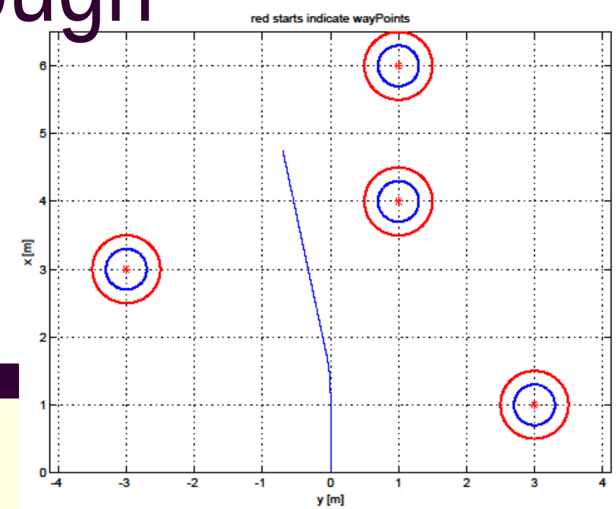
Guidance, Navigation and Control (GNC) of UGV for Travelling through Waypoints



Atilla Dogan

XX4378 and XX5378 – Introduction to UVS

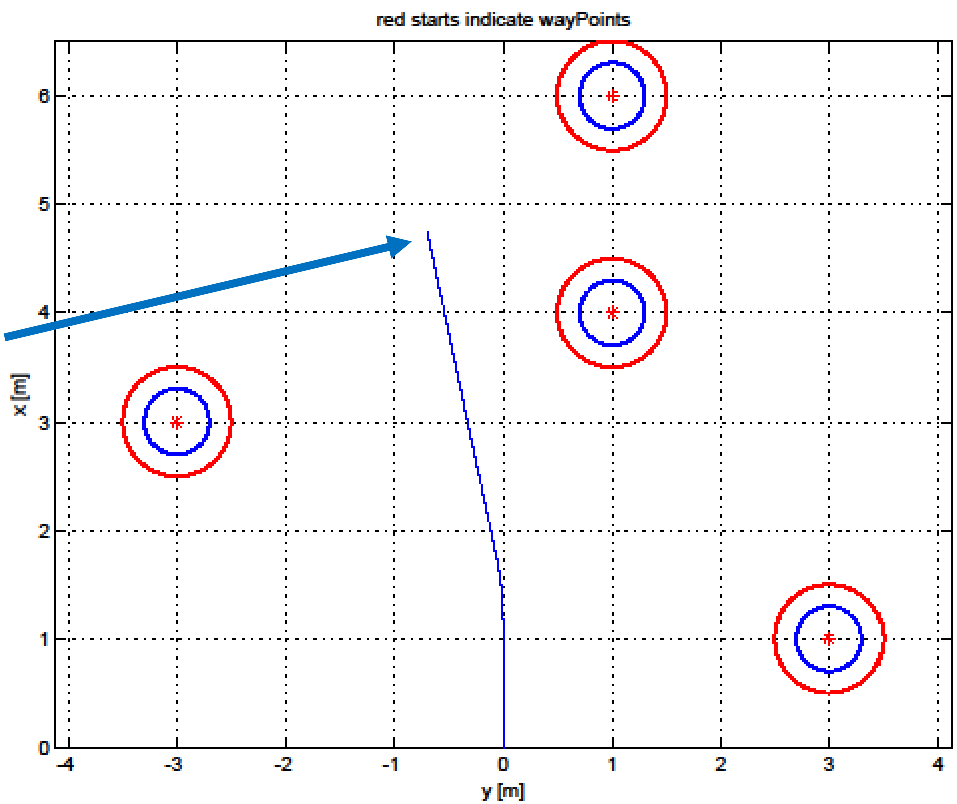
September 27,October 2 and 4, 2017

Fall 2017

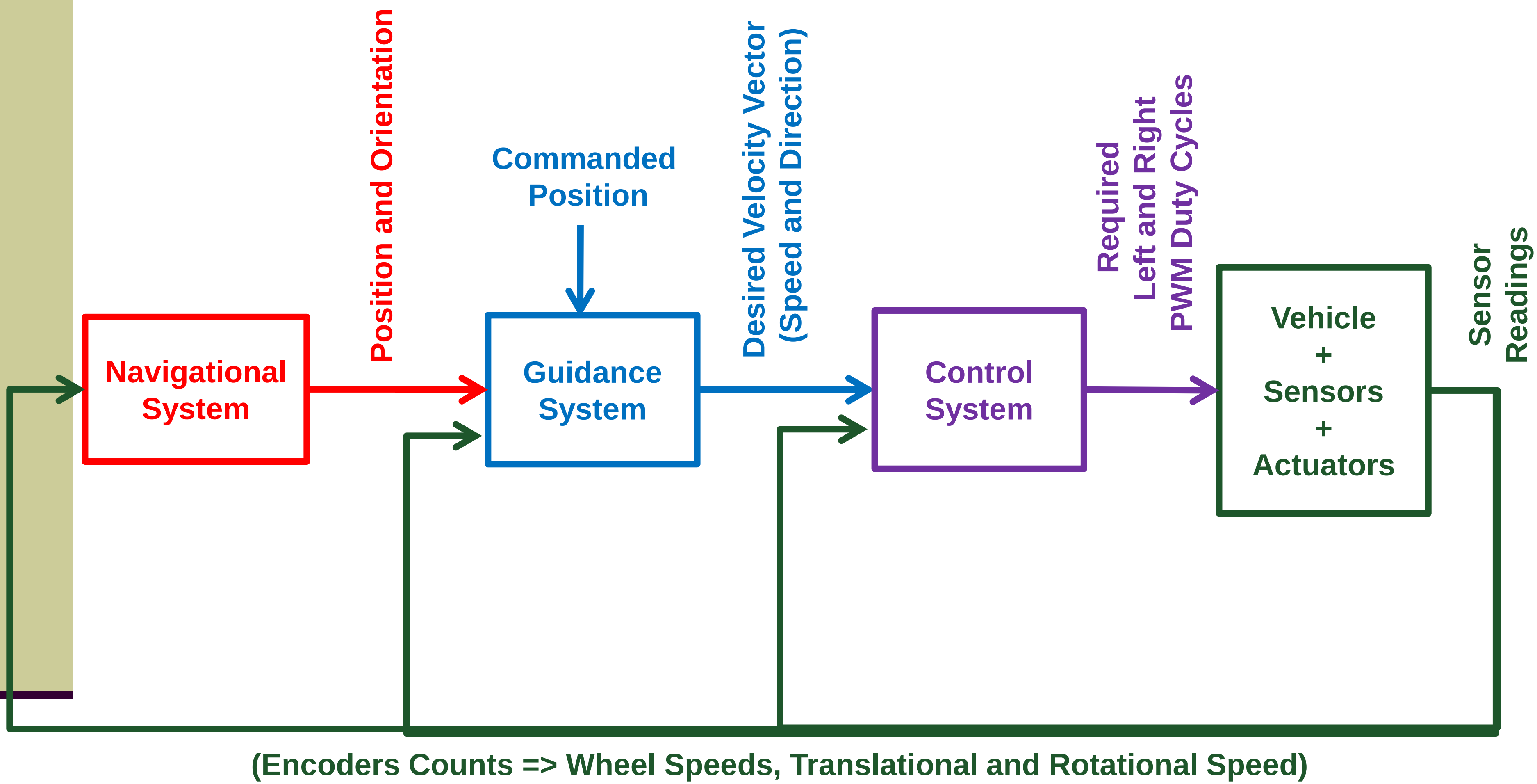
Guidance Navigation and Control (GNC)

- **Navigation:** What are the position and orientation of the vehicle?
- **Guidance:** What should be the vehicle's velocity (speed and direction) to get where it needs to go?
- **Control:** How to achieve the required velocity vector?

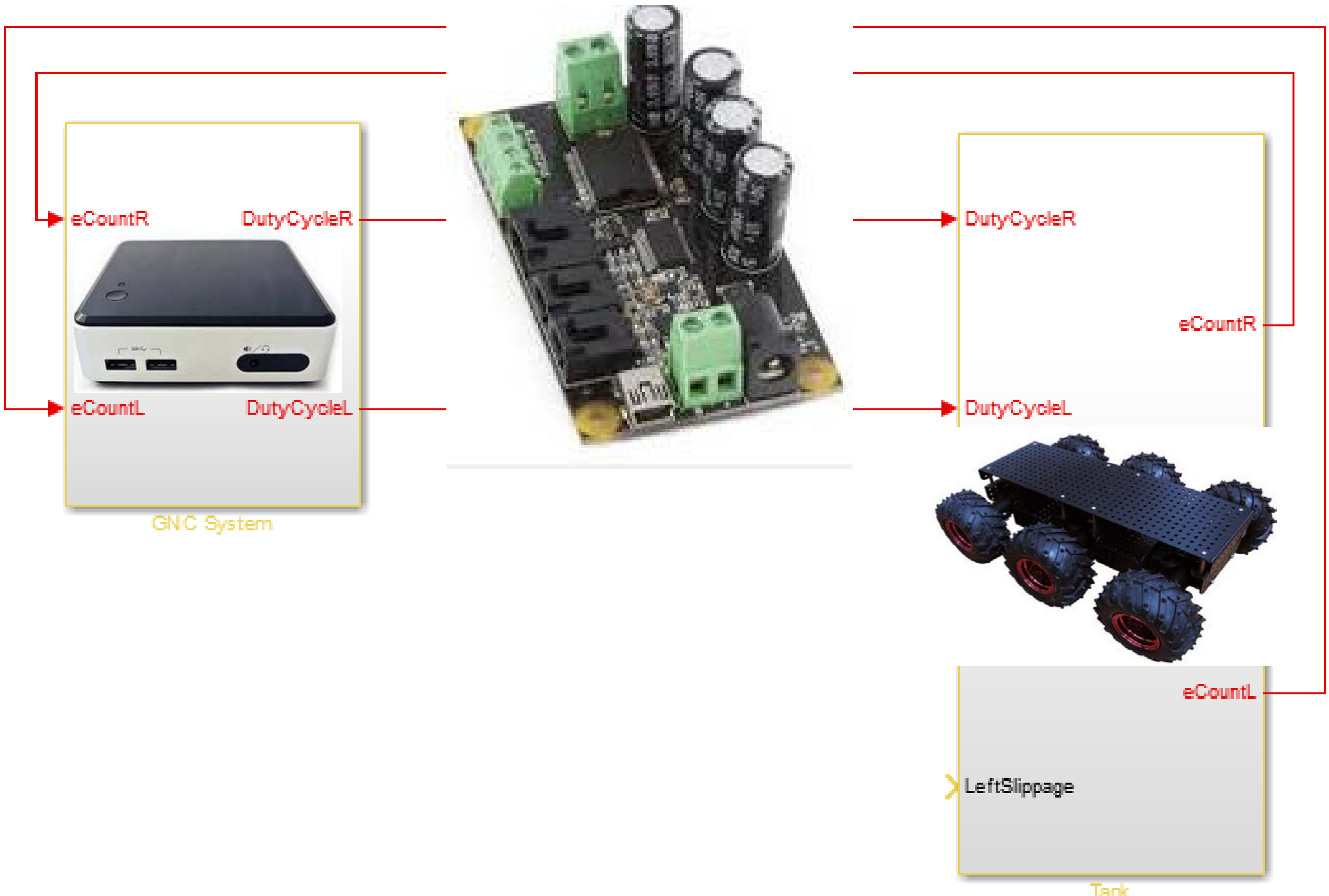
Where am I?
Where to go?
How to get there?



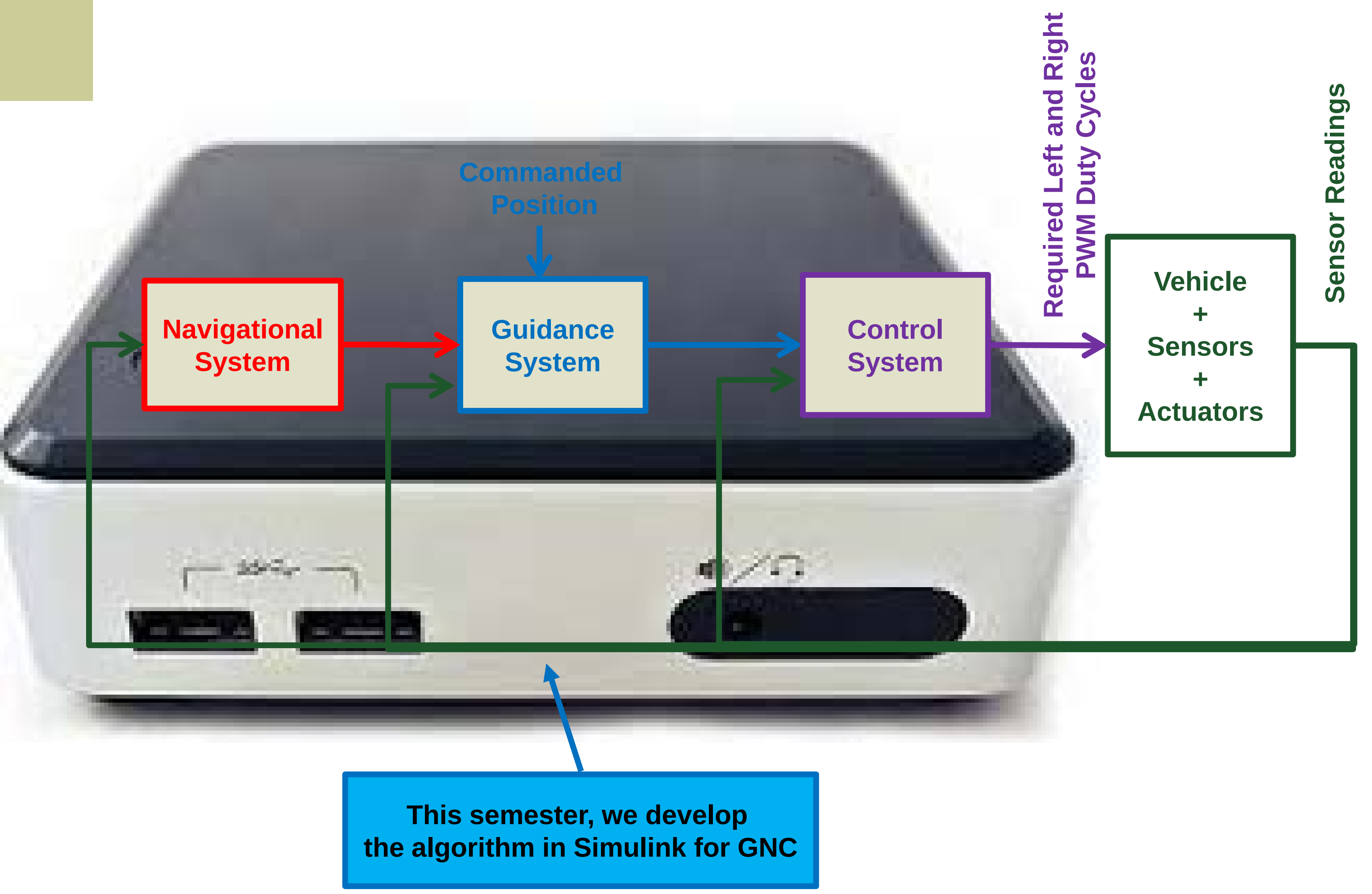
GNC of the UGV



Hardware Implementation (in Spring Class)



GNC of the UGV



Position and Orientation Estimation

- For wayPoint navigation
  - current position and orientation (angle) should be known
- Need sensors that measure position and orientation such as
  - GPS for position
  - Compass/Magnetometer for orientation
- The UGV operates indoor
  - GPS denied environment
  - Magnetometers not reliable indoor
- Encoder reading can be used to estimate position and orientation

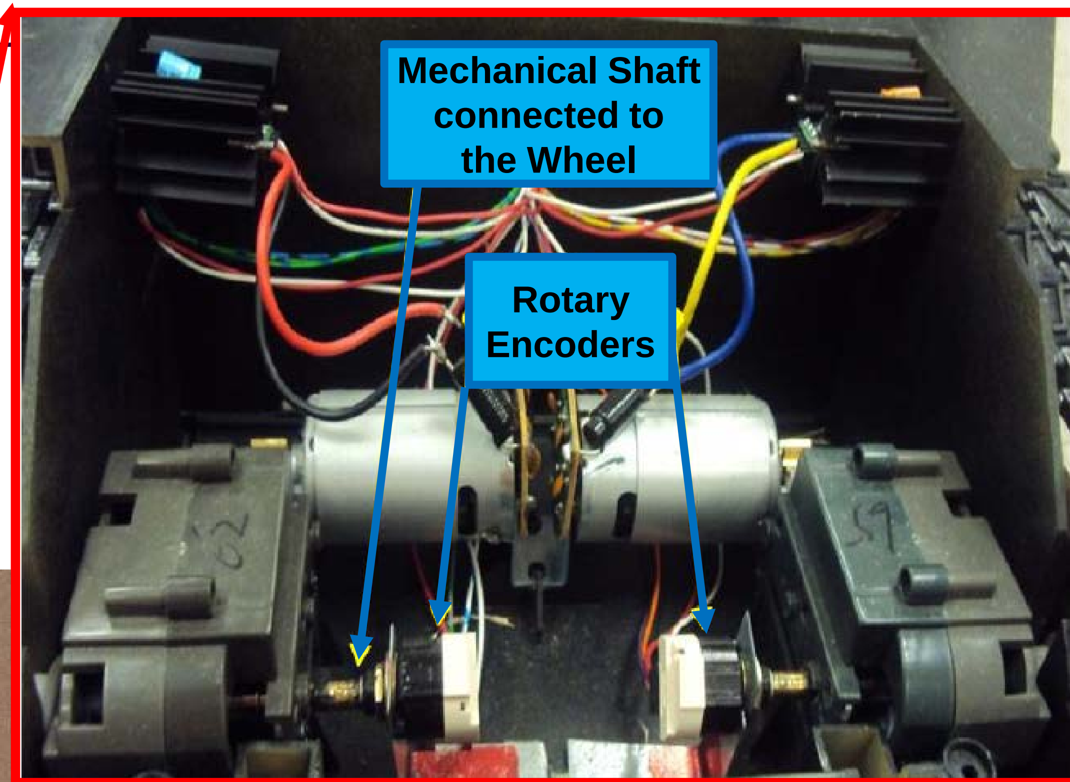
Dead Reckoning Method

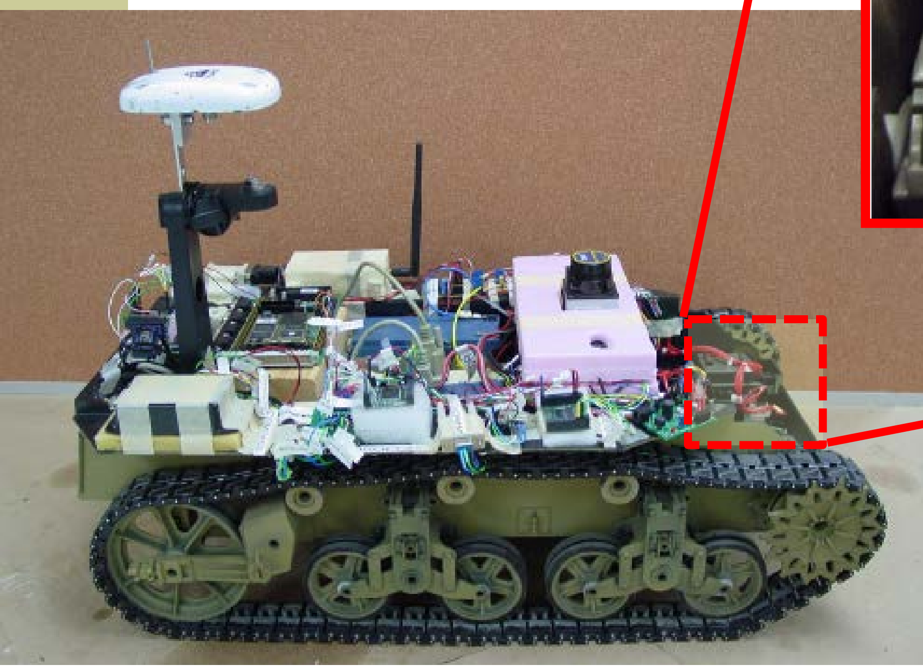
- In Navigation, it is the method of advancing position based on known or estimated speed over elapsed time*
- In general, it is the process of estimating the value of a variable by*
  - using an earlier value and
  - adding changes occurred in the meantime
- For the UGV,
  - Advancing position & orientation based on wheel speed estimates from encoder readings
- Dead reckoning is subject to cumulative errors*

* http://en.wikipedia.org/wiki/Dead_reckoning

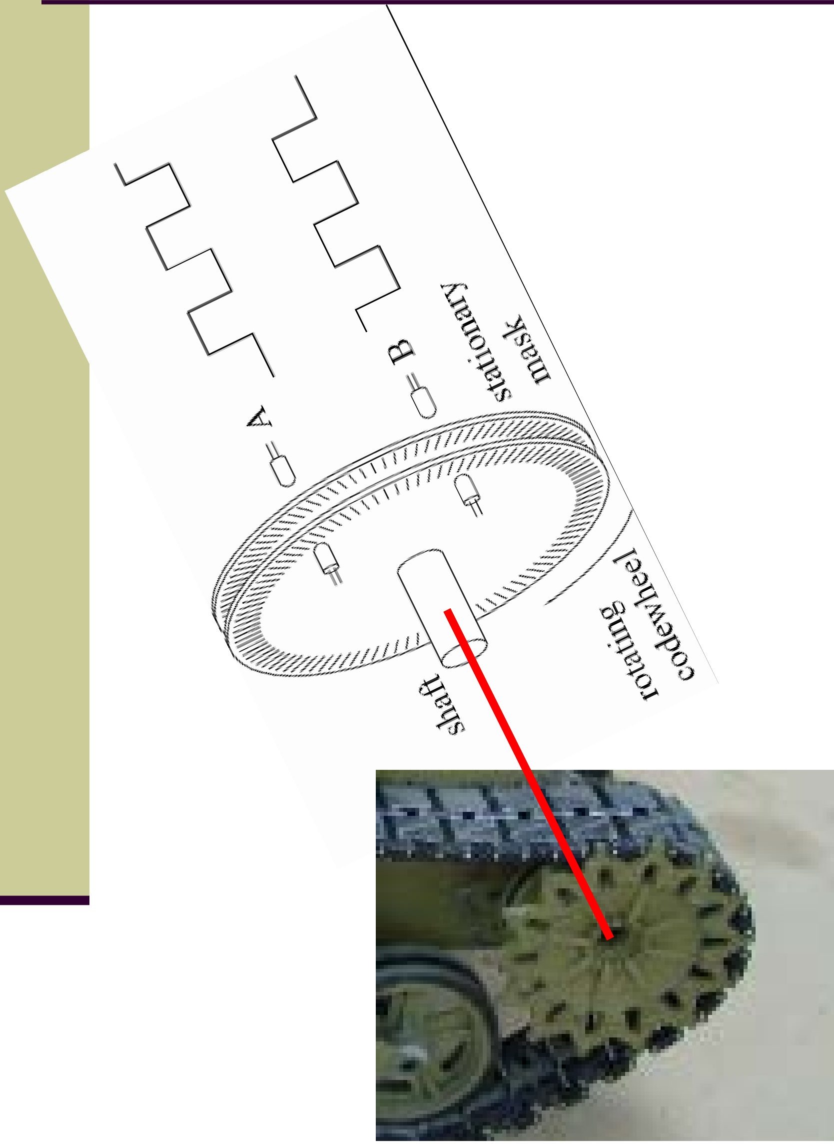
Encoders

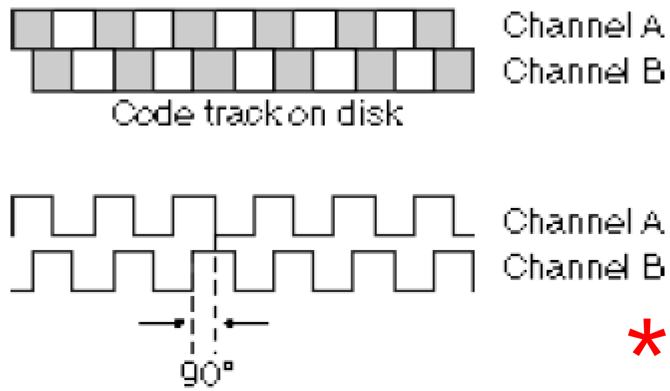
Encoder on each wheel
is to provide wheel speed





Encoders





• Using two code tracks with sectors positioned 90 degrees out of phase,

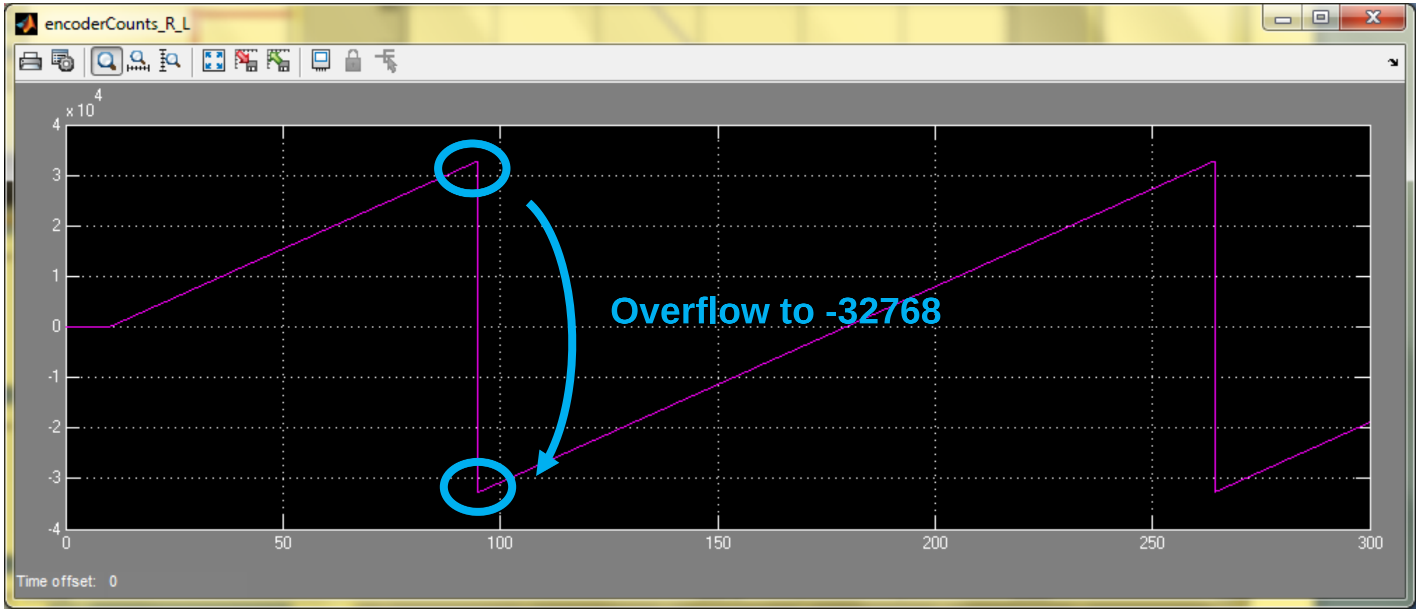
• Two output channels (A and B) of the quadrature encoder indicate
    • position
    • direction of rotation

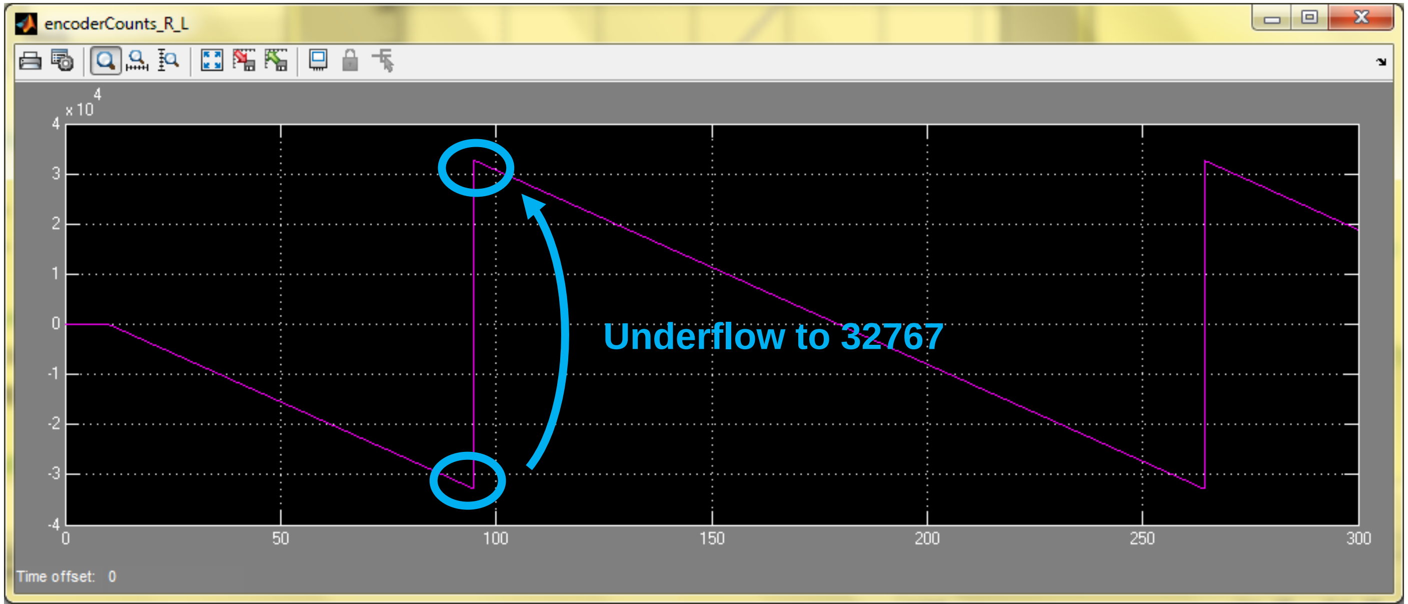
* http://www.ni.com/white-paper/4763/en/

Encoder Reading

- In the processor side hosting GNC (Guidance, Navigation and Control) code, encoder signals should be "decoded"
- For realtime implementation,
  - Position information from quadrature encoder hardware is decoded
  - The relative phase of a pair of input signals determines direction of movement.
  - The signals are decoded to increment or decrement the position counter
- In normal mode, the position counter is incremented or decremented for each valid transition on either channel.
  - The counter increments when the primary channel is ahead and decrements when the primary channel lags.
  - A switch in the phase relationship indicates a change of direction
  - The counter is "16 bit integer" and free flowing (that is, it overflows to -32768, and underflows to 32767)

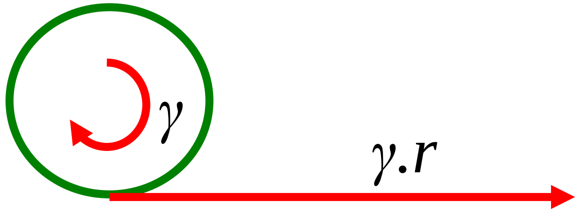
Encoder Reading





Encoder Reading from the UGV

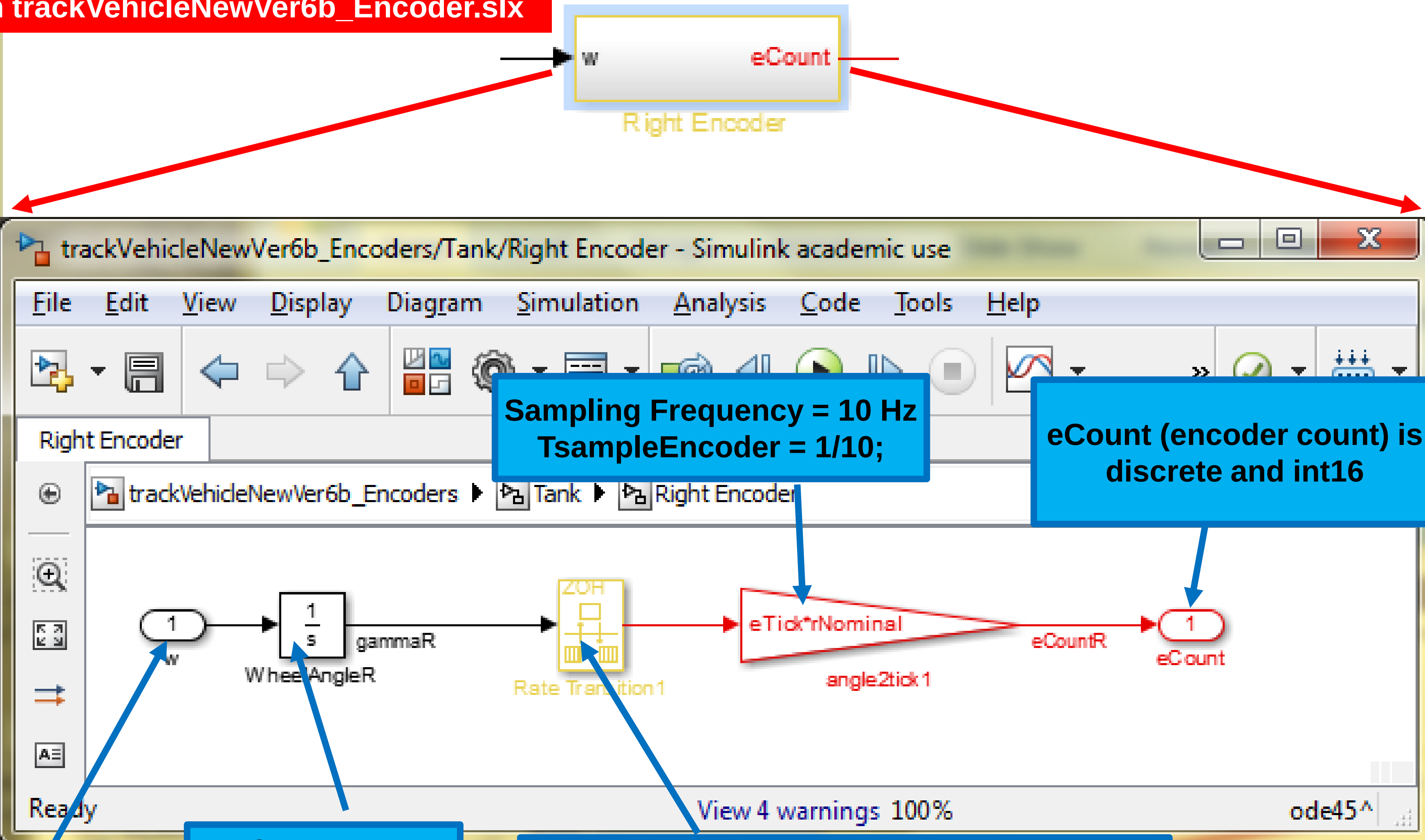
- Experiments revealed
  - encoder give 22-23 ticks per inch traveled by the track
  - This is about 900 ticks per meter traveled
  - *eTick* = 900 [1/m];
- With wheel radius of r = 0.052959 [m]
- With γ radian of wheel rotation,
  - the track travels $\gamma.r$ [m]



  - Encoder counter increments by $\gamma.r.eTick$

Encoder Modeling in Simulink

Open trackVehicleNewVer6b_Encoder.slx



Sampling Frequency = 10 Hz
TsampleEncoder = 1/10;

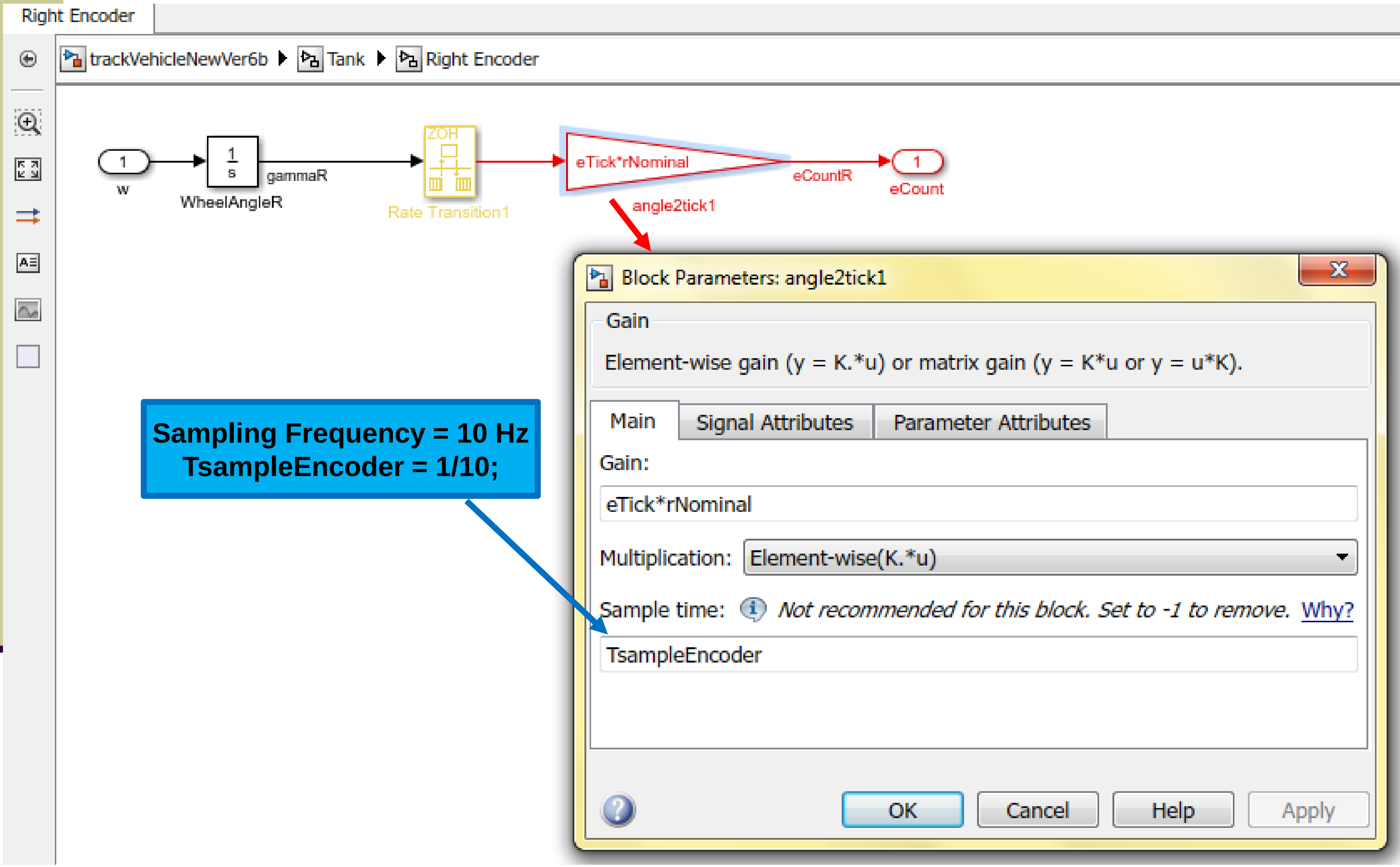
eCount (encoder count) is discrete and int16

Wheel angular speed

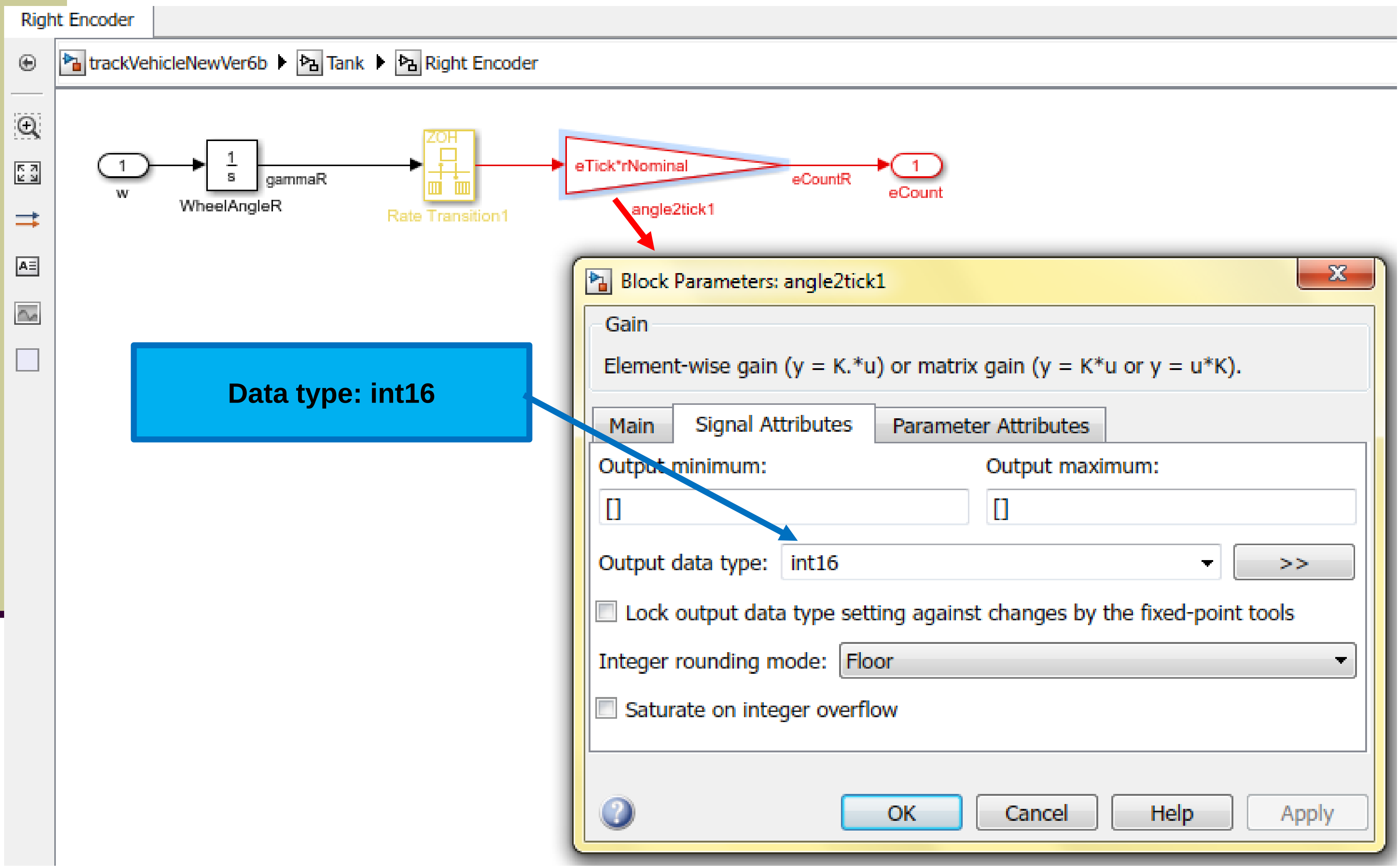
Integrate to compute the total rotation of the wheel

Rate Transition Because wheel rotation is analog (continuous) but encoder counts are discrete

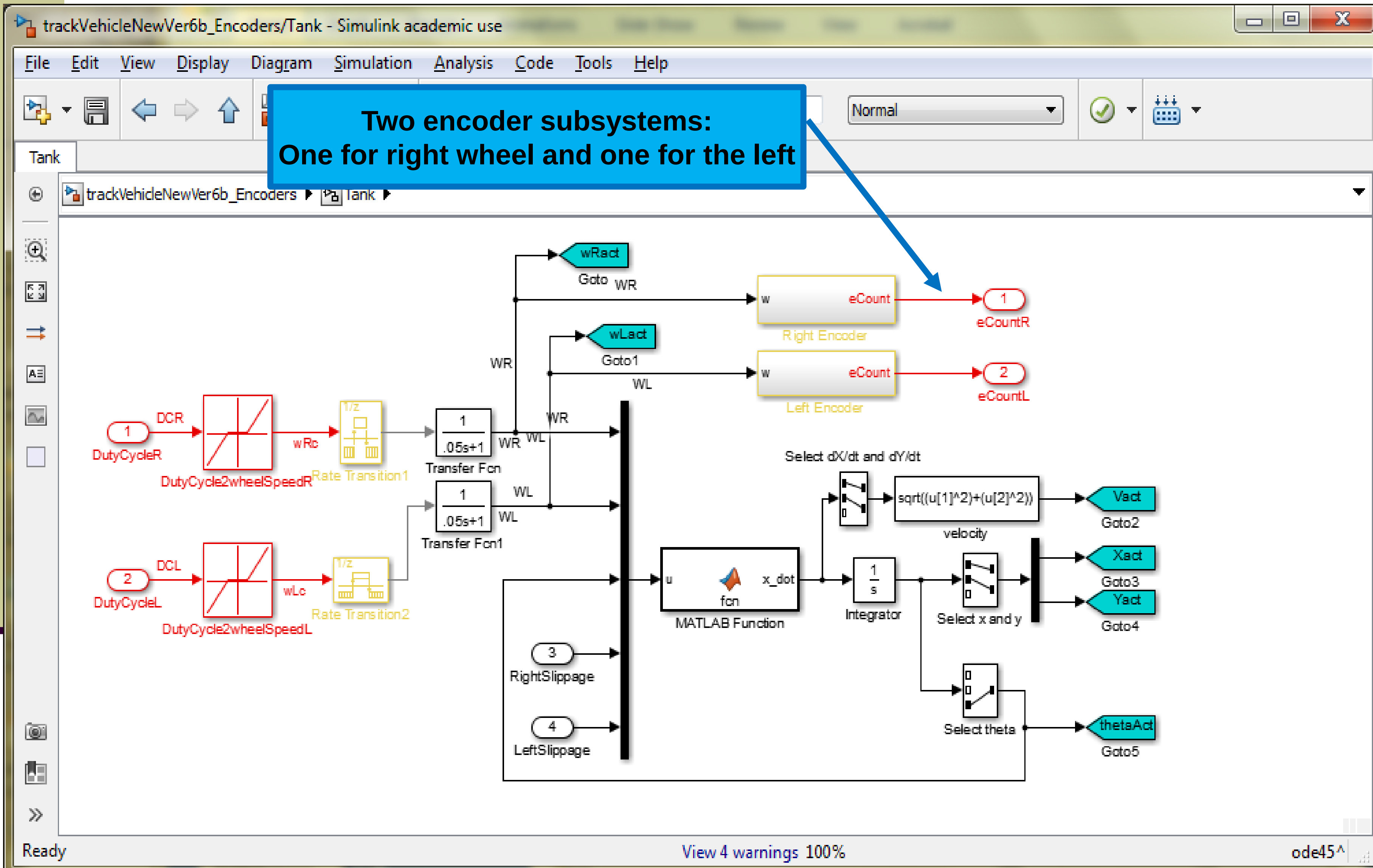
Encoder Modeling in Simulink



Encoder Modeling in Simulink



Encoder Modeling in Simulink

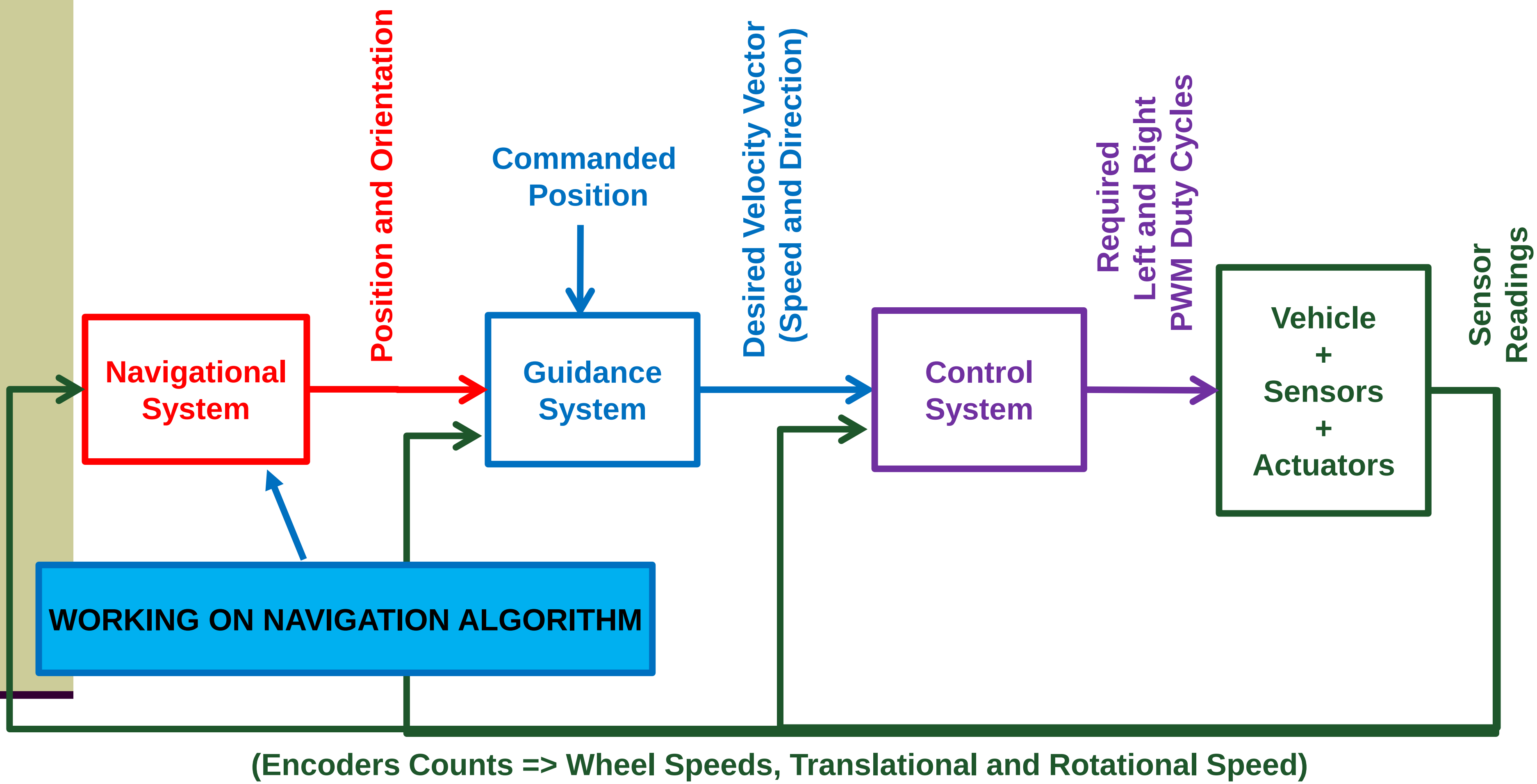


Example Runs for Encoder Counts

Open trackVehicleNewVer6b_Encoder.slx

- Run Simulation in various cases
- Show Right and Left Encoder Counts
  - Moves Forward
  - Moves Backward
  - Turns left or right

GNC of the UGV



Wheel Speed Calculation

- *eCount(k)*
  - encoder count at current sample time
- *eCount(k-1)*
  - encoder count at the previous sample time
- Wheel speed

$$\hat{\omega}(k) = \frac{\overbrace{eCount(k) - eCount(k-1)}^{\text{Number of ticks in one sampling period}}}{\underbrace{eTick}_{\text{Ticks per [meter]}}.\underbrace{r}_{\text{Wheel radius [meter]}}.\underbrace{TsampleEncoder}_{\text{Sampling period [sec]}}}$$

Wheel Speed Calculation

- Recall *eCount* (16 bit integer) overflow/underflow
  - $\left[eCount(k)-eCount(k-1)\right]$ will be very large!
  - Need to detect overflow and underflow
- When overflow/underflow, the counter jumps
  - From "32767 to -32768"
  - From "-32768 to 32767"

That corresponding to max speed would be enough

- When $\left|eCount(k)-eCount(k-1)\right|>32000$
  - UGV cannot move that fast
  - Overflow is detected!
  - Previous samples of the counter used

$$\left[eCount(k-1)-eCount(k-2)\right]$$

Two steps before

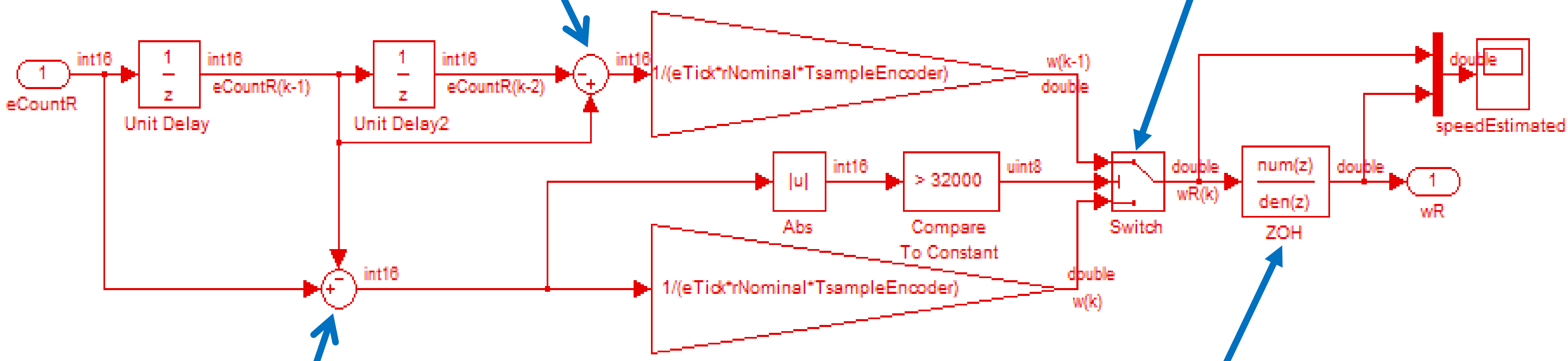
ASK STUDENTS TO BUILD WHEEL SPEED ESTIMATION BLOCK FROM ENCODER COUNT

Wheel Speed Calculation in Simulink

Open trackVehicleNewVer6b_Encoders_DR.slx

$$\left[eCount(k-1) - eCount(k-2)\right]$$

Switches to upper path only when overflow/underflow



Discrete first order filter to smooth the calculation

$$\left[eCount(k) - eCount(k-1)\right]$$

- This is for right wheel
- The same blocks repeated for the left wheel

Wheel Speed Calculation in Simulink

Open trackVehicleNewVer6b_Encoders_DR.slx

- Run Simulation in various cases
- Show Right and Left Encoder Counts
- Show wheel speed calculated versus actual
  - Moves Forward
  - Moves Backward
  - Turns left or right

UGV Speed & Orientation Estimation

- Recall Kinematics Equations of UGV

$$\dot{x} = \frac{1}{2}[(1-s_L)r_L\omega_L + (1-s_R)r_R\omega_R]\cos\theta$$
$$\dot{y} = \frac{1}{2}[(1-s_L)r_L\omega_L + (1-s_R)r_R\omega_R]\sin\theta$$
$$\dot{\theta} = \frac{1}{b}[(1-s_L)r_L\omega_L - (1-s_R)r_R\omega_R]$$

- Assuming no slippage and $r_L = r_R = r$

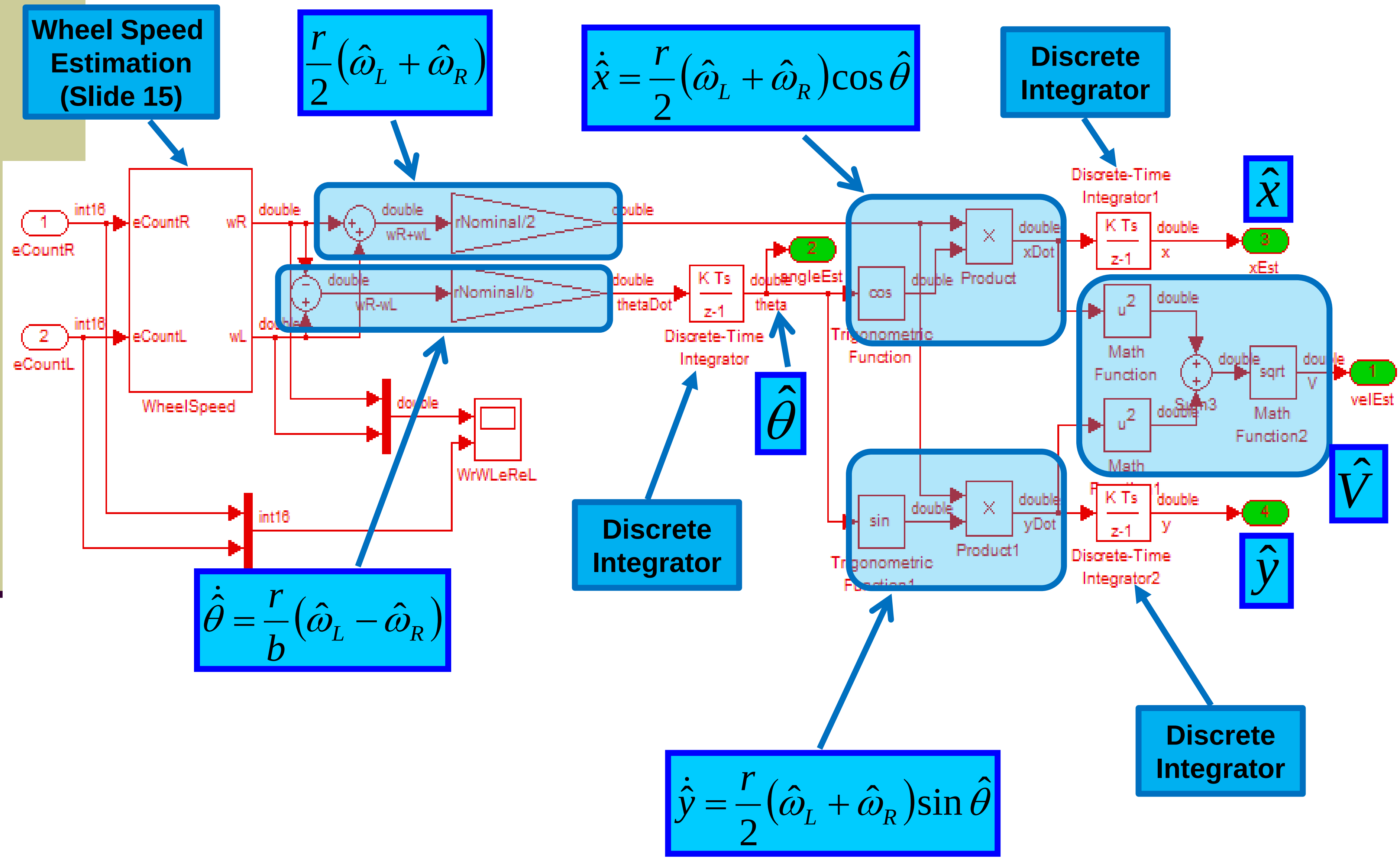
Estimated translational and angular speeds

Estimated wheel speeds

Continuous integration of those will give estimated position and orientation

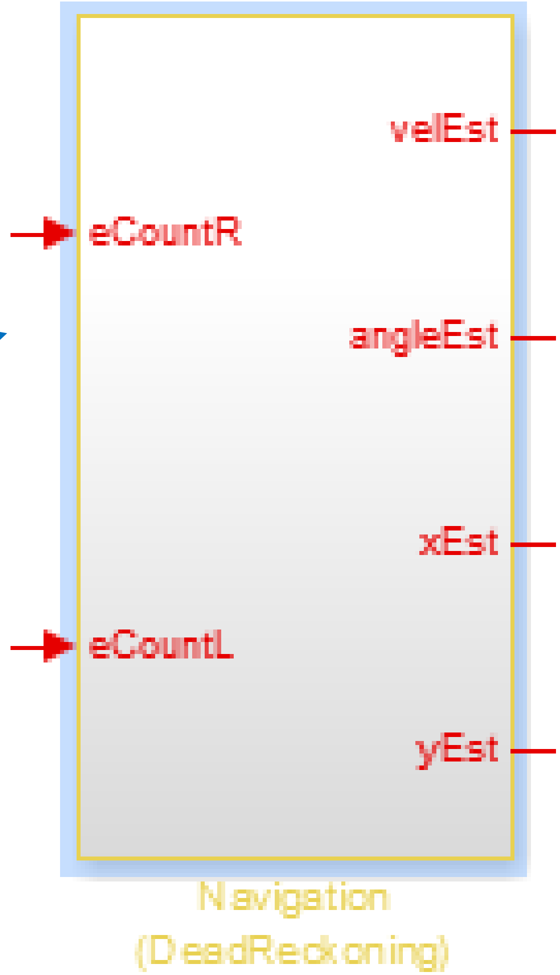
$$\left\{\begin{aligned} \dot{\hat{x}} &= \frac{r}{2}(\hat{\omega}_L + \hat{\omega}_R)\cos\hat{\theta} \\ \dot{\hat{y}} &= \frac{r}{2}(\hat{\omega}_L + \hat{\omega}_R)\sin\hat{\theta} \\ \dot{\hat{\theta}} &= \frac{r}{b}(\hat{\omega}_L - \hat{\omega}_R) \end{aligned}\right.$$

Dead Reckoning in Simulink



Navigation Subsystem

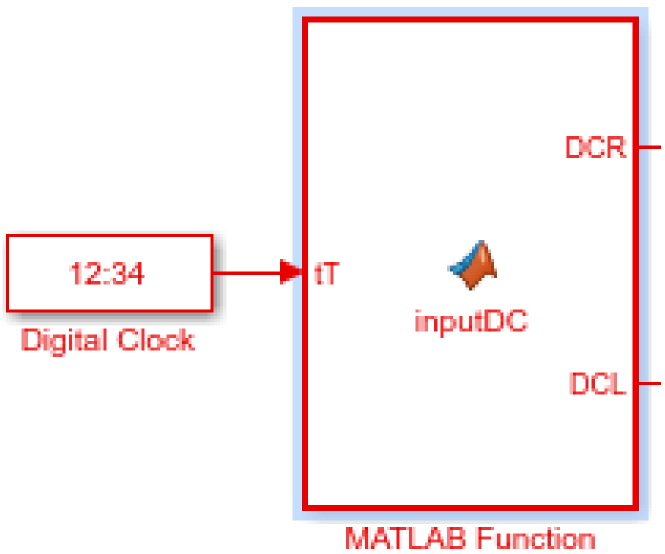
Takes Left and Right Encoder Readings



Computes
(1) Speed
(2) Orientation
(3) Position

Simulation Results (No slippage)

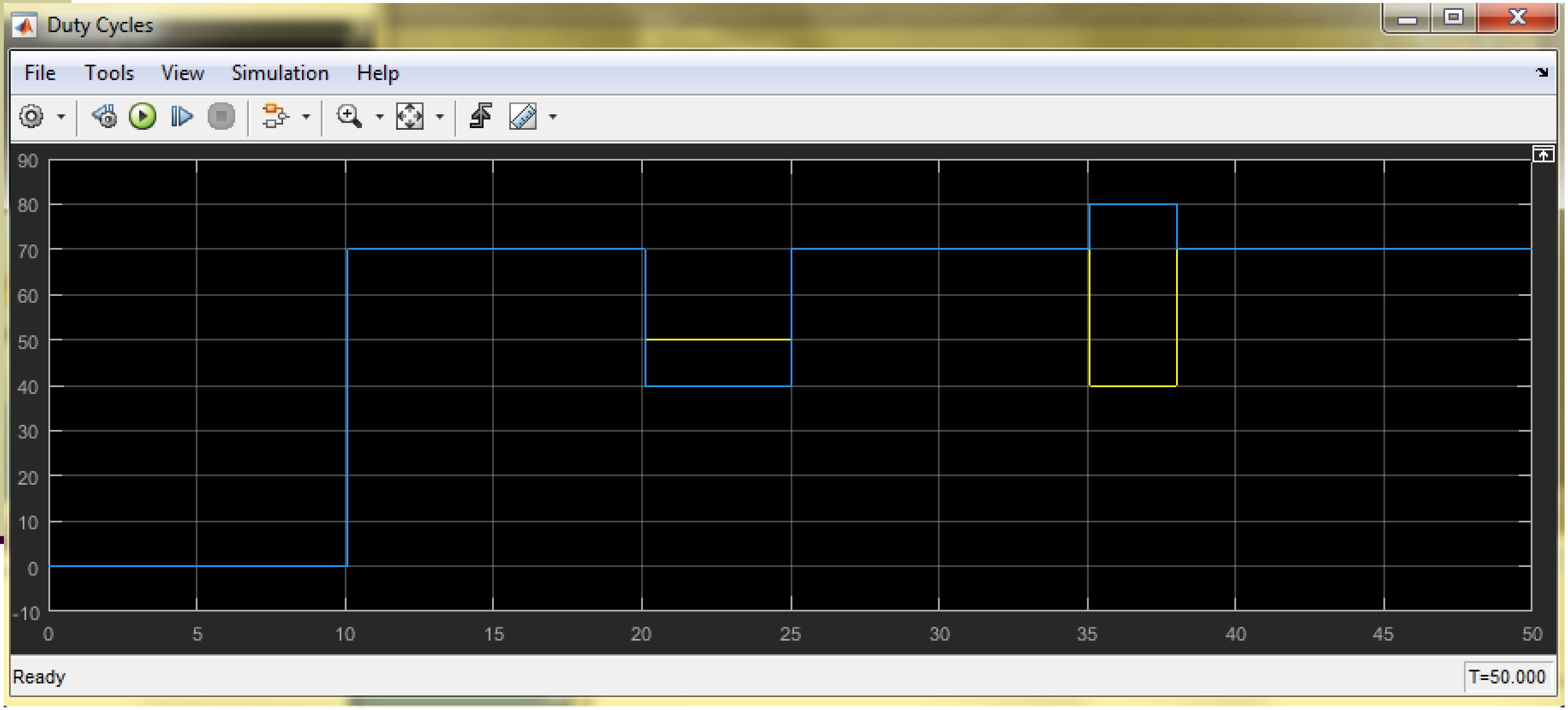
Open trackVehicleNewVer6b_Encoders_DR.slx




```
if tT > 10
    DCR=70;
    DCL=70;
end

if tT > 20 && tT < 25
    DCR = 50;
    DCL = 40;
end

if tT > 35 && tT < 38
    DCR = 40;
    DCL = 80;
end
```

Simulation Results (No slippage)

- PWM %DutyCycle Schedule

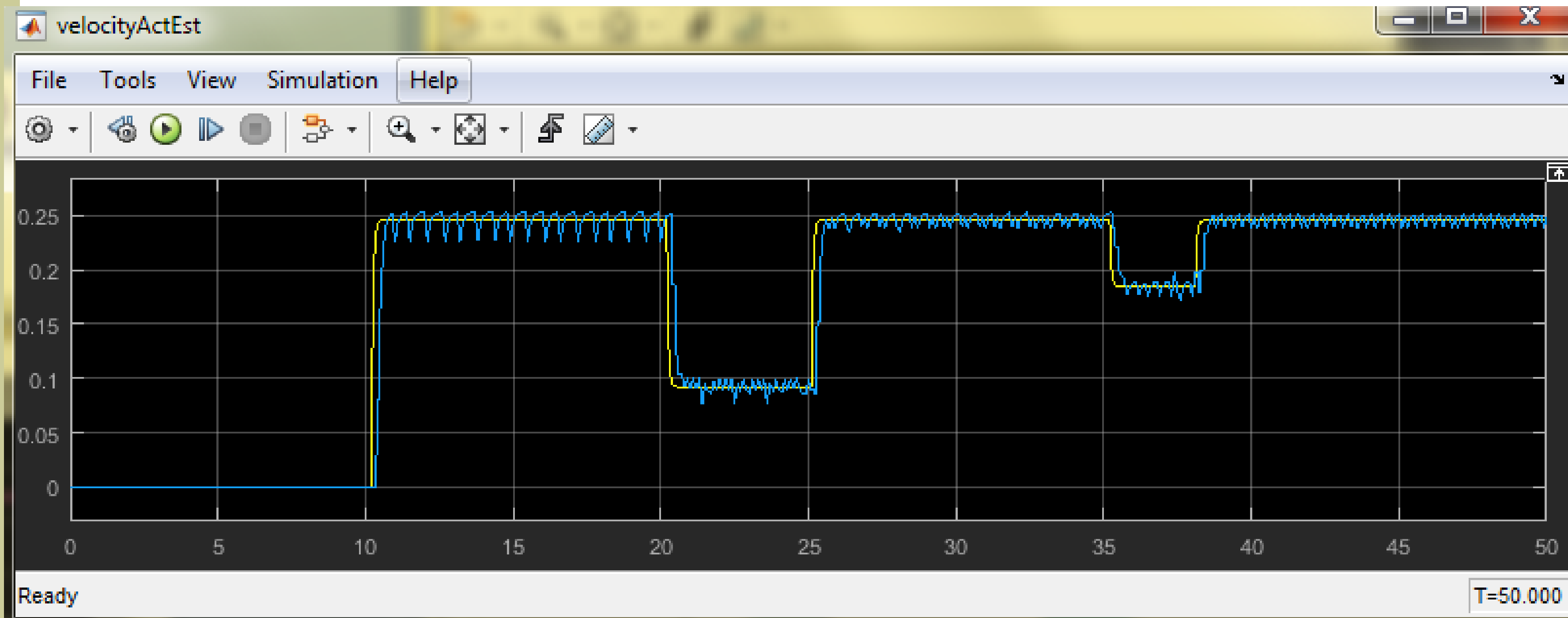


Simulation Results (No slippage)

- Estimated Left & Right Wheel Speeds based on encoder counts

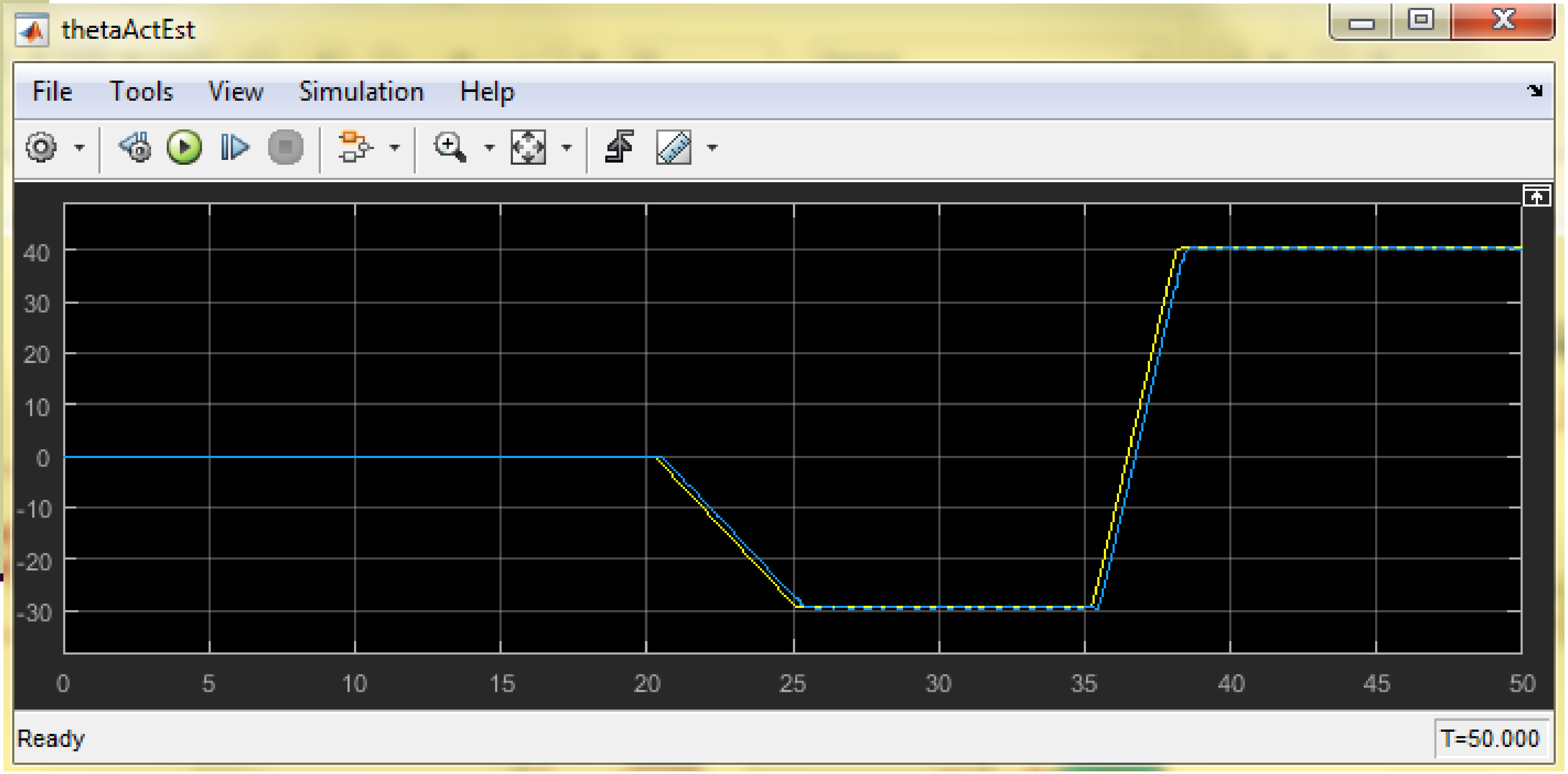
Simulation Results (No slippage)

- Actual vs Estimated Speed



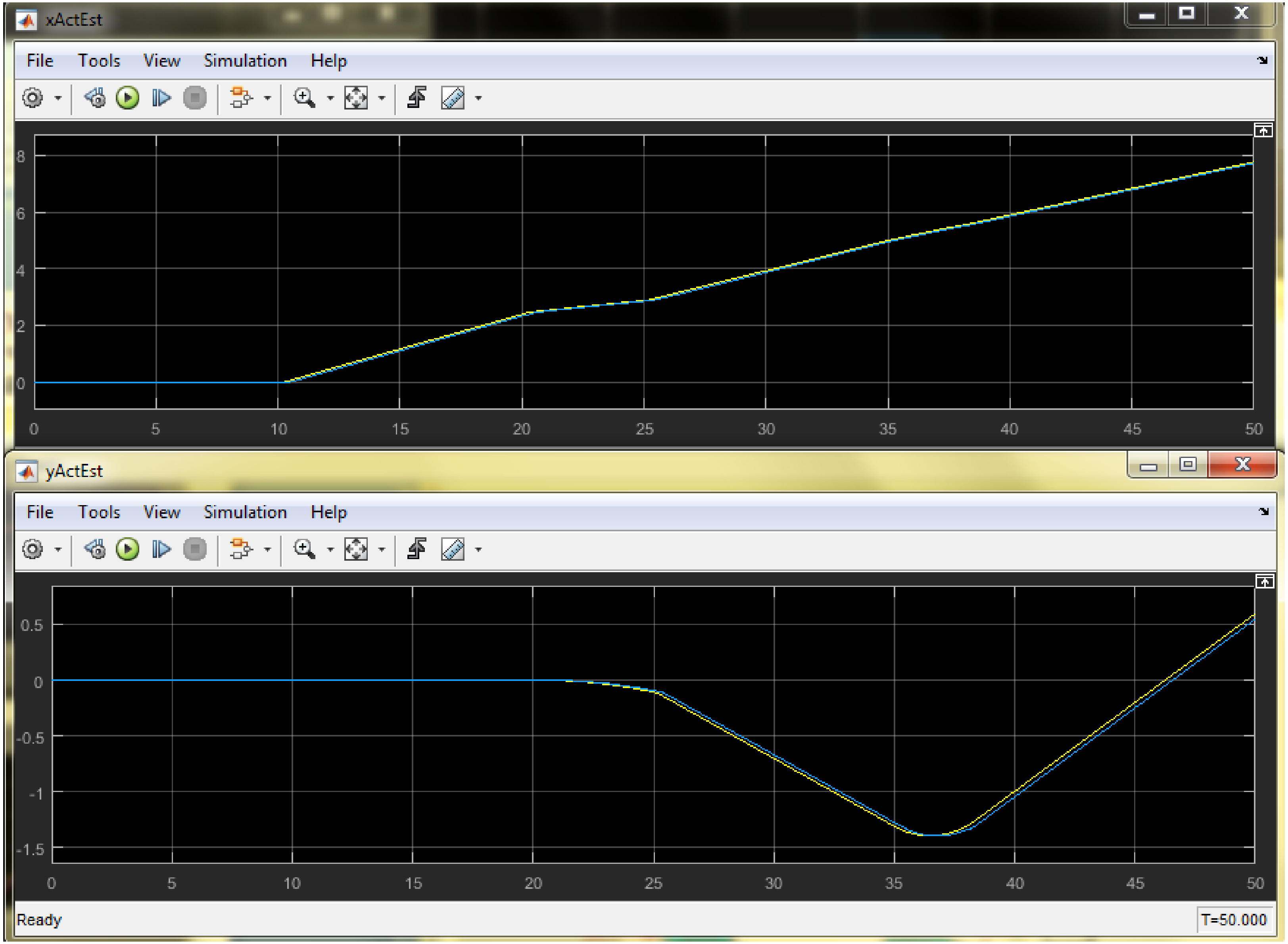
Simulation Results (No slippage)

- Actual vs Estimated Orientation

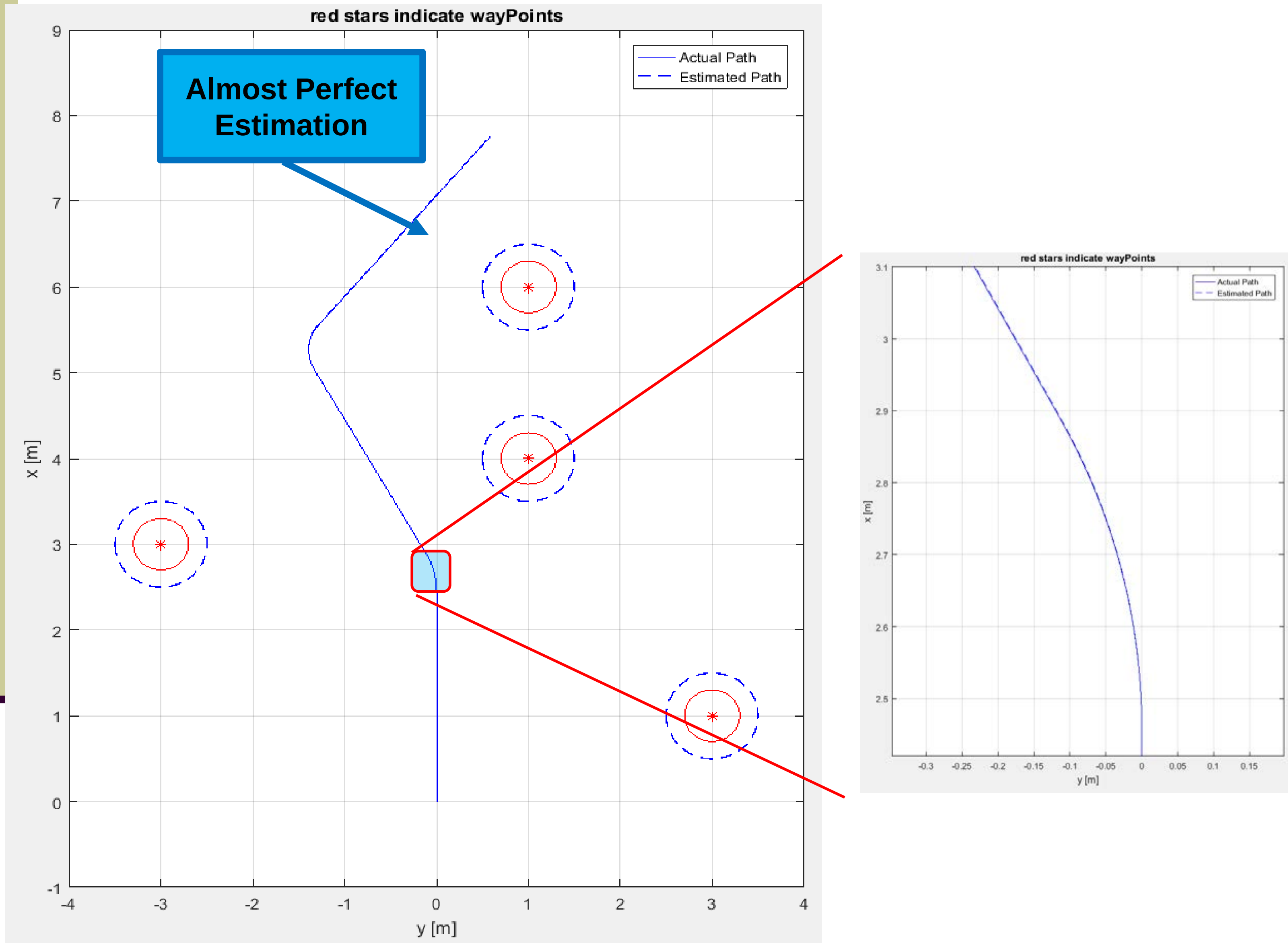


Simulation Results (No slippage)

- Actual vs Estimated x and y Coordinates



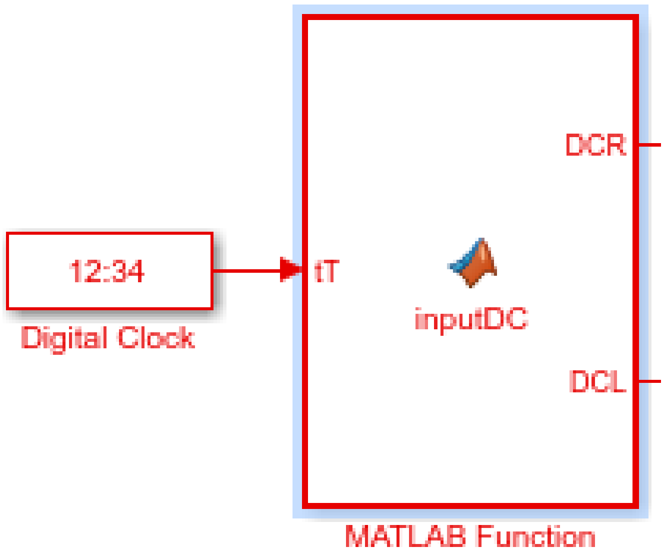
Simulation Results (No slippage)



Simulation Results (With Slippage)

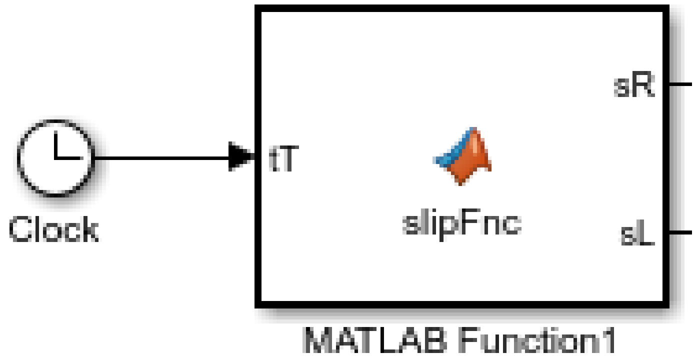
Open trackVehicleNewVer6b_Encoders_DR.slx

SAME AS BEFORE




```
if tT > 10
    DCR=70;
    DCL=70;
end

if tT > 20 && tT < 25
    DCR = 50;
    DCL = 40;
end

if tT > 35 && tT < 38
    DCR = 40;
    DCL = 80;
end
```

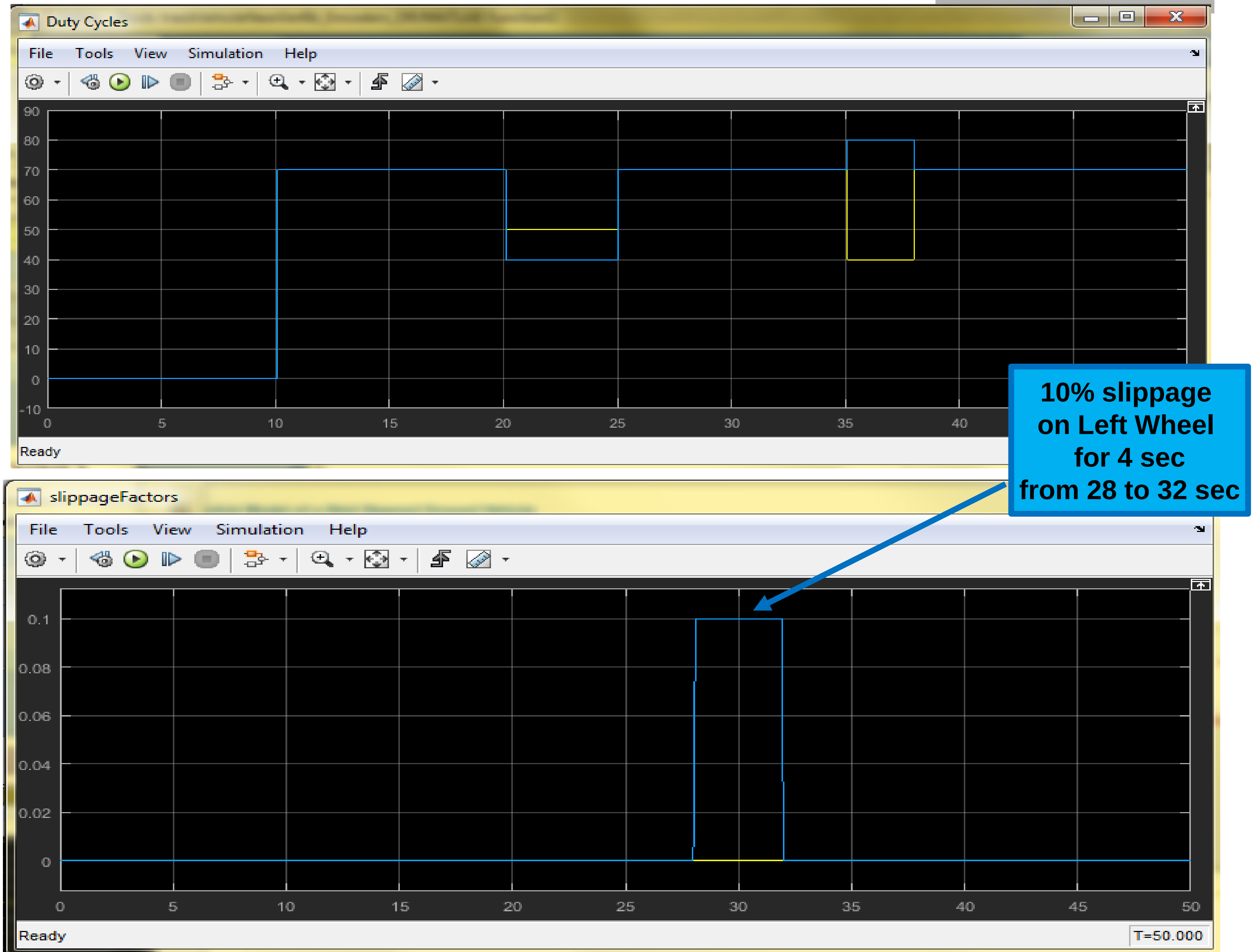
Simulation Results (With Slippage)



10% slippage on Left Wheel for 4 sec from 28 to 32 sec

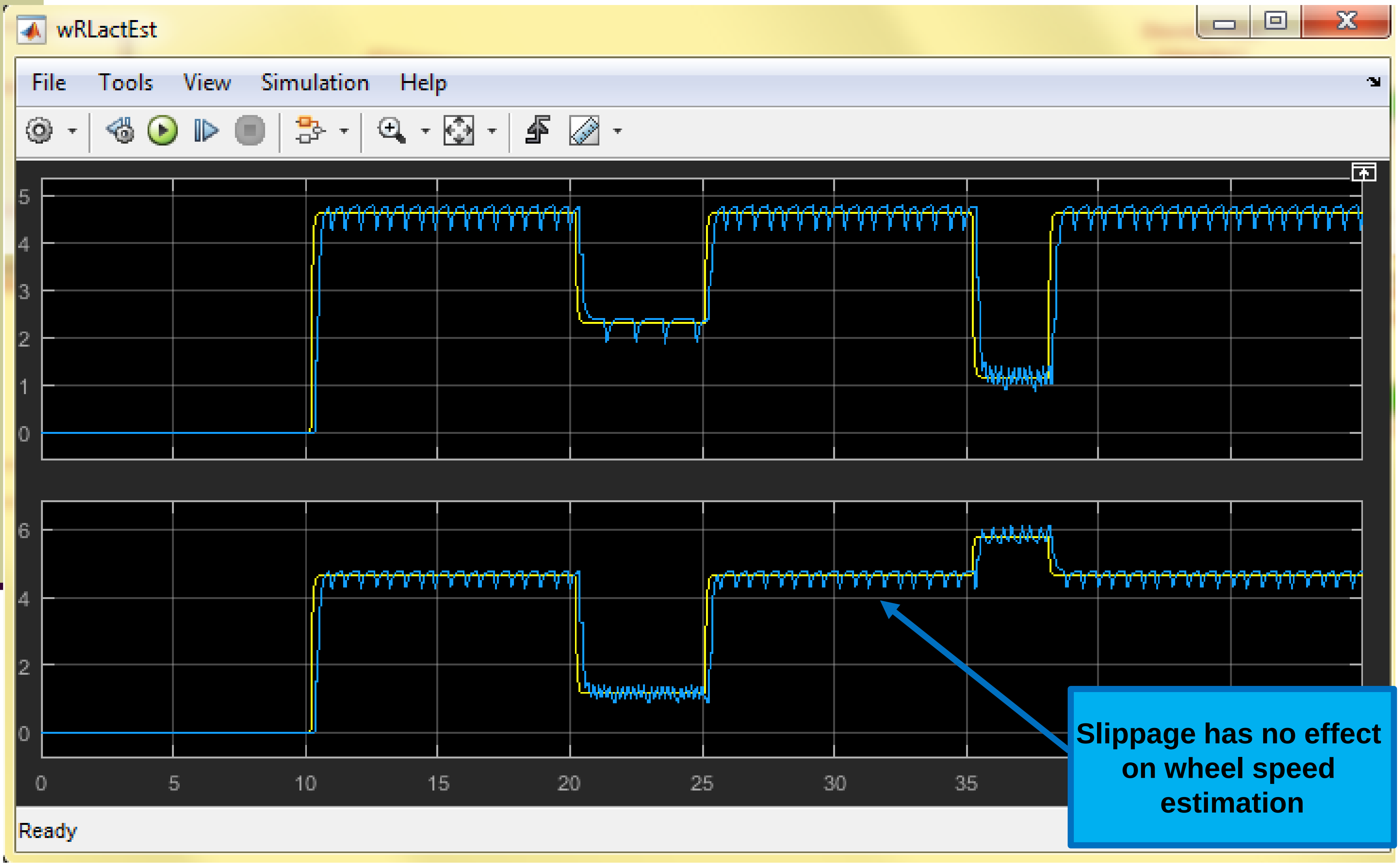
```
if tT > 28 && tT < 32
%       sR=0.2;
        sL=0.1;
end
```

Simulation Results (With Slippage)



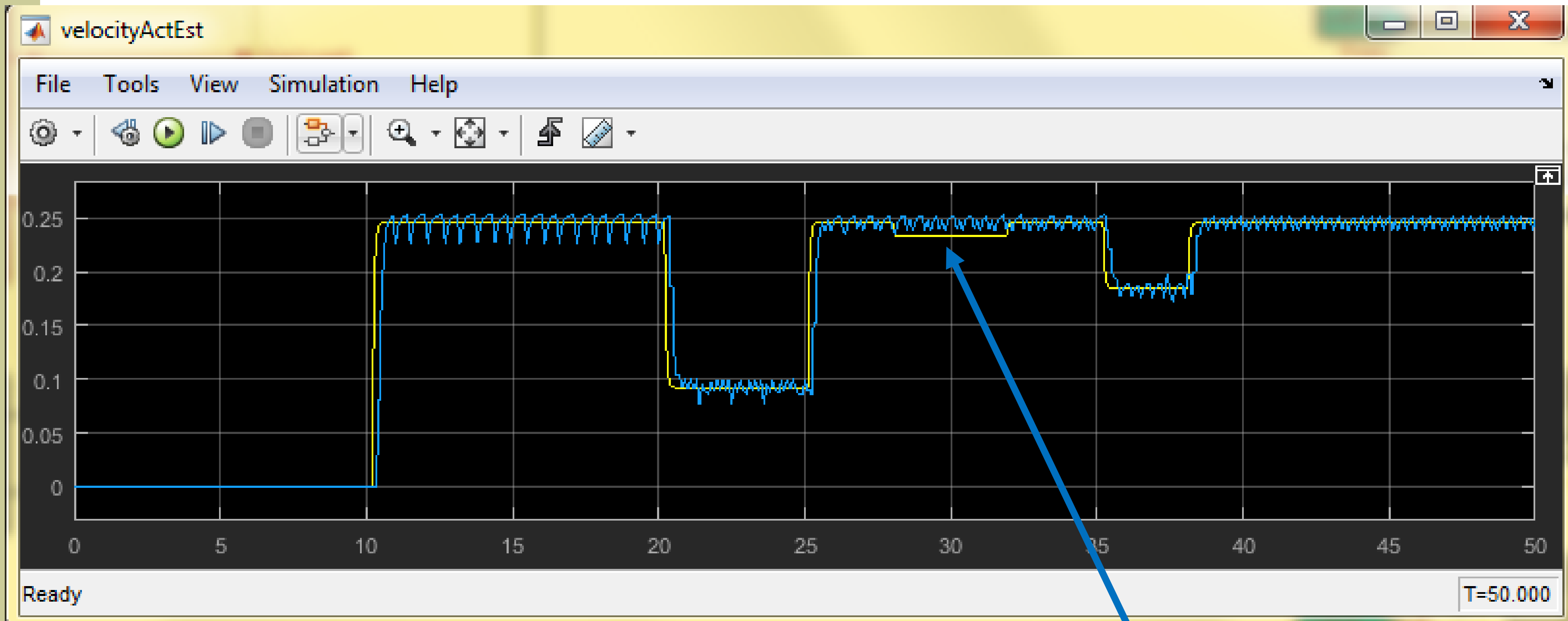
Simulation Results (With Slippage)

- Actual vs Estimated Wheel Speeds



Simulation Results (With Slippage)

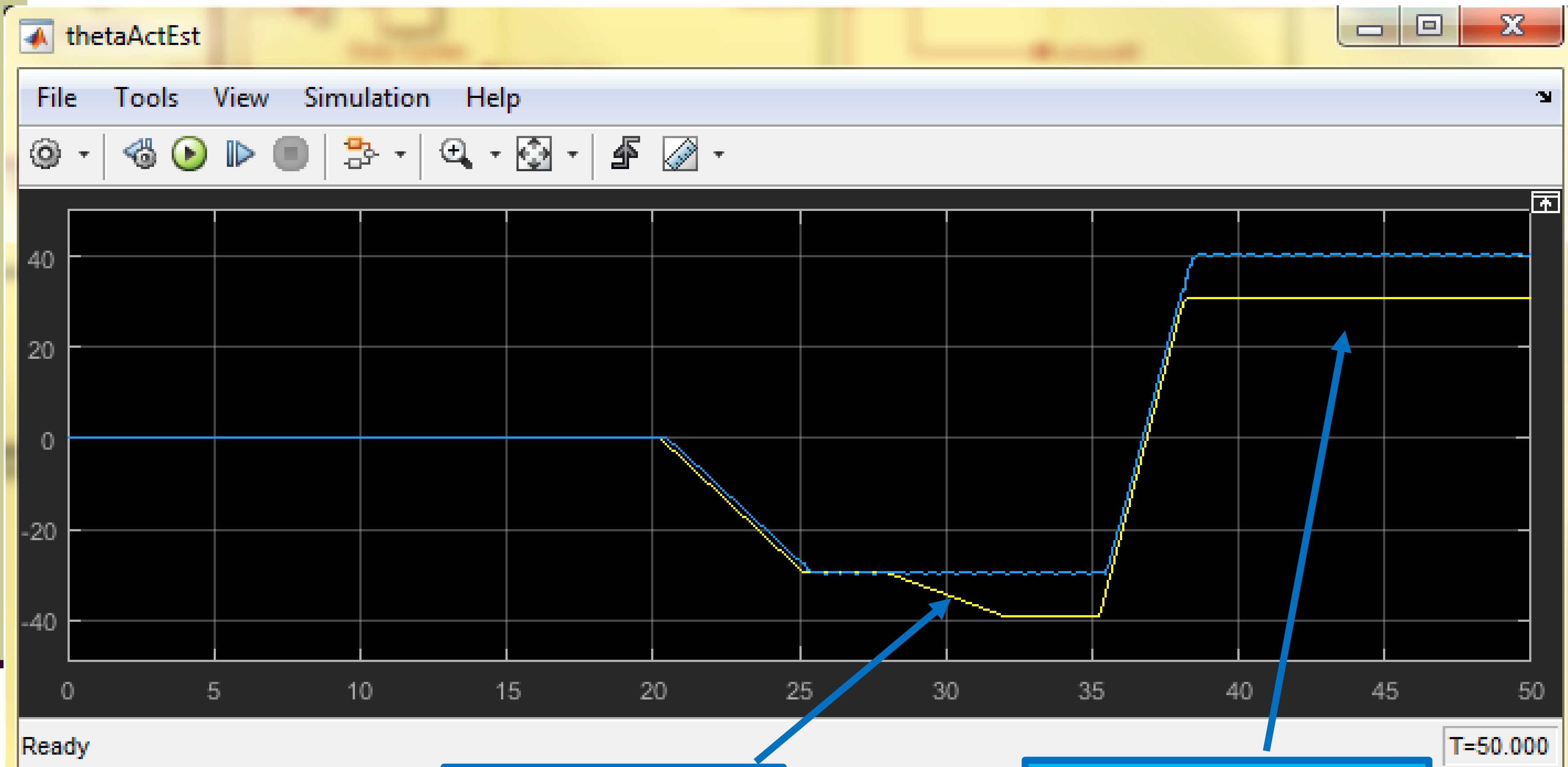
- Actual vs Estimated Speed



Estimated Speed has error during the slippage

Simulation Results (With Slippage)

- Actual vs Estimated Angle

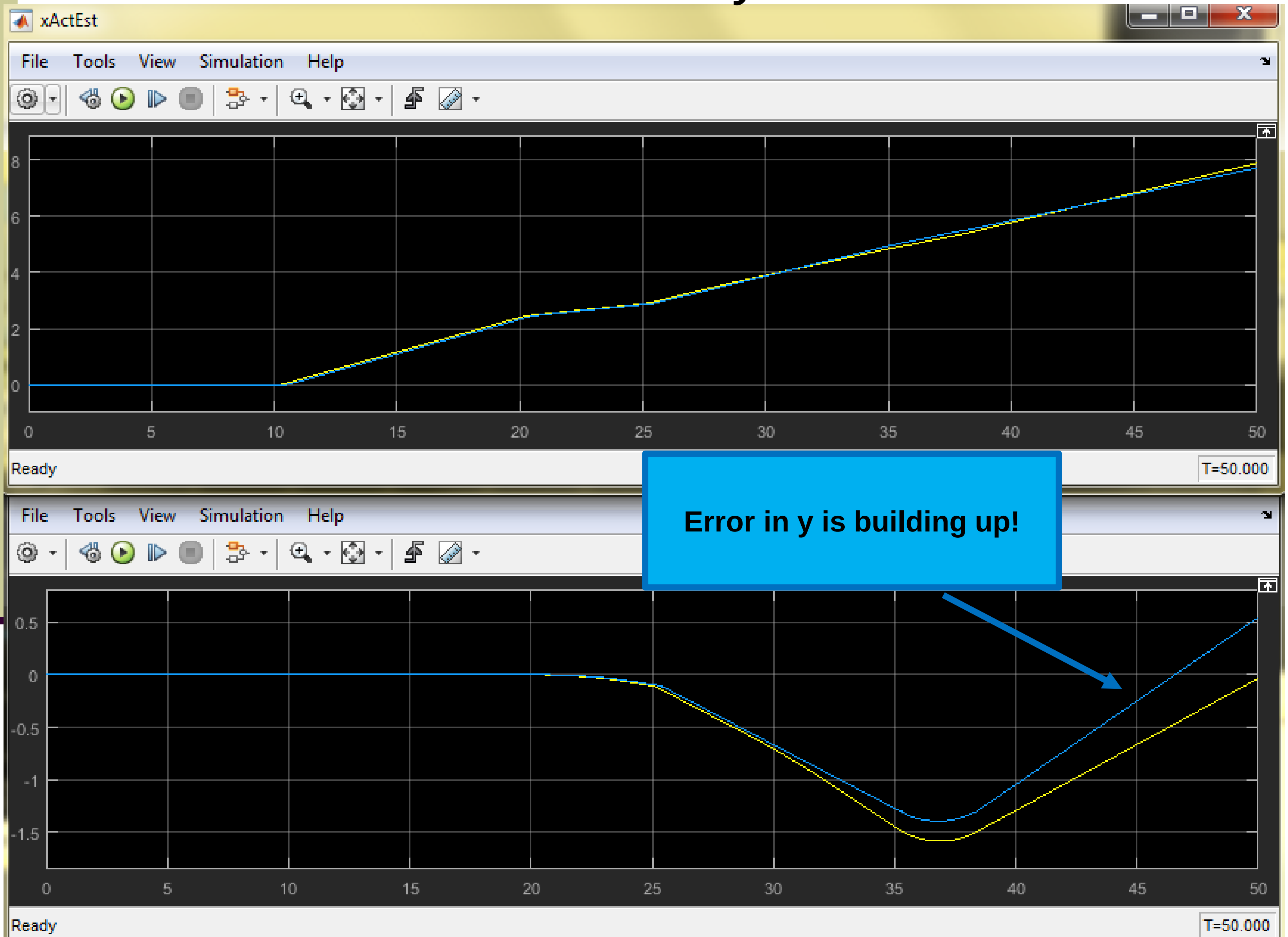


Estimated Angle has increasing error during the slippage

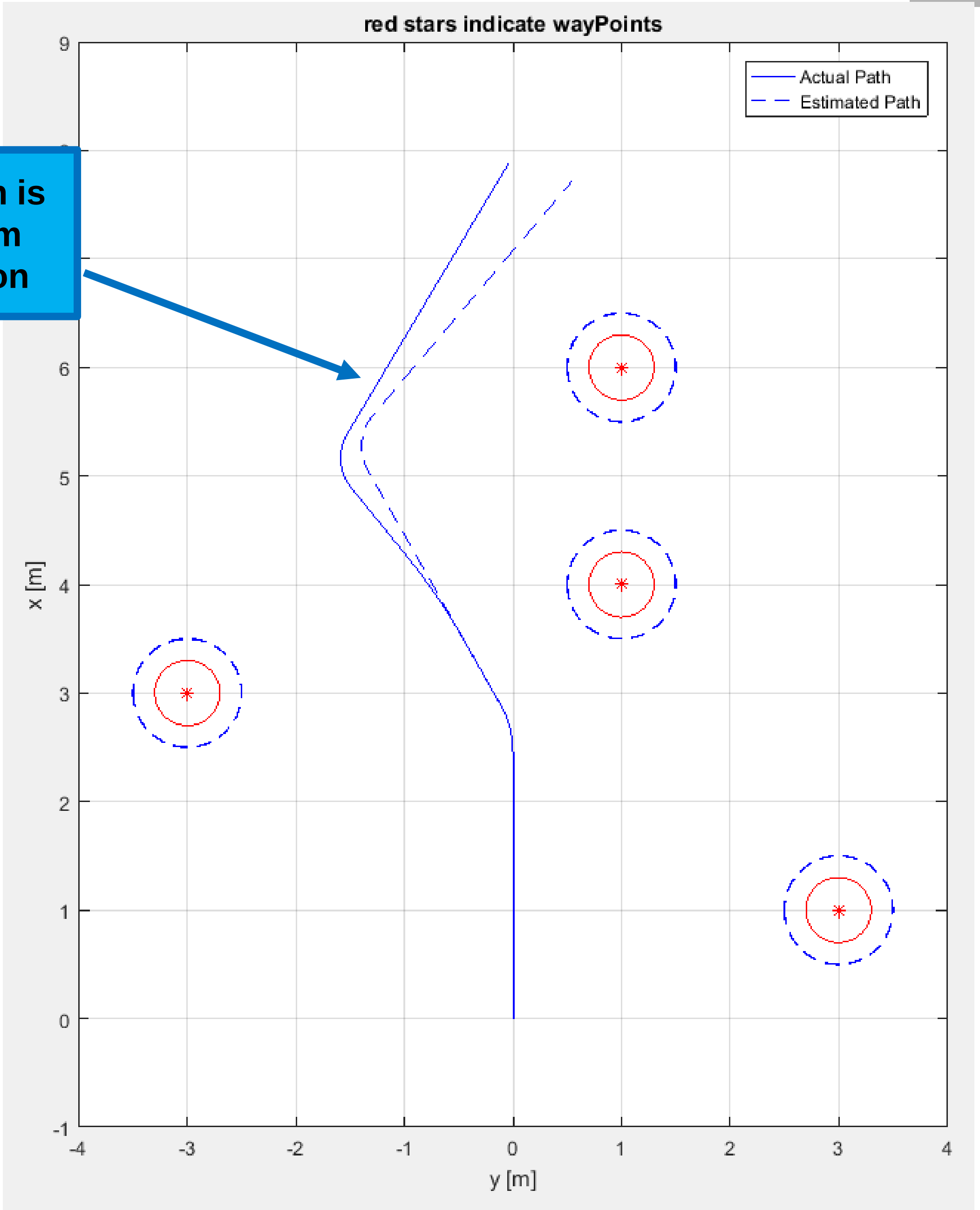
Error in angle stays even after the slippage

Simulation Results (With Slippage)

- Actual vs Estimated x & y coordinates



Simulation Results (With Slippage)



Estimated position is drifting away from the actual position

Simulation Results (Model Error)

parameterVehicle.m

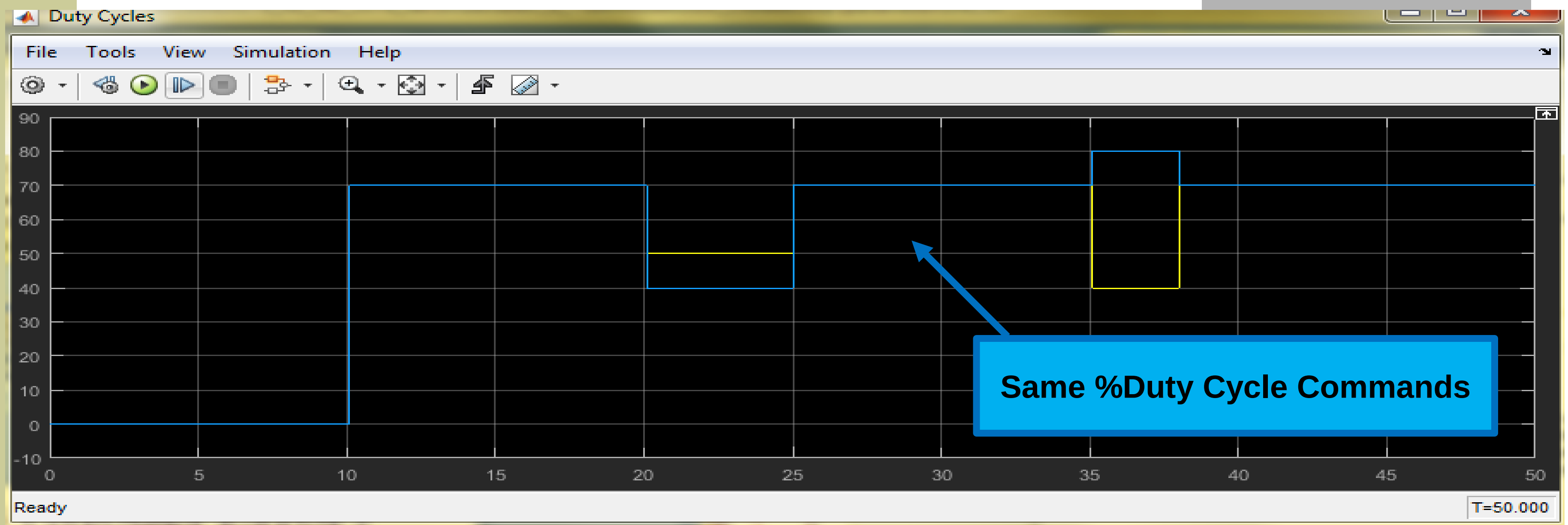
```
%%%%%%%%%%%%%%% Track Vehicle Parameter %%%%%
b = 0.5842 ; % [m] Effective Platform Width =
             % Actual Width = 0.3556 m
rNominal = 0.052959; % [m] Nominal Wheel Radi

Vmax = 0.43; %132/866.1417; % [m/s] Maximum s
% wMax = Vmax/rNominal; % [rad/s] Maximum ang

rr = 1*rNominal; %% Effective vehicle right w
rl = 1*rNominal; %%  1%% Effective vehicle le

%%%%%%%%%%%%%%% Encoder Parameter %%%%%%%%%%%
eTick = 237;%236.8852;%900; % 866.1417-905.51
TsampleEncoder = 1/10; %1/100; % 0.1 [s] Enco
%%
%%%%%%%%%%%%%%% Duty Cycle -> Speed - Convers
dcArray = [-100  -30 0  30  100];
wArray  = [-Vmax, 0, 0,  0, Vmax]./rNominal;

%%%%%%%%%%%%%%%%%% Initial Conditions %%%%%%%
xIC = 0;
yIC = 0;
thetaIC = 0*(pi/180);
```

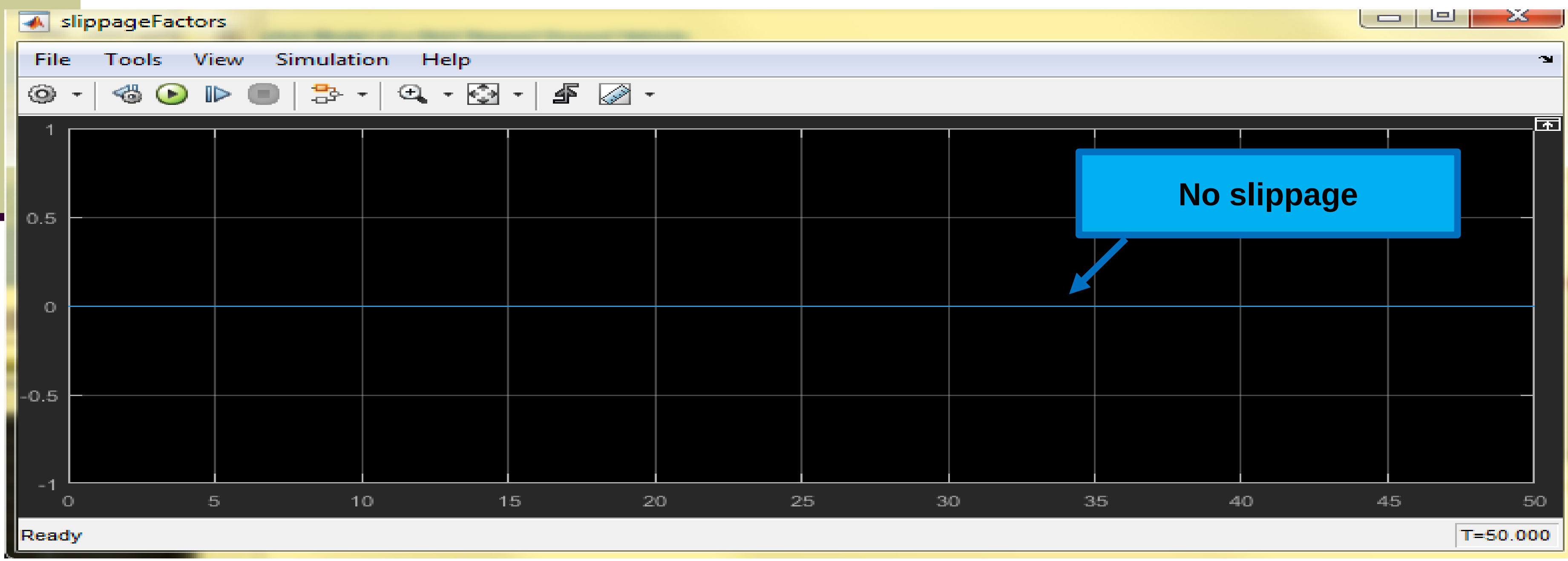
parameterGNC.m

```
% Navigation Parameters
% if we want the estimation parameters ide
VmaxDR = Vmax;
rNominalDR = rNominal;
bDR = 0.85*b;%b;%
eTickDR = eTick;
TsampleEncoderDR = TsampleEncoder;
TauEncoderDR = 0.1;
%%

%%%%%%%%%%%%%%%%%%% control parameter %%%
% 1:speed, 2:angle
KP1=10; %20;     % 1 is velocity controlle
KP2=3;%10;  %7 ->   % 2 is angle controlle
KI1=.5;%10;%0.3; %0.001; %0.001;%10;
KI2=0.001; %0.001; %0.001;%1; % 3 ->
KD1=0; %3; %0; %3; %0;%3;
KD2=0;% 3;%  %.5; %0; %0.5; %0;%0.5; %0.5
Tsample = 1/10; %1/100; %1/10;   %samplin
Tmodel=Tsample;
Tau1 = 0.01;     %time constant of filter
Tau2 = 0.1;%0.1;       %time constant of

% wheel speed - DC conversion
wArrayC  = [-VmaxDR, -VmaxDR/100, 0,  Vma
dcArrayC = [-100  -30 0  30  100];
```

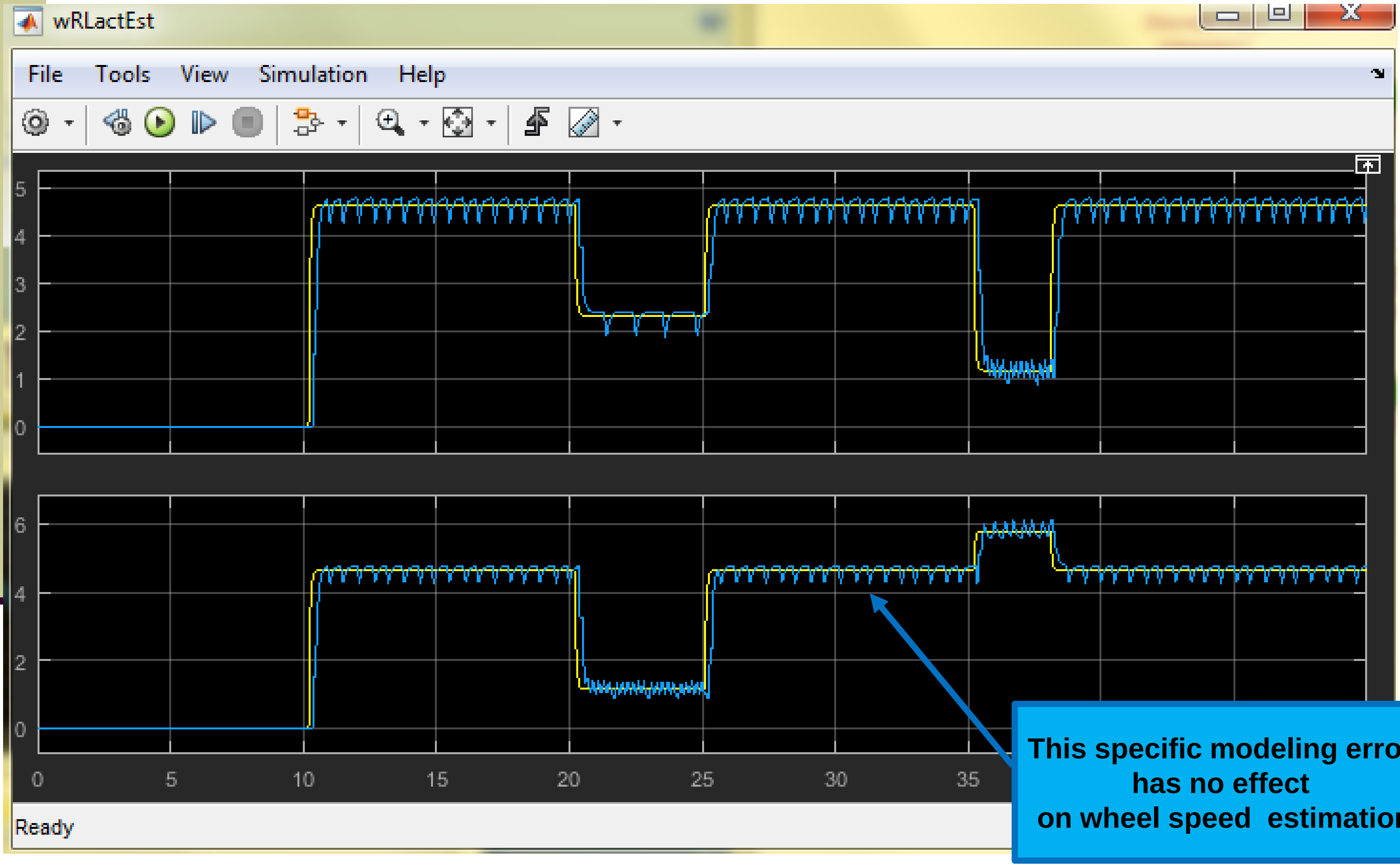
15% error in vehicle parameter b

Simulation Results (Model Error)



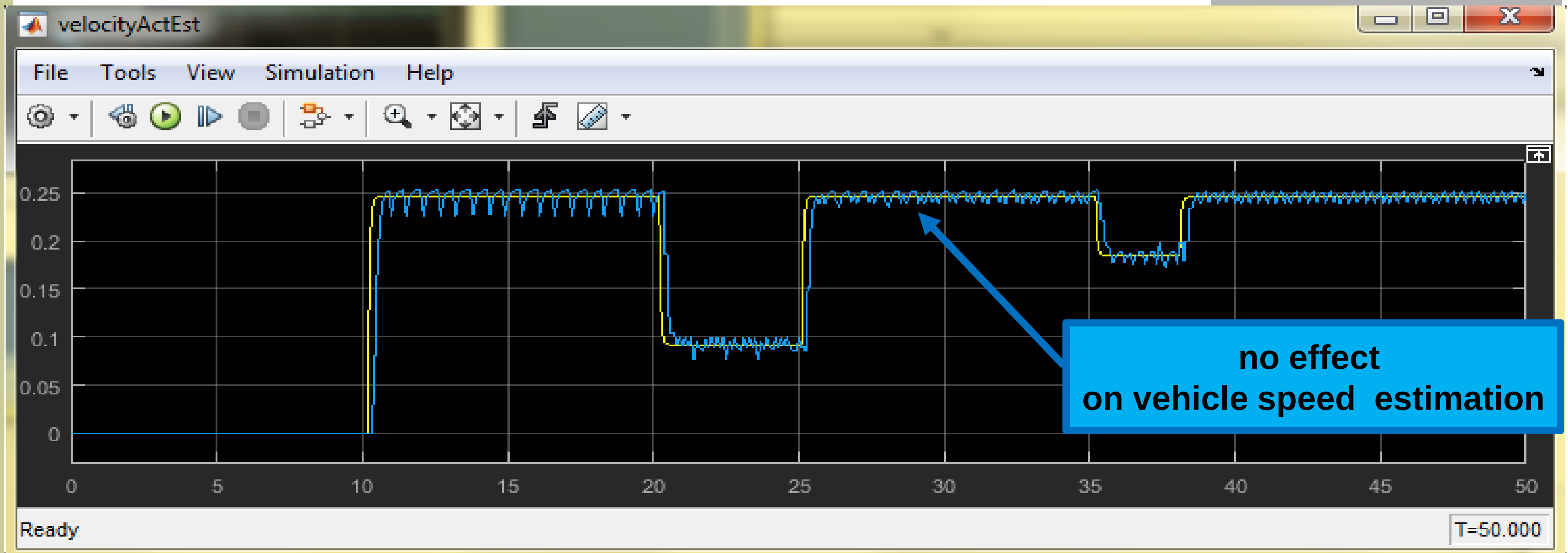


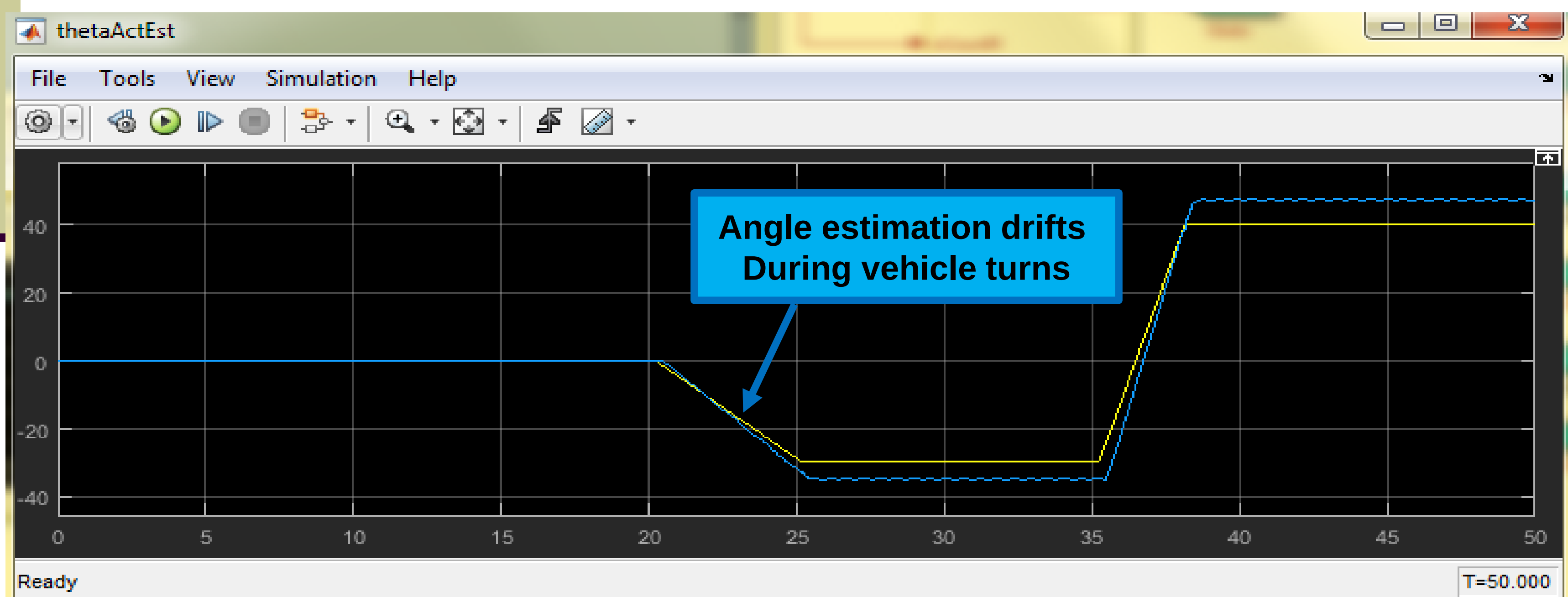
Simulation Results (Model Error)



This specific modeling error has no effect on wheel speed estimation

Simulation Results (Model Error)





Simulation Results (Model Error)

Recall Equations used in Dead reckoning

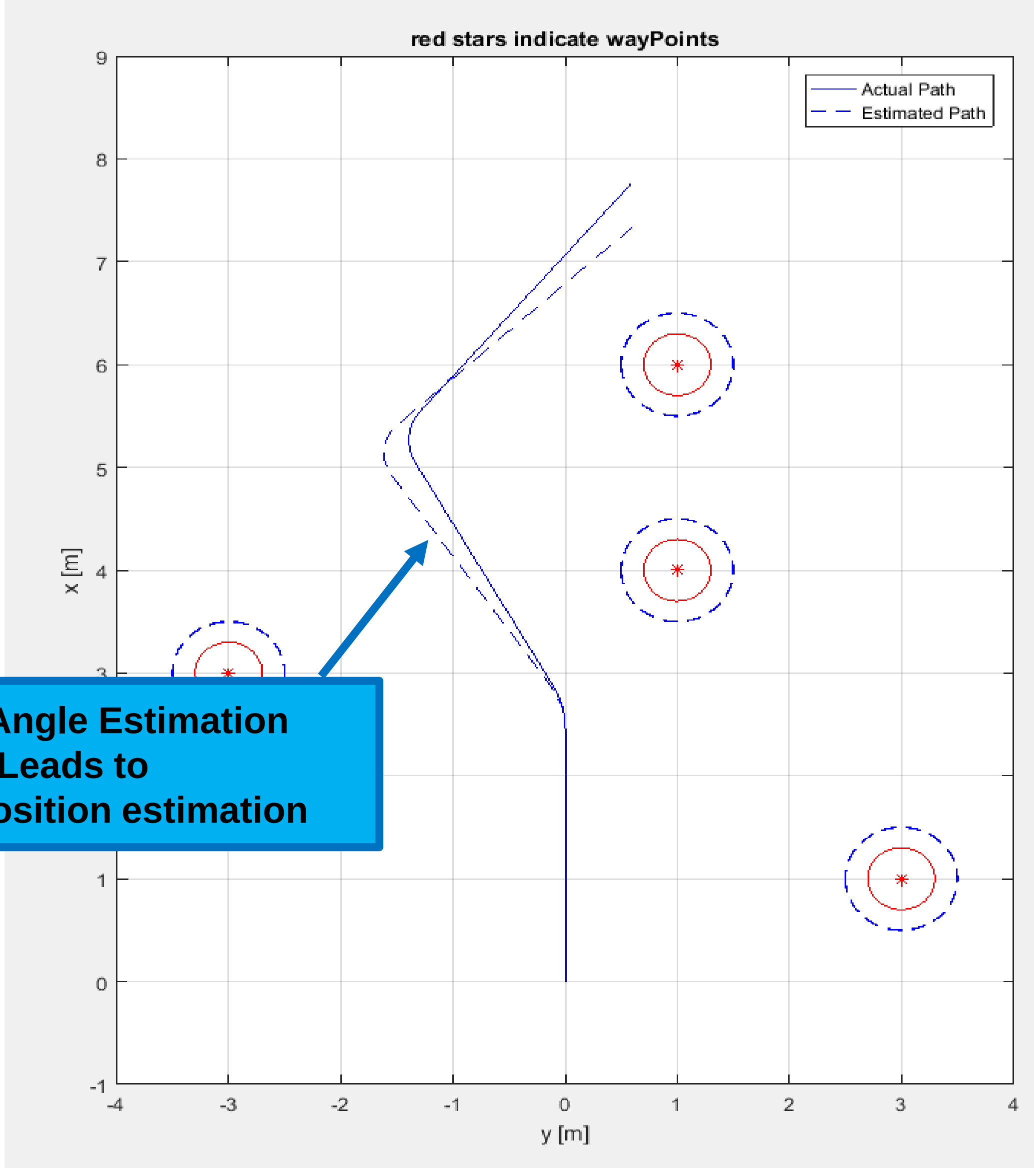
$$\dot{\hat{x}} = \frac{r}{2}\left(\hat{\omega}_L + \hat{\omega}_R\right)\cos\hat{\theta}$$

$$\dot{\hat{y}} = \frac{r}{2}\left(\hat{\omega}_L + \hat{\omega}_R\right)\sin\hat{\theta}$$

$$\dot{\hat{\theta}} = \frac{r}{b}\left(\hat{\omega}_L - \hat{\omega}_R\right)$$

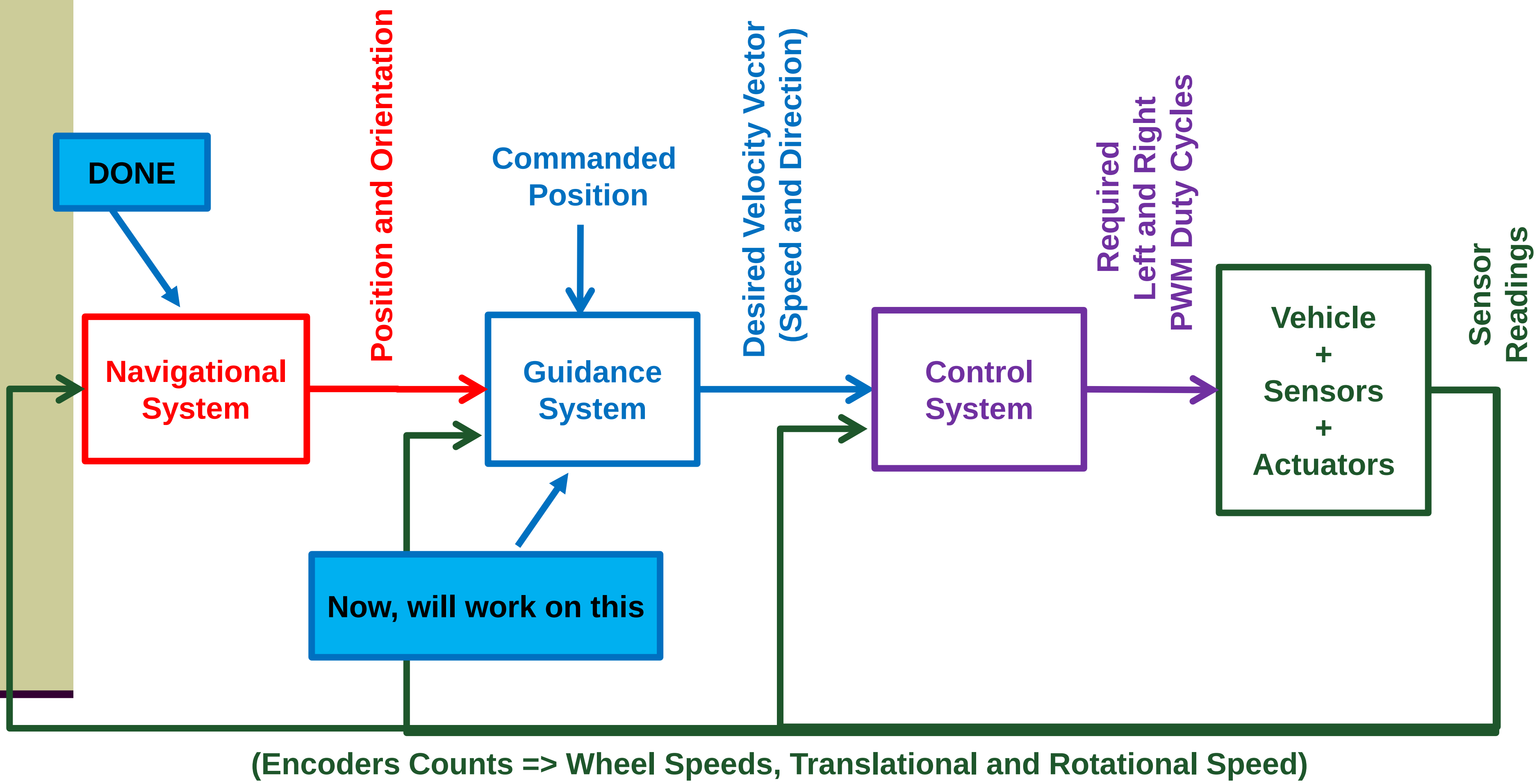
Parameter b appears in rotational speed equation

Simulation Results (Model Error)

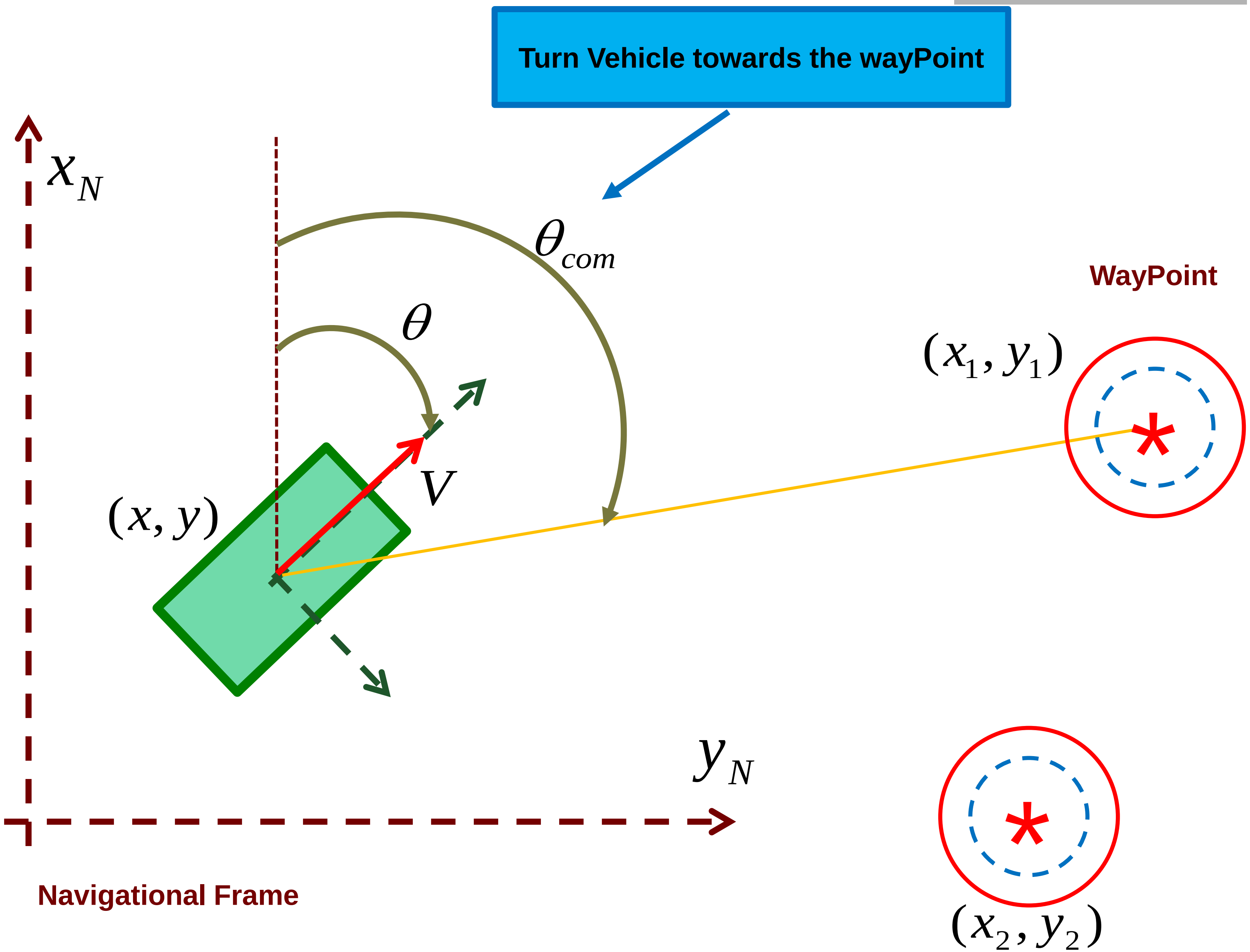


Drift in Angle Estimation
Leads to
Drift in position estimation

GNC of the UGV



Guidance Strategy - Direction



Commanded Direction

- Angle to turn towards to the wayPoint

$$\theta_{com} = \tan^{-1}\left(\frac{y - y_i}{x - x_i}\right), i = \{1, 2, \ldots, n\}$$

where *n* is the total number of wayPoints

- Distance to the wayPoint

$$|r| = \sqrt{(y - y_i)^2 + (x - x_i)^2}$$

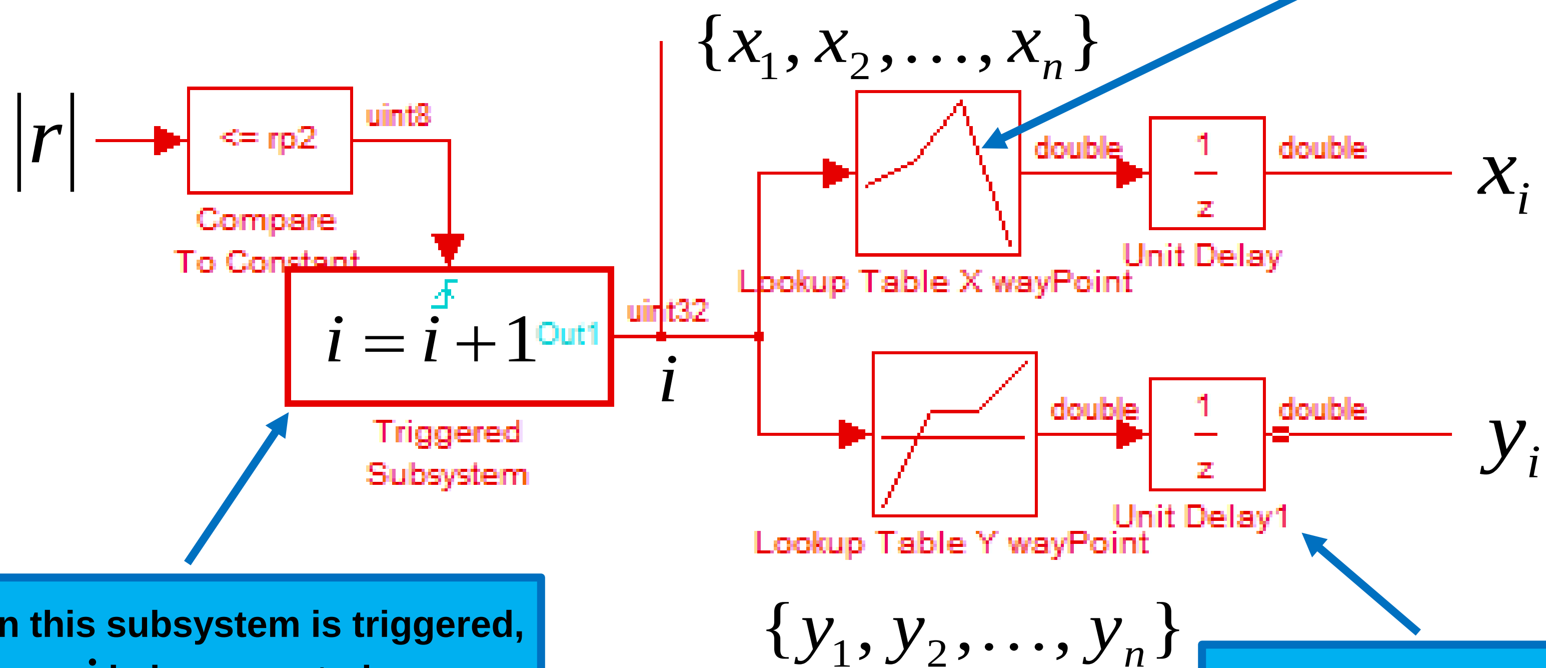
- When $|r| \leq r_{p2}, i = i + 1$ switch to the next wayPoint

Implementation of the Switch - 1

Open trackVehicleNewVer6b_Encoders_DR_GNC.slx

- Using Triggered Subsystem

LookUp tables contains the x and y coordinates of the wayPoints

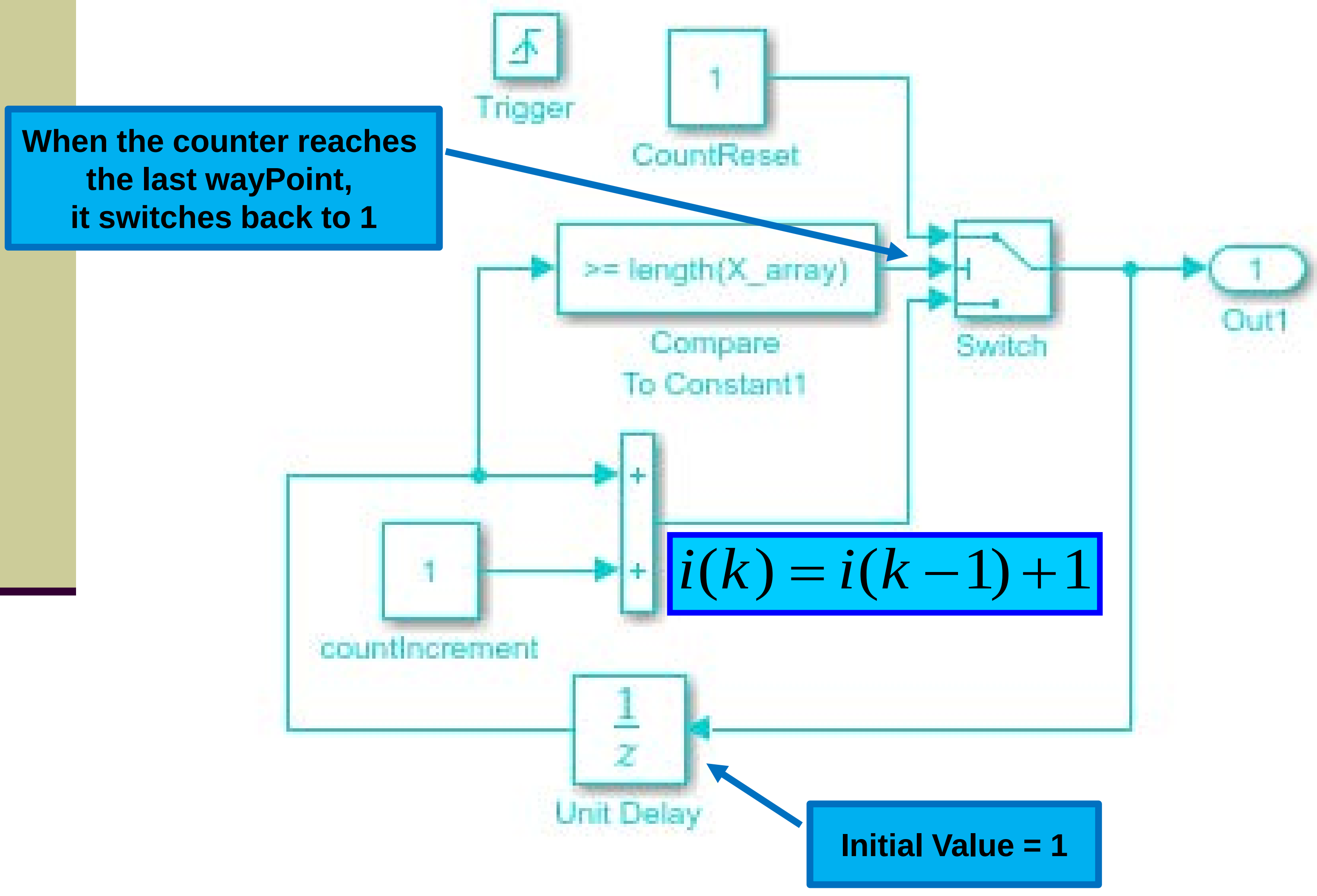


When this subsystem is triggered, ***i*** **is incremented**

Unit Delays are to avoid algebraic loops

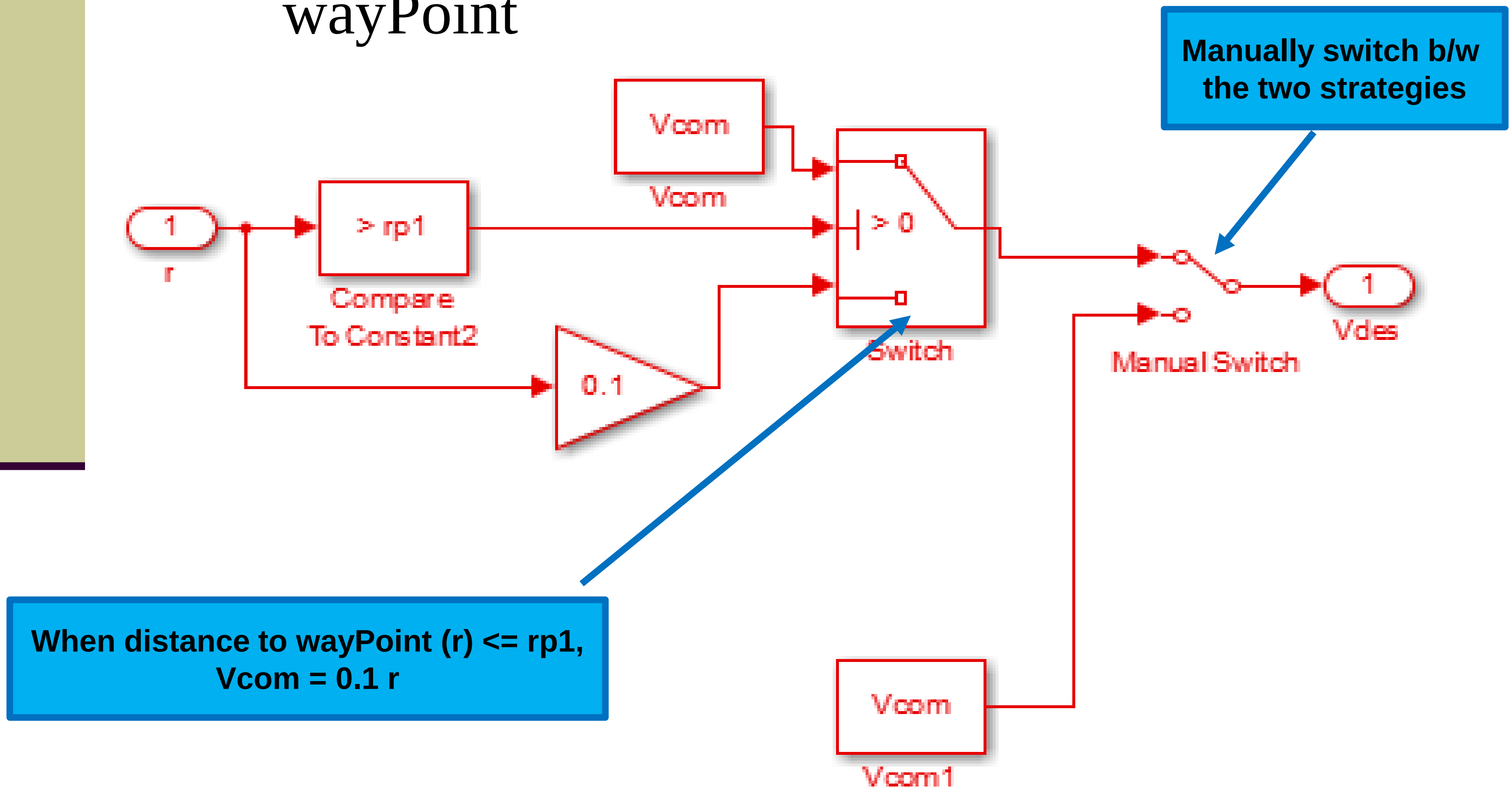
Implementation of the Switch - 2

Triggered Subsystem:



Guidance Strategy - Speed

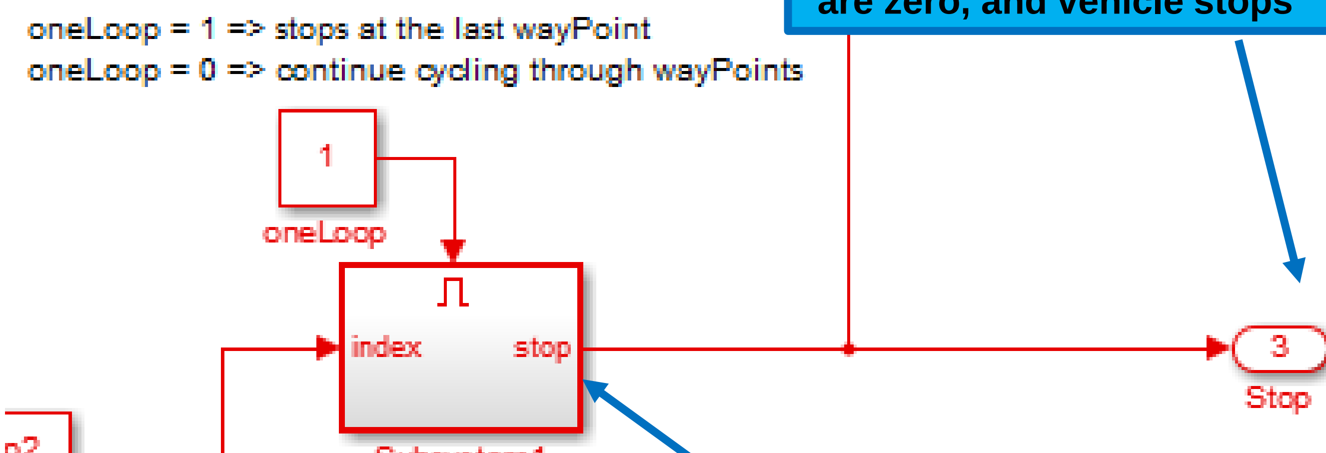
- Two strategies:
    1. Constant commanded speed always < Max Speed
    2. Commanded speeds is reduced when close to wayPoint



Stop at last wayPoint or Continue -1

- Using Enabled Subsystem

“Stop” can take 0 or 1
It multiplies the wheel speed commands
1) When it is 1, the commanded speeds are what calculated by Guidance
2) When it is 0, the commanded wheel speed are zero, and vehicle stops

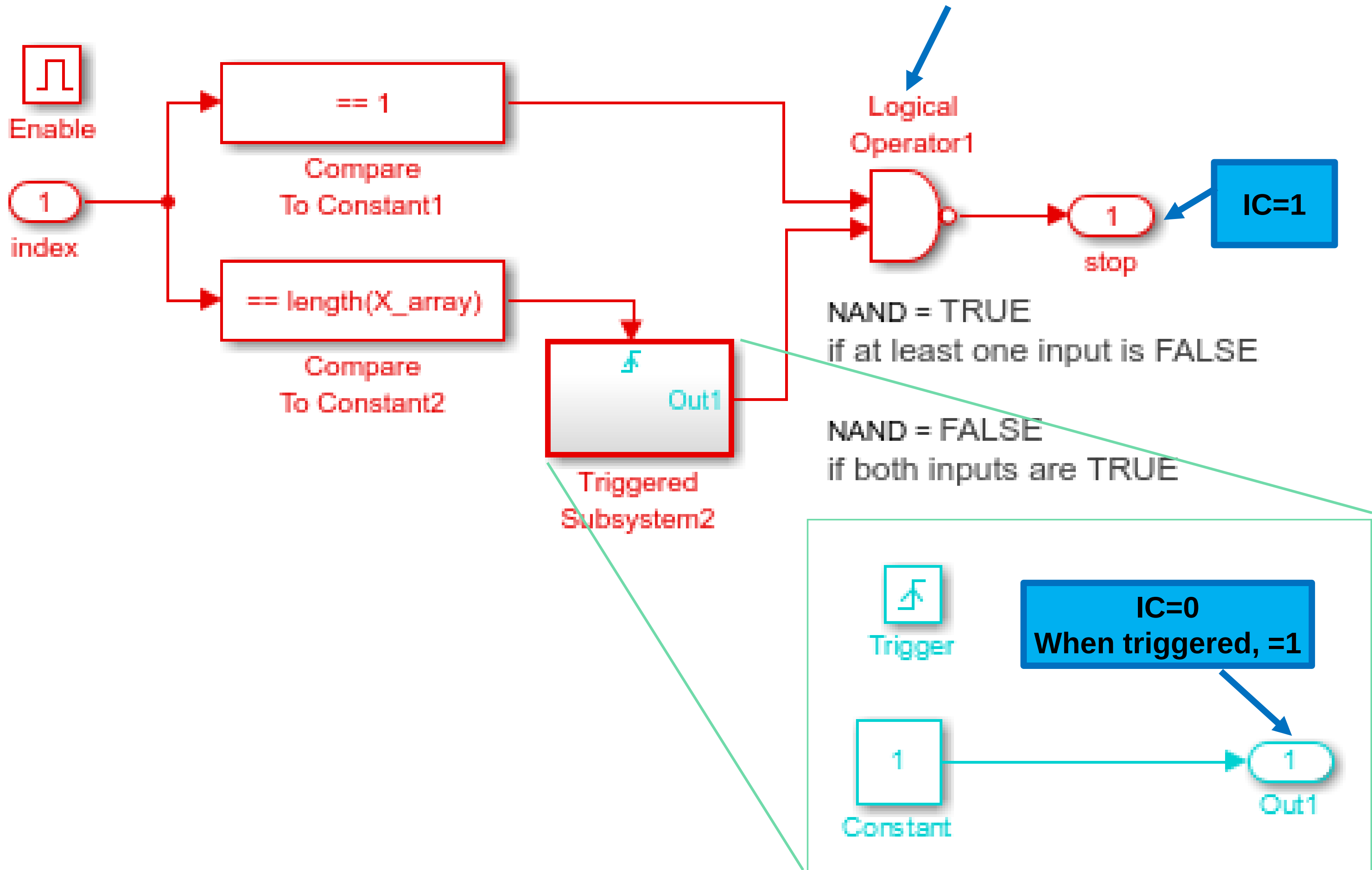


If enabled,
it outputs 0 when last wayPoint is reached

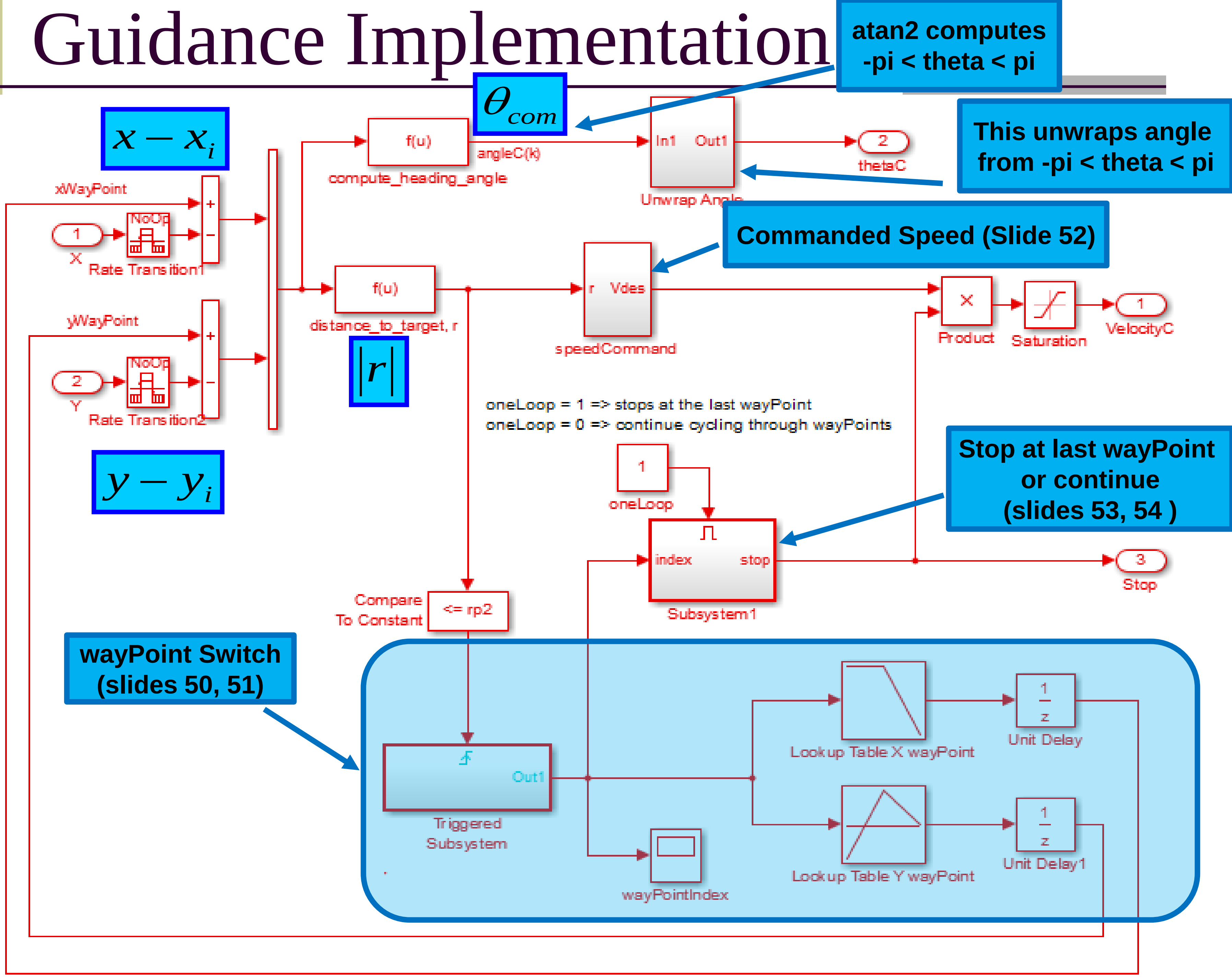
Stop at last wayPoint or Continue -2

- Enabled Subsystem

When both inputs are TRUE, i.e.,
Vehicle reached the last wayPoint AND the wayPoint index was switched to 1
(This is to exclude the initial time when the wayPoint index is 1)
The output is FALSE (=0)

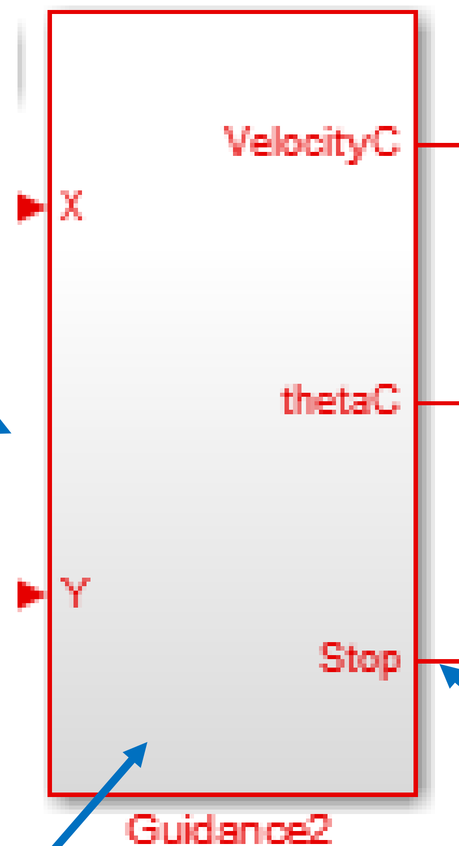


Guidance Implementation



Guidance Subsystem

Takes
Estimated Position
as inputs

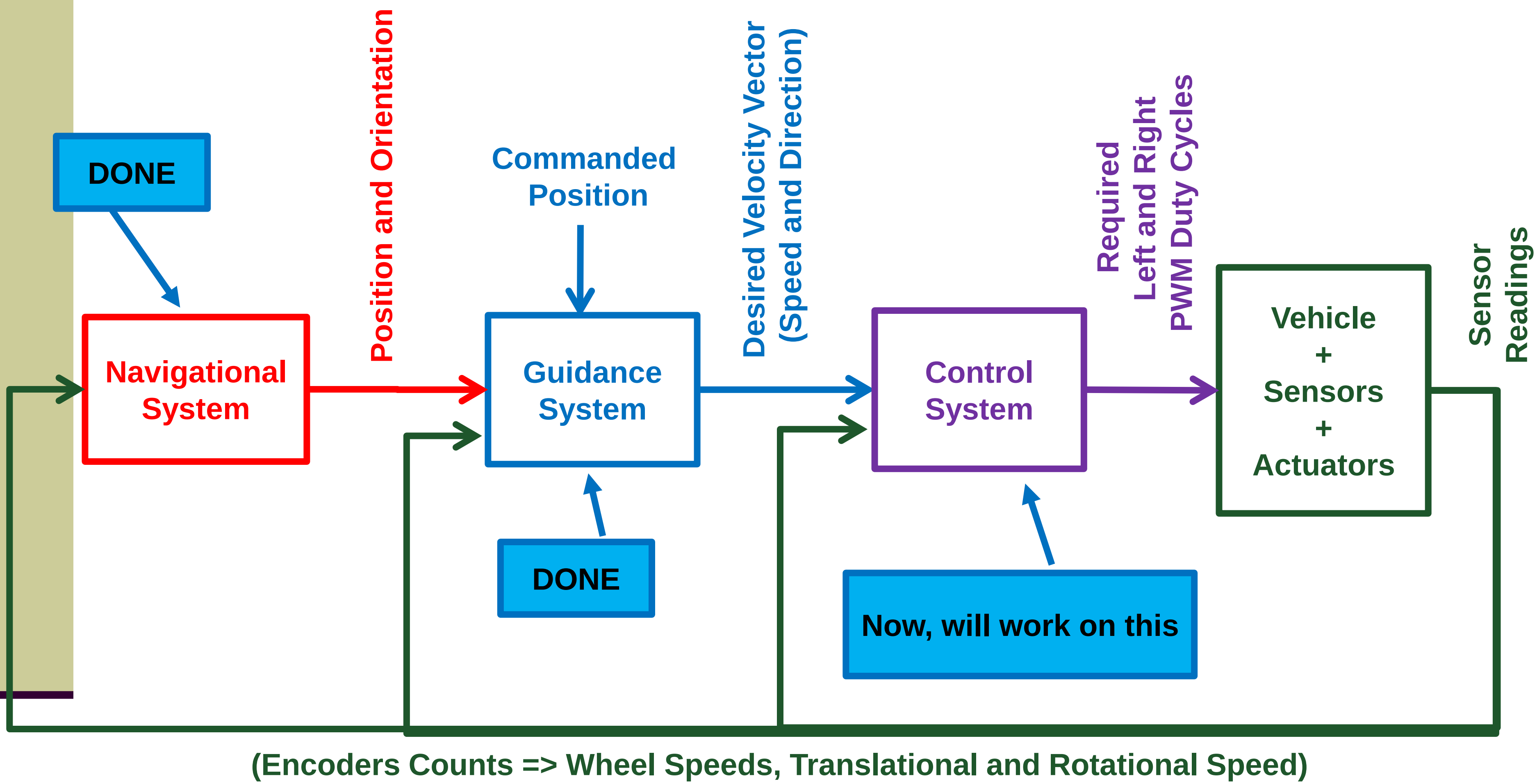


Computes
Commanded
Speed and Direction

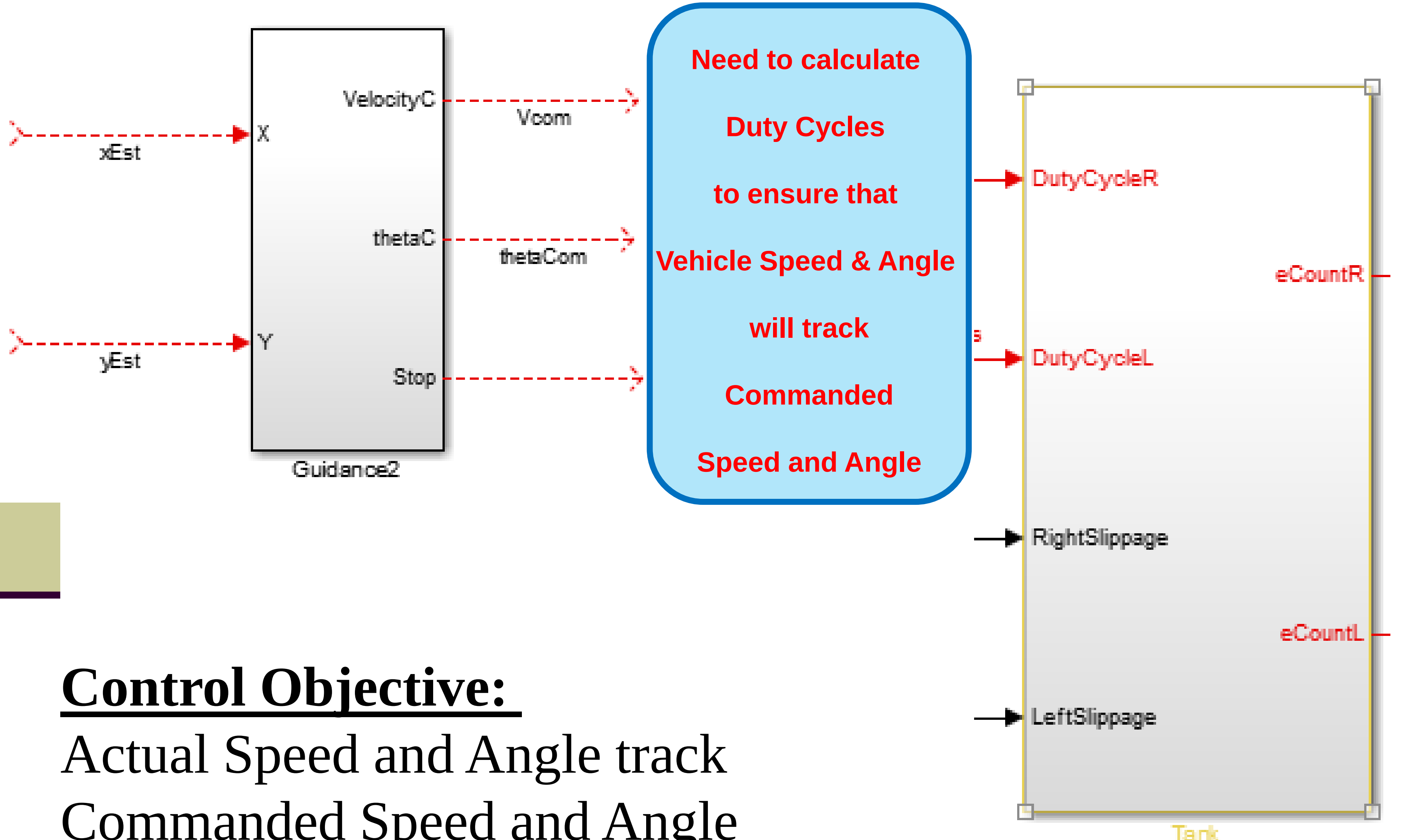
Also takes
the coordinates of
the wayPoint

If last wayPoint, Stop = 0
(when stop at the last wayPoint is selected)

GNC of the UGV



Controller – Why Needed?



<u>Control Objective:</u>

Actual Speed and Angle track Commanded Speed and Angle

Control Design Approach

- Recall (when no slippage and right/left tracks identical)

$$\dot{x} = \frac{r}{2}\left(\omega_L + \omega_R\right)\cos\theta$$

$$\dot{y} = \frac{r}{2}\left(\omega_L + \omega_R\right)\sin\theta$$

$$\dot{\theta} = \frac{r}{b}\left(\omega_L - \omega_R\right)$$

- Speed

$$V = \sqrt{\dot{x}^2 + \dot{y}^2} = \frac{r}{2}\left(\omega_L + \omega_R\right)$$

- Define total and differential wheel speed

$$\tilde{\omega} = \omega_L + \omega_R$$

$$\Delta\omega = \omega_L - \omega_R$$

Control Design Approach

- Translational and Angular Speed

$$V = \frac{r}{2}\tilde{\omega}$$

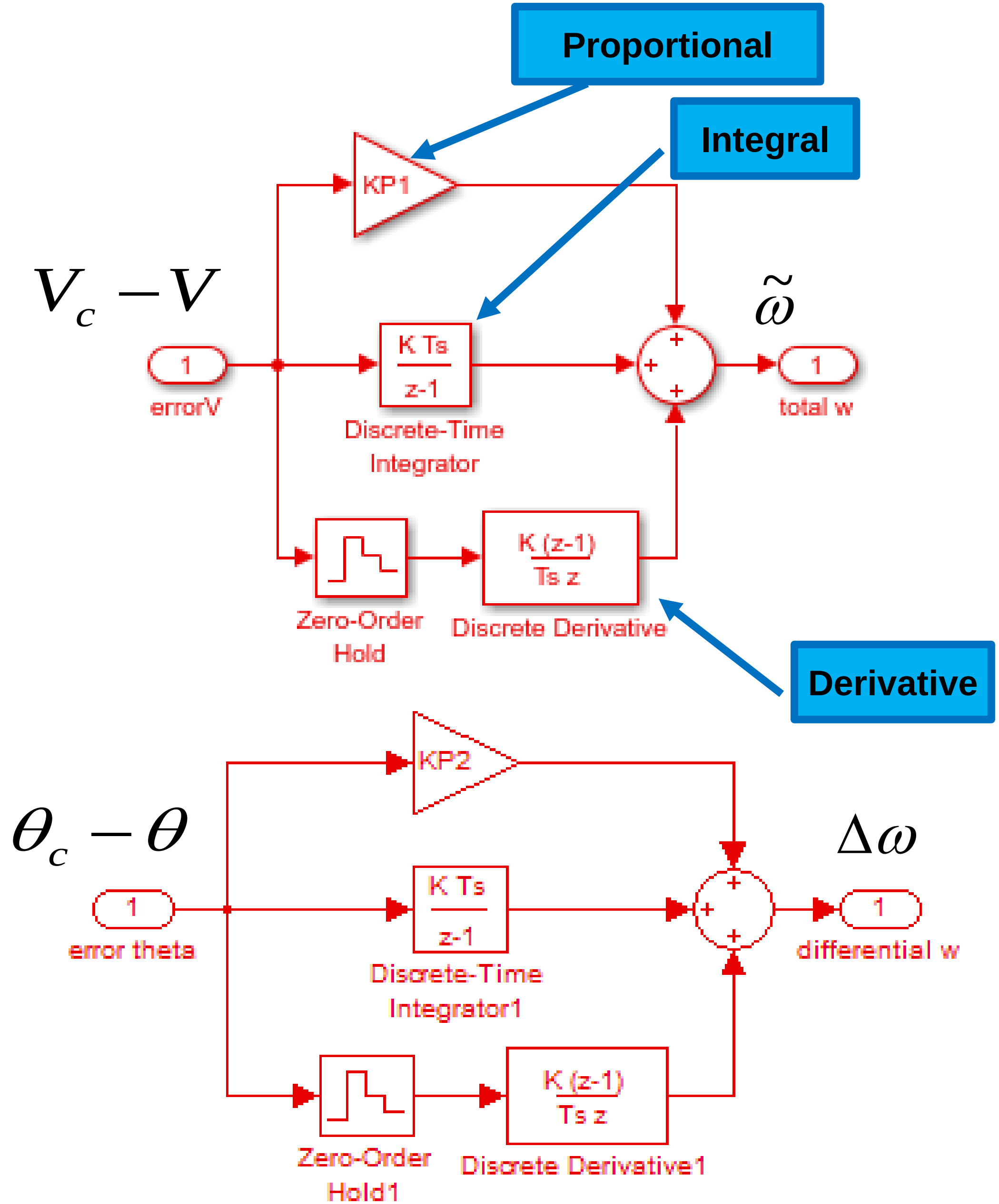
Total wheel appears only in V

$$\dot{\theta} = \frac{r}{b}\Delta\omega$$

Differential wheel appears only in V

- We use total wheel speed to control *V*
- We use differential wheel speed to control θ
- We use PID (Proportional-Integral-Derivative) Control

Two PID Controllers



$$\tilde{\omega} = \omega_L + \omega_R$$

$$\Delta\omega = \omega_L - \omega_R$$

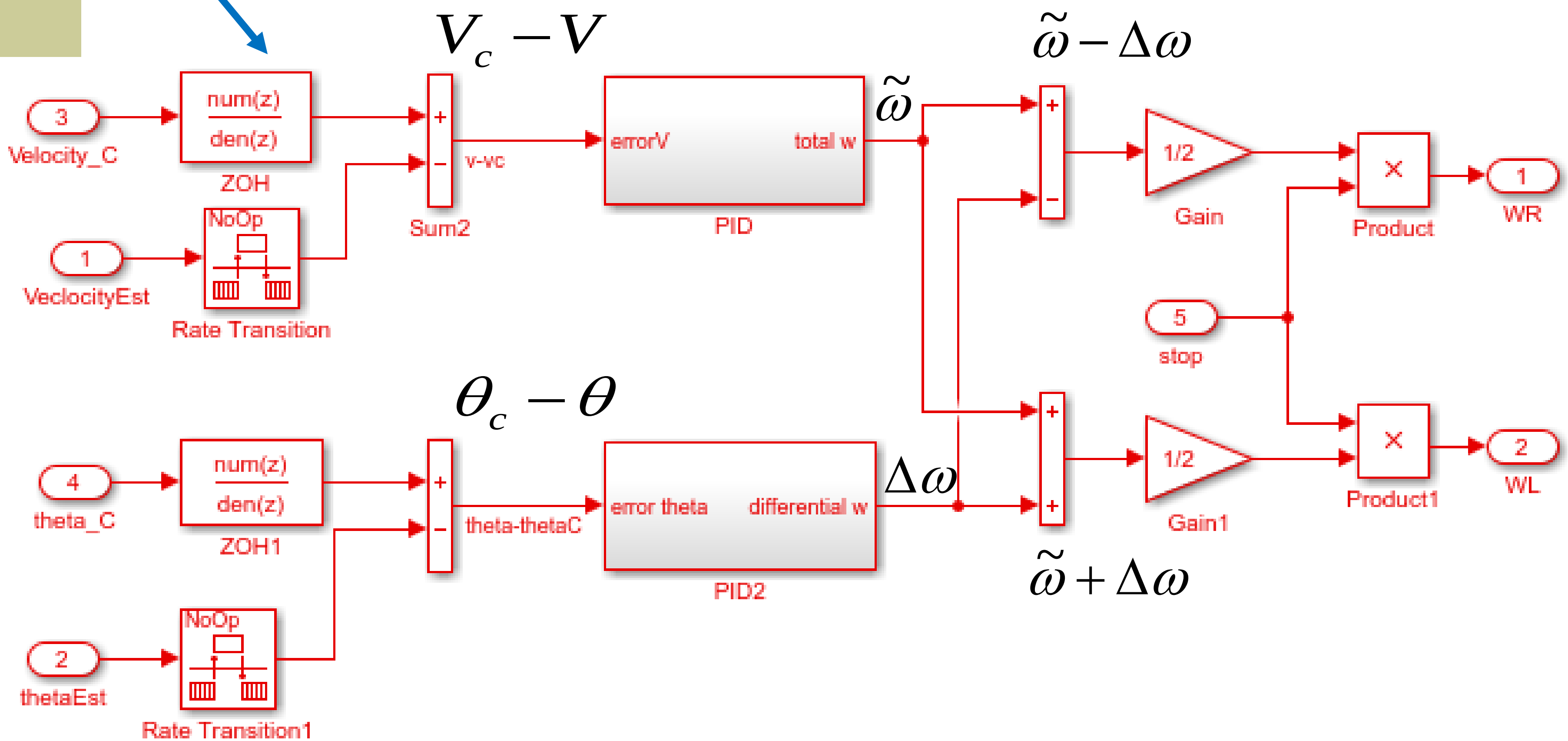
$$\Rightarrow$$

$$\begin{cases} \omega_R = \dfrac{\tilde{\omega} - \Delta\omega}{2} \\ \omega_L = \dfrac{\tilde{\omega} + \Delta\omega}{2} \end{cases}$$

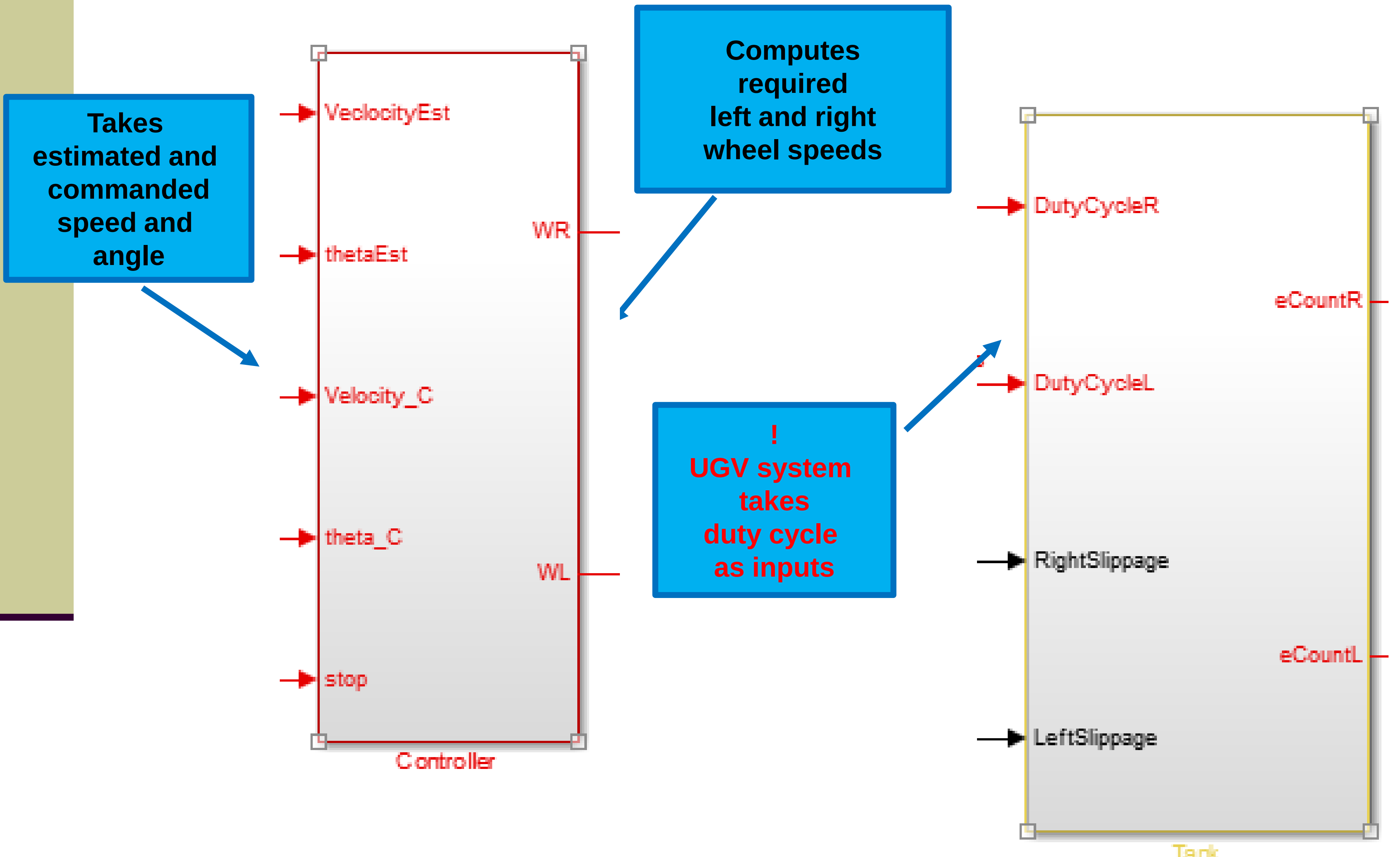
Once total and differential wheel speed are calculated, Required Right and Left Wheel Speeds

Controller Implementation

1st-order filter to smoothen the signal



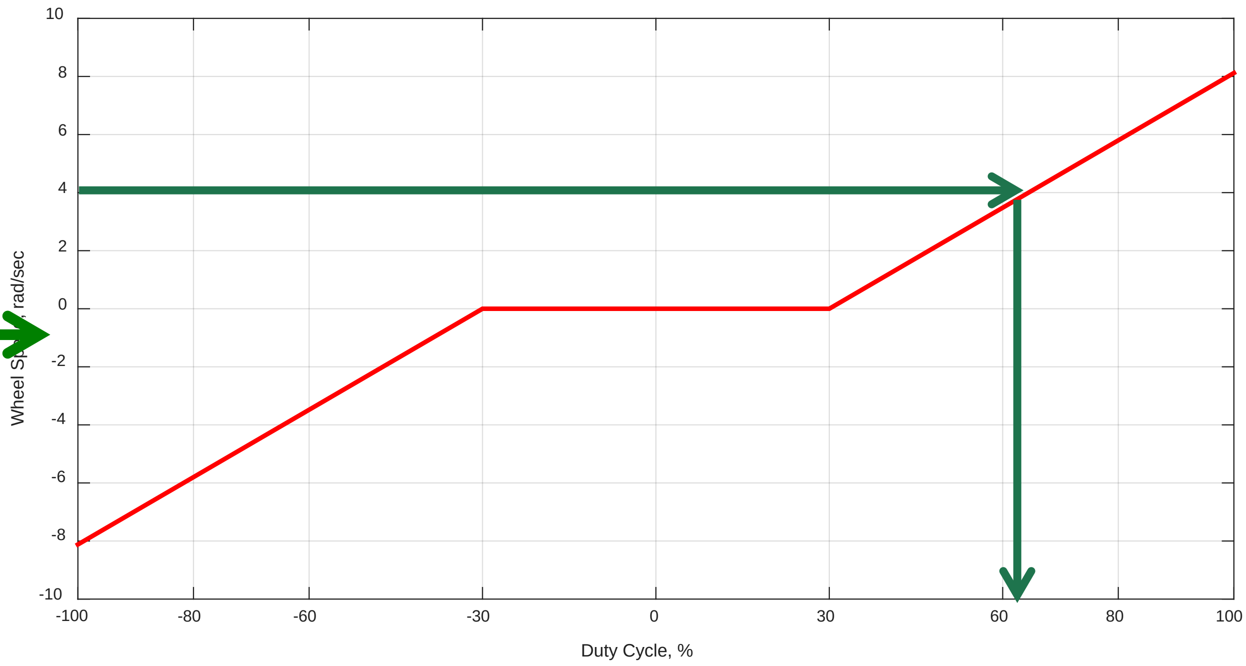
Controller Subsystem



Duty Cycle Commands

- Recall “Wheel Speed – Duty Cycle” Relation

Required Wheel Speed

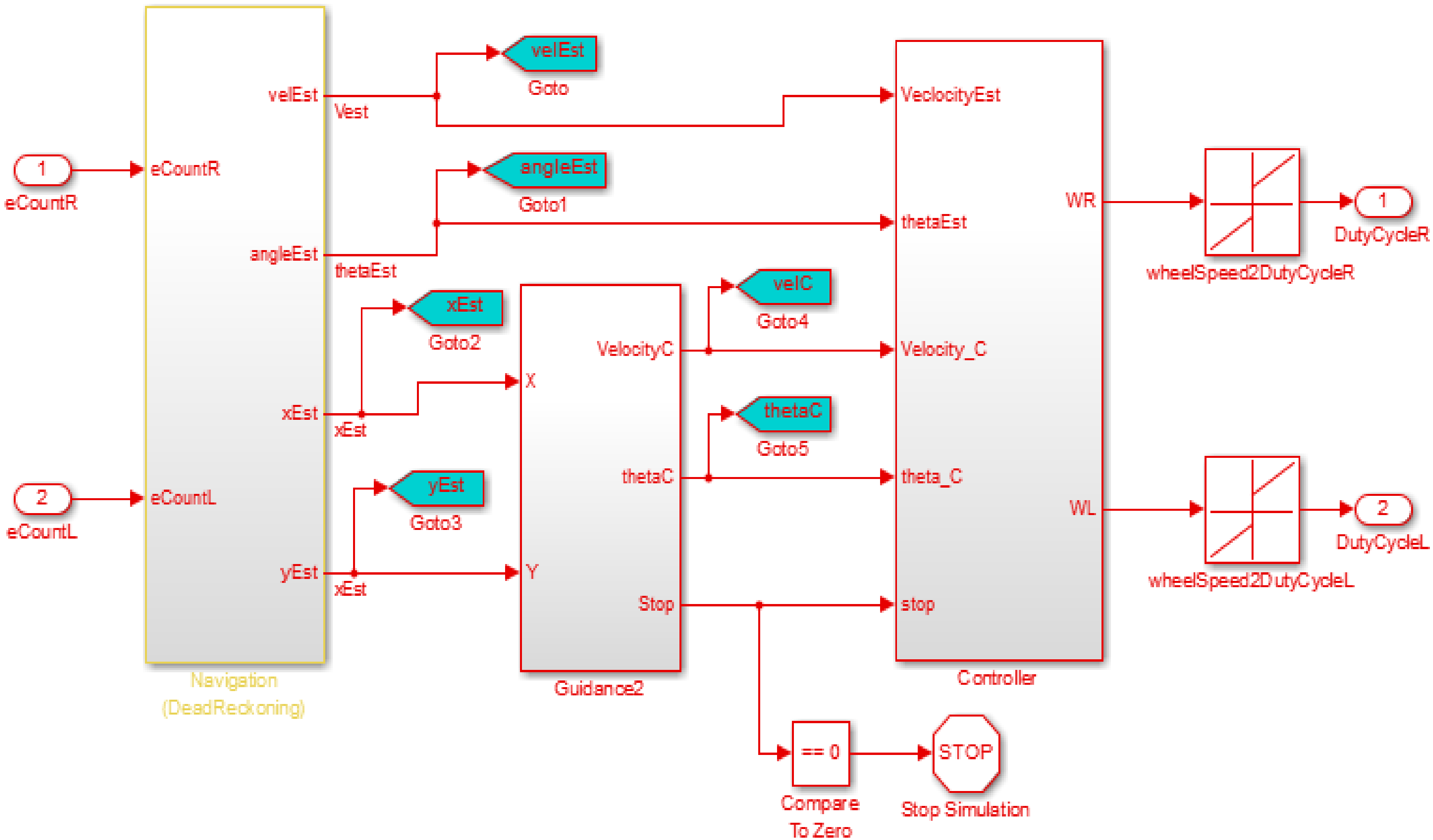


Percent Duty Cycle

LookUp Table

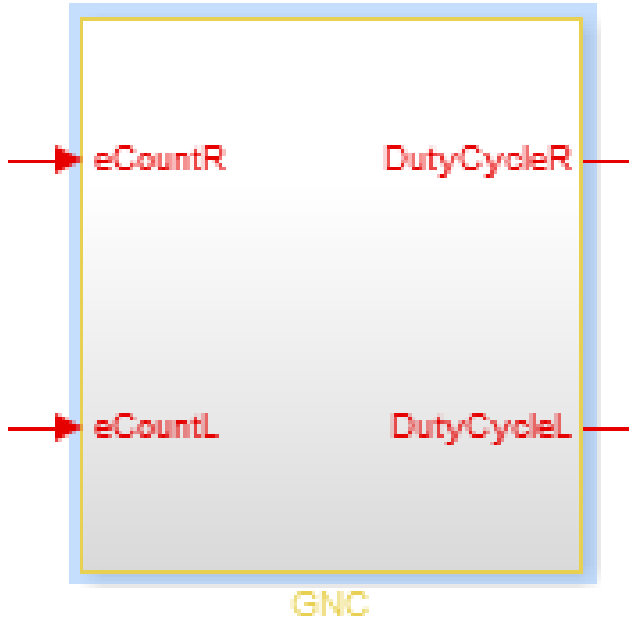
GNC Implementation

- All pieces put together



GNC Subsystem

Takes
Left and right wheel
Encoder counts



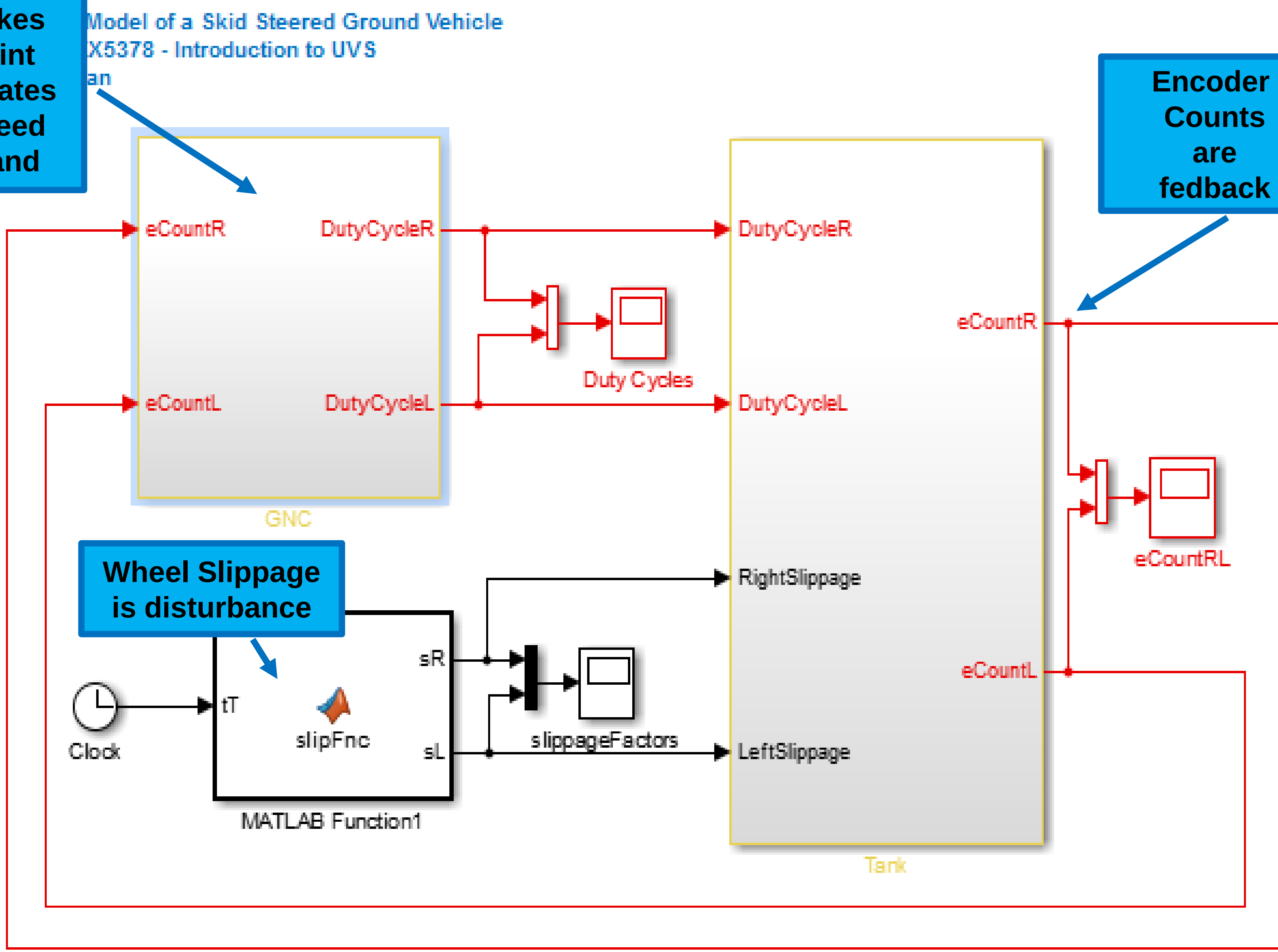
Computes
%Duty Cycles
For Left and Right
Backward and Forward

Closed-Loop System

GNC takes wayPoint Coordinates And speed command

Encoder Counts are fedback

Wheel Slippage is disturbance

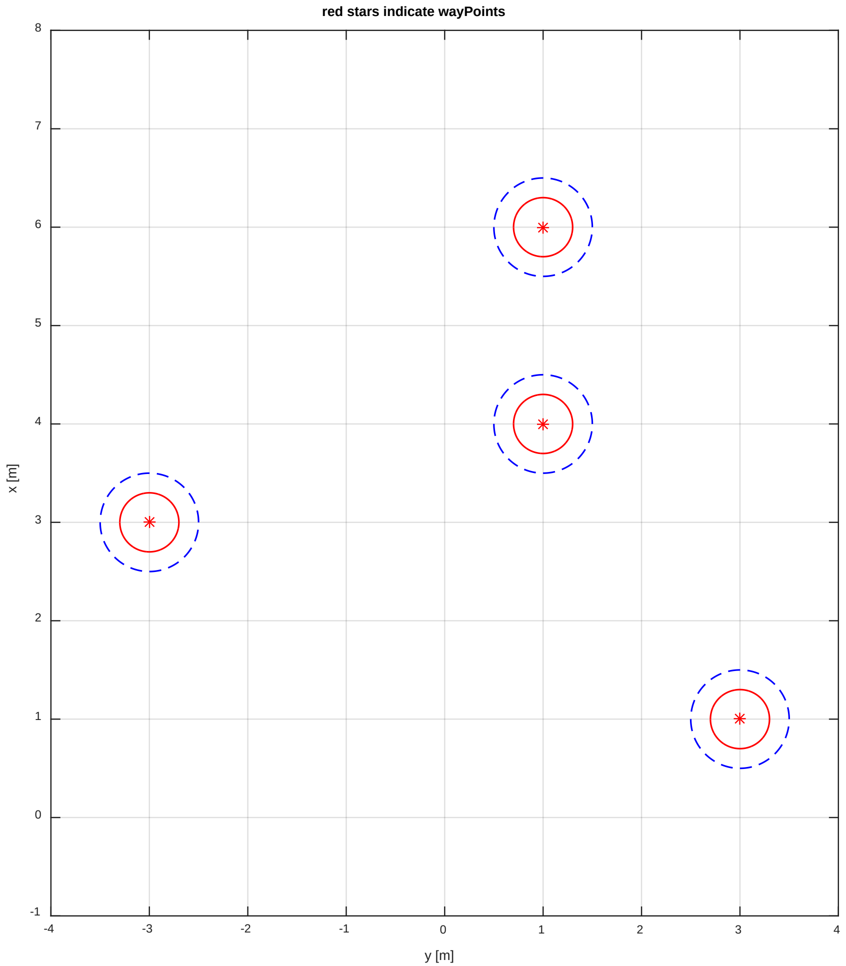


Simulation Results

- GNC takes the UGV through the assigned wayPoints
- Stop at the last wayPoint or Continue
- Two speed guidance strategies
  - Constant Commanded Speed
  - Slow down at wayPoints
- What happens if there is slippage
- What happens if there is modeling error
- What happens if “Unwrap Angle” block not used
- Adjusting PID gains

Assigned wayPoints

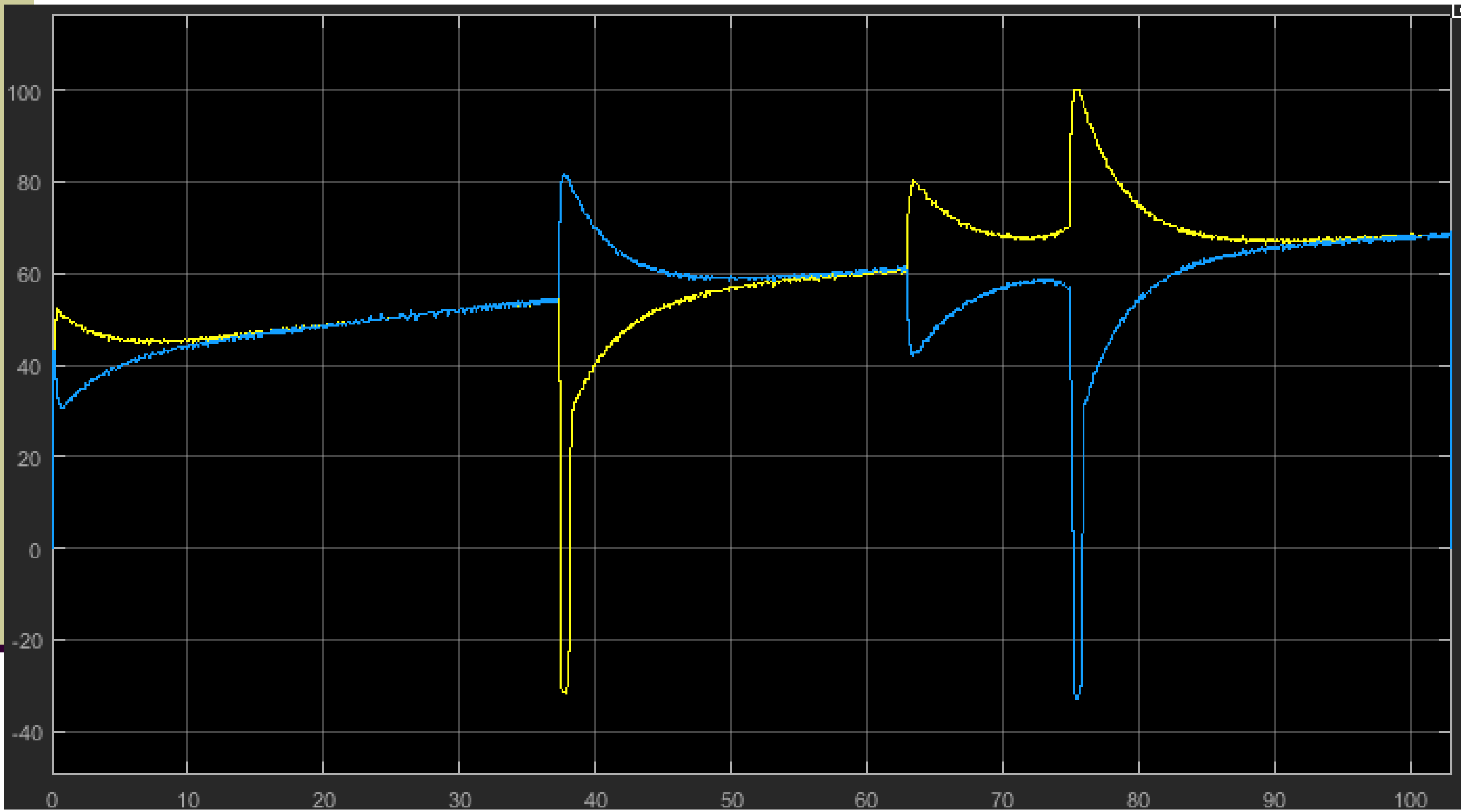
```
%%%%%%%%%%%%%%%%%% Way Points %%%%%%%%%%%%%%%%%%%%%%%%%%%
X_array  = [ 3 4 6 1];
Y_array  = [-3 1 1 3];
```



Simulation Case - 1

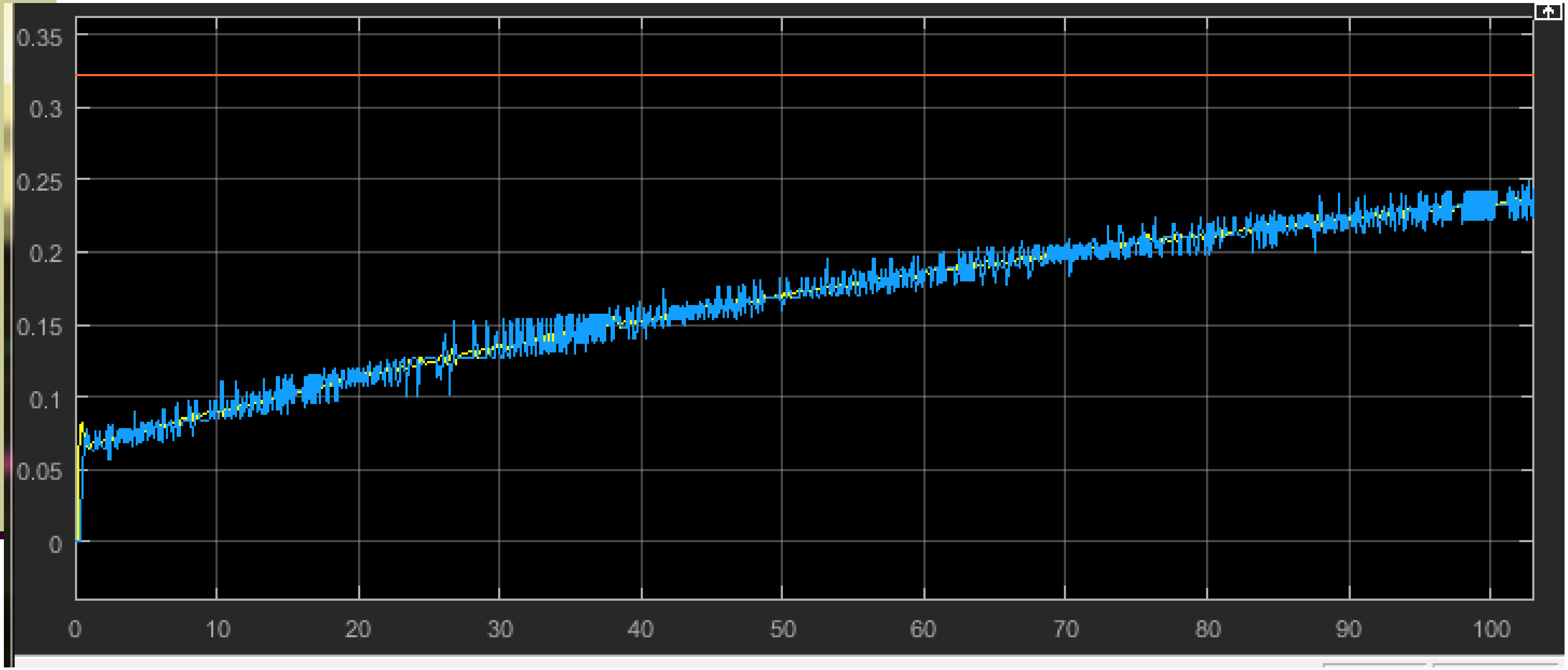
- Vcom = Vmax always
- Stops at the last wayPoint

Duty Cycle Responses



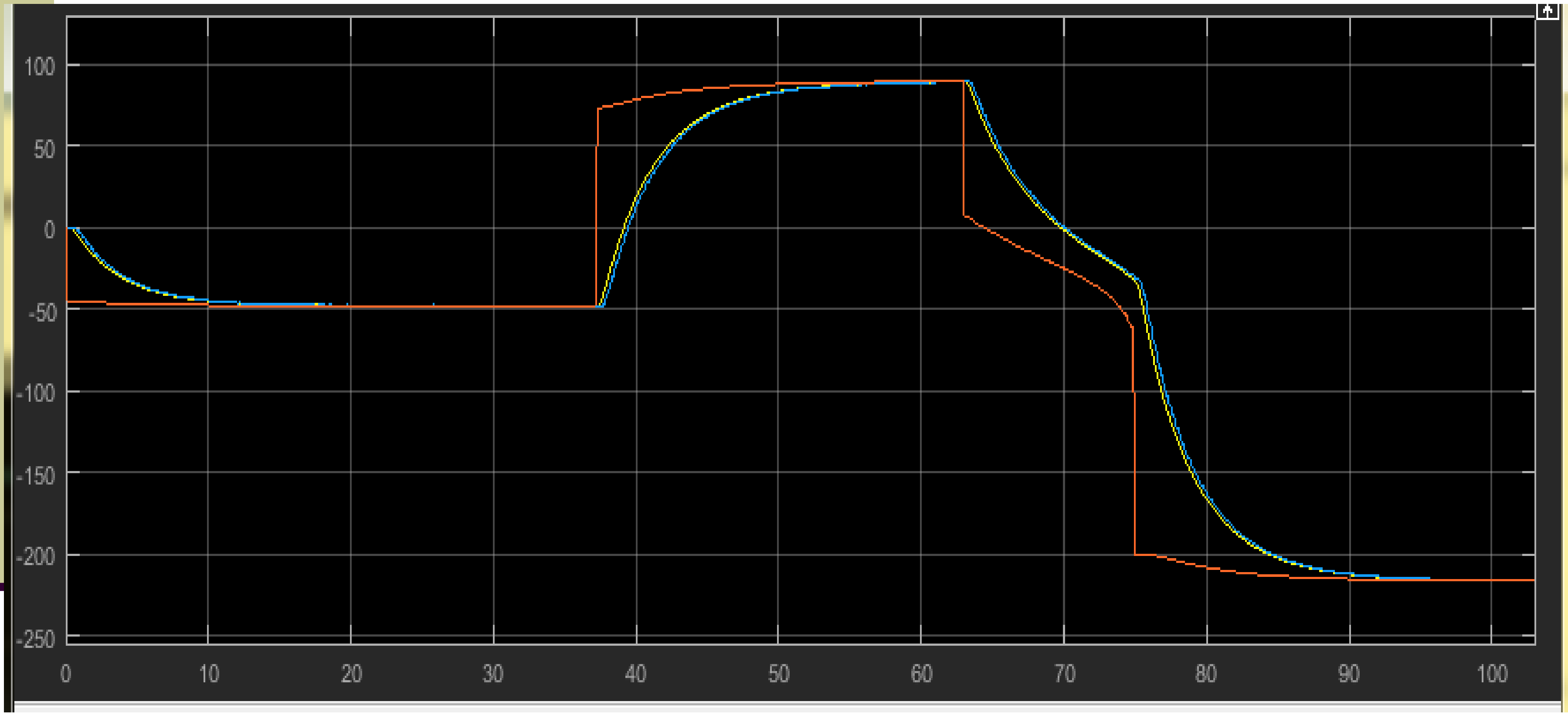
Speed Response

Actual, Estimated and Commanded

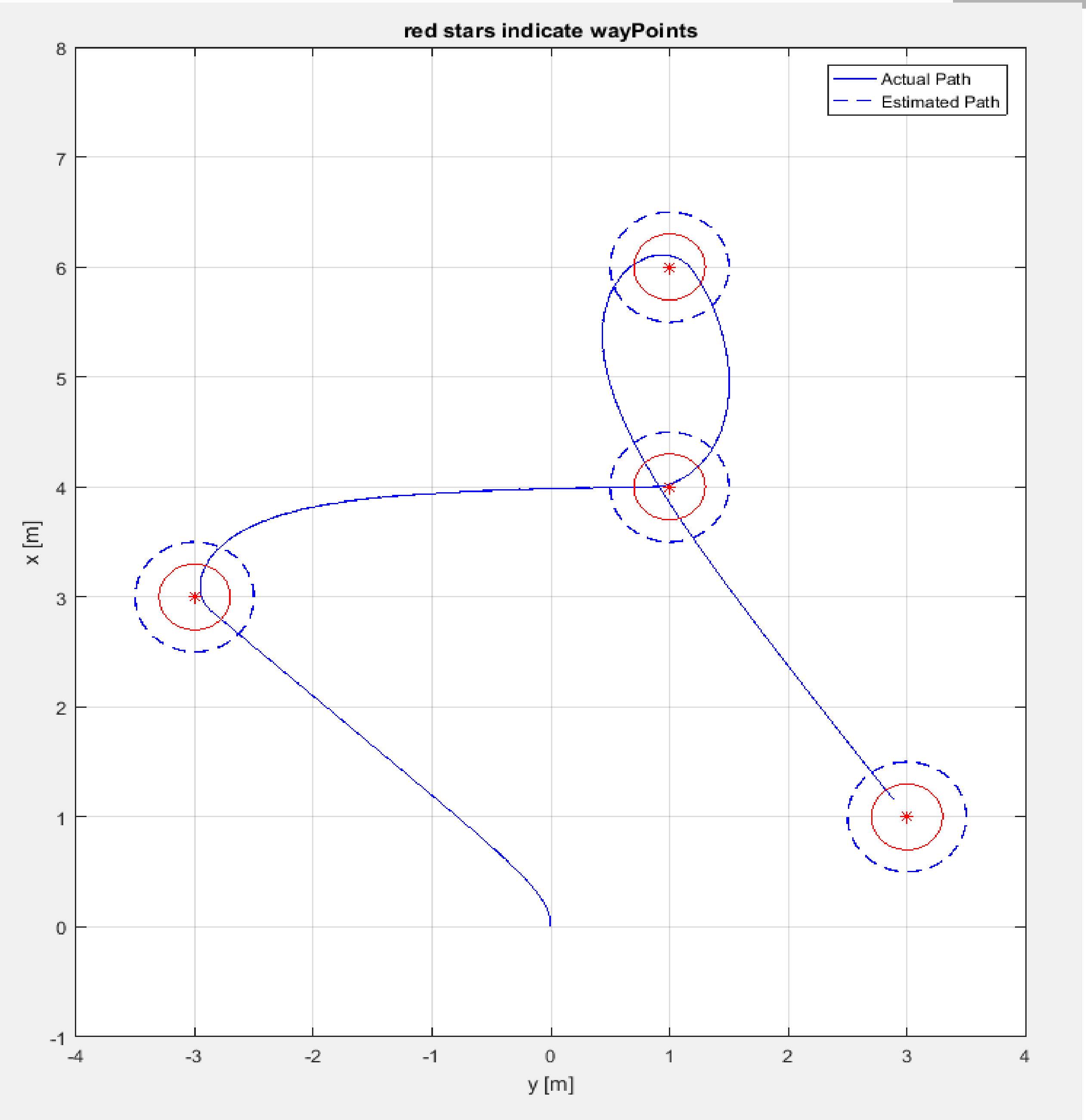


Angle Response

Actual, Estimated and Commanded

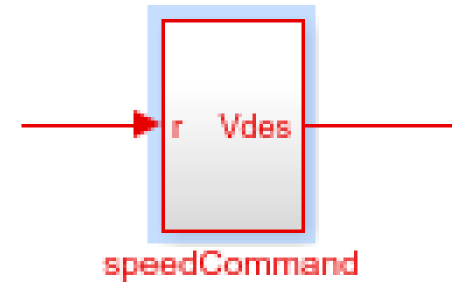


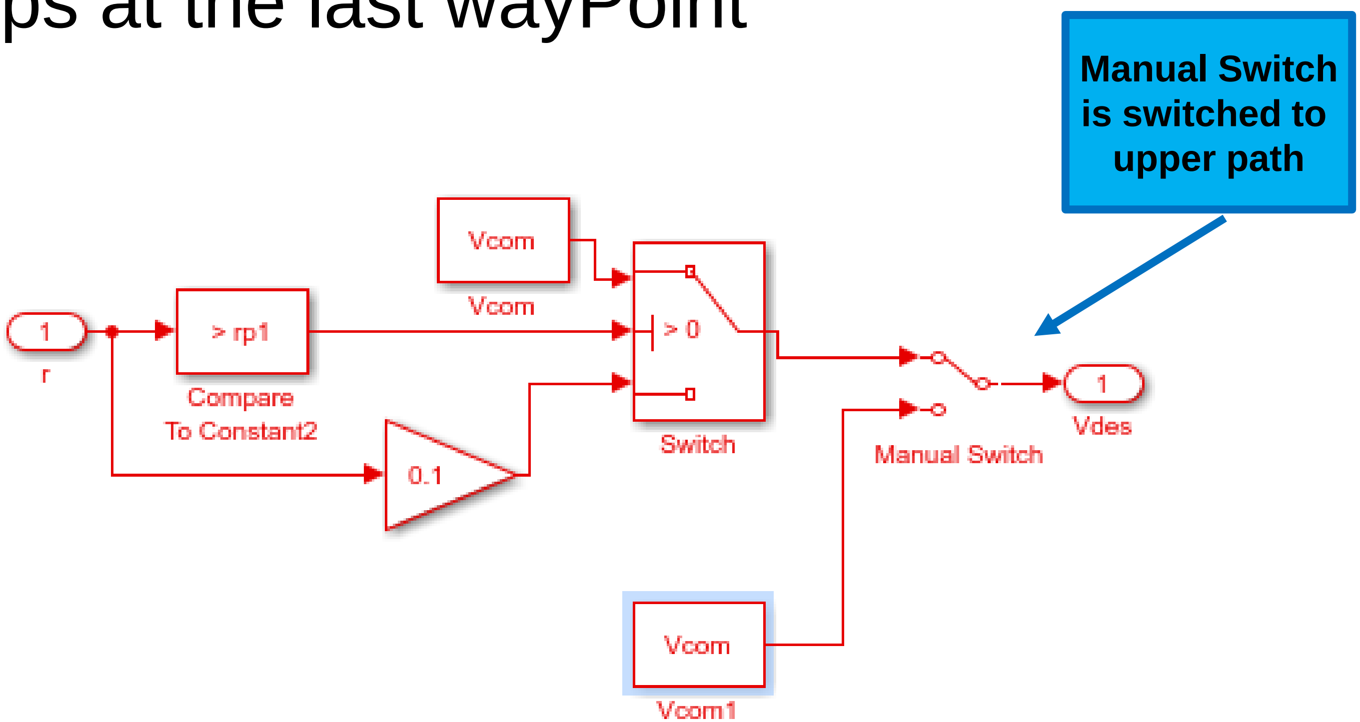
Trajectory Response



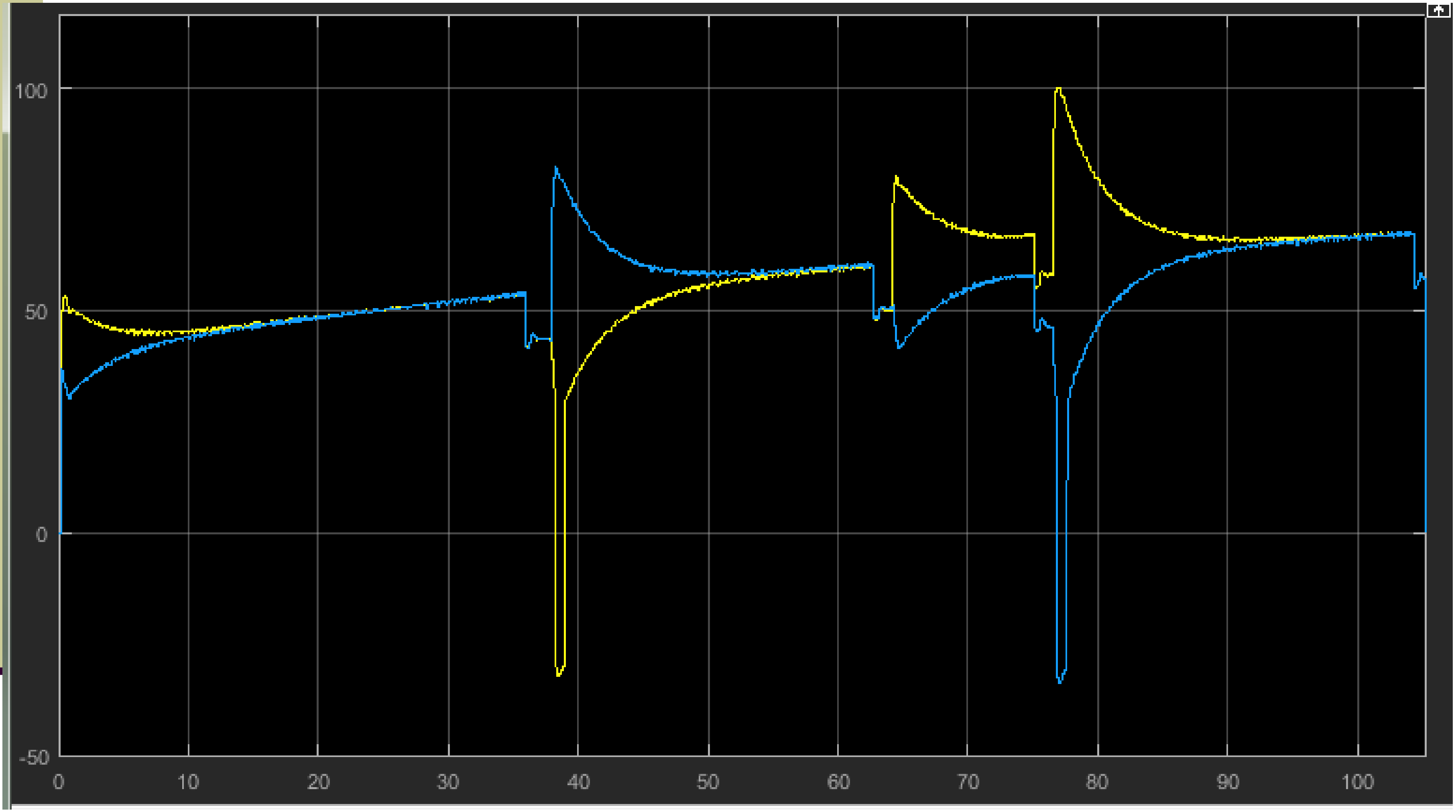
Simulation Case - 2

- Vcom = Vmax
- slows down when close to wayPoint
- Stops at the last wayPoint



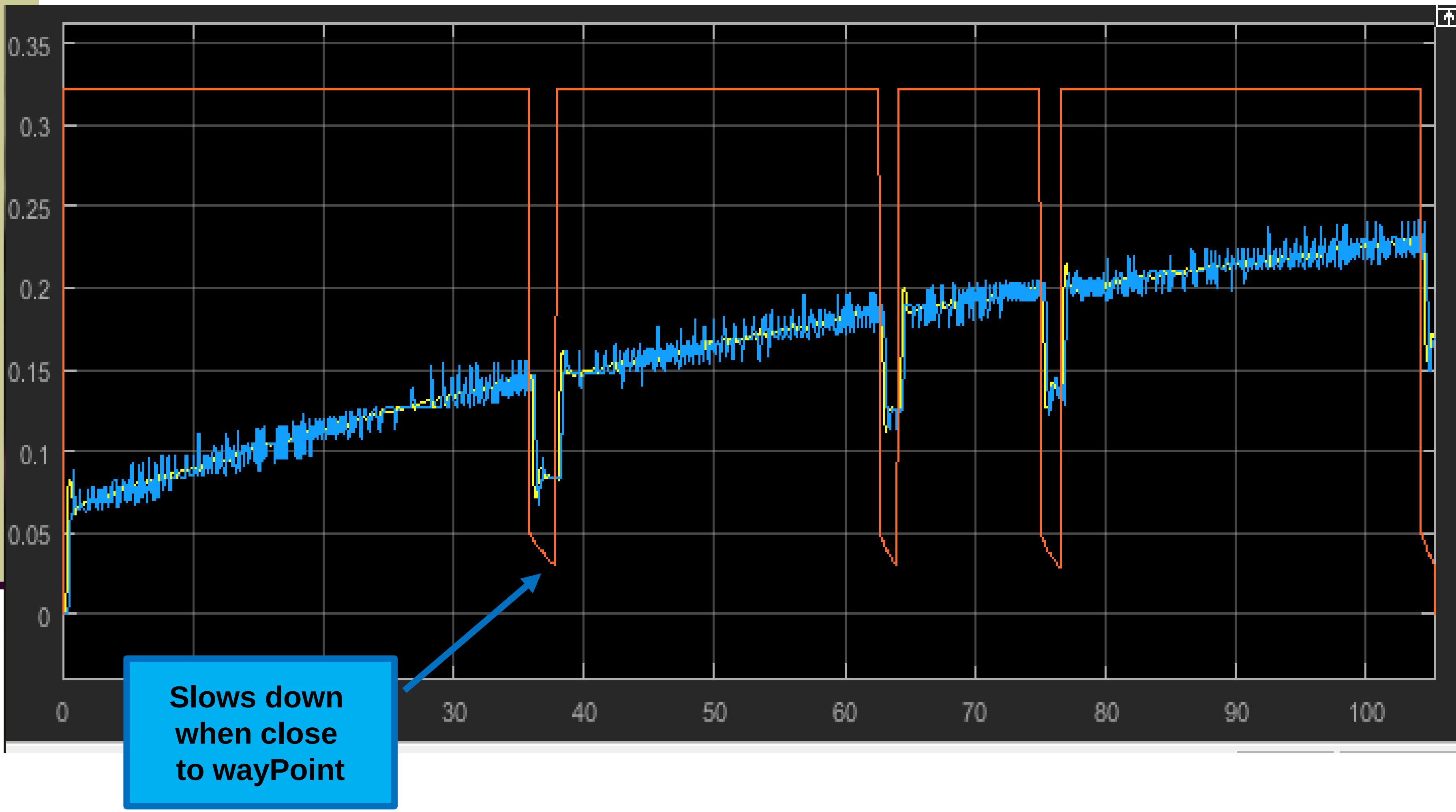


Duty Cycle Responses



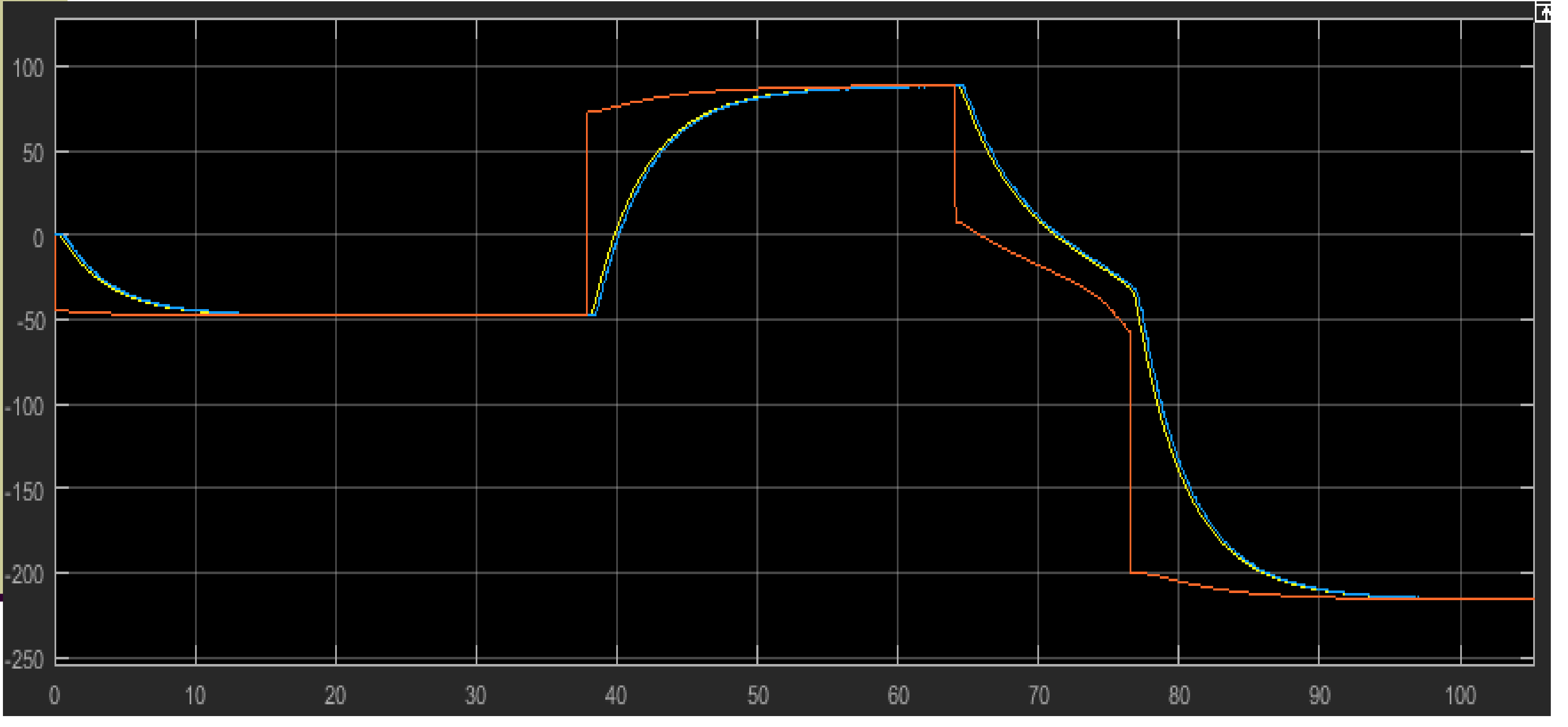
Speed Response

Actual, Estimated and Commanded

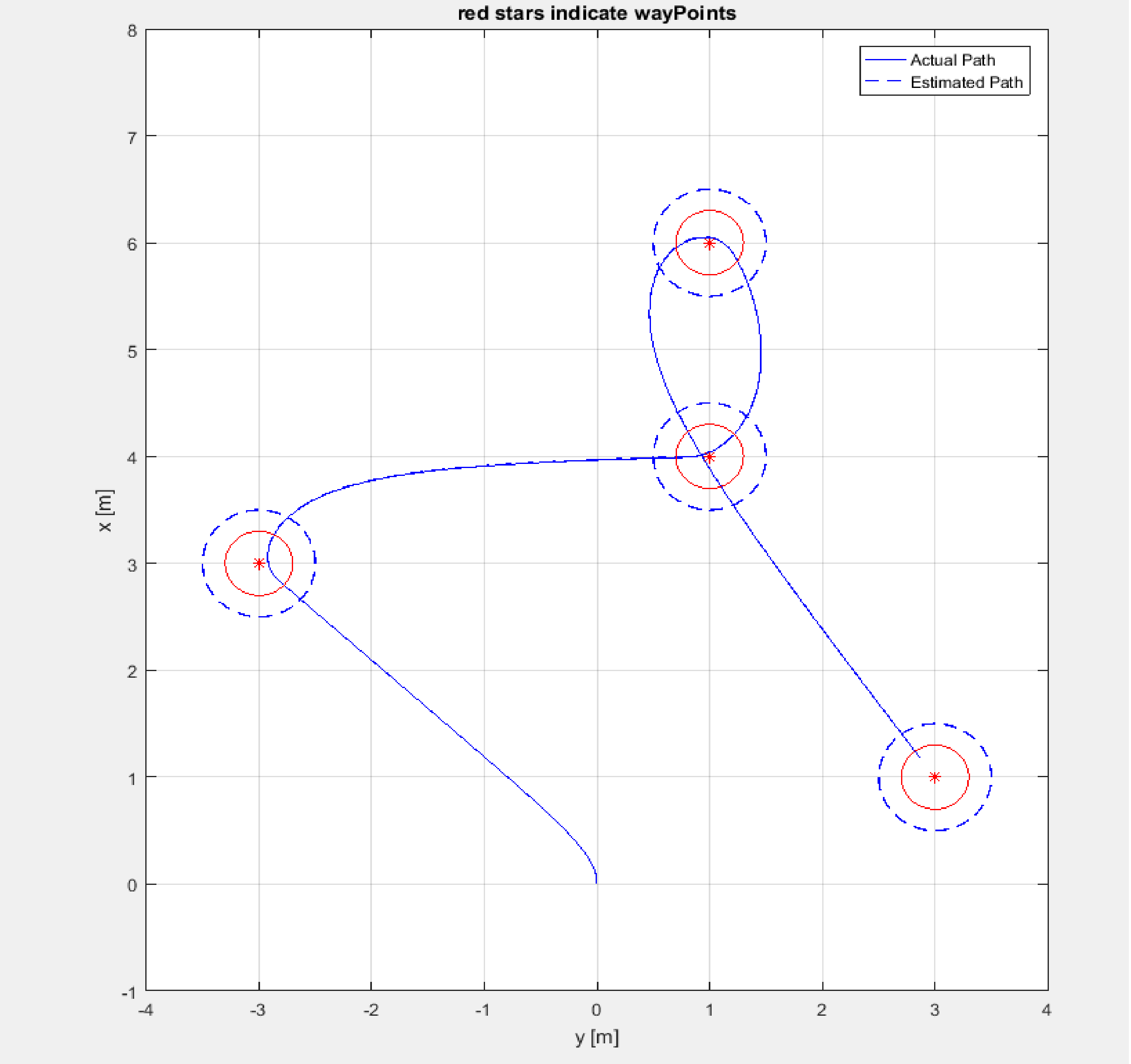


Angle Response

Actual, Estimated and Commanded



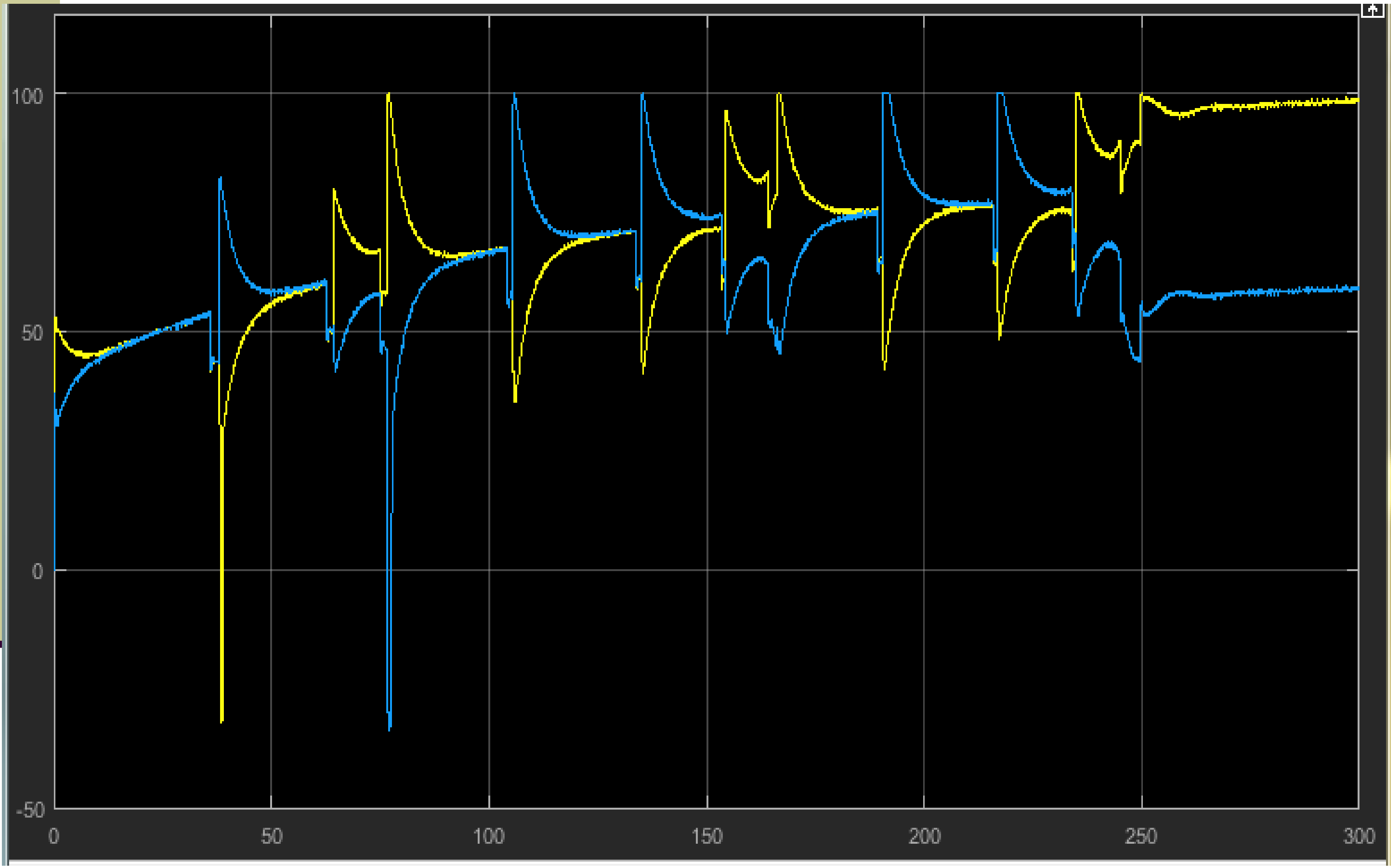
Trajectory Response



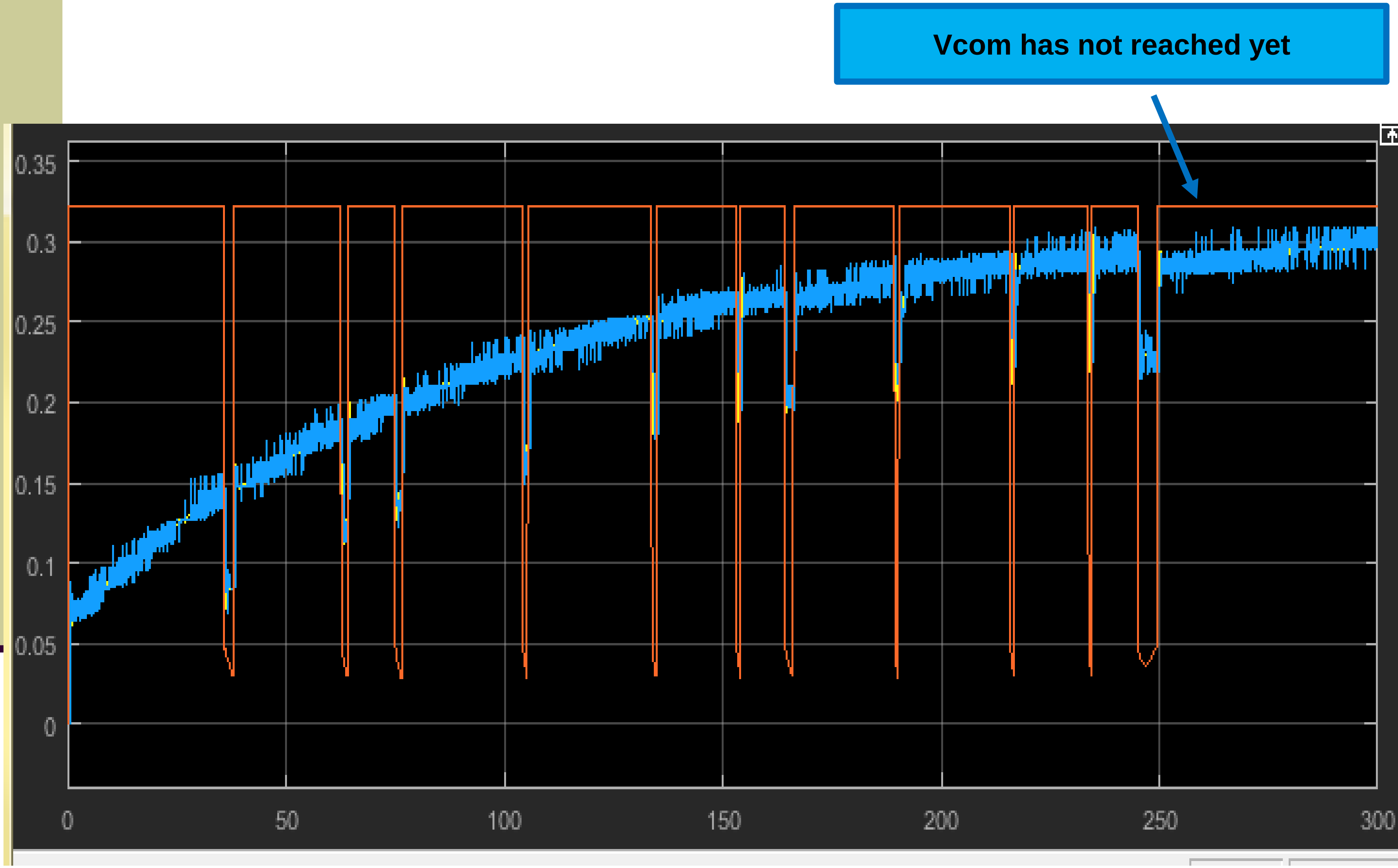
Simulation Case - 3

- slows down when close to wayPoint
- Does NOT stop at the last wayPoint
- loop through wayPoints
- Sim Stop time is 300 sec

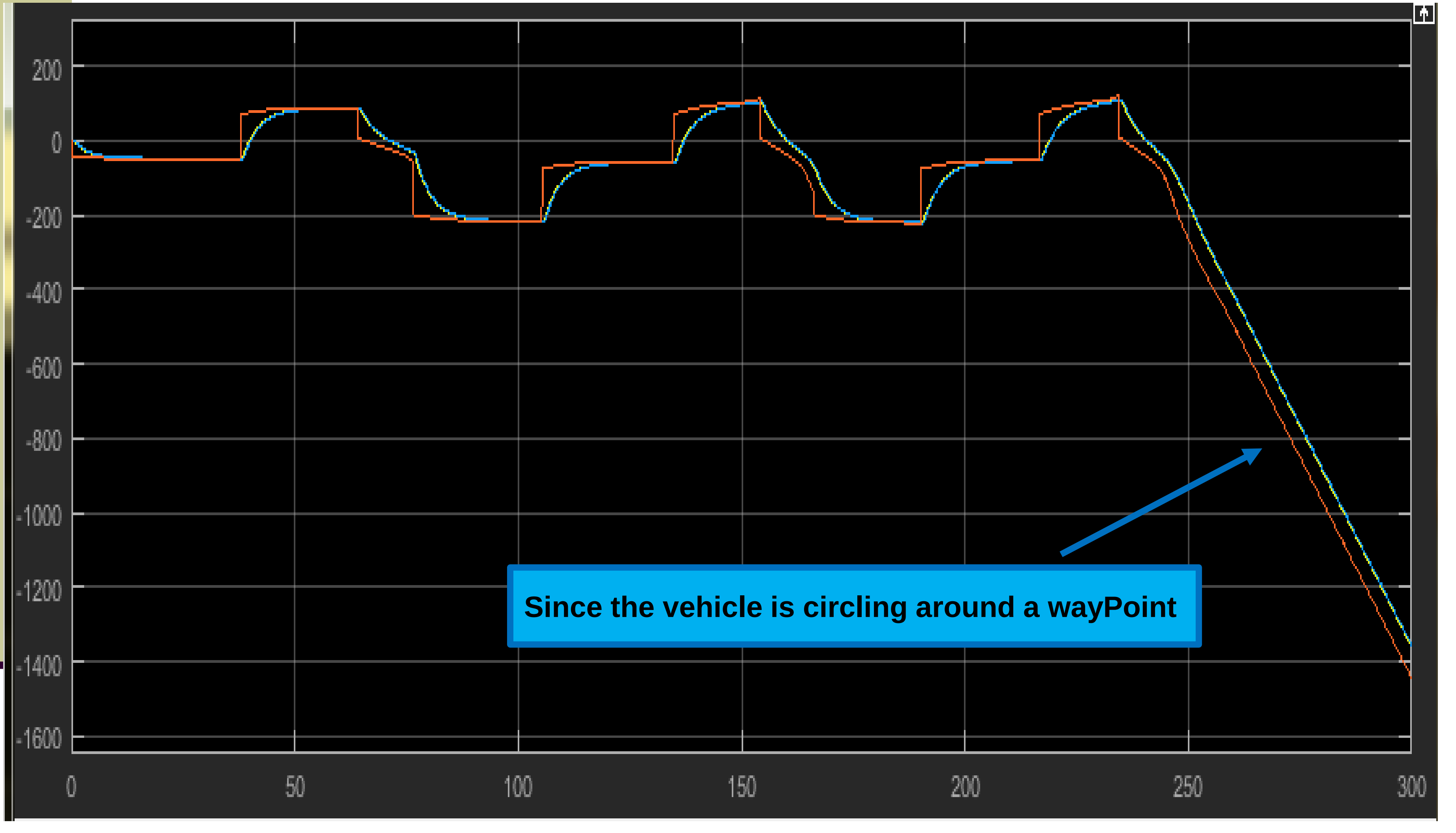
Duty Cycle Responses



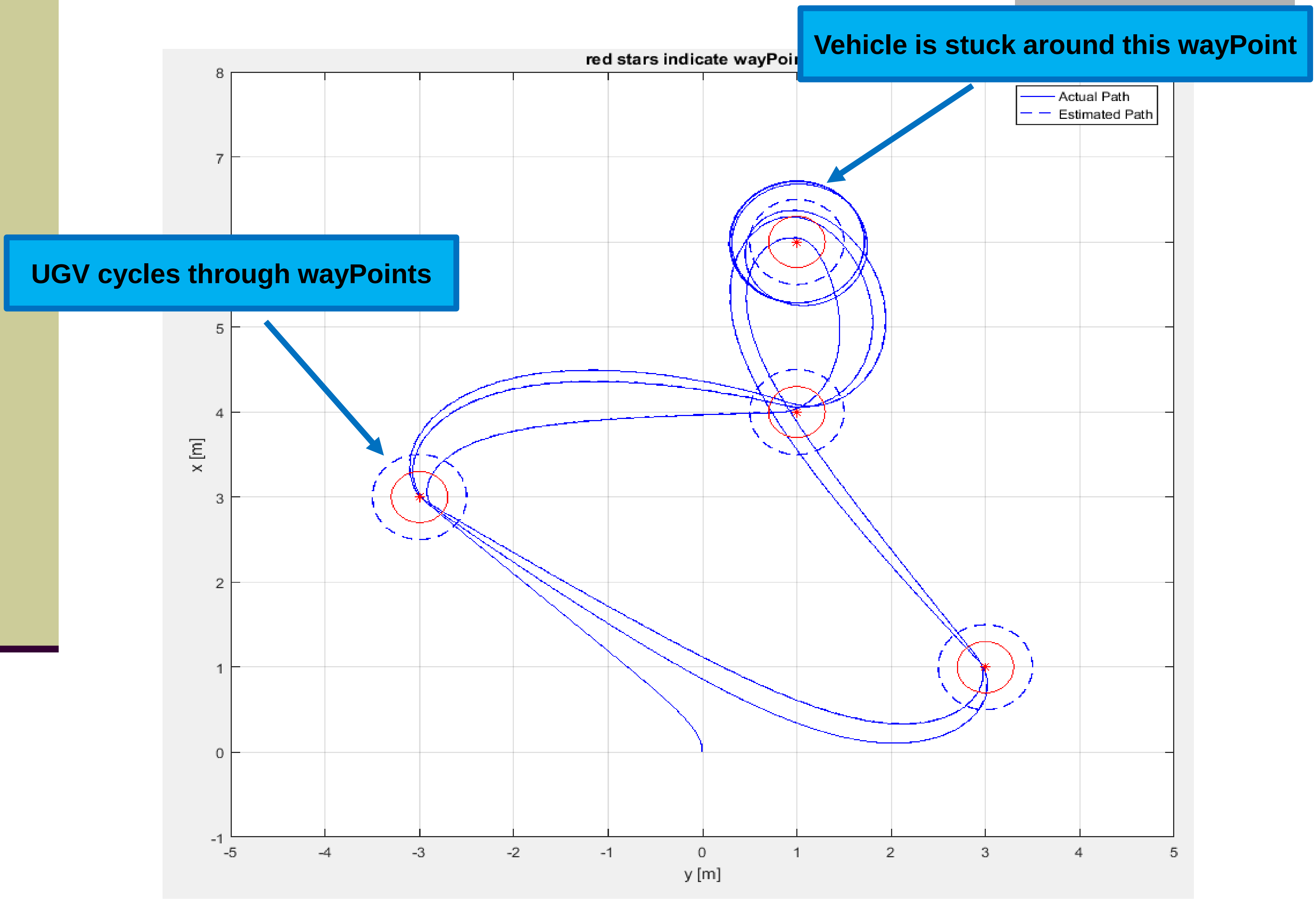
Speed Response



Angle Response



Trajectory Response



Simulation Case - 4

- slows down when close to wayPoint
- Does NOT stop at the last wayPoint
- loop through wayPoints
- Sim Stop time is 300 sec
- We will use different PID gains

Simulation Case - 4

- In parameterGNC.m
  - Gains for Angle Control are increased

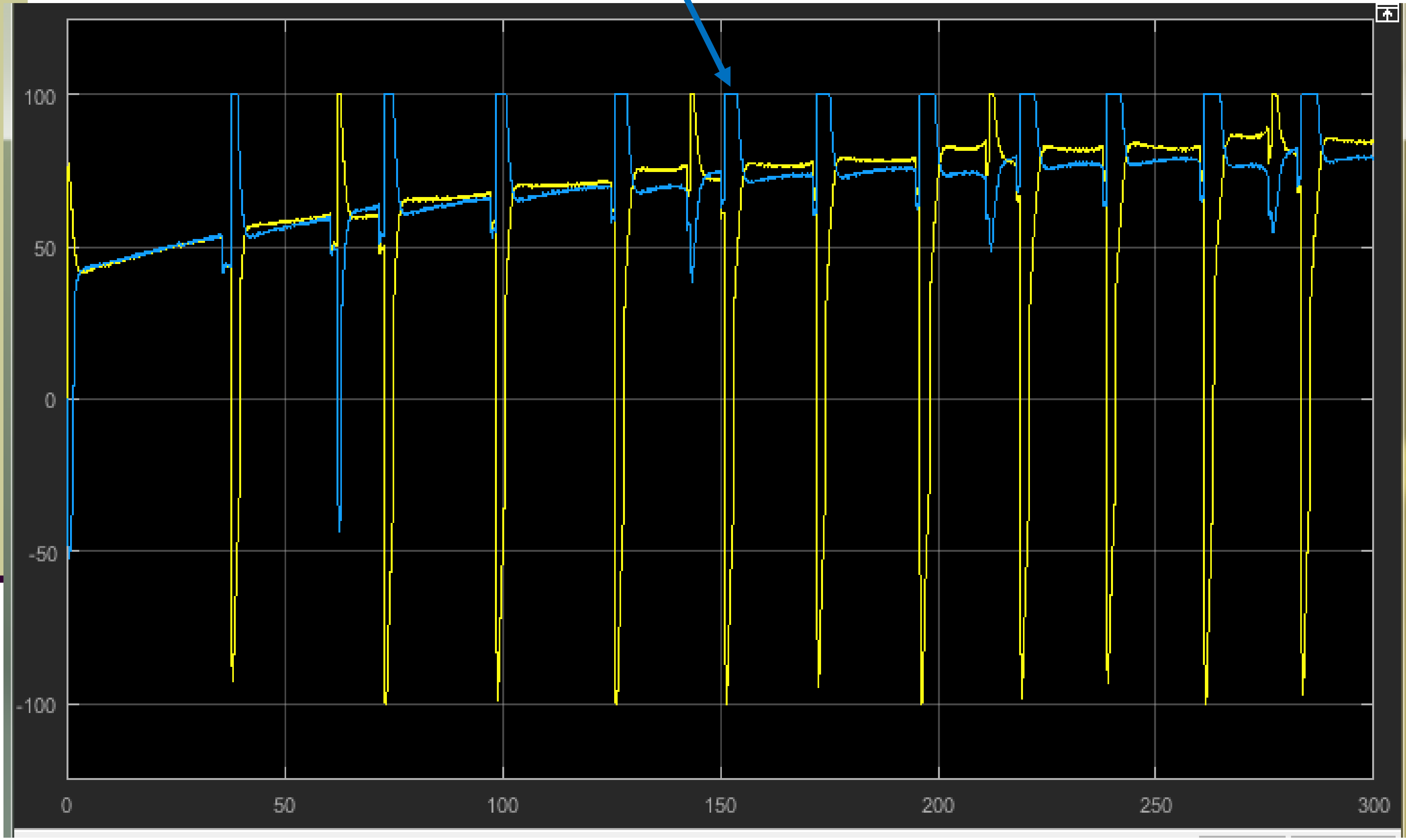
```
% Gains for
KP1=10; %20;
KP2=3;%10;
KI1=.5;%10;%
KI2=0.001; %
KD1=0; %3; %
KD2=0;% 3;%
```

NEW Gains

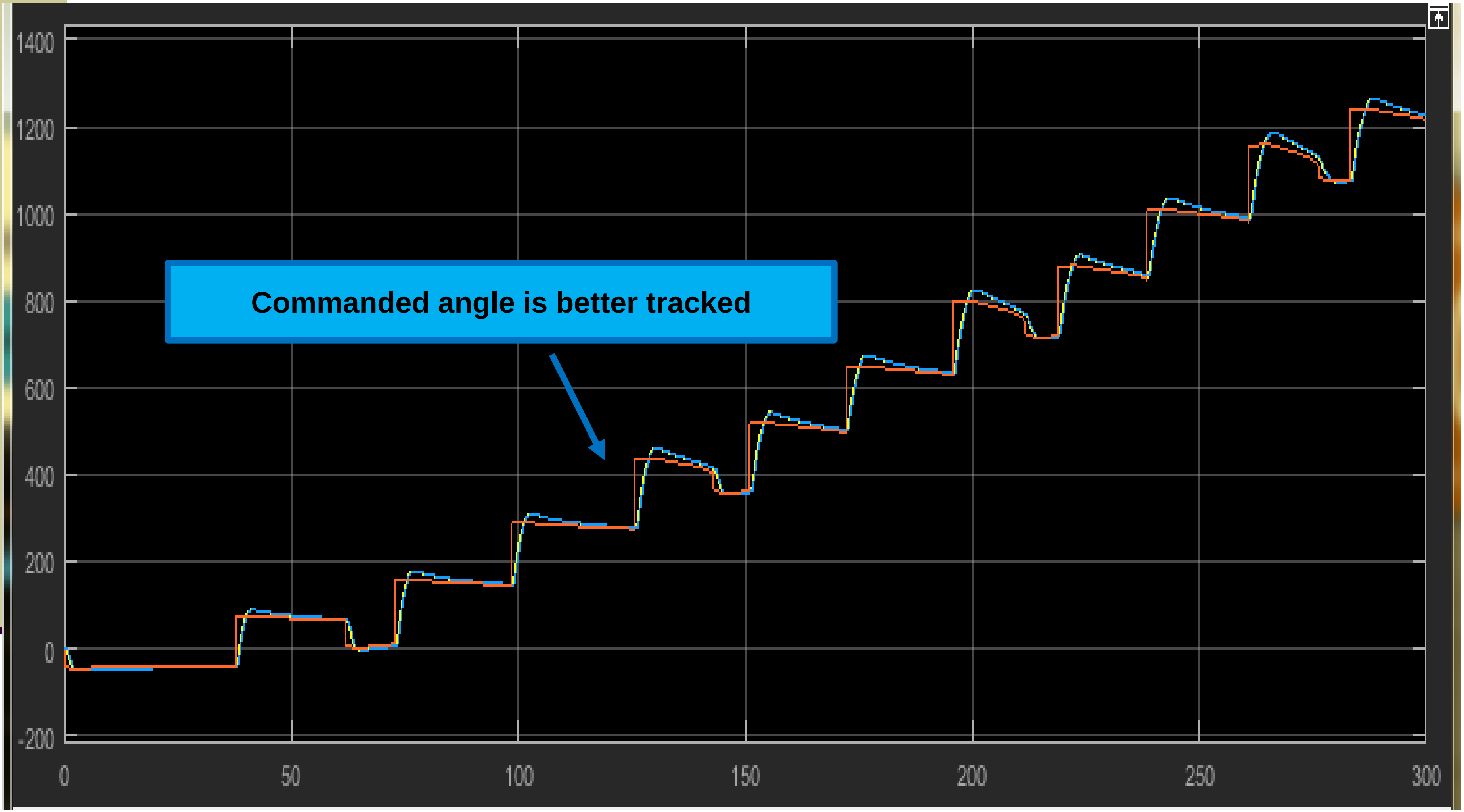
```
% Gains for
KP1=10; %20
KP2=10;%10;
KI1=.5;%10;
KI2=1; %0.0
KD1=0; %3;
KD2=0;% 3;%
```

Simulation Case - 4

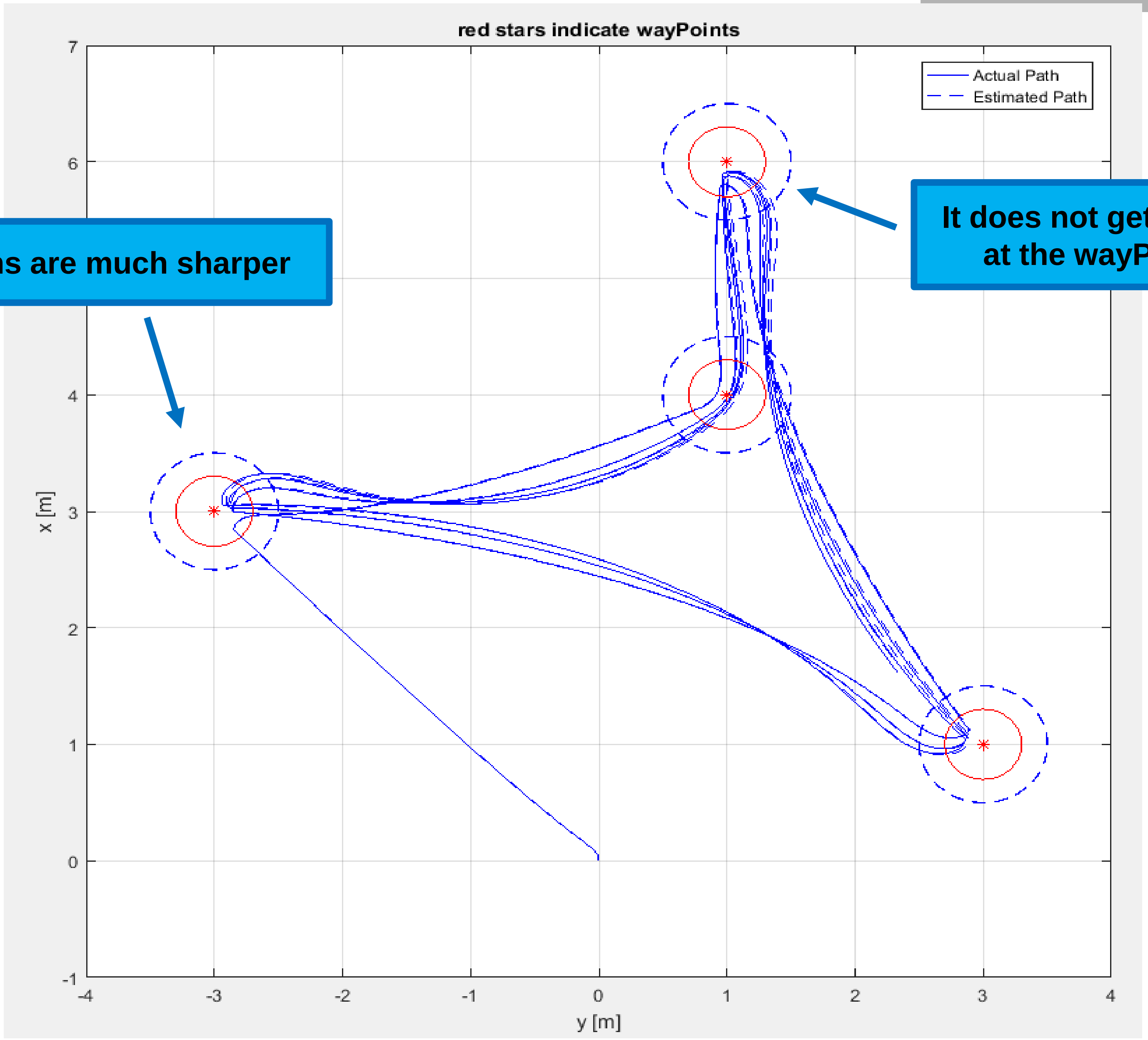
Larger %DutyCycles are commanded in turns



Simulation Case - 4



Simulation Case - 4



Turns are much sharper

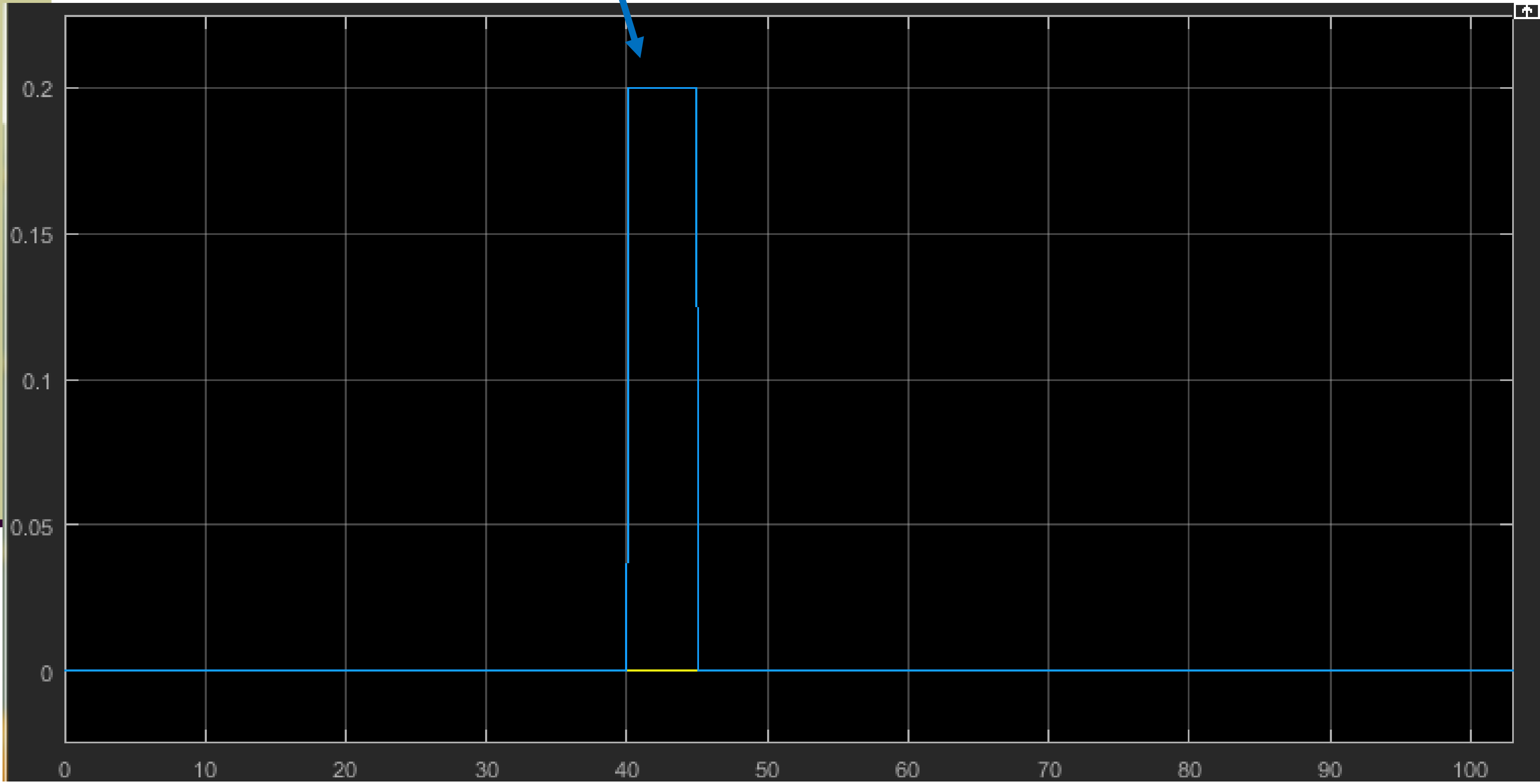
It does not get stuck at the wayPoint

Simulation Case - 5

- Switch back to the 1set set of PID gains
- Vcom = Vmax always,
- Stop at the last wayPoint
- Left Wheel Slips 20% from 40 sec to 45 sec

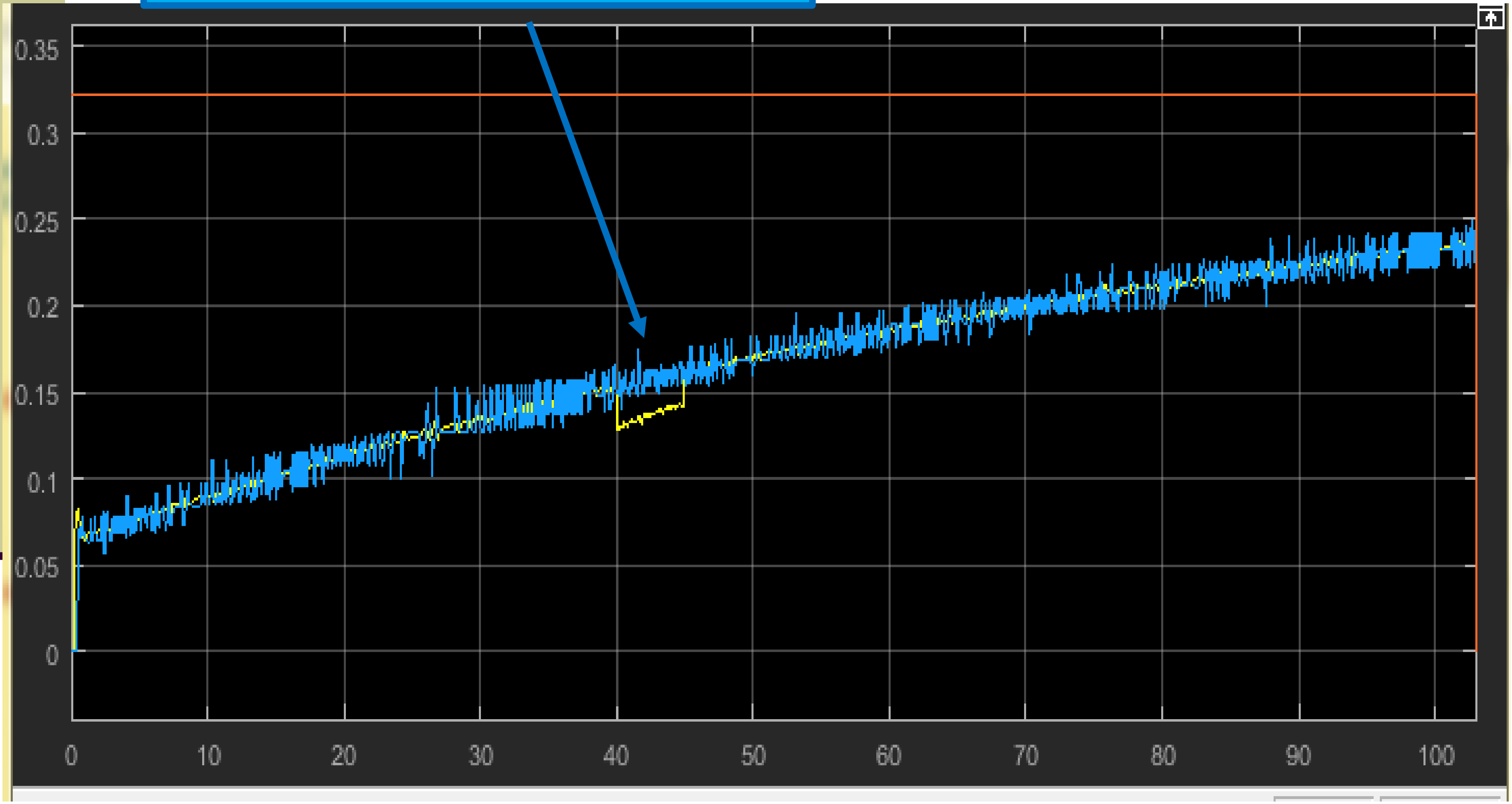
Slippage Factor

20% slippage on Left Wheel in 40-45 sec



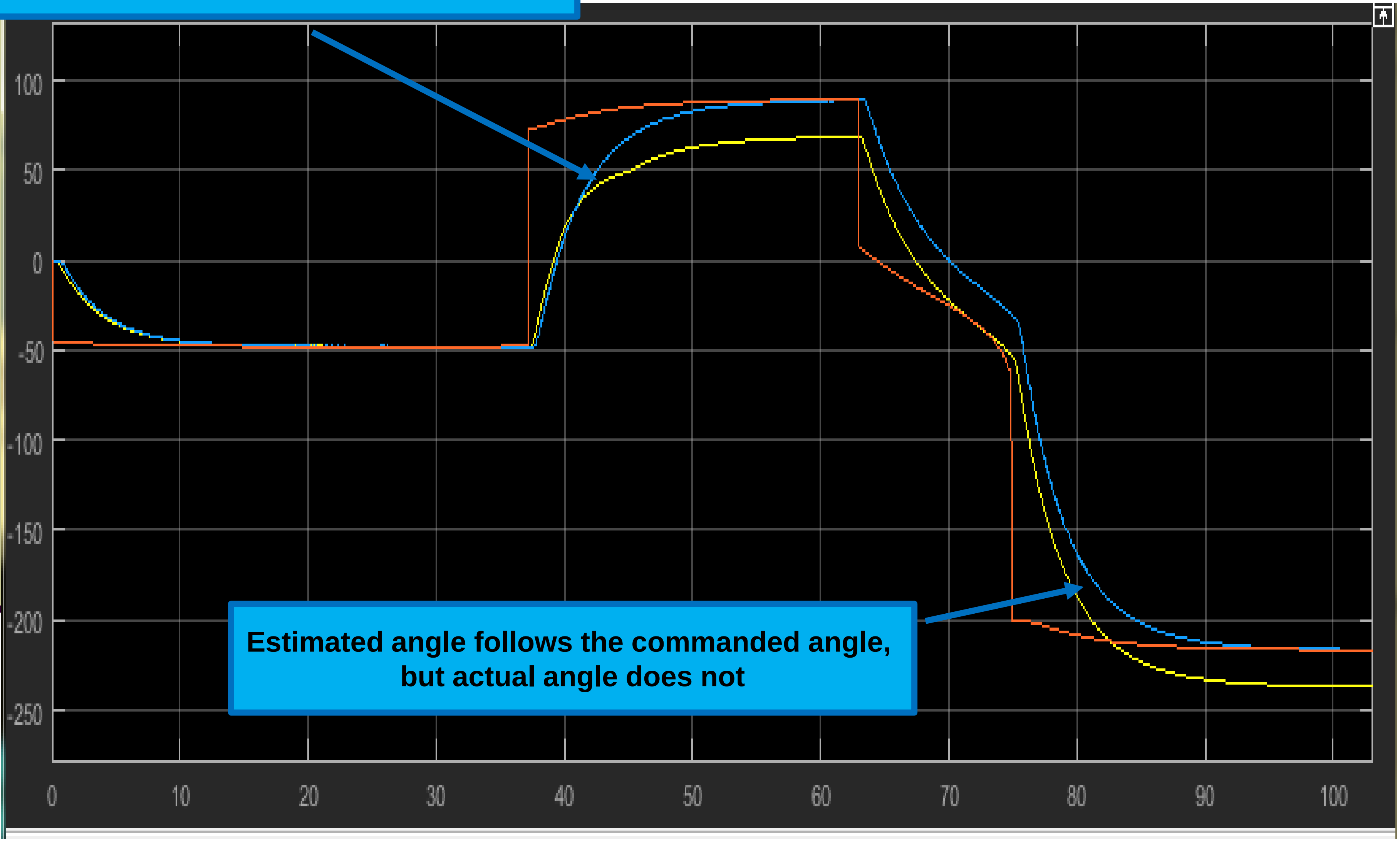
Speed Response

Speed estimates is wrong during slippage



Angle Response

Angle estimates is wrong during slippage

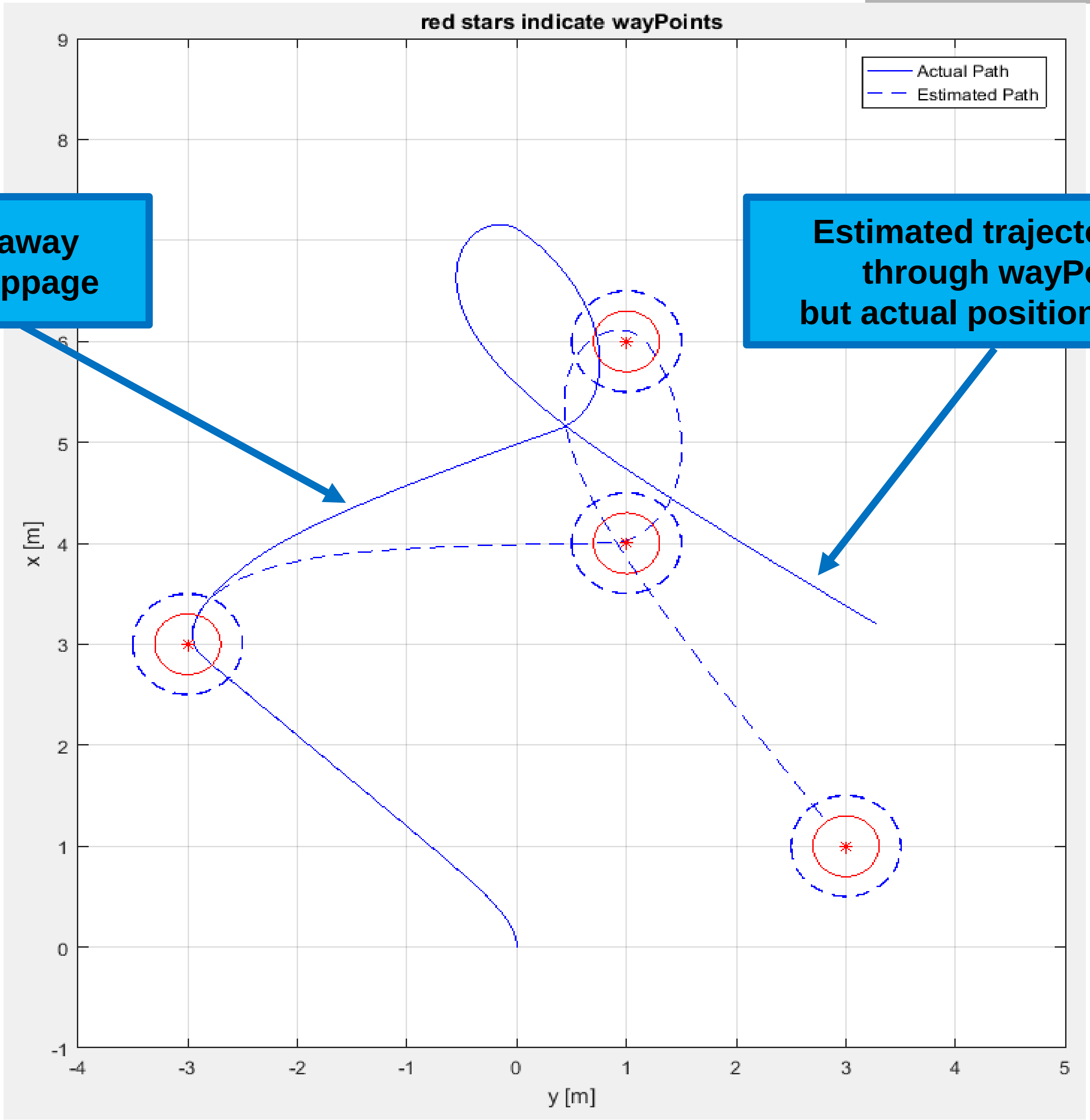


Estimated angle follows the commanded angle, but actual angle does not

Trajectory Response

Estimated trajectory drifts away
rom actual position after slippage

Estimated trajectory goes
through wayPoints,
but actual position does not



Simulation Case – 6 (WITH unwrap Angle)

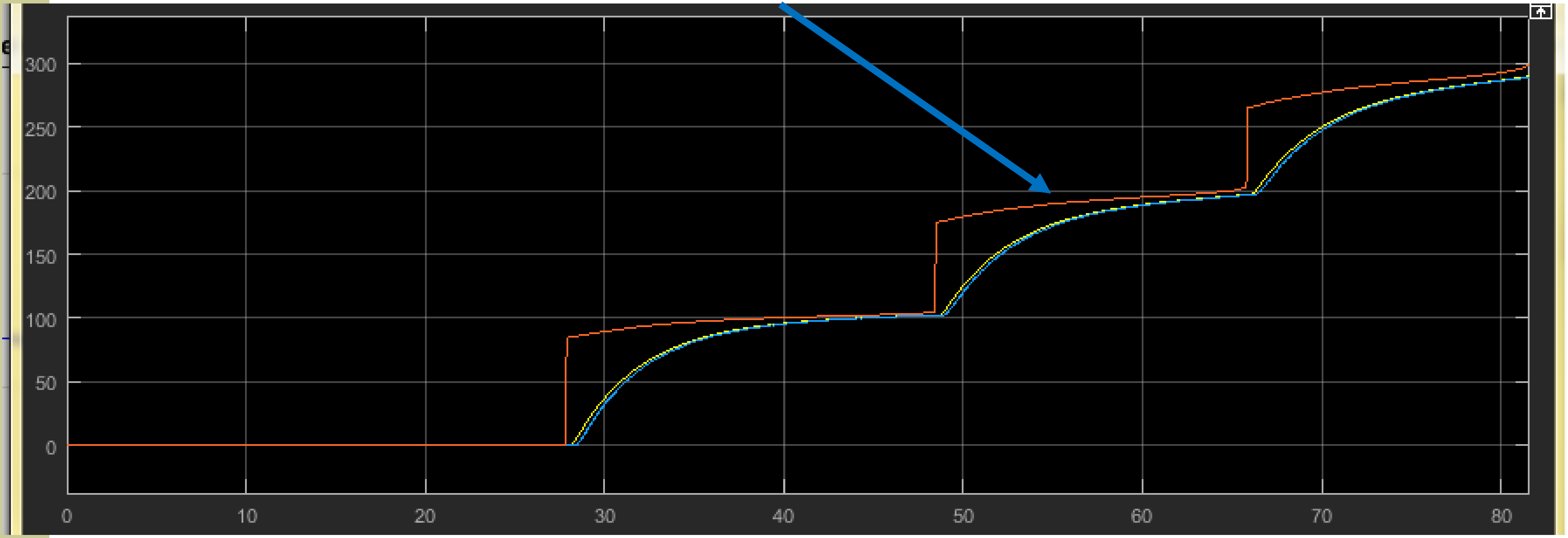
- Switch back to the 1set set of PID gains
- Vcom = Vmax always,
- Stop at the last wayPoint
- No Slippage
- New Set of Way Points

```
X_array  = [3  3  0  0];
Y_array  = [0  3  3  0];
```

- To show the importance of “Unwrap Angle” Block
- Change the sim end time to 100 sec

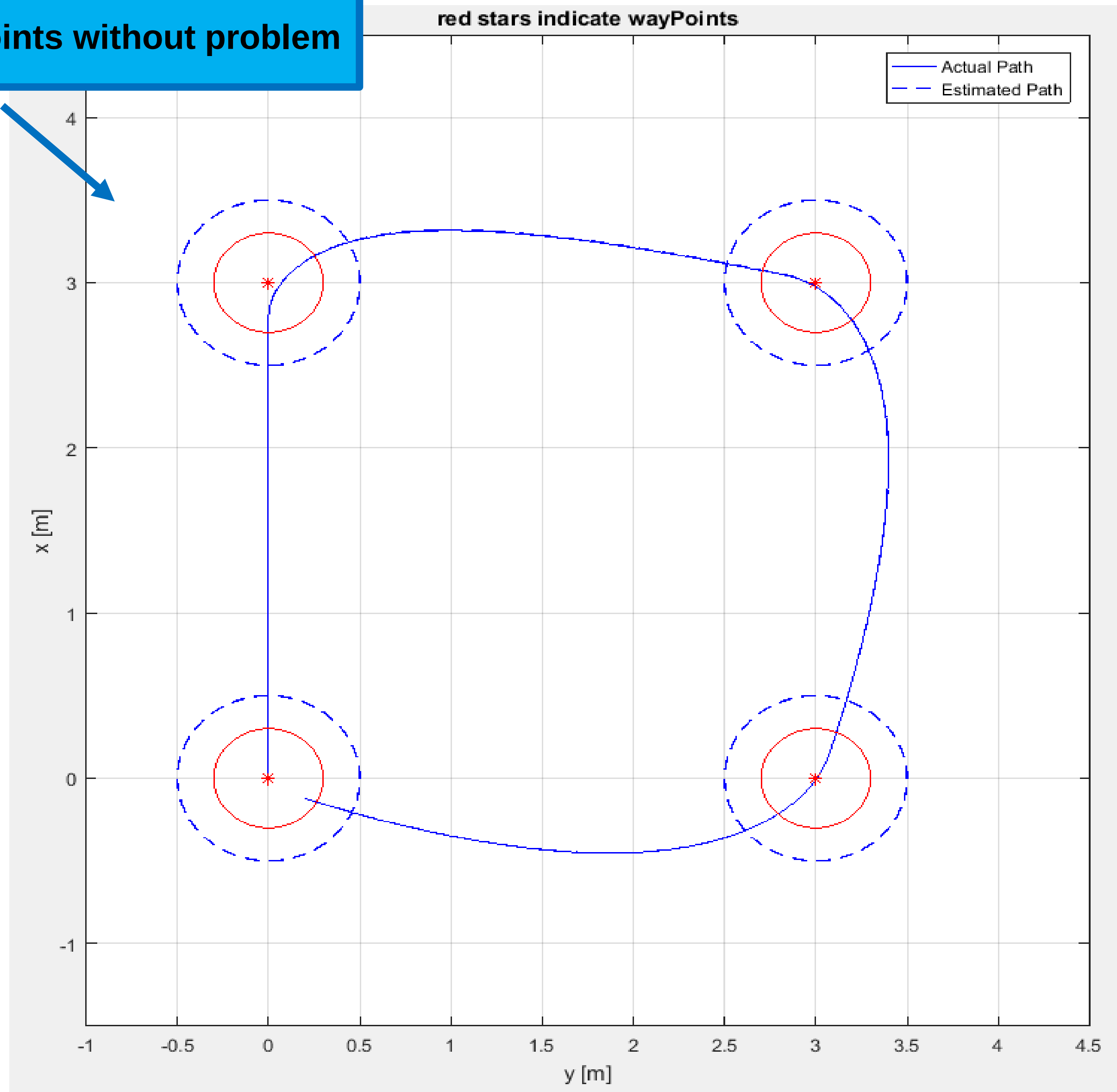
Simulation Case – 6 (WITH unwrap Angle)

Commanded Angle becomes larger than 180 deg



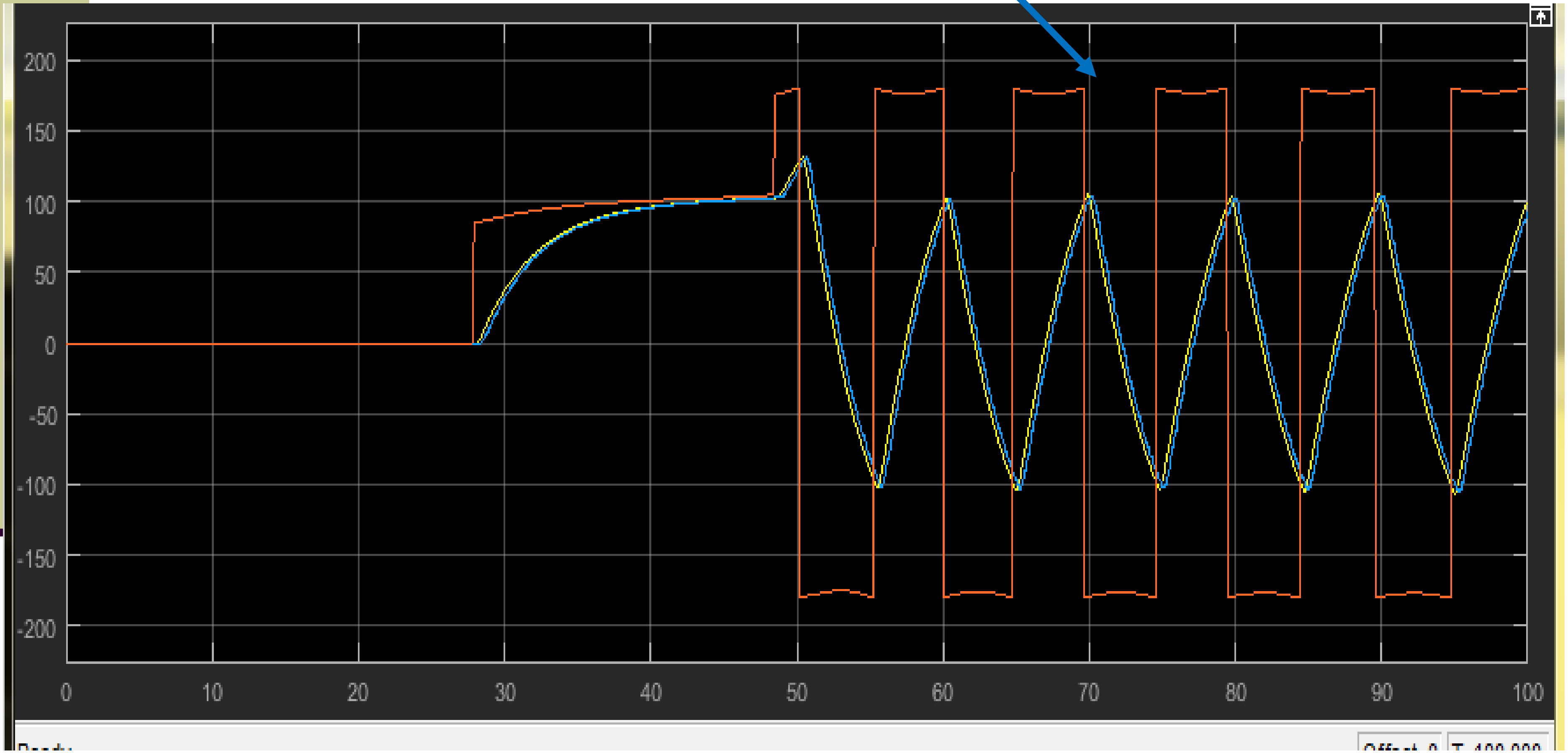
Simulation Case – 6 (WITH unwrap Angle)

Vehicle goes through wayPoints without problem



Simulation Case – 6 (NO unwrap Angle)

Commanded angle is limited between -180 & 180 deg
=>
There are jumps between -180 and 180 when angle reaches one of the limits



Simulation Case – 6 (NO unwrap Angle)

Vehicle cannot turn towards the last wayPoint because
the commanded angle switches between -180 & 180