Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-imu

INERTIAL MEASUREMENT UNIT C++ MODEL

Prepared by	S. Carnahan

Status: Tested
Scope/Contents
The Basilisk Inertial Measurement Unit (IMU) module is responsible for producing sensed body rates and acceleration from calculated values. It also provides a change in velocity and change in attitude value for the time between IMU calls. The IMU module applies Gauss-Markov process noise to the true body rates and acceleration. A unit test has been written which validates outputs of the IMU.

Rev	Change Description	By	Date
1.0	First draft	J. Alcorn	20170713
1.1	Added Math Documentation and User Guide. Implemented AutoTeX	S. Carnahan	20170713
1.2	Rewrote for new IMU implementation and test	S. Carnahan	20170914

Contents

1 Model Description

There two types of RW control torque interfaces, analog and digital. This modules assumes the RW is controlled through a set of voltages sent to the RW motors. This module is developed in a general manner where a voltage deadband is assumed and the module can be run in a pure open-loop manner, or with a closed-loop torque tracking control mode. Finally, if a RW availability message is present, then the RW is set to zero if the corresponding availability is set to `UNAVAILABLE`.

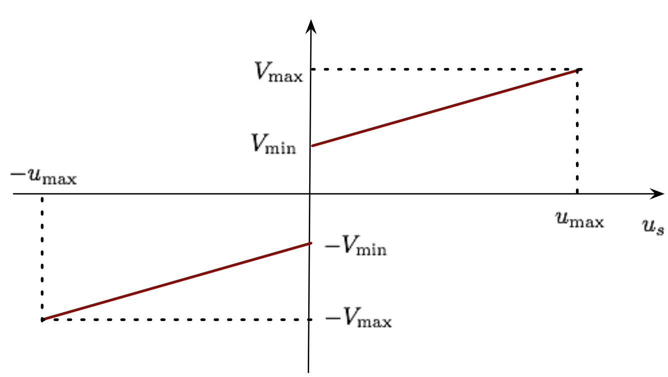

Fig. 1: Illustration of RW motor torque to voltage conversion

1.1 Open-loop voltage conversion

This module requires the RW configuration message to contain the maximum RW motor torque values u_{max}. The user must specify the minimum and maximum output voltages as shown in Figure 1. The minimum voltage is a voltage below which the motor doesn't apply a torque, i.e. a deadzone.

Let the intermediate voltage value V_{int} as

$$V_{\text{int}} = \frac{V_{\text{max}} - V_{\text{min}}}{u_{\text{max}}} u_s \tag{1}$$

The output voltage is thus determined through

$$V = V_{\text{int}} + V_{\text{min}} * \text{sgn}(V_{\text{int}}) \tag{2}$$

1.2 RW Availability

If the input message name `rwAvailInMsgName` is defined, then the RW availability message is read in. The voltage mapping is only performed if the individual RW availability setting is `AVAILABLE`. If it is `UNAVAILABLE` then the output voltage is set to zero.

1.3 Closed-loop commanded torque tracking

The requested RW motor torque is given by u_s. The RW wheel speed Ω is monitored to see if the actual torque being applied matches the commanded torque. Let J_s be the RW spin inertia about the RW spin axis $\hat{\boldsymbol{g}}_s$. In the following development the motor torque equation is approximated as

$$u_s = J_s \dot{\Omega} \tag{3}$$

where the assumption is made that the spacecraft angular accelerations are small compared to the RW angular accelerations. The $\dot{\Omega}$ term is digitally evaluated using a backwards difference method:

$$\dot{\Omega}_n = \frac{\Omega_n - \Omega_{n-1}}{\Delta t} \tag{4}$$

Care is taken that the old RW speed information Ω_{n-1} is not used unless a history of wheel speeds is available, in particular, after a module reset. Thus, the actual RW torque is evaluated as

$$u_n = J_s \dot{\Omega}_n \tag{5}$$

Finally, the closed loop motor torque value is computed with a proportional feedback component as

$$u_{s,CL} = u_s - K(u_n - u_s) \tag{6}$$

where $K > 0$ is a positive feedback gain value. Finally, this $u_{s,CL}$ is fed to the voltage conversion process in Eq. (1).

2 Model Functions

The code performs the following functions:

- **Reset**: Resets the rwMotorVoltage module to original settings.
- **Closed loop**: Evaluates commanded torque based on closed loop controls.
- **Torque to Voltage**: Uses gains to convert a given torque to a voltage output.
- **Saturation**: Checks that calculated output is within min/max bounds.

3 Model Assumptions and Limitations

This code makes the following assumptions:

- **Linear**: This code assumes that there is a linear relationship between the torque desired and the voltage required to create that torque.

4 Unit Test Description

A series of unit tests are performed to check the validity of this module's operation.

4.1 Test 1

The first test uses an input vector of $\boldsymbol{u}_s = [0.05, 0.0, -0.15, -0.2]$ Nm. The RW spin inertia is set to $J_s = 0.1$ kg m^s, while the maximum RW motor torque is set to $u_{\text{max}} = 0.2$ Nm. In this test case no RW availability or wheel speed messages are set. The simulation is first run for 1.5 seconds with a 0.5 second control update period. Next, the module is reset and run for another 1.5 sections. With only the open-loop voltage conversion active and the RW motor torque The resulting actual values and differences with hand-computed values are shown in Table 2. The reset of the module should have not impact on the voltage conversion, which is the case.

4.2 Test 2

This test repeats the values of test 1, except that the RW motor torque input vector is set to $\boldsymbol{u}_s = [0.5, 0.0, -0.15, -0.5]$ Nm. This should saturate the first and last RW voltage output, which is seen in Table 3.

4.3 Test 3

This test repeats the values of test 1, except that a RW availability message is created. Here all RWs have a status of `AVAILABLE` except for the 3rd RW which is `UNAVAILALBE`. The 3rd RW voltages should thus all be 0.0 in this case, which is seen in Table 4.

4.4 Test 4

This test repeats the values of test 1, except that a RW wheel speed message is created. The feedback gain is set to $K = 1.5$. For the first 1.0 seconds the RW wheel speeds are set to $\boldsymbol{\Omega} = [1.0, 2.0, 1.5, -3.0]$ rad/sec. At 1.0 seconds the RW wheel speeds are set to $\boldsymbol{\Omega} = [1.1, 2.1, 1.1, -4.1]$ rad/sec. Then the module is reset at 1.5 seconds and the simulation continued for another 1.5 second for a 3 second total simulation time. The speed message remains the same after the reset. Table 5 shows the results of the actual values, and the differences with the hand-computed values.

5 Test Parameters

Test parameters are interspersed with the test description above.

6 Test Results

Results for each test are shown in the tables below:

All of the tests passed:

Table 6: Test results

Test	Pass/Fail
1	PASSED
2	PASSED
3	PASSED
4	PASSED

Table 2: RW voltage output for case `useLargeVoltage = False`, `useAvailability = False`, `useTorqueLoop = False`.

time [s]	$u_{s,1}$	Error	$u_{s,2}$	Error	$u_{s,3}$	Error	$u_{s,4}$	Error
0	3.5	0	0	0	-8.5	0	-11	0
0.5	3.5	0	0	0	-8.5	0	-11	0
1	3.5	0	0	0	-8.5	0	-11	0
1.5	3.5	0	0	0	-8.5	0	-11	0
2	3.5	0	0	0	-8.5	0	-11	0
2.5	3.5	0	0	0	-8.5	0	-11	0
3	3.5	0	0	0	-8.5	0	-11	0

Table 3: RW voltage output for case `useLargeVoltage = True`, `useAvailability = False`, `useTorqueLoop = False`.

time [s]	$u_{s,1}$	Error	$u_{s,2}$	Error	$u_{s,3}$	Error	$u_{s,4}$	Error
0	11	0	0	0	-8.5	0	-11	0
0.5	11	0	0	0	-8.5	0	-11	0
1	11	0	0	0	-8.5	0	-11	0
1.5	11	0	0	0	-8.5	0	-11	0
2	11	0	0	0	-8.5	0	-11	0
2.5	11	0	0	0	-8.5	0	-11	0
3	11	0	0	0	-8.5	0	-11	0

Table 4: RW voltage output for case `useLargeVoltage = False`, `useAvailability = True`, `useTorqueLoop = False`.

time [s]	$u_{s,1}$	Error	$u_{s,2}$	Error	$u_{s,3}$	Error	$u_{s,4}$	Error
0	3.5	0	0	0	0	0	-11	0
0.5	3.5	0	0	0	0	0	-11	0
1	3.5	0	0	0	0	0	-11	0
1.5	3.5	0	0	0	0	0	-11	0
2	3.5	0	0	0	0	0	-11	0
2.5	3.5	0	0	0	0	0	-11	0
3	3.5	0	0	0	0	0	-11	0

Table 5: RW voltage output for case `useLargeVoltage = False`, `useAvailability = False`, `useTorqueLoop = True`.

time [s]	$u_{s,1}$	Error	$u_{s,2}$	Error	$u_{s,3}$	Error	$u_{s,4}$	Error
0	3.5	0	0	0	-8.5	0	-11	0
0.5	3.5	0	0	0	-8.5	0	-11	0
1	3.5	0	0	0	-8.5	0	-11	0
1.5	5.75	-1.77636e-15	-2.5	-1.33227e-15	-11	0	-9.5	-5.32907e-15
2	3.5	0	0	0	-8.5	0	-11	0
2.5	3.5	0	0	0	-8.5	0	-11	0
3	7.25	0	0	0	-11	0	-11	0

7 User Guide

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

This module is often called similar to:

```
from Basilisk.fswAlgorithms import rwMotorVoltage
cmd2volt = rwMotorVoltage.rwMotorVoltageConfig()
cmd2voltWrap = sim.setModelDataWrap(cmd2volt)
cmd2voltWrap.ModelTag = "commandToVoltageConverter"
cmd2volt.VMin = -100.
cmd2volt = 100.
cmd2volt.K = 5.
cmd2volt.voltageOutMsgName = "voltageCommands"
cmd2volt.torqueInMsgName = "torquesToBeConverted"
cmd2volt.rwParamsInMsgName = "reactionWheelParameters"
cmd2volt.inputRWSpeedsInMsgName = "reactionWheelSpeeds"
```

where sim is the simulation you are running. The wrap is used because this is a .c file rather than a .c++, so it helps Basilisk to interface with the module routines. The only thing important about the InMsgNames is that they match the output message of the appropriate modules that create the messages.

—