
# Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

## Basilisk Technical Memorandum

**Document ID: Basilisk-saturate**

**SATURATE C++ MODEL**

| | |
|---|---|
| Prepared by | S. Carnahan |

| |
|---|
| **Status:** Tested |
| **Scope/Contents** |
| The Basilisk saturate model is used to saturate inputs between high and low allowable values. It works on Eigen::VectorXd values for states and uses Eigen::MatrixXd (n x 2) as the state bounds inputs. |

| Rev | Change Description | By | Date |
|---|---|---|---|
| 1.0 | First draft | S. Carnahan | 20180108 |

# Contents



---

# 1 Model Description

$$x_{sat,1} = \min(x_{in}, x_{max}) \tag{1}$$

$$x_{sat} = \max(x_{sat,1}, x_{min}) \tag{2}$$

# 2 Model Functions

- **Set Bounds**: Takes in a matrix of state max and mins.
- **Saturate**: Saturates the input vector according to the max and mins.

# 3 Model Assumptions and Limitations

This code makes no real assumptions. It is up to the user not to give max values that are less than min values and any other inputs that could cause trouble.

# 4 Test Description and Success Criteria

This test is located at `SimCode/sensors/imu_sensor/_UnitTest/test_imu_sensor.py`. In order to get good coverage of all the aspects of the module, the test is broken up into several parts:

## 4.1 Primary Test Method

In order to thoroughly test arbitrary outputs from the IMU, The equations of motion for the sensor were formulated as functions of the center of mass of the spacecraft as opposed to the equations seen in the model description which were formulated about the body frame. The truth values for the following test are set up in the following way:

$$\boldsymbol{r}_{S/N} = \boldsymbol{r}_{C/N} + \boldsymbol{r}_{S/C} \tag{3}$$

$$\dot{\boldsymbol{r}}_{S/N} = \dot{\boldsymbol{r}}_{C/N} + \boldsymbol{r}'_{S/C} + \boldsymbol{\omega}_{B/N} \times \boldsymbol{r}_{S/C} \tag{4}$$

Knowing that:

$$\boldsymbol{r}'_{S/C} = -\boldsymbol{c}' \tag{5}$$

Eq. 4 becomes:

$$\dot{\boldsymbol{r}}_{S/N} = \dot{\boldsymbol{r}}_{C/N} - \boldsymbol{c}' + \boldsymbol{\omega}_{B/N} \times \boldsymbol{r}_{S/C} \tag{6}$$

So, again substituting in the center of mass velocity in the $\mathcal{B}$ frame,

$$\ddot{\boldsymbol{r}}_{S/N} = \ddot{\boldsymbol{r}}_{C/N} - \boldsymbol{c}'' - 2\boldsymbol{\omega}_{B/N} \times \boldsymbol{c}' + \dot{\boldsymbol{\omega}}_{B/N} \times \boldsymbol{r}_{S/C} + \boldsymbol{\omega}_{B/N} \times \boldsymbol{\omega}_{B/N} \times \boldsymbol{r}_{S/C} \tag{7}$$

$\boldsymbol{c}''$ and $\boldsymbol{c}'$ can be solved for in terms of $\ddot{\boldsymbol{c}}$ and $\dot{\boldsymbol{c}}$ using the transport theorem:

$$\boldsymbol{c}' = \dot{\boldsymbol{c}} - \boldsymbol{\omega}_{B/N} \times \boldsymbol{c} \tag{8}$$

$$\boldsymbol{c}'' = \ddot{\boldsymbol{c}} - 2\boldsymbol{\omega}_{B/N} \times \boldsymbol{c}' - \dot{\boldsymbol{\omega}}_{B/N} \times \boldsymbol{c} - \boldsymbol{\omega}_{B/N} \times \boldsymbol{\omega}_{B/N} \times \boldsymbol{c} \tag{9}$$

Now, given the states at a previous time step, $t_{n-1}$, and the accelerations (linear and angular), new states can be calculated for $t_n$:

$$\Delta t = t_n - t_{n-1} \tag{10}$$

$$\dot{\boldsymbol{r}}_{B/N,n} = \dot{\boldsymbol{r}}_{B/N,n-1} + \frac{\ddot{\boldsymbol{r}}_{B/N,n-1} + \ddot{\boldsymbol{r}}_{B/N,n}}{2}\Delta t \tag{11}$$

$$\boldsymbol{r}_{B/N,n} = \boldsymbol{r}_{B/N,n-1} + \frac{\dot{\boldsymbol{r}}_{B/N,n-1} + \dot{\boldsymbol{r}}_{B/N,n}}{2}\Delta t \tag{12}$$

$$\boldsymbol{\omega}_{B/N,n} = \boldsymbol{\omega}_{B/N,n-1} + \frac{\dot{\boldsymbol{\omega}}_{B/N,n-1} + \dot{\boldsymbol{\omega}}_{B/N,n}}{2}\Delta t \tag{13}$$

$$[\boldsymbol{B}] = (1-\sigma^2)[I_{3x3}] + 2[\tilde{\boldsymbol{\sigma}}] + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T \tag{14}$$

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}[\boldsymbol{B}]^{\mathcal{B}}\boldsymbol{\omega} \tag{15}$$

$$\boldsymbol{\sigma}_{B/N,n} = \boldsymbol{\sigma}_{B/N,n-1} + \frac{\dot{\boldsymbol{\sigma}}_{B/N,n-1} + \dot{\boldsymbol{\sigma}}_{B/N,n}}{2}\Delta t \tag{16}$$

The same can then be done for the position $\boldsymbol{r}_{C/N}$ with its derivatives. Also, knowing that:

$$\ddot{\boldsymbol{c}} = \ddot{\boldsymbol{r}}_{C/N} - \ddot{\boldsymbol{r}}_{B/N} \tag{17}$$

the same can be done for $\boldsymbol{c}$ and its derivatives. All of this numerical integration must be done in the inertial frame.

At this point, all of the information needed to solve for Eq. 7 is known. Additionally, the delta-v accumulated between $t_{n-1}$ and $t_n$ can be added to the total delta-v

## 4.2 Test Descriptions

1. Clean The IMU is run with all clean inputs, i.e. nonzero accelerations and angular accelerations of the spacecraft and this is compared to the truth values generated in python. No noise, discretization, saturation, etc. is applied.

   **Success Criteria**: The outputs match to acceptable tolerance and are visually confirmed.

2. Noise The IMU is run with inputs as in the clean test, i.e. nonzero accelerations and angular accelerations of the spacecraft and this is compared to the truth values generated in python. Gaussian noise and random walk are applied.

   **Success Criteria**: The output standard deviations match the inputs to acceptable tolerance.

3. Bias The IMU is run with all clean inputs, i.e. nonzero accelerations and angular accelerations of the spacecraft and this is compared to the truth values generated in python. Bias is then added.

   **Success Criteria**: The outputs match to acceptable tolerance and are visually confirmed to include bias.

4. Saturation The IMU is run with all clean inputs, i.e. nonzero accelerations and angular accelerations of the spacecraft and this is compared to the truth values generated in python. Out of bounds values are floored or ceilinged.

   **Success Criteria**: The outputs match to acceptable tolerance and are visually confirmed to be capped.

5. Discretization The IMU is run with all clean inputs, i.e. nonzero accelerations and angular accelerations of the spacecraft and this is compared to the truth values generated in python. Outputs are discretized.

   **Success Criteria**: The outputs match to acceptable tolerance and are visually confirmed to be discretized. Note. Two points in time always fail this test. This has to do with the python generated and c++ generated values being ever-so-slightly off and not discretizing at the same point. They match at the next timesteps and have been ignored for the test.

As an additional check, $[PB]$ is calculated separately for the truth values and $yaw$, $pitch$, and $roll$ are fed to the IMU which calculates this value independently. In this way, the multiple set-up options for the IMU are validated.

# 5 Test Parameters

This section summarizes the specific error tolerances for each test. Error tolerances are determined based on whether the test results comparison should be exact or approximate due to integration or other reasons. Error tolerances for each test are summarized in table 4.

**Table 2:** Error tolerance for each test. Note that tolerances are relative $\frac{truth-output}{truth}$

| Test | Tolerance | GyroLSB | AccelLSB | RotMax | TransMax | RotNoise | TransNoise | RotBias | TransBias |
|---|---|---|---|---|---|---|---|---|---|
| Clean | 1e-08 | 0e+00 | 0e+00 | 1.0e+03 | 1.0e+03 | 0.0 | 0.0 | 0.0e+00 | 0.0e+00 |
| Noise | 1e-01 | 0e+00 | 0e+00 | 1.0e+03 | 1.0e+03 | 0.1 | 0.1 | 0.0e+00 | 0.0e+00 |
| Bias | 1e-08 | 0e+00 | 0e+00 | 1.0e+03 | 1.0e+03 | 0.0 | 0.0 | 1.0e+01 | 1.0e+01 |
| Sat. | 1e-08 | 0e+00 | 0e+00 | 1.0e+00 | 5.0e+00 | 0.0 | 0.0 | 0.0e+00 | 0.0e+00 |
| Disc. | 1e-08 | 5e-02 | 5e-01 | 1.0e+02 | 1.0e+03 | 0.0 | 0.0 | 0.0e+00 | 0.0e+00 |

For all tests, the gyro has a scale factor of 1 applied to each axis while the accelerometer has a scale factor of two. This functionality is easily verified in the Noise test, which has 2x the standard deviation that was given only for the linear outputs.

# 6 Test Results

All checks within test_imu_sensor.py passed as expected. Table 3 shows the test results. The figures below the table show that the truth values matched the output values for all values checked.

**Table 3:** Test results

| Test | Pass/Fail |
|---|---|
| Clean | PASSED |
| Noise | PASSED |
| Bias | PASSED |
| Sat. | PASSED |
| Disc. | PASSED |

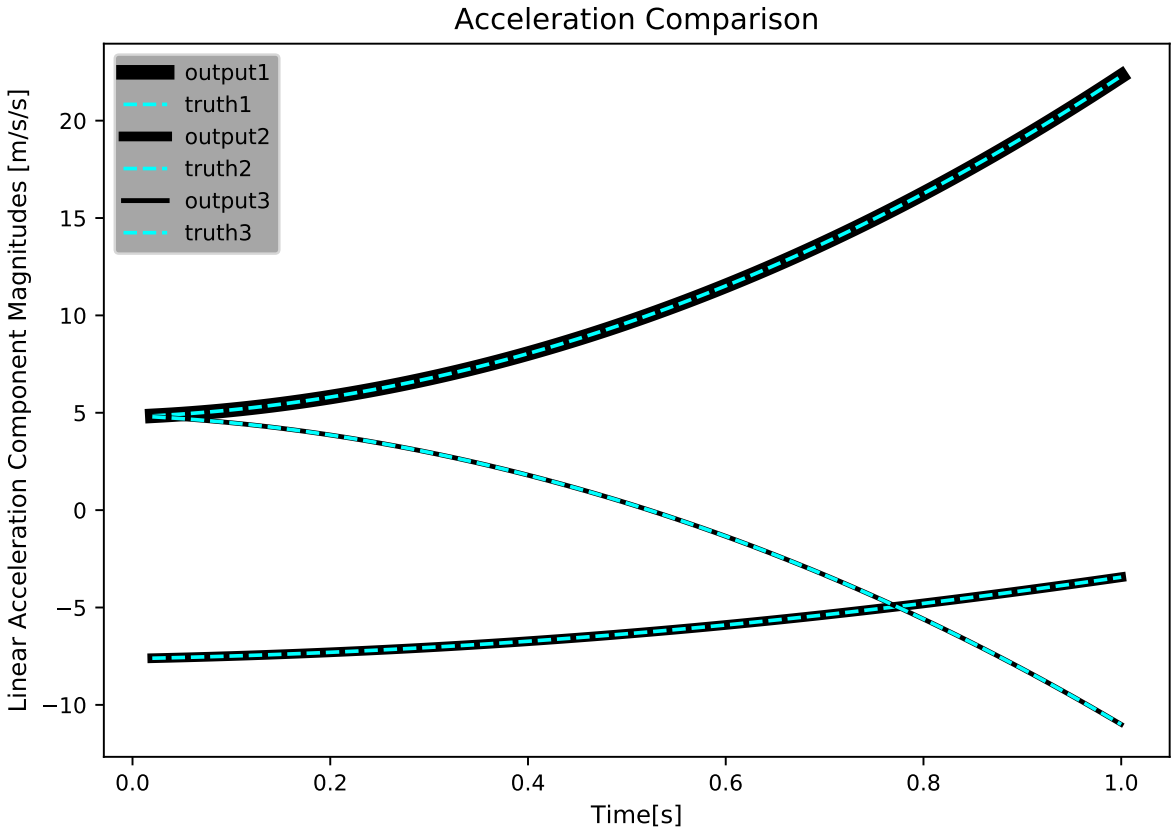


**Fig. 1:** Plot Comparing Sensor Linear Accelertaion Truth and Output for test: clean. Note that 1, 2, and 3 indicate the components of the acceleration.

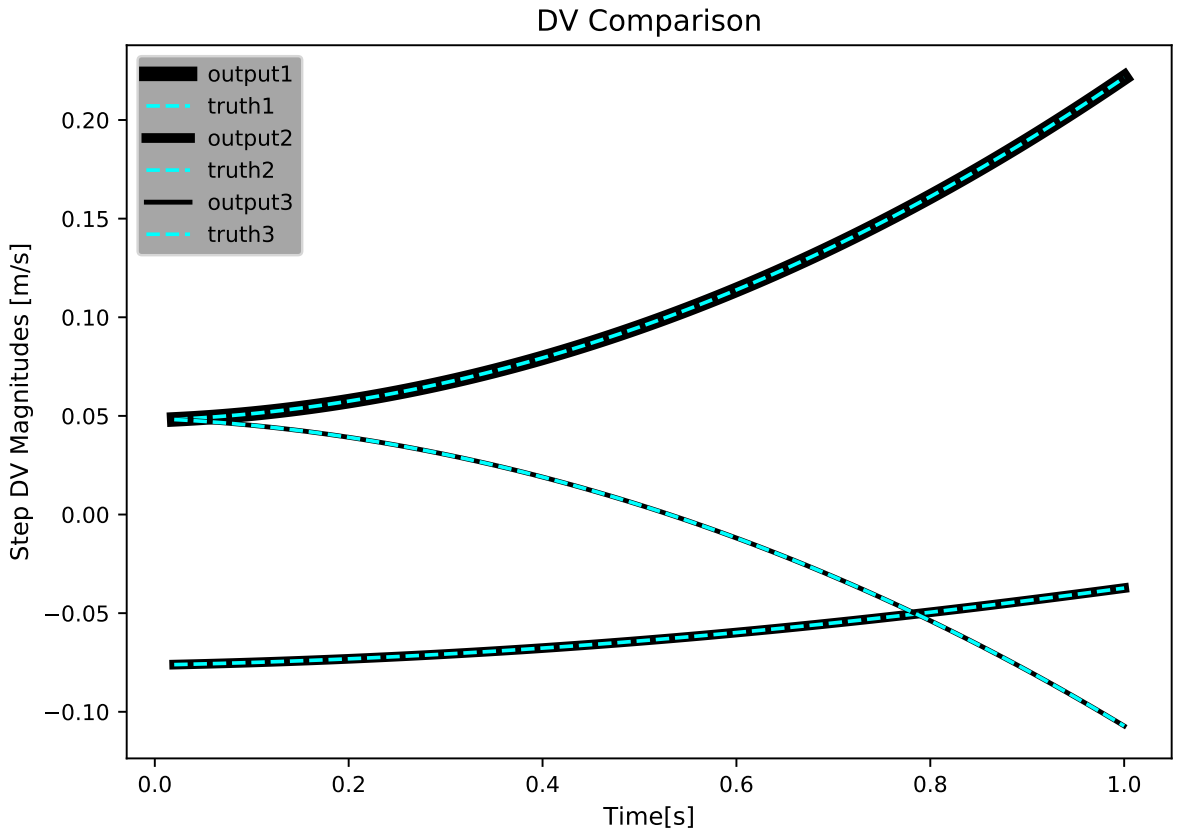


**Fig. 2:** Plot Comparing Time Step DV Truth and Output for test: clean. Note that 1, 2, and 3 indicate the components of the velocity delta.

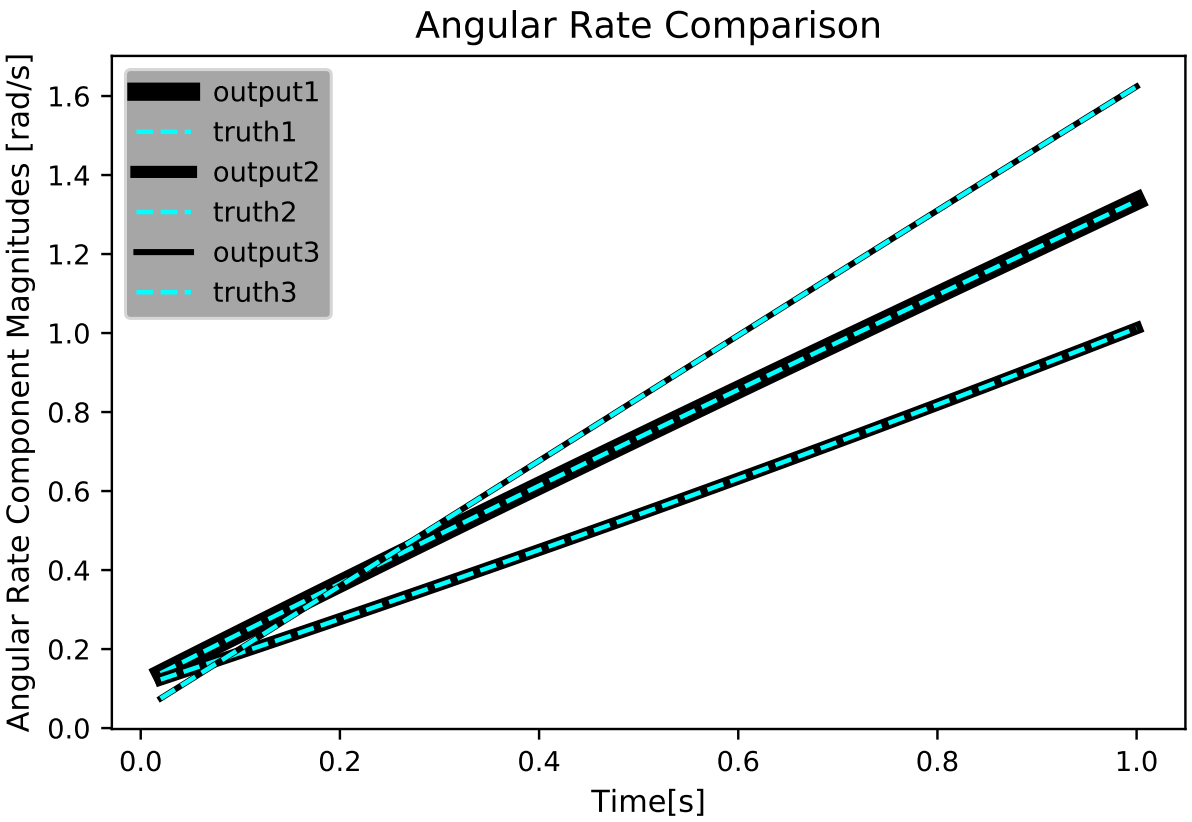


**Fig. 3:** Plot Comparing Angular Rate Truth and Output for test: clean. Note that 1, 2, and 3 indicate the components of the angular rate.

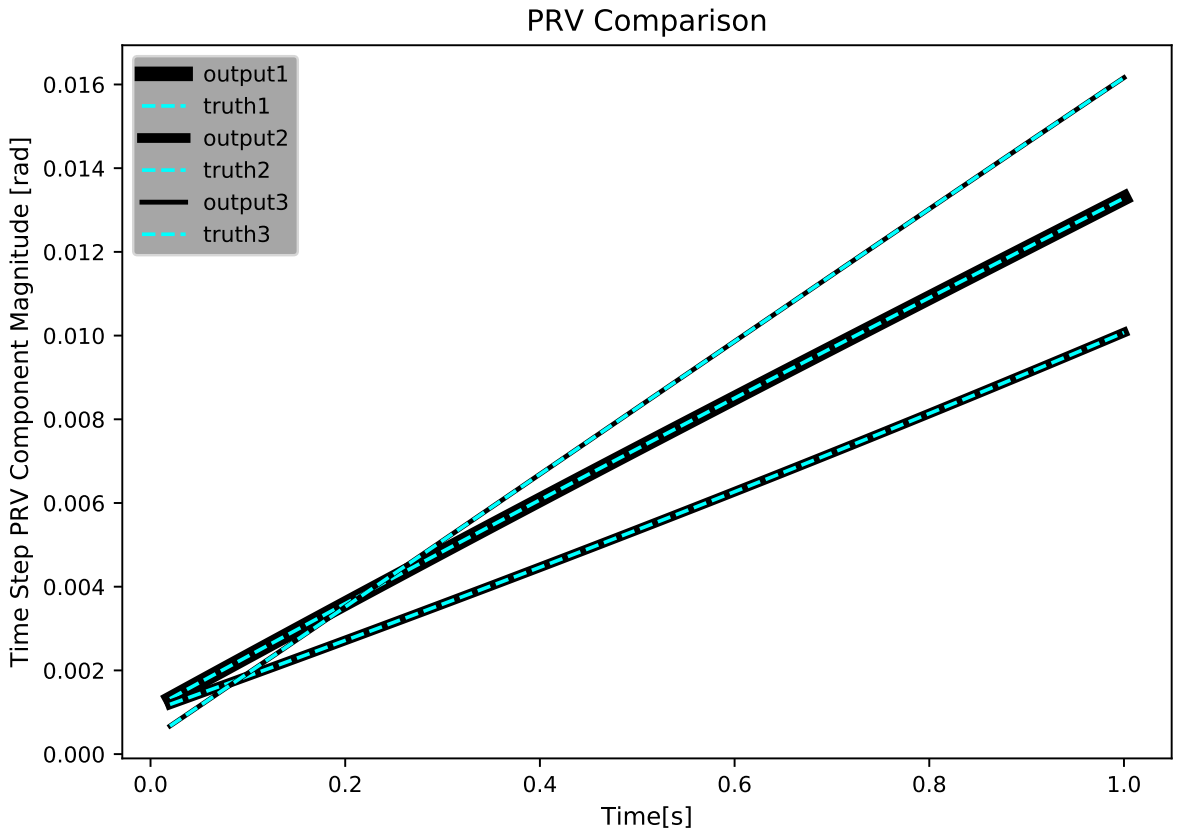


**Fig. 4:** Plot Comparing Time Step PRV Truth and Output for test: clean. Note that 1, 2, and 3 indicate the components of the principal rotation vector.

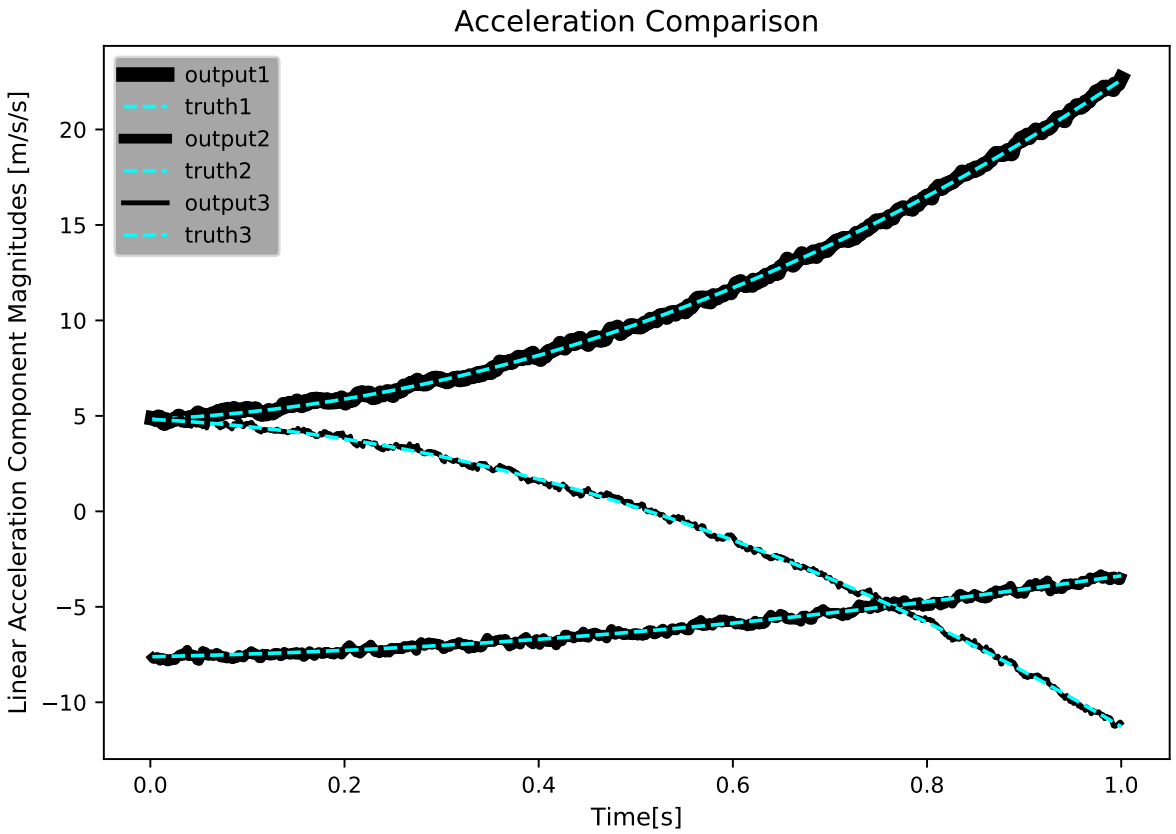


**Fig. 5:** Plot Comparing Sensor Linear Accelertaion Truth and Output for test: noise. Note that 1, 2, and 3 indicate the components of the acceleration.

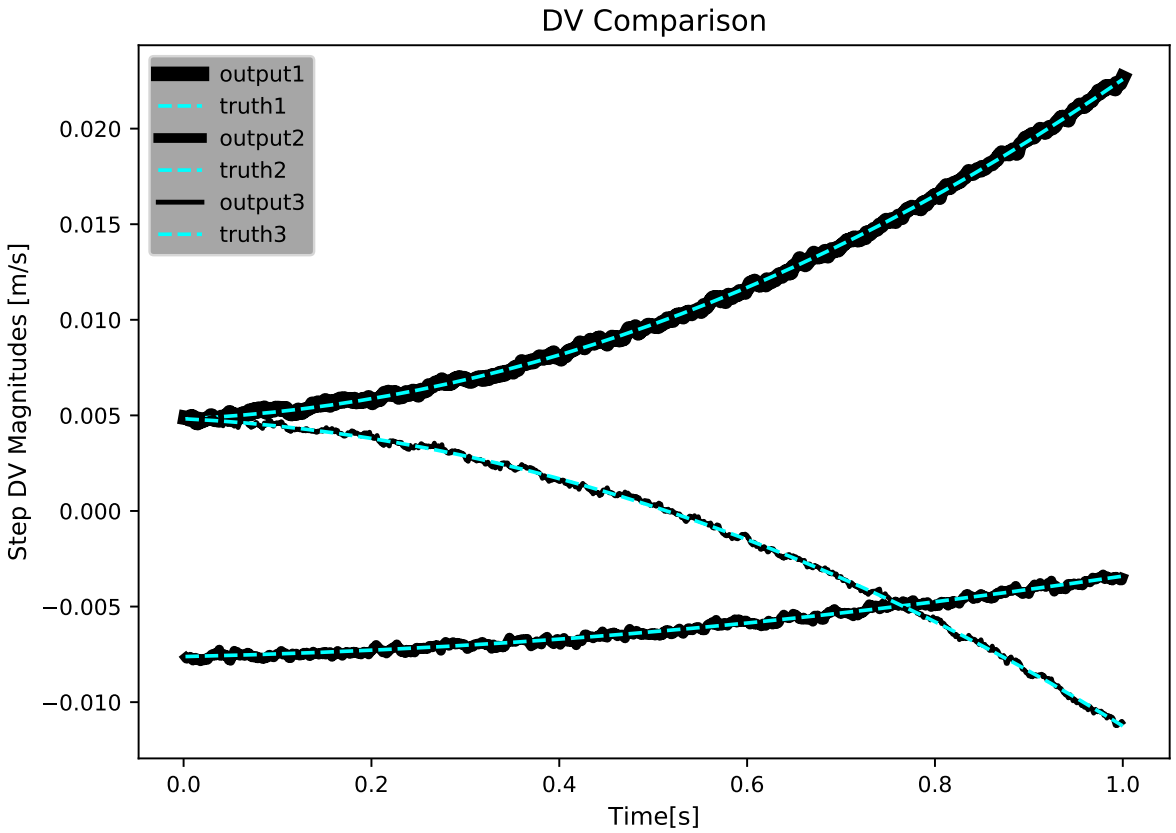


**Fig. 6:** Plot Comparing Time Step DV Truth and Output for test: noise. Note that 1, 2, and 3 indicate the components of the velocity delta.

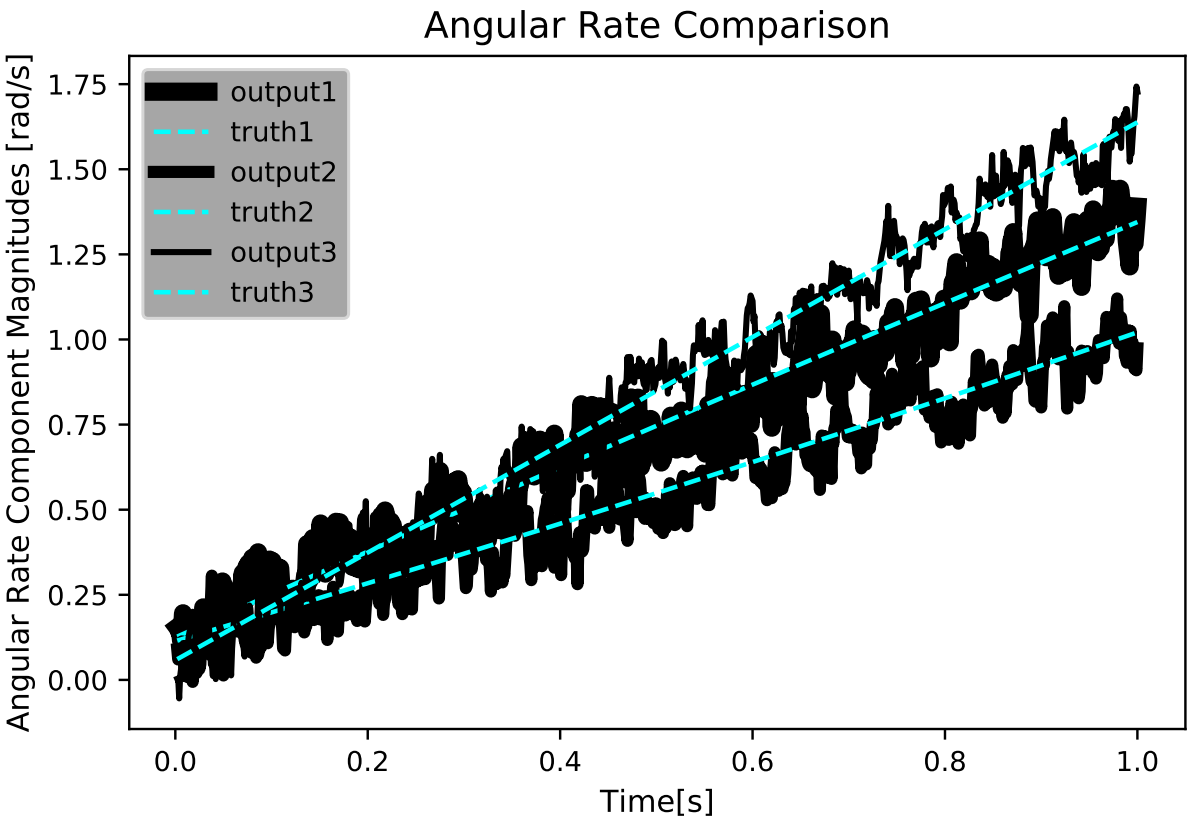


**Fig. 7:** Plot Comparing Angular Rate Truth and Output for test: noise. Note that 1, 2, and 3 indicate the components of the angular rate.

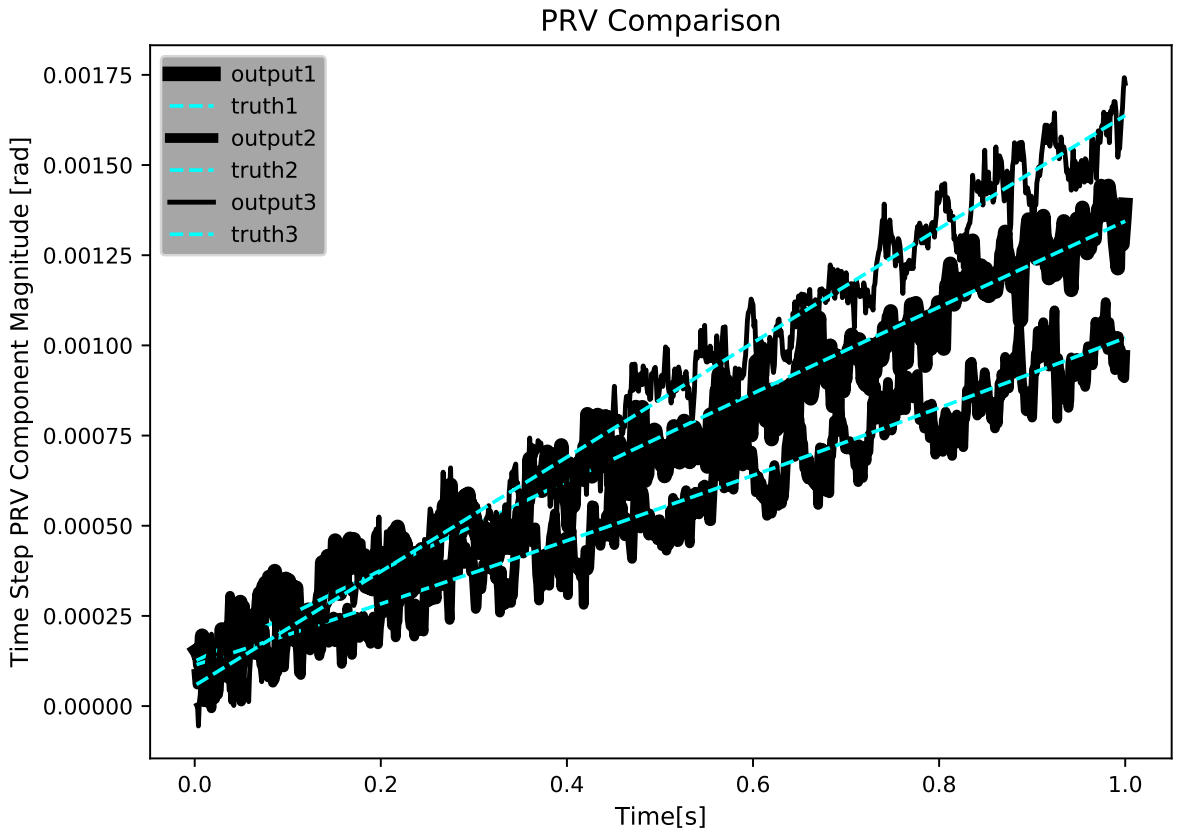


**Fig. 8:** Plot Comparing Time Step PRV Truth and Output for test: noise. Note that 1, 2, and 3 indicate the components of the principal rotation vector.

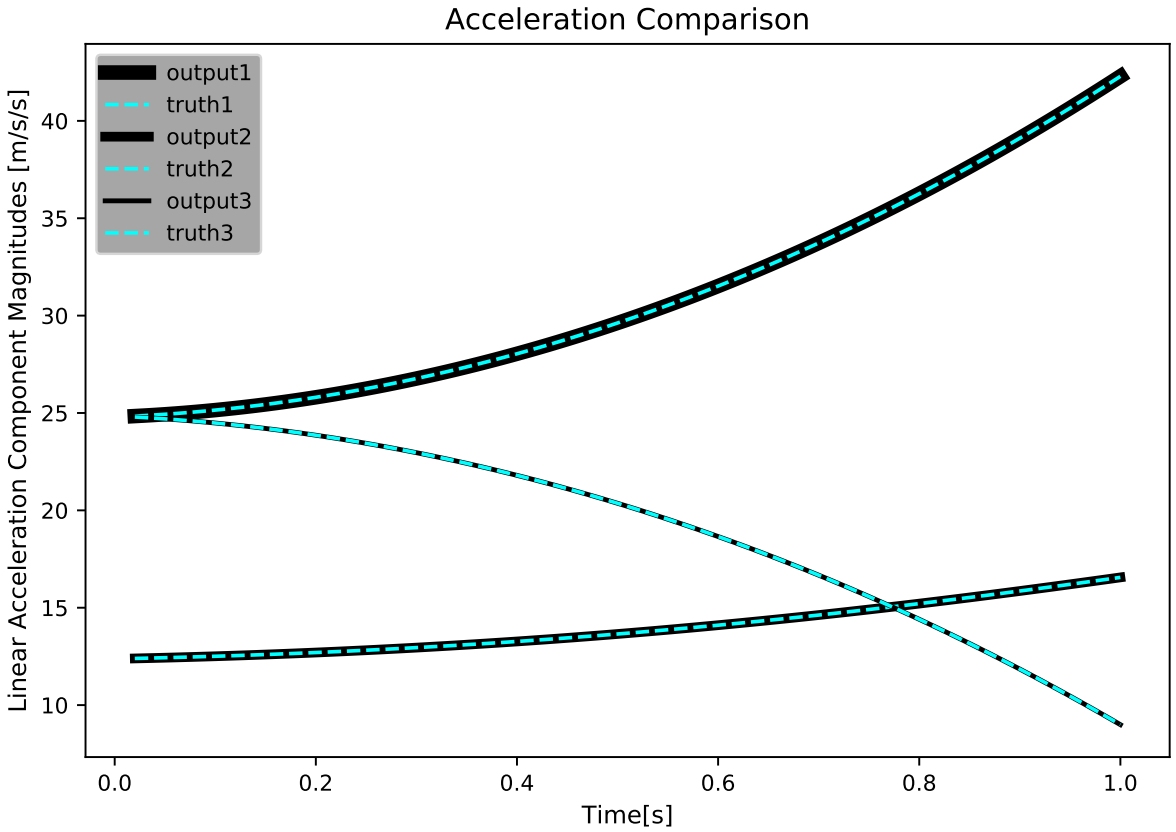


**Fig. 9:** Plot Comparing Sensor Linear Accelertaion Truth and Output for test: bias. Note that 1, 2, and 3 indicate the components of the acceleration.

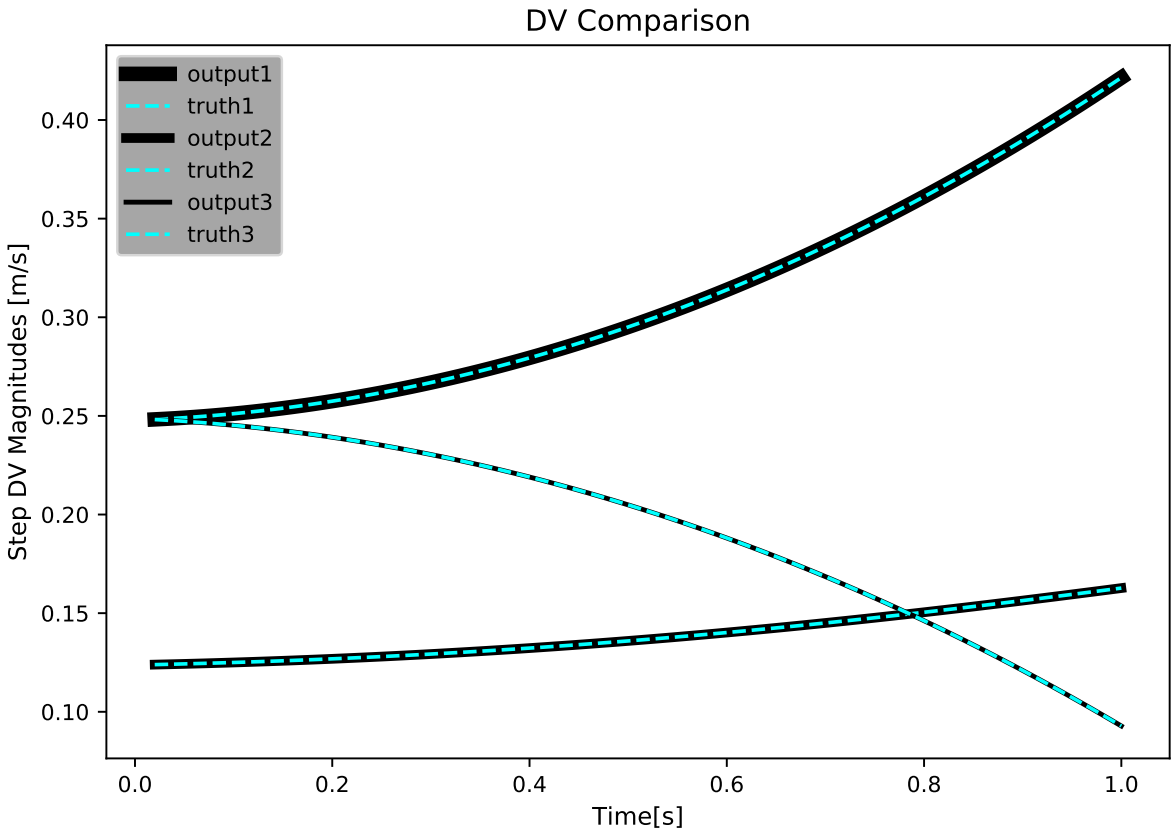


**Fig. 10:** Plot Comparing Time Step DV Truth and Output for test: bias. Note that 1, 2, and 3 indicate the components of the velocity delta.

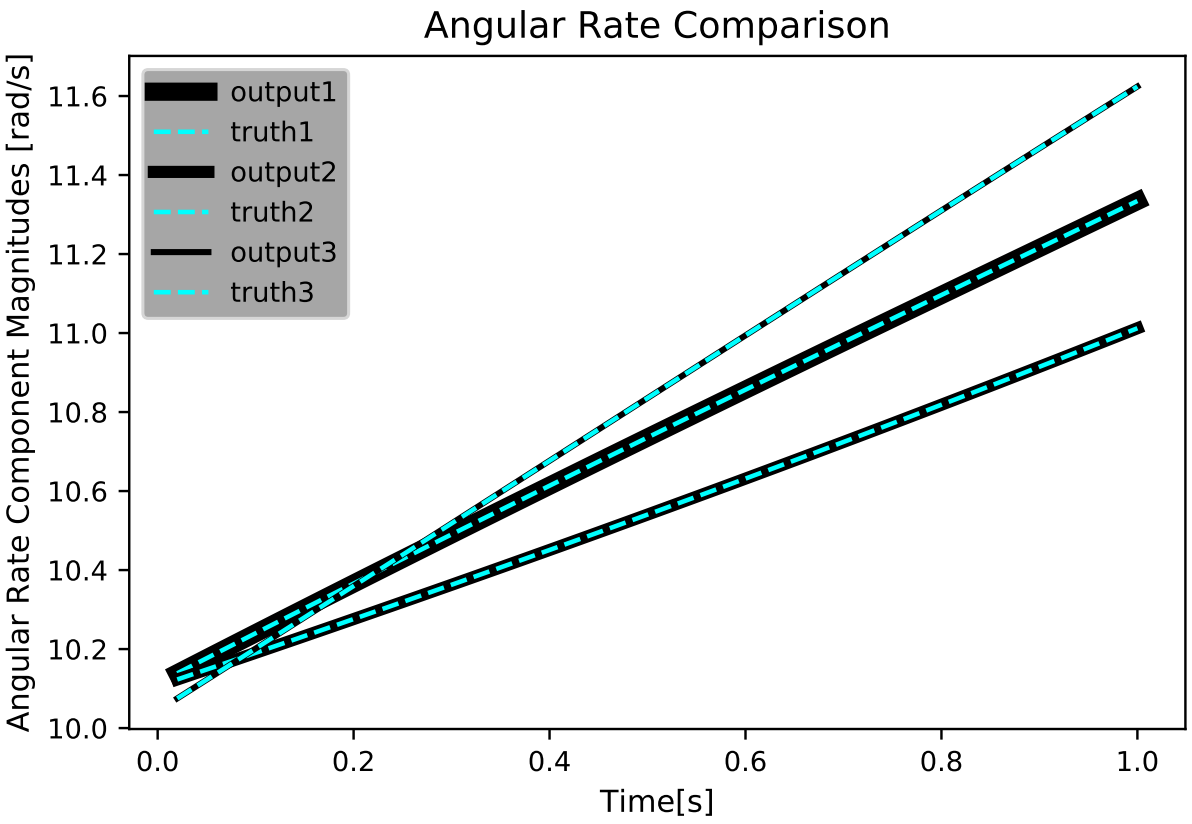


**Fig. 11:** Plot Comparing Angular Rate Truth and Output for test: bias. Note that 1, 2, and 3 indicate the components of the angular rate.

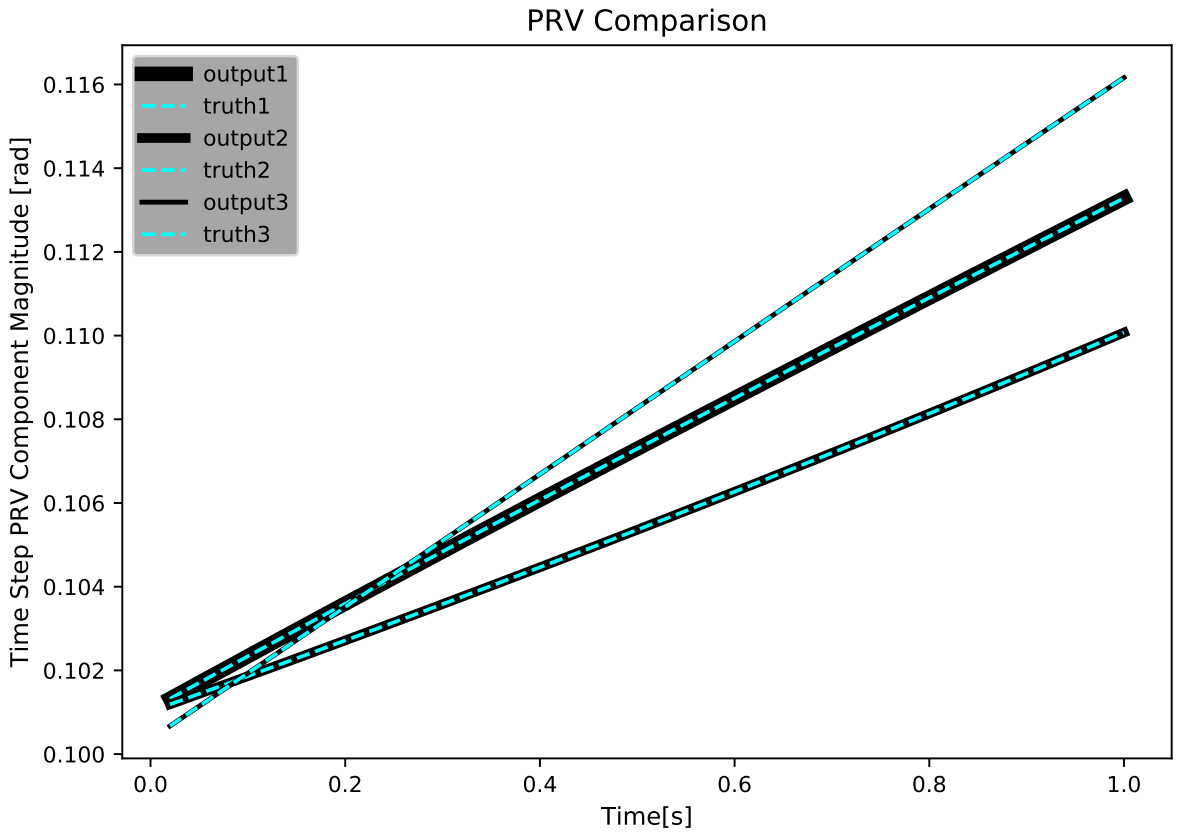


**Fig. 12:** Plot Comparing Time Step PRV Truth and Output for test: bias. Note that 1, 2, and 3 indicate the components of the principal rotation vector.

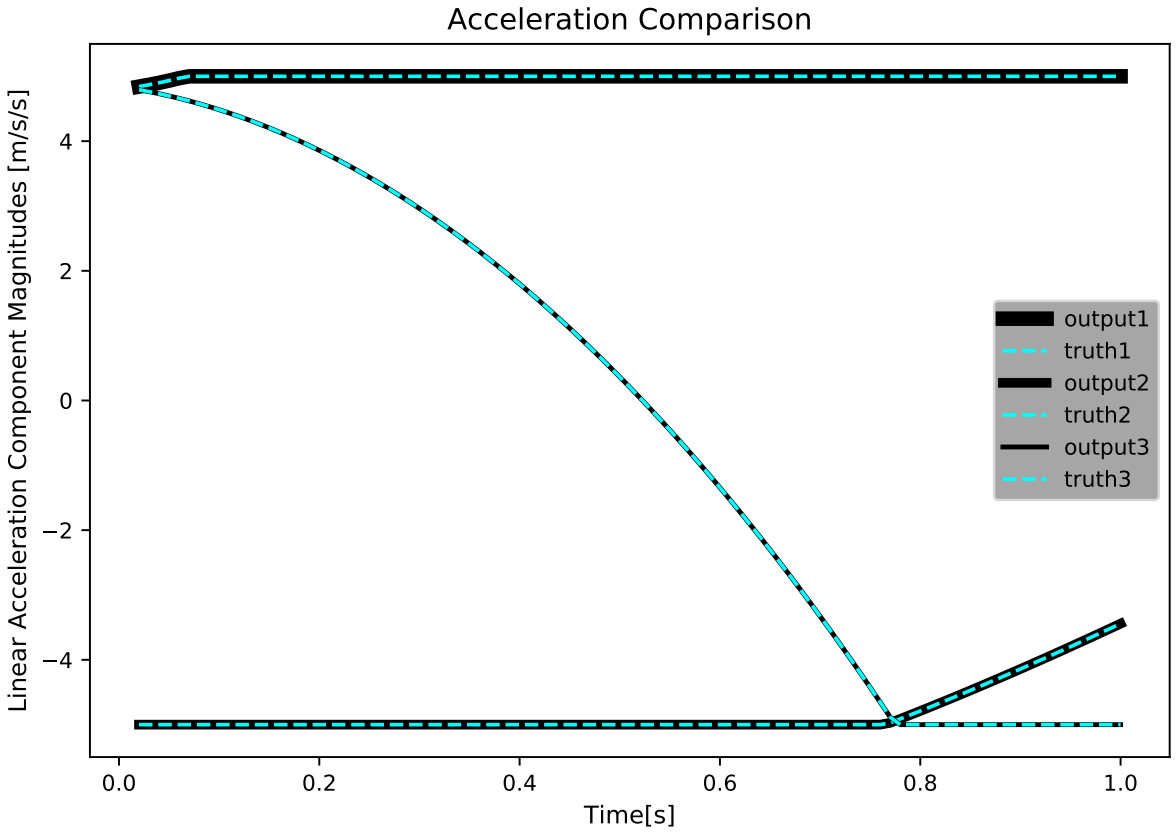


**Fig. 13:** Plot Comparing Sensor Linear Accelertaion Truth and Output for test: saturation. Note that 1, 2, and 3 indicate the components of the acceleration.

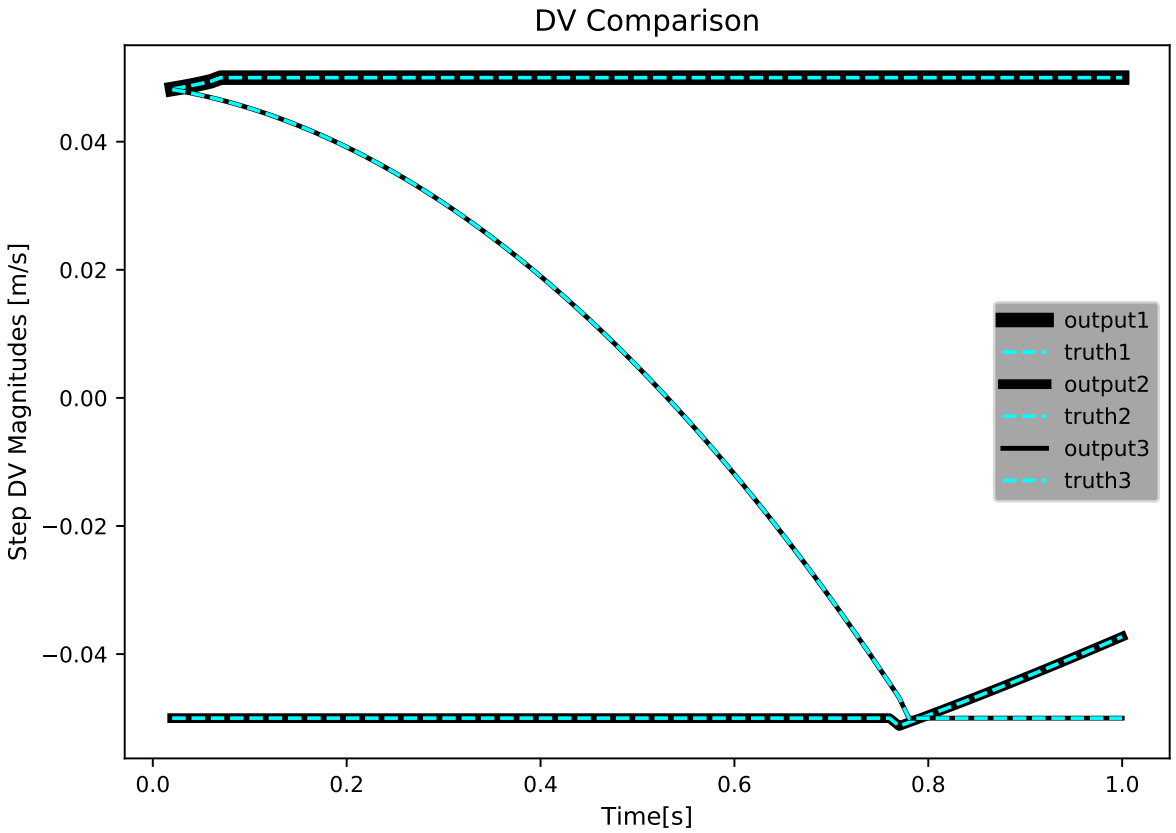


**Fig. 14:** Plot Comparing Time Step DV Truth and Output for test: saturation. Note that 1, 2, and 3 indicate the components of the velocity delta.

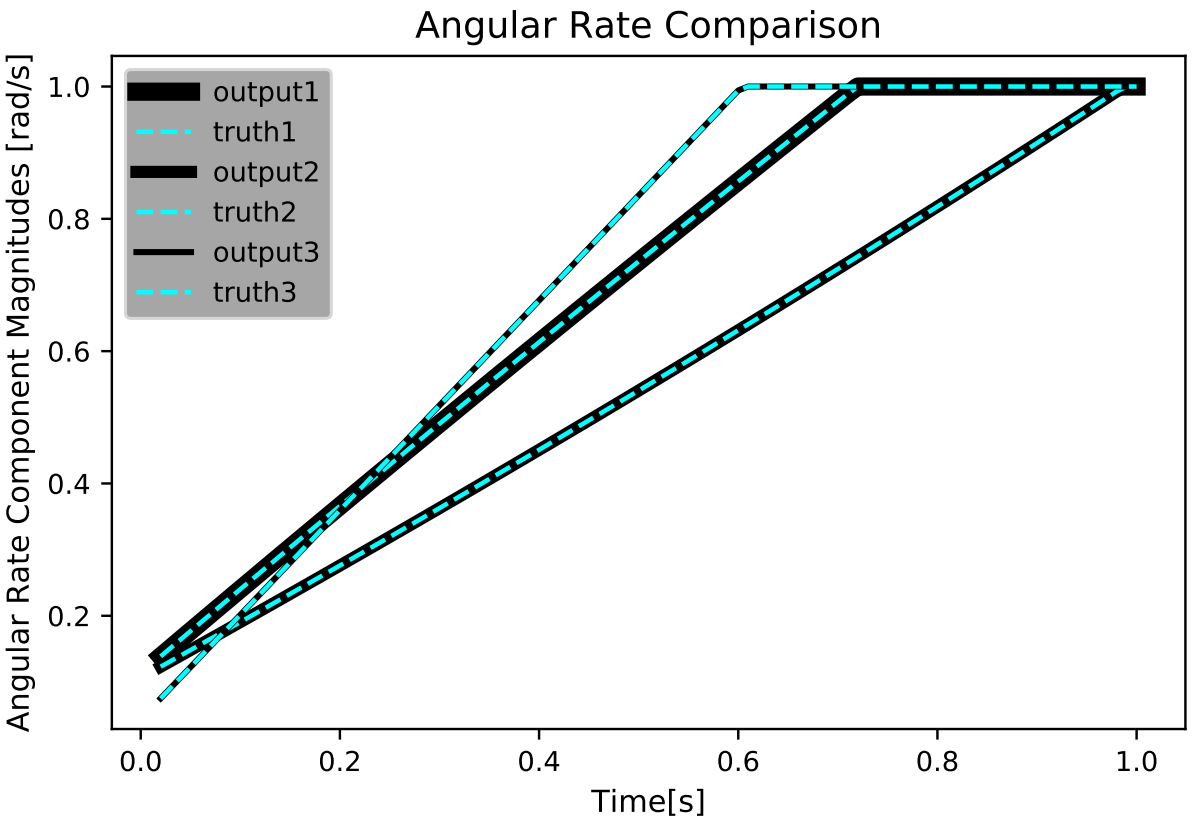


**Fig. 15:** Plot Comparing Angular Rate Truth and Output for test: saturation. Note that 1, 2, and 3 indicate the components of the angular rate.

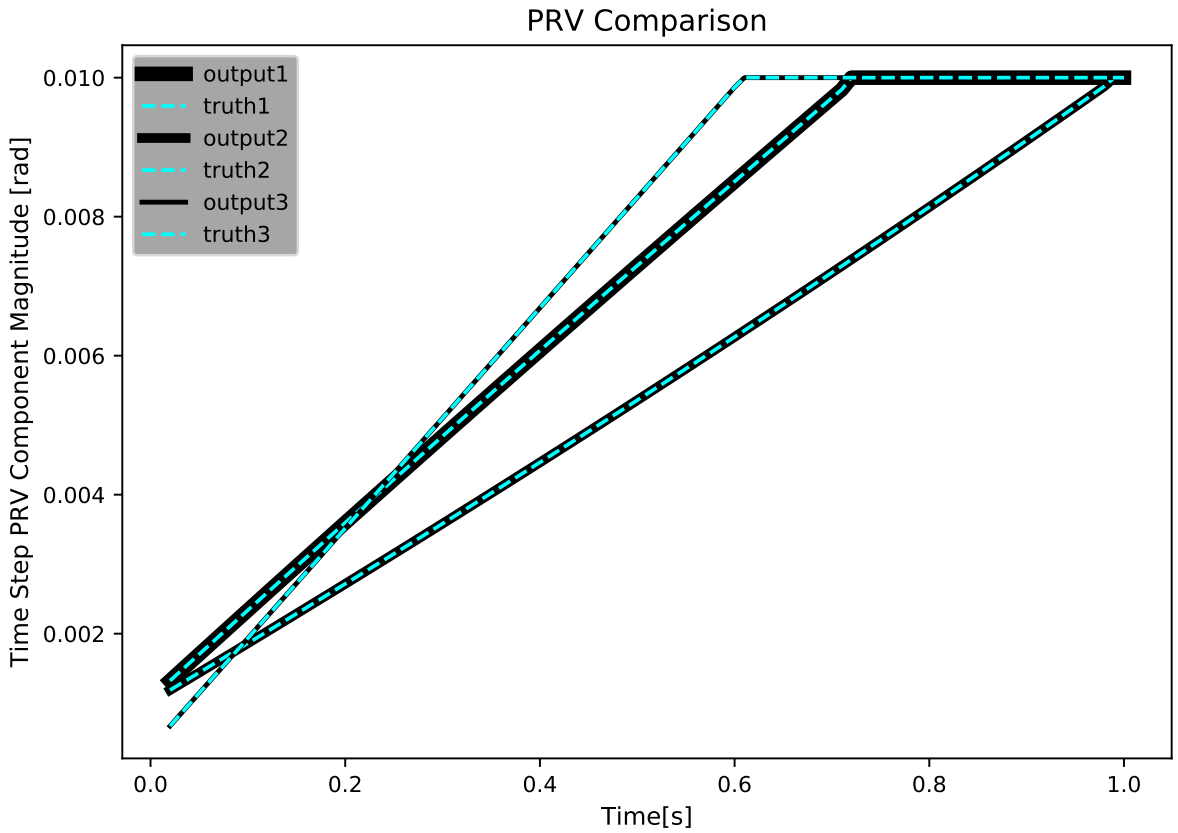


**Fig. 16:** Plot Comparing Time Step PRV Truth and Output for test: saturation. Note that 1, 2, and 3 indicate the components of the principal rotation vector.

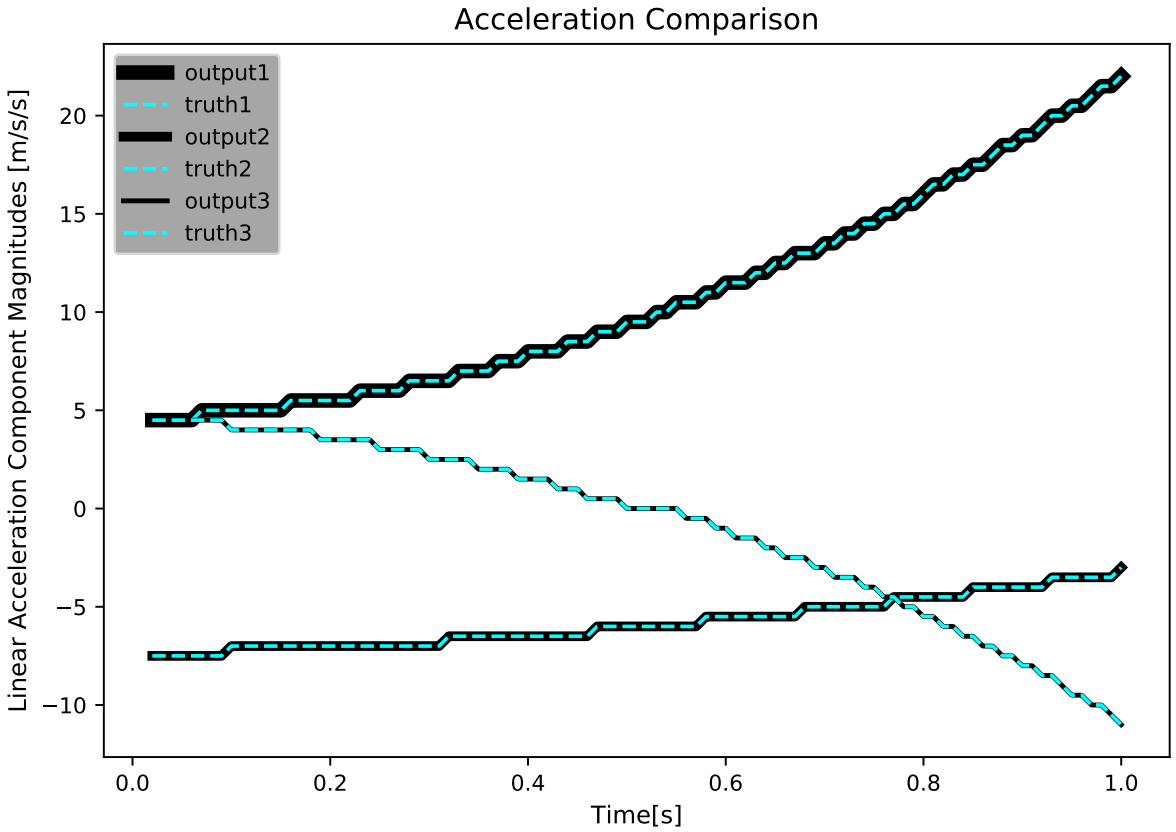


**Fig. 17:** Plot Comparing Sensor Linear Accelertaion Truth and Output for test: discretization. Note that 1, 2, and 3 indicate the components of the acceleration.

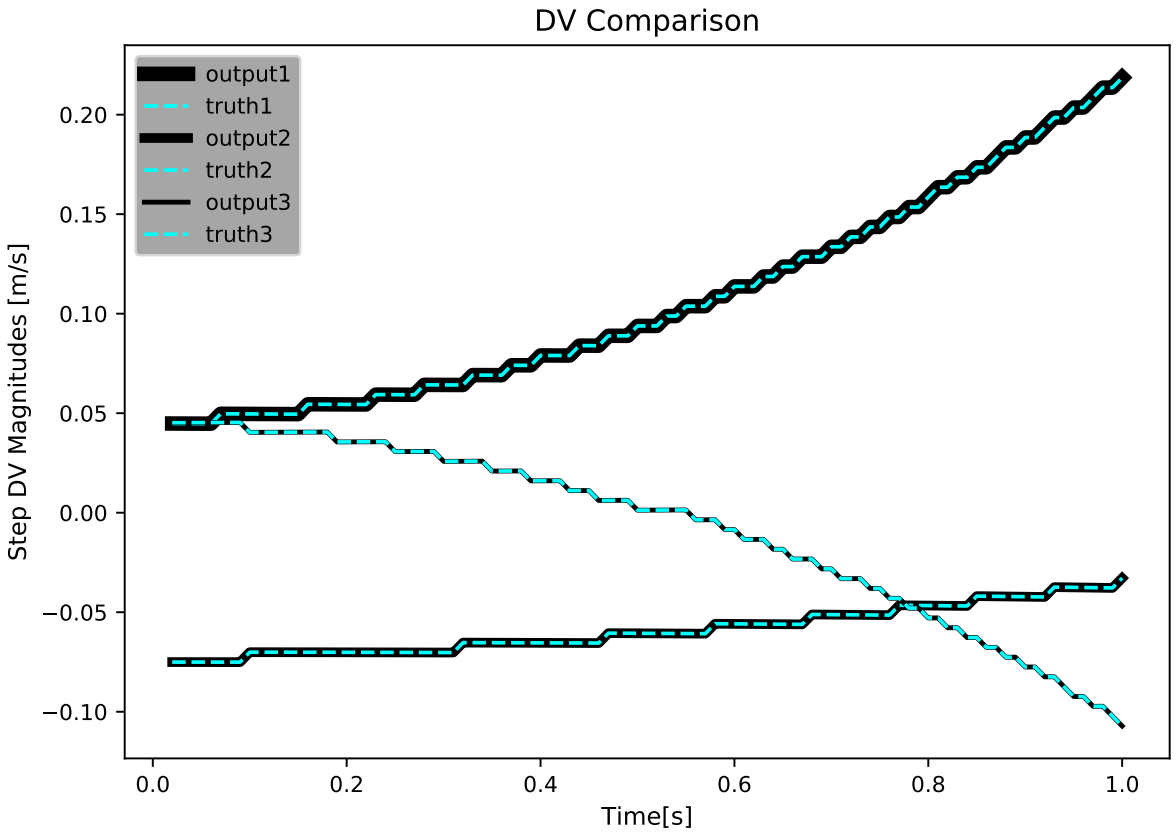


**Fig. 18:** Plot Comparing Time Step DV Truth and Output for test: discretization. Note that 1, 2, and 3 indicate the components of the velocity delta.

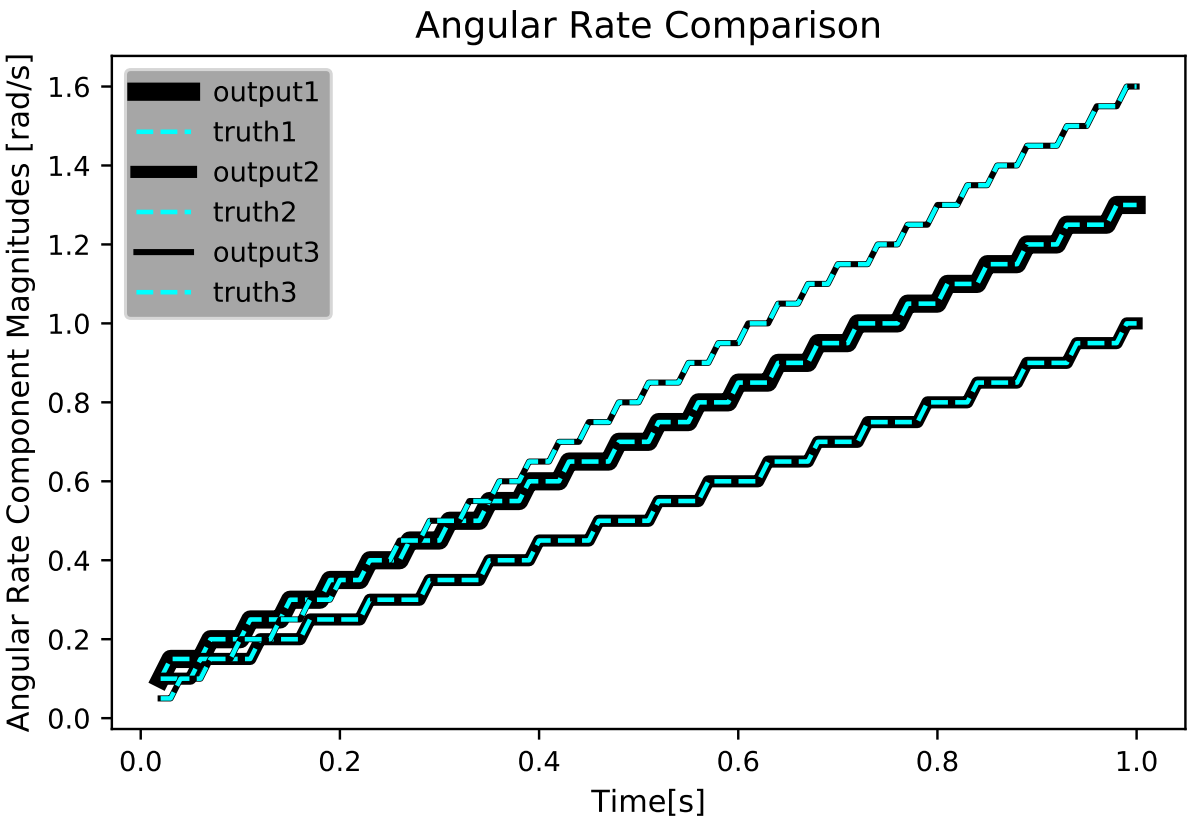


**Fig. 19:** Plot Comparing Angular Rate Truth and Output for test: discretization. Note that 1, 2, and 3 indicate the components of the angular rate.

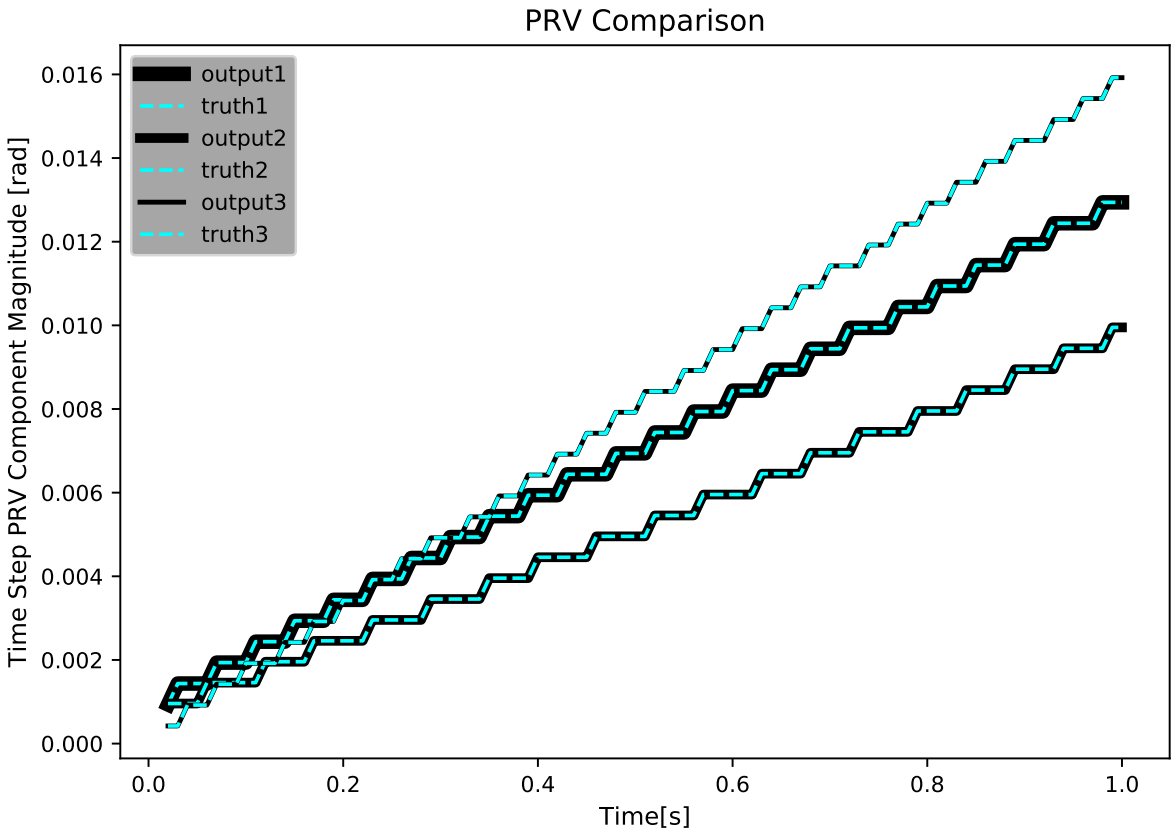


**Fig. 20:** Plot Comparing Time Step PRV Truth and Output for test: discretization. Note that 1, 2, and 3 indicate the components of the principal rotation vector.

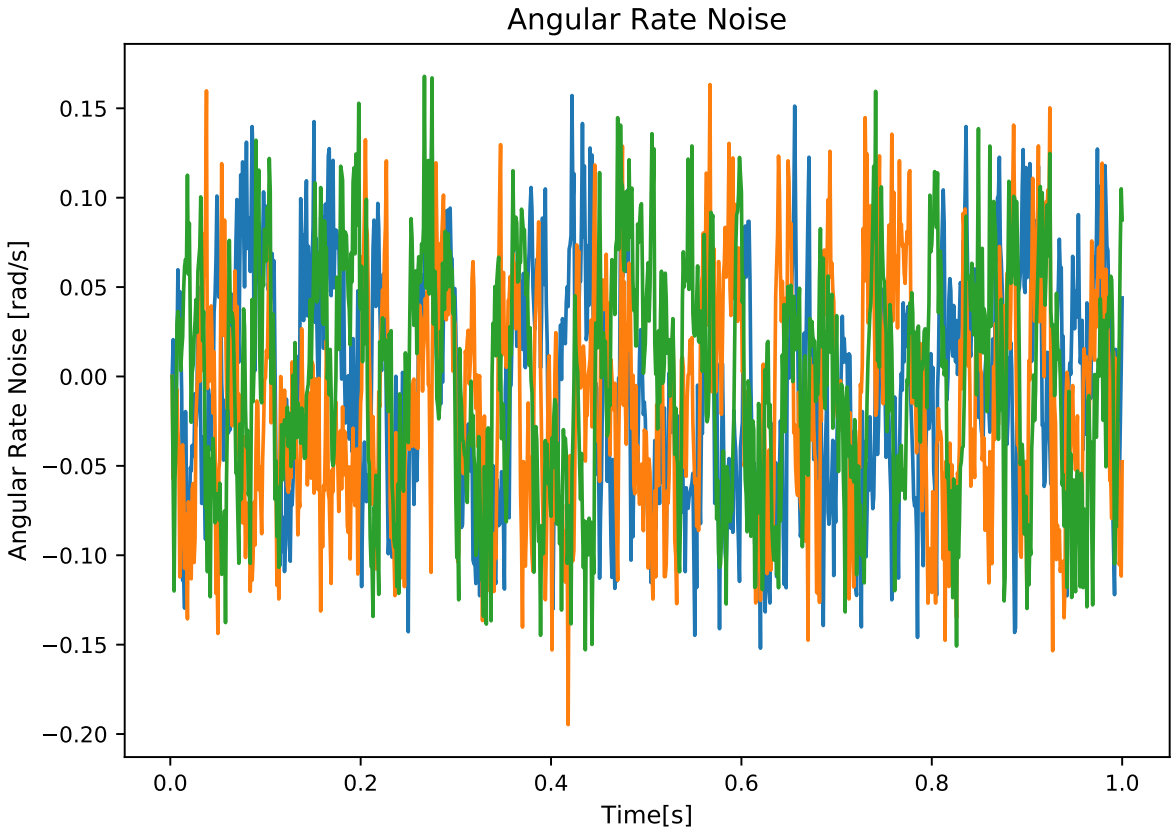


**Fig. 21:** Plot of Angular Rate noise along each component for the noise test.

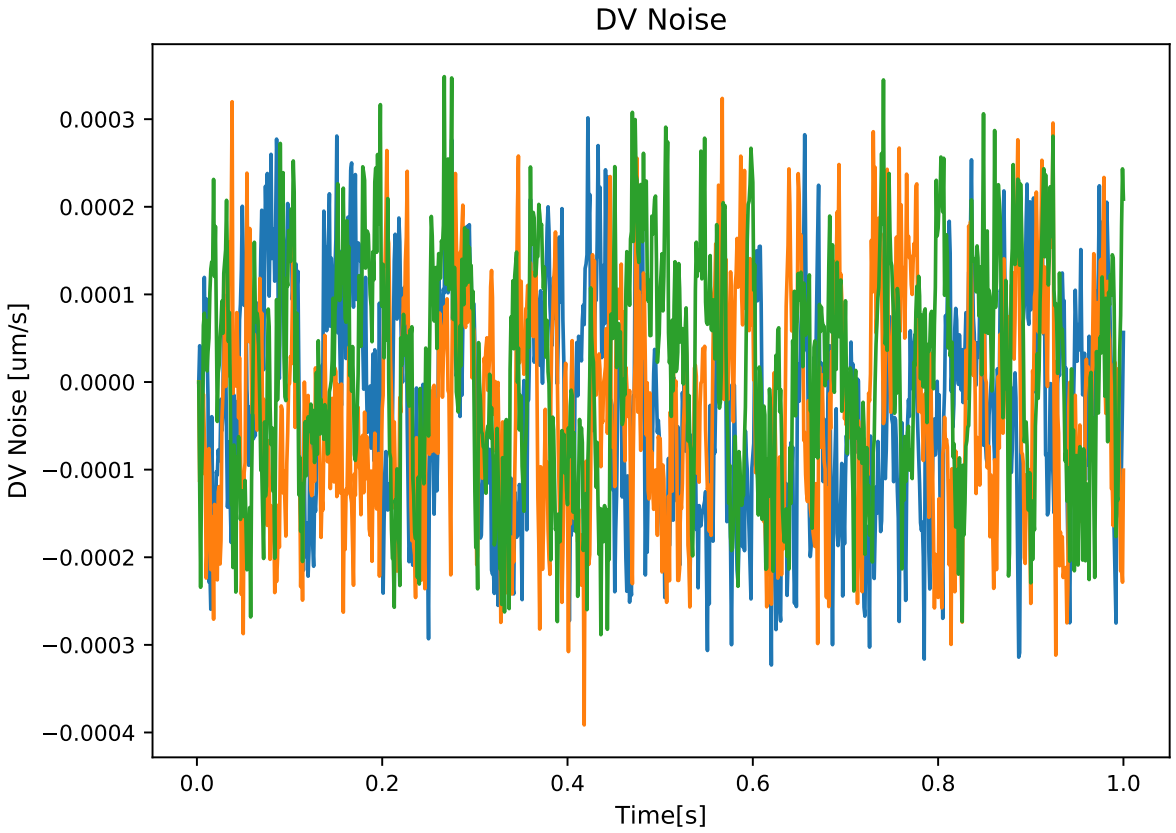


**Fig. 22:** Plot of DeltaV noise along each component for the noise test.

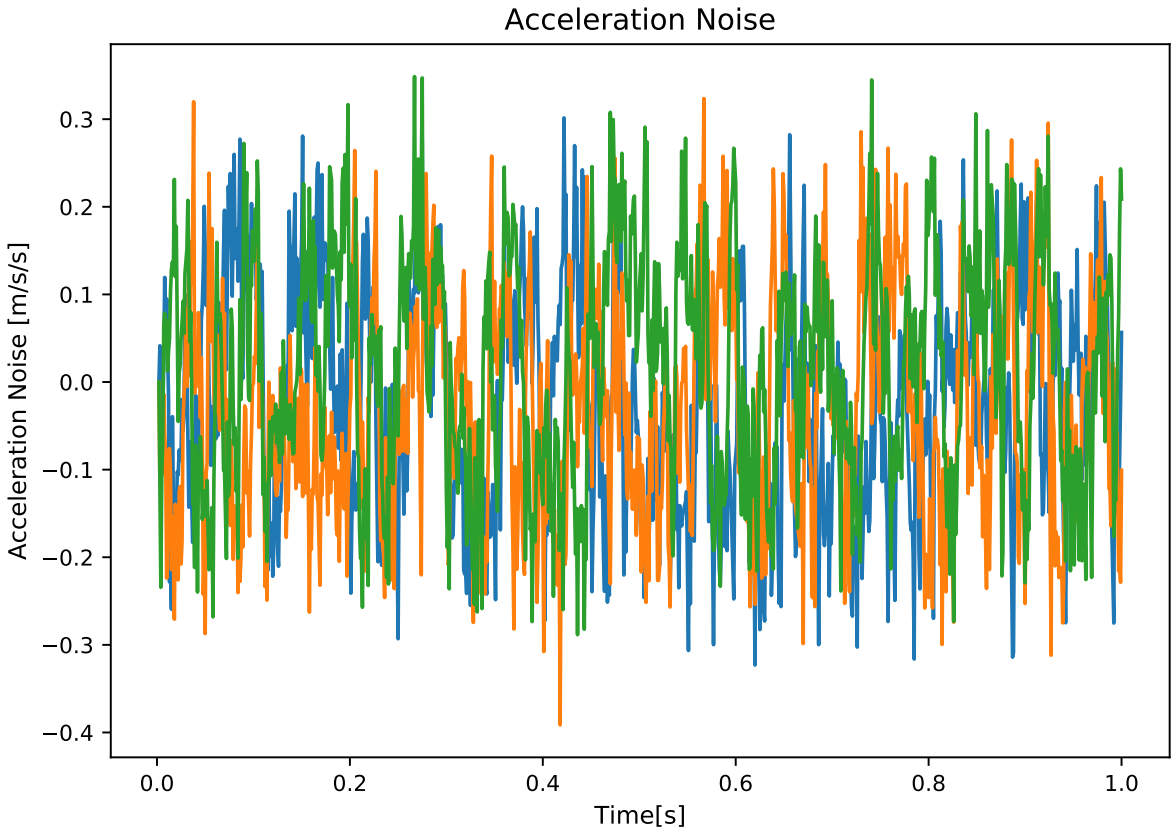


**Fig. 23:** Plot of acceleration noise along each component for the noise test.

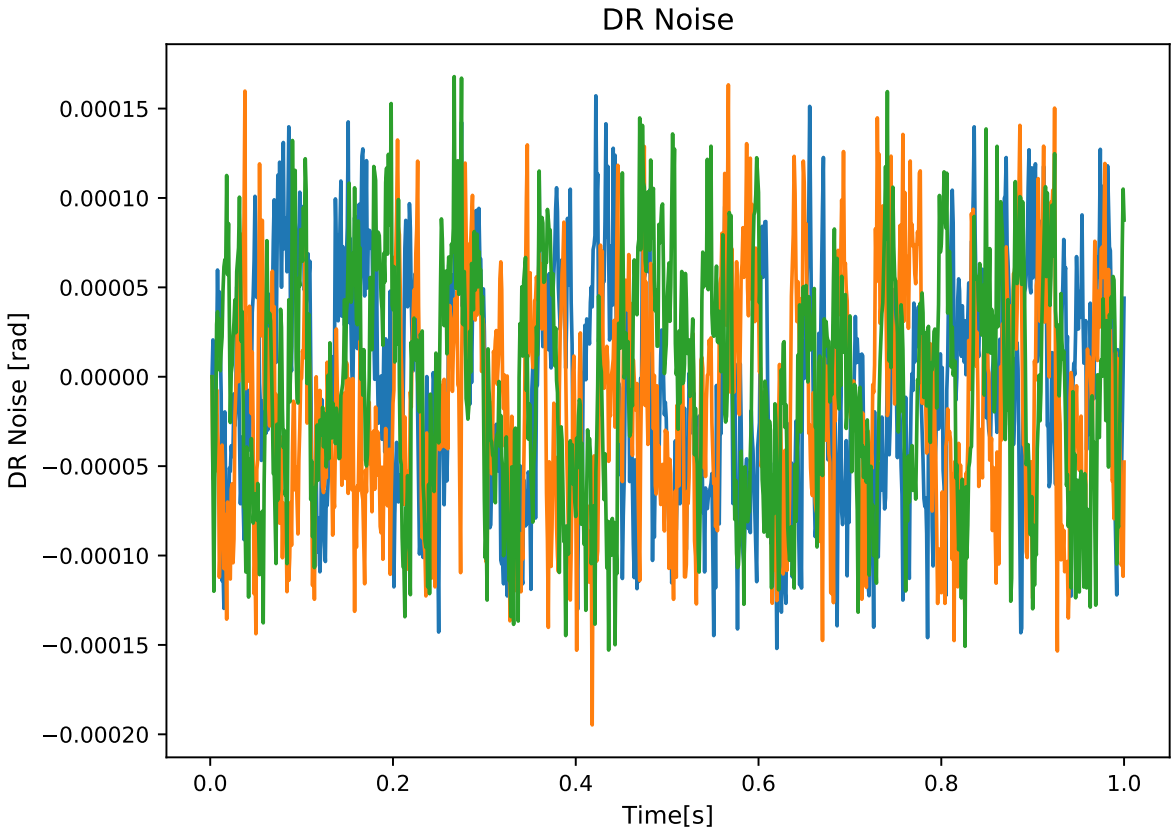


**Fig. 24:** Plot of PRV noise along each component for the noise test.

# 7 User Guide

This section contains conceptual overviews of the code and clear examples for the prospective user.

## 7.1 Code Diagram

The diagram in Fig. 25 demonstrates the basic logic of the IMU module. There is additional code that deals with auxiliary functions. An example of IMU use is given in test_imu_sensor.py in the imu_sensor_UnitTest folder. Application of each IMU function follows a simple, linear progression until realistic IMU outputs are achieved and sent out via the messaging system.

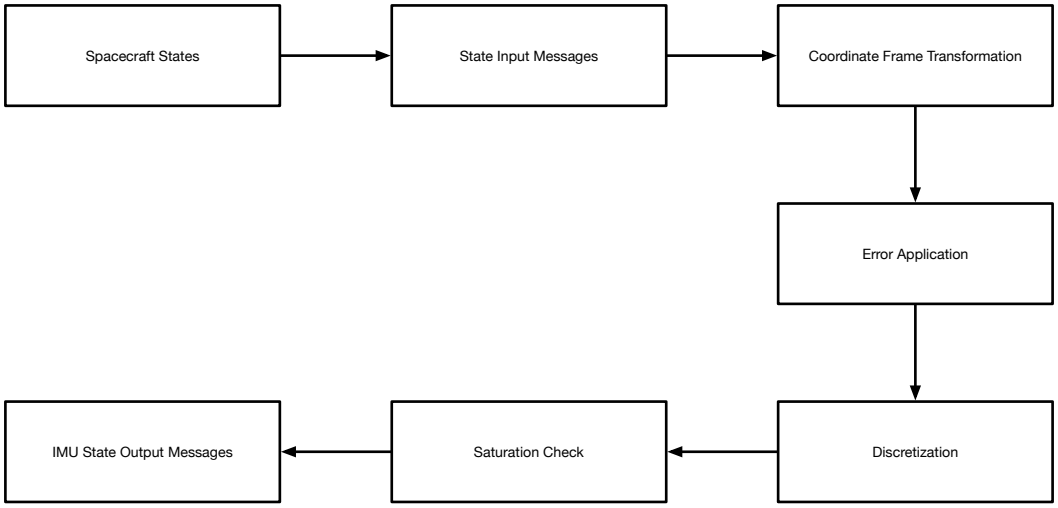


**Fig. 25:** A pseudo-code diagram demonstrating the flow of inputs and outputs in the IMU module.

## 7.2 Variable Definitions

The variables in Table 4 are available for user input. Variables used by the module but not available to the user are not mentioned here. Variables with default settings do not necessarily need to be changed by the user, but may be.

**Table 4:** Definition and Explanation of Variables Used.

| Variable | LaTeX Equivalent | Variable Type | Notes |
|---|---|---|---|
| InputStateMsg | N/A | string | Default setting: "inertial_state_output". This is the message from which the IMU receives spacecraft inertial data. |
| OutputDataMsg | N/A | string | Default setting: "imu_meas_data". This message contains the information output by the IMU. |
| SensorPos_B | N/A | double [3] | [m] Required input - no default. This is the sensor position in the body frame relative to the body frame. |
| roll, pitch, yaw | N/A | double, double, double | Default setting: (0,0,0). To set non-zero initial angles between imu and spacecraft body, call setBodyToPlatformDCM(roll, pitch, yaw) |
| dcm_PB | N/A | double [3][3] | Default setting: Identity. Setting dcm_PB is equivalent to calling setBodyToPlatformDCM(roll, pitch yaw) above. Use one method or the other. |
| senRotBias | $\mathbf{e}_{\text{bias}}$ | double [3] | [r/s] Default setting: zeros. This is the rotational sensor bias value for each axis. |
| senTransBias | $\mathbf{e}_{\text{bias}}$ | double [3] | [m/s2] Default setting: zeros. This is the translational sensor bias value for each axis |
| senRotMax | $\mathbf{m}_{\text{sat,max}}$ | double | [r/s] Required input - no default. This is the gyro saturation value. |
| senTransMax | $\mathbf{m}_{\text{sat,max}}$ | double | [m/s2] Required input - no default. This is the accelerometer saturation value. |
| PMatrixAccel | N/A yet | double [3][3] | Default: zeros. This is the covariance matrix used to perturb the state. |
| PMatrixGyro | N/A yet | double [3][3] | Default: zeros. This is the covariance matrix used to perturb the state. |
| walkBoundsGyro | N/A yet | double [3] | Default: zeros. This is the "3-sigma" errors to permit for gyro states |
| walkBoundsAccel | N/A yet | double [3] | Default: zeros. This is the "3-sigma errors ot permit for acceleration states. |
| accelLSB | (LSB) | double | Default: 0.0. This is the discretization value (least significant bit) for acceleration. Zero indicates no discretization. |
| gyroLSB | (LSB) | double | Default: 0.0. This is the discretization value (least significant bit) for acceleration. Zero indicates no discretization. |