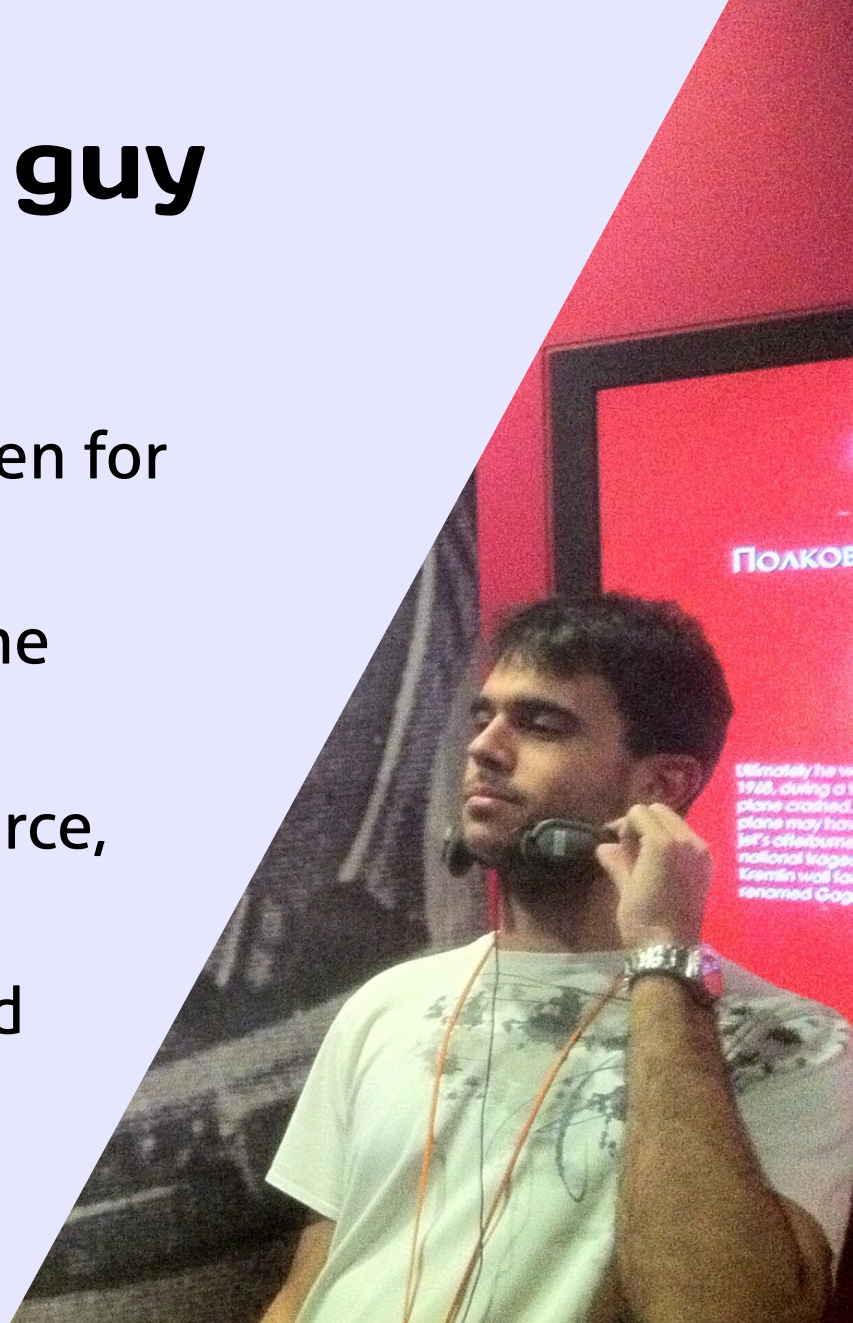
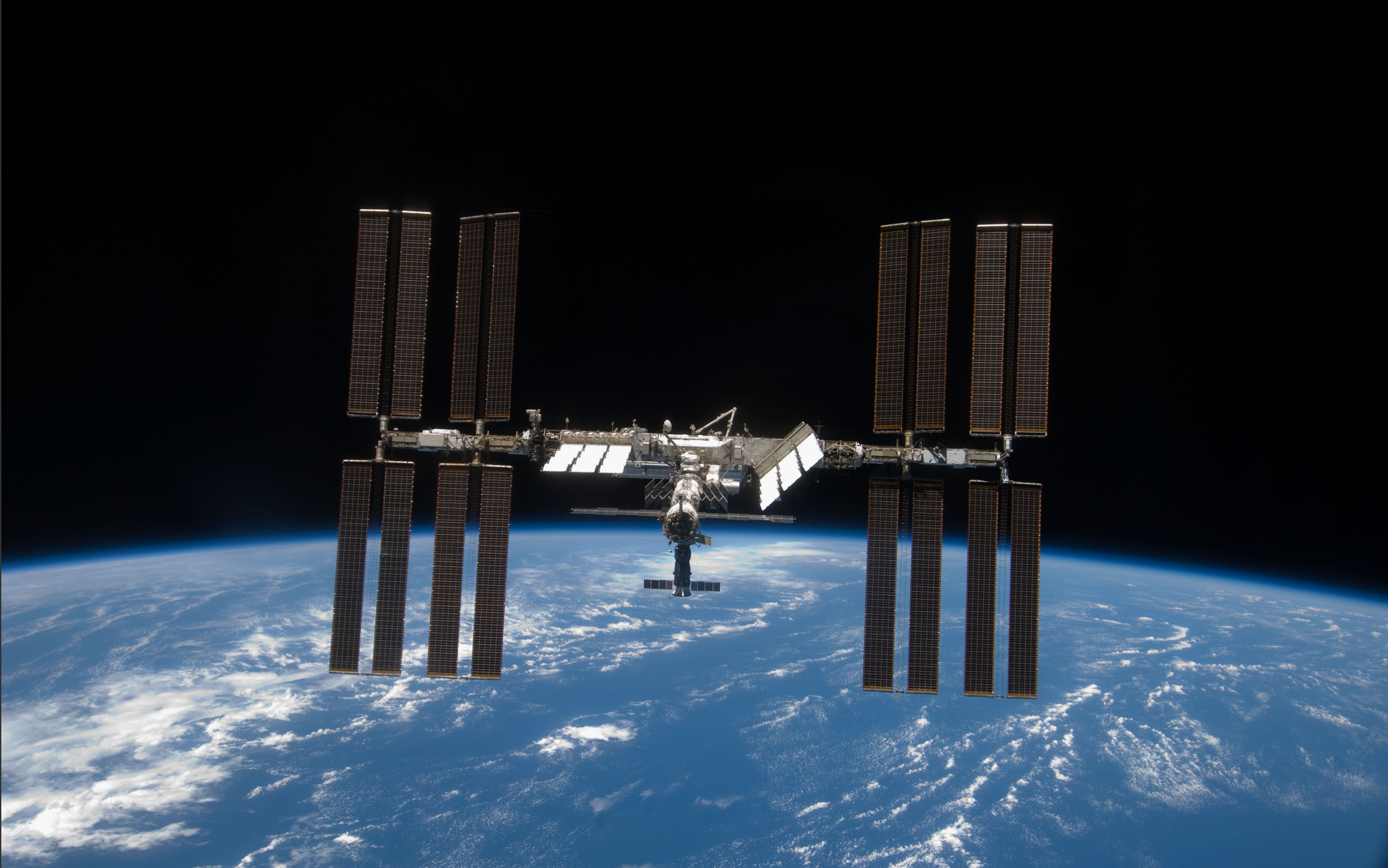
Per Python ad *A*stra

Juan Luis Cano @astrojuanlu
EuroPython @Bilbao – 2016-07-20

Who is this guy

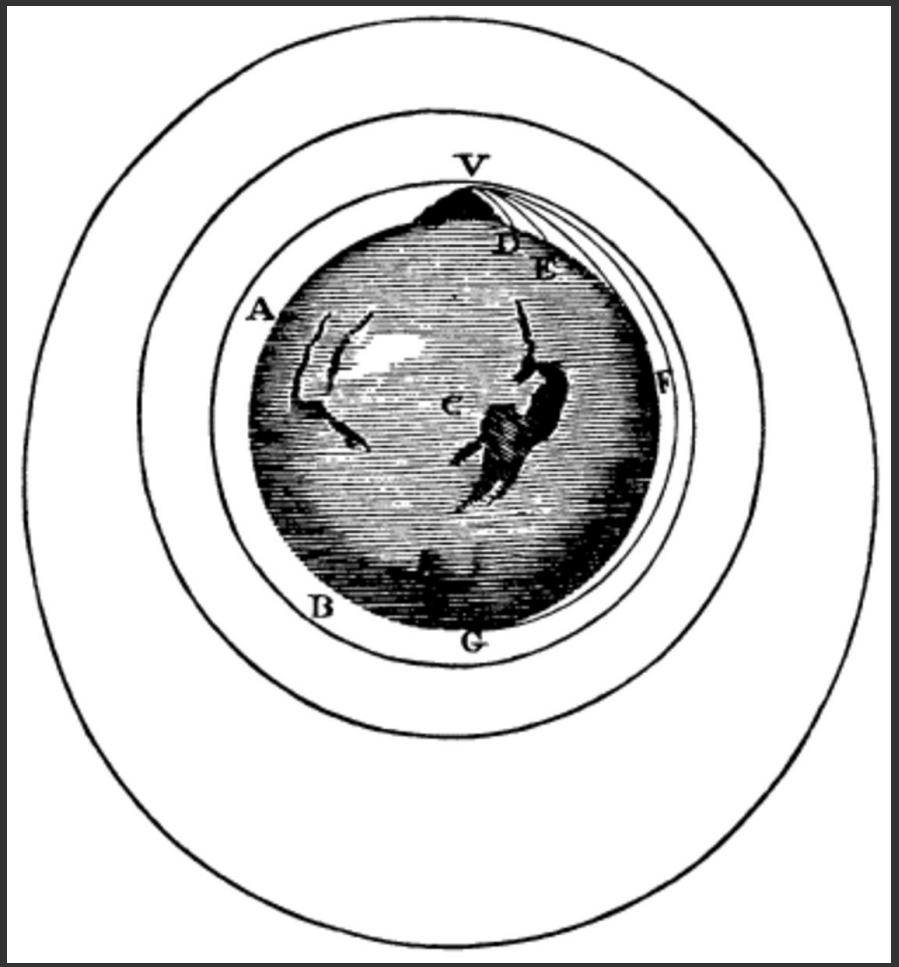
- *Almost* aerospace engineer
- Python developer in finance at Indizen for BBVA
- Mostly self-taught programmer (some Fortran 90 at the University)
- Passionate about **open culture** – source, hardware, science
- Chair of **Python Spain non-profit** and organizer of Python Madrid monthly meeting





Orbiting baseballs

...and Newton's cannonball



What is Astrodynamics then?

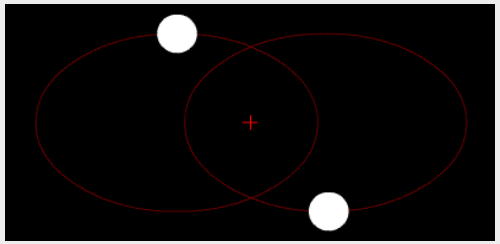
Physics > Mechanics > Celestial Mechanics > Astrodynamics

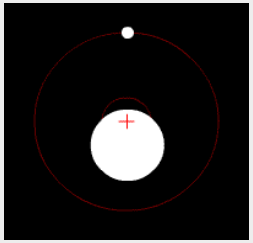
“A branch of Mechanics (itself a branch of Physics) that studies practical problems concerning the motion of human-made objects trough space”

Warning: This *is* rocket science!

Two-body problem

- Main problem in Celestial Mechanics
  - Two **point masses**
  - **Only gravitational force considered**
- The two motions are now decoupled!





$$\ddot{\mathbf{r}} = -\frac{\mu}{r^2}\hat{\mathbf{r}}$$

Kepler problem

- It's the **initial value problem (IVP)** of the two-body problem, also known as **propagation**
- *Statement: determine the position and velocity of a body in a specified moment in time, given its state in a previous moment*
- For elliptic orbits:

$$M = E - e \sin E$$

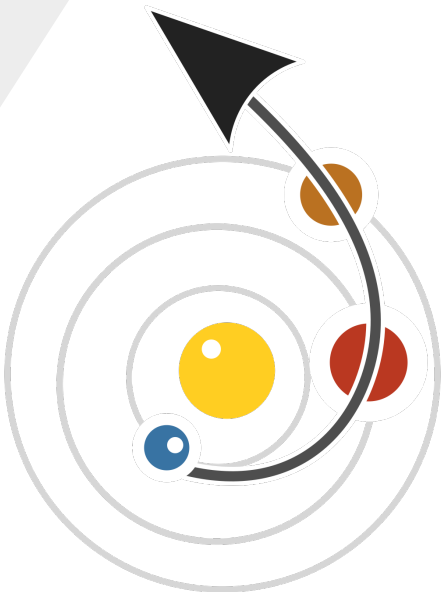
$$M = n(t - t_0)$$

Lambert problem

- It's the **boundary value problem (BVP)** of the two-body problem
- *Statement: determine the trajectory between two positions to be traveled between two moments in time*
- In the earliest phase we can assume that planets are point masses and consider only Sun's gravity ("patched conic approximation")

poliastro: Astrodynamics in Python

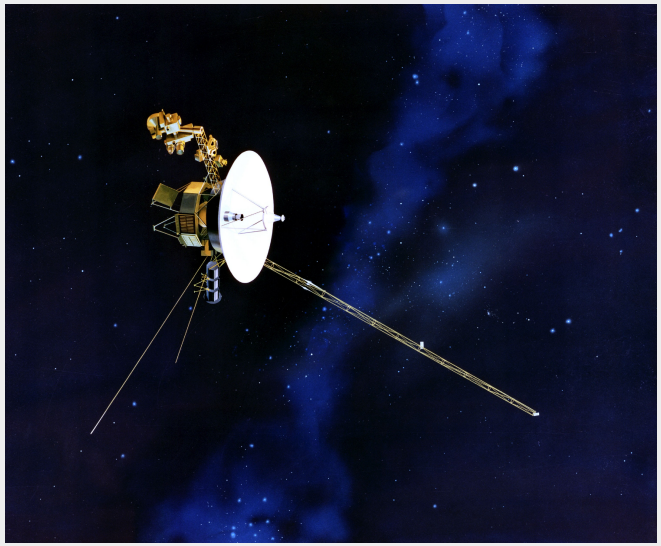
- **Pure Python**, accelerated with **numba**
- **MIT License** (permissive)
- **Physical units handling** (thanks to astropy)
- Analytical and numerical orbit propagation
- Conversion between position/velocity, classical and equinoctial orbital elements
- Simple 2D trajectory plotting (thanks to matplotlib)
- Hohmann and bielliptic maneuvers computation
- Initial orbit determination (Lambert problem)
- Planetary ephemerides through SPK SPICE kernels (thanks to jplephem)



astropy: Astronomy in Python

- Common library for Astronomy projects in Python
  - **Physical units** (`astropy.units`): static typing for engineers 😀
  - **Dates and times** (`astropy.time`): time vectors, conversion to Julian dates (JD), SOFA routines
  - **Reference systems conversion** (`astropy.coordinates`)
- Other: cosmological computations (`astropy.cosmology`), FITS data (`astropy.io.fits`)

jplephem: planetary ephemerides

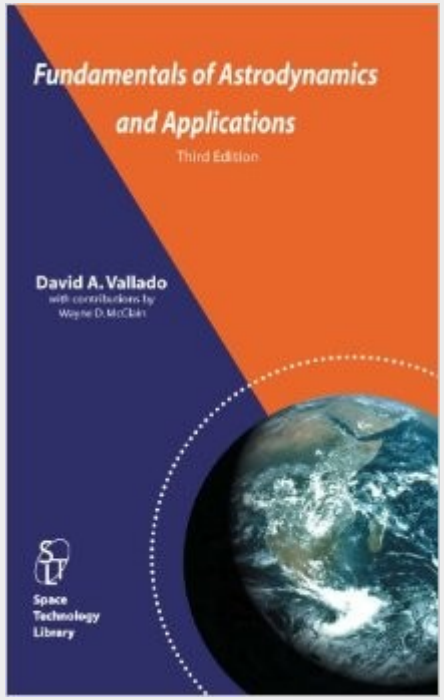


- NASA and JPL provide planetary positions (**ephemerides**) with great accuracy along broad time ranges (100s or 1000s years) in a binary format (SPK kernels)
- jplephem, by Brandon Rhodes♥, reads SPK files

♥Otras bibliotecas: **python-sgp4**, **python-skyfield**

Algorithms in compiled languages

- Most analysis require solving these problems **thousands of times**
  - Orbital groundtracks
  - Launch window opportunities
  - Trajectory optimization
- Online: Fortran, C, MATLAB, Java
  - Pros: Good performance
  - Cons: Poorly written, no testing, works-on-my-computer state, wrapping



numba: JIT for numerical Python

- numba is a BSD licensed, just-in-time compiler for *numerical* Python code
- Optimized to work with NumPy arrays
- Support for a (expanding) subset of the language (highly dynamical features tend to hurt performance)
- **Compiles to LLVM**, hence leveraging its power to this powerful toolset
- Support for GPUs too!

The results against Fortran

Version	Min	Max	Median	Relative
Intel ifort, -O2	594620.8	654121.4	623536.2	**1.0**
GNU gfortran, -O2	358478.2	505127.0	454613.6	**0.729**
Python + numba	197610.9	206153.2	203615.8	**0.327**
pure Python	3502.7	3703.0	3639.6	**0.006**

Table: Benchmarking results

This is PYTHON!

The journey of Juno

https://www.youtube.com/watch?v=sYp5p2oL51g

Conclusions

- Python not only rocks as a language: it can be *fast enough* using some tricks
- The ecosystem of libraries is simply awesome and super high quality
- Several things missing in poliastro: 3D plotting, better APIs
- Open development and good documentation make progress and collaboration accessible to anyone

Gracias a todos
Eskerrik asko
Keep on dreaming

@astrojuanlu
hello@juanlu.space