Design & Simulate SolarPanel Tracking System

Prepared By:
Nishantkumar V Patel

About Project

The solar tracking system will be Single axis in my project.

The solar tracking system includes 2 main subsystem. One is Panel and second is Motor. Here I will design the Simulink model using few equations regarding Motor and Panel motion. Then simulate and check the output of panel's angular position.

To make my system precise here I would apply the PID controller to minimize the error which will occurred as result of difference between actual sun's position and panel's position.

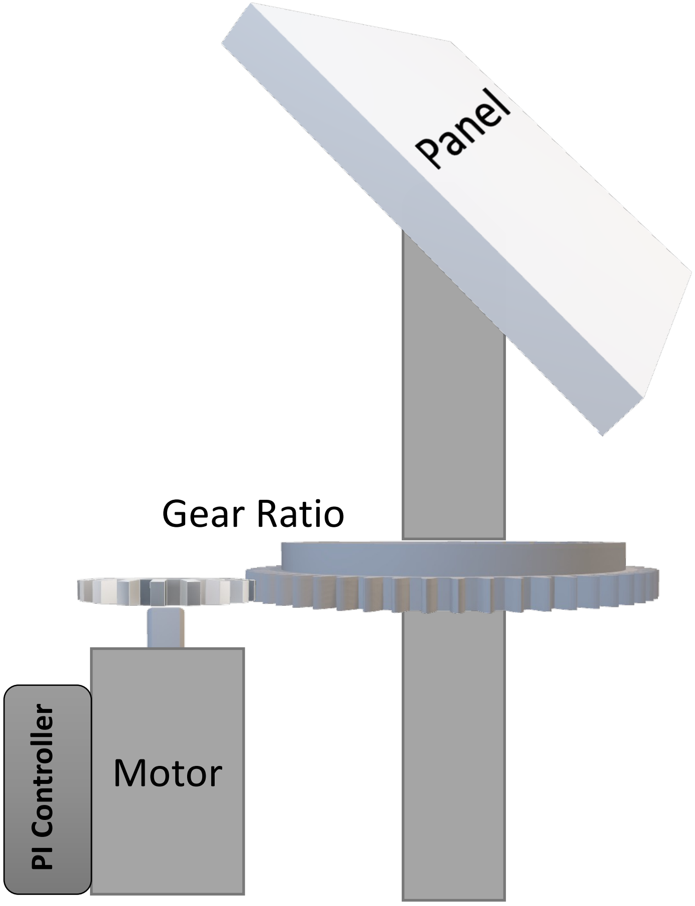
At the end, I compare the reference input so called Sun's different position during the whole day with my designed panel's position.

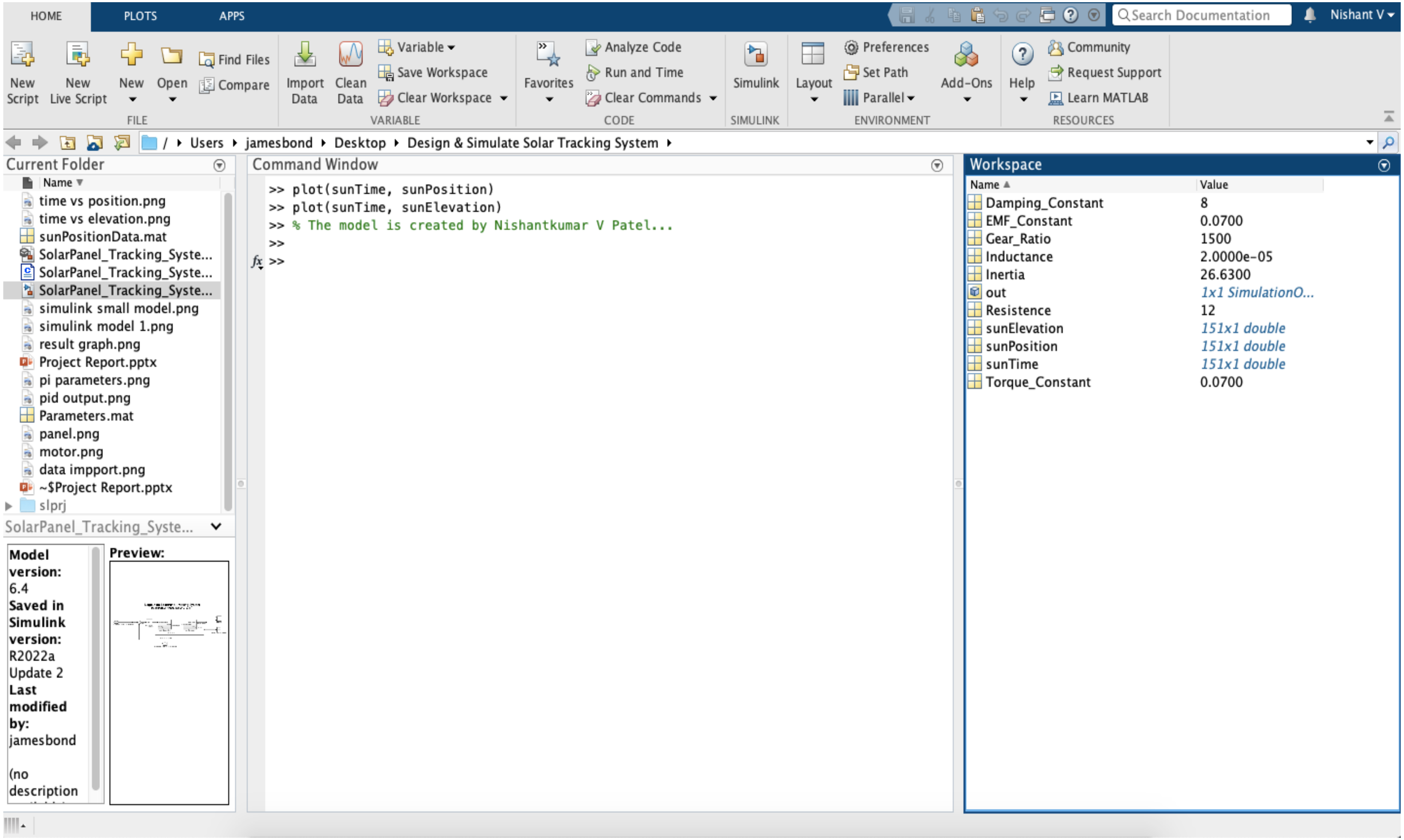
To ensure that whether the panel correctly tracks the sun over the period of time or not.

So, lets get Started…

Assumptions:

- I have taken the case of vertically rotating solar tracker.
- The dc motor I have taken into account with common dc motor specifications.
- I have taken the Solar panel’s dimesions and weight hypothetically.
- The above picture gives you the idea about how my tracking system will working.





Current Folder

Name
- time vs position.png
- time vs elevation.png
- sunPositionData.mat
- SolarPanel_Tracking_Syste...
- SolarPanel_Tracking_Syste...
- SolarPanel_Tracking_Syste...
- simulink small model.png
- simulink model 1.png
- result graph.png
- Project Report.pptx
- pi parameters.png
- pid output.png
- Parameters.mat
- panel.png
- motor.png
- data impport.png
- ~$Project Report.pptx
- slprj

SolarPanel_Tracking_Syste...

Model version: 6.4
Saved in Simulink version: R2022a Update 2
Last modified by: jamesbond

(no description

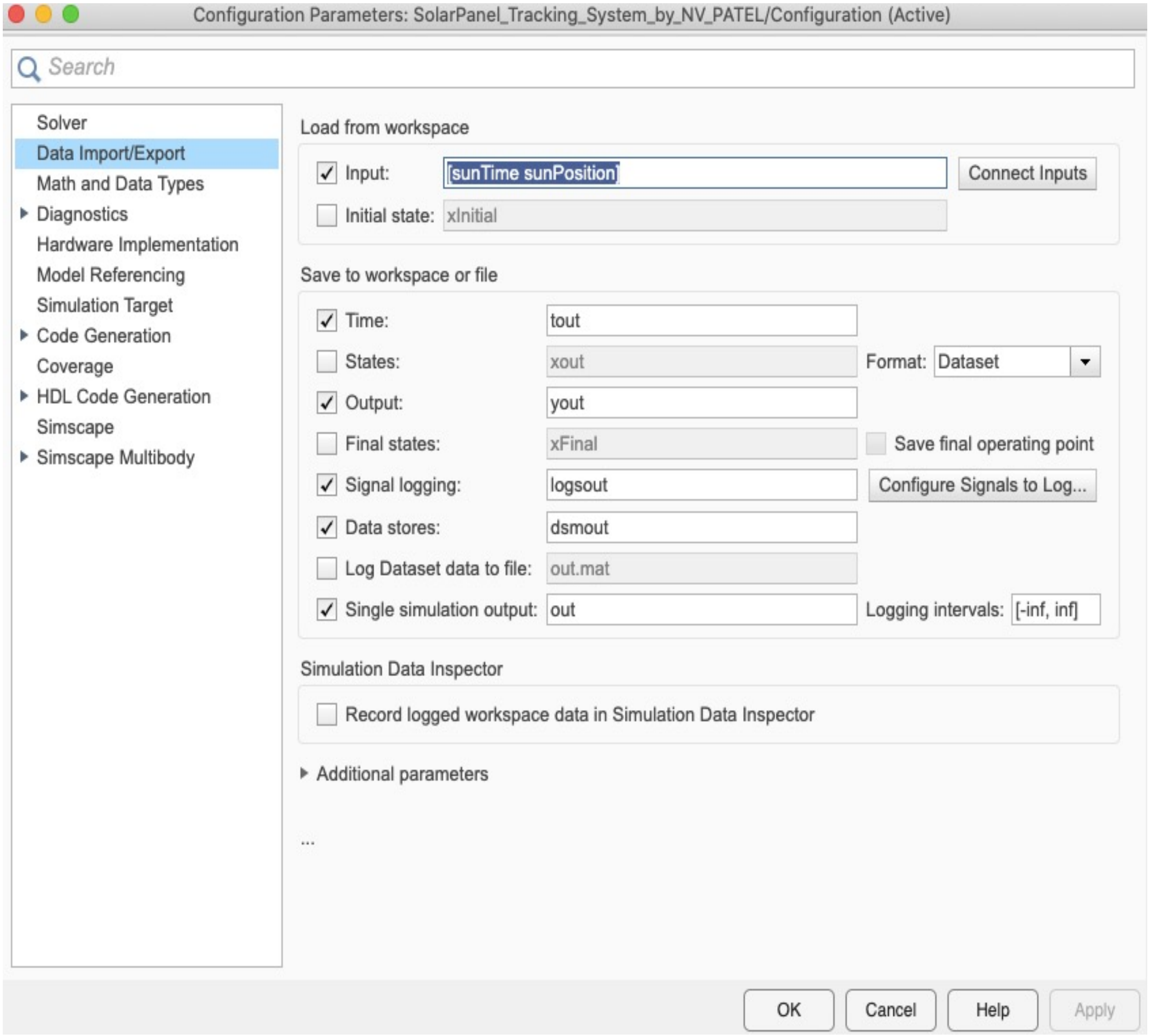
Preview:

Command Window

```
>> plot(sunTime, sunPosition)
>> plot(sunTime, sunElevation)
>> % The model is created by Nishantkumar V Patel...
>>
fx >>
```

Workspace

Name	Value
Damping_Constant	8
EMF_Constant	0.0700
Gear_Ratio	1500
Inductance	2.0000e-05
Inertia	26.6300
out	*1x1 SimulationO...*
Resistence	12
sunElevation	*151x1 double*
sunPosition	*151x1 double*
sunTime	*151x1 double*
Torque_Constant	0.0700



Configuration Parameters: SolarPanel_Tracking_System_by_NV_PATEL/Configuration (Active)

Load from workspace

☑ Input: sunTime sunPosition

□ Initial state: xInitial

Save to workspace or file

☑ Time: tout

□ States: xout Format: Dataset

☑ Output: yout

□ Final states: xFinal □ Save final operating point

☑ Signal logging: logsout

☑ Data stores: dsmout

□ Log Dataset data to file: out.mat

☑ Single simulation output: out Logging intervals: [-inf, inf]

Simulation Data Inspector

□ Record logged workspace data in Simulation Data Inspector

- To import the sun position and time data of .mat file I went to the ‘model parameters/ configuration parameters’ the by typing the ‘.mat file names’ in the ‘Load from workspace’ section I did it so.
- It is important to note that always type in

[time_data other_data] format to import correctly.

The click on the ‘connect inputs’ button.

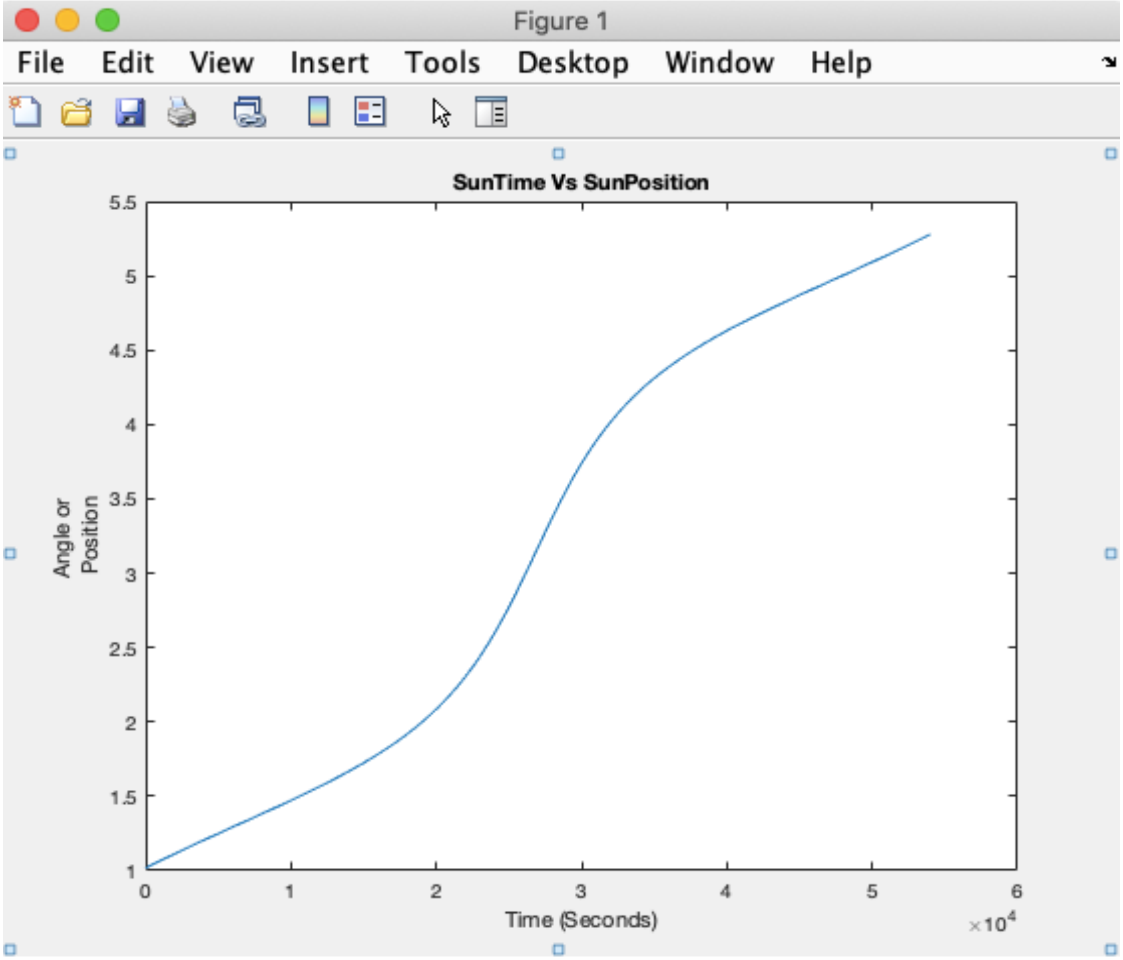


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
SunTime Vs SunPosition
Angle or Position
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0
1
2
3
4
5
6
Time (Seconds)
×10^4

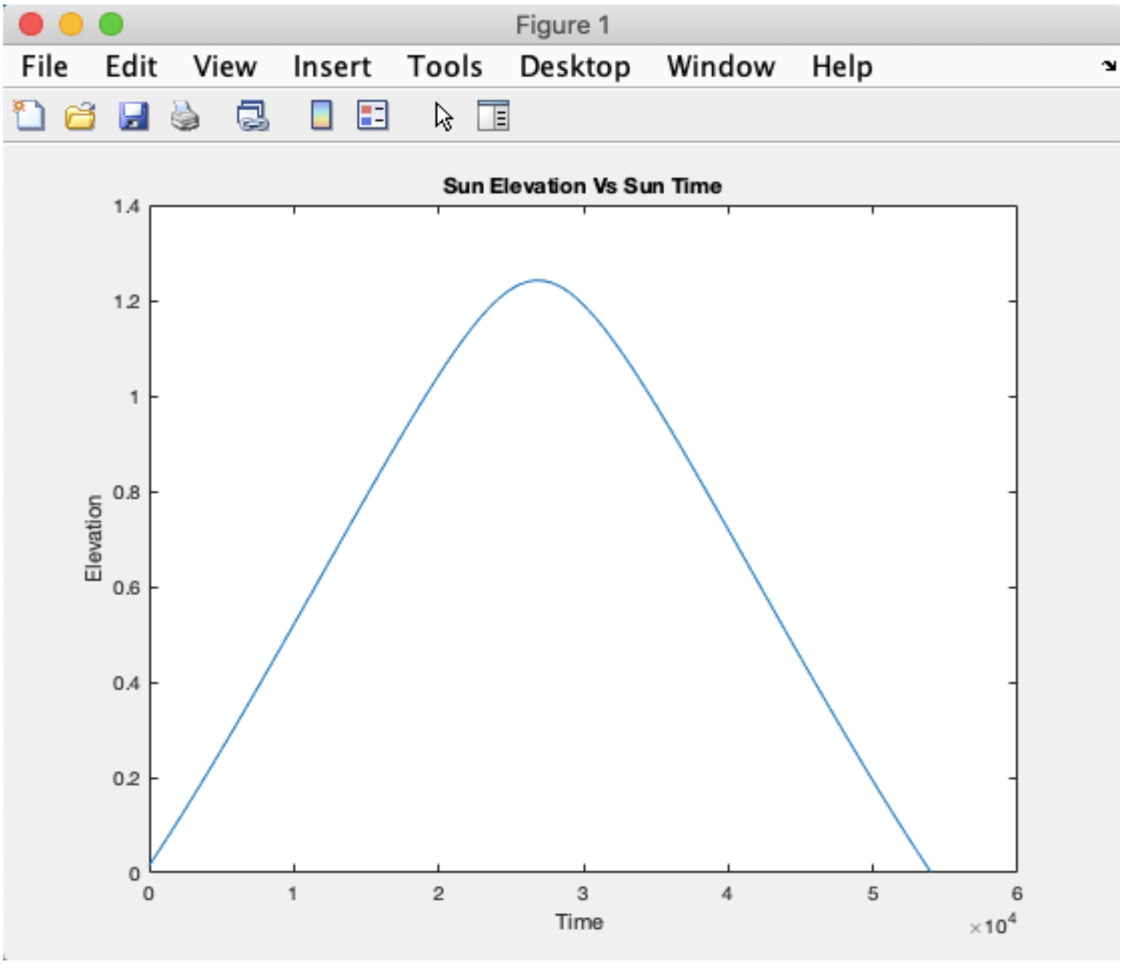


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Sun Elevation Vs Sun Time
Elevation
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
Time
×10^4

Panel Subsystem

The Solar panel size and its diemsions are important factors to be considered because from which one can derived the Inertia of the rectangle shaped panel.

<u>Panel Parameters:</u>

Mass (m)= 75 kg

Width (w)= 1.5 m

Length (l)= 2 m

Depth (d)= 0.15 m

Elevation Angle (α)= 45^0 (ideal angle for solar panel as 45 is the most effective angle for parabolic motions)

Damping Constant (Kd)= 8

θ.

The Inertia formula for rectangular shape,

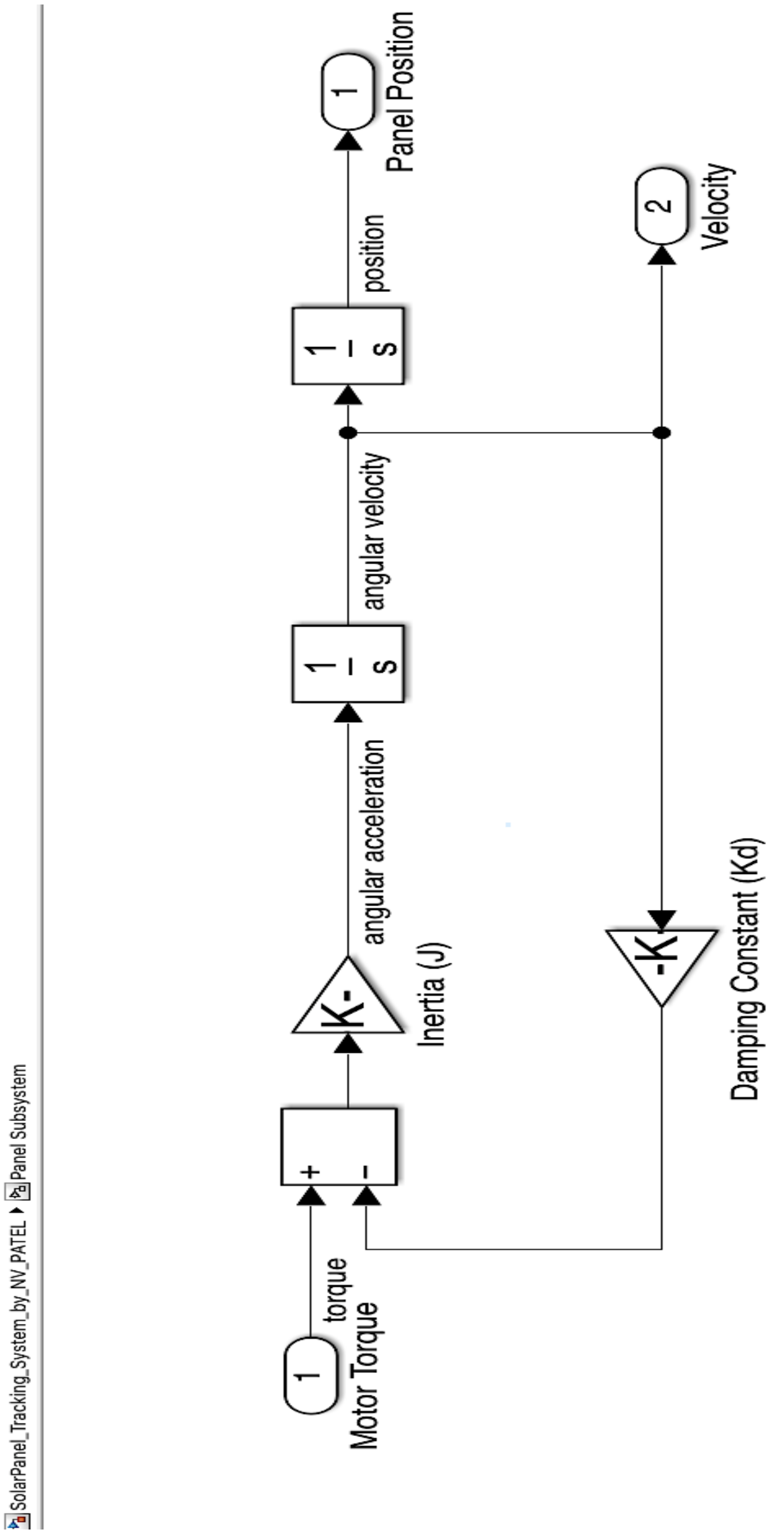
$J= m/12*[l^2*\cos\alpha^2 + d^2*\cos\alpha^2 + w^2]$

$J= 26.63\ kg*m^2$

This is the solar panel motion equation,

$$\therefore\ d^2\theta/dt^2 = 1/J *[T - (Kd*d\theta/dt)]$$

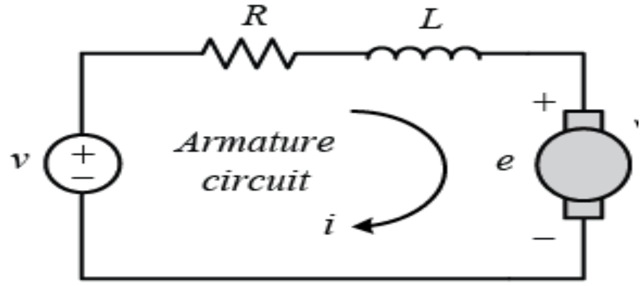
: $\frac{d^2\theta}{dt^2} = 1/J * [T - (Kd * d\theta/dt)]$



This equation is mathematically modelled and I have made the panel subsystem using different blocks like Subtract, Gain, Integrator. There is one Inport and Outport available in Panel subsystem which will taking input data and delivering the output data respectively.

Motor Subsystem

DC Motor Calculation:



<u>Motor Parameters:</u>

Emf constant (Kf)= 0.07 v/(rad/s)

Torque Constant (Kt)= 0.07 Nm/A

Inductance (L)= $2*10^{-5}$ H

Resistence (R)= 12 Ohms

Gear Ratio (Kg)= 1500

- 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,

$$\therefore di/dt= \int 1/L *[v-Kg*Kf*(d\theta/dt)-i(t)R]$$

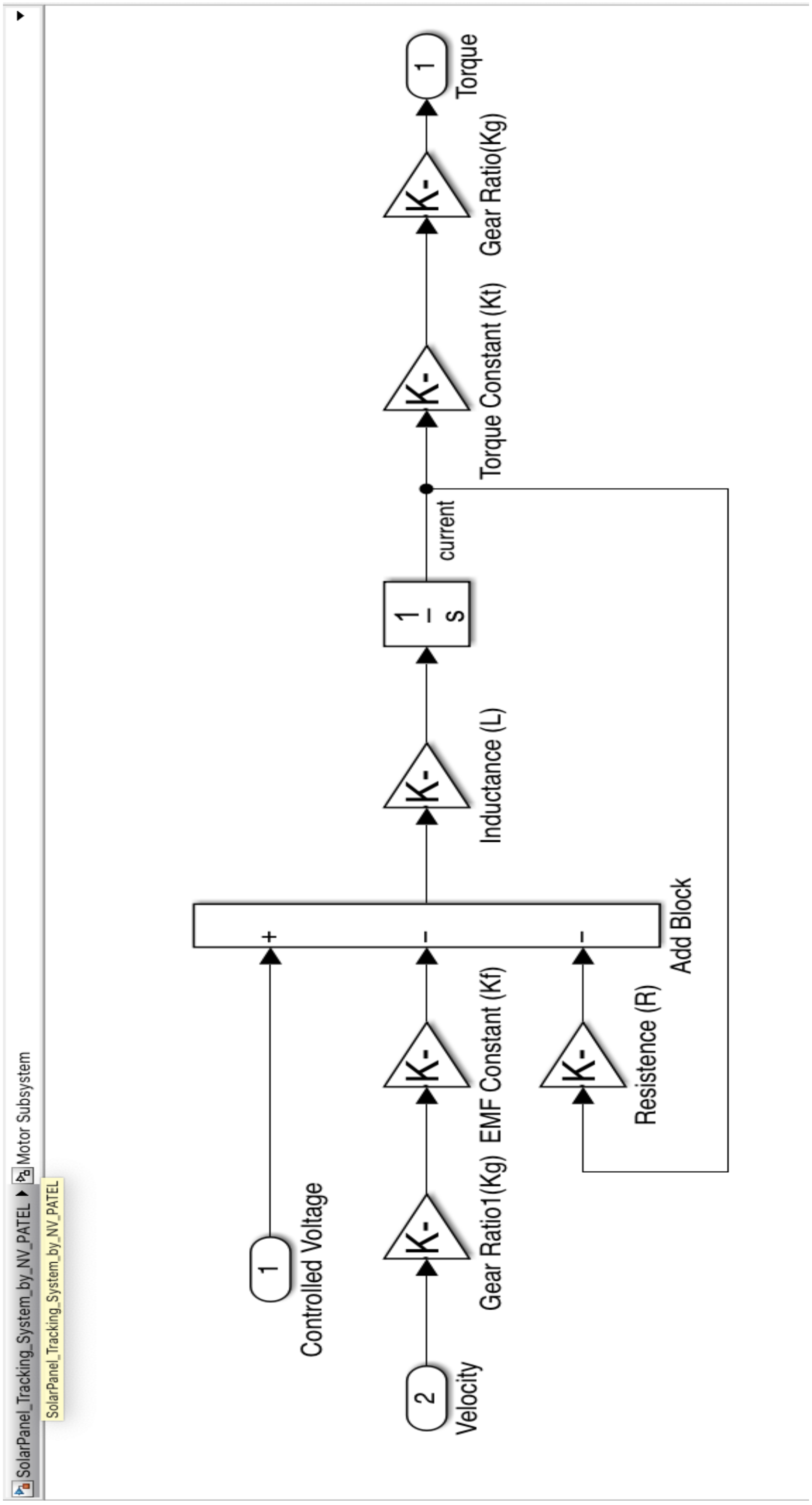
- As the current is directly proportional to motor torque,

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

Based on these two equations I have modelled the subsystem in simulink.

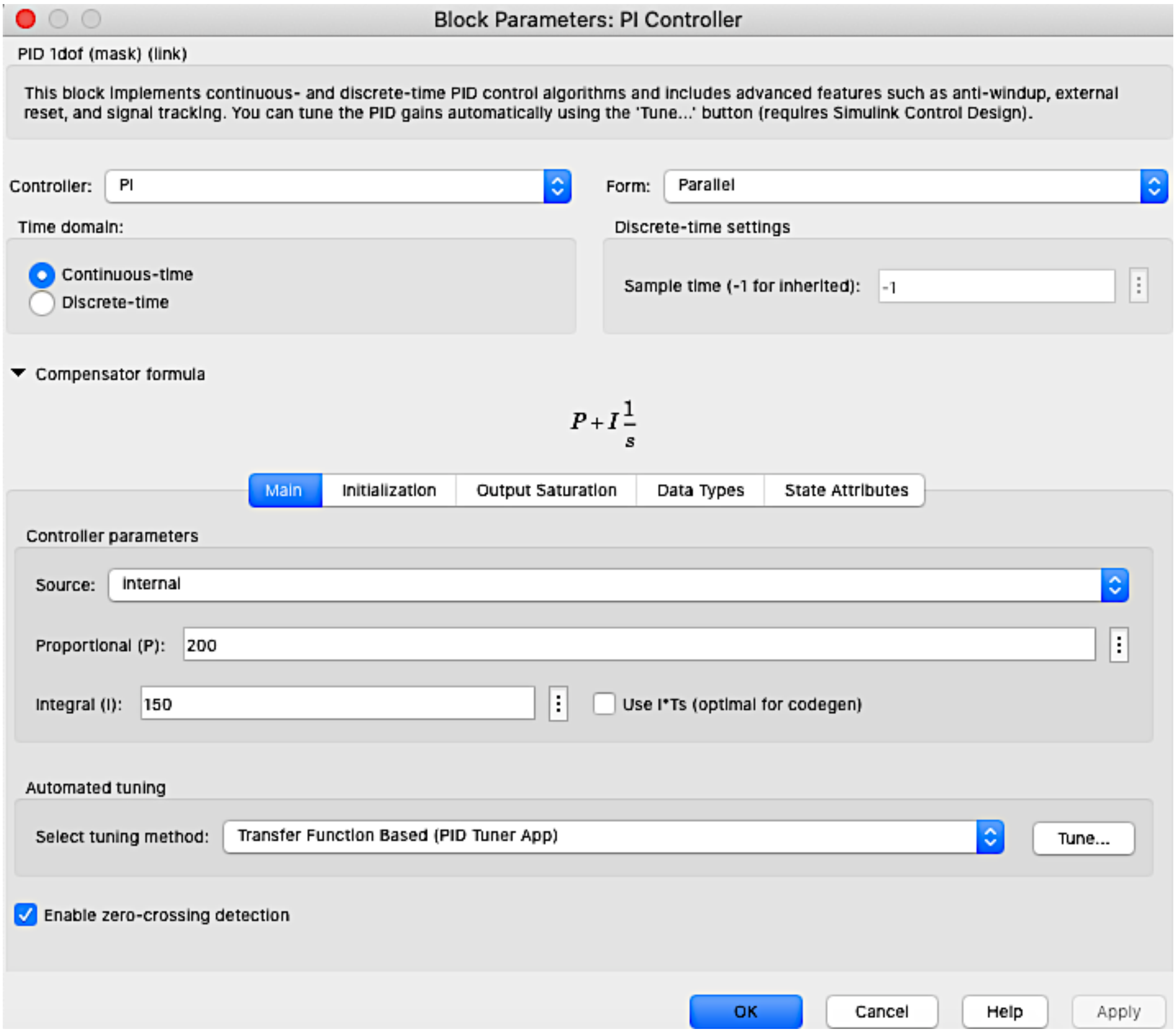
:: T = Kg*Kt*i(t)

:: di/dt= ∫ 1/L *[v−Kg*Kf*(dθ/dt)−i(t)R]



The above two equations which are completely describe the motor is mathematically modelled in simulink with the help of diffent blocks like Gain, Add, Integrator. There are two Inports and one Outport present in motor subystem.

PI Controller



PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PI

Form: Parallel

Time domain:

● Continuous-time
○ Discrete-time

Discrete-time settings

Sample time (-1 for inherited): -1

Compensator formula

$$P + I\frac{1}{s}$$

Controller parameters

Source: internal

Proportional (P): 200

Integral (I): 150

□ Use I*Ts (optimal for codegen)

Automated tuning

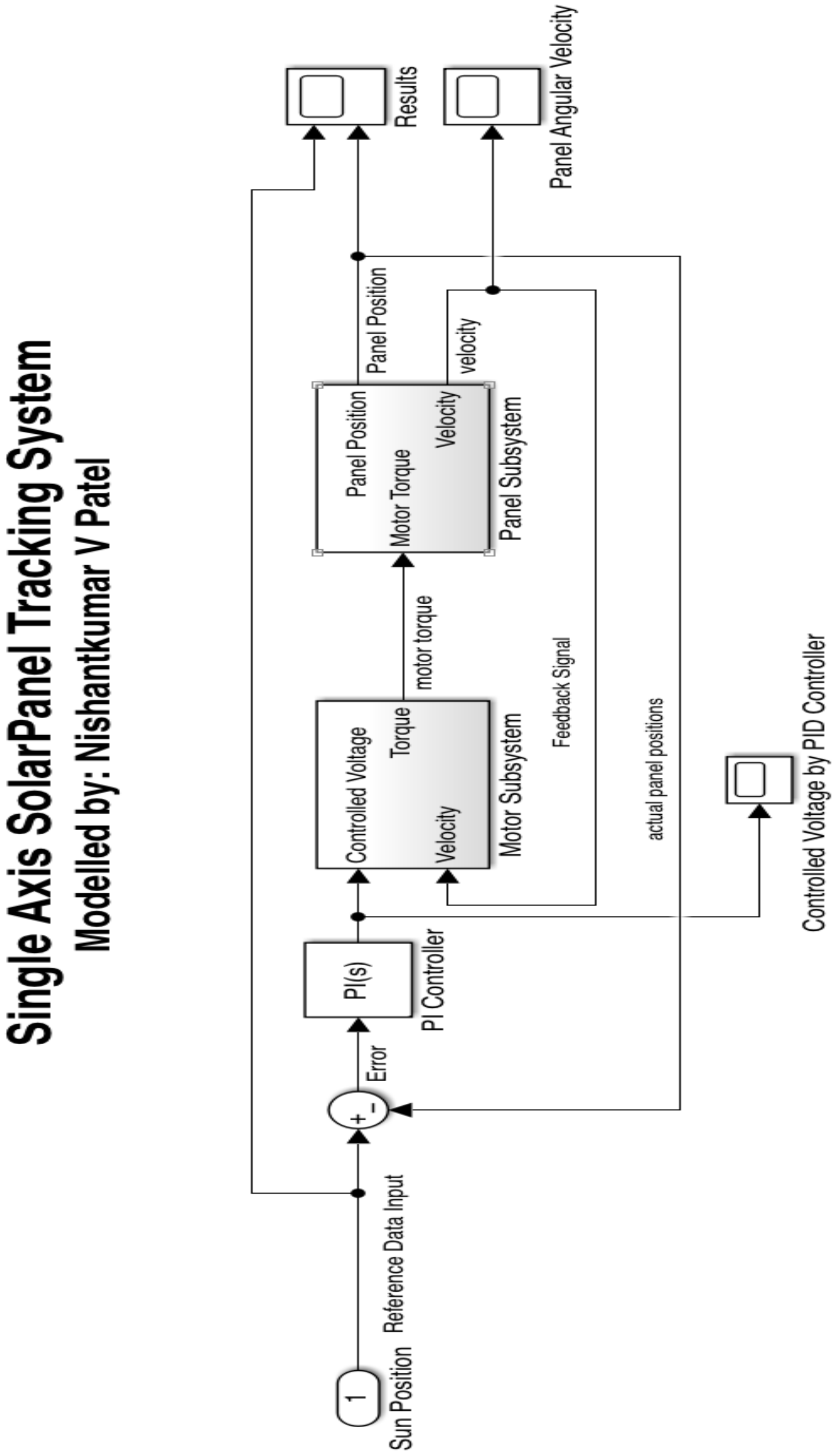
Select tuning method: Transfer Function Based (PID Tuner App)

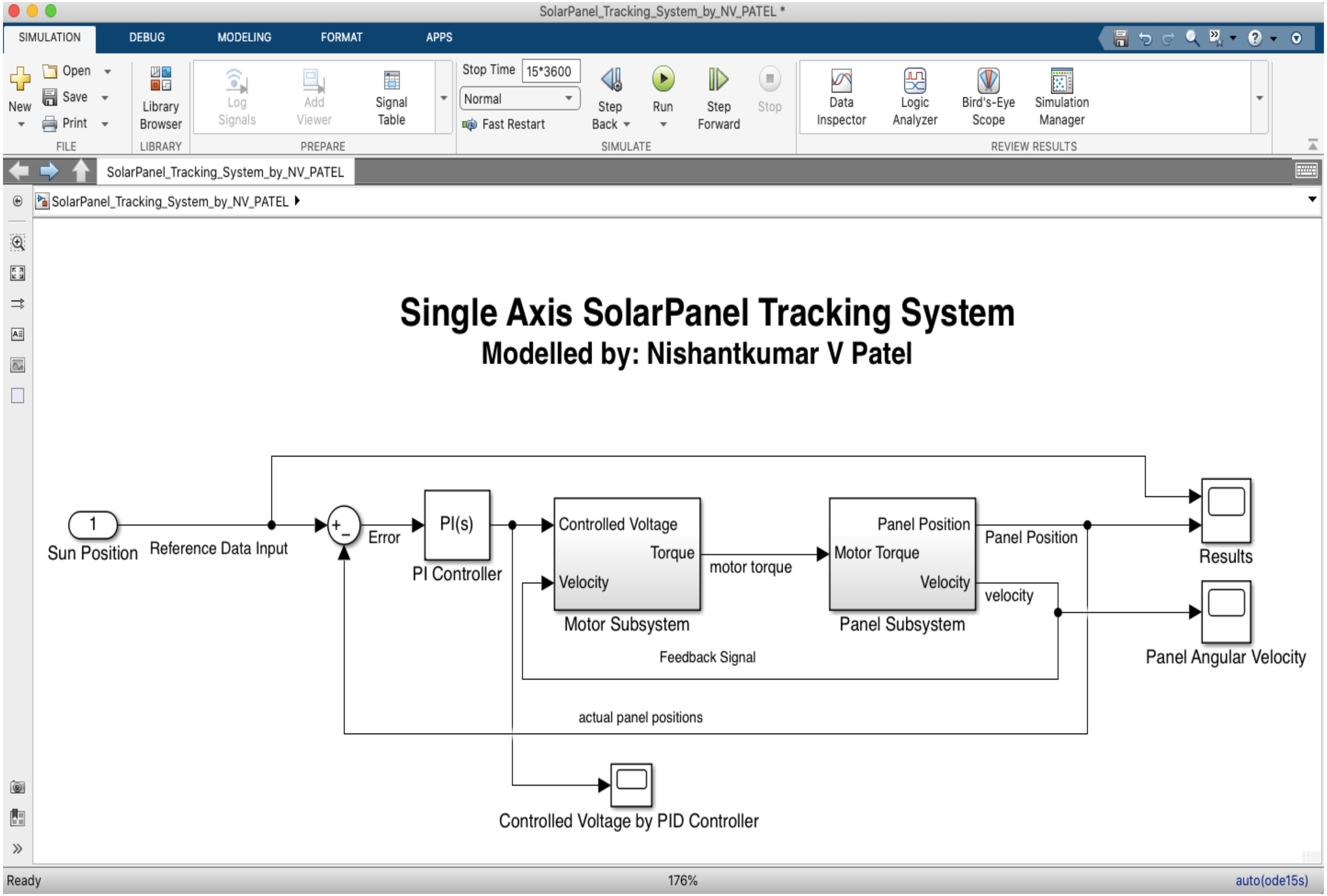
☑ Enable zero-crossing detection

- I have take PI gain values and neglecting the derivative gain by keeping its value 0. The reason is sun is moving steadily over the period of time and not chaning its position so fast so there is not rate of change of error produced in the closed loop system.

Proportional Gain=200, Integral Gain=150

Simulink Model





SolarPanel_Tracking_System_by_NV_PATEL *

Stop Time 15*3600

Normal

Single Axis SolarPanel Tracking System

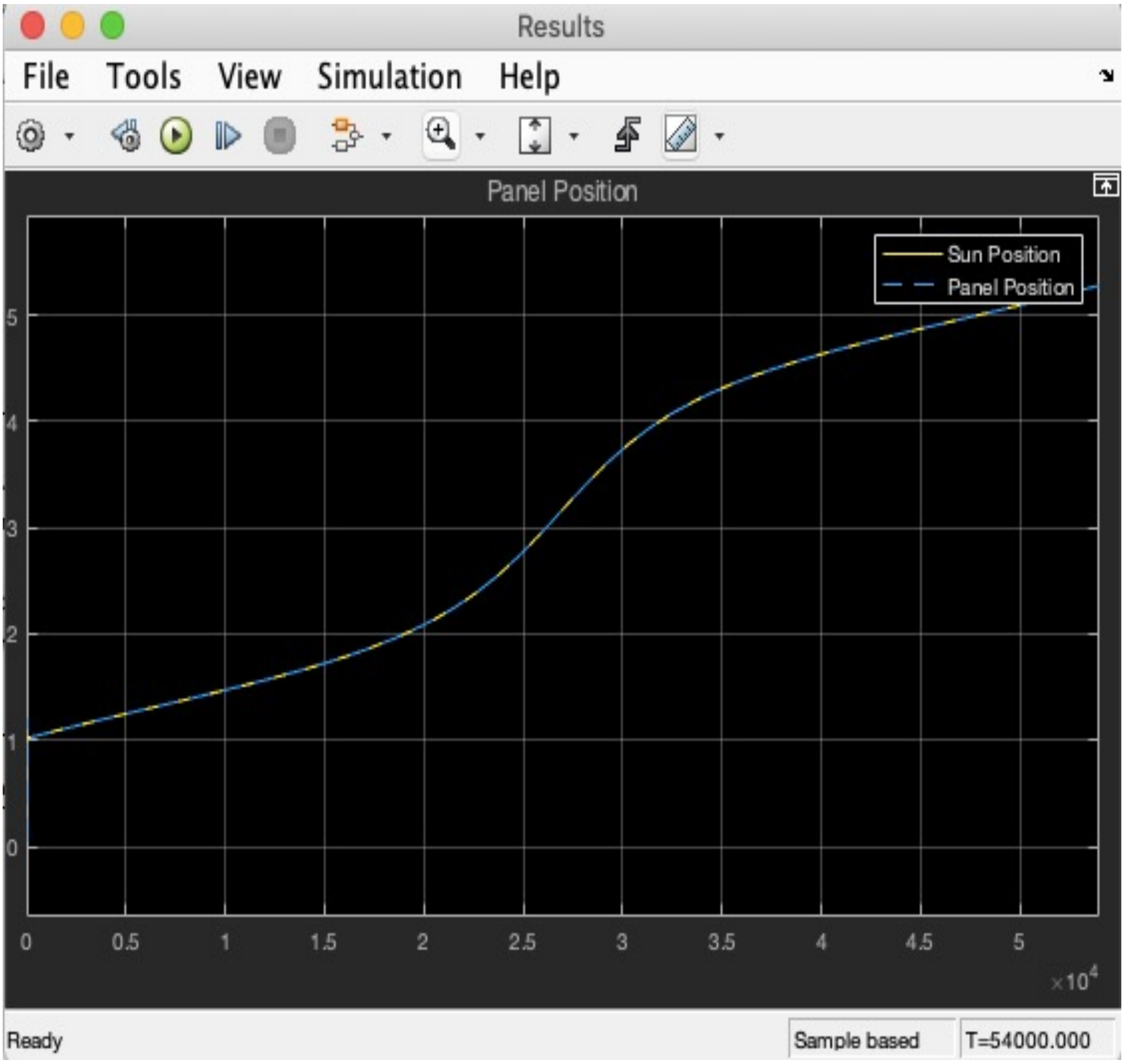
Modelled by: Nishantkumar V Patel

Ready

176%

auto(ode15s)

Testing the Model



- As You can see the the panel position constantly pointing with the sun position during the day. Here I took the simulation time as per the data of 15 hours means 54,000 seconds for the day.
- The X-axis is indicating simulation time 54,000 sec.
- The Y-axis is indicating the angle with the scale from 1 to 5.27 as per the sunPosition data.