# Lab 2 Report
# Lateral Dynamics – State estimation

SD2231– Applied vehicle dynamics control

May 4, 2020

Anirvan Dutta
Akash Singh

| Postal address | Visiting address | Telephone | Internet |
|---|---|---|---|
| Royal Institute of Technology | Teknikringen 8 | +46 8 790 6000 | www.ave.kth.se |
| KTH Vehicle Dynamics | Stockholm | **Telefax** | |
| SE-100 44 Stockholm | | +46 8 790 9304 | |
| Sweden | | | |

# Contents

# 1 Task 1 Washout filtering approach of side - slip estimation

## 1.1 Task 1.a

The body side-slip is defined as:

$$\beta = \arctan \frac{v_y}{v_x} \tag{1}$$

Three estimators have been designed in code, whose architecture are illustrated one-by-one.

#### 1.1.1 Model-based Estimator

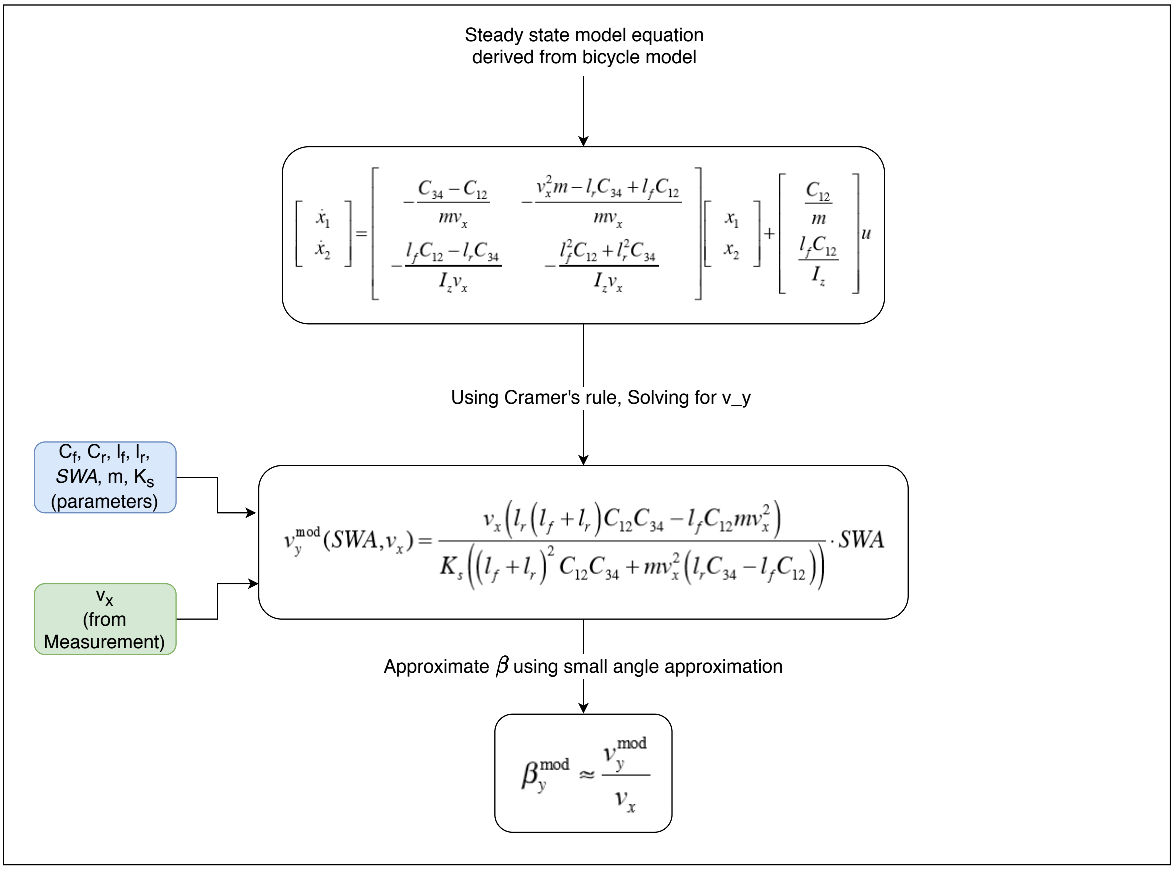



Figure 1: Architecture of Model-based Estimator

### 1.1.2 Integration-based Estimator

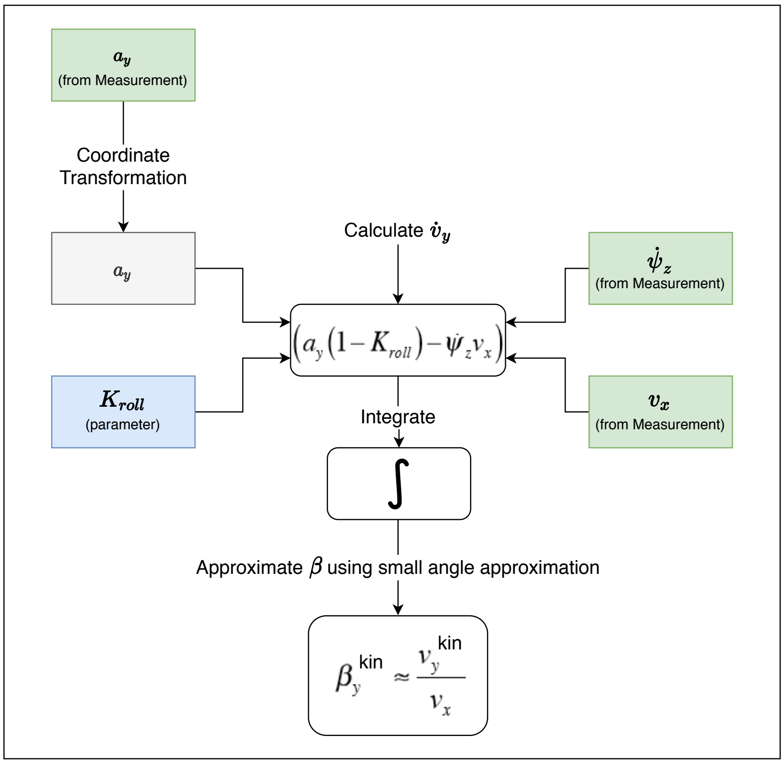



Figure 2: Architecture of Integration-based Estimator

### 1.1.3 Wash-Out Filter

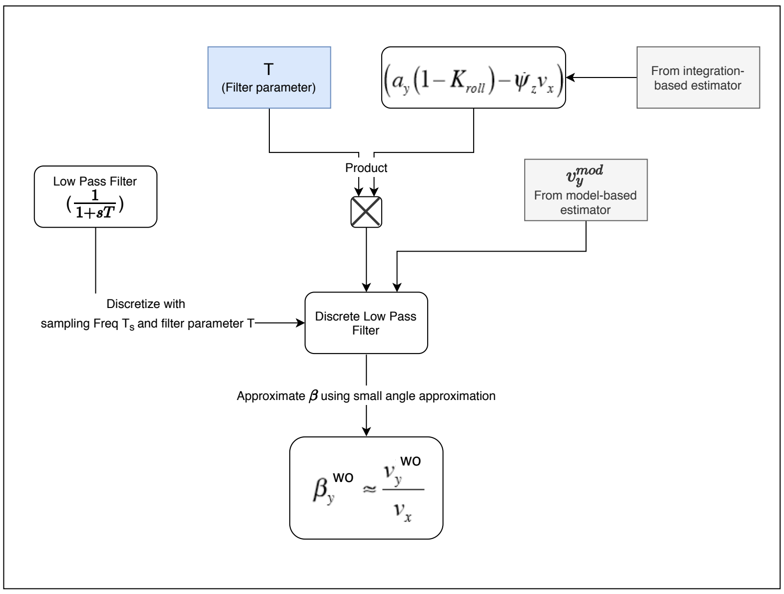



Figure 3: Architecture of Washout Filter-based Estimator

## 1.2 Task 1.b

Among the vehicle parameters, the tyre parameters $C_f$ and $C_r$ have been tuned to improve the results of the model-based estimator.

We compared the results of the measured lateral velocity, `vy_VBOX`, with the lateral velocity, $v_y^{mod}$ calculated using the steady-state model. Various articles on the internet gave a starting idea for the supposed value of the lateral stiffness of front and rear axles. After iterative testing, the chosen the values that optimized the result for all the 4 maneuvers were -

Final Values was chosen to be
$C_f = 160000$ & $C_r = 225000$.

The result is evident from the comparison of $v_y$ values calculated from model before and after tuning the vehicle parameters.

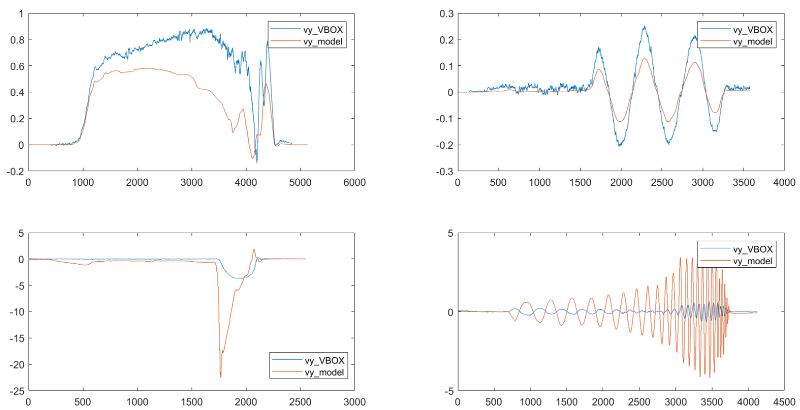



Figure 4: Calculated $v_y^{mod}$ vs measured $v_y$ before tuning

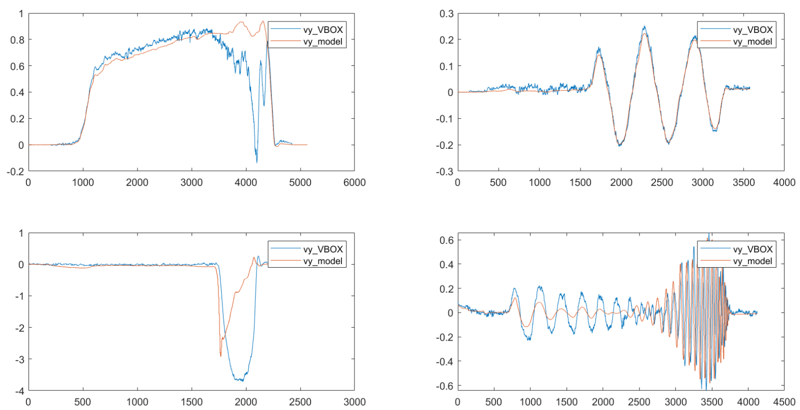



Figure 5: Calculated $v_y^{mod}$ vs measured $v_y$ after tuning

## 1.3 Task 1.c

Wash-out filter is an attempt at taking best out of both of the estimators. The model-based estimator work better during steady state, while the integration-based estimator work better(supposedly) during transient conditions.

Since integration-based estimator suffered from drift and had significant mean square error, we reduce the T parameter, to give more weightage to model-based estimator. Final value of T was settled at 0.05 which in turn followed mostly the model-based values and ignored the integration based values of beta. The MSE was the basis of the iterative tuning of the parameter and a lower value of T was avoided to completely ignore the integration based estimation.

The comparison of before and after tuning of the wash-out filter is presented below.

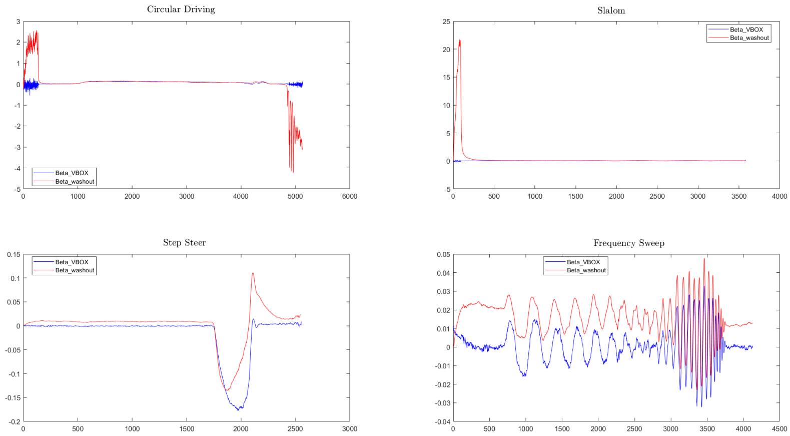



Figure 6: Calculated $\beta$ vs measured $\beta$ for T=1

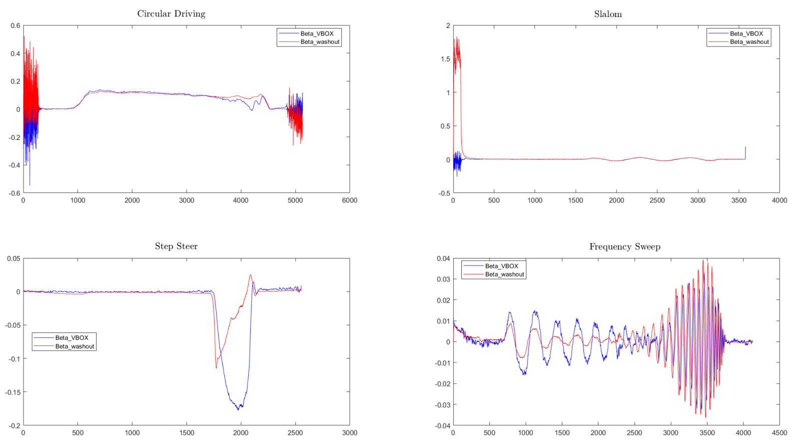



Figure 7: Calculated $\beta$ vs measured $\beta$ for T=0.05

## 1.4 Task 1.d

Quality of estimators in terms of the Mean Squared Error(MSE) and Max Error is presented below.

Table 1: Quality of the three estimators

| Maneuver | MSE | | | Max Error | | |
|---|---|---|---|---|---|---|
| | Model-based | Integration-based | WashOut Filter | Model-based | Integration-based | WashOut Filter |
| Circular | $8.23e-4$ | $6.54e+2$ | $4.57e-3$ | $1.65e-1$ | $1.79e+2$ | $6.99e-1$ |
| Slalom | $2.9e-5$ | $1.21e+1$ | $9.34e-2$ | $4.52e-2$ | $3.25e1$ | 2.38 |
| Step Steer | $1.57e-3$ | $1.046e-1$ | $1.56e-3$ | $1.49e-1$ | 1.69 | $1.49e-1$ |
| Frequency Sweep | $6.3e-5$ | $1.14e-1$ | $9.42e-5$ | $3.06e-2$ | $5.6e-1$ | $4.06e-2$ |

We have also presented the $\beta$ estimation for all the filters on all manoeuvres as in Figure 8.

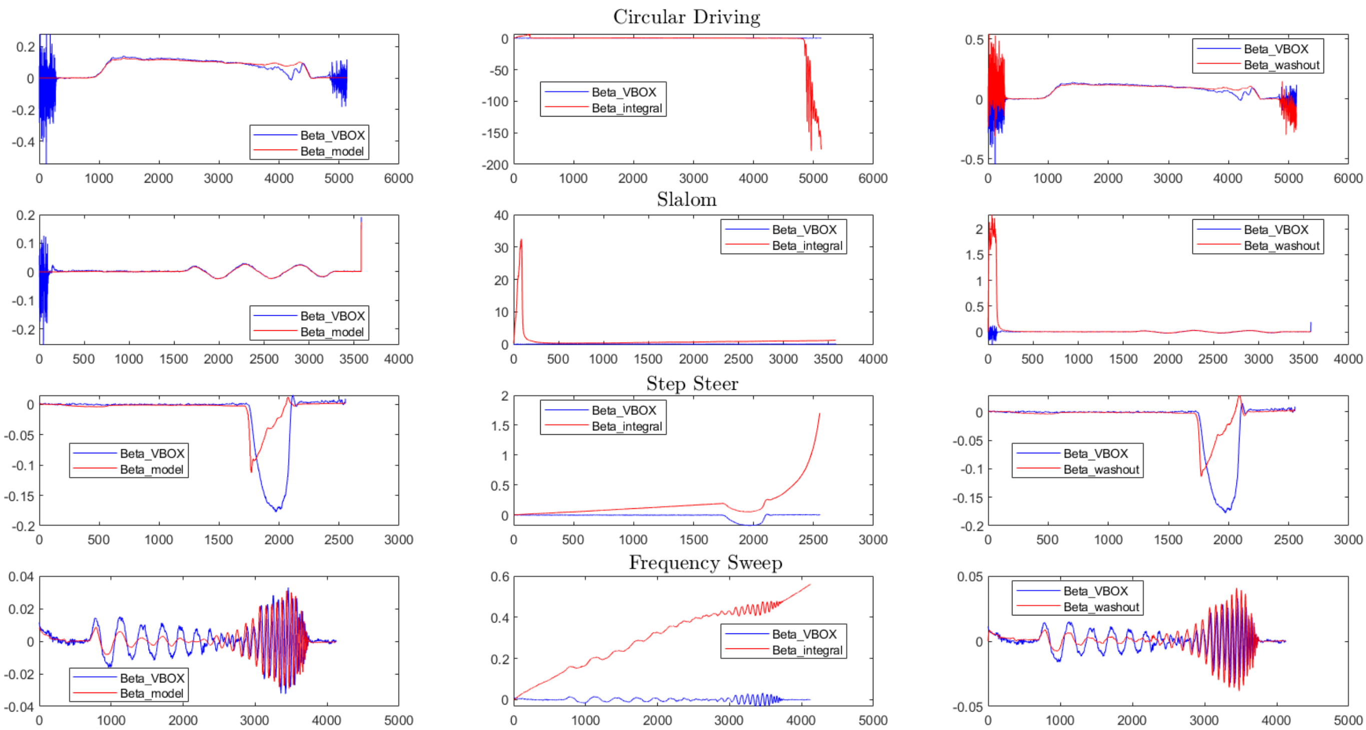



Figure 8: Estimated $\beta$ vs Measured $\beta$ on all manoeuvres with T=0.05

## 1.5 Task 1.e

Out of the three estimators only the integration based estimator has the tendency to drift, because along with the actual quantity it also integrates the noise in the reading.

The model based estimator works fine most of the times, especially when vehicle is in a steady state or when the lateral velocity is approx æ10km/h and the change is not too quick. Whereas the integral estimator is able to track those transient behaviours, but without handling the trend in its calculation, it is not of much use. For eg, the steep steer scenario where the model-based estimator performed the worst, the other estimator was able to match the transient nature of the side slip, though it suffers from drift.

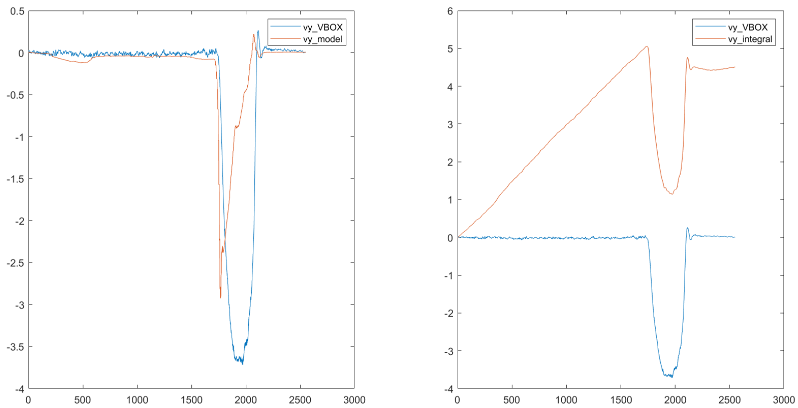



Figure 9: Comparison of two estimators for step steer manuever

Additionally, as the filter co-efficient $T$ of the wash-out filter was kept low, it did not suffer from drift, however on increasing $T$, even washout filter showed signs of drift, as shown in Figure.6.

## 1.6 Extra Task 1.g

In order to imrove the wash-out filter, we focused on the integral estimator since it is the one most susceptible to errors.
We took the following measures:

- It was observed that the lateral acceleration $a_y$ measurement coming from VBOX gave low values even when the lateral velocity $v_y$ was almost zero. This was corrected by finding the mean of the $a_y$ when the car did not have significant lateral velocity ($v_y < \varepsilon$), where $\varepsilon$ is the minimum floating-point value in `MATLAB`.

- Introduced a low pass filter on the lateral acceleration measurements before the integral, in order to filter out the high frequency noise.

- Introduced a high pass filter on the integrated quantity in order to remove the drift which usually is sustained by very low frequencies.

Further, in wash-out filter the low pass filter and high pass filter were separated in order to employ different filter co-efficient utilising the following equation -

$$v_y^{wo} = \frac{1}{1+sT_{lo}} v_y^{mod} + \frac{sT_{hi}}{1+sT_{hi}} v_y^{kin} \tag{2}$$

After careful iterative tuning, $T_{lo}$ and $T_{hi}$ was selected as 1, 0.35 respectively. In addition to this, estimated $\beta$ was zeroed out when the there was minimal lateral acceleration i.e $a_y^{COG}$ was less than 0.1 to remove the erroneous start and end values.The formulated code for wash-out filter is represented in the Figure,10

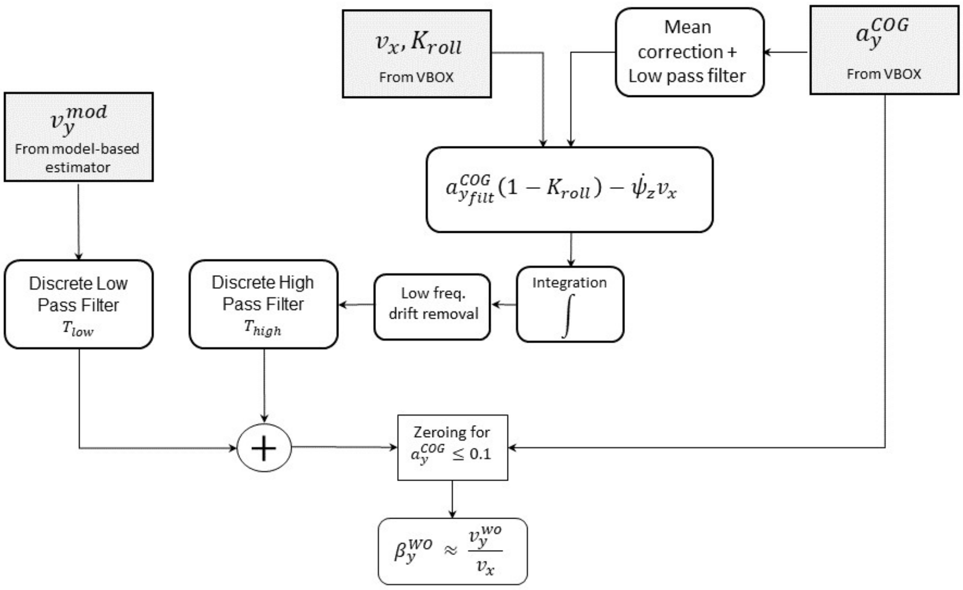



Figure 10: Architecture of advanced Washout Filter-based Estimator

The implementation drastically removed the drift from the integration based estimator as shown in Figure.11 and 12.

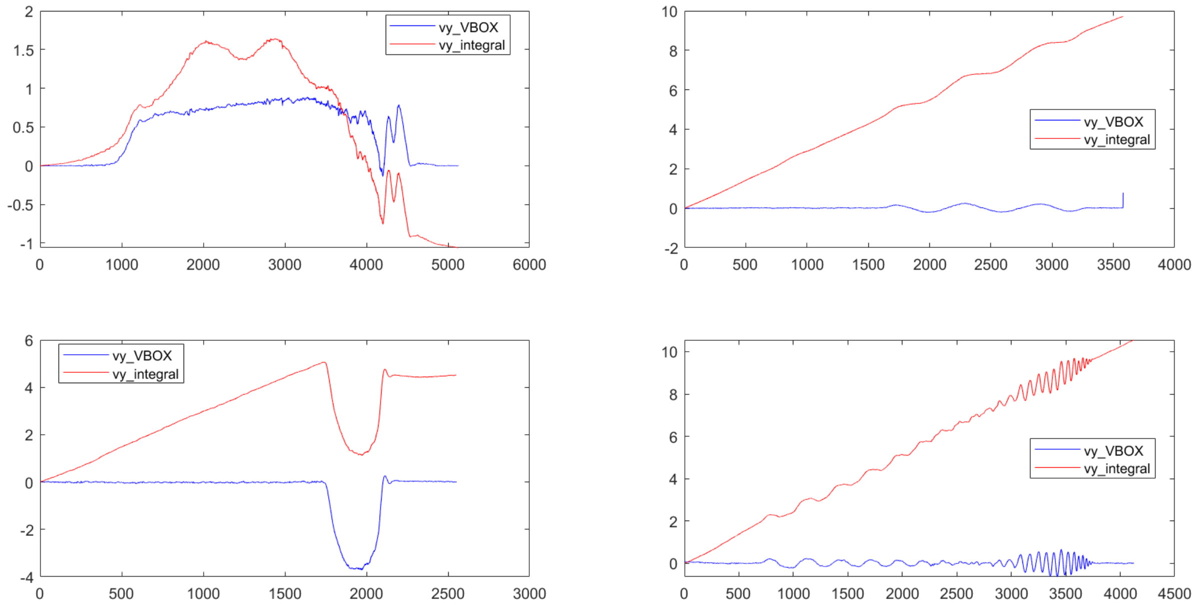



Figure 11: Integration-based estimator before applying filters

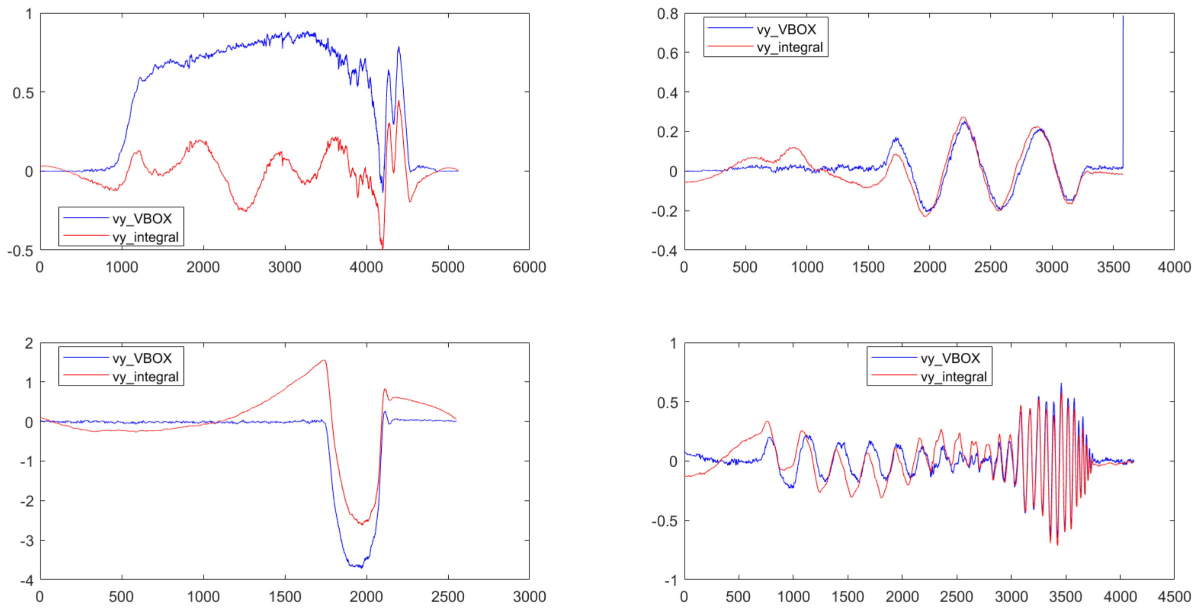



Figure 12: Effect on the Integration-based estimator after applying filters

As a result the error in estimation of most of the scenarios reduced, except for in the case of circular steer as can be seen in figure 11 & 12.

Table 2: Quality of the three estimators after further tuning

| Maneuver | MSE Model-based | MSE Integration-based | MSE WashOut Filter | Max Error Model-based | Max Error Integration-based | Max Error WashOut Filter |
|---|---|---|---|---|---|---|
| Circular | $8.23e-4$ | $7.34e-3$ | $9.61e-4$ | $1.65e-1$ | $1.65e-1$ | $1.65e-1$ |
| Slalom | $2.9e-5$ | $4.17e-4$ | $4.26e-4$ | $4.52e-2$ | $9.04e-1$ | $9.04e-1$ |
| Step Steer | $1.57e-3$ | $7.14e-4$ | $1.1e-3$ | $1.49e-1$ | $6.57e-2$ | $1.31e-1$ |
| Frequency Sweep | $6.3e-5$ | $1.01e-4$ | $7.66e-5$ | $3.06e-2$ | $3.51e-2$ | $3.34e-2$ |

The effect of improvement in integration based estimator is evident in washout-filter. The table 2 shows a marked improvement in the max error and mse statistics over the results in table 1.
We have also presented the improved $\beta$ estimation for all the filters on all manoeuvres as in Figure 13.

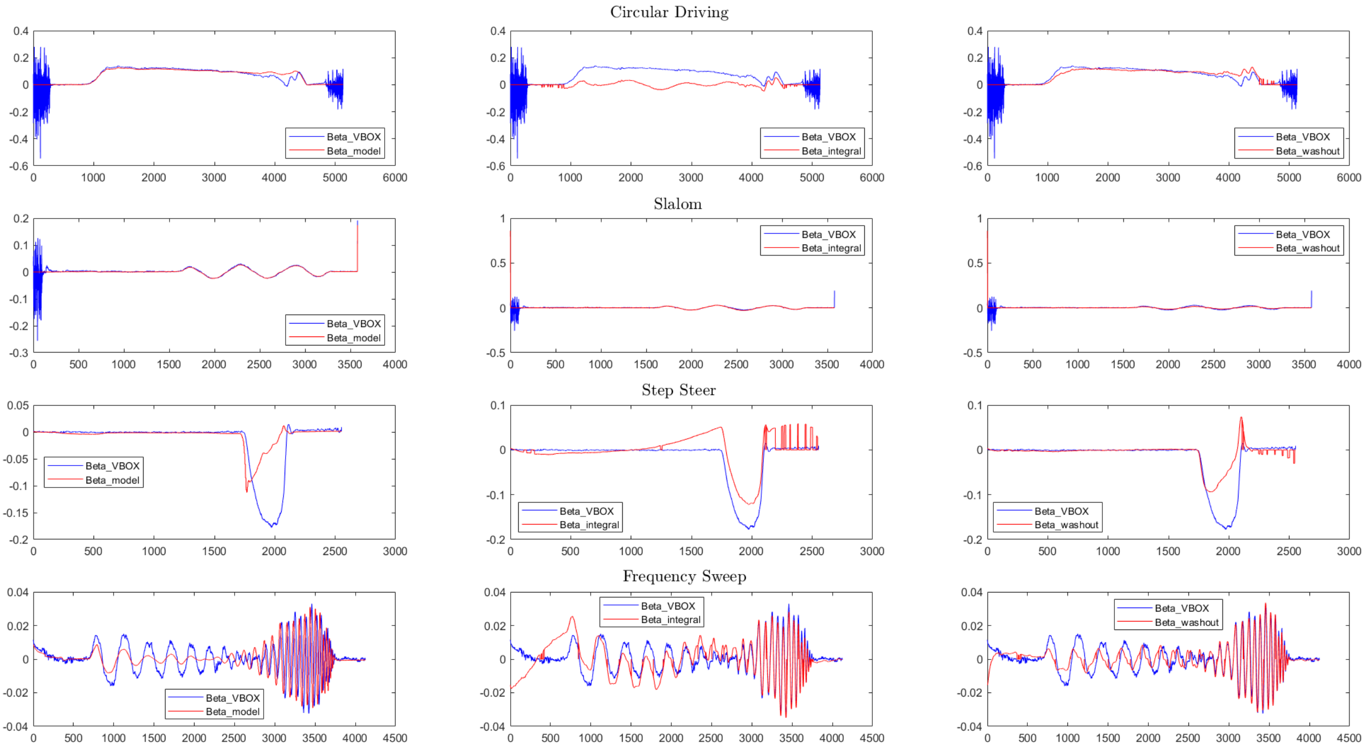



Figure 13: Estimated $\beta$ vs Measured $\beta$ on all manoeuvres

It is evident from the above figure, that wash-out filter is indeed taking the best out of both of the estimators- following the model during steady state and following integration during transient motion.

# 2 Task 2 Unscented Kalman Filter estimation

## 2.1 Task 2.a

In order to implement the Unscented Kalman Filter (UKF), state update and measurement prediction equations were implemented. Based on the instruction, the state variables were selected as $v_x, v_y$ and $\dot{\psi}_z$. Following equations were used to formulated the state update Eq.3

$$\alpha_{12} = arctan(\frac{v_y + \dot{\psi}_z l_f}{v_x}) - \delta \tag{3a}$$

$$\alpha_{34} = arctan(\frac{v_y - \dot{\psi}_z l_r}{v_x}) \tag{3b}$$

$$F_{12} = -C_f \alpha_{12} \tag{3c}$$

$$F_{34} = -C_r \alpha_{34} \tag{3d}$$

$$f(\dot{v}_x) = \dot{\psi}_z v_y - \frac{F_{12} sin(\delta)}{m} \tag{3e}$$

$$f(\dot{v}_y) = -\dot{\psi}_z v_x + \frac{F_{34} + F_{12} cos(\delta)}{m} \tag{3f}$$

$$f(\ddot{\psi}_z) = \frac{l_f F_{12} cos(\delta) - l_r F_{34}}{I_z} \tag{3g}$$

*Runga Kutta* was used in order to perform numerical integration to generate the states at $t+1$ given the states at time $t$. Similarly, $v_x, a_y, \dot{\psi}_z$ was used as measurement prediction variables. The measurement equations are used to generate measurements from the predicted state at a given time $t$ and used in the Kalman update step with the actual measurement from the VBOX. As $a_y$ was not part of the state, following relation was used

$$a_y = \frac{F_{34} + F_{12} cos(\delta)}{m} \tag{4a}$$

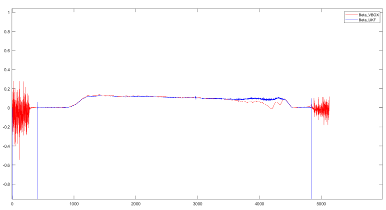
where, $F_{34}$ and $F_{12}$ was computed using Eq.3(a-d) given the states at time $t$. After formulating the state and measurement equations, `ukfpredict1` function was used to perform the UKF predict step and `ukfupdate1` function was used to perform the UKF update/correction in iterative manner.

## 2.2 Task 2.b

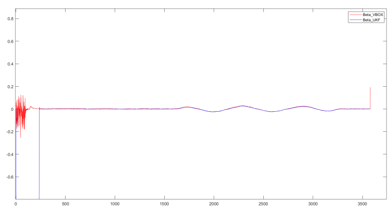
To obtain the UKF filter results, same tuned values of $C_f = 160000$ and $C_r = 225000$ was used. The process noise $Q = 0.1$ was kept for all the states and the measurement noise $R = 0.01$ was kept for all the measurement dimension. The measurement data $a_y$ coming from VBOX was corrected similar to previous implementation to $a_y^{COG}$. The results obtained by UKF is presented in Table.3 and the estimated side slip $\beta$ for all the four manoeuvres is presented in Fig.14.

Table 3: Quality of untuned UKF

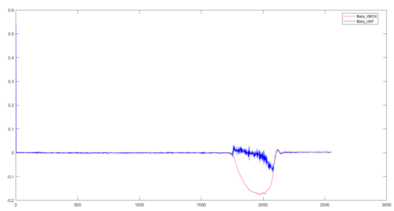
| Maneuver | MSE | Max Error |
|---|---|---|
| Circular | $3.022e-1$ | 1.567 |
| Slalom | $1.403e-1$ | 1.554 |
| Step Steer | $2.245e-3$ | $5.39e-1$ |
| Frequency Sweep | $1.585e-3$ | 1.541 |



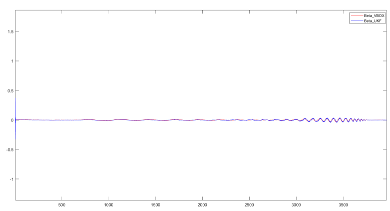
(a) Circular driving to the left



(b) Slalom



(c) Step steer to the left



(d) Frequency sweep

Figure 14: Estimated $\beta$ using UKF filter for different maneuvers vs measured $\beta$

***Comparision***

It is clear from mean square error results and the plots that without tuned UKF under-performs compared to model based and wash-out filter. This is primarily due to extremely high and incorrect estimation during start and end of each manoeuvres. To further analyse the reason, the variance evolution was generated as shown in figure.15, which signifies the confidence level of the $\beta$ estimation over time. We can see that during the initial portion of the trajectory, the variance is getting reduced for $v_x$ and increasing for $v_y$ and $\dot{\psi}_z$ which might occur due to noisy measurements coming from VBOX and the predicted measurement is not able to match it. Another prominent reason could be due to the simplified model utilised in state estimation.

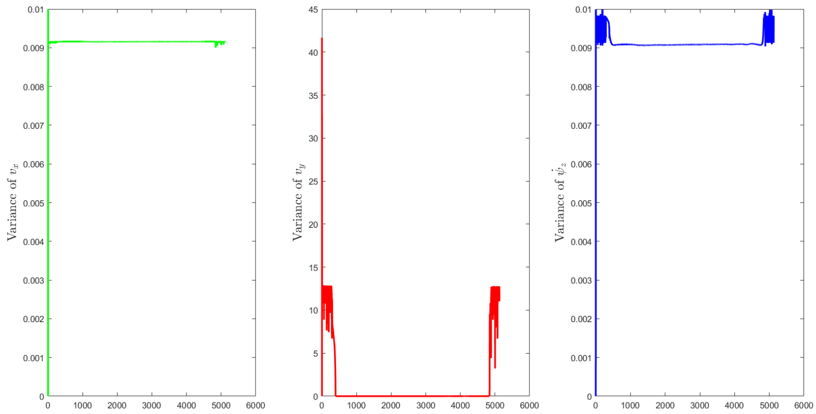



Figure 15: Variance of the state estimation for Circular driving to the left

The variance plots of other manoeuvres were omitted to keep the length of the report limited, but similar results followed as well.

## 2.3 Extra Task 2.c

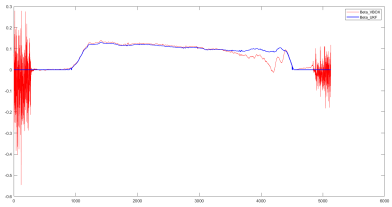
This insight of the variance evolution was later used in tuning the process $Q$ and measurement noise $R$ parameter and better results were obtained. Additionally, it was observed that the ratio of $\frac{Q}{R}$ played a role in improving the $\beta$ estimation result. Increasing the ratio, resulted in relying more on measurement of the VBOX while decreasing it, meant relying on the state estimation from the model. After careful tuning, following $Q$ and $R$ values were selected. $Q = [0.5, 0, 0; 0, 0.5, 0; 0, 0, 0.25]$, $R = [0.01, 0, 0; 0, 0.05, 0; 0, 0, 0.01]$.

Additionally, initial co-variance matrix was selected such that the initial VBOX measurement is close to the initial state estimate - $P_0 = [1e-1, 0, 0; 0, 1e-1, 0; 0, 0, 1e-2])$. It was observed that the $alpha$, $beta$ and $kappa$ values used to extract the sigma points did not effect the performance of the $\beta$ estimation too much. However $alpha$ and $beta$ was varied in a grid search manner and finally was selected as 0.5 and 0.1 based on the mean square error and max error performance of estimated and measured $\beta$.

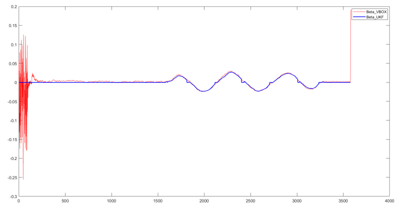
Further, to improve the performance of the UKF filter, $a_y^{COG}$ measurement data was mean corrected and lowpass filtered similar to the Task 1.g. In addition to this $\beta$ was zeroed out when the there was minimal lateral acceleration i.e $a_y^{COG}$ was less than 0.1 to remove the erroneous start and end values. All these improved the UKF performance considerably and resulted in Table.4. and Fig. 16.

Table 4: Quality of tuned UKF

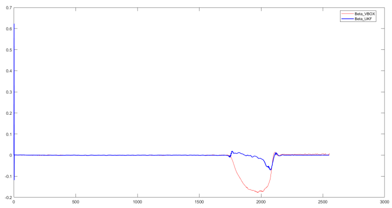
| Maneuver | MSE | Max Error |
|---|---|---|
| Circular | $8.806e-4$ | $1.649e-1$ |
| Slalom | $3.158e-5$ | $4.533e-2$ |
| Step Steer | $2.270e-3$ | $6.21e-1$ |
| Frequency Sweep | $9.338e-5$ | $1.347e-1$ |



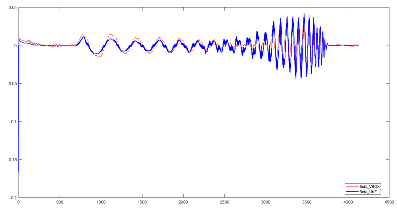
(a) Circular driving to the left



(b) Slalom



(c) Step steer to the left



(d) Frequency sweep

Figure 16: Estimated $\beta$ using UKF filter for different maneuvers vs measured $\beta$

Except the *step steer* manoeuvre, the tuned UKF outperforms the basic implementation to great extent and outperforms wash-out filter in *circular driving* and *slalom* manoeuvres.

## 2.4 Extra Task 2.e

As instructed, the UKF code was carefully analysed and all the codes are extracted and implemented everything in the `UKF_start.m`. The `vehicle_state_eq.m` and `vehicle_measure_eq.m` was kept the same as it was utilised as function evalution in the main loop. The correctness of implementation was verified by matching the mean square error of $\beta$ to that of Task 2.c.

```
%-----------------------------------
% FILTERING LOOP FOR UKF
%-----------------------------------
disp(' ');
disp('Filtering the signal with UKF...');

alpha = 0.5;
beta = 0.1;
kappa = 0;
```

```
M = x_0;
P(:,:,1) = P_0;
 Beta_ukf = zeros(size(SteerAngle));

%% Generate sigma weights
n = size(x_0,1);
lambda = alpha^2 * (n + kappa) - n;

WM = zeros(2*n+1,1);
WC = zeros(2*n+1,1);
 for j=1:2*n+1
     if j==1
         wm = lambda / (n + lambda);
         wc = lambda / (n + lambda) + (1 - alpha^2 +
             beta);
     else
         wm = 1 / (2 * (n + lambda));
         wc = wm;
     end
     WM(j) = wm;
     WC(j) = wc;
end

c = n + lambda;

totalTime = length(Time);

for iter = 2:totalTime-1
    predictParam.input = SteerAngle(iter);
    mu_bar = M(:, iter-1);
    P_bar = P(:,:,iter-1);
    %% Predict Step
    % Compute sigma points on the initial state
        distribution mu_bar
    A = schol(P_bar);
    X_bar = [zeros(size(mu_bar)) A -A];
    X_bar = sqrt(c)*X_bar + repmat(mu_bar,1,size(X_bar
        ,2));


    % Propagate the sigma points through the state
        estimation function
    X_hat = [];
    for i=1:size(X_bar,2)
      X_hat = [X_hat feval(state_func_UKF,X_bar(:,i),
          predictParam)];
    end

```

```
    % Compute state distribution mu_hat from the
        propagated sigma points
    mu_hat = zeros(size(X_hat,1),1);
    P_hat  = zeros(size(X_hat,1),size(X_hat,1));
    for i=1:size(X_hat,2)
      mu_hat = mu_hat + WM(i) * X_hat(:,i);
    end
    for i=1:size(X_hat,2)
      P_hat = P_hat + WC(i) * (X_hat(:,i) - mu_hat) * (
          X_hat(:,i) - mu_hat)';
    end
    % Add the process noise in the co-variance
    P_hat = P_hat + Q;

    %% Update Step

    % Compute sigma points on the updated state
        distribution mu_hat
    % (distribution changed due to the addition of
        process noise)
    A = schol(P_hat);
    XU_hat = [zeros(size(mu_hat)) A -A];
    XU_hat = sqrt(c)*XU_hat + repmat(mu_hat,1,size(
        XU_hat,2));

    % Propagate the sigma points through the
        measurement estimation function
    Y_hat = [];
    for i=1:size(XU_hat,2)
      Y_hat = [Y_hat feval(meas_func_UKF,XU_hat(:,i),
          predictParam)];
    end

    % Compute measurement distribution z_hat from the
        sigma points Y_hat
    z_hat = zeros(size(Y_hat,1),1);
    S_hat  = zeros(size(Y_hat,1),size(Y_hat,1));
    C_hat  = zeros(size(mu_hat,1),size(Y,1));
    for i=1:size(Y_hat,2)
      z_hat = z_hat + WM(i) * Y_hat(:,i);
    end
    for i=1:size(Y_hat,2)
      S_hat = S_hat + WC(i) * (Y_hat(:,i) - z_hat) * (
          Y_hat(:,i) - z_hat)';
      C_hat = C_hat + WC(i) * (XU_hat(1:size(mu_hat,1),
          i) - mu_hat) * (Y_hat(:,i) - z_hat)';
    end

```

```
    %Kalman update equations
    S_hat = S_hat + R;
    K = C_hat / S_hat;
    mu_hat = mu_hat + K * (Y(:,iter) - z_hat);
    P_hat = P_hat - K * S_hat * K';
    P(:,:,iter) = P_hat;
    M(:,iter) = mu_hat;

    if iter==round(totalTime/4)
        disp(' ');
        disp('1/4 of the filtering done...');
        disp(' ');
    end
    if iter==round(totalTime/2)
        disp(' ');
        disp('1/2 of the filtering done...');
        disp(' ');
    end
    if iter==round(totalTime*(3/4))
        disp(' ');
        disp('3/4 of the filtering done... Stay tuned
            for the results...');
        disp(' ');
    end
end
```

# Bibliography