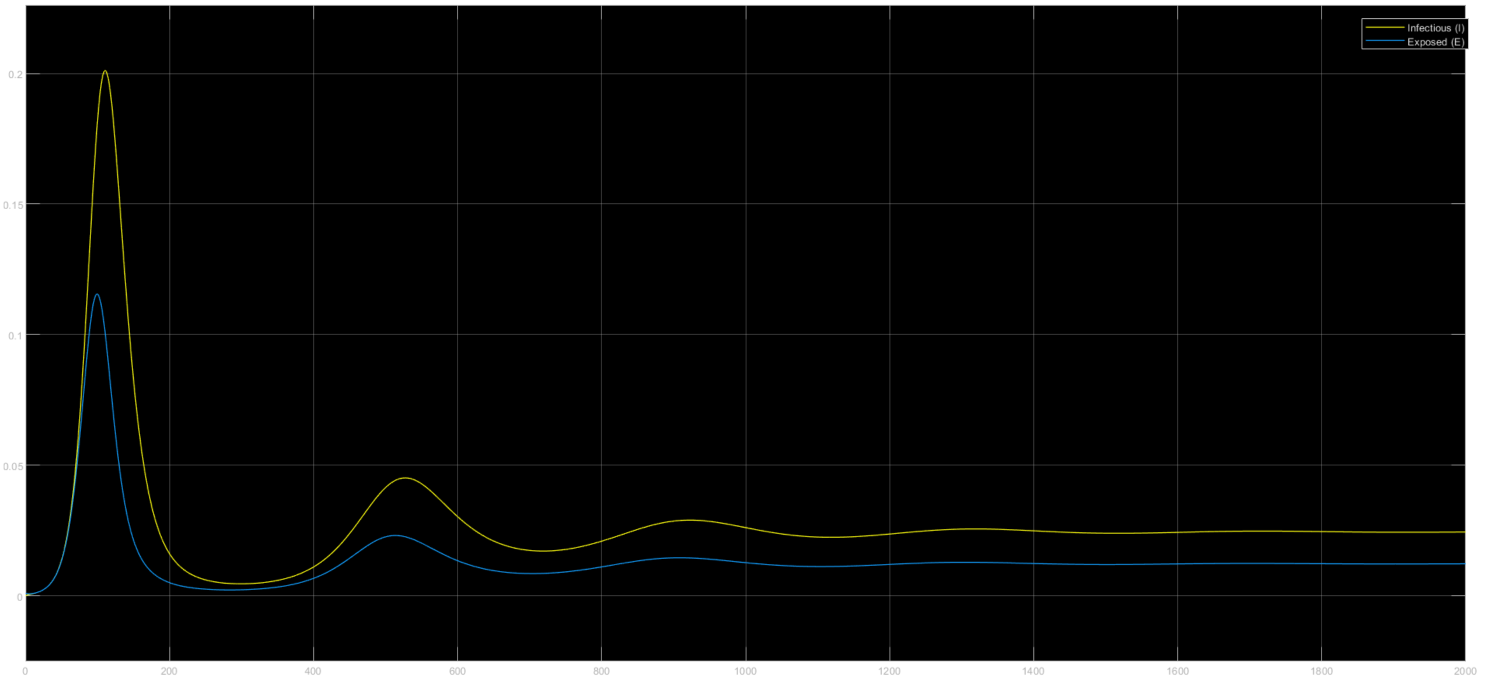

Fig 1.11 Infected and Exposed Signals Zoomed for better Visibility

5) Changing the Solver
   - Maximum step size: auto
   - Relative tolerance: 1e-5
   - Solver: ode113 (variable step size)

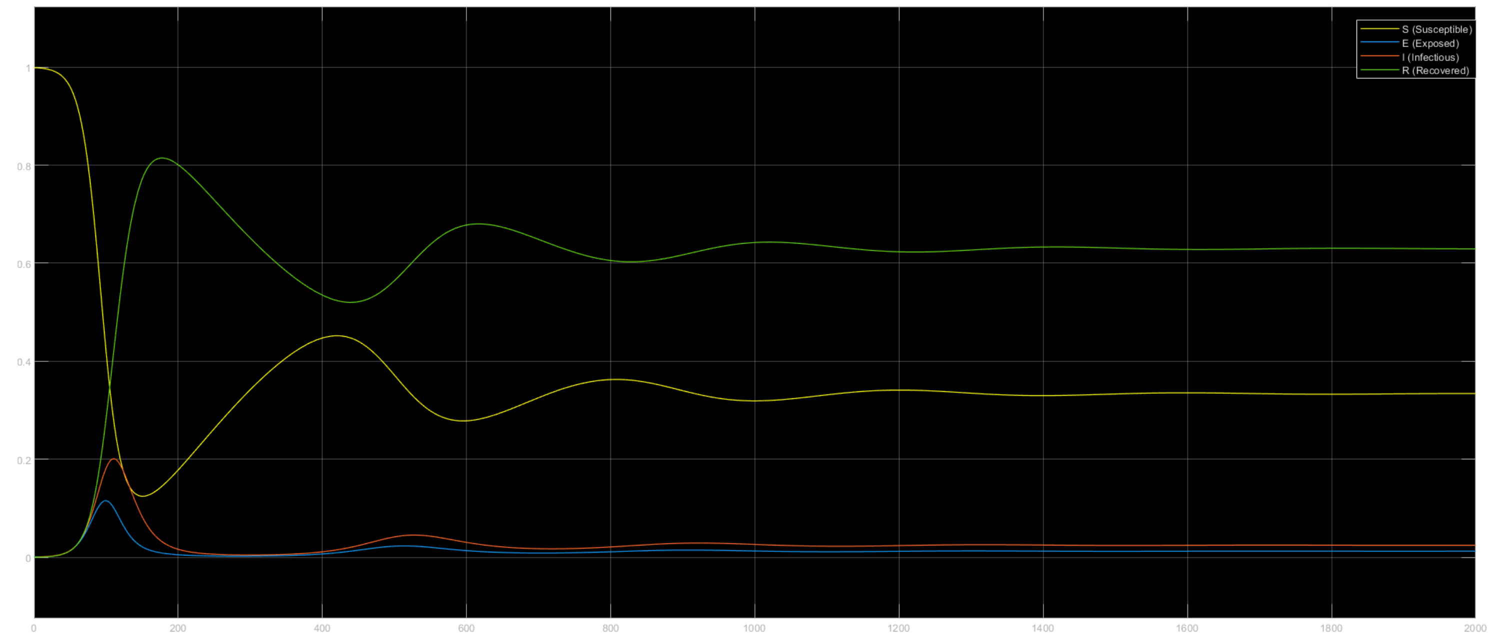

Fig 1.12 Simulink SEIRS's Model Solution

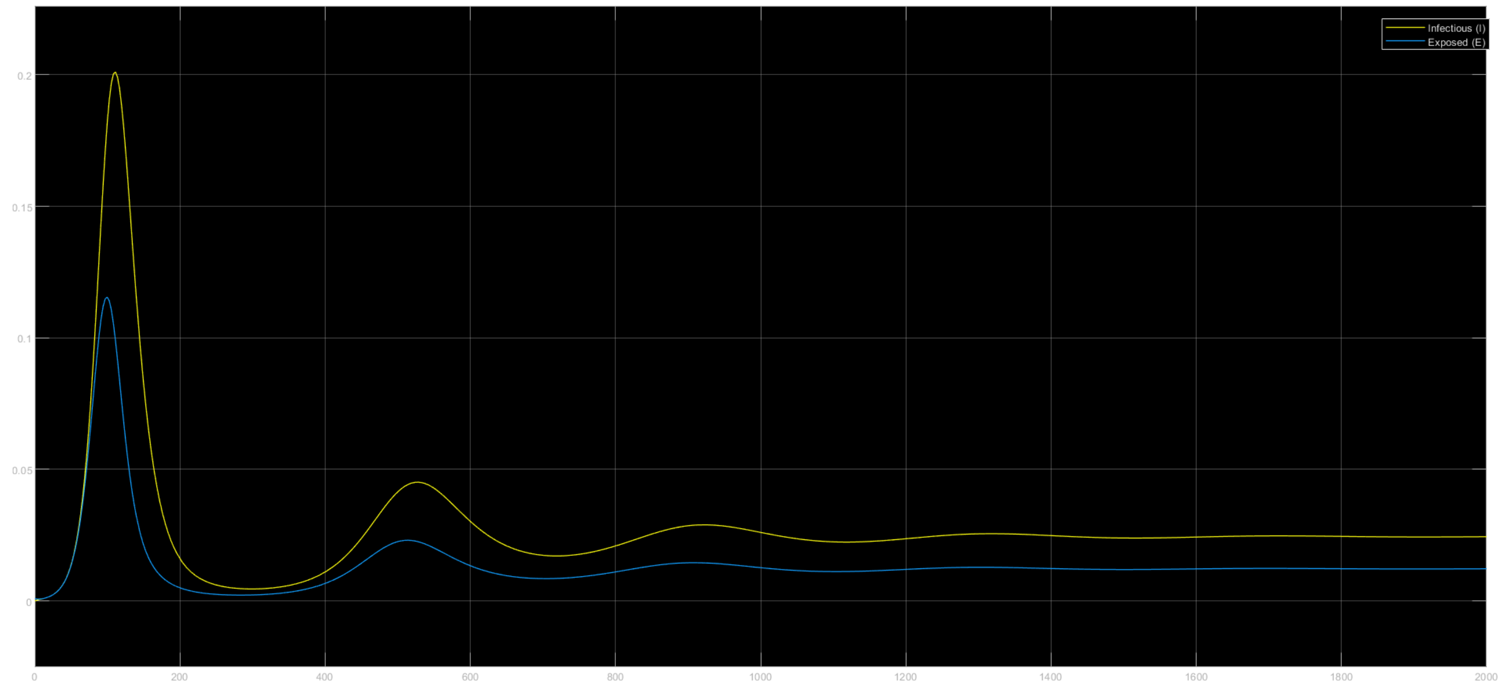

Fig 1.13 Infected and Exposed Signals Zoomed for better Visibility

6) Changing the Solver and Tolerance
- Maximum step size: auto
- Relative tolerance: 1e-1
- Solver: ode113 (variable step size)

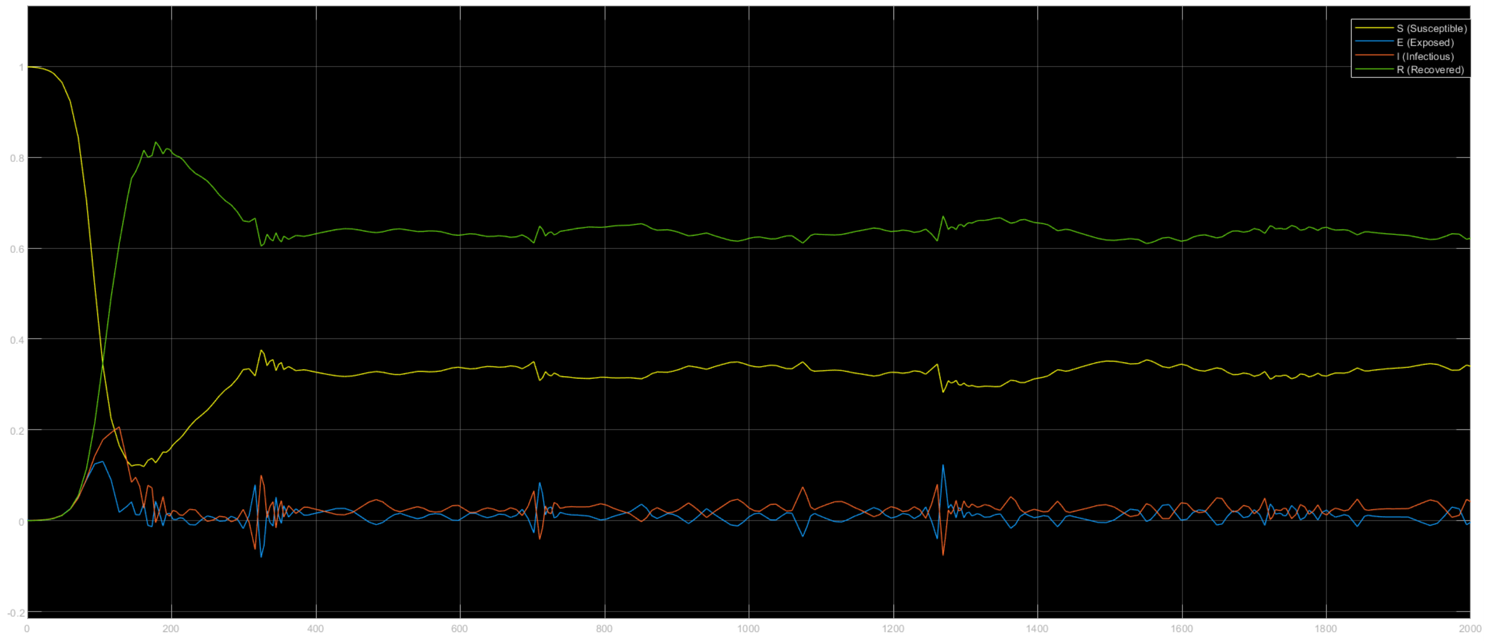

Fig 1.14 Simulink SEIRS's Model Solution

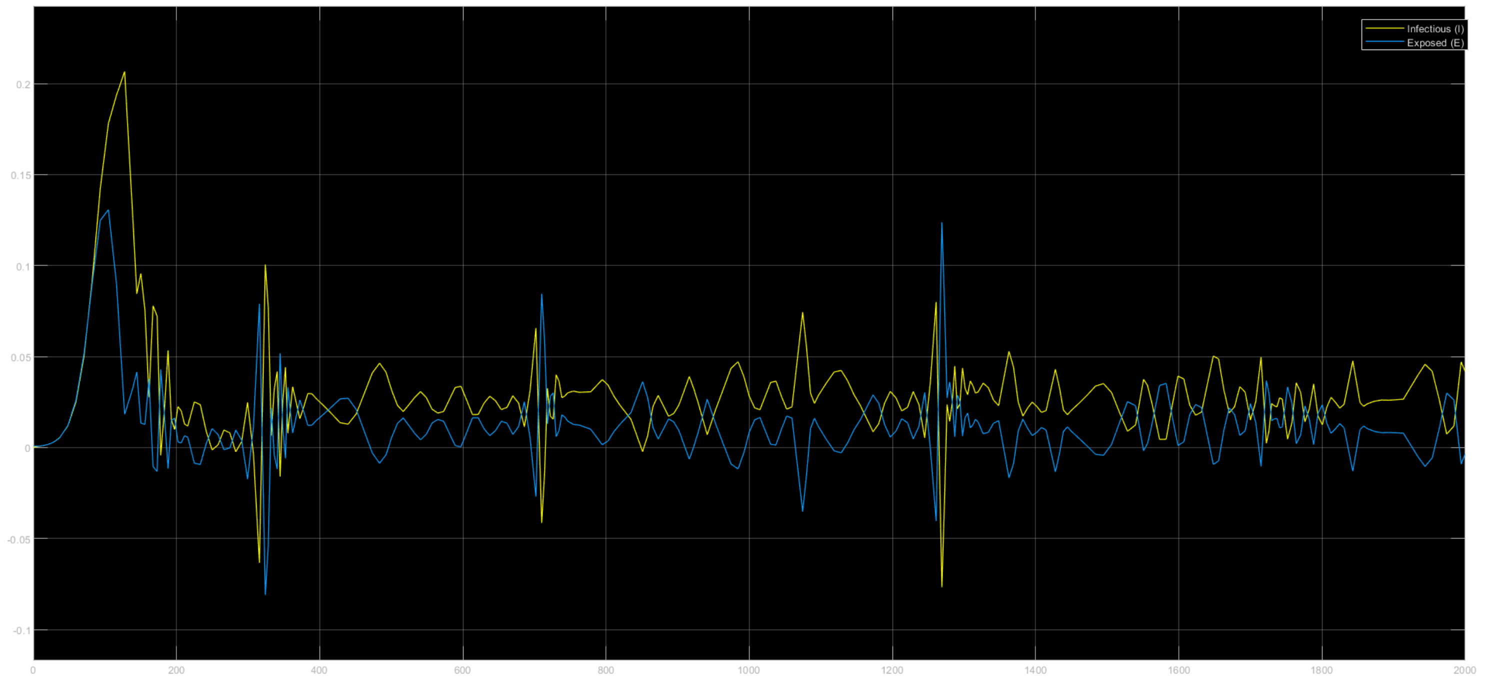

Fig 1.15 Infected and Exposed Signals Zoomed for better Visibility

7) Changing the Solver and Step size
   - Fixed step size: 0.5
   - Solver: ode8

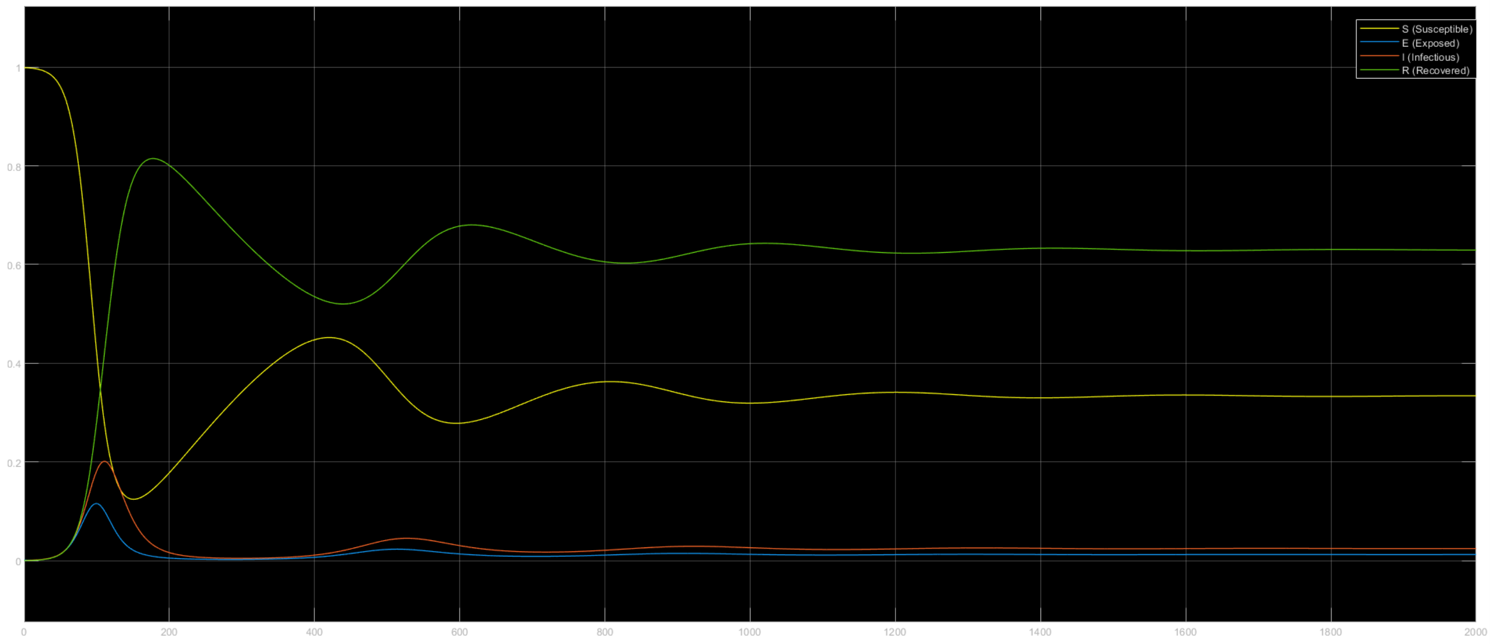

Fig 1.16 Simulink SEIRS's Model Solution

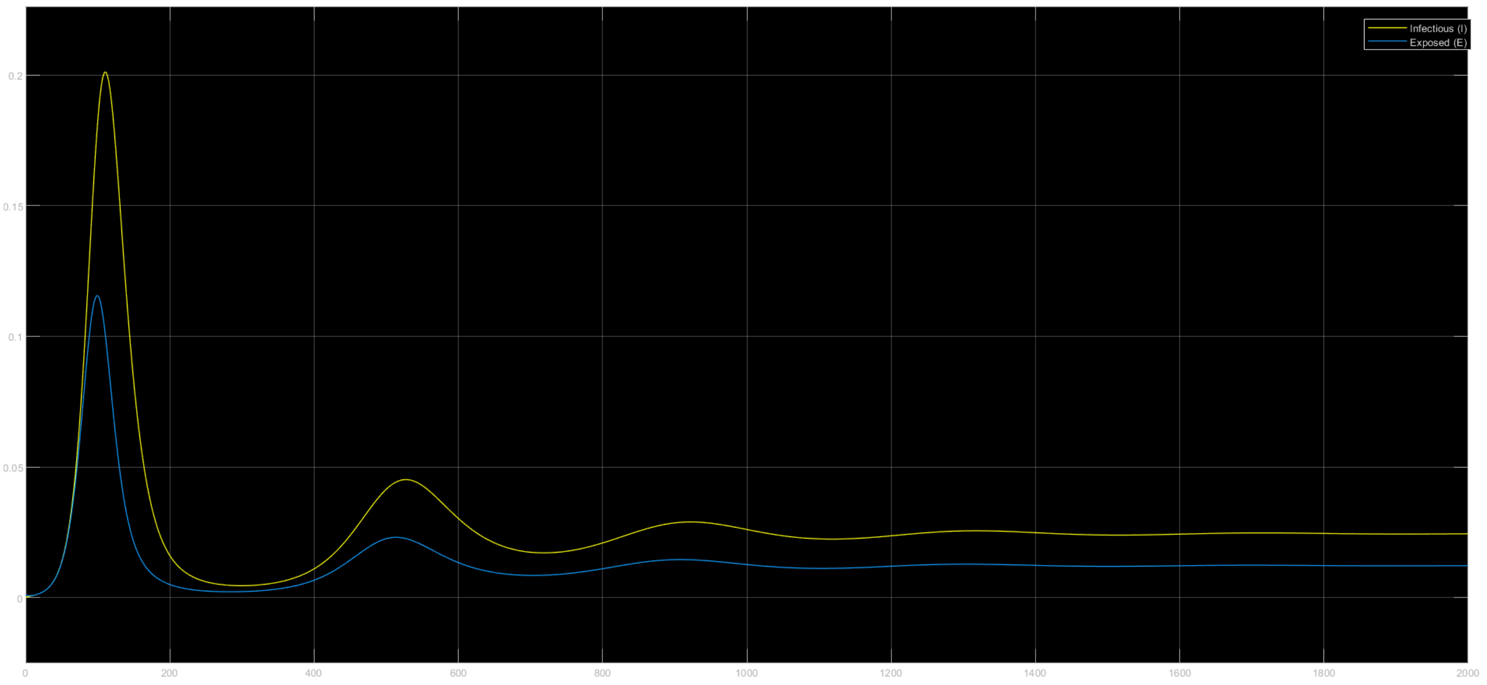

Fig 1.17 Infected and Exposed Signals Zoomed for better Visibility

8) Changing the Step size
   - Fixed step size: 2
   - Solver: ode8

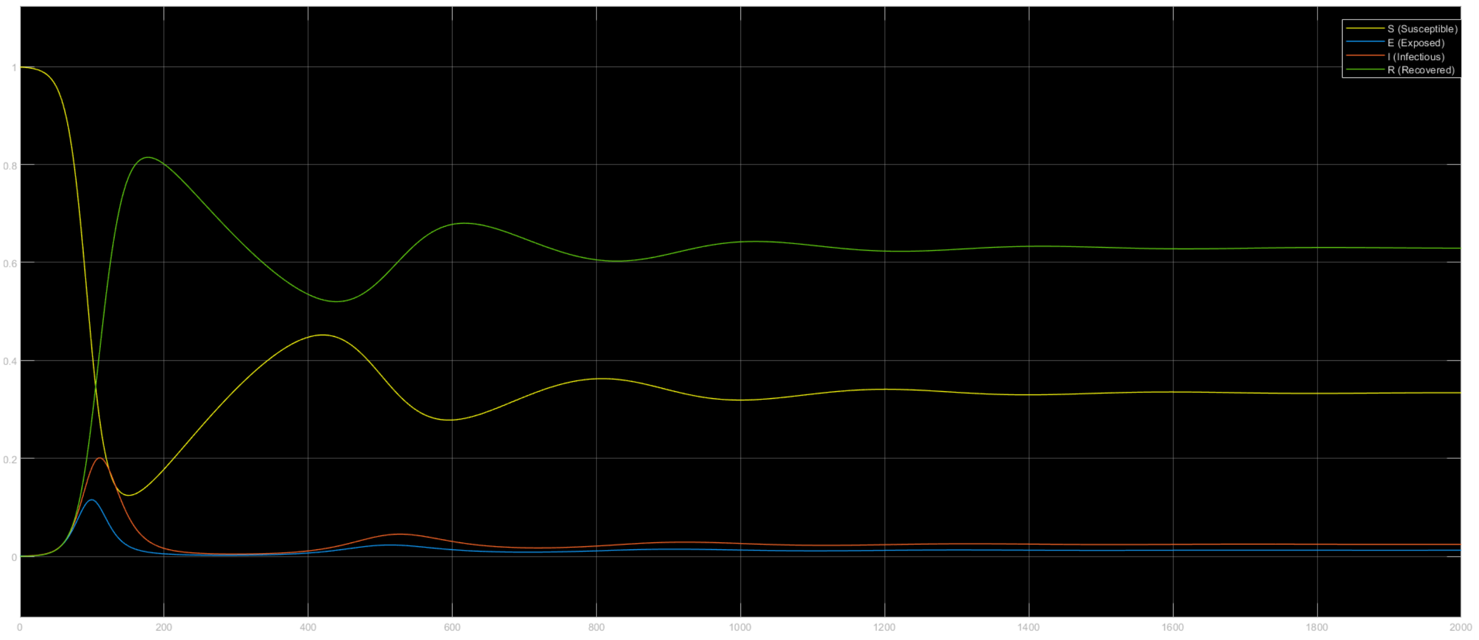

Fig 1.18 Simulink SEIRS's Model Solution

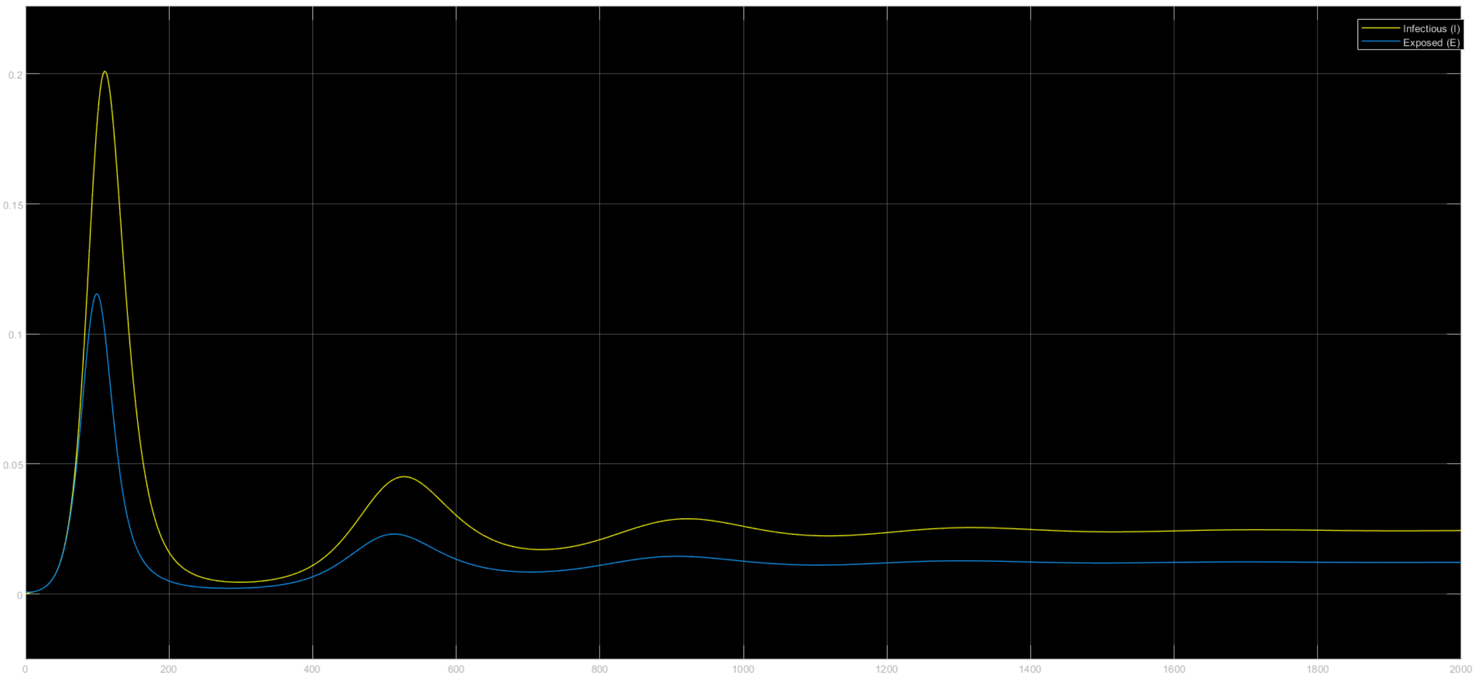

Fig 1.19 Infected and Exposed Signals Zoomed for better Visibility

9) Changing the Solver and Step size
- Fixed step size: 0.5
- Solver: ode4

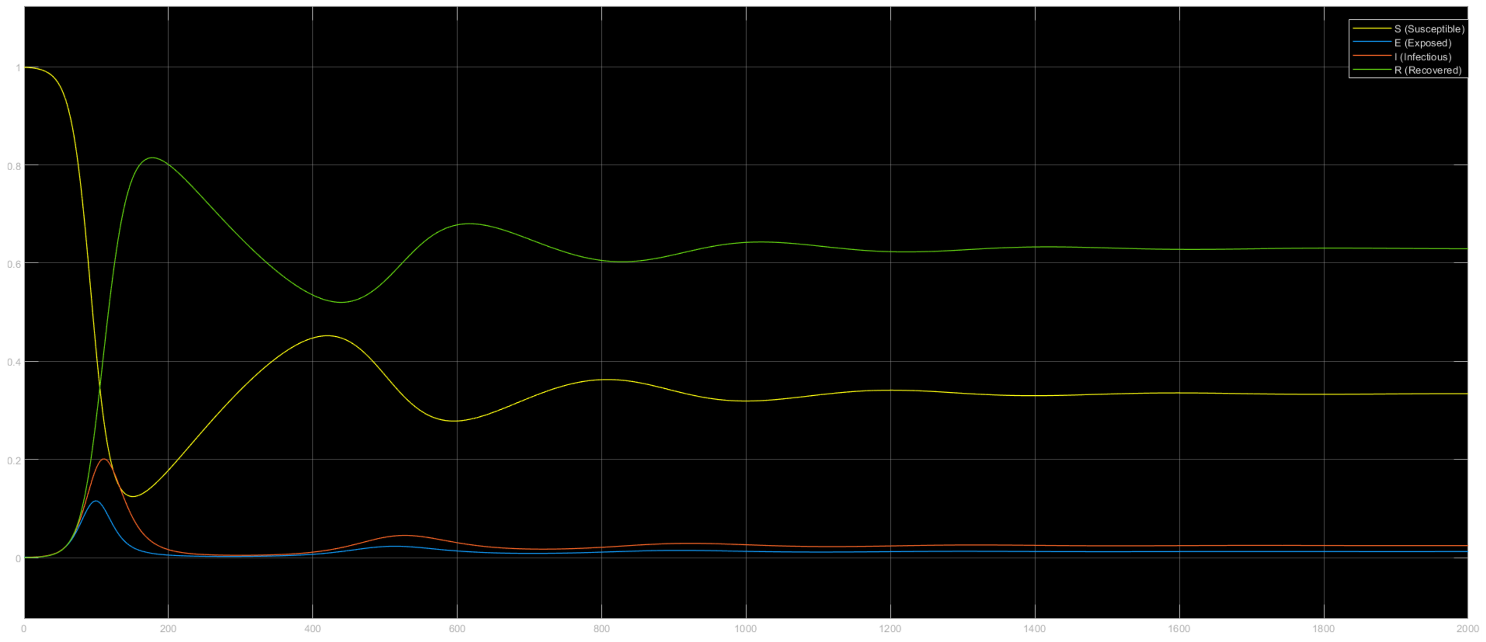

Fig 1.20 Simulink SEIRS's Model Solution

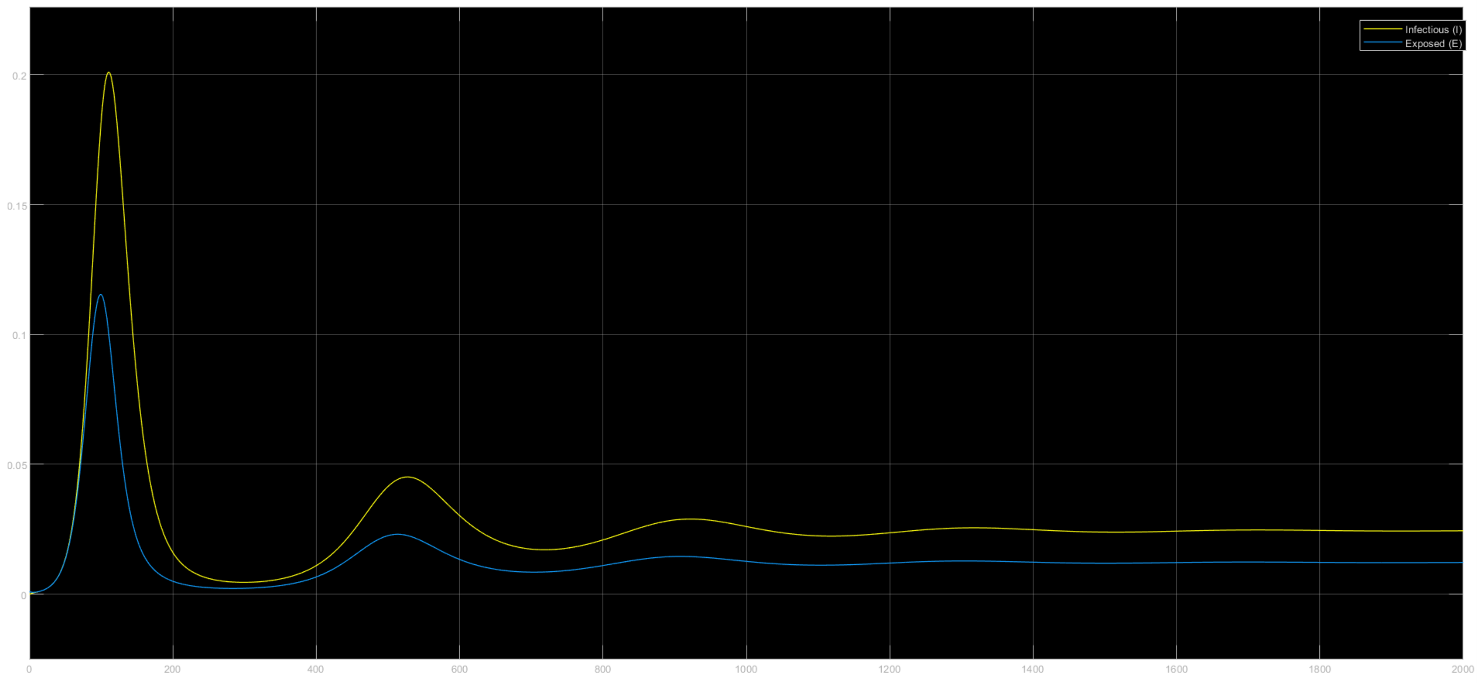

Fig 1.21 Infected and Exposed Signals Zoomed for better Visibility

10) Changing the Solver and Step size

- Fixed step size: 10
- Solver: ode4

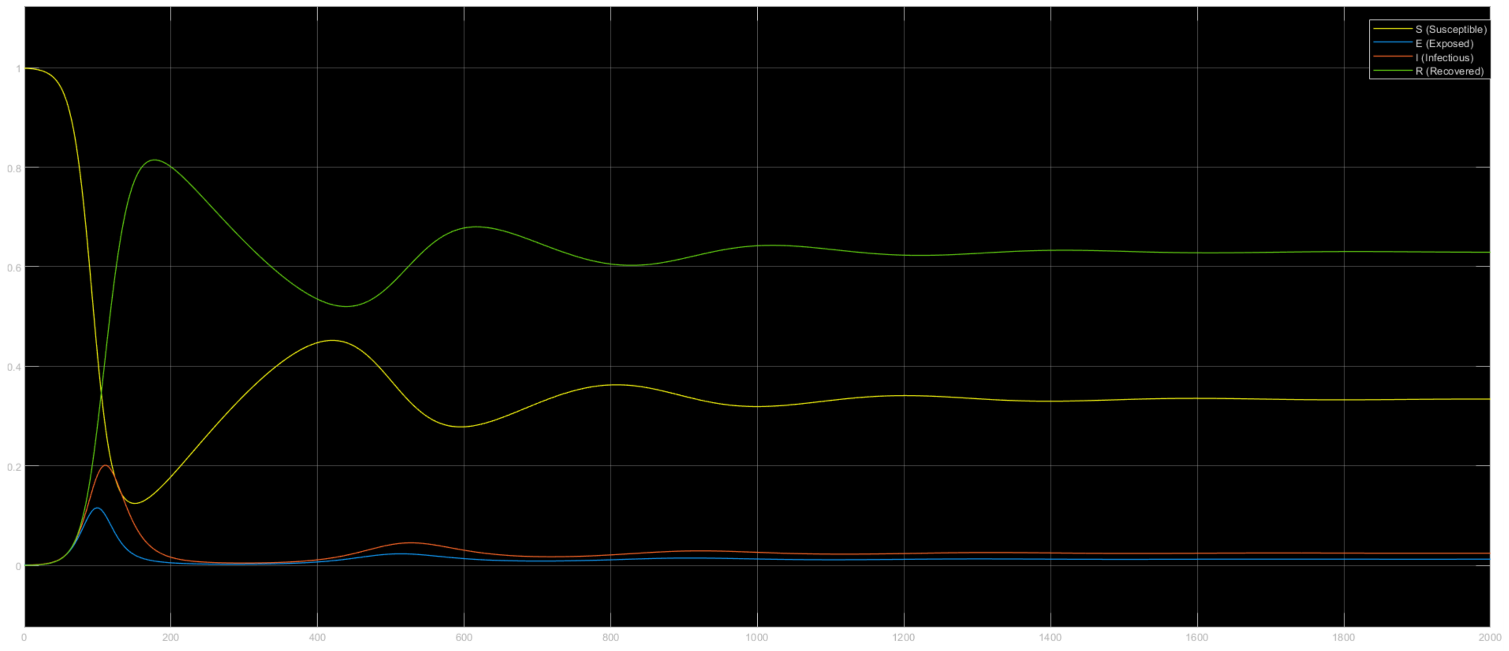

Fig 1.22 Simulink SEIRS's Model Solution

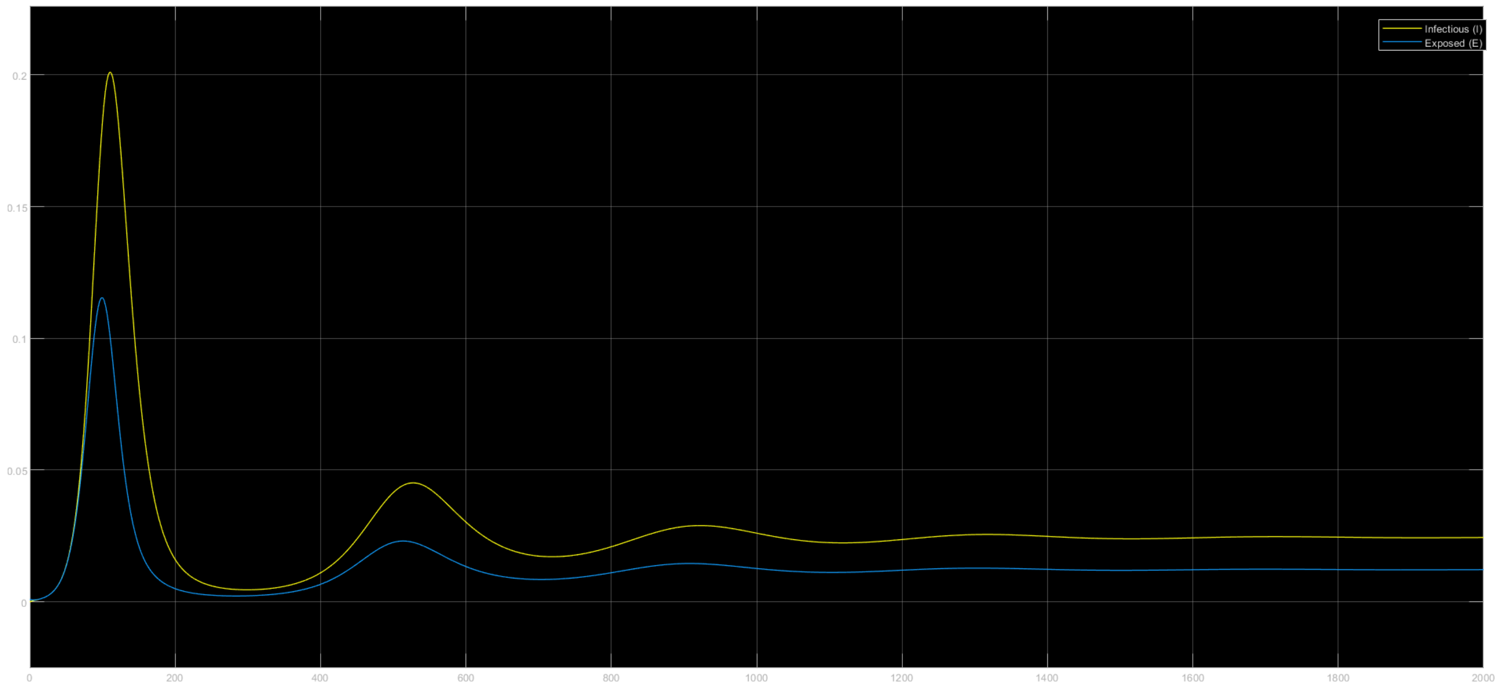

Fig 1.23 Infected and Exposed Signals Zoomed for better Visibility

11) Changing the Step size
- Fixed step size: 0.1
- Solver: ode8

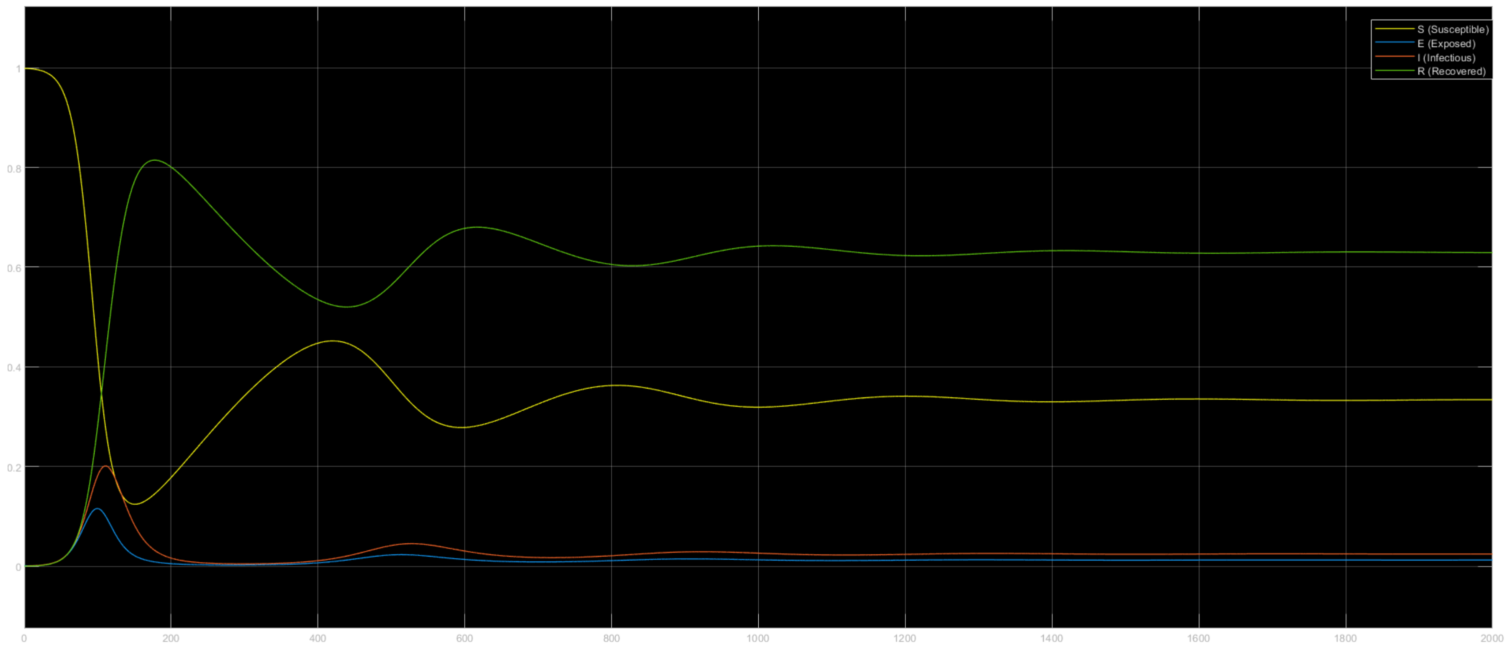

Fig 1.24 Simulink SEIRS's Model Solution

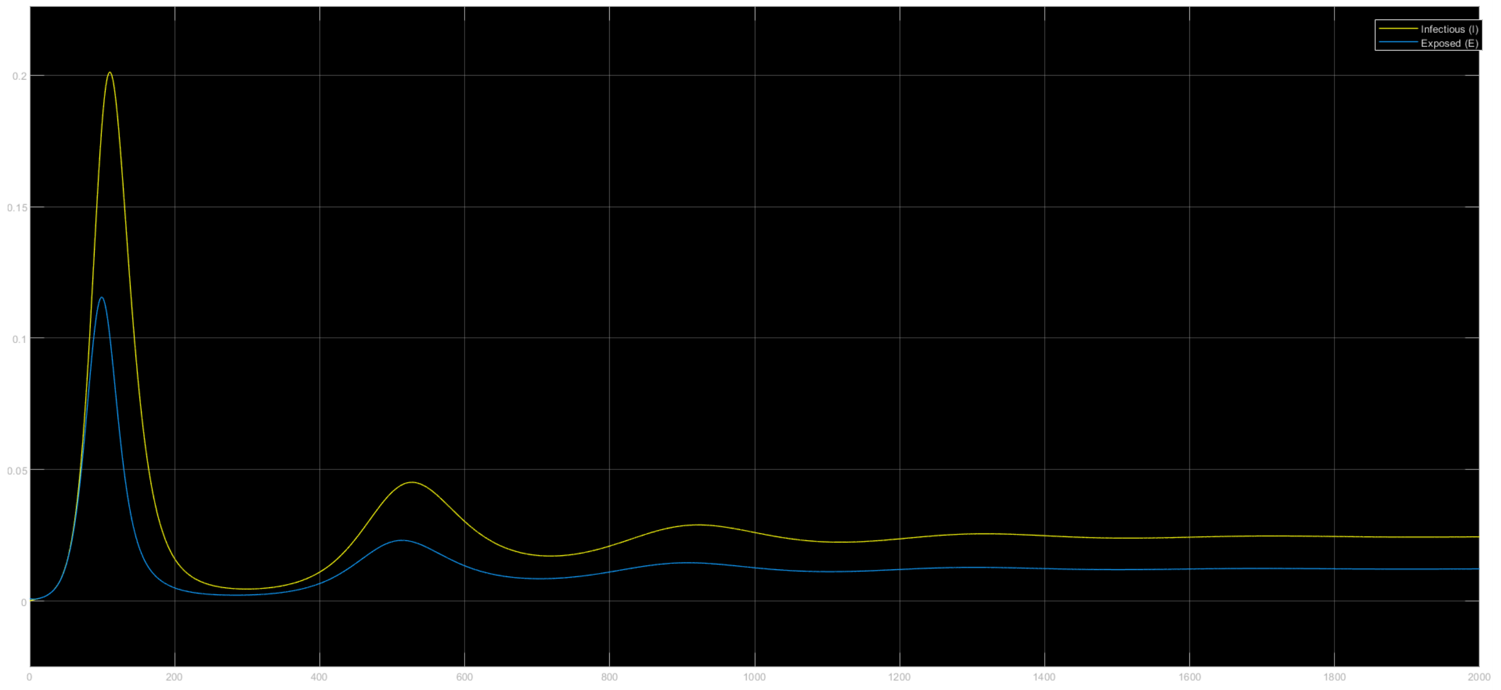

Fig 1.25 Infected and Exposed Signals Zoomed for better Visibility

12) Changing the Solver and Step size

- Fixed step size: 0.1
- Solver: ode1

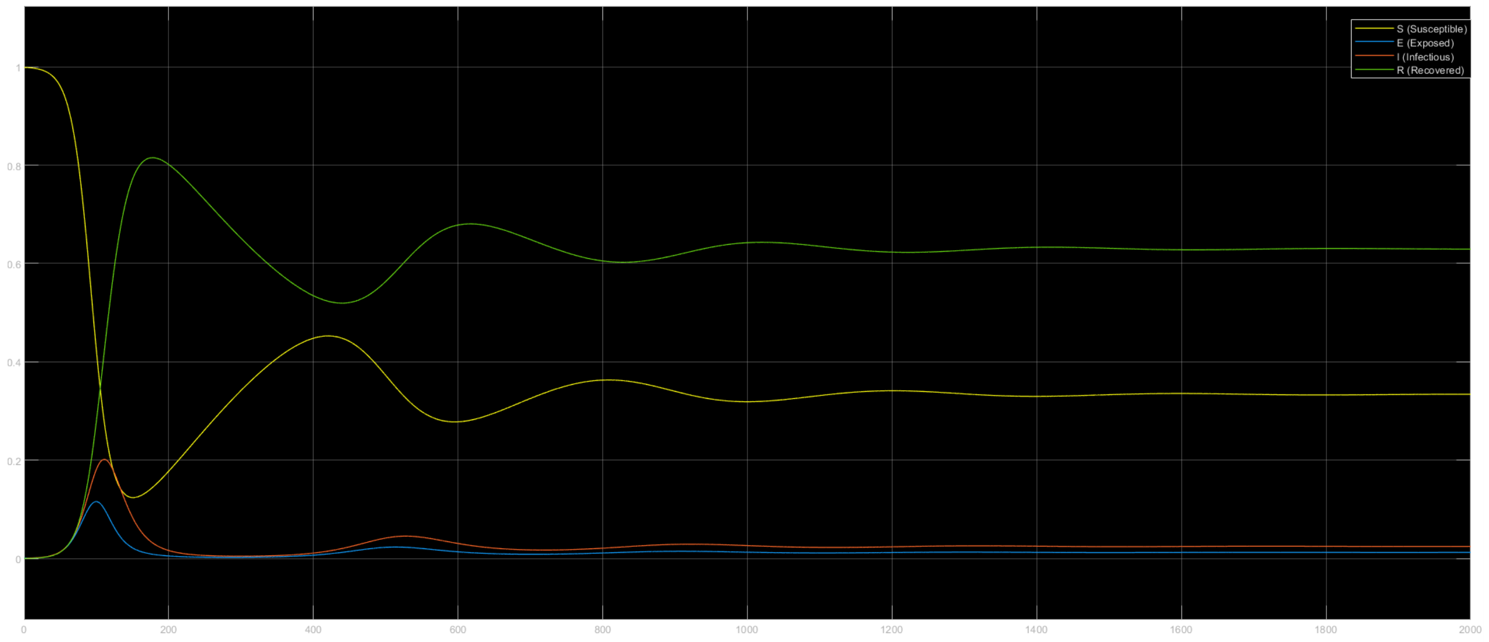

Fig 1.28 Simulink SEIRS's Model Solution

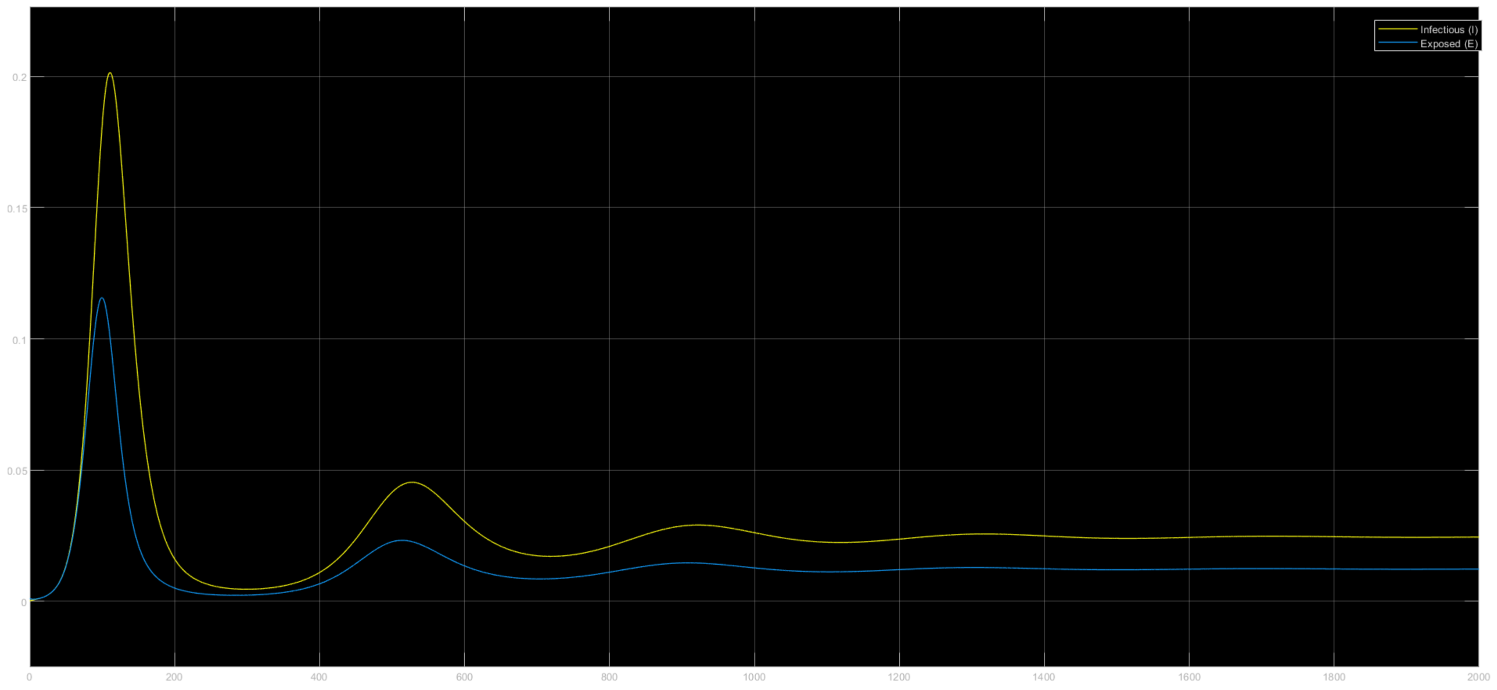

Fig 1.27 Infected and Exposed Signals Zoomed for better Visibility

13) Changing the Solver and Step size

- Fixed step size: 5
- Solver: ode1

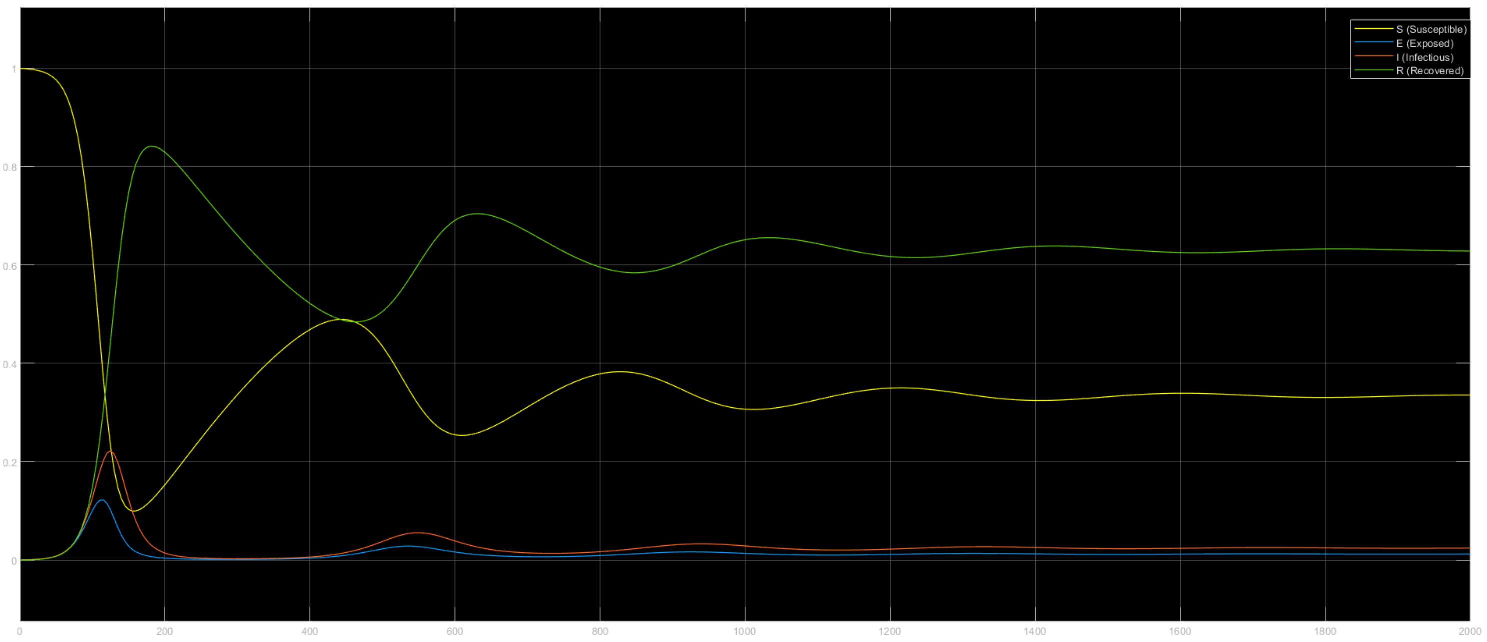

Fig 1.28 Simulink SEIRS's Model Solution

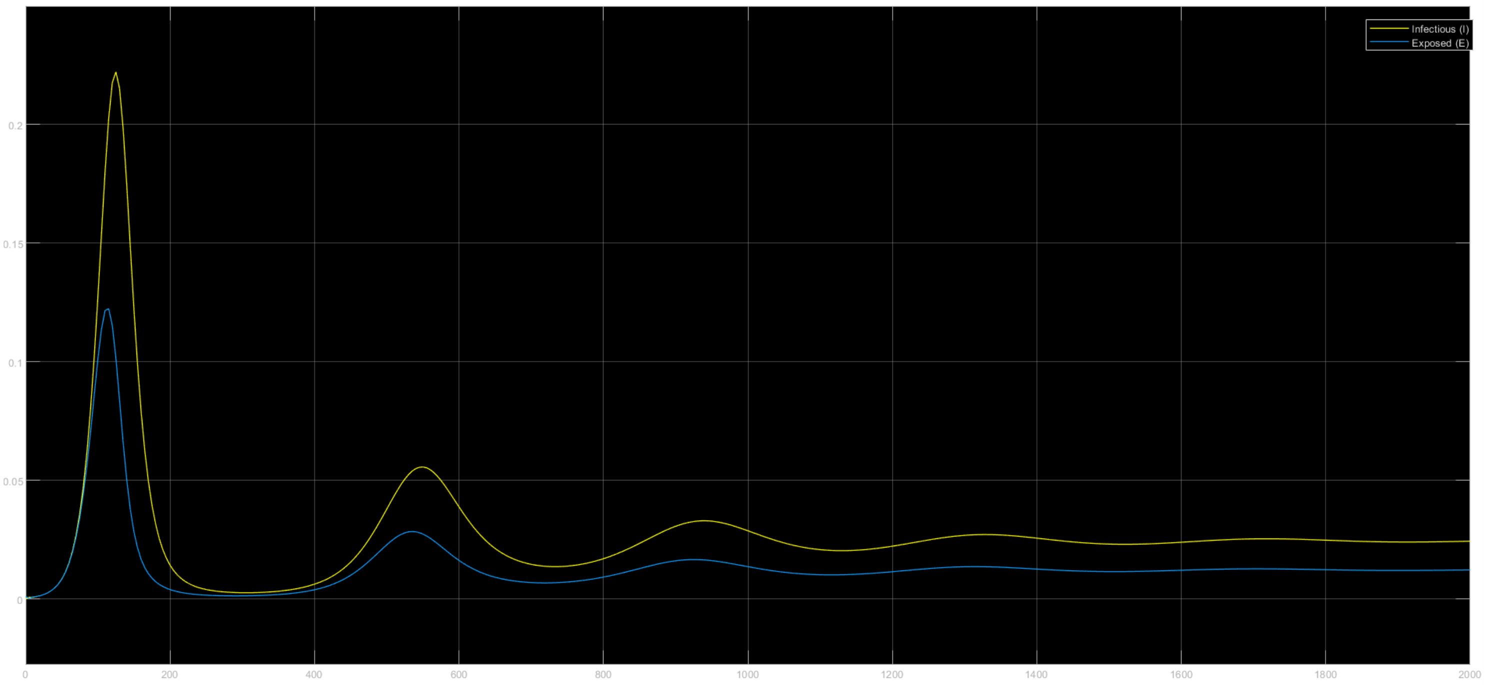

Fig 1.29 Infected and Exposed Signals Zoomed for better Visibility

**Comments**:

Since the Maximum Step size was altered by increasing it for the same solver, we achieved the same numerical solutions as indicated in the above graphs thus interpreting that the region of stability for the SEIRS model is very large thus if we exceed the ROS which may be greater than the step size of 100 we may reach instability, also by increasing the tolerance value we have seen the same behavior as instructed in the lecture that inconsistent points are generated for the same solver if the tolerance values are increased by a certain factor. Also one key thing to note is that the more we increase the step size the more overlapping of the signals in initial time steps occur which cause inconsistencies in those intermediate time steps causing the local error to be larger and global error to be non-zero up to these time steps. We also conclude that there is a difference (error) between the actual solution and the simulation results where the simulation accuracy seems to be better for small values of h and the stability seems to depend on the step size h while different systems require a different step size h for reaching the stable solution.

b)

1) To implement our own explicit Runge-Kutta algorithm based on Lecture 11 we first create a set of differential equations for the governing system, the method is of order 4 with the code given below for the standard example given in the lecture of $\dot{x} = -10x$. For a suitable value of h it must be chosen such that $|1 - 10h| < 1 \rightarrow h < 0.2$

```
% Some figure formatting
clear all;
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Runge Kutta 3 stage's method for the IVP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f = @(t,x)(-10*x); % you could use any other first-order ODE here
% Initial condition
x0 = 1;
% =======================
% Simulation
% =======================
tend = 2; % final time
h = 0.001; % small step size; this is very close to the actual solution
x(1) = x0; % state values during the simulation
time(1) = 0; % simulation time %


for kk = 1:tend/h
    x(kk+1) = RK4(time(kk),x(kk),h,f);
    time(kk+1) = kk*h;
end
disp(['Final value of x for h = 0.001: ', num2str(x(end))]);
figure;
plot(time,x);
xlabel('time [sec]')
ylabel('x')
grid on;
hold all;

% Different step size
h = 0.05; % larger step size
clear x time;
```

```
    x(1) = x0;
    time(1) = 0;
    for kk = 1:tend/h
        x(kk+1) = RK4(time(kk),x(kk),h,f);
        time(kk+1) = kk*h;
    end
    disp(['Final value of x for h = 0.05: ', num2str(x(end))]);
    plot(time,x);

    % Different step size
    h = 0.1; % larger step size
    clear x time;
    x(1) = x0;
    time(1) = 0;
    for kk = 1:tend/h
        x(kk+1) = RK4(time(kk),x(kk),h,f);
        time(kk+1) = kk*h;
    end
    disp(['Final value of x for h = 0.1: ', num2str(x(end))]);
    plot(time,x);

    % Different step size
    h = 0.16; % larger step size
    clear x time;
    x(1) = x0;
    time(1) = 0;
    for kk = 1:tend/h
        x(kk+1) = RK4(time(kk),x(kk),h,f);
        time(kk+1) = kk*h;
    end
    disp(['Final value of x for h = 0.16: ', num2str(x(end))]);
    plot(time,x);

    % Different step size
    h = 0.25; % larger step size
    clear x time;
    x(1) = x0;
    time(1) = 0;
    for kk = 1:tend/h
        x(kk+1) = RK4(time(kk),x(kk),h,f);
        time(kk+1) = kk*h;
    end
    disp(['Final value of x for h = 0.25: ', num2str(x(end))]);
    plot(time,x);

    legend({'h = 0.001','h = 0.05','h = 0.1','h = 0.16','h = 0.25'},'FontSize',14)
    title('$\dot{x} = -10x$', 'Interpreter', 'latex')
    set(findall(gcf,'Type','line'),'LineWidth',2)
    set(findall(gcf,'-property','FontSize'),'FontSize',14);

function xNext = RK4(t,x,h,f)
% =============================
% Function computes one step of Heun's method
% Inputs
% - x: current state
% - h: step size
% - f: right hand side of the IVP to be simulated (given as function
% handle): https://www.mathworks.com/help/matlab/matlab_prog/pass-a-function-to-another-
function.html
% Output
% - xNext: state in the next time step

% Butcher Tableu Values use in Runge Kutta 4 stages%
    c2 = 1/2;
    c3 = 1/2;
    c4 = 1;
    a21 = 1/2;
    a31 = 0;
    a32 = 1/2;
    a41 = 0;
    a42 = 0;
    a43 = 1;
    b1 = 1/6;
    b2 = 1/3;
    b3 = 1/3;
    b4 = 1/6;
% End of Butcher Tableu Values%

k1 = f(t,x);
k2 = f(t+c2*h,x+a21*h*k1);
k3 = f(t+c3*h,x+h*a31*k1+h*a32*k2);
k4 = f(t+c4*h,x+h*a41*k1+h*a42*k2+h*a43*k3);

xNext = x + h*(b1*k1+b2*k2+b3*k3+b4*k4);
```

```
Output of the code is as follows:
Final value of x for h = 0.001: 2.0612e-09
Final value of x for h = 0.05: 2.0941e-09
Final value of x for h = 0.1: 3.0243e-09
Final value of x for h = 0.16: 1.5278e-07
Final value of x for h = 0.25: 0.031257
```

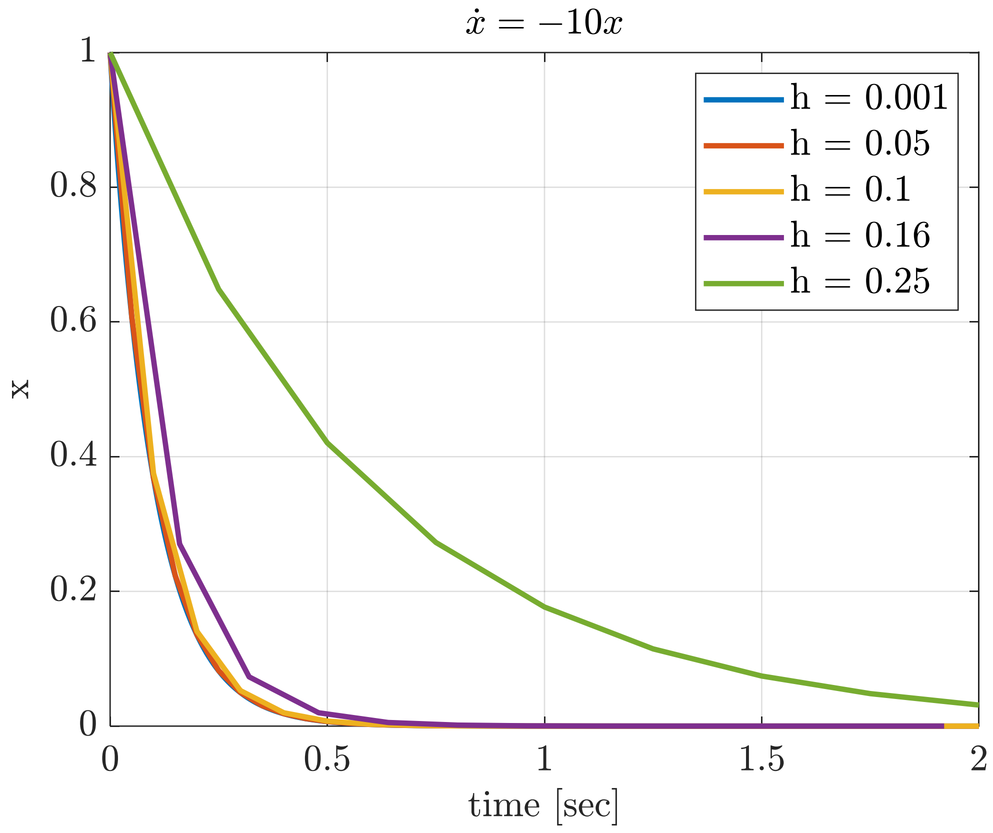

Fig 1.30 RK4 Method for $\dot{x} = -10x$ with varying step sizes

2) Learn about the realization of a variable step size. The relevant chapter of the textbook is provided on odtuclass. Summarize the main implementation steps you need for a solver with variable step size.

Ans: To effectively solve ODEs and DAEs numerically, it's crucial to ensure that the local error at each integration step remains within a specified tolerance, $\varepsilon$. If the local error estimate $|\eta_{k+1}|$ exceeds this tolerance, the step size must be reduced to maintain accuracy. Conversely, if the local error is within the acceptable range, the step size can be increased to improve efficiency.

The error estimation is based on the difference between two solutions obtained from Runge-Kutta methods of different orders ($p$ and $p-1$). For the k-th step, the local error is approximated as:

$$\eta_{k+1} = y(t_{k+1}) - \hat{y}_{k+1} = y_{k+1} - \hat{y}_{k+1} = \hat{C}h^p + O(h^{p+1})$$

where $\hat{y}^{k+1}$ is the result from the lower order method. If the local error estimate $|\eta^{k+1}|$ exceeds the tolerance $\varepsilon$, a new step size $\bar{h}$ is computed using $\bar{h} = h\left(\frac{\varepsilon}{|\eta^{k+1}|}\right)^{\frac{1}{p}}$. This new step size ensures that the local error for the subsequent step meets the desired tolerance. The step size adjustment formula typically includes factors to prevent drastic changes in step size: $\bar{h} = h\min\left(fac_1, \max\left(fac_0, \beta\left(\left(\frac{|\eta^{k+1}|}{\varepsilon}\right)^{-\frac{1}{p}}\right)\right)\right)$ with typical values for these parameters being $\beta = 0.9, fac_0 = 0.2, and\ fac_1 = 5$, ensuring that $0.2h \le \bar{h} \le 5h$.

For vector functions $y(t)$, error estimation and step size computation consider the scaling of elements:

$$\sigma = \left[ \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left( \frac{|\eta_i^{(k+1)}|}{\mathtt{atol}_i + \mathtt{rtol}_i |y_i^{(k+1)}|} \right)^2 } \right]^{-1/p}$$

$$\bar{h} = h \min(\mathtt{fac}_1, \max(\mathtt{fac}_0, \beta\sigma)).$$

This ensures that the computed step size adjusts appropriately according to both absolute $(atol)$ and relative $(rtol)$ error tolerances. Embedded Runge-Kutta methods are particularly efficient for these schemes as they provide both the primary solution and an error estimate from a single set of function evaluations, facilitating adaptive step size control.

3) and 4)

The code has been reconfigured to handle the SEIRS model with a system of ODE’s and also now includes the step size adaptation and multiple fixed step sizes to have a comparative model as well as tic and toc to find the relative speed up of the model compared to the fixed step size.

% FIXED STEP SIZE CODE %

```
clear all;
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Runge-Kutta 4 stage's method for the IVP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R0 = 3;
Gamma = 1/14;
Sigma = 1/7;
Omega = 1/365;
Mue = 1/(76*365);
Alpha = 0;
Beta = R0 * Gamma;

tend = 2000; % final time

% Initial condition of ODE's
S0 = 0.999;
E0 = 0.001;
I0 = 0.000;
R0 = 0.000;

time_steps = [0.01, 0.1, 1, 2, 5, 10]; % different time steps

% Loop over different time steps
for j = 1:length(time_steps)
    h = time_steps(j);

    % Initialize time and solution vectors
    time = 0:h:tend;
    y = zeros(4, length(time));
    y(:,1) = [S0; E0; I0; R0]; % Initial values
    tic; % Start timing fixed step size method
    for i = 1:length(time)-1
        k1 = ODESystem(time(i), y(:,i), Alpha, Beta, Gamma, Sigma, Mue, Omega);
        k2 = ODESystem(time(i)+h/2, y(:,i)+h*k1/2, Alpha, Beta, Gamma, Sigma, Mue, Omega);
        k3 = ODESystem(time(i)+h/2, y(:,i)+h*k2/2, Alpha, Beta, Gamma, Sigma, Mue, Omega);
        k4 = ODESystem(time(i)+h, y(:,i)+h*k3, Alpha, Beta, Gamma, Sigma, Mue, Omega);
        y(:,i+1) = y(:,i) + (1/6)*(k1+2*k2+2*k3+k4)*h;
    end
    fixed_time = toc; % End timing fixed step size method

    S = y(1,:);
    E = y(2,:);
    I = y(3,:);
    R = y(4,:);

    disp(['Fixed Step Size Method - Execution Time for h = ', num2str(h), ': ',
num2str(fixed_time), ' seconds']);
    disp(['Final values for h = ', num2str(h), ': S = ', num2str(S(end)), ', E = ',
num2str(E(end)), ', I = ', num2str(I(end)), ', R = ', num2str(R(end))]);

    figure;
    plot(time, S, 'k', 'DisplayName', 'Susceptible');
    hold on;
    plot(time, E, 'g', 'DisplayName', 'Exposed');
    plot(time, I, 'r', 'DisplayName', 'Infected');
    plot(time, R, 'b', 'DisplayName', 'Recovered');
    xlabel('Time [days]');
    ylabel('Population');
    legend('show');
    title(['SEIR Model Simulation using Fixed Step Size Method (h = ', num2str(h), ')']);
    hold off;
end

% Nested function for ODE system
function dydt = ODESystem(time, y, Alpha, Beta, Gamma, Sigma, Mue, Omega)
    S = y(1);
    E = y(2);
    I = y(3);
    R = y(4);
    dydt = zeros(4,1);
    dydt(1) = Mue * (S + E + I + R) - (Beta * I * S / (S + E + I + R)) + Omega * R - Mue * S;
    dydt(2) = (Beta * I * S / (S + E + I + R)) - Sigma * E - Mue * E;
    dydt(3) = Sigma * E - Gamma * I - (Mue + Alpha) * I;
    dydt(4) = Gamma * I - Omega * R - Mue * R;
end
```

Output:

```
Fixed Step Size Method - Execution Time for h = 0.01: 1.0581 seconds
Final values for h = 0.01: S = 0.33408, E = 0.012233, I = 0.024429, R = 0.62926

Fixed Step Size Method - Execution Time for h = 0.1: 0.097566 seconds
Final values for h = 0.1: S = 0.33408, E = 0.012233, I = 0.024429, R = 0.62926

Fixed Step Size Method - Execution Time for h = 1: 0.0091883 seconds
Final values for h = 1: S = 0.33408, E = 0.012233, I = 0.024429, R = 0.62926

Fixed Step Size Method - Execution Time for h = 2: 0.0048642 seconds
```

```
Final values for h = 2: S = 0.33408, E = 0.012233, I = 0.024429, R = 0.62926

Fixed Step Size Method - Execution Time for h = 5: 0.0023608 seconds
Final values for h = 5: S = 0.33408, E = 0.012233, I = 0.024429, R = 0.62926

Fixed Step Size Method - Execution Time for h = 10: 0.0010452 seconds
Final values for h = 10: S = 0.33408, E = 0.012233, I = 0.024429, R = 0.62926
```

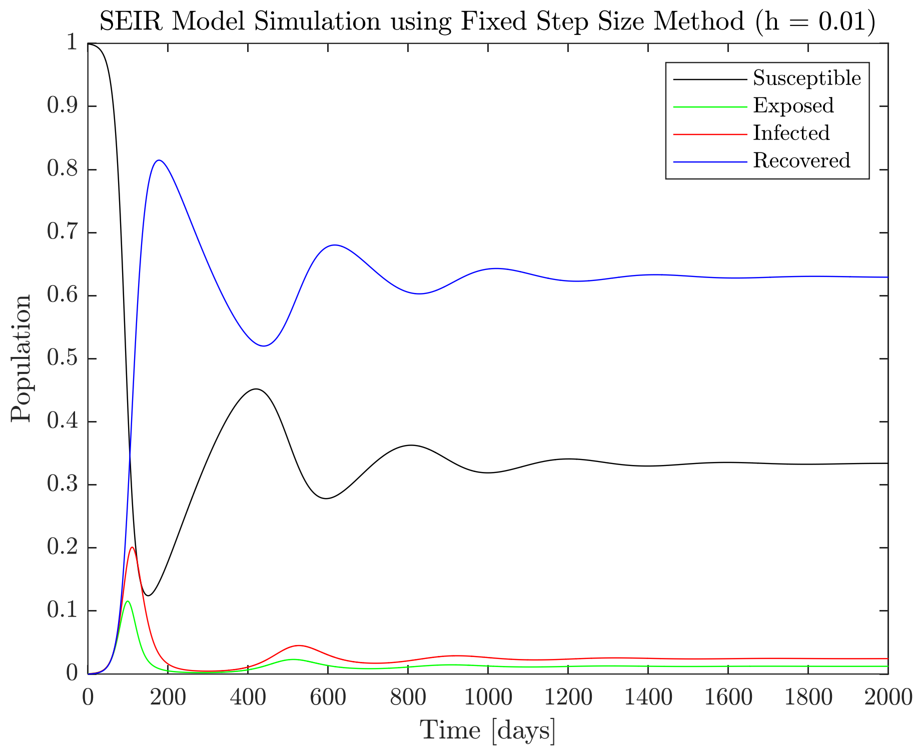

Fig 1.31 SEIRS Model with Fixed Step Size of h =0.01

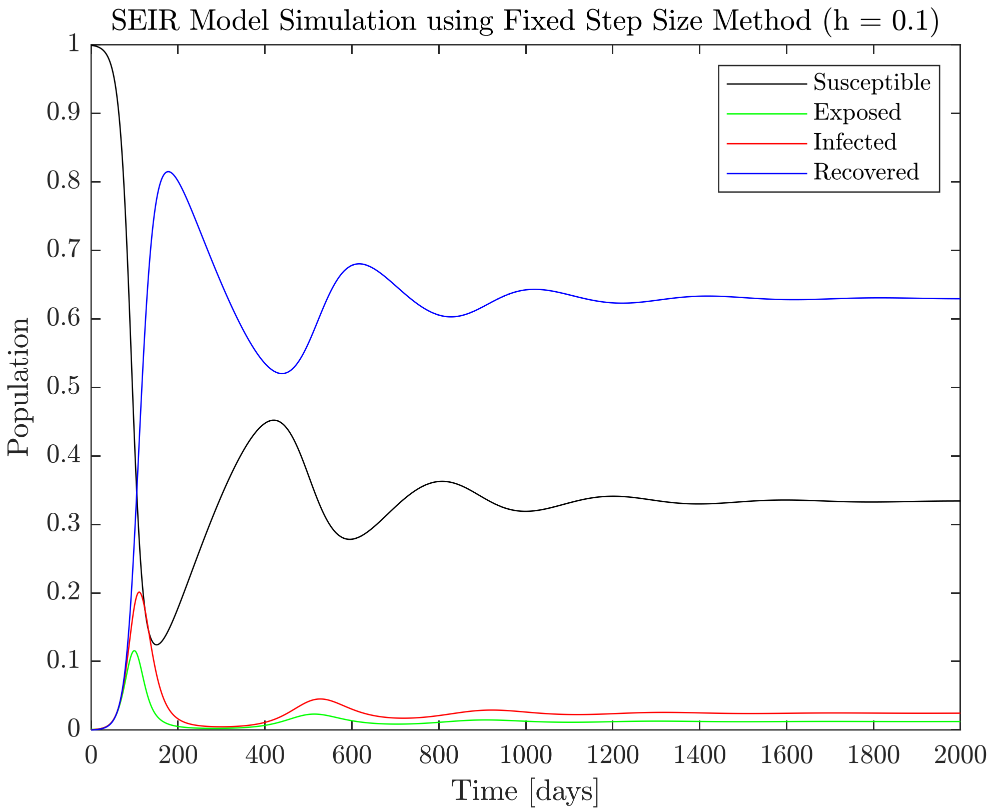

Fig 1.31 SEIRS Model with Fixed Step Size of h =0.1

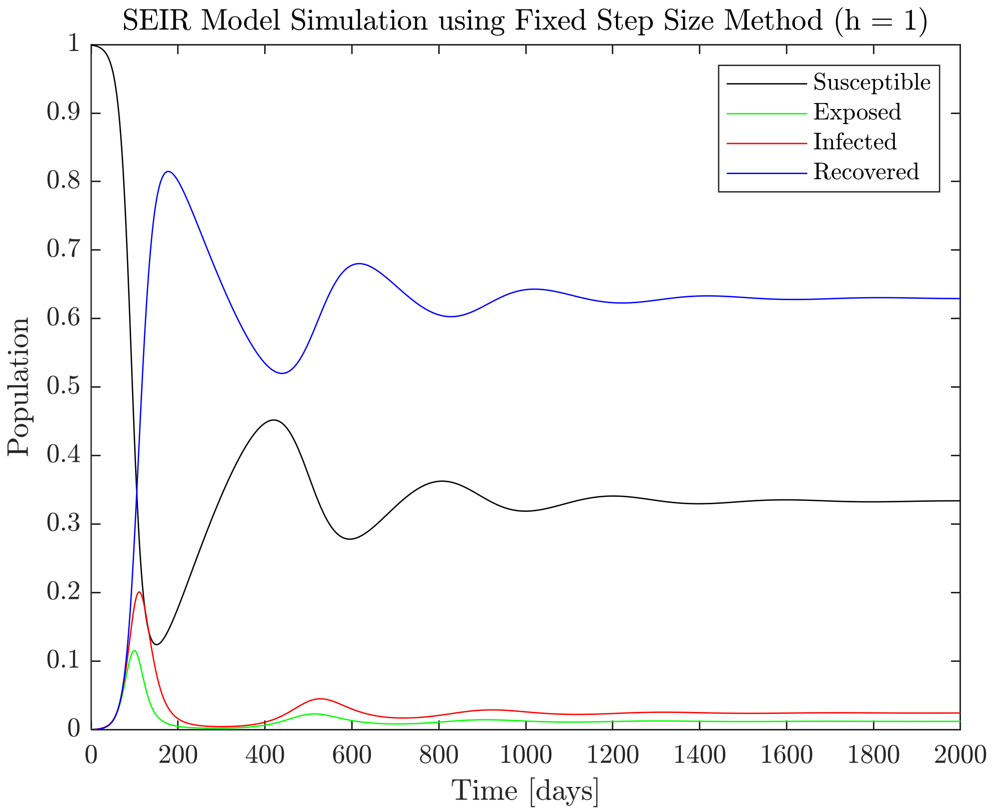

Fig 1.31 SEIRS Model with Fixed Step Size of h =1

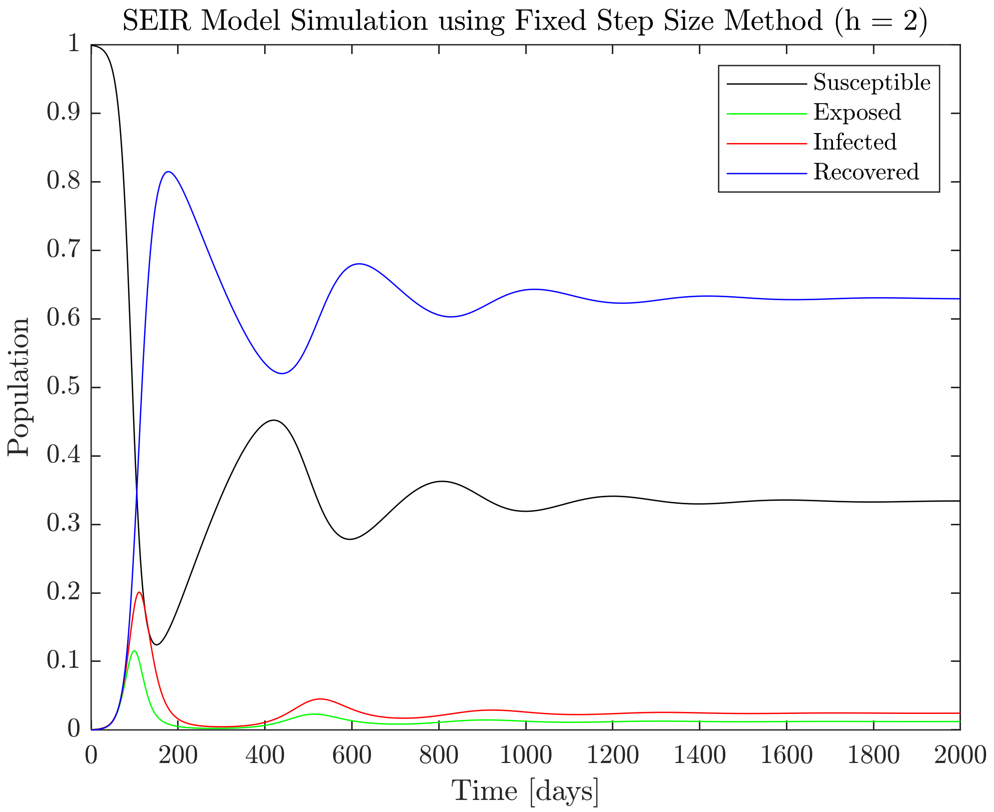

Fig 1.31 SEIRS Model with Fixed Step Size of h = 2

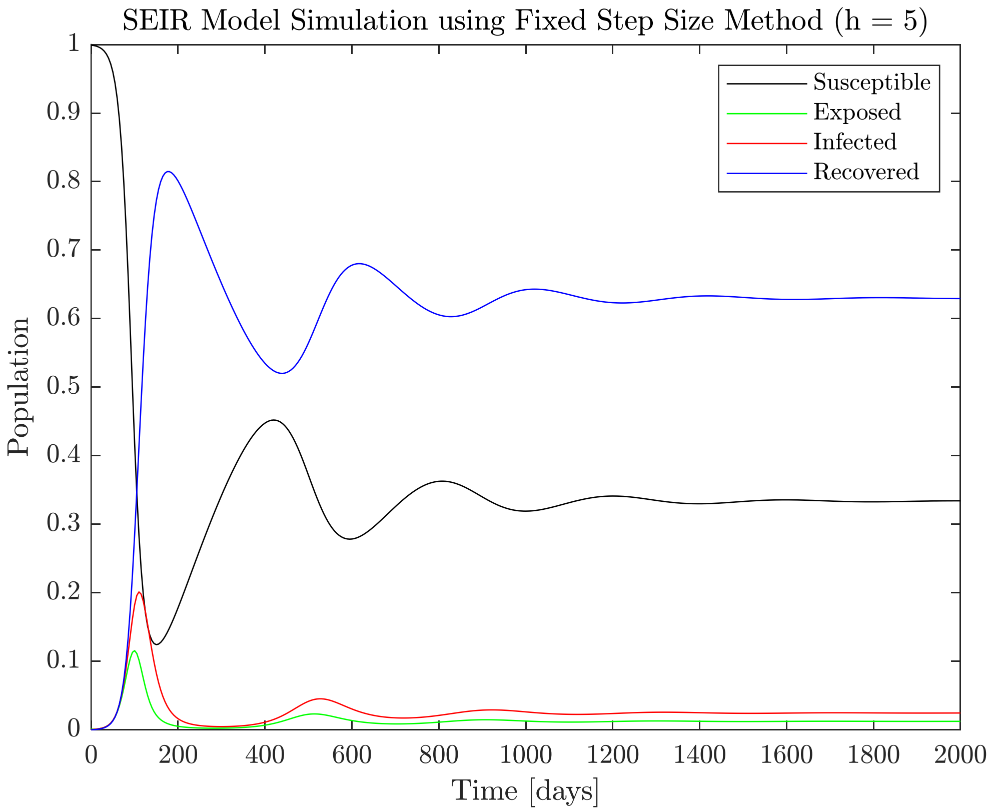

Fig 1.31 SEIRS Model with Fixed Step Size of h = 5

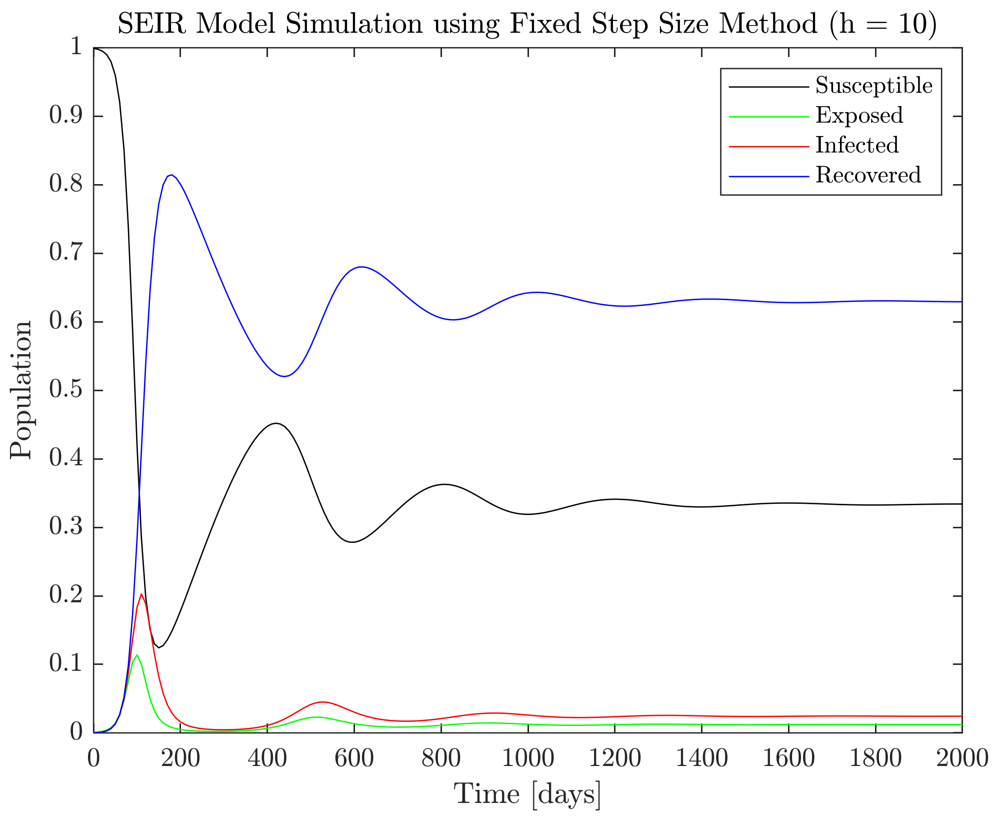

Fig 1.31 SEIRS Model with Fixed Step Size of h = 10

## % Adaptive Size Code for Runge Kutta %

```
clear all;
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Runge-Kutta method with adaptive step sizes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R0 = 3;
Gamma = 1/14;
Sigma = 1/7;
Omega = 1/365;
Mue = 1/(76*365);
Alpha = 0;
Beta = R0 * Gamma;

t0 = 0; % initial time
tf = 2000; % final time
h = 0.5; % initial step size

% Tolerances and parameters for adaptive step size control
atol = 1e-6;
rtol = 1e-3;
fac1 = 5;
```

```
fac0 = 0.2;
beta = 0.9;
hmin = 0.1;
MAX_ITER = 10000;

% Initial condition of ODE's
S0 = 0.999;
E0 = 0.001;
I0 = 0.000;
R0 = 0.000;

y = [S0;E0;I0;R0];
t = t0;
time = t;
result = y;

ITER = 0;

tic; % Start timing adaptive step size method
while t < tf && ITER < MAX_ITER
    k1 = ODESystem(t, y, Alpha, Beta, Gamma, Sigma, Mue, Omega);
    k2 = ODESystem(t + h/2, y + h*k1/2, Alpha, Beta, Gamma, Sigma, Mue, Omega);
    k3 = ODESystem(t + h/2, y + h*k2/2, Alpha, Beta, Gamma, Sigma, Mue, Omega);
    k4 = ODESystem(t + h, y + h*k3, Alpha, Beta, Gamma, Sigma, Mue, Omega);
    y_new = y + h/6*(k1 + 2*k2 + 2*k3 + k4);

    % Error estimation
    eta = (k1 - 2*k2 + 2*k3 - k4) * h/6;
    sigma = norm(eta ./ (atol + rtol * abs(y_new)), 2) / sqrt(length(y_new));

    if sigma <= 1
        t = t + h;
        y = y_new;
        time = [time t];
        result = [result y];

        if t >= tf
            break;
        end
    end

    % Update step size
    h = h * min(fac1, max(fac0, beta * sigma^(-1/5)));
    if h < hmin
        disp(['Warning: Step size below minimum threshold at iteration ', num2str(ITER), ', t = ',
num2str(t)]);
        h = hmin; % Set h to minimum threshold instead of error
    end

    ITER = ITER + 1;
end
adaptive_time = toc; % End timing adaptive step size method
S = result(1,:);
E = result(2,:);
I = result(3,:);
R = result(4,:);
```

```
disp(['Adaptive Step Size Method - Execution Time: ', num2str(adaptive_time), ' seconds']);
disp(['Final values: S = ', num2str(S(end)), ', E = ', num2str(E(end)), ', I = ', num2str(I(end)),
', R = ', num2str(R(end))]);

figure;
plot(time, S, 'k', 'DisplayName', 'Susceptible');
hold on;
plot(time, E, 'g', 'DisplayName', 'Exposed');
plot(time, I, 'r', 'DisplayName', 'Infected');
plot(time, R, 'b', 'DisplayName', 'Recovered');
xlabel('Time [days]');
ylabel('Population');
legend('show');
title('SEIR Model Simulation using Adaptive Runge-Kutta 4th Stage Method');
hold off;

function dydt = ODESystem(time, y, Alpha, Beta, Gamma, Sigma, Mue, Omega)
    S = y(1);
    E = y(2);
    I = y(3);
    R = y(4);
    dydt(1,1) = (Mue * (S + E + I + R) - ((Beta * I * S) / (S + E + I + R)) + Omega * R - Mue *
S);
    dydt(2,1) = ((Beta * I * S) / (S + E + I + R) - Sigma * E - Mue * E);
    dydt(3,1) = (Sigma * E - Gamma * I - (Mue + Alpha) * I);
    dydt(4,1) = (Gamma * I - Omega * R - Mue * R);
end
```

```
Output:
Adaptive Step Size Method - Execution Time: 0.029543 seconds
Final values: S = 0.33408, E = 0.012239, I = 0.024427, R = 0.62926
```

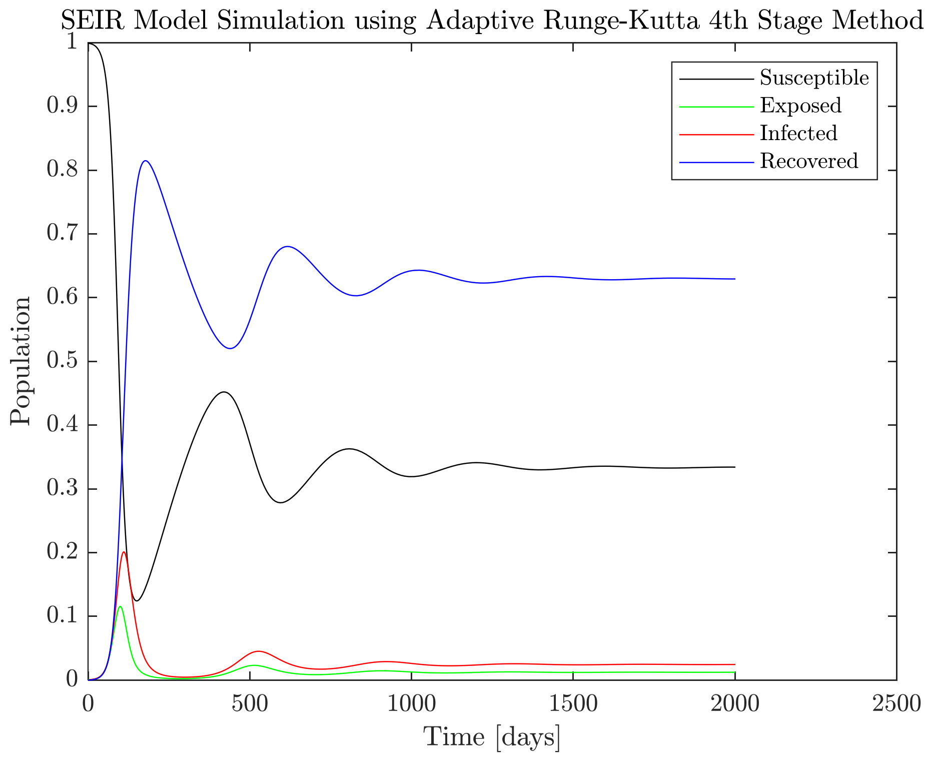

Fig 1.32 SEIRS Model with Adaptive Step Size

Comments: In comparing the computation time between the fixed step size method (tictac_fixed) and the adaptive step size method (AdaptiveSize), it's clear that the choice of step size (h) in the fixed method greatly influences the execution time.

For the fixed step size method:

- When $h = 0.01$, the execution time is 1.0641 seconds.
- As $h$ increases, the computation time decreases significantly. For instance, with $h = 0.1$, the time drops to 0.10142 seconds.
- Further increasing $h$ to 1 result in an execution time of 0.010201 seconds.
- With $h = 2$, the time is reduced to 0.0047649 seconds.
- For $h = 5$, the time further decreases to 0.0022873 seconds.
- At $h = 10$, the execution time is the lowest at 0.0010131 seconds.

For the adaptive step size method: The execution time is 0.0256 seconds.

When comparing these two methods, the adaptive step size method (0.0256 seconds) is generally more efficient than the fixed step size method at finer resolutions (e.g. $h = 0.01$, 1.0641 seconds) but less efficient compared to the coarser resolutions of the fixed method (e.g. $h = 10$, 0.0010131 seconds).

However, the adaptive method provides a balance between computational efficiency and accuracy without the need to manually adjust the step size. This adaptability ensures stability and precision, making it particularly advantageous in scenarios where the appropriate step size is not known a priori or where the dynamics of the system require varying step sizes for accurate results.

5) Plot the step sizes you observe when applying the method in (3).

To plot the step sizes, I have modified the code to include the step sizes in the subplot as shown below:

```
clear all;
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Runge-Kutta method with adaptive step sizes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R0 = 3;
Gamma = 1/14;
Sigma = 1/7;
Omega = 1/365;
Mue = 1/(76*365);
Alpha = 0;
Beta = R0 * Gamma;

t0 = 0; % initial time
tf = 2000; % final time
h = 0.5; % initial step size

% Tolerances and parameters for adaptive step size control
atol = 1e-6;
rtol = 1e-3;
fac1 = 5;
fac0 = 0.2;
beta = 0.9;
hmin = 0.1;
MAX_ITER = 10000;
```

```
% Initial condition of ODE's
S0 = 0.999;
E0 = 0.001;
I0 = 0.000;
R0 = 0.000;

y = [S0;E0;I0;R0];
t = t0;
time = t;
result = y;
step_sizes = h;

ITER = 0;

tic; % Start timing adaptive step size method
while t < tf && ITER < MAX_ITER
    k1 = ODESystem(t, y, Alpha, Beta, Gamma, Sigma, Mue, Omega);
    k2 = ODESystem(t + h/2, y + h*k1/2, Alpha, Beta, Gamma, Sigma, Mue, Omega);
    k3 = ODESystem(t + h/2, y + h*k2/2, Alpha, Beta, Gamma, Sigma, Mue, Omega);
    k4 = ODESystem(t + h, y + h*k3, Alpha, Beta, Gamma, Sigma, Mue, Omega);
    y_new = y + h/6*(k1 + 2*k2 + 2*k3 + k4);

    % Error estimation
    eta = (k1 - 2*k2 + 2*k3 - k4) * h/6;
    sigma = norm(eta ./ (atol + rtol * abs(y_new)), 2) / sqrt(length(y_new));

    if sigma <= 1
        t = t + h;
        y = y_new;
        time = [time t];
        result = [result y];
        step_sizes = [step_sizes h];

        if t >= tf
            break;
        end
    end

    % Update step size
    h = h * min(fac1, max(fac0, beta * sigma^(-1/5)));
    if h < hmin
        disp(['Warning: Step size below minimum threshold at iteration ', num2str(ITER), ', t = ',
num2str(t)]);
        h = hmin; % Set h to minimum threshold instead of error
    end

    ITER = ITER + 1;
end
adaptive_time = toc; % End timing adaptive step size method
S = result(1,:);
E = result(2,:);
I = result(3,:);
R = result(4,:);

disp(['Adaptive Step Size Method - Execution Time: ', num2str(adaptive_time), ' seconds']);
disp(['Final values: S = ', num2str(S(end)), ', E = ', num2str(E(end)), ', I = ', num2str(I(end)),
', R = ', num2str(R(end))]);
```

```
figure;
subplot(2,1,1);
plot(time, S, 'k', 'DisplayName', 'Susceptible');
hold on;
plot(time, E, 'g', 'DisplayName', 'Exposed');
plot(time, I, 'r', 'DisplayName', 'Infected');
plot(time, R, 'b', 'DisplayName', 'Recovered');
xlabel('Time [days]');
ylabel('Population');
legend('show');
title('SEIR Model Simulation using Adaptive Runge-Kutta 4th Stage Method');
hold off;

subplot(2,1,2);
plot(time(1:end), step_sizes, 'm', 'DisplayName', 'Step Size');
xlabel('Time [days]');
ylabel('Step Size');
legend('show');
title('Step Sizes over Time');
hold off;

function dydt = ODESystem(time, y, Alpha, Beta, Gamma, Sigma, Mue, Omega)
    S = y(1);
    E = y(2);
    I = y(3);
    R = y(4);
    dydt = zeros(4,1);
    dydt(1) = Mue * (S + E + I + R) - (Beta * I * S / (S + E + I + R)) + Omega * R - Mue * S;
    dydt(2) = (Beta * I * S / (S + E + I + R)) - Sigma * E - Mue * E;
    dydt(3) = Sigma * E - Gamma * I - (Mue + Alpha) * I;
    dydt(4) = Gamma * I - Omega * R - Mue * R;
end
```

Output:

```
Adaptive Step Size Method - Execution Time: 0.023215 seconds
Final values: S = 0.33408, E = 0.012239, I = 0.024427, R = 0.62926
```

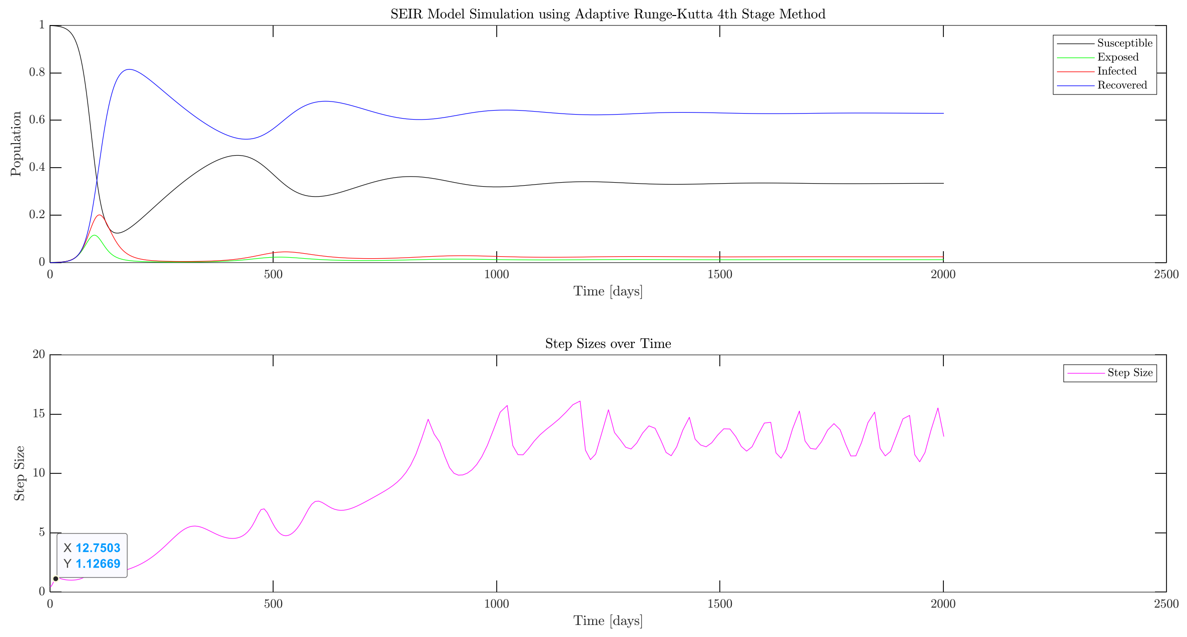

Fig 1.32 SEIRS Model with Adaptive Step Size Plots for each step size