Design a Tesla Model S P85 Speed Controller

Prepared by:
Nishantkumar V Patel

About the Project

- I Have done the project independetly and on my own on the platform of MATLAB & Simulink.
- I have design the PID which is speed controller in this particular case to check and examine the different velocity outputs by doing the different gain values of PID tuning.
- First of all, I have done some dynamics and forces calculations which are very necessary to take into account wile making the Simulink model.
- I have made the equation for the brushed DC motor which is I used to put in this Tesla car hypothetically. (although 3-phase 4-pole AC induction motor is actually used in this car) to make the model little bit of simple to understand.
- I did find the general characteristics of Tesla Model S car through the internet.
- I assumed that the car will be moving at constant speed.I am not going to accelerate dor decelerate the car. Therefore; Inertia does not matter when the car is in constant speed.
- I will simulate the model with one person (who driving the car) of having 72 kg body weight.
- As we predefine the input as we want or reference input that is speed in kmph and then simulate the whole model to get the output of the model which will be vehicle speed in kmph.

General Characteristics of Car

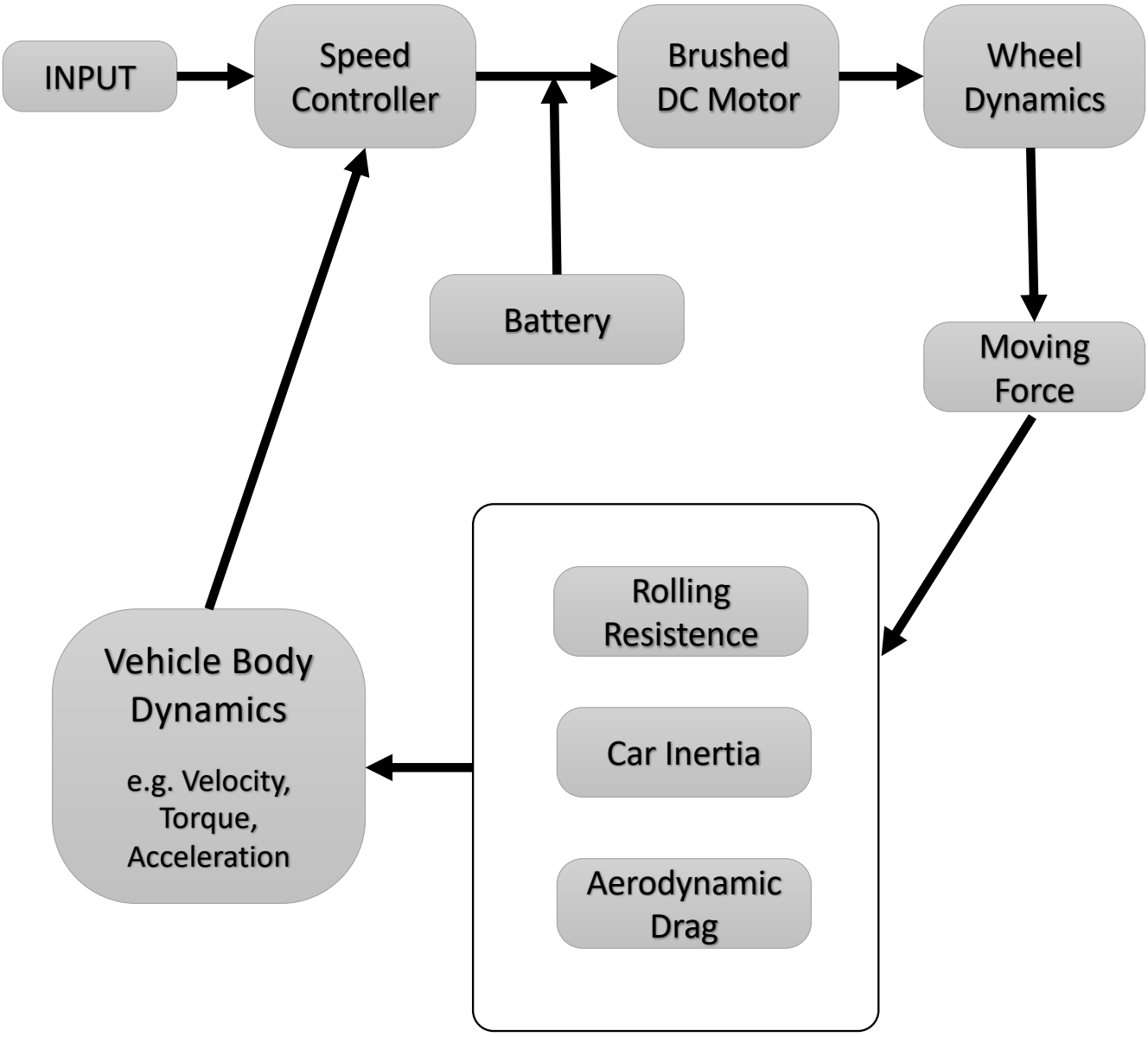
- P85 Battery is used.
- 16 Modules and 6 groups in each module.
- 85 KWh Battery Capacity.
- 3.6 V Nominal Cell Voltage
- So Nominal Battery Voltage would be,

  $\therefore$ 3.6*6*16= 346 V
- 9.73 Gear Ratio
- 2108kg+72 kg= 2180 kg Total Car Weight
- 0.24 m Wheel Radius of car (including alloy and tyre width)
- 2.3 m^2 Frontal Area of car

- 0.02 Coefficient of Friction (on asphalt/ dry road)
- 0.24 Drag Coefficient
- 1.225 kg/m^3 Air Density

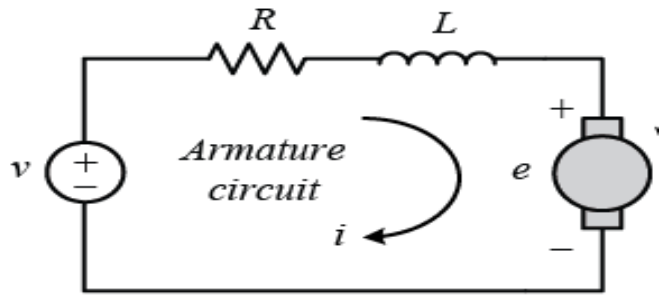
- Motor specifications:
- Resistence(R)= $5.3*10^{-3}$ Ohms
- Inductance(L)= $493*10^{-9}$ Henry
- Emf Constant(K_E)= 0.12 Vs/ rad
- Torque Constant(Kt)= 0.25 Nm/Amp
- 310 KW Motor Power
- 600 Nm Motor Torque

Car Diagram for making Simulink Model



Mathematical Calculations

A.) DC Motor Calculation:



- As you can see this is the simple diagram for the DC motor.
- v is represents the applied voltage, e is emf (electro motive force) induced inside the motor.
- Applied voltage (v) is directly proportional to rotational velocity of the motor. Whereas, applied(i) current is proportional to torque of the motor.
- Thus, ultimately by regulating or fluctuating the voltage we can change the motor speed and consecutively the wheel velocity or vehicle speed.
- According to kirchoff law applied in this circuit,

$$\therefore v = i(t)R + L[di(t)/dt] + e(t)$$

As the current is directly proportional to motor torque,

$$\therefore T = Kt*i(t)$$

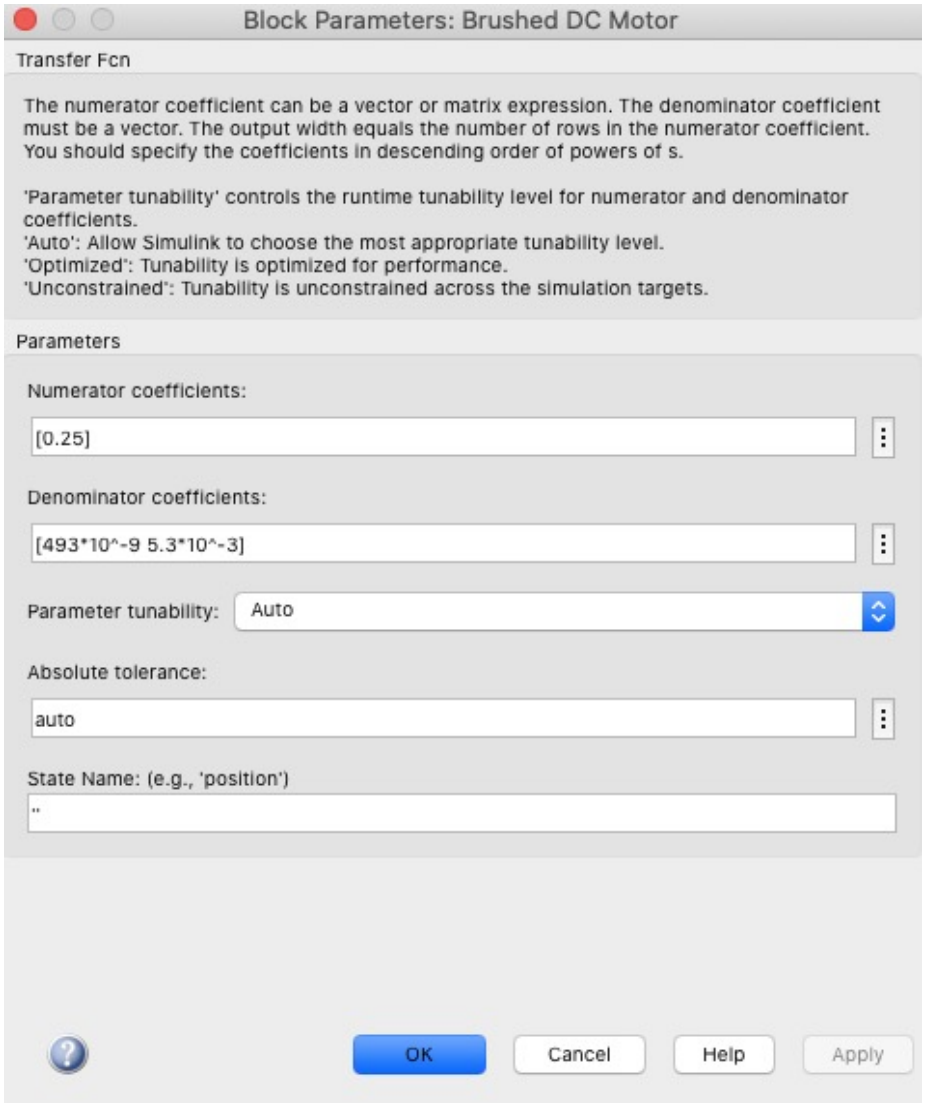
Now, converting the above equation from Time-domain to Frequency-domain or S-domain. For that purpose we need the Laplace tranform approach.

$$\therefore V(s) = R*I(s) + sL*I(s) + K_E*\omega(s)$$

$$\therefore I(s) = [V(s) - K_E*\omega(s)]/ (R + sL)$$ putting I(s) into T(s) euqation.

$$\therefore T(s) = K_t*I(s)$$

$$\therefore T(s) = K_t* [V(s) - K_E*\omega(s)]/ (R + sL)$$



The brushed dc motor is represented by Transfer function block in Simulink. The above, one can see that I put the values of Inductance, Resistence and Torque Constant (Kt) values in the frequency domain.

In Simulink;
Numerator= [S^0] with coefficient [0.25]
Denominator= [S^1 S^0] with coefficients [$493*10^{-9}$ $5.3*10^{-3}$]

means;
= $(0.25)/ (s\ 493*10^{-9} + 5.3*10^{-3})$ In equation.

B.) Tractive Force/ Wheel Dynamics:

F_T= Tractive/ Traction Force (N)

T= Motor Torque

R_w= Wheel Radius = 0.24 m

G_r= Gear Ratio= 9.73

$$\therefore F_T = (T/ R_w)* G_r$$

$$\therefore F_T = (40.54*T)$$

$$\therefore F_T = 0.6*(40.54*T) \quad N$$

- The Transimssion/ Mechanical efficiecny I assumed is 60 %. Because there are heat, frictional, transmission losses as well as slipping need to consider for making the model more realistic.

C.) Aerodynamic Drag:

D= Aerodynamic Drag force (N)

A= Frontal Area= 2.3 m^2

C_d= Drag coefficient= 0.24

ρ= Air Density= 1.225 kg/ m^3

$$\therefore F_A = 0.5*\rho*A*V^2* C_d$$

$$\therefore F_A = (0.3381\ V^2) \quad N$$

The aerodyanmic drag is also need to take into account as it becomes significant when the car is at moving state.

D.) Rolling Resistence:

Fr= Rolling Resistence force (N)

m_t= Total Car Weight including person= 2180 kg

G= Gravitational Constant= 9.81 m/s^2

Cr= Coefficient of rolling resistence= 0.02 (for asphalt/ dry road)

:. Fr= Cr*mt*g

:. Fr= (4237.7) N

Total Forces acting on vehicle $(\Sigma F)= F_T + F_A + Fr$

According to Newton's Second Law of Motion,

:. $\Sigma F= m*a$

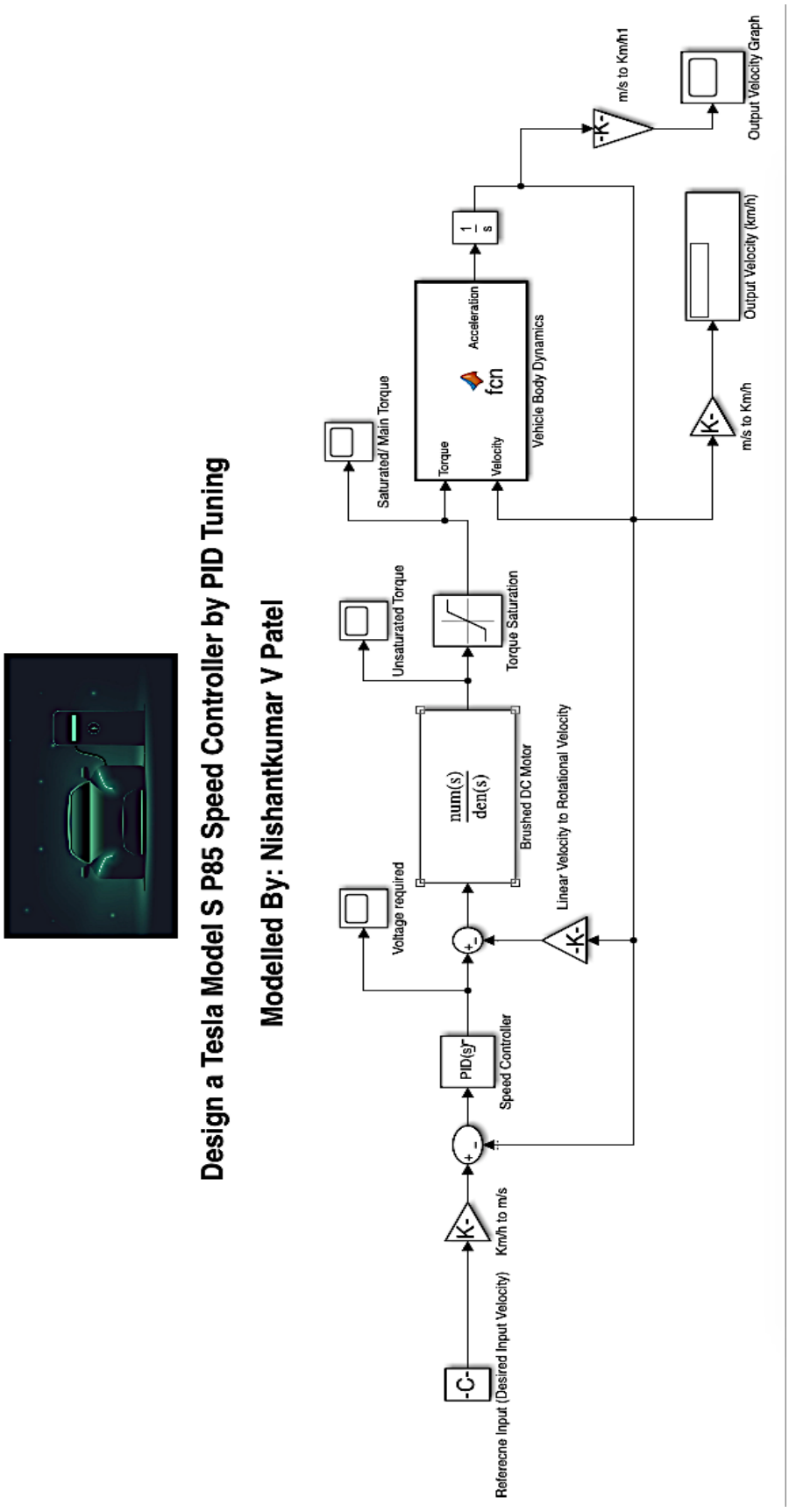
:. $[(0.7*40.54*T) – (427.7) – (0.3381V^2)]= m_t*(dV/dt$

:. $(dV/dt)= [(0.6*40.54*T) – (427.7) – (0.3381*V^2)]/ 2180$

X

- This equation- X is in Time-domian.

Setting Up the Closed Loop Simulink Model



NV_PATEL_Tesla_Model_S_Speed_Controller

SIMULATION DEBUG MODELING FORMAT APPS

New Open Save Print | Library Browser | Log Signals Add Viewer Signal Table | Stop Time 100 Normal Fast Restart | Step Back Run Step Forward Stop | Data Inspector Logic Analyzer Bird's-Eye Scope Simulation Manager

FILE LIBRARY PREPARE SIMULATE REVIEW RESULTS

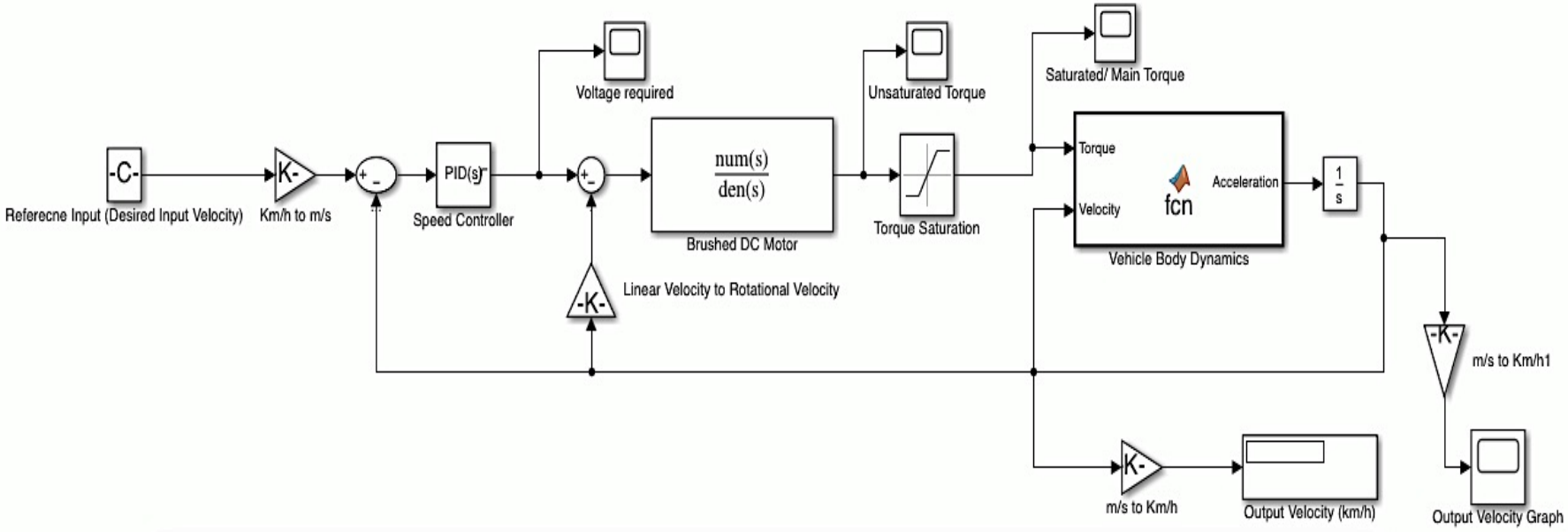
Model Browser

NV_PATEL_Tesla_Model_S_Speed_Controller

NV_PATEL_Tesla_Model_S_Speed_Controller

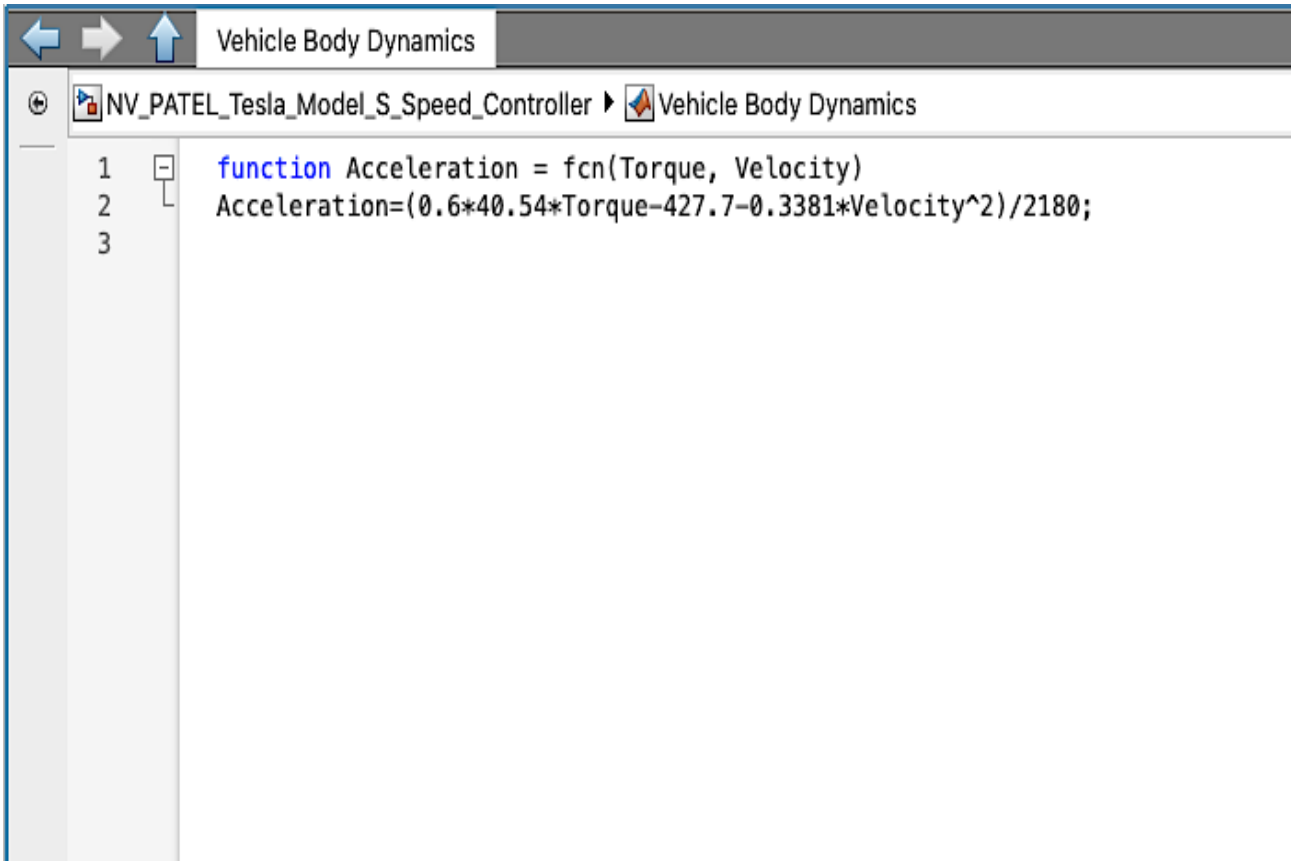
Design a Tesla Model S P85 Speed Controller by PID Tuning

Modelled By: Nishantkumar V Patel



Property Inspector

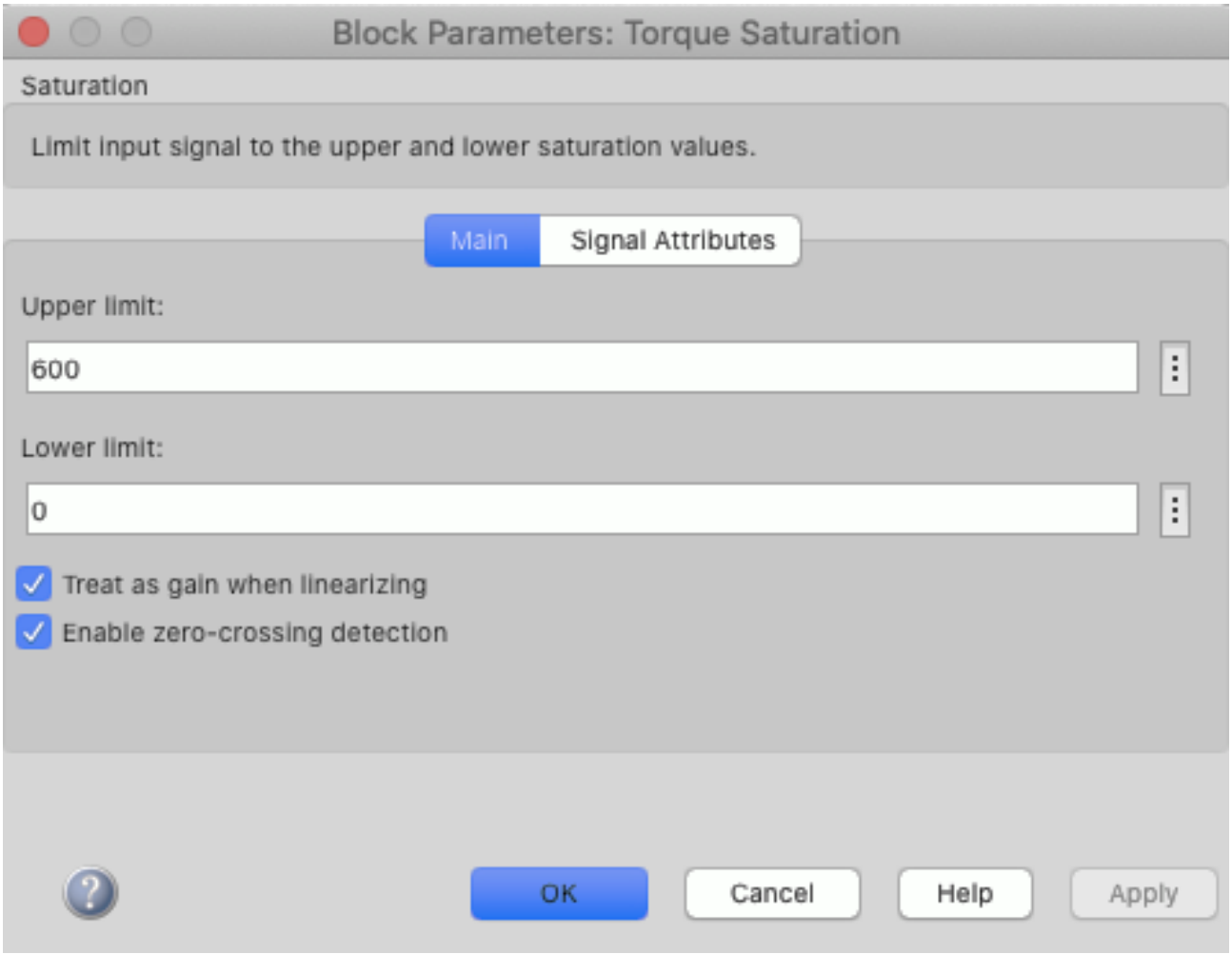
Ready 110% VariableStepAuto



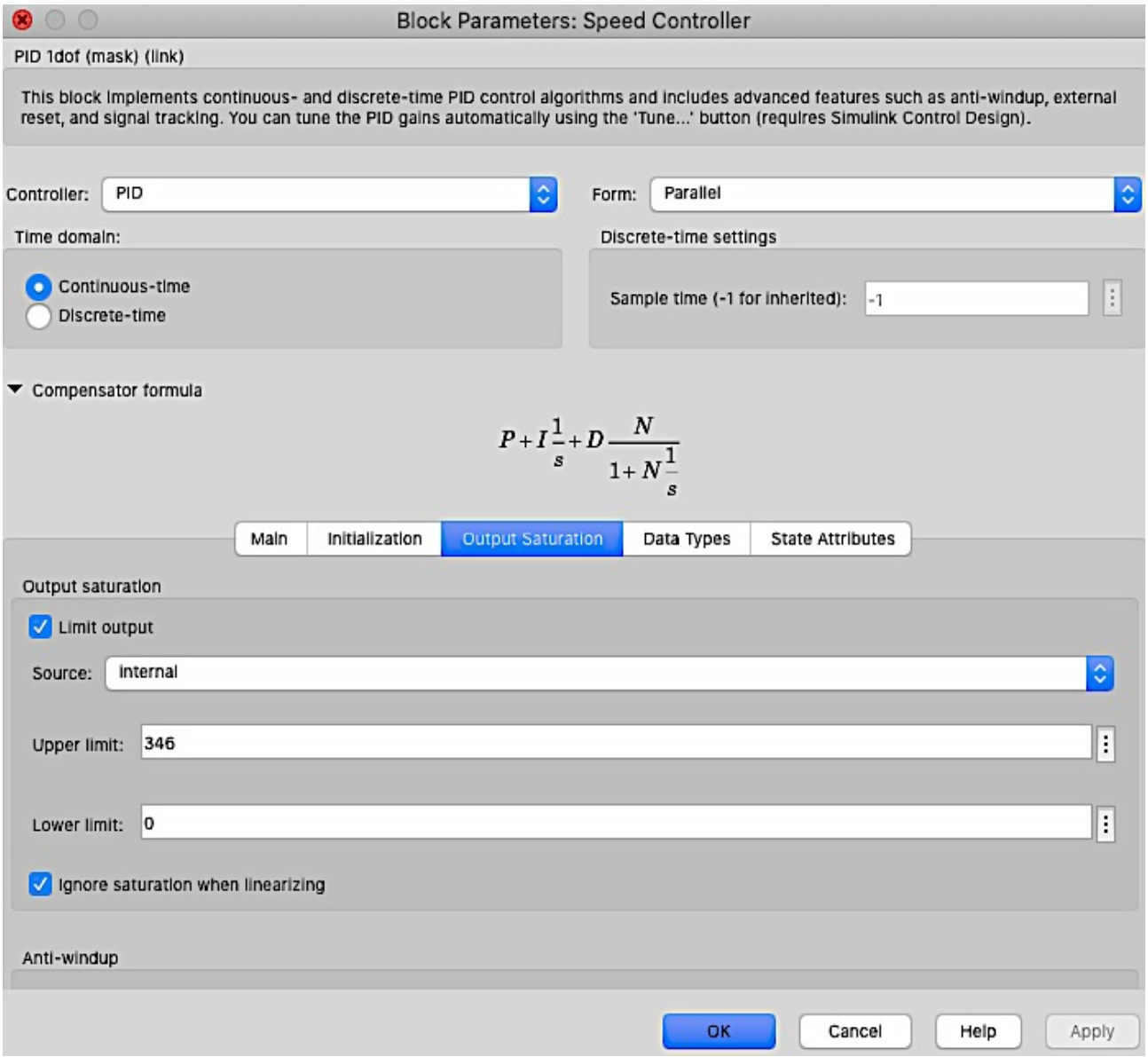
- I made a MATLAB Function in which I have defined the equation- A which is calculated previously.

- As you can see, this function takes Torque and Velocity (which are unknown quantity) as Input and at the end by doing some calculations it gives the Acceleration (dV/dt) as output.

- The beauty about the MATLAB & Simulink software is it automatically convert the time-domain to frequency-domain and vice versa.



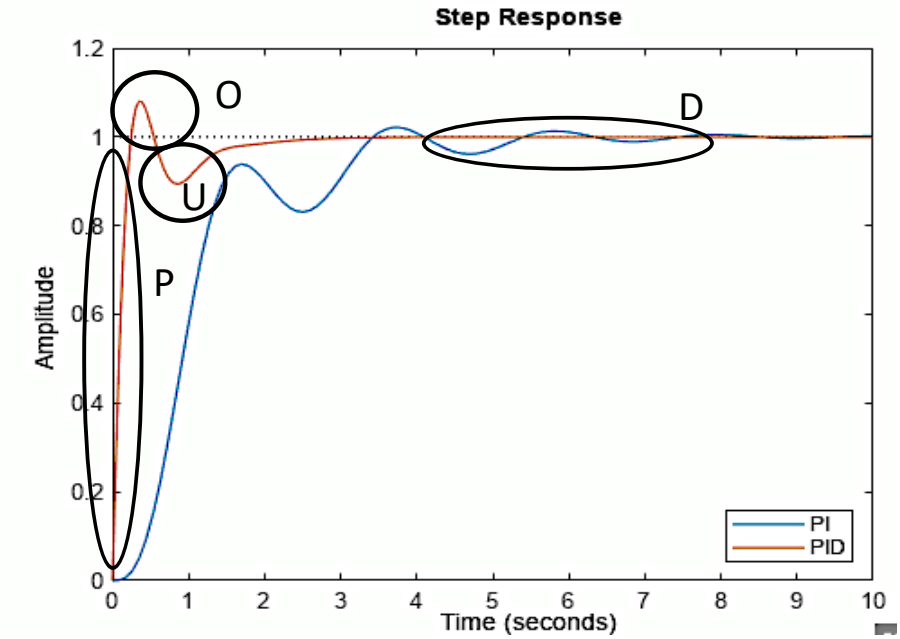
- As, the specified maximum torque of the Tesla Model S car is 600 Nm. So, I saturated the Torque limits ranges from 0 to 600. So, that we can get the appropriate and reasonalbe output.



- As per the battery calculations 346 V is maximum voltages Tesla Model S P85 battery can produce. So, as we have discussed before the voltage is the deciding factor to regulate and control the speed of vehicle.
- So, here I set the limits for PID Controller to tune always between 0 to 346 V. Otherwise the PID Tuner goes beyond the 346 value which is not valid to make a realistic simulink model.
- **So, here PID Contoller acts as a voltage controller or voltage regulator which is ultimately acts as so called Speed Controller.**

About PID Controller

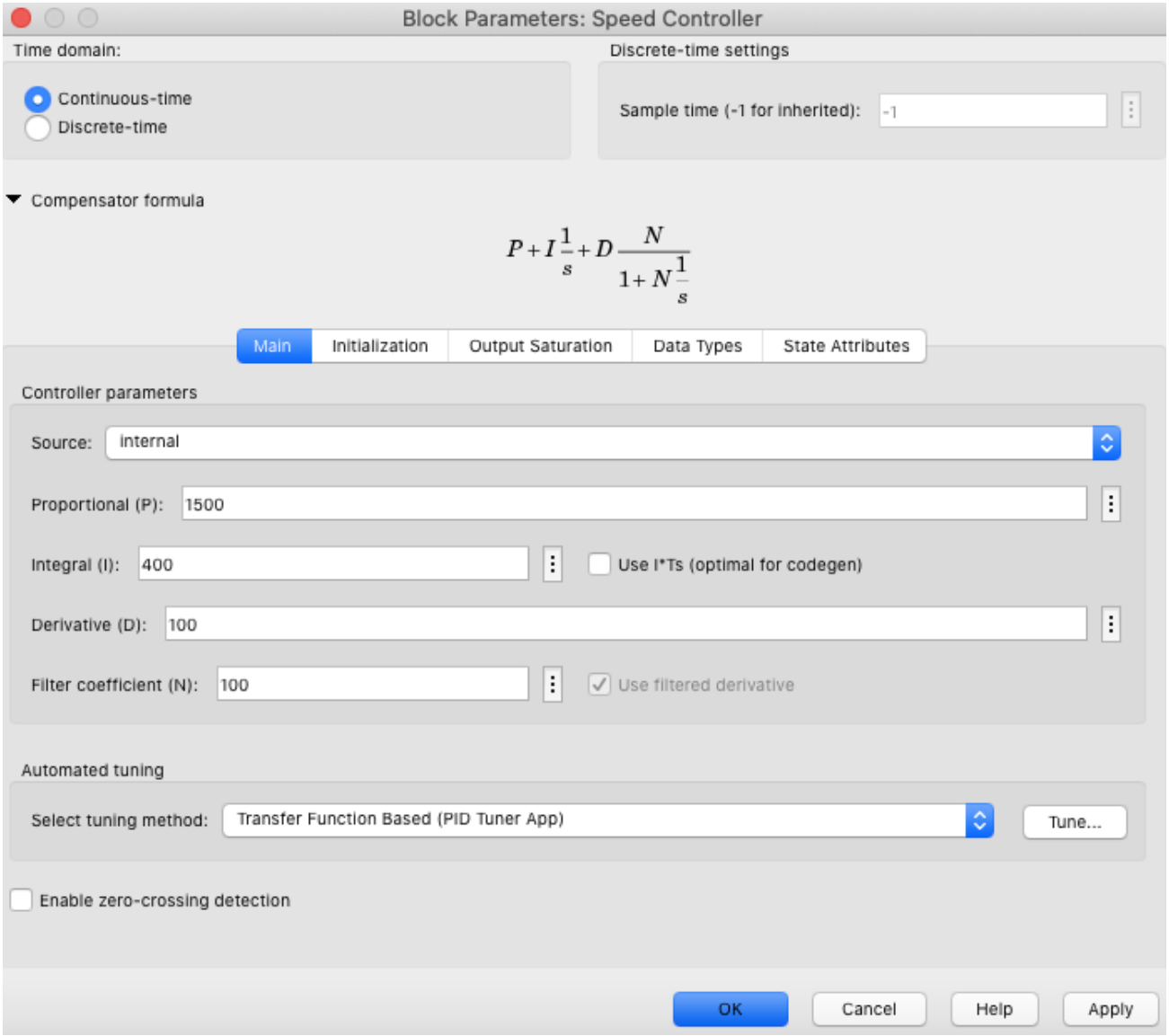
- The PID controller is basically works on time as well as frequency domain.
- **By means of tuning the PID contrtoller means changes the Gain values of Proportional, Integral and Derivative parameters seperately.**
- The main fucntion of PID controller is to minimize the error by these parameters. Each of parameters have a different role to control the error value.
- The error is the difference between the *Reference Speed Input* we want and feed to oput model and *Actual Speed Output* this simulink model offers which we get afterward by runing and simulating the it.
- Error= (Reference Speed Input – Actual Speed Output)
- This error is removed by the PID controller gradually with time.
- So, one can achieve the actual vehicle speed as output the Reference speed input as he/ she wants and expects from vehicle by delivering and feeding to model. To make possible the expectations of driver PID controller comes into picture and helps the model to achieve the output that driver tells the model to give.
- How PID controller helps the vehicle model to achieve this kind of things? So for that purpose, one needs to tune the PID parameters in a best possible way to make appropriate.
- Here, I also mentioned the output saturation of the *anti wind-up methods* e.g. none, clamping to get the most possible adequate results.



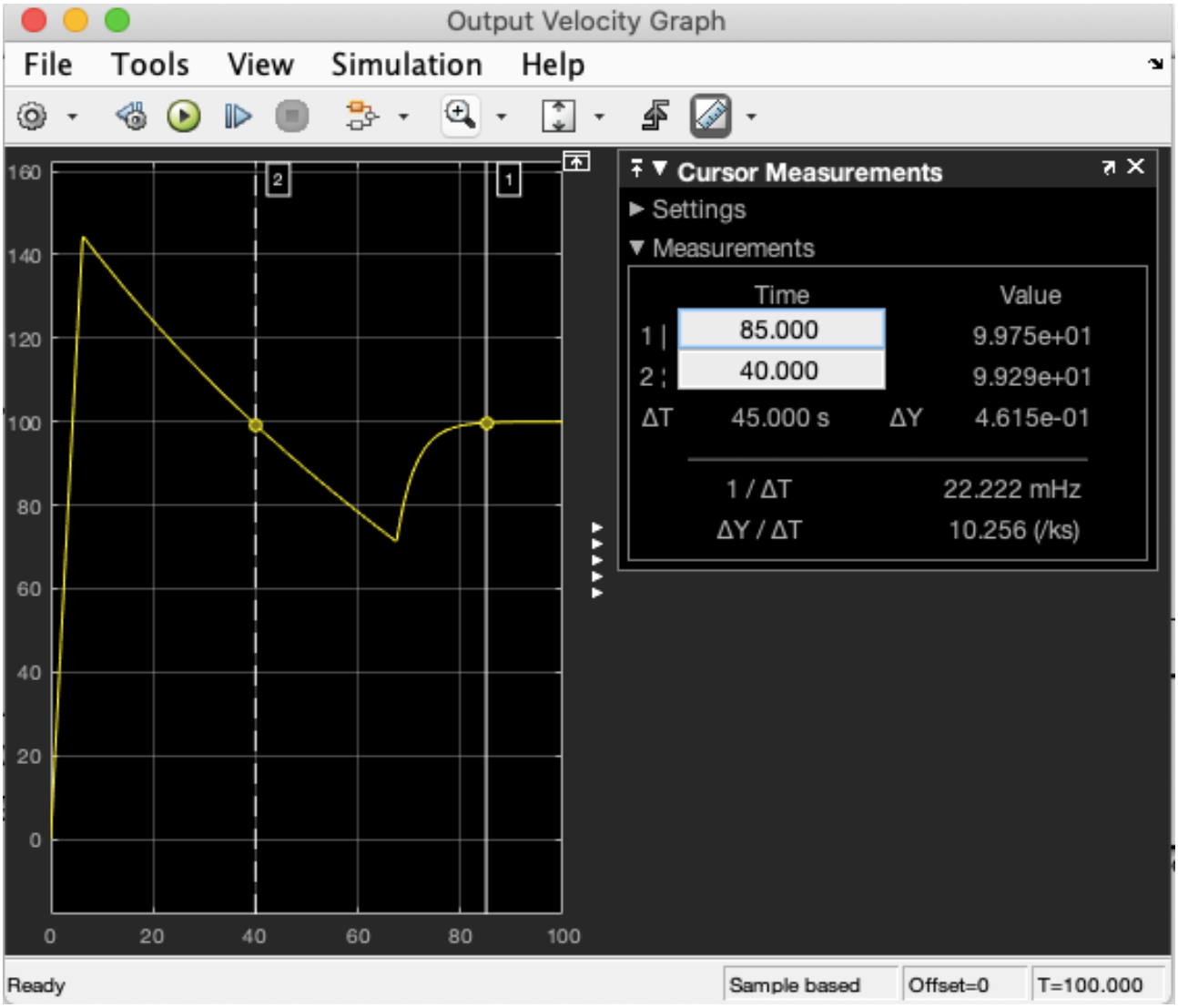
- **Proportional:** The proportional parameter trys to minimize the error very at initially shown in P-circle untill the steady state error left. So, it is important to keep the proportional gain value high to reach the error very close to zero as fast as possible.
- **Integral:** The Integral keeps increasing and bring the error to zero in closed loop system. It acts on the integrated error and try to reduce the error and bring to zero.
- **Derivative:** The derivative factor try to minimize the error which will occurred in the future. Becuase this parameter derive the error with time So, it breaks down the error and then eliminate. As one can see that without derivate factor only in PI the error is still there afterwards once the system becomes stabilize as seen in D-circle in figure. But in PID the error gets removed afterwards once the system becoms stabilize as seen in the same D-circle.
- Clamping method try to eliminate the overshoot as well as undershoot shown in O-circle and U-circle respectively. Because these methods put the limits on the Integral and Derivative parameters which soemtimes bring the error above or below zero.

Different PID Tuning for Testing the Speed Controller

- **Case-1: without Clamping (if Driver wants 100 kmph speed)**

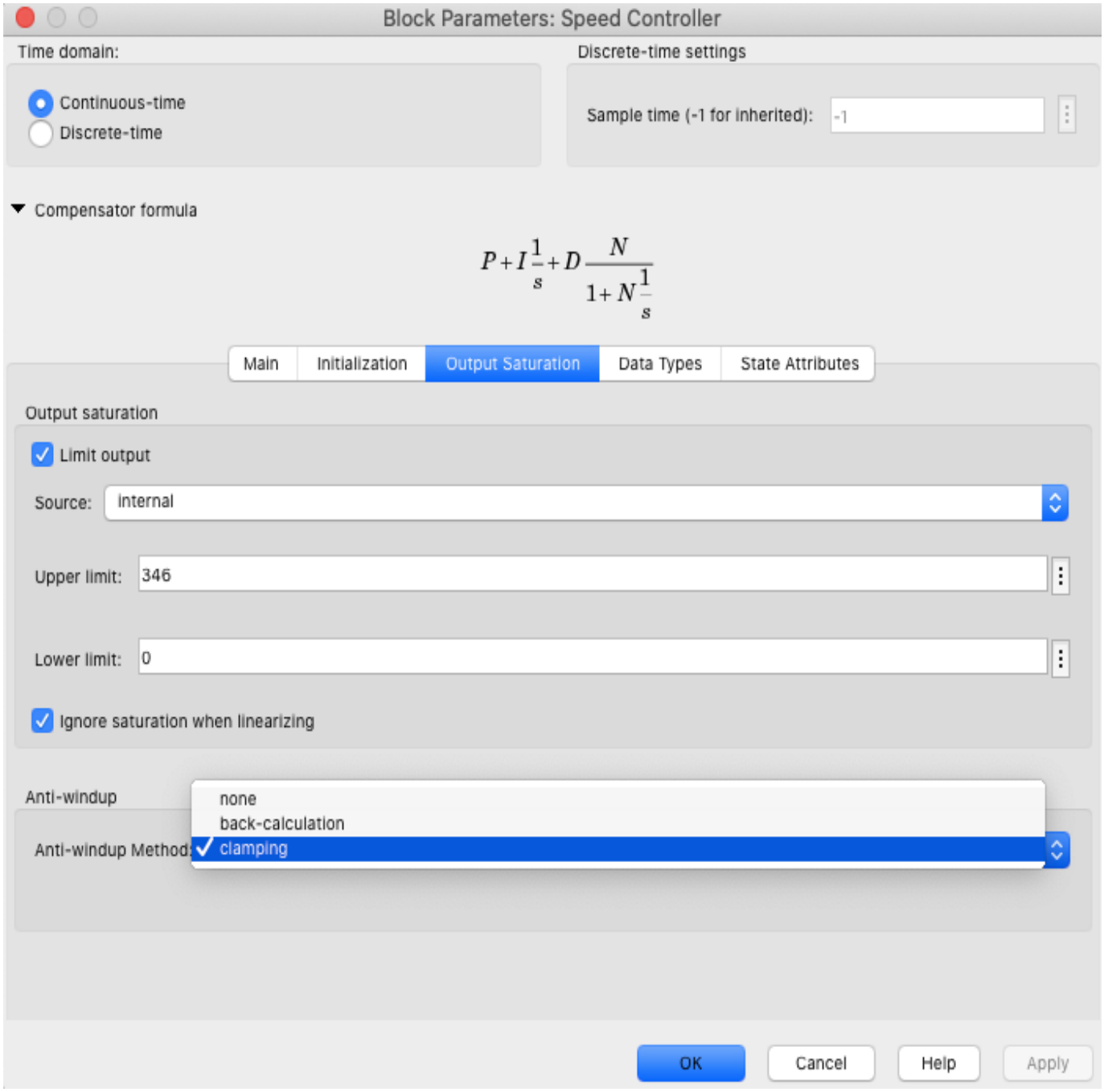


- I have set the Proportional gain value(P) at 1500 and Integral value(I) at 400. The derivate gain(D) is 100.The Filter Coefficient(N) at 100. without clamping and without zero-crossing detection.
- Let us see the output velocity graph.

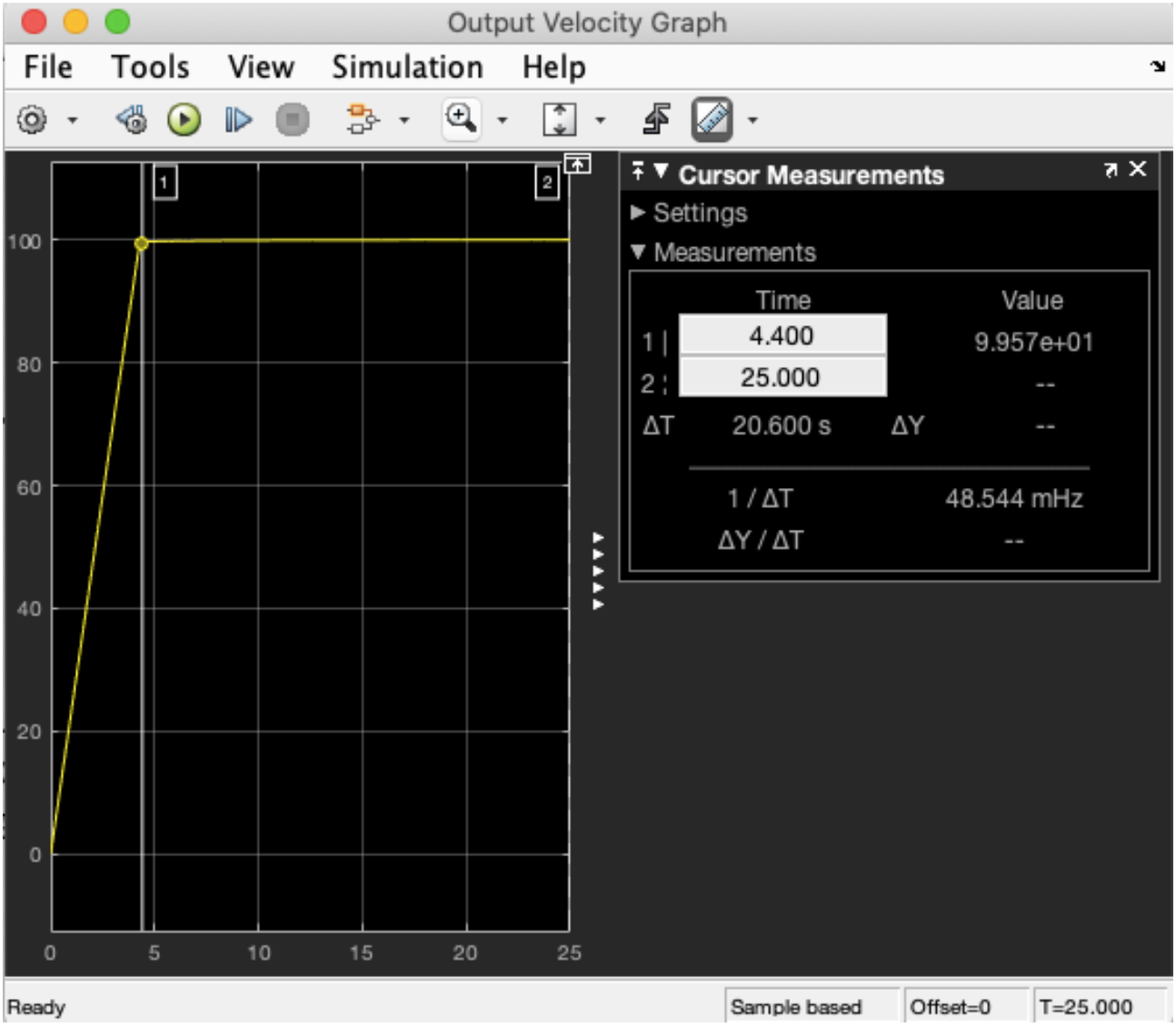


- Output Velocity graph depicts that at the very starting the Proportional parameter try to reduce the error right after the steady state error is left to reduce where the Integral parameter comes into action and try to reduce the lefted steady state error which eventually reduce that much of error that it goes down below 100 kmph at 40 seconds then it realizes to reduce itself and again goes up near to 100 kmph and at the end at around 85 seconds it touches the exact velocity which it is told by the driver to obtain.
- This is occur due to lack of anti wind up and zero-crossing detection it ramps up and down from the target speed.
- Here, the controller takes approximately about 85 seconds to reach the target speed at mentioned tuning values.

- **Case-2: with Clamping (if Driver wants 100 kmph speed)**



- Now enable the zer-crossing detection and clamping in this case to see what will be the result.



- Now, one can observed that the graph has became very smooth and uniform try to reach to target speed without any overshoot or undershooting. At about 4.4 seconds the car reach 100 kmph speed.

- So, here in this case PID controller takes roughly about 4.4 s to minimize the error with the same tuning values as taken in previous case-1, but including clamping and zero cross detection.

- I have neglected the Inertia because as I said I pressumed the vehicle at constant speed or in moving condition. So, for that inertia can be neglected.
- By this simulink model which I have made, the driver can achieve any speed he or she wants whether it is 100 kmph or 60 kmph or 150 kmph.
- I have tried my best to make it as much as realistic.