Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-mrpRotation

MRP ROTATION ATTITUDE GUIDANCE MODULE

Prepared by	H. Schaub

Status: First Release
Scope/Contents
This attitude guidance module creates a dynamic reference frame attitude state message where the initial orientation relative to the input reference frame is specified through an MRP set, and the angular velocity vector is held fixed as seen by the resulting reference frame. Besides setting the desired MRP and angular velocity values directly within the module, they can also be read in from an optional input message. This input message is checked for new content with each update call. The reset process will cause these desired attitude states from to be re-read from this input message. If a reset is called without an input attitude state message, then the last stored attitude states are used to continue the rotation. Optionally, the compute attitude states relative to the input frame can be fed to an output message as well.

Rev	Change Description	By	Date
1.0	Initial Release	H. Schaub	May 22, 2018

Contents

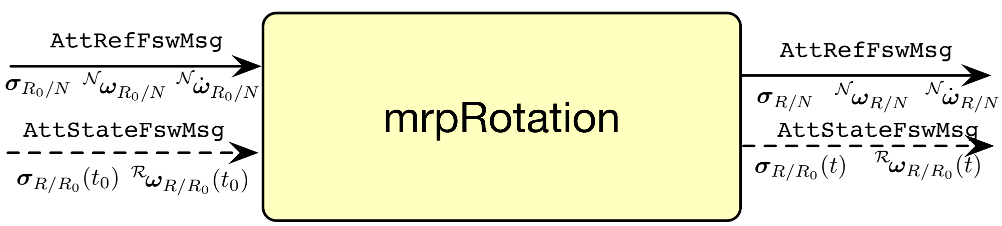

Fig. 1: Illustration of the mrpRotation() input and output messages. Required messages are shown with a solid line, while optional message have a dashed line.

1 Model Description

The purpose of this `mrpRotation` module is to add a constant rotation relative to the input frame $\mathcal{R}_0$. The output reference frame is called $\mathcal{R}$. The initial orientation is specified through an MRP[1] set $\boldsymbol{\sigma}_{R/R0}$, while the $\mathcal{R}$-frame angular velocity vector ${}^{\mathcal{R}}\boldsymbol{\omega}_{R/R0}$ is held constant in this module.

Assume that the input reference frame $\mathcal{R}_0$ is given through an attitude state input message containing $\boldsymbol{\sigma}_{R_0/N}$, ${}^{\mathcal{N}}\boldsymbol{\omega}_{R_0/N}$ and ${}^{\mathcal{N}}\dot{\boldsymbol{\omega}}_{R_0/N}$ as illustrated in Figure 1. The MRP set is mapped into the corresponding Direction Cosine Matrix or DCM[1] using

$$[R_0N] = [R_0N(\boldsymbol{\sigma}_{R_0/N})] \tag{1}$$

The goal of the motion is to compute the attitude of $\mathcal{R}$ relative to input frame $\mathcal{R}_0$ such that

$$\dot{\boldsymbol{\sigma}}_{R/R_0} = \frac{1}{4}[B(\boldsymbol{\sigma}_{R/R_0})]^{\mathcal{R}}\boldsymbol{\omega}_{R/R_0} \tag{2}$$

$$\frac{{}^{\mathcal{R}}\mathrm{d}\boldsymbol{\omega}_{R/R_0}}{\mathrm{d}t} = \mathbf{0} \tag{3}$$

Assume the initial $\boldsymbol{\sigma}_{R/R_0}(t_0)$ set and the $\mathcal{R}$-frame relative invariant ${}^{\mathcal{R}}\boldsymbol{\omega}_{R/R_0}$ vector are provided to the module. The current $\boldsymbol{\sigma}_{R/R_0}(t_0)$ value is then obtained by Eq. (2). The current DCM of the $\mathcal{R}$-frame is thus

$$[RN] = [RR_0(\boldsymbol{\sigma}_{R/R_0}(t)][R_0N] \tag{4}$$

Next, the angular velocity vector is transformed to inertial frame $\mathcal{N}$-frame components using

$${}^{\mathcal{N}}\boldsymbol{\omega}_{R/R_0} = [RN]^T \, {}^{\mathcal{R}}\boldsymbol{\omega}_{R/R_0} \tag{5}$$

to find the inertial angular velocity of the output reference frame:

$${}^{\mathcal{N}}\boldsymbol{\omega}_{R/N} = {}^{\mathcal{N}}\boldsymbol{\omega}_{R/R_0} + {}^{\mathcal{N}}\boldsymbol{\omega}_{R_0/N} \tag{6}$$

Finally, the inertial angular acceleration of the output reference frame is found using the transport theorem:

$$\dot{\boldsymbol{\omega}}_{R/N} = \frac{{}^{\mathcal{R}}\mathrm{d}\boldsymbol{\omega}_{R/R_0}}{\mathrm{d}t} + \boldsymbol{\omega}_{R/N} \times \boldsymbol{\omega}_{R/R_0} + \dot{\boldsymbol{\omega}}_{R_0/N} = \boldsymbol{\omega}_{R_0/N} \times \boldsymbol{\omega}_{R/R_0} + \dot{\boldsymbol{\omega}}_{R_0/N} \tag{7}$$

where $\boldsymbol{\omega}_{R/N} \times \boldsymbol{\omega}_{R/R_0} = (\boldsymbol{\omega}_{R/R_0} + \boldsymbol{\omega}_{R_0/N}) \times \boldsymbol{\omega}_{R/R_0} = \boldsymbol{\omega}_{R_0/N} \times \boldsymbol{\omega}_{R/R_0}$ is used. Expressed in $\mathcal{N}$ frame components, this vector equation is numerically evaluated using:

$${}^{\mathcal{N}}\dot{\boldsymbol{\omega}}_{R/N} = {}^{\mathcal{N}}\boldsymbol{\omega}_{R_0/N} \times {}^{\mathcal{N}}\boldsymbol{\omega}_{R/R_0} + {}^{\mathcal{N}}\dot{\boldsymbol{\omega}}_{R_0/N} \tag{8}$$

2 Module Functions

The `mrpRotation` module has the following design goals

- **Constant Spin**: The angular velocity vector between the input and output frame is constant as see by output reference frame
- **MRP attitude representation**: The initial and output attitude is described through an MRP coordinate set
- **Flexible Setup**: The desired rotation state can be described through an initial MRP and angular velocity vector specified in module internal variables, or read in through a Basilisk `AttStateFswMsg` message.

3 Module Assumptions and Limitations

- On reset the next time step doesn't yield an integration as the integration time evaluation requires at least a second time step.
- If the desired rotation states are read in with an input message, then this message is checked each update cycle for new content. On reset the commanded frame states are reset to zero such that they are re-read in again in the next update cycle.
- If the desired rotation is specified with module internal states, then on reset the prior internal states are re-used unless they are over-written after the reset call.

4 Test Description and Success Criteria

The module is run on its own with specified inputs to ensure the outputs are correct. The outputs are evaluated dynamically using a support python script, and then compared to the Basilisk evaluated results. A nominal simulation length of 1 second is used with a time step of 0.5 seconds, yielding 3 return values.

If the rotation states are set directly in the module by specifying `mrpSet` and `omega_RR0_R`, then the values $\boldsymbol{\sigma}_{R/R_0}(t_0)$ =[0.3 0.5 0.] and ${}^{\mathcal{R}}\boldsymbol{\omega}_{R/R_0}$ =[0.1 0. 0.]deg/sec are used. The simulation flag `cmdStateFlag` determines if the rotation states are specified through an input message. If yes, then the values $\boldsymbol{\sigma}_{R/R_0}(t_0)$ =[0.1 0. -0.2] and ${}^{\mathcal{R}}\boldsymbol{\omega}_{R/R_0}$ =[0.1 1. 0.5]deg/sec are used instead.

If the simulation flag `stateOutputFlag` is true then the optional attitude states of $\mathcal{R}$ relative to $\mathcal{R}_0$ are provided in an output message.

If the simulation flag `testReset` is true then the simulation will run an addition 1 second, but after a reset function is called.

Table 2: Test Scenarios.

Check	`cmdStateFlag`	`stateOutputFlag`	`testReset`
1	False	False	False
2	True	False	False
3	False	True	False
4	False	False	True
5	True	False	True

5 Test Parameters

The output variables being tested are listed in Table 3, including the test tolerance value.

Table 3: Error tolerance for each test.

Output Value Tested	Tolerated Error
`attRefOutMsgName.sigma_RN`	1e-12
`attRefOutMsgName.omega_RN_N`	1e-12
`attRefOutMsgName.domega_RN_N`	1e-12
`attitudeOutMsgName.state`	1e-12
`attitudeOutMsgName.rate`	1e-12

6 Test Results

The results of the unit test are listed in Table 4. All of the tests passed:

Table 4: Test results

Check	Pass/Fail
1	PASSED
2	PASSED
3	PASSED
4	PASSED
5	PASSED

7 User Guide

7.1 Specifying Desired Rotation

If the `mrpRotation` module is set directly with the desired rotation states, then the modules variables `mrpSet` and `omega_RR0_R` must be set.

If instead the desired rotation states are to be read in, then then input message name `desiredAttInMsgName` must be specified, and a corresponding message of type `AttStateFswMsg` created.

7.2 Required Input and Output Messages

The $\mathcal{R}_0$ input reference frame state message is specified through `attRefInMsgName`. The output message name is specified through the `attRefOutMsgName`.

7.3 Optional Output Message

If desired, the $\boldsymbol{\sigma}_{R/R_0}$ and ${}^{\mathcal{R}}\boldsymbol{\omega}_{R/R_0}$ states can be written to the message `attitudeOutMsgName`. If this message name is not set, then this output message is not created.

7.3.1 Module Reset Behavior

If the module is reset, then the `priorTime` flag is reset, meaning it take another time step to compute the sampling period used to integrate the kinematic differential equations.

REFERENCES

[1] Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems*. AIAA Education Series, Reston, VA, 4th edition, 2018.