EE498 Project Report

Adem Deniz Pişkin *2375673*

Part 1: Lane Keeping and Obstacle Avoidance with MPC Control

a. Modelling

In our problem formulation we actually use 3 different models. First we use the semi-linearized model in the paper to model the vehicle dynamics in the simulation. The second is the so called conservative model, which is the linearized model that we are using for our controller. Lastly the third model is the "Overreacting model", this model is used only in the constraint in order to make sure our doesn't stray to much from the actual non-linear dynamics. We introduce those models in the following sections. Our state variables for the control formulation are selected as $\omega = \begin{bmatrix} \dot{y} & \dot{\psi} & e_\psi & e_y & \delta \end{bmatrix}^T$, additionally we have the longitudinal dynamic variables s and $\dot{x}$ that have separate dynamics.

i. Non-Linear Model

This model is taken from eqn(7) of the paper. We assume here that $\dot{x} = v$ where $v \in \mathbb{R}$ is a constant.

$$\ddot{x} = g\mu\beta - \frac{k_d}{m}\dot{x}^2 = 0 \tag{1}$$

$$\ddot{y} = v\dot{\psi} + \frac{F_{y_f} + F_{y_r}}{m} \tag{2}$$

$$\ddot{\psi} = \frac{aF_{y_f} - bF_{y_r}}{I} \tag{3}$$

$$\dot{e_\psi} = \dot{\psi} - v\psi_r \tag{4}$$

$$\dot{e_y} = \dot{y} + ve_\psi \tag{5}$$

$$\dot{\delta} = u \tag{6}$$

$$\dot{s} = v \tag{7}$$

From the fact that $\dot{x} = v$ we can find β as

$$\beta = \frac{k_d v^2}{mg\mu} \tag{8}$$

Here F_{y_f} and F_{y_r} are given by the nonlinear relation below:

$$F_{y_f} = \mu g \frac{b - e\beta}{a + b} \sqrt{1 - \beta^2} \sin(C \arctan(B\alpha_f)) \tag{9}$$

$$F_{y_r} = \mu g \frac{b + e\beta}{a + b} \sqrt{1 - \beta^2} \sin(C \arctan(B\alpha_r)) \tag{10}$$

Where $\alpha_f = \frac{\dot{y} + a\dot{\psi}}{v} - \delta$ and $\alpha_r = \frac{\dot{y} - b\dot{\psi}}{v}$.

ii. Conservative model

This model is described in equation (9) of the paper. We basically linearize equations 9 and 10 with respect to the slip angles $\alpha_{f,r}$. Since $\alpha_{f,r}$ are a linear combination of the state variables so we can write the state variables as a linear model with an additive constant.

$$\begin{bmatrix} \ddot{y} \\ \ddot{\psi} \\ \dot{e_\psi} \\ \dot{e_y} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \frac{C_{f_L}+C_{r_L}}{mv} & \frac{C_{f_L}a-C_{r_L}b}{mv} - v & 0 & 0 & -\frac{C_{f_L}}{m} \\ \frac{aC_{f_L}-bC_{r_U}}{Iv} & \frac{a^2C_{f_U}+b^2C_{r_U}}{Iv} & 0 & 0 & -\frac{aC_{f_L}}{I} \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & v & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{y} \\ \dot{\psi} \\ e_\psi \\ e_y \\ \delta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u - \begin{bmatrix} 0 \\ 0 \\ v\psi_r \\ 0 \\ 0 \end{bmatrix} \tag{11}$$

Equation 13 can be compactified as as:

$$\dot{\omega} = A_{c,cm}\omega + B_c u + e_c \tag{12}$$

iii. Overreacting Model

For the overreacting model we again linearize equations 9 and 10, but this time we use the worst estimates of the line and find to find the appropriate constants. The new affine equation is very similar to equation 13 and is given in **??**.

$$\begin{bmatrix} \ddot{y} \\ \ddot{\psi} \\ \dot{e_\psi} \\ \dot{e_y} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \frac{C_{f_U}+C_{r_U}}{mv} & \frac{C_{f_U}a-C_{r_U}b}{mv} - v & 0 & 0 & -\frac{C_{f_U}}{m} \\ \frac{aC_{f_U}-bC_{r_L}}{Iv} & \frac{a^2C_{f_L}+b^2C_{r_U}}{Iv} & 0 & 0 & -\frac{aC_{f_L}}{I} \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & v & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{y} \\ \dot{\psi} \\ e_\psi \\ e_y \\ \delta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u - \begin{bmatrix} 0 \\ 0 \\ v\psi_r \\ 0 \\ 0 \end{bmatrix} \tag{13}$$

Once again we can compactify the notation as:

$$\dot{\omega} = A_{c,om}\omega + B_c u + e_c \tag{14}$$

iv. Discretization

We have obtained the CM and OM models in continuous time but in order to implement MPC we need to discretize our models. However taking matrix exponential can prove difficult in a complicated model, here we are going to assume that the sampling time T is small and do a first order approximation in the Taylor's Series.

$$\begin{aligned} A &= e^{A_cT} = I + A_cT + \frac{(A_cT)^2}{2} + \mathcal{O}(T^3) \approx I + A_cT \\ B &= \int_0^T e^{A_ct}dtB_c \approx \int_0^T (I + A_cT)dtBc \\ &= \int_0^T B_cdt + \int A_cB_cTdt = B_cT + A_cB_cT^2 \approx B_cT \end{aligned}$$

Finally we have the constant term, this one is just an added constant to the derivative so it is easy to see that it's effect can easily be found by integrating, if we assume that e_c is a slow process we can find an easy approximation as follows.

$$\omega_{k+1} = \int_{kT}^{(k+1)T} (A_c\omega + B_c u + e_c)dt = A\omega_k + Bu_k + \int_{kT}^{(k+1)T} e_c dt = A\omega_k + Bu_k + e_c(t_k)T$$

If we define $e_k = e_c(t_k)T$ we can write our linear models with offset as follows:

$$\omega_{k+1} = A\omega_k + Bu_k + e_k \tag{15}$$

b. MPC Formulation

We have an almost linear model that we can now use. However to be able more realistic we need to also implement constraints in our model. In practise most of the those variables in a car would be bounded, moreover those bounds can be relatively small sometimes meaning we can't ignore them. In order to be able to implement those constraints we will need an MPC controller. An MPC controller is a great choice with dealing with linear constraints which are the type of constraints that we are going to introduce. Moreover MPC of a affine system (linear + constant) can be solved by an MPC easily as a convex problem. We will show that the resultant optimization problem is still a convex problem that can be solved with quadratic programming.

i. Cost Function

The cost function can be written as follows:

$$J = ||\omega_{N|k}||^2_{Q_f} + \sum_{j=0}^{N-1} ||\omega_{j|k}||^2_Q + ||u_{j|k}||^2_R$$

Here $||.||_W$ simply denotes the weighted L_2 norm defined as $||x||^2_W = x^TWx$. We will now observe what the additional e_k term on the equations of model do to the time series. Let us look at the first 3 terms to see the pattern.

$$\begin{aligned}
\omega_1 &= A\omega_0 + Bu_0 + e_0 \\
\omega_2 &= A\omega_1 + Bu_1 + e_1 = A^2\omega_0 + ABu_0 + Bu_1 + Ae_0 + e_1 \\
\omega_3 &= A\omega_2 + Bu_2 + e_2 = A^3\omega_0 + A^2Bu_0 + ABu_1 + Bu_2 + A^2e_0 + Ae_1 + e_2
\end{aligned}$$

Using induction one can easily prove that:

$$\omega_k = A^k \omega_0 + \sum_{j=0}^{k-1} A^{k-1-j} B u_j + A^{k-1-j} e_j \tag{16}$$

We see that the vector form of the equations of motion will involve an extra term made of e_j multiplied by a matrix similar to the G matrix we multiply the U vector where B is replaced with identity.
Let Ω be the vectorization of ω_j vectors, U be the vectorization of the u_j vectors and finally E being the vectorizaiton of the e_j vectors (stacking on top of each other). Let's define the following matrices:

$$G = \begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ AB & B & 0 & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix} \qquad H = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}$$

$$K = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ I & 0 & 0 & \dots & 0 \\ A & I & 0 & \dots & 0 \\ A^2 & A & I & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ A^{N-1} & A^{N-2} & \dots & A & I \end{bmatrix}$$

Then the equations of motion for the horizon N can be written as follows:

$$\Omega = GU + H\omega_0 + KE$$

We can now define the final matrices $\underline{Q}, \underline{R}$ as:

$$\underline{Q} = \begin{bmatrix} Q & 0 & \dots & 0 & 0 \\ 0 & Q & 0 & \dots & \vdots \\ \vdots & 0 & \ddots & 0 & \vdots \\ \vdots & \dots & 0 & Q & 0 \\ 0 & \dots & \dots & 0 & Q_f \end{bmatrix} \qquad \underline{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & 0 & \vdots \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}$$

The Cost function then can be easily rewritten as:

$$\begin{aligned} J &= \Omega^T \underline{Q} \Omega + U^T \underline{R} U \\ &= (GU + H\omega_0 + KE)^T \underline{Q} (GU + H\omega_0 + KE) + U^T \underline{R} U \\ &= U^T G^T \underline{Q} G U + 2(H\omega_0 + KE)^T \underline{Q} G U + (H\omega_0 + KE)^T \underline{Q} (H\omega_0 + KE) + U^T \underline{R} U \\ &= U^T (G^T \underline{Q} G + \underline{R}) U + 2(H\omega_0 + KE)^T \underline{Q} G U + (H\omega_0 + KE)^T \underline{Q} (H\omega_0 + KE) \end{aligned}$$

The right most term is constant so we can remove it, if we define $M = G^T \underline{Q} G + \underline{R}$ and $\alpha = G^T \underline{Q}^T (H\omega_0 + KE)$ the cost function simply becomes:

$$J = U^T M U + 2\alpha^T U \tag{17}$$

Which is a convex optimization problem since M is symmetric. We can use quadratic programming with M and α.

ii. Constraints

We are using the constraint types given in the paper. We want to put the constraints into the form of $FU \leq V$. Most constraints in the paper boil down to a linear constraint in ω_j, so let's first consider the case that: $C_l \leq f_v \omega \leq C_u$.
Define the matrix $\underline{f_v}$ as the block diagonal matrix created by putting the f_v vector in the diagonal $N + 1$ times. Let $\underline{C_l}, \underline{C_u}$ be the constant put to a vector form. Then the constraint becomes $\underline{C_l} \leq \underline{f_v}\Omega \leq \underline{C_u}$. We can replace here $\Omega = GU + H\omega_0 + KE$ and we get two equations:

$$\underline{f_v} G U \leq \underline{C_u} - \underline{f_v}(H\omega_0 + KE)$$
$$\underline{f_v} G U \geq \underline{C_l} - \underline{f_v}(H\omega_0 + KE)$$

Those can be combined into one equation as:

$$\begin{bmatrix} \underline{f_v} G \\ -\underline{f_v} G \end{bmatrix} U \leq \begin{bmatrix} \underline{C_u} - \underline{f_v}(H\omega_0 + KE) \\ \underline{f_v}(H\omega_0 + KE) - \underline{C_l} \end{bmatrix} \tag{18}$$

Equation 18 will be the basis for almost all our constraints.

Actuator limits

We have limits on the steering angle (δ) and it's rate of change $\dot{\delta} = u$, so basically on the input. Those constraint are written as:

$$-\delta_{lim} \leq \delta \leq \delta_{lim} \tag{19}$$
$$-\dot{\delta}_{lim} \leq u \leq \dot{\delta}_{lim} \tag{20}$$

Defining $f_\delta = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ we can easily write the constraints using the results of equation 18. The other constraint on the rate of change is just directly proportional to u so it is even easier. Let I_N denote the $N \times N$ identity matrix.

$$\begin{bmatrix} \underline{f_\delta} G \\ -\underline{f_\delta} G \\ I_N \\ -I_N \end{bmatrix} U \leq \begin{bmatrix} \underline{\delta_{lim}} - \underline{f_\delta}(H\omega_0 + KE) \\ \underline{\delta_{lim}} + \underline{f_\delta}(H\omega_0 + KE) \\ \underline{\dot{\delta}_{lim}} \\ \underline{\dot{\delta}_{lim}} \end{bmatrix} \tag{21}$$

Slip Angle Limits
We have limits on the slip angles simply given as $|\alpha_{f,r}| \leq \alpha_{f,r,lim}$. Since those angles are a linear combinations of ω the constraints can be easily written as such. If we define $f_{\alpha_f} = \begin{bmatrix} \frac{1}{v} & \frac{a}{v} & 0 & 0 & -1 \end{bmatrix}$ and $f_{\alpha_r} = \begin{bmatrix} \frac{1}{v} & \frac{-b}{v} & 0 & 0 & 0 \end{bmatrix}$ we can write the constraints simply as $-\alpha_{f,lim} \leq f_{\alpha_f}\omega \leq \alpha_{f,lim}$ and $-\alpha_{r,lim} \leq f_{\alpha_r}\omega \leq \alpha_{r,lim}$. At this point the constraints can simply be written as follows:

$$\begin{bmatrix} \underline{f_{\alpha_f}}G \\ -\underline{f_{\alpha_f}}G \\ \underline{f_{\alpha_r}}G \\ -\underline{f_{\alpha_r}}G \end{bmatrix} U \leq \begin{bmatrix} \underline{\alpha_{f,lim}} - \underline{f_{\alpha_f}}(H\omega_0 + KE) \\ \underline{\alpha_{f,lim}} + \underline{f_{\alpha_f}}(H\omega_0 + KE) \\ \underline{\alpha_{r,lim}} - \underline{f_{\alpha_r}}(H\omega_0 + KE) \\ \underline{\alpha_{r,lim}} + \underline{f_{\alpha_r}}(H\omega_0 + KE) \end{bmatrix} \tag{22}$$

Lane Width and Obstacle Avoidance
Those two constraints go hand in hand since they both are in a sense effecting the state e_y. Let's define the bounds as follows:

$$e_{y,r_{lim}} = \begin{cases} -\frac{l_w}{2}, & \text{if no obstacle.} \\ e_{y,r_{safe}}, & \text{if there is obstacle.} \end{cases} \tag{23}$$

$$e_{y,l_{lim}} = \begin{cases} \frac{l_w}{2}, & \text{if no obstacle.} \\ e_{y,l_{safe}}, & \text{if there is obstacle.} \end{cases} \tag{24}$$

Where $e_{y,l_{safe}}$ and $e_{y,r_{safe}}$ are the left and right side of the safe zone when there is an obstacle. Then the problem is simply $e_{y,r_{lim}} \leq e_y \leq e_{y,l_{lim}}$ which can be easily written as:

$$\begin{bmatrix} \underline{f_y}G \\ -\underline{f_y}G \end{bmatrix} U \leq \begin{bmatrix} e_{y,l_{lim}} - \underline{f_y}(H\omega_0 + KE) \\ -e_{y,r_{lim}} + \underline{f_y}(H\omega_0 + KE) \end{bmatrix} \tag{25}$$

Here $f_y = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix}$ since $f_y\omega = e_y$.
Overall Constraint
Putting everything together we can get the mega constraint matrices F, V that satisfy $FU \leq V$ as given in equation 26. Note that we are using 2 models for constraints. We use the full horizon of 45 steps for the linear model since this is also the model we use in the cost function, we also use for a shorter horizon of 20 steps the overreacting model in order to prevent the model from diverging too much from the actual non-linear model of the vehicle. So the overall constraint will be the combination of both where $F = \begin{bmatrix} F_{cm} \\ F_{om} \end{bmatrix}$ and $V = \begin{bmatrix} V_{cm} \\ V_{om} \end{bmatrix}$ where $\{F_{cm}, V_{cm}\}$ and $\{F_{om}, V_{om}\}$ pairs are created using equation 26 and the A matrix of the respective model.

$$F = \begin{bmatrix} \underline{f_\delta}G \\ -\underline{f_\delta}G \\ I_N \\ -I_N \\ \underline{f_{\alpha_f}}G \\ -\underline{f_{\alpha_f}}G \\ \underline{f_{\alpha_r}}G \\ -\underline{f_{\alpha_r}}G \\ \underline{f_y}G \\ -\underline{f_y}G \end{bmatrix} \qquad V = \begin{bmatrix} \underline{\delta_{lim}} - \underline{f_\delta}(H\omega_0 + KE) \\ \underline{\delta_{lim}} + \underline{f_\delta}(H\omega_0 + KE) \\ \underline{\dot{\delta}_{lim}} \\ \underline{\dot{\delta}_{lim}} \\ \underline{\alpha_{f,lim}} - \underline{f_{\alpha_f}}(H\omega_0 + KE) \\ \underline{\alpha_{f,lim}} + \underline{f_{\alpha_f}}(H\omega_0 + KE) \\ \underline{\alpha_{r,lim}} - \underline{f_{\alpha_r}}(H\omega_0 + KE) \\ \underline{\alpha_{r,lim}} + \underline{f_{\alpha_r}}(H\omega_0 + KE) \\ e_{y,l_{lim}} - \underline{f_y}(H\omega_0 + KE) \\ -e_{y,r_{lim}} + \underline{f_y}(H\omega_0 + KE) \end{bmatrix} \tag{26}$$

iii. Cost Matrices

We need to determine the cost matrices Q, Q_f, R. In general using dual mode prediction provides stability and better results thus we are going to use dual mode where $Q_f = P$ where P is the solution of the discrete time Lyapunov equation. This leaves the choice of R and Q. Note that our input is scalar hence R is also a scalar say ρ, the exact value will be determined experimentally by trying several configurations. Eventually after several experiments similar as of those in section d. the most suitable value was found to be 0.1

Lastly we need to decide on the structure of Q. Since this is a lane keeping problem we would like to follow the lane as close as possible thus e_y must be minimized. moreover to not get out of the lane the angle with respect to the road must also be tried to minimize hence e_ψ must go to zero. Other parameters don't have to be necessarily zero since as the car turns while following the road, they can take non-zero values. So our Q must minmize those 2 states, we select Q as follows:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{27}$$

We gave e_y a larger weight since it is more important for it to converge then e_ψ.

d. Simulation Results

This section discusses the simulation experiments and their results. We will have 4 main experiments in this session. First we will see how the car drives on a curved road. Secondly we are going to see the disturbance rejection where the car will be placed at a offset position with respect to a straight road. In the third test a smaller disturbance test will be made on a curved road. Unfortunately we were not able to perform obstacle avoidance.

Parameters & Constraints

Some of the parameters were selected the same as the paper. Some were taken from other real life cars found on the internet. Below is the list of all the parameters and constraints.

$$
\begin{aligned}
m &= 2050kg & I &= 3344kg \cdot m^2 \\
\mu &= 0.3 & k_d &= 0.1838 \\
b &= 1.52m & a &= 0.92m \\
e &= 1.112m & l_w &= 4.6m \\
B &= 10.8 & C &= 0.908 \\
C_{f_L} &= 1.445 \times 10^4 & C_{r_L} &= 1.834 \times 10^4 \\
C_{f_U} &= 2.38 \times 10^4 & C_{r_U} &= 3.022 \times 10^4 \\
\delta_{lim} &= 15^\circ & \dot{\delta}_{lim} &= 30^\circ \\
\alpha_{f,lim} &= -8^\circ & \alpha_{r,lim} &= 8^\circ
\end{aligned}
$$

i. Curved Road

First we experimented with speed 40m/sec. The road is a circular road with a radius of 10km. The path of the vehicle along the lane is given in figure 1, the errors e_y and e_ψ are given in figure 2 and the other states are given in figure 3.
We see that the vehicle manages the follow the road along this curved road. The error graphs show however some steady state error. The error in the angle is around 0.3 degrees which is extremely small and can be ignored, there is some error however in the position as well. The vehicle seems to converge to about 10cm off the line. This is an acceptable result as 10cm is a small error and we see that the vehicle keeps the error constant and doesn't diverge. We also see that the other states also converge to some values to keep the vehicle in steady state while on the curved road.

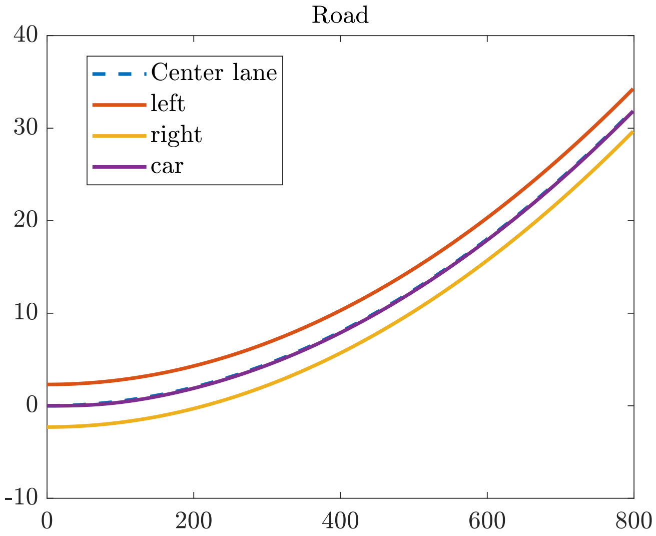


Figure 1: Vehicle's travel along the lane when $v = 40\text{m/sec}$

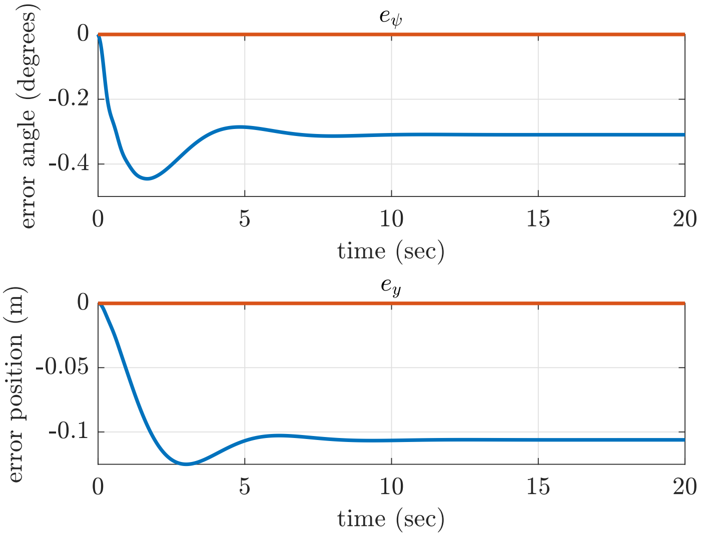


Figure 2: Errors while following the curved road when $v = 40\text{m/sec}$

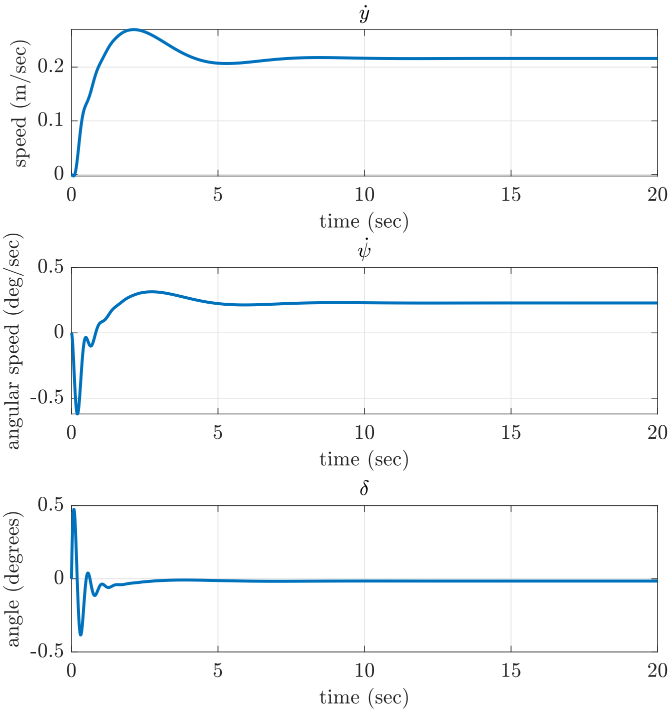


Figure 3: States while following the curved road when $v = 40$m/sec

Now we are going to test for $v = 60$ and 20 m/sec case. The vehicles path, error state plots for the 60m/sec case are given in figures 4, 5 and 6. The same plot respectively. The same plots for the $v = 20$m/sec case are given in figure 7,8,9 respectively.

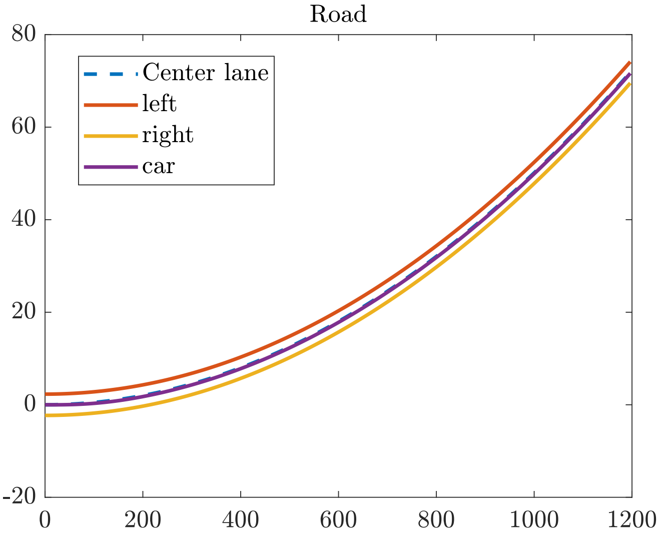


Figure 4: Vehicle's travel along the lane when $v = 60$m/sec

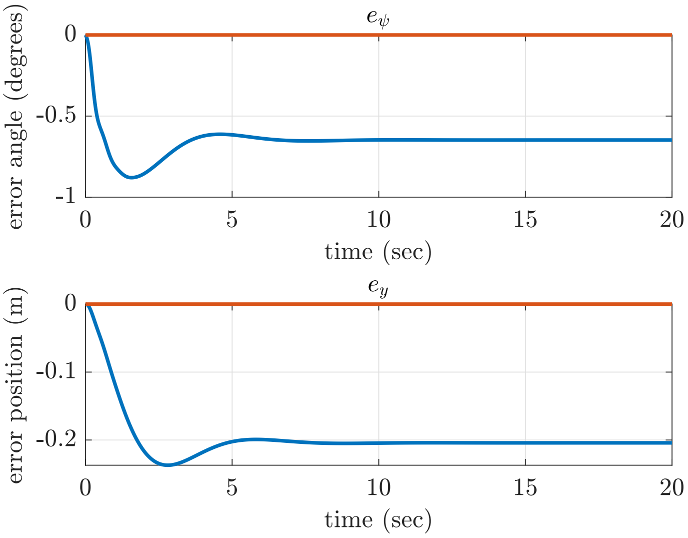


Figure 5: Errors while following the curved road when $v = 60$m/sec

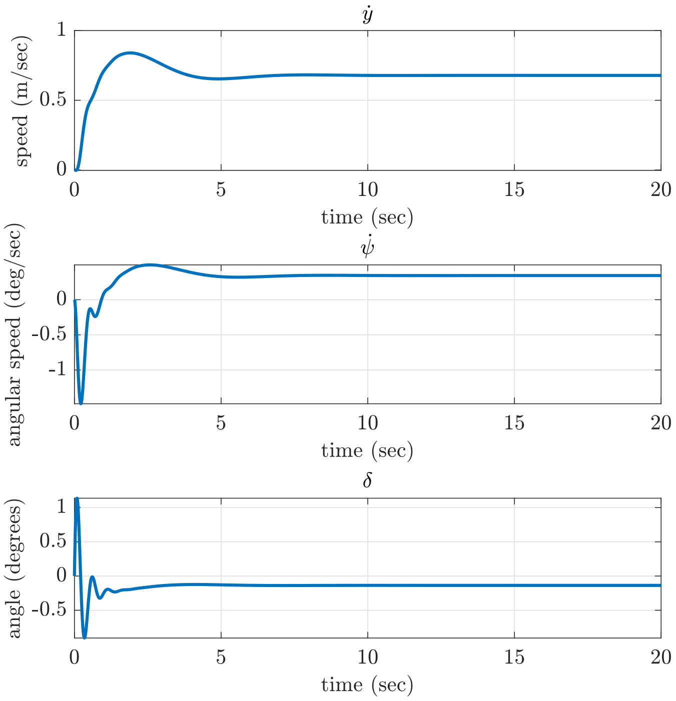


Figure 6: States while following the curved road when $v = 60$m/sec

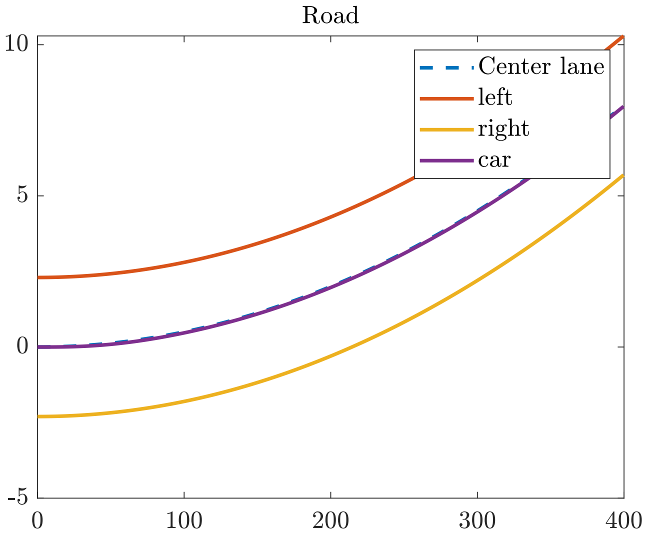


Figure 7: Vehicle's travel along the lane when $v = 20$m/sec

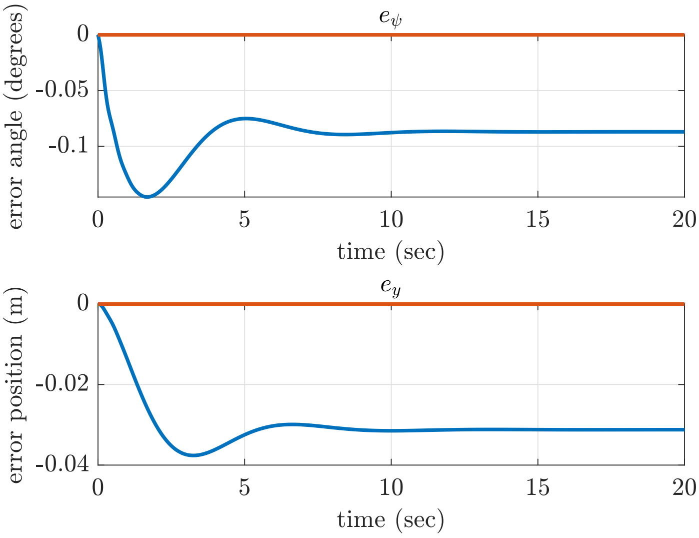


Figure 8: Errors while following the curved road when $v = 20$m/sec

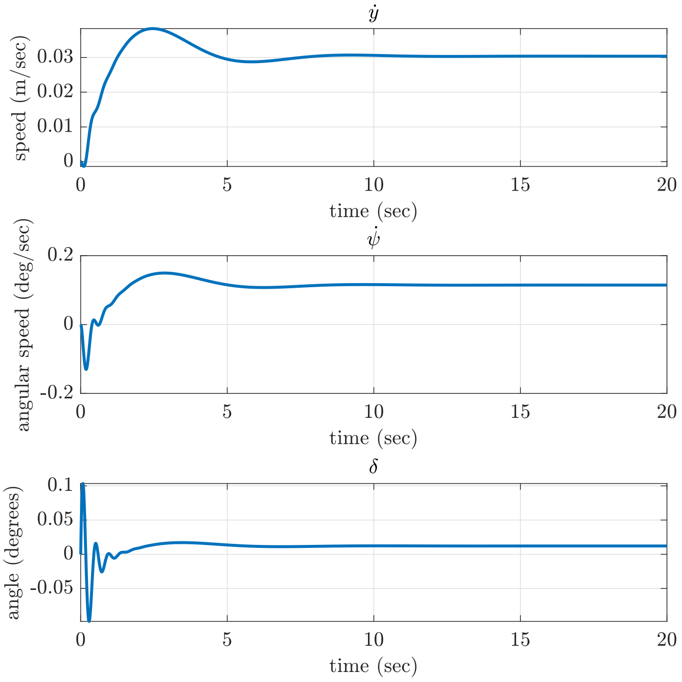


Figure 9: States while following the curved road when $v = 20$m/sec

In all cases we see that the car manages to follow the road without problem so we have not accidents where the car flies off the road. When it comes to error we see that as speed increases the steady state error increases. For e_ψ that doesn't seem to be a problem as the maximum error becomes 1 degree which is very small and negligible. However sometimes the e_y error becomes little more troublesome, especially in the v = 60m/sec case we have around 20cm steady state error, sometimes this might not be very desirable, however it is important to keep in mind that 60m/sec is 216km/h which is extremely high so such error might be expected. The 20m/sec case however is significantly smaller, around 3.5 cm which is completely tolerable.

We also note that the settling time of the car seems to be independent of the speed of the car and it is around 10sec. This is not a very long time considering the car doesn't do a very drastic behaviour in the process.Overall we see that in the range of 20-60 m/sec (72-216 km/h) the car could do lane keeping without a lot of trouble.

ii. Disturbance

Here we will let the car start with an offset of about 1m in a straight road. We are going to observe how the cars converge to the center lane. Again tests are performed at 20,40,60 m/sec speeds. The figures for the 60m/sec case are given in figures 10,11,12. The v = 40m/sec case are given in figures 13,14,15. Last but not leas the v = 20m/sec case results are given in figures 16,17,18.

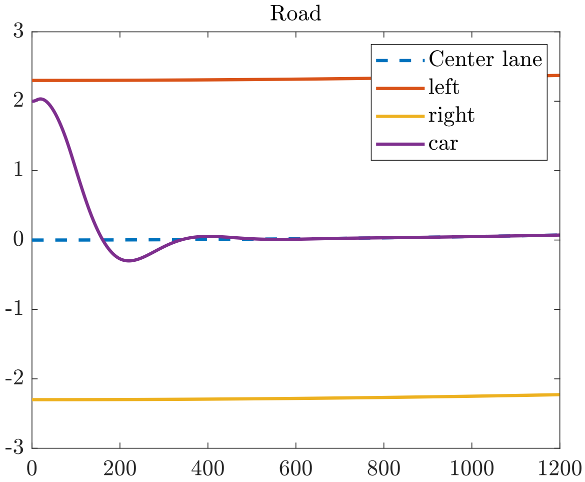


Figure 10: Vehicle disturbed traveling along the lane at v = 60m/sec

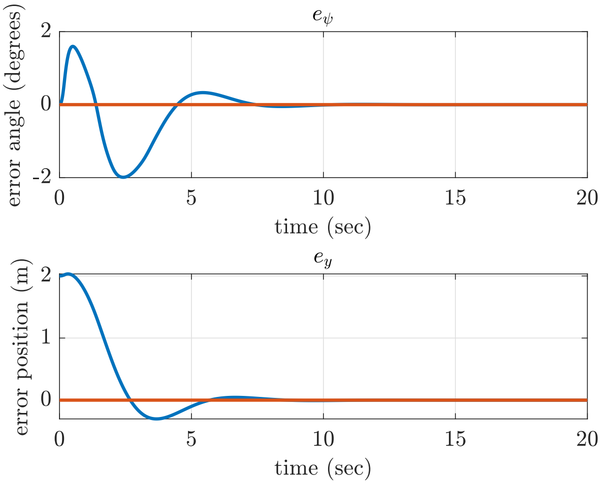


Figure 11: Errors plots when disturbed at $v = 60$m/sec

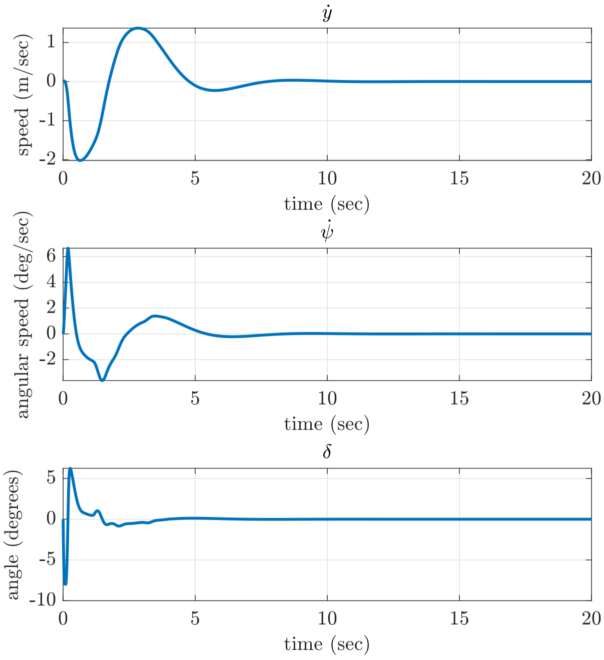


Figure 12: States while being disturbed at $v = 60$m/sec

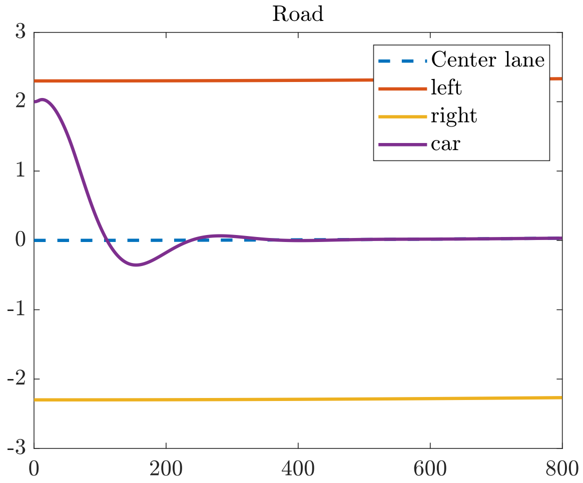


Figure 13: Vehicle disturbed traveling along the lane at $v = 40$m/sec

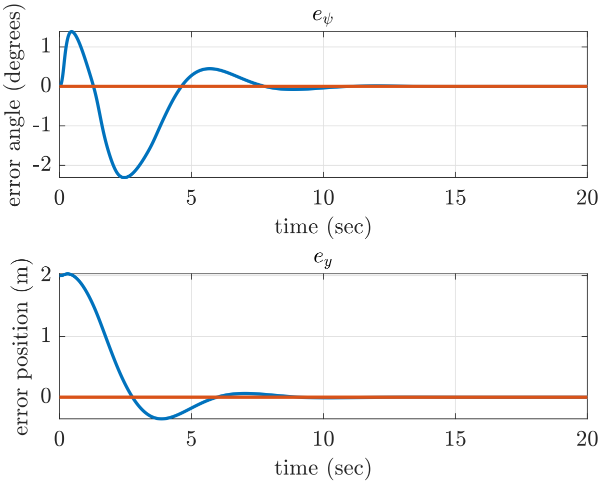


Figure 14: Errors plots when disturbed at $v = 40$m/sec

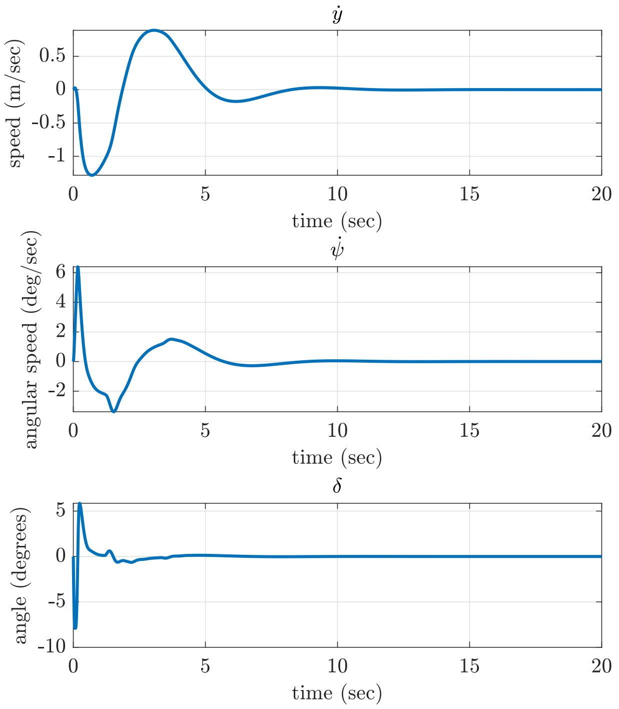


Figure 15: States while being disturbed at $v = 40$m/sec

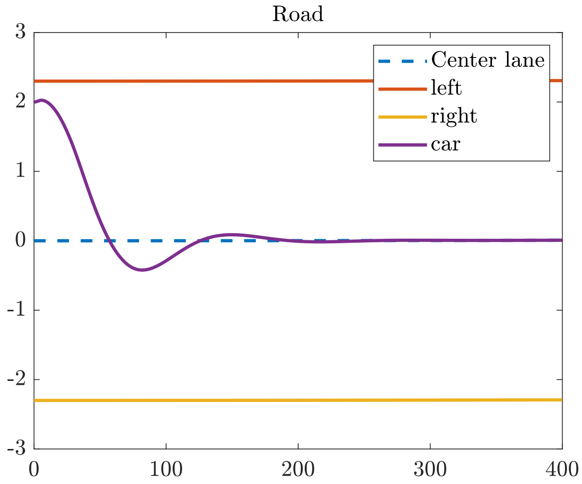


Figure 16: Vehicle disturbed traveling along the lane at $v = 20$m/sec

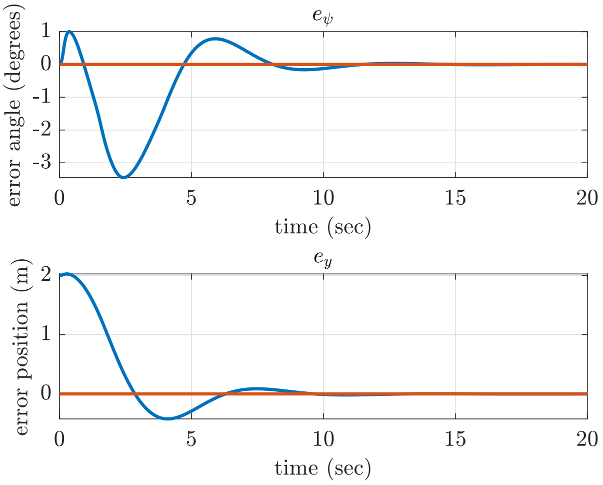


Figure 17: Errors plots when disturbed at $v = 20\text{m/sec}$

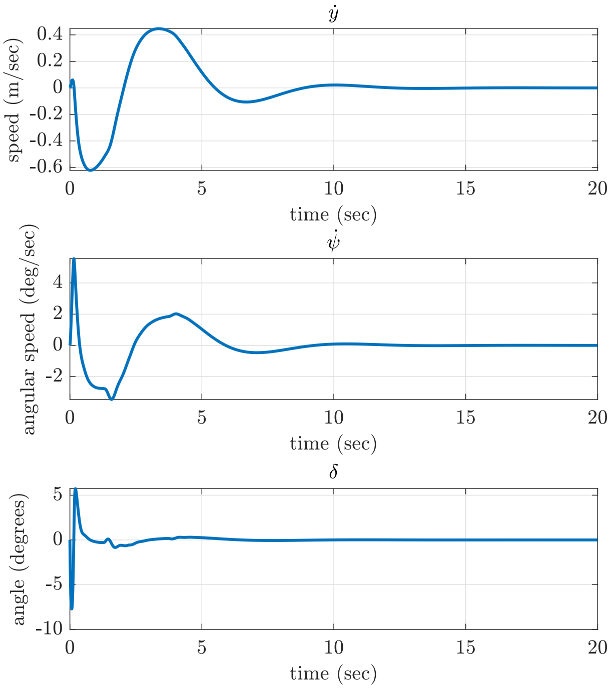


Figure 18: States while being disturbed at $v = 20\text{m/sec}$

We see that in all cases e_y converges to zero in about 5 sec after-some overshoot, independent of the speed of the vehicle. The time could be improved in a real life scenario however it is not unacceptable. We see also that e_ψ converges to zero as well in about 7 sec, little bit longer. The overshoot in e_psi seems to be the larges when v = 20m/sec (figure 17) with about -3 degrees overshoot, this is not necessarily an expected result however 3 degrees is not a very large error hence this is not a big issue. Overall we see that the disturbance has been rejected in an efficient manner in all the speeds almost equally.

iii. Curved Road + Disturbance

For this case we are going to only try the v = 40m/sec case. The road is little less curved $\psi_r = 10^{-5}$. The disturbance now is 1m in the e_y. The results are given in figures 19, 20 and 21. We see that once again despite the curved road the system can reject disturbances while also following the curved road without a problem. The steady state error here is almost non-existent and convergence once again happens around 5 seconds. We see that the system can successfully perform both tasks at once.

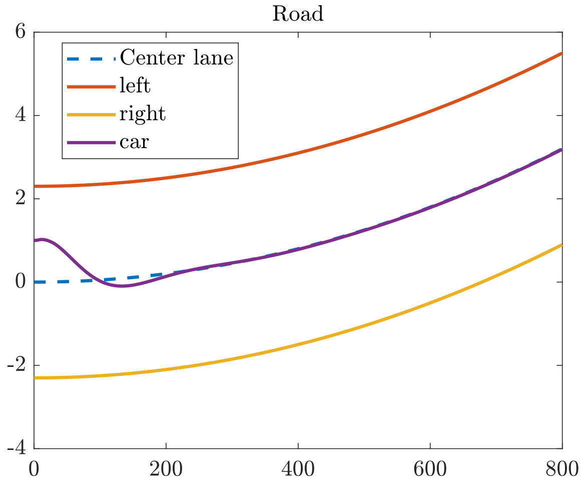


Figure 19: Vehicle disturbed traveling along curved lane at v = 40m/sec

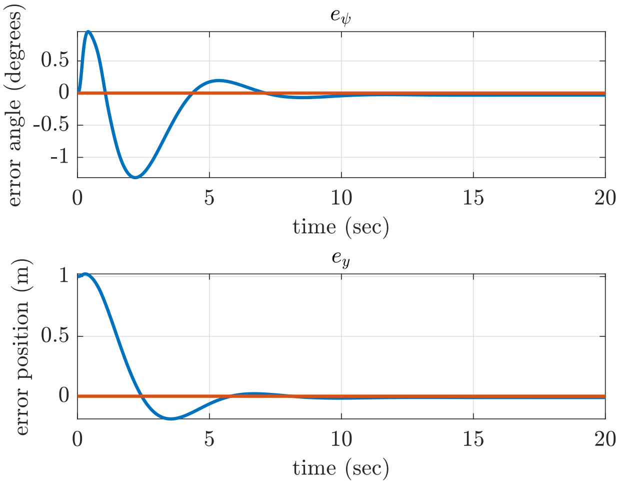


Figure 20: Errors plots when disturbed at $v = 40$m/sec in curved lane

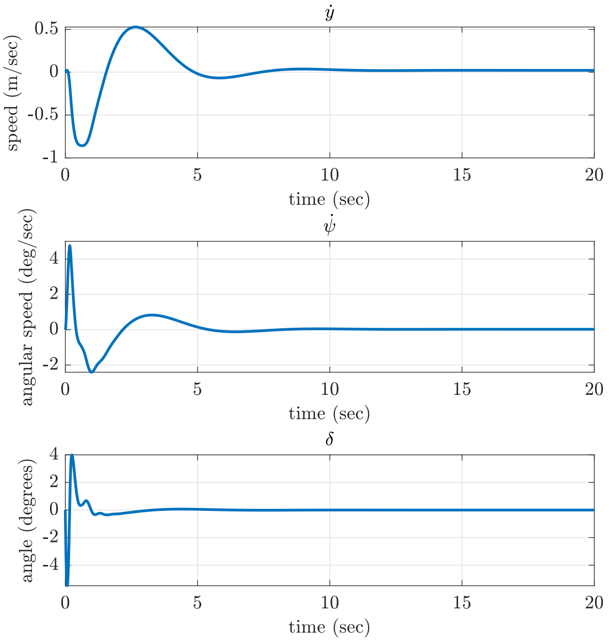


Figure 21: States while being disturbed at $v = 40$m/sec at curved lane

Part 2: Epidemic Simulation

In this part we are simulating the SARS Epidemic, here our parameters are chosen to be the same as in the paper, namely $\omega = 1/year$, $\gamma = 1/(14days)$, $\sigma = 1/(7days)$, $\mu = 1/(76years)$, $\beta = 0.21days$ and $\alpha = 0$. The initial conditions are $S(0) = 0.999$, $E(0) = 0.001$, $I(0) = R(0) = 0$.

a. Simulink

Using the ode45 solver we first looked at various changes in the simulation as we changed the relative tolerance from 10^{-5} up until 10^{-1}. Then we also varied the maximum step size to see if that did any difference. Lastly we tried the fixed step size method for various step sizes.

i. Relative Tolerance

We tried 4 different values of the relative tolerance, in these cases the maximum step sizes were set to "auto". The relative tolerance values tried are 10^{-5}, 10^{-3}, 10^{-2} and 10^{-1}. The results are given in figures 22,23 and 24 respectively. The relative tolerance of 10^{-1} cannot be presented as the result diverged and gave an error.

We see that the results are similar to those in the paper. We see the similar damped oscillation like structure. We can also check the steady state values. The paper has found that $R(\infty) = 0.63$,$S(\infty) = 0.33$, $I(\infty) = 0.024$, $E(\infty) = 0.012$. We see that our steady state results are very close, the R and S values are only little bit off by 1%. We also notice that as the relative tolerance increases the accuracy starts to decrease as seen from the 10^{-2} case (figure 24), moreover further increase can cause instability as in the case of 10^{-1} where the plot can't be even drawn properly.

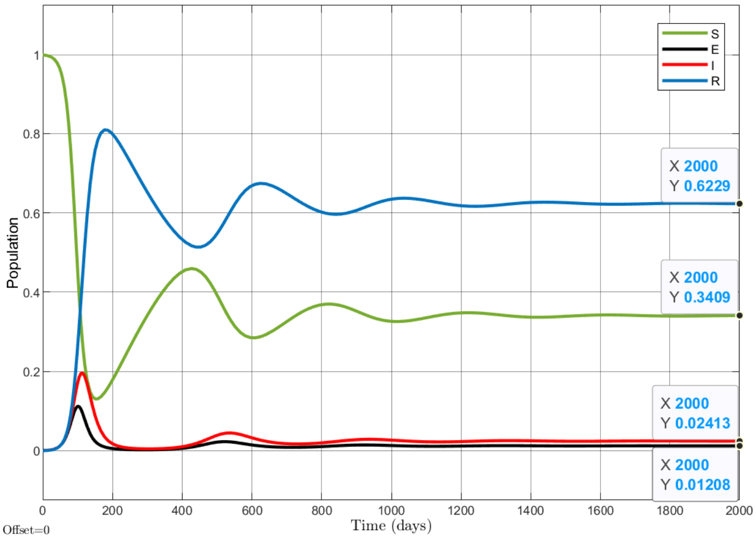


Figure 22: Epidemic Simulation with Simulink variable step size rel_tol = 10e-5

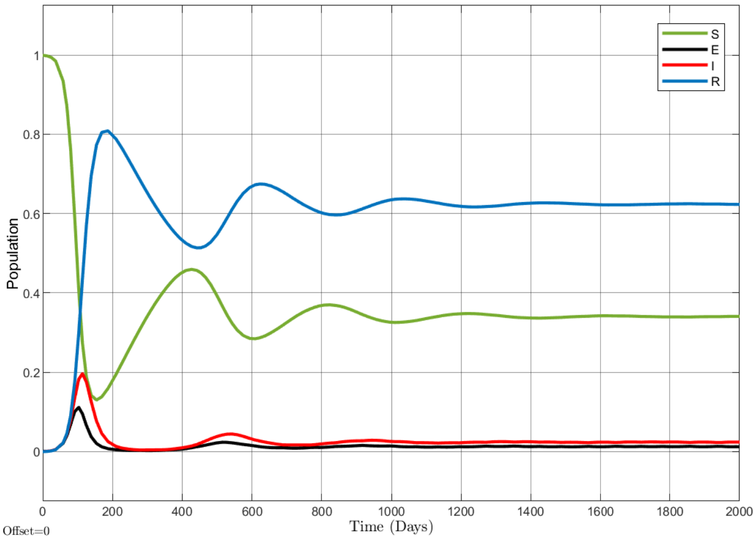


Figure 23: Epidemic Simulation with Simulink variable step size rel_tol = 10e-3

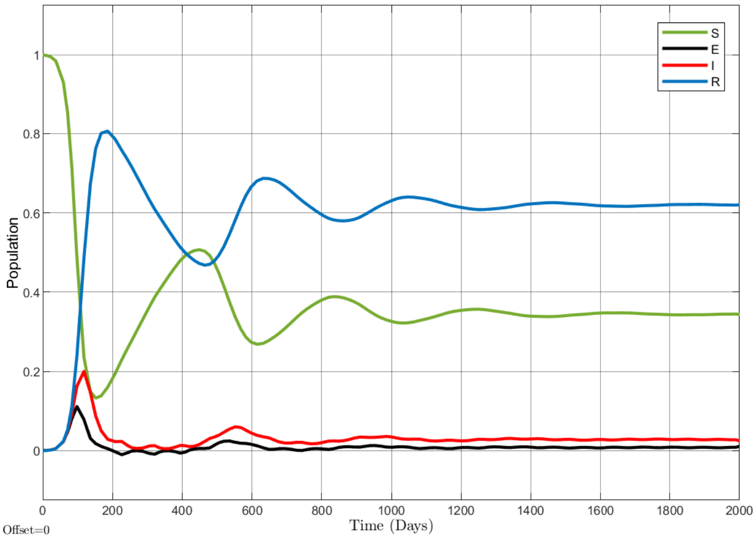


Figure 24: Epidemic Simulation with Simulink variable step size rel_tol = 10e-2

ii. Maximum Step-size

Now we have reduced the maximum step-size to 1 and run the simulation for the same relative tolerance values. The results are given in figures 25,26,27,28.

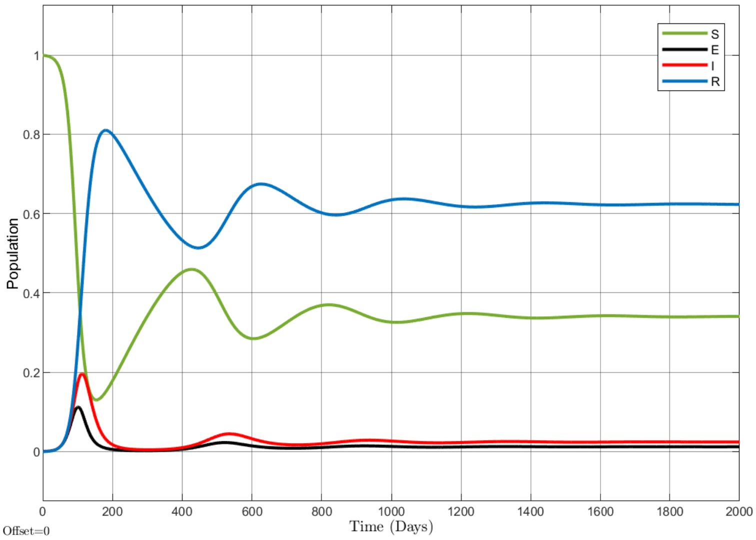


Figure 25: Epidemic Simulation with Simulink variable step size rel_tol = 10e-5 maxstep=1

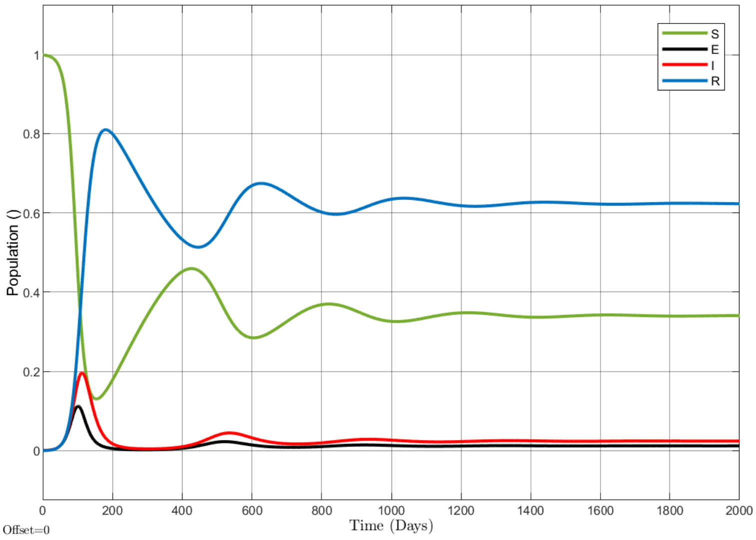


Figure 26: Epidemic Simulation with Simulink variable step size rel_tol = 10e-3 maxstep=1

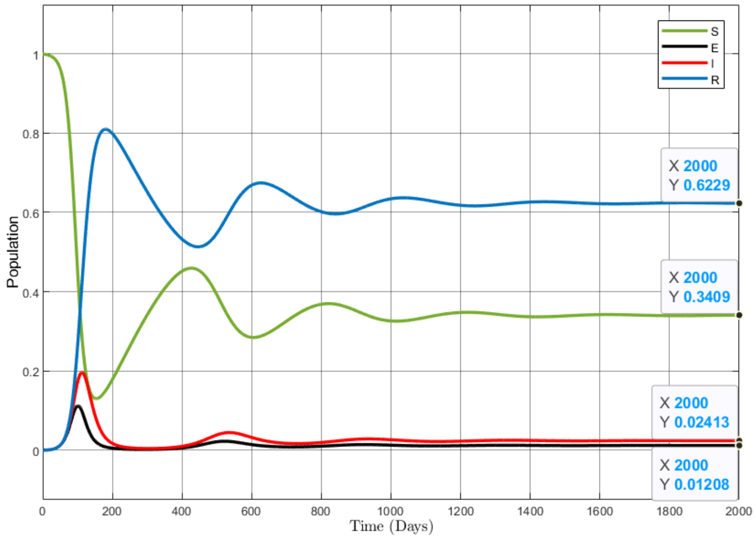


Figure 27: Epidemic Simulation with Simulink variable step size rel_tol = 10e-2 maxstep=1

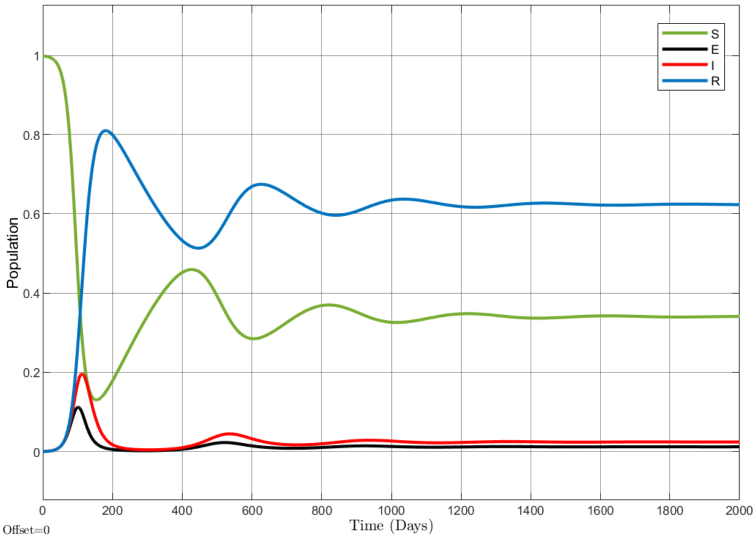


Figure 28: Epidemic Simulation with Simulink variable step size rel_tol = 10e-1 maxstep=1

Looking at figures 25 and 26 we see almost no changes in the result, for accurate simulations decreasing the step size seems to not have a large effect. On the other hand if we look at figure 27 we see that the inaccuracies present from figure 24 have now been remedied, thus decreasing the maximum step-size improved the accuracy of a low accuracy simulation. Even more importantly now at relative tolerance of 10^{-1} we have a stable and accurate simulation (figure 28) whereas before the simulation was unstable. So decreasing the maximum step-size can cure instability. This makes sense since we know that large step tend to cause instability and low accuracy, in the simulation, limiting the maximum-step size can prevent those effects from occurring.

Now we are also going to discuss briefly what happens when we increase the maximum step-size. We have let maximum step-size take the extreme value of 1000 and simulated the relative tolerance of 10^{-3}, the result is given in figure 29. We see that increasing the maximum step-size has not caused instability or any (or extremely small) decrease in the accuracy. This shows that increasing the maximum step-size in a stable and accurate simulation has almost no drawbacks.

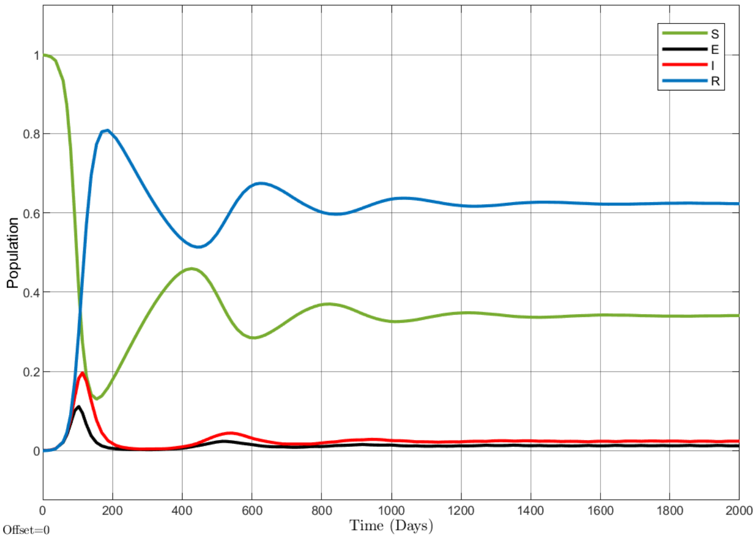


Figure 29: Epidemic Simulation with Simulink variable step size rel_tol = 10e-3 maxstep=1000

iii. Fixed Step Size

For the fixed step size we have run simulations on various fixed step sizes and observed the accuracy. The order 4 (ODE4) Runge-Kutta method was used in order to make comparison easier with our own implementation. For step sizes 1,10 and 12 days simulation results are given in figures 30, 31, 32 respectively.

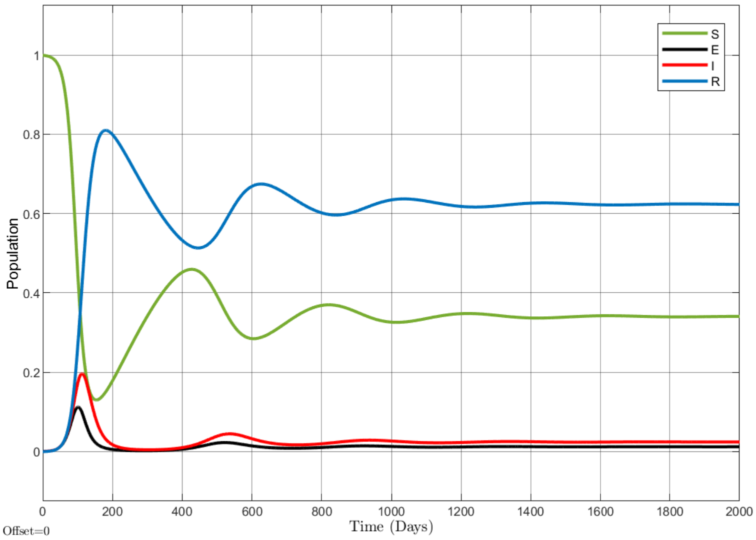


Figure 30: Epidemic Simulation with Simulink fixed step size h=1 days

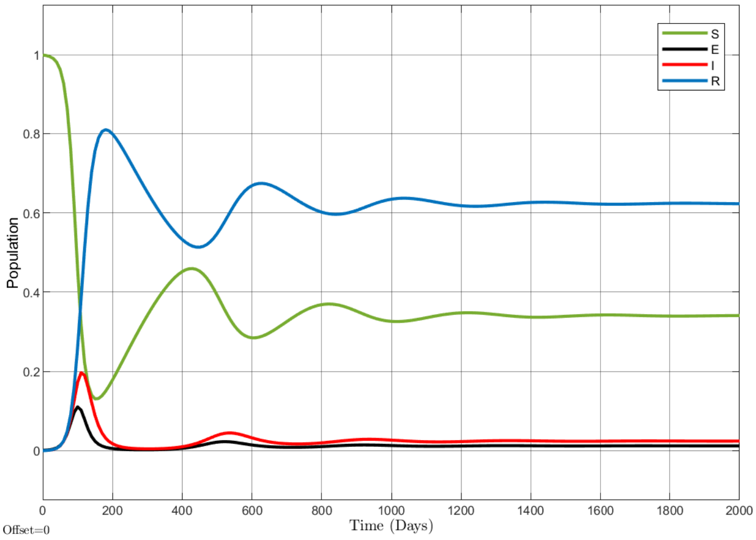


Figure 31: Epidemic Simulation with Simulink fixed step size h=10 days

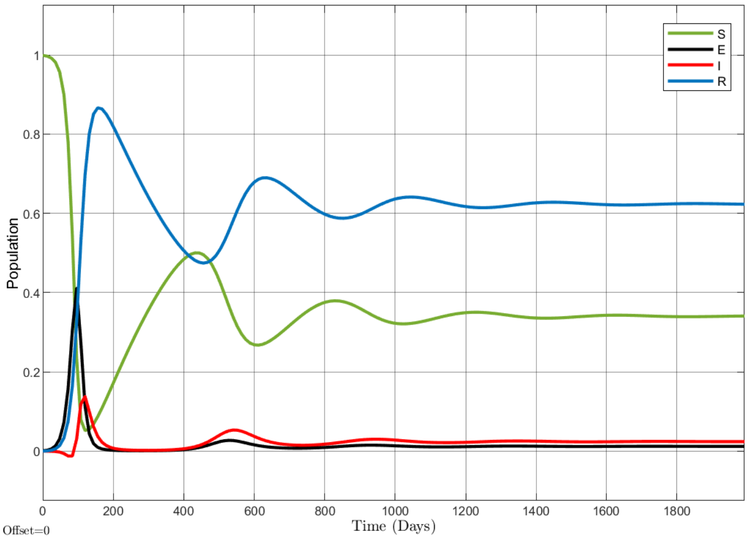


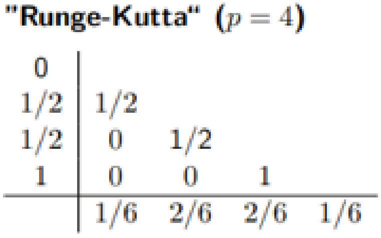
Figure 32: Epidemic Simulation with Simulink fixed step size h=12 days

We see that the results for h=1,10 days are sufficiently accurate and represent the previous plots, however h=12 days starts to break and has some decreased accuracy. This is expected since as step size increases the local error becomes less bound and hence it can take larger values. Moreover when we increase the step size to more than 13 the simulation gives error and becomes unstable, meaning that the large step size causes the simulation to get out of the region of stability. This also makes sense since we know that explicit methods are polynomials that occupy only finite regions and as we increase the step size we will eventually get out of the region of stability.

b. Our Implementation

i. Fixed Step Size

We implement the Runge-Kutta method of order 4 with fixed step size. The butcher tablau is given in figure



"Runge-Kutta" ($p = 4$)

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	2/6	2/6	1/6

Figure 33: Butcher Tablue of the Runge-Kutta method

We have selected a suitable step size of $h = 1$day since we knew from the previous part that this was a working step-size. The result is given in figure 34, we see that the result is very similar to the Simulink results and we managed to successfully simulate the epidemic.

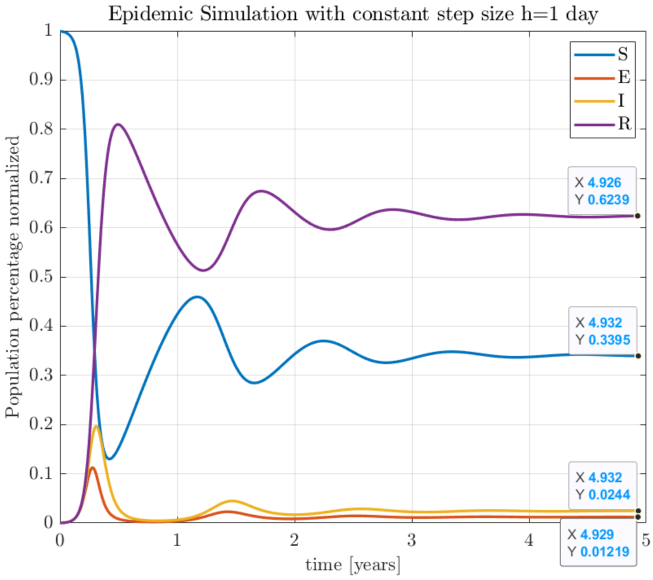
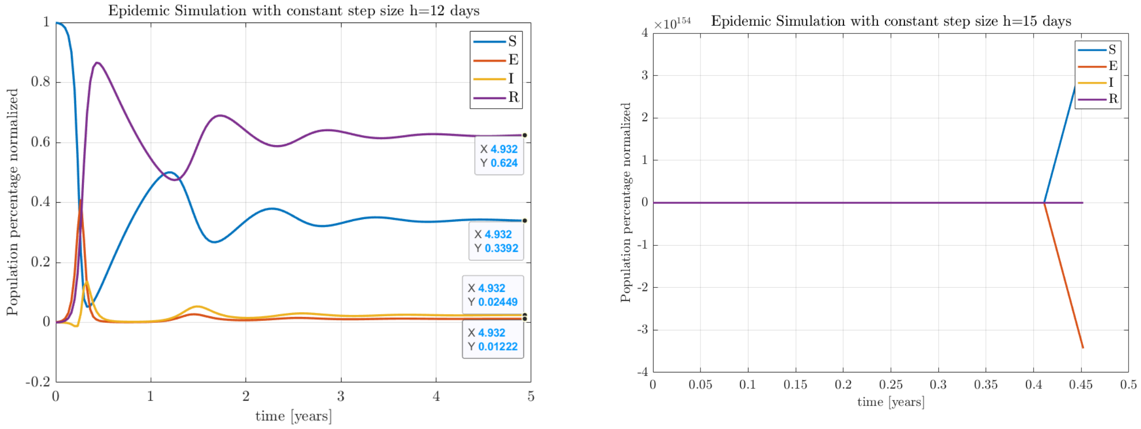


Figure 34: Epidemic Simulation with Runge-Kutta method of fixed step size h=1 day

We have also tried to simulate for other values as well. Simulations with step size of 12 days and 15 days are given in figures 35a and 35b. We see a similar distortion in the results as in Simulink's case when h=12 days and we can now see the diverging simulation in the unstable case of h=15 days.



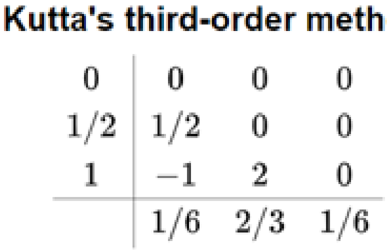
(a) h=12 days

(b) h=15 days

Figure 35: Epidemic Simulation with Runge-Kutta method of fixed step sizes

ii. Variable Step Size Theory

For the variable step size methods we closely followed the instructions given in Brian's book. We first require another solver of order 3 to work together with the Runge-Kutta method. For this purpose we selected the Kutta method with the Butcher tablau given in figure 36.



Kutta's third-order method

0	0	0	0
1/2	1/2	0	0
1	−1	2	0
	1/6	2/3	1/6

Figure 36: Butcher Tablue of the Kutta method

Let $y^(3)(t_k)$ and $y^{(4)}(t_k)$ denote the output at time t_k for the third and fourth order methods respectively. The error between them is calculated as:

$$|\eta_i| = |y_i^{(3)} - y_i^{(4)}|$$

We want this error to be smaller than some tolerance value in the 1 dimensional cases. However in our model we have a 4 dimensional state vector $y = \begin{bmatrix} S & E & I & R \end{bmatrix}$, and now we have 4 error, hence we need a variation of those errors. This is given as:

$$\sigma = \sqrt{\frac{1}{4}\sum_{i=1}^{4}(\frac{|\eta_i|}{\mathbf{atol} + \mathbf{rtol}|y_i^{(4)}|})^2} \tag{28}$$

This quantity measures if the error is small enough to keep the calculated y value. If ≤ 1 then the error is small enough and the calculated y value is taken and the time is updated with the current time step. If it is larger then 1 then the error is too large and step sizes must be made smaller before updating the time series.

The step update is done via the step update equation given as:

$$h_{i+1} = h_i \min(\mathbf{fac1}, \max(\mathbf{fac0}, \beta\sigma^{-\frac{1}{4}})) \tag{29}$$

This equation is used at every iteration regardless of the error.It makes the step size smaller when $\sigma > 1$, making sure the error in the next iteration becomes smaller and hopefully inside the tolerance range. When $\sigma \leq 1$ the step size gets larger making sure the simulation computation gets done faster in the given tolerance range. The min and max operations make sure that **fac0**$h_i \leq h_{i+1} \leq$ **fac1**h_i so that the step size doesn't change very rapidly. We have selected those parameters the recommended values by Brian in the book. Thus **fac0**=0.2, **fac1**=5 and $\beta = 0.9$. This way the step size can change up to 5 times at most.

iii. Variable Step Size Simulation

We have selected the absolute and relative tolerance values equal in our simulations for simplicity. For tolerance 10^{-5} we have obtained accurate results, as given in figure 37.

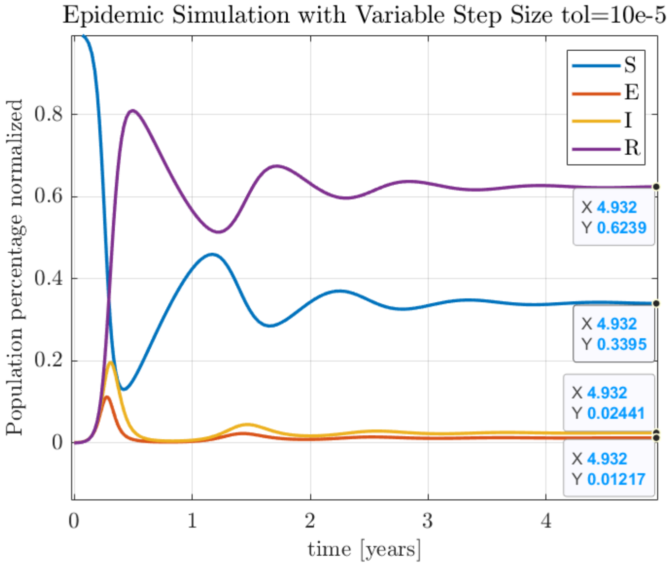
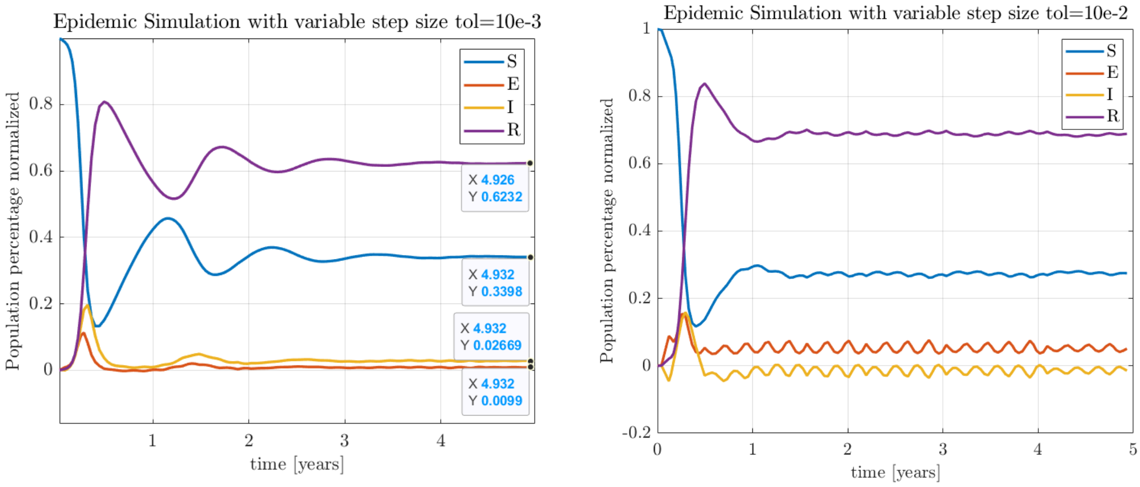


Figure 37: Epidemic Simulation with Runge-Kutta method (ode34) of variable step size and tolerance of 10e-5

Just like in the Simulink case we can test out larger tolerance values and see if the quality degrades, in figure 38 we see how the quality degrades when we increase the tolerance values. Moreover our method seems to get distorted faster but this is expected since in Simulink we used ODE45 and here we use ODE34 method which is a lower order method.



(a) tol=10e-3

(b) tol=10e-2

Figure 38: Epidemic Simulation with Runge-Kutta method with variable step-size (ode34)

iv. Time Comparison

The time elapsed for various configurations of the fixed and variable step methods are given in table 1. The step-size for the variable methods indicates the initial step-size. Looking at the results we clearly see a relation between the step-size and the time in fixed variable methods. Time obviously increases as step size decreases. On the other hand variable step size methods seem to converge to more or less same computation time. The computational time of the variable method seems to be about the same as the fixed size method with step equal to 1 day. Normally one can expect an even smaller time for the variable step size method but probably because

Method	Step-size (Initial)	Tolerance	Time
fixed	0.1	-	0.1348
fixed	1	-	0.0228
fixed	10	-	0.0095
variable	0.1	10e-5	0.0215
variable	0.1	10e-3	0.0225
variable	0.1	10e-2	0.0216
variable	1	10e-3	0.0219
variable	10	10e-3	0.0228

Table 1: Time Consumed for various configurations of the Simulation

it has larger computation for each time step such calculation of error variation (σ) and also using to solvers at once the computational cost is not way lower than all the fixed size methods but it is lower than the average.

v. Time Steps

In figure 39 we see the time step's taken at each iteration. We see at first the time step increases meaning that our error was small. Than at around 20 iteration the step size decreases meaning the error increased. Probably because of the fast change in the simulation at the beginning of the simulation. Later the step size increases again. The reason is as the infected population values start to settle down the simulation becomes slower hence a smaller step size is no longer needed. At the end it keeps bouncing of around the same values. The last one drop is to make sure t_f is aligned.

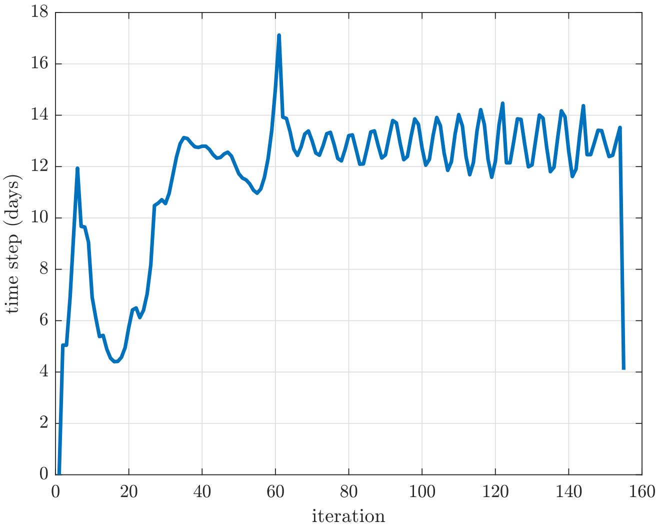


Figure 39: Time Steps of the Runge-Kutta method (ode34) of variable step size and tolerance of 10e-5