ENAE 404 - 0101

Homework 03: Ground Tracks

Due on March 13, 2025 at 09:30 AM

Dr. Barbee, 09:30

Vai Srivastava

March 11, 2025

Problem 1:

Conceptual questions:

1. Give the semi-major axis, eccentricity and inclination of an orbit whose ground track is a point.
2. Explain why argment of periapsis equal to 0° or 180° produces equatorial symmetry.
3. Consider the Molniya orbit. Which one orbital element would you change so that the spacecraft would spend a long time veiewing the Southern hemisphere (rather than the Northern hemisphere)? Identify the orbital element and the value of this orbital element that would preserve the same structure of the ground track (just flipped to observe the Southern hemisphere).

Solution

Problem 2:

Plot one day ground tracks for the following orbits:

Spacecraft ID	a(km)	e	i(°)	Ω(°)	ω(°)
A	42164	0.3	40	0	0
B	26562	0.3	40	0	0
C	42164	0.3	60	0	70
D	42164	0.3	30	0	70
E	42164	0.3	120	0	0

Compare and contrast the ground track for Orbit E to that of Orbit A.

Solution

```python
from astropy import units as u
from poliastro.util import time_range
from poliastro.bodies import Earth
from poliastro.twobody import Orbit
from poliastro.earth import EarthSatellite
from poliastro.earth.plotting import GroundtrackPlotter
```

Listing 1: P02 Python Code

Part A

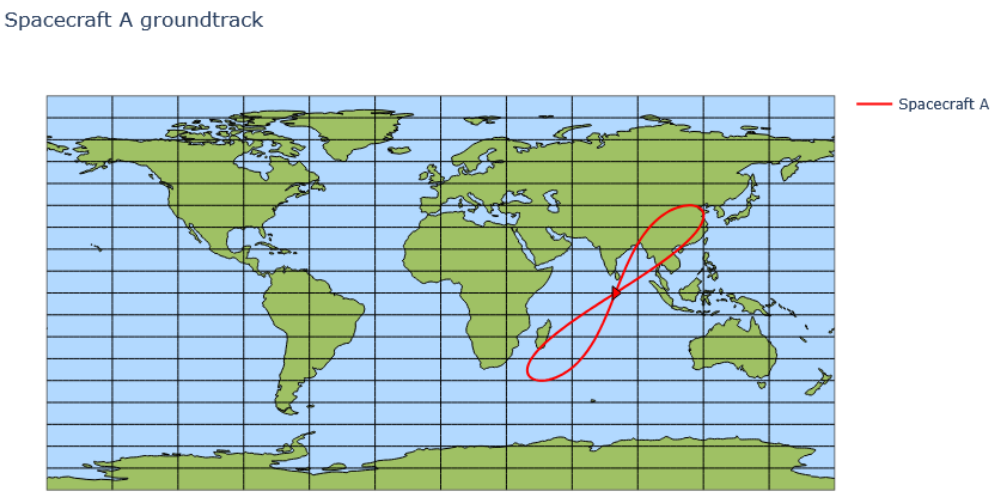

Figure 1: Spacecraft A groundtrack

```python
a = 42164 * u.km          # semi-major axis
ecc = 0.3 * u.one         # eccentricity
inc = 40 * u.deg          # inclination
raan = 0 * u.deg          # right ascension of the ascending node
argp = 0 * u.deg          # argument of perigee
nu = 0 * u.deg            # true anomaly (set to 0 for the initial state)
```

```python

Orbit_A = Orbit.from_classical(Earth, a, ecc, inc, raan, argp, nu)
Spacecraft_A = EarthSatellite(Orbit_A, None)
t_span = time_range(start=Orbit_A.epoch, end=Orbit_A.epoch + 24.0 * u.h, num_values=150)

gp = GroundtrackPlotter()
gp.update_layout(title="Spacecraft A groundtrack")

gp.plot(
    Spacecraft_A,
    t_span,
    label="Spacecraft A",
    color="red",
    marker={
        "size": 10,
        "symbol": "triangle-right",
        "line": {"width": 1, "color": "black"},
    },
)

# gp.update_geos(projection_type="orthographic")
gp.fig.show()
```

Listing 2: P02a Python Code

Part B

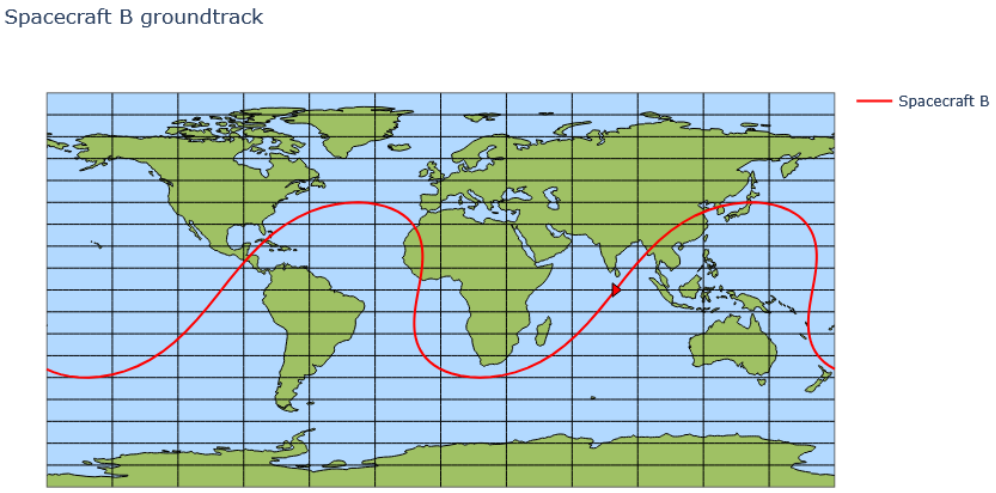

Figure 2: Spacecraft B groundtrack

```python
a = 26562 * u.km         # semi-major axis
ecc = 0.3 * u.one        # eccentricity
inc = 40 * u.deg         # inclination
raan = 0 * u.deg         # right ascension of the ascending node
argp = 0 * u.deg         # argument of perigee
nu = 0 * u.deg           # true anomaly (set to 0 for the initial state)

```

```python
Orbit_B = Orbit.from_classical(Earth, a, ecc, inc, raan, argp, nu)
Spacecraft_B = EarthSatellite(Orbit_B, None)
t_span = time_range(start=Orbit_B.epoch, end=Orbit_B.epoch + 24.0 * u.h, num_values=150)

gp = GroundtrackPlotter()
gp.update_layout(title="Spacecraft B groundtrack")

gp.plot(
    Spacecraft_B,
    t_span,
    label="Spacecraft B",
    color="red",
    marker={
        "size": 10,
        "symbol": "triangle-right",
        "line": {"width": 1, "color": "black"},
    },
)

# gp.update_geos(projection_type="orthographic")
gp.fig.show()
```

Listing 3: P02b Python Code

Part C

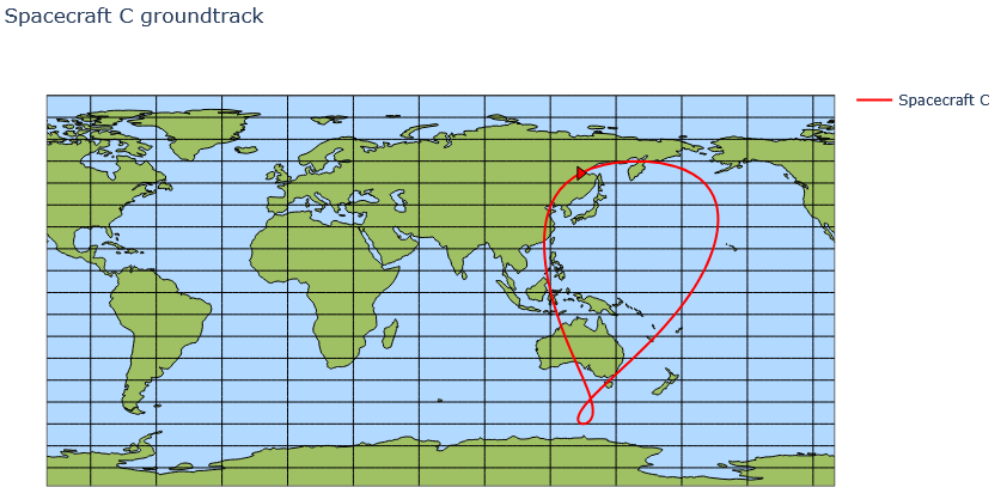

Figure 3: Spacecraft C groundtrack

```python
a = 42164 * u.km         # semi-major axis
ecc = 0.3 * u.one        # eccentricity
inc = 60 * u.deg         # inclination
raan = 0 * u.deg         # right ascension of the ascending node
argp = 70 * u.deg         # argument of perigee
nu = 0 * u.deg           # true anomaly (set to 0 for the initial state)

Orbit_C = Orbit.from_classical(Earth, a, ecc, inc, raan, argp, nu)
```

```python
Spacecraft_C = EarthSatellite(Orbit_C, None)
t_span = time_range(start=Orbit_C.epoch, end=Orbit_C.epoch + 24.0 * u.h, num_values=150)

gp = GroundtrackPlotter()
gp.update_layout(title="Spacecraft C groundtrack")

gp.plot(
    Spacecraft_C,
    t_span,
    label="Spacecraft C",
    color="red",
    marker={
        "size": 10,
        "symbol": "triangle-right",
        "line": {"width": 1, "color": "black"},
    },
)

# gp.update_geos(projection_type="orthographic")
gp.fig.show()
```

Listing 4: P02c Python Code

Part D

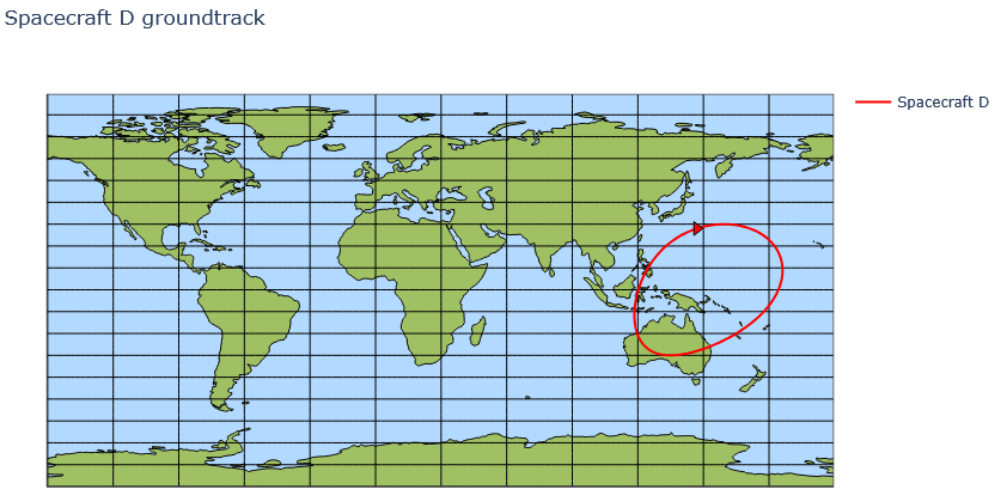

Figure 4: Spacecraft D groundtrack

```python
a = 42164 * u.km         # semi-major axis
ecc = 0.3 * u.one        # eccentricity
inc = 30 * u.deg         # inclination
raan = 0 * u.deg         # right ascension of the ascending node
argp = 70 * u.deg         # argument of perigee
nu = 0 * u.deg           # true anomaly (set to 0 for the initial state)

Orbit_D = Orbit.from_classical(Earth, a, ecc, inc, raan, argp, nu)
Spacecraft_D = EarthSatellite(Orbit_D, None)
```

```python
t_span = time_range(start=Orbit_D.epoch, end=Orbit_D.epoch + 24.0 * u.h, num_values=150)

gp = GroundtrackPlotter()
gp.update_layout(title="Spacecraft D groundtrack")

gp.plot(
    Spacecraft_D,
    t_span,
    label="Spacecraft D",
    color="red",
    marker={
        "size": 10,
        "symbol": "triangle-right",
        "line": {"width": 1, "color": "black"},
    },
)

# gp.update_geos(projection_type="orthographic")
gp.fig.show()
```

Listing 5: P02d Python Code

Part E

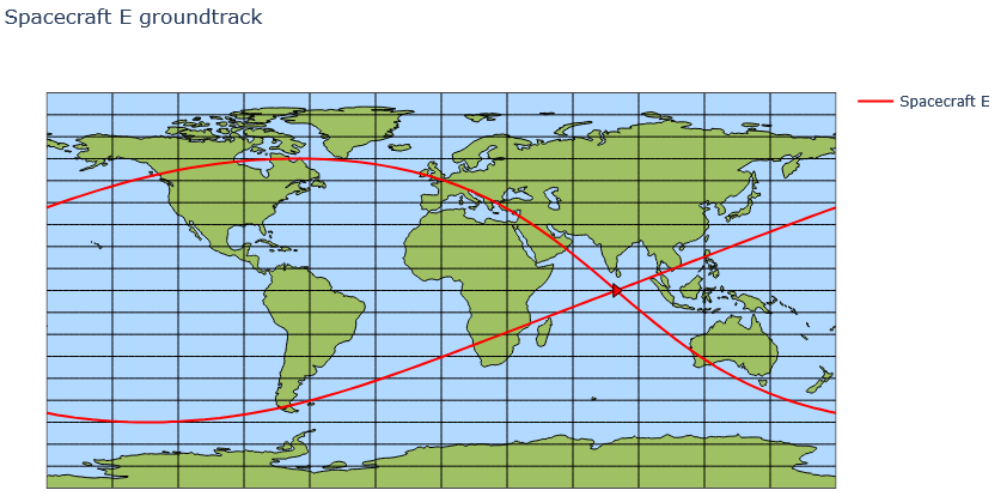

Figure 5: Spacecraft E groundtrack

```python
a = 42164 * u.km         # semi-major axis
ecc = 0.3 * u.one        # eccentricity
inc = 120 * u.deg         # inclination
raan = 0 * u.deg         # right ascension of the ascending node
argp = 0 * u.deg         # argument of perigee
nu = 0 * u.deg           # true anomaly (set to 0 for the initial state)

Orbit_E = Orbit.from_classical(Earth, a, ecc, inc, raan, argp, nu)
Spacecraft_E = EarthSatellite(Orbit_E, None)
t_span = time_range(start=Orbit_E.epoch, end=Orbit_E.epoch + 24.0 * u.h, num_values=150)
```

```
11
12 gp = GroundtrackPlotter()
13 gp.update_layout(title="Spacecraft E groundtrack")
14
15 gp.plot(
16     Spacecraft_E,
17     t_span,
18     label="Spacecraft E",
19     color="red",
20     marker={
21         "size": 10,
22         "symbol": "triangle-right",
23         "line": {"width": 1, "color": "black"},
24     },
25 )
26
27 # gp.update_geos(projection_type="orthographic")
28 gp.fig.show()
```

Listing 6: P02e Python Code

Problem 3:

Calculate the ΔV required to execute a Hohmann transfer from a circular orbit with radius 14×10^3 km to a circular orbit with radius 8×10^3 km. Assume the central body is the Earth. State whether the maneuvers increase or decrease the spacecraft's velocity.

Solution