QLUSTER

Spacecraft Formation Flying Visualisation Tool in Python

In the Spirit of the Open Source Cube-Sat Workshop (OSCW), 9 December 2021

Why Qluster?

- Many softwares available to design and propagate orbits… none that can directly design relative orbits!
- Instead of …

1. Start with the orbit elements
2. Propagate + Hill frame transformation into RIC coordinates
3. Visualize the resulting formation geometry!

- Why not:

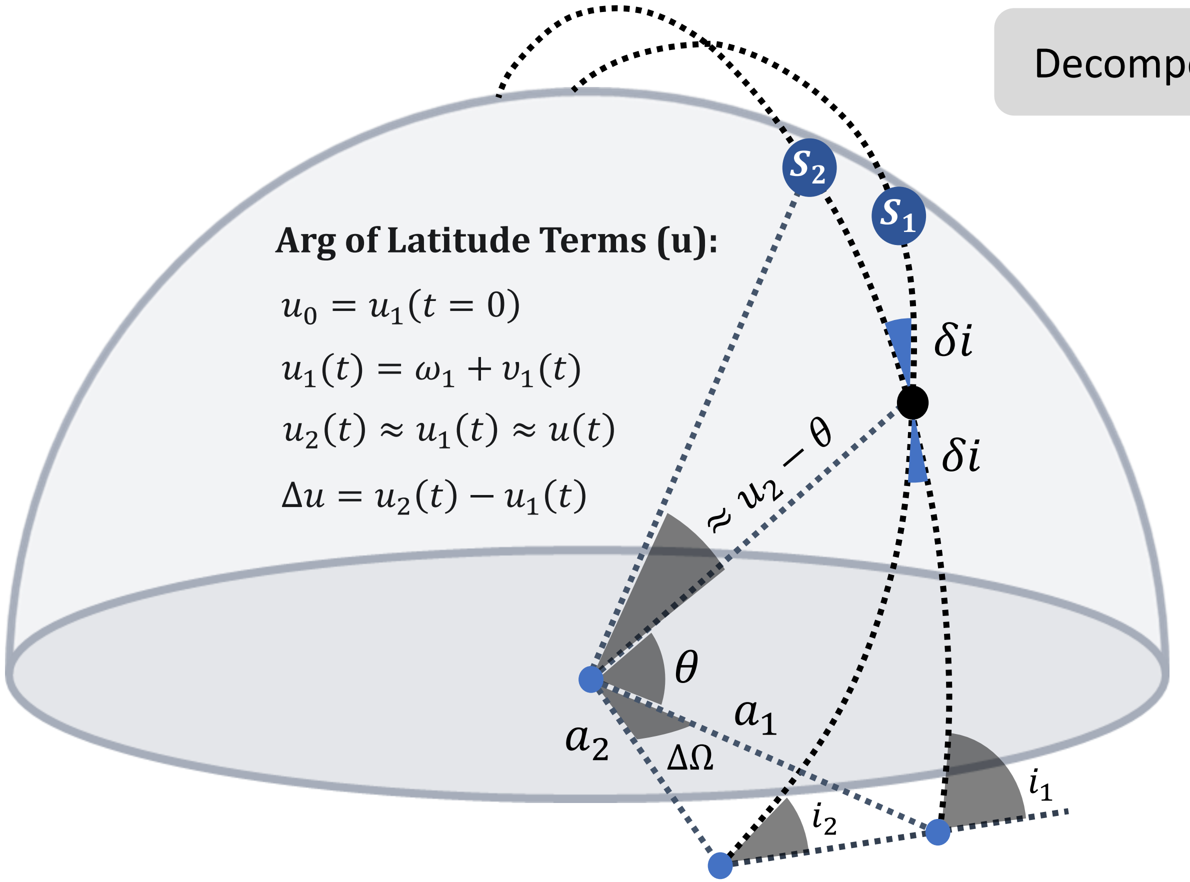
1. Visualize the resulting formation geometry!
2. Invert the Hill-Frame (or ROE frame) coordinates
3. Return the orbit elements!

Quick Math: Relative Orbital Element (ROE) Space

Formation design via classical method:
{ *a, e, i, ω, Ω, ν* } ➔ Not intuitive!

Can we re-design our future distributed satellite missions using **Hill Frame coordinates**, by linearizing the Hill-Clohessy-Wiltshire equations?

Decomposition into: **Inclination vector** and **eccentricity vector** separation.



$$\overrightarrow{\Delta e} \equiv \begin{Bmatrix} \Delta e_x \\ \Delta e_y \end{Bmatrix} = \begin{Bmatrix} e_2 \cos\omega_2 - e_1 \cos\omega_1 \\ e_2 \sin\omega_2 - e_1 \sin\omega_1 \end{Bmatrix} \approx \delta e \begin{Bmatrix} \cos\varphi \\ \sin\varphi \end{Bmatrix}$$

$$\overrightarrow{\Delta i} \equiv \begin{Bmatrix} \Delta i_x \\ \Delta i_y \end{Bmatrix} = \sin\delta i \begin{Bmatrix} \cos\theta \\ \sin\theta \end{Bmatrix} \approx \begin{Bmatrix} \Delta i \\ \Delta\Omega \sin i \end{Bmatrix}$$

Use standard orbital elements notation, with subscript 1 ➔ chief satellite, 2 ➔ deputy satellite, linearized about the chief elements.

a → *Semi-major axis*
e → *Eccentricity*
i → *Inclination*
ω → *Argument of Perigee*
Ω → *Right Ascension*
ν → *True Anomaly*

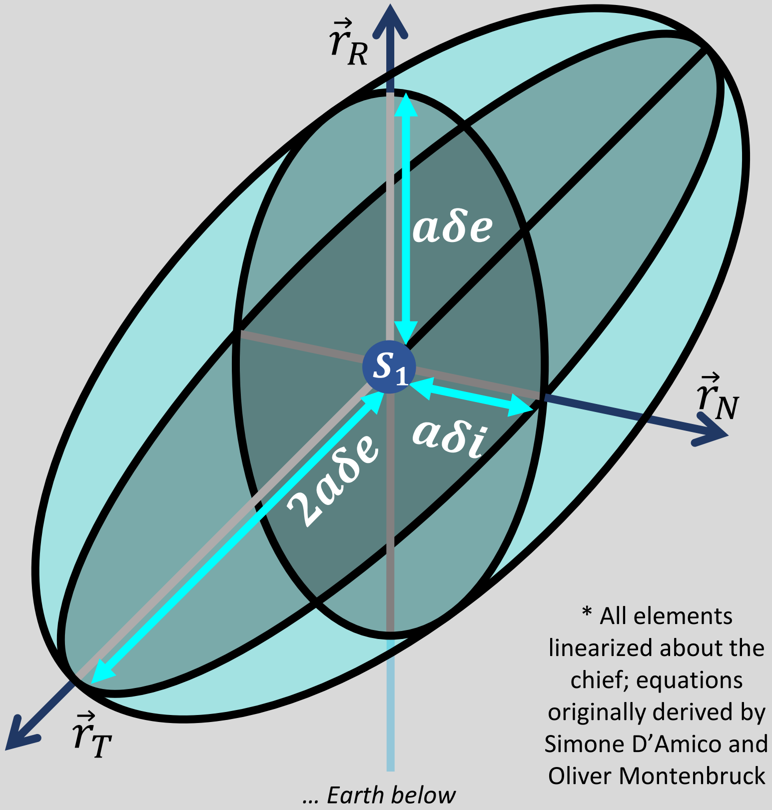
Quick Math: Relative Orbital Element (ROE) Space

Formation design via classical method: { *a, e, i, ω, Ω, ν* } ➔ **Not intuitive!**

Can we re-design our future distributed satellite missions using **Hill Frame coordinates**, by linearizing the Hill-Clohessy-Wiltshire equations?

Decomposition into: **Inclination vector** and **eccentricity vector** separation.

Instead of designing for orbits, can we just specify the radial, in-track, and cross-track variations, as well as the relative phasing between the eccentricity and inclination vectors, to get the orbital elements we need?



* All elements linearized about the chief; equations originally derived by Simone D'Amico and Oliver Montenbruck

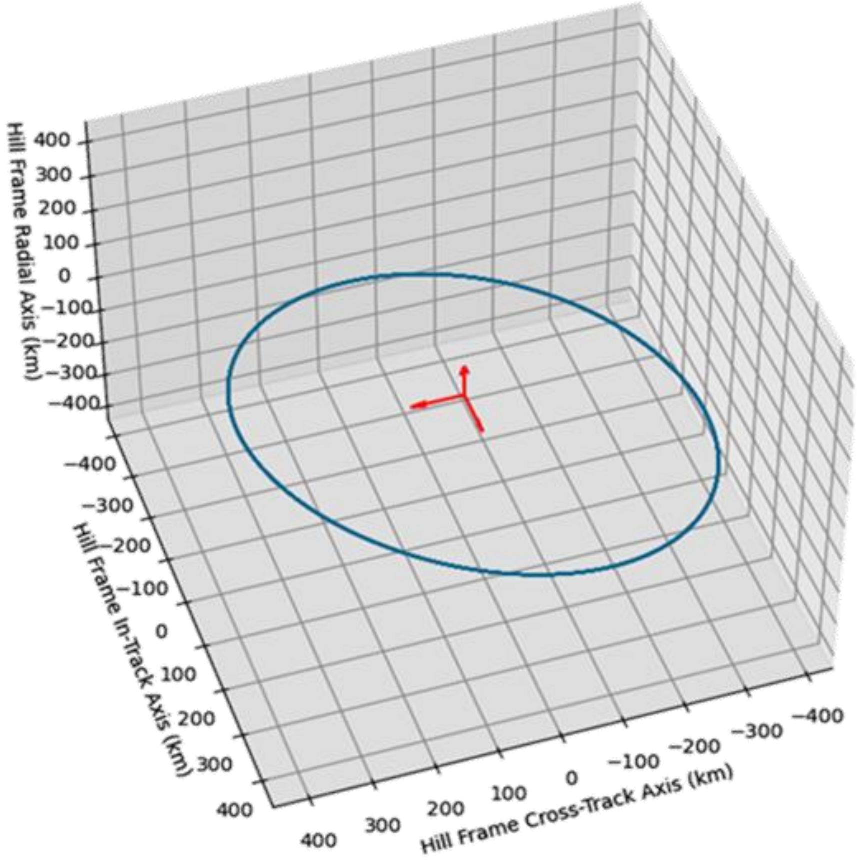
$$\begin{bmatrix} \Delta r_R/a \\ \Delta r_T/a \\ \Delta r_N/a \\ \Delta \dot{r}_R/v \\ \Delta \dot{r}_T/v \\ \Delta \dot{r}_N/v \end{bmatrix} = \begin{bmatrix} \Delta a/a & 0 & -\Delta e_x & -\Delta e_y \\ \Delta u + \Delta\Omega \cos i & -3\Delta a/2a & -2\Delta e_y & +2\Delta e_x \\ 0 & 0 & -\Delta i_y & +\Delta i_x \\ 0 & 0 & -\Delta e_y & +\Delta e_x \\ -3\Delta a/2a & 0 & +2\Delta e_x & +2\Delta e_y \\ 0 & 0 & +\Delta i_x & +\Delta i_y \end{bmatrix} \times \begin{bmatrix} 1 \\ u - u_0 \\ \cos u \\ \sin u \end{bmatrix}$$

... of course!

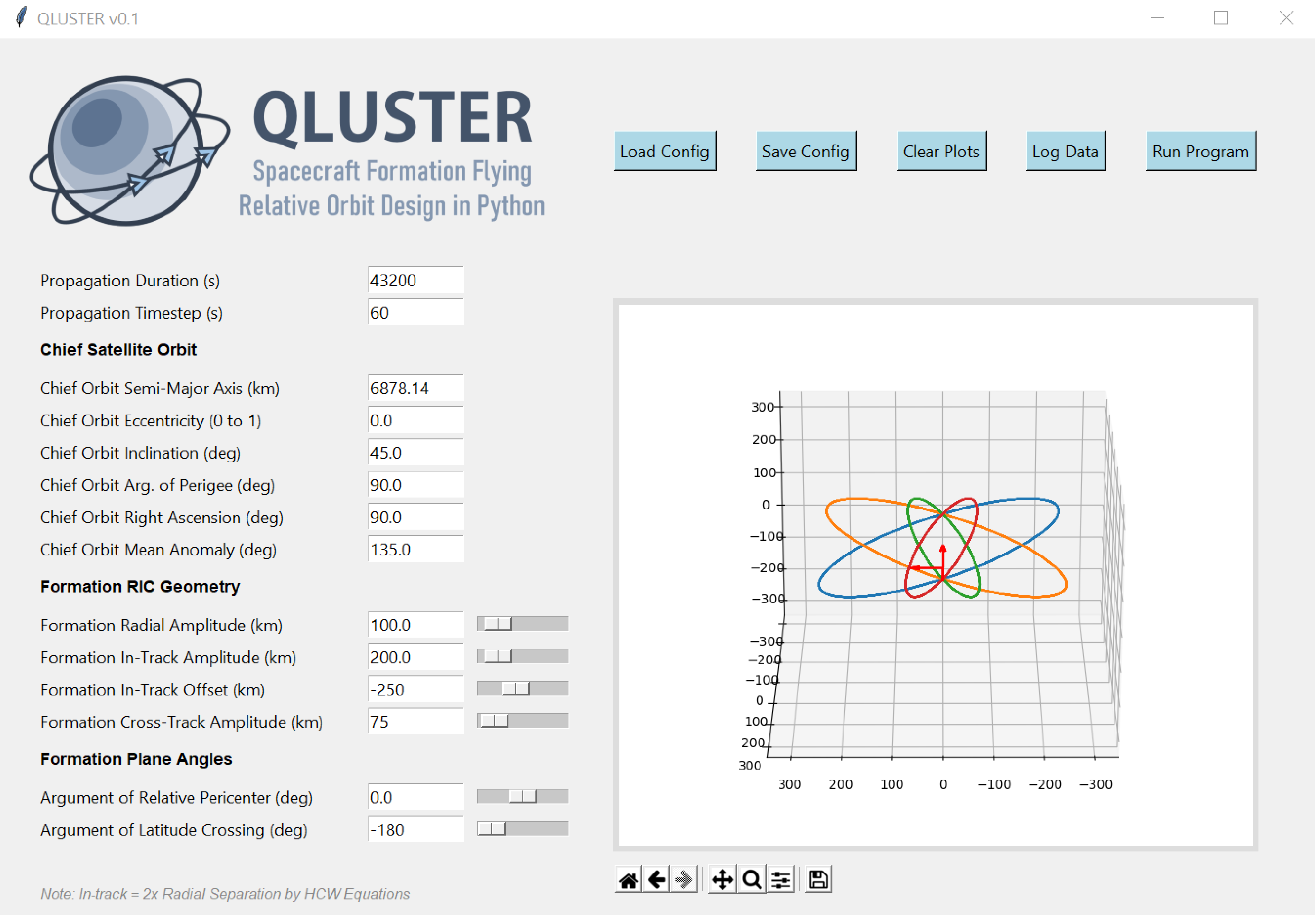
Some Beginnings…

- Version 1: In PoliAstro!

(Thanks to Juan Luis and the Poliastro contributor team!)




```python
import RelativeOrb
from poliastro.bodies import Earth
from poliastro.twobody import Orbit

# 1. Initialize an example Satellite A as the chief spacecraft.
satC = Orbit.from_classical( attractor = Earth,
                             a         = 6918.140 * u.km,
                             ecc       = 1e-6     * u.one,
                             inc       = 63.4     * u.deg,
                             raan      = 70.0     * u.deg,
                             argp      = 90.0     * u.deg,
                             nu        = 1.65     * u.deg)

# 2. Initialize an example Satellite B as the deputy spacecraft.
satD = Orbit.from_classical( attractor = Earth,
                             a         = 6918.140 * u.km,
                             ecc       = 0.012    * u.one,
                             inc       = 65.8     * u.deg,
                             raan      = 72.35    * u.deg,
                             argp      = 135.0    * u.deg,
                             nu        = -46.5725 * u.deg)

# 3. Instantiate the relative orbits object.
relativeSat = RelativeOrb( satC, satD )

# 4. Propagate the relative orbit
relativeSat.propagate()

# 5. Plot the relative trajectory in the chief VVLH Frame.
relativeSat.plot()
```

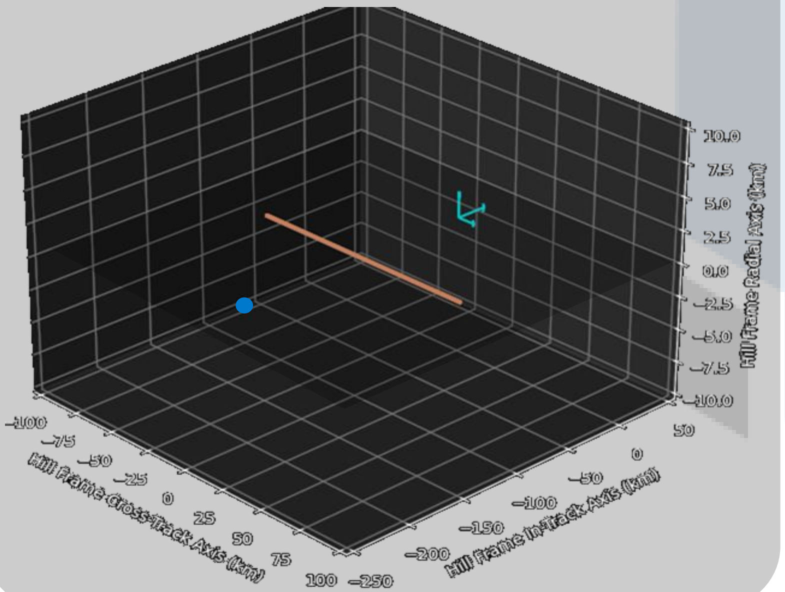

QLUSTER v0.1
QLUSTER
Spacecraft Formation Flying
Relative Orbit Design in Python
Load Config
Save Config
Clear Plots
Log Data
Run Program
Propagation Duration (s)
43200
Propagation Timestep (s)
60
Chief Satellite Orbit
Chief Orbit Semi-Major Axis (km)
6878.14
Chief Orbit Eccentricity (0 to 1)
0.0
Chief Orbit Inclination (deg)
45.0
Chief Orbit Arg. of Perigee (deg)
90.0
Chief Orbit Right Ascension (deg)
90.0
Chief Orbit Mean Anomaly (deg)
135.0
Formation RIC Geometry
Formation Radial Amplitude (km)
100.0
Formation In-Track Amplitude (km)
200.0
Formation In-Track Offset (km)
-250
Formation Cross-Track Amplitude (km)
75
Formation Plane Angles
Argument of Relative Pericenter (deg)
0.0
Argument of Latitude Crossing (deg)
-180
Note: In-track = 2x Radial Separation by HCW Equations
300
200
100
0
−100
−200
−300
−300
−200
−100
0
100
200
300
300
200
100
0
−100
−200
−300

What has QLUSTER been used for?

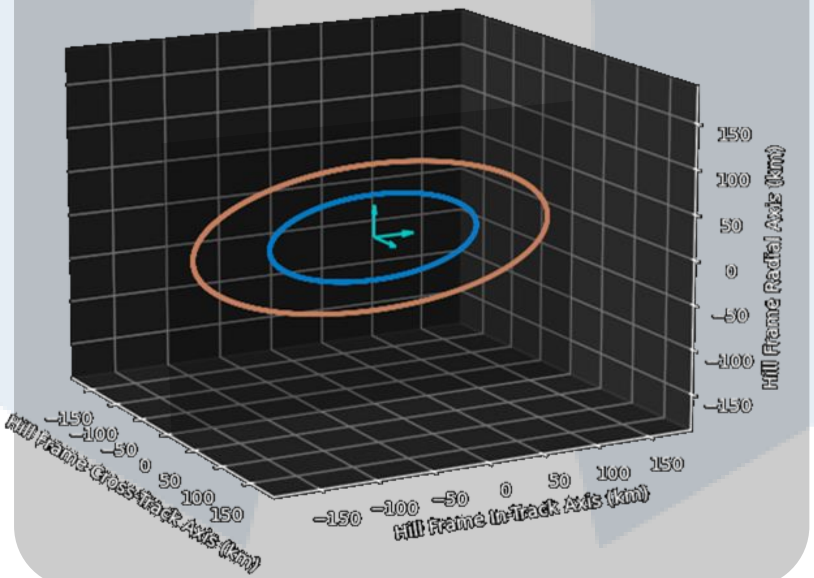
QLUSTER
Spacecraft Formation Flying
Relative Orbit Design in Python

Generating initial conditions for future formation flying mission concept designs…

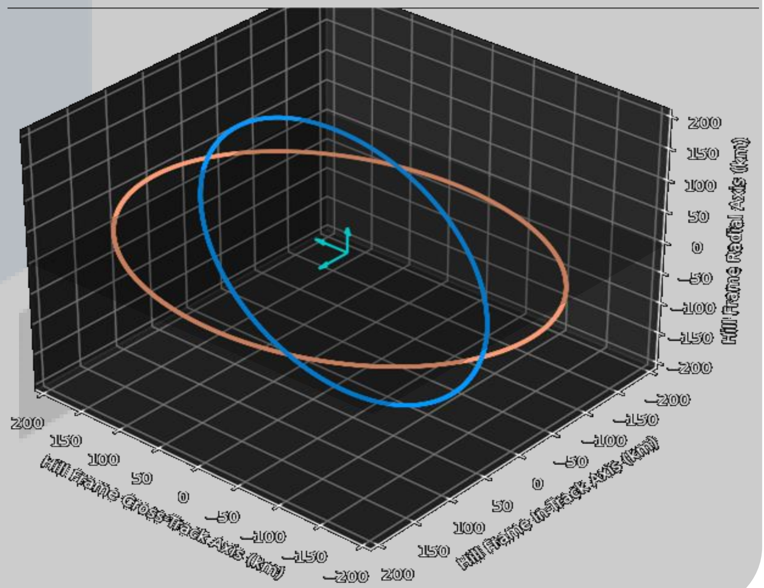
Inclination vector separation → pendulum formation. Applicable for ground emitter geo-location, although geometry is not persistently maximised.



Eccentricity vector separation → helix formation. Applicable for radar interferometry, or rendezvous proximity operations.



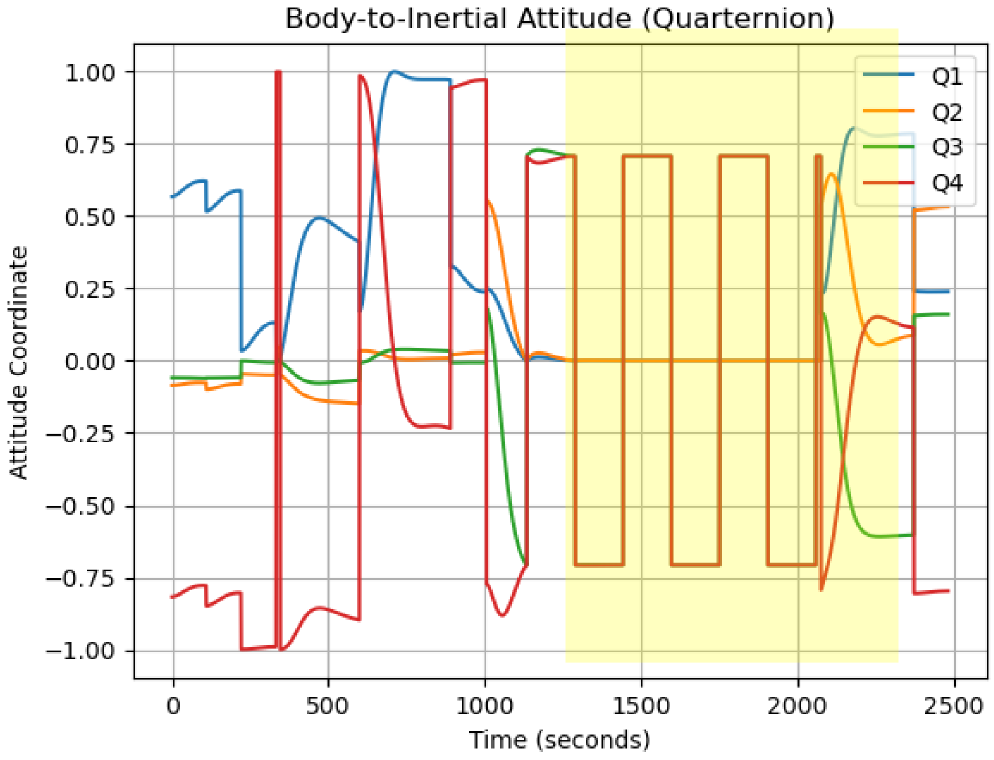
Both inclination and eccentricity vector separation → projected circular orbit formation. Applicable for navigation and geo-location.

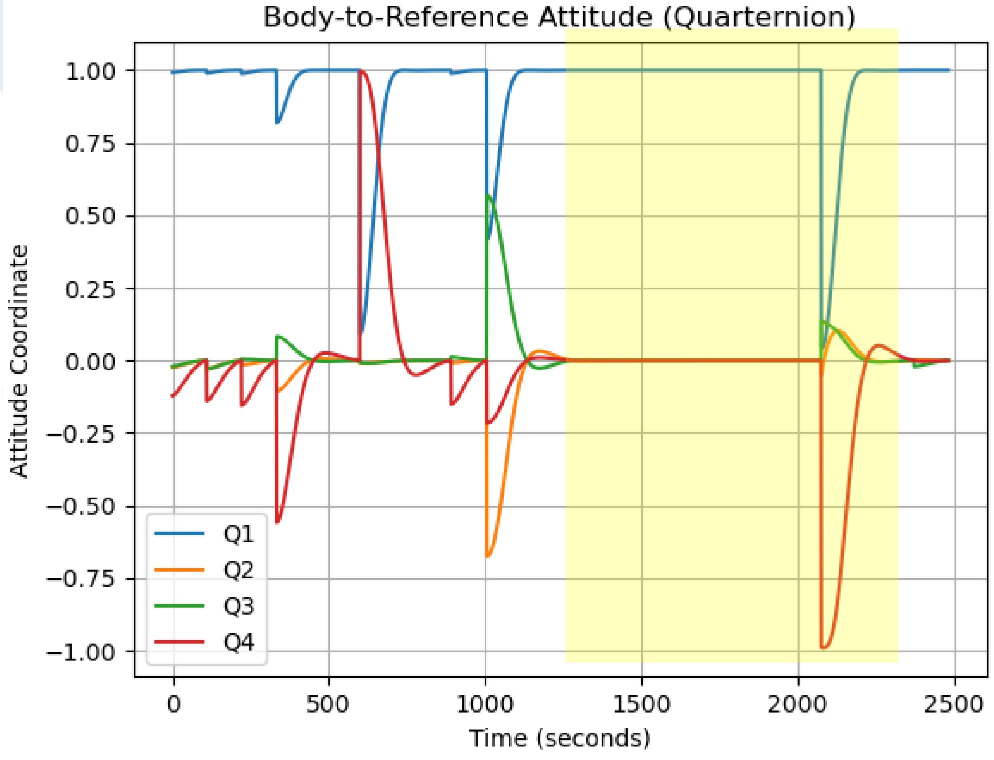


What has QLUSTER been used for?

QLUSTER
Spacecraft Formation Flying
Relative Orbit Design in Python

Attitude control experiments in different formation flying configurations (work in progress)…

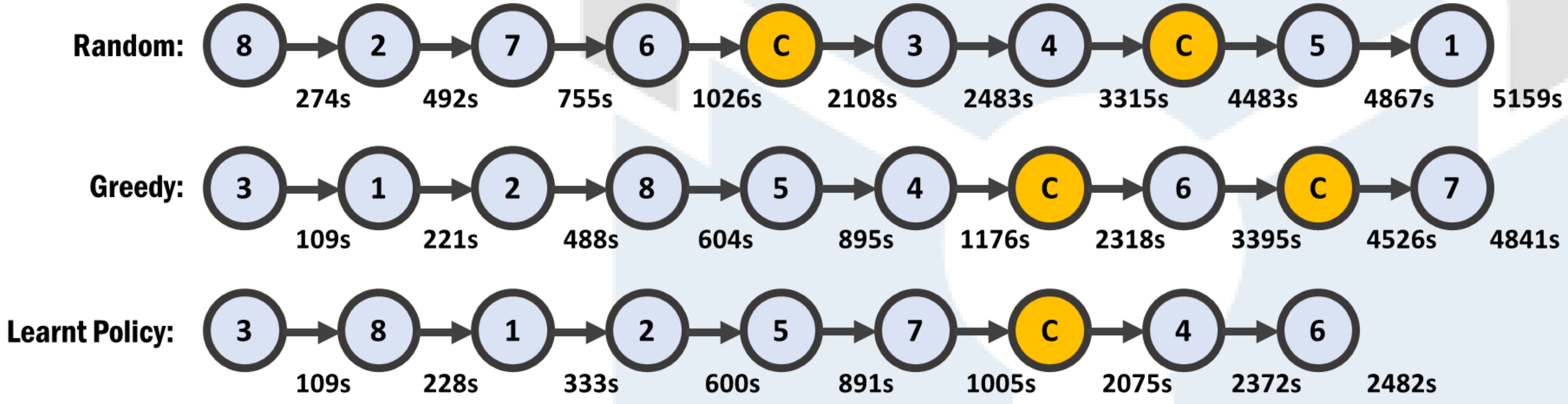


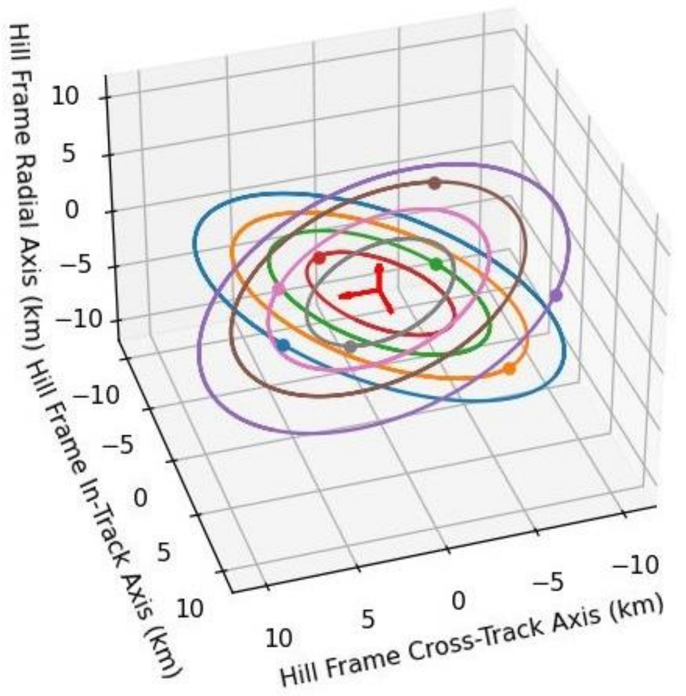


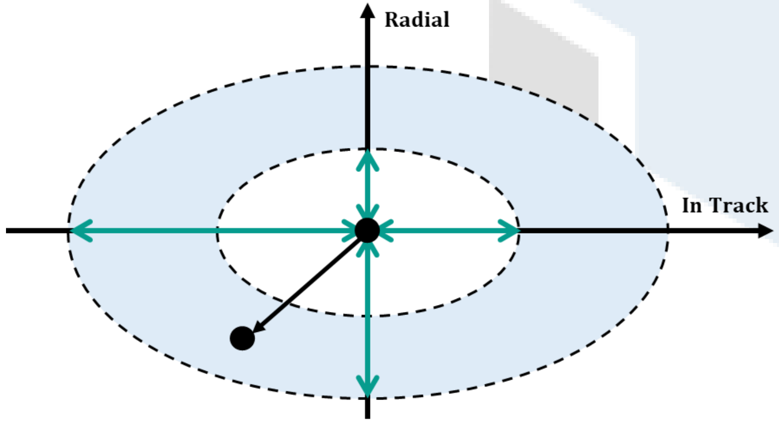
What has QLUSTER been used for?

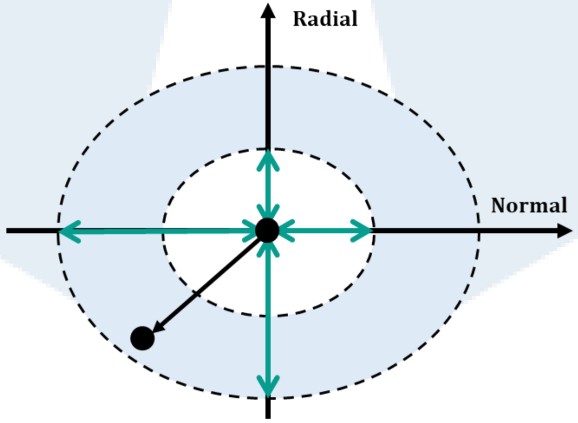
QLUSTER
Spacecraft Formation Flying
Relative Orbit Design in Python

Machine learning experiments where thousands of formation flying configurations can be iterated fast...









Future Work!

- Qluster will be a central part of the **ORQestra** Formation Flying Library!

QLUSTER
Spacecraft Formation Flying
Relative Orbit Design in Python

Future Features:

- High Precision Numerical Propagator (Geopotentials, Third Body, Drag).
- Common classes and objects that can be easily integrated into all the ORQestra libraries.
- Animated plotting + more logging features.
- Any suggestions and feedback are welcome!

Future Work!

- Qluster will be a central part of the **OrQestra** Formation Flying Library!

QLUSTER
Spacecraft Formation Flying
Relative Orbit Design in Python

QOMPASS
Spacecraft Formation Flying
GNSS Relative Navigation in Python

QUADRANT
Spacecraft Formation Flying
Attitude Control in Python

QONTROL
Spacecraft Formation Flying
Relative Orbit Control in Python