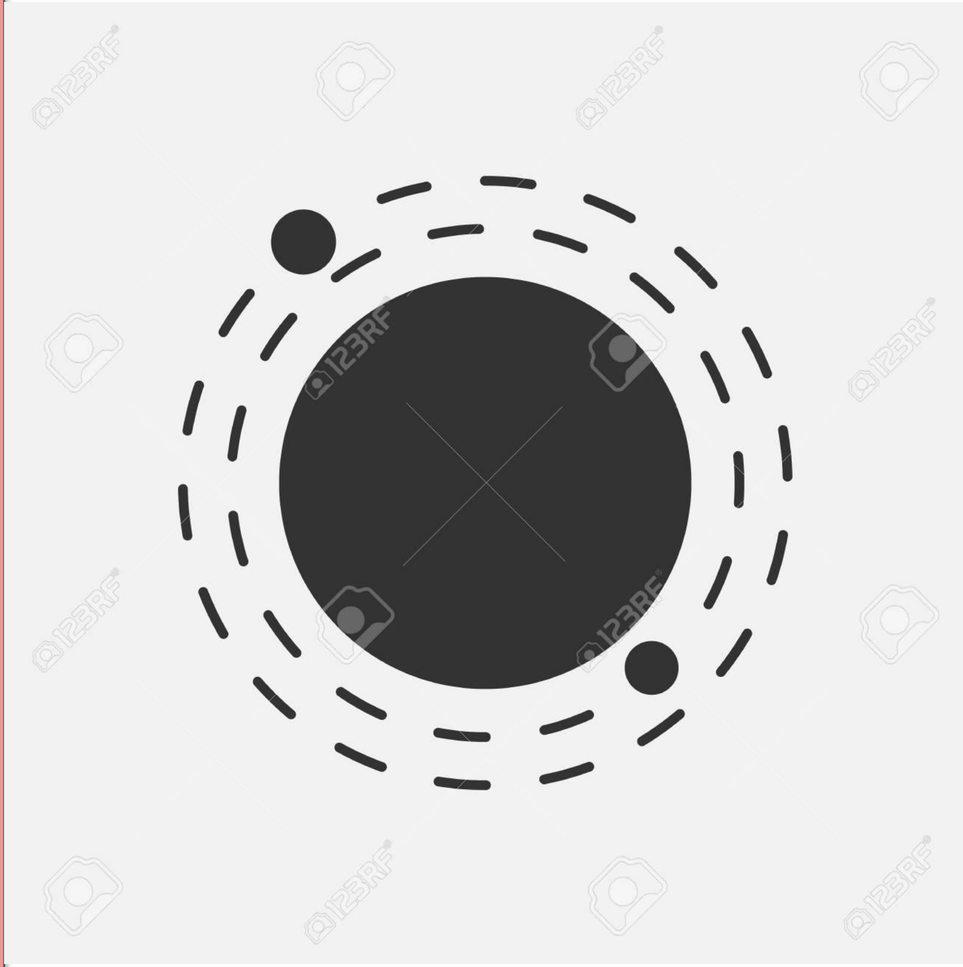
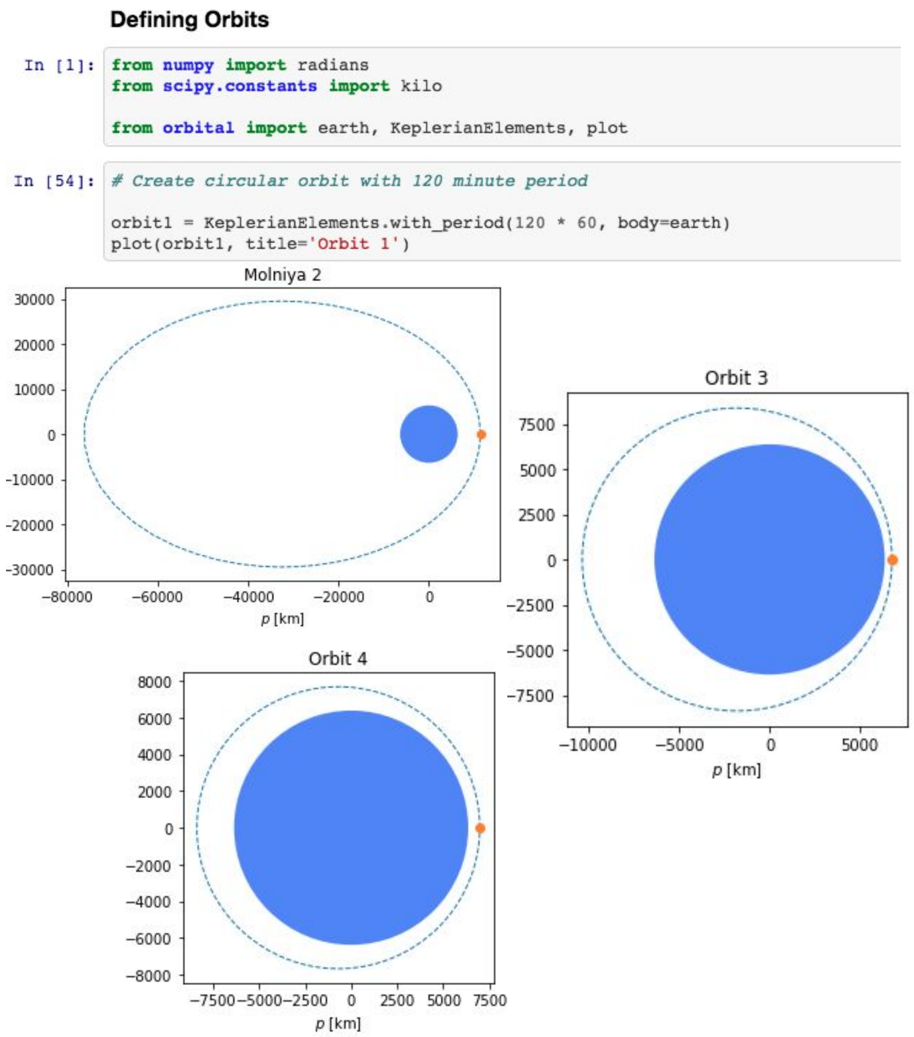
Python **Orbital Simulation**

Bryant Le

OrbitalPy



Object Definition

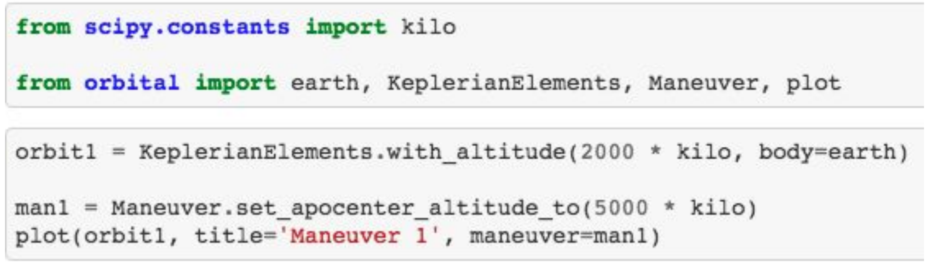


Defining Orbits

```
In [1]:  from numpy import radians
         from scipy.constants import kilo

         from orbital import earth, KeplerianElements, plot
```

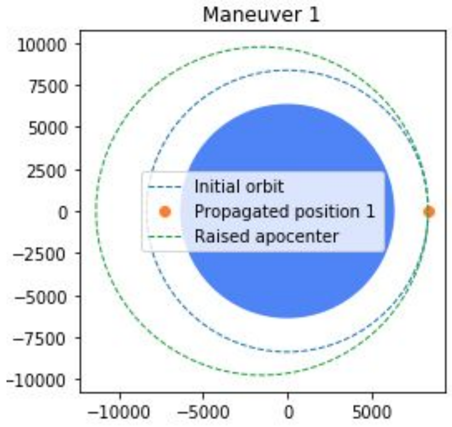
```
In [54]: # Create circular orbit with 120 minute period

         orbit1 = KeplerianElements.with_period(120 * 60, body=earth)
         plot(orbit1, title='Orbit 1')
```

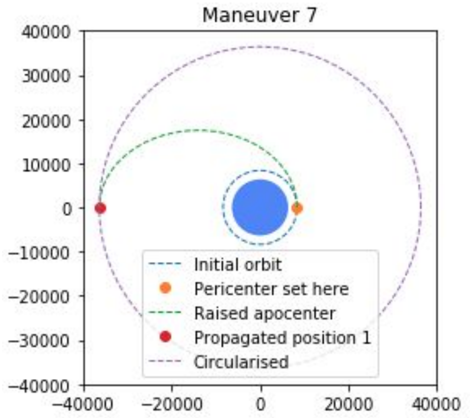
Maneuvers



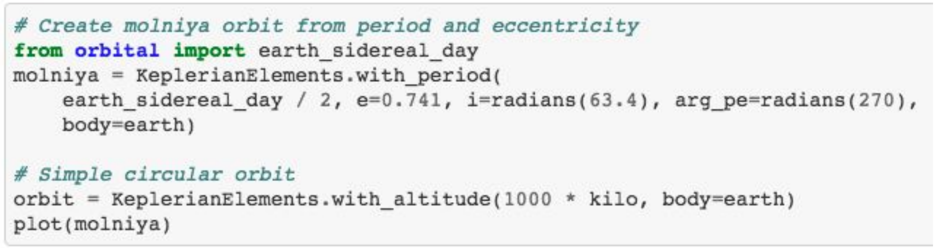
```
from scipy.constants import kilo

from orbital import earth, KeplerianElements, Maneuver, plot
```

```
orbit1 = KeplerianElements.with_altitude(2000 * kilo, body=earth)

man1 = Maneuver.set_apocenter_altitude_to(5000 * kilo)
plot(orbit1, title='Maneuver 1', maneuver=man1)
```

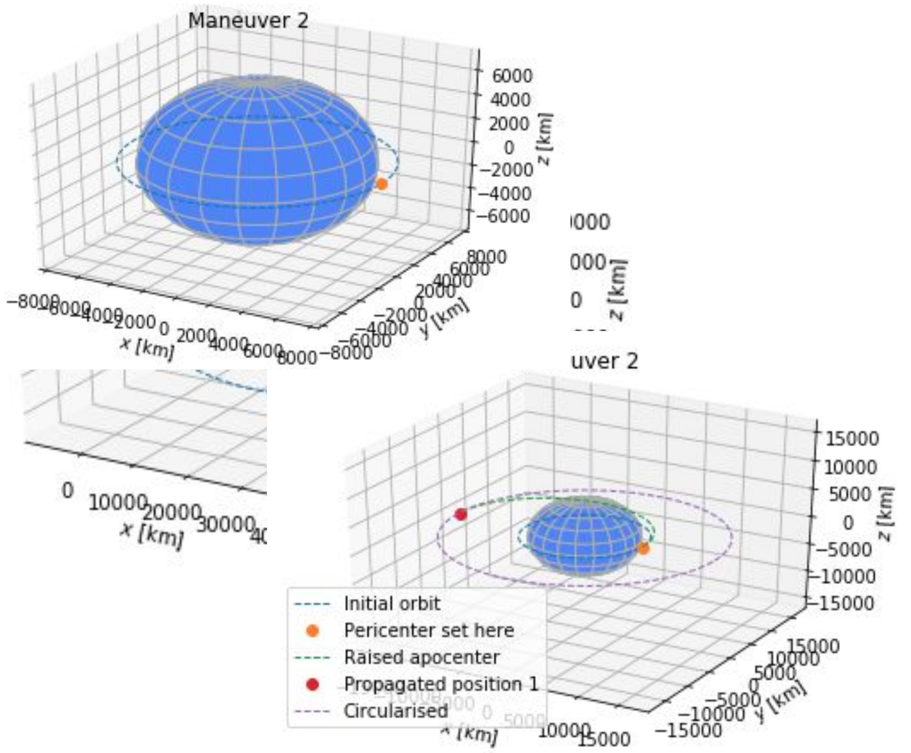




Plotting



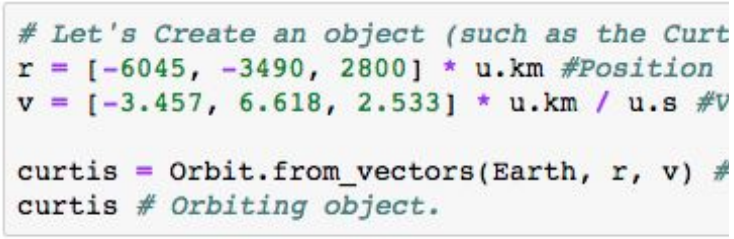
```python
# Create molniya orbit from period and eccentricity
from orbital import earth_sidereal_day
molniya = KeplerianElements.with_period(
    earth_sidereal_day / 2, e=0.741, i=radians(63.4), arg_pe=radians(270),
    body=earth)

# Simple circular orbit
orbit = KeplerianElements.with_altitude(1000 * kilo, body=earth)
plot(molniya)
```



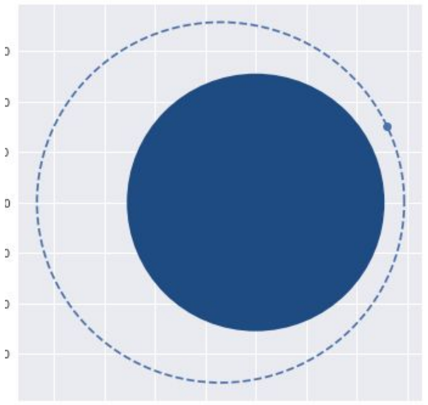
poliastro

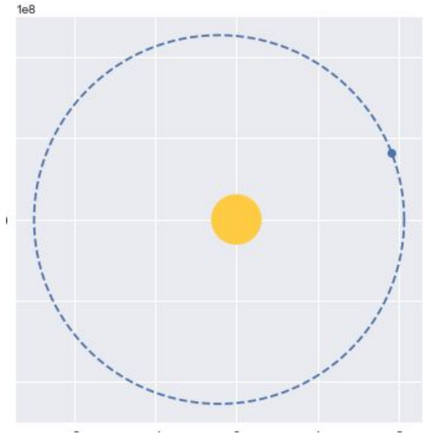
Astrodynamics in Python

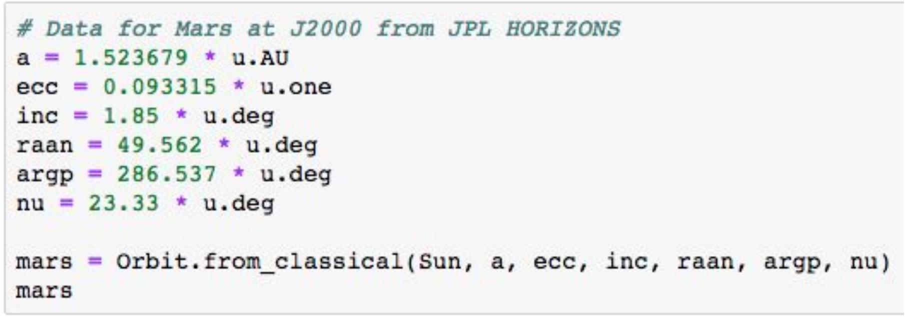
Object Definition



```
# Let's Create an object (such as the Curt
r = [-6045, -3490, 2800] * u.km #Position
v = [-3.457, 6.618, 2.533] * u.km / u.s #V

curtis = Orbit.from_vectors(Earth, r, v) #
curtis # Orbiting object.
```

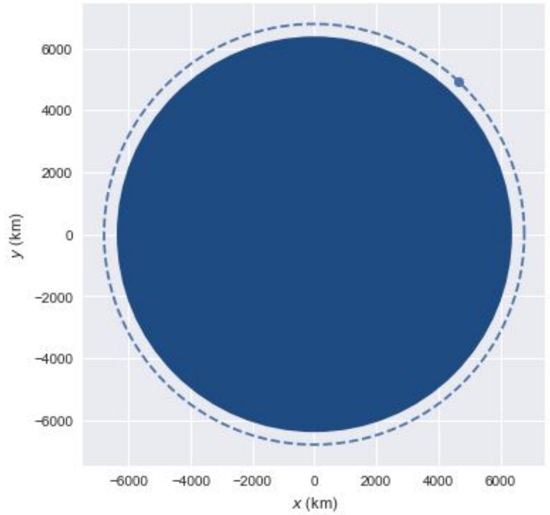


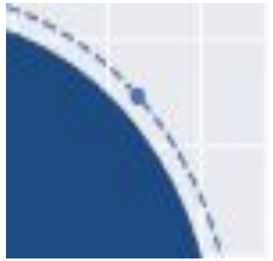


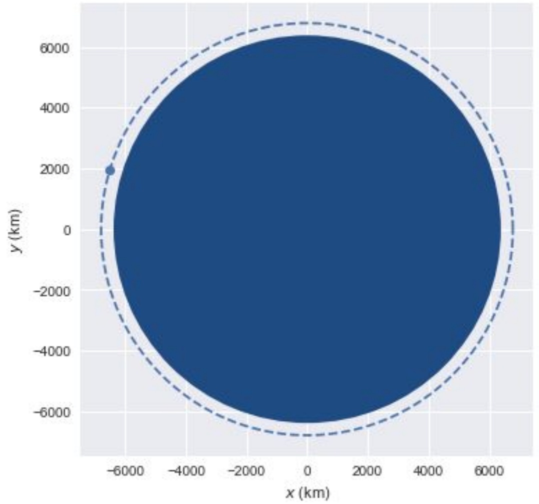


```
# Data for Mars at J2000 from JPL HORIZONS
a = 1.523679 * u.AU
ecc = 0.093315 * u.one
inc = 1.85 * u.deg
raan = 49.562 * u.deg
argp = 286.537 * u.deg
nu = 23.33 * u.deg

mars = Orbit.from_classical(Sun, a, ecc, inc, raan, argp, nu)
mars
```

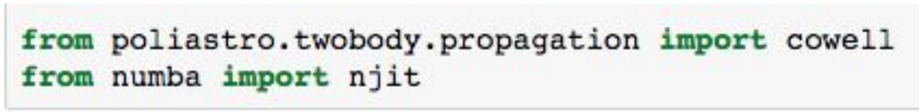
Propagations



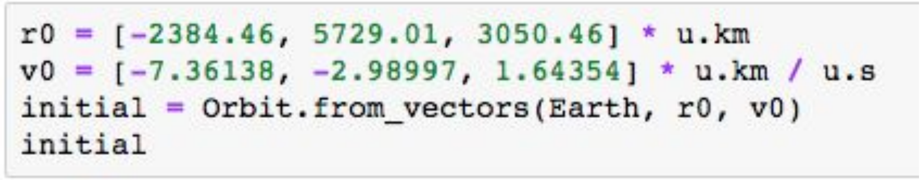




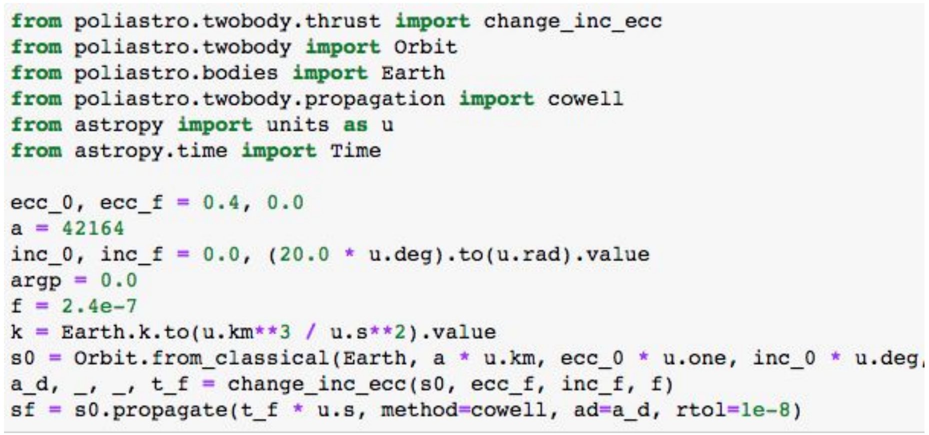
Perturbations (Thrusts)



```python
from poliastro.twobody.propagation import cowell
from numba import njit
```

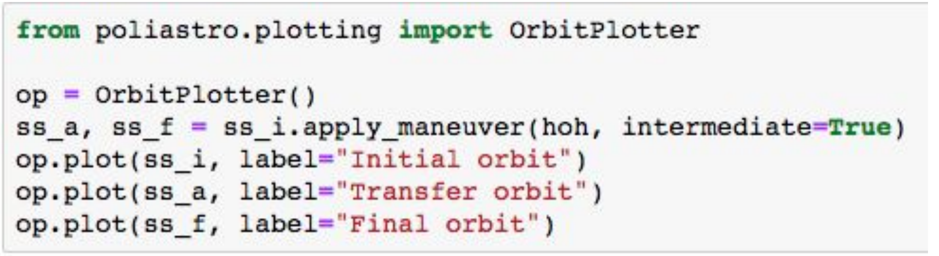


```python
r0 = [-2384.46, 5729.01, 3050.46] * u.km
v0 = [-7.36138, -2.98997, 1.64354] * u.km / u.s
initial = Orbit.from_vectors(Earth, r0, v0)
initial
```

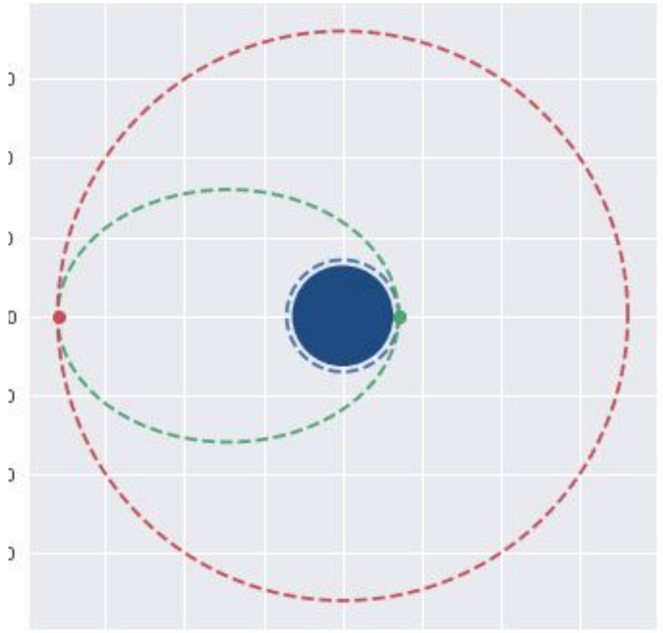


```python
from poliastro.twobody.thrust import change_inc_ecc
from poliastro.twobody import Orbit
from poliastro.bodies import Earth
from poliastro.twobody.propagation import cowell
from astropy import units as u
from astropy.time import Time

ecc_0, ecc_f = 0.4, 0.0
a = 42164
inc_0, inc_f = 0.0, (20.0 * u.deg).to(u.rad).value
argp = 0.0
f = 2.4e-7
k = Earth.k.to(u.km**3 / u.s**2).value
s0 = Orbit.from_classical(Earth, a * u.km, ecc_0 * u.one, inc_0 * u.deg,
a_d, _, _, t_f = change_inc_ecc(s0, ecc_f, inc_f, f)
sf = s0.propagate(t_f * u.s, method=cowell, ad=a_d, rtol=1e-8)
```

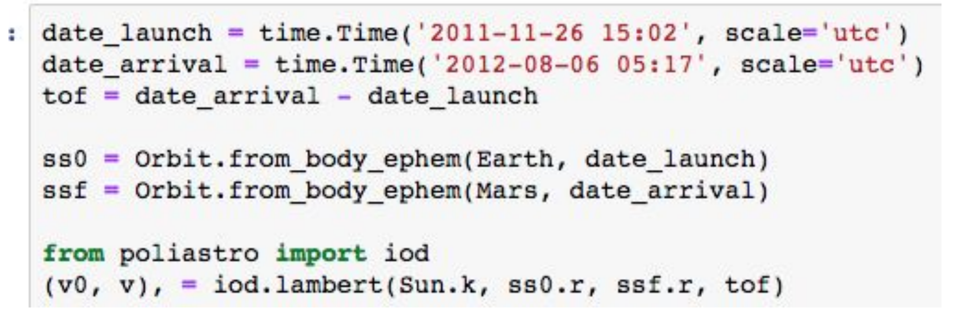
Maneuvering Objects



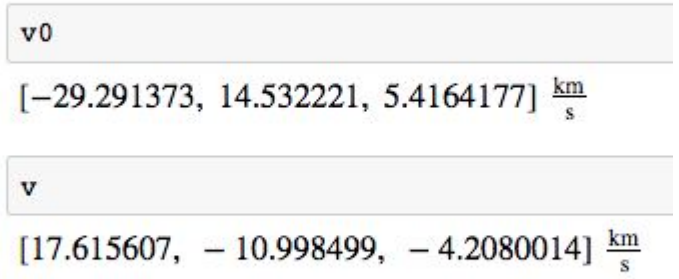
```python
from poliastro.plotting import OrbitPlotter

op = OrbitPlotter()
ss_a, ss_f = ss_i.apply_maneuver(hoh, intermediate=True)
op.plot(ss_i, label="Initial orbit")
op.plot(ss_a, label="Transfer orbit")
op.plot(ss_f, label="Final orbit")
```



Lambert's Problem



```python
date_launch = time.Time('2011-11-26 15:02', scale='utc')
date_arrival = time.Time('2012-08-06 05:17', scale='utc')
tof = date_arrival - date_launch

ss0 = Orbit.from_body_ephem(Earth, date_launch)
ssf = Orbit.from_body_ephem(Mars, date_arrival)

from poliastro import iod
(v0, v), = iod.lambert(Sun.k, ss0.r, ssf.r, tof)
```



```
v0
```

$$[-29.291373,\ 14.532221,\ 5.4164177]\ \frac{\mathrm{km}}{\mathrm{s}}$$

```
v
```

$$[17.615607,\ -10.998499,\ -4.2080014]\ \frac{\mathrm{km}}{\mathrm{s}}$$

Feature Summary

- Analytical and numerical orbit propagation
- Conversion between position and velocity vectors and classical orbital elements
- Coordinate frame transformations
- Hohmann and bielliptic maneuvers computation
- Trajectory plotting
- Initial orbit determination (Lambert problem)
- Planetary ephemerides (using SPICE kernels via Astropy)
- Computation of Near-Earth Objects