Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-Thrusters

THRUSTER DYNAMIC EFFECTOR MODULE

Prepared by	T. Teil

Status: Initial Document
Scope/Contents
This test compares the simulation's forces and torques due to thrusters with expected values. This test runs a variety of thrust scenarios, creates truth values, and compares them point by point to the simulation. It therefore analyzes the thrust behavior down to test rate precision.

Rev	Change Description	By	Date
1.0	Initial Revision	T. Teil	11/2016
1.1	Validation Revision	T. Teil	07/2017

Contents

1 Model Description **1**

2 Introduction **1**
2.1 Output 1

3 Model Functions **1**

4 Model Assumptions and Limitations **1**
4.1 Assumptions 1
4.2 Limitations 1

5 Test Description and Success Criteria **2**
5.1 Test inputs 2
5.2 Test sections 2
5.3 Test success criteria 3

6 Test Parameters **3**

7 Test Results **4**
7.1 Pass/Fail 4
7.2 Test Coverage 12
7.3 Conclusions 12

8 User Guide **13**

---

1 Model Description

2 Introduction

The Thruster model in the AVS Basilisk simulation is used to emulate the effect of a vehicle's thrusters on the overall system. Its primary use is to generate realistic forces/torques on the vehicle structure and body. This is accomplished by apply a force at a specified location/direction in the body and using the current vehicle center of mass to calculate the resultant torque. Each individual thruster in a given model has its own ramp-up/ramp-down profile specified as part of its initialization data and it follows those profiles during start-up and shutdown.

The thruster model also contains a mechanism that is used to change the current vehicle mass properties as the thruster fires propellant overboard. This model uses the thruster ISP (specific impulse, also specified with configuration data) to calculate how much mass is being removed during a given thruster firing and decrements the mass properties included in the thruster linearly as a function of mass.

2.1 Output

This module computes forces and torques, and is called as a Dynamic Effector to integrate the state.

3 Model Functions

The Thrusters module contains methods allowing it to perform several tasks:

- **Set thrusters**: Define and set several thrusters with different parameters, and locations. Parameters include: Minimum On-Time, I_{sp}, direction of thrust...
- **Ramp On/Off**: Define ramps that model the imperfect on time and off time for a thruster
- **Compute Forces and Torques**: Gets the forces and torques on the SpaceCraft given the previous definitions

4 Model Assumptions and Limitations

4.1 Assumptions

The Thruster module assumes that the thruster is thrusting exactly along it's position axis. Even if the position is dispersed, the thrust will be constant along that dispersed axis and will not vary.

4.2 Limitations

The ramps in the thrusters modules are made by defining a set of elements to the ramp. They therefore form, by definition, a piecewise-linear ramp. If enough points are added, this will strongly ressemble a polynomial, but the ramps are in essence piecewise constant.

The I_{sp} used for each thruster is considered constant throughout a simulation. This means there is no practical way of doing variable I_{sp} simulations. It can nevertheless be done by stopping the simulation and modifying the values.

5 Test Description and Success Criteria

The unit test for the thruster_dynamics module is located in:

`SimCode/dynamics/Thrusters/_UnitTest/unit_ThrusterDynamicsUnit.py`

5.1 Test inputs

The thruster inputs that were used are:

- The specific impulse: $I_{sp} = 266.7$s

  A thrusters potential to deliver force per mass flow rate.

- The maximum thrust: $F_{\max} = 1.0$ N

  The scaling factor yielding the thrust.

- The minimum On time: $t_{\min} = 0.006$s

  The minimum time that the thruster can be fired.

5.2 Test sections

This unit test is designed to functionally test the simulation model outputs as well as get complete code path coverage. The test design is broken up into several parts:

1. Instantaneous On/Off Factor: The thrusters are fired with an instantaneous ramp to ensure that the firing is correct. This gives us a base case.
2. Short Instantaneous Firing: A "short" firing that still respects the rules of thumb above is fired to ensure that it is still correct.
3. Short Instantaneous Firing with faster test rate: A "short" firing that still respects the rules of thumb above but with a faster test rate to see the jump.
4. Instantaneous On/Off Factor with faster test rate: The thrusters are fired with an instantaneous ramp to ensure that the firing is correct given a different test rate. This shouldn't modify the physics.
5. Thruster Angle Test: The output forces/torques from the simulation are checked with a thruster pointing in a different direction.
6. Thruster Position Test: The output forces/torques from the simulation are checked with a thruster in a different position.
7. Thruster Number Test: The output forces/torques from the simulation are checked with two thruster in different positions, with different angles.
8. Ramp On/Ramp Off Firing: A set of ramps are set for the thruster to ensure that the ramp configuration is respected during a ramped firing.
9. Short ramped firing: A thruster is fired for less than the amount of time it takes to reach the end of its ramp.
10. Ramp On/Ramp Off Firing with faster test rate: A set of ramps are set for the thruster to ensure that the ramp configuration is respected during a ramped firing with different test rate.
11. Cutoff firing: A thruster is commanded off (zero firing time) in the middle of its ramp up to ensure that it correctly cuts off the current firing
12. Ramp down firing: A thruster is fired during the middle of its ramp down to ensure that it picks up at that point in its ramp-up curve and reaches steady state correctly.

These scenarios create a set of different runs. These are run in the same test using pytest parameters. Therefore 12 different parameter sets were created to cover all of the listed parts.

5.3 Test success criteria

This thrusters test is considered successful if, for each of the scenarios, the output data matches exactly the truth data that is computed in python. This means that at every time step, the thrust is the one that is expected down to near machine precision ($\epsilon = 10^{-9}$).

This leave no slack for uncertainty in the thrusters module.

6 Test Parameters

In order to have a rigorous automated test, we have to predict the forces and torques that will apply on the spacecraft. We use the following equations to compute the thrust at each time step. We call α the angle in which the thruster is pointing, $\boldsymbol{r} = r\hat{\boldsymbol{e}}_r = (r_x, r_y, r_z)$ it's position vector,

1. With one thruster: The forces are simply the projections of the thrust force on the axes of interest. The torque along x is the arm along z times the projection of the force along y, the torque along z is the arm along x times the projection of the force along y, the torque along z is the arm along x times the projection of the force along y.

$$F_x = F_{\max} \cos\alpha \qquad T_x = -F_{\max} \sin(\alpha) r_z \tag{1}$$
$$F_y = F_{\max} \sin\alpha \qquad T_y = F_{\max} \cos(\alpha) r \sin(\arctan(r_z/r_x)) \tag{2}$$
$$F_z = 0 \qquad T_z = F_{\max} \sin(\alpha) r_x \tag{3}$$

2. With two thrusters: By giving indices 1 and 2 for each of the thrusters, we just need to add the Forces and torques defined above to get the total Forces and Torques:

$$F_x = F_{x1} + F_{x2} \qquad T_x = T_{x1} + T_{x2} \tag{4}$$
$$F_y = F_{y1} + F_{y2} \qquad T_y = T_{y1} + T_{y2} \tag{5}$$
$$F_z = F_{z1} + F_{z2} \qquad T_z = T_{z1} + T_{z2} \tag{6}$$

3. With ramps thruster: When the thrusters now ramp up and down, we create a normalized ramp function ρ. An example is given in 1 in the case of a cutoff fire and renewed fire.

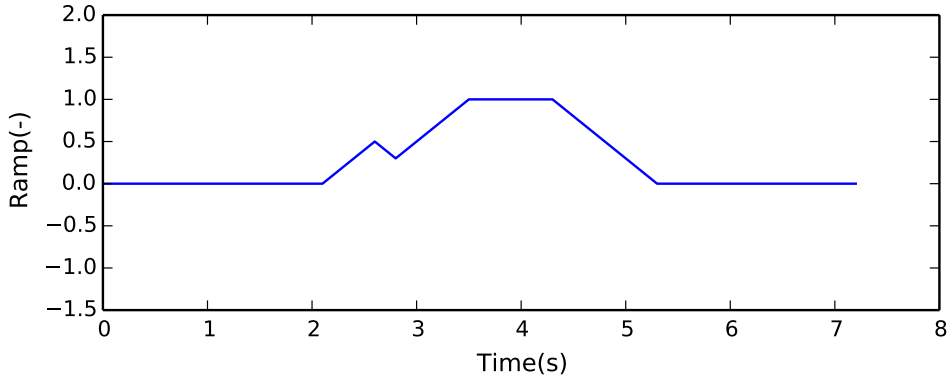

Fig. 1: Example of ramp function

We then prolong the force and torque end times as a function of the ramp slope, and multiply the initial functions by this ramping function:

$$\tilde{F}_x = \rho F_x \qquad \tilde{T}_x = \rho T_x \tag{7}$$
$$\tilde{F}_y = \rho F_y \qquad \tilde{T}_y = \rho T_y \tag{8}$$
$$\tilde{F}_z = \rho F_z \qquad \tilde{T}_z = \rho T_z \tag{9}$$

7 Test Results

7.1 Pass/Fail

Parameter Sets	Test Result	Error Tolerance
1	Passed	10^{-9}
2	Passed	10^{-9}
3	Passed	10^{-9}
4	Passed	10^{-9}
5	Passed	10^{-9}
6	Passed	10^{-9}
7	Passed	10^{-9}
8	Passed	10^{-9}
9	Passed	10^{-9}
10	Passed	10^{-9}
11	Passed	10^{-9}
12	Passed	10^{-9}

1. Instantaneous On/Off Factor:

   The thruster is set at 30° off the x-axis, in the position $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The test is launched using 1 thruster, for 5.0 seconds. The test rate is 10 steps per second

   Figures 2 and 3 show the force and torque behaviors (respectfully) for the thruster unit test.

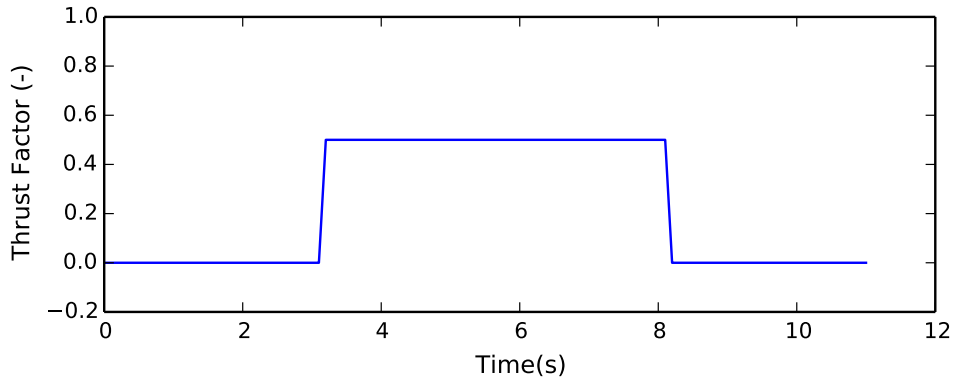

Fig. 2: Force on Y with 1 thrusters, for 5 sec at 30 deg Rate10

   As Figure 4 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

2. Short Instantaneous Firing:

   The thruster is set at 30° off the x-axis, in the position $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The test is launched using 1 thruster, for 0.1 seconds. The test rate is 10 steps per second

   Figure 5 shows the force behavior given this short input. We see that the test rate begin small next to the thrust duration, doesn't capture the jump quite well.

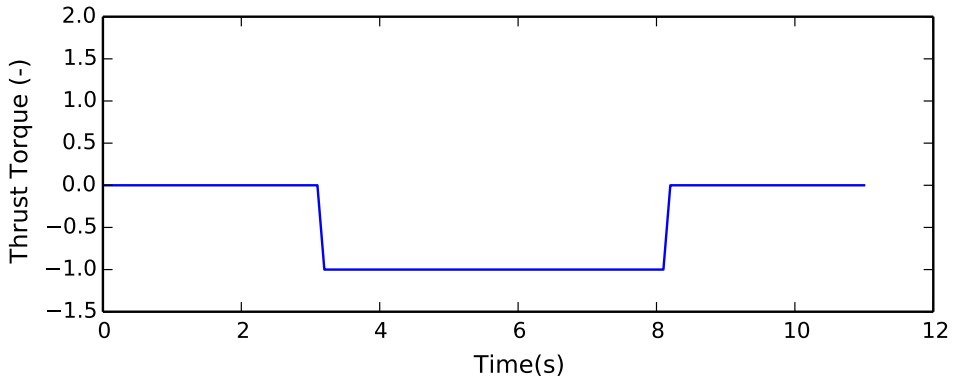

Fig. 3: Torque on X with 1 thrusters, for 5 sec at 30 deg Rate10

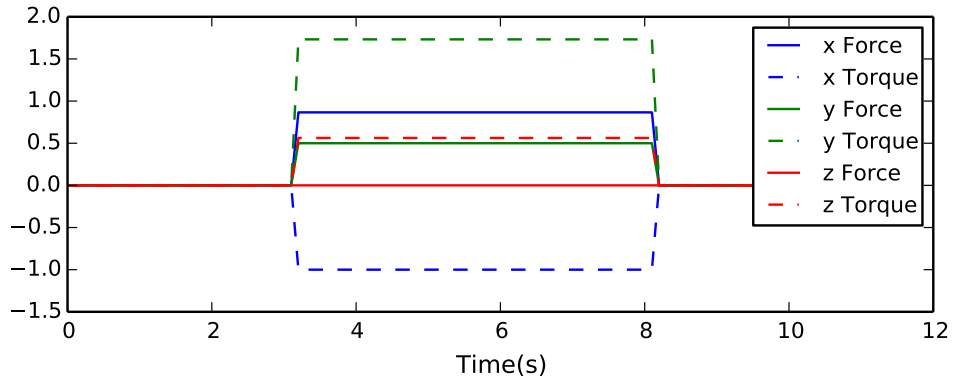

Fig. 4: All Forces and Torques 1 thrusters, for 5 sec at 30 deg Rate10

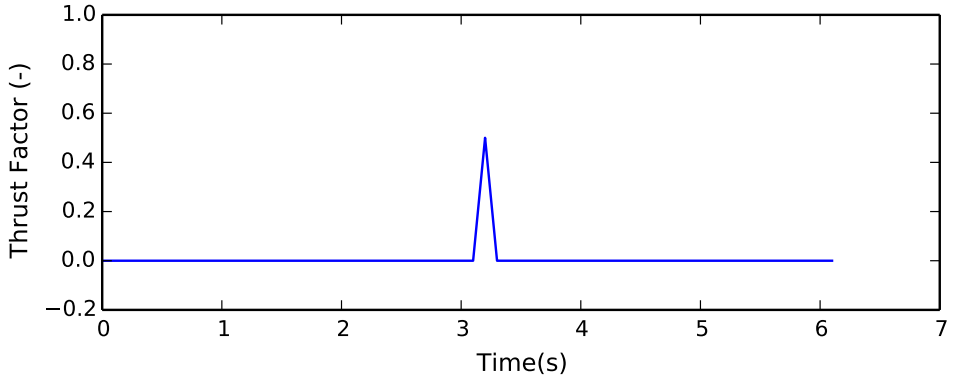

Fig. 5: Force on Y with 1 thrusters, for 0 sec at 30 deg Rate10

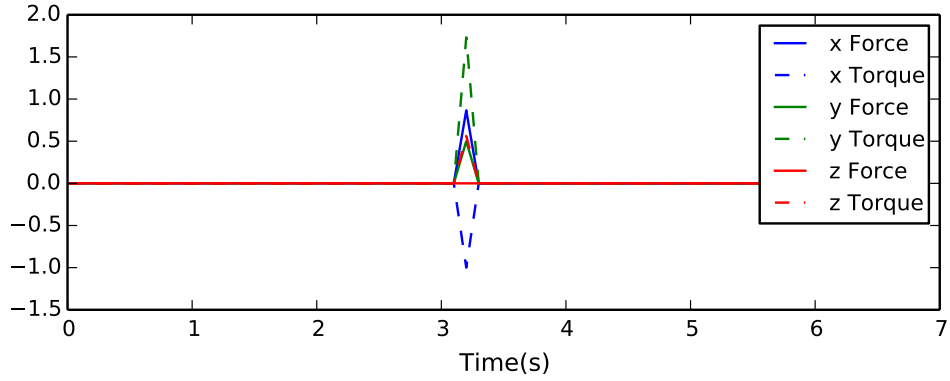

Fig. 6: All Forces and Torques 1 thrusters, for 0 sec at 30 deg Rate10

As Figure 6 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. Despite the lower test rate, the forces and torques behave appropriately. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

3. Short Instantaneous Firing with faster test rate:

   The thruster is set at 30° off the x-axis, in the position $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The test is launched using 1 thruster, for 0.1 seconds. The test rate is 1000 steps per second

   Figure 7 shows the force behavior given the same short input as previously. We now see that the jump is well resolved.

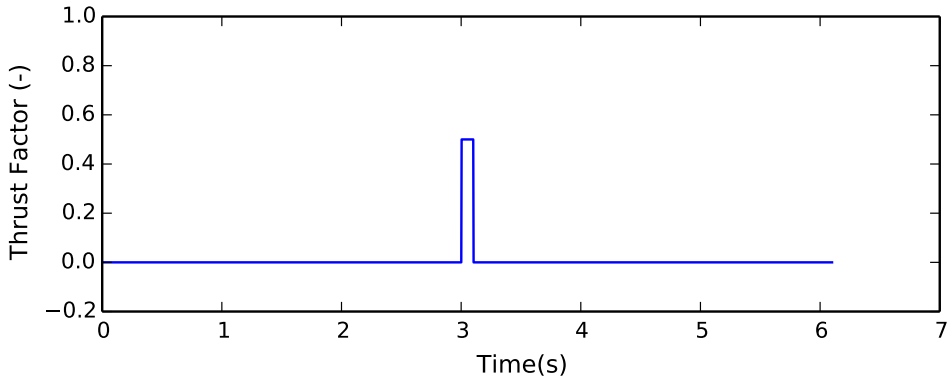

Fig. 7: Force on Y with 1 thrusters, for 0 sec at 30 deg Rate1000

As Figure 8 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

4. Instantaneous On/Off Factor with faster test rate:

   The thruster is set at 30° off the x-axis, in the position $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The test is launched using 1 thruster, for 5.0 seconds. The test rate is 100 steps per second

   The thrust command given is now 5 seconds long, as in the base test. The difference is that the test rate is now augmented in order to guarantee that it does not affect the test.

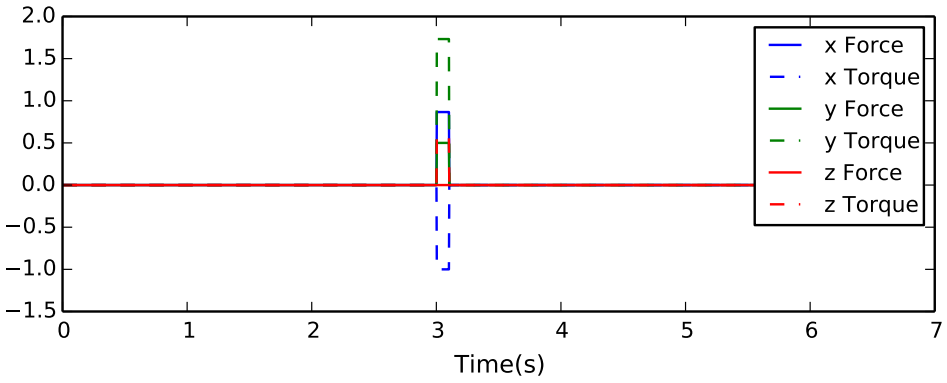

Fig. 8: All Forces and Torques 1 thrusters, for 0 sec at 30 deg Rate1000

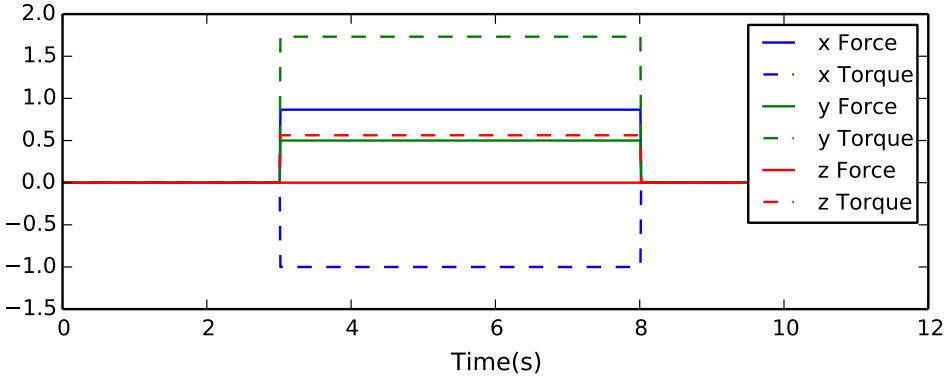

Fig. 9: All Forces and Torques 1 thrusters, for 5 sec at 30 deg Rate100

As Figure 9 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

5. Thruster Angle Test:

The thruster is set at 10° off the x-axis, in the position $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The test is launched using 1 thruster, for 5.0 seconds. The test rate is 10 steps per second

The test now shows that the simulation still behaves with different thruster orientations.

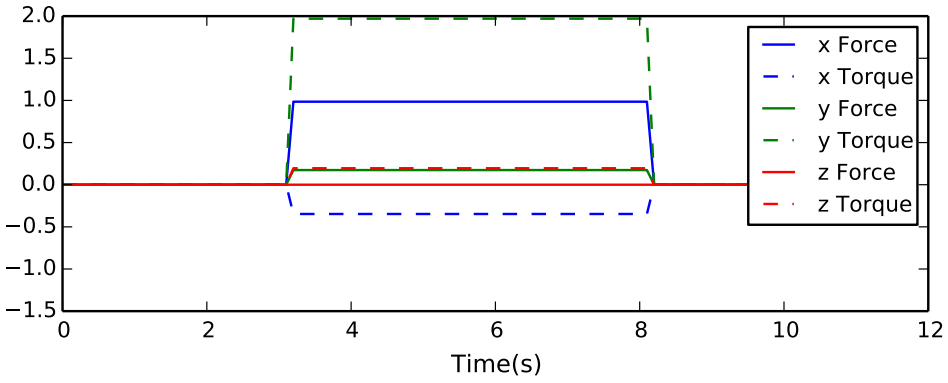

Fig. 10: All Forces and Torques 1 thrusters, for 5 sec at 10 deg Rate10

As Figure 10 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

6. Thruster Position Test:

The thruster is set at 30° off the x-axis, in the position $\boldsymbol{r} = (1.0, 0.0, 0.0)$. The test is launched using 1 thruster, for 5.0 seconds. The test rate is 10 steps per second

This test shows that different locations still give correct values for forces and torques.

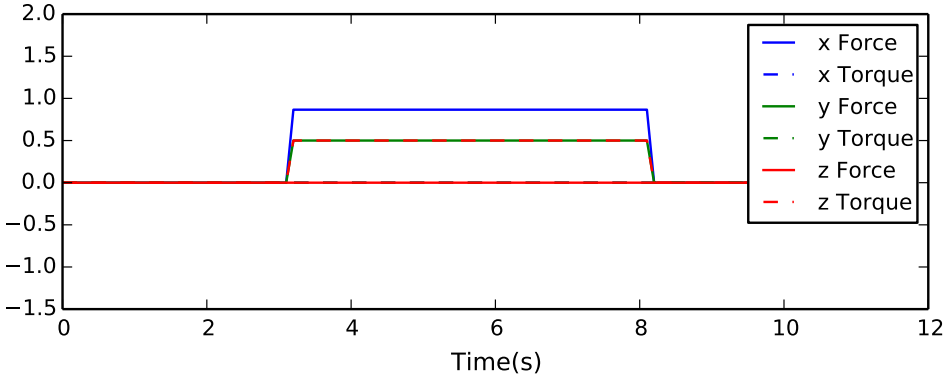

Fig. 11: All Forces and Torques 1 thrusters, for 5 sec at 30 deg Rate10

As Figure 11 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays.

Test successful.

7. Thruster Number Test:

   The thruster is set at 30° off the x-axis, in the position $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The test is launched using 2 thruster, for 5.0 seconds. The test rate is 10 steps per second

   This test shows that the thruster model can incorporate several thrusters correctly. We add a second thruster and use the modified truth function for the forces and torques.

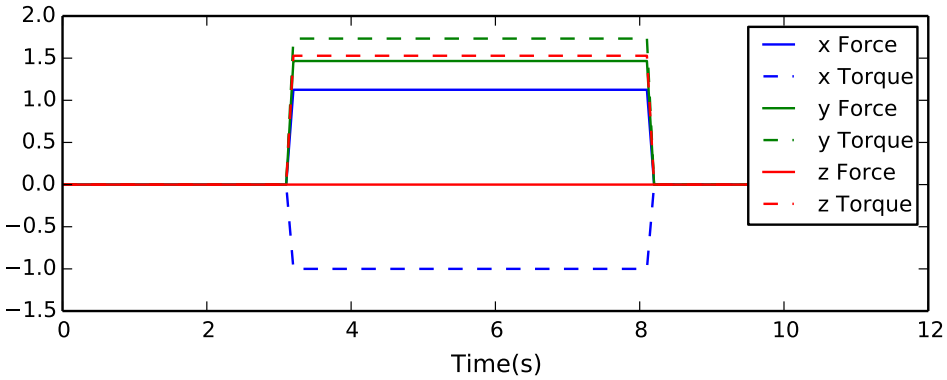

Fig. 12: All Forces and Torques 2 thrusters, for 5 sec at 30 deg Rate10

   As Figure 12 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

8. Ramp On/Ramp Off Firing:

   We test the ramped thrust with 10 incremental steps. The single thruster is set at the default 30°, at $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The thrust is set for 5.0 seconds with a test rate of 10 steps per second. The Cutoff test is OFF

   This test now ramps the thrust up and down. We use a 10 step ramp that takes 0.1s to climb and fall. This ramp time is slightly exaggerated in order to see the ramp clearly.

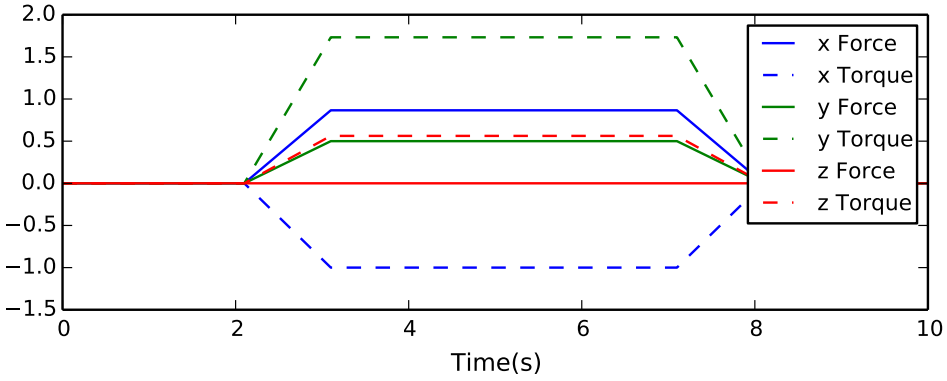

Fig. 13: All Forces and Torques with 10 step Ramp, thrust for 5s. Cutoff OFF, testRate10

As Figure 13 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

9. Short ramped firing:

We test the ramped thrust with 10 incremental steps. The single thruster is set at the default 30°, at $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The thrust is set for 0.5 seconds with a test rate of 10 steps per second. The Cutoff test is OFF

Using the same ramp, the thruster fires for 0.5s. We expect to see a climb and immediate fall of the thrust factor.

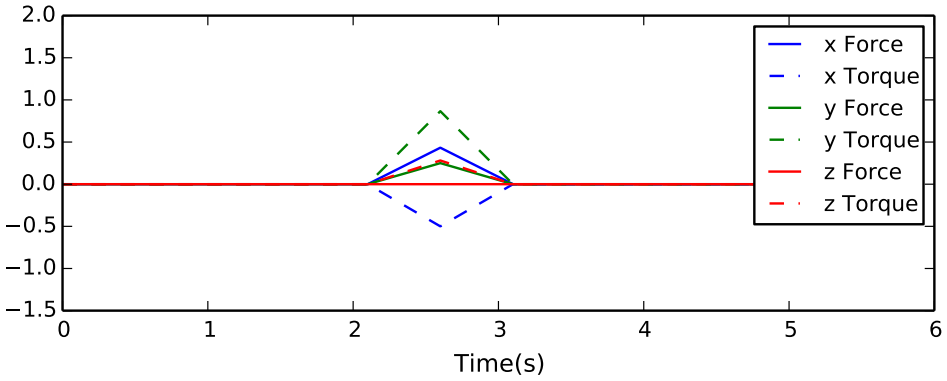

Fig. 14: All Forces and Torques with 10 step Ramp, thrust for 0s. Cutoff OFF, testRate10

As Figure 14 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

10. Ramp On/Ramp Off Firing with faster test rate:

We test the ramped thrust with 10 incremental steps. The single thruster is set at the default 30°, at $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The thrust is set for 5.0 seconds with a test rate of 100 steps per second. The Cutoff test is OFF

Using once again the same ramp, we run the test for 5 seconds with a faster test rate. We seek to validate that the test rate has no impact on the simulation.

As Figure 15 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

11. Cutoff firing:

We test the ramped thrust with 10 incremental steps. The single thruster is set at the default 30°, at $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The thrust is set for 5.0 seconds with a test rate of 10 steps per second. The Cutoff test is ON

Using the same ramp, we start firing the thruster with an initial command of 5 seconds. After just 0.2 seconds of thrust ramping, we change the test command and thrust for 0.3 seconds. This leads to a total thrust of 0.5 seconds, and validates the fact that the command was properly overridden.

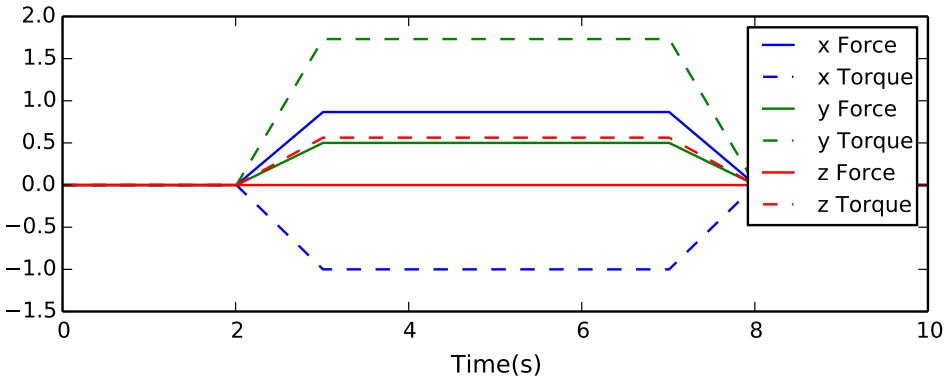

Fig. 15: All Forces and Torques with 10 step Ramp, thrust for 5s. Cutoff OFF, testRate100

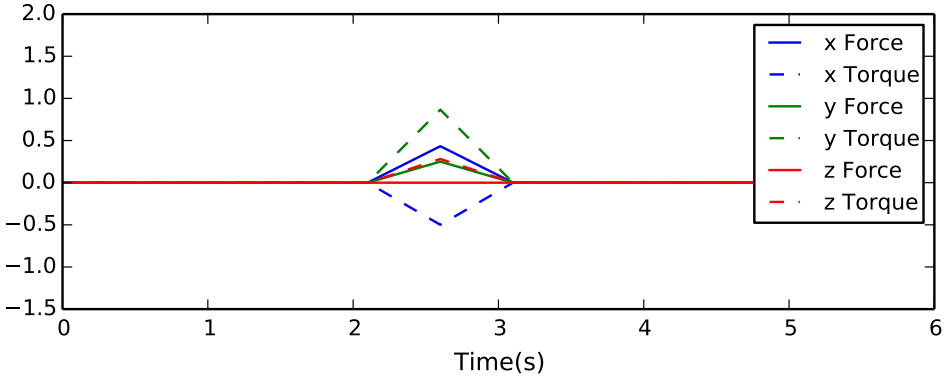

Fig. 16: All Forces and Torques, with a 10 step Ramp, thrust for 5s. Cutoff ON, testRate10

As Figure 16 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

12. Ramp down firing:

We test the ramped thrust with 10 incremental steps. The single thruster is set at the default 30°, at $\boldsymbol{r} = (1.125, 0.0, 2.0)$. The thrust is set for 0.5 seconds initially with a test rate of 10 steps per second. The Cutoff test is ON the RampDown test is ON.

In this test, the initial command is of 0.5 seconds. On the falling side of the ramp, a new command is given for 1.5s. We expect to see the thrust climb again to steady state and last for the expected command time.

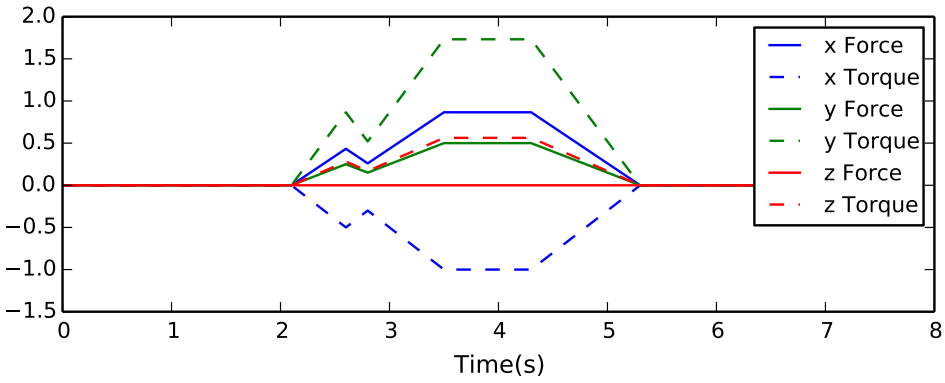

Fig. 17: All Forces and Torques, with a 10 step Ramp, Cutoff ON, RampDownON testRate10

As Figure 17 shows, the desired behavior is captured exactly for each firing in the test for all of the forces and torques. This is validated by the exact same predicted and simulated thrust arrays. Test successful.

7.2 Test Coverage

The method coverage for all of the methods included in the spice_interface module are tabulated in Table 2

Table 2: Test Analysis Results

Method Name	Unit Test Coverage (%)	Runtime Self (%)	Runtime Children (%)
SelfInit	100.0	0.0	0.0
CrossInit	100.0	0.0	0.0
AddThruster	100.0	0.0	0.0
UpdateState	100.0	0.0	0.0
WriteOutputMessages	100.0	0.0	0.0
ReadInputs	100.0	0.0	0.0
ConfigureThrustRequests	100.0	0.0	0.0
ComputeDynamics	100.0	0.0	9.8
ComputeThrusterFire	100.0	0.0	0.0
ComputeThrusterShut	100.0	0.0	0.0
updateMassProperties	100.0	0.0	0.6

For all of the methods in the spice_interface modules, the code coverage percentage is 100% which meets our test requirements. Additionally, 100% of all code branches in the thruster_dynamics source code were executed by this test.

The main CPU usage of the thruster_dynamics source code occurs in the ComputeDynamics method that is called by the dynamics source. The ThrusterDynamics methods themselves account for very little of the processing and it is the vector/matrix manipulation utilities called from the source that are the main culprits. While the thruster model's ComputeDynamics function is using 50% of the dynamics processing, that is only amounting to 10% of the overall simulation processing. The rest of the architecture in Basilisk should allow us to take the processing hit that we are getting from the ThrusterDynamics module without issue.

7.3 Conclusions

The thruster module has sufficient fidelity to accomplish the analysis that we need to perform thrust maneuvers. All model capabilities were tested and analyzed in this document with all observed performance being nominal compared to the going-in expectation. Every line of source code was successfully tested and the integrated model performance was analyzed and is acceptable. Furthermore many thrust scenarios were tested in order to cover all outcomes of a maneuver and the robustness of the simulation.

8 User Guide

The model can be configured according to the user's wishes, but the following rules of thumb should probably be respected unless the user is confident:

1. The internal simulation dynamics step time should be less than or equal to the thruster ramp-up/ramp-down time steps
2. The internal simulation dynamics step time should be less than or equal to the desired thruster discretization level
3. The internal simulation dynamics step time should be less than one-tenth of the expected minimum allowable thruster firing duration

A common set up for thrusters, contains:

- `thrusterSet = thrusterDynamicEffector.ThrusterDynamicEffector()`: Construct the Thruster Dyn Effector
- `thrusterSet.ModelTag = "ACSThrusterDynamics"`: Set the model tag
- `thruster1 = thrusterDynamicEffector.THRConfigSimMsg()`: Create a individual thruster
- `thruster1.thrLoc_B = [[1.0] ,[ 0.], [0.]]` : Set the thruster's location
- `thruster1.thrDir_B = [[math.cos(anglerad)], [math.sin(anglerad)], [0.0]]`: Set the thruster thrust direction
- `thruster1.MaxThrust = 1.0`: Set the max thrust
- `thruster1.steadyIsp = 226.7`: Set the I_{sp}
- `thruster1.MinOnTime = 0.006`: Set the minimum on time
- `thrusterSet.addThruster(thruster1)`: Add thruster to the Dyn Effector

If setting up a ramp, the user must also perform this:

- `rampOnList = []`

- `rampOffList = []`
- `for i in range(rampsteps):`

  ```
  fnewElement = thrusterDynamicEffector.THRTimePairSimMsg()
  fnewElement.TimeDelta = (i + 1.)  * 0.1
  fnewElement.ThrustFactor = (i + 1.0) / 10.0
  fnewElement.IspFactor = (i + 1.0) / 10.0
  frampOnList.append(newElement)
  fnewElement = thrusterDynamicEffector.THRTimePairSimMsg()
  fnewElement.TimeDelta = (i + 1) * 0.1
  fnewElement.ThrustFactor = 1.0 - (i + 1.0) / 10.0
  fnewElement.IspFactor = newElement.ThrustFactor
  frampOffList.append(newElement)
  ```

- `thrusterSet.thrusterData[0].ThrusterOnRamp = thrusterDynamicEffector.ThrusterTimeVector(ra`
  Add the on ramp
- `thrusterSet.thrusterData[0].ThrusterOffRamp = thrusterDynamicEffector.ThrusterTimeVector(r`
  Add the off ramp