Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

ORBELEMCONVERT

Rev:	Change Description	By	Date
1.0	First version - Mathematical formulation and implementation	G. Chapel	07/27/17

Contents

---

1 Model Description

The orb_elem_convert module is responsible for converting between a set of six orbital elements and three-component position and velocity vectors. The module determines which conversion to perform by which one of the 3 input messages is connected. To complete these conversions, the module uses the *elem2rv* and *rv2elem* functions in `/SimUtilities/orbitalMotion.c`. Both conversions require a gravitational constant as an input in addition to the appropriate state parameters. The math involved for conversion as well as implementation is shown and explained below. These calculations can also be found in Vallado's *Fundamentals of Astrodynamics and Applications*[1] and Schaub's *Analytical Mechanics of Space Systems*.[2] Methods and results of the code unit tests are also presented and discussed.

1.1 Mathematical model

1.1.1 Elements to Cartesian

Keplerian orbital elements are parameters used to define the state of a body in a specific orbit. Orbits may also be identified with Cartesian vectors, though, and it is sometimes more appropriate to do so. Converting elements to Cartesian requires *Elements2Cart* to be *True* and the input of six orbital elements and a gravitational constant. The necessary orbital elements for converting to position and velocity vectors include the semimajor axis a, eccentricity e, inclination i, ascending node Ω, argument of periapsis ω, and true anomaly f. The gravitational constant μ is used in place of the body's mass and gravitational force. An illustration of the orbital elements is displayed in Fig. 1 *.

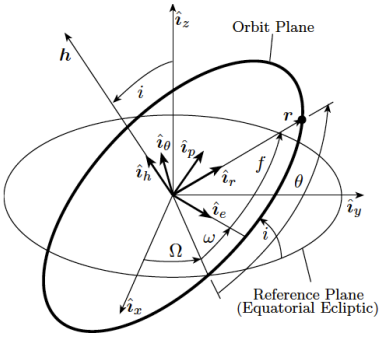


Fig. 1: Illustration of Keplerian Orbital Elements and Plane

The mathematical methods for conversion vary depending on parameter values, specifically semimajor axis and eccentricity inputs. Outlined below are three cases for varying a and e. Although the first case is supported when converting from orbital elements to Cartesian, it is not accurate for the reverse process and is considered a limitation of this module. The mathematical model is still briefly explained below but the first case is not tested. These cases all result in three-component position and velocity vectors.

1.1.1.1 Case 1 ($e = 1.0$, $a > 0$): This case specifies a rectilinear, elliptical orbit. The first steps in obtaining position and velocity vectors is to find the orbit radius and the magnitude of rectilinear velocity. The orbit radius r depends on the semimajor axis, eccentricity, and true anomaly. The velocity, as expected, depends on the orbit radius, semimajor axis, and gravitational constant. Both of these values can be found using the following equations:

$$r = a(1 - e\cos(f)) \tag{1}$$

$$v = \sqrt{\frac{2\mu}{a(r - \mu)}} \tag{2}$$

Fig. 1 illustrates the orbit frame *O* with components $\{\hat{\boldsymbol{i}}_e, \hat{\boldsymbol{i}}_p, \hat{\boldsymbol{i}}_h\}$, the inertial frame *N*= $\{\hat{\boldsymbol{i}}_x, \hat{\boldsymbol{i}}_y, \hat{\boldsymbol{i}}_z\}$, and the frame *M* which tracks the position of the body with components $\{\hat{\boldsymbol{i}}_r, \hat{\boldsymbol{i}}_\theta, \hat{\boldsymbol{i}}_h\}$. For rectilinear orbits, the Euler angle set $\{\Omega, i, \omega\}$ relates the orientation of frame *M* to *N*, providing the ability to parameterize the direction cosine matrix and solve for position and velocity. This is shown by Eq. 3 below.

$$\boldsymbol{r} = r \begin{pmatrix} \cos(\Omega)\cos(\omega) - \sin(\Omega)\sin(\omega)\cos(i) \\ \sin(\Omega)\cos(\omega) + \cos(\Omega)\sin(\omega)\cos(i) \\ \sin(\omega)\sin(i) \end{pmatrix} \tag{3}$$

* Figure from *Analytical Mechanics of Space Systems*

The same parameterization is used when determining both position and velocity for a rectilinear orbit, giving

$$\dot{\boldsymbol{r}} = v \begin{pmatrix} \cos(\Omega)\cos(\omega) - \sin(\Omega)\sin(\omega)\cos(i) \\ \sin(\Omega)\cos(\omega) + \cos(\Omega)\sin(\omega)\cos(i) \\ \sin(\omega)\sin(i) \end{pmatrix} \tag{4}$$

1.1.1.2 Case 2 ($e = 1.0$, $a < 0$): This case indicates a parabolic orbit, which naturally has an infinite semimajor axis. For computational purposes and to distinguish this case from a rectilinear orbit, the negative radius at periapsis will be used instead of an infinite semimajor axis. The radius is used to determine a semiparameter p, which describes the size of the orbit's conic section. These relationships are as follows:

$$r_p = -a \tag{5}$$

$$p = 2r_p \tag{6}$$

As with Case 1, the orbit radius, velocity magnitude, and parameterized direction cosine matrix are used to compute the Cartesian vectors. The radius and velocity are determined largely using Kepler's first law, which gives the general trajectory equation shown here:

$$r = \frac{p}{1 + e\cos(f)} \tag{7}$$

For this case, the relationship between frames *M* and *N* uses the Euler angle set $\{\Omega,\ i,\ \theta\}$ where $\theta = \omega + f$ is the true latitude. To find the position vector, θ is substituted for ω in the vector from Eq. 3. This can be written as

$$\boldsymbol{r} = r \begin{pmatrix} \cos(\Omega)\cos(\theta) - \sin(\Omega)\sin(\theta)\cos(i) \\ \sin(\Omega)\cos(\theta) + \cos(\Omega)\sin(\theta)\cos(i) \\ \sin(\theta)\sin(i) \end{pmatrix} \tag{8}$$

Taking the derivative of this equation gives the velocity vector, which implements angular momentum h as shown in the following equations:

$$\dot{\boldsymbol{r}} = -\frac{\mu}{h} \begin{pmatrix} \cos(\Omega)(\sin(\theta) + e\sin(\omega)) + \sin(\Omega)(\cos(\theta) + e\cos(\omega))\cos(i) \\ \sin(\Omega)(\sin(\theta) + e\sin(\omega)) + \cos(\Omega)(\cos(\theta) + e\cos(\omega))\cos(i) \\ -(\cos(\theta) + e\cos(\omega))\sin(i) \end{pmatrix} \tag{9}$$

$$h = \sqrt{\mu p} \tag{10}$$

$$\dot{\boldsymbol{r}} = -\sqrt{\frac{\mu}{p}} \begin{pmatrix} \cos(\Omega)(\sin(\theta) + e\sin(\omega)) + \sin(\Omega)(\cos(\theta) + e\cos(\omega))\cos(i) \\ \sin(\Omega)(\sin(\theta) + e\sin(\omega)) + \cos(\Omega)(\cos(\theta) + e\cos(\omega))\cos(i) \\ -(\cos(\theta) + e\cos(\omega))\sin(i) \end{pmatrix} \tag{11}$$

1.1.1.3 Case 3 ($0 < e < 1.0$, $a > 0$) or ($e > 1.0$, $a < 0$): This case applies to any scenario not included in cases 1 or 2, that is where eccentricity is bounded between 0 and 1.0 for a positive semimajor axis and greater than 1.0 for a negative semimajor axis.

The only difference between calculating Cartesian vectors in this case and the second is the method of determining the semiparameter, as to avoid a negative value. This is shown by the following equation:

$$p = a(1 - e^2) \tag{12}$$

Eq. 7 - 11 remain the same for this case and can be used to find the Cartesian vectors. From Eq. 10, it is clear that the orbit's angular momentum is proportional to $\sqrt{p}$, resulting in an imaginary angular momentum when given a negative semiparameter. This is why limits are set for a and e in this case.

1.1.2 Cartesian to Elements

Since Keplerian orbital elements are the parameters primarily used to identify orbits, conversion from Cartesian vectors to these elements is an important function. That is, converting position and velocity vectors to the six elements previously discussed (a, e, i, Ω, ω, f). This conversion requires the input of a position and velocity vector, both consisting of three components, and a gravitational constant μ, like the previous conversion. Presented below is the mathematical process for converting from Cartesian to orbital elements.

Like the reverse process, this conversion has several cases, this time depending on the semimajor axis, eccentricity, *and* inclination. This means that the cases do not present themselves until those parameters are to be defined. That said, the first step is to calculate the position and velocity magnitudes, r_0 and v_0. Magnitudes of position and velocity can be determined from the vectors as follows:

$$r_0 = \sqrt{\boldsymbol{r} \cdot \boldsymbol{r}} \tag{13}$$

$$v_0 = \sqrt{\dot{\boldsymbol{r}} \cdot \dot{\boldsymbol{r}}} \tag{14}$$

Since eccentricity relies only on position, velocity, and the gravity constant μ, it can now be calculated. The equation for the eccentricity vector is

$$\boldsymbol{e} = \frac{(v_0^2 - \frac{\mu}{r_0})\boldsymbol{r} - (\boldsymbol{r} \cdot \dot{\boldsymbol{r}})\dot{\boldsymbol{r}}}{\mu} \tag{15}$$

Once the magnitude e is found, it can be used with r_0 and v_0 to find the semimajor axis. However, this is when unique cases begin to emerge. They are defined below.

Semimajor Axis Case 1 (non-parabolic finite a):
For non-parabolic orbits, the semimajor axis is given by the following energy equation:

$$a = (\frac{2}{r_0} - \frac{v_0^2}{\mu})^{-1} \tag{16}$$

Semimajor Axis Case 2 (parabolic infinite a):
For parabolic orbits, the semimajor axis is infinite, so a semiparameter is used instead for computational purposes.

$$a = -\frac{p}{2} \tag{17}$$

The semiparameter p is given by

$$p = \frac{h^2}{\mu} \tag{18}$$

To find the inclination of the orbit, it is neccessary to first calculate the angular momentum. In vector form, the angular momentum is calculated with a cross product as shown below.

$$\boldsymbol{h} = \boldsymbol{r} \times \dot{\boldsymbol{r}} \tag{19}$$

This vector is normalized and the magnitude is obtained with the following equations:

$$h = \sqrt{\boldsymbol{h} \cdot \boldsymbol{h}} \tag{20}$$

After finding angular momentum, it can be used to determine the inclination of the orbit plane. This inverse cosine expression involves only the second term of the angular momentum vector, denoted by $\hat{\boldsymbol{h}}_2$, and the magnitude. It can be written as

$$i = \cos^{-1}(\frac{\hat{\boldsymbol{h}}_2}{h}) \tag{21}$$

In addition to the inclination angle, it is important to find the line of nodes. The line of nodes is necessary, specifically, to identify inclined orbits and is not useful for equatorial orbits. First, set up a unit vector u to cross with the angular momentum vector, resulting in a line of nodes vector, and then calculate the magnitude.

$$\boldsymbol{u} =< 0, 0, 1.0 > \tag{22}$$

$$\boldsymbol{n} = \boldsymbol{u} \times \boldsymbol{h} \tag{23}$$

$$n = \sqrt{\boldsymbol{n} \cdot \boldsymbol{n}} \tag{24}$$

The vector $\boldsymbol{n}$ consists of three components, which will be referred to as $\hat{\boldsymbol{n}}_1$, $\hat{\boldsymbol{n}}_2$, and $\hat{\boldsymbol{n}}_3$. This provides the final parameter needed to find the remaining orbital elements and, ultimately, the orbit type. The methods for determining the ascending node, argument of periapsis, and true anomaly angles depend on the eccentricity and inclination, introducing the four cases presented below.

1.1.2.1 Case 1 (Non-circular, inclined orbit):

A non-circular orbit is specified with $e > 0$ and an inclined orbit has an inclination $i > 0$. Since it is for inclined orbits, this case depends on the line of nodes. Each remaining orbital element is calculated with cosine expressions, so depending on the sign of the line of nodes components, the angles may need to be adjusted by a 2π radians. Subcases are shown for these adjustments.

$$\Omega = \begin{cases} \cos^{-1}(\frac{\hat{\boldsymbol{n}}_1}{n}) & \hat{\boldsymbol{n}}_2 > 0 \\ 2\pi - \cos^{-1}(\frac{\hat{\boldsymbol{n}}_1}{n}) & \hat{\boldsymbol{n}}_2 < 0 \end{cases} \tag{25}$$

Calculating the ascending node requires the eccentricity vector found previously in Eq. 15.

$$\omega = \begin{cases} \cos^{-1}(\frac{\boldsymbol{n} \cdot \boldsymbol{e}}{ne}) & \hat{\boldsymbol{n}}_3 > 0 \\ 2\pi - \cos^{-1}(\frac{\boldsymbol{n} \cdot \boldsymbol{e}}{ne}) & \hat{\boldsymbol{n}}_3 < 0 \end{cases} \tag{26}$$

Finding the true anomaly also requires the eccentricity vector as well as the position vector, its magnitude, and the velocity vector.

$$f = \begin{cases} \cos^{-1}(\frac{\boldsymbol{e} \cdot \boldsymbol{r}}{er_0}) & \boldsymbol{r} \cdot \dot{\boldsymbol{r}} > 0 \\ 2\pi - \cos^{-1}(\frac{\boldsymbol{e} \cdot \boldsymbol{r}}{er_0}) & \boldsymbol{r} \cdot \dot{\boldsymbol{r}} < 0 \end{cases} \tag{27}$$

1.1.2.2 Case 2 (Non-circular, equatorial orbit):

An equatorial orbit is specified with $i > 0$. The true anomaly can still be calculated with Eq. 27 but the methods for finding the ascending node and argument of periapsis differ from those in the first case. An equatorial orbit has no ascending node, meaning

$$\Omega = 0 \tag{28}$$

It also uses the true longitude of periapsis, defining

$$\omega = \begin{cases} \cos^{-1}(\frac{\hat{\boldsymbol{e}}_1}{ne}) & \hat{\boldsymbol{e}}_2 > 0 \\ 2\pi - \cos^{-1}(\frac{\hat{\boldsymbol{e}}_1}{ne}) & \hat{\boldsymbol{e}}_2 < 0 \end{cases} \tag{29}$$

1.1.2.3 Case 3 (Circular, inclined orbit):

This case also differs from the first by two elements, the argument of periapsis and true anomaly. The ascending node can still be determined with Eq. 25. A circular orbit has no argument of periapsis, so for this case

$$\omega = 0 \tag{30}$$

The true anomaly substitutes eccentricity for the line of nodes vector and magnitude in Eq. 27. It also gets adjusted by 2π depending on the sign of the position vector's third component, denoted by $\hat{\boldsymbol{r}}_3$. This results in the following equation

$$f = \begin{cases} \cos^{-1}(\frac{\boldsymbol{n}\cdot\boldsymbol{r}}{nr_0}) & \hat{\boldsymbol{r}}_3 > 0 \\ 2\pi - \cos^{-1}(\frac{\boldsymbol{n}\cdot\boldsymbol{r}}{nr_0}) & \hat{\boldsymbol{r}}_3 < 0 \end{cases} \tag{31}$$

1.1.2.4 Case 4 (Circular, equatorial orbit): This case is a combination of the previous two, where the orbit has neither an ascending node or an argument of periapsis, giving

$$\Omega = 0 \tag{32}$$

$$\omega = 0 \tag{33}$$

Due to the eccentricity and the inclination both being equal to zero, the true anomaly also changes. It now only depends on the position vector's first and second terms, $\hat{\boldsymbol{r}}_1$ and $\hat{\boldsymbol{r}}_2$, and its magnitude. This can be written as

$$f = \begin{cases} \cos^{-1}(\frac{\boldsymbol{r}_1}{r_0}) & \hat{\boldsymbol{r}}_2 > 0 \\ 2\pi - \cos^{-1}(\frac{\boldsymbol{r}_1}{r_0}) & \hat{\boldsymbol{r}}_2 < 0 \end{cases} \tag{34}$$

Lastly, to ensure the appropriate value is used for the true anomaly, it should be checked whether the orbit is parabolic or hyperbolic. For both of these orbit types, a true anomaly greater than π and less then 2π would mean the body is outside of the orbit. Thus, a limit must be applied to the true anomaly for parabolic and hyperbolic orbits, as shown below. The sign of f remains the same after changing the magnitude.

$$f = \begin{cases} \pm 2\pi & |e| \geqslant 1.0 \quad \text{and} \quad |\pm f| > \pi \end{cases} \tag{35}$$

2 Model Functions

The main functions in this model include converting from Keplerian orbital elements to Cartesian vectors and converting from Cartesian vectors to Keplerian orbital elements. Orbital elements used for this conversion include the semimajor axis, eccentricity, inclination, ascending node, argument of periapses, and true anomaly. The Cartesian parameters consist of position and velocity vectors. These conversions are able to be performed with a variety of inputs, so listed below are the orbit types available for accurate conversion.

- **Elliptic Orbit** $(\boldsymbol{0 < e < 1.0}, \quad \boldsymbol{a > 0})$
  - Inclined: $i > 0, \quad \Omega > 0$
  - Equatorial: $i = 0, \quad \Omega = 0$
- **Circular Orbit** $(\boldsymbol{e = 0}, \quad \boldsymbol{a > 0}, \quad \boldsymbol{\omega = 0})$
  - Inclined: $i > 0, \quad \Omega > 0$
  - Equatorial: $i = 0, \quad \Omega = 0$
- **Parabolic orbit** $(\boldsymbol{e = 1.0}, \quad \boldsymbol{a = -r_p})$
  - Inclined: $i > 0, \quad \Omega > 0$
  - Equatorial: $i = 0, \quad \Omega = 0$
- **Hyperbolic orbit** $(\boldsymbol{e > 1.0}, \quad \boldsymbol{a < 0})$
  - Inclined: $i > 0, \quad \Omega > 0$
  - Equatorial: $i = 0, \quad \Omega = 0$

3 Model Assumptions and Limitations

3.1 Assumptions

- The origin of the inertial frame is coincident with the geocentric equatorial system.
- The attracting body is specified by the supplied gravity constant $\mu[\frac{km^3}{s^2}]$.

3.2 Limitations

- **For input $e \geqslant 1.0$**
  - Rectilinear orbits are not supported with this module because their angular momentum equals zero. However, Cartesian vectors can be obtained from orbital elements because this only affects the Cartesian to element conversion. Regardless, this case will not be needed when using the orb_elem_convert module.
  - The semimajor axis input must be negative.
- **For input $e = 0$**
  - The argument of periapsis input must be zero.
- **For input $a = 0$**
  - The ascending node input must be zero.
- **For input $a < 0$**
  - The eccentricity must be greater than or equal to 1.0.
- **Orbital element angles (i, Ω, ω, f)**
  - Must be greater than or equal to zero
  - Must be radians

4 Test Description and Success Criteria

The unit test, test_orb_elem_convert.py, validates the internal aspects of the Basilisk orb_elem_convert module by comparing module output to expected output. The unit test validates the conversions performed between Keplerian orbital elements and Cartesian vectors. This test begins by converting elements to Cartesian and is immediately followed by converting the results back to orbital elements, providing the opportunity to compare the initial input with the final output. This is repeated for a variety of parameters.

Differences between simulated and calculated output are measured, and those that approximately equal zero (typically $\delta < 10^{-9}$) qualify as a successful test. However, since the semimajor axis and the Cartesian vectors are greater than the other parameters by a factor of at least 10^7, a higher discrepency is more tolerable. These tolerances are more clearly defined with the results.

4.1 Element to Cartesian

This test verifies the conversion from orbital elements to Cartesian vectors by comparing position and velocity vectors calculated in Python to those determined in the module. It uses six orbital elements and a gravitational constant as inputs. Also, *Elements2Cart* and *inputsGood* must be set to *True*, which is discussed in more detail in the next section. The *useEphemFormat* parameter is alternated between *True* and *False* to test the use of both spacecraft and planet data. The calculations for this conversion are performed seperately in the test module to compare with the simulated results. Multiple tests were performed, varying semimajor axis and eccentricity inputs.

4.1.1 Calculation vs. Module

The first check compares calculated Cartesian vectors to the simulated results and verifies that the discrepancy is less than 10^{-9}.

4.2 Cartesian to Element

This test verifies, via two checks, that the conversion from Cartesian vectors to orbital elements is accurate. This test requires position and velocity vectors, which are obtained through initially converting from orbital elements to Cartesian, as described above. It also needs a gravitational constant input. The *Elements2Cart* is set to *False* and *inputsGood* must be set to *True*. Like in the previous conversion test, *useEphemFormat* is alternated between *True* and *False* to ensure that the Cartesian vectors can use both planet and spacecraft state data identically and accurately. The calculations for this conversion are also performed seperately in the test to compare with the simulated results.

4.2.1 Calculation vs. Module

Similar to the first conversion test, this check compares the calculated orbital elements to the elements obtained from the module.

4.2.2 Input vs. Module

This second check is performed last and compares the final result, comprised of orbital elements, with the orbital elements initially used as inputs for the first conversion test. This verifies that the two conversion processes can be combined to reattiain the original orbital elements.

5 Test Parameters

The previous description only briefly mentions the input parameters required to perform the tests on this module, so this section details the parameters further. In addition to the parameters being converted, the unit test requires inputs that identify which conversion is performed, what type of state data are being used, and whether or not to proceed with the given input.

1. Euler Angles:
   Euler angles, consisting of i, Ω, and ω, define the orbit plane orientation and may vary, depending on the orbit type. All non-zero values for these angles were pulled from an existing module and are actually used in many other Basilisk tests in the scenarios directory. This parameter set gives the following inclination, ascending node, and argument of periapses inputs:

$$i = 33.3^\circ = 0.5812\text{rad} \tag{36}$$

$$\Omega = 48.2^\circ = 0.8412\text{rad} \tag{37}$$

$$\omega = 347.8^\circ = 6.0703\text{rad} \tag{38}$$

   Inputs alternate between zero and these given values depending on the orbit type, which is specified in the results section. All angles must be input as radians, so the conversion from degrees is performed as shown in Eq. 36 - 38.

2. True Anomaly:
   The true anomaly input stays the same for each parameter set because the different orbits do not require a unique value, as they may for inclination, ascending node, and argument of periapsis. The true anomaly value, however, is pulled from the same module as those parameters, giving

$$f = 85.3^{\circ} = 1.489\text{rad} \tag{39}$$

3. Standard Gravitational Parameter:
   The gravitational parameter μ is necessary for any conversion between orbital elements and Cartesian vectors. It is the product of a body's gravitational constant and mass, specifying the attracting body in units of $\frac{m^3}{s^2}$. This test specifies Earth as the astronomical body.

$$\mu = GM_{Earth} = 3.986\text{e+}14 \quad \frac{m^3}{s^2} \tag{40}$$

4. Conversion Identifier:
   In order to specify which conversion to perfom, that is Element to Cartesian or the reverse, an identifier must be set. In the test as well as the c++ file, this is a boolean flag, referred to as *Elements2Cart* and indicating the cases displayed in the table below.

Table 2: Elements2Cart Cases

Case	Conversion
True	Element to Cartesian
False	Cartesian to Element

5. Astronomical Body Identifier
   Like the previous identifier, this is a boolean parameter that specifies how to use the module. Since the state data may describe any body in orbit, a value is needed to determine whether the conversions are being performed for a spacecraft or a planet. As stated previously, the test uses *useEphemFormat* for this, providing the following options:

Table 3: useEphemFormat Cases

Case	Body
True	Planet State Data
False	Spacecraft State Data

6. Valid Input Identifier
   The last necessary parameter is another boolean flag, this time indicating whether the inputs are valid. This value is read from a message and distinguishes if it is written succesfully. If succesful, this allows the rest of the module to be run, initiating the conversion process. The test uses *inputsGood* to execute this check where *True* lets the module proceed and *False* does not, so this value should always be *True* for the module to execute properly.

The orbital element parameters are varied throughout the test to validate a range of orbits. Since the test performs both conversions consecutively and the Cartesian results depend on the Keplerian inputs, the position and velocity vectors used for the second conversion also vary. These parameters are discussed with the results in the next section.

Table 4: Error Tolerance-Note: Absolute Tolerance is abs(truth-value)

Test	Calc/Sim Tolerance	Input/Output Tolerance
a	10^{-9} km	10^{-7} km
e	10^{-9}	10^{-9}
i	10^{-9} rad	10^{-9} rad
Ω	10^{-9} rad	10^{-10} rad
ω	10^{-9} rad	10^{-9} rad
f	10^{-9} rad	10^{-9} rad
$\boldsymbol{r}$	10^{-9} km	N/A
$\boldsymbol{v}$	10^{-9} $\frac{\text{km}}{\text{s}}$	N/A

6 Test Results

All checks within test_orb_elem_convert.py passed as expected with a tolerance limit of about 10^{-9}. This means all comparisons should have a discrepancy of no more than 10^{-9}. This may vary depending on the parameters in question, for example, when comparing the semimajor axis, since it is a factor of at least 10^4 larger than the other parameters. The tables in this section show the results of each test for various parameters followed by descriptions of the parameter sets.

- **Element to Cartesian**

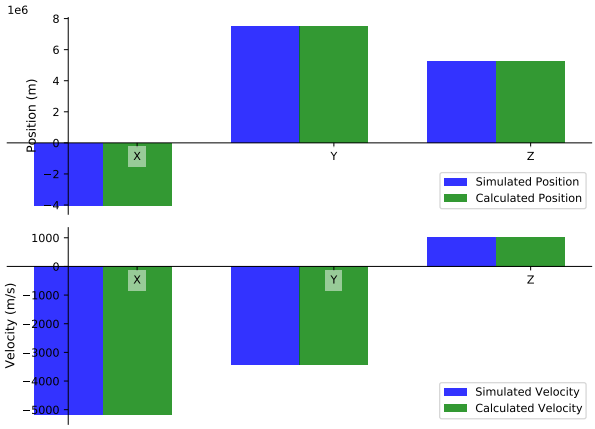
Table 5: Element to Cartesian Test Results

Parameter Sets	Results	Notes
1	PASSED	
2	PASSED	
3	PASSED	
4	PASSED	
5	PASSED	
6	PASSED	
7	PASSED	
8	PASSED	

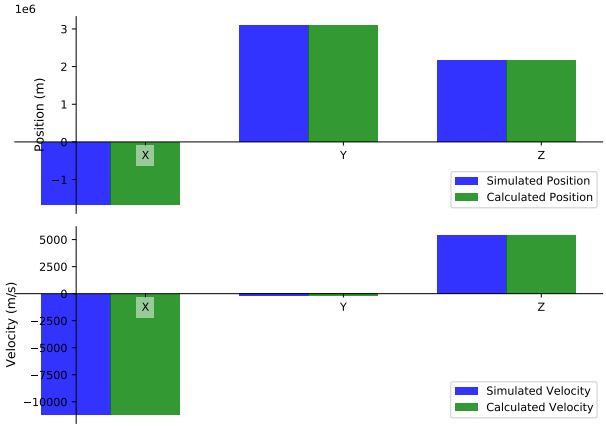
Cartesian vectors are tested by comparing the simulated results of the conversion with calculated vectors. The discrepancy is expected to be no greater than 10^{-9} for both position and velocity. The two cases presented by the astronomical body identifier (*useEphemFormat*), discussed in the Test Parameters section, produce identical results. Thus, they will not be referred to as seperate cases any further and only the various orbits will be analyzed. Plots comparing the position and velocity vectors are shown for each orbit. Only the upper and lower limits of the tests are displayed, providing insight into the effect of the changing parameters without an overwhelming amount of data.

1. Inclined Elliptic Orbit $(0 < e < 1.0, \quad a > 0)$

   To test elliptic orbtits, a range of semimajor axis inputs from $10 - 10^7$km is used. The eccentricity is also varied with inputs from $0.01 - 0.75$. As a changes, $e = 0.5$ and as e changes, $a = 10^7$km. All parameter sets passed for this case since the difference when comparing simulated and calculated results was always zero, as shown by Fig. 2 and 3.

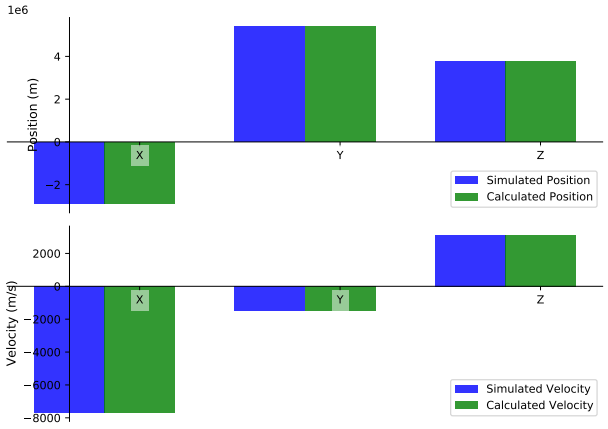


(a) $e = 0.01$ and $a = 10^7$km

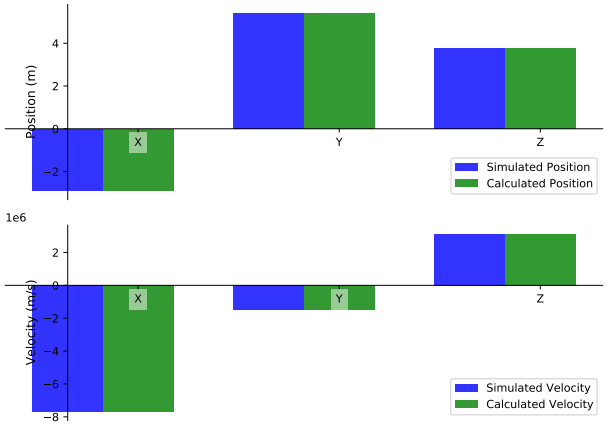


(b) $e = 0.75$ and $a = 10^7$km

Fig. 2: Inclined Elliptical Orbit Varying e



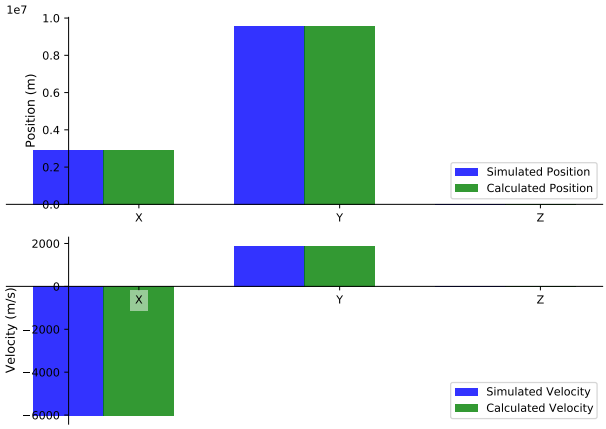
(a) $e = 0.5$ and $a = 10^7$km



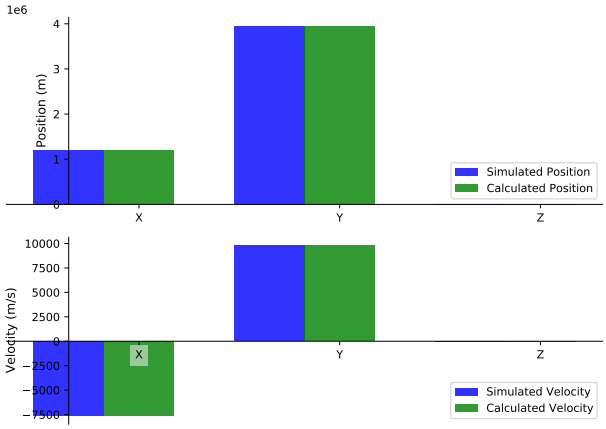
(b) $e = 0.5$ and $a = 10$km

Fig. 3: Inclined Elliptical Orbit Varying a

2. Equatorial Elliptic Orbit $(0 < e < 1.0, \quad a > 0, \quad i = 0, \quad \Omega = 0)$
The equatorial elliptic orbit is tested with the same range of e and a inputs as the inclined case. All variations also passed this test with a tolerance of zero, as shown in the following set of figures:

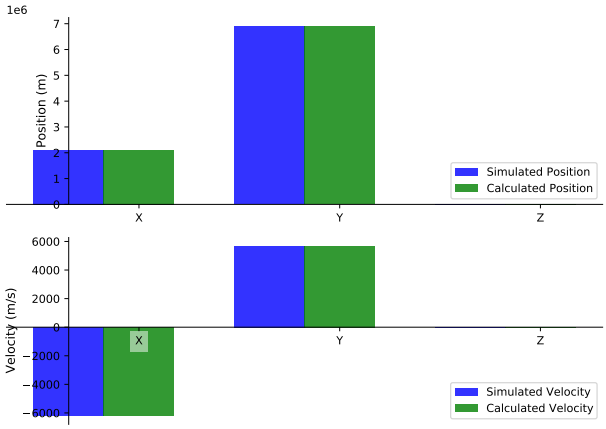


(a) $e = 0.01$ and $a = 10^7$km

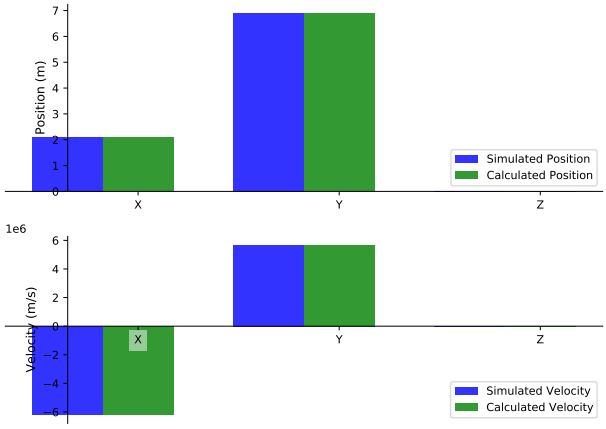


(b) $e = 0.75$ and $a = 10^7$km

Fig. 4: Equatorial Elliptical Orbit Varying e



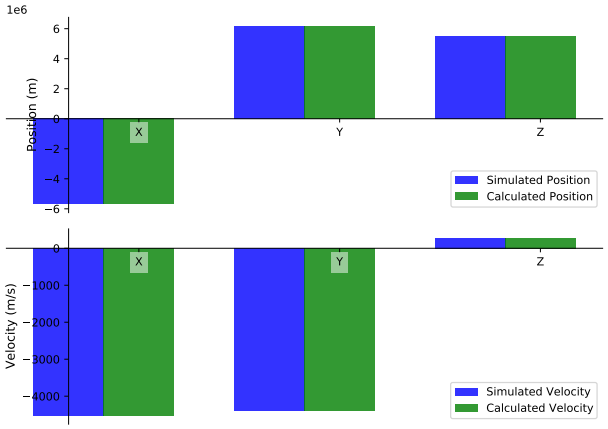
(a) $e = 0.5$ and $a = 10^7$km



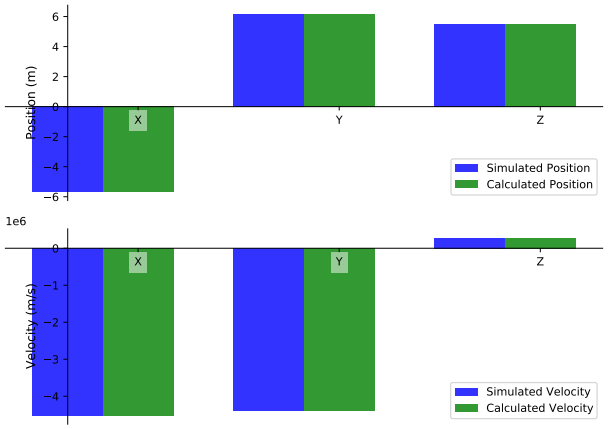
(b) $e = 0.5$ and $a = 10$km

Fig. 5: Equatorial Elliptical Orbit Varying a

3. Inclined Circular Orbit $(e = 0, \quad a > 0, \quad \omega = 0)$
   Since $e = 0$ for a circular orbit, the only varied input is a, which is limited to the same range as in the elliptical cases. All variations passed for the inclined circular orbit, again, with no discrepancy.



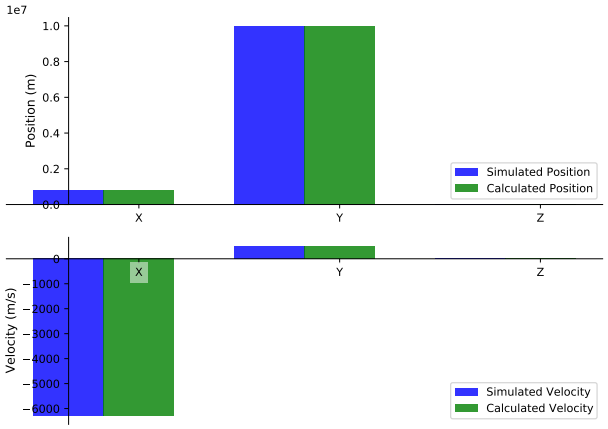
(a) $a = 10^7$km



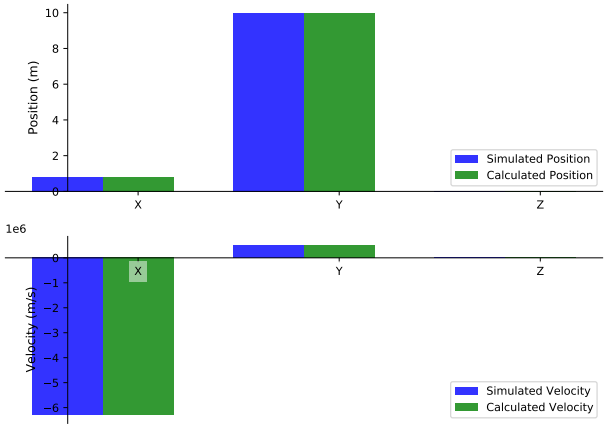
(b) $a = 10$km

Fig. 6: Inclined Circular Orbit Varying a

4. Equatorial Circular Orbit $(e = 0, \quad a > 0, \quad \omega = 0, \quad i = 0, \quad \Omega = 0)$
   Like the inclined case, the only input varied for equatorial circular orbits is a. Also, all variations passed with no discrepancy, as shown in Fig. 7.



(a) $a = 10^7$km

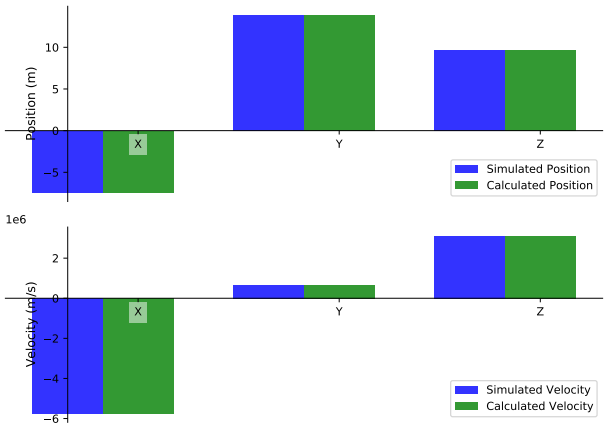


(b) $a = 10$km

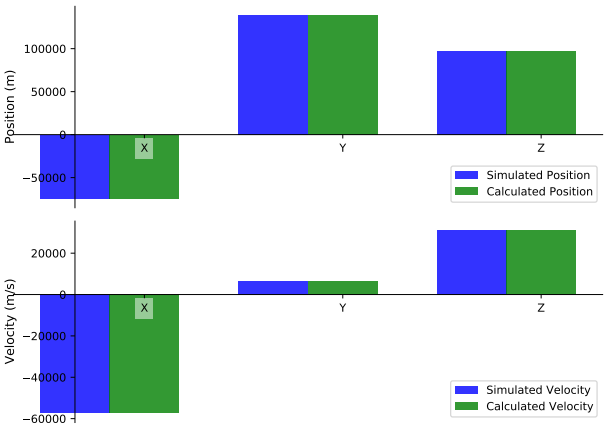
Fig. 7: Equatorial Circular Orbit Varying a

5. Inclined Parabolic Orbit ($e = 1.0, \quad a = -r_p$)

   For the parabolic orbit, the input is varied with -10^5km $< a < -10$km. As discussed in the model description, it is crucial that a remains negative in the parabolic case. All variations passed for this case with discrepancies equal to zero displayed below.



(a) $a = 10^7$km

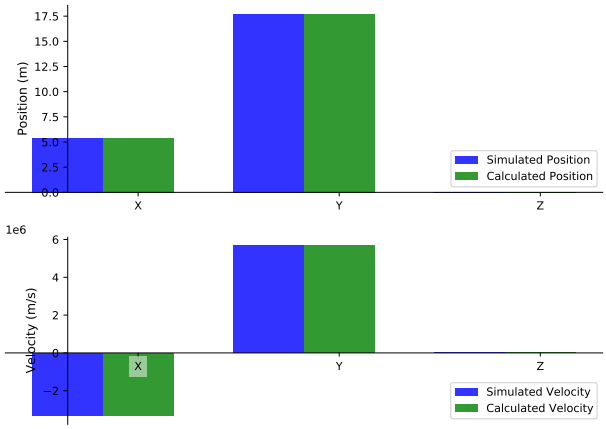


(b) $a = 10$km

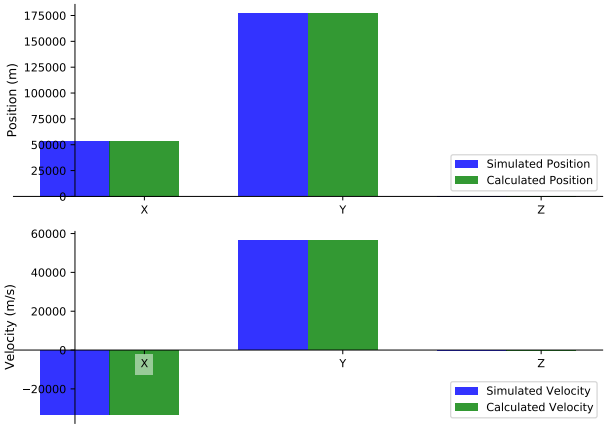
Fig. 8: Inclined Parabolic Orbit Varying a

6. Equatorial Parabolic Orbit ($e = 1.0, \quad a = -r_p, \quad i = 0, \quad \Omega = 0$)

   The equatorial parabolic orbit is tested with the same range of a as the inclined case. This orbit also passed for all variations and the results are shown in Fig. 9.



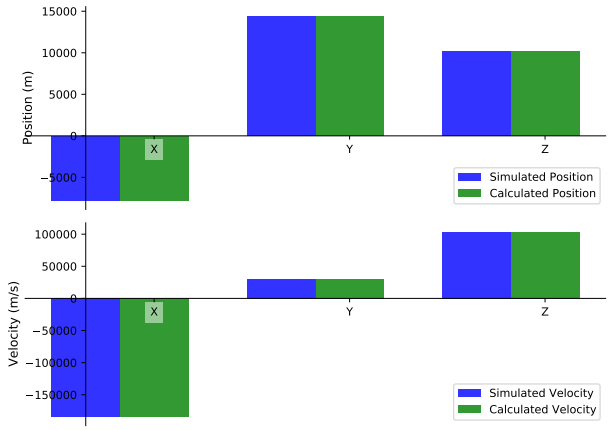
(a) $a = 10^7$km



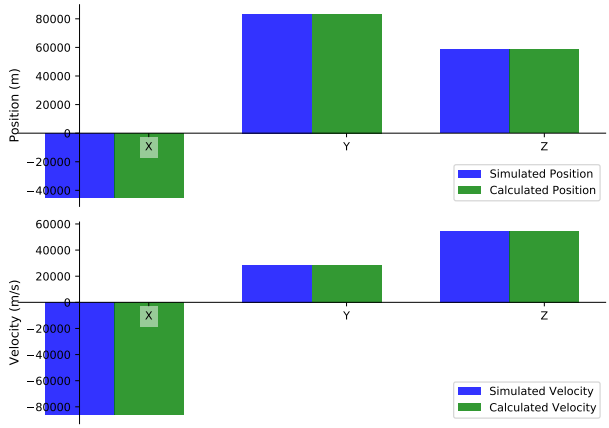
(b) $a = 10$km

Fig. 9: Equatorial Parabolic Orbit Varying a

7. Inclined Hyperbolic Orbit ($e > 1.0, \quad a < 0$)
When testing hyperbolic orbits, e and a are both varied. While changing e within the range $1.1 < e < 1.5$, $a = -10^4$km. While changing a in the range -10^5km $< a < -10$km, $e = 1.3$. All variations passed the tests, still with no discrepancies. The plots below compare the simulated and calculated results.

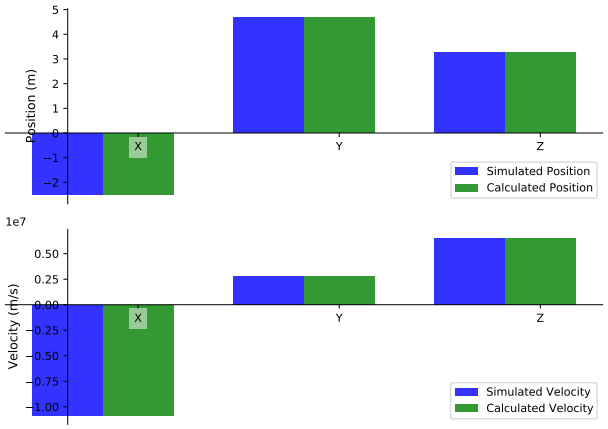


(a) $e = 1.1$, $a = -10^5$km

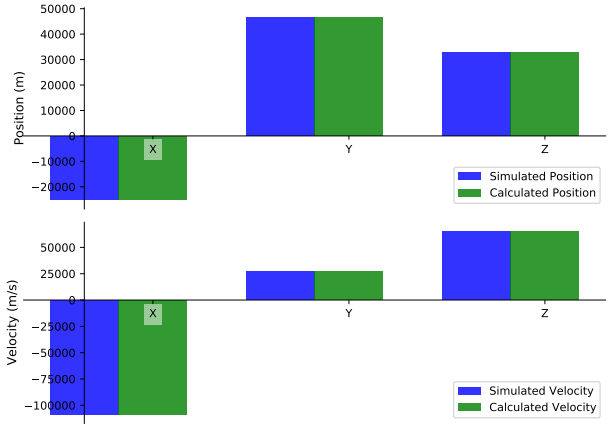


(b) $e = 1.5$, $a = -10^5$km

Fig. 10: Inclined Hyperbolic Orbit Varying e



(a) $e = 1.3$, $a = -10$km

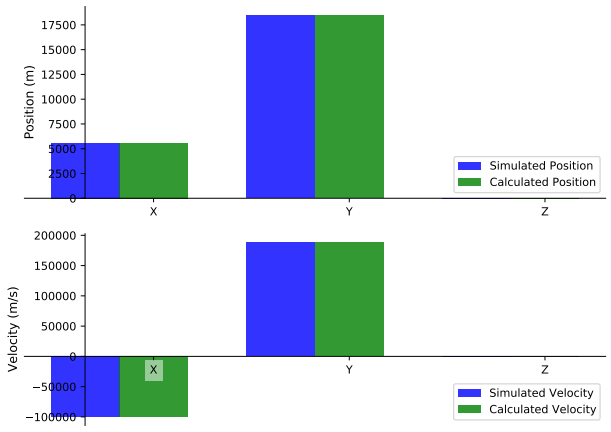


(b) $e = 1.3$, $a = -10^5$km

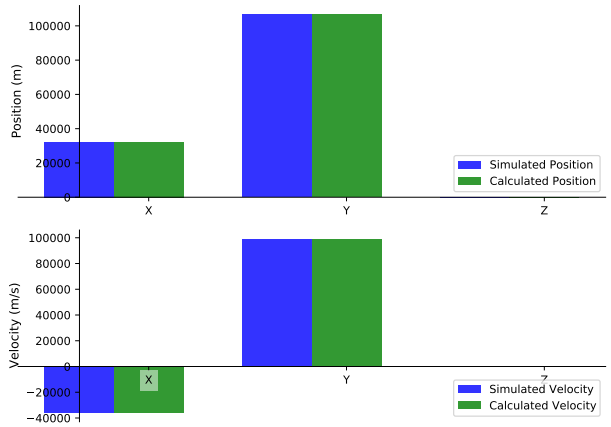
Fig. 11: Inclined Hyperbolic Orbit Varying a

8. Equatorial Hyperbolic Orbit ($e > 1.0, \quad a < 0, \quad i = 0, \quad \Omega = 0$)

Equatorial hyperbolic orbits are tested with the same range of e and a inputs as the inclined case. Also, all variations passed without discrepancy, as shown in the following Fig. 12 and 13:

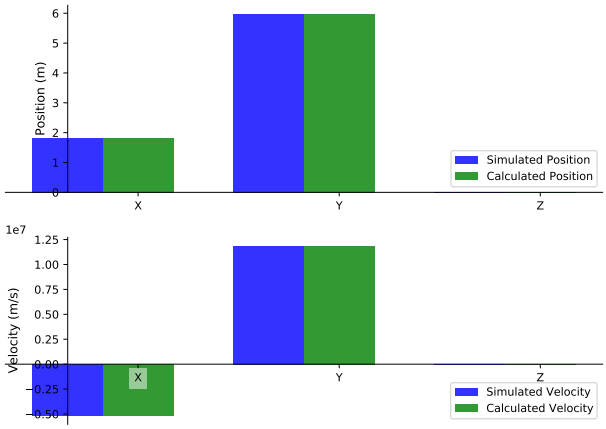


(a) $e = 1.1$,$a = -10^5$km

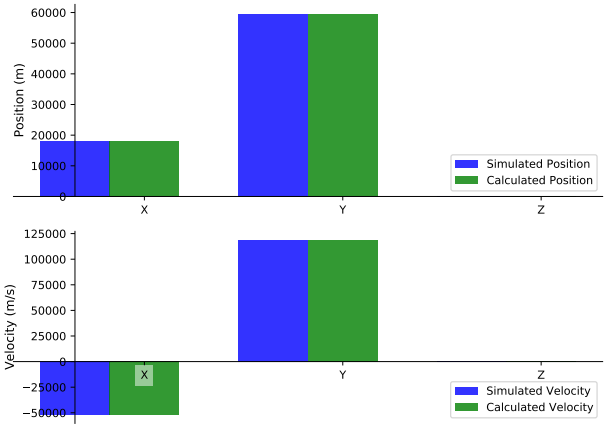


(b) $e = 1.5$, $a = -10^5$km

Fig. 12: Equatorial Hyperbolic Orbit Varying e



(a) $e = 1.3$, $a = -10$km



(b) $e = 1.3$, $a = -10^5$km

Fig. 13: Equatorial Hyperbolic Orbit Varying a

- **Cartesian to Element**

Table 6: Cartesian to Element Test Results

Parameter Sets	Results	Notes
1	PASSED	
2	PASSED	
3	PASSED	
4	PASSED	
5	PASSED	
6	PASSED	
7	PASSED	
8	PASSED	

Orbital elements are tested by making two comparisons. The first compares the simulated output with the calculated output, like in the previous conversion. The second compares elements initially used as inputs for the first conversion with those produced as the output of the second. As observed when testing the previous conversion, the comparison between calculated and simulated elements results in complete agreement. Also, the two cases presented by the astronomical body identifier, once again, produce identical results. Thus, the main focus of the Cartesian to element test results is the input/output comparison.

Displayed below are explanations of the results after converting Cartesian to elements. Graphical representations of the parameter comparisons for each orbit type are not provided for this test, since the output is already specified above as the input when converting from elements to Cartesian. Figures from above can also be referenced for the Cartesian inputs.

1. Inclined Elliptic Orbit $(0 < e < 1.0, \quad a > 0)$
   All variations passed the tests performed, however, the discrepancy between semimajor axis initial input and simulated result increases as eccentricity approaches 1.0. This is because it approaches the parabolic case, which requires a different conversion method and makes the elliptic calculations less accurate. The discrepancy with the tested parameters never increases past 10^{-7}km, though, and since the semimajor axis is typically on the scale of at least 10^{7}km, this is acceptable. Fig. 14 shows the change in semimajor axis discrepancy as eccentricity reaches 0.999.

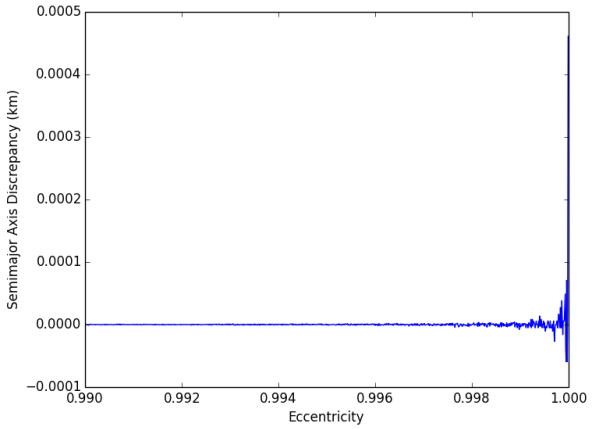


Fig. 14: Input/Output Discrepancy for Semimajor Axis as Eccentricity Approaches 1.0

2. Equatorial Elliptic Orbit $(0 < e < 1.0, \quad a > 0, \quad i = 0, \quad \Omega = 0)$
   All variations also passed for this case with much lower discrepancies than the previous one, due to the initial input $e = 0.5$. The largest difference still occurs when comparing the semimajor axis, but this time it only reaches 10^{-15}km.
3. Inclined Circular Orbit $(e = 0, \quad a > 0, \quad \omega = 0)$
   All variations passed in the inclined circular orbit tests with the maximum tolerance at about 10^{-9}. As initial input a decreases, so does the tolerance, resulting in no discrepancy when $a = 10$km.
4. Equatorial Circular Orbit $(e = 0, \quad a > 0, \quad \omega = 0, \quad i = 0, \quad \Omega = 0)$
   All variations passed as well, this time resulting in a tolerance of zero for almost all parameters. The only non-zero discrepancy is eccentricity with a difference of 10^{-17} between input and output.
5. Inclined Parabolic Orbit $(e = 1.0, \quad a = -r_p)$
   All variations passed for this orbit with tolerances well below the maximum allowed and decreasing as initial input a becomes more negative.
6. Equatorial Parabolic Orbit $(e = 1.0, \quad a = -r_p, \quad i = 0, \quad \Omega = 0)$
   This orbit also passed for all variations with discrepancies similar to those seen in the inclined case.
7. Inclined Hyperbolic Orbit $(e > 1.0, \quad a < 0)$
   All tests passed with discrepancies well below the maximum limit.
8. Equatorial Hyperbolic Orbit $(e > 1.0, \quad a < 0, \quad i = 0, \quad \Omega = 0)$
   Each variation passed when tested, satisfying the tolerance limitation. The equatorial case exhibits lesser discrepancies than the inclined.

7 User Guide

The three possible input messages are:

- `scStateInMsg`: spacecraft state input message
- `spiceStateInMsg`: spice state input message
- `elemInMsg`: classical orbit elements input message

Note that only one can be connected to.

The three output messages are:

- `scStateInMsg`: spacecraft state input message
- `spiceStateInMsg`: spice state input message
- `elemInMsg`: classical orbit elements input message

Here you can connect to any of these messages. This makes it possible to take a single input and convert into multiple output formats at the same time.

The gravitational constant μ must be defined in all cases.

```
orb_elemObject = orbElemConvert.OrbElemConvert()
orb_elemObject.ModelTag = "OrbElemConvertData"
orb_elemObject.mu = mu
```

REFERENCES

[1] Vallado, D. A., and McClain, W. D., *Fundamentals of Astrodynamics and Applications, 4th ed.* Hawthorne, CA: Published by Microcosm Press, 2013.

[2] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems, 3rd ed.*. Reston, VA: American Institute of Aeronautics and Astronautics.