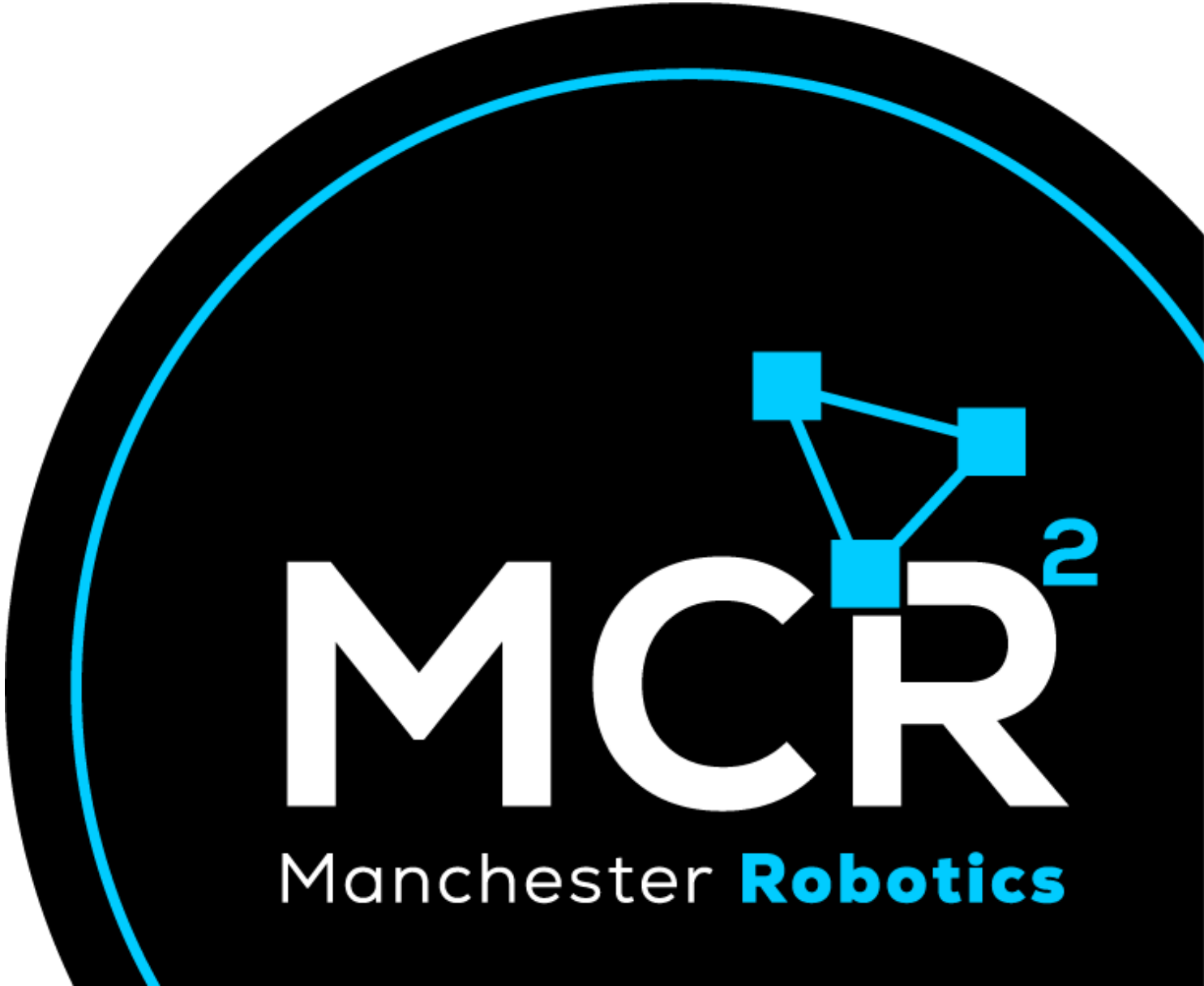
Challenges

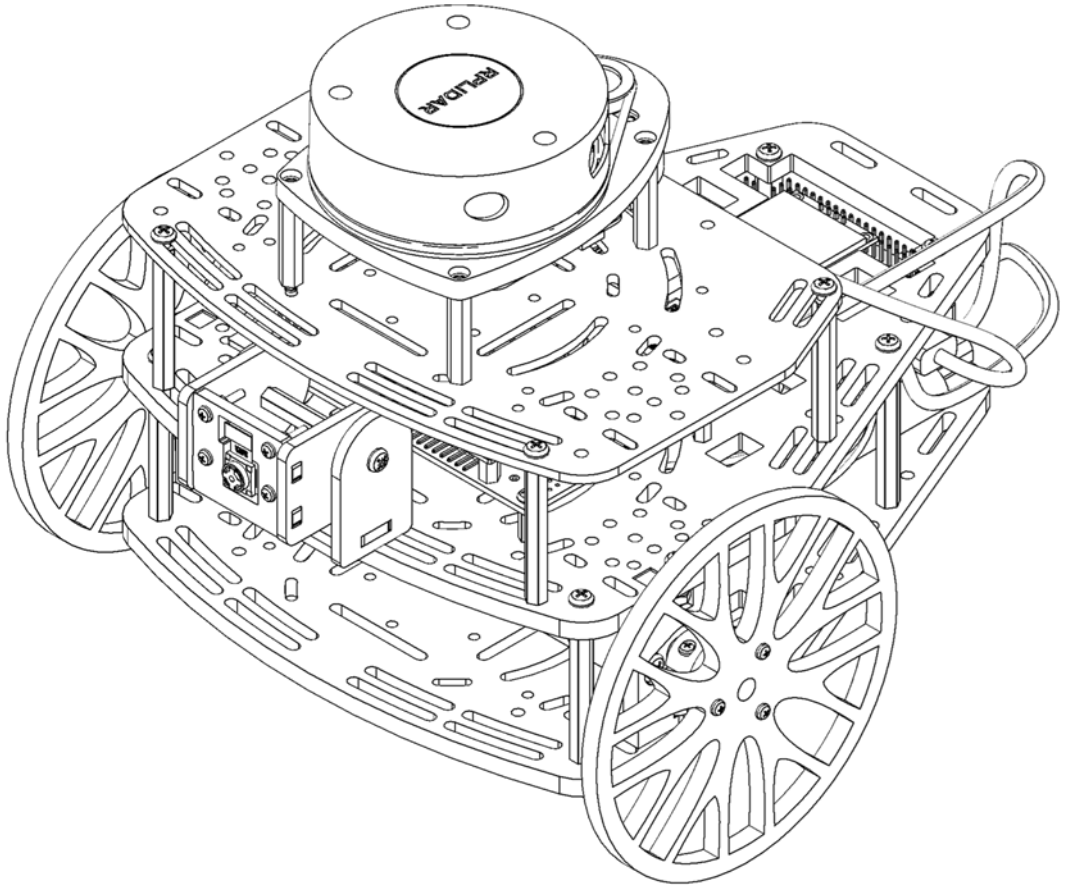
Mini challenge 2

{Learn, Create, Innovate};



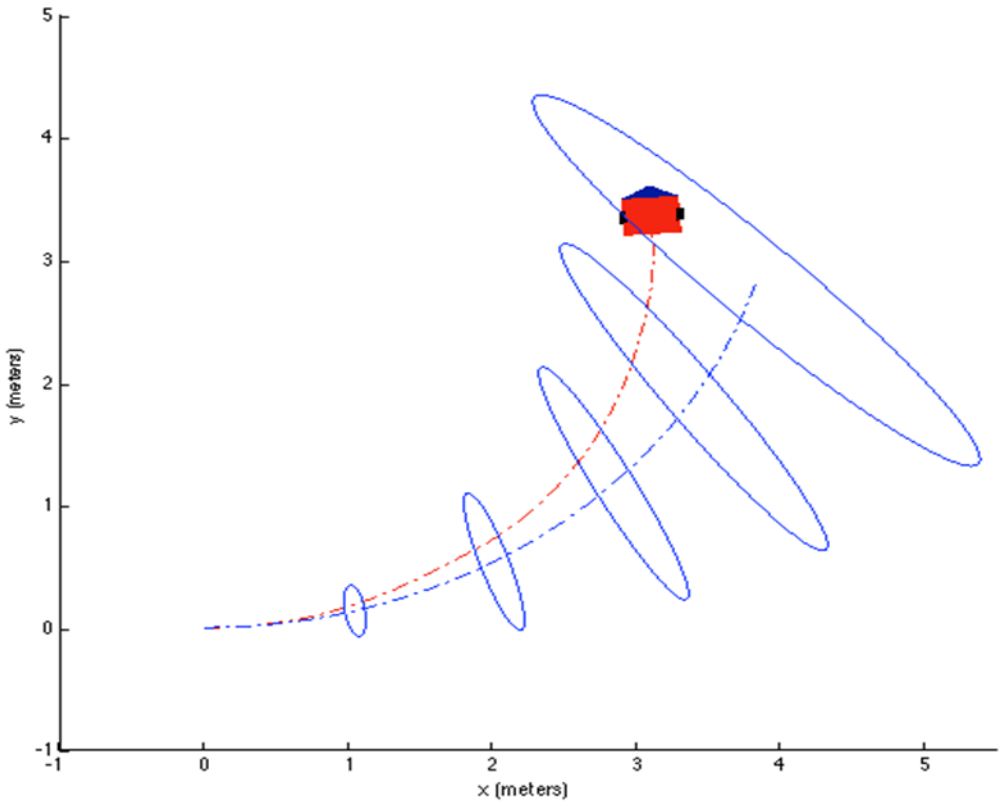
Mini challenge 2

- This challenge is intended for the student to review the concepts introduced in this week.
- This challenge aims to show the students how the noise and other perturbations affect robotic platforms.
- This challenge will be divided in different sections,.



Mini challenge 2

- Using the simulator developed for the Mini challenge 1, and the different tests to analyse the position uncertainty; estimate the uncertainty distribution and plot the confidence ellipsoid for the Puzzlebot.



Mini challenge 2

Task 1:

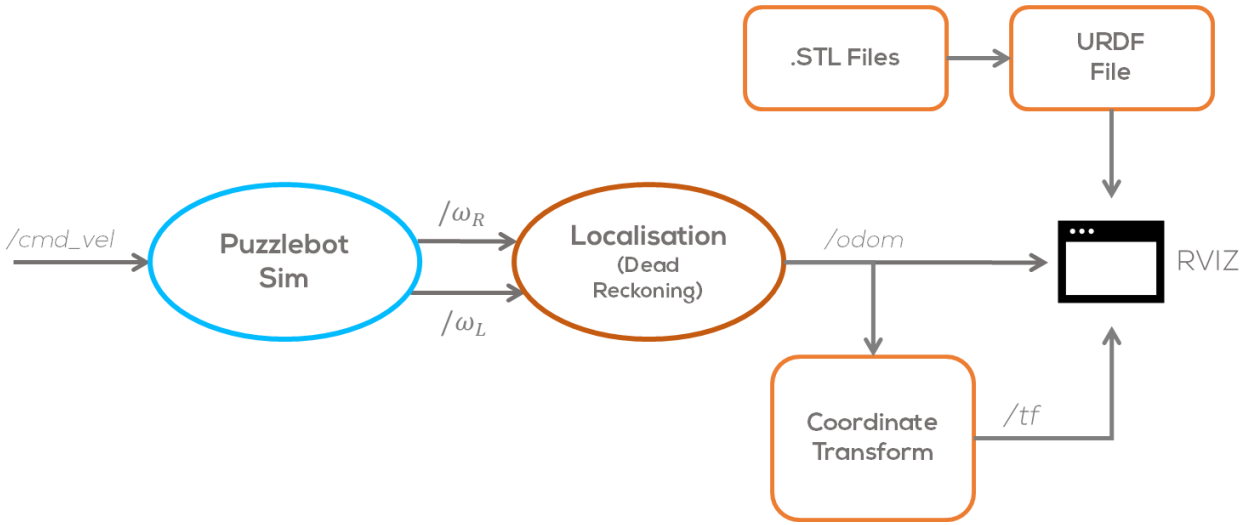
- Develop a Localisation Node.
  - Based on Dead reckoning localisation.

$$\begin{bmatrix} s_{x,k} \\ s_{y,k} \\ s_{\theta,k} \end{bmatrix} = \begin{bmatrix} s_{x,k-1} \\ s_{y,k-1} \\ s_{\theta,k-1} \end{bmatrix} + \begin{bmatrix} \Delta d \cos(s_{\theta,k-1}) \\ \Delta d \sin(s_{\theta,k-1}) \\ \Delta\theta \end{bmatrix}$$

$$\Delta d = v \cdot \Delta t \qquad \Delta\theta = \omega \cdot \Delta t$$

$$\begin{bmatrix} \Delta d \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ 1/l & -1/l \end{bmatrix} \begin{bmatrix} \Delta d_r \\ \Delta d_l \end{bmatrix}$$

  - Plot the result of the localisation in RVIZ, using coordinate transformations or robot state publisher and URDF files.
  - The output message must be in a topic called "/odom"
  - The output message to be used is "nav_msgs/Odometry.msg"
    - For this part of the task leave the **Covariance matrices** for the pose and velocity empty.



Mini challenge 2

Odom Message:

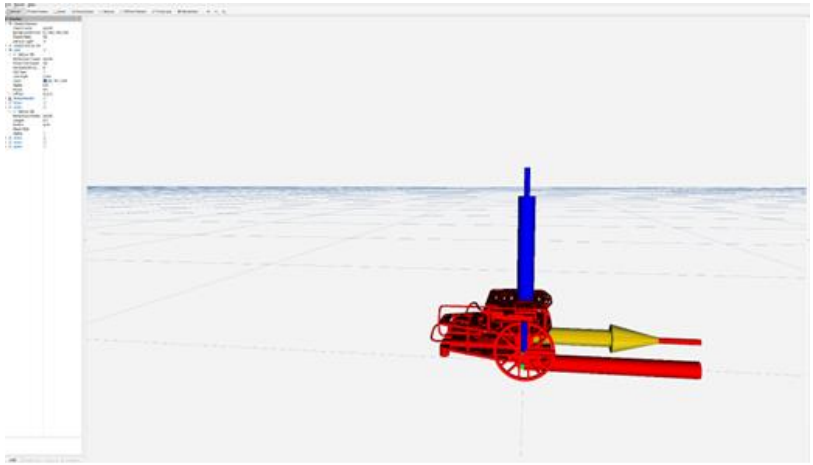
- Odometry():

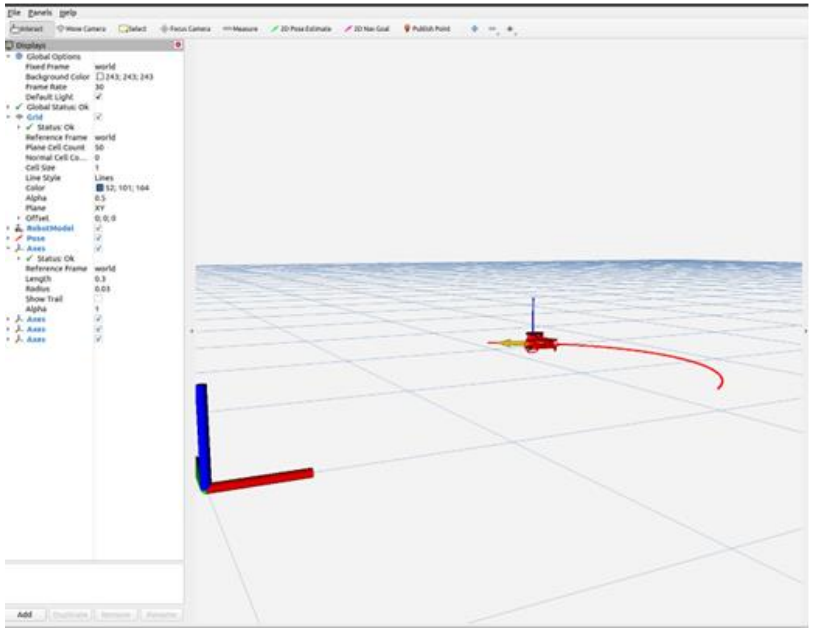
```
odometry.header.stamp = rospy.Time.now()                    #time stamp
odometry.header.frame_id = "world“                          #parent frame (joint)
odometry.child_frame_id = "base_link“                       #child frame
odometry.pose.pose.position.x = 0.0                         #position of the robot “x” w.r.t “parent frame”
odometry.pose.pose.position.y = 0.0                         # position of the robot “x” w.r.t “parent frame”
odometry.pose.pose.position.z = (Wheel Radius)              #position of the robot “x” w.r.t “parent frame”
odometry.pose.pose.orientation.x = 0.0                      #Orientation quaternion “x” w.r.t “parent frame”
odometry.pose.pose.orientation.y = 0.0                      #Orientation quaternion “y” w.r.t “parent frame”
odometry.pose.pose.orientation.z = 0.0                      #Orientation quaternion “z” w.r.t “parent frame”
odometry.pose.pose.orientation.w = 0.0                      #Orientation quaternion “w” w.r.t “parent frame”
odometry.pose.covariance = [0]*36                           #Pose Covariance 6x6 matrix (empty for now)
odometry.twist.twist.linear.x = 0.0                         #Linear velocity “x”
odometry.twist.twist.linear.y = 0.0                         #Linear velocity “y”
odometry.twist.twist.linear.z = 0.0                         #Linear velocity “z”
odometry.twist.twist.angular.x = 0.0                        #Angular velocity around x axis (roll)
odometry.twist.twist.angular.y = 0.0                        #Angular velocity around x axis (pitch)
odometry.twist.twist.angular.z = 0.0                        #Angular velocity around x axis (yaw)
odometry.twist.covariance = [0]*36                          #Velocity Covariance 6x6 matrix (empty for now)
```

- The odometry message is specially used for estimating the pose, velocity and covariance of a robot.
- This message is read by RVIZ and plotted automatically.
- ROS automatically relates the transformation tree, with the header and child frame id’s to plot the results in RVIZ, alongside the covariance ellipsoids (if implemented).

Mini challenge 2

- As per mini challenge 1, the student must define the transformation to be used.
- The students must define the required launch files for this activity.
- The simulation must be tested under different conditions, i.e., different speeds.
- The students must define a correct sampling time for the simulation .
- The students must solve the differential equations using numerical methods.
- The usage of any library is strictly **forbidden**.





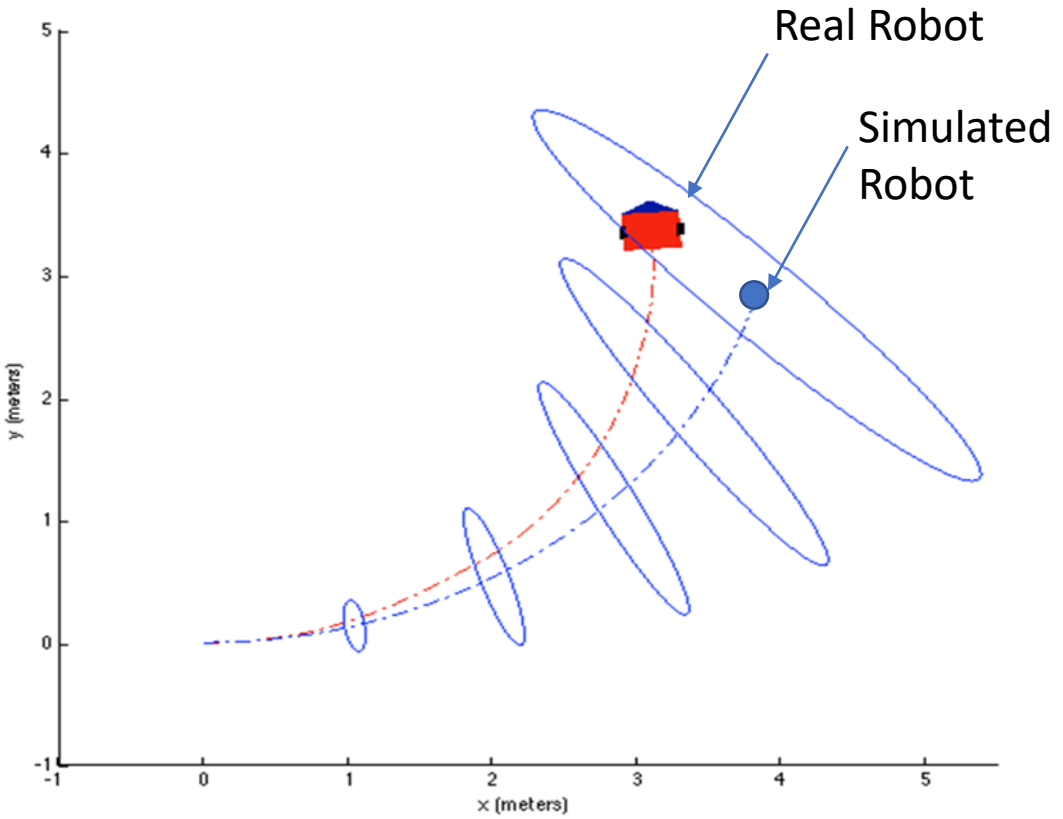
Expected results

- The expected results should look the same as the ones in challenge 1.

Mini challenge 2

Task 2:

- Using the uncertainty propagation model, and the different tests to analyse uncertainty, calibrate the model and plot the covariance ellipsoid for the pose of the robot.
- For this part of the challenge the student must fill the 6x6 pose covariance matrix of the odometry message in the localisation node, done in the previous task.



Mini challenge 2

Tips:

The robot pose $\mathbf{s}_k$ is given by a random variable

$$\mathbf{s}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Where, the mean vector $\boldsymbol{\mu}_k$ is the best estimate of the pose, and $\boldsymbol{\Sigma}_k$ is the covariance matrix.

The pose $\boldsymbol{\mu}_k$ is the one given by your dead reckoning localisation node.

The covariance $\boldsymbol{\Sigma}_k$ can be approximated by:

$$\boldsymbol{\Sigma}_k = \mathbf{H}_k \boldsymbol{\Sigma}_{k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{Q}_k$$

Where, $\boldsymbol{\Sigma}_k$ is a 3x3 covariance matrix.

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \end{bmatrix}$$

- θ represents rotation around z-axis (yaw)

Mini challenge 2

$\mathbf{H}_k$ is a 3x3 Linear model Jacobian of the robot.

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & -\Delta t \cdot v_k \cdot \sin(\mu_{\theta,k-1}) \\ 0 & 1 & \Delta t \cdot v_k \cdot \cos(\mu_{\theta,k-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

- The matrix Q_k is the nondeterministic error matrix, given by

$$Q_k = \nabla_{\omega_k} \cdot \Sigma_{\Delta,k} \cdot \nabla_{\omega_k}^T$$

Where,

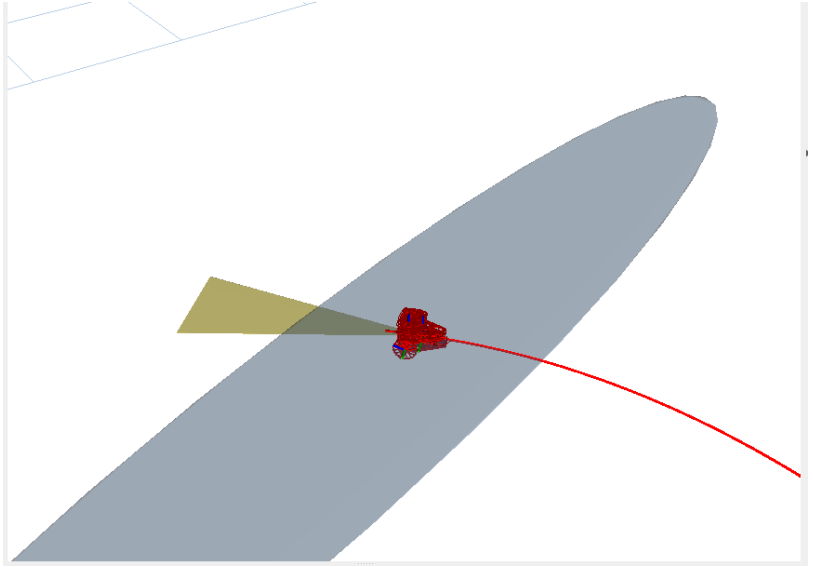
$$\Sigma_{\Delta,k} = \begin{bmatrix} k_r|\omega_{r,k}| & 0 \\ 0 & k_l|\omega_{l,k}| \end{bmatrix}$$

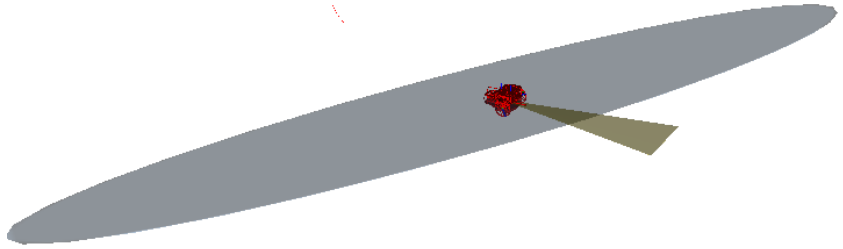
The values of k_r and k_l must be tunned according to the test realised for the mini challenge 1.

$$\nabla_{\omega_k} = \frac{1}{2} r \Delta t \begin{bmatrix} \cos(s_{\theta,k-1}) & \cos(s_{\theta,k-1}) \\ \sin(s_{\theta,k-1}) & \sin(s_{\theta,k-1}) \\ \frac{2}{l} & -\frac{2}{l} \end{bmatrix}$$

Mini challenge 2

- RVIZ automatically plots the covariance ellipsoid for the pose of the robot, given that the correct message and transformation are used (odometry message).
- As per the previous task, the student must define the transformation to be used.
- The students must define the required launch files for this activity.
- The simulation must be tested under different conditions, i.e., different speeds.
- The students must define a correct sampling time for the simulation .
- The students must solve the differential equations using numerical methods.
- The usage of any library is strictly **forbidden**.





Rules

- This is challenge **not** a class. The students are encouraged to research, improve tune explain their algorithms by themselves.
- MCR2(Manchester Robotics) Reserves the right to answer a question if it is determined that the questions contains partially or totally an answer.
- The students are welcomed to ask only about the theoretical aspect of the classed.
- No remote control or any other form of human interaction with the simulator or ROS is allowed (except at the start when launching the files).
- It is **forbidden** to use any other internet libraires with the exception of standard libraires or NumPy.
- If in doubt about libraires please ask any teaching assistant.
- Improvements to the algorithms are encouraged and may be used as long as the students provide the reasons and a detailed explanation on the improvements.
- All the students must be respectful towards each other and abide by the previously defined rules.
- Manchester robotics reserves the right to provide any form of grading. Grading and grading methodology are done by the professor in charge of the unit.