Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-Atmosphere

ATMOSPHERE

Prepared by	A. Harris

Status: Released	
Scope/Contents	
The `Atmosphere` class used to calculate temperature / density above a body using multiple models. This class is used to hold relevant atmospheric properties and to compute the density for a given set of spacecraft relative to a specified planet. Planetary parameters, including position and input message, are settable by the user. Internal support is provided for Venus, Earth, and Mars. In a given simulation, each planet of interest should have only one Atmosphere model associated with it linked to the spacecraft in orbit about that body.	

Rev	Change Description	By	Date
1.0	Initial release	A. Harris	02-26-2019
1.1	Update with new unit tests	H. Schaub	03-09-2019

Contents

1 Model Description

The purpose of this module is to implement neutral atmospheric density and temperature models. Atmosphere represents a single interface to a variety of atmospheric models within the BSK framework. These are briefly summarized here for reference. By invoking the atmosphere module, the default values are set such that this basic exponential atmosphere model is simulated about Earth. The reach of the model controlled by setting the variables `envMinReach` and `envMaxReach` to positive values. These values are the altitude above the planets surfaces assuming an equatorial radius and spherical shape. The default values are -1 which turns of this checking where the atmosphere model as unbounded reach.

1.1 Exponential Atmosphere

Under the assumption of an isothermal atmosphere, an expression for atmospheric density can be readily arrived at by combining the equations of hydrostatic equilibrium with the ideal gas law and integrating, yielding:

$$\rho = \rho_0 e^{\frac{-h}{h_0}} \tag{1}$$

where ρ_0 is the density at the planet's surface, h is the altitude, and h_0 is the atmospheric scale height derived from the weighted-average behavior of the species that make up the atmosphere. A list of these parameters for atmospheric bodies in the solar system is available from NASA here. For more detail, see Reference 1.

1.2 NRLMSISE-00

NRLMSISE-00 is an empircally derived neutral density and temperature model maintained by the Naval Research Laboratory. Basilisk uses the C implementation of NRLMSISE-00 published by Dr. Dominik Brodowski.

At present, this functionality is not implemented.

2 Module Functions

This module will:

- **Compute atmospheric density and temperature**: 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.
- **Communicate neutral density and temperature**: 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.
- **Support for multiple spacecraft and model types** Only one Atmosphere module is required for each planet, and can support an arbitrary number of spacecraft. Output messages for individual spacecraft are automatically named based on the environment type.

3 Module Assumptions and Limitations

Individual atmospheric models are complex and have their own assumptions. At present, all non-exponential models are Earth-specific. For details about tradeoffs in atmospheric modeling, the 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 setEnvType

This test verifies that the module is able adjust its internal state to each of the implemented atmospheric models.

4.1.2 setEpoch

This test verifies that the user can set the initial date ("epoch") arbitrarily.

4.1.3 addSpacecraftToModel

This test verifies that the user can both add additional spacecraft to the module. This is accomplished by checking the number of input and output messages of the module after adding multiple spacecraft.

4.2 Model-Specific Tests

4.2.1 test_integratedTestAtmosphere.py

This integrated runs a section of an orbit and verifies that the exponential atmosphere model both correctly calculates the orbit altitude and the atmospheric density against a Python implementation of the model. A single spacecraft is simulated about Earth, and no minimum or maximum reach is set. No planet ephemeris message is setup causing the simulation to assume the planet center is at the coordinate frame origin.

4.2.2 test_unitTestAtmosphere.py

This unit test only runs the atmosphere Basilisk module with two fixed spacecraft state input messages. The simulation option `useDefault` checks if the default atmosphere parameters for an Earth-based exponential atmosphere module are used, or if the exponential model information is setup manually. The option `useMinReach` dictates if the minimum altitude check is performed, while the option `useMaxReach` checks if the maximum reach check is performed. The option `usePlanetEphemeris` checks if a planet state input message should be created. All permutations are checked.

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
`length(atmo.scStateInMsgNames)`	`0 (int)`
`length(atmo.envOutMsgNames)`	`0 (int)`
`atmo.envType`	`0 (string)`
`neutralDensity`	1e-05 (relative)

6 Test Results

The following two tables show the test results. All tests are expected to pass.

Table 3: Test result for test_integratedTestAtmosphere.py

Check	Pass/Fail
1	PASSED

Table 4: Test result for test_unitTestAtmosphere.py

useDefault	useMinReach	useMaxReach	usePlanetEphemeris	**Pass/Fail**
False	False	False	False	PASSED
False	False	False	True	PASSED
False	False	True	False	PASSED
False	False	True	True	PASSED
False	True	False	False	PASSED
False	True	False	True	PASSED
False	True	True	False	PASSED
False	True	True	True	PASSED
True	False	False	False	PASSED
True	False	False	True	PASSED
True	False	True	False	PASSED
True	False	True	True	PASSED
True	True	False	False	PASSED
True	True	False	True	PASSED
True	True	True	False	PASSED
True	True	True	True	PASSED

7 User Guide

7.1 General Module Setup

This section outlines the steps needed to add an Atmosphere module to a sim. First, the atmosphere must be imported and initialized:

```
from Basilisk.simulation import atmosphere
newAtmo = atmosphere.Atmosphere()
newAtmo.ModelTag = "ExpAtmo"
```

By default the model assumes an exponential atmosphere model about the Earth.

Optionally, if a different model type is required, the desired model type must be set by calling "`setEnvType`":

```
newAtmo.setEnvType("exponential")
```

The model can then be added to a task like other simModels. Each Atmosphere calculates atmospheric parameters based on the output state messages for a set of spacecraft.

To add spacecraft to the model the spacecraft state output message name is sent to the `addScToModel` method:

```
scObject = spacecraftPlus.SpacecraftPlus()
scObject.ModelTag = "spacecraftBody"
newAtmo.addSpacecraftToModel(scObject.scStateOutMsgName)
```

7.2 Planet Ephemeris Information

The optional planet state message name can be set by directly adjusting that attribute of the class:

```
newAtmo.planetPosInMsgName = "PlanetSPICEMsgName"
```

If SPICE is not being used, the planet is assumed to reside at the origin.

7.3 Setting the Model Reach

By default the model doesn't perform any checks on the altitude to see if the specified atmosphere model should be used. This is set through the parameters `envMinReach` and `envMaxReach`. Their default values are -1. If these are set to positive values, then if the altitude is smaller than `envMinReach` or larger than `envMaxReach`, the density is set to zero.

7.4 Exponential atmosphere user guide

If the module's `envType` is set to `"exponential"`, the parameters of the exponential atmosphere can be set by calling

```
newAtmo.exponentialParams.baseDensity = baseDensityValue
newAtmo.exponentialParams.scaleHeight = scaleHeightValue
newAtmo.planetRadius = planetRadiusValue
```

REFERENCES

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