Orekit for the Global Trajectory Optimization Competitions

3rd Orekit Talk

Romain Serra [1]

[1] Astrodynamicist &
Orekit developper

April 19th, 2024

Outline

Outline

Global Trajectory Optimization Competition

Key facts

- Proposes an optimization problem related to space mission analysis and design
- Aimed at fostering research in the field and beyond
- Teams have 4 weeks after release to submit their best solution
- Organized every 1-to-2 years by the previous winner

A few important dates

2005 Introduction by Dario Izzo (ESA's Advanced Concept Team)

2017 First online validation system (GTOC 9)

2025 GTOC 13?

Fun facts

- There is always a fictitious story for context
- Registration is free and there is no prize money
- Since 2017, it has a live leaderboard and a real-time scoring factor
- The organizers are given carte blanche
- The trophy represents the solution to GTOC 1
- NASA's Jet Propulsion Lab has won the most medals
- Sometimes dubbed the America Cup of Rocket Science
- Inspired the creation of the China Trajectory Optimization Competition

The actual optimization

High-level part (combinatorics) e.g.:

- Selecting ships and asteroids/stars/debris
- Sequencing flybys/rendezvous

Low-level part (trajectory arcs) e.g.:

- Keplerian dynamics, possibly with patched conics
- Control, possibly impulsive
- Boundary conditions, possibly rendezvous

Assumptions are fairly simple but the problem's size makes it hard

The actual optimization

There are always surprises in GTOCs:

- 9 Dynamics including J_2 zonal harmonics
- X Dynamics not even (un)perturbed Keplerian motion
- 11 Scheduling aspect
- 12 Number of available ships strongly limited by the solution's efficiency

And don't underestimate compliance with the submission system!

Personal experience

- Participating since 2017 (GTOC 9)
- In total about 20 different teammates (recurring: Carlos Ortega and Ivan Sumelzo)
- Best performance is rank 6, achieved twice (GTOCs 9&11)
- Recurring roles: trajectory propagation, team lead, optimal control
- Programming languages used: Matlab, C/C++, Python
- Used Orekit only for last edition

Outline

Disclaimer

One does NOT win with just astrodynamics, not even optimal control, as high scores require a well-mixed cocktail:

- Preliminary analysis
- Search algorithms for combinatorics
- Efficient heuristics and simplified models
- State-of-the-art non-linear programming
- Global and local refining techniques
- Computational power
- Time, time, time

Usually no need for accurate frame transformations, elaborate force models or analytical averaging theories.

Why use Orekit then?

General reasons

- Open source
- Reasonably fast
- Object oriented
- Well validated
- Fancy orbit propagation

Personal motivations

- Fair knowledge of the code
- Advocate for the library

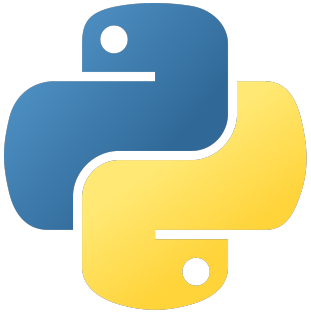
"Sustainable Asteroid Mining" in a nutshell

- Send spacecraft out to drop mass collectors on asteroids
- Need to return them to Earth
- 15 years timeframe
- 60000 possible targets available
- When active, thrust is constrained in magnitude
- Gravity assists are possible at Venus, Earth and Mars
- List of mined asteroids is updated live and impacts the score
- Max. number of allowed ships depends on average collection
- Very stringent tolerances for submission, especially with mass

Our team

- Called The Mean Anomalies (abbreviation: the Ms)
- Partial recast from GTOC 11 (The Eccentric Anomalies, or Es)
- 6 people active, half of them "newbies"
- Limited time and computational resources compared to previous participations
- Ranked 13 out of 28
- Only team using Orekit?

Library set up



Version 11.3 via Python wrapper (thanks Petrus)

Pros of the language

- Handy for prototyping and plotting
- Loads of open-source libraries for scientific computing, including other astrodynamics ones e.g. ESA's *pykep*

Cons

- Performance downgrade (back and forth calls)
- Difficulties of Java exposure (casting, inheritance, etc.)

What to expect with Orekit $\geq$ 12

Code has been adapted to **version 12.0** (API changes) after the competition, but not really optimized.

Some relevant perspectives from newer versions:

- Optimizable event detection with *AdaptableInterval* (12.0)
- Better computational performance from numerical propagation, including with state transition matrix (12.1?)
- Faster calculations involving *CalculusFieldElement* and *Gradient* in particular (12.1?)

Optimal Control Problems

A clear need to design arcs joining planets and/or asteroids. Let **x** be the state vector and **u** the control:

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{u}) \text{ s. t. } \mathbf{u} \in \mathcal{U}, (t_0, \mathbf{x}_0) \in \mathcal{D}_0, (t_f, \mathbf{x}_f) \in \mathcal{D}_f$$

Might as well minimize w.r.t. to some cost function J i.e. use optimal control. There are 2 main types of **local** methods for such a problem:

Indirect
1. Derive optimality conditions (with theorems like Pontryagin's Maximum Principle)
2. Discretize to solve for them

Direct
1. Discretize
2. Solve optimality conditions (requires Non-Linear Programming)

Always need to compute partial derivatives!

Optimal Control Approach

Not intuitive to decide what J should be. Since starting from scratch, used incremental developments:

Indirect

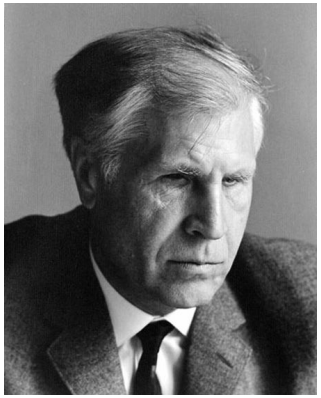
1. Min. acceleration energy with time-fixed boundaries
2. Min. force energy with time-fixed boundaries
3. ~~Min. time~~

Direct

1. Min. fuel with time-fixed boundaries
2. Min. time

Unfortunately there was no time to exploit the direct methods!

Indirect approach



According to Pontryagin's Maximum Principle, there are adjoint variables **p** satisfying optimality conditions:

Adjoint equations Differential system coupled with optimal state $\mathbf{x}^*$, dependent on dynamics and coordinates

Hamiltonian's maximization Relationship between **p**, $\mathbf{x}^*$ and optimal control $\mathbf{u}^*$, also dependent on cost function

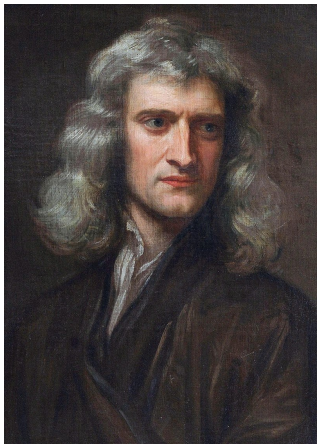
Transverse equations Boundary conditions, linked to the state's ones

Indirect shooting

With fixed final (resp. initial) conditions, the problem is fully represented by the initial (resp. final) adjoint variables.

Single shooting principle

1. Start with a guess (finding it can be an issue)
2. Iterate with Newton-like algorithm
3. Stop when optimality conditions are met within tolerance



The Ms' indirect shooting

Assumptions made for the early developments:

- Fixed initial and final times
- Cartesian coordinates
- "Energy" $\int |\mathbf{u}|^2$ as cost function

The actual min. fuel is $\int |\mathbf{u}|$, but :

- Solutions are smoother (not bang-bang, so better convergence)
- We also started GTOC 11 with min. enegy trajectories

The Ms' indirect shooting

How to implement with Orekit:

- *(Field)AdditionalDerivativeProvider* for adjoint variables and equations
- *(Field)EventDetector* for control singularities spanning from Hamiltonian maximization
- *IntegratedEphemeris* to produce ephemeris needed for submission
- *Gradient* and *FieldNumericalPropagator* for automatic differentiation

Actually started with constrained control acceleration rather than force:

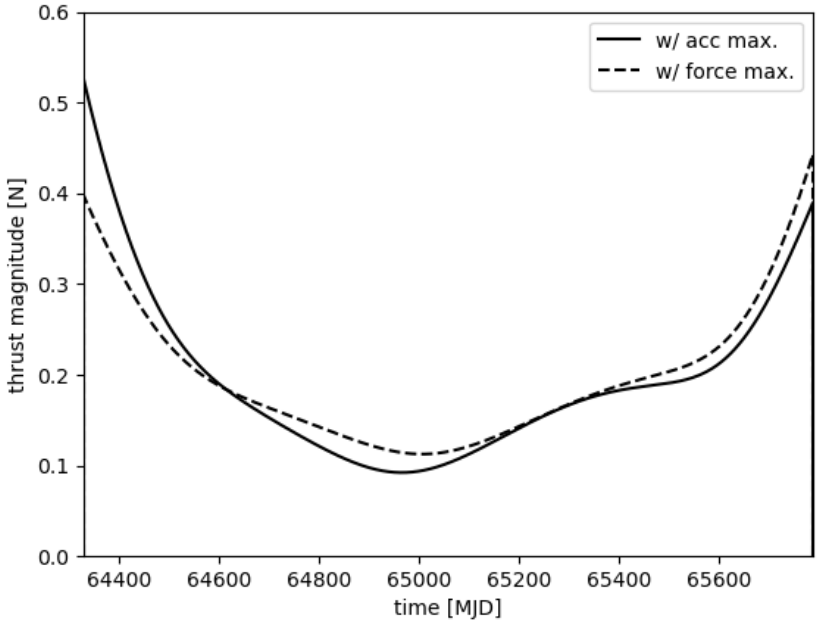
Advantage Mass is no longer a state variable

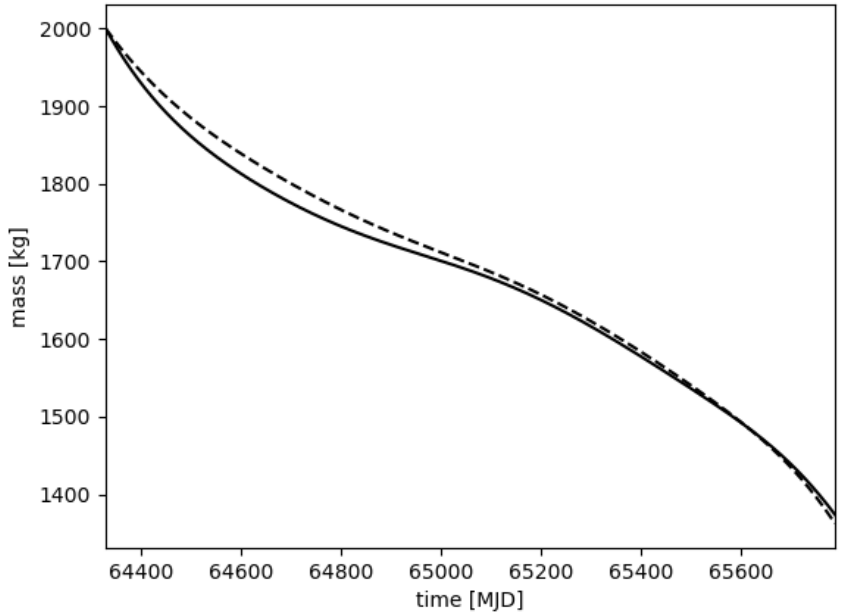
Inconvenient Need to decide on a magnitude bound (made easier with backward propagation)

An example

Scenario

- Transfer from Earth to Asteroid 200 in 4 years
- Initial mass is 2000 kg
- Forward force- and acceleration-constrained solutions





Feedback on indirect shooting

Good stuff

- Used successfully for all our submissions, shifting initial/final times at higher level in the design process
- First estimate of adjoint vector was refined later with tighter tolerances to pass validation checks

Things not working

- Min. time convergence, likely due to choice of coordinates?
- Min. fuel, didn't even try it due to low performance in Python

Room for improvement

- Orekit SI units not ideal for heliocentric orbits, might want to go back down at Hipparchus level
- Use slower state variables w.r.t. independent one
- Implementation in pure Java, at least for some bricks

Direct approach

Two main strategies to get finite-dimension, non-linear problem:

Collocation Discretize the independent variable down to integration-step level

Shooting Parametrize control law a priori and still propagate state vector

Then rely on solvers (for Karush-Kuhn-Tucker conditions, generalizing Lagrange multiplers).

Better provide accurate partial derivatives (so with auto. diff.)!

The Ms' direct shooting

Choices

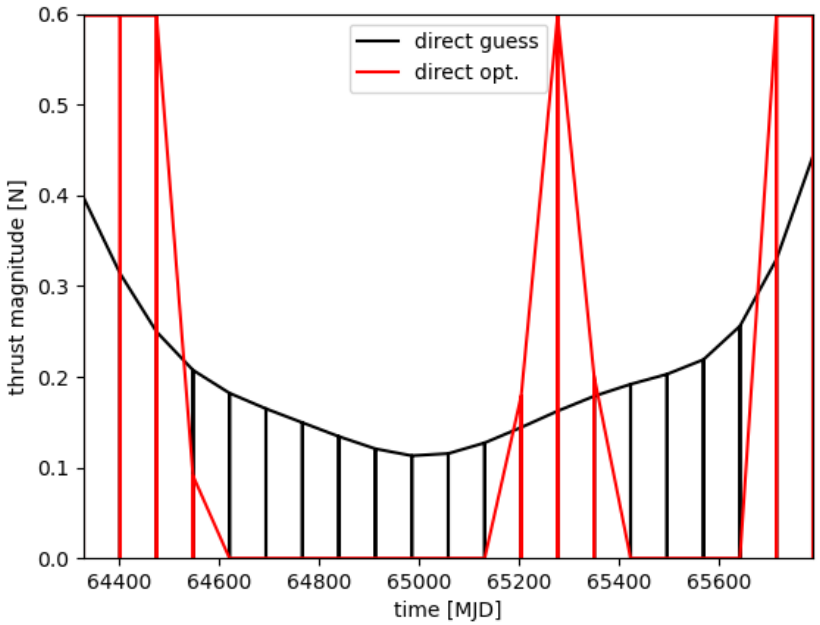
- Scipy's NLP
- Single shooting (piecewise-constant control in spherical coordinates)
- Write boundary conditions with Cartesian or equinoctial variables
- Initial guess derived from indirect

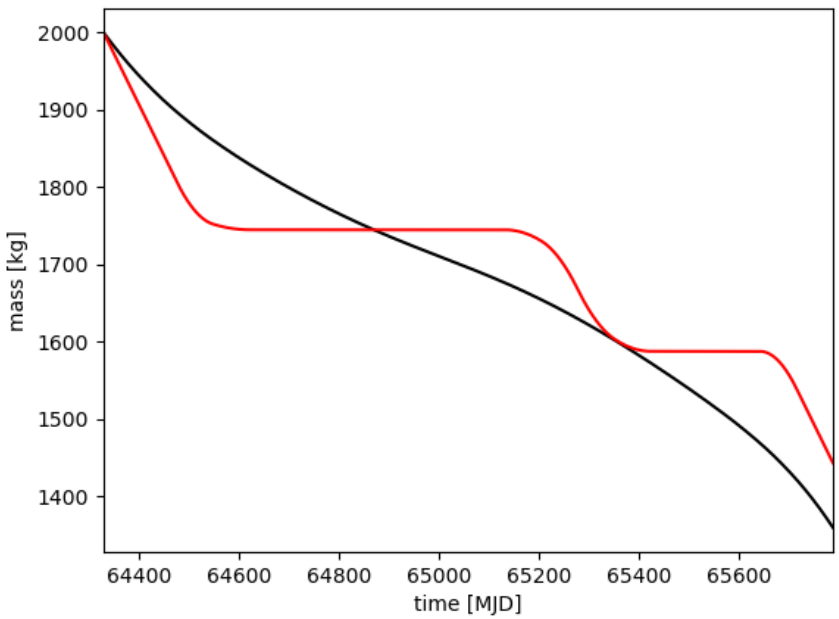
My (questionable) implementation

- *FieldNumericalPropagator* (with *Gradient*)
- Custom implementation of *ForceModel*

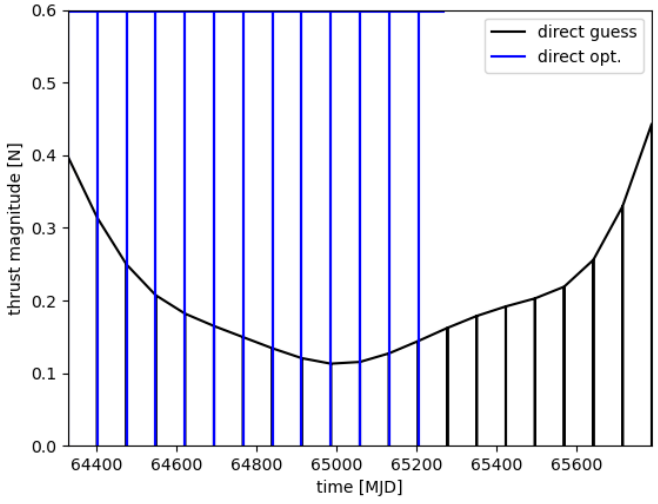
Min. fuel example

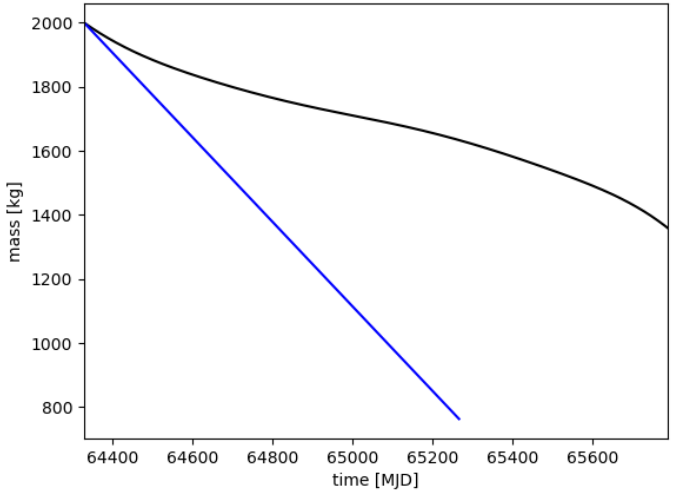
20 subintervals

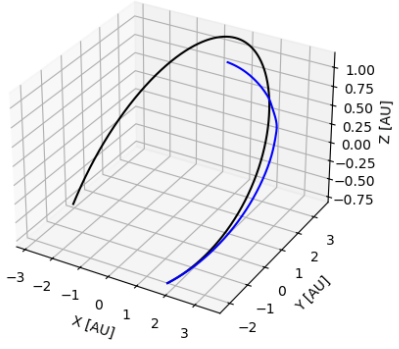




Min. time example







Feedback on direct shooting

Good stuff

- Offered more optimization perspectives, yet too late

Things not working

- Optimization on series of drop-off/pick-up, no time to properly try

Room for improvement

- Use *Maneuver* interface instead of parent *ForceModel*?
- Switch to Orekit's State Transition Matrices rather than full-*Field* propagation (for performance)

Outline

Summary

About the GTOC in general

- Problem's difficulty comes from combinatorics coupled with the short deadline
- Astrodynamics is secondary (building brick)
- Dynamics is always simple, yet propagation important too

About Orekit in this context

- Powerful numerical integration framework
  - ✓ Backward/forward propagation
  - ✓ Event detection
  - ✓ Dense output
  - ✓ Secondary variables
  - ✓ Automatic differentiation
- Still need to watch bottle necks, even in pure Java

Perspectives

- Likely to participate again, probably use Orekit too
- Could contribute some optimal control features (adjoint equations) to library and tutorials
- Will try to keep on improving Orekit's computational performance

Reach out to me with your ideas on the forum (username: Serrof)

Some references

M. Cerf.

Optimization techniques ii: Discrete and functional optimization.

In *Optimization Techniques II*. EDP Sciences, 2023.

G. Colasurdo and L. Casalino.

Indirect methods for the optimization of spacecraft trajectories.

Modeling and Optimization in Space Engineering, pages 141–158, 2013.

D. Izzo, D. Hennes, L. F. Simões, and M. Märtens.

Designing complex interplanetary trajectories for the global trajectory optimization competitions.

Space Engineering: Modeling and Optimization with Case Studies, pages 151–176, 2016.

Some references (our contributions)

C. Ortega, L. A. Ricciardi, M. Di Carlo, C. Greco, R. Serra, M. Polnik, A. Vroom, A. Riccardi, E. Minisci, and M. Vasile.

Gtoc9: Methods and results from strathclyde university. on the generation and evolution of multiple debris removal missions.

Acta Futura, 2017.

C. Ortega, R. Serra, I. Sumelzo, N. Charpigny, J. Labroquère, V. Munoz, J. Olympio, and V. Rodriguez-Fernandez.

Design of impulsive asteroid flybys and scheduling of time-minimal optimal control arcs for the construction of a dyson ring (gtoc 11).

Acta Astronautica, 201:94–110, 2022.