Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-FacetDrag

FACETDRAG

Prepared by	A. Harris

Status: Released
Scope/Contents
The `facetedDrag` class used to calculate drag forces acting on a spacecraft modeled as a collection of flat, angled facets. Spacecraft geometry is settable by the user. In a given simulation, each spacecraft should have only one drag effector associated with it.

Rev	Change Description	By	Date
1.0	Initial release	A. Harris	05-15-2019

Contents

---

1 Model Description

1.1 General Module Function

The purpose of this module is to implement simple, attitude-dependent drag forces and torques for spacecraft with convex (i.e., non-self-shadowing) shape models. Spacecraft are considered as collections of facets consisting of an area, an offset from Point B, a surface normal (defined in the body frame), and a drag coefficient. Facets can be added from the Python level by using the `addFacetToModel` method. This module does not support self-shadowing.

1.2 Facet Model

The simplest model of spacecraft drag that considers the dependence of attitude is a faceted model. These models compute drag by considering the spacecraft as a collection of facets. The drag force contributed by an individual facet is given by

$$\mathbf{F}_D = -\frac{1}{2} C_D \rho A_{\text{facet}} (\hat{n} \cdot \hat{v}) |\mathbf{v}|^2 \hat{v} \tag{1}$$

where C_D is the facet's drag coefficient, ρ is the local neutral density, A_{facet} is total facet area, $\hat{n}$ is the facet surface normal unit vector, and $\mathbf{v}$ is the atmosphere-relative velocity.

2 Module Functions

This module will:

- **Compute atmospheric drag**: Each of the provided models is fundamentally intended to compute the neutral atmospheric density and temperature for a spacecraft relative to a body. These parameters are stored in the AtmoPropsSimMsg struct. Supporting parameters needed by each model, such as planet-relative position, are also computed.
- **Support simple spacecraft geometry dependence**: This module interfaces with modules that subscribe to neutral density messages via the messaging system.
- **Subscribe to model-relevant information:** Each provided atmospheric model requires different input information to operate, such as current space weather conditions and spacecraft positions. This module automatically attempts to subscribe to the relevant messages for a specified model.

3 Module Assumptions and Limitations

This module is only intended for simple convex geometries that do not self-shadow; facets that are turned "away" from the flow are considered to be non-interacting, while facets that are turned "into" the flow are, regardless of other panel geometry. Additionally, specular reflection is not considered, so lift effects are not calculated.

Drag modeling is complex and subject to a variety of assumptions and limitations. For further details, an interested reader is pointed to 1.

4 Test Description and Success Criteria

This section describes the specific unit tests conducted on this module.

4.1 General Functionality

4.1.1 setDensityMessage

This test verifies that the user can specify the atmospheric density message used by the module.

4.1.2 testDragForce

This test verifies that the module correctly calculates the drag force given the model's assumptions. It also implicitly tests the compatibility of facetDrag and exponentialAtmosphere.

4.1.3 testShadow

This test verifies that panels that are not in the flow are correctly ignored for the purposes of drag calculation.

4.2 Model-Specific Tests

4.2.1 test_unitFacetDrag.py

This unit test runs setDensityMessage, testDragForce, and testShadow to verify the functionality of the module.

5 Test Parameters

The simulation tolerances are shown in Table 2. In each simulation the neutral density output message is checked relative to python computed true values.

Table 2: Error tolerance for each test.

Output Value Tested	Tolerated Error
facetDragDynamicEffector.atmoDensInMsgName	0 (str)
newDrag.forceExternal_N	0.001 (relative)

6 Test Results

The following table shows the test result.

Table 3: Test result for test_unitFacetDrag.py

Check	Pass/Fail
1	PASSED

7 User Guide

7.1 General Module Setup

This section outlines the steps needed to add a facetDrag effector to a spacecraft, add facets to that module, and connect said module to an atmosphere module.

First, import the module and set its tag:

```
from Basilisk.simulation import facetDragDynamicEffector
newDrag = facetDragDynamicEffector.FacetDragDynamicEffector()
newDrag.ModelTag = "FacetDrag"
```

Assuming an atmospheric density model has already been set up, the density message to be used by this drag effector can be set by calling

```
newDrag.setDensityMessage(atmoModule.envOutMsgs[0])
```

By default, the module has no facets and drag calculation is skipped. To add a facet, call th addFacet function with an area, drag coefficient, body-frame normal vector, and body-frame facet location. For example, to add multiple facets, call

```
scAreas = [1.0, 1.0]
scCoeff = np.array([2.0, 2.0])
B_normals = [np.array([1, 0, 0]), np.array([0, 1, 0])]
B_locations = [np.array([0.1,0,0]), np.array([0,0.1,0])]

for ind in range(0,len(scAreas)):
newDrag.addFacet(scAreas[ind], scCoeff[ind], B_normals[ind], B_locations[ind])
```

Finally, add the module to a spacecraft object using the addDynamicEffector method:

```
scObject = spacecraft.Spacecraft()
scObject.ModelTag = "spacecraftBody"
scObject.addDynamicEffector(newDrag)
```

REFERENCES

[1] David Vallado. *Fundamentals of Astrodynamics and Applications*. Microcosm press, 4 edition, 2013.