69th International Astronautical Congress (IAC), Bremen, Germany, 1-5 October 2018.

IAC-18,C1,9,13,x47354

Hybrid Differential Dynamic Programming Algorithm for Low-Thrust Trajectory Design Using Exact High-Order Transition Maps

Michele Maestrini[a], Pierluigi Di Lizia[b], Roberto Armellin[c], Ryan P. Russell[d]

[a] *Department of Aerospace Science and Technology, Politecnico di Milano, 34 Via Giuseppe La Masa, Milano, MI, Italia 20156,* michele.maestrini@polimi.it
[b] *Department of Aerospace Science and Technology, Politecnico di Milano, 34 Via Giuseppe La Masa, Milano, MI, Italia 20156,* pierluigi.dilizia@polimi.it
[c] *Surrey Space Centre, University of Surrey, BA Building, Guildford GU2 7XH, UK,* r.armellin@surrey.ac.uk
[d] *Department of Aerospace Engineering and Engineering Mechanics, Cockrell School of Engineering, University of Texas at Austin, 301 E. Dean Keeton Street, Austin, Texas, United States 78712-2100,* ryan.russell@austin.utexas.edu

Abstract

Optimal orbital trajectories are obtained through the solution of highly nonlinear large-scale problems. In the case of low-thrust propulsion applications, the spacecraft benefits from high specific impulses and, hence, greater payload mass. However, these missions require a high count of orbital revolutions and, therefore, display augmented sensitivity to many disturbances. Solutions to such problems can be tackled via a discrete approach, using optimal feedback control laws. Historically, differential dynamic programming (DDP) has shown outstanding results in tackling these problems. A state of the art software that implements a variation of DDP has been developed by Whiffen (2006) and it is used by NASA's DAWN mission. One of the latest techniques implemented to deal with these discrete constrained optimizations is the Hybrid Differential Dynamic Programming (HDDP) algorithm, introduced by Lantoine and Russell (2012). This method complements the reliability and efficiency of classic nonlinear programming techniques with the robustness to poor initial guesses and the reduced computational effort of DDP. The key feature of the algorithm is the exploitation of a second order state transition matrix procedure to propagate the needed partials, decoupling the dynamics from the optimization. In doing so, it renders the integration of dynamical equations suitable for parallelization. Together with the possibility to treat constrained problems, this represents the greatest improvement of classic DDP. Nevertheless, the major limitation of this approach is the high computational cost to evaluate the required state transition matrices. Analytical derivatives, when available, have shown a significant reduction in the computational cost and time for HDDP application. This work applies differential algebra (DA) to HDDP to cope with this limitation. DA is introduced to obtain state transition matrices as polynomial maps. These maps come directly from the integration of the dynamics of the system, removing the dedicated algorithmic step and reducing its computational cost. Moreover, by operating on polynomial maps, all the solutions of local optimization problems are treated through differential algebraic techniques. This approach allows users to deal with higher order expansions of the cost, without modifying the algorithm. From the examples provided, it emerges that increasing the order of the expansions does not yield a better convergence rate. Additionally, it causes numerical instability of the algorithm to arise, as well as a noticeable increase on computational time due to the number of polynomial coefficients that ought to be computed.

Nomenclature

- $x_{i,j} \in \mathbb{R}^{n_{x,i}}$ vector of states of dimension $\mathbb{R}^{n_{x,i}}$ at phase i and stage j;
- $u_{i,j} \in \mathbb{R}^{n_{u,i}}$ dynamic controls of dimension $\mathbb{R}^{n_{u,i}}$ at phase i and stage j;
- $w_i \in \mathbb{R}^{n_{\omega,i}}$ static controls of dimension $\mathbb{R}^{n_{\omega,i}}$ at phase i and stage j;
- $\Gamma_i: \mathbb{R}^{n_{\omega,i}} \to \mathbb{R}^{n_{x,i}}$ functions describing initial states of phase i;
- $F_{i,j}: \mathbb{R}^{n_{x,i}} \times \mathbb{R}^{n_{u,i}} \times \mathbb{R}^{n_{\omega,i}} \to \mathbb{R}^{n_{x,i}}$ transition functions that propagate the states from stage j to stage $j+1$ of phase i;
- $L_{i,j}: \mathbb{R}^{n_{x,i}} \times \mathbb{R}^{n_{u,i}} \times \mathbb{R}^{n_{\omega,i}} \to \mathbb{R}$ stage cost functions at phase i and stage j;
- $g_{i,j}: \mathbb{R}^{n_{x,i}} \times \mathbb{R}^{n_{u,i}} \times \mathbb{R}^{n_{\omega,i}} \to \mathbb{R}^{n_{g,i}}$ stage constraints at phase i and stage j;
- $\psi_{i,j}: \mathbb{R}^{n_{x,i}} \times \mathbb{R}^{n_{\omega,i}} \times \mathbb{R}^{n_{x,i+1}} \times \mathbb{R}^{n_{\omega,i+1}} \to \mathbb{R}^{n_{\psi,i}}$ boundary phase constraints between phase i and $i+1$;
- $\phi_{i,j}: \mathbb{R}^{n_{x,i}} \times \mathbb{R}^{n_{\omega,i}} \times \mathbb{R}^{n_{x,i+1}} \times \mathbb{R}^{n_{\omega,i+1}} \to \mathbb{R}^{n_{\phi,i}}$ terminal cost of phase between phase
- i phase $i+1$;

- $J := \sum_{i=1}^{M} \left[\sum_{j=1}^{N_i} (L_{i,j}(x_{i,j}, u_{i,j}, \omega_i)) + \phi_i(x_{i,N_i+1}, \omega_i, x_{i+1,1}, \omega_{i+1}) \right]$ generic formulation of the cost function;
- $\tilde{\phi}_i(x_{i,N_i+1}, \omega_i, x_{i+1,1}, \omega_{i+1}, \lambda_i) := \phi_i(x_{i,N_i+1}, \omega_i, x_{i+1,1}, \omega_{i+1}) + \lambda_i^T \psi_i(x_{i,N_i+1}, \omega_i, x_{i+1,1}, \omega_{i+1}) + \sigma ||\psi_i(x_{i,N_i+1}, \omega_i, x_{i+1,1}, \omega_{i+1})||^2$ Augmented cost function as formulated in HDDP.
- $\max_{\lambda_i} \min_{u_{i,j}, \omega_i} \sum_{i=1}^{M} [\sum_{j=1}^{N_i} (L_{i,j}(x_{i,j}, u_{i,j}, \omega_i)) + \tilde{\phi}_i(x_{i,N_i+1}, \omega_i, x_{i+1,1}, \omega_{i+1}, \lambda_i)]$ Formulation of the cost function in HDDP

While in the framework of DA operations:

- δx represents the infinitesimal variation of a variable;
- $\mathcal{M}$ represents the polynomial map at high order (i.e. all the coefficients of a series expansion except the one of order 0);
- $[x]$ Represent the DA object obtained via series expansion of the quantity x including its infinitesimal and constant part.

1. Introduction

Low thrust optimal control has found growing interest among researchers and practitioners thanks to its remarkable mass savings. With respect to chemical propulsion, ion-thrusters allow for higher specific impulses, yielding greater useful mass. For example, a solar sail mission could reduce the propellant mass stored virtually to zero, as the thrust is provided by solar wind. The downsides in the design of these methods are twofold. First, they increase the size of the problem that needs to be solved due to the large amount of decision variables that must be chosen. Secondly, it causes an accumulation of errors due to the large actuation times needed to produce significant delta-V's [1]. The aim of this thesis is to add to the current literature regarding optimal control for low-thrust trajectory design. In order to do so, the focus is placed on implementing DA techniques for the treatment of a well validated DDP-based technique: HDDP [2]. The work starts by addressing the question whether higher than second order expansions could lead to an improvement on convergence speed of the algorithm. This enhancement in speed can be seen as an augmented convergence region of the algorithm. In particular, HDDP is based on a trust region quadratic programming (TRQP) algorithm, whereas this work attempts at using higher orders. The increase in order of the expansion allows also for the evaluation of nonlinear optimal feedback. To these purposes, this paper starts by replicating HDDP algorithm, which is then enhanced by the addition of DA techniques implemented in the software DACE developed in Politecnico di Milano's Department of Aerospace Science and Technology. Four test cases are provided at the end of this work in order of growing complexity: I) a validation Linear-Quadratic problem; II) a Mono-Dimensional Landing problem; III) an interplanetary Earth-Mars Transfer and, to conclude, IV) a Satellite Constellation Refueling Problem.

The expected improvement of the algorithm is tested on the Satellite Constellation Refueling case. The dynamic model for the system is modified by adding the effect of Earth's zonal harmonic perturbation J2 to the system of dynamical equations. The algorithm achieves convergence without need for the user to compute and assemble complicated partials via a symbolic external software (e.g. MAPLE). Overall, this work provides a first attempt at the application of DA techniques in the Differential Dynamic Programming framework. By doing so, it creates the first building block for further research in this field. Moreover, it adds to the current Differential Dynamic Programming literature by exploiting high order nonlinear optimal feedback controls, also dealing with constraints. Furthermore, it tries to improve the user's capability of exploiting HDDP software by removing the tedious step of obtaining partials from symbolic external software and interfacing it with Fortan/C code.

1.1 Literature Review

In the past, several approaches have been implemented to find optimal trajectories. Among the many solutions, particularly challenging is the case of low-thrust propulsion. Approaches to these optimization challenges are classically divided in two main categories: indirect methods and direct methods [3].

1.1.1 Indirect Methods

These methods rely on calculus of variation or Pontryagin's Maximum Principle [4] to retrieve the necessary optimality conditions. The problem is then reduced to a two-point boundary value problem (TPBVP) and solved. Indirect approaches introduce additional states, the so called "co-states" (or adjoint states), that have little physical meaning most of the time. Additionally, the equations and gradients of co-states, needed to retrieve necessary conditions for optimality, are not easily formulated. Therefore, one of the main disadvantages of indirect methods, is the necessity of the user to have deep knowledge of the problem [4]. Their solution is obtained when, for each time instant, the adjoint states (Lagrange multipliers) extremize the Hamiltonian. Moreover, the boundary conditions together with the dynamics of state and adjoint states must be satisfied. The classical approaches to solution of

these problems include single-shooting and multiple-shooting techniques. Single shooting methods solve a TPBVP with a first guess on the initial conditions, then this trial guess is updated as a function of the final error, details can be found in [5]. Keeping this in mind, a first guess on Lagrange multipliers is also necessary, subsequently this guess is evaluated and iterated until optimality is achieved. The major downside of these techniques is that the first guess on Lagrange multipliers is not always obvious and the solution of the problem is strongly affected by it, as shown in [3,6,7]. Multiple shooting techniques attempt to solve the TPBVP by splitting the time interval in a succession of subintervals. Then, single-shooting is applied between the subintervals and continuity is then ensured via linking constraints. The main advantage of multiple shooting, is the reduction of sensitivity to bad initial conditions and the possibility to parallelize the computations, being the subintervals independent as shown by Betts and Huffmann [8]. On the other hand, the number of decision variables increases quite rapidly with the number of subintervals, causing an increase of computational effort.

1.1.2 Direct Methods

In direct methods, the problem is reformulated from an optimal control problem to a nonlinear (often quadratic) programming problem. To reformulate it as NLP problem, the optimal control problem is discretized and parametrized with polynomials or other functions whose coefficients become the new decision variables. The idea behind this method, is that the research for the approximating function of the solution is restricted in a finite dimensional space of functions. Such approximations are usually piecewise polynomials as explained by Von Stryck and Bulirsch in [4]. Usually, direct methods rely on physical quantities rather than abstract Lagrange multipliers, and their solutions are only approximated. The reformulation is based on a selected decision vector and an iterative procedure which adjusts it during iterations until all convergence criteria are met. The key step of the NLP procedures is the assembly of a Hessian containing all the second order pure and mixed partial derivatives of the cost function with respect to the decision variables. After it is assembled, the Hessian is inverted, however the size of the Hessian grows rapidly with the number of decision variables as explained in [2]. In fact, decision vectors for classic NLP increase linearly with the number of discretization points, while the size of the Hessian grows with the square of the discretized variables (sometimes with the cube) [9]. This causes an increase in computational effort which is twofold. First, inverting the sparse Hessian matrix is notoriously computationally expensive. Secondly, the process of assembling large Hessians is expensive per se. This is owed to the repeated chain rule applications needed to compute all the necessary sensitivities with respect to decision variables [10]. Regardless their broad usage, direct methods are in general less accurate than the indirect ones, and they may converge to a local minimum as shown by Kraft [11]. The main advantage of these direct methods over indirect ones, is the reduced dependence of the solution on initial choice of adjoint variables. This allows for less expert users to tackle the problems, as reduced insight is needed. An additional note should be made regarding Differential dynamic programming. This method is classified as a direct one, but if implemented in a first order version, it retrieves the same equations of calculus of variations, minimizing the Hamiltonian at each iteration [12,13]. The necessary conditions for optimality may not be formulated by DDP as they were for indirect methods, however its solution is influenced by them.

1.1.3 Constrained Optimization

Methods requiring constraint handling, introduce Lagrange multipliers in their cost and new conditions necessary for optimality, i.e. Karush-Kuhn-Tucker conditions [14]. These conditions make it possible to treat linear constraint, therefore nonlinear constraints must be linearized to be addressed. The typical solution to these equality constrained problems, are based on elimination methods (also called primal methods) [15]. These approaches restrict the number of inputs in a subset, so that the optimization problem to be solved becomes unconstrained. The optimal solution is then found in this subset which is compliant with the constraints. Such methods are for example the null-space method and the range space method [16]. The approach of linearizing the constraints is widely used also in methods where the Lagrangian function has a quadratic order expansion such as the Sequential Quadratic Programming [17]. Some methods implement a quadratic approximation of the constraints [18], but they require also quadratic control feedback implementation. Another category of constraint handling procedures is Penalty Methods, which are only approximate solutions to the optimization problem. They include in the objective function a penalty term for the violation of constraints. In this way, the problem is treated as unconstrained and the quality of the constraint satisfaction is dictated by the entity of the penalty. However, penalty methods require the penalty term to be raised to infinity to give exactly zero constraint violation. This characteristic of penalty methods, results in conditioning problems of the Hessian matrix [19].
The necessity to accurately treat inequality constraints, has led to many other solution approaches. A first approach, is to add slack variables so that an inequality can be transformed into an equality, adding to the set of tuning parameters of the problem. The second approach is, instead, to use an active set of constraints to determine whether a constraint is active or not during optimization [16,20]. Finally, an Augmented Lagrangian method can

be used [21,22]. These strategies constitute a hybrid between the Lagrangian approach and the penalty function method. The gist of this method is that the Lagrangian is augmented with a penalty function which increases with the violation of constraints. The introduction of the Lagrange multipliers solves the conditioning problem posed by pure penalty methods [23]. The strategy for solving these problems is by primal-dual approach. This means dividing the problem into an inner loop and an outer loop. The inner loop, is where the decision variables are updated to minimize the cost function. Whereas, the outer loop updates the Lagrange multipliers, after the internal loop has converged, to maximize the cost function. For this reason, these problems are known also as minimax problems. It is paramount to have a correct Lagrange multipliers update formulation, which can be linear as well as nonlinear [2]. The inner loop rigorous convergence is not necessary for the complete algorithm to converge and some problems showed convergence also when the inner loop is only approximately solved as in [24]. The prevalence of one of these methods on another is yet to be proven, however the tendency of penalty function method to increase nonlinearity and slow down convergence rates is well known and documented [23]. An improved method, based on a mixed approach of augmented Lagrangian and range space active set method, was implemented by Yakowitz [25] and reprised by Lantoine and Russell in HDDP [2].

1.1.4 Differential Dynamic Programming

To solve larger problems with an admissible amount of computational power, DDP was introduced. DDP is a technique available for problems that can be reformulated as a dynamic optimization problem. What DDP does is discretizing the optimization into smaller subproblems, dividing the decision vector in smaller sub vectors that only influence local and future optimization steps. Optimizing for each stage on the control variables of that stage, yields a linear scaling of the problem with the number of control variables [26,27]. The founding assumption of DDP is Bellman's Principle of Optimality developed for pure Dynamic Programming [28]. This solution method allows for the computation of an optimal feedback control law from the trajectory for any initial conditions. Unfortunately, this requires prohibitive amount of computational power/storage, hence generating the so-called *"curse of dimensionality"*. DDP is, therefore, a development of pure Dynamic Programming based on a quadratic expansion of the cost in the neighbourhood of a reference trajectory, effectively rendering the minimization a localized problem, hence sacrificing globality. The reference trajectory is then updated iteratively and so are its second order expansions. A feedback law, based on the perturbations of states, is generated during a process called "backward-sweep" so as to improve the following iterate. Instead, during the "forward-sweep", the dynamics of the system subject to this feedback law is re-computed. Analytical quadratic expansions, moreover, allow for improved convergence with respect to previous linear problems or to numerically approximated second order problems. However, DDP is mainly suitable for smooth, unconstrained problems. This is, in fact, the original form in which it was formulated [26]. Several attempts were made at applying DDP also to constrained problems, and an overview of the spectrum of different approaches is presented by Yakowit [29]. The latest methods have started to include well validated NLP techniques in DDP, for example in the work of Lantoine and Russell [2,31]. The state of the art in low-thrust optimal control is Mystic software developed by Whiffen [31,32] and it is based on a DDP variant. Mystic exploits a Hessian shifting technique to enforce convexity of the optimization problems and has a penalty function for treating constraints. Another method that relies on DDP is HDDP [2,31,33,34] which is a multi-phase, multi-stage method that uses augmented Lagrangian technique to treat phase constraints. On the other hand, each stage is constrained via null-space methods directly during optimization. Moreover, instead of a generic Hessian shifting technique, HDDP implements a trust region algorithm to guarantee boundedness of the solution and convexity of the cost function. Being this algorithm multiple-phase but non multiple-shooting, the subproblems are solved in a succession and not independently in parallel. The attempt at applying multiple shooting principles to HDDP was made by Pellegrini [35].

1.1.5 Robust Control

Especially in space applications, the uncertainties are many and they have a great impact on the mission's outcome. The impact of unmodeled dynamical perturbations, as well as the scarce knowledge of the system initial state, produces a very rapid error accumulation in the trajectory. Together with the accumulation of uncertainties on system parameters, non-deterministic constraints and terminal condition can be detrimental to the mission's reference path [1]. Instead of using empirical margins to estimate the propellant necessary for corrections, there has been a growing interest in a robust optimal low-thrust trajectory design strategy. The main benefit coming from this technique would be the possibility to reduce the modelling cost for small satellites, allowing for some errors, which would reduce the cost for research rendering it more accessible [36]. Classical approaches to the solution of optimization problems under uncertainties, rely on the assumption of linearity both of dynamics and feedback control [37]. To face also nonlinear problems, Theodorou et al. [38] have exploited Stochastic Differential Dynamic Programming

(SDDP) to minimize the expected value of the cost function. Finally, a new set of optimization techniques is introduced. They do not rely on any statistical hypothesis on the uncertainties and they belong to the so-called semi-analytic methods. They try to add to this debate by introducing new mathematical tools to treat uncertainties. One of the possible approaches is Interval Analysis, a mathematical tool developed by Moore [39]. The main idea of Interval Analysis is the substitution of real numbers with intervals of real numbers. Arithmetic and analysis of intervals are substituted to the usual ones acting on real numbers. Therefore, acting on the interval of all possible initial values will yield all possible final values as outputs. This technique can be exploited to propagate effectively errors and uncertainties, but it may as well result in an artificially high overestimation of the solution (i.e. "wrapping effect") [40]. Following this train of thoughts has led to the development of a substitute for interval analysis that still retains the idea of propagating more than a single state vector. Differential algebra (DA) guarantees such an instrument. After reducing the optimization problem to a two-point boundary value problem, DA is used to expand the solution of the optimal problem with respect to initial and terminal conditions about a reference trajectory. In doing so, the optimal trajectory and feedback law are computed only once as polynomial maps, and the retrieval of the optimal feedback control policy is obtained as simple polynomial evaluation when different final or initial conditions are imposed [41]. The feedback laws obtained with this approach can be evaluated to an arbitrarily high order.

1.1.6 Differential Algebra

In this framework, an introduction to differential algebra is paramount for the following work. The reason why DA was invented in the first place, was to obtain the solution to analytical problems applying algebraic techniques [42]. The application of DA was extended by Berz [43] for the solution of differential equations and partial differential equations. DA's founding assumption is that it is possible to convey more information about functions than just their values at specific points, being this extra information the Taylor expansion of the function at a certain location up to arbitrary order. Historically, numerical treatment of functions was, in fact, based on floating point arithmetic operations at some specific evaluation points. To achieve the same goal for functions, computer programs were developed, so that they could implement operations between them in a similar way as they are implemented for real numbers. The key feature of DA is in fact its ability to efficiently represent functions in a computer environment in a way that they can be easily manipulated via usual arithmetic expressions. For each operation defined between functions, an equivalent one is coded to act on their Taylor expansions. In this way, the Taylor expansion of the result of an operation between functions is equivalent to applying a corresponding operator to the operands and vice versa. The computer implementation of differential algebra, allows for the computation of a function's values and its Taylor expansion coefficients up to an arbitrary order with a fixed amount of computational effort. Similarly to floating points numbers, elementary operators and algorithm to perform more complex tasks can be implemented in computer environment [42,44].

1.2 Work Outline

In this work, the application of DA for the improvement of some aspects of HDDP is studied. DA is exploited for its ability to retrieve precise high order derivatives of functions along with the functions values. These properties are exploited to propagate the needed partials of the cost function and to obtain feedback laws as inversion of polynomial maps at arbitrarily high order. Additionally, this allows to consider perturbed dynamics with little to no effort. After the theoretical setup has been laid, the methodology is introduced in section 2, in which a clear explanation of the application of DA techniques in HDDP is provided. Following this section, some examples are provided in section 3 and 4, where models and their results are detailed. Finally, the discussion of the results and conclusions are presented in the final section 5, where further research directions are also suggested.

2. Material and methods

A general overview of DDP has been given by Yakowitz [29], the reader is directed to his work for details. This technique is paramount to understand the work of Lantoine and Russell [2] in HDDP which is a stepping stone for this paper. The reader is encouraged to go through the referenced texts to better understand this work, which is not reviewed here in order to avoid repetition.

2.1 HDDP Based on Polynomial Maps

The general outline of the modifications implemented to HDDP is given in this section. The DA makes it possible to compute the derivatives of a function f in v variables up to order n along with the computation of the function's value.

This has important consequences when the dynamical function that maps the states from stage k to stage $k+1$ is obtained via numerical integration. Without a loss of generality, one can express the expansion of an ODE in one variable as:

$$\begin{cases} \dot{x} = f(x) \\ x(t_0) = x_0 \end{cases} \tag{1}$$

The solution of this ODE equation requires algebraic operations to be performed, together with the evaluation

of f at several time instants. Thanks to this, if the initial point is initialized as its constant part plus the DA identity (i.e. the Taylor expansion of its identity function) $[x_0] = x_0 + \delta x_0$, then the Taylor expansion of the solution at each integration step is obtained as a function of variation from reference initial conditions. The procedure is described with a first order Euler integration scheme, but any ODE scheme ideally exploits the same algebraic operations. Using a first order scheme as $x_{k+1} = x_k + f(x_k)\Delta t$, then by the properties of DA [42] this new value can be expressed as $[x_{k+1}] = [x_k] + f([x_k])\Delta t$. The extraction of the operator $f([x_k])$ gives the Taylor expansion of the function f about the starting condition x_k as a function of δx_k, expressed as $f(x_k) + \mathcal{M}_f(\delta x_k)$ (constant and differential part of the expansion). The remaining algebraic operations help to compute $[x_{k+1}] = x_{k+1} + \mathcal{M}_{x_{k+1}}(\delta x_k)$ which represents the expansion of x_{k+1} with respect to the initial value x_k about the reference point x_{k+1}. If the procedure is continued, then at the step $k + 2$:

$$\begin{aligned}&[x_{k+2}] = [x_{k+1}] + \Delta t f([x_{k+1}]) = \\ &= x_{k+1} + \mathcal{M}_{x_{k+1}}(\delta x_k) + \Delta t f(x_{k+1} + \mathcal{M}_{x_{k+1}}(\delta x_k)); \\ &[x_{k+2}] = x_{k+1} + \mathcal{M}_{x_{k+2}}(\delta x_k);\end{aligned} \quad (2)$$

This procedure gives, $[x_{k+2}] = x_{k+2} + \mathcal{M}_{x_{k+2}}(\delta x_k)$ and can be repeated for each integration step from x_0 up to the desired integration interval N, so that the final result is: $[x_N] = x_N + \mathcal{M}_{x_N}(\delta x_0)$. This result comes from the fact that propagating ODE system in the DA framework only requires evaluations of the right-hand side, algebraic operations and composition of DA polynomials. This result holds its validity for multi-variable functions propagated through higher order ODE schemes. In HDDP, this is exploited for estimating the partials removing the STM computation step. During the forward sweep of the algorithm, the variables at step k are initialized as their value on the reference trajectory plus their DA identity, one per each independent variable needed (i.e. controls, states, Lagrange multipliers).

$$\begin{aligned}&[x_k] = \overline{x}_k + \delta x_k; \\ &[u_k] = \overline{u}_k + \delta u_k; \\ &[\omega] = \overline{\omega} + \delta\omega; \\ &[\lambda] = \overline{\lambda} + \delta\lambda;\end{aligned} \quad (3)$$

Then the dynamic transition function of the system is applied to these DA objects and the result is:

$$\begin{aligned}&[x_{k+1}] = F([x_k], [u_k], [\omega]) = x_{k+1} + \\ &\mathcal{M}_{x_{k+1}}(\delta x_k, \delta u_k, \delta\omega)\end{aligned} \quad (4)$$

Notice how, regardless of the shape of F, being it analytical or numerically integrated via ODE, this expression retains its value and guarantees the correct partials. This equation represents the nominal value of the new state and its high order polynomial expansion $\mathcal{M}_{x_{k+1}}$ as a high order polynomial of $\delta x_k, \delta u_k, \delta\omega$ about the reference starting condition at instant k. At each stage these maps are stored, they contain all the information on partial derivatives needed for propagation of partials of the cost function.

2.2 Backward Sweep on Stages

When treating the backward sweep across the stages of a phase, the index of the phase is removed to reduce notation complexity. At the beginning of the backward sweep, the state x_{N+1} is available as well as the value of Lagrange multipliers λ, and they are augmented with their DA identity functions generating $[x_{N+1}]$ and $[\lambda]$. Without loss of generality, the final cost can be expressed as:

$$\left[\tilde{\phi}\right] = \tilde{\phi}([x_{N+1}], [\lambda], [\omega]) = \tilde{\phi} + \mathcal{M}_{\phi}(\delta x_{N+1}, \delta\lambda, \delta\omega); \quad (5)$$

In the forward sweep, the transition maps $\delta x_{k+1} = \mathcal{M}_{x_{k+1}}(\delta x_k, \delta u_k, \delta\omega)$ were stored, therefore it is possible to retrieve the last map $\mathcal{M}_{x_{N+1}}$ for usage. To start the backward sweep, this polynomial map is composed with that of $\tilde{\phi}$, allowing for the expression of:

$$\begin{aligned}&[J_{N+1}] = \tilde{\phi} + \mathcal{M}_{\phi}(\delta x_{N+1}, \delta\lambda, \delta\omega) = \\ &= \tilde{\phi} + \mathcal{M}_{\phi}(\mathcal{M}_{N+1}(\delta x_N, \delta u_N, \delta\omega), \delta\lambda, \delta\omega); \\ &[J_{N+1}] = J_{N+1} + \mathcal{M}_{J_{N+1}}(\delta x_N, \delta u_N, \delta\omega, \delta\lambda);\end{aligned} \quad (6)$$

To the last term of this equation, the last stage cost (provided there is one) must be added $[L_N] = L_N + \mathcal{M}_{L_N}(\delta x_N, \delta u_N, \delta\omega)$.

$$[J_N] = [J_{N+1}] + [L_N] = J_N + \mathcal{M}_{L_N}(\delta x_N, \delta u_N, \delta\omega); \quad (7)$$

The next step of the algorithm, is to solve he optimization problem: the gradient of J_N with respect to u_N must be set to 0. Therefore, a control law can be retrieved by exploiting DA maps inversion. The derivatives of J_N with respect to u_N can be taken in the neighbourhood of u_N thanks to the differentiation operator available in DACE, and then they can be set to zero:

$$[J_{u,N}] = J_{u,N} + \mathcal{M}_{J_{u,N}}(\delta x_N, \delta u_N, \delta\omega, \delta\lambda) = 0; \quad (8)$$

The feedback law $\delta u_N(\delta x_N, \delta\omega, \delta\lambda)$ is now obtained via map inversion as in Equation (9). To guarantee the feasibility of this inversion, some identity polynomial maps have been introduced to exploit the inversion

operator available in DACE. Such operator is based on reducing the inversion problem to a fixed-point problem.

$$\begin{aligned}&\delta J_{u,N} = \mathcal{M}_{J_{u,N}}(\delta x_N, \delta u_N, \delta\omega, \delta\lambda);\\ &\delta u_N = \mathcal{M}^{-1}_{J_{u,N}}(\delta J_{u,N}, \delta x_N, \delta\omega, \delta\lambda);\\ &\delta u_N = \delta u_N\big(\delta J_{u,N}, \delta x_N, \delta\omega, \delta\lambda\big);\end{aligned} \tag{9}$$

The polynomials relative to δu_N are extracted and their coefficients stored for usage in the forward sweep, after their evaluation in $\delta J_{u,N} = -J_{u,N}$ to guarantee that the feedback law extremizes the cost. The final input expression is, therefore, Equation (10).

$$\delta u_N^* = \delta u_N^*(\delta x_N, \delta\omega, \delta\lambda); \tag{10}$$

Recovering now the DA function J_N, it is possible to compose the polynomial $[J_N]$ with these maps of δu_N^* and obtain the optimized cost as Equation (11).

$$\begin{aligned}&[J_N^*] = J_N + \mathcal{M}_{J_N}(\delta x_N, \delta u_N^*(\delta x_N, \delta\omega, \delta\lambda), \delta\omega, \delta\lambda) =\\ &J_N^* + \mathcal{M}_{J_N^*}(\delta x_N, \delta\omega, \delta\lambda);\end{aligned} \tag{11}$$

The expected reduction can now be estimated by evaluating on the reference trajectory:

$$ER_N = [J_N] - [J_N^*] = J_N - J_N^*; \tag{12}$$

Once the step N is performed, the cost is propagated backward:

$$\begin{aligned}&[J_{N-1}] = L_{N-1} + \mathcal{M}_{L_{N-1}}(\delta x_{N-1}, \delta u_{N-1}, \delta\omega) + J_N^*\\ &\qquad + \mathcal{M}_{J_N^*}(\delta x_N, \delta\omega, \delta\lambda);\\ &= L_{N-1} + \mathcal{M}_{L_{N-1}}(\delta x_{N-1}, \delta u_{N-1}, \delta\omega) + J_N^* +\\ &\mathcal{M}_{J_N^*}\big(\mathcal{M}_{x_N}(\delta x_{N-1}, \delta u_{N-1}, \delta\omega), \delta\omega, \delta\lambda\big) =\\ &= J_{N-1} + \mathcal{M}_{J_{N-1}}(\delta x_{N-1}, \delta u_{N-1}, \delta\omega, \delta\lambda);\end{aligned} \tag{13}$$

Having shown how to start the iteration process, at the generic step k the process performs the following steps:

- $[J_k] = [L_k] + [J_{k+1}^*]$ is composed with the dynamic mappings obtained during the forward sweep and stored as $\delta x_{k+1} = \mathcal{M}_{x_{k+1}}(\delta x_k, \delta u_k, \delta\omega)$ to obtain $[J_k] = J_k + \mathcal{M}_{J_k}(\delta x_k, \delta u_k, \delta\omega, \delta\lambda)$.
- Derivatives of $[J_k]$ are extracted and the gradient with respect to the controls is equated to zero as in Equation (8).
- The gradient $\delta J_{u,k}$ is augmented with identity maps to guarantee inversion and inverted as in Equation (9).
- The feedback law in polynomial form is evaluated in $\delta J_{u,k} = -J_{u,k}$ and the feedback polynomial law is substituted in J_k to obtain J_k^* as in Equation (11).
- The expected reduction is obtained by extracting the difference between the nominal value of J_k and J_k^* on the reference trajectory (i.e. $\delta x_k = 0, \delta\omega = 0, \delta\lambda = 0$), the algorithm proceeds to $k-1$.

The process can be repeated per each step until step 0 of the phase. It is important to notice how this method, when the arbitrary order is set to $n = 2$, retrieves the same partials mapping and linear feedback law as pure HDDP, without the cumbersome formulation of partial derivatives of dynamic functions, or integration of a system of ODEs of size $N + N^2 + N^3$, where N is the number of variables.

2.3 Constraints Handling Techniques

The constraint handling techniques are the same as HDDP but with slight modifications. To enforce null-space methods as in the standard HDDP for control bounds, the nominal polynomial feedback is evaluated on the reference trajectory (i.e.$\delta x_k = 0, \delta\omega = 0, \delta\lambda = 0$). If a constraint is violated, the polynomial function is substituted with a constant δu_k that fixes the controls at the next iteration on the control bounds, the control is removed from the set of active controls and the process is repeated with the remaining constraints (e.g. Mono-Dimensional Landing Problem in section 3.2). The treatment of nonlinear constraints requires the introduction of other DA variables, the Lagrange multipliers of stage nonlinear active constraints ν_k. The set of active constraints are estimated by checking if the nominal δu_k violates them exactly as in HDDP. Then, the Lagrangian $\mathcal{L}$ is constructed adding the term $\nu_k(g_k)$ to the cost function, and it is minimized with respect to δu_k and $\delta\nu_k$.

$$\begin{aligned}&\begin{pmatrix}\delta\mathcal{L}_{u,k}\\ \delta\mathcal{L}_{\nu,k}\end{pmatrix} = \begin{pmatrix}\mathcal{M}_{\mathcal{L}_{u,k}}\\ \mathcal{M}_{\mathcal{L}_{\nu,k}}\end{pmatrix}\begin{pmatrix}\delta x_k\\ \delta u_k\\ \delta\omega\\ \delta\lambda\\ \delta\nu_k\end{pmatrix};\\ &\delta u_k = \begin{pmatrix}\mathcal{M}_{\mathcal{L}_{u,k}}\\ \mathcal{M}_{\mathcal{L}_{\nu,k}}\end{pmatrix}^{-1}\begin{pmatrix}\delta\mathcal{L}_{u,k}\\ \delta\mathcal{L}_{\nu,k}\\ \delta x_k\\ \delta\omega\\ \delta\lambda\end{pmatrix};\end{aligned} \tag{14}$$

The optimal constrained feedback is obtained by evaluating δu_k in $\delta\mathcal{L}_{u,k} = -\mathcal{L}_{u,k}$ and $\delta\mathcal{L}_{\nu,k} = -\mathcal{L}_{\nu,k}$. The process can then continue as before, noting that there is no need to store these Lagrange multipliers as they are not needed during the iteration.

2.4 Backward Sweep on Phases

To extend the backward sweep across phases after all the stages on phase i have been minimized, another

composition of maps is necessary. At the end of a phase, it is necessary to initialize other variables as DA variables, adding their identity polynomials. There are $\delta\omega_i, \delta\omega_{i-1}, \delta\lambda_i, \delta\lambda_{i-1}$ and δx_i. The notation is here simplified again, values indexed $-$ refer to multipliers, controls and states pertinent to phase $i-1$, whereas the index $+$ refers to the same quantities at phase i. First, the initial conditions of a stage are exploited. At the end of the sweep on phase i, the cost is:

$$[J_+^*] = J_+^* + \mathcal{M}_{J_+^*}(\delta x_+, \delta\omega_+, \delta\lambda_+); \quad (15)$$

To this cost, it is necessary to add the termination cost of phase $i-1$ which, thanks to DA, is expressed as:

$$[\tilde{\phi}_-] = \tilde{\phi}_- + \mathcal{M}_{\tilde{\phi}_-}(\delta x_-, \delta\omega_-, \delta x_+, \delta\omega_+, \delta\lambda_-); \quad (16)$$

Therefore, in a similar way as for HDDP, applying $[\Gamma(\omega_+)] = [x_+]$ and composing the cost function with it, gives:

$$[J_-] = J_+^* + \tilde{\phi}_- + \mathcal{M}_{\tilde{\phi}_-}(\delta x_-, \delta\omega_-, \delta x_+, \delta\omega_+, \delta\lambda_-) + \mathcal{M}_{J_+^*}(\delta x_+, \delta\omega_+, \delta\lambda_+) = J_- + \mathcal{M}_{J_-}(\delta x_-, \delta\omega_-, \delta\omega_+, \delta\lambda_-, \delta\lambda_+); \quad (17)$$

This substitution effectively removes the dependency from δx_+. To retain the same scheme of HDDP and obtaining the same results when the order of the problem is set to 2, the problem is uncoupled as it was done in HDDP. The derivatives with respect to $\delta\lambda_+$ will only present mixed terms with $\delta\omega_+$ as the only component of Equation (17) in which $\delta\lambda_+$ appears is Equation (15). Therefore, the following solution is setup by equating the gradient of this equation with respect to $\delta\lambda_+$ to zero.

$$\begin{aligned} &\delta J_{\lambda_+,-} = \mathcal{M}_{J_{\lambda_+,-}}(\delta\lambda_+, \delta\omega_+); \\ &\delta\lambda_+ = \mathcal{M}^{-1}_{J_{\lambda_+,-}}(\delta J_{\lambda_+,-}, \delta\omega_+); \end{aligned} \quad (18)$$

This equation retrieves a feedback law for $\delta\lambda_+(\delta\omega_+)$ by evaluating it in $\delta J_{\lambda_+,-} = -J_{\lambda_+,-}$. By composing this polynomial map with Equation (17), Equation (19) is retrieved.

$$[J_-^{*1}] = J_-^{*1} + \mathcal{M}_{J_-^{*1}}(\delta x_-, \delta\omega_-, \delta\omega_+, \delta\lambda_-); \quad (19)$$

Once again, the feedback law for $\delta\omega_+$ is obtained by equating to zero the gradient of Equation (19) with respect to these variables:

$$\begin{aligned} &\delta J^{*1}_{\omega_+,-} = \mathcal{M}_{J^{*1}_{\omega_+,-}}(\delta\omega_+, \delta x_-, \delta\omega_-, \delta\lambda_-); \\ &\delta\omega_+ = \mathcal{M}^{-1}_{J^{*1}_{\omega_+,-}}\left(\delta J^{*1}_{\omega_+,-}, \delta x_-, \delta\omega_-, \delta\lambda_-\right); \end{aligned} \quad (20)$$

This is the final step in the phase optimization procedures, as now the feedback law of $\delta\omega_+(\delta x_-, \delta\omega_-, \delta\lambda_-)$ can be retrieved by imposing $\delta J^{*1}_{\omega_+,-} = -J^{*1}_{\omega_+,-}$. If this feedback law is substituted into Equation (19), the optimal function at the beginning of phase $i-1$ is found, and the stages backward sweep can re-start:

$$[J_-^*] = J_-^{*1} + \mathcal{M}_{J_-^{*1}}(\delta x_-, \delta\omega_-, \delta\omega_+(\delta x_-, \delta\lambda_-, \delta\omega_-), \delta\lambda_-) = J_-^* + \mathcal{M}_{J_-^*}(\delta x_-, \delta\omega_-, \delta\lambda_-); \quad (21)$$

The expected reduction of this step can still be computed as the difference between Equation (21) and Equation (19) after substituting the optimal $\delta\omega_+^*$ feedback law and evaluating it on the reference trajectory (all variations set to 0).

$$ER_- = [J_-] - [J_-^*]; \quad (22)$$

Moreover, the feedback law for $\delta\omega_+$ is now available, and the updating scheme of the Lagrange multipliers can be obtained as composition of:

$$\begin{aligned} &\delta\lambda_+(\delta\omega_+); \\ &\delta\omega_+(\delta x_-, \delta\lambda_-, \delta\omega_-); \\ &\delta\lambda_+ = \delta\lambda_+(\delta\omega_+(\delta x_-, \delta\lambda_-, \delta\omega_-)); \end{aligned} \quad (23)$$

2.5 Trust Region Algorithm

For the treatment of Trust Region limitation and to guarantee a descent direction, the same approach as standard HDDP is used. The TRQP algorithm is applied on the Hessians and gradient. Afterwards, the cost function is modified by adding a shift of γ to the coefficients of second order of the cost function J, so that the new modified cost J_m has the same Hessian as the shifted cost. To conclude, the gradient of the modified cost is taken equal to 0, and the feedback laws are retrieved. For example, to shift the input Hessian the procedure is the following:

- Starting from $[J_k]$, the derivatives with respect to δu_k are stored in the gradient and the Hessian matrix.
- These matrices are used in the TRQP procedure.
- The $[J_m]$ function is obtained: $[J_m] = J_k + \gamma\delta u_k^T \delta u_k$
- This modified J is used to obtain the feedback law, by imposing the gradient to be zero.
- The algorithm proceeds by substituting this feedback into the cost function $[J_k]$ to obtain $[J_k^*]$.

This algorithm poses a limitation on the size of the quadratic trust-region, even for higher orders. Future research directions aim at modifying this step to exploit the full potential of DA and higher orders.

3. Study Cases

Three example applications are here reported, a first validation example, one application of mono-dimensional landing and a test case for orbital perturbations.

3.1 Linear-Quadratic Problem

The solution of this exercise is implemented according to Lantoine and Russell [31] with some minor modifications. This example is known to converge for HDDP and the preliminary test is to assess whether DA introduction spoiled convergence. This kind of problems is linear in the controls and quadratic in cost, moreover it has linear constraints. This structure makes it possible to retain quadratic augmented cost, thanks to the linearity of the constraints. Therefore, these problems should converge in only one iteration for methods based on augmented Lagrangian.

Similarly to the test case in [31], this example exploits 2 phases ($M = 2$) and 5 stages per each phase ($N1 = N2 = 5$). The transition functions $F_{i,j}$ are defined as:

$$x_{i,j+1} = F_{i,j}(x_{i,j}, u_{i,j}) = \begin{bmatrix} r_{i,j+1} \\ v_{i,j+1} \end{bmatrix} = \begin{bmatrix} r_{i,j} + v_{i,j} \\ v_{i,j} + u_{i,j} \end{bmatrix}; \quad (24)$$

The stage constraints $g_{i,j}$ are not present, on the contrary the phase constraints ψ_1 at the end of phase 1 and ψ_2 at the end of phase 2 are defined as follows:

$$\begin{aligned} \psi_1 &= x_{2,1} - x_{1,6} = 0; \\ \psi_2 &= x_{2,6} - x_t = 0; \end{aligned} \quad (25)$$

The final objective state here is defined by x_t and it is a targeted point at the end of the second phase. The first phase constraint is just imposing continuity between the last stage of the first phase and the starting stage of the second phase. The constant controls ω are defined only on the second phase, as in the first phase every parameter is fixed. ω_2 are used to parametrize the initial conditions of the second phase as $x_{2,1} = \Gamma(\omega_2) = \omega_2$. Finally, the stage cost function is formulated as $L_{i,j} = \|u_{i,j}\|^2$.

3.2 Mono-Dimensional Landing

The dynamic formulation of this problem is reprised from Lantoine and Russell [34]. The objective function is formulated as $J_1 = -m_f$, hence with only one phase whose cost is $-m(t_f)$ and no stage cost L_k. The dynamics that need to be integrated are:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{m} \end{bmatrix} = \begin{bmatrix} v \\ -g + \frac{T}{m} \\ -\frac{T}{g0Isp} \end{bmatrix}; \quad (26)$$

The number of stages considered are ten, and the constraints on those are control bounds with $u^U = 1.227$ and $u^L = 0.0$. They can be treated with null space method or with range space active set method. The phase constraint is:

$$\begin{bmatrix} x(t_f) \\ v(t_f) \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}; \quad (27)$$

Finally, the initial conditions and parameters are:

$$\begin{bmatrix} x(0) \\ v(0) \\ m(0) \\ TOF \\ g_0 I_{sp} \\ g \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.783 \\ 1.0 \\ 1.397 \\ 2.349 \\ 1.0 \end{bmatrix}; \quad (28)$$

The provided model of the system is already scaled for improved convergence. The integration scheme used for the dynamics is a direct Euler first order method, which proved to be sufficient for this example.

3.3 Satellite Constellation Rephasing

The last application of modified HDDP, is to a case of large satellite constellations. The orbital data selected for this example is the one of IRIDIUM–NEXT constellation. The source of Keplerian elements for this satellite was the Air Force Space Command. The case studied is the rephasing of 32° of one of the constellation's satellite. The initial guess for the algorithm was created using MATLAB. The time of flight guess was retrieved by using a Lambert arc approach. Indeed, a sampling of the orbit in 100 points was made. Then, for each of these sampled positions, a Lambert arc was created, with a TOF in a range between 0 and the orbital period. The final target position could be evaluated propagating in time the position of the target satellite for a time equal to the TOF considered plus the time of departure. A Porkchop graph was produced (SEE Fig. 6), where only one period in the departure time range is considered. To estimate the minimum Δv budget necessary to reach one satellite, a manoeuvre of two impulses was studied and their total contribute is reported. The initial guess for the time of flight obtained is TOF = 1.516 h. This initial guess was introduced in the modified HDDP algorithm. The problem was scaled with the same scheme as the Earth-Mars Transfer case. The problem is divided in 100 stages. The dynamics that need to be integrated are reported in Equation (29).

$$\begin{cases} \dot{x} = \dot{x}; \\ \dot{y} = \dot{y}; \\ \dot{z} = \dot{z}; \\ \ddot{x} = -\frac{\mu}{r^3}x + \frac{T_x}{m}; \\ \ddot{y} = -\frac{\mu}{r^3}y + \frac{T_y}{m}; \\ \ddot{z} = -\frac{\mu}{r^3}z + \frac{T_z}{m}; \\ \dot{m} = -\frac{T}{I_{sp}g_0}; \end{cases} \quad (29)$$

The integration scheme used is a 7/8 Dormand-Prince (8th order solution for propagation, 7th order solution for step size control) Runge-Kutta scheme and the initial conditions to this problem, as obtained from the preliminary Porkchop graph study, are reported in Equation 30.

$$\mathbf{r_0} = [1085.028 \quad -241.357 \quad -7071.888]\text{km}; \\ \mathbf{v_0} = [-5.922 \quad 4.407 \quad -1.0743]\text{km s}^{-1}; \quad (30)$$

The target position of the satellite is reported in Equation (31).

$$\mathbf{r_t} = [1444.3563 \quad 509.9112 \quad 6992.1188]\text{km}; \\ \mathbf{v_t} = [5.8468 \quad 4.3874 \quad 1.5421]\text{km s}^{-1}; \quad (31)$$

The cost function is again an augmented Lagrangian one, which minimizes the energy as formulated in Equation (32).

$$J = \sum_{k=0}^{N}(||u_k||^2) + \lambda^T(x(t_f) - x_t) + \sigma_0||x(t_f) - x_t||^2; \quad (32)$$

The phase constraint function is the violation of the target $\boldsymbol{\psi}(t_f) = \boldsymbol{x}(t_f) - \boldsymbol{x_t}$, while there is no final phase cost.
Additionally, the stage cost from Equation (32) is $L_k = \|\boldsymbol{u}_k^2\|$.

The maximum thrust available for the system is $T_{max} = 310$ N to guarantee sufficient thrust to achieve the target in the selected TOF. Again, the constraint to be imposed on stages is a nonlinear one, it requires that the amplitude of the thrust is smaller than the maximum allowed value $\|\boldsymbol{u}_k\| < T_{max}$.

After optimizing this first case, a second case is developed to show the flexibility of this algorithm. This case considers the presence of the J2 perturbation effect in the dynamic model, which, is reformulated as Equation (33).

$$\begin{cases} \dot{x} = \dot{x}; \\ \dot{y} = \dot{y}; \\ \dot{z} = \dot{z}; \\ \ddot{x} = -\frac{\mu}{r^3}x + \frac{T_x}{m} - \frac{3J_2\mu R_\oplus^2}{2r^5}(1 - 5\frac{z^2}{r^2})x; \\ \ddot{y} = -\frac{\mu}{r^3}y + \frac{T_y}{m} - \frac{3J_2\mu R_\oplus^2}{2r^5}(1 - 5\frac{z^2}{r^2})y; \\ \ddot{z} = -\frac{\mu}{r^3}z + \frac{T_z}{m} - \frac{3J_2\mu R_\oplus^2}{2r^5}(3 - 5\frac{z^2}{r^2})z; \\ \dot{m} = -\frac{T}{I_{sp}g_0}; \end{cases} \quad (33)$$

4. Results and Discussion

4.1 Linear-Quadratic Problem

The algorithm converges in exactly one iteration. The problem is set up in a way to permit the user modification of number of phases and stages in the process. The results are shown with 2 phases and 5 stages each, but this number can be modified accordingly. As the time step is fixed, increasing *N* or *M* corresponds to increasing the total time of the process, yielding smaller input and therefore smaller cost function. The target point is $x_t = [2.0;\ 4.0;\ 1.0;\ -0.5;\ 1.5;\ -2.5]$, whereas the initial point of the algorithm is set to $x_0 = [1.0;\ 1.0;\ 1.0;\ 1.0;\ 1.0;\ 1.0]$, while the first guess for inputs and Lagrange multipliers is all zeros. Also, the initial conditions for constant controls on phase 2, that corresponds to the initial condition of the states of phase 2, are set to zero. As it can be observed in Fig. 1, the continuity constraints between phases are respected, moreover the final target is achieved in one iteration only. The result is consistent if the number of stages or phases is changed and the convergence is achieved in exactly one iteration. Several trials with different target points, different *M* and *N* have reported the same result. Finally, the input trajectory is reported in the following Fig. 2. For completeness, also the value of the Lagrange multipliers for the phase constraints are reported in Table 1.

Table 1. Value of the Lagrange multipliers for the Linear-Quadratic Problem

Phase	Lagrange Multipliers
1	[0.0545455; 0.224242; -0.139394; 0.327273; 0.0121212; 0.630303]
2	[0.0545455; 0.224242; -0.139394; 0.0545455; -1.10909; 1.32727]

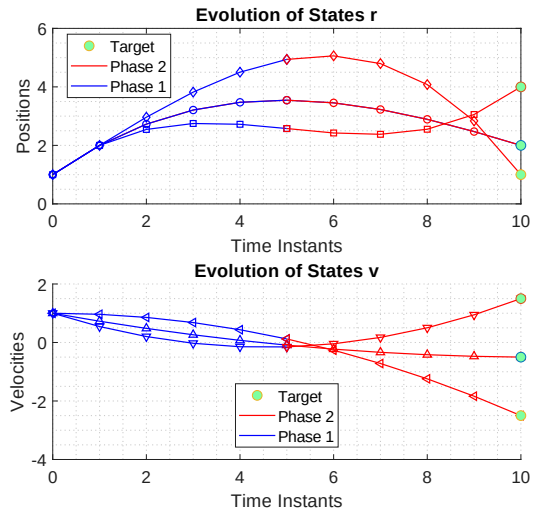

Fig. 1. Evolution of states for the linear-quadratic Problem.

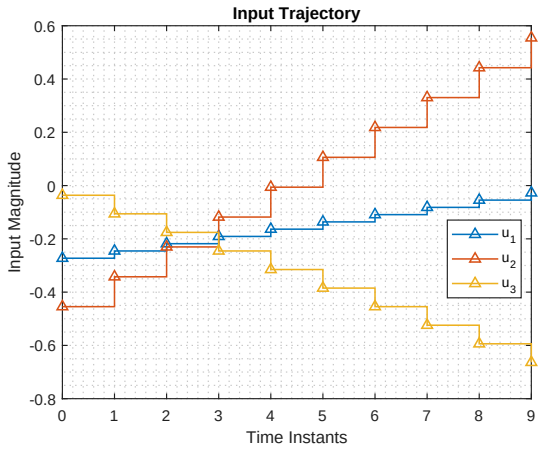

Fig. 2. Evolution of controls for the linear-quadratic Problem.

4.2 Mono-Dimensional Landing Problem

The final norm of constraint violation is $f = 7.6952\ \mathrm{X}\ 10^{-6}$, while the final mass is $m_f = 0.392$.

First, the results for second order are reported, which yield the same results as in [55] and [90]. The evolution of the height h and the velocity v is represented in Fig. 3, whereas the control input is reported in Fig.5.

The next step is to analyse the behaviour of the solution for different orders of expansion. This task is performed in Table 2. Orders higher than 6 do not converge for this problem. Heuristics and experience indicate that a high-order feedback controller behaves better when close to the solution of the problem.

Bearing in mind that the starting position is far enough from the optimal solution, while looking at Fig. 4 it is possible to observe that order 2 behaves better than the others. While at the beginning of the algorithm the solver is far from the solution, higher-order exhibits greater violation of constraints. On the other hand, after some iterations, the higher orders tend to have smaller violation of constraints, due to the proximity to the optimal trajectory. For really high orders, numerical difficulties arise, spoiling the convergence. As it can be already seen for order 6, feedback terms of high order tend to render the constraint violation reduction noisier. Probably, order 6 is used outside the convergence radius of the polynomial, resulting in this behaviour. On the contrary, the examples up to order 4 show quite a good result. To conclude, if the polynomials are used properly (inside their region of convergence), some improvements with respect to order 2 may arise. However, these enhancements were not significant in this case, therefore the use of higher than second order is not justified.

Table 2. High order effects on the mono-dimensional landing problem.

Order	Iterations Accepted	Iterations Total	Constraint Violation	Computational time [s]
2	87	124	7.7E-6	3.7
3	91	137	8.6E-6	3.4
4	71	83	8.5E-6	2.8
6	213	258	8.9E-6	23.6

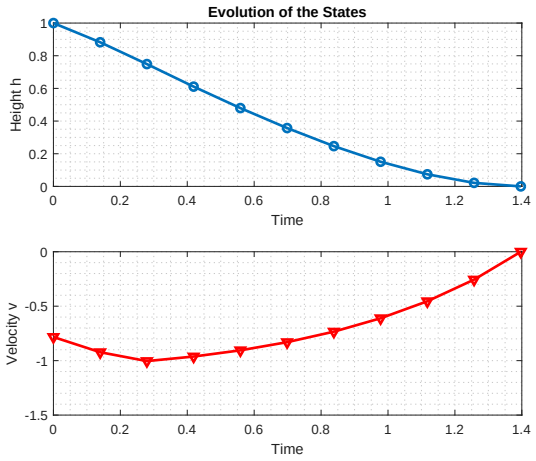

Fig. 3. Evolution of states for the mono-dimensional landing problem.

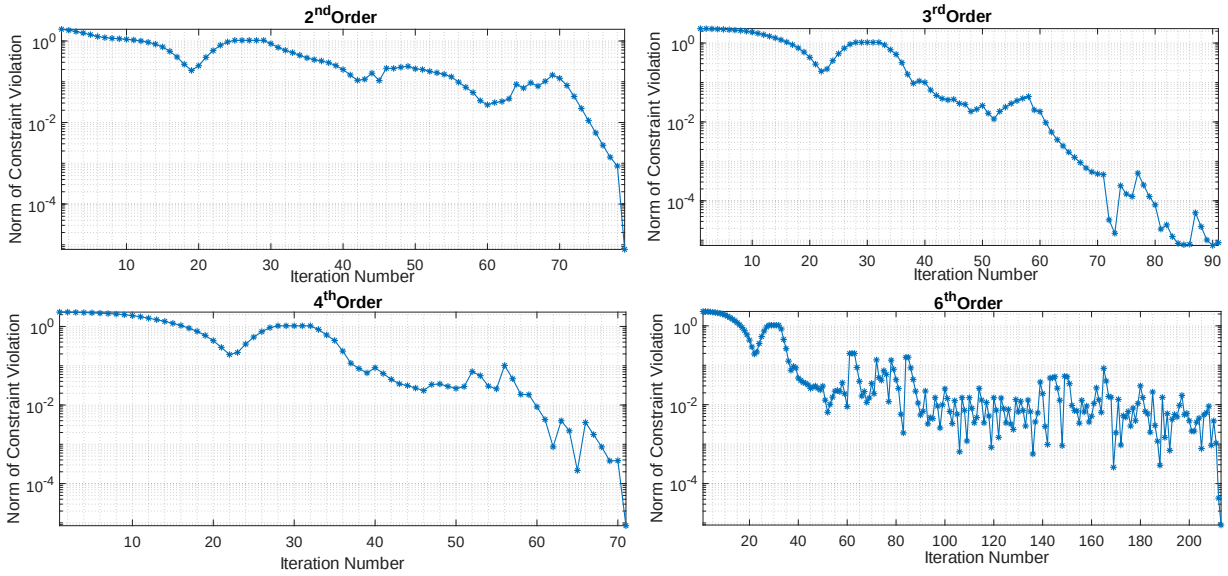

Fig. 4. Norm of constraint violation f for the mono-dimensional landing problem for orders 2(a), 3(b), 4(c) and 6(d).

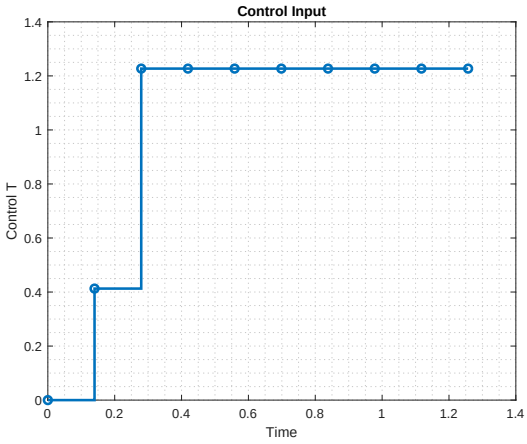

Fig. 5. Evolution of control for the mono-dimensional landing problem.

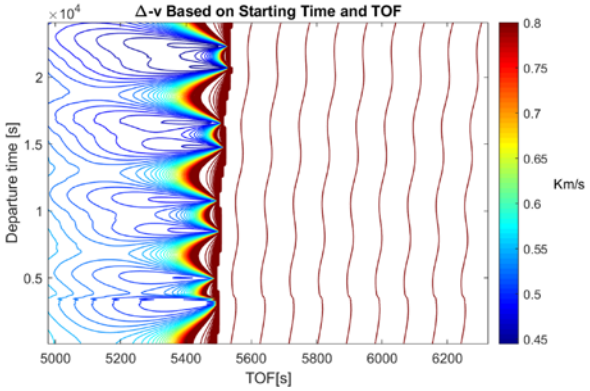

Fig. 6. Porkchop graph for the generation of the initial guess of the rephasing problem.

4.3 Satellite Rephasing

Starting from the unperturbed results, the trajectory of the transfer is reported in Fig. 7. The control trajectory for this example is reported in Fig. 8.

After the solution has been retrieved, the perturbation J2 is included in the dynamics, and the study is repeated. The solution is extremely close to the unperturbed one, as the time of flight (~ 2 h) is not sufficiently large for the effects of the perturbation to appear on the final solution. The only modification necessary to run this example with respect to the previous one is the change in dynamics for the ODE propagation, as DA will automatically compute the higher order derivatives. The most significant differences can be observed in the thrust magnitude in Fig. 9. Moreover, a discrepancy can be observed in the different Lagrange parameters reached at convergence in Table 3.

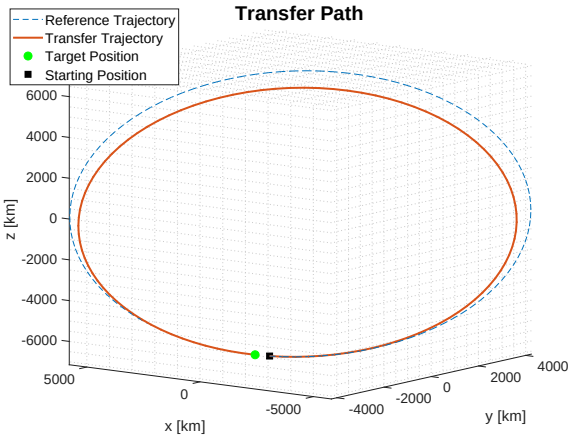

Fig. 7. Optimal transfer path for the Satellite Constellation Refueling Problem.

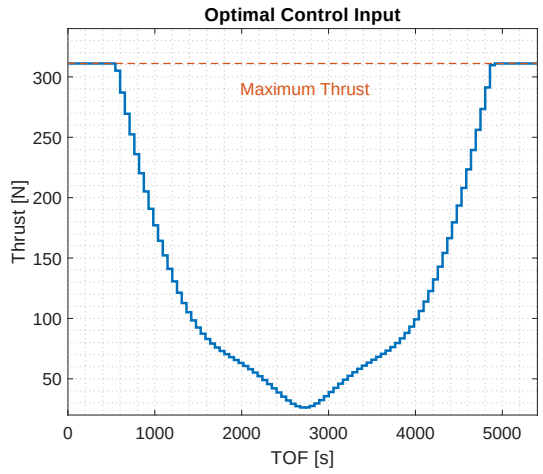

Fig. 8. Optimal control evolution for the Satellite Constellation Refueling Problem.

Table 3. Value of the Lagrange multipliers for the Perturbed and Unperturbed Rephasing Problem

	Lagrange Multipliers
Unperturbed	[-0.860903; 0.509705; 1.52568; 1.47022; -1.03679; -0.469851]
Perturbed	[-0.938731; 0.554697; 1.6653; 1.61168; -1.12907; -0.512154]

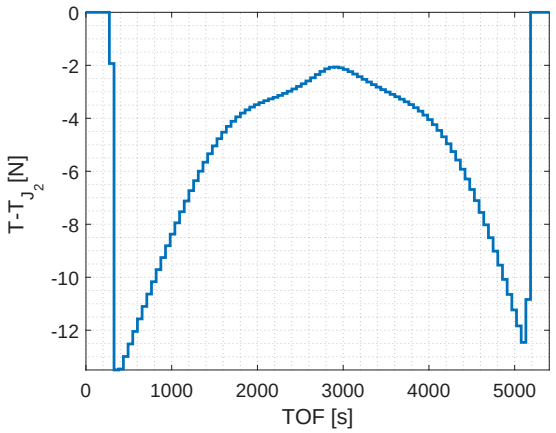

Fig. 9. Control differences between perturbed and unperturbed Rephasing Problem

5. Conclusions

This paper investigated the problem of low-thrust optimal control through the combination of Differential Algebra and Differential Dynamic Programming. The use of DA in HDDP grants the possibility of computing higher order feedback and approximations in the neighbourhood of the reference trajectory. The hypothesis formulated at the beginning of this study is that high order feedback and expansions could enlarge the convergence region of the algorithm, leading to a faster convergence with respect to standard second order methods. Nonetheless, this hypothesis seems to be disproved. Through the examples, a clear improvement in the number of iterations for higher orders is not verified. In addition, the higher order feedback terms seem to worsen numerical errors of the algorithm. In particular, the constraint violation is smoother for lower orders and becomes noisy for higher orders. Moreover, even though in some cases the number of iterations necessary for convergence of the algorithm does not change substantially, the computational time increases (see Table 2). Having addressed this issue, the thesis improves the HDDP algorithm by relieving some of the workload on the user. The difficulty in implementing the model for perturbed dynamics arises when the necessary partial derivatives of the dynamics need to be computed. The problems are evidenced by Pellegrini [45]. For complicated dynamical models, the partials required for the propagation of the dynamics must be computed with a symbolic manipulator software and sometimes they are not even possible to obtain. Moreover, these partials need to be converted into useful code (i.e. C++, Fortran etc.). Usually, these symbolic manipulators yield inefficient code, largely due to insufficient factoring. The use of DA to reduce the user's effort is demonstrated in this work, and completely removes these difficulties as DA will automatically take care of computing the higher order partials. An example is given in this study by introducing the J2 dynamical perturbation, which is handled by the same algorithm as the unperturbed case.

As any preliminary study, this work may suffer from several limitations. First of all, the algorithm still relies on a Hessian shifting technique to guarantee a convex objective function. This technique is based on a quadratic trust region procedure. Secondly, the algorithm does not improve the standard limitations of HDDP as far as tuning is concerned. In fact, a lot of tuning parameters must be set accurately, in order for this solver to converge to a solution. Finally, the scaling is performed via a non-automatic procedure, which slows down solution considerably. On the other hand, this work provides several contributions. First, it provides a first attempt to apply Differential Algebraic techniques in the Differential Dynamic Programming. By doing this, it creates the first building block for further research in this direction. In addition, it adds to the current Differential Dynamic Programming literature by exploiting high order nonlinear optimal feedback controls, capable of dealing with constraints. Lastly, this paper improves the user's experience with HDDP software by removing the tedious step of obtaining partials from symbolic external software and interfacing it with Fortan/C code. Finally, taking into consideration the above-mentioned limitations and contributions, future researches might obtain enhanced results, taking this thesis as a starting point. For example, an alternative to the Hessian shifting problem could be formulated exploiting high-orders.

Indeed, the cost function is not quadratic, therefore, an optimal expansion point where the Hessian is positive definite could be located. Afterwards, the trajectory could be re-expanded about this new reference condition, so that a descent direction is guaranteed. Furthermore, the high-order terms could be better exploited by using the convergence radius of the cost function to estimate the elliptical region of convergence automatically. This improvement could, at the same time, remove several tuning parameters from the algorithm.

References

[1] F. Bernelli-Zazzera, M. Lavagna, R. Armellin, P. Di Lizia, A. Morselli, J. Olympio, D. Izzo, L. Summerer, Trajectory optimisation under uncertainties, ESA/ACT, Ariadna Final Report, id:10/4101, Oct. 1, 2012, Contract Number: 4000103161.

[2] G. Lantoine, R.P. Russell, A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory, Journal of Optimization Theory and Applications, 154 (2012) 382–417.

[3] J.D. Aziz, D. Scheeres, G. Lantoine, Differential dynamic programming in the three-body problem, 28th Space Flight Mechanics Meeting, Kissimmee, Florida, 2018, 8-12 January.

[4] O. Von Stryk, R. Bulirsch, Direct and indirect methods for trajectory optimization, Annals of Operations Research, 37(1992) 357–373.

[5] A. Betts, Practical methods for optimal control using nonlinear programming, Applied Mechanics Reviews, 55 (2002) 68-68.

[6] M. Osborne, On shooting methods for boundary value problems, Journal of Mathematical Analysis and Applications, 27 (1969) 417–433.

[7] A. Rao, K. Mease, Dichotomic basis approach to solving hyper-sensitive optimal control problems, Automatica, 35 (1999) 633–642.

[8] J.T. Betts, W.P. Huffman, Trajectory optimization on a parallel processor, Journal of Guidance, Control, and Dynamics, 14 (1991) 431–439.

[9] D. Murray, S. Yakowitz, The application of optimal control methodology to nonlinear programming problems, Mathematical Programming, 21 (1981) 331–347.

[10] G. Lantoine, A Methodology for Robust Optimization of Low-thrust Trajectories in Multibody Environments. Georgia Institute of Technology, 2010.

[11] D. Kraft, On converting optimal control problems into nonlinear programming problems, in: Computational Mathematical Programming, K. Schittkowski, Ed., ser. NATO ASI Series, Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 261–280.

[12] S. Dreyfus, Dynamic programming and the calculus of variations, Journal of Mathematical Analysis and Applications, 1 (1960) 228–239.

[13] A. Bryson, Dynamic Optimization., Addison-Wesley Longman, California, 1999.

[14] H.W. Kuhn, A.W. Tucker, Nonlinear programming, Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, California, 1950, 31 July-12 August.

[15] D.G. Luenberger, Y. Ye, Linear and nonlinear programming, in, ser. International Series in Operations Research & Management Science vol 116, Springer US, 2008.

[16] R. Fletcher, Practical Methods of Optimization, Wiley, New York, 1987.

[17] M.S. Bazaraa, H.D. Sherali, C.M. Shetty, Nonlinear programming: Theory and algorithms, Wiley, New York, 2005.

[18] P. Patel, D.J. Scheeres, A second-order optimization algorithm using quadric control updates for multistage optimal control problems, Optimal Control Applications and Methods, 30 (2009) 525–536.

[19] W. Murray, Analytical expressions for the eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions, Journal of Optimization Theory and Applications, 7 (1971) 189–196.

[20] D.M. Murray, S.J. Yakowitz, Constrained differential dynamic programming and its application to multireservoir control, Water Resources Research, 15 (1979) 1017– 1027.

[21] M. Powell, Algorithms for nonlinear constraints that use Lagrangian functions, Mathematical Programming, 14 (1978) 224–248.

[22] M. Hestenes, Multiplier and gradient methods, Journal of Optimization Theory and Applications, 4 (1969) 303–320.

[23] D. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, Academic Press, New York, 1982.

[24] A.R. Conn, N.I.M. Gould, P. Toint, A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds, SIAM Journal on Numerical Analysis, 28 (1991) 545–572.

[25] S. Yakowitz, The stagewise Kuhn-Tucker condition and differential dynamic programming, IEEE Transactions on Automatic Control, 31 (1986) 25–30.

[26] D. Jacobson, D. Mayne, Differential Dynamic Programming, American Elsevier Publ. Co., New York, 1970.

[27] L. Liao, C. Shoemaker, Advantages of differential dynamic programming over newton's method for

discrete-time optimal control problems, Cornell University, New York, 1992.
[28] R. Bellman, Dynamic Programming, 1st ed., Princeton University Press, New Jersey, 1957.
[29] S. Yakowitz, Algorithms and computational techniques in differential dynamic programming, Advances in Aerospace Systems Dynamics and Control Systems, 31 (1989) 75–91.
[30] G. Whiffen, Static/dynamic control for optimizing a useful objective, US patent 6496741, Dec. 2002.
[31] G. Lantoine, R.P. Russell, A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 2: Application, Journal of Optimization Theory and Applications, 154 (2012) 418–442.
[32] G. Whiffen, Mystic: Implementation of the static dynamic optimal control algorithm for high-fidelity, low-thrust trajectory design, AAS/AIAA Astrodynamics Specialist Conference, Keystone, Colorado, 2006, 21-24 August.
[33] G. Lantoine, R.P. Russell, A fast second-order algorithm for preliminary design of low-thrust trajectories, IAC-08-C1.2.5, 59th International Astronautical Congress, Glasgow, United Kingdom, 2008, 29 September-3 October.
[34] G. Lantoine, R.P. Russell, A hybrid differential dynamic programming algorithm for robust low-thrust optimization, AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Honolulu, Hawaii, 2008, 18-21 August.
[35] E. Pellegrini, R.P. Russell, A multiple-shooting differential dynamic programming algorithm, AAS/AIAA Space Flight Mechanics Meeting, San Antonio, Texas, 2017, 5-9 February.
[36] N. Ozaki, S. Campagnola, C.H. Yam, R. Funase, Differential dynamic programming approach for robust-optimal low-thrust trajectory design considering uncertainty, 25th International Symposium on Space Flight Dynamics, Munich, Germany, 2015, 19-23 October.
[37] J. Gil-Fernández, M. Graziano, M.A. Gómez-Tierno, E. Milic, Autonomous low-thrust guidance: Application to SMART-1 and BepiColombo, Annals of the New York Academy of Sciences, 1017 (2004) 307–327.
[38] E. Theodorou, Y. Tassa, E. Todorov, Stochastic differential dynamic programming, Proceedings of the 2010 American Control Conference, Baltimore, Maryland, 2010, 30 June-2 July.
[39] R. Moore, Interval Analysis. Prentice-Hall, New Jersey, 1966.
[40] F. Bernelli-Zazzera, M. Vasile, M. Massari, P. Di Lizia, M. Markót, Assessing the accuracy of interval arithmetic estimates in space flight mechanics, ESA/ACT, Ariadna Final Report id:04/4105, Contract Number: 18851/05/NL/MV, Jan. 8, 2006, pp. 1–189.
[41] P. Di Lizia, R. Armellin, F. Bernelli-Zazzera, M. Berz, High order optimal control of space trajectories with uncertain boundary conditions, Acta Astronautica, 93 (2014) 217–229.
[42] M. Berz, Modern Map Methods in Particle Beam Physics, Academic Press, United Kingdom, 1999.
[43] M. Berz, Differential algebraic treatment of beam dynamics to very high orders including applications to spacecharge, AIP Conference Proceedings, San Diego, California, 1988, 19-21 January.
[44] M. Berz, K. Makino, COSY INFINITY version 8.1 user's guide and reference manual, April 2001, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.201.2818&rep=rep1&type=pdf, (accessed 26.08.2018).
[45] E. Pellegrini, R.P. Russell, Multiple-shooting differential dynamic programming with applications to spacecraft trajectory optimization, UT Electronic Theses and Dissertations, Austin, Texas, 2017.