Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-MsisAtmosphere

MSISATMOSPHERE

Prepared by	A. Harris

Status: First Review
Scope/Contents
The `MsisAtmosphere` class used to calculate temperature, density, and neutral species for a spacecraft in Earth orbit up to 1000km using the NRLMSISE-00 neutral density model.

Rev	Change Description	By	Date
1.0	Initial release	A. Harris	06-20-2019

Contents

1 Model Description 1

2 Module Functions 1

3 Module Assumptions and Limitations 1

4 Test Description and Success Criteria 1
4.1 General Functionality . . . 2
4.2 Model-Specific Tests . . . 2
4.2.1 test_unitTestNrlmsise00.py . . . 2

5 Test Parameters 2

6 Test Results 2

7 User Guide 2
7.1 General Module Setup . . . 2
7.2 Planet Ephemeris Information . . . 3
7.3 Setting the Model Reach . . . 3
7.4 NRLMSISE-00 atmosphere user guide . . . 3

1 Model Description

The purpose of this module is to wrap the Brodowski implementation of the NRLMSISE-00 atmospheric neutral density model* for use with other Basilisk modules. NRLMSISE-00 is a high-fidelity, empirically-derived model of the Earth's atmosphere that considers the effects of common space phenomena (magnetic/solar influences) on neutral atmospheric density. A portfolio of individual models is valid up to 1000km, and provides both the neutral temperature and species in addition to the total mass density.

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 AtmoPropsMsgPayload 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.

* https://ccmc.gsfc.nasa.gov/models/modelinfo.php?model=MSISE

- **Support for multiple spacecraft** Only one NRLMSISE-00 atmosphere model is needed to compute densities for several spacecraft.
- **Support dynamic space-weather coupled density forecasting**: A primary benefit of the NRLMSISE-00 model is its ability to provide forecasts of neutral density in response to space weather events, changing atmospheric conditions, and the like that are not captured in simpler models.

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.

NRLMSISE-00, specifically, is highly dependent on "space weather parameters" such as Ap, Kp, $F10.7$, among others. The outputs of this model can vary greatly with the selected parameters.

4 Test Description and Success Criteria

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

4.1 General Functionality

The unit test check the neutral density calculation for a single spacecraft at a particular epoch time. The epoch is either specified through an epoch input message, or set directly by specifying the `epochDoy` day of year value.

4.2 Model-Specific Tests

By design, this module shares much of its functionality with `exponentialAtmosphere`; shared functionality is not checked by redundant tests.

4.2.1 test_unitTestNrlmsise00.py

This integrated test evaluates the NRLMSISE-00 model at a given point in an orbit with zero'd (i.e., nonphysical) space weather inputs and verifies its outputs against the outputs of the base C model with identical inputs. This is a comprehensive test of the NRLMSISE-00 implementation in BSK, as it checks out the end-to-end functionality of the space weather messaging system, the geodetic conversion, and the interface to NRLMSISE-00 itself.

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
Output Neutral Mass Density	1e-08 (relative)
Output Temperature	1e-08 (relative)

6 Test Results

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

Table 3: Test result for test_unitMsisAtmosphere.py

orbitCase	setEpoch	**Pass/Fail**
LPO	Direct	PASSED
LTO	Direct	PASSED
LPO	Msg	PASSED
LTO	Msg	PASSED
LPO	Default	PASSED
LTO	Default	PASSED

7 User Guide

7.1 General Module Setup

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

```
from Basilisk.simulation import msisAtmosphere
newAtmo = msisAtmosphere.MsisAtmosphere()
newAtmo.ModelTag = "MsisAtmo"
```

By design this module is only applicable to Earth. The earth radius is set automatically within the module. As with all planetary environment models, the epoch case be set in three ways:

1. **Default:** If no epoch information is provided, then the BSK default epoch date and time are set.
2. **Use Epoch Message:** An epoch input message is connected. This is read in on reset.

   ```
   epochMsg = unitTestSupport.timeStringToGregorianUTCMsg('2019 Jan 01 00:00:00.00 (UTC)')
   newAtmo.epochInMsg.subscribeTo(epochMsg)
   ```

3. **Direct Setting:** The module variable `epochDoy` can be set directly as well. However, note that if the input message is specified, the `epochDoy` value is not used.

   ```
   newAtmo.epochDoy = 1
   ```

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 `addSpacecraftToModel` method common to environmental models:

```
scObject = spacecraft.Spacecraft()
scObject.ModelTag = "spacecraftBody"
newAtmo.addSpacecraftToModel(scObject.scStateOutMsg)
```

This `addSpacecraftToModel()` method subscribes the module to the spacecraft state output message, and it creates the corresponding vector entry for the environment output message. The output message order is the same as the order in which the spacecraft state messages are added.

7.2 Planet Ephemeris Information

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

```
newAtmo.planetPosInMsg.subscribeTo(planetMsg)
```

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 NRLMSISE-00 atmosphere user guide

NRLMSISE-00 is dependent on a variety of space weather indexes, times, and locations. During initialization, a starting date must be set; this will be updated as the sim progresses using the simulation time. NRLMSISE-00 will attempt to subscribe to a standard set of message names that can be produced by the WIP space-weather data factory module, or set by hand. These messages are

```
sw_msg_names = [
"ap_24_0", "ap_3_0", "ap_3_-3","ap_3_-6","ap_3_-9",
"ap_3_-12","ap_3_-15","ap_3_-18","ap_3_-21","ap_3_-24",
"ap_3_-27", "ap_3_-30","ap_3_-33","ap_3_-36","ap_3_-39",
"ap_3_-42", "ap_3_-45", "ap_3_-48","ap_3_-51","ap_3_-54",
"ap_3_-57","f107_1944_0","f107_24_-24"
]
```

REFERENCES

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