PYGMO AND PYKEP: OPEN SOURCE TOOLS FOR MASSIVELY PARALLEL OPTIMIZATION IN ASTRODYNAMICS (THE CASE OF INTERPLANETARY TRAJECTORY OPTIMIZATION)

Dario Izzo

Advanced Concepts Team – European Space Research and Technology Centre (ESTEC)

ABSTRACT

At the intersection between computer science and astrodynamics lies a fertile ground for improving methodologies and performances of spacecraft trajectory computations. In this paper we present two open source projects (written in C++ and exposed to Python) that are focussed around computational efficiency and that allow to script massively parallel optimization of aerospace related problems. In particular, we will show their use for interplanetary trajectory optimization. After having described novel findings and technologies powering these two projects, we will show some use examples. We show results on asteroid selection for human mission to asteroids, on the Global Trajectory Optimization Competitions and, finally, on a novel idea to obtain on-line adaptive mesh during the direct optimization of interplanetary low-thrust problems.

1. INTRODUCTION

We describe two open source projects that the Advanced Concepts Team of the European Space Agency has been developing in the past years and that have recently reached a maturity level sufficient for unexpert users to benefit from them. The combined use of these two scientific libraries allows to easily script massively parallel optimization tasks for use in astrodynamics research. Examples range from interplanetary trajectory optimization (chemical and low-thrust) to legged robot gait design, to rover path planning experiments etc.

Both projects share the same basic structure: a C++ adhoc class hierarchy implementing all speed critical algorithms and capabilities, and a simple Python interface that allows to combine the ease of use of the Python language to the speed of C++ implementations. In Section 2 we describe the first of these projects: PyGMO, followed by the second project, PyKEP, in Section 3. We then start describing in details a number of new technologies used and algorithms that are implemented in these projects and that are exploited in the applications later shown. In Section 4 we describe in detail the implementation of the algorithm Monotonic Basin Hopping in the case of a non linear constrained optimization problem. In the following Section 5 we discuss PyKEP's novel implementations of direct transcription methods for interplanetary trajectory optimization. The following Section 5 concludes introducing three examples where the combined use of PyKEP and PyGMO allows to perform cutting edge scientific research in astrodynamics, fully exploiting modern multicore architectures in a transparent, simple way. We touch upon the use for the GTOC problem, we show some results for the Human Mission to Asteroid project and we conclude introducing some preliminary analysis on-line mesh adaptation.

2. PYGMO

PyGMO is a scientific library for the easy distribution of massive optimization tasks over multiple CPUs. At the core of PyGMO (Parallel Global Multiobjective Optimizer) is a novel paradigm called "generalized island model" [8] for the coarse grained parallelization of optimization algorithms. Candidate solutions to a given optimization problem are represented as individuals. A population of individuals over which an algorithm $\mathcal{A}$ acts to improve the solutions quality with respect to agiven problem is called an island. An ensamble of islands that can share solutions along a defined topology and thus learn on their reciprocal progress is called an archipelago (see Figure 2 for a visualization of a n archipelago). Once the user defines its own problem, thus coding the objective function and possibly the constraints, the optimization can be defined by a short script as shown in Figure 1

Different algorithms can be placed on different islands, populations of different sizes can be employed (also made by one single individual) in one archipelago. PyGMO offers the usr a great number of algorithms. A non exhaustive list of these algorithms is given in Table 1, most of them are coded directly in PyGMO, others use external dependencies and are wrapped up in PyGMO and thus need third pary libraries installed separately (NLOPT, SciPy, SNOPT, GSL). In all cases the algorithms are offered to the user with the same simple syntax and can be mixed up in archipelagos to perform research on hyper-

```
from PyGMO import *
prob = problem.schwefel(dim = 50)
algo = algorithm.de_self_adaptive(gen = 500)
t = topology.ring()
archi = archipelago(algo,prob,8,20,topo = t)
archi.evolve(10)
print min([isl.population.champion.f for isl in archi])
```

Figure 1. A simple quick start script for PyGMO (Python). 8 islands containing 20 individuals will solve (on 8 parallel threads) the 50 dimensional Schwefel problem using an adaptive version of Differential Evolution (jDE) [1] and exchanging individuals along a ring topology.

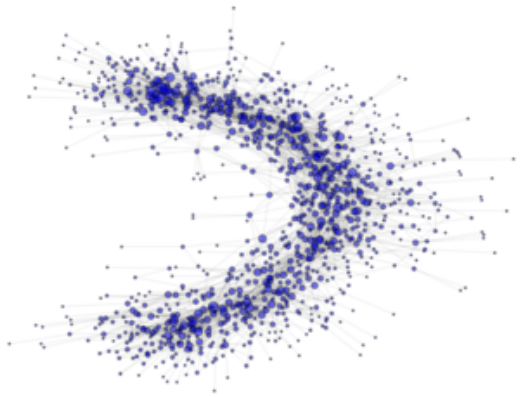

Figure 2. PyGMO: visualization of a large archipelago (CPU network) containing 1024 islands connected with an ageing clustered Barabasi-Albert topology

heuristics (e.g. [7]). The user can also code his own algorithm (both in C++ or directly in Python) and the generalized island model with automatically perform the coarse parallelization for the user. The main philosophical driver of the whole project is to make complex technologies such as the generalized island model and new trends in algorithms coming from evolutionary computing or operation research, available to any user that understand these technologies at a very minimum level. For more information about the generalized island model see [8], while for a systematic study on the effect of archipelago's topologies on the resulting optimization problem see [11].

PyGMO was selected to be part of the 2010 Google Summer of Code (GSoC), an experience that led to a great improvement of the sotware itself, and convinced us to advocate for the implementation of a similar idea at the Europan Space Agency. In 2011, the first Summer of Code in Space (SOCIS), organized by the Advanced Concepts Team of the European Space Agency, innovated on the way Space Agencies can help developing aerospace applications.

Table 1. Some of the algorithms in PyGMO as of V1.0. The type of problems each algorithm can be applied to is reported (Continuous, Integer or Mixed Integer)-(Constrained, Unconstrained)-(Single, Multi-objective).

Name	Type
Differential Evolution	C-U-S
Self-adaptive Differential Evolution	C-U-S
Particle Swarm optimization	C-U-S
Simple Genetic Algorithm	MI-U-S
Artificial Bee Colony	C-U-S
Improved Harmony Search	MI-U-M
Covariance Matrix Adaptation-ES	C-U-S
Monotonic Basin Hopping	MI-U-M
Simulated Annealing	C-U-S
Nelder-Mead	C-U-S
COBYLA	C-C-S
BOBYQA	C-C-S
BFGS	C-U-S
Augmented Lagrangian	C-C-S
SNOPT	C-C-S
IPOPT	C-C-S
NLOPT	N/A
SciPy	N/A

3. PYKEP

PyKEP is a scientific library for simple astrodynamical computations (hence the "keplerian" in the title) centered on the efficiency of its computations. It contains implementations of ephemerides computations a multiple revolution Lambert's solver, propagators of keplerian dynamics based on universal variables as well as on Lagrange coefficients, Taylor intagrators for constant thrust trajectories, patched conics relations, interplanetary trajectories direct transcriptions and more. The software has been validated extensively during the Global Trajectory Optimization Competitions [5, 6] where it has been employed by the Advanced Concepts Team as a primary computational tool, as well as in other mission analysis computations performed in cooperation with international partners such as NASA's Jet Propulsion Laboratory [14].

4. PYGMO'S GENERALIZED MBH

Monotonic Basin Hopping [3] is a global optimization algorithm which, in its original version, works on unconstrained, single objective, continuous problems. It performs a Monte-Carlo type of search substituting the original objective function $f(\mathbf{x})$ with $g(\mathbf{x}) = \mathcal{S}(\mathbf{x})$, where $\mathcal{S}$ is, typically, the result of a local search procedure starting from $\mathbf{x}$. The PyGMO implementation generalizes mbh defining it as a meta-algorithm as shown in Algorithm 1. To implement a MBH algorithm one then needs a procedure $\mathcal{G}$ to generate a random initial population, a procedure $\mathcal{S}$ (pop) that runs an optimization on pop and a comparison criteria $Best$ to decide the best out of two populations.

Algorithm 1: Monotonic Basin Hopping (generalized)

```
pop ← G();
pop* ← S(pop);
while k < MNI do
    pop_new ← P(pop*);
    pop*_new ← S(pop_new);
    if Best(pop*_new, pop*) = pop*_new then
        pop* ← pop*_new;
    end
end
```

In PyGMO, one can use any optimization algorithm to define the $\mathcal{S}$ procedure, while the $\mathcal{G}$ procedure initialize at random the population within the problem's box bounds. The procedure $\mathcal{P}$ perturbing the population perturbs each element of an individual by a percentage defined by the user (with respect to the box bounds width). Finally, the comparison criteria $Best$ is problem dependent (a virtual function) and the user can reimplement it when defining the optimization problem. If the user does not provide its own, PyGMO will use a default implementation which works generically on all problem types comparing the best individuals of the two populations:

> The number of constraint satisfied (equality and inequality) are first compared, if both individuals are equally feasible (to a fixed tolearance) the norm of the constrained violations is evaluated and compared. If both individuals have no constraint violation, the one that is dominated by the least number of individuals is considered best and if also this fails, then the individuals are considered, simply, as different and the comparison fails returning false.

This default criteria is general enough to deal with mixed integer, continuous, integer, constrained, non constrained, multi objective and single objective problems.

5. THE REPRESENTATION OF ONE INTERPLANETARY LEG

Consider the Sims-Flanagan transcription method [13] to transcribe the optimal control problem (OCP) of low-thrust interplanetary trajectorties into a non linear programming problem (NLP). In such an approach, implemented in the widely used tools GALLOP and MALTO [15, 16], each trajectory leg (or phase) is divided into equally time spaced segments and an impulsive ΔV is applied in the mid-point of each segment. Backward and forward propagation are used to compute the spacecraft state mismatch in the middle of each leg as defined by the sequence of impulses. The resulting Non Linear Proramming problem resulting from imposing the mismatch to be zero as a constraint is solved by means of a Sequentail Quadratic Programming method (SNOPT [4] is the most widely used algorithm for this purpose). This simple approach is, arguably, today's most efficient direct optimization method in preliminary trajectory design, as demonstarted also by the sucess (in terms of ranks achieved) of the teams that used it during the Global Trajectory Optimization Competitions. The main advantages of the method are 1) its easy algoritmic implementation, 2) its ample radius of convergence also when multiple fly-bys are involved, 3) the efficiency of the trajectory computations that involve, essentially, a sequence of keplerian propagations. The method drawback is, essentially, its low-fidelity nature (perturbations cannot be accounted for, the thrust is approximated as a sequence of impulses).

Let us describe, formally, the Sims-Flanagan transcription. We consider a single trajectory leg that we consider divided into N segments of which the first n_{fwd} are forward propagation segments and the following n_{bck} are backward propagation segments (see Figure 3). Consider the following definitions:

Forward propagation:

$$\begin{aligned} \mathbf{x}_{i+1}^{m^-} &= \Phi_{\mathbf{x}_i}(\Delta t/2)\mathbf{x}_i \\ \mathbf{x}_{i+1}^{m^+} &= \mathbf{x}_{i+1}^{m^-} + \Delta\mathbf{x}_{i+1} \\ \mathbf{x}_{i+1} &= \Phi_{\mathbf{x}_{i+1}^{m^+}}(\Delta t/2)\mathbf{x}_{i+1}^{m^+} \end{aligned} \qquad \forall i = 0, ..., n_{fwd} - 1 \quad (1)$$

Backward propagation:

$$\begin{aligned} \tilde{\mathbf{x}}_{i-1}^{m^+} &= \Phi_{\tilde{\mathbf{x}}_i}(-\Delta t/2)\tilde{\mathbf{x}}_i \\ \tilde{\mathbf{x}}_{i-1}^{m^-} &= \tilde{\mathbf{x}}_{i-1}^{m^+} - \Delta\tilde{\mathbf{x}}_{i-1} \\ \tilde{\mathbf{x}}_{i-1} &= \Phi_{\tilde{\mathbf{x}}_{i-1}^{m^-}}(-\Delta t/2)\mathbf{x}_{i-1}^{m^-} \end{aligned} \qquad \forall i = N, ..., N - n_{bck} + 1 \quad (2)$$

where $\Phi_{\mathbf{x}}(t)$ is the keplerian transition matrix advancing the spacecraft state $\mathbf{x} = (\mathbf{r}, \mathbf{v}, m)$ of t and $\Delta\mathbf{x}_i = (\mathbf{0}, \Delta\mathbf{V}_i, [\exp(-\Delta V_i/I_{sp}g_0) - 1]m_i^-)$ represents the discontinuity in each segment mid-point modelling the spacecraft thrust. To illustrate how, starting from the previous definitions, we can transcribe an optimal control problem of an interplanetary spacecraft, we consider the problem of transferring the spacecraft from the starting

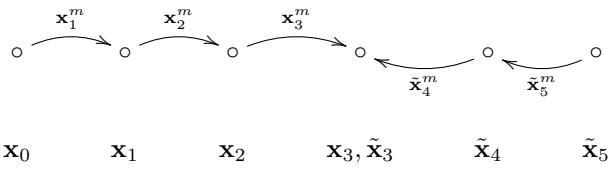

Figure 3. An example of a Sims-Flanagan trajectory leg having $N = 5$ segments, where $n_{fwd} = 3$ and $n_{bck} = 2$. The state at the third grid node as evaluated both by forward and backward propagation needs to be the same for the leg to be dynamically feasible.

state $\mathbf{r}_s, \mathbf{v}_s, m_s$ to the final $\mathbf{r}_f, \mathbf{v}_f$ in the fixed time ΔT and maximizing the final mass m_f.

$$\begin{array}{rl} \text{find:} & \mathbf{y} = (m_f, \Delta\mathbf{V}_1 ..., \Delta\mathbf{V}_N) \\ \text{to maximize:} & m_f \\ \text{subject to:} & \tilde{\mathbf{x}}_{N-n_{bck}} = \mathbf{x}_{n_{fwd}} \\ & \mathbf{x}_0 = (\mathbf{r}_s, \mathbf{v}_s, m_s) \\ & \tilde{\mathbf{x}}_N = (\mathbf{r}_f, \mathbf{v}_f, m_f) \\ & \Delta V_i \leq T_{max}\Delta T/N \\ & \mathbf{lb} \leq \mathbf{y} \leq \mathbf{ub} \end{array} \tag{3}$$

where T_{max} is the maximum thrust of the thruster. The first equality constraint in the above equations, is called state mismatch constraint, while the first N inequality constraints are called thruster constraints. It is possible to compute the derivatives of all the constraints and of the objective functions, thus enhancing the efficiency of NLP solvers. This can be done by differentiating in cascade Eqq.(1,2), which requires the differentiation of the transition matrix Φ and, essentially, of the Lagrange coefficients [**?**].

5.1. PyKEP interplanetary leg

In our open source project PyKEP (see Section 3) a modified version of the Sims-Flanagan model is implemented. The modifications made were motivated by the need to implement a direct transcription method easily used in the global optimization framework of PyGMO, and to improve the trajectory model fidelity so that more a effective representation of the thrust and a more precise propagation of the dynamics could be included in the model.

We thus introduce the throttles $\boldsymbol{\eta}_i$ to encode in the decision vectors the various thrust actions, instead of the $\Delta\mathbf{V}_i$. We define the throttles using the equation:

$$\Delta\mathbf{V}_i = \boldsymbol{\eta}_i T_{max}\Delta T/N$$

the NLP becomes:

$$\begin{array}{rl} \text{find:} & \mathbf{y} = (m_f, \boldsymbol{\eta}_1 ..., \boldsymbol{\eta}_N) \\ \text{to maximize:} & m_f \\ \text{subject to:} & \tilde{\mathbf{x}}_{N-n_{bck}} = \mathbf{x}_{n_{fwd}} \\ & \mathbf{x}_0 = (\mathbf{r}_s, \mathbf{v}_s, m_s) \\ & \tilde{\mathbf{x}}_N = (\mathbf{r}_f, \mathbf{v}_f, m_f) \\ & \eta \leq 1 \\ & \mathbf{lb} \leq \mathbf{y} \leq \mathbf{ub} \end{array} \tag{4}$$

This description has several advantages, one of them being the simplified form of box-bounds on the throttles which now are, simply, $-\mathbf{1} \leq \boldsymbol{\eta}_i \leq \mathbf{1}$. We then add the possibility (hugh-fidelity mode) to substitute the two keplerian propagations in each segment with the numerical propagation of a constant thrust segment, so that, in place of Eqq.(1,2) we consider the simpler definitions:

Forward propagation:

$$\mathbf{x}_{i+1} = \varphi(\Delta t, \mathbf{x}_i),\ \forall i = 0, ..., n_{fwd} - 1 \tag{5}$$

Backward propagation:

$$\tilde{\mathbf{x}}_{i-1} = \varphi(-\Delta t, \mathbf{x}_i),\ \forall i = N, ..., N - n_{bck} + 1 \tag{6}$$

where $\varphi(t, \mathbf{x})$ is computed as the result of the full propagation of the spacecraft state subject to a constant thrust $\mathbf{T} = \boldsymbol{\eta} T_{max}$. Such a propagation has a higher computational cost with respect to the keplerian propagation. Infact it is roughly two orders of magnitude slower (depending on the implementyation) if we consider, say, a Runge-Kutta-Fehlberg method. Such a slow down is unacceptable and leads to an almost unusable method. Fortunately, having now divided the trajectory into segments of constant low-thrust we may use a Taylor integration scheme [9, 12] to compute $\varphi(t, \mathbf{x})$. In PyKEP, Taylor integration is implemented in the simplest case of no disturbing accelerations. With respect to the keplerian propagation our implementation results in a slow down of a factor 6 to 10, when modern compilers able to exploit vectorization (i.e. gcc $\geq$ 4.5) are used.

The resulting transcription of the Optimal Control Problem has several properties that are rather attactive:

1. Disturbances can now be accounted for in the trajectory propagation $\varphi(\Delta t, \mathbf{x}_i)$. The Taylor propagator needs to be re-implemented each time a distrubance is added, but the procedure can be made automatic [9].
2. The mismatch constraints, for the very same decision vector, can be computed using both propagation schemes, the faster keplerian low-fidelity one and the slower high-fidelity one. The switch between propagation models can be made also during the optimization process (typically one would start using the keplerian propagation to then complete the trajectory design using the high-fidelity mode). If the mismatch constraints are satisfied in low-fidelity, in high fidelity they will be close to be satisfied as long as the number of segments is adequate and disturbances are small.

3. When using the high-fidelity mode, one can double the number of segments easily by just copying each throttle twice producing a perfectly feasible initial guess for the resulting larger NLP.

5.2. PyKEP adaptive meshing

Aside the implementation of the PyKEP interplanetary leg described above, PyKEP also contains an experimental "adaptive leg" based on a new idea that builds on the work of Yam et al.[17]. In essence, we define a pseudo-time s using the generalization of the Sundmann transformation:

$$dt = cr^{\alpha}ds \tag{7}$$

This allows us to play with the parameters c and α as to slow down or accelerate the time along a trajectory and thus obtain segments of different lengths. To do so, we consider the state as containing also the time t: $\mathbf{x} = [\mathbf{r}, \mathbf{v}, m, t]$. We consider the new equation of motions (11) in the pseudo-time s and, as reported in the Appendix, we build a new Taylor intagration scheme to evaluate their solution and thus build $\varphi(\Delta s, \mathbf{x}_i)$. The Taylor s-integration results to be actually faster than the Taylor t-integartion (as detailed in Table 2 and Table 3) and thus the resulting objective function and constraint evaluation will benefit form the variable change.

We then consider an interplanetary leg as divided in equally spaced (in the pseudo-time) segments of length Δs and we introduce, similarly to what done above, the definitions:

Forward propagation:

$$\mathbf{x}_{i+1} = \varphi(\Delta s, \mathbf{x}_i), \ \forall i = 0, ..., n_{fwd} - 1 \tag{8}$$

Backward propagation:

$$\tilde{\mathbf{x}}_{i-1} = \varphi(-\Delta s, \mathbf{x}_i), \ \forall i = N, ..., N - n_{bck} + 1 \tag{9}$$

We then introduce a new transcritpion for the optimal control problem:

$$\begin{array}{rl} \text{find:} & \mathbf{y} = (s_f, m_f, \boldsymbol{\eta}_1 ..., \boldsymbol{\eta}_N) \\ \text{to maximize:} & m_f \\ \text{subject to:} & \tilde{\mathbf{x}}_{N-n_{bck}} = \mathbf{x}_{n_{fwd}} \\ & \mathbf{x}_0 = (\mathbf{r}_s, \mathbf{v}_s, m_s, 0) \\ & \tilde{\mathbf{x}}_N = (\mathbf{r}_f, \mathbf{v}_f, m_f, \Delta T) \\ & \eta \leq 1 \\ & \mathbf{lb} \leq \mathbf{y} \leq \mathbf{ub} \end{array} \tag{10}$$

With respect to the previous case, the NLP has one added dimension and one more non linear constraint as the optimizer will have to find also the value of the final pseudo-time s_f that satisfies the mismatch constraint on the added state, i.e. the real time.

6. USE EXAMPLES

In the following three subsections we describe some scientific achievments obtained thanks to the combined use of PyGMO and PyKEP. In particular, in all cases, we construct an optimization problem related to interplanetary trajectory design using elements from PyKEP and we solve sucha problem over multiple CPUs using PyGMO's available solvers, in particular the self-adaptive Differential Evolution jDE [1] (for box bounded single objective problems), Monotonic Basin Hopping [3] (for non linearly constrained single objective problems), Nelder-Mead [10] and Simulated Annealing [2].

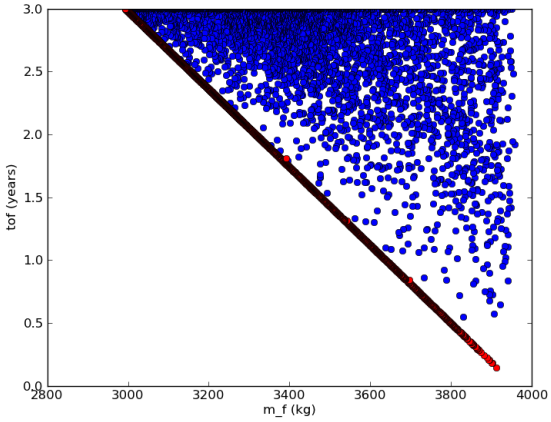

Figure 5. A plot of all launch opportunities found by the low-thrust global optimization of the Earth-asteroid low-thrust trajectory in the GTOC5 problem. Blue: the mass optimal opportunities. Red: the time optimal ones (the spacecraft thrust is always on, hence the linear disposition on the plot.

6.1. Results on GTOC

The Global Trajectory Optimization Competitions [5, 6] started in 2005 to challenge the international community with a extremely difficult problems on interplanetary trajectory optimization on the edge of what is currently possible with known technologies. The Advanced Concepts Team, at the European Space Agency initiated these competitions and was present in each edition either as the organizing institution or as a participating team. The competitions have been used to help and validate the development of PyKEP and PyGMO. While these tools have helped searching efficiently the solution space and ensure that trajectory submitted ranked among the top ones. As an example, during the 5th edition of the competition, where a complex multiple randezvous mission had to be designed for a low-thrust spacecraft, a search on the immense space of possible launch windows (7075 asteroids were possible in a window of fifteen years) was possible thanks to a quite simple PyGMO-PyKEP script (see

```
start = epoch(0)
end = epoch(340)
earth = planet_ss('earth')
mars = planet_ss('mars')
sc = sims_flanagan.spacecraft(4500,0.05,2500)
r,v = earth.eph(start)
x0 = sims_flanagan.sc_state(r,v,sc.mass)
r,v = mars.eph(start)
xe = sims_flanagan.sc_state(r, v ,sc.mass)
mu = MU_SUN
throttles = (1,0,0,0,0,1,0,0,0,0,1,0)
l = sims_flanagan.leg(start, x0, throttles,end, xe, sc, mu)
ceq = l.mismatch_constraints()
cineq = l.throttles_constraints()
```

Figure 4. Example of a PyKEP script to build an interplanetary leg for an Earth-Mars transfer and compute the resulting constraints.

Table 2. Performance of Taylor integartion over a constant thrust segment. Speed is measured in seconds and refers to 10000 integrations (forward+backward) using random initial conditions and thrust values. Max. Err. is the maximum integration error made in the 10000 propagations.

ϵ	s-integration ($\alpha = 1.5$)			t-integration		
	Max. Err.	Mean Err.	Speed (s)	Max. Err.	Mean Err.	Speed (s)
1e-5	3.32	0.0027	0.399158	2.93	0.0124	0.456787
1e-6	0.212	0.000296	0.494545	0.125	0.000955	0.600910
1e-7	0.00215	4.7e-06	0.784847	0.00206	1.47e-05	0.957887
1e-8	0.000353	5.96e-07	0.912959	0.00122	4.1e-06	1.205672
1e-9	4.13e-05	7.31e-08	1.57723	3.05e-05	2.32e-07	1.434423
1e-10	6.85e-06	1.01e-08	1.222635	2.22e-05	7.13e-08	1.666550
1e-11	7.33e-07	1.2e-09	1.384560	5.14e-07	3.85e-09	1.924356
1e-12	6.04e-08	1.61e-10	1.542333	3.89e-07	1.24e-09	2.231468
1e-13	3e-08	2.15e-11	1.724858	8.59e-09	6.48e-11	2.483096
1e-14	1.47e-10	3.38e-13	2.76578	1.54e-10	1.1e-12	3.79955
1e-15	2.6e-11	5.13e-14	2.290430	1.3e-10	3.72e-13	3.382649
1e-16	5.19e-12	7.6e-15	2.492409	3e-12	2.22e-14	3.719196
1e-17	6.39e-13	3.13e-15	2.685531	2.34e-12	1.15e-14	4.69951
1e-18	2.15e-13	2.82e-15	2.933384	8.8e-13	8.82e-15	4.432802

Figure 6) that launches a parallel global search of low-thrust trajectories to randezvous with all available asteroids. The script solves 70750 non linearly constrained global optimization problems transcribing low-thrust optimal control problems. The resulting problem has 39 dimensions, and 12 non linear inequality constraints and 7 equality constraints (the state mismatches, see Section 5. the leg is represented using 10 segments). To give an idea of the speed, the optimization (performed both for maximum final mass and minimal final time) completed, on our servers, in roughly eight hours.

6.2. Results on Human missions to asteroids

Human missions to asteroids (an idea revived by the US President Obama announcement on April 15 2011 at the Kennedy Space Centre) are, from the trajectory design point of view, different in several aspects from those to Mars or to the Moon. Among others because of the wide choice: there are millions of asetroids in the Asteroid belt between the orbits of Jupiter and Mars and thousands in the Near Earth vicinity (NEAs). Not all of these are equally easy and safe to reach and return from.

The Advanced Concepts Team, in cooperation with colleagues form the Jet Propulasion Laboratory [14], performed a preliminary screening of all known asteroids looking for good opportunities to have a first human mission to an asteroid. Using the MPCORB database a prliminary selection was made filtering objects having a diameter smaller than 50m.

We used PyKEP to build, for each asteroid in the MPCORB database, a box constrained problem by patching together two trajectory phases: Erath-Asteroid and Asteroid-Earth. In each phase the spacecraft was allowed one deep-space manouvre. A minimum stay of 20 days was forced on the asteroid in order for the mission to be

Table 3. Performance of Taylor integartion over a constant thrust segment. Speed is measured in seconds and refers to all the 10000 integrations (forward+backward) using random initial conditions and thrust values. Max. Err. is the maximum integration error made in the 10000 propagations.

ϵ	s-integration ($\alpha = 1.0$)			t-integration		
	Max. Err.	Mean Err.	Speed (s)	Max. Err.	Mean Err.	Speed (s)
1e-5	0.0821	0.00173	0.214227	0.0143	0.00718	0.236942
1e-6	0.00674	0.000218	0.279915	0.000624	0.000529	0.331037
1e-7	0.000106	3.23e-06	0.442771	1.04e-05	7.66e-06	0.491794
1e-8	1.82e-05	3.99e-07	0.537666	1.56e-06	1.97e-06	0.635365
1e-9	1.22e-06	5.03e-08	0.631750	1.32e-07	1.2e-07	0.741630
1e-10	2.39e-07	6.37e-09	0.728937	2.54e-08	3.39e-08	0.864929
1e-11	3.02e-08	7.97e-10	0.834116	1.69e-09	1.96e-09	1.8970
1e-12	5.73e-09	1.06e-10	0.944031	3.83e-10	5.81e-10	1.129998
1e-13	3.7e-10	1.26e-11	1.59414	3.55e-11	3.23e-11	1.278076
1e-14	1.81e-11	2.18e-13	1.310364	1.6e-13	5.59e-13	1.570269
1e-15	2.71e-11	4.01e-14	1.443797	6.12e-14	1.89e-13	1.737650
1e-16	6.77e-11	2.08e-14	1.584893	5.4e-14	2.95e-14	1.932467
1e-17	2.54e-11	1.4e-14	1.720881	3.74e-14	2.71e-14	2.85594
1e-18	3.2e-11	1.39e-14	1.875299	1.53e-14	2.61e-14	2.276236

```
from PaGMO import *
from PyKEP import *

for asteroid in gtoc5_ast:
  prob = problem.py_pl2pl(mass=4000, Tmax=0.5, Isp=2500, ...
              ... Vinf_0=5, departure=Earth, target=asetroid)
  algo = algorithm.snopt(feas_tol=1e-9)
  algo2 = algorithm.mbh(algo, stop = 5, perturb = 0.05)
  archi = archipelago(algo2,prob,10,1)
  archi.evolve(1)
```

Figure 6. PyGMO-PyKEP script used in GTOC5 to explore globally for good low-thrust launch opportunities to all 7075 asteroid in a 10 year launch window (the launch window is hidden in the definition of the py_pl2pl problem)

able to achieve the necessary science output. Three scenarions were run where the maximum mission duration was set to be 200, then 400 and finally 600 days as to explore all possible targets for different mission profiles. The Launch window considered was 2020-2035. A maximum hyberbolic excess velocity relative to Earth of 4.5 km/s for the return leg was considered as to allow for a safe landing of the return capsule and a maximum launch ΔV of 5.5 km/sec was required to the optimal trajectory.

The resulting 12th dimensional box constrained problem was solved in a PyGMO heterogeneous archipelago having 7 islands connected in a rim topology and visualized in Figure 9. The algorithm Differential Evolution and Simulated Annealing were used in each island, except for the final one, only receiving migrants, where a Nelder Mead algorithm was instantiated. The optimization results (a total of 16000 global optimization problems were solved) were later analysed producing the graph showed in Figure 7 and 8. The trajectories found were later further analyzed in JPL for low-thrust options and radiation issues.

Thanks to this joint project, a much greater insight was given to worldwide scientists on the number and human reachability of the thousands of asteroids that travel so close to our planet and yet so far.

6.3. Results on adaptive-mesh

A classical approach to meshing in direct method for optimal control is to use an initial uniform mesh, look at the result and refine the mesh according to where the optimization seems to be going. While this constitute an effective approach in many cases, it would be greatly better to have the algorithm take care of adapting the time-mesh while the optimization is advancing. Some attempt to obtain on-line mesh adaptation can be found in [**?**]. In that approach, the exact position of time points was optimized too using an objective function built ad hoc and rewarding bang-bang trajectories. While promising, the method was not developed further as it did restrict considerably the convergene radius. Here we use, instead, PyGMO's adaptive leg that uses points equally spaced in the s variable defined in Eq.(7). We clearly do not want to use

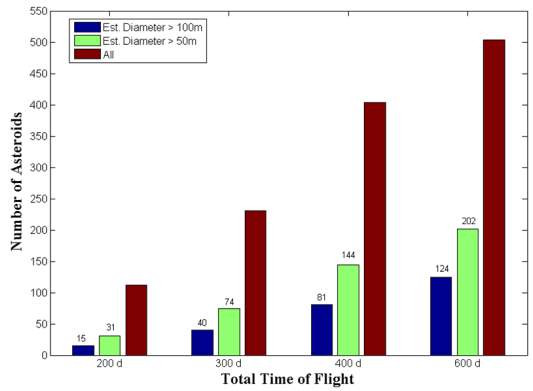

Figure 7. Number of candidate asteroids for a Human mission, divided by mission time (including Earth return trajectory)

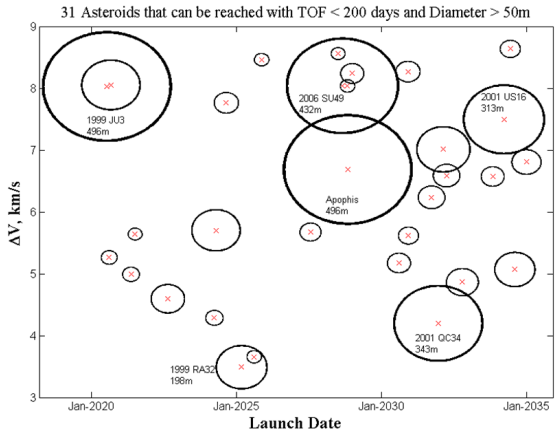

Figure 8. Visualization of the best 31 asteroids candidate for the human mission. The asteroid radius is also visualized.

many points in parts of the trajectory where the thrust is zero, while we would like to characterize trajectory parts with higher thrust. We thus will use the c coefficient of the generalized Sundmann transformation to slow down time when the thrust is high and to speed up the time where the thrust is low. In particular we use

$$c = \frac{1}{1 + \eta \frac{T}{T_{max}}}$$

with $\eta = 3$, so that in ballistic arcs the time will flow four times faster than in full thrust arcs.

We then run a comparison between the use of this adaptive-mesh technique and a standard optimization in two simple cases of single phase transfer. In each case we run monotonic basin hopping for 160 times and record the solution found only if feasibility has been reached. The results are shown in Table 4 for an Earth-Jupiter transfer and in 5 for an Earth-Mars transfer. In Figures 10 and 11 the best rajectries found are plotted in the Earth jupiter

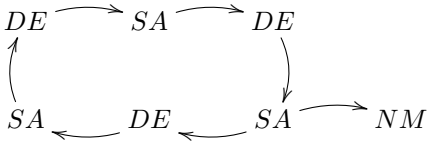

Figure 9. Visualization of the PyGMO's heterogeneous archipelago used in the Human Mission to Asteroid study. Note the alternating Differential Evolution and Simulated Annealing islands and the receiving Nelder Mead island.

Table 4. Earth-Jupiter case: summary of the 160 runs of Monotonic Basin Hopping

	sol. found	best	mean
Adaptive	69 (43%)	637.55 Kg	625.07 Kg
Normal	108 (67%)	635.85 Kg	569.49 Kg

case. There the effect of the adaptive mesh appears visibly in the ballistic arc (blue color).

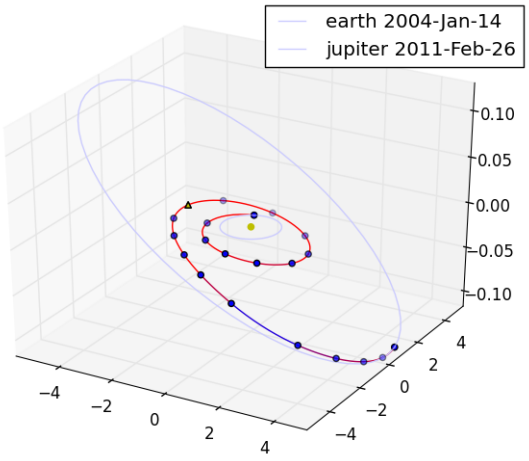

Figure 10. Best trajectory found for the Earth-Jupiter transfer (adapted mesh)

7. APPENDIX: TAYLOR INTEGRATION IN THE SUNDMANN VARIABLE

Consider the generalized Sundmann transformation:

$$dt = cr^{\alpha} ds$$

the following definitions:

$$r = \sqrt{x_1^2 + x_2^2 + x_3^2}$$
$$F = \sqrt{F_1^2 + F_2^2 + F_3^2}$$

Table 5. Earth-Mars case: summary of the 160 runs of Monotonic Basin Hopping

	sol. found	best	mean
Adaptive	25 (16%)	894.5 Kg	735.77 Kg
Normal	156 (97%)	895.2 Kg	875.96 Kg

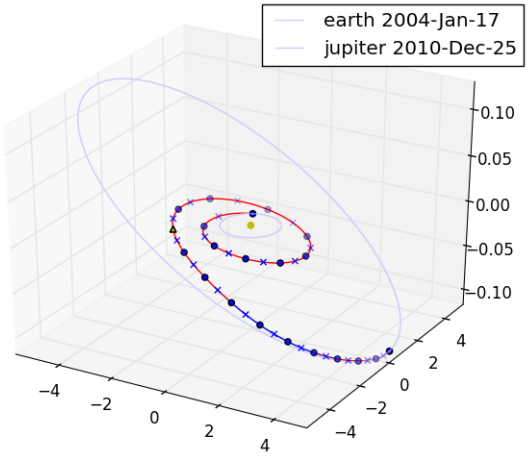

Figure 11. Best trajectory found for the Earth-Jupiter transfer (non-adapted mesh)

and the set of equations:

$$\begin{aligned}
\dot{x}_1 &= x_4 \\
\dot{x}_2 &= x_5 \\
\dot{x}_3 &= x_6 \\
\dot{x}_4 &= -\mu x_1/r^3 + F_1/x_7 \\
\dot{x}_5 &= -\mu x_2/r^3 + F_2/x_7 \\
\dot{x}_6 &= -\mu x_3/r^3 + F_3/x_7 \\
\dot{x}_7 &= -T/(I_{sp}g_0)
\end{aligned}$$

We apply the generalized Sundmann transformation to the above differential system to get the equivalent one:

$$\begin{aligned}
x_1' &= cx_4r^\alpha \\
x_2' &= cx_5r^\alpha \\
x_3' &= cx_6r^\alpha \\
x_4' &= -c\mu x_1/r^{(3-\alpha)} + cr^\alpha F_1/x_7 \\
x_5' &= -c\mu x_2/r^{(3-\alpha)} + cr^\alpha F_2/x_7 \\
x_6' &= -c\mu x_3/r^{(3-\alpha)} + cr^\alpha F_3/x_7 \\
x_7' &= -cF/(I_{sp}g_0)r^\alpha \\
t' &= cr^\alpha
\end{aligned} \tag{11}$$

We now develop a Taylor model to integrate efficiently the equations above. We will compute a step h for our integrator using the p-th order Taylor formula:

$$\mathbf{x}(t+h) = \sum_{n=0}^{p} \mathbf{x}^{[n]} h^n$$

where $\mathbf{x}^{[n]} = \frac{1}{n!}\mathbf{x}^{(n)}(t)$ is the generalized n-th derivative of $\mathbf{x}$ evaluated in t. The computations of the various

$$\begin{aligned}
u_1 &= x_1, \\
u_2 &= x_2, \\
u_3 &= x_3, \\
u_4 &= x_4, \\
u_5 &= x_5, \\
u_6 &= x_6, \\
u_7 &= x_7, \\
u_8 &= t \\
u_9 &= u_1u_1, \\
u_{10} &= u_2u_2, \\
u_{11} &= u_3u_3, \\
u_{12} &= u_9 + u_{10} + u_{11}, && \text{this is } r^2 \\
u_{13} &= u_{12}^{\frac{\alpha}{2}}, && \text{this is } r^\alpha \\
u_{14} &= u_{12}^{\frac{\alpha-3}{2}}, && \text{this is } 1/r^{(3-\alpha)} \\
u_{15} &= u_{14}u_1, && \text{this is } x_1/r^{(3-\alpha)} \\
u_{16} &= u_{14}u_2, && \text{this is } x_2/r^{(3-\alpha)} \\
u_{17} &= u_{14}u_3, && \text{this is } x_3/r^{(3-\alpha)} \\
u_{18} &= u_{13}/u_7, && \text{this is } r^\alpha/x_7 \\
u_{19} &= u_4u_{13}, && \approx \text{eq. 1} \\
u_{20} &= u_5u_{13}, && \approx \text{eq. 2} \\
u_{21} &= u_6u_{13}, && \approx \text{eq. 3} \\
u_{22} &= -\mu u_{15} + F_1u_{18}, && \approx \text{eq. 4} \\
u_{23} &= -\mu u_{16} + F_2u_{18}, && \approx \text{eq. 5} \\
u_{24} &= -\mu u_{17} + F_3u_{18}, && \approx \text{eq. 6} \\
u_{25} &= -F/(I_{sp}g_0)u_{13}, && \approx \text{eq. 7}
\end{aligned}$$

Figure 12. Taylor integrator auxiliary variables, $\gamma = \alpha/2$ and $\sigma = (\alpha-3)/2$

$\mathbf{x}^{[n]}$ is done in cascade using the automated differentiation approach. To do so, we define a number of auxiliary variables u_i as detailed in Figure 12. Using basic differentiation rules we may then write the identities detailed in Figure 13. Looping over these equations (over n= 0,1,2 ... p) one can compute all necessary coefficients to compute the Taylor formula.

We now have a fast integration scheme able to propagate the motion of a spacecraft subject to a constant thrust in the s-domain. Controlling the parameters c and α we can slow down or accelerate time on a per-segment basis. Implementing Jorba's step size control [9] and using a modern compiler able to make use of vectorization (e.g. gcc $\geq$ 4.5) this Tayor integartion scheme results in a very efficient propagation of the spaceraft dynamics (see Table 2 and Table 3)

REFERENCES

[1] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646–657, 2006.

[2] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm corrigenda for this article is available here. *ACM*

$$
\begin{aligned}
u_1^{[n]} &= x_1^{[n]} \\
u_2^{[n]} &= x_2^{[n]} \\
u_3^{[n]} &= x_3^{[n]} \\
u_4^{[n]} &= x_4^{[n]} \\
u_5^{[n]} &= x_5^{[n]} \\
u_6^{[n]} &= x_6^{[n]} \\
u_7^{[n]} &= x_7^{[n]} \\
u_8^{[n]} &= t^{[n]} \\
u_9^{[n]} &= \textstyle\sum_{j=0}^{n} u_1^{[n-j]} u_1^{[j]} \\
u_{10}^{[n]} &= \textstyle\sum_{j=0}^{n} u_2^{[n-j]} u_2^{[j]} \\
u_{11}^{[n]} &= \textstyle\sum_{j=0}^{n} u_3^{[n-j]} u_3^{[j]} \\
u_{12}^{[n]} &= u_9^{[n]} + u_{10}^{[n]} + u_{11}^{[n]} \\
u_{13}^{[n]} &= \frac{1}{n u_{12}^{[0]}} \textstyle\sum_{j=0}^{n-1} \left[(n\gamma - j(\gamma+1))\, u_{12}^{[n-j]} u_{13}^{[j]} \right] \\
u_{14}^{[n]} &= \frac{1}{n u_{12}^{[0]}} \textstyle\sum_{j=0}^{n-1} \left[(n\sigma - j(\sigma+1))\, u_{12}^{[n-j]} u_{14}^{[j]} \right] \\
u_{15}^{[n]} &= \textstyle\sum_{j=0}^{n} u_{14}^{[n-j]} u_1^{[j]} \\
u_{16}^{[n]} &= \textstyle\sum_{j=0}^{n} u_{14}^{[n-j]} u_2^{[j]} \\
u_{17}^{[n]} &= \textstyle\sum_{j=0}^{n} u_{14}^{[n-j]} u_3^{[j]} \\
u_{18} &= \frac{1}{u_7^{[0]}} \left[u_{13}^{[n]} - \textstyle\sum_{j=1}^{n} u_7^{[j]} u_{18}^{[n-j]} \right] \\
u_{19}^{[n]} &= \textstyle\sum_{j=0}^{n} u_{13}^{[n-j]} u_4^{[j]} \\
u_{20}^{[n]} &= \textstyle\sum_{j=0}^{n} u_{13}^{[n-j]} u_5^{[j]} \\
u_{21}^{[n]} &= \textstyle\sum_{j=0}^{n} u_{13}^{[n-j]} u_6^{[j]} \\
u_{22}^{[n]} &= -\mu u_{15}^{[n]} + F_1 u_{18}^{[n]} \\
u_{23}^{[n]} &= -\mu u_{16}^{[n]} + F_2 u_{18}^{[n]} \\
u_{24}^{[n]} &= -\mu u_{17}^{[n]} + F_3 u_{18}^{[n]} \\
u_{25}^{[n]} &= -F/(I_{sp} g_0) u_{13}^{[n]} \\
x_1^{[n+1]} &= \tfrac{1}{n+1} c u_{19}^{[n]} \\
x_2^{[n+1]} &= \tfrac{1}{n+1} c u_{20}^{[n]} \\
x_3^{[n+1]} &= \tfrac{1}{n+1} c u_{21}^{[n]} \\
x_4^{[n+1]} &= \tfrac{1}{n+1} c u_{22}^{[n]} \\
x_5^{[n+1]} &= \tfrac{1}{n+1} c u_{23}^{[n]} \\
x_6^{[n+1]} &= \tfrac{1}{n+1} c u_{24}^{[n]} \\
x_7^{[n+1]} &= \tfrac{1}{n+1} c u_{25}^{[n]} \\
t^{[n+1]} &= \tfrac{1}{n+1} c u_{13}^{[n]}
\end{aligned}
$$

Figure 13. Taylor series computations.

Transactions on Mathematical Software (TOMS), 13(3):262–280, 1987.

[3] J. David and J.P.K. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

[4] P.E. Gill, W. Murray, and M.A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM journal on optimization*, 12(4):979–1006, 2002.

[5] D. Izzo. 1st act global trajectory optimisation competition: Problem description and summary of the results. *Acta Astronautica*, 61(9):927–939, 2007.

[6] D. Izzo. Global Trajectory Optimization Competition portal, June 2009.

[7] D. Izzo, M. Rucinski, and C. Ampatzis. Parallel global optimisation meta-heuristics using an asynchronous island-model. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2301–2308. IEEE, 2009.

[8] D. Izzo, M. Rucinski, and F. Biscani. The generalized island model. *Parallel Architectures and Bioinspired Algorithms*, pages 151–169, 2012.

[9] A. Jorba and M. Zou. A software package for the numerical integration of odes by means of high-order taylor methods. *Experimental Mathematics*, 14(1):99–117, 2005.

[10] J.A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[11] M. Ruciński, D. Izzo, and F. Biscani. On the impact of the migration topology on the island model. *Parallel Computing*, 36(10):555–571, 2010.

[12] J.R. Scott and M.C. Martini. High-speed solution of spacecraft trajectory problems using taylor series integration. *Journal of Spacecraft and Rockets*, 47:199–202, 2010.

[13] J. A. Sims, P. A. Finlayson, E. A. Rinderle, M. A. Vavrina, and T. D. Kowalkowski. Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design. In *AIAA/AAS Astrodynamics Specialist Conference*, Aug. 2006.

[14] J.N. Strange, D.F. Landau, C. Yam, F. Biscani, and D. Izzo. Near earth asteroids accessible to human exploration in 2020-2035. 42nd annual meeting of the Division for Planetary Sciences of the American Astronomical Society, October 2010.

[15] C. H. Yam, T. T. McConaghy, K. J. Chen, and J. M. Longuski. Preliminary Design of Nuclear Electric Propulsion Missions to the Outer Planets. In *AIAA/AAS Astrodynamics Specialist Conference*, Aug. 2004.

[16] C. H. Yam, T. T. McConaghy, K. J. Chen, and M. Longuski, J. Design of Low-Thrust Gravity-Assist Trajectories to the Outer Planets. In *55th International Astronautical Congress*, Oct. 2004.

[17] C.H. Yam, D. Izzo, and F. Biscani. Towards a high fidelity direct transcription method for optimisation of low-thrust trajectories. International Conference on Astrodynamics Tools and Techniques - ICATT, 2010.