AIAA Guidance, Navigation, and Control Conference

Exact Hybrid Jacobian Computation for Optimal Trajectory Generation via Dual Number Theory

Vincenzo D'Onofrio*, Marco Sagliano†, Yunus E. Arslantas‡

In this paper the effects of the use of the dual-based hybrid Jacobian computation in combination with the Pseudospectral Methods are thoroughly inspected. The dual-step differentiation method is implemented in SPARTAN (SHEFEX-3 Pseudospectral Algorithm for Re-entry Trajectory ANalysis), a tool based on the use of the global Flipped Radau Pseudospectral method for the transcription of optimal control problems. The dual number theory is exploited to provide an exact computation of the Jacobian matrix associated with the NonLinear Programming (NLP) problem to be solved. The dual-step differentiation method is compared to standard differentiation schemes (the central difference and the complex-step approximations) and applied in the solution of two examples of optimal control problem using two different off-the-shelf NLP solvers (SNOPT and IPOPT). Differentiation based on dual number theory is proved to be a valid alternative to the traditional, well-known, differentiation schemes as its use improves, for the problems analysed, the accuracy of the results, especially in combination with SNOPT.

Nomenclature

$\mathbf{B}$	Boundary Conditions
$\mathbf{C}$	Path Constraint
D	Drag Force, lb
$\tilde{D}_{j,k}$	($\mathrm{j^{th}}$,$\mathrm{k^{th}}$) element of Discretization Matrix $\tilde{\mathbf{D}}_{j,k}$
$\tilde{\mathbf{D}}_{j,k}$	Discretization Matrix
$Dual$	Dual part
F	Force, N
f	Residual value vector
$\mathbf{f}$	Function representing the dynamics
g	Gravity acceleration, $\mathrm{ft/s^2}$
$\mathbf{g}$	Function representing the constraints
h	Altitude, ft
i	Imaginary Unit
i	Subscript
$\mathbf{I}_{N_S}$	$N_S \times N_S$
Im	Imaginary part
J	Jacobian Matrix
$\mathbf{Jac}_{Du}$	Dual jacobian
$\mathbf{Jac}_{Ps}$	Pseudospectral jacobian
$\mathbf{Jac}_{Th}$	Theoretical jacobian
L	Lift Force, lb
L_i	$\mathrm{i^{th}}$ Lagrange
LU	Canonical Length
m	Mass, kg (Orbit Raising problem)
MU	Canonical Mass

*M.Sc.student, Department of Aerospace Engineering, University of Naples Federico II.
†Research Engineer, Guidance, Navigation and Control Department, Institute of Space Systems, DLR.
‡PhD candidate, Guidance, Navigation and Control Department, Institute of Space Systems, DLR.

n	Number of collocations nodes
n_c	Number of controls
n_g	Number of constraints
n_s	Number of states
$\mathbf{p}$	Parameter
P_n	Legendre Polynomial of order n
q	Heat rate, BTU/ft^{2}s
r	Radius, ft (Space Shuttle entry problem)
r	Radius, LU (Orbit Raising problem)
Re	Real part
t	Time, s
T	Thrust magnitude, MU LU / TU^2
t_f	Final time, s
TU	Canonical Time
u	Control
$\mathbf{U}_k$	k$^{\text{th}}$ discrete control vector
v	Velocity modulus, ft/s
V_r	Radial speed, LU/TU
V_t	Tangential speed, LU/TU
x	State
X_{NLP}	NLP state vector
$\mathbf{X}_k$	k$^{\text{th}}$ discrete state vector Polynomial
α	Angle of attack, rad
β	Bank angle, rad
γ	Flight-path angle, rad
δ	Thrust angle, rad
ϵ	Dual Unit
ϵ_k	k$^{\text{th}}$ Hyperdual Unit
η	Discrepancy between numerical and analytical derivative
θ	Longitude, rad
τ	Pseudospectral time domain
ϕ	Latitude, rad (Space Shuttle entry problem)
ϕ	True anomaly, rad (Orbit Raising problem)
$\mathbf{\Phi}$	Mayer term of cost function
ψ	Velocity azimuth angle, rad
$\mathbf{\Psi}$	Lagrange term of cost function

Abbreviations

CD	Central Difference
CDn	Central Difference with n points
CS	Complex Step
DS	Dual Step
Du	Dual
FRPM	Flipped Radau Pseudospectral Method
GPM	Gauss Pseudospectral Method
IPOPT	Interior Point Optimizer
LPM	Legendre Pseudospectral Method
NLP	NonLinear Programming
OCP	Optimal Control Problem
Ps	Pseudospectral
RPM	Radau Pseudospectral Method
SHEFEX	SHarp Edge Flying EXperiment
SNOPT	Sparse Nonlinear Optimizer

SPARTAN	SHEFEX-3 Pseudospectral Algorithm for Reenty Trajectory ANalysis
Th	Theoretical
w.r.t.	with respect to

I. Introduction

Nowadays the new, increased capabilities of CPUs have encouraged researchers and engineers towards the investigation of numerical optimization as an analysis and synthesis tool to generate optimal trajectories and the controls to track them. In particular, direct methods are gaining widespread acceptance for solving optimization problems.
Direct methods use gradient-based techniques, and require the computation of the derivatives of the objective function and the constraints of the problem under analysis. The accuracy of these derivatives has a direct impact on the computational accuracy of the solutions. Therefore, the quality of the results and the computation time are strongly affected by the Jacobian matrix describing the discrete, transcribed Optimal-Control Problem (OCP), that is, the resulting NonLinear Programming (NLP) problem.
From this perspective, in this paper we will combine the dual-step differentiation method which provides exact (machine epsilon) first-order derivatives, with Pseudospectral Methods to provide exact Jacobian in the frame of implementing a NLP problem. The work is organized as follows: the transcription process from a generic OCP to the NLP using the Flipped Radau Pseudospectral Methd (FRPM) is described in Sec.II. Section III briefly summarizes the dual number theory and, in particular, it thoroughly analyses the dual-step differentiation method. Section IV focuses on SPARTAN (SHEFEX-3 Pseudospectral Algorithm for Reentry Trajectory ANalysis) and on its hybrid computation of the Jacobian matrix associated with the NLP to be solved. The dual-step differentiation method is integrated in the frame of the hybridization of the entire jacobian matrix. In Sec.V two examples taken from literature[1] are considered to show the results coming from the use of the proposed technique. The optimal-control problems considered are the maximization of the final crossrange of the space shuttle reentry trajectory, and the maximization of the final specific energy in an orbit raising problem. The NLP generated by SPARTAN are solved using two different off-the-shelf NLP solvers (SNOPT and IPOPT), and a comparison of the dual-step method with other differentiation schemes is performed for each of the tools. The results of the simulations in terms of accuracy and CPU time are reported to assess the effects of the use of the Pseudospectral methods in combination with the dual numbers. Finally, in Sec.VI some conclusions and the future outlook are reported.

II. Pseudospectral Methods

In Optimal Control it is desired to determine the inputs to a dynamical system that optimize (i.e., minimize or maximize) a specific performance index (the cost function) while satisfying any constraints on the system. Numerical methods for solving OCPs are divided in two major classes[1] : indirect methods and direct methods. Indirect methods are based on the Pontryagin Maximum Principle, which leads to a multiple-point boundary-value problem. Direct methods, instead, consist in the discretization of the OCP, transcribing it to a nonlinear optimization problem or NonLinear Programming Problem (NLP). It is seen[2] that indirect methods and direct methods emanate from two different philosophies. Indeed, the indirect approach solves the problem indirectly by converting the OCP to a boundary-value problem and, as a result, the optimal solution is found by solving a system of differential equations that satisfies endpoint and/or interior point conditions. On the other hand, using a direct approach, the optimal solution is found by transcribing an infinite-dimensional optimization problem to a finite-dimensional optimization problem.
PseudoSpectral Methods represent a particular area of interest in the frame of the wider class of direct methods. The basic idea behind the pseudospectral methods is, as in the other direct methods, to collocate the differential equations, the cost function and the constraints of the original OCP in a finite number of points to treat them as a set of nonlinear algebraic equations. In detail, in the PS methods, the collocation points are chosen as linear combinations of the roots of Legendre Polynomials and / or their derivatives.[2] In this way, the continuous (infinite-dimensional) OCP is transformed (i.e. transcribed) into a discrete (finite-dimensional) NLP problem, which can be efficiently solved with one of the off-the-shelf, well-known software.
It is possible to distinguish between two subcategories of pseudospectral methods: the symmetrical methods, like the Gauss Pseudospectral Method (GPM) and the Lobatto Pseudospectral Method (LPM), which use

symmetrical distributions of nodes[3–5] and the asymmetrical methods, represented by the Radau Pseudospectral Method (RPM)[6] in its direct and flipped form.
In this paper we focus on SPARTAN, an optimal-control package developed by the German Aerospace Center, which has already been used in literature[7–13] . SPARTAN uses the flipped version of the RPM: the global Flipped Radau Pseudospectral Method (FRPM), based on the flipped distribution of points w.r.t. the classical RPM. It has been shown that for the FRPM, as well as for all the PS methods, the following properties are valid:

- "Spectral" convergence in the case of a smooth problem
- The Runge phenomenon is avoided
- Straightforward implementation
- Sparse structure of the associated NLP problem
- Mapping between the discrete costates of the associated NLP and the continuous costates of the Optimal Control Problem (except for LPM) in virtue of the Pseudospectral Covector Mapping Theorem.[14]

In addition, the FRPM distinguishes itself from the other PS methods by the property of convergence of the costates. Therefore, it is useful to have a look at the FRPM and how it can be conveniently employed to solve OCPs, focusing on the transcription process which defines the corresponding NLP.

A. Discretization of the OCP

An optimal-control problem is posed formally as follows.[1] Determine the *state* (equivalently, the *trajectory* or *path*), $\mathbf{x}(t) \in \mathbb{R}^n$, the *control* $\mathbf{u}(t) \in \mathbb{R}^m$, the vector of static parameters $\mathbf{p} \in \mathbb{R}^q$, the initial time, $t_0 \in \mathbb{R}$, and the terminal time, $t_f \in \mathbb{R}$ (where $t \in [t_0, t_f]$ is the independent variable) that optimize the cost function

$$J[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}] \tag{1}$$

subject to the dynamic constraints (i.e., differential equation constraints),

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t); \mathbf{p}], \tag{2}$$

the path constraints

$$\mathbf{C}_{min} \leq \mathbf{C}[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}] \leq \mathbf{C}_{max}, \tag{3}$$

and the boundary conditions

$$\mathbf{B}_{min} \leq \mathbf{B}[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}] \leq \mathbf{B}_{max}. \tag{4}$$

The objective function (1), in a Bolza formulation of the OCP, can be expressed as follows

$$J = \mathbf{\Phi}[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f; \mathbf{p}] + \int_{t_0}^{t_f} \mathbf{\Psi}[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}]dt \tag{5}$$

where $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are the Mayer and the integrand of the Lagrange terms, respectively.
The differential equations (2) describe the dynamics of the system while the objective (1) is the performance index, which can be considered as a measure of the "quality" of the trajectory. When it is desired to minimize the performance index, a lower value of J is preferred; conversely, when it is desired to maximize the performance index, a larger value of J is preferred. In the PS methods, and specifically, in the FRPM, the transcription of the OCP as NLP is based on the choice of some "basis" functions to represent the continuous variables. The states and the controls are discretized as follows.

$$x(t_i) \cong X_i, \qquad i \in [0, n] \tag{6}$$

$$u(t_j) \cong U_j, \qquad j \in [1, n]. \tag{7}$$

The continuous states and controls are therefore approximated with polynomials which interpolate the values in the nodes X_i and U_i,

$$x(t) \cong \sum_{i=0}^{n} X_i L_i(t) \tag{8}$$

$$u(t) \cong \sum_{i=1}^{n} U_i L_i(t) \tag{9}$$

where

$$L_i(t) = \prod_{j=0, j\neq i}^{n} \frac{t - t_j}{t_i - t_j} \tag{10}$$

and t_j are the roots of linear combinations of Legendre Polynomials $P_n(t)$ and $P_{n-1}(t)$. Formal definitions of the Legendre Polynomials can be found in[15] , while an example of Legendre Polynomials of order $0-5$ is plotted in Fig. 1.

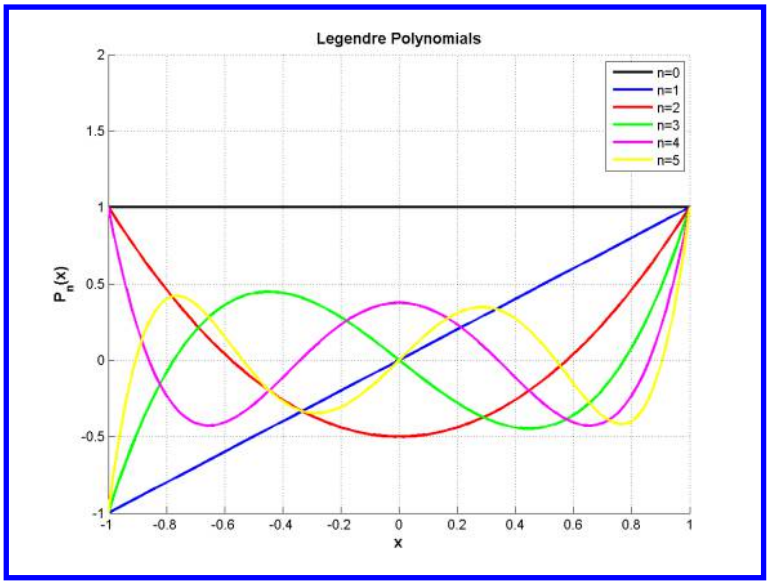

Figure 1: Legendre Polynomials of order 0-5.

The difference in the indexing in the first and the second relationships in Eq. (9) is due to the distinctions between discretization and collocation. While the discretization includes (in the FRPM) the initial point, the collocation does not. Hence, the controls will be approximated with a polynomial having a lower order and the NLP problem will not provide the initial values for the controls. These can be, in some cases, part of the initial set of known inputs, otherwise they can be extrapolated from the generated polynomial interpolating the N values of controls in the collocation nodes. The entire information related to the states and the controls is enclosed in their nodal values. The boundaries defined in Eq. (4) are evaluated in the same nodes used for the collocation of the differential equations, the cost function, and the constraints. The transcribed NLP problem can therefore be defined.

B. Transcription as NLP

The resulting NLP can be summarized as follows. Minimize (maximize) the cost function J

$$J = \Psi\left[t_f, \mathbf{X}_f, \mathbf{U}_f\right] + \frac{t_f - t_0}{2} \sum_{i=0}^{n} w_i \Phi\left[t_i, \mathbf{X}_i, \mathbf{U}_i(\tau)\right] \tag{11}$$

subject to the nonlinear algebraic constraints,

$$\mathbf{F}_i = \tilde{\mathbf{D}} \cdot \mathbf{X} - \frac{t_f - t_0}{2} \mathbf{f}(t_i, \mathbf{X}_i, \mathbf{U}_i) = \mathbf{0} \tag{12}$$

representing the differential equations, and to the boundary and path constraints Eqs. (13)-(15).

$$\mathbf{C}_{min,i} \leq \mathbf{C}\left(t_i, \mathbf{X}_i, \mathbf{U}_i\right) \leq \mathbf{C}_{max,i} \tag{13}$$

$$\mathbf{B}_{min,i} \leq \mathbf{B}(\mathbf{X}_i) \leq \mathbf{B}_{max,i} \tag{14}$$

$$i = 1, \ldots, n \tag{15}$$

The elements w_i which appear in the relationship Eq. (11) are the Gauss-Radau quadrature weights.[15] The matrix $\tilde{\mathbf{D}}$ is the discrete differential operator, and will be described in Sec.IV, while the term $\frac{t_f - t_0}{2}$ is a scale factor related to the transformation between the physical time domain t, defined between t_0 and t_f, and the pseudospectral time domain $\tau \in (-1, 1]$. The use of this pseudospectral domain is connected to the property of orthogonality of the Legendre polynomials, which allow to remove the Runge phenomenon.[3–6] The affine transformation between the time and pseudospectral time domains is given by

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{16}$$

$$\tau = \frac{2}{t_f - t_0}t - \frac{t_f + t_0}{t_f - t_0} \tag{17}$$

The NLP is therefore completely characterized.

III. Dual Numbers

A. Definition

In linear algebra, the dual numbers extend the real numbers by adjoining one new element ϵ with the property $\epsilon^2 = 0$ (ϵ is nilpotent). The collection of dual numbers forms a particular two-dimensional commutative associative algebra over the real numbers[16] . Every dual number has the form

$$z = a + b\epsilon \tag{18}$$

with a and b uniquely determined real numbers and, in particular,

$$\begin{aligned} a &= real(z) \qquad \text{Real Part} \\ b &= dual(z) \qquad \text{Dual Part} \end{aligned}$$

Dual numbers extend the real numbers in a similar way to the complex numbers. Indeed, as the dual numbers, the complex numbers adjoin a new element i, for which $i^2 = -1$, and every complex number has the form $z = a + bi$ where a and b are real numbers.
The definition given in Eq. (18) relies on the idea that $\epsilon^2 = 0$ with $\epsilon \neq 0$. This may not be mathematically possible; $\epsilon^2 = 0$ may require $\epsilon = 0$. For this reason, these numbers were also called "fake numbers"[17] with reference to their similarity with imaginary numbers and to acknowledge that this type of number may not formally exist.
Using matrices, dual numbers can be represented also in matrix form as

$$\epsilon = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad ; \qquad \mathbf{z} = \mathbf{a} + \mathbf{b}\epsilon = \begin{pmatrix} a & b \\ 0 & a \end{pmatrix}.$$

It is easy to see that the matrix form satisfies all the properties of the dual numbers.
In order to implement the dual numbers, algebraic operations on these numbers should be properly defined. It is important to underline that the dual number algebra is a non-division algebra; given two dual numbers, division is possible only if the real part of the divisor is different from zero.
The dual numbers have been implemented in MATLAB as a new class of numbers[18,19] , using operator overloading. The class includes definitions for standard algebraic operations, logical comparison operations, and other more general functions such as the exponential or the trigonometric functions. This class definition file allows a real-valued analysis code to be easily converted to operate on dual numbers by just changing the variable type declarations, while the structure of the code remains unchanged. The use of the dual numbes allow to compute exact first derivatives, as it will be explained in the next Sec.

B. Dual-Step Differentiation Method

The dual-step differentiation method uses the dual numbers to provide exact first order derivatives.
Consider the Taylor series of a function $f(x)$ for $x \in \mathbb{R}$ for a given perturbation value a.

$$f(x + a) = f(x) + af'(x) + \frac{1}{2!}a^2 f''(x) + \frac{a^3 f'''(x)}{3!} + \ldots. \tag{19}$$

If we assume that the perturbation a is the dual number

$$a = a_1\epsilon \qquad \text{with} \qquad \epsilon^2 = 0 \qquad \text{and} \qquad \epsilon \neq 0 \tag{20}$$

so that $a^2 = 0$, $a^3 = 0$, ..., the Taylor series in Eq. (19) truncates exactly at the first-derivative term, yielding the properties of the approximation that we are seeking:

$$f(x + a) = f(x) + a_1 f'(x)\epsilon. \tag{21}$$

So, to get $f'(x)$ it is necessary to simply read off the ϵ component and divide by a_1, yielding the dual-step first derivative formula:

$$f'(x) = \frac{Dual[f(x + a)]}{a_1}. \tag{22}$$

This formula clearly shows the advantages of the use of the dual-step differentiation method over the central difference and the complex-step approximations.[20] Indeed, since the dual-step derivative approximation does not involve a difference operation and no terms of the Taylor series are ignored, this formula is subject neither to truncation error, nor to round-off error. There is no need to make the step size small and the simplest choice is $a_1 = 1$, which eliminates the need to divide by the step size. Therefore, using the dual-step method, the error between numerical and analytical derivative ($\eta = |f' - f'_{ref}|/|f'_{ref}|$) is machine zero regardless of the selected step size, as illustrated in Fig. 2.

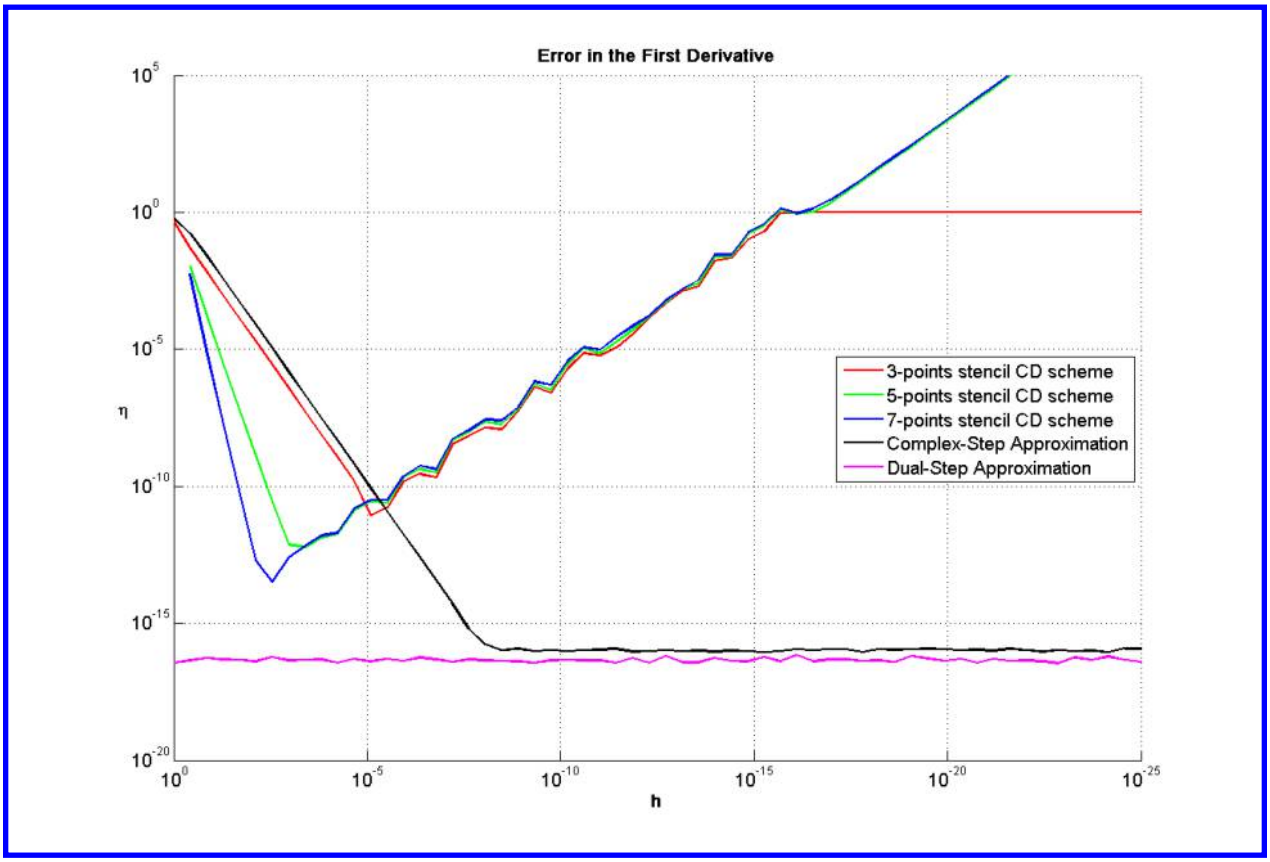

Figure 2: Error in the first derivative, function $f(x) = \frac{1}{x^2}$.

Considering the central difference (CD) and the complex-step approximations, instead, Fig. 2 shows that, as h decreases, the error decreases according to the order of the truncation error of the method. However, after a certain value of h (below a value of about 10^{-2} for the 7-points CD, 10^{-4} for 5-points CD, 10^{-5} for 3-points CD), the error for the central difference approximations tends to grow, while the error for the complex-step approximation continuously decreases. This shows the effect of the round-off error, which affects the central differences but not the first derivative complex-step approximation.

From the inspection of Eq. (21) it is possible to observe that each function extended in the dual plane "hides" its derivative in its dual part. Indeed, the mathematics of these numbers are such that, when operations are carried out on the real part of the number, derivative information for those operations is formed and stored in the non-real part of the number. The disadvantage is a larger computational cost and, in addition, the need of working with analytical functions (e.g., no look-up tables).

C. Hyper-Dual Numbers for n-order exact derivatives

Hyper-dual numbers are a larger dimensional extension of dual numbers in a similar way that the quaternions are a larger dimensional extension of ordinary complex numbers. They have been successfully used to solve optimization problems in the domain of CFD computations[21] . A hyper-dual number is of the form

$$x = x_1 + x_2\epsilon_1 + x_3\epsilon_2 + x_4\epsilon_1\epsilon_2. \tag{23}$$

It has one real part and three non-real parts with the following properties

$$\epsilon_1^2 = \epsilon_2^2 = (\epsilon_1\epsilon_2)^2 = 0, \tag{24}$$

where

$$\epsilon_1 \neq \epsilon_2 \neq \epsilon_1\epsilon_2 \neq 0 \tag{25}$$

or in other words

$$\epsilon_1 = \sqrt{0} \neq 0 \tag{26}$$

$$\epsilon_2 = \sqrt{0} \neq 0 \tag{27}$$

$$\epsilon_1\epsilon_2 = \sqrt{0} \neq 0 \tag{28}$$

Hyper-dual numbers can be used to compute exact first- and second-order derivatives to evaluate Gradients and Hessians for optimization methods. Indeed, the Taylor series for a function with a hyper-dual step truncates exactly at the second-derivative term, yielding

$$f(x + h_1\epsilon_1 + h_2\epsilon_2 + 0\epsilon_1\epsilon_2) = f(x) + h_1 f'(x)\epsilon_1 + h_2 f'(x)\epsilon_2 + h_1 h_2 f''(x)\epsilon_1\epsilon_2. \tag{29}$$

The larger order terms are all zero by the definition of $\epsilon_1^2 = \epsilon_2^2 = (\epsilon_1\epsilon_2)^2 = 0$, so there is no truncation error. The first and second derivatives are the leading terms of the non-real parts, meaning that if $f'(x)$ is desired simply look at the ϵ_1 or ϵ_2 part and divide by the appropriate step and if $f''(x)$ is desired look at the $\epsilon_1\epsilon_2$ part:

$$f'(x) = \frac{\epsilon_1 part[f(x + h_1\epsilon_1 + h_2\epsilon_2 + 0\epsilon_1\epsilon_2)]}{h_1} \tag{30}$$

$$f'(x) = \frac{\epsilon_2 part[f(x + h_1\epsilon_1 + h_2\epsilon_2 + 0\epsilon_1\epsilon_2)]}{h_2} \tag{31}$$

$$f''(x) = \frac{\epsilon_1\epsilon_2 part[f(x + h_1\epsilon_1 + h_2\epsilon_2 + 0\epsilon_1\epsilon_2)]}{h_1 h_2}. \tag{32}$$

The derivative calculations are not even subject to subtractive cancellation error so the use of hyper-dual numbers results in first and second derivative calculations that are exact, regardless of the step size.
The real part returns the original function evaluated in the real argument $Re(x)$, and it is mathematically impossible for the derivative calculations to affect the real part. Indeed, the use of this new number system to compute the first and second derivative involves converting a real-valued function evaluation to operate on these alternative types of numbers. Then, the derivatives are computed by adding a perturbation to the non-real parts and evaluating the modified function.
An example of use of hyperdual numbers is reported in Fig. 3, which shows the difficulty of computing accurate second derivatives using traditional methods. To compute exact second derivatives it is necessary to employ the hyper-dual numbers.

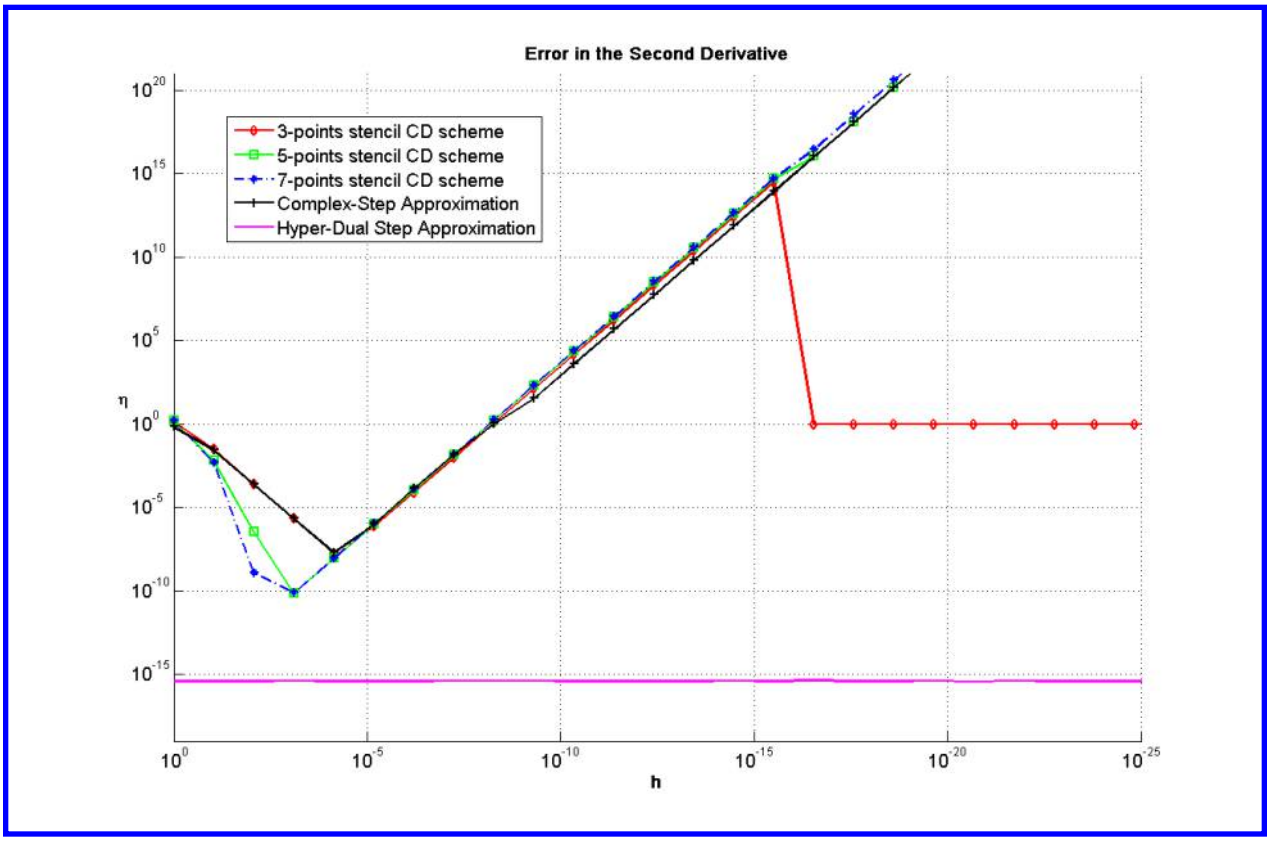

Figure 3: Error in the second derivative, function $f(x) = \frac{1}{x^2}$.

Methods for computing exact larger derivatives can be created by using more non-real parts. For instance, to produce n^{th} derivatives, n^{th} order hyper-dual numbers would be used. These n^{th} order hyper-dual numbers have n components $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ and all of their combinations. If only the first derivatives are needed, first order hyper-dual numbers would be used: the dual numbers.

IV. Hybridization of Jacobian

Let us now consider the general structure of the Jacobian associated with the NLP problem deriving from the application of FRPM. The hybridization described in[7] can be now merged with the framework provided by the Dual Number Theory to compute exact jacobian matrices.
In the most general case, considering n_s states, n_c controls, n_g constraints, n collocation points and unknown final time t_f, the Jacobian associated with the transcription of an autonomous system of equations (as in the examples here treated) will be expressed as a matrix having the following dimensions

$$\dim(J) = [n \cdot (n_s + n_g) + 1] \times [(n+1) \cdot n_s + n \cdot n_c + 1]\,. \tag{33}$$

In order to maintain a consistence between the states and the controls associated with each node, the following order for the NLP variable is proposed:

$$\mathbf{X}_{NLP} = \left\{ \begin{array}{c|cc|cc|cc|c} \mathbf{X}_0 & \mathbf{X}_1 & \mathbf{U}_1 & \mathbf{X}_2 & \mathbf{U}_2 & .. & .. & \mathbf{X}_n & \mathbf{U}_n & t_f \end{array} \right\}^T \tag{34}$$

We can observe how the initial control $\mathbf{U_0}$ does not appear in Eq. (34). This is due to the choice of the FRPM as transcription method instead of the traditional RPM. The initial control indeed can be extrapolated once the NLP is solved. Since the Jacobian is by definition the matrix representing the partial derivatives of a given set of functions (i.e. our NLP constraints) w.r.t. their variables, this set and its order must be defined.

For the NLP, they are all the constraints defined during the transcription of the problem, that is, the cost function J, the dynamics $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_n\}$, and, when defined, the constraints $\mathbf{G} = \{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_n\}$.

$$\mathbf{C}(\mathbf{X}_{NLP}) = \left\{ \; J \; \middle| \; \mathbf{f}_1 \quad \mathbf{f}_2 \quad ... \quad \mathbf{f}_n \; \middle| \; \mathbf{g}_1 \quad \mathbf{g}_2 \quad ... \quad \mathbf{g}_n \; \middle| \right\}^T \tag{35}$$

The Jacobian deriving from these definitions is reported in Eq. (36):

$$\mathbf{J} = \left[\frac{\partial \mathbf{C}}{\partial \mathbf{X}_{NLP}}\right] = \left[\begin{array}{c|ccc|cc|cc|ccc|c}
\frac{\partial J}{\partial \mathbf{X}_0} & \frac{\partial J}{\partial \mathbf{X}_1} & \frac{\partial J}{\partial \mathbf{U}_1} & \frac{\partial J}{\partial \mathbf{X}_2} & \frac{\partial J}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial J}{\partial \mathbf{X}_n} & \frac{\partial J}{\partial \mathbf{U}_n} & \frac{\partial J}{\partial t_f} \\
\frac{\partial \mathbf{f}_1}{\partial \mathbf{X}_0} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{f}_1}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{U}_n} & \frac{\partial \mathbf{f}_1}{\partial t_f} \\
\frac{\partial \mathbf{f}_2}{\partial \mathbf{X}_0} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{f}_2}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{U}_n} & \frac{\partial \mathbf{f}_2}{\partial t_f} \\
.. & .. & .. & .. & .. & .. & .. & .. & .. & .. \\
\frac{\partial \mathbf{f}_n}{\partial \mathbf{X}_0} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{f}_n}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{U}_n} & \frac{\partial \mathbf{f}_n}{\partial t_f} \\
\frac{\partial \mathbf{g}_1}{\partial \mathbf{X}_0} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{g}_1}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{U}_n} & \frac{\partial \mathbf{g}_1}{\partial t_f} \\
.. & .. & .. & .. & .. & .. & .. & .. & .. & .. \\
\frac{\partial \mathbf{g}_n}{\partial \mathbf{X}_0} & \frac{\partial \mathbf{g}_n}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{g}_n}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{g}_n}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{g}_n}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{g}_n}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{g}_n}{\partial \mathbf{U}_n} & \frac{\partial \mathbf{g}_n}{\partial t_f}
\end{array}\right] \tag{36}$$

This Jacobian matrix can be computed numerically in different ways (e.g. with the classical finite-differences schemes, or using the complex-step derivative technique). This is not the best approach since it does not consider the theoretical knowledge contained in the definition of the discrete operator $\mathbf{D}$, nor does it take full advantage from the intrinsic sparsity associated with the use of Pseudospectral methods. Instead, the approach followed in[7] is here considered. The results will be extended to the use of the dual numbers. On this purpose, let us then express the Jacobian as sum of three different contributions.

$$\mathbf{J} = \mathbf{Jac}_{Ps} + \mathbf{Jac}_{Du} + \mathbf{Jac}_{Th} \tag{37}$$

We can now analyze each of these terms and how to compute them.

A. PseudoSpectral Jacobian

This part of the Jacobian matrix is intrinsically related to the use of the FRPM. More specifically, it can be seen as the contribution to the Jacobian and to the constraints represented in Eq. (36) given by the use of the discrete differential matrix $\tilde{\mathbf{D}}$. In the frame of the discretization of the dynamics, it represents the term

$$\tilde{\mathbf{D}} \cdot \mathbf{X} \tag{38}$$

From a pure algebraic point of view, the differential operator can be seen as a set of linear combinations of the nodal values of each of the states. The PseudoSpectral Jacobian is entirely defined once the matrix $\tilde{\mathbf{D}}$ is computed. More explicitly, the Pseudospectral Jacobian Matrix can be defined as follows

$$\mathbf{Jac}_{Ps} = \left[\begin{array}{ccccc}
 & & \mathbf{O}_{1\times[(n+1)\cdot n_s + n\cdot n_c + 1]} & & \\
\tilde{\mathbf{D}}_{\mathbf{1,0}} & .. & .. & \tilde{\mathbf{D}}_{\mathbf{1,n}} & \\
.. & .. & .. & .. & \mathbf{O}_{[n\cdot(n_s+n_g)+1\times 1]} \\
\tilde{\mathbf{D}}_{\mathbf{n,0}} & .. & .. & \tilde{\mathbf{D}}_{\mathbf{n,n}} & \\
 & & \mathbf{O}_{n_g\times[(n+1)\cdot n_s + n\cdot n_c + 1]} & &
\end{array}\right] \tag{39}$$

where

$$\tilde{\mathbf{D}}_{\mathbf{i,j}} = \tilde{D}_{i,j} \cdot \mathbf{I}_{N_S}, \; j \in [0, n] \tag{40}$$

and $\mathbf{I}_{N_S}$ is the identity matrix of dimension N_S. The elements $\tilde{D}_{i,j}$ are the time derivative of the polynomials P defined in Eq. (10), evaluated in the collocation nodes. The Pseudospectral Jacobian can then be entirely computed just once, before the beginning of the real optimization process. Moreover, the accuracy of its computation is a consequence of how good the estimate of the roots of the Legendre-Radau Polynomials is, and not of the errors given by the approximation due to the use of numerical differentiation techniques.

B. Dual Jacobian

The Dual Jacobian refers to the cost functions, the differential equations and the path constraints which appear in the NLP problem defined in Eqs. (1)-(3). While Central Differences and Complex Step provide approximated derivatives for these terms, the use of Dual Number Theory permits to compute zero-epsilon derivatives. The only limit for the use of this technique is the same associated with the use of the complex-step, that is, the need to have analytical functions, i.e. no look-up tables are allowed. In case we are dealing with analytical functions, it is possible to compute their contribution to the Jacobian (i.e. considering the matrix $\mathbf{D}$ equal to $\mathbf{0}$), excluding the last column,

$$\mathbf{Jac}_{Du} = \left[\frac{\partial \mathbf{C}}{\partial \mathbf{X}_{NLP}}\right]_{\mathbf{D}=\mathbf{0}} = -k_t \begin{bmatrix} & \frac{\partial J}{\partial \mathbf{X}_1} & \frac{\partial J}{\partial \mathbf{U}_1} & \frac{\partial J}{\partial \mathbf{X}_2} & \frac{\partial J}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial J}{\partial \mathbf{X}_n} & \frac{\partial J}{\partial \mathbf{U}_n} & \\ & \frac{\partial \mathbf{f}_1}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{f}_1}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{U}_n} & \\ & \frac{\partial \mathbf{f}_2}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{f}_2}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{U}_n} & \\ \mathbf{O}_{[n\cdot(n_s+n_g)+1\times n_s]} & .. & .. & .. & .. & .. & .. & .. & .. & \mathbf{O}_{[n\cdot(n_s+n_g)+1\times 1]} \\ & \frac{\partial \mathbf{f}_n}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{f}_n}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{U}_n} & \\ & \frac{\partial \mathbf{g}_1}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{g}_1}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{U}_n} & \\ & .. & .. & .. & .. & .. & .. & .. & .. & .. \\ & \frac{\partial \mathbf{g}_n}{\partial \mathbf{X}_1} & \frac{\partial \mathbf{g}_n}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{g}_n}{\partial \mathbf{X}_2} & \frac{\partial \mathbf{g}_n}{\partial \mathbf{U}_2} & .. & .. & \frac{\partial \mathbf{g}_n}{\partial \mathbf{X}_n} & \frac{\partial \mathbf{g}_n}{\partial \mathbf{U}_n} & \end{bmatrix} \tag{41}$$

where k_t is the time mapping factor defined as $\frac{t_f - t_0}{2}$ for the elements related to the functions $\mathbf{f}$, and 1 for all the other terms of $\mathbf{Jac_{Du}}$. Each of the element of $\mathbf{Jac}_{Du}$ can be rewritten in dual form. We can therefore write

$$\mathbf{Jac}_{Du} = -k_t Dual\left[\mathbf{C}(\mathbf{X}_{NLP} + \epsilon)\right]_{\mathbf{D}=\mathbf{0}} \tag{42}$$

The differentiation operation becomes then an evaluation of the single elements of $\mathbf{C}(\mathbf{X}_{NLP})$ in dual sense, and the extraction of the dual part.

C. Theoretical Jacobian

Finally, a third contribution, the Theoretical Jacobian, arises in case we deal with problems having an unknown final time. The NLP state vector will then have a further variable, that is t_f. In this case, the Jacobian associated with this term is proportional to the output of the continuous functions in virtue of the time mapping reported in the relationships (16), (17).

$$\mathbf{Jac}_{Th} = -\frac{1}{2}\begin{bmatrix} & 0 \\ & \mathbf{f}_1 \\ & \mathbf{f}_2 \\ \mathbf{O}_{[n\cdot(n_s+n_g)+1]\times[(n+1)\cdot n_s + n\cdot n_c+1]} & .. \\ & \mathbf{f}_n \\ & \mathbf{O}_{n\cdot n_g\times 1} \end{bmatrix} \tag{43}$$

The hybridization of the Jacobian matrix makes the computation of the NLP problems solution more accurate, as no approximations are taken, except those associated with the transcription process. Hence, significant CPU time is saved when solving the NLP problem. To see the results, let us consider two significant examples, already used as examples in literature. The solutions have been generated using a number of nodes between 100 and 300. Cases with 100 nodes were solved with a cold start (e.g., no meaningful initial guess was provided to the solver). This solution was used as initial guess for the cases with 200 and 300 nodes to reduce the CPU time.

V. Numerical Examples

A. Space Shuttle Reentry Problem

The problem of the construction of the reentry trajectory for the Space Shuttle is a classic example of an optimal-control problem and it is of significant practical interest. The motion of the vehicle is defined by

the following set of differential algebraic equations[1] :

$$\dot{h} = v\,sin(\gamma), \tag{44}$$

$$\dot{\phi} = \frac{v\,sin(\psi)\,cos(\gamma)}{r\,cos(\theta)}, \tag{45}$$

$$\dot{\theta} = \frac{v}{r}\,cos(\gamma)\,cos(\psi), \tag{46}$$

$$\dot{v} = -\frac{D}{m} - g\,sin(\gamma), \tag{47}$$

$$\dot{\gamma} = \frac{L}{m\,v}cos(\beta) + cos(\gamma)\left(\frac{v}{r} - \frac{g}{v}\right), \tag{48}$$

$$\dot{\psi} = \frac{L}{m\,v\,cos(\gamma)}\,sin(\beta) + \frac{v}{r\,cos(\theta)}cos(\gamma)\,sin(\psi)\,sin(\theta), \tag{49}$$

The state variables are $\mathbf{x} = (h, \phi, \theta, v, \gamma, \psi)^T$ where h is the altitude (ft), ϕ is the longitude (rad), θ is the latitude (rad), v is the velocity (ft/sec), γ is the flight path angle (rad) and ψ is the azimuth (rad), and the control variables are $\mathbf{u} = (\alpha, \beta)^T$ where α is the angle of attack (rad) and β is the bank angle (rad).
The reentry trajectory begins at an altitude where the aerodynamic forces are quite small with the following initial conditions:

$$\begin{aligned} h_0 &= 260000\text{ft}, & v_0&=25600\text{ft/s}, \\ \phi_0 &= 0^\circ, & \gamma_0 &= -1^\circ \\ \theta_0 &= 0^\circ, & \psi_0 &= 90^\circ. \end{aligned}$$

The final point on the reentry trajectory occurs at the unknown final time t_f. The goal is to choose the control variables $\alpha(t)$ and $\beta(t)$ so that the final cross-range is maximized, which is equivalent to maximizing the final latitude $\theta(t_f)$. So, the cost function can be defined as follows:

$$J = \theta(t_f). \tag{50}$$

Furthermore, an upper bound on the aerodynamic heating of 70 BTU/ft^2/s is imposed. The aerodynamic heating has been calculated as $q = q_a q_r$ where In this case the Jacobian has all the three contributions. The

$$\begin{aligned} q_a &= c_0 + c_1\hat{\alpha} + c_2\hat{\alpha}^2 + c_3\hat{\alpha}^3, & q_r &= 1770\sqrt{\rho}(0.0001v)^{3.07}, \\ c_0 &= 1.0672181, & \rho &= \rho_0 exp[-h/h_r], \\ c_1 &= -0.19213774 \times 10^{-1}, & \rho_0 &= 0.002378 \\ c_2 &= 0.21286289 \times 10^{-3} & h_r &= 23800, \\ c_3 &= -0.10117249 \times 10^{-5}, & \hat{\alpha} &= 180\alpha/\pi. \end{aligned}$$

OCP has been implemented and solved with SPARTAN using different differentiation schemes: the 3-points stencil central difference scheme, the 5-points stencil central difference scheme, the 7-points stencil central difference scheme, the complex-step derivative approach and the dual-step derivative method.
Figures 4 and 5 illustrate the time histories of the states and the controls which are associated with the solution obtained using the dual-step derivative method and a number of nodes equal to 100. The red points are the discrete states and controls obtained with SPARTAN, and the blue lines are the interpolating functions approximating the continuous variables. The states and the controls are fully consistent with the solution obtained by Betts.[1] Figure 6 shows the discrepancy between the optimized states obtained with SPARTAN and the states obtained by propagating control inputs obtained with SPARTAN according to Runge-Kutta 45 scheme. The maximum discrepancy is about 60 ft in terms of altitude h, and 1 ft/s in terms of velocity, with a maximum percentage difference of about 0.3%. The groundtrack of the optimal trajectory and the optimal value for the final latitude are reported in Fig. 7. The crossrange is maximized as expected, with a final value equal to 30.71 deg.

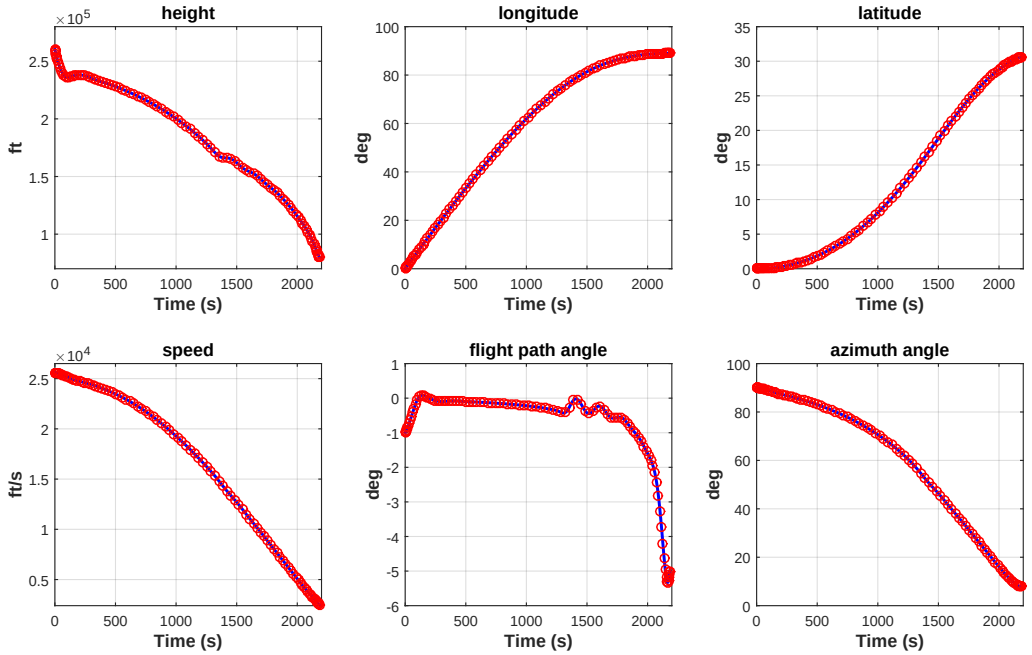

Figure 4: States Evolution for the Space Shuttle Reentry Problem.

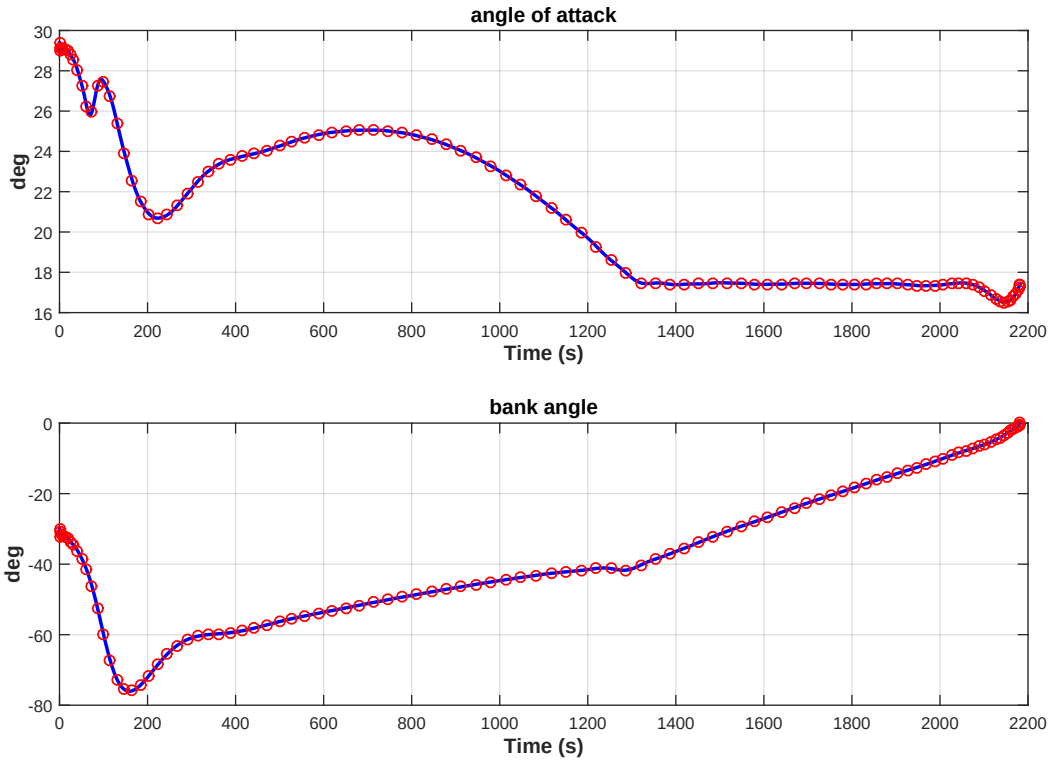

Figure 5: Controls Evolution for the Space Shuttle Reentry Problem.

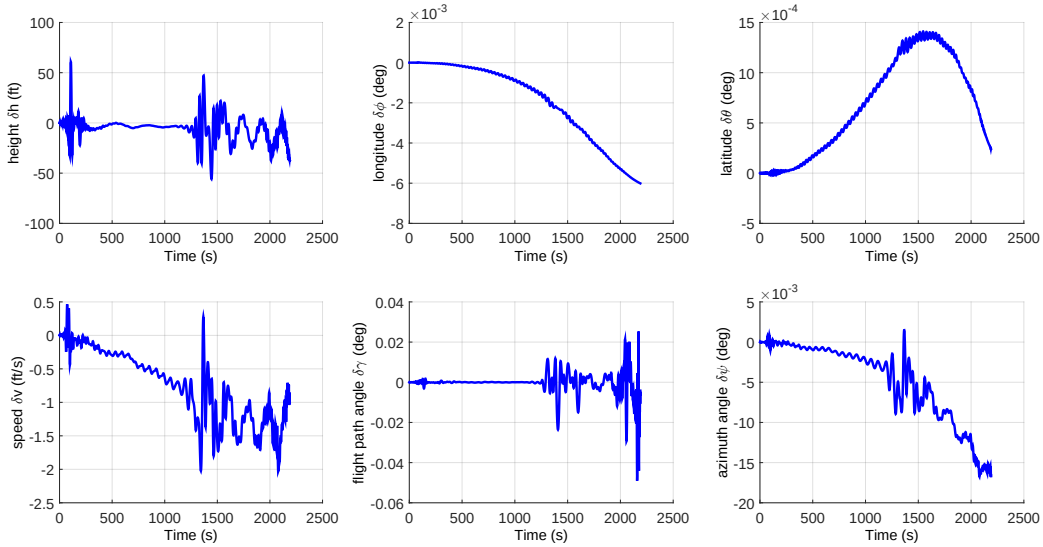

Figure 6: Discrepancy between optimized and propagated solutions for the Space Shuttle Reentry Problem.

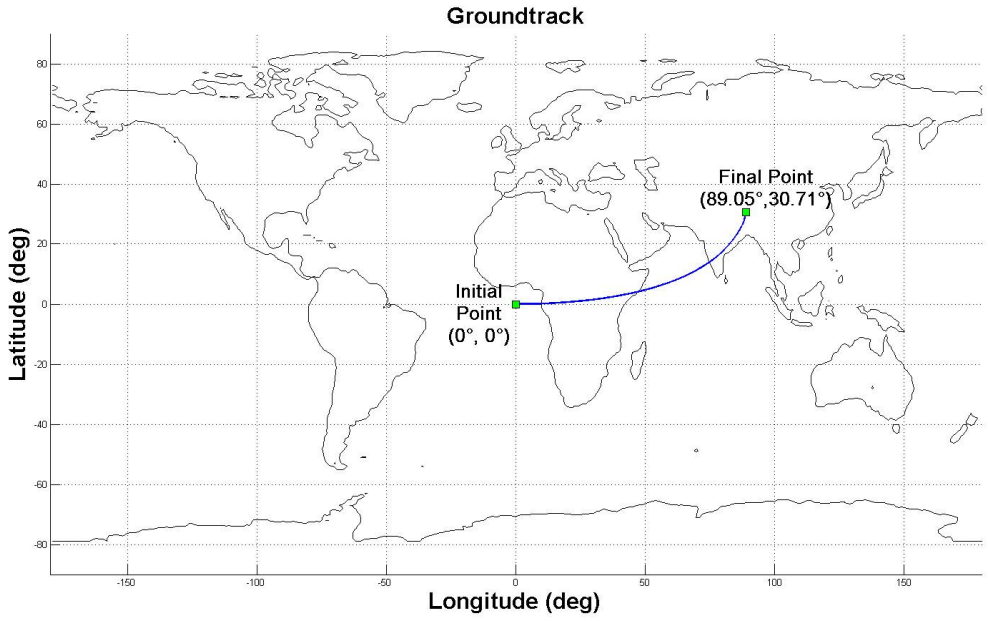

Figure 7: Space Shuttle Reentry Problem - Groundtrack of Trajectory Optimizing Final Crossrange.

Figures 8 and 9 illustrate the trend of the altitude mean error (in logarithmic scale) between optimized and propagated solutions as a function of the number of the nodes. When the dual-step differentiation method is integrated within the FRPM, the "spectral" convergence of the solution, typical of the pseudospectral methods, is verified, and shows a smoother behavior w.r.t. to the convergence obtained when other approximated techniques for numerical differentiation are used (in the example reported here the Complex-Step is used as comparison). As expected, an increase in the number of nodes brings the mean error asymptotically to zero. Tables 1 and 2 summarize the results obtained with SPARTAN using five different differentiation schemes: the 3-points central difference scheme, the 5-points central difference scheme, the 7-points central difference scheme, the complex-step derivative approach and the dual-step differentiation method. In the first table SNOPT (Sparse Nonlinear OPTimizer) is used to solve the NLP problem, while in the second table IPOPT (Interior Point OPTimizer) is used. The results in terms of accuracy and CPU time show that the effects of the use of the dual-step derivative method, as well as of the other schemes, in combination with the PS methods are strongly influenced by the number of the nodes used to discretized the problem under analysis, and by the NLP solver which has been selected. It is interesting to observe that for this specific problem and when SNOPT is used, only for solutions associated with 200 and 300 nodes a better accuracy is paid in terms of CPU time, while when solutions associated with 100 nodes are computed, the Dual-step approach provides slightly improved results and a smaller CPU time required to solve the problem. When IPOPT is used, results obtained with the dual-step are similar to those obtained with the Complex-step for the case associated with 100 nodes, but there is a significant increase in the CPU time required to solve the problem (about 65% more). When the number of nodes is increased, there are no significant differences in terms of accuracy when several differentiation schemes are implemented in SPARTAN, but different burden, expressed in terms of CPU time.

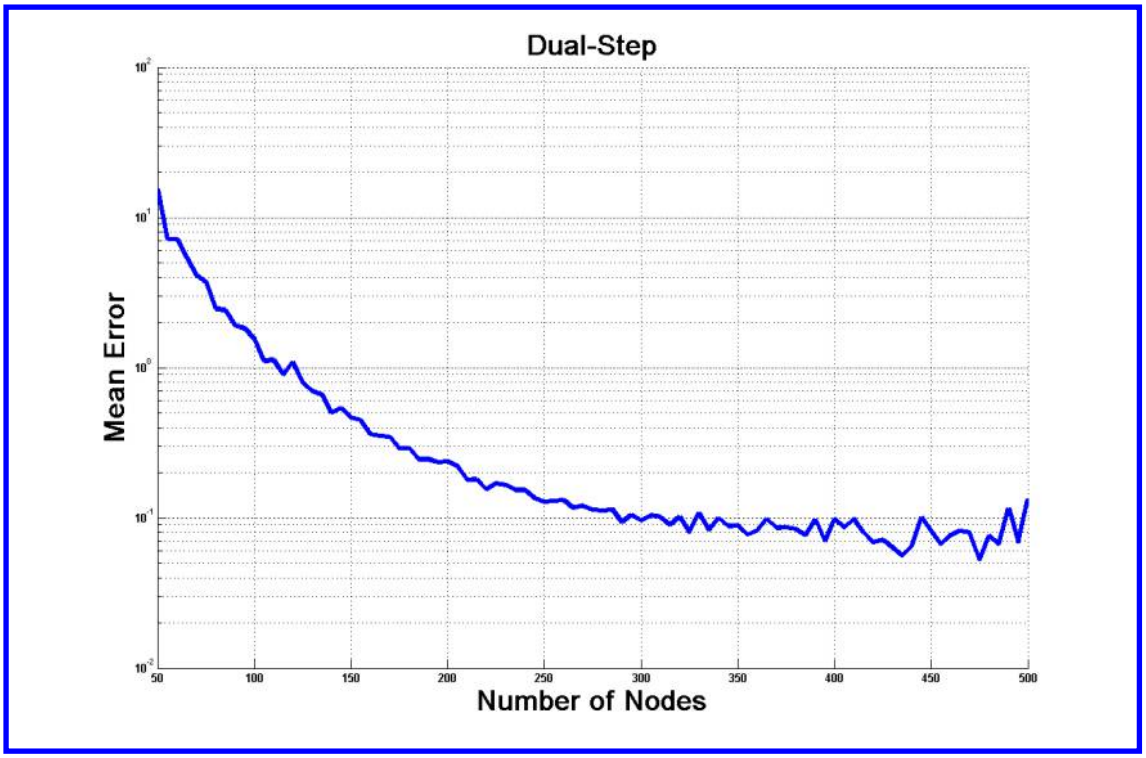

Figure 8: Altitude mean error - "Spectral" Convergence of the solution obtained with Dual-step approach.

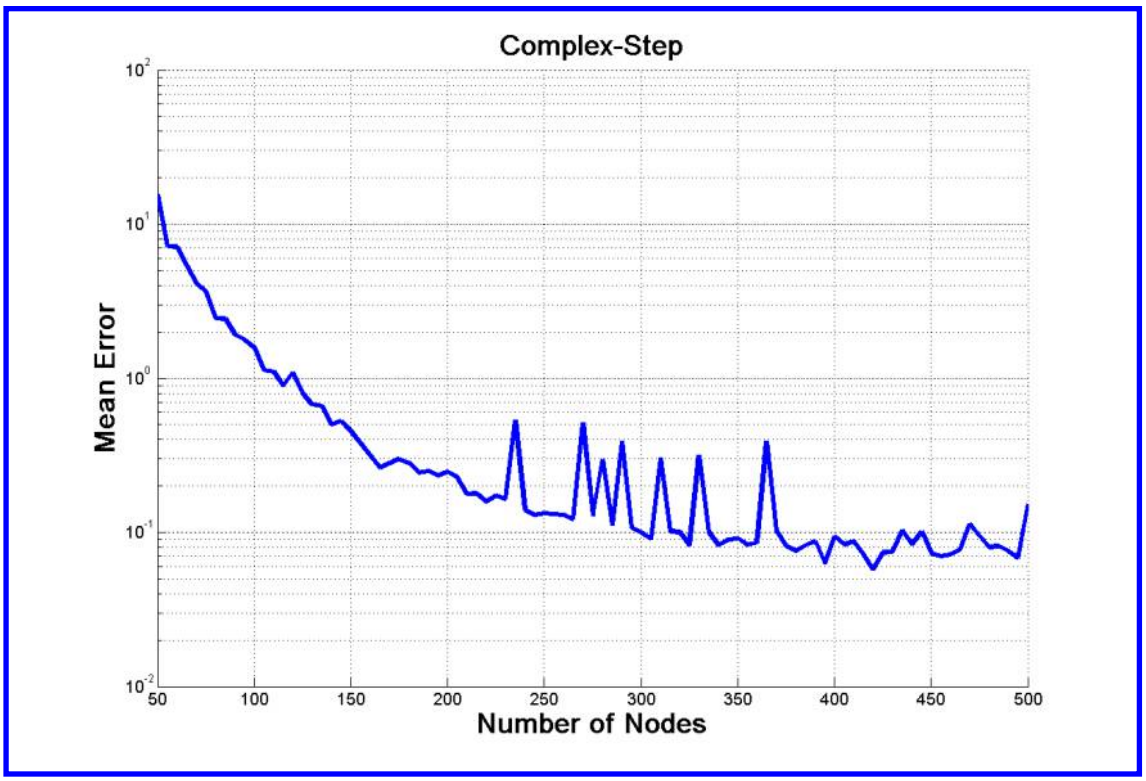

Figure 9: Altitude mean error - "Spectral" Convergence of the solution obtained with Complex-step approach.

SNOPT	100 Nodes				200 Nodes				300 Nodes			
	Mean Error	Max Error	Iter.	CPU (sec)	Mean Error	Max Error	Iter.	CPU (sec)	Mean Error	Max Error	Iter.	CPU (sec)
CD3	8.647	60.47	13	21.38	1.367	13.23	2	62.18	0.4963	4.378	3	248.29
CD5	8.558	60.35	13	16.21	1.363	13.44	10	83.79	0.5106	4.380	3	267.60
CD7	8.571	60.45	13	17.39	1.357	13.28	10	82.27	0.5171	4.408	3	268.38
CS	8.612	60.44	13	15.20	1.383	13.15	3	63.40	0.5337	4.441	3	249.23
DS	8.533	60.45	13	15.37	1.348	13.05	10	80.87	0.4902	4.350	3	270.06

Table 1: Accuracy and CPU Time comparison for the Space Shuttle Problem (SNOPT).

IPOPT	100 Nodes				200 Nodes				300 Nodes			
	Mean Error	Max Error	Iter.	CPU (sec)	Mean Error	Max Error	Iter.	CPU (sec)	Mean Error	Max Error	Iter.	CPU (sec)
CD3	6.287	62.78	483	472.1	7.402	26.52	3094	$1.2 \cdot 10^4$	13.917	48.33	1706	$2.8 \cdot 10^4$
CD5	6.306	62.80	767	675.8	7.547	27.01	6635	$2.7 \cdot 10^4$	13.911	48.32	1168	$1.9 \cdot 10^4$
CD7	6.280	62.87	687	628.85	6.867	24.63	1971	$8.1 \cdot 10^3$	13.920	48.31	1448	$2.5 \cdot 10^4$
CS	6.526	62.84	439	313.6	6.983	25.24	3044	$1.2 \cdot 10^4$	13.910	48.33	4177	$7.2 \cdot 10^4$
DS	6.285	62.76	574	518.7	6.932	21.93	3493	$1.3 \cdot 10^4$	13.857	47.14	2425	$4.1 \cdot 10^4$

Table 2: Accuracy and CPU Time comparison for the Space Shuttle Problem (IPOPT).

B. Orbit Raising Problem

This problem has been proposed more than once in literature[22] and deals with the maximization of the specific energy of a low-thrust spacecraft orbit transfer, in a given fixed time. It can be expressed considering an orbit subject to the following dynamics (expressed in canonical units)[7] ,

$$\frac{dr}{dt} = V_r \tag{51}$$

$$\frac{d\phi}{dt} = \frac{V_t}{r} \tag{52}$$

$$\frac{dV_r}{t} = \frac{V_t^2}{r} - \frac{\mu}{r^2} + Tsin(\delta) \tag{53}$$

$$\frac{dV_t}{dt} = -\frac{V_r V_t}{r} + Tcos(\delta) \tag{54}$$

where V_r and V_t are the radial and the tangential speed, respectively. r is the radius, ϕ is the true anomaly, δ is the thrust angle, T is the specific force, assumed to be constant and equal to 0.01. μ is the normalized gravitational parameter and δ is the angle between the direction of the thrust and the tangential velocity. The state variables are $\mathbf{x} = (r, \phi, V_r, V_t)^T$ whereas, the control variable is $u = \delta$.
The goal is to maximize the total specific energy at the final time, considering that the rocket engine provides a constant thrust acceleration to the spacecraft. Thus, the cost function can be defined as follows:

$$J = \frac{1}{r(t_f)} - \left\{ \frac{1}{2} \left[V_r^2(t_f) + V_t^2(t_f) \right] \right\}. \tag{55}$$

Since the final time is known, the Jacobian here will only consist of the pseudospectral and dual contributions. Figures 10, 11 and 13 illustrate states, controls and the discrepancy between optimized and propagated solutions. These results are obtained using the dual-step derivative approach, with a number of nodes equal to 100, and are fully consistent with those proposed in literature. The cost function is identical to the value obtained by Herman and Conway[22] . In this case the maximum percentage discrepancy is about 0.0014%. Figure 12 shows the trajectory optimizing the final orbit energy. As it can been seen, the optimal trajectory is a multi-revolution spiral away from attracting body[22] which has its center of mass located at the origin. As done in the previous example, Figs. 14 and 15 illustrate the trend of the altitude mean error (in logarithmic scale) between optimized and propagated solutions as a function of the number of the nodes. Also in this case the spectral convergence has been numerically verified, but no particular differences between the methods have been observed.

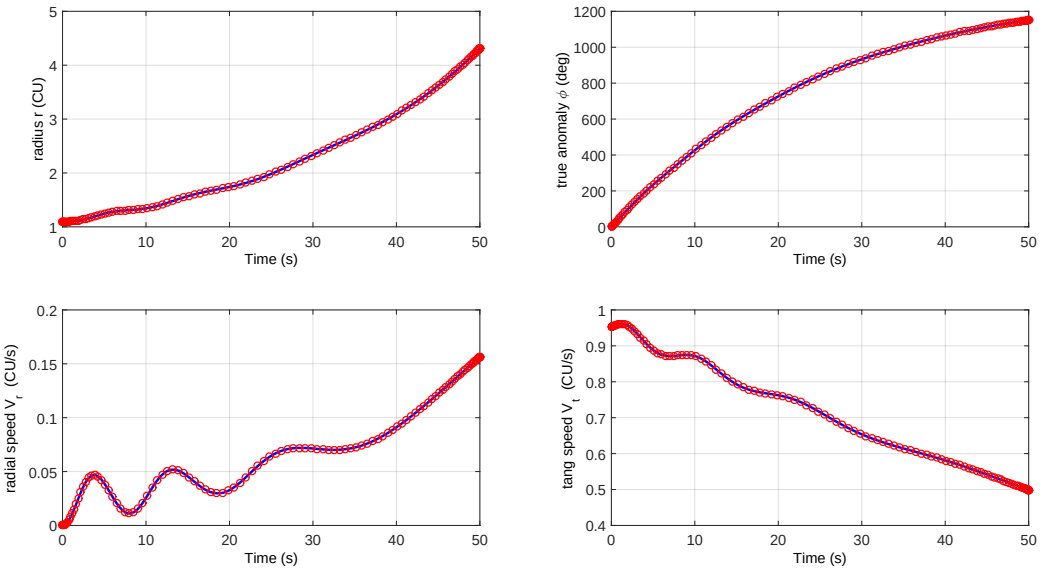

Figure 10: States Evolution for the Orbit Raising Problem.

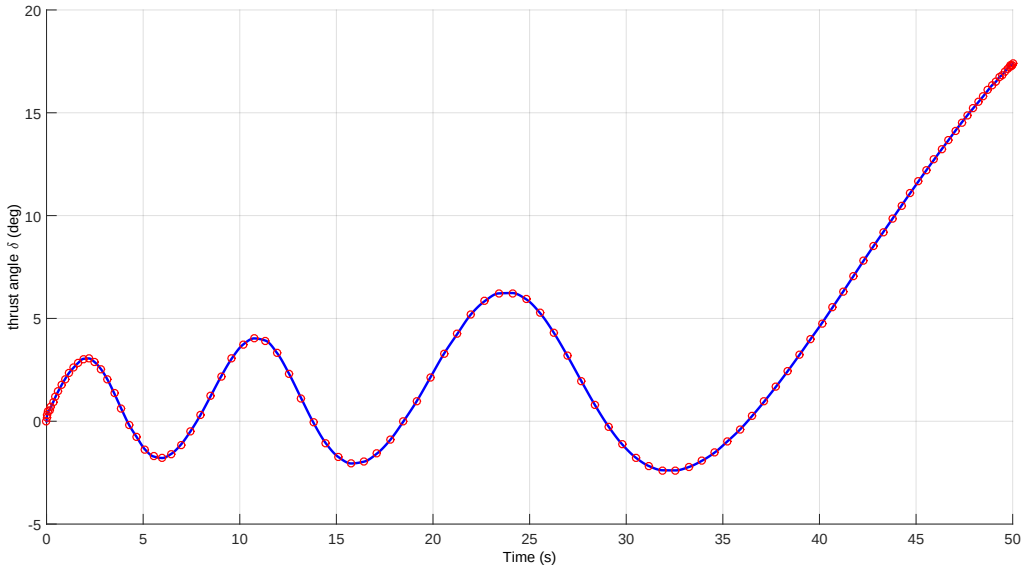

Figure 11: Control Evolution for the Orbit Raising Problem.

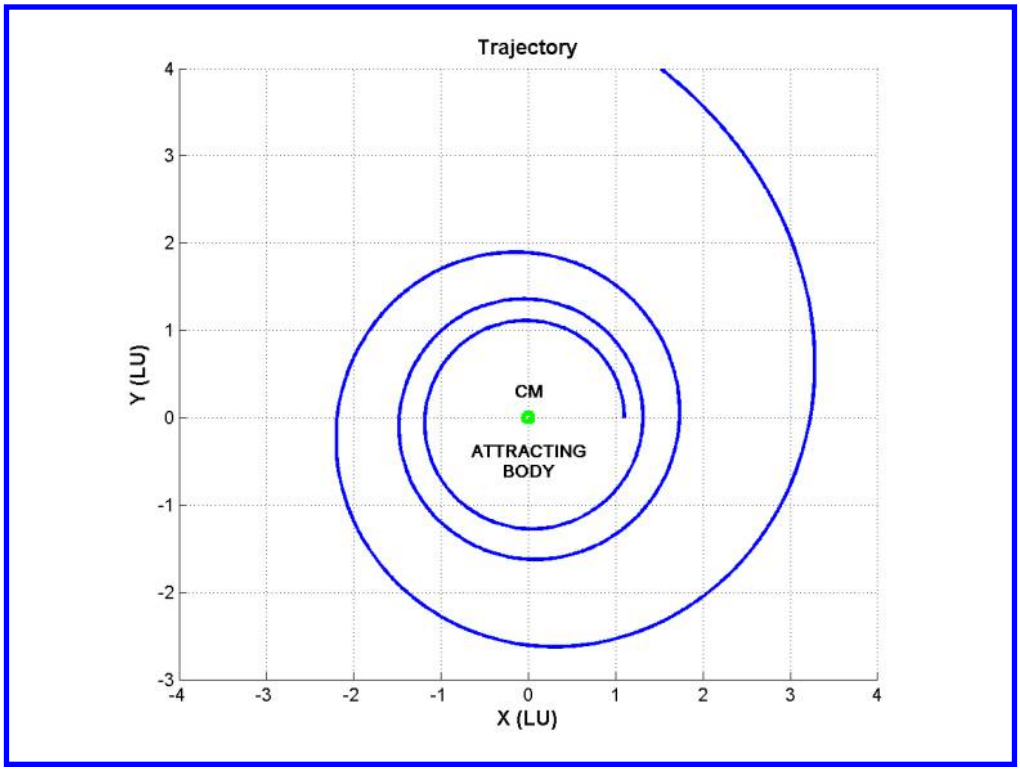

Figure 12: Orbit Raising Problem - Trajectory Optimizing Final Orbit Energy (LU=Unitary Length).

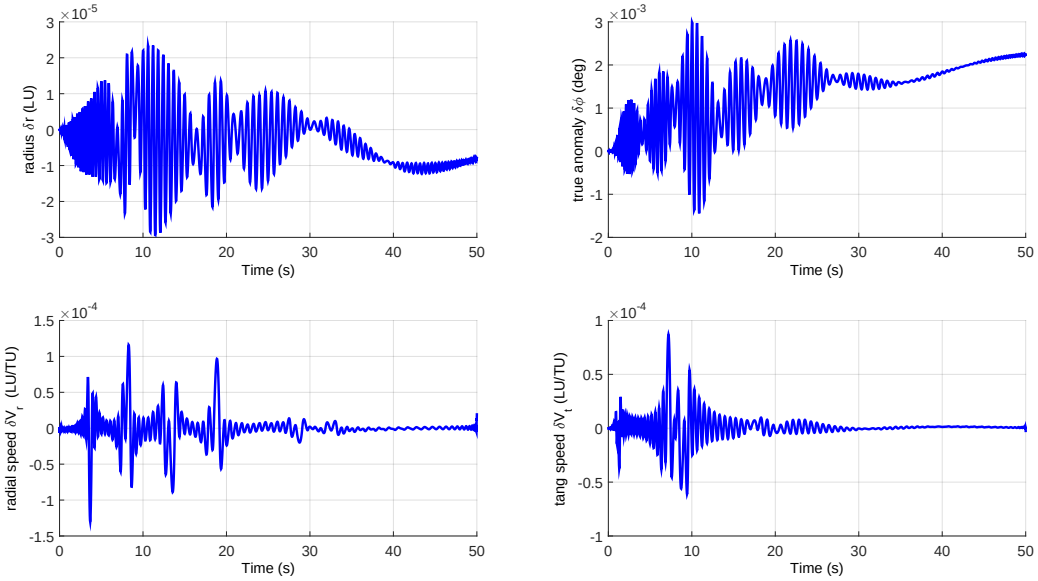

Figure 13: Discrepancy between optimized and propagated solutions for the Orbit Raising Problem.

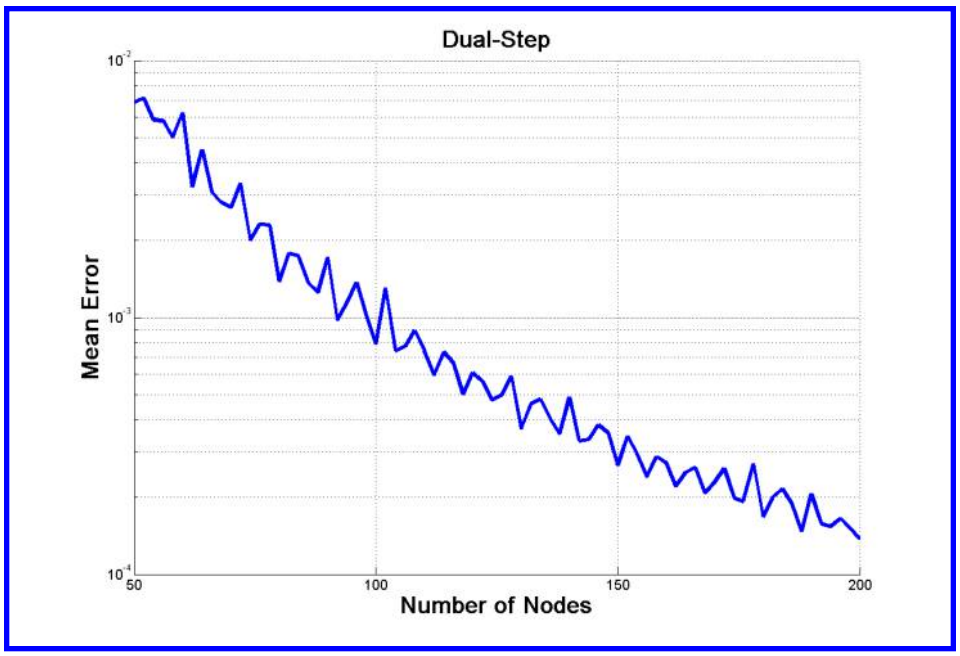

Figure 14: Altitude mean error - "Spectral" Convergence of the solution obtained with Dual-step approach.

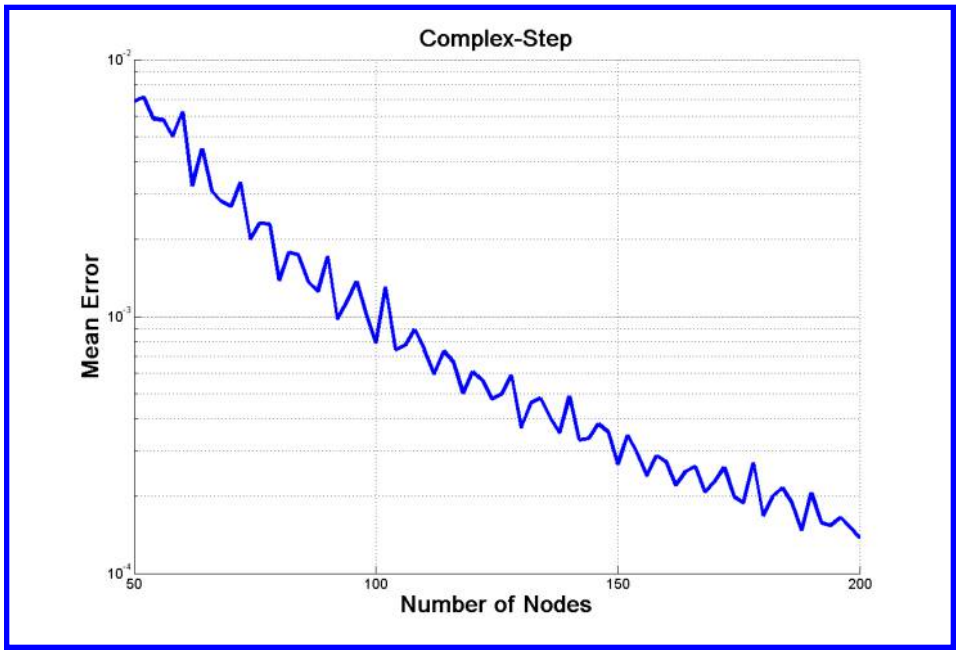

Figure 15: Altitude mean error - "Spectral" Convergence of the solution obtained with Complex-step approach.

Tables 3 and 4 summarize the results obtained with SPARTAN using five different differentiation schemes: the 3-points, the 5-points and the 7-points central difference scheme, the complex-step derivative approach and the dual-step differentiation method. In the first table SNOPT is used to solve the NLP problem, instead in the second table IPOPT is employed in SPARTAN.

SNOPT	100 Nodes				200 Nodes				300 Nodes			
	Mean Error ($\cdot 10^{-5}$)	Max Error ($\cdot 10^{-4}$)	Iter.	CPU (sec)	Mean Error ($\cdot 10^{-5}$)	Max Error ($\cdot 10^{-4}$)	Iter.	CPU (sec)	Mean Error ($\cdot 10^{-5}$)	Max Error ($\cdot 10^{-4}$)	Iter.	CPU (sec)
CD3	1.028	3.392	35	12.94	5.081	3.233	28	71.54	3.283	3.621	10	18.88
CD5	1.481	3.333	35	12.11	1.791	1.131	28	64.99	1.306	3.401	10	16.15
CD7	1.481	3.334	35	12.16	1.791	1.314	28	64.71	1.307	3.406	10	16.36
CS	1.482	3.333	35	12.19	1.792	1.132	28	62.70	1.307	3.406	10	15.76
DS	1.480	3.333	35	15.37	1.791	1.132	28	67.29	1.021	3.422	10	21.65

Table 3: Accuracy and CPU Time comparison for the Orbit Raising Problem (SNOPT).

IPOPT	100 Nodes				200 Nodes				300 Nodes			
	Mean Error	Max Error	Iter.	CPU (sec)	Mean Error	Max Error	Iter.	CPU (sec)	Mean Error	Max Error	Iter.	CPU (sec)
CD3	0.0019	0.0118	87	47.83	0.0037	0.0225	90	333.5	0.0054	0.0330	184	$1.61 \cdot 10^3$
CD5	0.0019	0.0118	72	40.65	0.0037	0.0225	87	317.7	0.0054	0.0330	103	934.9
CD7	0.0019	0.0118	86	46.99	0.0037	0.0225	108	388.8	0.0054	0.0330	126	$1.09 \cdot 10^3$
CS	0.0019	0.0118	110	58.13	0.0037	0.0225	101	360.8	0.0054	0.0330	125	$1.13 \cdot 10^3$
DS	0.0019	0.0118	75	42.0	0.0037	0.0225	114	394.4	0.0054	0.0330	175	$1.56 \cdot 10^3$

Table 4: Accuracy and CPU Time comparison for the Orbit Raising Problem (IPOPT).

In the case related to the use of SNOPT, when the number of nodes is larger (e.g. equal to 200), the Dual-step approach provides accuracy equal or better than the other methods, despite a slightly larger value of the CPU w.r.t. the case of the complex-step. The difference becomes larger when 300 nodes are used. For instance, when compared w.r.t. the complex-step, an increase of 30% in accuracy (relative to the mean error of radius) is paid with a larger CPU time ($\cong +20\%$). No significant differences in accuracy (with an increase in CPU time for the cases $n = 200, 300$) are observed when IPOPT is used.

VI. Conclusions and Future Works

In this paper a thorough analysis on the dual-step differentiation method has been performed; in particular the effects of the combination of the Dual-based hybrid Jacobian computation with the Flipped Radau

Pseudospectral Method have been analysed for two reference examples.
The rationale resides in the fact that the Dual-step differentiation method provides exact first-order derivatives; indeed, the Dual-step formula is subject neither to truncation error, nor to round-off error. Furthermore, methods for computing exact larger derivatives can be created by using hyper-dual numbers which are a larger dimensional extension of dual numbers. This is useful to compute exact Hessians for optimization problems.
The application of the proposed technique to two classical optimal-control problems shows that the use of the Dual-step sligthly improves the accuracy of the results for larger number of nodes, especially in combination with SNOPT, for the problems here considered. In some cases, this is paid in terms of CPU time, as operations with dual numbers are required. For the orbit raising problem, no particular improvements were observed when the dual-step approximation is used in combination with IPOPT, while the combination of SPARTAN and SNOPT provides marginal improvements paid in terms of CPU time when a larger set of nodes is used. The numerical proof of the spectral convergence for the cases analyzed here shows that the Dual-step method generates smoother profiles of error w.r.t. the number of nodes, as predicted by the theory. This would suggest that the results may be improved by using the Dual-step method. In conclusion, even if it may be not possible to define a-priori the most convenient differentiation method to be implemented, the general tendency suggests to take the priorities associated with the problem to be analyzed into account; a trade-off between the desired quality of the results, and the CPU time can be found according to the specific number of nodes, to the NLP solver used, and to the nonlinear behavior of the equations which describe the problem under analysis. For instance, in case very accurate solutions are required, Dual-step based computations may provide better results, while in case the CPU time is a priority, (e.g. in case a trajectory database, made of thousands of solutions, needs to be computed), the Complex-step may still be the best option. However, the Dual-step method has been demonstrated to be a valid alternative to the other traditional, well-known differentiation schemes, and it is well worth being considered as useful method to solve OCPs, especially in combination with SNOPT.
Future research will extend the use of Dual-step method to other OCPs to assess the behavior of the method over a larger set of examples (e.g., "hypersensitive" problems[1]). Moreover, the use of Hyper-Dual numbers for the computation of the Hessian in combination with SPARTAN will be explored, to verify the impact on the performance of IPOPT, which can use this information to improve the results, while for SNOPT this is not possible. In parallel to this research topic, an extension of the developed work to the Dual-Quaternion class[23] can provide a stable and efficient representation of the 6-DOF motion to improve the computational accuracy of several dynamical models.

References

[1] Betts J. T.: *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, SIAM-Society for Industrial and Applies Mathematics, Philadelphia, SECOND EDITION, 2010.

[2] Rao A. V.: *A Survey of Numerical Methods for Optimal Control*, AAS/AIAA Astrodynamics Specialist Conference, AAS Paper 09-334.

[3] Ross M., Sekhavat P., Fleming A., Gong Q.: *Pseudospectral Feedback Control: Foundations, Examples and Experimental Results*, AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, Colorado, August 2006, AIAA 2006-6354, doi: 10.2514/6.2006-6354.

[4] Fahroo F., Ross M.: *Pseudospectral Methods for Infinite-Horizon Optimal Control Problems*, JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS Vol. 31, No. 4, 2008, DOI: 10.2514/1.33117.

[5] Fahroo F., Ross M.: *Advances in Pseudospectral Methods for Optimal Control*,AIAA Guidance, Navigation, and Control Conference, Honolulu, HI, August 2008, doi: 10.2514/6.2008-7309.

[6] Garg D., *Advances in Global PseudoSpectral Methods for Optimal Control*, Ph.D. Dissertation, Aeronautics and Astronautics Dept., University of Florida., 2011.

[7] Sagliano M., Theil S.: *Hybrid Jacobian Computation for Fast Optimal Trajectories Generation*, AIAA Guidance, Navigation, and Control Conference, August 19-22, 2013, Boston, MA, doi: 10.2514/6.2013-4554.

[8] Sagliano M., Samaan M., Theil S., Mooij E.: *SHEFEX-3 Optimal Feedback Entry Guidance*, AIAA SPACE 2014 Conference and Exposition, AIAA 2014-4208, San Diego, CA, 2014, doi:10.2514/6.2014-4208.

[9] Sagliano M.: *Performance analysis of linear and nonlinear techniques for automatic scaling of discretized control problems*, Operations Research Letters, Volume 42, Issue 3, 2014, Pages 213–216, DOI: 10.1016/j.orl.2014.03.003.

[10] Arslantas Y. E., Oehlschlägel T., Sagliano M., Theil S., Braxmaier C., *Approximation of Attainable Landing Area of a Moon Lander by Reachability Analysis*, 17th International Conference and Control (HSSC), Berlin, Germany, 2014.

[11] Arslantas Y. E., Oehlschlägel T., Sagliano M., Theil S., Braxmaier C., *Safe Landing Area Determination for a Moon Lander by Reachability Analysis* International Astronautical Conference, IAC, Toronto, Canada, 2014, paper code:IAC-14,C1,7,2,x21078.

[12] Sagliano M., Oehlschlägel T., Theil S., Mooij E., *Real Time Adaptive Feedforward Guidance for Entry Vehicles*, Proceedings of the 3rd CEAS EuroGNC, Specialist Conference on Guidance, Navigation and Control, ISAE-SUPAERO and ENAC, Toulouse, France, April 2015.

[13] Huneker, L., Sagliano M., Arslantas Y.E., "SPARTAN: An Improved Global Pseudospectral Algorithm for High-Fidelity Entry-Descent-Landing Guidance Analysis, " *30^{th} International Symposium on Space Technology and Science*, Kobe, Japan, 2015.

[14] Gong Q., Ross M., Kang W., Fahroo F.: *Connections between the covector mapping theorem and convergence of pseudospectral methods for optimal control*, Comput Optim Appl (2008) 41: 307–335, doi:10.1007/s10589-007-9102-4

[15] Abramowitz M., Stegun I.: *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, National Bureau of Standards Applied Mathematics Series 55, June 1964.

[16] Kandasamy W. B. V., Smarandache F.: *Dual Numbers*, Zip Publishing, Ohio, 2012.

[17] Fike J. A.: *Numerically Exact Derivative Calculation Using Fake Numbers*, Stanford University, Department of Aeronautics and Astronautics, April 9, 2008.

[18] D'Onofrio V.: *Implementation of Advanced Differentiation Techniques for Optimal Trajectory Computation*, Master Thesis, Aerospace Engineering, University of Naples Federico II, IT, 2015.

[19] Fike J: *Hyper-Dual Numbers*, Aerospace Design Lab, Department of Aeronautics and Astronautics, Stanford University, http://adl.stanford.edu/hyperdual/

[20] Martins J., Sturdza P., Alonso J.: *The Complex-Step Derivative Approximation*, ACM Transactions on Mathematical Software, Vol.29, No.3, September 2003, DOI: 10.1145/838250.838251.

[21] Fike J., Alonso J.: *The Development of Hyper-Dual Numbers for Exact Second-Derivative Calculations*, 49th AIAA, 4-7 January 2011, Orlando, Florida, DOI: 10.2514/6.2011-886.

[22] Herman A. and Conway B.: *Direct Optimization Using Collocation Based on High-Order Gauss-Lobatto Quadrature Rules*, Journal of Guidance, Control, and Dynamics, Vol.19, No. 3, 1996, doi: 10.2514/3.21662.

[23] Kenwright B.: *A Beginners Guide to Dual-Quaternions What They Are, How They Work, and How to Use Them for 3D Character Hierarchies*, WSCG 2012 Communication Proceedings.