State and Parameter estimator for Erle rover

A Project Report

submitted by

SATHYA ASWATH GOVINDRAJU EE15B028

in partial fulfilment of requirements
for the award of the dual degree of

BACHELOR OF TECHNOLOGY AND MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

June 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **State and Parameter estimator for Erle rover**, submitted by **Sathya Aswath Govindraju**, to the Indian Institute of Technology, Madras, for the award of the degree of **Dual degree of Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Ramkrishna Pasumarthy
Research Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Dr. Nirav P Bhatt
Research Guide
Assistant Professor
Dept. of Biotechnology
IIT-Madras, 600 036

Place: Chennai

Date: 13 June 2020

ACKNOWLEDGEMENTS

I would like to specially thank Dr. Ramkrishna Pasumarthy and Dr. Nirav P Bhatt for giving me an opportunity to be part of the autonomous vehicles project and for their guidance through the whole period.

I would also like to thank Mr. Subhadeep Kumar, phD Research scholar for his support and guidance during the period of this project.

ABSTRACT

KEYWORDS: Vehicle model; State estimator; Parameter estimator; Unscented Kalman filter, Particle swarm optimization, Comprehensive learning Particle swarm optimization

A model that the describes the motion of a vehicle in a two dimensional plane has been developed. A state estimator that is suitable for estimating the states of the model in real time, has been implemented using a modified Unscented Kalman filter(UKF). A parameter estimator using Comprehensive Learning Particle swarm optimization(CLPSO), has been implemented for estimating the immeasurable parameters in the developed model.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
UKF	Unscented Kalman Filter
PSO	Particle Swarm Optimization
CLPSO	Comprehensive Learning Particle Swarm Optimization
NM	Nelder Mead algorithm
RSSE	Root Sum Square Error
ROS	Robot Operating System
EB3	Erle Brain 3
ESC	Electronic Speed Controller

CHAPTER 1

INTRODUCTION

This is a part of autonomous vehicles project under Dr. Ramkrishna Pasumarthy, Associate Professor,department of Electrical Engineering, IIT Madras. The aim of the autonomous vehicles project is to build intelligent systems for the maneuver of on-road autonomous vehicles. This includes developing the infrastructure and driving systems for the vehicles. Driving systems include developing local controllers, for which model based controllers were proposed to be used. The reliability and performance of these controllers depends on the model. These driving systems are implemented on scaled vehicles for testing their performance.

The aim of this project is to build a model for Erle rover figure(1.1), that is good enough for predicting the dynamics of the vehicle for a given input but simple enough so that it can simulated online in an embedded micro-controller. The model built has some states and parameters that cannot measured directly. So, these are estimated using a parameter estimator and a state estimator. The parameter estimation is done using CLPSO-NM algorithm(Cao *et al.* (2019)) . The state Estimation is done using a modified Unscented kalman filter(Wan and Van Der Merwe (2000)). The state estimator is suitable for being run on an embedded micro-controller on the Erle rover and the parameter estimator runs offline which provides the state estimator with parameters for the model.

Figure 1.1: Erle rover

CHAPTER 2

Problem Statements

1. **Model for the vehicle:** A model that describes the dynamics of the vehicle for a given input is required. The model has to be reasonably accurate and simple so that it is suitable for simulating in real time on an embedded micro-controller.A model is built using some sub-models already present and using basic laws of physics to integrate them.

2. **State Estimator:** The model built has some states that can not be sensed directly using a sensor. The states that can be sensed using a sensor will also be distorted with noise. A state estimator using the model developed for the vehicle has been implemented.

3. **Parameter Estimation:** The model built has some parameters that can only be measured by using complex methods and sophisticated equipment, which are either hard to conduct or not easily available. So, a parameter estimation method which uses the data collected from the available sensors on the vehicle to estimate the parameters has been implemented.

CHAPTER 3

Vehicle model

A model that describes the movement of vehicle in a 2 dimensional plane has been built. The motion along the third dimension has been neglected. The model is built as a combination of sub-models that already exist and combined together using basic laws of physics to form a Differential algebraic equations system that describes the motion of the vehicle in a 2 dimensional space.

3.1 Motor Model

The vehicle being used for the experiments is a buggy which is driven by a brushed motor attached to the rear two wheels with an open differential in between. For the motor, second order model is considered.

$$J_m\dot{\omega}_m + B_m\omega_m + T_m = K_m i_m \tag{3.1a}$$

$$L_m\dot{i}_m + R_m i_m + K_m\omega_m = V_m \tag{3.1b}$$

Where,

J_m - Moment of inertia of the rotor.

B_m - Friction coefficient.

T_m - Drive torque given by the rotor.

K_m - represents the torque constant and back emf constant which are taken as to be the same.

L_m - Inductance of the motor coil.

R_m - Resistance of the motor coil.

I_m - Current in the motor coil.

V_m - Voltage applied to the motor across its terminals.

3.2 Wheel Dynamics

The rotor of the motor is attached to the open differential through gears of gear ratio G. So,

$$GT_m = T_d$$
$$\omega_m = G\omega_d$$

Where T_d and ω_d are the torque and angular velocity at the differential. In an open differential the speed of the crown gear is shared by the wheels and the torque given to both the wheels will be equal,

$$\omega_d = \frac{\omega_{rL} + \omega_{rR}}{2}$$
$$T_L = T_R$$

Where, ω_{rL} and ω_{rR} are the angular velocities of the rear left and rear right wheels respectively. T_L and T_R are the torques at the rear left and right wheels. Combining the above equations and balancing the torques on the rear wheels gives the following equations,

$$J_w\dot{\omega}_{rj} = T_j - R_w f_{rjx} \tag{3.2a}$$
$$2(\omega_d) = \omega_{rR} + \omega_{rL} \tag{3.2b}$$
$$\omega_m = G\omega_d \tag{3.2c}$$

Balancing the torques on the front wheels gives,

$$J_w\dot{\omega}_{fj} = -R_w Fx_{fj} \tag{3.3}$$

Where,

J_w - Moment of Inertia of the wheels.

Fx_{rj} - frictional force acting on the rear tires tires in x direction.

j - L or R which represents left or right.

Combining the equations (3.1), (3.2), (3.3) and assuming no power loss during the transfer in open differential leads to the following set of equations.

$$\dot{\omega}_{rL} = \frac{K_m}{GJ_m + 2J_w} i_m + \left(\frac{GJ_m R_w}{2J_w(GJ_m + 2J_w)}\right) Fx_{rR} + \left[\frac{GJ_m R_w}{2J_w(GJ_m + 2J_w)} - \frac{R_w}{J_w}\right] Fx_{rL} - \frac{Bm}{GJ_m + 2J_w}\omega_m$$

$$\dot{\omega}_{rR} = \frac{K_m}{GJ_m + 2J_w} i_m + \left(\frac{GJ_m R_w}{2J_w(GJ_m + 2J_w)}\right) Fx_{rL} + \left[\frac{GJ_m R_w}{2J_w(GJ_m + 2J_w)} - \frac{R_w}{J_w}\right] Fx_{rR} - \frac{Bm}{GJ_m + 2J_w}\omega_m$$

$$\dot{\omega}_{fL} = -\frac{R_w}{J_w} Fx_{fL}$$

$$\dot{\omega}_{fR} = -\frac{R_w}{J_w} Fx_{fR}$$

$$\dot{i}_m = -\frac{R_m}{L_m} i_m - \frac{K_m}{L_m}\omega_m + \frac{1}{L_m} V_m$$

3.3 Tire force model

In this sub model, the frictional forces generated are modelled using slips at each wheel as given in You and Tsiotras (2017)

Longitudinal Slip

$$Sx_{ij} = \frac{(vx_{ij} - Rw * \omega_{ij})}{max(vx_{ij}, Rw * \omega_{ij})}$$

Special cases:

1. $vx_{ij} = R_w * \omega_{ij}$ then slip is made zero to avoid the 0/0 case.
2. $\omega_{ij} = 0$ and $vx_{ij} < 0$ then according to the definition slip becomes -∞. So, the slip is made -1(limiting the slip).
3. $vx_{ij} = 0$ and $\omega_{ij} < 0$ then according to the definition slip becomes ∞. So, the slip is made 1(limiting the slip).

Lateral Slip

$$Sy_{ij} = \frac{vy_{ij}}{(R_w * \omega_{ij})}$$

Special Cases:

1. $vy_{ij} = R_w * \omega_{ij}$ and $\omega_{ij} = 0$ then the vehicle is stationary. So, the slip is made zero.
2. $\omega_{ij} = 0$ then vehicle is skidding. So, the slip is as the sign(vy_{ij}).

Total Slip

$$S_{ij} = \sqrt{(Sx_{ij}^2 + Sy_{ij}^2)}$$

where,

vx_{ij} - velocity of the ij^{th} tire in x direction w.r.t the body frame.

vy_{ij} - velocity of the ij^{th} tire in y direction w.r.t the body frame.

i - f,r represents if it is front or rear and j - L, R represents if it is Left or Right.

Parameters for Tire model

These parameters used here are from magic formula given in(You and Tsiotras (2017))

$$Se_{ij} = S_{ij} - Sh$$
$$\mu_{ij} = D * sin(C * atan(B * Se_{fl} - E * (B * Se_{fl} - atan(Se_{fl})))) + Sv$$

where, B, C, D, E, S_h, S_v are constants from the magic formula.

Frictional Forces

$$Fx_{ij} = -\left(\left(\frac{Sx_{ij}}{S_{ij}}\right) * \mu_{ij} * Fz_{ij}\right) \tag{3.4a}$$
$$Fy_{ij} = -\left(\left(\frac{Sy_{ij}}{S_{ij}}\right) * \mu_{ij} * Fz_{ij}\right) \tag{3.4b}$$

Special Cases:

1. When the total slip is zero the frictional forces are made zero.

3.4 Car Model

Using the rigid body dynamics of the vehicle the relationship between velocity at the center of mass and the velocity at the wheels are found. The equations are given below,

$$vy_{rR} = vy - \dot{\psi} l_r \tag{3.5a}$$

$$vx_{rR} = vx + \dot{\psi} w_r \tag{3.5b}$$

$$vy_{rL} = vy - \dot{\psi} l_r \tag{3.5c}$$

$$vx_{rL} = vx - \dot{\psi} w_r \tag{3.5d}$$

$$vy_{fR} = -\sin(\delta)(\dot{\psi} w_f + V_x) + \cos(\delta)(\dot{\psi} l_f + vy) \tag{3.5e}$$

$$vx_{fR} = \cos(\delta)(vx + \dot{\psi} w_f) + \sin(\delta)(\dot{\psi} l_f + vy) \tag{3.5f}$$

$$vy_{fL} = -\sin(\delta)(-\dot{\psi} w_f + vx) + \cos(\delta)(\dot{\psi} l_f + vy) \tag{3.5g}$$

$$vx_{fL} = \cos(\delta)(vx - \dot{\psi} w_f) + \sin(\delta)(\dot{\psi} l_f + vy) \tag{3.5h}$$

$$\tag{3.5i}$$

where,

vx - velocity in x direction of vehicle's center of mass w.r.t body frame.

vy - velocity in y direction of vehicle's center of mass w.r.t body frame.

ψ - Yaw of the vehicle about the center of mass w.r.t global frame.

δ - Steering angle of the vehicle.

l_r - Distance of center of mass of the vehicle to the rear axle.

l_f - Distance of center of mass of the vehicle to the front axle.

w_f - half track width of the front axle.

w_r - half track width of the rear axle.

3.5 Double Track Model

The Double track model from (You and Tsiotras (2017)) is used here,

$$\dot{vx} = \frac{(Fx_f \cos(\delta) - Fy_f \sin(\delta) + Fx_r)}{Mv} + vy\dot{\psi} \tag{3.6a}$$

$$\dot{vy} = \frac{(Fx_f \sin(\delta) + Fy_f \cos(\delta) + Fy_r)}{Mv} + vx\dot{\psi} \tag{3.6b}$$

$$\ddot{\psi} = \frac{((Fy_f \cos(\delta) + Fx_f \sin(\delta))l_f - Fy_r l_r)}{J_z} \tag{3.6c}$$

where

Fx_f - total frictional force acting on the front wheels in x direction w.r.t body frame. This is the sum of individual frictional forces acting in x direction w.r.t body frame on each of the front wheels.

Fy_f - total frictional force acting on the front wheels in y direction w.r.t body frame. This is the sum of individual frictional forces acting in y direction w.r.t body frame on each of the front wheels.

Fx_r - total frictional force acting on the rear wheels in x direction w.r.t body frame. This is the sum of individual frictional forces acting in x direction w.r.t body frame on each of the rear wheels.

Fy_r - total frictional force acting on the rear wheels in y direction w.r.t body frame. This is the sum of individual frictional forces acting in y direction w.r.t body frame on each of the rear wheels.

J_z - Moment of inertia of the vehicle about z-axis w.r.t body frame.

Mv - Mass of the vehicle.

3.6 Normal Load constraints

The Normal load on a vehicle at any instant is found using the following constraints. These are obtained by balancing moments about the pitch axis(Limebeer and Rao (2015)).

$$\begin{bmatrix} Fz_{rR} \\ Fz_{rL} \\ Fz_{fR} \\ Fz_{fL} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -\frac{l_f Mg + aAF_{az}}{aA+aB} & -\frac{(D_r-1)hFy}{(1-D_r)w_r+D_r w_f} & -\frac{hFx}{aA+aB} \\ -\frac{l_f Mg + aAF_{az}}{l_f+l_r} & \frac{(D_r-1)hFy}{(1-D_r)w_r+D_r w_f} & -\frac{hFx}{aA+aB} \\ -\frac{l_r Mg + aBF_{az}}{aA+aB} & \frac{(D_r)hFy}{(1-D_r)w_r+D_r w_f} & \frac{hFx}{aA+aB} \\ -\frac{l_r Mg + aBF_{az}}{aA+aB} & -\frac{(D_r)hFy}{(1-D_r)w_r+D_r w_f} & \frac{hFx}{aA+aB} \end{bmatrix} \tag{3.7}$$

where,

Fz_{ij} - The force acting in the z direction or the normal load on the ij^{th} wheel.

aA - distance of the front axle from the center of gravity.

aB - distance of the rear axle from the center of gravity.

h - height of center of gravity from the ground.

Fx - Total frictional acting on the vehicle in x direction w.r.t body frame.

Fy - Total frictional acting on the vehicle in y direction w.r.t body frame.

F_{az} - Total aerodynamic drag force acting on the vehicle.

D_r - roll coefficient

3.7 The DAE system

All the equations obtained from the sub-models, when put together give an index 1 semi-explicit DAE system.
States of the model are,

1. **Differential states:**
   (a) **Position:** Position of the vehicle w.r.t 2 dimensional global frame.
   (b) **Velocity:** Velocity of the vehicle w.r.t 2 dimensional body frame.
   (c) **Yaw angle:** Yaw angle of the vehicle w.r.t 2 dimensional global frame.

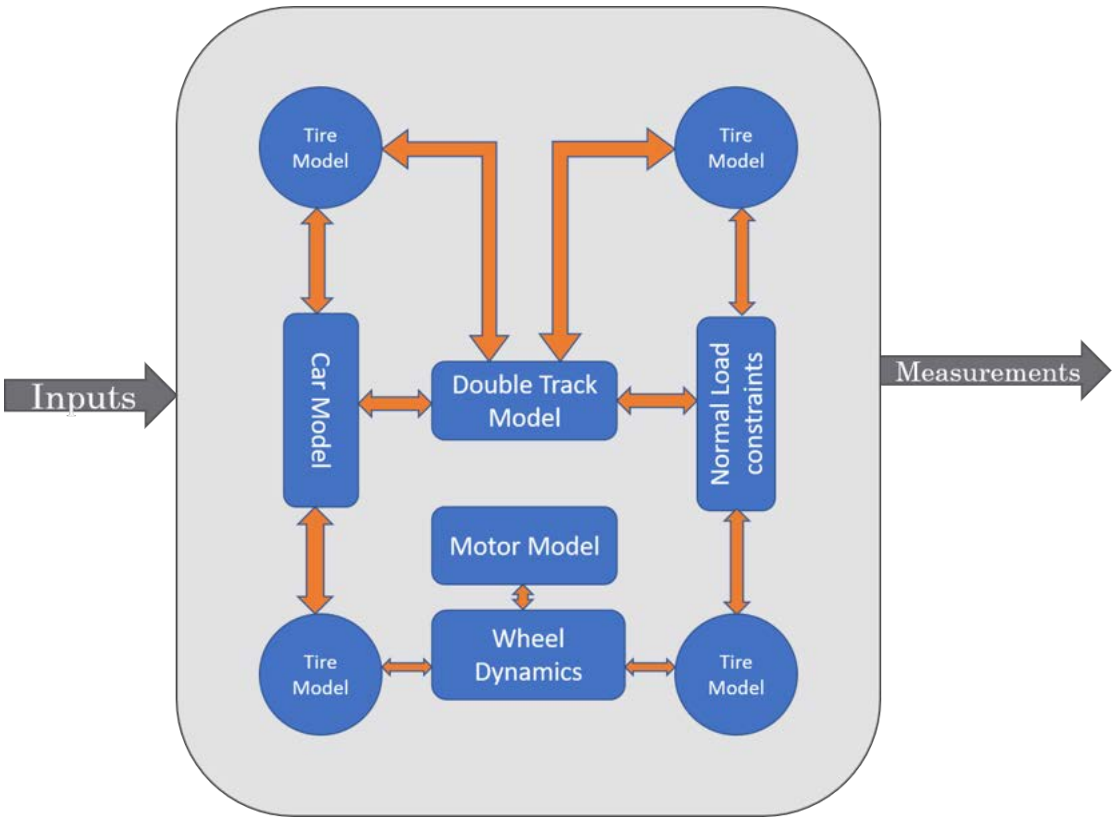

Figure 3.1: A diagram of all the sub systems in the model put together

(d) **Yaw rate:** Yaw rate of the vehicle w.r.t 2 dimensional body frame.

(e) **Wheel Angular Velocity:** Angular velocities of each wheel.

(f) **Motor Current:** Current in the coil of the rear drive motor.

2. **Algebraic states:**
   (a) **Normal Loads:** Normal loads on each of the wheels.

The inputs to the model are:

1. **Voltage:** Input voltage to the rear drive motor.
2. **Steering Angle:** Steering angle input for the steering servo.

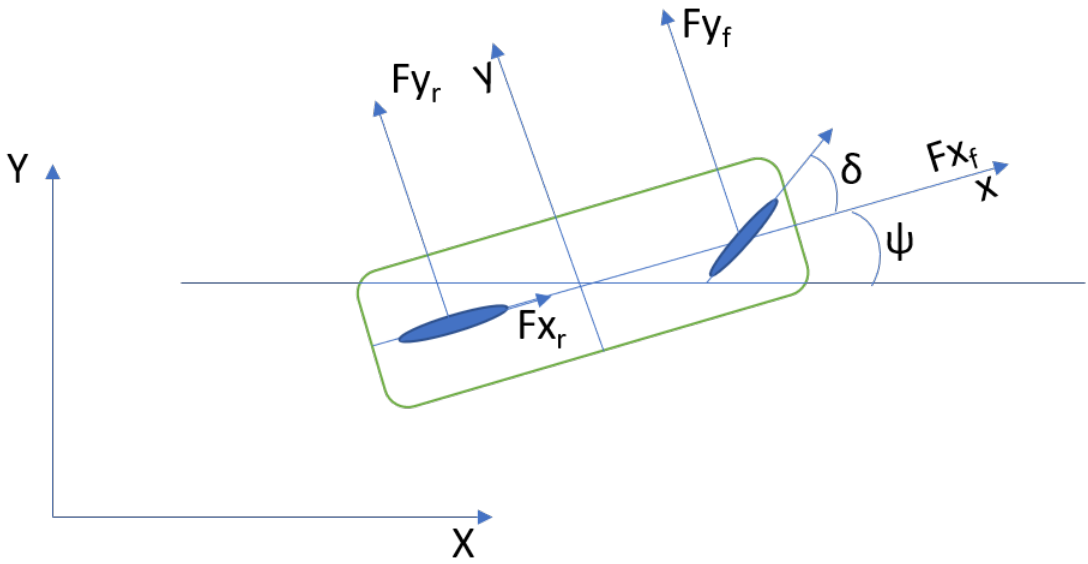

Figure 3.2: The frame of reference used for the model

3.7.1 The System of equations

ODE's

$$\dot{sx} = vx * \cos(\psi) - vy * \sin(\psi) - sy * \dot{\psi} \quad \text{global frame}$$

$$\dot{sy} = vx * \sin(\psi) + vy * cos(\psi) + sx * \dot{\psi} \quad \text{global frame}$$

$$\dot{vx} = \frac{(Fx_f * \cos(\delta) - Fy_f * \sin(\delta) + Fx_r)}{Mv} + vy * \dot{\psi} \quad \text{body frame}$$

$$\dot{vy} = \frac{(Fx_f * \sin(\delta) + Fy_f * \cos(\delta) + Fy_r)}{Mv} + vx * \dot{\psi} \quad \text{body frame}$$

$$\dot{\psi} = \dot{\psi}$$

$$\ddot{\psi} = \frac{(Fy_f * \cos(\delta) * l_f + Fx_f * \sin(\delta) * l_f + Fy_r * l_r)}{J_z}$$

$$\dot{\omega}_{fL} = -\frac{(R_w * Fx_{fL})}{J_w}$$

$$\dot{\omega}_{fR} = -\frac{(R_w * Fx_{fR})}{J_w}$$

$$\dot{\omega}_{rL} = \left(\frac{((K_m * G)}{(G^2 * J_m + 2 * J_m))}\right) * i_m + \left(\frac{(G^2 * J_m * R_w)}{(2 * J_w * (G^2 * J_m + 2 * J_w)}\right) * Fx_{rR}$$
$$+ \left(\left(\frac{(G^2 * J_m * R_w)}{(2 * J_w * (G^2 * J_m + 2 * J_w))}\right) - \left(\frac{R_w}{J_w}\right)\right) * Fx_{rL}$$
$$- \left(\frac{(B_m * G)}{(G^2 * J_m + 2 * J_w)}\right) * \omega_m$$

$$\dot{\omega}_{rr} = \left(\frac{((K_m * G)}{(G^2 * J_m + 2 * J_m))}\right) * i_m + \left(\frac{(G^2 * J_m * R_w)}{(2 * J_w * (G^2 * J_m + 2 * J_w)}\right) * Fx_{rL}$$
$$+ \left(\left(\frac{(G^2 * J_m * R_w)}{(2 * J_w * (G^2 * J_m + 2 * J_w))}\right) - \left(\frac{R_w}{J_w}\right)\right) * Fx_{rR}$$
$$- \left(\frac{(B_m * G)}{(G^2 * J_m + 2 * J_w)}\right) * \omega_m$$

$$\dot{i}_m = \frac{(V_m - R_m * i_m - K_m * \omega_m)}{L_m}$$

Algebraic Constraints

$$0 = Fz_{fL} - \frac{(l_r * Mv * g + aB * F_{az})}{(2 * (aA + aB))} - \frac{(D_r * h * Fy)}{(2 * ((1 - D_r) * w_r + D_r * w_f))} + \frac{(h * Fx)}{(2 * (aA + aB))}$$

$$0 = Fz_{fR} - \frac{(l_r * Mv * g + aB * F_{az})}{(2 * (aA + aB))} + \frac{(D_r * h * Fy)}{(2 * ((1 - D_r) * w_r + D_r * w_f))} + \frac{(h * Fx)}{(2 * (aA + aB))}$$

$$0 = Fz_{rL} - \frac{(l_f * Mv * g + aA * F_{az})}{(2 * (aA + aB))} + \frac{((D_r - 1) * h * Fy)}{(2 * ((1 - D_r) * w_r + D_r * w_f))} - \frac{(h * Fx)}{(2 * (aA + aB))}$$

$$0 = Fz_{rR} - \frac{(l_f * Mv * g + aA * F_{az})}{(2 * (aA + aB))} - \frac{((D_r - 1) * h * Fy)}{(2 * ((1 - D_r) * w_r + D_r * w_f))} - \frac{(h * Fx)}{(2 * (aA + aB))}$$

The Frictional forces are found by using the relationship in section(3.3).

CHAPTER 4

State Estimator

The model built for the vehicle is index 1 semi explicit non-linear DAE system. The measurements available from the rover are:

1. **Acceleration:** Acceleration of the vehicle w.r.t 2 dimensional body frame.
2. **Yaw angle:** Yaw angle of the vehicle w.r.t 2 dimensional global frame.
3. **Yaw rate:** Yaw rate of the vehicle w.r.t 2 dimensional body frame.
4. **Wheel angular Velocity:** Angular velocities of each wheel.
5. **Motor Current:** Current in the coil of the rear drive motor.

There are some states in the model that can't be directly measured from the vehicle and the available measurements of these states are not noise free. Therefore, a state estimator is required.

The most commonly used state estimator for linear systems Kalman Filter and its various extensions are used for non-linear systems . Some of them for non-linear systems are Extended Kalman Filter(EKF) and Unscented Kalman Filter(UKF), both of them have their advantages and disadvantages. EKF linearizes the system by taking the Jacobian at every operating point for finding a prediction of states at the next time instant. Linearizing the model built in section(3) is very cumbersome and the model has operating points where a Jacobian is not defined for it. UKF uses the non-linear model directly for the prediction which makes it suitable for our model. Usually for highly non-linear systems UKF is mostly used((Wan and Van Der Merwe (2000))). The problem with UKF is that it highly dependent on its hyper parameters.

4.1 Unscented Kalman Filter

An algorithm of any variant of Kalman filter can be divided into two basic steps. They are,

Prediction: Model is used to make prediction of the states at the next instant using a given model for the system using the current states and inputs.

Update: The measurements obtained from the real system are then used to update the predicted states of the system, using the process and measurement noise covariance matrices given by the user.

Unscented Kalman Filter(UKF), uses unscented transformation for prediction.

4.1.1 Unscented Transformation

When a Gaussian random variable with a known mean and covariance undergoes a non-linear transformation, the statistics of the transformed random variables can be found accurately upto third order using unscented transformation. If the random variables do not follow a Gaussian distribution then the estimation of the statistics of the output random variable would be accurate upto second order(Wan and Van Der Merwe (2000)).

Using the mean and covariance of the random variables, a minimal set of points are generated that represent the random variable, called sigma points(If the random variable follows gaussian then these points represent it accurately as first two moments accurately describe a gaussian random variable). These sigma points then undergo the non-linear transformation which are then used to find the mean and covariance of the transformed random variable. Figure(4.1) represents the procedure of Unscented transformation for a two dimensional case.

In UKF, the states are assumed to be random variables and the unscented transformation is performed on them to get a prediction of the state's mean and covariance at next instant.

4.1.2 Estimation of measurement noise statistics

As the covariance of the noise present in the measurements can be unknown . It is also estimated along with the states of the system. This done by storing the difference in measurements obtained from the real system to the measurements obtained from

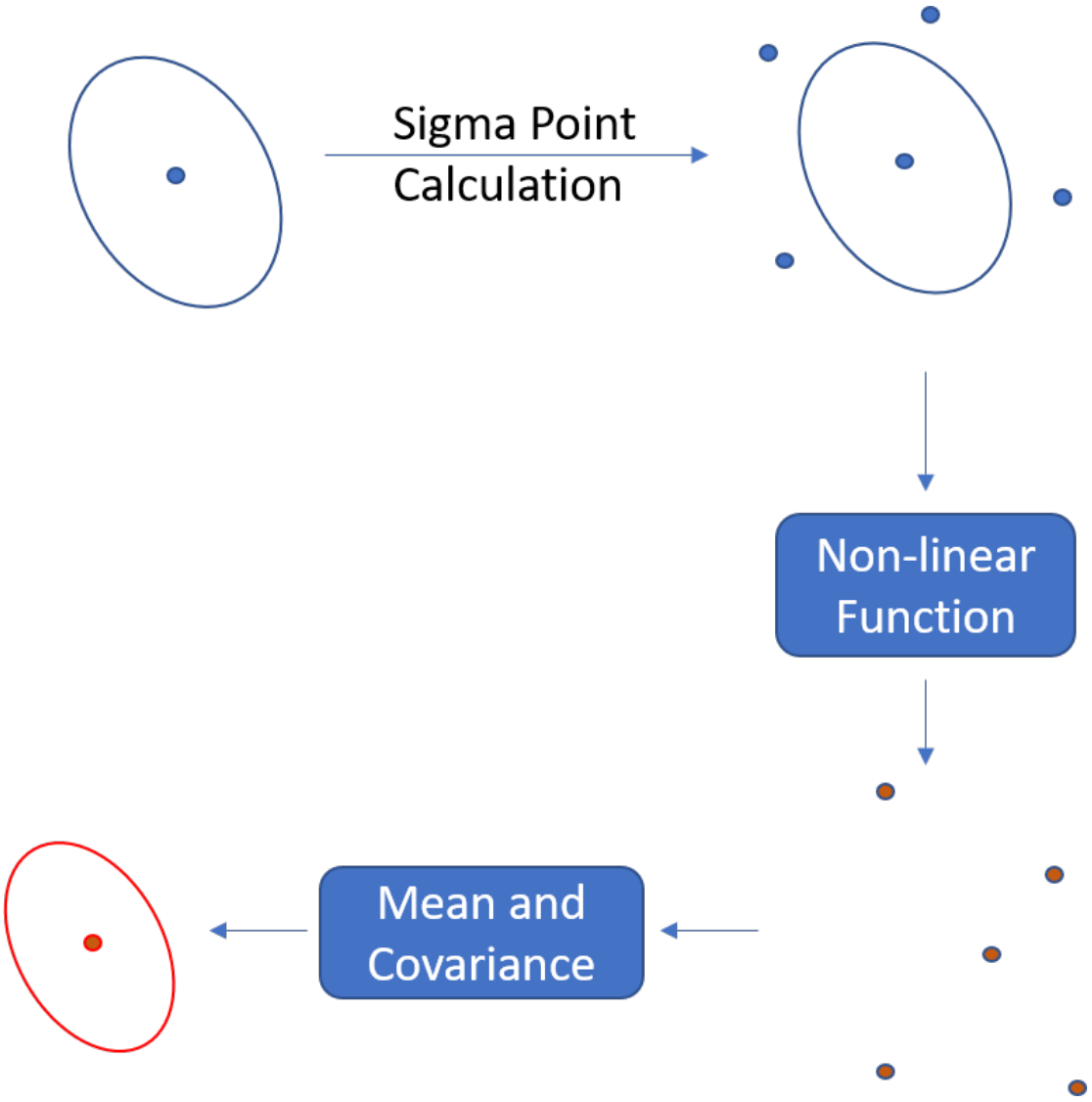

Figure 4.1: Unscented Transformation

the state estimator after updating. This is called residual based R-AUKF algorithm as mentioned in Das *et al.* (2014).

4.2 Algorithm

1. **Unscented transformation parameter initialization:**

$$\lambda = \alpha^2(L + \kappa) - L$$
$$W_o^m = \frac{\lambda}{L + \lambda}$$
$$W_o^c = \frac{\lambda}{L + \lambda} + 1 - \alpha^2 + \beta$$
$$W_i = \frac{0.5}{L + \lambda}, \text{i = 1, ... 2L}$$
$$\gamma = \sqrt{L + \lambda}$$

where L is the number of the states in the state vector (both differential and algebraic), α is a parameter that defines the spread of the sigma points, κ and β are hyper parameters of UKF which are usually set to 0 and 2 respectively (optimal for gaussian), W_o^m and W_0^c are the coefficients for the first sigma point while calculating mean and covariance, respectively, and W^i is coefficient for the remaining sigma points for calculating mean and covariance. Increasing the parameter α increases the spread of the sigma points.

2. **Sigma Point Calculation:**

$$\chi_{k-1} = \begin{bmatrix} \hat{x}_{k-1} & \hat{x}_{k-1} + \gamma\sqrt{P_{k-1}} & \hat{x}_{k-1} - \gamma\sqrt{P_{k-1}} \end{bmatrix}$$

where $\hat{x}_{k-1}$ is the estimated mean of the states(both differential and algebraic) in the previous iteration and χ_{k-1} denotes the matrix with sigma points as columns. $\sqrt{P_{k-1}}$ is the square root (cholesky decomposition) of the covariance matrix. In some cases, P_{k-1} can be negative definite for which an approximate square root is found.

3. **Estimating Algebraic states from Differential states:** The sigma points are calculated by adding a column of the square root of covariance matrix to the previous estimated state. This might not be consistent with algebraic constraints in the model. Therefore, the algebraic states are determined from the differential states for each of the sigma points upon solving the algebraic constraints, (Mandela *et al.* (2009)). The algebraic constraints can be written as,

$$AF_z = b$$

where F_z is a column vector representing the normal load on each wheel. A and b are calculated using differential states from each sigma point and the system of equations are solved to find the normal load at each wheel. These algebraic states are then appended to the differential states and are used in state prediction.

4. **State Prediction:**

$$\begin{aligned} \chi^-_{k|k-1} &= f(\chi_{k-1}, u_{k-1}) \\ \hat{x}^-_k &= \sum_{i=0}^{2L} W_i^m \chi^-_{i,k|k-1} \\ P^-_k &= \sum_{i=0}^{2L} W_i^c (\chi^-_{i,k|k-1} - \hat{x}^-_k)(\chi^-_{i,k|k-1} - \hat{x}^-_k)^T + Q_k \end{aligned}$$

where, $f(x, u)$ is the DAE model used for predicting the states at next instant. $\chi^-_{k|k-1}$ is the matrix containing the sigma points post transformation as columns. $\hat{x}^-_k$ is the predicted mean of the state vector at time instant k and P^-_k is the predicted error covariance matrix of the state vector at instant k. Q_k is the process noise covariance matrix at instant k which has to be initialized by the user.

5. **Measurement Update 1:**

$$\begin{aligned}
\chi_{k|k-1} &= \begin{bmatrix} \hat{x}_k^- & \hat{x}_k^- + \gamma\sqrt{P_k^-} & \hat{x}_k^- - \gamma\sqrt{P_k^-} \end{bmatrix} \\
Y_{k|k-1} &= h(\chi_{k|k-1}, u_{k-1}) \\
\hat{y}_k^- &= \sum_{i=0}^{2L} W_i^m Y_{i,k|k-1} \\
P_{y_k y_k}^* &= \sum_{i=0}^{2L} W_i^c (Y_{i,k|k-1} - \hat{y}_k^-)(Y_{i,k|k-1} - \hat{y}_k^-)^T + R_k \\
P_{x_k y_k}^* &= \sum_{i=0}^{2L} W_i^c (\chi_{i,k|k-1} - \hat{x}_k^-)(Y_{i,k|k-1} - \hat{y}_k^-)^T \\
K^* &= P_{x_k y_k}^* {P_{y_k y_k}^*}^{-1} \\
\hat{x}_k^* &= \hat{x}_k^- + K^*(y_k - \hat{y}_k^-) \\
P_k^* &= P_k^- - K P_{y_k y_k} K^T
\end{aligned}$$

h is the measurement function.The predicted state vector at time instant k, x_k^- and P_k^- are used to find sigma points which are represented together in by the matrix $\chi_{k|k-1}$. These sigma points are used in the measurement model to find predicted measurements, $Y_{k|k-1}$. The measurement sigma points are then used to find the mean y_k^-. The covariance matrix y_k and the cross-covariance matrix between x_k and y_k are found to find kalman gain(K^*) which is then used to find the preliminary updated state vector(x_k^*) and its error covariance matrix(P_k^*).

6. **Measurement noise covariance estimation:** The measurement noise covariance matrix is estimated using the available measurements and updated states(Das *et al.* (2014)).

$$\begin{aligned}
\chi_k^+ &= \begin{bmatrix} \hat{x}_k^* & \hat{x}_k^* + \gamma\sqrt{P_k^*} & \hat{x}_k^* - \gamma\sqrt{P_k^*} \end{bmatrix} \\
Y_k^+ &= h(\chi_k^+, u_{k-1}) \\
\hat{y}_k^+ &= \sum_{i=0}^{2L} W_i^m Y_{i,k/k-1} \\
res_k &= y_k - \hat{y}_k^+
\end{aligned}$$

The residual(res_k) is a measure of the difference in the estimated measurement and the measurement from the vehicle. Calculate the residual covariance as:

$$P_{res} = \frac{1}{k_w} \sum_{i=k-k_w+1}^{k} res_i \, res_i^T$$

(if, w_s is the window size then, $k_w = k$ if $k < w_s$ and $k_w = w_s$ if $k \geq w_s$)

$$\hat{R}_k = P_{res} + \sum_{i=0}^{2n} W_i^c (Y_{k_i}^+ - y_k^+)(Y_{k_i}^+ - y_k^+)^T$$

$\hat{R}_k$ is the estimated measurement noise covariance matrix.

7. **Measurement Update 2:**

$$P_{y_k y_k} = \sum_{i=0}^{2L} W_i^c (Y_{i,k|k-1} - \hat{y}_k^+)(Y_{i,k|k-1} - \hat{y}_k^+)^T + \hat{R}_k$$
$$P_{x_k y_k} = \sum_{i=0}^{2L} W_i^c (\chi_{i,k|k-1} - \hat{x}_k^*)(Y_{i,k|k-1} - \hat{y}_k^+)^T$$
$$K = P_{x_k y_k} P_{y_k y_k}^{-1}$$
$$\hat{x}_k = \hat{x}_k^* + K(y_k - \hat{y}_k^+)$$
$$P_k = P_k^* - K P_{y_k y_k} K^T$$

The estimated measurement noise covariance matrix is used to update the states and the error covariance matrix again. Steps 6 and 7 can be performed multiple times to make the estimates better. Here, it is only performed once because estimation has to performed real time.

8. **Finding algebraic states from differential states:** The algebraic states are again found from the differential states for maintaining the consistency of algebraic states in the model.

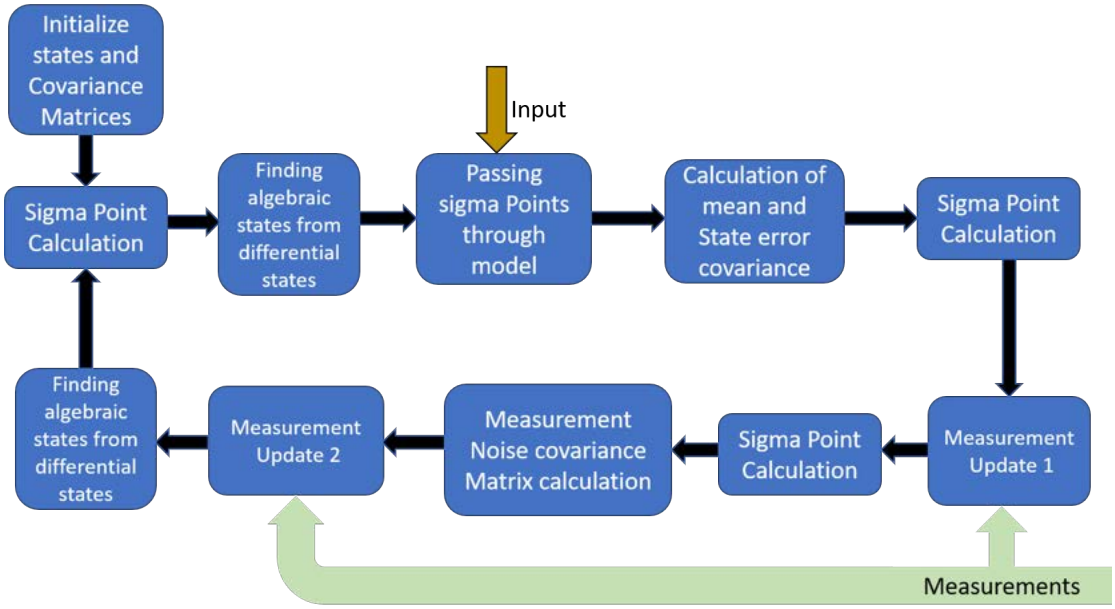

Figure 4.2: Flow diagram representing the UKF algorithm

CHAPTER 5

Parameter Estimation

All the parameters in the model built in chapter(3), can not be estimated using conventional methods because of various reasons. For example, the parameters B, C, D, E, S_h and S_v need sophisticated equipment for estimating them. For this reason the estimation of parameters is done using measurements obtained from the vehicle. That is by using the response of the vehicle for a known input given to it. The estimation is done by solving an optimization problem in parameter space.

5.1 Cost Function

The cost function used for the optimization problem is the root sum square error(RSSE) between the measurements from the rover and the estimated measurements from the model.

$$
\begin{aligned}
\min_{\theta} \quad & \sqrt{\sum_{j=1}^{P_n}(R\sum_{i=1}^{N}(Y_{ij}-\hat{y}_{ij})^2)} \\
\text{s.t.} \quad & \dot{x} = f(x,z,u,\theta,t) \\
& z = g(x,z,u,\theta,t) \\
& y = h(x,t) \\
& \theta_{min} \leq \theta \leq \theta_{max} \\
\text{where} \quad & \\
& x \in \mathbb{R}^{11\text{X}1} \\
& z \in \mathbb{R}^{4\text{X}1} \\
& u \in \mathbb{R}^{3\text{X}1} \\
& y \in \mathbb{R}^{9\text{X}1} \\
& \theta \in \mathbb{R}^{17\text{X}1} \\
& R \in \mathbb{R}^{9\text{X}9}
\end{aligned}
\tag{5.1}
$$

Where P_n is the total number of parameters to be estimated, N is the number of measurement samples, $Y^j_{mes_i}$ and $Y^j_{est_i}$ are measurements from the rover and estimated measurements, of the i^{th} measurement for the j^{th} parameter. $f(x, z, u, t)$ and $h(x, z, u, t)$ represent the DAE model for the vehicle. L and U are the upper and lower bounds for the parameters. R is a matrix used for scaling the error in preferred measurements.
With a given initialization of parameters and known initial states the DAE model is solved for N time intervals and at every interval the difference between the estimated measurement from the model and the measurements from the vehicle are squared and summed over all the measurements and time instants for computing its cost.

5.2 Optimization algorithm

The optimization is divided into two stages,

Particle Swarm Optimization: This is a global search method for finding the region in parameter space where the optimal parameters lie. A set of parameters that lie in the region are given as an initialization to the local search method.

Nelder-Mead(NM): This is a local search method that uses the initialization given by the global search algorithm and converges to the optimal parameters.

5.3 Particle swarm Optimization

The Particle swarm Optimization(PSO) is a swarm intelligence algorithm in which a swarm of particles are initialized stochastically (Kennedy and Eberhart (1995)). All the particles communicate with each other. Each particle has a position and a velocity which are updated at every iteration. The updation for every particle occurs at every iteration in such a way that the particles move in the direction which is the vector sum of the vectors from the particle and the best position reached by the particle, the best position reached among all the particles. A pictorial representation of this in two di-

mensions is shown in figure(5.1).

$$V_{id}^{n+1} = V_{id}^{n} + c_1 \times r_1 \times (pbest_{id}^{n} - x_{id}^{n}) + c_2 \times r_2 \times (gbest_{d}^{n} - x_{id}^{n})$$
$$x_{id}^{n+1} = x_{id}^{n} + V_{id}^{n+1}$$

Where, V is the velocity of the particle and x is its position. i is the particle number, n is

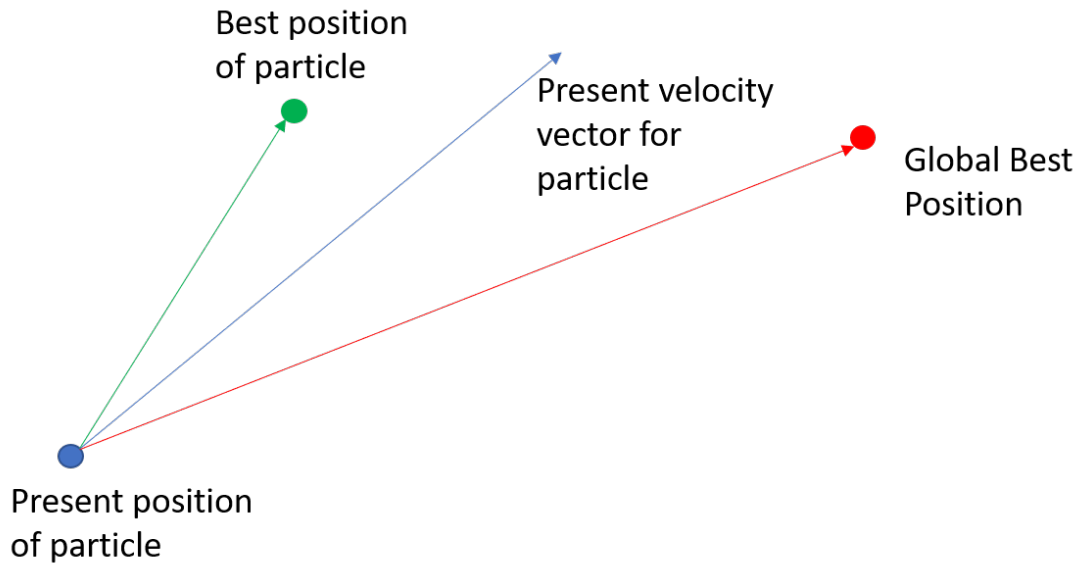

Figure 5.1: A two dimensional representation of particle swarm optimization algorithm

the iteration number and d is the dimension in the particle.r_1 and r_2 are random numbers that are uniformly distributed in the range[0,1]. c_1 and c_2 are learning factors. $gbest$ indicates the best position in the whole swarm and $pbest$ indicates the best position of the particle.

One of the problem with this algorithm is that it may prematurely converge to local minima. This problem will be very prominent in our case because of the stiffness in the model i.e. for most of the positions of particles(parameters) the cost function goes to infinity. Therefore, an extension to PSO, Comprehensive learning particle swarm optimization (CLPSO) is used ,(Liang *et al.* (2006)).

5.3.1 CLPSO

In this method the velocity is updated using a different position of particle for each dimension instead of the globally best position and the particle's best position. The

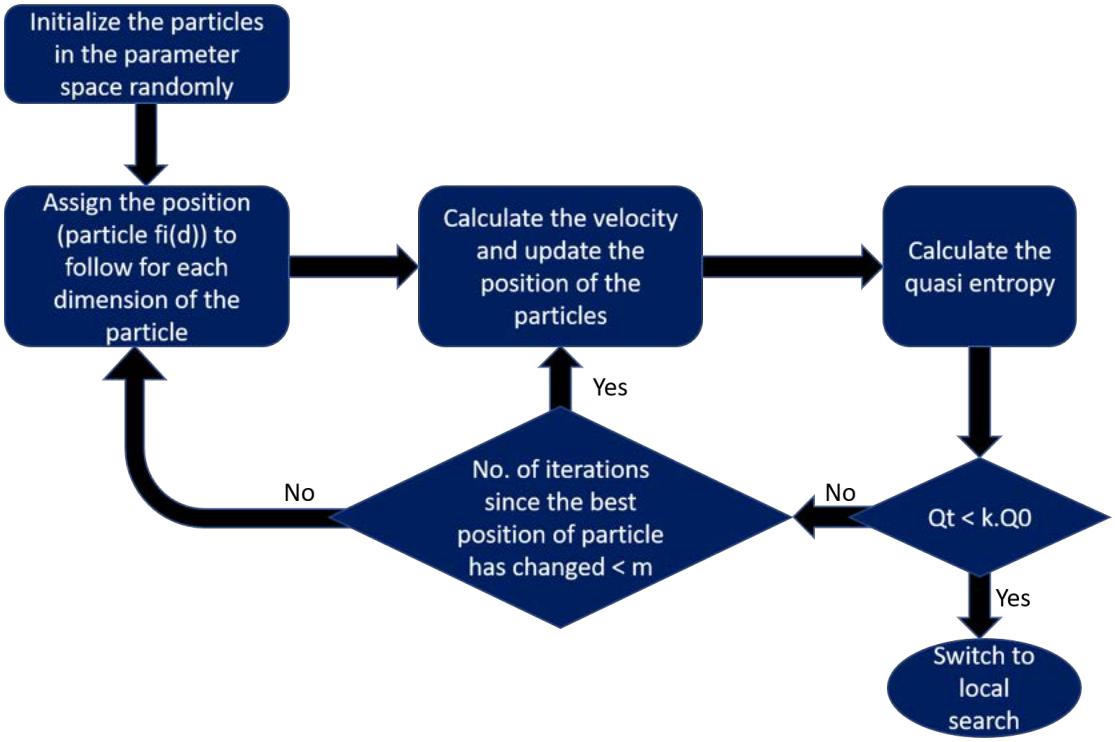

Figure 5.2: Flow diagram representing the algorithm of CLPSO

update policy is,

$$V_{id}^{n+1} = w * V_{id}^{n} + c \times r_1 \times (pbest_{fi(d)}^{n} - x_{id}^{n}) \tag{5.2}$$

Where, $pbest_{fi(d)}^{n}$ is the position that d^{th} dimension of particle i will move towards. This is selected differently for every dimension in each particle. The process of selecting $pbest$ is followed as given in (Liang *et al.* (2006)).A brief flow diagram of the algorithm used is shown in figure(5.2).

5.4 Nelder-Mead Optimization

This is a method is used for local search of parameters. The reason for using this method is because it does not require the calculation of derivatives.

The position of the globally best particle from the global search method is given as the initial value to it and a simplex with (n+1) particles is generated, where n is the number of parameters to be estimated. Then the vertices of this simplex are moved to obtain parameters that reduces the cost function(5.1). The algorithm used here is as given in (Singer and Nelder (2009)).

5.5 Switching

The switching can't be done after CLPSO converges because the amount time taken for it to occur is very large. So, a measure is used for finding the point where switching has to occur. Quasi-entropy(Q_n), (Cao *et al.* (2019)), a measure of how far off are the particles from the globally best particle. The measure is defined as follows,

$$Q_E^n = -\sum_{i=1}^{N} P_i^n \log(P_i^n) \tag{5.3}$$

where,

$$P_i^n = \frac{U_i^n}{\sum_{i=1}^{N} U_i^n}$$

and U_i^n is the cost of particle i at time instant n. The algorithm switches from global to local search after the Q_E^n is less than a fraction of the initial quasi entropy.

$$Q_E^n \leq \theta . Q_E^0$$

θ is a parameter to be tuned.

CHAPTER 6

Architecture

All the things discussed till now are put together for the estimation of model offline and state estimation.

1. Measurements and inputs are collected from the vehicle's sensors.
2. This data is used for the estimation of parameters.
3. Using the estimated parameters in the model, state estimation is done.

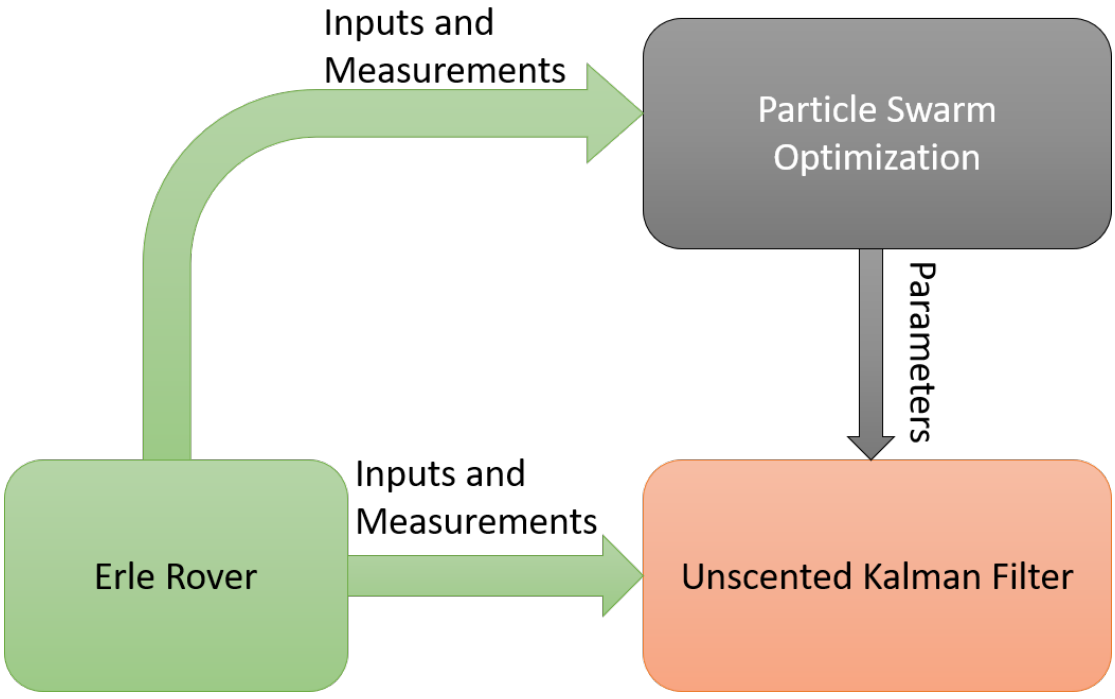

Figure 6.1: A flow Diagram representing the Data flow between processes.

CHAPTER 7

Simulations, Experiments and Results

The simulations were carried out in stages.

1. The performance of state estimation was checked using a simulation model built on matlab.
2. Available hardware was modified for data logging.
3. The data collected from an erle rover was used to test the complete architecture.

7.1 Response of the DAE system

Figures(7.1) - (7.6) are the state trajectories obtained for different inputs given to DAE system built in chapter(3). The DAE system was solved in Matlab using ODE15s solver.It was observed that the model is behaving ideally for the inputs given. So, this is model was used in parameter estimation and state estimation.

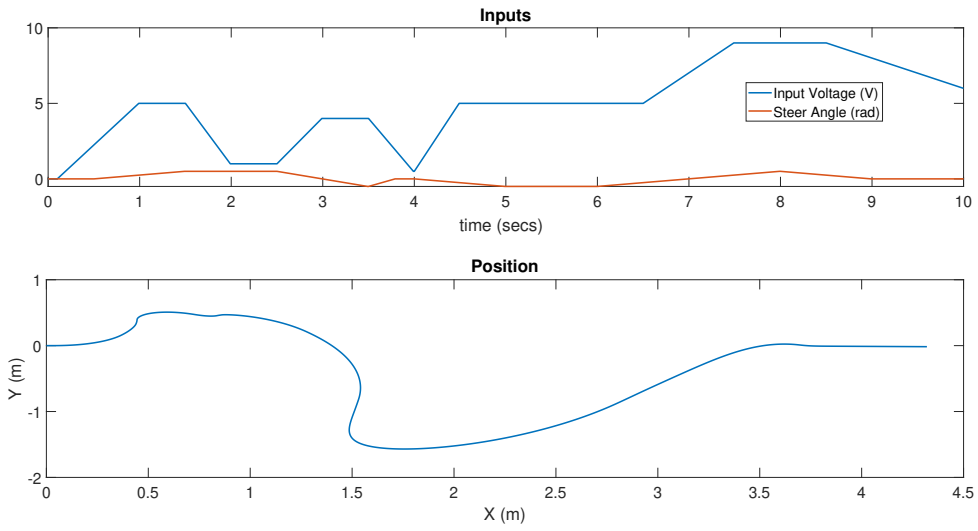

Figure 7.1: Position of the vehicle obtained from the model, the input is given in top plot

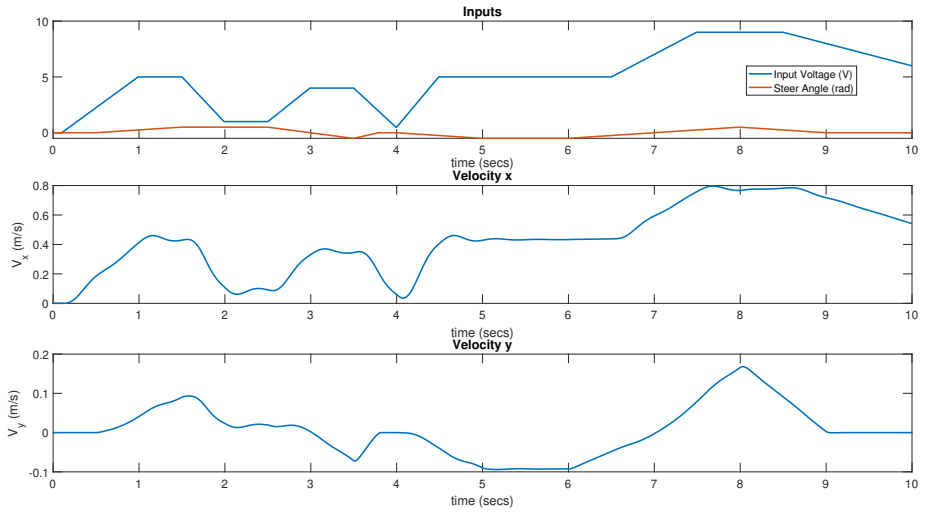

Figure 7.2: Velocity of the vehicle obtained from the model, the input is given in top plot

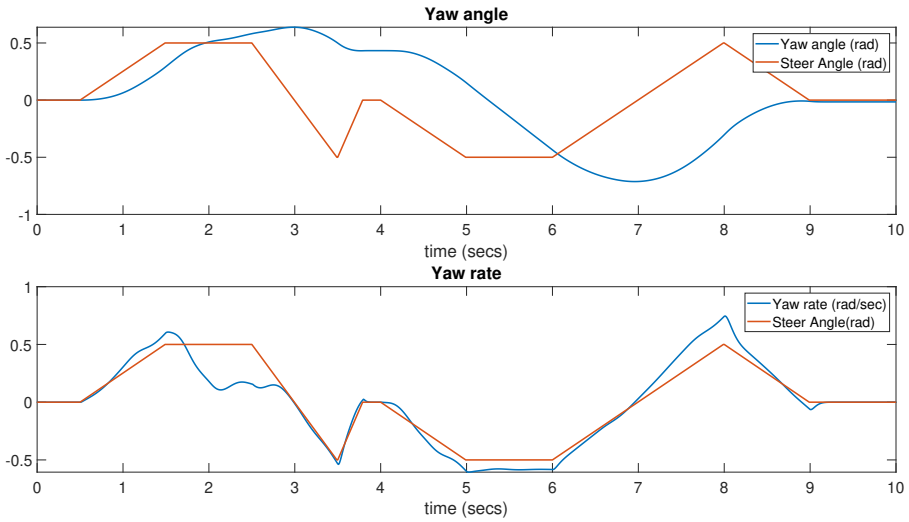

Figure 7.3: Yaw and Yaw rate of the vehicle obtained from the model

7.2 State estimation with simulation model

A simulation model built in section(7.1) was used to give the measurements at every time instant and noise was added to these measurements. These noise measurements and the inputs given to the simulation model are given to UKF in which the state estimation is performed. A flow diagram depicting this is shown in figure(7.7). The estimation results obtained are shown in figures (7.8) - (7.11). In the figures, the red curve represents the states from the simulation model, blue curves represent the states from UKF and yellow curves represent states estimated from the measurements of the

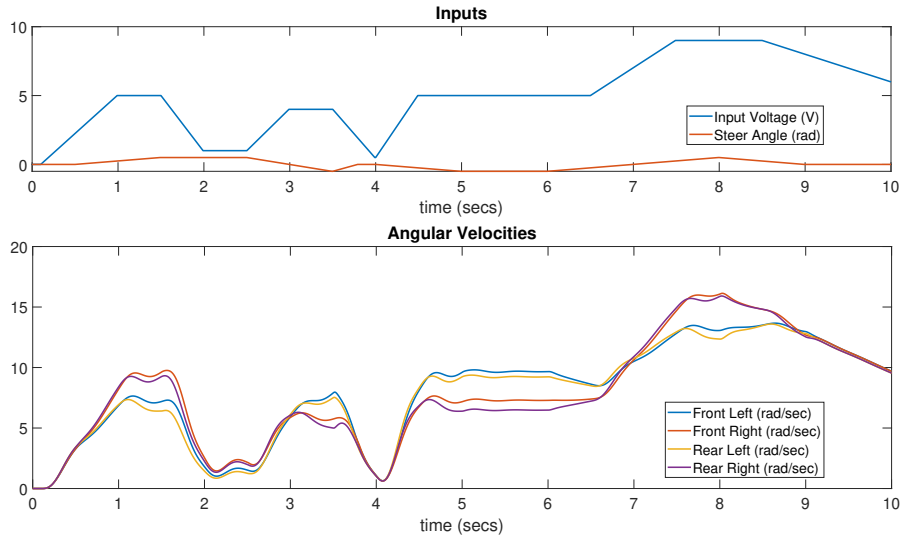

Figure 7.4: Angular velocity of the wheels obtained from the model, the input is given in top plot

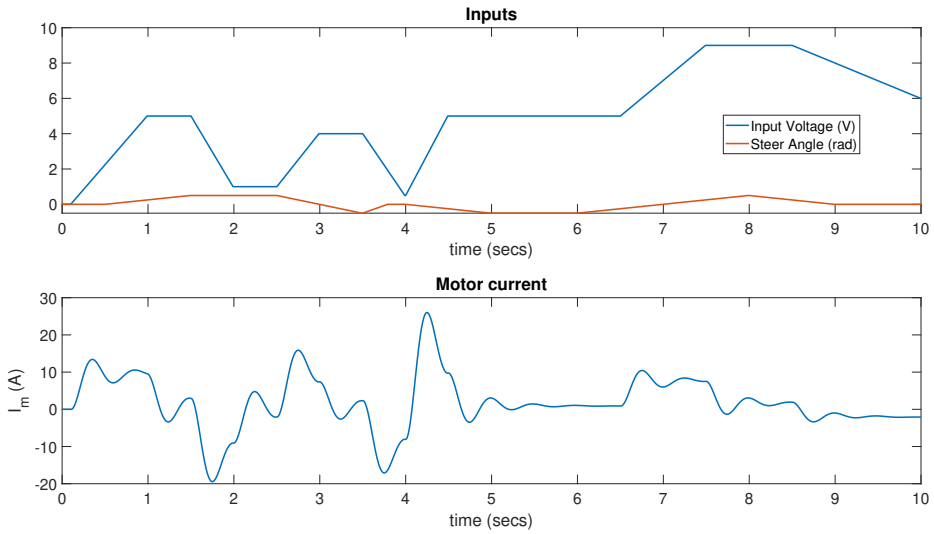

Figure 7.5: Current in the coil of the rear drive motor obtained from the model, the input is given in top plot

simulation model without any filter by using only the measurements. It can be observed that the estimation of filter is much better when compared to the case without filter. The RMS error values between the states of the simulation model and states of UKF, states estimated without filter are given in table(7.1, it can be observed that most of the RMS errors between states of simulation model and UKF are almost zero this because the simulation model and the model in UKF are identical. It can also be observed that the RMS error of position and velocity of the case without filter is large but in the case of UKF it is very small, this indicates that having a good model for the system improves

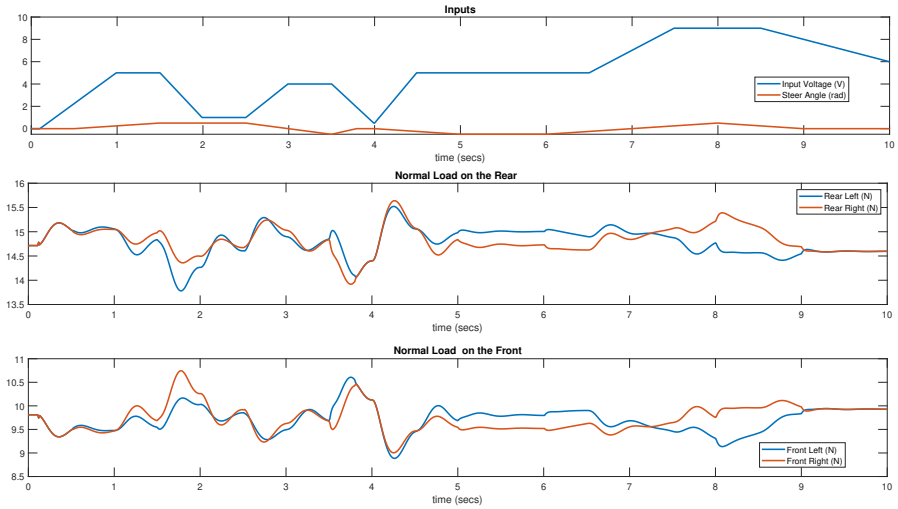

Figure 7.6: Normal Load on the wheels obtained from the model, the input is given in top plot

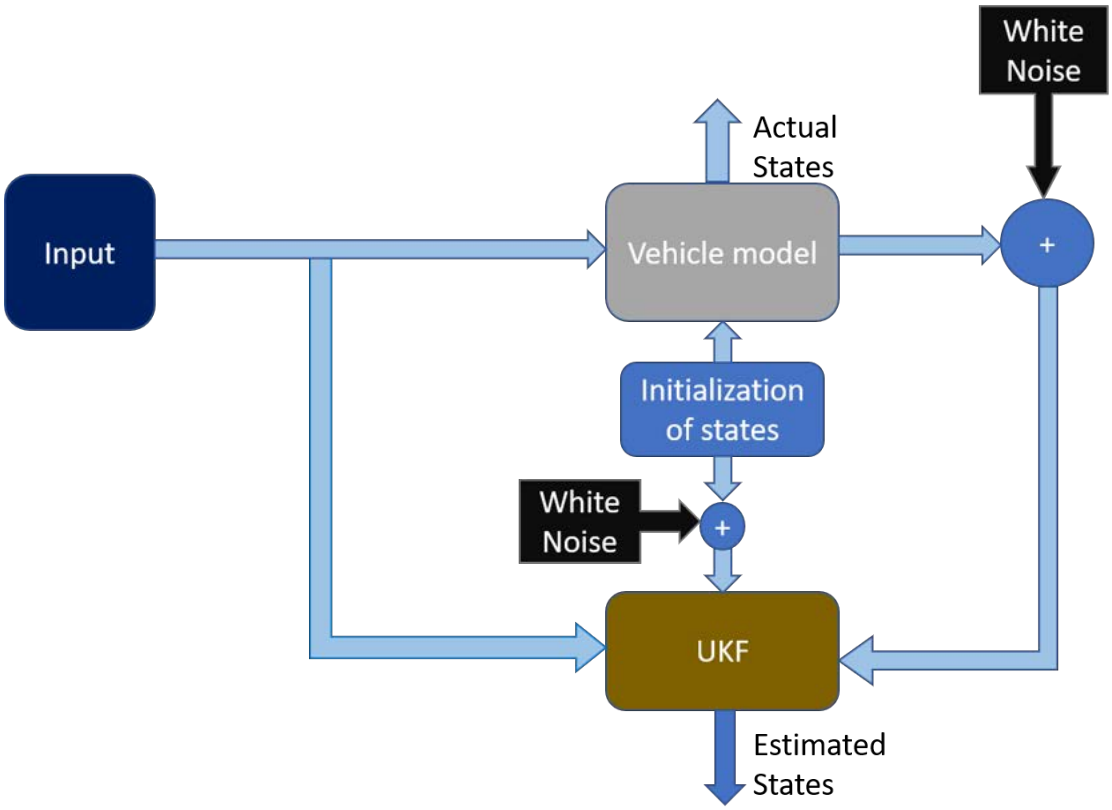

Figure 7.7: Flow diagram representing the implementation of simulations

the estimation. Although, in real case the model built for the vehicle and the behaviour of the vehicle in real world will not be identical as in the case of these simulations but the model built is closer to real world behaviour than the case of without the filter. Therefore, using UKF for estimation does improve the accuracy of state estimation.

Table 7.1: The RMS error values between the states of the simulation model and states of UKF, states estimated without filter. Where XxDiff is the RMS error between the states of simulation model and UKF, XintiDiff is the RMS error between states of simulation model and estimated states using only measurements without filter, and Compare is the difference between XxDiff and XinitDiff.

State	XxDiff	XintiDiff	Compare
sx	0.0032	2.4587	-2.4555
sy	0.0033	2.5806	-2.5772
vx	0.0000	0.0944	-0.0944
vy	0.0000	0.0932	-0.0932
ψ	0.0000	0.0001	0.0003
$\dot{\psi}$	0.0000	0.0000	-0.0000
ω_{fL}	0.0000	0.0004	-0.0003
ω_{fR}	0.0000	0.0004	-0.0004
ω_{rL}	0.0000	0.0004	-0.003
ω_{rR}	0.0000	0.0004	-0.004

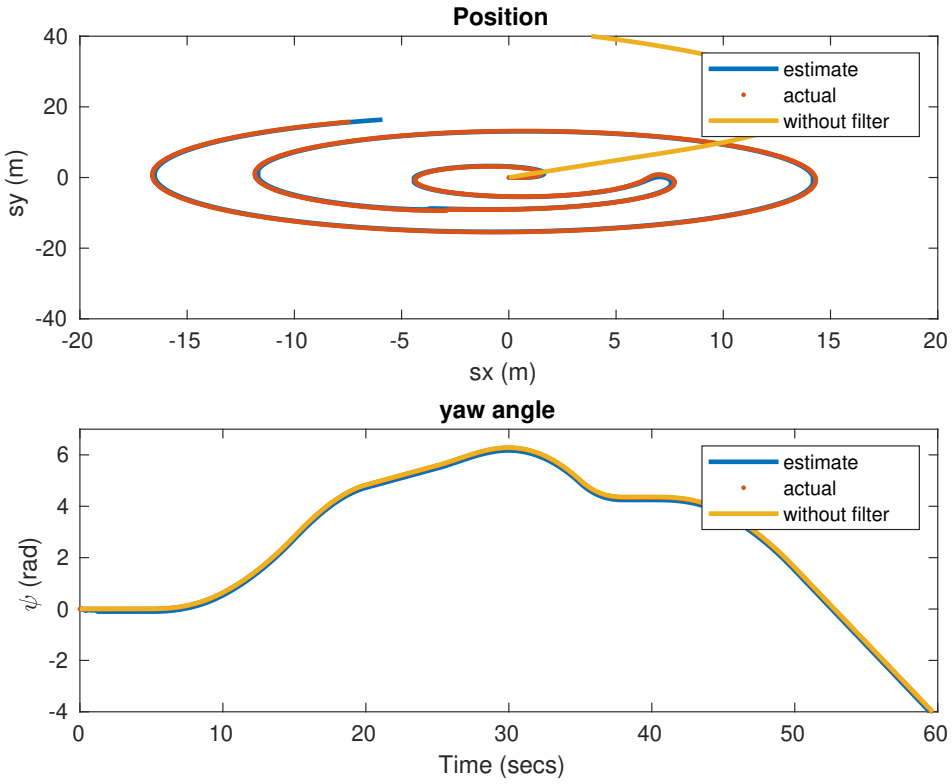

Figure 7.8: Position and Yaw angle in state estimation. Red plot represents the states from simulation model, blue plot represents the estimated states from UKF and Yellow plot represents states when they are directly without filter.

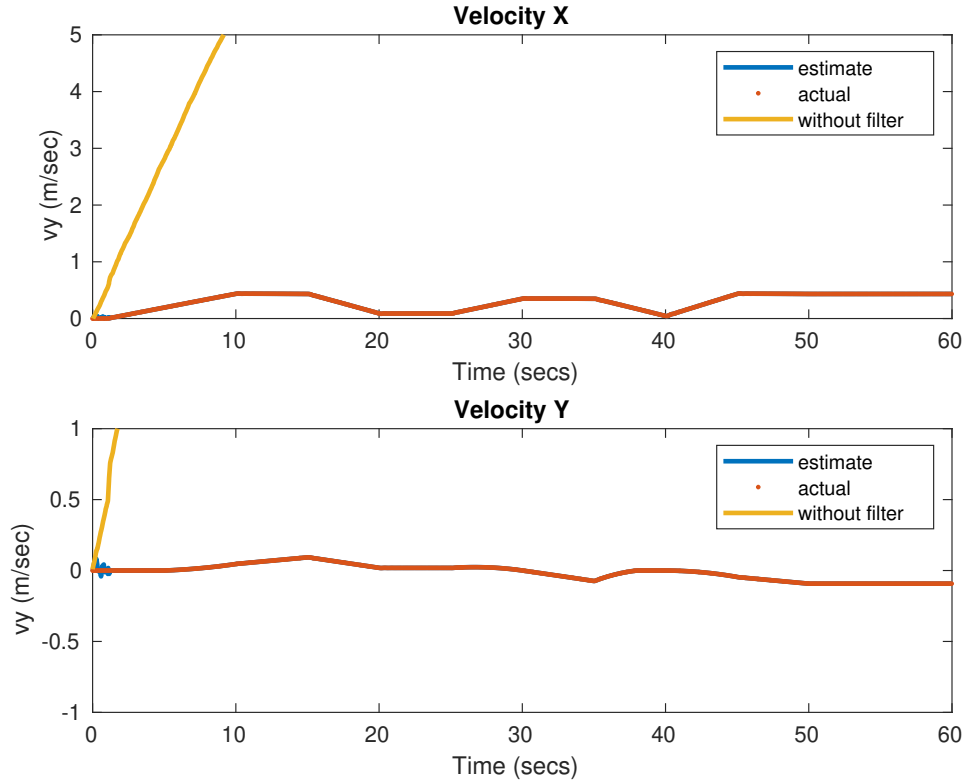

Figure 7.9: Velocity in state estimation. Red plot represents the states from simulation model, blue plot represents the estimated states from UKF and Yellow plot represents states when they are directly without filter.

7.3 Hardware for data logging

The vehicle used for the experiments is Erle rover(7.12). It is a rear drive vehicle, it comes with a micro-controller, Erle brain3(EB3) which is raspberry pi based. Steering of vehicle is done using a servo motor.

Erle brain 3 has an IMU in it which was used for obtaining accelerations, yaw rate and yaw angle. For controlling the motor an ESC was provided which was replaced with a motor driver from Pololu. This motor driver has a current sensor to it which was used for getting the current in the motor coil. For angular velocities infrared sensors and hall effect sensors were fixed to the wheels as wheel encoders.

An Arduino mega micro controller is used to collect the data from wheel encoders, motor current from motor driver, given inputs to motor driver and inputs to servo motor for steering.This data is then sent to Erle brain 3 for data logging.

Erle-brain 3 receives the inputs to be given to the motor and servo from a RC transmitter. It then sends this data to Arduino mega and receives the sensor information from

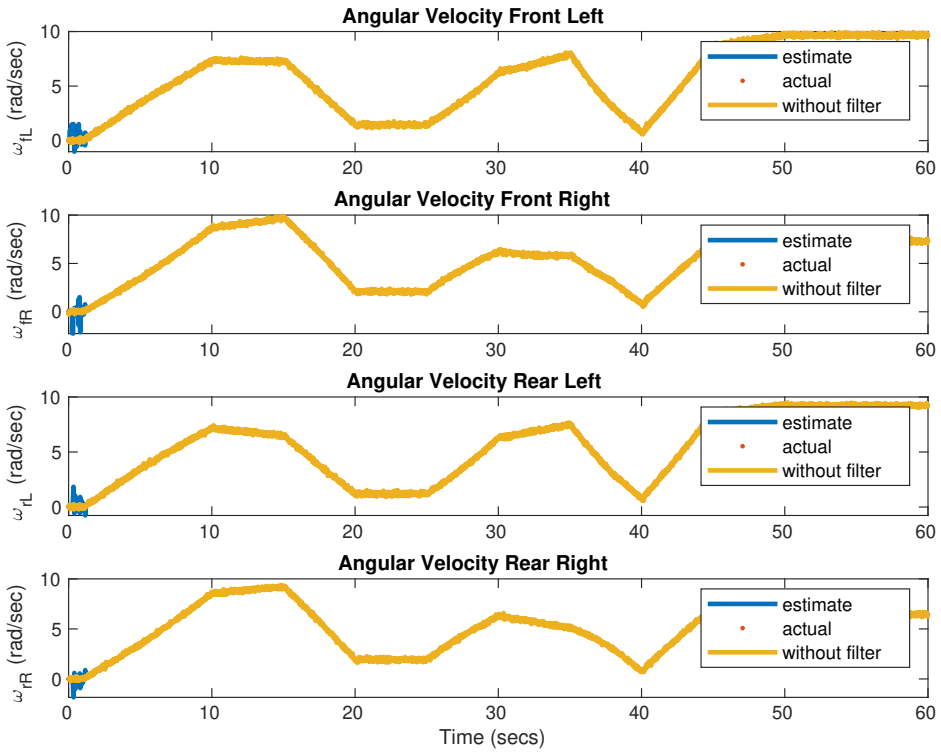

Figure 7.10: Angular Velocity of the wheels in state estimation.Red plot represents the states from simulation model, blue plot represents the estimated states from UKF and Yellow plot represents states when they are directly without filter.

it. This sensor information appended with the acceleration, yaw rate and yaw angle data from IMU are logged into a file in a USB attached to it. The Erle-brain 3 can be accessed wirelessly through WIFI by using ssh. All the programs on it run as ROS nodes, this is done because the data logging and input to the vehicle has to be done simultaneously. Arduino mega and EB3 communicate with each other using a serial communication. A pictorial representation of this system is shown in figure(7.13).

The data from the USB is then used for Parameter estimation and testing the performance of UKF offline.

7.3.1 Plots of data collected from Hardware for estimation

The data collected from the rover after smoothening and removing bias are shown in figures (7.15) - (7.22).

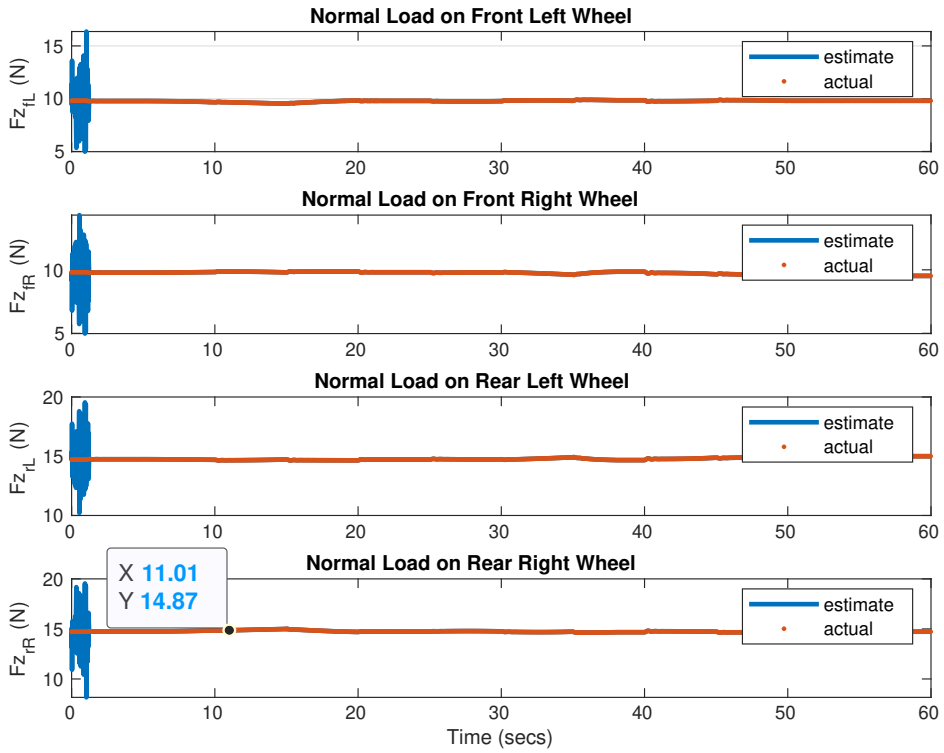

Figure 7.11: Normal Load in the wheel in state estimation. Red plot represents the states from simulation model and blue plot represents the estimated states from UKF.

7.4 Relationship between servo input and steering angle

The inputs to the model are steering angle and motor voltage. The input to the Erle rover is servo ppm input and not steering angle. A relationship between servo input and steering angle was found. This was found by curve fitting.

The steering angle response of the vehicle for a given servo input was collected. This data is used to find mathematical model that depicts the relationship between them.

The steering angle was measured by finding the yaw angle at each of the wheels for a given servo input. This was done because measuring the steering angle directly was not possible. These measurements are done on each wheel for 5 times with inputs to servo motor going from a lower bound to upper bound and the reverse. Data collected from multiple experiments was averaged and used for curve fitting.The experimental setup used for this is shown in figure (7.23)

As it can be seen from figure(7.24) that there is hysteresis in the response. There-

Figure 7.12: Erle Rover used for data logging

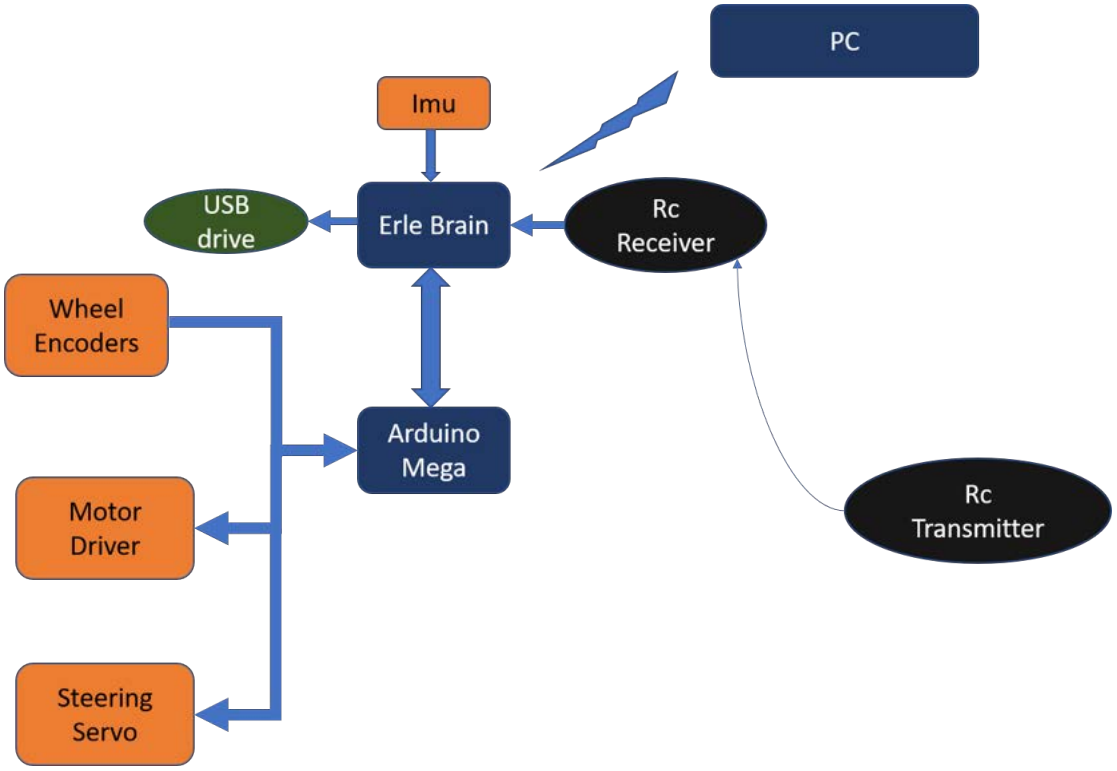

Figure 7.13: Communication among sensors, Micro-controllers, PC and Rc controller

fore, a switching model was used for the response, i.e. two different models, one for increasing servo input and the other for decreasing servo input.

$$\delta = -0.000073(x - 55)^3 + 10, \qquad \text{when servo input is decreased from its current value} \tag{7.1a}$$

$$\delta = 0.000077(124 - x)^{2.9} - 10, \qquad \text{when servo input is increased from its current value} \tag{7.1b}$$

Where, δ is the steering angle and x is the servo input. Figure(7.24), shows the estimated model and the true data collected from the experiments.

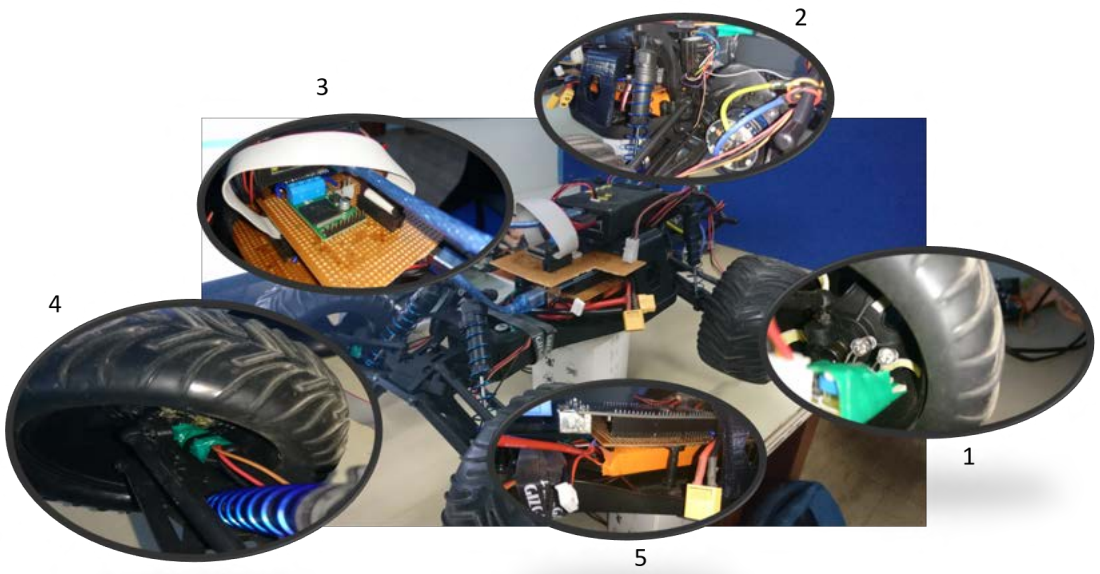

Figure 7.14: Figure with sensors and micro-controllers on Erle Rover. 1) IR Wheel encoders 2) RC receiver 3) Motor Driver 4)Hall Effect Wheel encoders 5) Arduino Mega

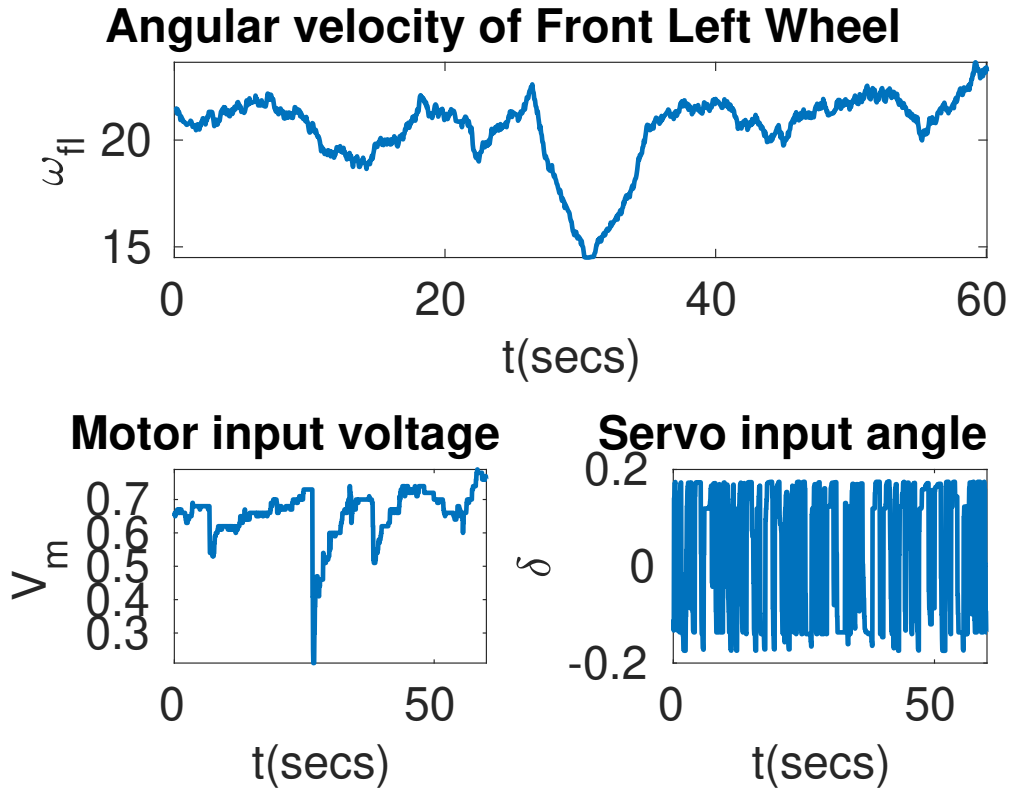

Figure 7.15: Angular velocity data of front left wheel, from the rover.

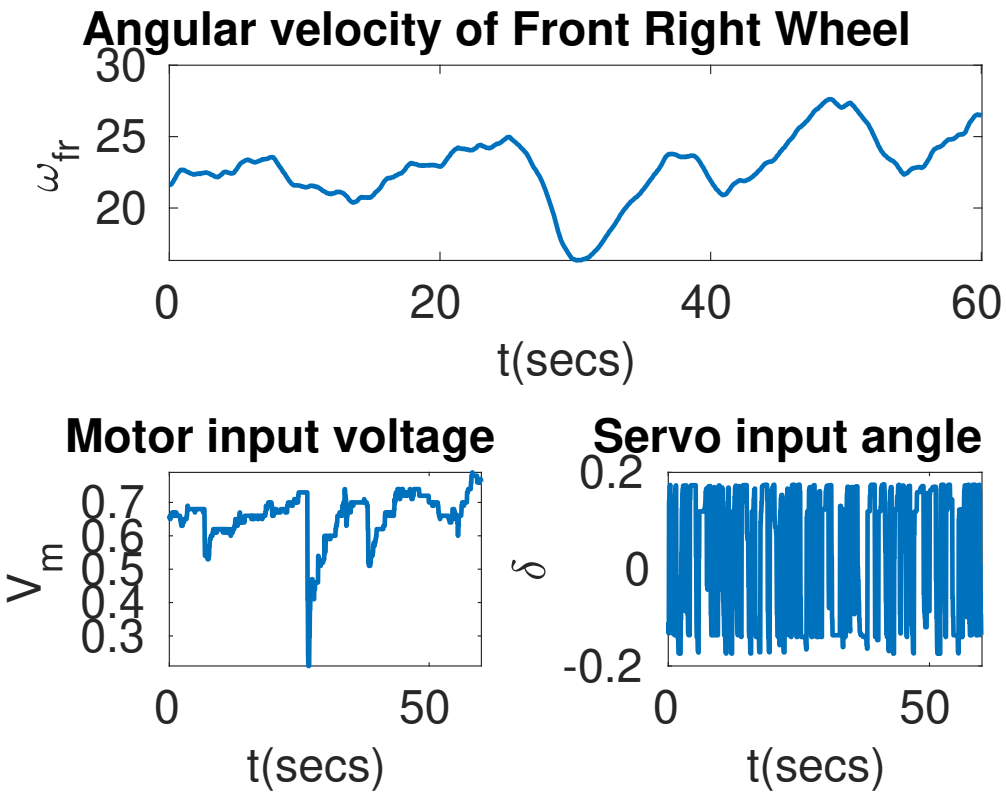

Figure 7.16: Angular velocity data of front right wheel, from the rover.

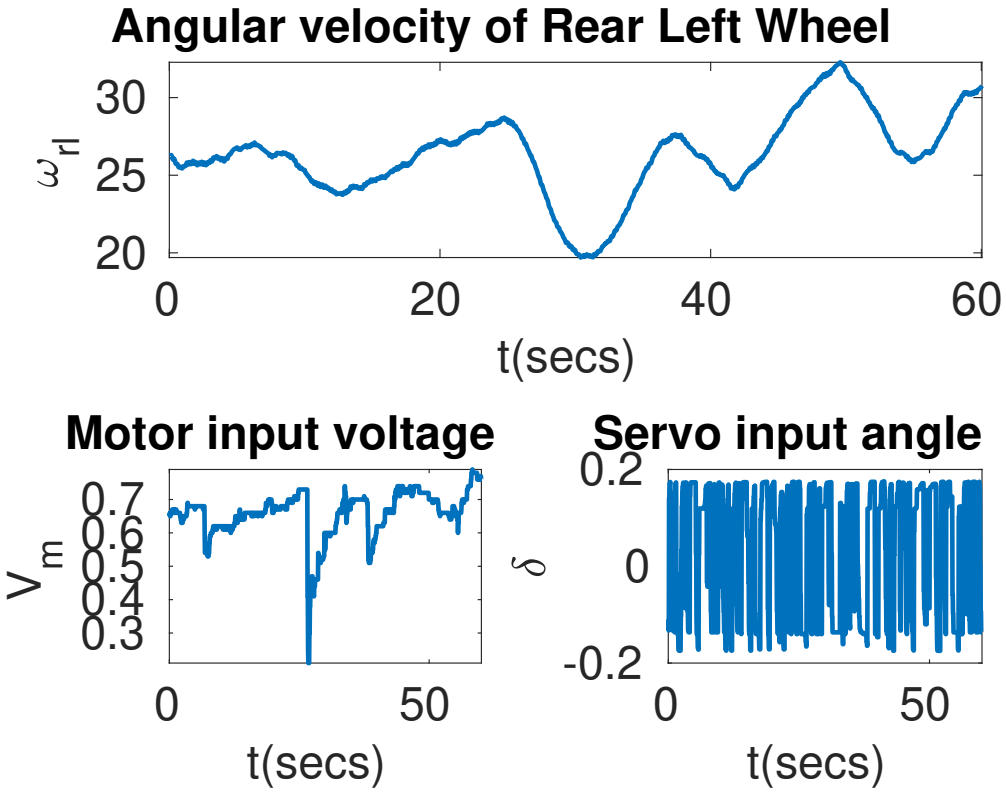

Figure 7.17: Angular velocity data of rear left wheel, from the rover.

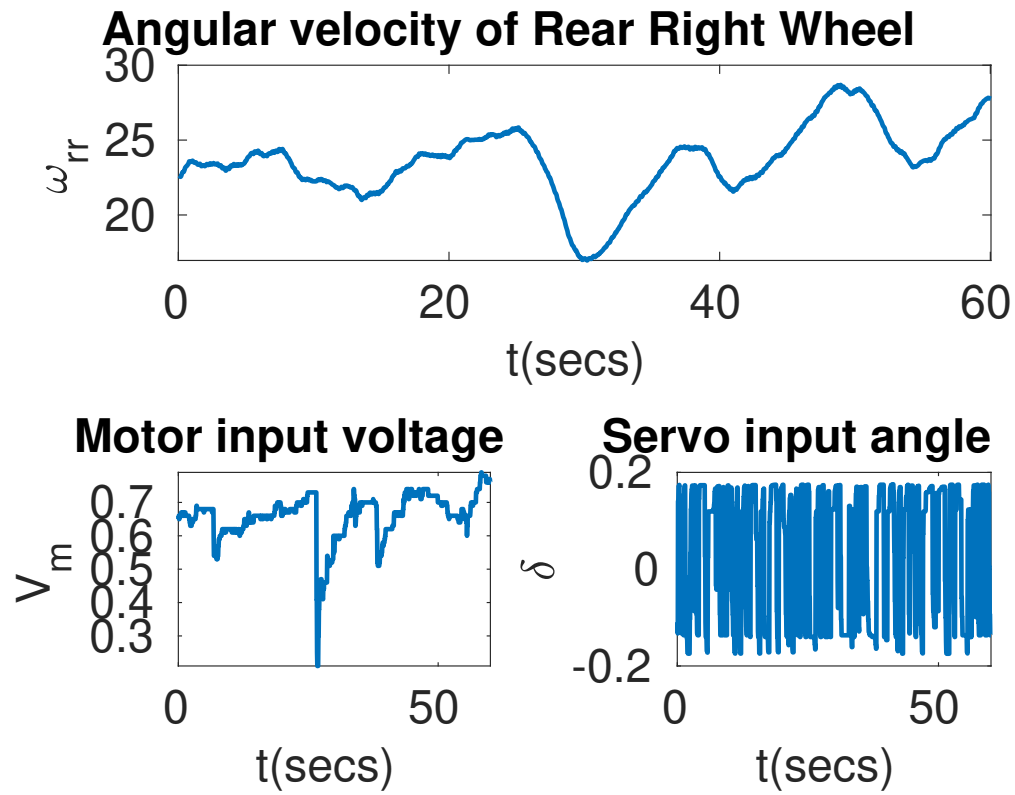

Figure 7.18: Angular velocity data of rear right wheel, from the rover.

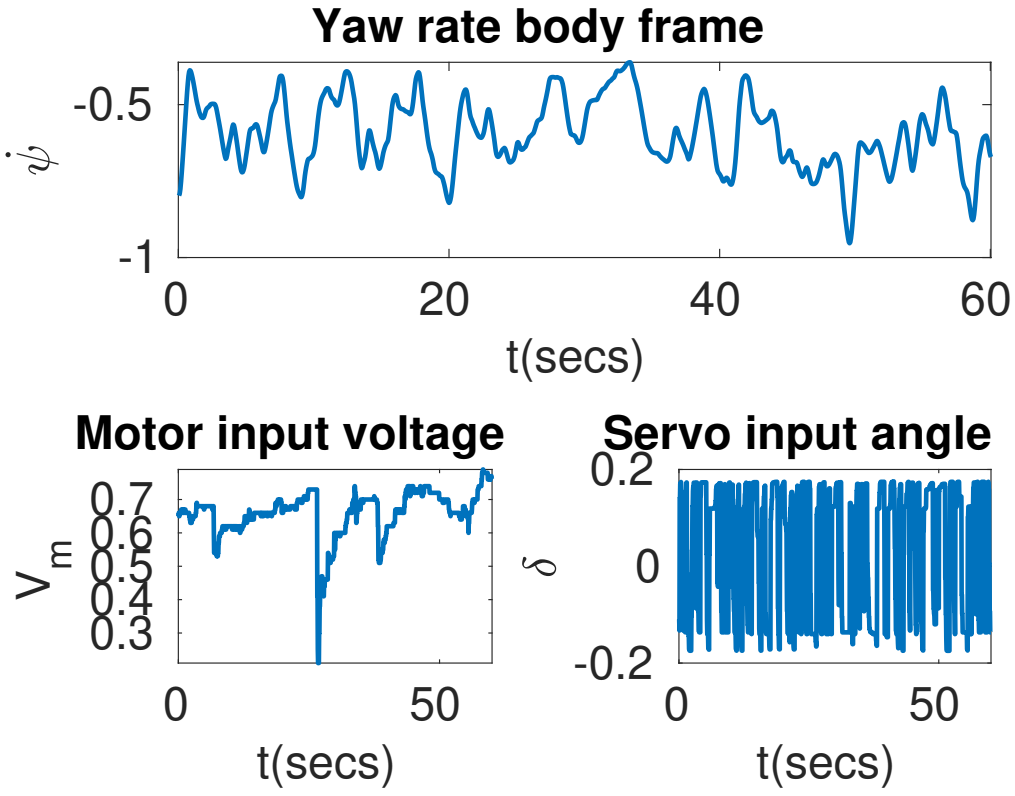

Figure 7.19: Yaw rate data from the rover.

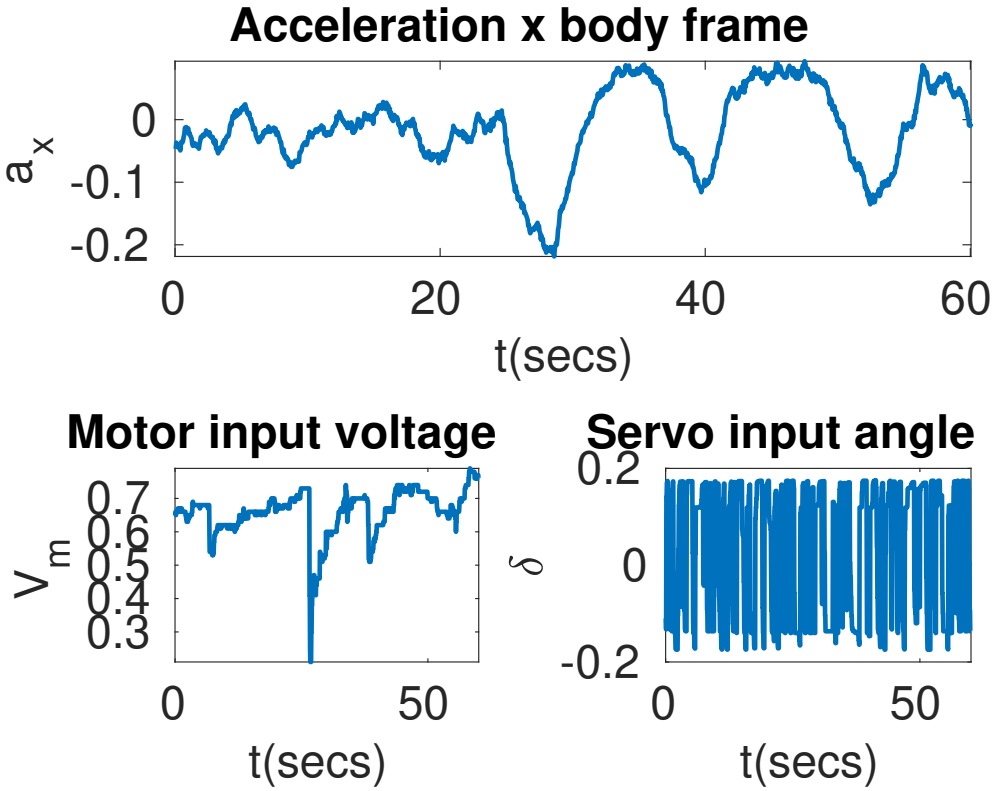

Figure 7.20: Acceleration along x direction w.r.t body frame, from the rover

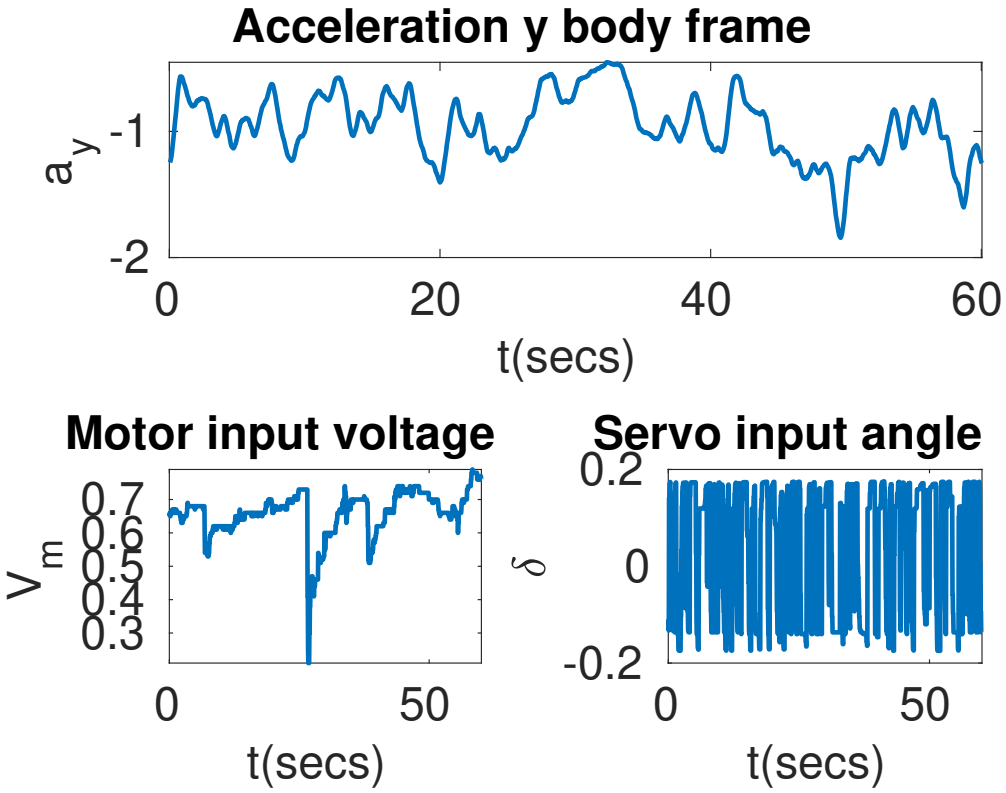

Figure 7.21: Acceleration along y direction w.r.t body frame, from the rover.

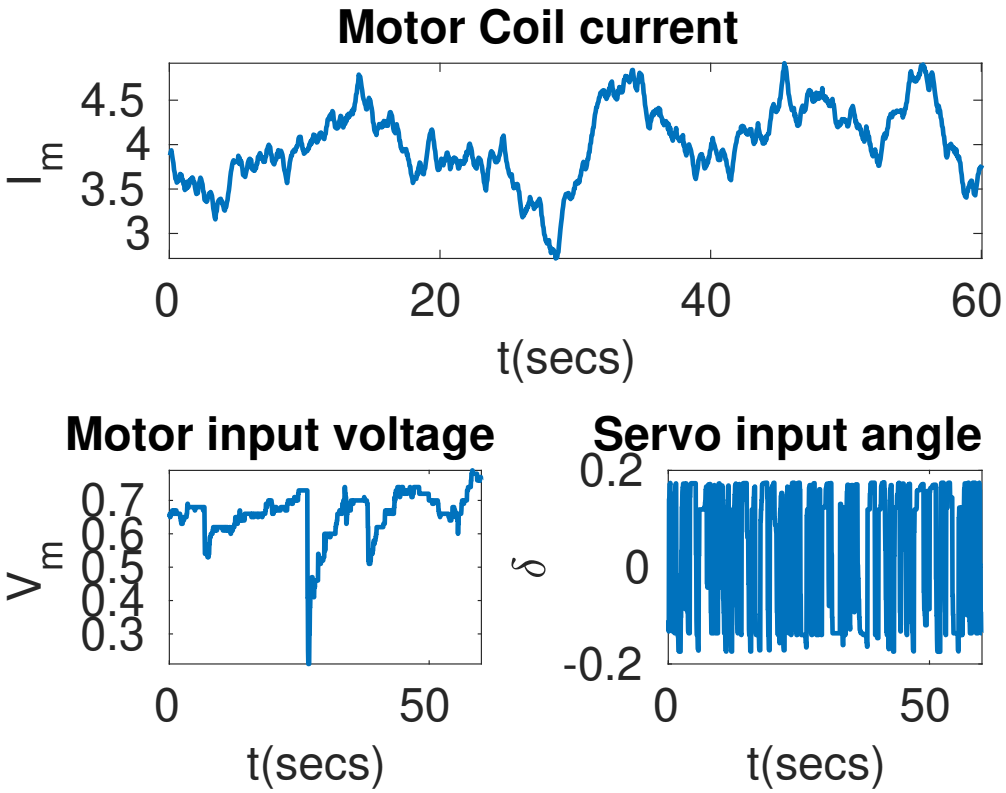

Figure 7.22: Current in the coil of the rear drive motor, from the rover.

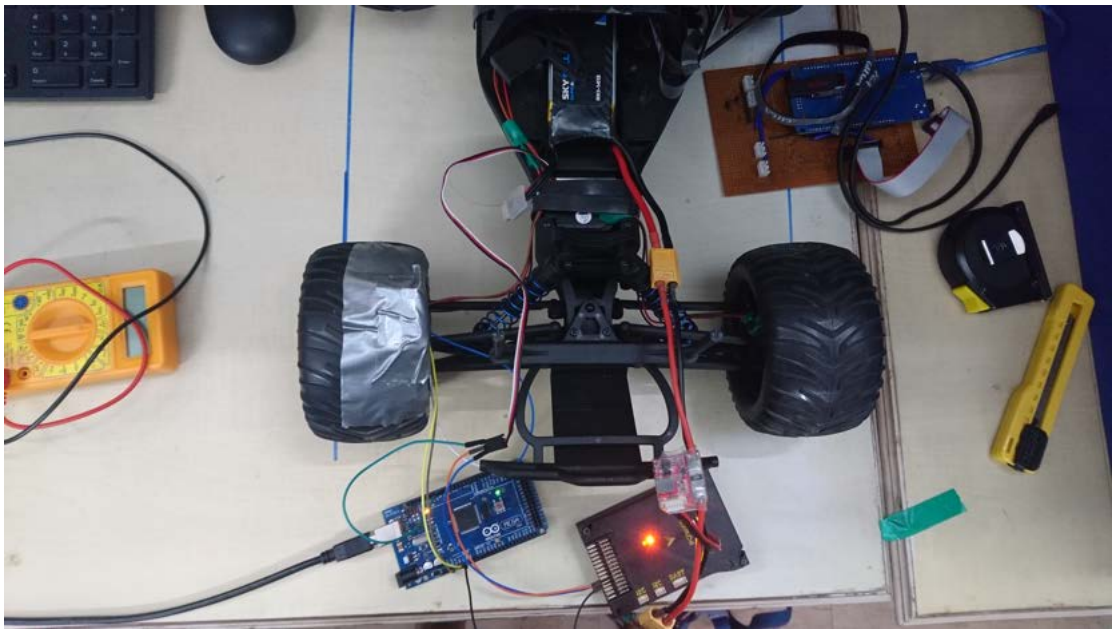

Figure 7.23: Experimental Setup for Steering angle model estimation

7.5 Parameter estimation with data from Erle rover

7.5.1 Parameter Space

The parameters present in the model built in chapter(3) are,

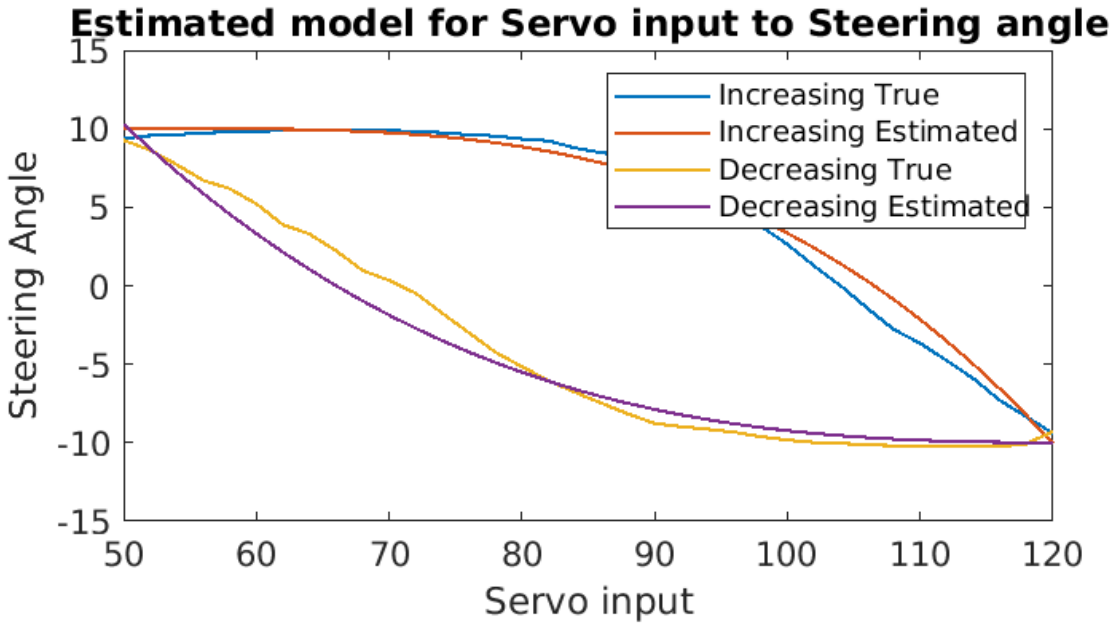

Figure 7.24: Plot of servo input vs steering angle with collected data and the estimated model

B, C, D, E, S_v, S_h - Tire parameters to be estimated and bounds are set as [0,10].

R_w - Radius of the tires is given in the data sheet of Erle rover but it is in resting condition with no additional load on it. So this value was used for setting the bounds as [0.01, 0.1] and the effective radius of the wheel was estimated.

J_w - Moment of Inertia of wheels was estimated and the bounds are set as [0.000001, 0.1].

D_r - Roll coefficient, was estimated and can't be measured using conventional techniques. While modelling it was know to vary between 0 and 1. So, the bounds are set as[0,1].

M_v - Mass of the vehicle, was measured to be 2.385 Kg.

l_f, l_r - The distance of center of mass to the front and rear axles. The wheel base of the rover was given as 29.5 cm in the data sheet of the vehicle. Therefore, one of these two parameters was estimated and the other was found by using the constraint that sum of the two parameters has to be equal to wheel base.

h - Height of center of gravity from the ground, was estimated and the bounds are set as [0.01, 0.2].

aA, aB - The distance of center of gravity from the front and rear axles, can be found if the normal loads on each wheels in rest condition is known. The normal loads are substituted in normal load constraints to find the parameters. In the model aA and aB only appear along with total aerodynamic drag force F_{az}, this is assumed to be zero in the experiments. Therefore, they are not estimated

w_f, w_r - Half track widths of front and rear axles. The values given in the datasheet are used for this, which is 13.6 cm.

$J_m, B_m, L_m, K_m, R_m,$ - Motor Parameters are estimated as there was no data sheet given for the motor. The bounds are set as [0, 1] for all the parameters.

G - Gear ratio, was given in the data sheet which was 11.93 and the same was used.

J_z - Moment of inertia of the vehicle about z axis, was estimated and the bounds are set as [0.001, 2].

So, a total of 17 parameters were estimated.
For Initial states, the vehicle was always started from rest condition. So, all the initial states except normal loads can be taken as zero. As for the initial normal loads they are calculated based on the l_f and l_r chosen for the particle and using the normal load constraints.

7.5.2 Optimizations done in code

When the cost function is calculated, it involves solving the DAE for N time instants. This process occurs serially and can not be done in parallel. Usually, N is set as 6500. This has to be done for every particle at every iteration. The number of particles are set around 40 and The number of iterations are around 5000. So, the evaluation of DAE is done for 6500x40x5000 times i.e. 1.3×10^9 and all the other operations do not take that significant time. This made the code to take days to complete.
For speeding up or optimizing the code two methods were employed. They are,

1. Evaluating the cost functions of particles in parallel. This was done by using multiple cores available on the PC. Each particles cost function was made to be evaluated in one core. The speedup achieved is dependent on the number of cores present in the PC.

2. Using Cython and optimizing the evaluation of DAE system. The function that evaluates and returns the cost was written in Cython and appended to the main code. The time taken by the code for one iteration was brought down from around 600 seconds to around 50 seconds.

7.5.3 Results

The estimated parameters are given in table(7.2). The CLPSO algorithm was run for 5000 iterations and then switched to NM local search. Figure(7.25) shows the progress of CLPSO algorithm. The plot on the top is the least cost function or the cost of globally best particle after every iteration. The plot on the bottom is the plot of entropy of the particles over iterations.

Table 7.2: Estimated parameters from the parameter estimator

S.no	Parameter	Estimated value
1	B	1.000e+01
2	C	2.355e-03
3	D	2.315e-03
4	E	1.928e-03
5	S_v	2.458e+00
6	S_h	4.048e-03
7	J_w	8.045e-02
8	R_w	1.089e-02
9	D_r	1.210e-02
10	l_f	1.475e-02
11	h	1.000e-02
12	J_m	3.486e-04
13	B_m	1.014e-08
14	L_m	2.225e-01
15	K_m	2.405e-03
16	R_m	5.514e-02
17	J_z	1.059e+00

7.6 State estimation with the estimated parameters.

The UKF state estimator described in section(4.1), has been tested here using real data. The parameters estimated in the previous section, shown in table(7.2), have been used to complete the model of the vehicle used in the state prediction step in the algorithm

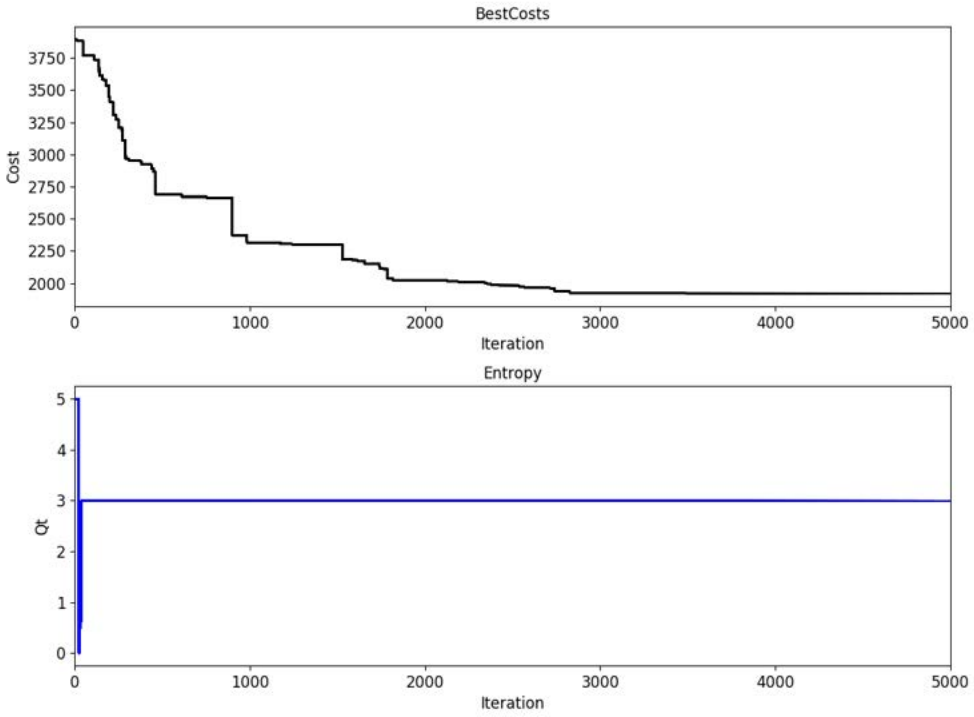

Figure 7.25: The progress of simulation over iterations.

in section(4.1). The real data used here is from a different experimental run than the one used for estimating the parameters. The vehicle starts from rest and the inputs are given such that it maintains a uniform motion along a predetermined path. The results obtained are discussed for each state in the following subsections.

7.6.1 Velocity (vx, vy)

The estimated velocities of center of mass w.r.t to body frame of the vehicle have been shown in figures(7.26 - 7.30). In figures(7.26), (7.28) two cases have been shown, yellow plot labelled as 'without filter' represents the states estimated by using a basic model for estimating velocities i.e. by integrating the obtained accelerations along each directions and blue plot labelled as 'estimate' represents the estimated velocities from UKF. It can be observed that the velocities in the case of without filter are increasing almost linearly along one direction with time, this is because of the noise in the acceleration data used. Whereas, the estimated velocities from UKF are contained and not increasing with time. Figures(7.27, 7.29) represent the estimated states alone. The validity of the estimated states can be checked by using some sanity checks as velocity

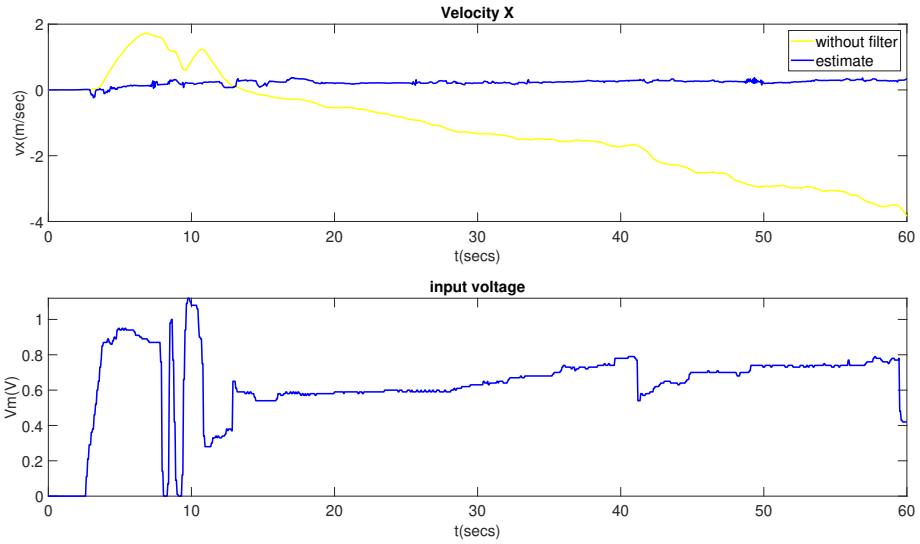

Figure 7.26: Velocity along X direction w.r.t body frame and the input voltage to the motor

of the vehicle was not available as a measurement.

Velocity along x direction(vx)

It can be said that velocity along x direction has to to mimic the input voltage given to the motor and the experiments were performed, the vehicle was operated such that the speed of the vehicle is maintained almost constant in periods of time. Observing figure(7.27), it can be said that the velocity of the vehicle does satisfy the conditions mentioned. Therefore, the estimate may be close to the actual value.

The vehicle was always operated in forward direction, therefore vx has to be positive always but when the vehicle starts from rest around 5 secs, it can seen in figure(**??**), the velocity is negative and in figure(7.29), it can be seen that the vy reaches a peak value of around 0.6 m/sec which is not possible because the vehicle has just started.When the vehicle starts from rest the transients that occurs are not accurately represented by the model built in chapter(3) because of this vx is estimated as negative and there is a spike in estimated vy.The estimate of vy from 10 seconds has been shown in figure(7.30) to clearly represent the variation of vy during uniform motion.

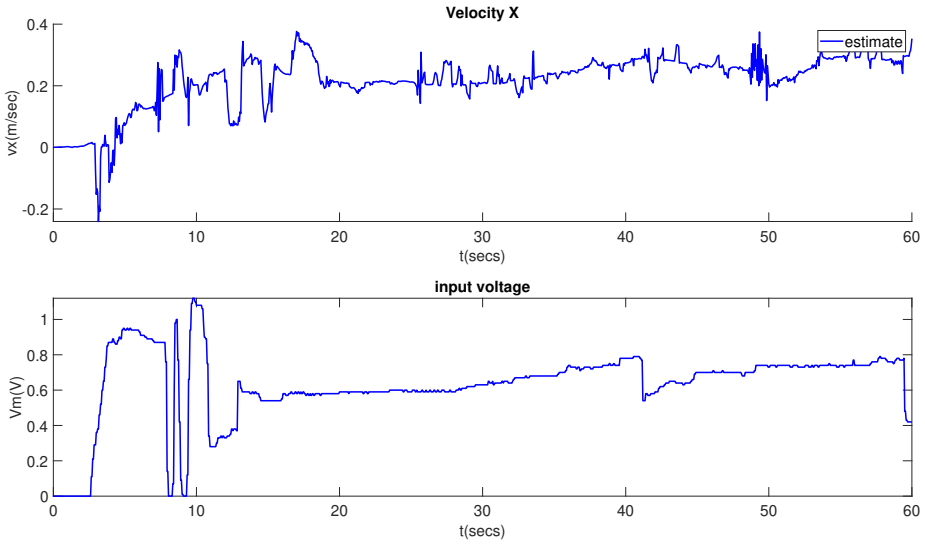

Figure 7.27: Velocity along X direction w.r.t body frame and the input voltage to the motor

Velocity along y direction(vy)

The velocity along y direction has to mimic the steering angle input and longer the steering angle input is applied the higher magnitude of velocity reached along the direction of steering. It can observed that velocity along y direction shown in figure(7.30) does satisfy these conditions. Therefore, the estimates might be true.

7.6.2 Angular Velocity(ω_{ij})

Figure(7.31), represents the angular velocity of the wheels of the vehicle. Yellow plot represents the measurements labelled as 'measurements', obtained from the wheel encoders directly and blue plot labelled as 'estimate' represents the estimates from UKF. It can be observed that measurements are noisy and has spikes at various locations, whereas the estimate plot is smooth with no irregularities in the middle as mentioned earlier regarding the transients when the vehicle starts from rest, the model is not accurate for that situation but it can be seen that the estimates are close to the measurements. This is because of the adaptive estimation of measurements noise covariance matrix, when the model is not accurate noise covariance was reduced to give more weight to the measurements when compared to model.

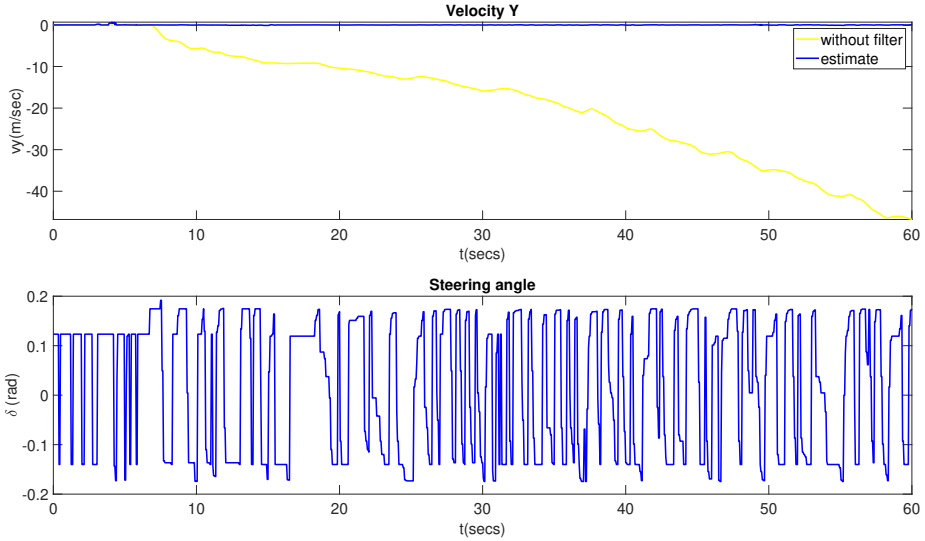

Figure 7.28: Velocity along Y direction w.r.t body frame and the input steering angle

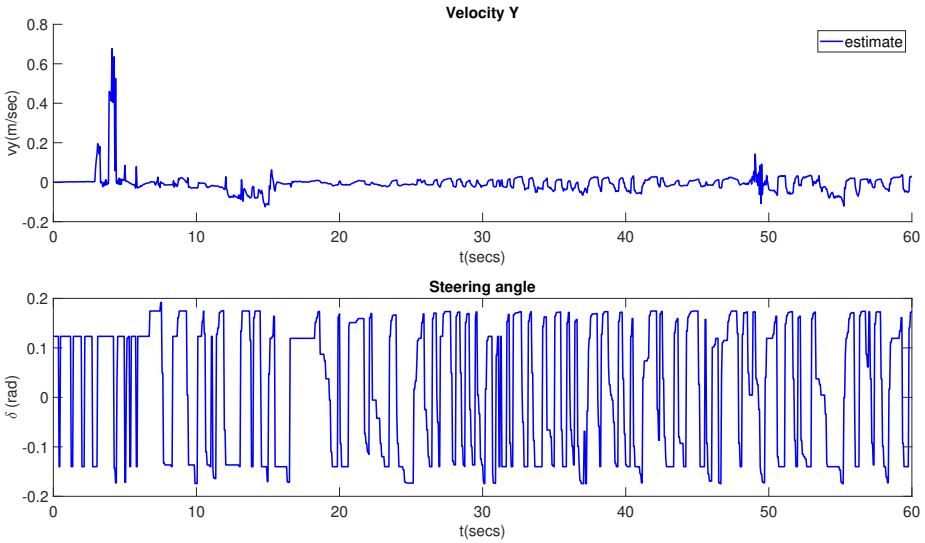

Figure 7.29: Velocity along y direction w.r.t body frame and the input steering angle

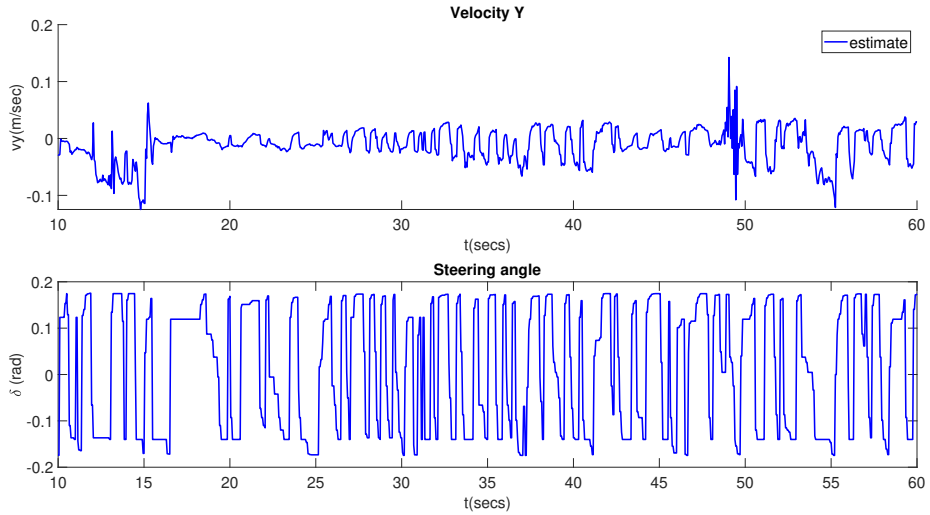

Figure 7.30: Velocity along y direction w.r.t body frame and the input steering angle

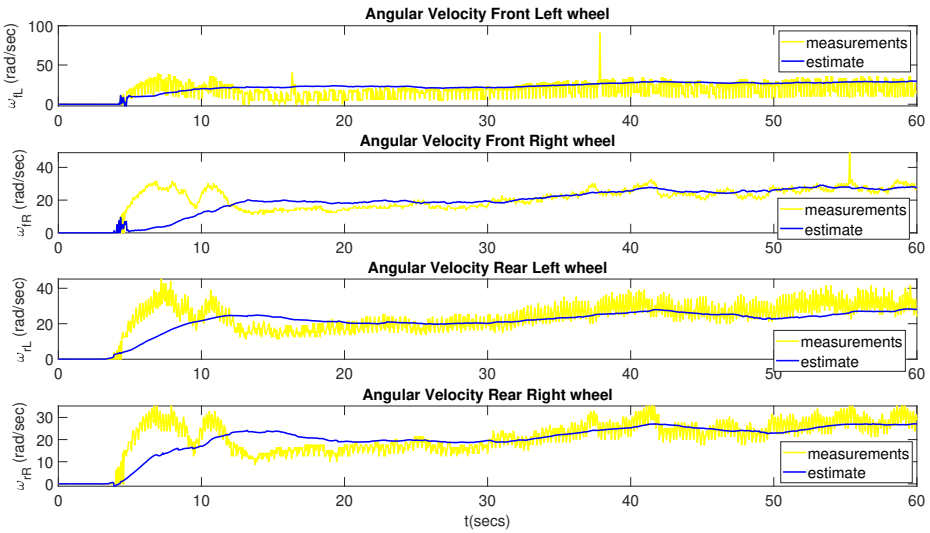

Figure 7.31: Angular velocity estimated by UKF

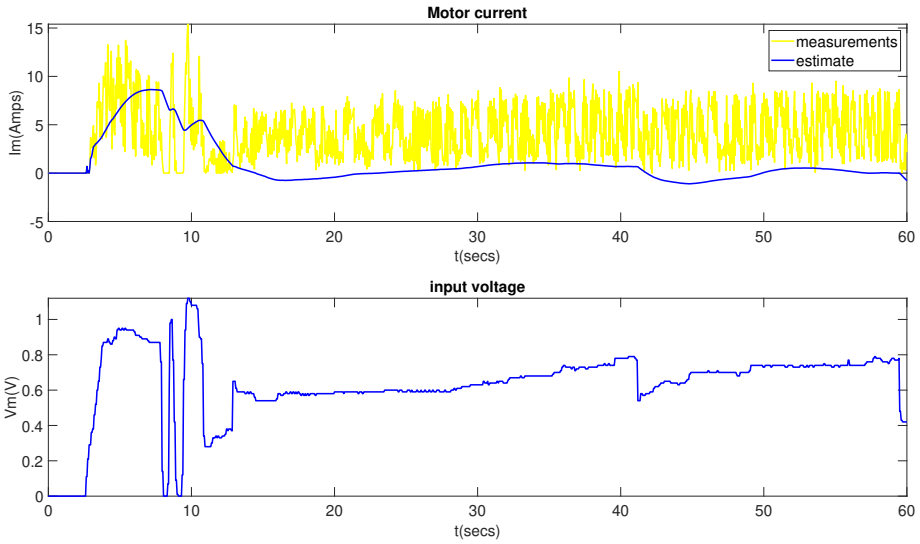

Figure 7.32: Motor current estimated by UKF

7.6.3 Motor Current

Figure(7.32), represents the current in the coils of the motor and the input voltage given to the motor. The yellow plot labelled as 'measurements', represents the measurements obtained from the current sensor on the motor driver and the blue plot labelled as 'estimate' represents the estimate from UKF. It can be observed that the measured values are fluctuating a lot whereas the estimate is smooth.

It can also be observed that the measurements are always positive but this is not true because when the vehicle is decelerating the input voltage will be less and the voltage generated by the rotation of the rotor will be large which makes the current negative and the power flows from the motor into driver. This power will be dissipated as heat in one of the components. This can be seen in estimate plot.

Some more sanity checks would be that when the vehicle starts from rest there will large current flowing into the motor for generating the required torque and later it reduces while it is in uniform motion. If the input voltage is increased to the motor, then the speed of motor increases by generating torque for this the current flow into the motor increases. The estimate satisfies these conditions.

7.6.4 Normal Loads

Figure(7.33), represents the estimate of normal load on each wheel of the vehicle.

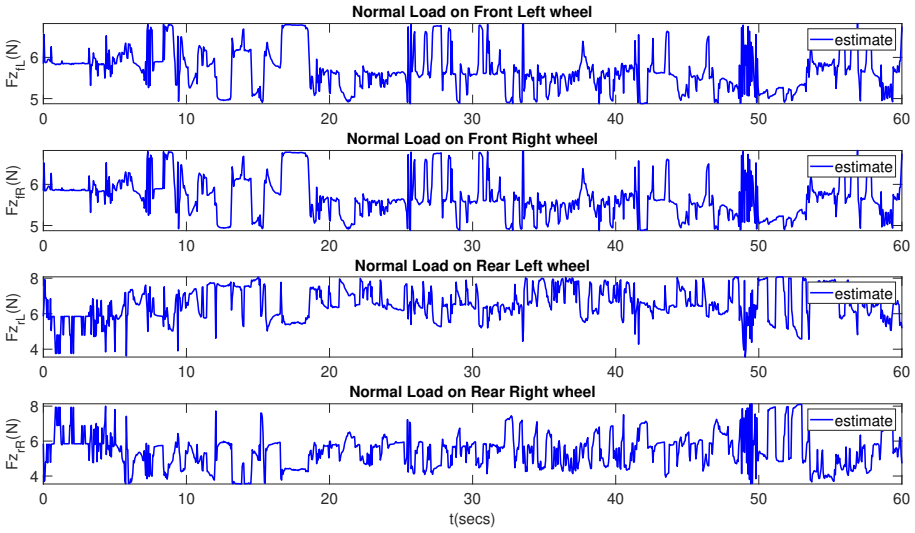

Figure 7.33: Normal Loads on the wheels estimated by UKF

7.6.5 Position(sx,sy) and Yaw(ψ)

Figure(7.34), represents the position the vehicle estimated in two cases. Yellow plot represents the case when a basic model i.e. by double integrating the acceleration measurements, for estimating position of the vehicle. Blue plot represents the position estimate obtained from UKF. It can be observed that the position of the vehicle in the case of using basic model, has grown exponentially over time and reached values of thousands of meters. This is because of the error in the acceleration measurements are integrated twice and added. This makes the error in the estimate grow exponentially over time. Figure(7.35), shows the estimated position and the Yaw angle from UKF. It can be seen that that position of the vehicle is contained in a region and not growing exponentially over time. This shows that the estimate is much better compared to the case with basic model. However, the estimate is not correct because it does not represent the path in which the rover was driven. Currently the estimate of velocity is differentiated to find the acceleration which is then used to update the predicted position and velocity after state prediction.This can be improved if there is a measurement of either velocity or position of the vehicle which can be directly used to correct the predictions.

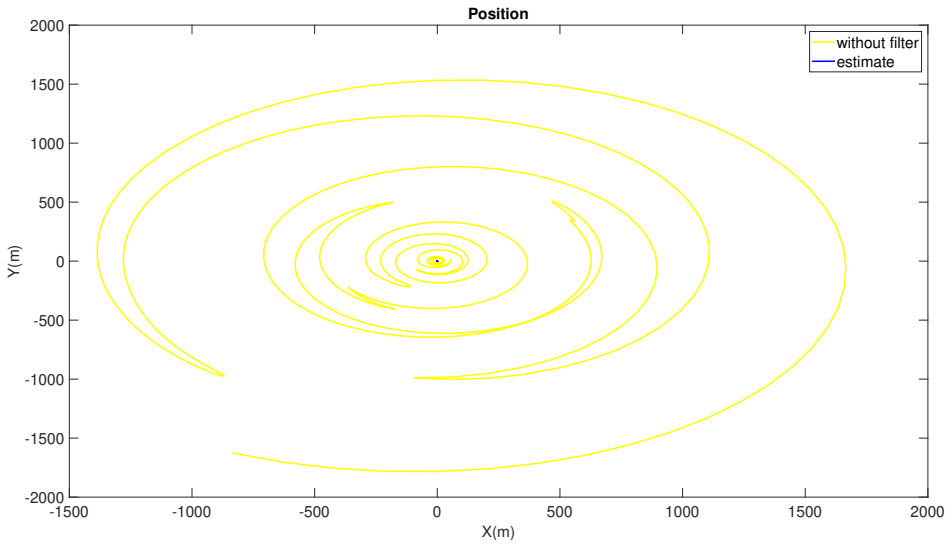

Figure 7.34: Estimated position of the vehicle by UKF

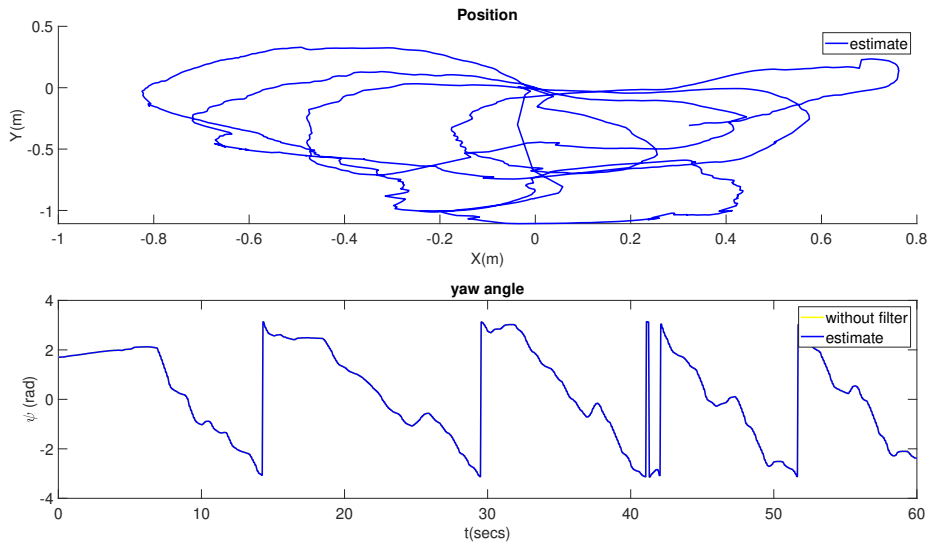

Figure 7.35: Estimated position and Yaw of the vehicle by UKF

CHAPTER 8

Conclusion and Future work

A model that represents the dynamics of a vehicle in a two dimensional space has been built. It is a index 1 semi explicit non-linear DAE system. In this model some states and parameters can not be measured. Hence, a state estimator and parameter estimator have been built. The state estimation was done using an Unscented Kalman filter and the parameter estimation was done by solving an optimization problem. For testing these methods experiments were conducted using Erle rover from which data was logged and simulations were performed. This logged data was used for estimating parameters and testing the performance of the UKF. The estimated parameters were used in UKF where the states were estimated.

The estimate of position was not accurate and it can be improved by including sensors to get the measurements of velocity. If these measurements are also included in parameter estimation then the parameter estimation can be done with greater accuracy.The model estimated can be used to check the accuracy of its predictions of the dynamics of the real system. The estimated model along with the state estimator can be used in model based controllers for predicting the states of the vehicle and finding the optimal inputs for performing specific tasks.

REFERENCES

1. **Cao, Y.**, **H. Zhang**, **W. Li**, **M. Zhou**, **Y. Zhang**, and **W. A. Chaovalitwongse** (2019). Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions. *IEEE Transactions on Evolutionary Computation*, **23**(4), 718–731.

2. **Das, M.**, **A. Dey**, **S. Sadhu**, and **T. K. Ghoshal**, Joint estimation of states and parameters of a reentry ballistic target using adaptive ukf. *In 2014 Fifth International Symposium on Electronic System Design*. 2014.

3. **Kennedy, J.** and **R. Eberhart**, Particle swarm optimization. *In Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4. 1995.

4. **Liang, J. J.**, **A. K. Qin**, **P. N. Suganthan**, and **S. Baskar** (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, **10**(3), 281–295.

5. **Limebeer, D. J. N.** and **A. V. Rao** (2015). Faster, higher, and greener: Vehicular optimal control. *IEEE Control Systems Magazine*, **35**(2), 36–56.

6. **Mandela, R. K.**, **R. Rengaswamy**, and **S. Narasimhan** (2009). Nonlinear state estimation of differential algebraic systems. *IFAC Proceedings Volumes*, **42**(11), 792 – 797. ISSN 1474-6670. URL `http://www.sciencedirect.com/science/article/pii/S1474667015303724`. 7th IFAC Symposium on Advanced Control of Chemical Processes.

7. **Singer, S.** and **J. Nelder** (2009). Nelder-Mead algorithm. *Scholarpedia*, **4**(7), 2928. Revision #91557.

8. **Wan, E. A.** and **R. Van Der Merwe**, The unscented kalman filter for nonlinear estimation. *In Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. 2000. ISSN null.

9. **You, C.** and **P. Tsiotras**, Vehicle modeling and parameter estimation using adaptive limited memory joint-state ukf. *In 2017 American Control Conference (ACC)*. 2017.