Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-thrFiringRemainder

ACCUMULATE UNIMPLEMENTED REQUESTED THRUSTER IMPULSES

Prepared by	H. Schaub

Status: First Release
Scope/Contents
A thruster force message is read in and converted to a thruster on-time output message. The module ensures the requested on-time is at least as large as the thruster's minimum on time. If not then the on-time is zeroed, but the unimplemented thrust time is kept as a remainder calculation. If these add up to reach the minimum on time, then a thruster pulse is requested. If the thruster on time is larger than the control period, then an on-time that is 1.1 times the control period is requested. .

Rev	Change Description	By	Date
1.0	First Release	H. Schaub	2019-03-28

Contents

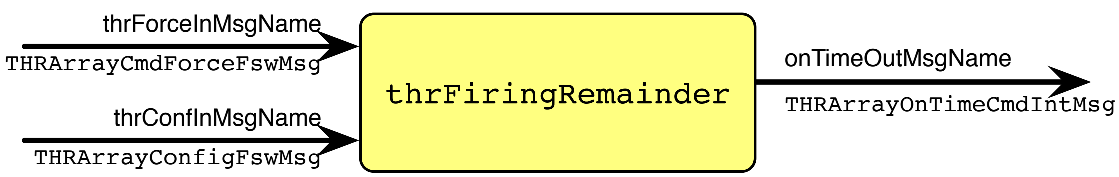

Fig. 1: Illustration of the module input and output messages.

1 Module Description

This module implements a remainder tracking thruster firing logic. More details can be found in Reference 1.

1.1 Module Input and Output Messages

As illustrated in Figure 1, the module reads in two messages. One message contains the thruster configuration message from which the maximum thrust force value for each thruster is extracted and stored in the module. This message is only read in on `Reset()`.

The second message reads in an array of requested thruster force values with every `Update()` function call. These force values F_i can be positive if on-pulsing is requested, or negative if off-pulsing is required. On-pulsing is used to achieve an attitude control torque onto the spacecraft by turning on a thruster. Off-pulsing assumes a thruster is nominally on, such as with doing an extended orbit correction maneuver, and the attitude control is achieved by doing periodic off-pulsing.

The output of the module is a message containing an array of thruster on-time requests. If these on-times are larger than the control period, then the thruster remains on only for the control period upon which the on-criteria is reevaluated.

1.2 Reset() Functionality

- The control period is dynamically evaluated in the module by comparing the current time with the prior call time. In `reset()` the `prevCallTime` variable is reset to 0.
- The thruster configuration message is read in and the number of thrusters is stored in the module variable `numThrusters`. The maximum force per thruster is stored in `maxThrust`.
- The on-time pulse remainder variable is reset for each thruster back to 0.0.

1.3 Update() Functionality

The goal of the `update()` method is to read in the current attitude control thruster force message and map these into a thruster on-time output message. Let Δt_{min} be the minimum on-time that can be implemented with a thruster. If the requested on-time is less than Δt_{min}, then the requested thruster on-time is clipped to zero. In the following algorithm unimplemented fractional on-times less than Δt_{min} are tracked and accumulated, providing additional pointing accuracy. For example, if the minimum on-time is 20 milli-seconds, an on-time algorithm without remainder calculation would create a deadband about the 20 milli-second control request. With the remainder logic, if 5 milli-second on-time requests are computed, these are accumulated such that every 4$^{\text{th}}$ control step a 20 milli-second burn is requested. This reduces the deadband behavior of the thruster and achieves better pointing. In this example the 5 milli-second un-implemented on-times are accumulated in the variable $\Delta t_{\text{partial}}$.

If the `update()` method is called for the first time after reset, then there is no knowledge of the control period Δt. In this case the thruster on-time values are set either to 0 (on-pulsing case) or 2 seconds (off-pulsing case). After writing out this message the module is exited. This logic is essence turns off the thruster-based attitude control for one control period.

If this is a repeated call of the `update()` method then the control period Δt is evaluated by differencing the current time with the prior call time. Next a loop goes over each installed thruster to map the requested force F_i into an on-time t_i. The following logic is used.

- If off-pulsing is used then $F_i \leqslant 0$ and we set

  $$F_i + = F_{\text{max}}$$

  to a reduced thrust to achieve the negative F_i force.

- Next, if $F_i < 0$ then it set to be equal to zero. This can occur if an off-pulsing request is larger than the maximum thruster force magnitude F_{max}.

- The nominal thruster on-time is computed using

  $$t_i = \frac{F_i}{F_{\text{max}}} \Delta t$$

  .

- If there un-implemented on-time requested $\Delta t_{\text{partial}}$ from earlier `update()` method calls, these are added to the current on-time request using

  $$t_i + = \Delta t_{\text{partial}}$$

  After this step the variable $\Delta t_{\text{partial}}$ is reset to 0 as the remainder calculation is stored in the on-time variable t_i.

- If $t_i < \Delta t_{\text{partial}}$ then on-time request is set to zero and the remained is set to $\Delta t_{\text{partial}} = t_i$
- If $t_i > \Delta$ then the requested force is larger than F_{max} and the control is saturated. In this case the on-time is set to $1.1\Delta t$ such that the thruster remains on through the control period.
- The final step is to store the thruster on-time into and write out this output message

2 Module Functions

- **Read in thruster configuration message**: This is used to determine the number of installed thrusters and what the maximum force is for each.
- **Convert thruster force requested into an on-time request**: Knowing how strong the thruster is, the on-time is scaled such that the effectively applied force is equal to the requested force.

3 Module Assumptions and Limitations

The module assumes that the incoming forces F_i can be both positive or negative, depending if an on- or off-pulsing mode is being implemented. The particular mode is set through `baseThrustState`.

4 Test Description and Success Criteria

The unit test creates a desired thruster force input vector and then runs the simulation for 3 seconds. If the `resetCheck` flag is true then a `reset()` method is called and the simulation is repeated for another 2.5 seconds. If the `dvOn` flag is set then the off-pulsing mode is checked.

5 Test Parameters

The simulation sets up 8 thrusters. All permutations with the `resetCheck` and `dvOn` states are run. The output is checked to the tolerance shown in Table 2.

Table 2: Error tolerance for each test.

Output Value Tested	Tolerated Error
`OnTimeRequest`	1e-12

6 Test Results

All of the tests passed:

Table 3: Test results

`resetCheck`	`dvOn`	**Pass/Fail**
False	False	PASSED
False	True	PASSED
True	False	PASSED
True	True	PASSED

7 User Guide

The following variables are all required parameter to operate this module:

- `thrForceInMsgName`: string containing the thruster force input message
- `thrConfInMsgName`: string containing the thruster configuration input message
- `onTimeOutMsgName`: string containing the thruster on-time output message

- `thrMinFireTime`: Minimum thruster on-time in seconds
- `baseThrustState`: Flag indicating either an on-pulsing (0) or off-pulsing (1) configuration

REFERENCES

[1] John Alcorn, Hanspeter Schaub, and Scott Piggott. Steady-state attitude and control effort sensitivity analysis of discretized thruster implementations. *AIAA Journal of Spacecraft and Rockets*, 54(5):1161–1169, 2017.