
**A HEADING HOLD PID CONTROLLER
FOR
THE TCS L-BAND GROUND TARGETS**

Version 2

27 October 2008

System Engineering Directorates

Prepared by:
Luis E. Alvarado

The views, opinions and findings contained in this report are those of the author and should not be construed as an official Department of the Army position, policy or decision, or an official document of the Rhino Corps unless so designated by other documentation

Approved by: _____

Teofilo Holguin
Program Manager

OUTLINE

Contents

Contents

1. SCOPE.....	4
2. ACTUATOR MODEL	5
3. VEHICLE STEERING MODEL.....	7
2.1 Heading Rate Model.....	7
2.2 Power Steering Model	8
4. PID CONTROLLER DIAGRAM.....	8
5. PRESENT CONTROLLER.....	9
6. SIMULINK MODEL - PID CONTROLLER	10
7. SIMULINK MODEL – PRESENT CONTROLLER	13
8. STEP/RAMP RESPONSES PID & PRESENT CONTROLLER	14
8.1 PID Heading Response to a 20 degree heading step change	14
8.2 Present Controller Heading Response to a 20 degree heading change	15
8.3 PID Controller Heading Rate Response to a 20 degree heading change.....	16
8.4 Present Controller Heading Rate Response to a 20 degree heading change	17
8.5 PID Controller Actuator Response to a 20 degree heading change	18
8.6 Present Controller Actuator Response to a 20 degree heading change	19
8.7 PID Controller Heading Response to a RAMP heading change.....	20
8.8 Present Controller Heading Response to a RAMP heading change.....	21
8.9 PID Controller Heading Rate Response to a RAMP heading change.....	22
8.10 Present Controller Heading Rate Response to a RAMP heading change.....	23
9. DATA REQUIRED TO MODEL TANKS	24
9.1 Data Recorded at the VIU.....	24
9.2 Driving Tests.....	24
10. PID CONTROLLER SOURCE CODE.....	25

LIST OF FIGURES

Figure 1 Comparing Real versus Simulated Actuator	5
Figure 2 Actuator Model SIMULINK Diagram	7
Figure 3 PID Controller SIMULINK Diagram	9
Figure 4 Truck Steering SIMULINK Model - PID	11
Figure 5 Truck Steering SIMULINK Model - Present Control.....	13
Figure 6 PID Heading Response to 20 deg heading step command	14
Figure 7 Present Controller Heading Response to a 20 deg heading step command	15
Figure 8 PID Heading Rate Response to a 20 deg heading step command	16
Figure 9 Present Controller Heading Rate Response to 20 deg heading step command	17
Figure 10 PID Actuator Response to 20 deg heading step command.....	18
Figure 11 Present Controller Actuator Response to 20 deg heading step command.....	19
Figure 12 PID Heading Response to a RAMP heading command	20
Figure 13 Present Controller Heading Response to a RAMP heading command	21
Figure 14 PID Heading Rate Response to a RAMP heading command	22
Figure 15 Present Controller Heading Rate Response to a RAMP heading command	23

LIST OF TABLES

Table 1 Simulation of Steering Actuator.....	6
Table 2 Heading Rate Model Data.....	8
Table 3 Simulation of Present Controller	10
Table 4 Steering Model Simulation Parameters	12

1. SCOPE

A model based design approach was used to develop a new control law for the L-Band TCS Heading Hold Control loop. First we developed a SIMULINK model of the ground vehicle and implemented both the present heading hold control algorithm and a new heading hold PID control system. Next we compared the algorithm responses from both algorithms when they were excited with step and ramp input signals. This report describes the SIMULINK models that were used as well as the test results showing the performance of both algorithms under identical circumstances. This report also provides the source code for the new PID controller as well as all the source code developed to create the SIMULINK model.

The purpose of this report is to emphasize that we need to implement a new PID controller for the L-band TCS targets. This algorithm will be required if we want to successfully control non-wheeled vehicles such the T72 tank and the BMP personal carrier. The source code for the new PID controller is provided in section 10. I recommend that this algorithm is implemented first in the SEDAN and then in the tanks.

This report also shows the type of data that will be needed to model the T-72 tank and the BMP. Once we develop the SIMULINK models it should be somewhat easier to adapt the PID controller to the tank and BMP vehicles.

2. ACTUATOR MODEL

The first step was to develop a simulation model of the current steering servo actuator. Figure 1 shows steering step voltage commands (blue line) and the actuator response (red line). The black line shows the simulated response.

Figure 1 Comparing Real versus Simulated Actuator

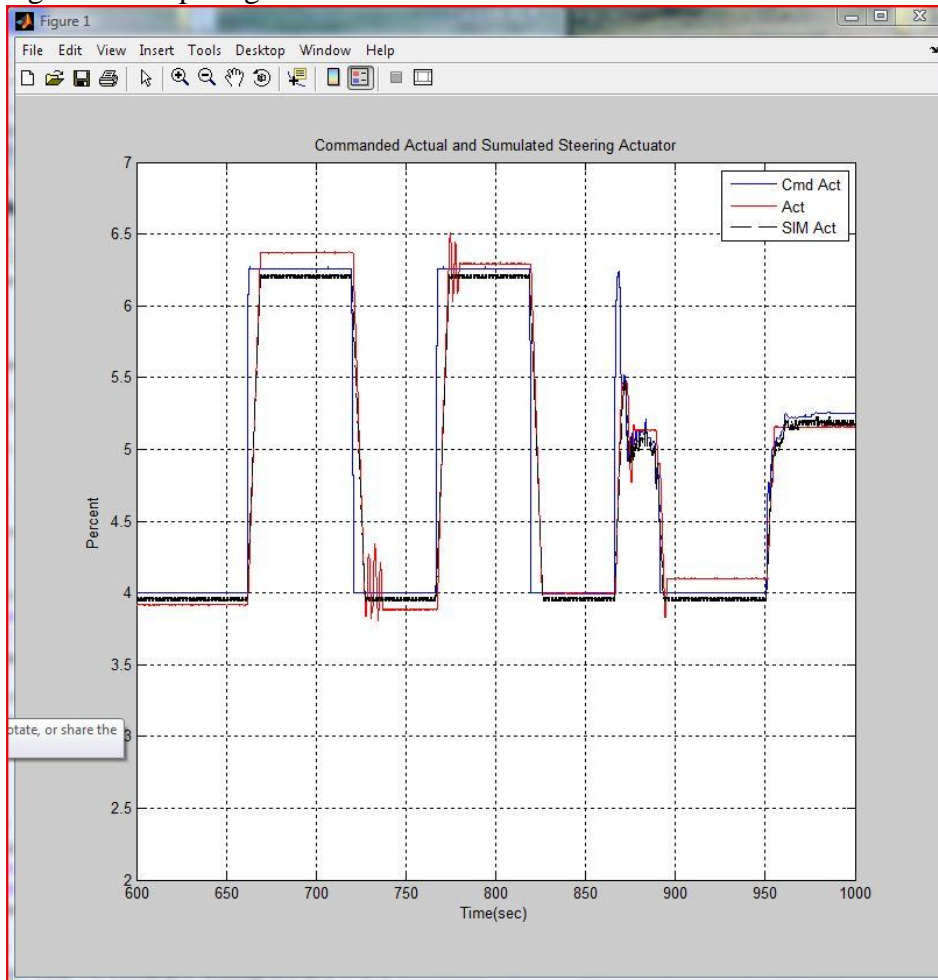
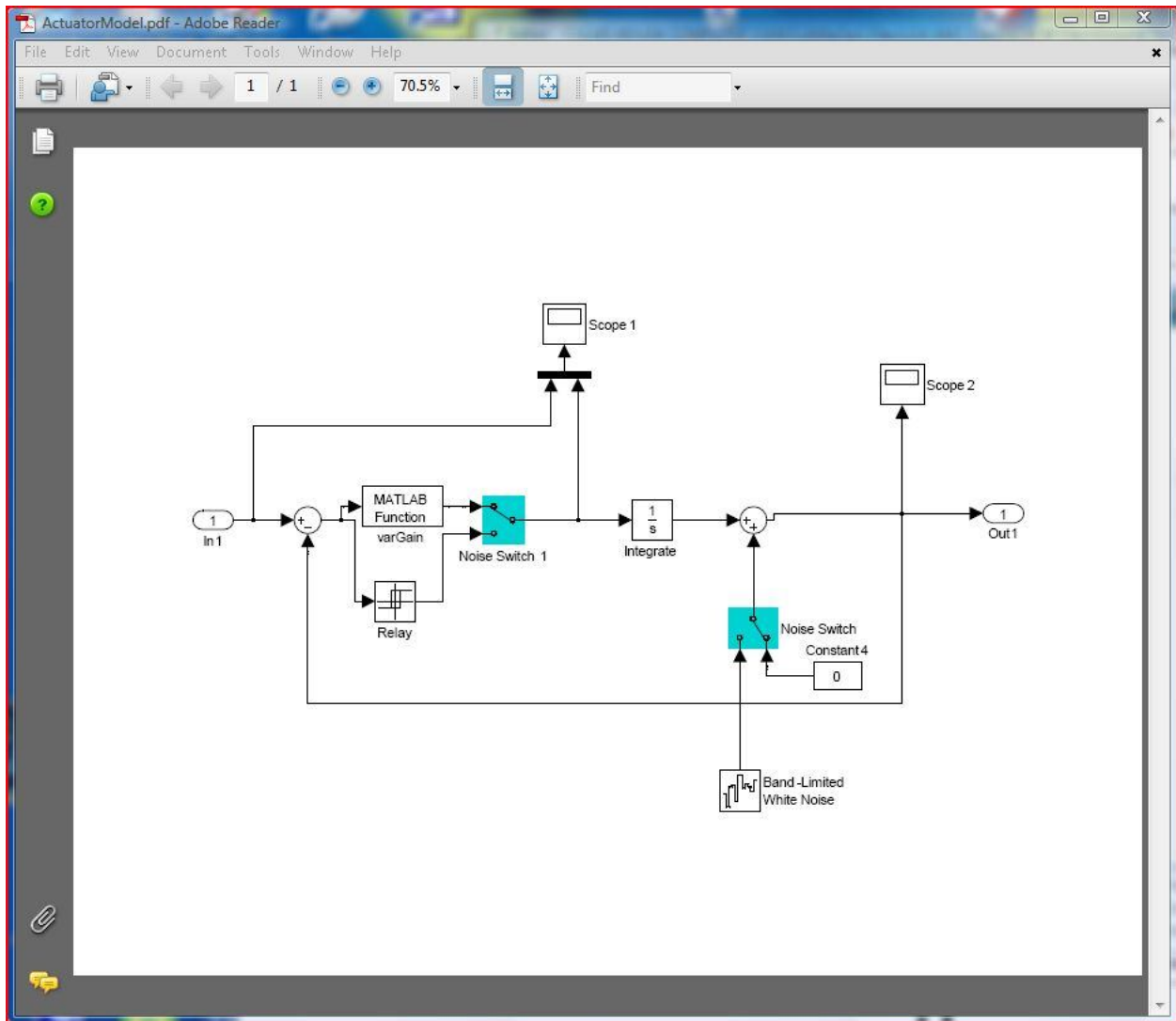


Table 1 Simulation of Steering Actuator

```
function var = varGain( error )
gain = 0.30; %Present servo controller
gain = 0.30;
if( error >= 0.01 )
    var = gain;
elseif( error < -0.01)
    var = -gain;
else
    var = 0;
end
end
```

Figure 2 shows the model used to simulate the steering actuator. The code depicted in Table 1 shows the code included in the box labeled MATLAB function and used to model the actuator incremental displacements. As depicted in Figure 2 these servo changes are integrated with time to compute the total servo displacement. The capability to add Gaussian noise to the actuator position was also added. This was done to simulate the poor control resolution of the present servo controller. (i.e. it commands 4.2 volts and the actuator moves to 4.25 or 4.15).

Figure 2 Actuator Model SIMULINK Diagram



3. VEHICLE STEERING MODEL

2.1 Heading Rate Model

Heading rate is inversely proportional to vehicle turn radius and directly proportional to ground speed as indicated in Equations 1 and 2.

$$\dot{\theta} = \frac{v}{r} \quad (1)$$

$$h = 57.3 \frac{v}{r} \quad (2)$$

Table 2 shows the data collected for one of the trucks. The first column shows steering voltage and the second column shows the truck turn radius for that particular steering voltage. The last

column indicates the turn slope, that is the ratio between the inverse of the turn radius and the steering voltage multiplied by 57.3.

Table 2 Heading Rate Model Data

Steering	Turn Radius	57.3/Turn Radius	Turn Slope
5.15 volts	75 feet	0.764	0.764/(5.15-4) = 0.663
6.30 volts	32.5 feet	1.763	1.763/(6.3-4) = 0.766

Equation 3 shows that heading rate can now be calculated by multiplying the computed turn slope times the ground speed in feet per second times the steering voltage.

$$\dot{\theta} = (\text{Turn Slope}) * (\text{Ground Speed} - 4) * (\text{Steering Voltage}) \quad (3)$$

2.2 Power Steering Model

A simple lag of 0.6 seconds was used to model the tire rotational delay caused by the power steering.

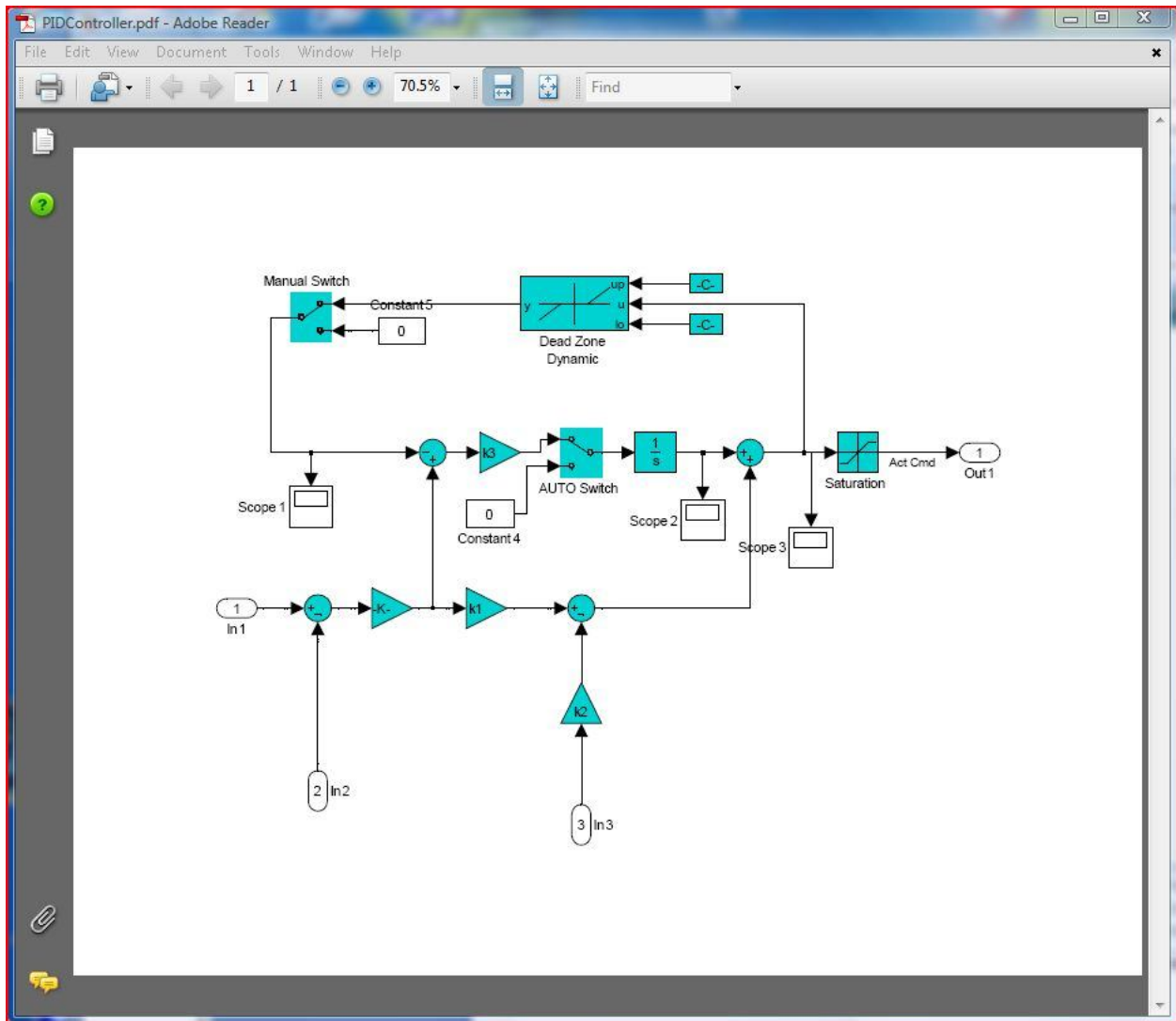
$$\tau = 2 * 0.6$$

+

4. PID CONTROLLER DIAGRAM

Figure 3 shows the block diagram of the proposed PID controller. The PID controller compares the heading command with current IMU heading to derive a heading error. The PID controller uses IMU yaw rate to provide some lead. The PID also integrates the heading error to compensate for steering biases. The “Dead Zone” feedback is used to make sure that the integrator is not wound up. Notice also that the integrator is “frozen”, that is, the input to the integrator is zeroed out, when the onboard system receives an AUTO uplink from the ground station. When the target is in AUTO mode, the ground software will be the one that detects and eliminates heading steering biases. We do not want the onboard integrator and the ground software control integral fighting each other.

Figure 3 PID Controller SIMULINK Diagram



5. PRESENT CONTROLLER

Table 3 shows the code for the present HH control loop. The steering command is generated based on heading error. If the heading error exceeds 4 degrees a steering increment voltage of 0.25 is commanded. If the heading error is between 3 and 4 degrees a steering increment voltage of 0.20 is commanded and so forth. Notice also that if the heading error is less than 0.5 degrees no action is taken. Eventually this artificial heading “bias” will make the vehicle diverge from center line until a heading change greater than 0.5 degrees is detected. Furthermore, if the steering dead zone exceeds 0.25 volts, (very common in tanks) the current algorithm will not steer the vehicle.

Table 3 Simulation of Present Controller

```
function steeringCmd = propController( error )

Tolerance1 = 4.0;           %tight tolerance
Tolerance2 = 3.0;           %tight tolerance
Tolerance3 = 2.0;           %tight tolerance
Tolerance = 0.5;           %degrees tight

absDelta=abs(error);
if( absDelta > Tolerance1 )
    steering = 0.25;
elseif( absDelta > Tolerance2 )
    steering = 0.20;
elseif( absDelta > Tolerance3 )
    steering = 0.15;
elseif( absDelta > Tolerance )
    steering = 0.10;
else
    steering= 0.0;
end

if( error >= 0 )
    steeringCmd = steering + 4;
else
    steeringCmd = 4 - steering;
end
[steeringCmd error]
end
```

6. SIMULINK MODEL - PID CONTROLLER

Figure 4 shows the block diagram for the proposed heading hold steering control system. It includes the PID controller, the actuator model and the tank steering dynamics. Table 4 shows simulation parameters such as PID control gains, center steer voltage, steering voltage limits, turn slope, and ground vehicle speed in miles per hour.

Figure 4 Truck Steering SIMULINK Model - PID

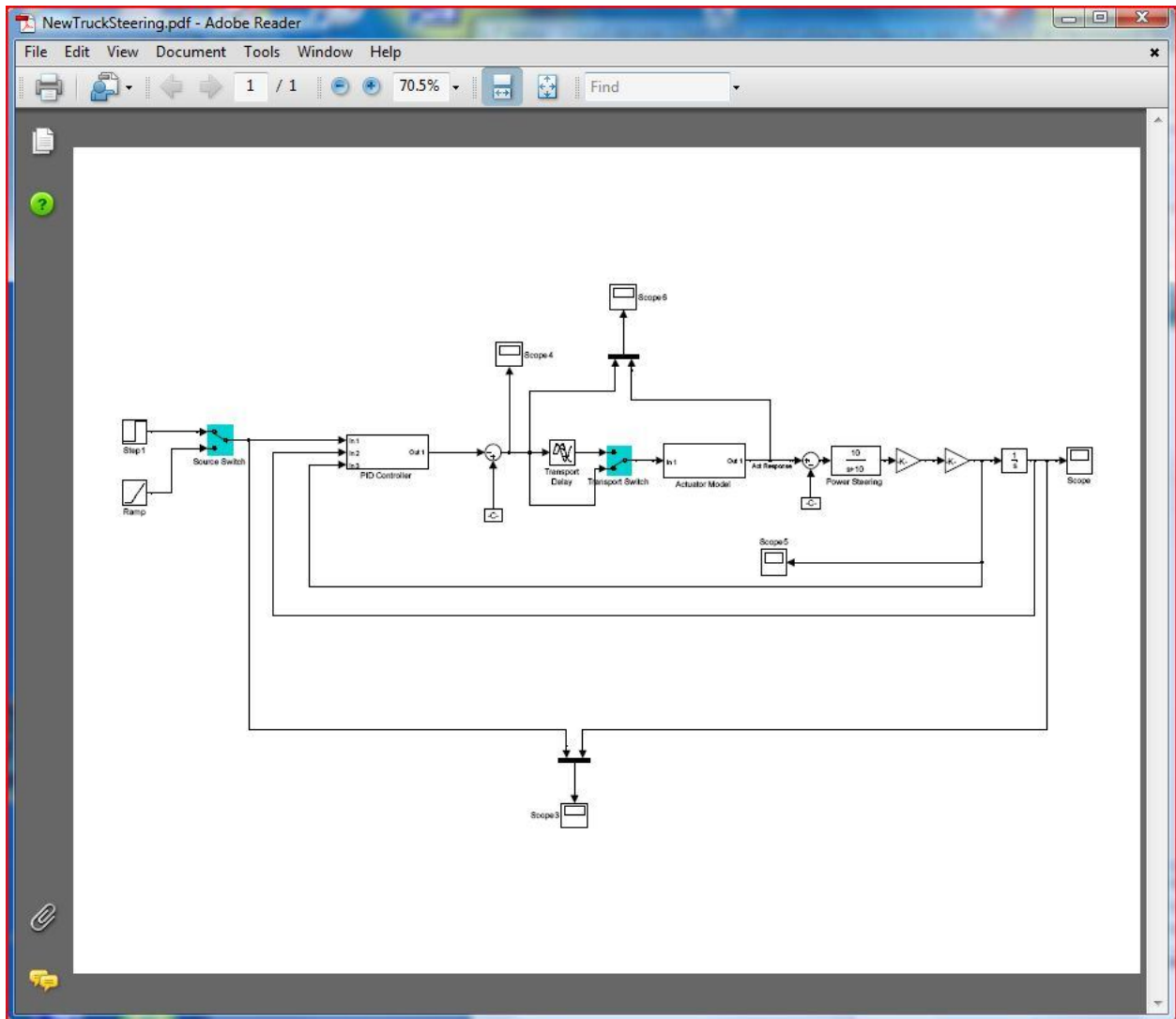


Table 4 Steering Model Simulation Parameters

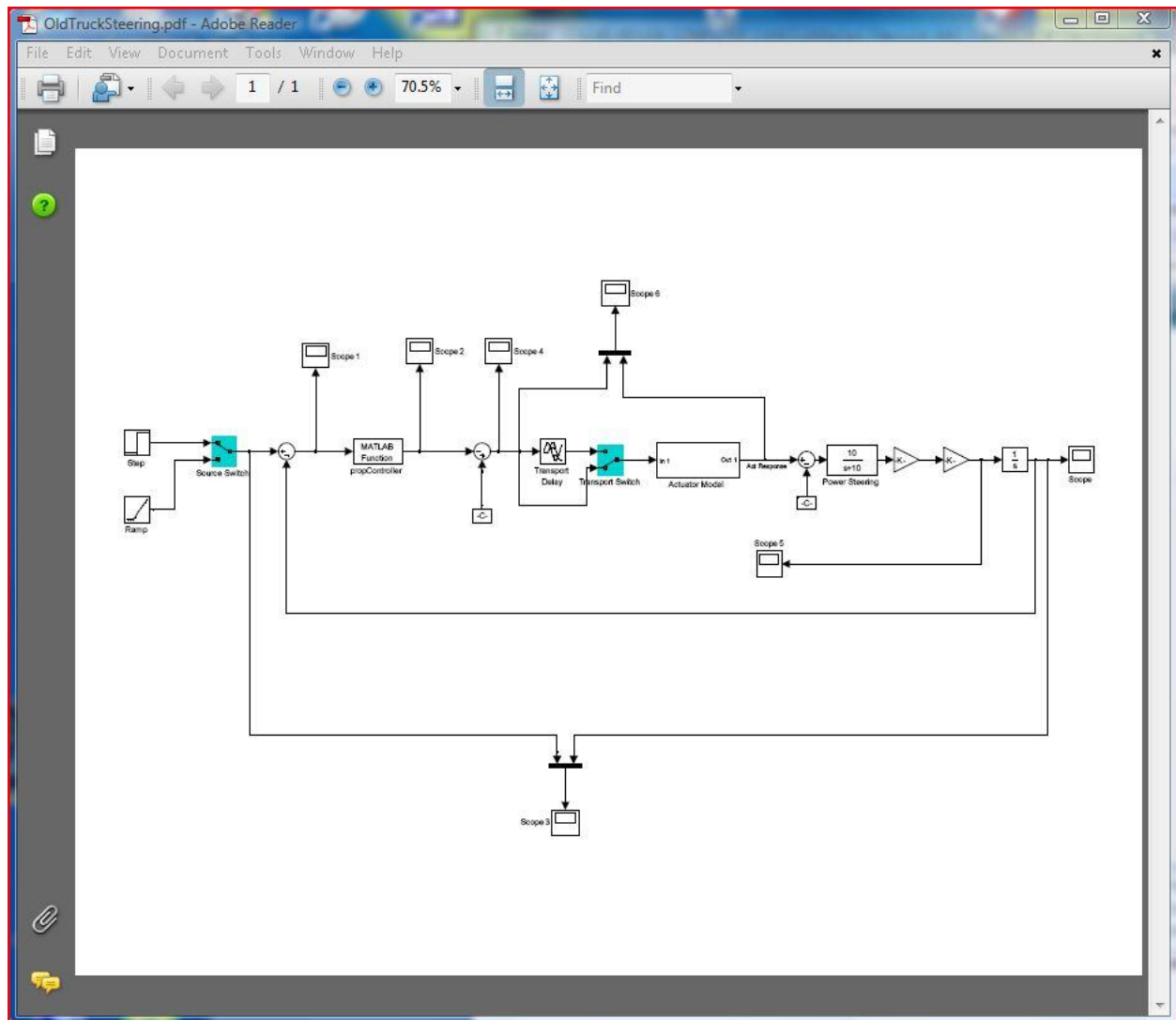
```
speedMPH = 60; % ground speed
groundSpeed = speedMPH * 1609/3600; % speed in meters per second
met2ft = 3.28;
ktotal=1.0;

k1=0.10; % Volts / deg-error
k2=0.35; % Volts / deg/sec
%k2=0;
k3=0.005; % Volts / deg-error per second
%k3=0;
k2=ktotal*k2;
%centerSteer = 4.25; %Volts
centerSteer = 4.10;
centerVoltage = 4.0;
deadLowerLimit = centerVoltage - 0.25;
deadUpperLimit = centerVoltage + 0.25;
steerLowerLimit = 0.5; % Volts
steerUpperLimit = 7.5; % Volts
turnSlope = 0.766 * met2ft; % degrees per second / feet-per-sec
```

7. SIMULINK MODEL – PRESENT CONTROLLER

Figure 4 shows the block diagram for the current heading hold steering system. It includes the present controller, the actuator model and the tank steering dynamics. Table 4 shows simulation parameters such as PID control gains, center steer voltage, steering voltage limits, turn slope, and ground vehicle speed in miles per hour.

Figure 5 Truck Steering SIMULINK Model - Present Control

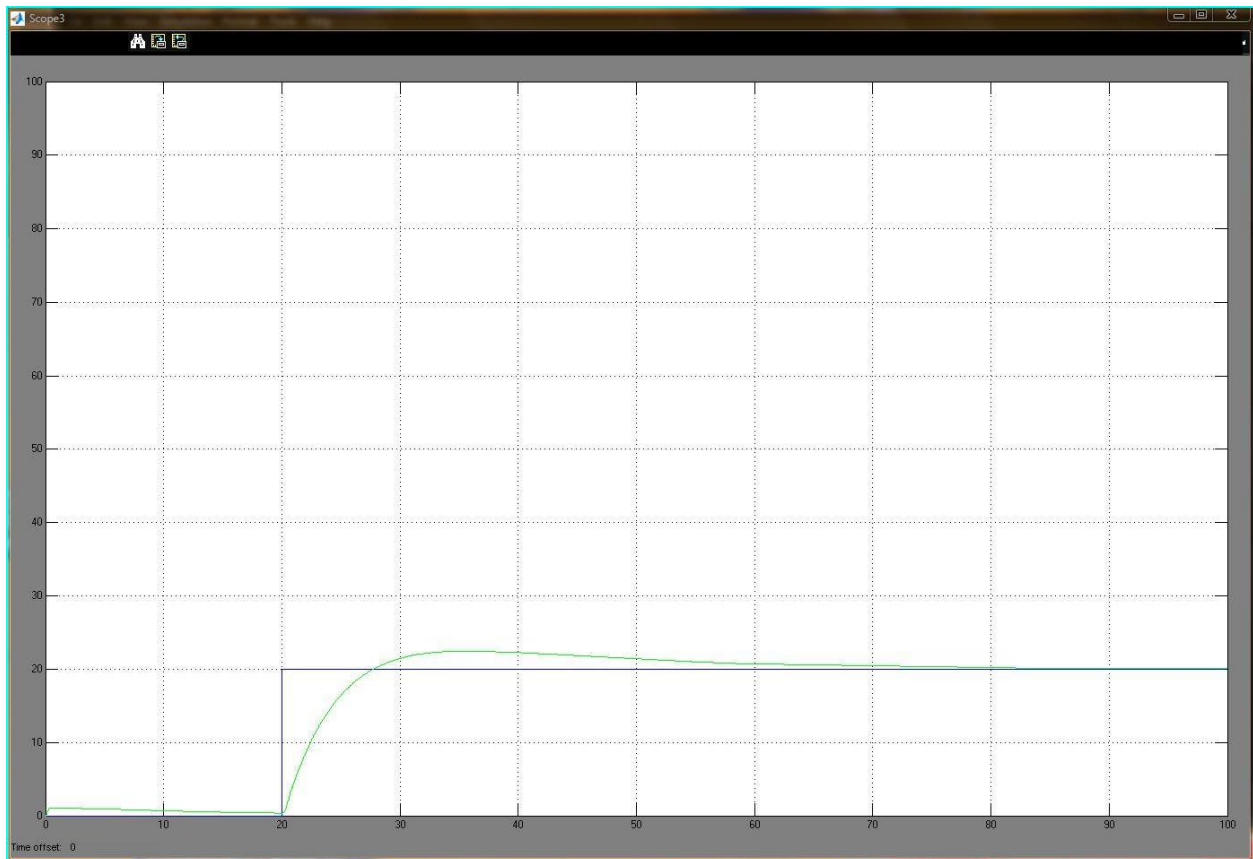


8. STEP/RAMP RESPONSES PID & PRESENT CONTROLLER

8.1 PID Heading Response to a 20 degree heading step change

Figure 6 shows the response of the new PID HH system to a step heading change of 20 degrees. Notice the rise time of the response is about 9 seconds and there are no heading biases.

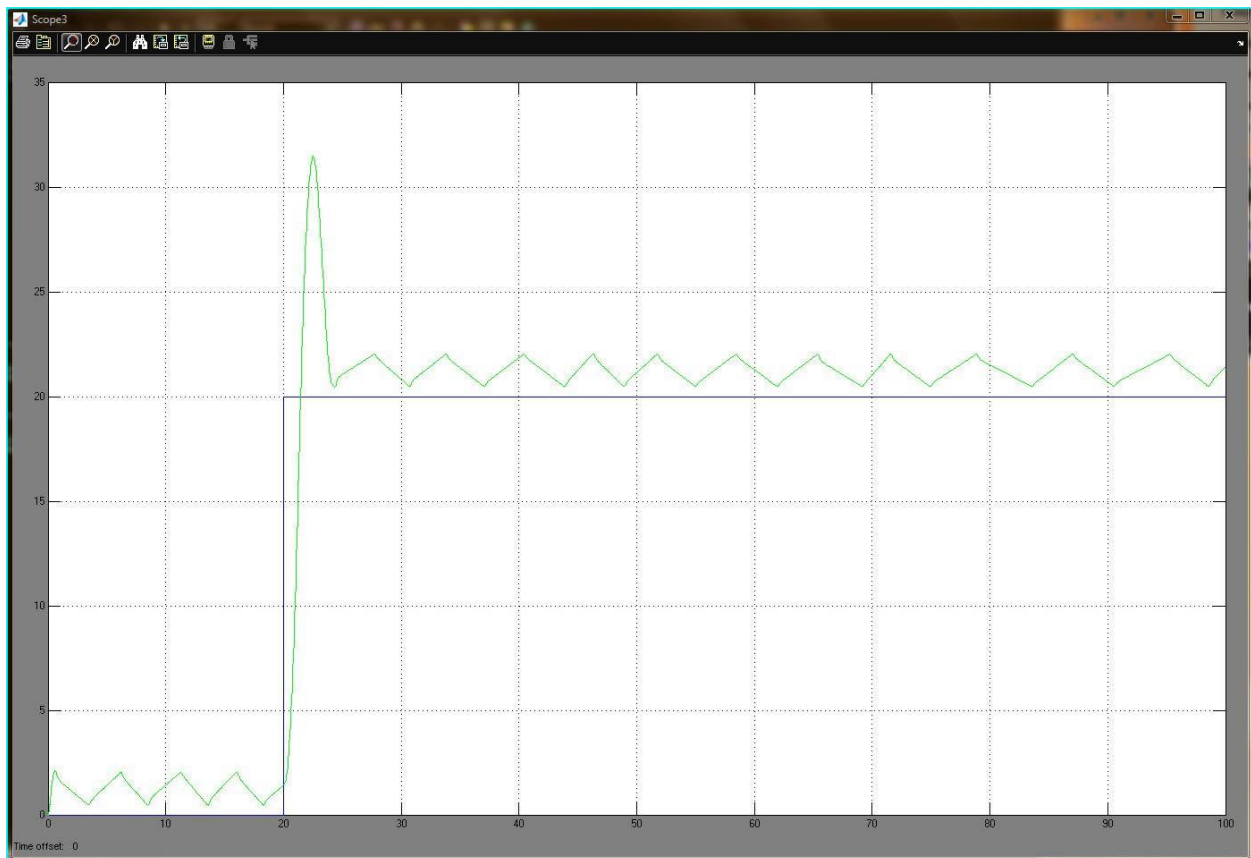
Figure 6 PID Heading Response to 20 deg heading step command



8.2 Present Controller Heading Response to a 20 degree heading change

Figure 6 shows the response of the present HH system to a step heading change of 20 degrees. Notice the rise time is very fast (no heading rate damping) and there is 50% overshoot. Notice also the small amplitude heading oscillation and the constant heading error. This behavior is caused by the simulated steering bias of 0.1 volt. The algorithm cannot properly eliminate the steering bias and it is constantly fighting it.

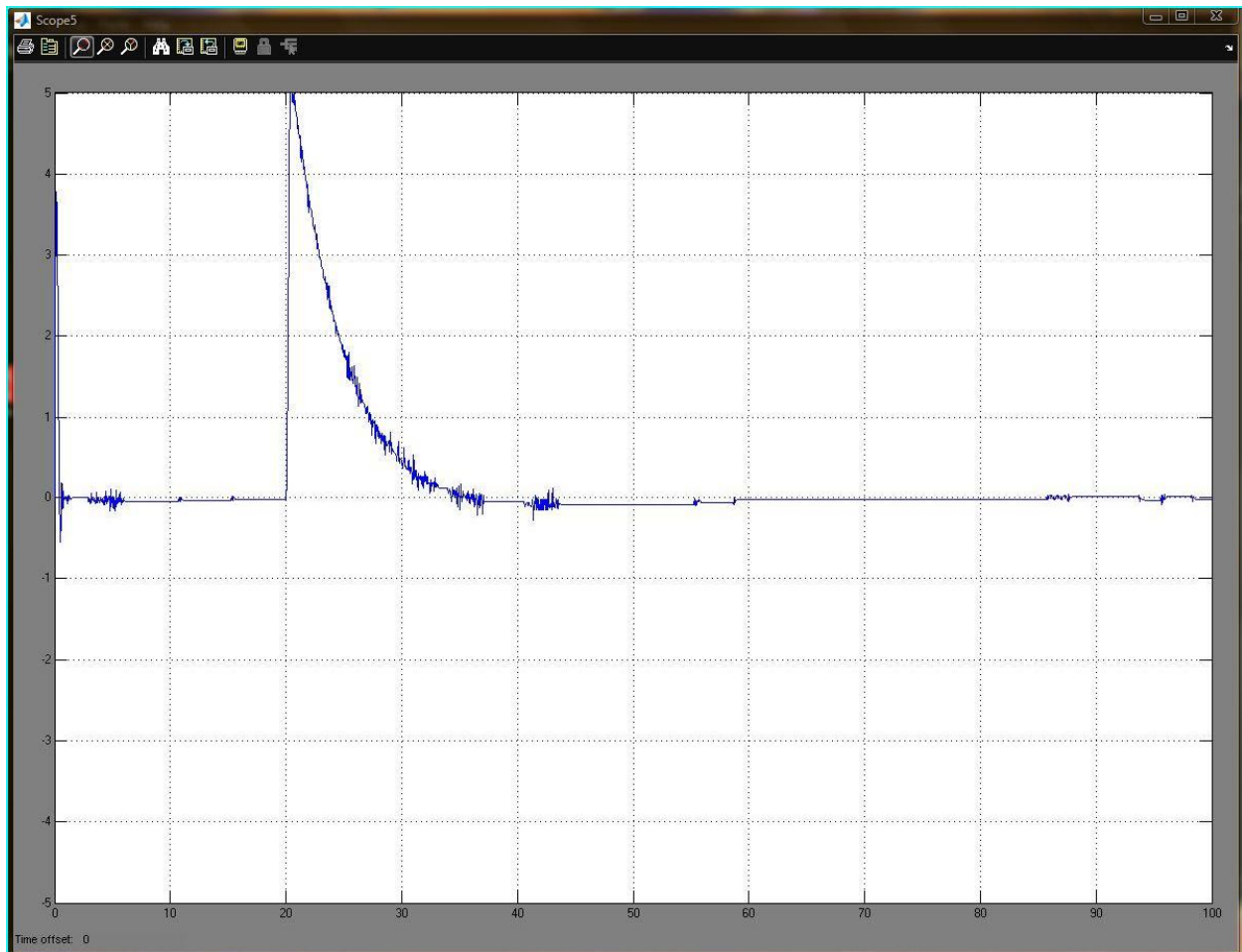
Figure 7 Present Controller Heading Response to a 20 deg heading step command



8.3 PID Controller Heading Rate Response to a 20 degree heading change

Figure 8 shows heading rate change of the ground vehicle during the 20 degree heading change when using the PID controller. Notice that the angular velocity decreases exponentially as the heading error decreases.

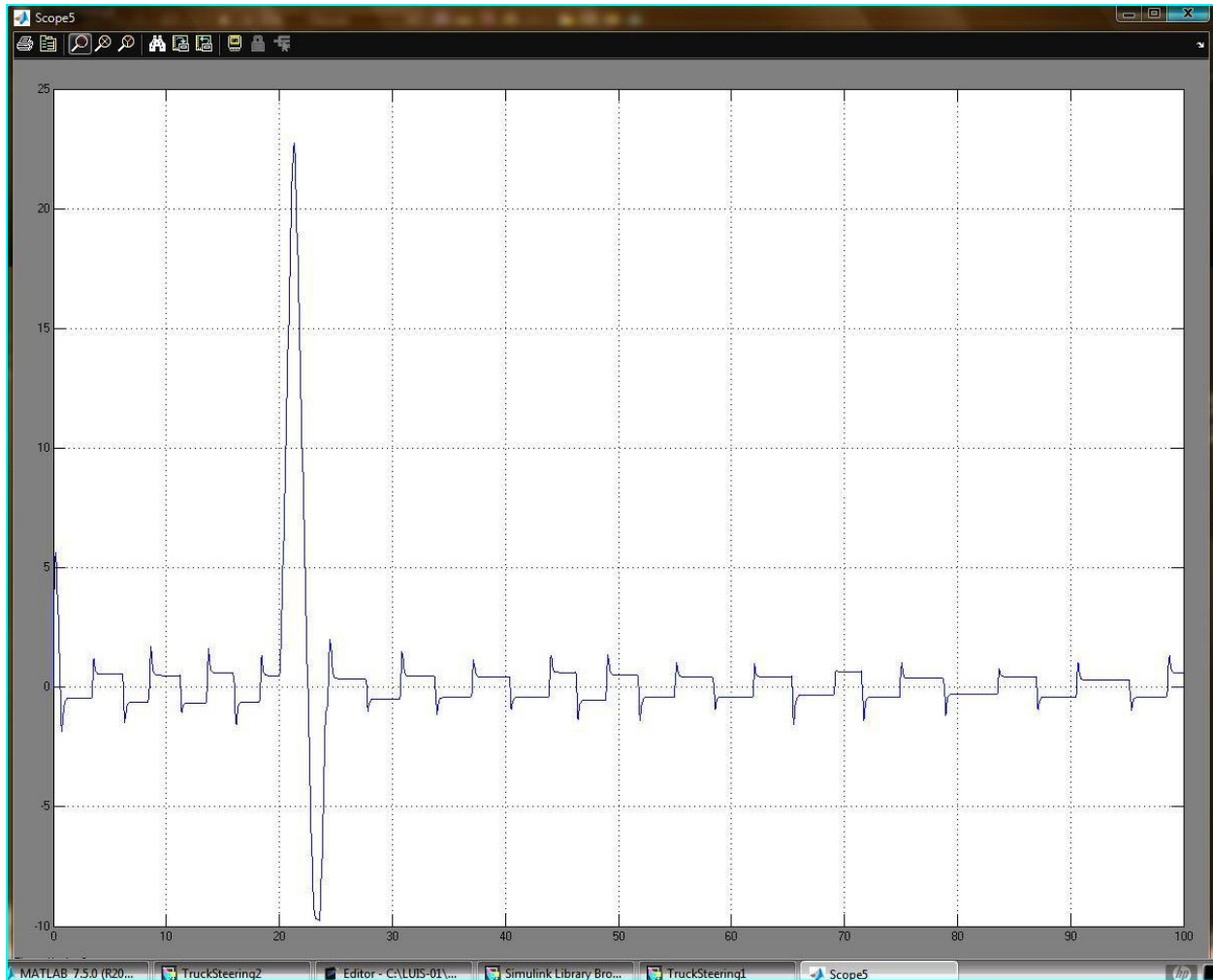
Figure 8 PID Heading Rate Response to a 20 deg heading step command



8.4 Present Controller Heading Rate Response to a 20 degree heading change

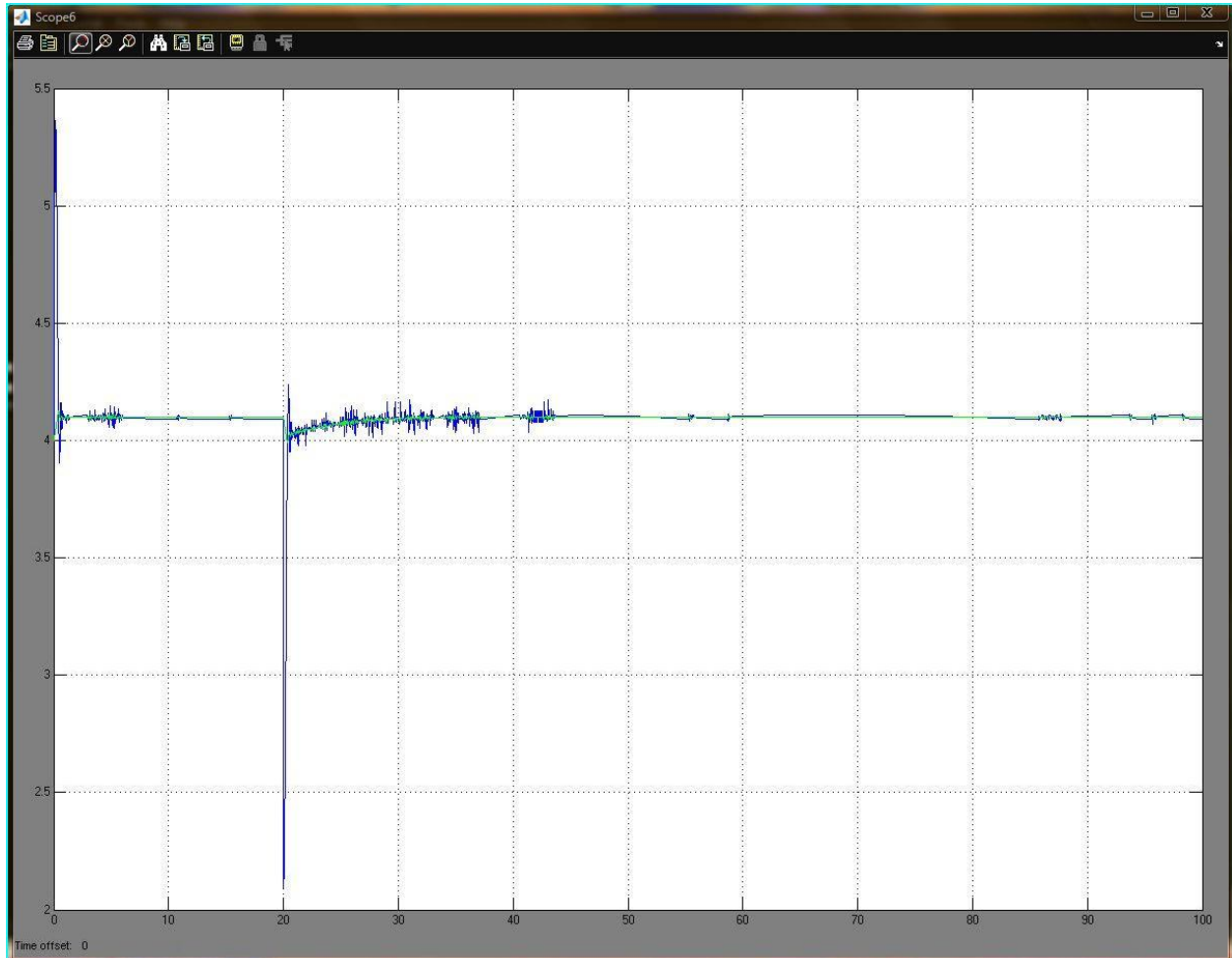
Figure 9 shows heading rate change of the ground vehicle during the 20 degree heading change when using the current heading hold controller. Notice the heading rate oscillation. This is what the driver senses during the Condron test runs.

Figure 9 Present Controller Heading Rate Response to 20 deg heading step command



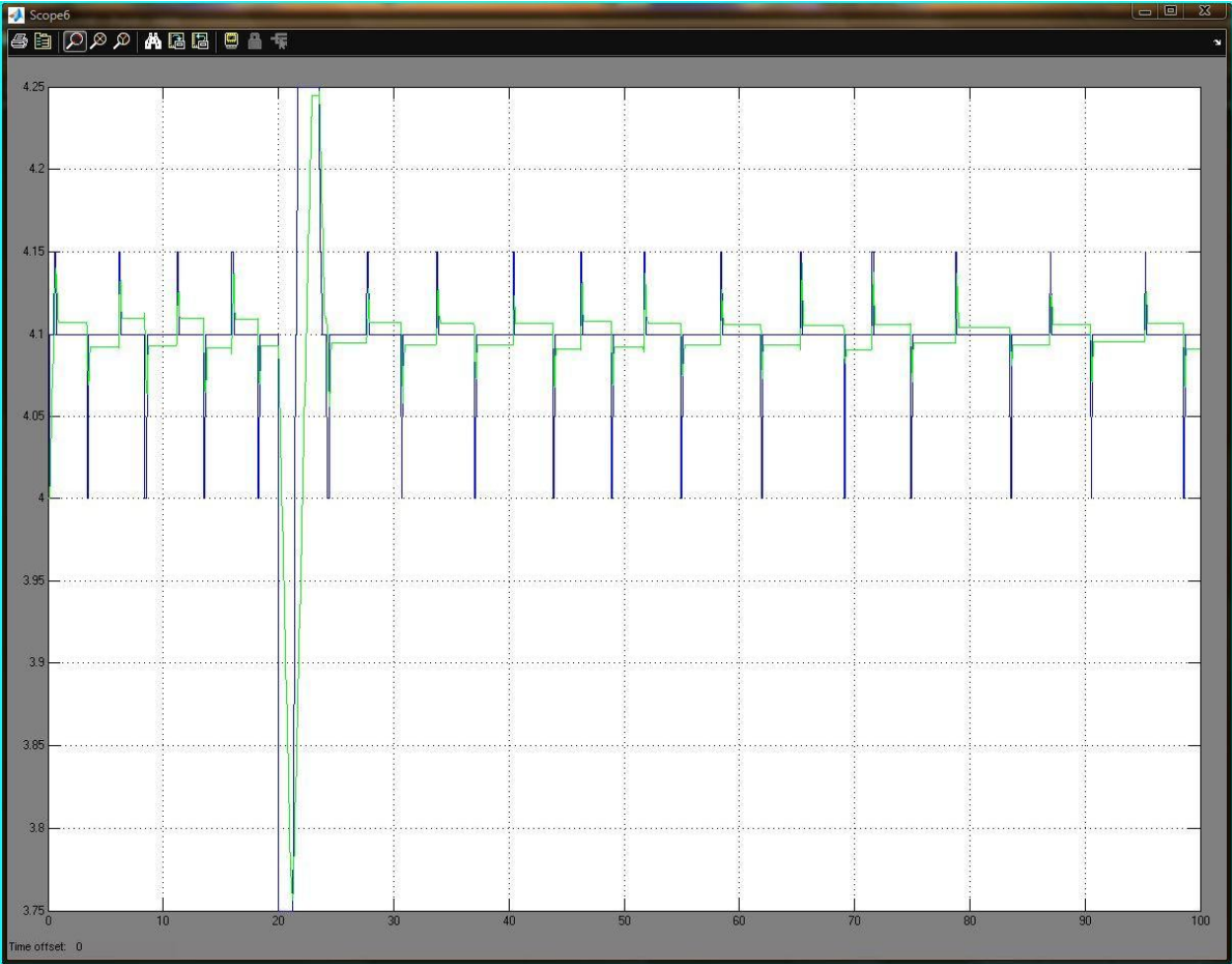
8.5 PID Controller Actuator Response to a 20 degree heading change

Figure 10 PID Actuator Response to 20 deg heading step command



8.6 Present Controller Actuator Response to a 20 degree heading change

Figure 11 Present Controller Actuator Response to 20 deg heading step command



8.7 PID Controller Heading Response to a RAMP heading change

Figure 12 shows the response of the PID HH system to heading RAMP change of 10 degree per second (i.e. turn).

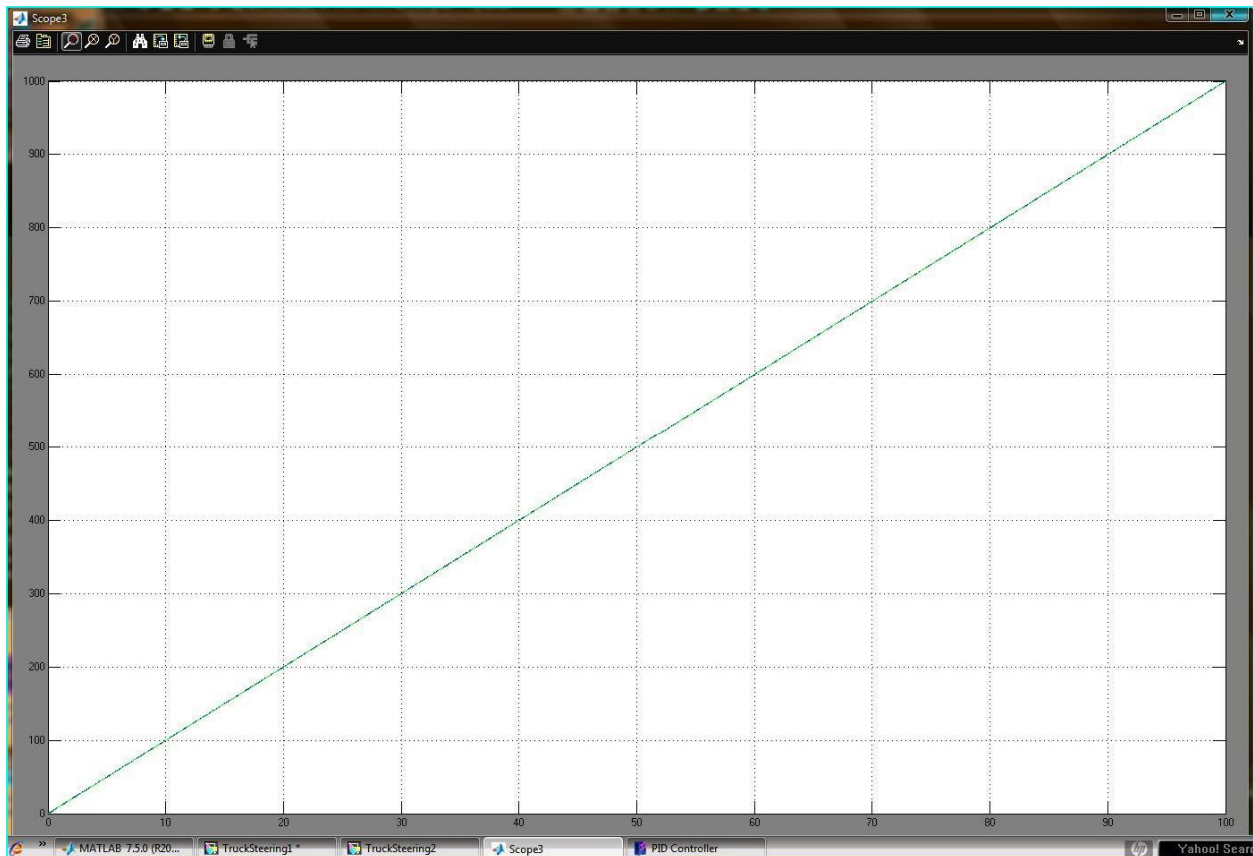
Figure 12 PID Heading Response to a heading RAMP command



8.8 Present Controller Heading Response to a RAMP heading change

Figure 13 shows the response of the current HH system to heading RAMP change of 1 degree per second..

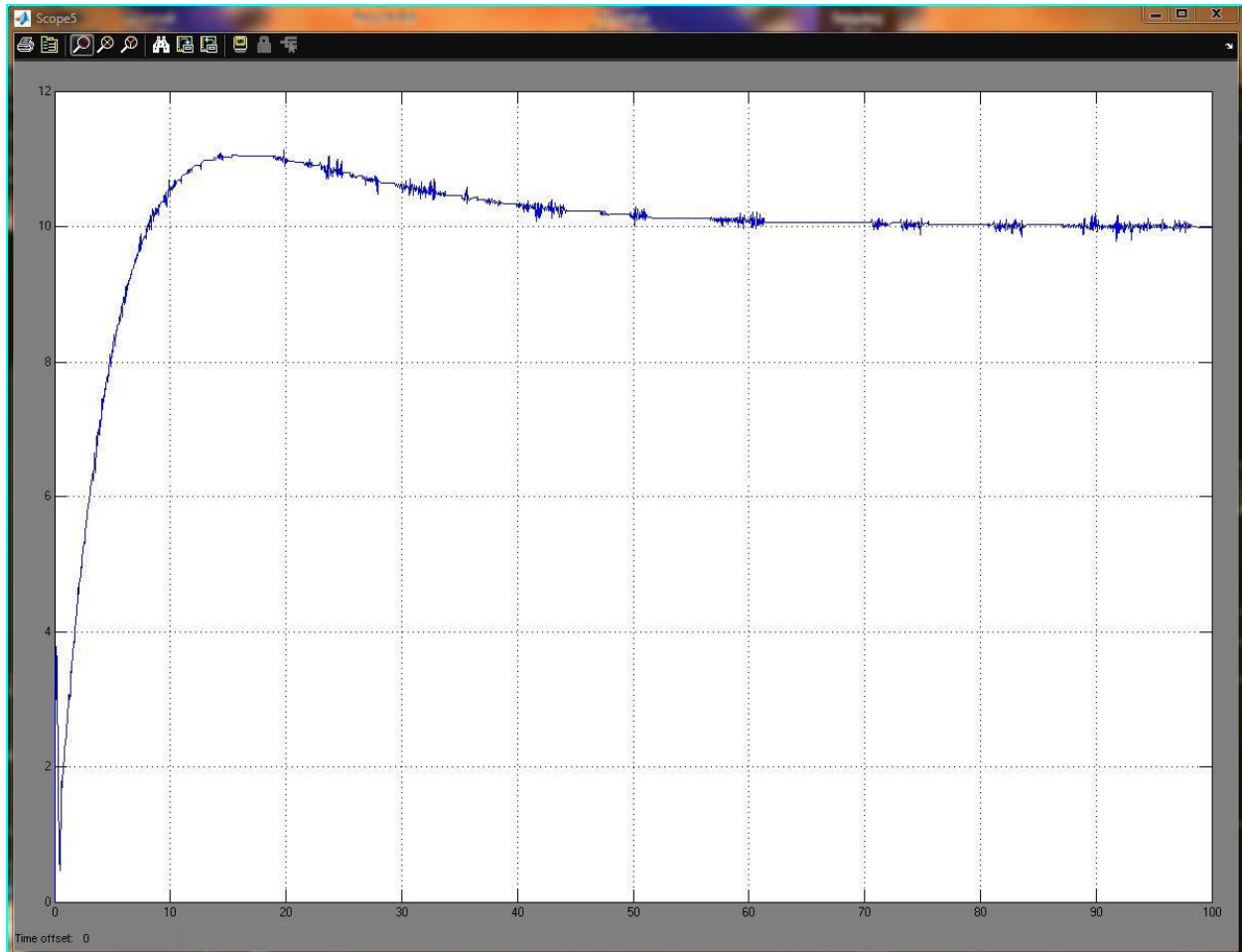
Figure 13 Present Controller Heading Response to a RAMP heading command



8.9 PID Controller Heading Rate Response to a RAMP heading change

Figure 14 shows the target heading rate for the PID controller during the ramping heading change of 1 degree per second. Notice the heading rate is changing at 10 degree per second as it is suppose to.

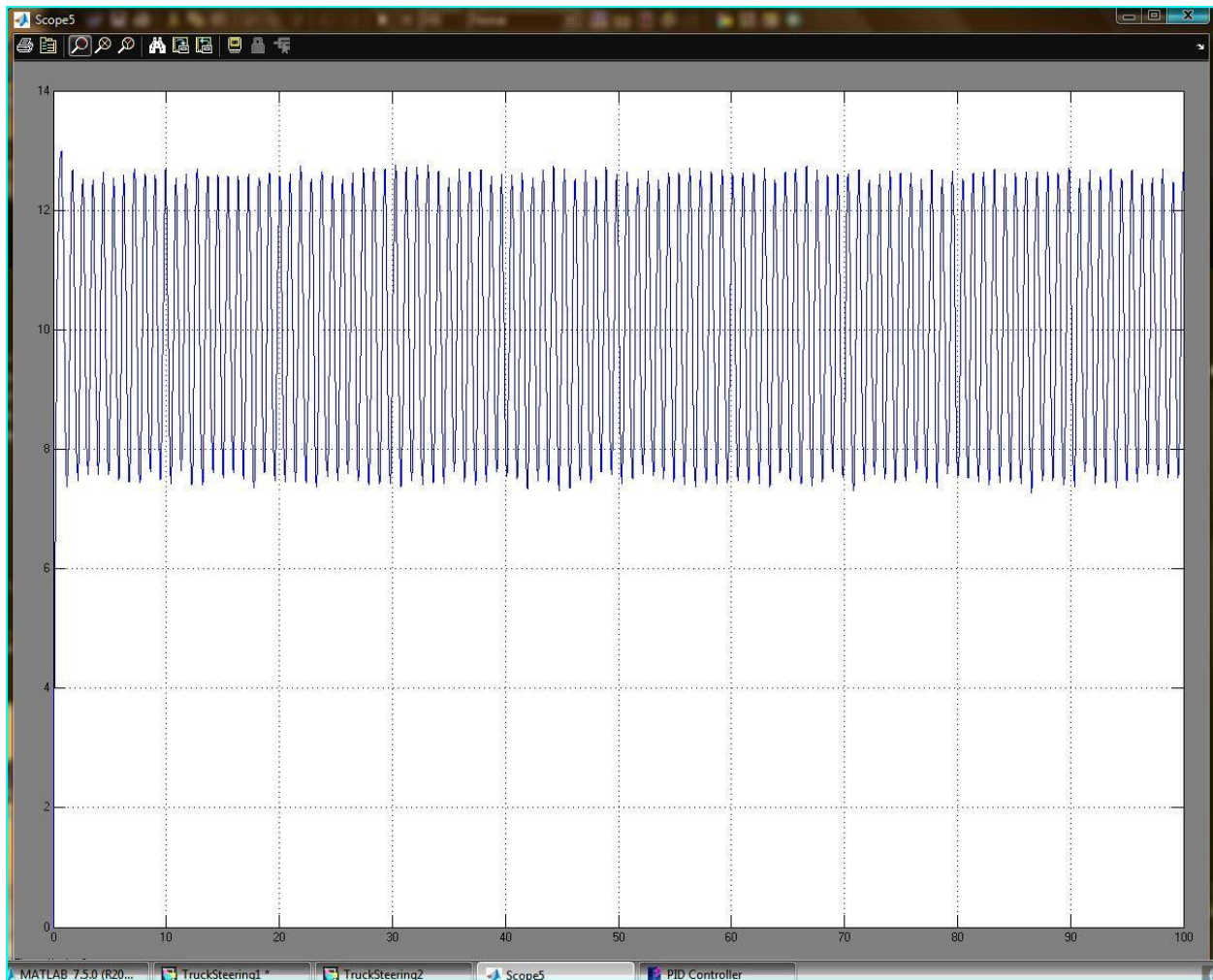
Figure 14 PID Heading Rate Response to a RAMP heading command



8.10 Present Controller Heading Rate Response to a RAMP heading change

Figure 15 shows the target heading rate for the present controller during the ramping heading change of 1 degree per second. Notice that the heading rate is never exactly 10 degrees per second. The value varies from 9 to 12 degrees per second. The steering servo is always working.

Figure 15 Present Controller Heading Rate Response to a RAMP heading command



9. DATA REQUIRED TO MODEL TANKS

9.1 Data Recorded at the VIU

Vehicle ground speed
Steering actuator response
Steering actuator position
Yaw rate
Heading
Throttle command
Throttle position

9.2 Driving Tests

Drive the tanks at slow speeds and make left and right turns in a sinusoidal pattern
Accelerate target from 0 to 20 mph in 5 mph increments
Perform 360 degrees left and right turns for different steering settings; 2, 3, 4, 5 and 6 volts.

10. PID CONTROLLER SOURCE CODE

```
/*=====
=====
NAME: calcDeadVal
DESCRIPTION: Calculate the 'dead' value used in the heading error integrator.
            Used to account for values outside the upper/lower limit values.
=====*/
double calcDeadVal( double val, double low, double high )
{
    double dead = 0.0;

    if ( val < high )
        dead = val - high;
    else if ( val > low )
        dead = val - low;

    return dead;
}

/*=====
=====
NAME: calcHeadingHoldSteerCmd
DESCRIPTION: Calculate the new steering command only if we are in heading
            hold mode.
=====*/
double calcHeadingHoldSteerCmd( int dronenum, gv_lband_uplink *uplink )
{
    double newSteerCmd = 0.0;

    if ( uplink && uplink->discretes.headingHold )
    {
        static const double HEADING_HOLD_K1 = 0.1;
        static const double HEADING_HOLD_K2 = 0.2;
        static const double HEADING_HOLD_K3 = 0.001;

        double headingError;
        double integralInput;
        double headingErrorIntegral;

        static double DEAD_LOWER_LIMIT;
        static double DEAD_UPPER_LIMIT;

        DEAD_LOWER_LIMIT = STEERING_CENTER_STEER + 0.25;
        DEAD_UPPER_LIMIT = STEERING_CENTER_STEER - 0.25;
    }
}
```

```

headingError = (uplink->steeringReferenceCmd -
                gDroneSimData[ dronenum ].prevHeading);

headingError = fmod( headingError, 360.0 );
if ( headingError >= 180.0 )
    headingError -= 360.0;
else if ( headingError < -180.0 )
    headingError += 360.0;

if ( gDroneSimData[dronenum].prevVelocity < 5.0 )
    gDroneSimData[dronenum].prevHeadingIntegral = STEERING_CENTER_STEER;

if ( uplink && uplink->discretes.inhibitIntegrator )
{
    integralInput = 0.0;
}
else
{
    integralInput = headingError - gDroneSimData[dronenum].prevDeadVal;
}

headingErrorIntegral = gDroneSimData[dronenum].prevHeadingIntegral +
    integralInput * HEADING_HOLD_K3 *
    gDroneSimData[dronenum].DELTA_TIME;

newSteerCmd = HEADING_HOLD_K1 * headingError;
newSteerCmd -= (HEADING_HOLD_K2 * gDroneSimData[ dronenum ].prevHeadingDot);
newSteerCmd += headingErrorIntegral;

gDroneSimData[dronenum].prevDeadVal = calcDeadVal( newSteerCmd,
    DEAD_LOWER_LIMIT,
    DEAD_UPPER_LIMIT );

gDroneSimData[dronenum].prevHeadingIntegral = headingErrorIntegral;

newSteerCmd = (2.0 * STEERING_CENTER_STEER) - newSteerCmd;
}
else if ( uplink )
{
    newSteerCmd = uplink->steeringReferenceCmd;
}
else
{
    fprintf(stderr, "ERROR: calcHeadingHoldSteerCmd - No uplink struct sent.\n");
    newSteerCmd = 0.0;
}

return newSteerCmd;
}

```