MS6021 Scientific Computation

Assignment 3

Dara Corr - 22275193

October 29, 2022

Contents

1 Introduction

This assignment is split into two parts. The first part of the assignment is concerned with three explicit methods of solving ODEs numerically: The Explicit Euler Method, The Predictor Corrector Method and the 4th Order Runge-Kutta Method. We will look at how each of these methods performs in solving three different problems.

Part two of this Assigment involves looking at the SIR model for disease modelling and fitting an SIR model to recorded incidence data from an Influenza B outbreak.

2 Part 1: Investigating Explicit Methods for ODEs

We will be applying three Explicit Methods to Three different problems in this part of the assignment, The methods we will be using are the Explicit Euler Method. The Predictor Corrector Method and The 4th Order Runge-Kutta method.
We will be applying these three methods to three different problems, given as systems of ODEs in the form $\frac{d}{dt}y = f(y,t)$:

$$\begin{aligned} \frac{d}{dt}y(t) &= -y(t) - 5e^{-t}sin(5t) \\ y(0) &= 1 \end{aligned} \tag{1}$$

$$\frac{d^2}{dt^2}\theta(t) + sin(\theta(t)) = 0 \tag{2}$$

$$\begin{aligned} \frac{d}{dt}y_1(t) &= -y_2(t) - y_3(t) \\ \frac{d}{dt}y_2(t) &= y_1(t) + ay_2(t) \\ \frac{d}{dt}y_3(t) &= b + y_3(t)(y_1(t) - c) \end{aligned} \tag{3}$$

equation 2 can be rewritten as:

$$\begin{aligned} \frac{d}{dt}y_1(t) &= y_2(t) \\ \frac{d}{dt}y_2(t) &= -sin(y_1(t)) \end{aligned}$$

2.1 Outline of Explicit Methods used:

The explicit methods we used have the following form:
Explicit Euler Method

$$\begin{aligned} y^{(0)} &= y(0) \\ y^{(n+1)} &= y^{(n)} + hf(t^{(n)}, y^{(n)}) \\ \text{where } t^{(n)} &= nh, \text{ n = number of iterations} \end{aligned} \tag{4}$$

Predictor Corrector Method

$$
\begin{aligned}
y^{(0)} &= y(0) \\
y^{*} &= y^{(n)} + hf(t^{(n)}, y^{(n)}) \\
y^{(n+1)} &= y^{(n)} + 0.5h[f(t^{(n)}, y^{(n)}) + f(t^{(n+1)}, y^{*})] \\
&\text{where } t^{(n)} = nh
\end{aligned}
\tag{5}
$$

4th Order Runge-Kutta

$$
\begin{aligned}
y^{(n+1)} &= y^{(n)} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \\
t^{(n+1)} &= t^{(n)} + h \\
k_1 &= f(t^{(n)}, y^{(n)}) \\
k_2 &= f(t^{(n)} + \frac{h}{2}, y^{(n)} + h\frac{k_1}{2}) \\
k_3 &= f(t^{(n)} + \frac{h}{2}, y^{(n)} + h\frac{k_2}{2}) \\
k_4 &= f(t^{(n)} + h, y^{(n)} + hk_3)
\end{aligned}
\tag{6}
$$

I implemented these methods as Matlab functions which I used to solve the three problems we are given.

```
function [tsol,ysol] = ExplicitEuler(f,tspan,y0,N)

%finds numeric solution to dy/dt = f(t,y0) using Explicit Euler Method
%user provides function f, tspan, y0 and N

h = (tspan(end) - tspan(1))/N

y = zeros(numel(y0),N+1);

T = tspan(1):h:tspan(end);

y(:, 1) = y0;

for j = 1:N
    y(:, j+1) = y(:, j) + h.* f(T(j),y(:, j));
end

ysol = y
tsol = T

end
```

```
function [tsol,ysol] = PredictorCorrector(f,tspan,y0,N)

%finds numeric solution to dy/dt = f(t,y0) using Predictor Corrector method
%user provides function f, tspan, y0 and N

h = (tspan(end) - tspan(1))/N

y_pred = zeros(numel(y0),N+1);
```

```
y = zeros(numel(y0),N+1);

T = tspan(1):h:tspan(end);

y(:, 1) = y0;
ypred(:, 1) = y0;

for j = 1:N
    ypred(:, j+1) = y(:, j) + h*f(T(j),y(:, j));
    y(:, j+1) = y(:, j) + 1/2 * h * ( f(T(j),y(:, j)) + f(T(j+1),ypred(:, j+1)
    ));
end

ysol = y
tsol = T
end
```

```
function [tsol,ysol] = RK4(f,tspan,y0,N)

h = (tspan(end) - tspan(1))/N

y = zeros(numel(y0),N+1);

T = tspan(1):h:tspan(end);

y(:, 1) = y0;

for j = 1:N
    k1 = f(T(j),y(:, j))
    k2 = f(T(j)+h/2,y(:, j)+h*k1/2)
    k3 = f(T(j)+h/2,y(:, j)+h*k2/2)
    k4 = f(T(j)+h/2,y(:, j)+h*k3)
    y(:, j+1) = y(:, j) + 1/6 * (k1 + 2*k2 + 2*k3 + k4)*h;
end

ysol = y
tsol = T
end
```

2.2 Problem 1

Problem 1 is a system consisting of one ODE. This system has an analytical solution $y(t) = e^{-t}cos(5t)$ which we can use to find the errors of our computed solutions. We run our three methods for N = 600 and N = 1200 for problem 1 and compare with the analytical solution.

Below are some code snippets of how I called the functions and calculated the differences finding the Max absolute difference between the numerical solution and the analytical solution accross all values of time t (time is the same for both). And then I plotted all 4 solutions against t.

```
%problem1 (N=600/1200)
%analytic solution


N = 1200
y_analytic1 = @(t)exp(-t) .* cos(5*t)


%Explicit Euler Method
yprime = @(t,y)-y -5*exp(-t)*sin(5*t)

tspan = [0 3]; yzero = 1;

[t,y1] = ExplicitEuler(yprime,tspan,yzero,N)

max(abs(y_analytic1(t) - y1))
%%
%Problem1 (N=600/1200) Predictor-Corrector Method
yprime = @(t,y)-y -5*exp(-t)*sin(5*t)
y_analytic1 = @(t)exp(-t) .* cos(5*t)
tspan = [0 3]; yzero = 1;
[t,y2] = PredictorCorrector(yprime,tspan,yzero,N)

max(abs(y_analytic1(t) - y2))

%%
%Problem1 (N=600/1200) 4th Order Runge-Kutta Method
yprime = @(t,y)-y -5*exp(-t)*sin(5*t)
y_analytic1 = @(t)exp(-t) .* cos(5*t)
tspan = [0 3]; yzero = 1;
[t,y3] = RK4(yprime,tspan,yzero,N)

max(abs(y_analytic1(t) - y3))

%%



plot(t,y_analytic1(t))
hold on
plot(t,y1)
hold on
plot(t,y2)
hold on
plot(t,y3)

title('Problem 1 plots')
xlabel('t')
ylabel('y')
legend("Analytical Solution","Explicit Euler Method","Predictor Corrector
    Method","Runge-Kutta 4",'Location','southwest')
```

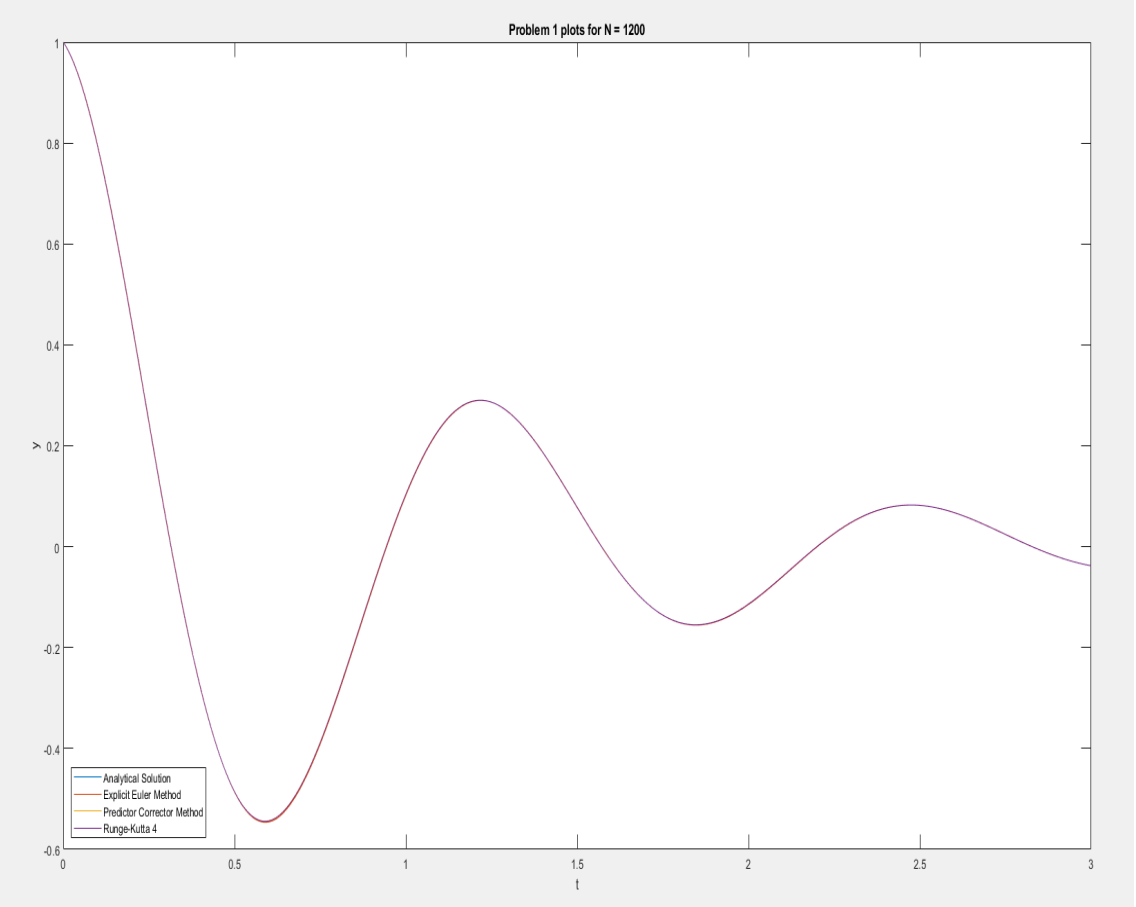
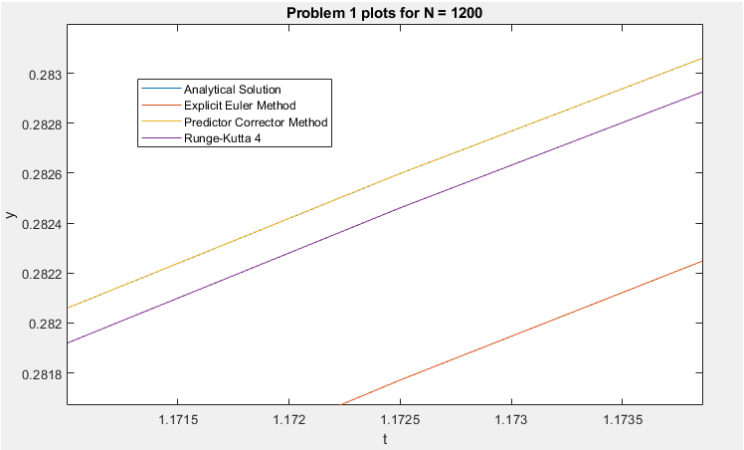
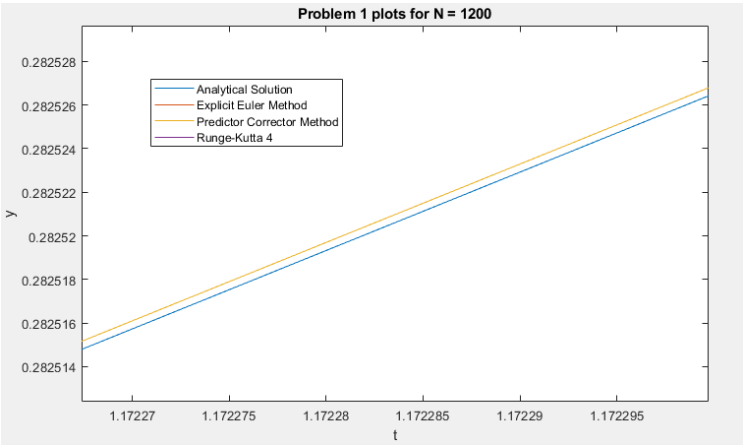


Figure 1: Plots of Solutions from the three methods and the analytical solution for Problem 1



(a)



(b)

Figure 2: (a) and (b) are zoomed in snippets of figure 1 showing how close the Numerical Solutions are in relation to the Analytical Solution of the ODE.

As we can see from the plots, all three numerical methods provide reasonably accurate solutions to the ODE for N = 1200 and the computation time was small for all three calculations. The three methods have similar errors here with Predictor-Corrector performing the best, followed by Runge-Kutta 4 and Explicit Euler Method. Next, we will examine the three methods for the other two problems so we can make better comparisons between the efficiency and accuracy of these explicit methods.

2.3 Problem 2 and 3

Problems 2 and 3 consist of systems of ODEs with initial conditions [y10; y20] = [1;1] and [y10 ; y20 ; y30 ;] = [1 ; 1 ; 1] respectively. There are no analytical Solutions provided for problems 2 and 3. Therefore we need to create a reference solution with ODE45() in MATLAB which is a built in ODE solver in MATLAB which uses a version of the explicit 4th order Runge-Kutta Method to solve non-stiff ODEs. We specify very small tolerances in ODE45 to get a very precise reference solution to make comparisons with. This is done by

setting Absolute tolerances as 1e-20 and setting Relative tolerances as 1e-13 in the options argument when calling ODE45. Once we have our reference solution then obtaining the errors is the same as before using $max|y^{(n)} - y(t^{(n)})|$ to find errors where $y^{(n)}$ is the reference solution here instead of the analytical solution we had for problem 1.
Here is a snippet of how I created the reference solution for problem 2 (the same procedure is repeated in problem 3 and the procedure for generating the errors is the same with reference solution used instead of analytical solution)

```
%problem 2: (N=500/1000)
tspan = [0 10];
yzero = [1; 1];

N=500
h = (tspan(end) - tspan(1))/N
tspan2 = tspan(1):h:tspan(end)


%Reference Solution
options = odeset('AbsTol',1e-20,'RelTol',1e-13);
[T2,Y2] = ode45(@pend,tspan2,yzero, options);
```

```
%Explicit Euler solution for Problem 2:
tspan = [0 10];
yzero = [1; 1];
[t,y4] = ExplicitEuler(@pend,tspan,yzero,N);

plot(t,y4(1,:))
%plot(t,y(2,:))
%plot(y(1,:),y(2,:)) %note plot is not continuous


y4 = y4';
max(abs(Y2 - y4))
```

When plotting the y_2 against y_1 from the results of problem 2 using the Explicit Euler Method I notice that there is a very obvious discontinuity in the plot. The 2nd order ODE in problem 2 is seen often in physics and engineering as the Simple Harmonic Equation $\frac{d^2x}{dt^2} + \omega^2 x = 0$ which models systems like planetary motions, motion of pendula and vibrations of systems. This equation here does not account for damping so there is no energy loss, so we would expect a continuous curve when plotting y_2 against y_1 here. Because of this discontinuity, I would consider the errors associated with the Explicit Euler method to be too large if you are modelling a physical system that you want to be precise.

This same issue is not encountered when using Predictor Corrector or 4th Order Runge-Kutta Methods which produce much smaller errors, especially Runge-Kutta 4 with errors of about e-08 - e-10 in magnitude. For problems 2 and 3, Runge-Kutta performs very well and Predictor Corrector performs quite well also with relatively small errors (Whether a method is a good method to use or not depends on the precision needed in the application it is being used for and whether short computation times are needed or not).

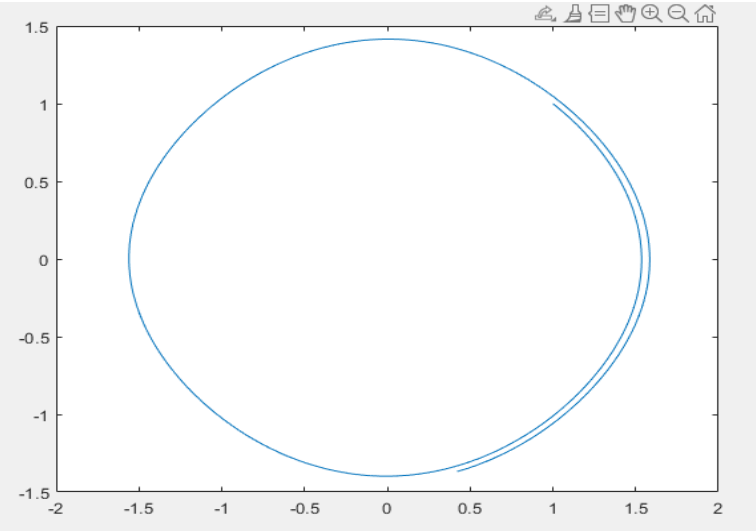


Figure 3: Plots of y_2 against y_1 for problem 2 using Explicit Euler Method for N = 1000

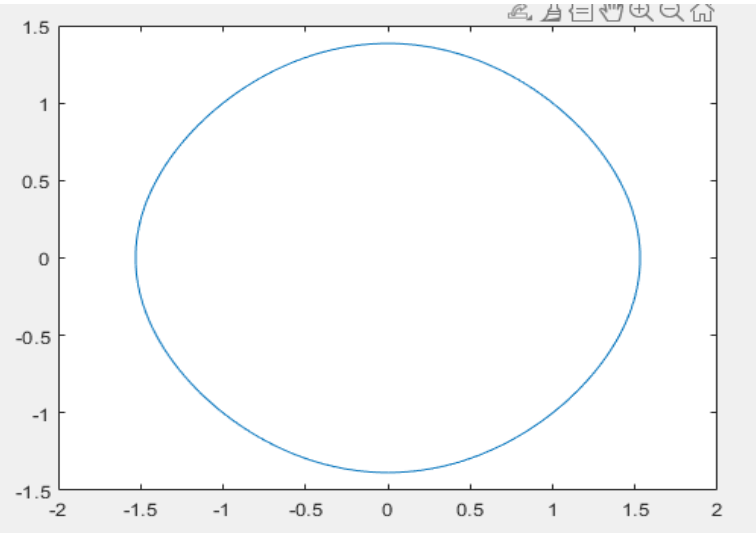


Figure 4: Plots of y_2 against y_1 for problem 2 using Predictor Corrector Method for N=1000

For more complicated systems like The Roessler system, it is not apparent if the method used to solve the ODE is precise enough by just looking at the plots and we must look at the errors. For Problem 3, like problem 2, there is a large error value associated with Explicit Euler Method, a smaller error for Predictor Corrector and a very small error for 4th Order Runge-Kutta Method.

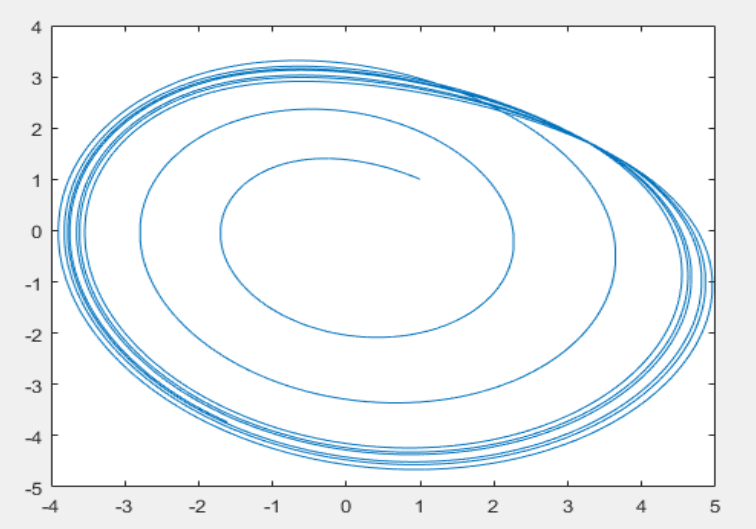


Figure 5: Plots of y_2 against y_1 for problem 3 using Predictor Corrector Method for N = 40000

	Problem (1) N = 600	Problem (1) N = 1200	Problem (2) N = 500	Problem (2) N = 1000	Problem (3) N = 20000	Problem (3) N = 40000
Explicit Euler	7.797e-03	3.900e-03	2.411e-01	1.148e-01	4.946e-01	2.451e-01
Predictor Corrector	6.497e-05	1.624e-05	2.566e-04	6.622e-05	4.032e-04	1.007e-04
4th Order Runge-Kutta	1.312e-03	6.576e-04	9.736e-09	6.196e-10	1.410e-10	8.957e-12

2.4 Conclusions

- There is a trade off between computation time and precision of the results when using explicit methods to solve ODEs numerically.
- While the Predictor Corrector method is the most precise for problem 1, with the smallest associated error, The 4th Order Runge-Kutta method is the best explicit method overall in terms of minimizing errors as the errors reduce drastically by multiple orders of magnitude when the model is more complex and has a higher number of iterations as we can see from the table. The other two explicit methods do not have their errors decreasing as fast as 4th Order Runge-Kutta does.
- The Explicit Euler Method generally produces solutions with relatively high error values and I think it is not a good method to be used in any practical applications. Predictor Corrector and Runge-kutta 4 methods produce consistently accurate solutions with low error values.
- One disadvantage of Runge-Kutta is that it is a slower method than the other two methods so it may increase computation times for solving very complex systems or real time systems where minimising computation times is important.

3 Part 2: Fitting the parameters of an SIR model to influenza data using Least Squares

In part 2 we look at the SIR model of modelling disease spread and we create an SIR model to model data of recorded infections from B-influenza and by fitting our model and finding the solution that minimizes the sum of squares difference between the model and the data we can estimate parameters of R number and t_0. Knowing R-number and t_0 is useful so we can source where an outbreak may have started and estimate further spread of a disease in a population.

3.1 Q1 The SIR model of Disease Spread

The SIR model of Disease Spread splits a population that we are modelling into three Cohorts:

- S(t) = The population of susceptible (healthy) individuals
- I(t) = The population of infected Individuals
- R(t) = The population of "recovered" individuals who were infected and now have immunity to the virus

We want to model the change in population of these cohorts over time, so we introduce the following set of equations:

$$\begin{aligned} \frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= +\beta SI - \gamma I \\ \frac{dR}{dt} &= +\gamma I \end{aligned} \tag{7}$$

Where

$$S(t) + I(t) + R(t) = 1 \tag{8}$$

$\frac{1}{\gamma}$ is the average recovery period of the Disease and $\beta = \gamma \cdot R_{\text{number}}$

The initial conditions for this problem is that there is one infected individual at time t_0 i.e.

$$\begin{aligned} I(t_0) &= \frac{1}{\text{population}} \\ S(t_0) &= 1 - I(t_0) \\ R(t_0) &= 0 \end{aligned} \tag{9}$$

We make an initial model by inputting the following parameters into the SIR model we just defined:

$$\frac{1}{\gamma} = 3 \text{ days}$$

$$\text{Population} = 5.2\text{e+}7$$

$$t_0 = 250 \text{ i.e. } 250\text{th day of the year}$$

Because of Equation (8), we only need to calculate $S(t)$ and $I(t)$ (from the first two equations in (7)) using ODE45() and we can compute R(t) as

$$R(t) = 1 - S(t) - I(t)$$

For this model tspan = [t0, 7*(46:72)]. We define our tspan this way because we will be comparing this model to data from weeks 47-72 in the data (our plots t axis is measured in days).

Using ODE45 with Absolute and Relative tolerances of 1e-20 and 1e-13, we can now then calculate S,I and R and plot them all against time.

```
function yprime = SIR(t,y,Beta,gamma)

%SIR model

yprime = [-Beta*y(1)*y(2); +Beta*y(1)*y(2) - gamma*y(2)];
```

```
%Q1
%Model SIR model
%measure time in units of [days]

%told gamma^-1 = 3 days
gamma = 1/3;
t0 = 250;
tspan = [t0, 7*(46:72)];
R_num = 1.30;
population = 5.2e+7;

yzero = [1-1/population,1/population];

%Beta = gamma * R
Beta = gamma*R_num;

%y = [S, I]
%yprime = [-Beta*S*I, +Beta*S*I - gamma*I]
%yprime = [-Beta*y(1)*y(2), +Beta*y(1)*y(2) - gamma*y(2)]

options = odeset('AbsTol',1e-20,'RelTol',1e-13);

[t, y] = ode45(@SIR,tspan,yzero,options,Beta,gamma)


S = y(:,1)
I = y(:,2)

plot(t,S,'b'); %plot of S(t) against t
hold on
```

```
plot(t,I,'r'); %plot of I(t) against t
hold on
R = 1 - S - I;

plot(t,R); %plot of R(t) against t
title('Plotting S(t),I(t) and R(t) against t')
xlabel('t (days)')
ylabel('Proportion of Population')
legend("Susceptible population S(t)","Infected Population I(t)","Recovered
    Population with immunity R(t)",'Location','southwest')
hold off
```

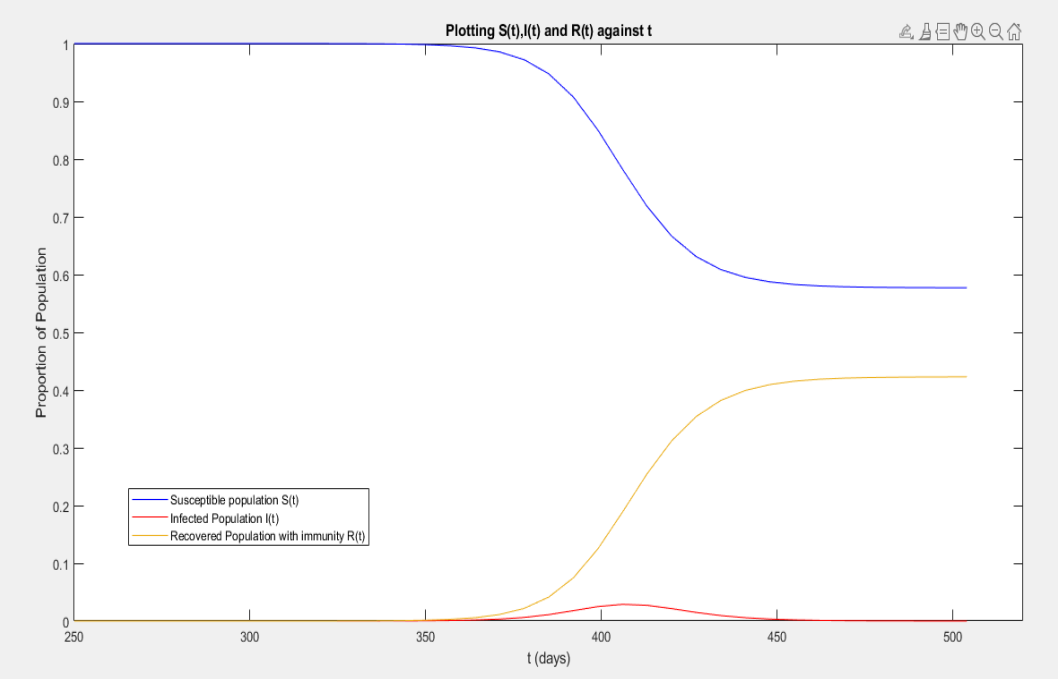


Figure 6: Plots of the initial SIR model for B influenza in Q1

3.2 Q2 Comparing the numerical results obtained in Q1 with the data using least squares metric

Next we plot the data of incidences of B influenza infections in the US Midwest in 2007 from the dataset. We are only interested in the week numbers and weekly incidences of B influenza from this Data. We discard all data rows for weeks < 47 since the incidence numbers are close to 0 for these weeks. Then we can plot the measured incidences against the days (weeks $\times$ 7) on the x axis.

```
%discard weeks <47 where incidence may = 0
incidence_data(1:7,:) = [];

incidence_data = incidence_data'
T = 7.* incidence_data(1,:);
incidences = incidence_data(2,:);

plot(T,incidences,'-o') %plots incidences for weeks 47 to 72 inclusive
title("Incidences of B influenza over time from Data")
xlabel('t (days)')
ylabel('number of incidences')

%calculate area from data
%(approx) area under curve in this case is just sum of incidences .* 7

area_d = 7.*sum(incidences)
%%
```

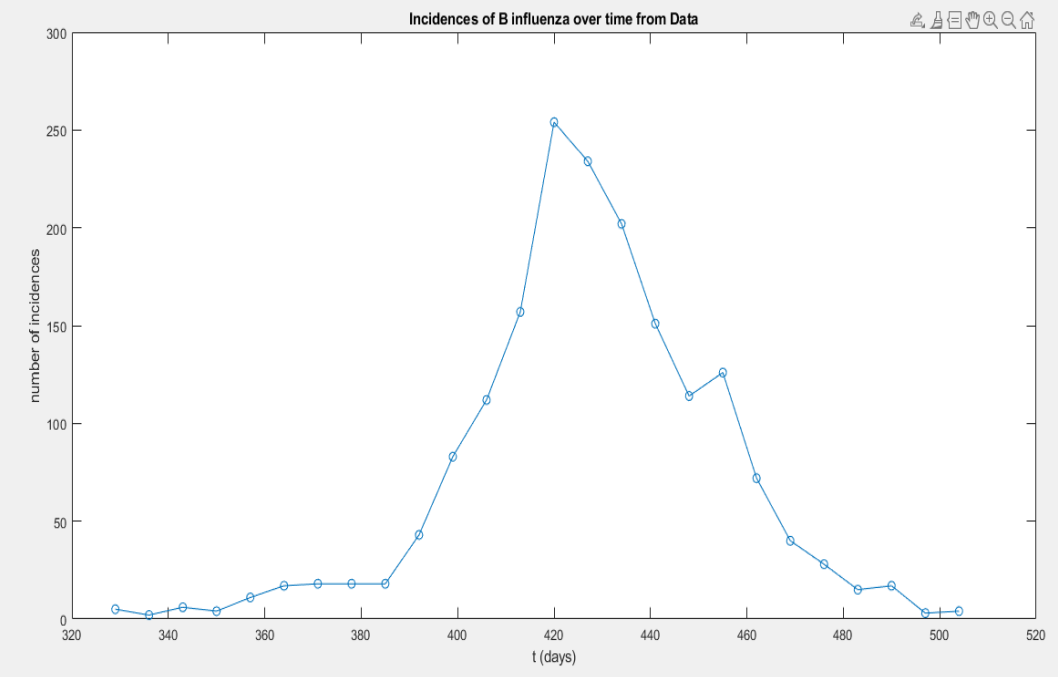


Figure 7: Plotting the incidence data of B influenza infections from the Data

We must make some changes to our numerical data from Question 1 to make it suitable for comparison with the recorded incidence data.

- The recorded data ignores recovering patients and measures the decrease in the Susceptible Population $S(t)$ for each corresponding week. The recorded incidence corresponds to -diff(S) so because of this we set incidence_ modelled = -diff(S)
- Because only a fraction of a the population was tested, we need to normalize the numerical data. To do this we calculate the area of the incidence data $= 7 \cdot Sum_{incidences}$ and the area of the numerical data $= 7 \cdot Sum_{num}$ and the normalization constant is then $= \frac{Sum_{inc}}{Sum_{num}}$
- The normalized numerical data is given by

$$\text{incidence\_modelled} = \left(\frac{Sum_{inc}}{Sum_{num}}\right) \cdot \text{incidence\_modelled}$$

Now the Numerical Data is prepared for comparison with the Recorded Incidences Data, we can plot both of them on the same plot and find the Sum of the Squared differences between the numerical and recorded data to find the least squares metric (important for Question 3 and 4 where we fit parameters in the model):

$$\text{Sum of Squares} = \sum_{i=1}^{n} (\text{incidences}_i - \text{incidences\_modelled}_i)^2$$

```
%discard initial values at initial time t0
S = y(:,1);
I = y(:,2);
R = 1 - S - I;

S(1) = []; I(1) = []; R(1) = [];


%incidence modelled = -diff(S)
incidence_modelled = -diff(S);

%area_numerical
area_n = 7 .* sum(incidence_modelled);

%normalise incidence modelled
incidence_modelled = incidence_modelled .* (area_d/area_n);
plot(T,incidences,'-o');
hold on
plot(T,incidence_modelled);

title('Plots of recorded incidences and modelled incidences of B influenza')
xlabel('t (days)')
ylabel('number of incidences')
legend("Recorded Incidences","Modelled Incidences",'Location','southwest')

```

```
26
27 hold off
28
29 %check areas are equal
30 area1 = 7 .* sum(incidence_modelled)
31 area2 = 7 .* sum(incidences)
32
33 incidence_modelled = incidence_modelled';
34
35 SQUARES = (incidences - incidence_modelled).^2;
36 SumofSQUARES = sum(SQUARES)
37 MSE = (1/numel(incidences)) .* SumofSQUARES %MSE = 5500.8
```

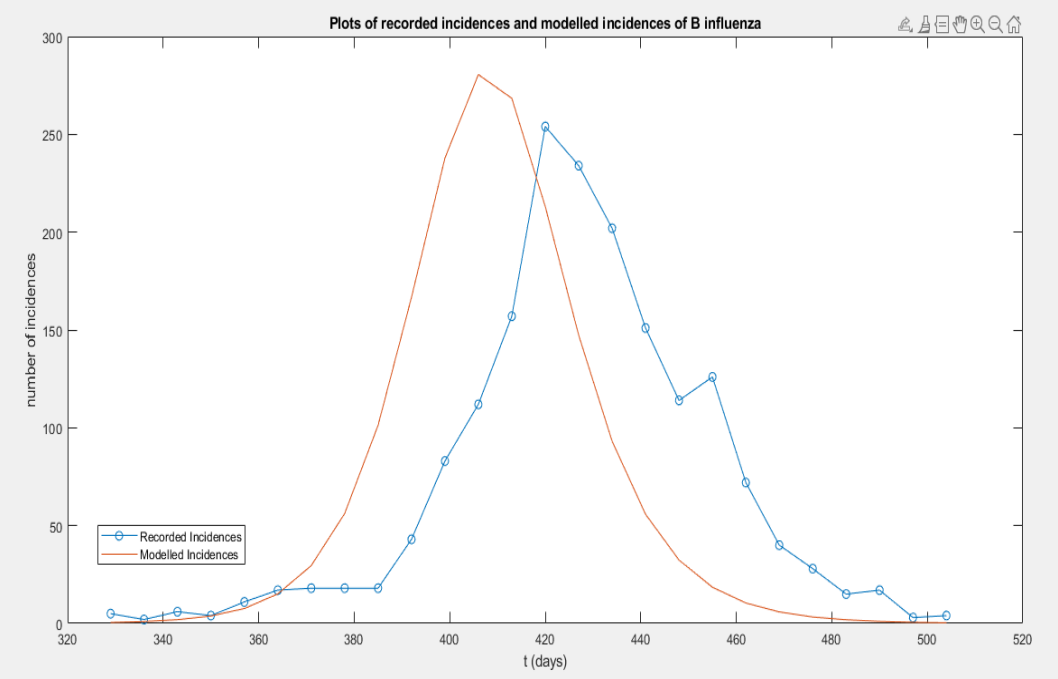


Figure 8: Plotting the Recorded Incidence Data and Modelled Incidences on the same plot

The Sum of Squares calculated here is = 1.4302e+05.

3.3 Q3 Fitting parameter R number in the model with $t_0 = 250$ fixed

For Q3 we try to model the R number in our SIR model to the data keeping $t_0 = 250$ fixed. To do this I created a column vector of R numbers to calculate:

$$\mathrm{R}_{\mathrm{number}} = [1.21{:}0.001{:}1.24]'$$

and I set k = the length of this vector.
Then I modified my code for f in .m file SIR2.m and modified my initial conditions for the problem so that the solution y, will store solutions $S(t)^{(1:k)}$ in columns 1:k and solutions

for $I(t)^{(1:k)}$ in columns k+1:2k. Then the code runs as it did in Q2 except it computes the Sum of Squares for each R number and adds it to a vector. Then the index of the least sum of squares in the vector corresponds to the index of the R number value which fits the incidence data the best.

```
function yprime = SIR2(t,y,Beta,gamma,k)

%SIR model adapted for question 3 and 4

yprime = [ -Beta.*[y(1:k)].*[y(k+1:2*k)] ; +Beta .* [y(1:k)].*[y(k+1:2*k)] -
    gamma.*[y(k+1:2*k)] ];

end
```

```
t0 = 250;
tspan = [t0, 7*(46:72)];
population = 5.2e+7;

R_num = [1.21:0.001:1.24]' %smaller size for test
k = length(R_num)

yzero = [(1-1/population).*ones(k,1);(1/population).*ones(k,1)];

%Beta = gamma * R
Beta = gamma.*R_num;

%y = [S, I]
options = odeset('AbsTol',1e-20,'RelTol',1e-13);

[t, y] = ode45(@SIR2,tspan,yzero,options,Beta,gamma,k)

S = y(:,1:k); %y(:,1:k)
%I = y(:,k+1:2*k)

S(1,:) = [];

for i = [1:k]
incidence_modelled_new(:,i) = -diff(S(:,i));

%area_numerical
area_n = 7 .* sum(incidence_modelled_new(:,i));
%normalise incidence modelled
incidence_modelled_new(:,i) = incidence_modelled_new(:,i) .* (area_d/area_n);
end

SumOfSquares_vec = [];

for j = 1:k
SQUARES = (incidences - incidence_modelled_new(:,j)).^2
SumofSQUARES = sum(SQUARES);
SumOfSquares_vec(end+1) = SumofSQUARES;
end

[leastsquare, index] = min(SumOfSquares_vec);
```

```
index

R_num(index)

%bestR_num = R_num(index)
plot(T,incidence_modelled_new(:,index),'r-')
hold on
plot(T',incidences,'b-o')

title('Plots of recorded incidences and modelled incidences of B influenza
    with t_0 fixed and R number optimised')
xlabel('t (days)')
ylabel('number of incidences')
legend("Modelled Incidences with t_0 fixed and R number fit to the data","
    Recorded Incidences",'Location','southwest')
hold off
```

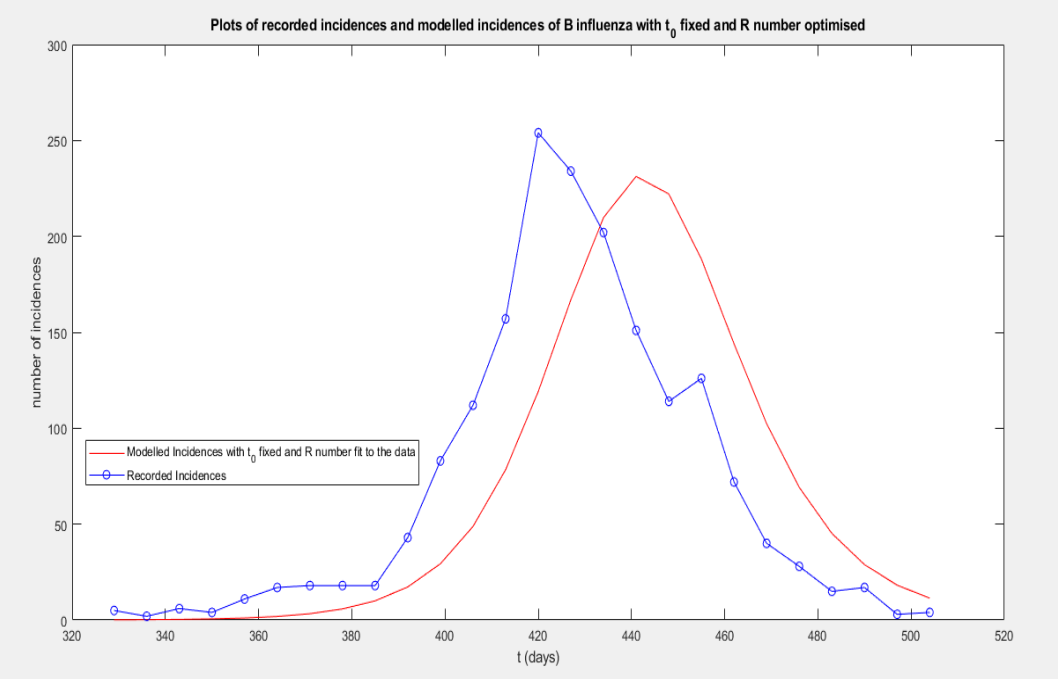


Figure 9: Plot of least square solution from fitting R number to the model for t0 fixed

The R number found from the least square solution in Q3 is $R_{\text{number}} = 1.24$

3.4 Q4 Fitting parameters R number and t_0 to SIR model

We see from the plot in the last section that fitting R_{number} with t_0 fixed yields a model that does not fit the data as well as we would like. This can be improved by fitting for parameters t_0 and R_{number} in the model. I achieved this by running my code for Q3 inside a loop in t0 = [200:250] and then computed a Matrix of Sum of Squares values with row index corresponding to t_0 index and column index corresponding to R_{number} index. Then by finding the index of the Least Square value inside this matrix we have found the index for the R_{number} and t_0 values that best fit the data we are modelling using the Least Squares metric.

```
%next model for R number and t0

gamma = 1/3;
population = 5.2e+7;
R_num = [1.21:0.001:1.24]'
k = length(R_num);
yzero = [(1-1/population).*ones(k,1);(1/population).*ones(k,1)];
Beta = gamma.*R_num;
options = odeset('AbsTol',1e-20,'RelTol',1e-13);



SumOfSquares_Matrix2 = []
t0 = [200:250]

for t0 = [200:250]

tspan = [t0, 7*(46:72)];

[t, y] = ode45(@SIR2,tspan,yzero,options,Beta,gamma,k);

S = y(:,1:k); %y(:,1:k)
%I = y(:,k+1:2*k)
S(1,:) = [];

for i = [1:k]

incidence_modelled_new(:,i) = -diff(S(:,i));

%area_numerical
area_n = 7 .* sum(incidence_modelled_new(:,i));
%normalise incidence modelled
incidence_modelled_new(:,i) = incidence_modelled_new(:,i) .*(area_d/area_n);

end

SumOfSquares_vec2 = [];


for j = 1:k
SQUARES2 = (incidences - incidence_modelled_new(:,j)).^2;
SumofSQUARES2 = sum(SQUARES2);
```

```
SumOfSquares_vec2(end+1) = SumofSQUARES2;
end

SumOfSquares_Matrix2 = vertcat(SumOfSquares_Matrix2,SumOfSquares_vec2)
end

%%
%minimum = min(min(SumofSquares_Matrix))
[t0_index,R_num_index] = find(SumOfSquares_Matrix2 == min(SumOfSquares_Matrix2(:)))


R_num_optimum = R_num(R_num_index) %R_num = 1.2390

t0 = [200:250];
t0_optimum = t0(t0_index) %t0 = 234
```

```
%and t0

tspan = [t0_optimum, 7*(46:72)];
R_num = R_num_optimum;
Beta = gamma*R_num;
yzero = [1-1/population,1/population];
gamma = 1/3;


[t, y] = ode45(@SIR,tspan,yzero,options,Beta,gamma);

S = y(:,1); %y(:,1:k)
%I = y(:,k+1:2*k)
S(1,:) = [];

InfluenzaModel = -diff(S);

%area_numerical
area_n = 7 .* sum(InfluenzaModel);
%normalise incidence modelled
InfluenzaModel = InfluenzaModel .*(area_d/area_n);

plot(T,InfluenzaModel,'r-')
hold on
plot(T',incidences,'b-o')

title('Plot of model of B influenza with parameters R number = 1.2390 and t_0 = 234 days fitted by comparison against recorded incidences of the disease')
xlabel('t (days)')
ylabel('number of incidences')
legend("Modelled Incidences with t_0 and R number fit against the data","Recorded Incidences",'Location','southwest')
```

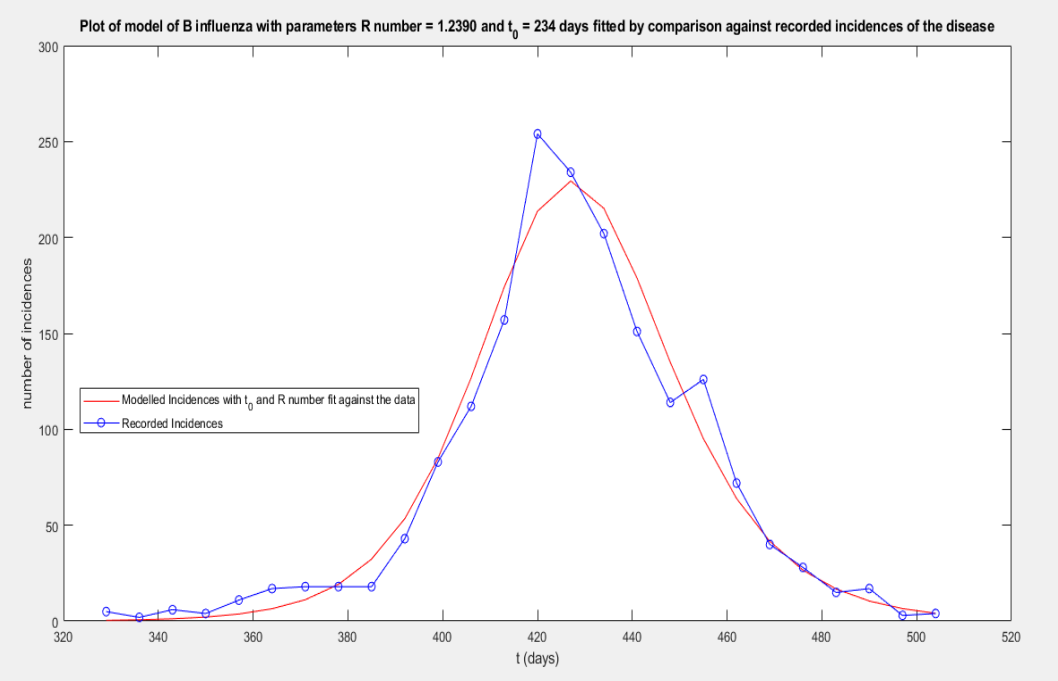


Figure 10: Plot of least square solution from fitting R number and t0 to the model

We see that the model now makes a good fit with the data. This model gives us a value of $R_{\text{number}} = 1.239$ and $t_0 = 234$ days.

3.5 Comments

The R number is also know as the Basic Reproduction Number and it describes a diseases ability to spread to others. For example, an R number of 2 for a disease, means that on average, if someone is infected with the disease they will spread it to 2 other people. The R number is very important for Epidemiologists in modelling disease spread.

The SIR model is a very basic model in disease modelling but it gives us a good idea of how diseases are modelled. More advanced models can factor in more variables or can modify the SIR model i.e. introducing another cohort for people who die from the diseases or can reintroduce members from R(t) into S(t) when immunity wears off.