New ideas in Direct Optimization Methods for Interplanetary Trajectory Design

Invited Talk at the Fifth International Meeting on Celestial Mechanics (CELMECV)

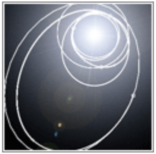
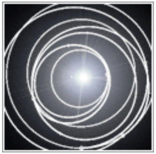
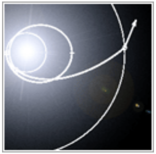
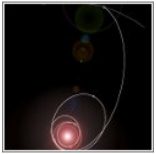
Dario Izzo and Francesco Biscani

Advanced Concepts Team
Institutional Matters and Strategic Studies Office
European Space Agency
www.esa.int/act

07/09/2009

Outline

The Optimal Control Problem

Problem Statement

Optimise:	$\phi(t_s, t_f, \mathbf{x}_s, \mathbf{x}_f) + \int_{t_s}^{t_f} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t)dt$	
Subject to:	$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$	dynamic con.
	$\mathcal{G}(t_s, t_f, \mathbf{x}_s, \mathbf{x}_f) \leq 0$	boundary con.
	$\mathbf{u} \in \mathcal{U}(\mathbf{x}, t)$	propulsion / power con.

Direct Approaches

In a nutshell

1. Transform the optimal control problem (OCP) into a non linear programming problem (NLP)
2. Do not make use of Pontryagin necessary conditions

Some more...

1. There are many ways of creating the NLP Problem Transcription
2. Most transcriptions create highly constrained large scale NLP
3. Allow to describe complex trajectories with fly-bys and different planetocentric phases
4. The efficiency of the NLP solver depends on the transcription adopted
5. Some commonly used NLP solvers: SNOPT, IPOPT, SOCS

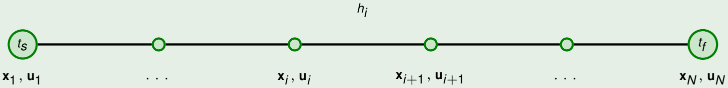
A basic example

Simple Earth-Mars transfer

$$
\begin{array}{rll}
\text{Find:} & \mathbf{x}(t) = [\mathbf{r}(t), \mathbf{v}(t), m(t)], \mathbf{u}(t), t_s, t_f & \\
\text{To maximise:} & m_f = m(t_f) & \\
\text{Subject to:} & \dot{\mathbf{r}} = \mathbf{v}, \dot{\mathbf{v}} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{u}/m & \text{dynamic con.} \\
 & \dot{m} = -\frac{|\mathbf{u}|}{I_{sp} g_0} & \\
 & \mathbf{r}_s = \mathbf{r}_E(t_s), |\mathbf{v}_s - \mathbf{v}_E(t_s)| \le V_\infty & \text{Earth dep.} \\
 & m_s = m_0 & \\
 & \mathbf{r}_f = \mathbf{r}_M(t_f), \mathbf{v}_f = \mathbf{v}_M(t_f) & \text{Mars arr.} \\
 & |\mathbf{u}| \le T_{max} & \text{NEP con.}
\end{array}
$$

A basic example

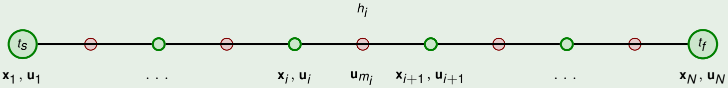
Example (Trapezoidal transcription)



$$\mathbf{x}_{i+1} = \mathbf{x}_i + h_i \mathbf{f}(\mathbf{x}_i , \mathbf{u}_i)$$

A basic example

Example (Runge-Kutta transcription)

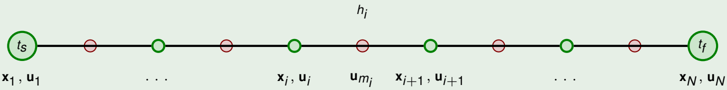


$$\mathbf{x}_{i+1} = \mathbf{x}_i - \frac{h_i}{6}\left(\mathbf{k}_{1_i} + 2\mathbf{k}_{2_i} + 2\mathbf{k}_{3_i} + \mathbf{k}_{4_i}\right)$$

$$\begin{aligned}
\mathbf{k}_{1_i} &= \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i) \\
\mathbf{k}_{2_i} &= \mathbf{f}(\mathbf{x}_i + \frac{\mathbf{k}_{1_i}}{2}, \mathbf{u}_{m_i}) \\
\mathbf{k}_{3_i} &= \mathbf{f}(\mathbf{x}_i + \frac{\mathbf{k}_{2_i}}{2}, \mathbf{u}_{m_i}) \\
\mathbf{k}_{4_i} &= \mathbf{f}(\mathbf{x}_i + \mathbf{k}_{3_i}, \mathbf{u}_{i+1})
\end{aligned}$$

A basic example

Example (Hermite-Simpson transcription)



$$\mathbf{x}_{i+1} = \mathbf{x}_i + \frac{h_i}{6}\left[\mathbf{f}\left(\mathbf{x}_{i+1}, \mathbf{u}_{i+1}\right) + 4\mathbf{f}\left(\mathbf{x}_m, \mathbf{u}_m\right) + \mathbf{f}\left(\mathbf{x}_i, \mathbf{u}_i\right)\right] \quad \text{Simpson Rule}$$

$$\mathbf{x}_m = \frac{1}{2}\left(\mathbf{x}_{i+1} + \mathbf{x}_i\right) + \frac{h_i}{8}\left[\mathbf{f}\left(\mathbf{x}_i, \mathbf{u}_i\right) - \mathbf{f}\left(\mathbf{x}_{i+1}, \mathbf{u}_{i+1}\right)\right] \quad \text{Hermite Interpolation for the states}$$

What matters?

In a nutshell

1. Does the NLP solver converge?
2. In how many iterations?
3. Does the solution found suite the original OCP problem?

Some more...

1. Problem dimension. How many variables? How many constraints?
2. Sparsity of the constraint gradients
3. Sensitivity to the initial guess
4. Numerical error of the implicit integration

What for?

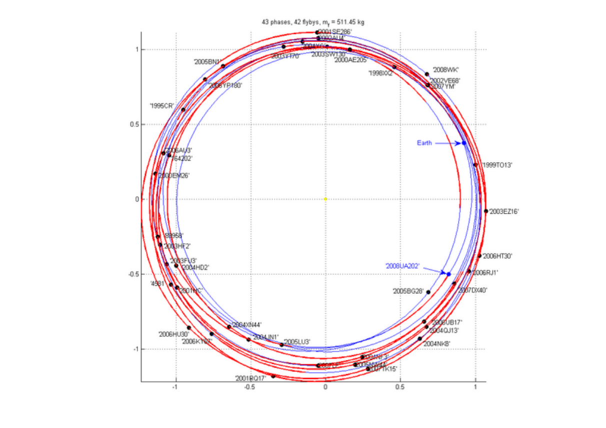


Figure: A multiple (42) asteroid fly-by tour. Ranked third in GTOC4, Credits: Advanced Concepts Team, (ESA)

What for?

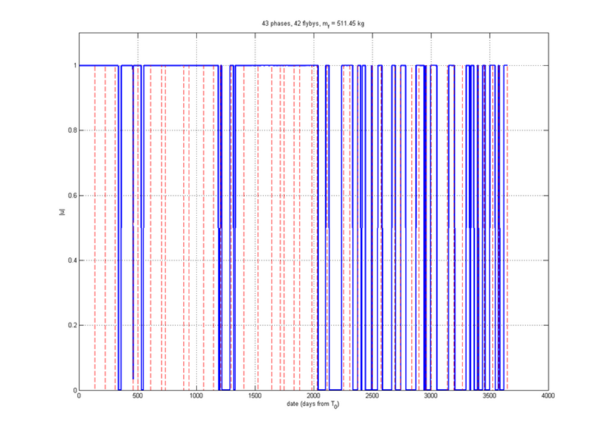
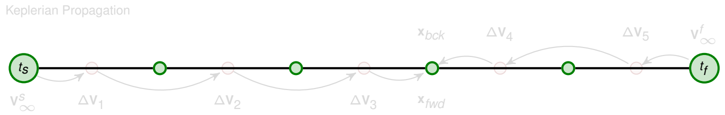


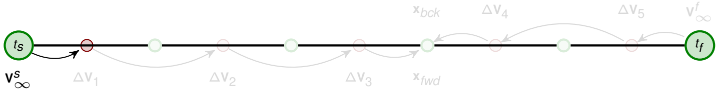
Figure: Optimal control structure

Sims-Flanagan transcription

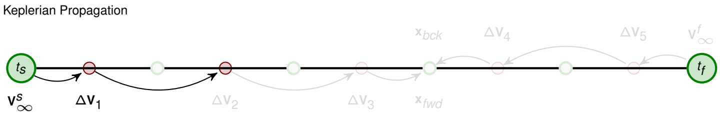


Sims-Flanagan transcription

Keplerian Propagation

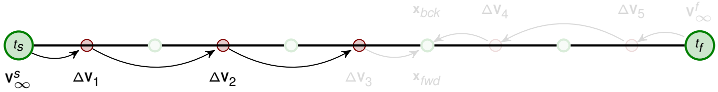


Sims-Flanagan transcription



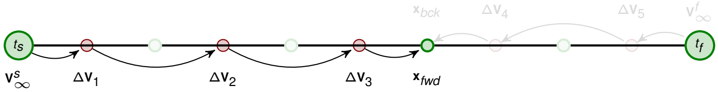
Sims-Flanagan transcription

Keplerian Propagation

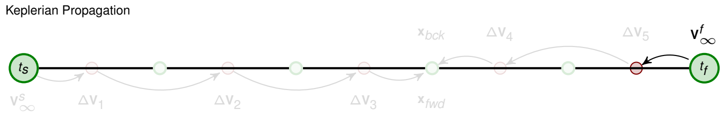


Sims-Flanagan transcription

Keplerian Propagation

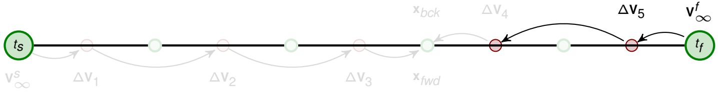


Sims-Flanagan transcription

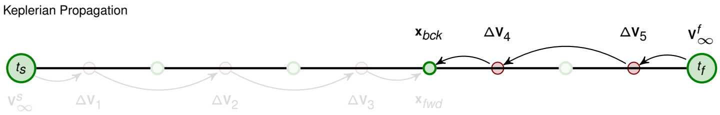


Sims-Flanagan transcription

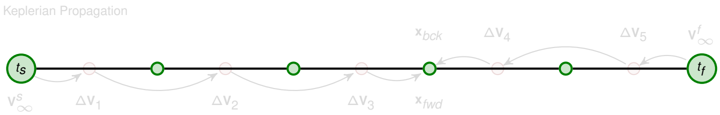
Keplerian Propagation



Sims-Flanagan transcription



Sims-Flanagan transcription



find: $\mathbf{p} = [\mathbf{V}_\infty^{s,f}, \Delta\mathbf{V}_i, t_{s,f}]$

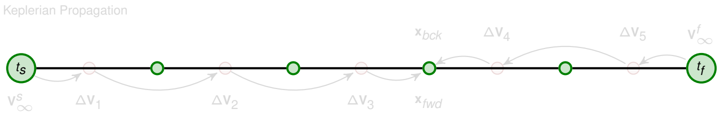
to maximize: $J(\mathbf{p})$

subject to: $\mathbf{x}_{fwd} = \mathbf{x}_{bck}$

$\Delta V_i < T_{max} h_i$

others

Sims-Flanagan transcription



find: $\mathbf{p} = [\mathbf{v}_\infty^{s,f}, \Delta\mathbf{v}_i, t_{s,f}]$

to maximize: $J(\mathbf{p})$

subject to: $\mathbf{x}_{fwd} = \mathbf{x}_{bck}$

$\Delta V_i < T_{max} h_i$

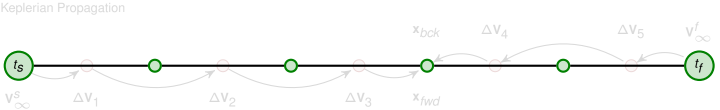
others

"Pros & Cons"

- NLP problem dimension is very small
- Dynamics is explicitly integrated with negligible numerical error (Kepler's equation slover)
- Small computational cost to evaluate the objectives and the constraints.
- Radius of convergence exceptionally good (dynamic is always satisfied)

- Ballistic dynamics needs to be Keplerian
- No automatic differentiation is possible (Kepler equation is solved iteratively for each segment)
- The solution is only an approximation of a feasible trajectory as the thrust is modelled as impulsive

Sims-Flanagan transcription



find: $\mathbf{p} = [\mathbf{V}_\infty^{s,f}, \Delta\mathbf{V}_i, t_{s,f}]$

to maximize: $J(\mathbf{p})$

subject to: $\mathbf{x}_{fwd} = \mathbf{x}_{bck}$

$\Delta V_i < T_{max} h_i$

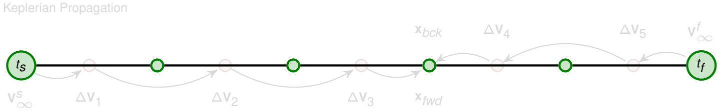
others

"Pros & Cons"

- NLP problem dimension is very small
- Dynamics is explicitly integrated with negligible numerical error (Kepler's equation slover)
- Small computational cost to evaluate the objectives and the constraints.
- Radius of convergence exceptionally good (dynamic is always satisfied)

- Ballistic dynamics needs to be Keplerian
- No automatic differentiation is possible (Kepler equation is solved iteratively for each segment)
- The solution is only an approximation of a feasible trajectory as the thrust is modelled as impulsive

Sims-Flanagan transcription



find: $\mathbf{p} = [\mathbf{V}_\infty^{s,f}, \Delta\mathbf{V}_i, t_{s,f}]$

to maximize: $J(\mathbf{p})$

subject to: $\mathbf{x}_{fwd} = \mathbf{x}_{bck}$

$\Delta V_i < T_{max} h_i$

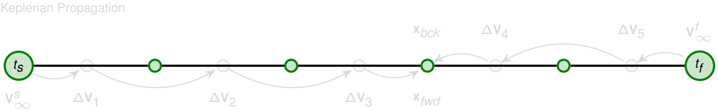
others

"Pros & Cons"

- NLP problem dimension is very small
- Dynamics is explicitly integrated with negligible numerical error (Kepler's equation slover)
- Small computational cost to evaluate the objectives and the constraints.
- Radius of convergence exceptionally good (dynamic is always satisfied)

- Ballistic dynamics needs to be Keplerian
- No automatic differentiation is possible (Kepler equation is solved iteratively for each segment)
- The solution is only an approximation of a feasible trajectory as the thrust is modelled as impulsive

Sims-Flanagan transcription



find: $\mathbf{p} = [\mathbf{V}_\infty^{s,f}, \Delta\mathbf{V}_i, t_{s,f}]$

to maximize: $J(\mathbf{p})$

subject to: $\mathbf{x}_{fwd} = \mathbf{x}_{bck}$

$\Delta V_i < T_{max} h_i$

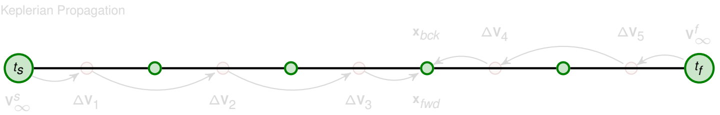
others

"Pros & Cons"

- NLP problem dimension is very small
- Dynamics is explicitly integrated with negligible numerical error (Kepler's equation slover)
- Small computational cost to evaluate the objectives and the constraints.
- Radius of convergence exceptionally good (dynamic is always satisfied)

- Ballistic dynamics needs to be Keplerian
- No automatic differentiation is possible (Kepler equation is solved iteratively for each segment)
- The solution is only an approximation of a feasible trajectory as the thrust is modelled as impulsive

Sims-Flanagan transcription



find: $\mathbf{p} = [\mathbf{V}_\infty^{s,f}, \Delta\mathbf{V}_i, t_{s,f}]$

to maximize: $J(\mathbf{p})$

subject to: $\mathbf{x}_{fwd} = \mathbf{x}_{bck}$

$\Delta V_i < T_{max} h_i$

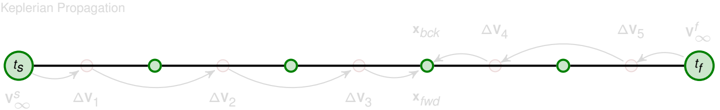
others

"Pros & Cons"

- NLP problem dimension is very small
- Dynamics is explicitly integrated with negligible numerical error (Kepler's equation slover)
- Small computational cost to evaluate the objectives and the constraints.
- Radius of convergence exceptionally good (dynamic is always satisfied)

- Ballistic dynamics needs to be Keplerian
- No automatic differentiation is possible (Kepler equation is solved iteratively for each segment)
- The solution is only an approximation of a feasible trajectory as the thrust is modelled as impulsive

Sims-Flanagan transcription



find: $\mathbf{p} = [\mathbf{V}^{s,f}_\infty, \Delta\mathbf{V}_i, t_{s,f}]$

to maximize: $J(\mathbf{p})$

subject to: $\mathbf{x}_{fwd} = \mathbf{x}_{bck}$

$\Delta V_i < T_{max} h_i$

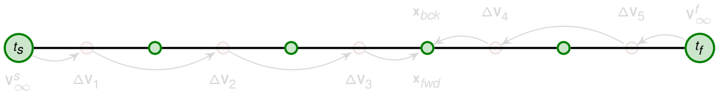
others

"Pros & Cons"

- NLP problem dimension is very small
- Dynamics is explicitly integrated with negligible numerical error (Kepler's equation slover)
- Small computational cost to evaluate the objectives and the constraints.
- Radius of convergence exceptionally good (dynamic is always satisfied)

- Ballistic dynamics needs to be Keplerian
- No automatic differentiation is possible (Kepler equation is solved iteratively for each segment)
- The solution is only an approximation of a feasible trajectory as the thrust is modelled as impulsive

Sims-Flanagan transcription



find: $\mathbf{p} = [\mathbf{V}^{s,f}_\infty, \Delta\mathbf{V}_i, t_{s,f}]$

to maximize: $J(\mathbf{p})$

subject to: $\mathbf{x}_{fwd} = \mathbf{x}_{bck}$
$\Delta V_i < T_{max} h_i$
others

"Pros & Cons"

- NLP problem dimension is very small
- Dynamics is explicitly integrated with negligible numerical error (Kepler's equation slover)
- Small computational cost to evaluate the objectives and the constraints.
- Radius of convergence exceptionally good (dynamic is always satisfied)

- Ballistic dynamics needs to be Keplerian
- No automatic differentiation is possible (Kepler equation is solved iteratively for each segment)
- The solution is only an approximation of a feasible trajectory as the thrust is modelled as impulsive

Hybrid GO on LT trajectories

Let us try.....

- The NLP obtained has only one non-linear constraint
- We attempt to solve the problem using Global optimization algorithms based on meta-heuristics (DE, SA, GA, ES, PSO, ACO)

Table: From Chit Hong Yam et al. ISTS 2009

N=10

	Convergence Rate	Best	Worst
DE	98/100	1372.3	1319.3
SA-AN	100/100	1372.3	1236.5
Lambert	70/100	1372.3	1217.26
Random	63/100	1372.3	1183.79

Hybrid GO on LT trajectories

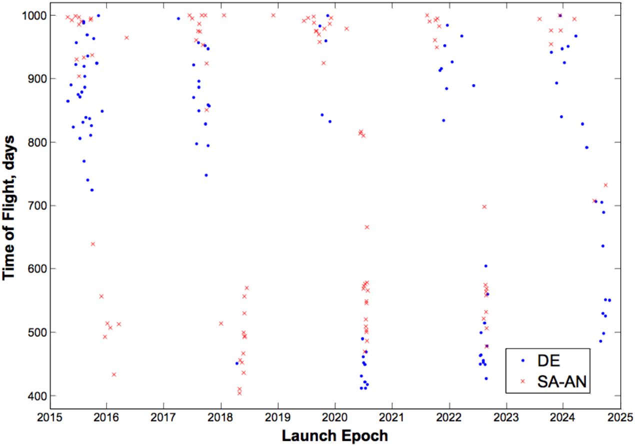


Figure: All trajectories

Hybrid GO on LT trajectories

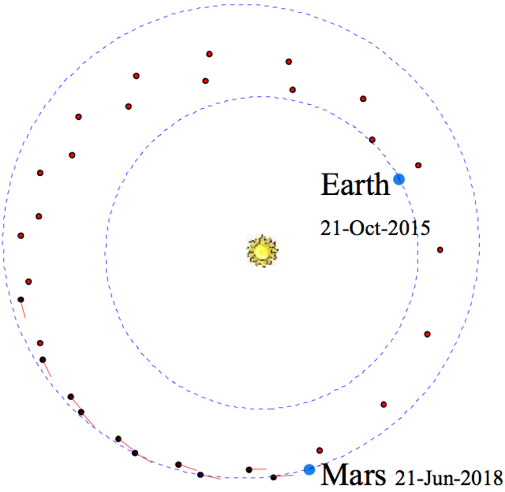


Figure: Visualization of the global optima

Hybrid GO on LT trajectories

- Only 10 segments are enough!!! For N=20, N=30 Best is still 1372.1
- A random initial guess still brings to convergence!!
- DE and SA algorithms work efficiently acting on diverse areas of the search space

Some observed facts....

- GAs are outperformed by others...
- Results need to be confirmed also for multiple phases trajectories, how complex can we go?
- Fully automated approach (no initial guess needed) extending to LT the work done on MGA and MGA-DSM problems

Hybrid GO on LT trajectories

- Only 10 segments are enough!!! For N=20, N=30 Best is still 1372.1
- A random initial guess still brings to convergence!!
- DE and SA algorithms work efficiently acting on diverse areas of the search space

Some observed facts....

- GAs are outperformed by others...
- Results need to be confirmed also for multiple phases trajectories, how complex can we go?
- Fully automated approach (no initial guess needed) extending to LT the work done on MGA and MGA-DSM problems

Hybrid GO on LT trajectories

- Only 10 segments are enough!!! For N=20, N=30 Best is still 1372.1
- A random initial guess still brings to convergence!!
- DE and SA algorithms work efficiently acting on diverse areas of the search space

Some observed facts....

- GAs are outperformed by others...
- Results need to be confirmed also for multiple phases trajectories, how complex can we go?
- Fully automated approach (no initial guess needed) extending to LT the work done on MGA and MGA-DSM problems

Hybrid GO on LT trajectories

- Only 10 segments are enough!!! For N=20, N=30 Best is still 1372.1
- A random initial guess still brings to convergence!!
- DE and SA algorithms work efficiently acting on diverse areas of the search space

Some observed facts....

- GAs are outperformed by others...
- Results need to be confirmed also for multiple phases trajectories, how complex can we go?
- Fully automated approach (no initial guess needed) extending to LT the work done on MGA and MGA-DSM problems

Automated differentiation

What to modify?

- We add to the NLP variables the eccentric anomalies differences between subsequent velocity increments
- Kepler equations are forced in the constraints

To get....

- Automatic differentiation can now be used.
- The problem can be written in an explicit form using metalanguges such as AMPL or GAMS
- A new direct transcription retaining the convergence properties of the original SF

Automated differentiation

What to modify?

- We add to the NLP variables the eccentric anomalies differences between subsequent velocity increments
- Kepler equations are forced in the constraints

To get....

- Automatic differentiation can now be used.
- The problem can be written in an explicit form using metalanguges such as AMPL or GAMS
- A new direct transcription retaining the convergence properties of the original SF

Automated differentiation

What to modify?

- We add to the NLP variables the eccentric anomalies differences between subsequent velocity increments
- Kepler equations are forced in the constraints

To get....

- Automatic differentiation can now be used.
- The problem can be written in an explicit form using metalanguges such as AMPL or GAMS
- A new direct transcription retaining the convergence properties of the original SF

Automated differentiation

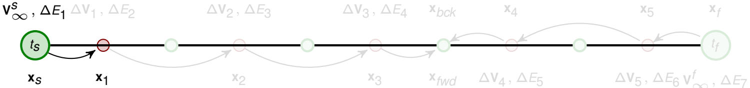
What to modify?

- We add to the NLP variables the eccentric anomalies differences between subsequent velocity increments
- Kepler equations are forced in the constraints

To get....

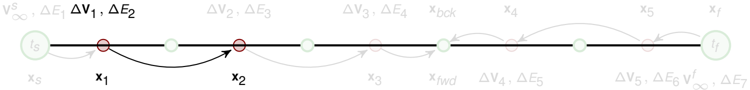
- Automatic differentiation can now be used.
- The problem can be written in an explicit form using metalanguges such as AMPL or GAMS
- A new direct transcription retaining the convergence properties of the original SF

Automated differentiation



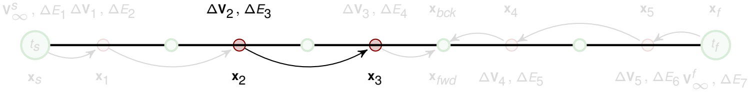
$$\mathbf{x}_1 = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{v}_1 \end{bmatrix} = \begin{bmatrix} F(\Delta E_1) & G(\Delta E_1) \\ F_t(\Delta E_1) & G_t(\Delta E_1) \end{bmatrix} \begin{bmatrix} \mathbf{r}_S \\ \mathbf{v}_S + \mathbf{v}_\infty^S \end{bmatrix}$$

Automated differentiation



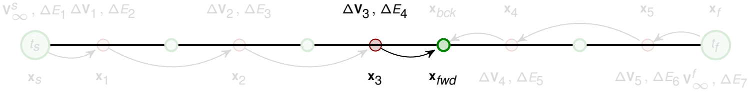
$$\mathbf{x}_2 = \begin{bmatrix} \mathbf{r}_2 \\ \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} F(\Delta E_2) & G(\Delta E_2) \\ F_t(\Delta E_2) & G_t(\Delta E_2) \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{v}_1 + \Delta\mathbf{V}_1 \end{bmatrix}$$

Automated differentiation



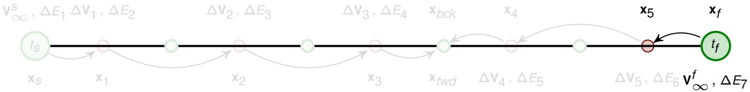
$$\mathbf{x}_3 = \begin{bmatrix} \mathbf{r}_3 \\ \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} F(\Delta E_3) & G(\Delta E_3) \\ F_t(\Delta E_3) & G_t(\Delta E_3) \end{bmatrix} \begin{bmatrix} \mathbf{r}_2 \\ \mathbf{v}_2 + \Delta \mathbf{V}_2 \end{bmatrix}$$

Automated differentiation



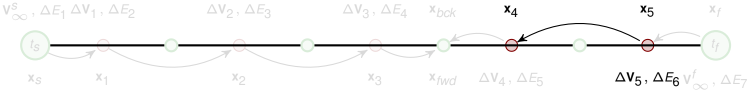
$$\mathbf{x}_{fwd} = \begin{bmatrix} \mathbf{r}_{fwd} \\ \mathbf{v}_{fwd} \end{bmatrix} = \begin{bmatrix} F(\Delta E_4) & G(\Delta E_4) \\ F_t(\Delta E_4) & G_t(\Delta E_4) \end{bmatrix} \begin{bmatrix} \mathbf{r}_3 \\ \mathbf{v}_3 + \Delta\mathbf{V}_3 \end{bmatrix}$$

Automated differentiation



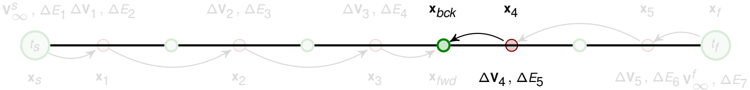
$$\mathbf{x}_5 = \begin{bmatrix} \mathbf{r}_5 \\ \mathbf{v}_5 \end{bmatrix} = \begin{bmatrix} F(\Delta E_7) & G(\Delta E_7) \\ F_t(\Delta E_7) & G_t(\Delta E_7) \end{bmatrix} \begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f + \Delta\mathbf{V}_\infty^f \end{bmatrix}$$

Automated differentiation



$$\mathbf{x}_4 = \begin{bmatrix} \mathbf{r}_4 \\ \mathbf{v}_4 \end{bmatrix} = \begin{bmatrix} F(\Delta E_6) & G(\Delta E_6) \\ F_t(\Delta E_6) & G_t(\Delta E_6) \end{bmatrix} \begin{bmatrix} \mathbf{r}_5 \\ \mathbf{v}_5 + \Delta\mathbf{V}_5 \end{bmatrix}$$

Automated differentiation



$$\mathbf{x}_{bck} = \begin{bmatrix} \mathbf{r}_{bck} \\ \mathbf{v}_{bck} \end{bmatrix} = \begin{bmatrix} F(\Delta E_5) & G(\Delta E_5) \\ F_t(\Delta E_5) & G_t(\Delta E_5) \end{bmatrix} \begin{bmatrix} \mathbf{r}_4 \\ \mathbf{v}_4 + \Delta\mathbf{V}_4 \end{bmatrix}$$

Automated differentiation

find: $\mathbf{p} = [\mathbf{V}^{s,f}_\infty, \Delta \mathbf{V}_i, \Delta E_i, t_{s,f}]$

to maximize: $J(\mathbf{p})$

subject to:

$$\mathbf{x}_{fwd} = \mathbf{x}_{bck}$$

$$\Delta V_i < T_{max} h_i$$

$$\Delta M_i - \Delta E_i - \frac{\sigma_i}{\sqrt{a_i}}(1 - \cos \Delta E_i) + (1 - \frac{r_i}{a_i}) \sin \Delta E_i = 0$$

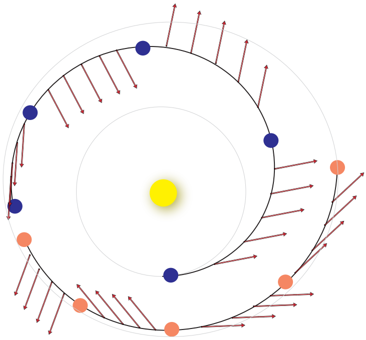
others

Eureka!!

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

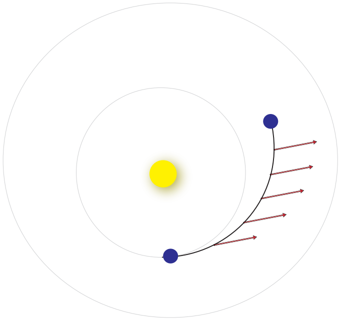


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

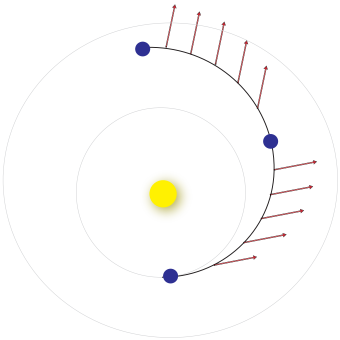


- Each feasible point of the NLP is a "flyable" trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

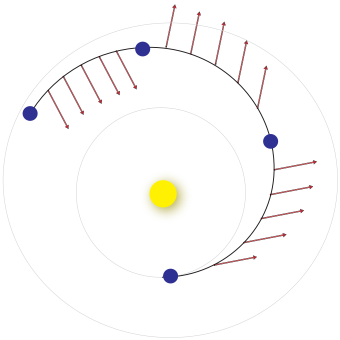


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

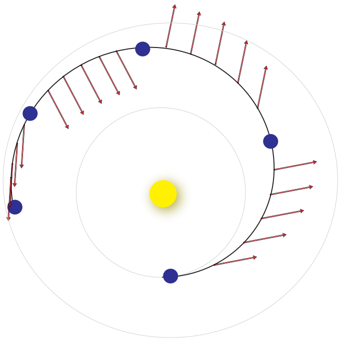


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

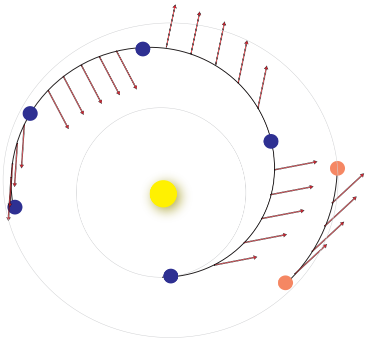


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

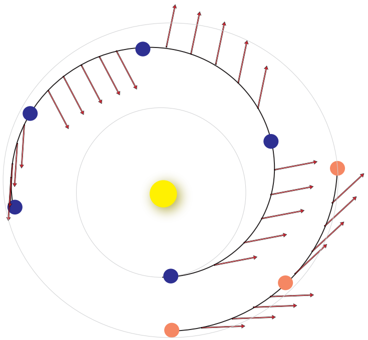


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

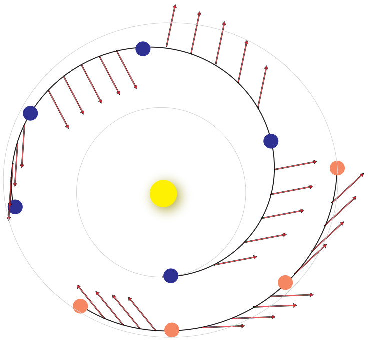


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

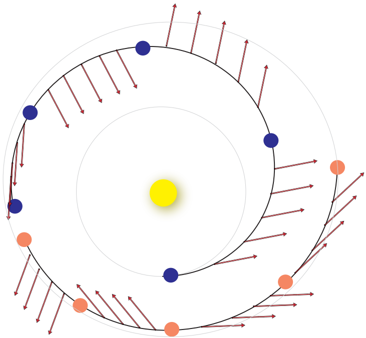


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

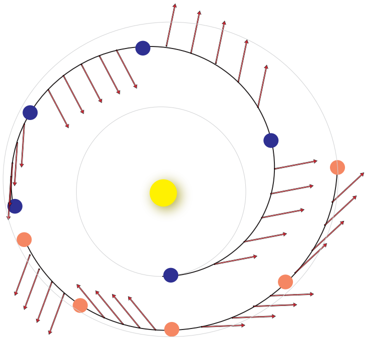


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

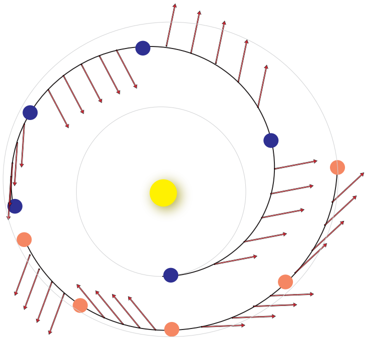


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

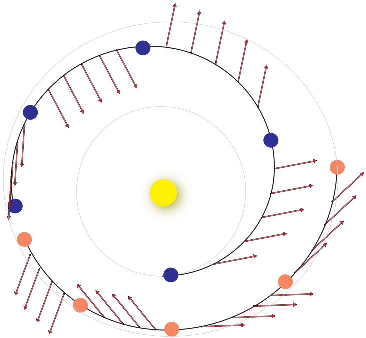


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

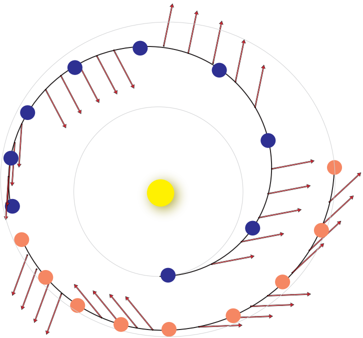


- Each feasible point of the NLP is a “flyable” trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?

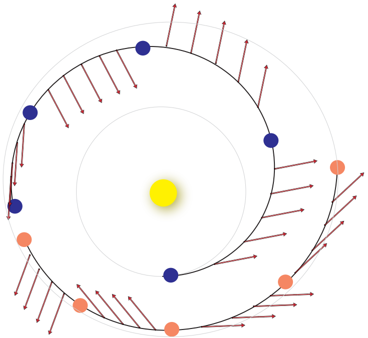


- Each feasible point of the NLP is a "flyable" trajectory.
- The same problem description can be used throughout the trajectory design process
- Nodes can be added to a feasible solution obtaining a feasible solution.

One Step Beyond....

Question....

What happens if we substitute the Keplerian propagation with a fixed constant thrust propagation?



What would help.....

$$\mathbf{x}_{i+1} = \phi(\Delta t, \mathbf{T})\mathbf{x}_i$$

...is the transition matrix for the fixed-thrust problem

Exact Solutions

Dynamical system

- Keplerian force plus external constant force field
- Force field aligned by convention to the $\hat{\mathbf{x}}$ axis
- Acceleration is given by:

$$\mathbf{a} = -\frac{\mathbf{r}}{r^3} + \varepsilon \hat{\mathbf{x}},$$

  where ε is the acceleration induced by the thrust
- Hamiltonian:

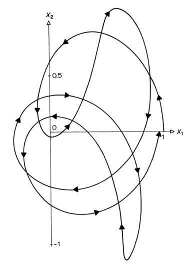
$$\mathcal{H} = \frac{1}{2} v^2 - \frac{1}{r} - \varepsilon x$$

Analyitical approaches

Exact Solutions

- 2D version of the dynamical system is solvable using Levi-Civita variables
- KS theory: closed form solution for the 3D problem using elliptic functions (Kirchgraber, 1970)

Example of bounded motion:



Approximate approaches

Perturbative approach

- Thrust acceleration ε is typically much smaller than gravitational acceleration ($\lesssim 10^{-3}$)
- Idea: use perturbation methods to obtain a (good) approximation of the real motion
- Potential advantages: simpler form, performance, derivatives (for optimization), ...

Perturbative method

Lie series approach

- Canonical transformations depending on a small parameter (Deprit, 1969)
- Well-suited for computer-assisted algebraic manipulation
- Explicit formulation of the transformations
- Hamiltonian in modified Delaunay variables:

$$\mathcal{H} = -\frac{1}{2\Lambda^2} + \varepsilon \mathcal{H}_1 \left(\Lambda, P, Q; \lambda, p, q\right)$$

- Perturbing Hamiltonian $\mathcal{H}_1$ is developed into Poisson series form

Perturbative method - first few terms of $\mathcal{H}_1$

$-\Lambda^2 + \Lambda \mathcal{C}_2^{-1} Q + \frac{1}{2}\Lambda P \mathcal{C}_2 + P\mathcal{C}_2^{-1}Q - \frac{1}{2} PQ - \frac{1}{2} P^2 + \frac{1}{64}P^2\mathcal{C}_2^2$	$\cos(\lambda)$
$-\frac{1}{3} \Lambda^{\frac{1}{2}} P^{\frac{3}{2}} \mathcal{C}_2^{\frac{3}{2}}$	$\cos(3p + 4\lambda)$
$-\Lambda\mathcal{C}_2^{-1}Q - P\mathcal{C}_2^{-1}Q + \frac{1}{2} PQ$	$\cos(\lambda + 2q)$
$-\frac{1}{2}\Lambda^{\frac{3}{2}}P^{\frac{1}{2}}\mathcal{C}_2^{\frac{1}{2}} + \frac{1}{2} \Lambda^{\frac{1}{2}}P^{\frac{1}{2}}\mathcal{C}_2^{-\frac{1}{2}}Q + \frac{1}{4}\Lambda^{\frac{1}{2}}P^{\frac{3}{2}}\mathcal{C}_2^{-\frac{1}{2}} + \frac{3}{8}\Lambda^{\frac{1}{2}}P^{\frac{3}{2}}\mathcal{C}_2^{\frac{3}{2}}$	$\cos(p + 2\lambda)$
$\frac{3}{2}\Lambda^{\frac{1}{2}}P^{\frac{1}{2}}\mathcal{C}_2^{-\frac{1}{2}}Q$	$\cos(p - 2q)$
$-\frac{1}{8} PQ$	$\cos(2p + \lambda - 2q)$
$\frac{3}{2}\Lambda^{\frac{3}{2}}P^{\frac{1}{2}}\mathcal{C}_2^{\frac{1}{2}} - \frac{3}{2}\Lambda^{\frac{1}{2}}P^{\frac{1}{2}}\mathcal{C}_2^{-\frac{1}{2}}Q - \frac{3}{4}\Lambda^{\frac{1}{2}}P^{\frac{3}{2}}\mathcal{C}_2^{-\frac{1}{2}}$	$\cos(p)$
$-\frac{1}{8}\Lambda P\mathcal{C}_2 + \frac{1}{8} PQ + \frac{1}{8} P^2 - \frac{1}{24}P^2\mathcal{C}_2^2$	$\cos(2p + \lambda)$
$-\frac{1}{2}\Lambda^{\frac{1}{2}}P^{\frac{1}{2}}\mathcal{C}_2^{-\frac{1}{2}}Q$	$\cos(p + 2\lambda + 2q)$
$-\frac{3}{128}P^2\mathcal{C}_2^2$	$\cos(4p + 3\lambda)$
...	...

Perturbative method

Work in progress and challenges

- Approximate analytical solution of the fixed-thrust problem
- Assessment of the requirements (accuracy vs performance tradeoff)
- Cope with high-eccentricity orbits: Fourier-Bessel series in which the Taylor series for each $J_n(ne)$ is truncated according to a specified tolerance (Sengupta, 2007)
- Integration within the optimization framework

Final Remarks

- We propose a new direct method that combines computational speed with accuracy in the trajectory description
- The solution of the fixed-thrust problem is central to the efficiency of this new method
- We are trying the use of perturbative methods as an approach to obtain the solution of the fixed-thrust problem
- The final goal is to obtain a transcription that is suitable in each phase of mission design, from the preliminary studies to the operational details