Assignment 1: BASICS-2

1. Converting a Kepler state vector

Convert the following Cartesian state-vector to Keplerian elements:

$$x = 10157768.1264; y = -6475997.0091; z = 2421205.9518; [m]$$

$$\dot{x} = 1099.2953996; \dot{y} = 3455.1059240; \dot{z} = 4355.0978095; [m/s]$$

1.1. Initial state

One first defines current state of the Cartesian components into the Cartesian state vector. There are two vectors, the position (radius), pointing from the orbited body to the orbiting body and the velocity of the orbiting object with respect to the orbited object.

$$r = \{x, y, z\} \tag{1}$$

$$v = \{\dot{x}, \dot{y}, \dot{z}\} \tag{2}$$

Furthermore, since the satellite is orbiting the earth, the standard gravitational parameter μ can be set to that of the earth.

$$\mu = 398600.4418 \;\; 10^9[m^3/s^2] \tag{3}$$

1.2. Conversion process

We can define several parameters that can aid in the conversion process. These parameters are frequently used when working with Kepler orbits. The first one is the specific angular momentum (see eq. 4). In Kepler orbits this is a preserved quantity:

$$h = r \times v \tag{4}$$

There is also the nodal vector. This is the vector that points from the center orbited body (Earth) to the ascending node of the orbit. It can be extracted from the angular momentum using:

$$n = \{0, 0, 1\} \times h \tag{5}$$

We can start the conversion process with the eccentricity. It is derived from the eccentricity vector (see eq. 6) . The actual eccentricity is the L2 norm of the eccentricity vector:

$$\vec{e} = \frac{v \times h}{\mu} - \frac{\boldsymbol{r}}{\|r\|} \tag{6}$$

$$e = \left\| \vec{e} \right\| \tag{7}$$

The semi major axis can be found (see eq. 8). A special case exists for the semi major axis, if the orbit is parabolic, then the semi major axis is defined as infinity.

$$a = \begin{cases} \frac{1}{\frac{2}{\|r\|} - \frac{\|v\|^2}{\mu}} & e \neq 1 \\ \infty & e = 1 \end{cases} \tag{8}$$

The other orbital parameters can be deduced of the previous values with the sets of equations below (see eq. 8-11).

$$i = \cos^{-1}\left(\frac{h_z}{\|h\|}\right) \tag{9}$$

$$\Omega = \cos^{-1}\left(\frac{n_x}{\|n\|}\right) \tag{10}$$

$$\omega = \cos^{-1}\left(\frac{n.\vec{e}}{e\|n\|}\right) \tag{11}$$

$$\nu = \cos^{-1}\left(\frac{\vec{e}.r}{e\|r\|}\right) \tag{12}$$

However special care needs to be taken that the angles are in the correct quadrant. This is done using various quadrant checks.

- $n_y < 0 \Longrightarrow \Omega = 2\pi - \Omega$
- $\vec{e}_z < 0 \Longrightarrow \omega = 2\pi - \omega$
- $r \cdot v < 0 \Longrightarrow \nu = 2\pi - \nu$

1.3. Results

The equations where evaluated using a coputational tool, Mathematica[1]. For the Cartesian state, the computed Keplerian elements are:

- $a = 1.216495891 \times 10^7 [m]$
- $e = 0.01386952771[-]$
- $i = 52.67767044[°]$
- $\omega = 151.4337673[°]$
- $\Omega = 318.6663261[°]$
- $M = 222.9126712[°]$

2. Converting Cartesian state vector

Convert the following state-vector from Kepler elements to Cartesian components:

- $a = 12269687.5912[m]$
- $e = 0.004932091570[-]$
- $i = 109.823277603[°]$
- $\omega = 106.380426142[°]$
- $\Omega = 134.625563565[°]$
- $M = 301.149932402[°]$

2.1. Conversion process

For easy conversion between the Cartesian and Keplerian state vectors one should use the equivalent parameter true anomaly ν instead of the mean anomaly M. This conversion is done via the eccentric anomaly E. This anomaly can be determined by iterating equation 13. The start value used was the mean anomaly.

$$\mathrm{E}_{i+1} = M + e \cdot sin(\mathrm{E}_i) \tag{13}$$

Then using the eccentric anomaly the true anomaly can be found using simple trigonometry rules:

$$\nu = \tan^{-1}\left(\frac{\sqrt{1-e^2}\sin(\mathrm{E})}{\cos(\mathrm{E})-e}\right) \tag{14}$$

Now we can convert the Keplerian elements into r and v. This is done in two steps (based on the approach in [2]). First one converts the Kelper elements to r_{pqw} and v_{pqw} in the PQW reference system. Then both the position and velocity vector are rotated to the ECI Frame. Computing r_{pqw} and v_{pqw} is done based on the following set of equations:

$$r_{\text{pqw}} = \left\{\frac{p\cos(\nu)}{e\cos(\nu)+1}, \frac{p\sin(\nu)}{e\cos(\nu)+1}, 0\right\} \tag{15}$$

$$v_{\text{pqw}} = \left\{-\sqrt{\frac{\mu}{p}}\sin(\nu), \sqrt{\frac{\mu}{p}}(e+\cos(\nu)), 0\right\} \tag{16}$$

With $p = a(1-e^2)$

To transform r_{pqw} and v_{pqw} to the ECI frame, We set up a transformation matrix. This matrix is build up of the 313 rotation of the angles defining the orbital plane (see eq. 17).

$$T^{\text{xyz}}_{\text{pqr}} = T_z(\omega) \cdot T_x(i) \cdot T_z(\Omega) \tag{17}$$

This leads to the following equations for the ECI velocity en position vector:

$$r_{\text{xyz}} = T^{\text{xyz}}_{\text{pqr}}(\omega, i, \Omega) \cdot r_{\text{pqw}} \tag{18}$$

$$v_{\text{xyz}} = T^{\text{xyz}}_{\text{pqr}}(\omega, i, \Omega) \cdot v_{\text{pqw}} \tag{19}$$

2.2. Results

Then performing the above computations in Mathematica[1] yields the following Cartesian state vector:

$$r_{xyz} = \{-3.69645903910^6, 8.06926849910^6, 8.42653655810^6\}[m] \tag{20}$$

$$v_{xyz} = \{3884.880912, -2064.829168, 3646.340862\}[m/s] \tag{21}$$

References

[1] W. Research. (2008) Mathematica edition: Version 7.0. Champaign, Illinois.

[2] D. A. Vallado and W. D. McClain, *Fundamentals of Astrodynamics and Applications*. Microcosm Press, 2004.

[3] R. Noomen, *AE4-879 Basics*, TUDelft Lecture Slides, 2010.

Additional information

Estimated work time:

$\sim$ 1h Studying theory + $\sim$ 1.5h making assignment + $\sim$ 2.5h writing report = $\sim$ 5h

Made by

Simon Billemont

Stud Nr: 1387855

s.billemont@student.tudelft.nl

Version history

Version 1: Initial document

A. Mathematica source code

This is the source code in the form of two Mathematica notebooks. One (CartesianToKepler.nb) is for the conversion from the Cartesian to the Keplerian state vector. KeplerToCartesian.nb is for the conversion in the other direction.

CartesianToKepler.nb | 1

```
In[398]:= x = 10157768.1264; y = -6475997.0091; z = 2421205.9518;
          ẋ = 1099.2953996; ẏ = 3455.1059240; ż = 4355.0978095;

In[400]:= r = {x, y, z}(* Radius [m]*)
Out[400]= {1.01578×10^7, -6.476×10^6, 2.42121×10^6}

In[401]:= v = {ẋ, ẏ, ż}(*Velocity [m/s]*)
Out[401]= {1099.3, 3455.11, 4355.1}

In[402]:= μ = 398600.4418; (* mhu km^3/s^2*)
          μ = μ 10^9 (* mhu m^3/s^2*)
Out[403]= 3.986×10^14

In[404]:= h = r×v(* Specific angular momentum*)
Out[404]= {-3.65691×10^10, -4.15765×10^10, 4.22152×10^10}

In[405]:= n = {0, 0, 1}×h (* Nodal vector*)
Out[405]= {4.15765×10^10, -3.65691×10^10, 0.}

In[406]:= ē = v×h/μ - r/‖r‖;(* Eccentricity vector *)
          e = ‖ē‖ (* eccentricity[-]*)
Out[407]= 0.0138695

In[408]:= If[e == 1, (*Parabolic orbit ?*)
           a = Infinity,
           a = 1 / (2/‖r‖ - ‖v‖^2/μ)] (*SemiMajor axis [m]*)
Out[408]= 1.2165×10^7
```

In[409]:=
```
i = ArcCos[h[[3]]/‖h‖];(* inclination [rad] *)
i 180/π // N(* show inclination in [degree] *)
(* //N is convert to numeric form *)
```

Out[410]=
```
52.6777
```

In[411]:=
```
Ω = ArcCos[n[[1]]/Norm[n]];(* Right ascension of ascending node [rad] *)
If[n[[2]] < 0, Ω = 2 π - Ω]; (* Checking for quadrant *)
Ω 180/π // N (* Show raan in degree *)
```

Out[413]=
```
318.666
```

In[414]:=
```
ω = ArcCos[Dot[n, ē]/(Norm[n] e)] ;(* Argument of perigee [rad] *)
If[ē[[3]] < 0, ω = 2 π - ω];(* Checking for quadrant *)
ω 180/π // N(* Show ω in degree *)
```

Out[416]=
```
151.434
```

In[417]:=
```
ν = ArcCos[Dot[ē, r]/(e Norm[r])]; (* True anomaly [rad] *)
If [Dot[r, v] < 0, ν = 2 π - ν];(* Checking for quadrant *)
ν 180/π // N(* Show true anomaly in degree *)
```

Out[419]=
```
222.913
```

In[420]:=
```
fromRad = 180/π;
Print["a = ", NumberForm[a, 10], " [ m ]"];
Print["e = ", NumberForm[e, 10], " [ – ]"];
Print["i = ", NumberForm[i fromRad , 10], " [ ° ]"];
Print["ω = ", NumberForm[ω fromRad, 10], " [ ° ]"];
Print["Ω = ", NumberForm[Ω fromRad, 10], " [ ° ]"];
Print["ν = ", NumberForm[ν fromRad, 10], " [ ° ]"];
```


In[234]:=
```
a = 12 269 687.5912;
e = 0.004932091570;
i = 109.823277603 (π / 180);
ω = 106.380426142 (π / 180);
Ω = 134.625563565 (π / 180);
M = 301.149932402 (π / 180);
```

In[240]:=
```
μ = 398 600.4418; (* mhu km^3/s^2*)
μ = μ 10^9 (* mhu m^3/s^2*)
```

Out[240]=
```
3.986 × 10^14
```

In[241]:=
```
(*function E = M + e Sin[E] to iterate over*)
EAnew[M_, e_, EA_] := M + e Sin[EA]

EA = M;
EA2 = Infinity;
While[Abs[EA - EA2 ] > 10^-18,
 EA2 = EA;
 EA = EAnew[M, e, EA2]]
EA 180/π // N (*Eccentric anomaly [degree]*)
```

Out[245]=
```
300.907
```

In[246]:=
```
ν = ArcTan[Cos[EA] - e, Sqrt[1 - e^2] * Sin[EA]]; (*ArcTan[x, y] form*)
ν 180/π // N (*True anomaly [degree]*)
```

Out[246]=
```
-59.3353
```

In[247]:=
```
p = a * (1 - e * e)
```

Out[247]=
```
1.22694 × 10^7
```

In[248]:=
```
r_pqw = {p Cos[ν]/(1 + e Cos[ν]), p Sin[ν]/(1 + e Cos[ν]), 0}
```

Out[248]=
```
{6.24185 × 10^6, -1.05272 × 10^7, 0}
```

In[249]:=
```
v_pqw = {-Sqrt[μ/p] Sin[ν], Sqrt[μ/p] (e + Cos[ν]), 0}
```

Out[249]=
```
{4902.75, 2935.07, 0}
```

In[250]:=
```
T[ω_, i_, Ω_] := RotationMatrix[Ω, {0, 0, 1}].
  RotationMatrix[i, {1, 0, 0}].RotationMatrix[ω, {0, 0, 1}] (* T_pqr^xyz *)
T[ω_, i_, Ω_] // MatrixForm
```

Out[251]//MatrixForm=

```
( Cos[ω_] Cos[Ω_] - Cos[i_] Sin[ω_] Sin[Ω_]   -Cos[Ω_] Sin[ω_] - Cos[i_] Cos[ω_] Sin[Ω_]   Sir
  Cos[i_] Cos[Ω_] Sin[ω_] + Cos[ω_] Sin[Ω_]   Cos[i_] Cos[ω_] Cos[Ω_] - Sin[ω_] Sin[Ω_]   -Co
  Sin[i_] Sin[ω_]                              Cos[ω_] Sin[i_]
```

In[252]:=
```
r_xyz = T[ω, i, Ω].r_pqw;
NumberForm[r_xyz, 10]
```

Out[253]//NumberForm=

$\{-3.696459039 \times 10^6, 8.069268499 \times 10^6, 8.426536558 \times 10^6\}$

In[254]:=
```
v_xyz = T[ω, i, Ω].v_pqw;
NumberForm[v_xyz, 10]
```

Out[255]//NumberForm=

{3884.880912, -2064.829168, 3646.340862}