Astrodynamics Standards
Shared Library

Astrodynamic Utility Functions (AstroFunc)

Version 9.4

May 2024

Contents

1. Introduction

The **AstroFunc** shared library provides functions for converting between orbital elements and for transforming between coordinate systems. It also provides a variety of other useful astrodynamic utility functions.

If you are on Windows, the shared library files will end in ".dll". For example, "AstroFunc.dll". If you are on Linux, the shared library will begin with "lib" and end in ".so", and will be all lowercase. For example, libastrofunc.so.

2. Prerequisites

The following shared libraries MUST be loaded and initialized before using AstroFunc:

- DllMain
- EnvConst
- TimeFunc

3. Getting Started

To get started, please read the README.txt file that came in the root directory of your distribution. In addition to an overall description contained in the distribution, it has a description of a "**wrapper**".

To get started with **AstroFunc**, there is a "wrapper" specific to AstroFunc, under the **SampleCode** directory. Under your language of choice, you will see a "**DriverExample/wrapper**" subdirectory. The files under this directory will have all the Application Programming Interfaces (APIs) available. For AstroFunc specific APIs, you should see a source file labelled with "AstroFunc" in the file name. This will be where you will find all the APIs for that specific library. The "DriverExample" directory will also contain several examples of applications that should run by simply running the runExample.bat or runExample.sh script. You can use these examples as a starting point for building your application.

If you do not see your programming language under "SampleCode", look in the HTML documentation for the APIs. Open a browser to the "Documentation/APIDocs/index.html" file. This document will show all the APIs regardless of programming language.

The Astrodynamics Standards libraries should work with any language capable of using Dynamic Link Library (on Windows) or Shared object (on Linux) files.

4. Arrangement of the Orbital Elements in Arrays

Array Description	Array Size	Array Arrangement
Keplerian elements	6	1. semi-major axis (km) 2. eccentricity (unitless) 3. inclination (degree) 4. mean anomaly (degree)

		5. right ascension of the ascending node (degree) 6. argument of perigee (degree)
Equinoctial elements	6	1. Af (unitless) 2. Ag (unitless) 3. Chi (unitless) 4. Psi (unitless) 5. L : mean longitude (degree) 6. N : mean motion (revs/day)
Classical elements	6	1. mean motion (revs/day) 2. eccentricity (unitless) 3. inclination (degree) 4. mean anomaly (degree) 5. right ascension of the ascending node (degree) 6. argument of perigee
LLH (geodetic latitude, longitude, height)	3	1. north latitude (degree) 2. east longitude (degree) 3. height (km) from geoid

Notes: The **Array Arrangement** column lists the order of the elements in the array. It is not necessarily the subscript of the element in the array since this is language-dependent. For example, in C/C++ the first element in every array is the zero-subscripted element. In other languages, 1-based indexing might be used instead.