MOBILE ROBOT CONTROL
LAB # 1
University of Genova
EMARO+ 2015-2017

Razeen Hussain & Dipendra Subedi

OBJECTIVE:

The objective of this lab was to model the (2,0) and (1,1) wheeled mobile robots and demonstrate the robustness of the various control laws to tackle the problem of trajectory and posture tracking. For evaluation purposes, MATLAB was used to create simulations and to observe the effects of varying the parameters on the control laws.

INTRODUCTION:

The control of a wheeled mobile robot has been the focus of many research studies for the past few years, especially the presence of non-holonomic constraints have encouraged the development of non-linear control laws.

Mobile robots have been classified into various types but for the purpose of this lab, only the (2,0) and (1,1) types are analyzed. The (2,0) mobile robot, also called unicycle-type mobile robot, is comprised of two independently actuated wheels (usually fixed wheels) on a common axle linked rigidly to the mobile robot chassis. A passively orientable wheel (castor wheel) is also attached to the robot chassis to support the robot structure.

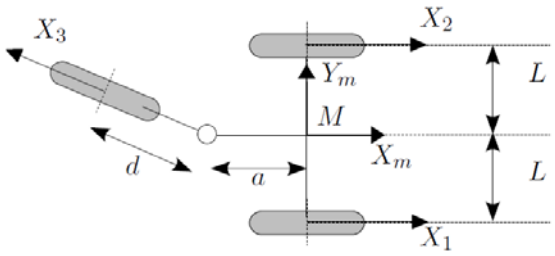
The (1,1) mobile robot, also called car-like mobile robot, is composed of a motorized wheeled-axle at the rear of the chassis and an orientable front wheel for steering purposes, however, other motorization schemes also exist for this class of mobile robots. Typically, two fixed wheels and a steerable wheel is used for this type of robot.

The goal of trajectory tracking is for a point, ‘P’ located on the chassis of the moving robot to follow a given curvature which evolves with time. The error that one has to minimize is the distance between the point ‘P’ and the curve at any given time, ‘t’. The same principle can be extended for posture tracking as well.

The lab first focuses on the development of the direct kinematic and inverse kinematic model of two classes of wheeled mobile robots and then analyzes the static, dynamic and lyapunov control schemes for trajectory tracking applied to the two robot classes.

KINEMATIC MODELING:

The schematic of the (2,0) mobile robot is shown below. The front direction of the robot is towards the two fixed wheels.



The table below shows the parameters of the (2,0) robot. These parameters are used to create the kinematic model of the mobile robot.

W	L	α	d	β	γ	φ	ψ
1f	*L*	*-π/2*	*0*	*0*	*π/2*	φ_{1f}	*0*
2f	*L*	*π/2*	*0*	*0*	*- π/2*	φ_{2f}	*0*
3c	*a*	*π*	*d*	β_{3c}	*0*	φ_{3c}	$\pi+\beta_{3c}$

The configuration vector is as follows, where x, y, θ are the coordinates of the robot frame M in the absolute reference frame.

$$q = (x \quad y \quad \theta \quad \beta_{3c} \quad \varphi_{1f} \quad \varphi_{2f} \quad \varphi_{3c})^T$$

From this, we can derive the configuration kinematic model as shown below:

$$J_1 = \begin{pmatrix} 1 & 0 & L \\ 1 & 0 & -L \\ -\cos\beta_{3c} & -\sin\beta_{3c} & a\sin\beta_{3c} \end{pmatrix}$$

$$J_2 = \begin{pmatrix} -r & 0 & 0 \\ 0 & -r & 0 \\ 0 & 0 & -r \end{pmatrix}$$

$$C_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ \sin\beta_{3c} & -\cos\beta_{3c} & d + a\cos\beta_{3c} \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix}$$

$$\Sigma(\beta_s) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Since the base of the *Ker*(C_1^*) (shown above) has 2 columns. The degree of mobility of the (2,0) mobile robot is $\delta_m = 2$. And since the steerable wheel does not exist so the degree of steerability is $\delta_s = 0$.

Taking, the linear and angular velocities as the inputs to the system, we can derive the system equation as:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} V \\ \omega \end{pmatrix}$$

With this the configuration kinematic model can be established as follows:

$$\begin{pmatrix} \dot{\varphi}_{1f} \\ \dot{\varphi}_{2f} \\ \dot{\varphi}_{3c} \\ \dot{\beta}_{3c} \end{pmatrix} = \begin{pmatrix} -\frac{\sin\beta_{3c}}{d} & -\frac{d + a\cos\beta_{3c}}{d} \\ \frac{1}{r} & \frac{L}{r} \\ \frac{1}{r} & -\frac{L}{r} \\ -\frac{\cos\beta_{3c}}{r} & \frac{a\sin\beta_{3c}}{r} \end{pmatrix} \begin{pmatrix} V \\ \omega \end{pmatrix}$$

Since the $\delta_m = 2$, two wheels have to be motorized by selecting two independent rows from the above equation. For our case, we chose to motorize the two fixed wheels which is also the standard way for the (2,0) robot. This motorization ensures that there is no singularity in the system, as the determinant of the middle two rows of the matrix is non-zero.

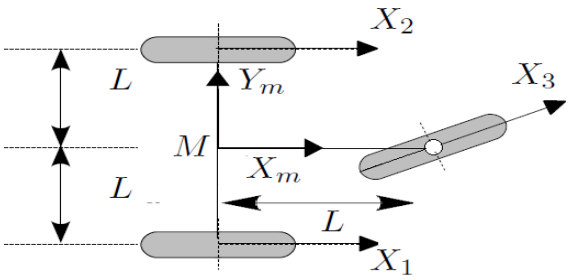
We now express the velocities of the motorized joints in terms of our input vector. This is referred to as the inverse kinematic model.

$$\begin{pmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{r} & \frac{L}{r} \\ \frac{1}{r} & -\frac{L}{r} \end{pmatrix} \begin{pmatrix} V \\ \omega \end{pmatrix}$$

The inverse of this gives the direct kinematic model.

$$\begin{pmatrix} V \\ \omega \end{pmatrix} = \begin{pmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2L} & -\frac{r}{2L} \end{pmatrix} \begin{pmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{pmatrix}$$

Similarly, we can extend this approach for the kinematic modeling of the (1,1) mobile robot whose schematic is shown below:



The parameter table for the (1,l) robot is as follows:

W	L	α	d	β	γ	φ	ψ
1f	*L*	*-π/2*	*0*	*0*	*π/2*	φ_{1f}	*0*
2f	*L*	*π/2*	*0*	*0*	*- π/2*	φ_{2f}	*0*
3s	*L*	*0*	*d*	β_{3s}	*0*	φ_{3s}	β_{3s}

The configuration vector is:

$$q = (x \quad y \quad \theta \quad \beta_{3s} \quad \varphi_{1f} \quad \varphi_{2f} \quad \varphi_{3s})^T$$

Likewise, now we can derive the configuration kinematic model for the (1,1) mobile robot as shown below:

$$J_1 = \begin{pmatrix} 1 & 0 & L \\ 1 & 0 & -L \\ \cos\beta_{3s} & \sin\beta_{3s} & L\sin\beta_{3s} \end{pmatrix}$$

$$J_2 = \begin{pmatrix} -r & 0 & 0 \\ 0 & -r & 0 \\ 0 & 0 & -r \end{pmatrix}$$

$$C_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ -\sin\beta_{3s} & \cos\beta_{3s} & L\cos\beta_{3s} \end{pmatrix}$$

$$\Sigma(\beta_s) = \begin{pmatrix} 1 \\ 0 \\ \frac{\tan\beta_{3s}}{L} \end{pmatrix}$$

Since the base of the *Ker*(C_1^*) (shown above) has only 1 column. The degree of mobility of the (1,1) mobile robot is $\delta_m = 1$. And since there exist one steerable wheel in our model so the degree of steerability is $\delta_s = 1$.

Taking, the linear and angular velocities as the inputs to the system, we can derive the system equation as:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\beta_{3s}} \end{pmatrix} = \left(\begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ \frac{\tan\beta_{3s}}{L} & 0 \\ 0 & 1 \end{bmatrix} or \begin{bmatrix} \cos\theta\cos\beta_{3s} & 0 \\ \sin\theta\cos\beta_{3s} & 0 \\ \frac{\sin\beta_{3s}}{L} & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{pmatrix} V \\ \dot{\beta_{3s}} \end{pmatrix}$$

With this the configuration kinematic model can be established as follows:

$$\dot{\varphi} = \frac{1}{r} \left(\begin{bmatrix} 1 + \tan\beta_{3s} \\ 1 - \tan\beta_{3s} \\ \cos\beta_{3s} + \sin\beta_{3s}\tan\beta_{3s} \end{bmatrix} or \begin{bmatrix} \cos\beta_{3s} + \sin\beta_{3s} \\ \cos\beta_{3s} - \sin\beta_{3s} \\ 1 \end{bmatrix} \right) u_m$$

Since the $\delta_m = 1$, one wheel has to be motorized. For our case, we chose to motorize the angular rotation of the steerable wheel. However, as we can see that due to the presence of the $tan\beta_{3s}$ term in our model, there exist singularities. To tackle this, we can multiply our equations with $cos\beta_{3s}$ to avoid this. That is why, there are two values for the matrix present in the model.

We now express the velocities of the motorized joints in terms of our input vector. This is referred to as the inverse kinematic model.

$$\begin{pmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{pmatrix} = \begin{pmatrix} \dot{\varphi_{3s}} \\ \dot{\beta_{3s}} \end{pmatrix} = \left(\begin{bmatrix} \frac{\cos\beta_{3s} + \sin\beta_{3s}\tan\beta_{3s}}{r} & 0 \\ 0 & 1 \end{bmatrix} or \begin{bmatrix} \frac{1}{r} & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{pmatrix} V \\ \dot{\beta_{3s}} \end{pmatrix}$$

The inverse of this gives the direct kinematic model.

$$\begin{pmatrix} V \\ \dot{\beta_{3s}} \end{pmatrix} = \left(\begin{bmatrix} \frac{r}{\cos\beta_{3s} + \sin\beta_{3s}\tan\beta_{3s}} & 0 \\ 0 & 1 \end{bmatrix} or \begin{bmatrix} r & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{pmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{pmatrix}$$

For our model, we will use the following values for the constants in the system:

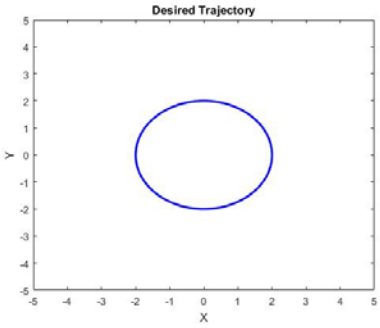
- r = 0.1m
- L = 0.13m

DESIRED TRAJECTORY, INITIAL POSITION AND SIMULATION CASES:

For simulation purposes, we will try to make a point 'P' on the mobile robot follow a circular trajectory of radius R = 2 and angular velocity ω_d = 0.5 rad/s. Thus the time variation of the input trajectory is shown below:

$$h_d = \begin{pmatrix} 2\cos(0.5t) \\ 2\sin(0.5t) \end{pmatrix}$$

The plot of the input trajectory is shown below:



The initial position of the mobile is as follows:

- x_0 = 2.3m
- y_0 = 0m
- $\theta_0 = \pi$ rad

The simulations made consider three different cases which are:

1) the values of 'r' and 'L' are fully known
2) there is a 10% error (alpha=0.1) in the values of 'r' and 'L'
3) there is a -10% error (alpha=-0.1) in the values of 'r' and 'L'

The simulations run for 26 seconds which is approximately equal to the robot making two complete circles.

STATIC DECOUPLING CONTROL:

For the case of the (2,0) mobile robot, we will control the following point on the robot. This point is expressed in the absolute reference frame.

$$h = \begin{pmatrix} x + d\cos\theta \\ y + d\sin\theta \end{pmatrix}$$

We can differentiate this posture coordinates to get a relation between the posture coordinate velocities and our input vector as follows:

$$\dot{h} = \begin{pmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \end{pmatrix} \begin{pmatrix} V \\ \omega \end{pmatrix}$$

Taking the inverse, we can get our control law.

$$\begin{pmatrix} V \\ \omega \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\frac{\sin\theta}{d} & \frac{\cos\theta}{d} \end{pmatrix} W$$

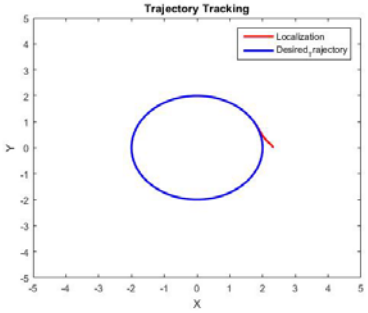
The input to the control law is an auxiliary control command, 'W' which can be expressed as shown below:

$$W = \dot{h_d} + K_p(h_d - h)$$

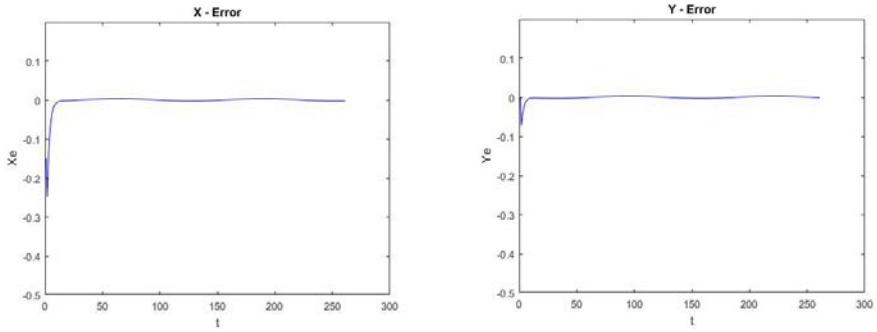
The system equations ensure asymptotic stability and this is also demonstrated by the error graphs in the simulations. In the static decoupling control, we are not controlling the orientation of the robot.

The simulation results of the three discussed cases for the (2,0) robot are shown below. Also, a special case is also included where the robot was given a fixed point to reach.

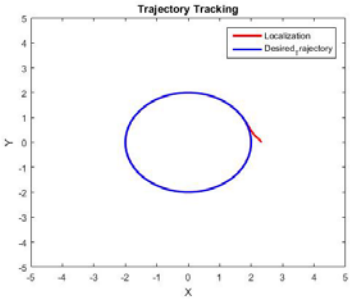
- **CASE 1 (2,0) Static Control**
  The output curve can be seen below. It can be observed that robot follows the desired trajectory.



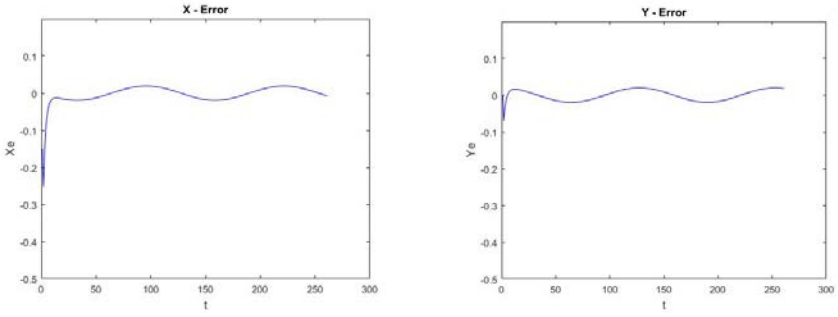
  The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.



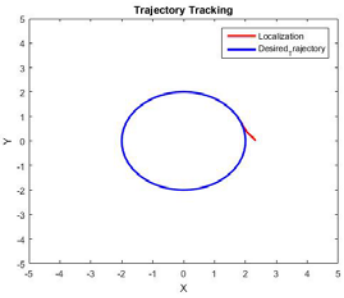
- **CASE 2 (2,0) Static Control**
  The output curve can be seen below. It can be observed that robot follows the desired trajectory.



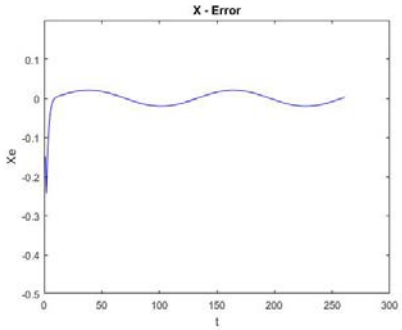
  The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.

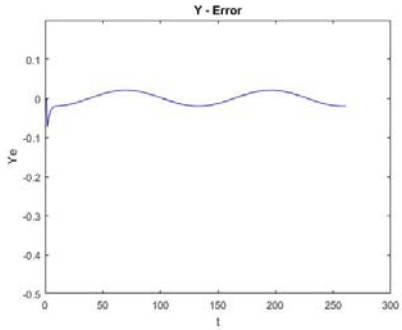


- **CASE 3 (2,0) Static Control**
  The output curve can be seen below. It can be observed that robot follows the desired trajectory.

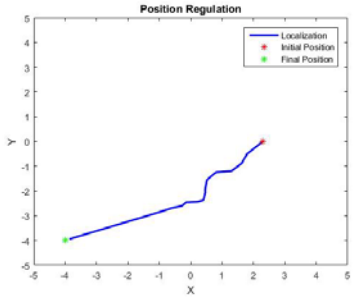


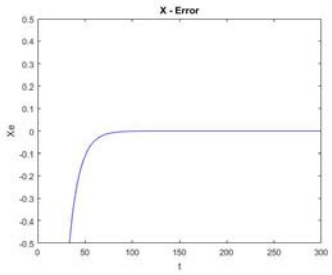
  The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.

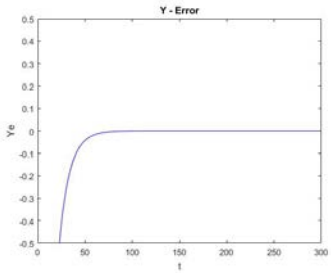




- **CASE 4 (2,0) Static Control for a Fixed Point**
  The robot was to move to a final point of [-4,-4] and the movement was observed. The output curve can be seen below. It can be observed that robot reaches near the final point but there is a steady state error in the system. The error curves are also shown.







The control law established is robust enough as there is minimal change in output as the value of alpha varies.

Similarly for the case of the (1,1) mobile robot, we will control the following point on the robot. This point is expressed in the absolute reference frame.

$$h = \begin{pmatrix} x + L\cos\theta + d\cos(\theta + \beta_{3s}) \\ y + L\sin\theta + d\sin(\theta + \beta_{3s}) \end{pmatrix}$$

We can differentiate this posture coordinates to get a relation between the posture coordinate velocities and our input vector as follows:

$$\dot{h} = \begin{pmatrix} \cos\theta - \sin\theta\tan\beta_{3s} - \dfrac{d}{L}\sin(\theta + \beta_{3s})\tan\beta_{3s} & -d\sin(\theta + \beta_{3s}) \\ \sin\theta + \cos\theta\tan\beta_{3s} + \dfrac{d}{L}\cos(\theta + \beta_{3s})\tan\beta_{3s} & d\cos(\theta + \beta_{3s}) \end{pmatrix} \begin{pmatrix} V \\ \dot{\beta}_{3s} \end{pmatrix}$$

Taking the inverse, we can get our control law.

$$\begin{pmatrix} V \\ \dot{\beta}_{3s} \end{pmatrix} = \begin{pmatrix} \cos\beta_{3s}\cos(\theta+\beta_{3s}) & \cos\beta_{3s}\sin(\theta+\beta_{3s}) \\ \frac{-\sin(\theta+\beta_{3s})}{d} + \frac{\cos(\theta+\beta_{3s})\sin\beta_{3s}}{L} & \frac{\cos(\theta+\beta_{3s})}{d} - \frac{\sin(\theta+\beta_{3s})\sin\beta_{3s}}{L} \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}$$

The input to the control law is an auxiliary control command, 'W' which can be expressed as shown below. This is same as for the case of (2,0) robot.

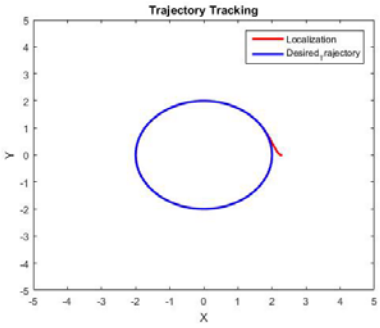
$$W = \dot{h_d} + K_p(h_d - h)$$

The system equations for the (1,1) mobile robot also ensure asymptotic stability and this is demonstrated by the error graphs in the simulations. In the static decoupling control, we are not controlling the orientation of the robot similar to the (2,0) case.

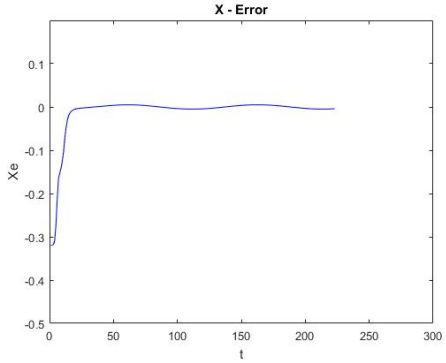
The simulation results of the three discussed cases for the (1,1) robot are shown below.

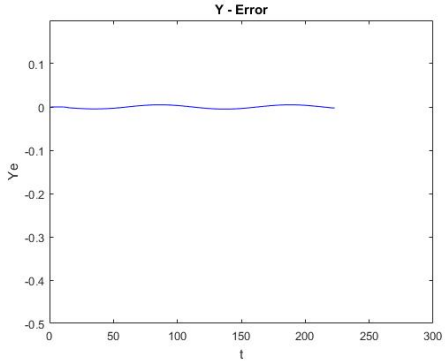
- **CASE 1 (1,1) Static Control**

  The output curve can be seen below. It can be observed that robot follows the desired trajectory.



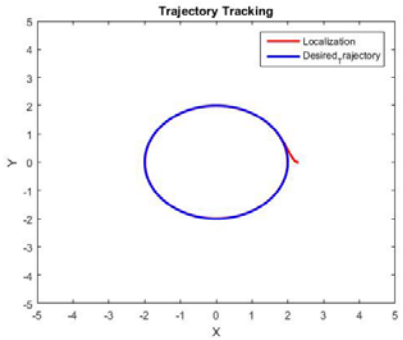
  The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.



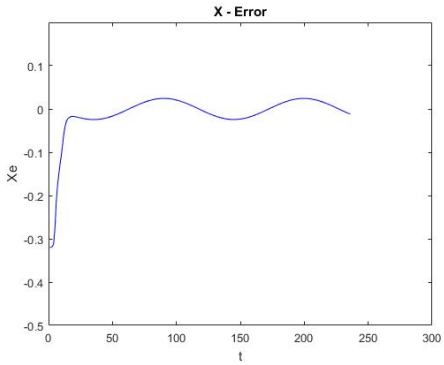


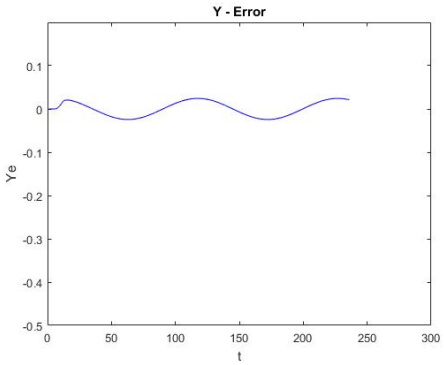
- **CASE 2 (1,1) Static Control**

  The output curve can be seen below. It can be observed that robot follows the desired trajectory.



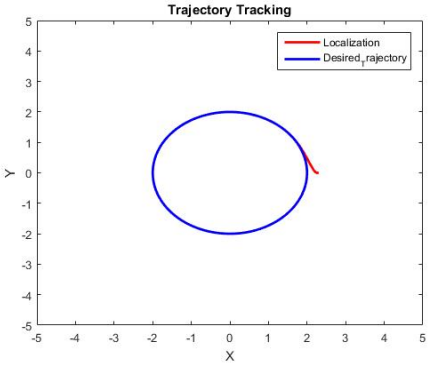
The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.



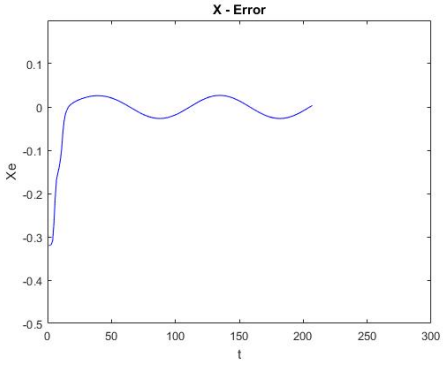


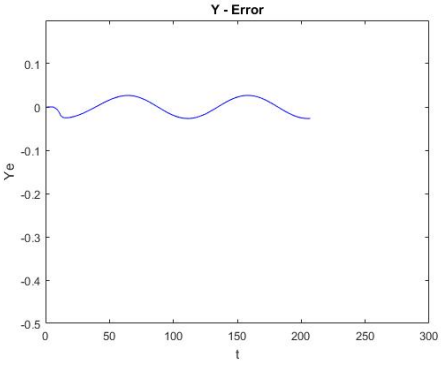
- **CASE 3 (1,1) Static Control**

The output curve can be seen below. It can be observed that robot follows the desired trajectory.



The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.





The control law established is robust enough as there is minimal change in output as the value of alpha varies.

DYNAMIC DECOUPLING CONTROL:

For the case of the (2,0) mobile robot, we will control the following point on the robot. This point is expressed in the absolute reference frame.

$$h = \begin{pmatrix} x \\ y \end{pmatrix}$$

We can differentiate this posture coordinates twice to get a relation between the posture coordinate velocities and our input vector as follows:

$$\ddot{h} = \begin{pmatrix} \cos\theta & -V\sin\theta \\ \sin\theta & V\cos\theta \end{pmatrix} \begin{pmatrix} \dot{V} \\ \omega \end{pmatrix}$$

Taking the inverse, we can get our control law.

$$\begin{pmatrix} \dot{V} \\ \omega \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\dfrac{\sin\theta}{V} & \dfrac{\cos\theta}{V} \end{pmatrix} \ddot{h}$$

The input to the control law is an auxiliary control command, 'W' which can be expressed as shown below:

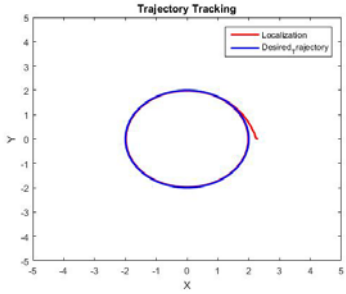
$$W = \ddot{h_d} + K_d(\dot{h_d} - \dot{h}) + K_p(h_d - h)$$

The system equations ensure asymptotic stability and this is also demonstrated by the error graphs in the simulations. In the static decoupling control, we are not controlling the orientation of the robot.

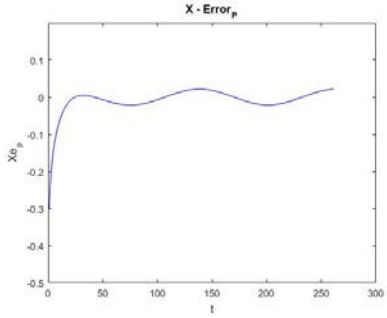
The simulation results of the three discussed cases for the (2,0) robot are shown below.

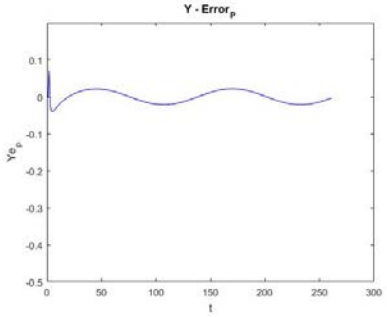
- **CASE 1 (2,0) Dynamic Control**

  The output curve can be seen below. It can be observed that robot follows the desired trajectory.

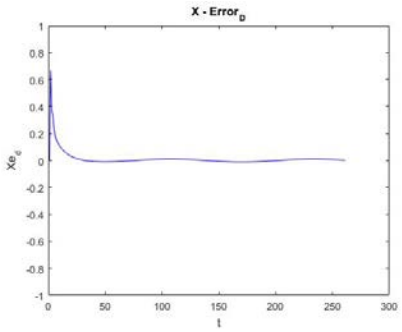


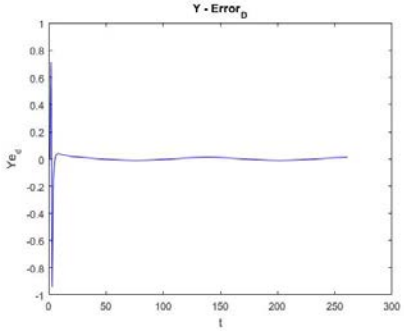
  The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.





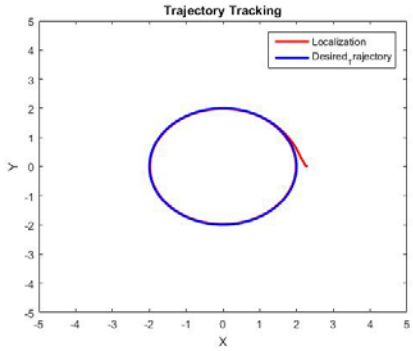
  The error in the linear velocities of the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.



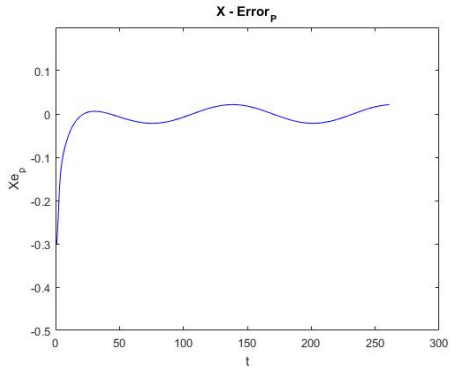


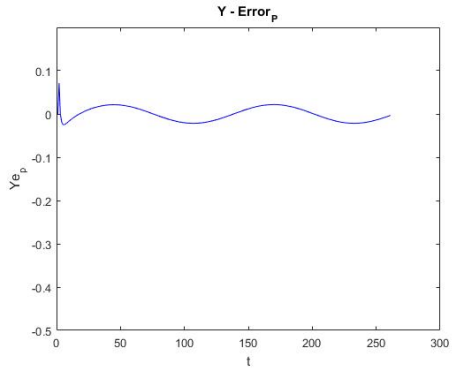
- **CASE 2 (2,0) Dynamic Control**

The output curve can be seen below. It can be observed that robot follows the desired trajectory.

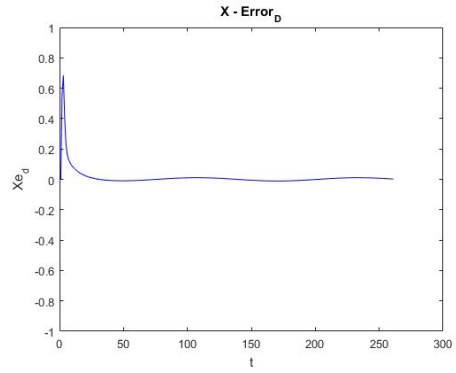


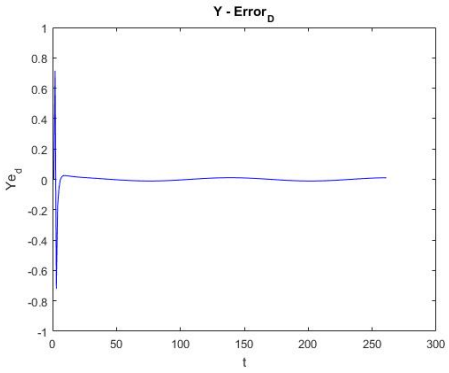
The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.





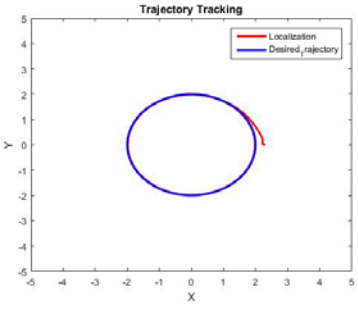
The error in the linear velocities of the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.



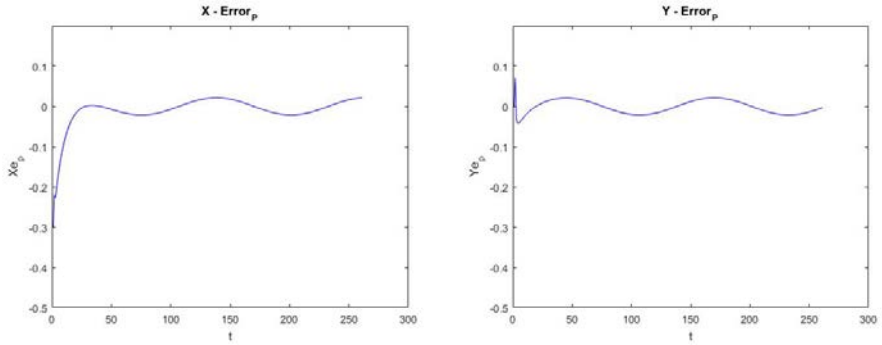


- **CASE 3 (2,0) Dynamic Control**

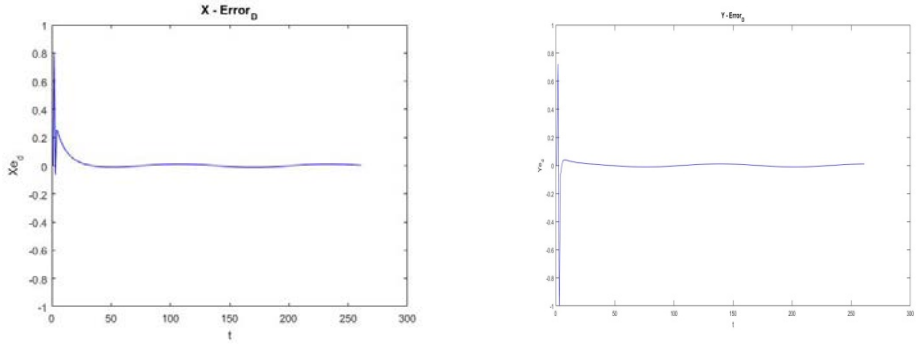
The output curve can be seen below. It can be observed that robot follows the desired trajectory.



The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.



The error in the linear velocities of the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.



The control law established is robust enough as there is minimal change in output as the value of alpha varies.

Similarly for the case of the (1,1) mobile robot, we will control the following point on the robot. This point is expressed in the absolute reference frame.

$$h = \begin{pmatrix} x + d\cos\theta \\ y + d\sin\theta \end{pmatrix}$$

We can differentiate this posture coordinates twice to get a relation between the posture coordinate velocities and our input vector as follows:

$$\ddot{h} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} = K \begin{pmatrix} \dot{V} \\ \dot{\beta}_{3s} \end{pmatrix} + \gamma$$

$$K = \begin{pmatrix} \cos\theta\cos\beta_{3s} - \frac{d}{L}\sin\theta\sin\beta_{3s} & -\frac{d\,V}{L}\sin\theta\cos\beta_{3s} - V\cos\theta\sin\beta_{3s} \\ \sin\theta\cos\beta_{3s} + \frac{d}{L}\cos\theta\sin\beta_{3s} & \frac{d\,V}{L}\cos\theta\cos\beta_{3s} - V\sin\theta\sin\beta_{3s} \end{pmatrix}$$

$$\gamma = V^2 \begin{pmatrix} -\frac{1}{L}\sin\theta\cos\beta_{3s}\sin\beta_{3s} - \frac{d}{L^2}\cos\theta\sin^2\beta_{3s} \\ \frac{1}{L}\cos\theta\cos\beta_{3s}\sin\beta_{3s} - \frac{d}{L^2}\sin\theta\sin^2\beta_{3s} \end{pmatrix}$$

Taking the inverse, we can get our control law.

$$\begin{pmatrix} \dot{V} \\ \dot{\beta}_{3s} \end{pmatrix} = K^{-1}(\ddot{h} - \gamma)$$

$$K^{-1} = \begin{pmatrix} \cos\theta\cos\beta_{3s} - \frac{L}{d}\sin\theta\sin\beta_{3s} & \sin\theta\cos\beta_{3s} + \frac{L}{d}\cos\theta\sin\beta_{3s} \\ -\frac{L}{d\,V}\sin\theta\cos\beta_{3s} - \frac{1}{V}\cos\theta\sin\beta_{3s} & \frac{L}{d\,V}\cos\theta\cos\beta_{3s} - \frac{1}{V}\sin\theta\sin\beta_{3s} \end{pmatrix}$$

The input to the control law is an auxiliary control command, 'W' which can be expressed as shown below. This is same as for the case of (2,0) robot.

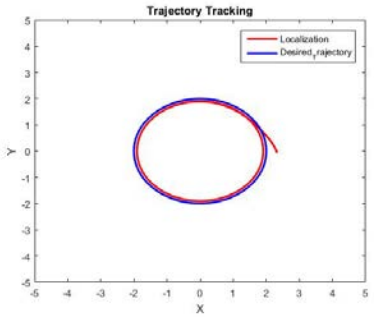
$$W = \ddot{h_d} + K_d(\dot{h_d} - \dot{h}) + K_p(h_d - h)$$

The system equations ensure asymptotic stability and this is also demonstrated by the error graphs in the simulations. In the static decoupling control, we are not controlling the orientation of the robot.

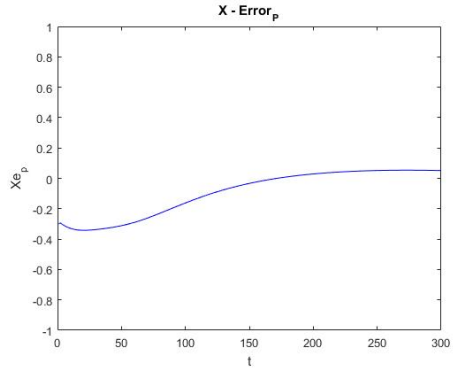
The simulation results of the three discussed cases for the (1,1) robot are shown below.

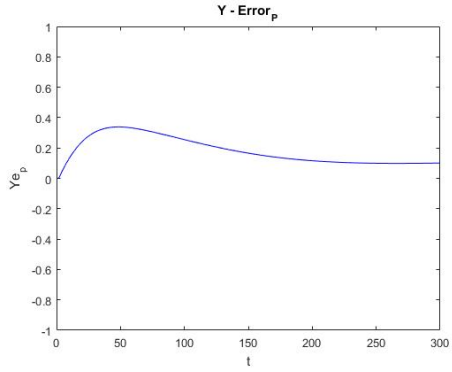
- **CASE 1 (1,1) Dynamic Control**

  The output curve can be seen below. It can be observed that robot follows the desired trajectory.

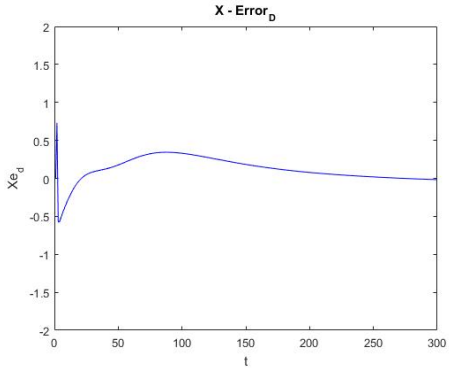


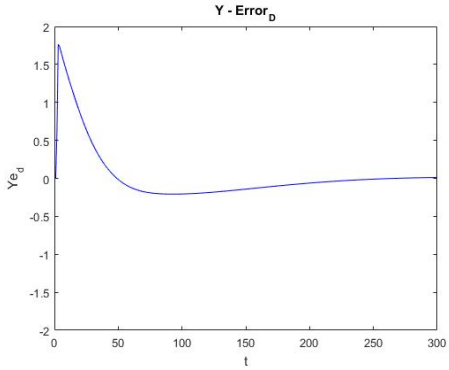
The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.





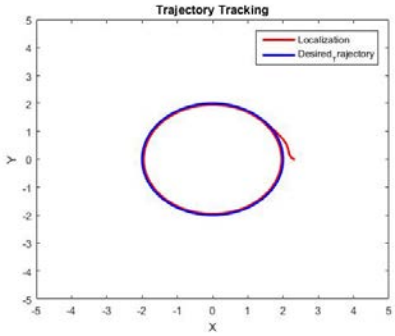
The error in the linear velocities of the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.



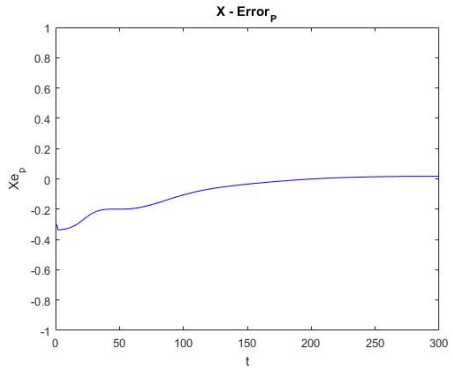


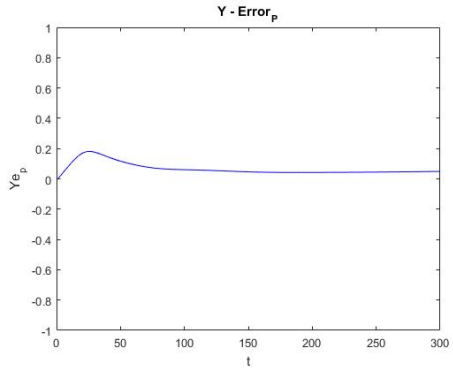
- **CASE 2 (1,1) Dynamic Control**

The output curve can be seen below. It can be observed that robot follows the desired trajectory.

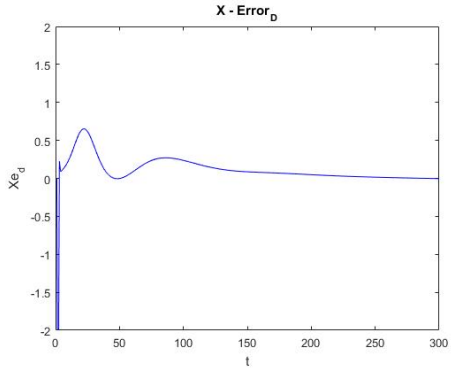


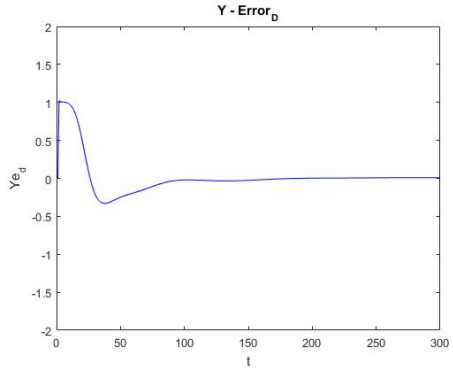
The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.





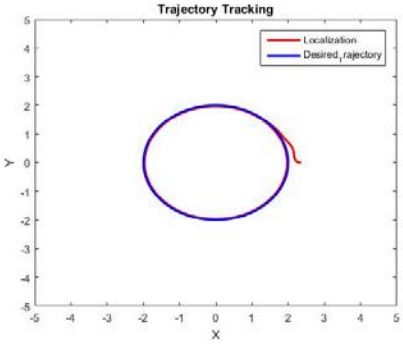
The error in the linear velocities of the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.



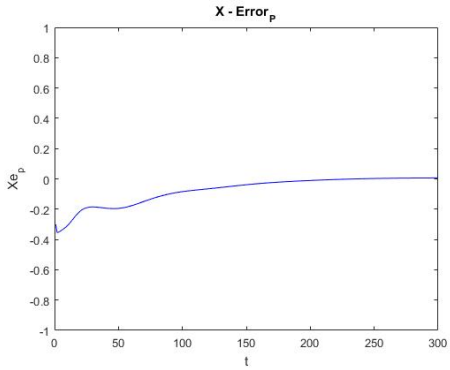


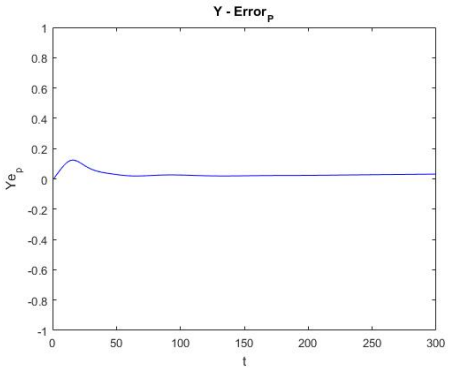
- **CASE 3 (1,1) Dynamic Control**

The output curve can be seen below. It can be observed that robot follows the desired trajectory.

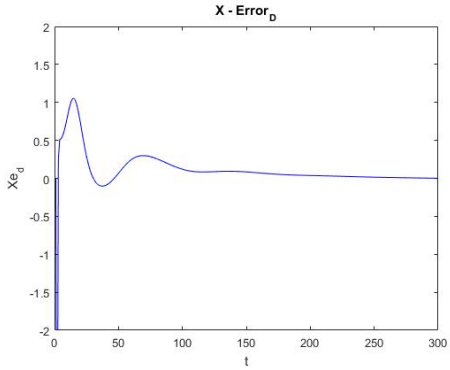


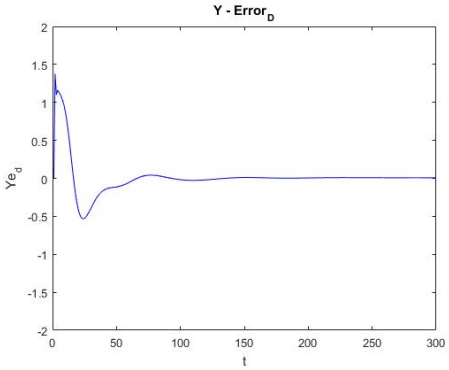
The error in the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.





The error in the linear velocities of the x and y coordinates can be seen below. Both errors converge to a stable value, thus ensuring asymptotic stability.





The control law established is robust enough as it follows the desired trajectory regardless of the value of alpha. However, tuning the gains was hard for the dynamic control of (1,1) robot. There was a tradeoff between the time for the robot to reach the desired radius circle and the smoothness of the trajectory.

LYAPUNOV CONTROL LAW:

We will consider the following posture error:

$$\xi^e = \xi^d - \xi$$

This can be expressed in the robot frame 'M' as shown below:

$${}^m\xi_e = \begin{pmatrix} {}^m x_e \\ {}^m y_e \\ {}^m \theta_e \end{pmatrix} = {}^m R_0 \, ({}^0\xi_d - {}^0\xi)$$

The control error can be given by:

$$u_e = u_d - u$$

By differentiating w.r.t. time, we can obtain the following expression. Here for the case of the (2,0) robot the inputs are the linear and angular velocities.

$$\begin{pmatrix} \dot{x_e} \\ \dot{y_e} \\ \dot{\theta_e} \end{pmatrix} = \begin{pmatrix} 0 & \omega_d & V_d \dfrac{\cos\theta_e - 1}{\theta_e} \\ -\omega_d & 0 & V_d \dfrac{\sin\theta_e}{\theta_e} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ \theta_e \end{pmatrix} + \begin{pmatrix} 1 & -y_e \\ 0 & x_e \\ 0 & 1 \end{pmatrix} \begin{pmatrix} V_e \\ \omega_e \end{pmatrix}$$

Thus, the control law can be defined as:

$$u = \begin{pmatrix} V \\ \omega \end{pmatrix} = \begin{pmatrix} V_d \cos {}^m\theta_e + K_x {}^m x_e \\ \omega_d + K_\theta {}^m\theta_e + K_y V_d {}^m y_e \dfrac{\sin {}^m\theta_e}{{}^m\theta_e} \end{pmatrix}$$

For the case of the (1,1) robot, the same formulas can be used but with one substitution and the control block outputs are the linear velocity and the steering angle, 'β_{3s}'

$$\beta_{3s} = \sin^{-1}\left(\frac{L\omega}{V}\right)$$

The Lyapunov function used to derive this expression is:

$$W = \frac{{}^m x_e^2 + {}^m y_e^2 + \dfrac{{}^m\theta_e^2}{K_y}}{2}$$

This is a Lyapunov function since it is always positive definite due to the presence of the square terms (W > 0). Its derivative is as follows:

$$\dot{W} = {}^m x_e {}^m\dot{x}_e + {}^m y_e {}^m\dot{y}_e + \frac{{}^m\theta_e {}^m\dot{\theta}_e}{K_y}$$

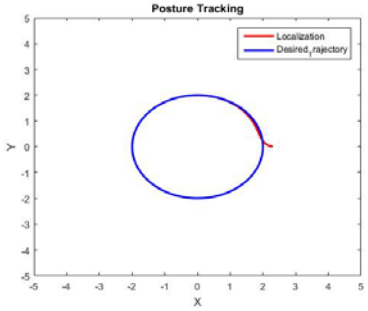
$$\dot{W} = -K_x {}^m x_e^2 - \frac{K_\theta {}^m\theta_e^2}{K_y}, \qquad {}^m y_e = 0 \; when \; V \neq 0$$

This ensures that $\dot{W}$ is negative definite, but it can be 0 as well so it is semi-negative definite and locally stable in terms of Lyapunov.

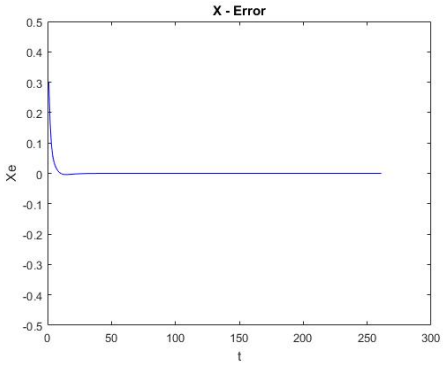
The simulation results of the three cases for the (2,0) and (1,1) robots are shown below.

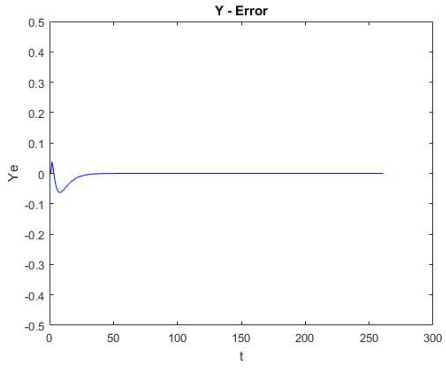
- **CASE 1 (2,0) Lyapunov Control**

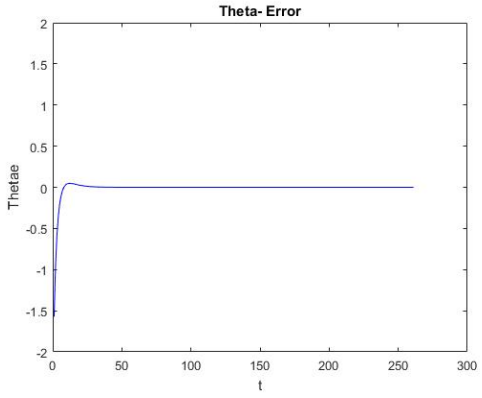
  The output curve can be seen below. It can be observed that robot follows the desired trajectory.



  The error in the x, y and θ can be seen below. All errors converge to a stable value, thus ensuring asymptotic stability.

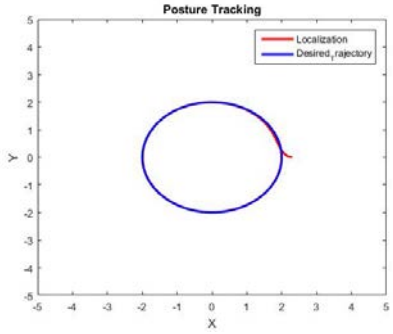




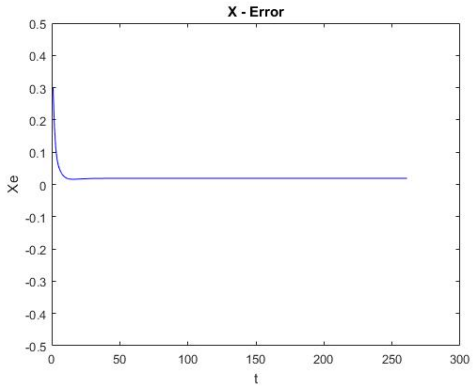


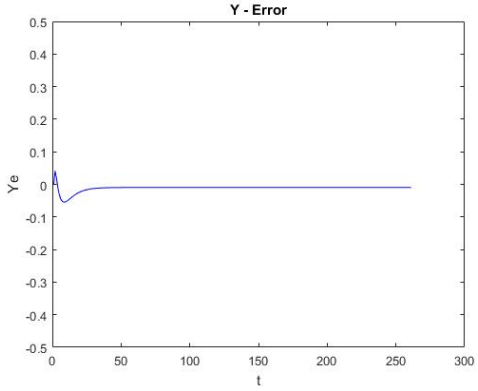
- **CASE 2 (2,0) Lyapunov Control**

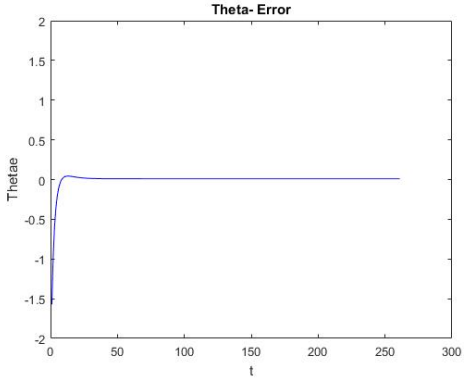
The output curve can be seen below. It can be observed that robot follows the desired trajectory.



The error in the x, y and θ can be seen below. All errors converge to a stable value, thus ensuring asymptotic stability.

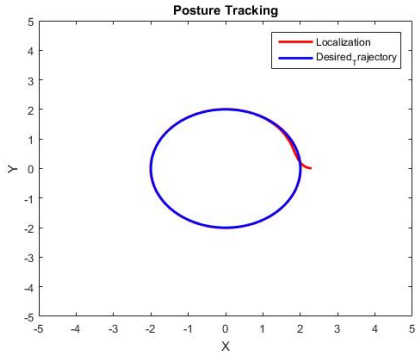




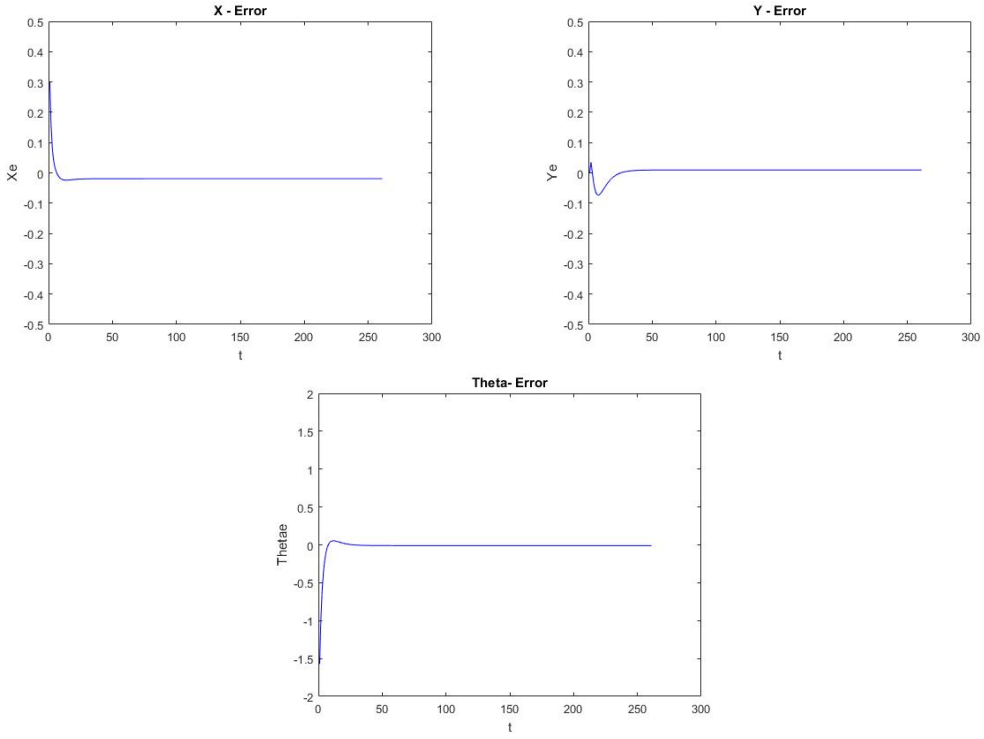


- **CASE 3 (2,0) Lyapunov Control**

The output curve can be seen below. It can be observed that robot follows the desired trajectory.

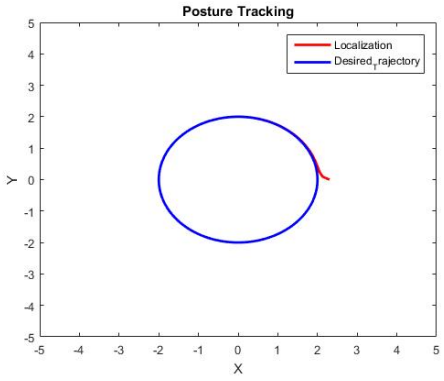


The error in the x, y and θ can be seen below. All errors converge to a stable value, thus ensuring asymptotic stability.

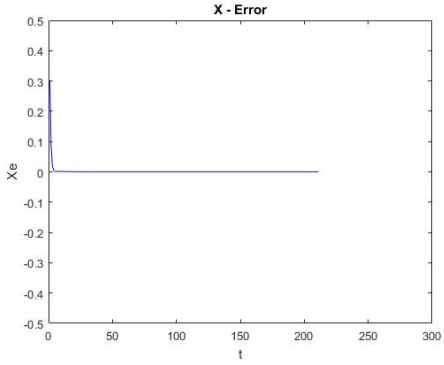


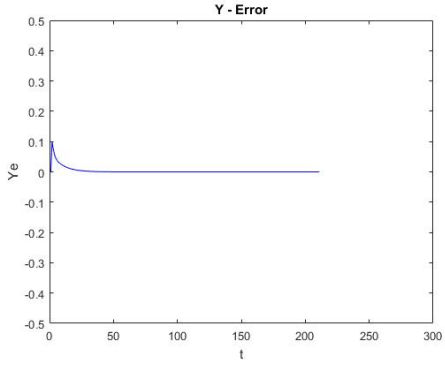
- **CASE 1 (1,1) Lyapunov Control**

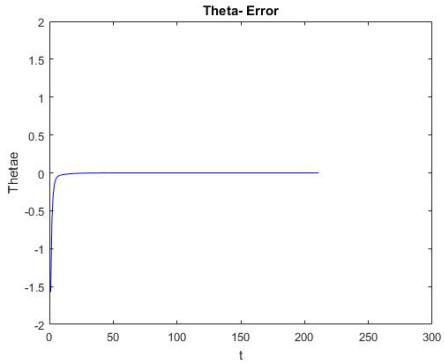
The output curve can be seen below. It can be observed that robot follows the desired trajectory.



The error in the x, y and θ can be seen below. All errors converge to a stable value, thus ensuring asymptotic stability.

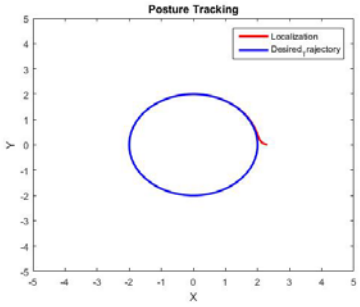




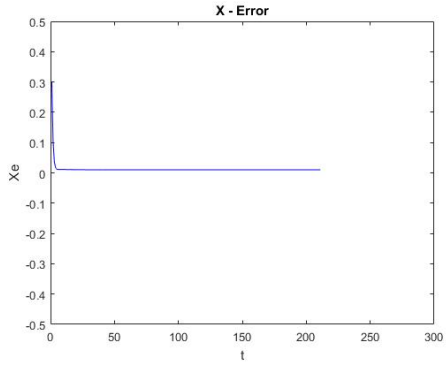


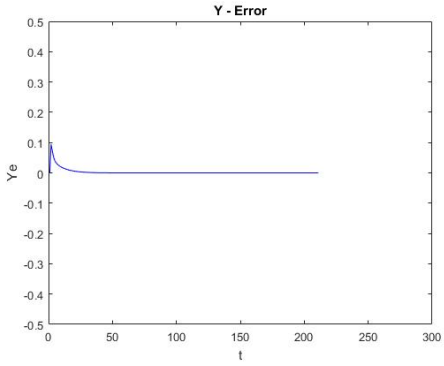
- **CASE 2 (1,1) Lyapunov Control**

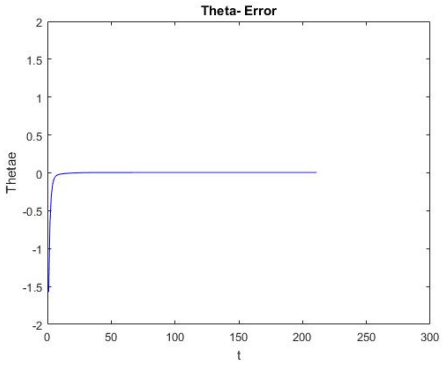
The output curve can be seen below. It can be observed that robot follows the desired trajectory.



The error in the x, y and θ can be seen below. All errors converge to a stable value, thus ensuring asymptotic stability.

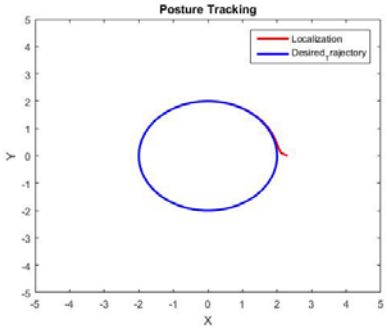




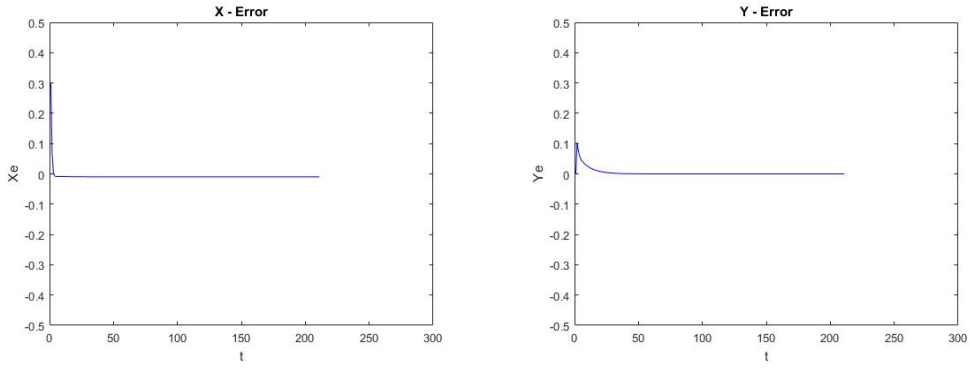


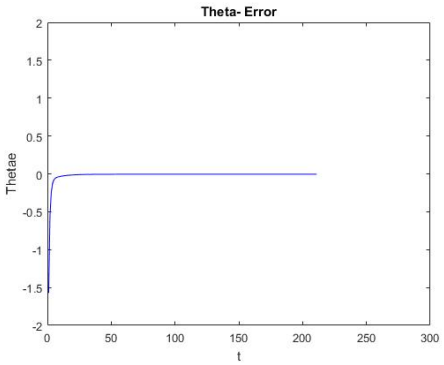
- **CASE 3 (1,1) Lyapunov Control**

  The output curve can be seen below. It can be observed that robot follows the desired trajectory.



The error in the x, y and θ can be seen below. All errors converge to a stable value, thus ensuring asymptotic stability.





The control law established is the most robust among the three control strategies as the convergence of the error to zero is best in Lyapunov control. Unlike the previous controls, the Lyapunov allows the tracking of the orientation.