(Preprint) AAS 23-365

A HIGHER-ORDER DIFFERENTIAL CORRECTION SCHEME FOR THE TWO-POINT BOUNDARY VALUE PROBLEM

Sharad Sharan[*], Roshan T. Eapen[†], Puneet Singla[‡], Robert G. Melton[§]

The conventional differential corrections approach for solving two-point boundary value problems (TPBVP) using Newton's method has several limitations, such as slow convergence rates, limited validity regions, and sensitivity to initial guesses. This paper proposes an alternative fourth-order iterative scheme, inspired by the super-Halley method, for solving TPBVPs. This approach utilizes higher-order state transition tensors computed using an efficient data-driven technique, that involves optimal sampling in a domain around a reference trajectory, to obtain a polynomial model for differential corrections. Improved robustness, reduced computational cost, and simplicity of implementation are demonstrated using examples.

INTRODUCTION

Two-Point Boundary Value Problems (TPBVPs) are often encountered in several engineering disciplines. In astrodynamics, a classic example is the Lambert problem, that seeks a path connecting two fixed points in a given amount of time is sought.[1] Indirect methods to approach an optimal control problem tend to reduce it to a TPBVP, following the derivation of the state and co-state equations.[2] For most nonlinear systems, an analytical solution to this problem does not exist, and one has to resort to iterative numerical methods to solve the TPBVP.[2–4] One such method is differential correction (DC),[2,5] also known as the shooting method.

The DC scheme has been used in various applications in astrodynamics, for example, the determination of periodic orbits[6–8] in the Circular Restricted three Body Problem (CR3BP), low-thrust spacecraft trajectory targeting,[9] station-keeping for geostationary spacecraft,[10] and spacecraft formation flying,[11] to name a few. In the DC process, the state dynamical equations are integrated using an initial guess for the unknown states and parameters. The terminal conditions are then defined in terms of an objective function, such that a root of that function indicates a satisfactory solution to the TPBVP. Determining the root of a nonlinear function necessitates an iterative numerical scheme. The nature of the iterative scheme determines the quality of the correction step.

The iterative scheme generally employed in the DC process is Newton's method.[12] It has quadratic convergence, and is often the preferred method for use in the DC process. This preference is due to the fact that the Newton's scheme only requires first-order sensitivities of the terminal conditions

[*]Graduate student, Department of Aerospace Engineering, Pennsylvania State University, State College, PA-16802.
[†]Assitant Professor, Department of Aerospace Engineering, Pennsylvania State University, State College, PA-16802.
[‡]Professor, AIAA Associate Fellow, AAS Fellow, Department of Aerospace Engineering, Pennsylvania State University, State College, PA-16802.
[§]Professor, AAS Fellow, AIAA Associate Fellow, Department of Aerospace Engineering, Pennsylvania State University, State College, PA-16802.

with respect to the initial guess. Determining these sensitivities involves the computation of partial derivatives of the system dynamics with respect to the unknown initial states and parameters, which may not always be readily available. Determining first-order sensitivities may itself prove to be an arduous task for dynamical systems. Therefore, alternate iterative schemes employing higher-order sensitivities, which require higher-order partial derivatives, are seldom used on grounds of practicality.

However, an issue with using Newton's scheme in a DC process lies in the fact that the correction step is derived from a linear Taylor series approximation of the nonlinear function around the initial guess. It is illustrated later in the paper that the linear Taylor series approximation about any reference is only valid for a very small region around the reference. Thus, in order for a good correction step that subsequently leads to the root of the aforementioned objective function, the initial guess must be close enough to the actual solution. This makes the Newton's scheme highly sensitive to the initial guess. In the context of a TPBVP, at times, the unknown initial conditions and parameters may not yield appropriate physical intuition for one to make an educated initial guess.

In order to tackle this problem of coming up with a good initial guess, heuristic methods are often used,[13] a well renowned one being the particle swarm optimization technique.[14,15] In such cases, a candidate solution is sought within a given range for each unknown initial guess through random initialization and subsequent propagation to check if terminal conditions are being satisfied. Given a large pool of randomized initial guess samples, some samples are bound to converge to a solution within a specified tolerance. Note that the heuristic methods demand a lot of computational power, and is heavily dependent on the desired tolerance for the solutions being sought by such methods. Since the Newton's method has stringent requirements on the quality of the initial guess, the aforementioned tolerance for the heuristic method must be sufficiently low in order to generate a good initial guess for the Newton's method to work with. A consequence of low tolerance is high computation cost.

On the other hand, the stringent requirement of a high-quality initial guess can be reduced by employing correction schemes that use higher-order sensitivities, that in turn are more robust to the initial guess. However, recall that obtaining higher-order sensitivities is challenging, therefore returning us to the very reason why Newton's method is widely preferred. The challenge of employing higher-order iterative schemes for a relaxed initial guess, coupled with the computational cost of heuristic methods to generate a good initial guess for the Newton's scheme, form a notorious loop that is difficult to break. The undoing of this loop in its entirety is the goal of this paper. This goal is approached by defining two objectives.

The first objective seeks to derive a higher-order correction scheme applicable to the DC process, that is more robust to the initial guess than the conventional Newton's method. Newton's method only uses first-order sensitivities. A well-known scheme that uses second-order sensitivities is the Halley's method,[16] which exhibits third-order convergence.[17] Several modifications to the Halley's scheme have been explored, like the super-Halley method,[18] followed by a variant of super-Halley with fourth-order convergence.[19] However, methods using third-order sensitivities are rarely investigated, as the cost of evaluating third-order partial derivatives is much too high for any meaningful reward in terms of computation. In this work, an iterative scheme that uses fourth-order sensitivity information is proposed, inspired by the derivation of Halley's method. This scheme allows one to come up with an initial guess that is not very close to the actual solution, yet converges to the solution by using fourth-order sensitivity information at subsequent iteration points.

The second objective seeks to address the computational aspect of implementing the proposed higher-order correction scheme. Recall that the major impediment is the evaluation of partial derivatives in order to determine the appropriate sensitivities to effect the higher-order correction. In this paper, a derivative-free, stochastic approach is employed to determine these higher-order sensitivities in a given domain. First, a domain is defined around the reference trajectory's parameters of interest at the initial time. A deterministic sampling of the domain is performed using Conjugate Unscented Transform (CUT) points. CUT has been demonstrated to efficiently capture the characteristics of a domain with good accuracy,[20] and has been applied to various problems successfully.[21–24] The parameters of interest at the final time are evaluated for the sampled initial points, using which a least-squares approach is adopted to fit a polynomial model to the sampled data. This polynomial model maps the final state and parameters to the initial state and parameters, and is valid in the aforementioned domain. The coefficients of the polynomial model are nothing but the different sensitivities of the system that one needs in order to perform a DC process. The order of the sensitivities depends upon the order of the polynomial model. A detailed implementation of this procedure is discussed in the methodology section. Thus, the higher-order sensitivities are obtained in a computationally efficient manner without need for knowledge of any partial derivatives.

For convenience, the aforementioned higher-order sensitivities are referred to as CUT-STT henceforth in the paper. Note that although it is termed as the CUT-STT, it involves the sensitivities of other parameters in the system in addition to the states. Therefore, it is not a direct equivalent to the State Transition Tensor (STT), which only involves the sensitivities of the states of the system between two given time instances. These CUT-STTs have been demonstrated to work well in the context of uncertainty propagation in the CR3BP.[25] They are employed in this paper to tackle the aforementioned impediment of computational burden in using higher-order schemes in the DC process.

The paper is organized as follows. First, a description of the TPBVP is provided to describe the issues associated with it, followed by a detailed motivation for the current research objectives. Following this, traditional methods of DC that are employed to address a TPBVP are discussed, before elaborating on the higher-order scheme proposed in this work. Subsequently, the CUT-STT, which alleviates the computational difficulty associated with the implementation of the higher-order scheme, is discussed. The higher-order scheme and the CUT-STT are then combined, and applied to a few well-known problems to demonstrate their advantages, and address the disadvantages.

PROBLEM STATEMENT

Consider a dynamical system of the form

$$\dot{\mathbf{x}} = \mathbf{f}[t, \mathbf{x}(t), \boldsymbol{\eta}] \tag{1}$$

where $\mathbf{x}$ is an n-dimensional vector of the states of the system, and $\boldsymbol{\eta}$ is a vector of additional parameters dictating system dynamics. A set of initial conditions is generally known based on the problem specifications, i.e.,

$$x_i(t_0), \text{ for } i = 1, 2, \ldots n_1 \tag{2}$$

If $n_1 = n$, then this constitutes a well-posed Initial Value Problem (IVP). However, based on the problem characteristics, one does not always end up with an IVP. In addition to Eq. (2), the problem may specify certain terminal conditions, given by

$$\psi_j(t_f, \mathbf{x}(t_f), \boldsymbol{\eta}), \text{ for } j = 1, 2, \ldots n_2 \tag{3}$$

Some systems also possess certain states and parameters defined by algebraic functions, thereby giving rise to Differential Algebraic Equation (DAE) systems. The knowledge of certain boundary conditions at the initiation of the integration, and the rest at its termination, constitutes what is known as the TPBVP, given by Eqs. (1) - (3). For the TPBVP to be well-posed, $n_1 + n_2 = n$.

Apart from a few simple problems, solving a TPBVP is generally quite challenging.[26] Several numerical techniques exist for solving a TPBVP.[2–4] This paper is concerned with the method of DC.[2,5]

Differential Correction

The DC process begins by assuming guesses for the unknown initial conditions of the TPBVP, thereby formulating an IVP. The guess is usually made for unknown parameters, or states, or both at the initial time, and the dynamics is propagated to the final time. Note that there are problems where the final time (t_f) is not known, and t_f becomes one of the parameters that one needs to solve for in the TPBVP. In such cases, time is non-dimensionalized by defining $\tau = \frac{t}{t_f}$, and the integration is carried out for $0 < \tau < 1$. The dynamical equations are modified appropriately to reflect τ as the independent variable.

The guessed initial conditions and parameters which are to be corrected at each iteration are referred to as the design variables ($\boldsymbol{\theta}$). With the known initial conditions and the guessed design variables, one can propagate the system dynamics using Eq. (1) to the final time. At the end of the integration, the error between the obtained terminal conditions corresponding to the design variables, and the required terminal conditions dictated by the TPBVP is calculated. The goal of DC is to drive this error to zero, theoretically, through successive corrections applied to the guess variables. This error can be written as an objective function defined at t_f,[27]

$$\mathbf{F}(\boldsymbol{\theta}) = \boldsymbol{\psi}(t_f, \boldsymbol{\theta}) - \boldsymbol{\psi}_{desired} \tag{4}$$

where $\boldsymbol{\psi}(t_f, \boldsymbol{\theta})$ represents the terminal conditions obtained as a function of the design variables $\boldsymbol{\theta}$, and $\boldsymbol{\psi}_{desired}$ is given by Eq. (3). Now, a numerical root-finding scheme is required to find $\boldsymbol{\theta}$ such that $\mathbf{F}(\boldsymbol{\theta}) = 0$.

A commonly used iterative scheme in DC is the Newton's method, which has quadratic convergence.[12] In order to find a solution to any $F(\boldsymbol{\theta}) = 0$, Newton's iterative method can be derived from a first-order Taylor series expansion of $F(\boldsymbol{\theta})$ about any guessed value of $\boldsymbol{\theta}$. Consider the difference between this guess value and the actual root to be $\delta\boldsymbol{\theta}$. Using a first-order Taylor series expansion, one can solve for this $\delta\boldsymbol{\theta}$, in order to theoretically arrive at the actual root.

Let $\boldsymbol{\theta}^* = \boldsymbol{\theta} + \delta\boldsymbol{\theta}$ be a root of $F(\boldsymbol{\theta})$. Applying a Taylor expansion up to the first-order,

$$F(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) = F(\boldsymbol{\theta}) + F'(\boldsymbol{\theta})\delta\boldsymbol{\theta} \tag{5}$$

where the required first-order derivative is given by

$$F'_{ij}(\boldsymbol{\theta}) = \frac{\partial F_i(\boldsymbol{\theta})}{\partial \theta_j} \qquad i, j = 1 \dots n \tag{6}$$

where n is the dimension of the system. However, $\boldsymbol{\theta} + \delta\boldsymbol{\theta}$ is assumed to be a root, therefore $F(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) = 0$. Eq. (5) now becomes

$$0 = F(\boldsymbol{\theta}) + F'(\boldsymbol{\theta})\delta\boldsymbol{\theta} \tag{7}$$

$$\implies \delta\boldsymbol{\theta} = -[F'(\boldsymbol{\theta})]^{-1} F(\boldsymbol{\theta}) \tag{8}$$

The issue with this however, is the fact that a first-order Taylor series expansion is simply an approximation of the actual nonlinear function, valid in a small neighborhood around the guess value of $\boldsymbol{\theta}$. Therefore, simply adding $\delta\boldsymbol{\theta}$ to $\boldsymbol{\theta}$ only yields the root of the linear approximation. A simple solution to this issue is to use the approximated root as a new guess, and find the roots of successive linear approximations iteratively, until the approximated root converges to the actual root. This is the essence of the Newton's root-finding algorithm, given by

$$\boldsymbol{\theta}_{p+1} = \boldsymbol{\theta}_p - [F'(\boldsymbol{\theta}_p)]^{-1} F(\boldsymbol{\theta}_p) \tag{9}$$

where p is the iteration index. Note that when applied to the DC process under discussion, Eq. (6) is essentially a matrix of first-order sensitivity of the terminal condition to the initial guess. The determination of this sensitivity is later discussed in detail.

Now, the updated initial design variables given by Eq. (9) are used to solve a new IVP, and the root of the objective function given in Eq. (4) is evaluated again. Since the updates were effected using a linear approximation, the desired terminal condition is seldom satisfied after just one iteration of the process. Therefore, this process is performed repeatedly until $|\mathbf{F}(\boldsymbol{\theta})| < \epsilon$ for practical purposes, where ϵ is a specified tolerance.

In summary, the steps involved in the DC process can be outlined as follows.[28]

1. Guess unknown initial conditions and parameters, $\boldsymbol{\theta} = [\mathbf{x}(t_0), \boldsymbol{\eta}]$ to formulate an IVP.
2. Propagate system dynamics from initial to final state using Eq. (1).
3. Calculate the error in terminal condition $\mathbf{F}(\boldsymbol{\theta})$ using Eq. (4). Terminate process if $|\mathbf{F}(\boldsymbol{\theta})| < \epsilon$.
4. Utilize Newton's method to find a correction to the guessed design variables $\boldsymbol{\theta}$, using Eq. (8). Use the new $\boldsymbol{\theta}$ to set up a new IVP. Repeat steps $2 - 4$.

Since a linear approximation is assumed at each iteration to generate new guess variables, the quality of the initial guess is of high importance, as the guess variables must lie within a reasonable neighborhood of the actual root of $F(\boldsymbol{\theta})$. Depending on the quality of the initial guess, the DC process may converge after a number of iterations, or diverge altogether. In order to demonstrate the approximation qualities of different order approximations, let us consider a Taylor series expansion of a simple univariate function,

$$g(x) = e^x - 2x^2 \tag{10}$$

An m^{th} order Taylor series expansion of this function is given by

$$g(x) = \sum_{i=0}^{m} \frac{g^{(i)}(x^*)}{i!} \delta x^i \tag{11}$$

where $g^{(i)}(x^*)$ refers to the i^{th} derivative of $g(x)$ evaluated at x^*. Approximations of Eq. (10) about the reference point $x^* = 1$, up to the seventh order, are illustrated in Fig. 1. Note that the first-order approximation is only valid in a very small neighborhood around the reference point $x^* = 1$. As the approximation order increases, the size of the neighborhood where the approximation follows the true nonlinear function also increases. Eventually, a seventh-order Taylor approximation was found to exactly follow the example function $g(x)$ in the interval illustrated in Fig. 1. This example

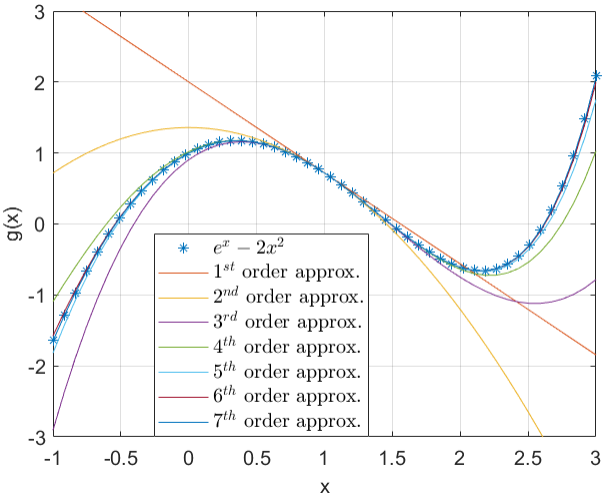


Figure 1: Example of Taylor series approximations of varying orders

motivates the primary question addressed in this paper: if higher-order approximations are valid in a larger neighborhood of the reference point (this is the initial guess in the context of a DC process), why not use a higher-order iterative scheme in the DC process to obtain updates that are closer to the actual root at each iteration, compared to the linear approximation in Newton's method? This leads to the first objective of this paper.

Objective 1 Derive a higher-order root-solving scheme for the DC process.

If one can formulate such a higher-order scheme, what are its pros and cons? Computational tractability is an important factor when it comes to solving numerically challenging systems. Recall that the traditional methodology to calculate higher-order sensitivities is a computationally expensive task. This leads to the second objective.

Objective 2 Tackle the computational challenge of determining higher-order sensitivities, so that implementation of the developed higher-order scheme is feasible.

The following section explores some conventional higher-order schemes from existing literature, and outlines the proposed methodologies in this paper.

METHODOLOGY

Following the description of the DC process in the previous section, it is clear that the contributions in this paper revolve around the root-finding portion of the DC process. To this end, some commonly used iterative root-solving schemes are reviewed in the first part of this section. Following this, the proposed fourth-order scheme is outlined, applied to a simple problem, and compared with the traditional methods. In dynamical systems, higher-order iterative schemes are scarcely preferred due to the high computational complexity of higher-order sensitivities necessary for the DC process. However, we adopt a computationally inexpensive methodology of finding those sensitivities, which ultimately opens the door to implementing higher-order schemes without much computational effort. This methodology is outlined in the second subsection.

Higher-order correction schemes

Beyond the Newton's method, a well-known higher-order scheme is the Halley's method,[16] also known as the method of tangent hyperbolas. In order to derive this method, the Taylor series expansion given in Eq. (5) is extended to include the second derivative $F''(\boldsymbol{\theta})$.[29]

$$F(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) = F(\boldsymbol{\theta}) + F'(\boldsymbol{\theta})\delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^T F''(\boldsymbol{\theta})\delta\boldsymbol{\theta} \tag{12}$$

$$F''_{ijk}(\boldsymbol{\theta}) = \frac{\partial^2 F_i(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k} \qquad i, j, k = 1 \dots n \tag{13}$$

Now, the goal is to determine a value for $\delta\boldsymbol{\theta}$, such that $\boldsymbol{\theta} + \delta\boldsymbol{\theta}$ is a root of F. Thus, Eq. (12) becomes

$$0 = F(\boldsymbol{\theta}) + F'(\boldsymbol{\theta})\delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^T F''(\boldsymbol{\theta})\delta\boldsymbol{\theta} \tag{14}$$

$$\implies -F(\boldsymbol{\theta}) = \left[F'(\boldsymbol{\theta}) + \frac{1}{2}F''(\boldsymbol{\theta})\delta\boldsymbol{\theta}\right]\delta\boldsymbol{\theta} \tag{15}$$

Substituting Eq. (8) for the $\delta\boldsymbol{\theta}$ within the bracket,

$$-F(\boldsymbol{\theta}) = \left[F'(\boldsymbol{\theta}) - \frac{1}{2}F''(\boldsymbol{\theta})[F'(\boldsymbol{\theta})]^{-1}F(\boldsymbol{\theta})\right]\delta\boldsymbol{\theta} \tag{16}$$

$$\therefore\ \delta\boldsymbol{\theta} = -\left[F'(\boldsymbol{\theta}) - \frac{1}{2}F''(\boldsymbol{\theta})F'(\boldsymbol{\theta})^{-1}F(\boldsymbol{\theta})\right]^{-1} F(\boldsymbol{\theta}) \tag{17}$$

where $\delta\boldsymbol{\theta}$ is the Halley's update. Note that $(\boldsymbol{\theta} + \delta\boldsymbol{\theta})$ is only an approximation of the root of a second-order approximation of F, since it uses Newton's update as an intermediary. Therefore, similar to the Newton's method, an iterative algorithm is required to converge on the root of the actual nonlinear function F. Using Eq. (17), the Halley's algorithm can be written as

$$\boldsymbol{\theta}_{p+1} = \boldsymbol{\theta}_p - \left[F'(\boldsymbol{\theta}_p) - \frac{1}{2}F''(\boldsymbol{\theta}_p)F'(\boldsymbol{\theta}_p)^{-1}F(\boldsymbol{\theta}_p)\right]^{-1} F(\boldsymbol{\theta}_p) \tag{18}$$

Halley's method has been shown to exhibit cubic convergence.[17] Several variants of the Halley's method have been explored in literature, a notable one being the super-Halley method, given by[18]

$$\boldsymbol{\theta}_{p+1} = \boldsymbol{\theta}_p - \left[I + \frac{1}{2}L_F(\boldsymbol{\theta}_p)[I - L_F(\boldsymbol{\theta}_p)]^{-1}\right][F'(\boldsymbol{\theta}_p)]^{-1}F(\boldsymbol{\theta}_p) \tag{19}$$

$$L_F(\boldsymbol{\theta}_p) = [F'(\boldsymbol{\theta}_p)]^{-1}F''(\boldsymbol{\theta}_p)[F'(\boldsymbol{\theta}_p)]^{-1}F(\boldsymbol{\theta}_p) \tag{20}$$

A variant of the super-Halley method with fourth-order convergence can be found in literature.[19,30] In such a scheme, the operand of the $F''(.)$ operator is modified within the $L_F(.)$ operator in Eq. (20) as

$$L_F(\boldsymbol{\theta}_p) = [F'(\boldsymbol{\theta}_p)]^{-1}F''(\mathbf{u}_p)[F'(\boldsymbol{\theta}_p)]^{-1}F(\boldsymbol{\theta}_p) \tag{21}$$

$$\mathbf{u}_p = \boldsymbol{\theta}_p - \frac{1}{3}[F'(\boldsymbol{\theta}_p)]^{-1}F(\boldsymbol{\theta}_p) \tag{22}$$

This fourth-order super-Halley method can be extended to utilize more information about the surface in question, by including the third derivative of $F(\boldsymbol{\theta})$, given by

$$F'''_{ijkq}(\boldsymbol{\theta}) = \frac{\partial^3 F_i(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k \partial\theta_q} \qquad i,j,k,q = 1 \ldots n \tag{23}$$

The proposed scheme can be derived from a Taylor series expansion for F about one of its zeros, similar to the derivation shown for the Halley's method, and is given by

$$\boldsymbol{\theta}_{p+1} = \boldsymbol{\theta}_p - B^{-1}(\boldsymbol{\theta}_p) F(\boldsymbol{\theta}_p) \tag{24}$$

$$B(\boldsymbol{\theta}_p) = F'(\boldsymbol{\theta}_p) + \frac{A^T(\boldsymbol{\theta}_p)}{6}\left[3F''(\boldsymbol{\theta}_p) + A^T(\boldsymbol{\theta}_p) F'''(\boldsymbol{\theta}_p)\right] \tag{25}$$

$$A(\boldsymbol{\theta}_p) = -\left[F'(\boldsymbol{\theta}_p) - \frac{1}{2}F''(\boldsymbol{\theta}_p)F'(\boldsymbol{\theta}_p)^{-1}F(\boldsymbol{\theta}_p)\right]^{-1} F(\boldsymbol{\theta}_p) \tag{26}$$

Note that $A(\boldsymbol{\theta}_p)$ is simply Halley's update, which is used as an intermediate approximation, and combined with the information provided by $F'''(\boldsymbol{\theta}_p)$ to calculate the update term in the proposed higher-order scheme given by Eq. (24). The proposed iterative scheme given by Eqs. (24) - (26) is referred to as the Third-Order Root Solver (TORS). The nomenclature here reflects the highest-order derivative of $F(\boldsymbol{\theta})$, third in this case, that one would require in order to implement the scheme to find the root of $F(\boldsymbol{\theta})$. This should not be confused with the order of convergence of the scheme.

Having derived the TORS, a similar procedure can be applied to go one order higher than the TORS, and utilize fourth-order derivative information given by

$$F^{(\mathrm{iv})}_{ijkqr}(\theta) = \frac{\partial^4 F_i(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k \partial\theta_q \partial\theta_r} \qquad i,j,k,q,r = 1 \ldots n \tag{27}$$

The corresponding iterative scheme that uses $F^{(\mathrm{iv})}_{ijkqr}(\theta)$ information can now be derived from a fourth-order Taylor series expansion about the root, and is given by

$$\boldsymbol{\theta}_{p+1} = \boldsymbol{\theta}_p - E^{-1}(\boldsymbol{\theta}_p) F(\boldsymbol{\theta}_p) \tag{28}$$

$$E(\boldsymbol{\theta}_p) = F'(\boldsymbol{\theta}_p) + \frac{D^T(\boldsymbol{\theta}_p)}{24}\left[12F''(\boldsymbol{\theta}_p) + 4D^T(\boldsymbol{\theta}_p)F'''(\boldsymbol{\theta}_p) + D^T(\boldsymbol{\theta}_p)F^{(\mathrm{iv})}(\theta)D(\boldsymbol{\theta}_p)\right] \tag{29}$$

$$D(\boldsymbol{\theta}_p) = -B^{-1}(\boldsymbol{\theta}_p) F(\boldsymbol{\theta}_p) \tag{30}$$

where $B(\boldsymbol{\theta}_p)$ is given by Eq. (25). Once again, note that the TORS update is used as an intermediate approximation in developing this fourth-order scheme. A nomenclature similar to the TORS is adopted, and the proposed iterative scheme given by Eqs. (28) - (30) is simply termed the Fourth-Order Root Solver (FORS). Compared to the Newton's method, one does incur more computations every iteration in order to implement the higher-order schemes. A comparison can be made by plotting approximate values of the FLoating point OPerations (FLOPs) per iteration demanded by each iterative scheme. This comparison is illustrated in Fig. 2 with respect to the dimension of a given problem.

Note that the FORS requires computations an order of magnitude higher than that required by the Newton's method. However, as is evident from the examples to follow, the higher-order schemes provide superior robustness to the initial guess, which is an important quality to have while tackling a TPBVP. Moreover, for some problems, higher-order schemes converge in significantly lesser number of iterations than Newton's method, thereby proving to be much more computationally efficient. This is entirely dependent on the problem at hand. With the derivation of the FORS, the first objective of this paper is accomplished, which is to develop a higher-order iterative scheme to solve $\mathbf{F} = 0$.

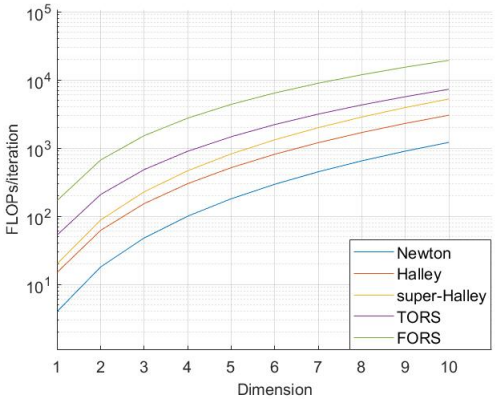


Figure 2: Comparison of FLOPs per iteration demanded by the different iterative schemes

However, it is necessary to demonstrate that the proposed iterative method works, before addressing the issue of computing higher-order derivatives. For this purpose, the Newton, Halley and FORS schemes are applied to find the root of an analytical system of nonlinear equations given by

$$16x_1^5 + 16x_2^4 + x_3^4 - 16 = 0 \tag{31}$$

$$x_1^5 - x_2^2 e^{x_1} + x_3^2 - 3 = 0 \tag{32}$$

$$x_1^5 \sin(x_3) - x_2 = 0 \tag{33}$$

Through all the three methods, the roots obtained for an initial guess of $[2, 2, 2]$ were

$$[x_1, x_2, x_3] = [0.83966136, 0.41240179, 1.72524016] \tag{34}$$

correct to twelve decimal digits of accuracy. However, the number of iterations and average execution time per iteration were different, and are tabulated in Table 1. Note that the number of

Table 1: Comparison of iteration methods

Method	**Iterations**	**Avg. time per iteration (s)**
Newton	10	0.107 34
Halley	6	0.115 30
FORS	5	0.129 79

iterations required by FORS is half that of Newton's method. The average time per iteration for

FORS is slightly larger than Newton's, due to the incorporation of derivatives up to the third order. However, the total run time of Newton's is considerably higher than FORS due to the greater number of iterations. It must also be noted that the super-Halley performs equally well in this case. In general, there are documented scenarios where the Newton's method performs just as good as Halley's.[31] Computational performance is completely dependent on the multidimensional surface under consideration, and changes from system to system, and the quality of the initial guess. However, as demonstrated in the aforementioned example, there are systems where the higher-order methods outperform Newton's method by a significant margin. Such root-solving methods are extremely useful in optimization.

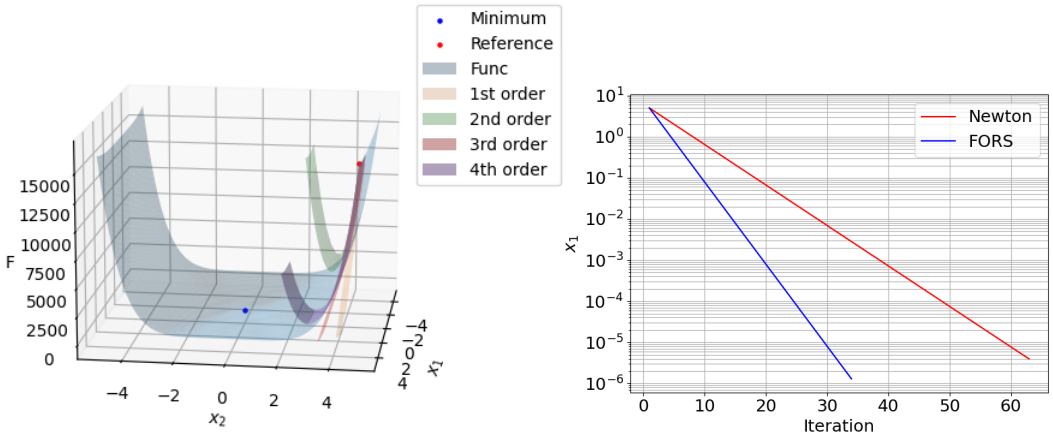
Iterative schemes are applied to find the roots of the gradient of a function, which are essentially the stationary points one looks for in optimization. Newton's method, along with several second-order variants[32–36] are widely used in optimization procedures. In order to compare such second-order methods with a higher-order method, let us consider a simple example.

$$\min_{x_1,x_2} F = x_1^6 + x_2^4 \tag{35}$$

The solution to Eq. (35) is known to be the origin of x_1 and x_2. Surface approximations of the function F up to the fourth order are illustrated in Fig. 3a. Intuitively, the derivatives of a function at a given point carry information about the function in a vicinity around that point. This is the notion behind the Taylor series. The higher the derivative, the greater the quality of information about the function in a bigger vicinity around the reference point. Note that the optimization procedure uses these derivatives at the reference point to predict where the minimum would be. Naturally, this would only be a prediction using the derivative information at the reference point, and the actual minimum may not be this point. However, this prediction is taken as a new reference point where derivatives are computed, in order to come up with another prediction. Subsequently, this iterative process converges to the minimum of the actual surface within a specified tolerance. Therefore, with more information of the higher-order derivatives of the surface at each iteration point, the minimum of F is reached with fewer iterations. This is accomplished using the FORS, as it uses a fourth-order approximation in its iterations. This fact is evident on comparing the FORS iterates with those of the second-order Newton's method, as illustrated in Fig. 3b. Note that it takes almost twice as many iterations for the Newton's method, compared to the FORS, to achieve an accuracy of 10^{-5}. Hence, FORS has great potential to be utilized in several optimization applications.

The main reason for Newton's method being widely preferred is due to the cost of calculating the higher-order derivatives, especially in systems where calculating the partial derivatives analytically is not tractable. The examples seen so far do not capture the difficulty of this effort, as the systems defined by Eqs. (31) - (33), and Eq. (35) were analytically differentiable without much effort. However, for systems governed by differential equations, it is a completely different scenario. For such dynamical systems, the F'' and F''' tensors involve the computation of partial derivatives of the state rates with respect to the states, followed by rigorous integration. This is where the usefulness of the CUT-STT is realized. In that case, the complicated system dynamics is approximated by a polynomial surogate model using the CUT-STTs, and brought down to a tractable form, similar to the aforementioned analytical system of equations. The necessary higher-order derivatives required by the FORS then become simple to evaluate analytically.

Therefore, by eliminating the need to compute partial derivatives to build the cumbersome F'' and F''' tensors in the traditional way, one can reap the rewards of implementing a higher-order root solver, without having to pay as much in terms of computational cost. On the other hand, in



(a) Function surface approximation to various orders (b) Decrease in error with respect to iteration

Figure 3: Illustration of the FORS and Newton's method in a minimization procedure

cases where a simple Newton's method is sufficient, one doesn't incur a significant loss in terms of effort expended in employing the FORS. As mentioned earlier, these CUT-STTs have already been proven to work well,[25] and the process of using them to generate a polynomial model is discussed in the following section.

Computation of sensitivities

When trying to effect a correction using an iterative root-solving scheme in the DC process, one requires knowledge of the sensitivities of the terminal states with respect to the unknown initial conditions and parameters. Recall that in the case of analytical systems, these sensitivities can be obtained by means of straightforward partial differentiation. However, in dynamical systems, this is a cumbersome process.

In a dynamical system, the sensitivities of the terminal states with respect to initial states are given by the STM, in the case of first-order sensitivity, and by an m^{th} order STT for m^{th} order sensitivities. Obtaining these STTs is a tedious process, as one has to find several partial derivatives of the system dynamics with respect to the states, in order to build the $n + n^2 + \cdots + n^m$ equations before integrating them. For example, in order to be able to apply the FORS scheme, which is known to incorporate fourth-order sensitivities, one would need to integrate Eqs. (36) - (39).[37]

$$\dot{\Phi}_{i,a} = f_{i,\alpha}\Phi_{\alpha,a} \tag{36}$$

$$\dot{\Phi}_{i,ab} = f_{i,\alpha}\Phi_{\alpha,ab} + f_{i,\alpha\beta}\Phi_{\alpha,a}\Phi_{\beta,b} \tag{37}$$

$$\dot{\Phi}_{i,abc} = f_{i,\alpha}\Phi_{\alpha,abc} + f_{i,\alpha\beta}(\Phi_{\alpha,a}\Phi_{\beta,bc} + \Phi_{\alpha,ab}\Phi_{\beta,c} + \Phi_{\alpha,ac}\Phi_{\beta,b}) + f_{i,\alpha\beta\gamma}\Phi_{\alpha,a}\Phi_{\beta,b}\Phi_{\gamma,c} \tag{38}$$

$$\begin{aligned}\dot{\Phi}_{i,abcd} = {} & f_{i,\alpha}\Phi_{\alpha,abcd} + f_{i,\alpha\beta}(\Phi_{\alpha,abc}\Phi_{\beta,d} + \Phi_{\alpha,abd}\Phi_{\beta,c} + \Phi_{\alpha,acd}\Phi_{\beta,b} + \Phi_{\alpha,ab}\Phi_{\beta,cd} \\ & + \Phi_{\alpha,ac}\Phi_{\beta,bd} + \Phi_{\alpha,ad}\Phi_{\beta,bc} + \Phi_{\alpha,a}\Phi_{\beta,bcd}) + f_{i,\alpha\beta\gamma}(\Phi_{\alpha,ab}\Phi_{\beta,c}\Phi_{\gamma,d} \\ & + \Phi_{\alpha,ac}\Phi_{\beta,b}\Phi_{\gamma,d} + \Phi_{\alpha,ad}\Phi_{\beta,b}\Phi_{\gamma,c} + \Phi_{\alpha,a}\Phi_{\beta,bc}\Phi_{\gamma,d} + \Phi_{\alpha,a}\Phi_{\beta,bd}\Phi_{\gamma,c} \\ & + \Phi_{\alpha,a}\Phi_{\beta,b}\Phi_{\gamma,cd}) + f_{i,\alpha\beta\gamma\delta}\Phi_{\alpha,a}\Phi_{\beta,b}\Phi_{\gamma,c}\Phi_{\delta,d}\end{aligned} \tag{39}$$

where the initial condition for the integration is simply $\Phi_{i,a} = 1$, if $i = a$, in the case of the STM in Eq. (36), and zero for the remaining elements of the STT. The highest-order partial derivative necessary to implement the FORS scheme is of fourth-order, given by $f_{i,\alpha\beta\gamma\delta} = \frac{\partial^4 f_i}{\partial x_\alpha \partial x_\beta \partial x_\gamma \partial x_\delta}$. Deriving these partials beyond the second order becomes cumbersome. Note that the aforementioned process only yields the sensitivities with respect to the initial states. In the case of additional parameters in the system with unknown initial values, one has to find their sensitivities too. For example, such a parameter sensitivity determination is encountered in the Zermelo problem, which is illustrated in detail in the following section.

Significant difficulties arise when calculating the higher-order partial derivatives, and applying the tensor operations given in Eqs. (37) - (39).[38] Several efforts to handle the difficulty of obtaining the partials have been explored by means of automated computational tools, a notable one being the object oriented coordinate embedding method.[39] With respect to the tensor operations, special representations of tensors were proposed by Majji et al.[38] for efficient computation. Note that most of the effort was focused on handling the difficulties presented by Eqs. (36) - (39) in determining the sensitivities. In this paper, we adopt an alternate derivative-free approach to determine the necessary sensitivities without a need for the aforementioned tensor equations.

The alternate approach is based on a Least-Squares (LS)[40] methodology. The fundamental idea behind the LS process is to fit a model to a given input-output (I/O) data. Now, consider the context of a DC process. The root-solving schemes in DC help in determining an update to the initial guess, informed by the sensitivity of the terminal condition to the initial guess. In an LS sense, the initial guess ($\boldsymbol{\theta}$) can be regarded as the input, while the terminal condition ($\mathbf{F}(\boldsymbol{\theta})$) corresponding to the initial guess can be regarded as the output. Assuming a polynomial model to fit the I/O data,

$$\mathbf{F}(t_f, \boldsymbol{\theta}) \approx K\mathbf{p}(t_0, \boldsymbol{\theta}) \tag{40}$$

where $p(\boldsymbol{\theta})$ is a vector of assumed polynomial basis functions evaluated at the input points, and K is a matrix of unknown coefficients to be solved for. A least-squares problem can now be set up as

$$\min_K J = \frac{1}{2} \left\langle \left[\mathbf{F}(t_f, \boldsymbol{\theta}) - K\mathbf{p}(t_0, \boldsymbol{\theta})\right], \left[\mathbf{F}(t_f, \boldsymbol{\theta}) - K\mathbf{p}(t_0, \boldsymbol{\theta})\right] \right\rangle_\rho \tag{41}$$

where $\langle ., . \rangle$ represents an inner product with respect to ρ, which is the probability distribution function (pdf) of the input in the defined domain of interest. The minimization procedure leads to the expression

$$K = LM^{-1} \tag{42}$$

$$\text{where,} \quad M_{ij} = \langle p_i(t_0, \boldsymbol{\theta}), p_j(t_0, \boldsymbol{\theta}) \rangle_\rho \tag{43}$$

$$L_{ij} = \langle F_i(t_f, \boldsymbol{\theta}), p_j(t_0, \boldsymbol{\theta}) \rangle_\rho \tag{44}$$

Using the coefficients obtained in the K matrix, a polynomial model as shown in Eq. (40) can now be defined for $\mathbf{F}(t_f, \boldsymbol{\theta})$. The coefficients contained in the K matrix are essentially the sensitivities of the output with respect to the input. Therefore, if certain sensitivities of states or parameters at the final time with respect to those at the initial time are required for the DC process, all one has to do is define the input and output data appropriately, in order to obtain the required sensitivities in the form of the K matrix. The order of the obtained sensitivities depends on the order of the assumed polynomials making up the model.

Note that in Eq. (43), inner products of the polynomial basis functions are taken to construct the M matrix. As the order of the approximation increases, so does the number of polynomial basis functions. This M matrix has to be inverted to obtain K, as shown in Eq. (42). Computing the inverse of a large matrix contributes significantly to the complexity of the operation. To that end, if one were to choose a set of orthogonal polynomial basis functions, then $M_{ij} = 0 \ \forall \ i \neq j$. Therefore, M becomes a diagonal matrix, the inverse of which is trivial to compute. Thus, the computation of higher-order sensitivities can be performed with minimal computational load. For all examples in this paper, the Legendre class of orthogonal polynomials is employed. The choice of orthogonal polynomials varies based on the assumed pdf of $\boldsymbol{\theta}$ in a defined domain. The Legendre polynomials are generally used when a uniform pdf of the input is assumed.[41]

In order to sample data in a defined domain, and evaluate the multidimensional integrals given by Eqs. (43) and (44), the CUT quadrature points[21] are used. CUT provides an optimal sampling of points in a domain, while still capturing the pertinent characteristics of the domain to a good degree of accuracy. The output $\mathbf{F}(\boldsymbol{\theta})$ is evaluated at all these CUT points, and this forms the data set necessary to initiate the LS process to determine the sensitivities. CUT has been used in several applications[21–24] with great success, and its comparison with other quadrature schemes has been well documented.[22,25,42,43] Due to the usage of CUT quadrature points, and for ease of reference, the sensitivities contained in the K matrix of Eq. (42) are termed the CUT-STT. However, the information contained in K could be any sensitivity, based on the I/O data of the LS process, and not necessarily the state sensitivities which are generally referred to as the STT.

The LS-based stochastic approach to evaluate sensitivities eliminates the need to know the numerous partial derivatives, and perform tensor operations, thereby alleviating the computational burden of traditional methods. Recall that the said challenges of partial derivatives and tensor operations arise due to the nature of description of STTs given in Eqs. (37) - (39). Such a description in turn finds its origin in Taylor series expansions about a reference.[38] This means that the STTs obtained using Eqs. (37) - (39) are valid in a small neighborhood around the reference, similar to the one-dimensional example illustrated in Fig. 1. While the size of the neighborhood increases with increase in the order of the STT, there are no definite bounds that one can define, within which it remains valid.

On the other hand, with the CUT-STT, the description first starts with the definition of a domain for the input ($\boldsymbol{\theta}$), within which the CUT points are populated to acquire data for the LS procedure. Therefore, the CUT-STT is valid within a well-defined domain. For this reason, they are not exactly equivalent to traditional sensitivities, yet, they serve the same purpose when it comes to a DC process. Having explored the FORS on simple analytical systems, the following sections examine its application to DC schemes on dynamical systems, with the help of the CUT-STT.

ZERMELO PROBLEM

The Zermelo problem is chosen as the first example to study the combined efficacy of the CUT-STT and the FORS algorithm on dynamical systems. The problem deals with the design of a control policy for the steering angle of a ship, in order to reach a desired target in minimum time, in the presence of ocean currents.[2,44] The dynamical equations are given by[2]

$$\dot{x} = V \cos\gamma + u(x, y) \tag{45}$$

$$\dot{y} = V \sin\gamma + v(x, y) \tag{46}$$

where u and v are the ocean currents as functions of the position of the ship, γ is the steering angle, and V is the constant speed at which the ship is moving. The Hamiltonian of the system for the minimum time problem is given by

$$H = \lambda_x(V\cos\gamma + u) + \lambda_y(V\sin\gamma + v) + 1 \tag{47}$$

where $\boldsymbol{\lambda}$ is the costate vector. The necessary conditions for optimality, in addition to Eqs. (45) and (46) are

$$\dot{\lambda}_x = -\frac{\partial H}{\partial x} \tag{48}$$

$$\dot{\lambda}_y = -\frac{\partial H}{\partial y} \tag{49}$$

The stationarity condition yields the steering angle as a function of the costates.

$$\frac{\partial H}{\partial \gamma} = V(-\lambda_x\sin\gamma + \lambda_y\cos\gamma) = 0 \tag{50}$$

$$\implies \gamma = \tan^{-1}\left(\frac{\lambda_y}{\lambda_x}\right) \tag{51}$$

Eqs. (45), (46), (48), and (49) form a TPBVP, given the initial position of the ship and its desired final destination. An analytical solution to a simplified Zermelo problem is provided by Bryson and Ho.[2]

In the simplified version of the problem, v is considered to be zero. A scaling factor h is introduced for the length units. In this special case, h and V are each considered to be unity. The current u is given by

$$u = -V\left(\frac{y}{h}\right) \tag{52}$$

The initial position of the ship is

$$\frac{x_0}{h} = 3.66\,, \qquad \frac{y_0}{h} = -1.86$$

The target destination is the origin. We shall address the same simplified problem in order to verify the results obtained using FORS. First, we implement a conventional numerical DC scheme employing up to the second-order STT, in order to highlight the differences between first and second-order DC methods. For this problem, the unkown parameters, also referred to as the design variables for the DC scheme are $\boldsymbol{\theta} = [\lambda_{x_0}, \lambda_{y_0}, t_f]$. The objective function at the final time, whose root is desired, is given by $\mathbf{F} = [x_f, y_f, H_{t_f}]$. Note that $H_{t_f} = \frac{\partial H}{\partial t_f} = 0$ is an additional necessary condition, since t_f is a free parameter[2] in the setup of this TPBVP. Note that since t_f is an unknown parameter, the integration is carried out with respect to a non-dimensionalized time variable $\tau = \frac{t}{t_f}$. Therefore, the dynamical equations have to be changed appropriately to reflect τ as the independent variable.

Starting from an initial guess of the design variables, an iterative correction process is carried out by employing Newton's method, which is referred to as first-order DC. Rewriting Eq. (9) for the

parameters of this problem, the necessary update equation at each iteration of the DC process can be written as

$$\begin{bmatrix} \lambda_{x_0} \\ \lambda_{y_0} \\ t_f \end{bmatrix}_{new} = \begin{bmatrix} \lambda_{x_0} \\ \lambda_{y_0} \\ t_f \end{bmatrix}_{old} - \begin{bmatrix} \frac{\partial x_f}{\partial \lambda_{x_0}} & \frac{\partial x_f}{\partial \lambda_{y_0}} & \frac{\partial x_f}{\partial t_f} \\ \frac{\partial y_f}{\partial \lambda_{x_0}} & \frac{\partial y_f}{\partial \lambda_{y_0}} & \frac{\partial y_f}{\partial t_f} \\ \frac{\partial H_{t_f}}{\partial \lambda_{x_0}} & \frac{\partial H_{t_f}}{\partial \lambda_{y_0}} & \frac{\partial H_{t_f}}{\partial t_f} \end{bmatrix}^{-1} \begin{bmatrix} x_f \\ y_f \\ H_{t_f} \end{bmatrix} \tag{53}$$

In Eq. (53), note that columns one and two of the sensitivity matrix consist of elements of the STM, which are given by Eq. (36). Column three consists of sensitivities with respect to the free parameter, which is not given by the STM. In order to obtain these sensitivities, a procedure similar to the derivation of Eq. (36) can be adopted, but with respect to the free parameter (t_f), instead of the states. On doing so, one obtains

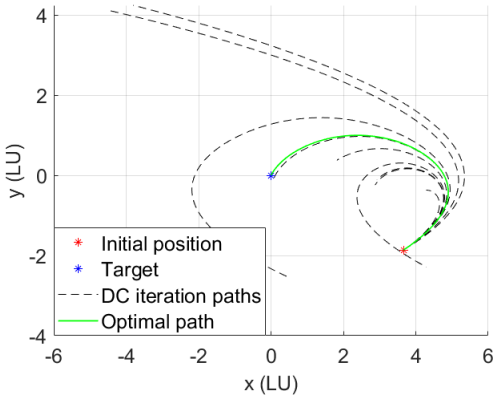
$$\dot{\boldsymbol{\psi}} = \frac{\partial \mathbf{f}}{\partial t_f} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \boldsymbol{\psi} \tag{54}$$

where $\boldsymbol{\psi}$ is a vector of sensitivity of each state to the free parameter, and f is simply the system dynamics. The initial condition $\boldsymbol{\psi}_0$ for Eq. (54) is simply a zero vector. A second-order DC would require a second-order STT and parameter sensitivity, in addition to the STM, in order to implement Halley's algorithm given in Eq. (18).

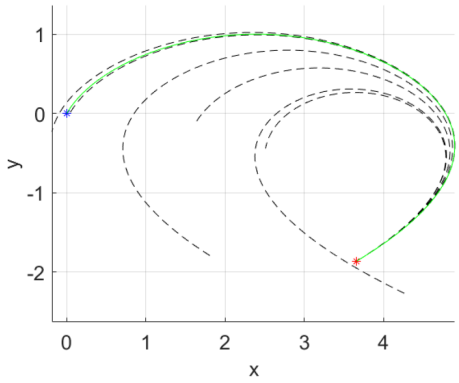
An initial guess of $[\lambda_{x_0}, \lambda_{y_0}, t_f] = [0.59, -1.77, 6.46]$ is assumed, and the TPBVP is solved using both the first-order and second-order DC schemes. The paths corresponding to each iterate of the first-order and the second-order DC schemes are illustrated in Figs. 4a and 4b, respectively. Recall that first-order DC employs Newton's iterative root-finding scheme, while second-order DC employs Halley's iterative scheme. These paths are also visualized in terms of λ_x and λ_y on the surface of the objective function in Fig. 4c. Note that the second-order update is of higher quality, compared to the first-order. This is evident in the progression of the iteration paths from the initial guess to the converged solution.

Note that for this problem, a first-order DC scheme requires the integration of 35 equations per iteration, while a second-order DC scheme requires the integration of 160 equations per iteration. The issue lies in the fact that first-order DC converged to the optimal solution in eleven iterations, while second-order DC converged in eight iterations. The computational expense of three more iterations of first-order DC is far lesser than the computational expense of implementing a second-order DC. The prime reason for the high cost of the latter lies in the evaluation of the second-order STT. It is for this reason that first-order DC is widely preferred, and higher-order methods are very rarely used. However, the CUT-STT helps in tackling this issue, so that higher-order methods become feasible to employ in DC. Henceforth in the paper, all DC schemes of second-order and higher are implemented using the CUT-STTs.

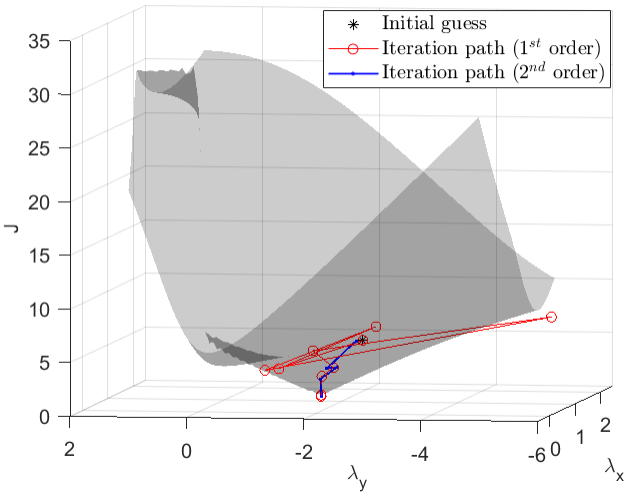
The example illustrated in Fig. 4 showcased the advantage of a second-order DC scheme (using Halley's algorithm) in tackling the TPBVP. In order to highlight the effectiveness of the proposed higher-order schemes, a stochastic study is undertaken. The goal of this study is to examine the robustness of the higher-order schemes to the initial guess, in comparison to the Newton's method. For the TPBVP illustrated in Fig. 4, 500 random initial condition samples are chosen from within predefined domains for $[\lambda_{x_0}, \lambda_{y_0}, t_f]$. Three such cases are considered, where the domains have



(a) First-order differential correction



(b) Second-order differential correction



(c) Iteration paths viewed on the objective function's surface, with respect to λ_x and λ_y

Figure 4: Comparison of first-order and second-order differential correction schemes

increasing sizes from case-1 to case-3, as illustrated in Table 2. Note that these domains are built around the actual solution to the TPBVP. Therefore, from case-1 to case-3, the quality of the randomly chosen initial guesses in each case gets progressively worse.

In each case, all 500 initial guess samples are put through five different solvers. The first would be the regular DC utilizing Newton's algorithm, and generates the required sensitivities using Eqs. (36) and (54). The second solver also utilizes Newton's algorithm, but generates the required sensitivities from a polynomial model corresponding to the domain under consideration. This model can be constructed by implementing Eqs. (40) - (44). A fourth-order polynomial model is adopted for all the test cases. The necessary sensitivities can simply be analytically derived from the polynomial model to effect the corrections process. This is referred to as the non-intrusive approach to DC, and is the main focus of this work, as it enables a computationally efficient way to perform higher-order DC. It is termed non-intrusive, as one does not require explicit knowledge of the system dynamics in order to construct the polynomial model. Model construction is simply driven by data

generated at the CUT points. Therefore, for all intents and purposes, the system dynamics could essentially be a black box that yields output data for any input passed into it, and the non-intrusive approach can still perform a successful DC process. The higher-order sensitivities supplied by the polynomial model enable the implementation of Halley's algorithm, the TORS and the FORS, without computational burden. Out of the 500 initial guess samples, the number of those that lead to a converged result when put through the different solvers is recorded. Using this observation, the corresponding convergence probability for each method is tabulated in Table 2. This convergence probability is used to quantify the robustness of the methods to the initial guess.

Table 2: Convergence probability of various methodologies in finding a solution to a given TPBVP, tabulated for a study of 500 initial guess samples

Case	**Variable bounds**			**Convergence Probability**				
	$\boldsymbol{\lambda_x}$	$\boldsymbol{\lambda_y}$	$\boldsymbol{t_f}$	**Regular DC (Newton)**	**Non-intrusive approach**			
					Newton	**Halley**	**TORS**	**FORS**
1	$[0.2, 0.8]$	$[-2.2, -1.5]$	$[4.5, 6.5]$	71%	72%	72.6%	84.2%	91.6%
2	$[0, 1]$	$[-2.4, -1.3]$	$[3.5, 7.5]$	38.4%	44.4%	44%	52.2%	72.6%
3	$[-0.2, 1.2]$	$[-2.6, -1.1]$	$[2.5, 8.5]$	24%	19.6%	32.2%	33.2%	49%

It is known that a regular DC method using Newton's algorithm requires an initial guess close enough to the actual solution of the TPBVP, in order to achieve successful convergence. Getting started is an infamous difficulty one often faces while dealing with gradient methods. Higher-order methods alleviate that difficulty, as observed in the results shown in Table 2. In case-1, a small domain around the nominal solution is chosen to populate the initial random guesses. Note that 71% of those samples are good enough for the Newton's scheme to start with and reach the solution iteratively. Essentially, this means that the quality of the remaining samples in the domain chosen in case-1 are insufficient for Newton's method to reach the solution. However, note the increasing convergence probability as one increases the order of the DC scheme employed. Halley's scheme does not yield a significant advantage over Newton's for this problem, however, the TORS and FORS provide considerable improvement with convergence probabilities of 84.2% and 91.6%, respectively.

One has to keep in mind that the necessary higher-order sensitivities for the Halley, TORS and FORS algorithms are supplied by the polynomial model, which is an approximation of the dynamics in the first place. Nevertheless, the derivatives supplied by the model provide good, useful information in generating the higher-order updates to achieve convergence to the solution.

Case-2 considers a bigger domain than case-1 as indicated in Table 2. This time, only 38.4% of the samples are feasible for the Newton's scheme to work with and converge to the solution. On the other hand, the FORS yields a success probability of 72.6%, which is nearly double that of a conventional first-order DC scheme. It must be noted that as the size of the domain increases, the quality of the approximation of dynamics captured by the polynomial model decreases. Regardless, the FORS is still considerably more robust to the quality of the initial guess than conventional DC.

The domain chosen in case-3 is considerably large, consequently leading to a significant reduction in convergence probability of the conventional DC scheme. The large size of the domain affects the

quality of the polynomial model, thereby affecting the convergence probabilities of the higher-order DC schemes as well. From Table 2, it must be noted that even at this stage, the FORS is twice as likely as the conventional Newton's method to converge to the solution. This study bears significant evidence of the superior robustness of higher-order schemes, especially the FORS, to the quality of the initial guess.

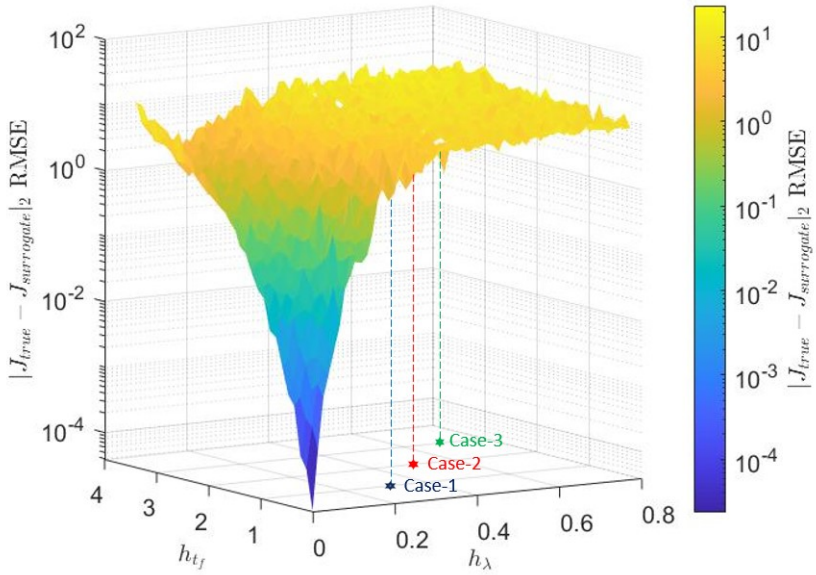


Figure 5: Surface plot of the RMSE between the true Jacobian and the approximated Jacobian generated by the polynomial model, illustrating the sensitivity of the approximation quality to the domain sizes of $\boldsymbol{\lambda}$ and t_f

The domains discussed in Table 2 were simply referred to as being smaller or larger compared to one another. However, one can obtain a better idea of the sensitivity of the polynomial approximation to the domain size of each unknown parameter by performing a parameter sensitivity analysis. The bounds for the domains discussed in Table 2 were defined using $[\lambda_x^* + h_\lambda, \lambda_y^* + h_\lambda, t_f^* + h_{t_f}]$, where λ_x^*, λ_y^* and t_f^* represent the actual solution to the TPBVP, and h_λ and h_{t_f} control the size of the domain around that solution, from which the initial guess samples are chosen at random. One can comprehend the quality of approximation produced by the polynomial model, by comparing the two norm error between the Jacobian calculated by the conventional method (using Eq. (36)), and that calculated using the polynomial model, as a function of the domain size of $\boldsymbol{\lambda}$ and t_f.

Fig.5 illustrates the aforementioned error, and provides grounds for judiciously choosing the domain sizes. Each point on the h_λ-h_{t_f} plane in Fig. 5 represents a domain $[\lambda_x^* + h_\lambda, \lambda_y^* + h_\lambda, t_f^* + h_{t_f}]$. The Jacobian RMSE corresponding to each domain is calculated by considering a pool of 50 random samples, to obtain insight into the maximum values of h_λ and h_{t_f} one could choose without compromising on the robustness of the higher-order methods. The cases summarized in Table 2 are also indicated in Fig. 5 for reference. For case-3, with $h_\lambda \approx 0.7$ and $h_{t_f} \approx 3$, it is observed that the Jacobian RMSE is close to an order of 10^1. Recall that the convergence probability of the FORS was approximately 50% for this case. Therefore, this point could be considered as a cut-off point beyond which it would be worthwhile to construct a new polynomial model for different values of λ_x^*, λ_y^* and t_f^*.

Note that λ_x^*, λ_y^* and t_f^* simply refer to arbitrary reference values around which a domain is considered, and they do not need to be the actual solution to the problem at hand. They were only chosen to be the true solution in this work for the purpose of the case study in Table 2, which was to demonstrate robustness of the FORS to the initial guess.

The values of h_λ and h_{t_f} are primarily problem dependent, as well as on one's knowledge of whereabouts of the solution, if a physical intuition exists. Often times, such intuitions are hard to develop when tackling TPBVPs relevant to optimal control problems, as the co-states generally do not carry physical meaning. This is one of the many reasons why one can benefit from a DC scheme that is more robust to initial guess, compared to commonly used schemes in the present day. The FORS is presented as one such scheme in this paper. Having successfully applied higher-order schemes using a non-intrusive approach to the DC process on the Zermelo problem, let us move on to study the performance of the proposed methodology on problems in the CR3BP, given the highly sensitive and chaotic nature of the CR3BP system. For this reason, the non-intrusive DC approach is put to the task of solving TPBVPs to find periodic orbits in the CR3BP in the following section.

CIRCULAR RESTRICTED THREE BODY PROBLEM

The CR3BP is formulated in a frame that rotates along with the primaries, called the synodic frame. The $\hat{x}$ basis vector points from the origin, which is at the barycenter of the Earth-Moon system, toward the Moon. The $\hat{y}$ basis vector is perpendicular to it and lies in the plane of motion of the primaries. The $\hat{z}$ vector is given by the cross product of $\hat{x}$ and $\hat{y}$. A canonical system of units is employed where one length unit (LU) is equal to the distance between the two primaries and one time unit (TU) is chosen such that the mean motion of the primaries is unity. For the Earth-Moon system, $1\,\mathrm{LU} = 384\,400\,\mathrm{km}$ and $1\,\mathrm{TU} = 4.3424$ days.

The Cartesian CR3BP equations of motion are

$$\dot{v}_x = \Omega_x + 2v_y \tag{55a}$$

$$\dot{v}_y = \Omega_y - 2v_x \tag{55b}$$

$$\dot{v}_z = \Omega_z \tag{55c}$$

where $v_x = \dot{x}$, $v_y = \dot{y}$, $v_z = \dot{z}$, and Ω is the pseudo-potential in the synodic frame. Ω_x, Ω_y and Ω_z are the partial derivatives of Ω with respect to x, y and z, respectively.

$$\Omega = \frac{1}{2}(x^2 + y^2) + \frac{(1-\mu)}{r_1} + \frac{\mu}{r_2} \tag{56}$$

where $r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}$, $r_2 = \sqrt{(x+\mu-1)^2 + y^2 + z^2}$, and $\mu = \frac{m_2}{m_1+m_2}$. r_1 and r_2 are the magnitudes of the position vectors of the spacecraft relative to the Earth and the Moon, respectively. m_1 and m_2 are the masses of the Earth and the Moon, respectively. μ is the characteristic parameter in the synodic frame. For the Earth-Moon system, $\mu = 0.012151$. There exists an integral of motion in the CR3BP known as the Jacobi integral or the Jacobi constant (C).

$$C = 2\Omega - (v_x^2 + v_y^2 + v_z^2) \tag{57}$$

The CR3BP admits several periodic solutions. One such solution is the Lyapunov orbit. Determining the initial conditions that would yield a Lyapunov orbit in the chaotic, highly nonlinear environment of the CR3BP, requires the use of numerical techniques and a good initial guess.

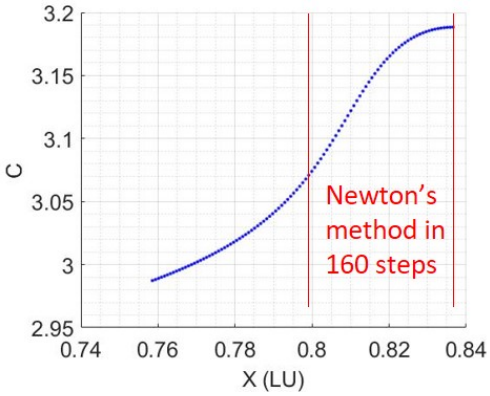
The process starts with a linear approximation of the dynamics in the vicinity of the L_1 Lagrange point. A periodic solution to the linear dynamics is found, and put through a DC process in order to obtain the solution to the true nonlinear system. The DC process to find a Lyapunov orbit is traditionally set up using Newton's method as

$$\begin{bmatrix} \dot{y}_0 \\ \tau \end{bmatrix}_{new} = \begin{bmatrix} \dot{y}_0 \\ \tau \end{bmatrix}_{old} - \begin{bmatrix} \Phi_{y,\dot{y}} & \dot{y}_f \\ \Phi_{\dot{x},\dot{y}} & \ddot{x}_f \end{bmatrix}^{-1} \begin{bmatrix} y_f \\ \dot{x}_f \end{bmatrix} \tag{58}$$

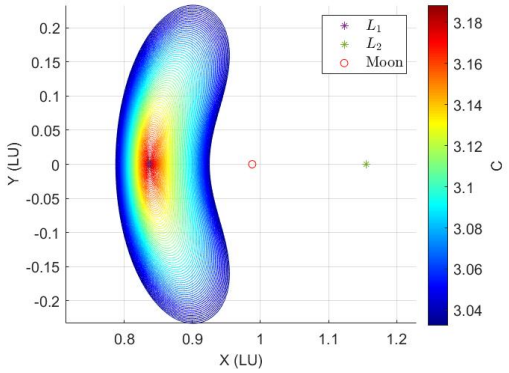
where τ here is the half period of the orbit. Once a solution is obtained, a continuation process is adopted by stepping along x_0 to create a new boundary condition. Subsequently, the new TPBVP is solved to obtain the second periodic solution of the family, and the process is repeated until termination of the family. The characteristics of a family vary from system to system. Each CR3BP system is uniquely defined by the mass parameter μ. Therefore, in order to study the periodic families in different systems, a DC process combined with continuation is necessary. For analytic purposes, one might be interested in the size and characteristics of these periodic orbit families as a whole. In order to obtain insight about the size of a family, one has to go through the continuation process sequentially, and cannot afford to skip ahead to the end of the family, as there is no means to do that if a particular CR3BP system hasn't already been catalogued.

The drawback here lies in the fact that the stepping distance in the continuation process is an important parameter, as too big a step can sometimes result in the inability of the differential corrector to find a solution. Therefore, small steps are taken cautiously, with provision to reduce the step size should the corrector experience trouble along the way. For the purpose of demonstration, a traditional DC scheme is first employed to find a series of orbits belonging to the L_1 Lyapunov family by performing the continuation process with a fixed step size of $120\,\mathrm{km}$. It was observed that 160 steps were able to be completed before the step size proved too big a difference in the initial guess for the Newton's scheme to converge to a solution. These set of orbits are illustrated in Fig. 6b.

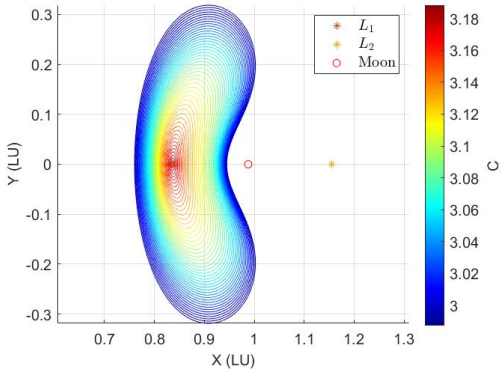
Now, let us employ a similar continuation process starting from the same initial orbit as the first-order DC scheme, but using a second-order DC scheme this time with the non-intrusive approach. Using the CUT-STT, a polynomial model of second order is constructed. For the second-order DC process, a step size of $300\,\mathrm{km}$ is adopted. Recall that one of the significant results obtained in the previous case study of the Zermelo problem, was that the necessity for a good initial guess is relaxed when using a higher-order DC scheme, as opposed to a first-order one. This feature is tested in the CR3BP by adopting a higher step size in the continuation process. The goal is to go from family initiation to a certain bound on the family with a reduced number of steps, in comparison to the traditional first-order scheme. This can be observed in Fig. 6a and Fig. 6c. For demonstration purposes in this paper, the continuation process is only carried out for 100 steps using second-order DC. In comparison with Fig. 6b, note the sparse collection of orbits in Fig. 6c, which is a visual indication of the contrast in step size difference. From Fig. 6a, by examining the range of C, it is understood that the second-order DC generates about twice the range of orbits yielded by the first-order method. This difference is a major advantage provided by the second-order DC scheme over the traditional first-order method, and is a testament to the superior robustness of the second-order scheme to initial guess in this particular problem. Studying the bounds of families of periodic orbits in a CR3BP system becomes much easier when the sequential process of continuation takes larger steps. Larger steps tend to reduce the quality of the initial guess for the subsequent TPBVP



(a) Range of orbits yielded by 2^{nd} order DC indicated by their C values and initial XZ plane crossing point



(b) 1^{st} order DC with small continuation steps



(c) 2^{nd} order DC with larger continuation steps

Figure 6: Comparison of continuation steps necessary between 1^{st} and 2^{nd} order DC for generation of Lyapunov orbit family

in the purview of the Newton's scheme, hence forcing a smaller step size. However, higher-order schemes, now made computationally feasible by the non-intrusive DC approach, can work well with an average quality initial guess, in order to obtain a successful solution.

It is to be noted that in the second-order DC as implemented here, at each step of the continuation process, a new polynomial model is generated to perform the iterations. This is because a new continuation step creates a new prescribed boundary condition. Therefore, a polynomial model corresponding to the new condition is generated in an arbitrarily chosen domain of the unknowns $\dot{y}_0$ and τ. However, in performing a second-order non-intrusive DC, only 8 CUT points were utilized. Therefore, generating a new polynomial model at every continuation step required 48 equations to be integrated. In other words, it takes 48 integrations for each orbit in the family. On the other hand, when using a conventional DC, one is required to integrate 42 equations each iteration, in order to find one orbit. On average, a conventional Newton's scheme yields a converged solution in 5 iterations. This means that one has to integrate 210 equations on average to find one orbit using conventional DC, while on the other hand the same can be achieved using 48 integrations using the non-intrusive DC approach. Moreover, the FLOPs necessary for the second-order DC involving Halley's scheme, is not significantly more than that required by the Newton's algorithm,

as illustrated in Fig. 2. Therefore, the second-order non-intrusive DC proves to be significantly more efficient than the traditional DC scheme of first order.

In comparison, if one wishes to employ second-order DC traditionally, without the polynomial model for this problem, one would be required to integrate 258 equations per iteration in order to determine the second-order STT necessary for the DC process. This was a huge factor that prevented the usage of such second-order schemes in the CR3BP, until now. With the advent of non-intrusive DC, a significant computational burden is alleviated, thus opening the door for the fruits of higher-order DC schemes to be enjoyed.

CONCLUSION

A novel methodology involving a higher-order iteration scheme termed FORS, with the necessary higher-order derivatives supplied by a polynomial model, was developed to aid in solving TPBVPs. The FORS was observed to outperform the traditional DC scheme in terms of its robustness to the quality of the initial guess. Moreover, a derivative-free approach was adopted to construct the polynomial model in a given domain, that made the implementation of the higher-order scheme much more easy to handle, as the computation of higher-order sensitivities was one of the major factors prohibiting the widespread use of higher-order iterative schemes so far.

The proposed methodology was successfully applied to examples to demonstrate the aforementioned claim of robustness to initial guess, in tandem with the polynomial model, which tackled the difficulty of obtaining higher-order sensitivities. A parameter sensitivity study was carried out in the Zermelo problem to determine the feasible size of the domain in which one can expect good results from the proposed methodology. Future work could take into account the observations made from this study, in order to dynamically update the domain and learn a new polynomial model during the correction process. In such a case, the algorithm would have no problems arriving at the solution by switching between models depending on the domain in which each iterate obtained during the correction belongs to. Such a scheme would be beneficial to spacecraft guidance problems, which will be explored in the future.

REFERENCES

[1] D. Izzo, "Revisiting Lambert's problem," *Celestial Mechanics and Dynamical Astronomy*, Vol. 121, 2015, pp. 1–15.

[2] A. Bryson and Y.-C. Ho, "Applied optimal control, Hemisphere," *New York*, 1975.

[3] S. R. Vadali, *Solution of the two-point boundary value problems of optimal spacecraft rotational maneuvers*. PhD thesis, Virginia Polytechnic Institute and State University, 1982.

[4] D. Griffith, J. Turner, S. Vadali, and J. Junkins, "Higher order sensitivities for solving nonlinear two-point boundary-value problems," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2004, p. 5404.

[5] J. L. Junkins and J. D. Turner, *Optimal spacecraft rotational maneuvers*. Elsevier, 2012.

[6] J. V. Breakwell and J. V. Brown, "The 'halo'family of 3-dimensional periodic orbits in the Earth-Moon restricted 3-body problem," *Celestial mechanics*, Vol. 20, No. 4, 1979, pp. 389–404.

[7] K. Howell, "Three-dimensional, periodic,'halo'orbits," *Celestial mechanics*, Vol. 32, No. 1, 1984, pp. 53–71.

[8] M. T. Ozimek, *A low-thrust transfer strategy to earth-moon collinear libration point orbits*. PhD thesis, School of Aeronautics and Astronautics, Purdue University, 2006.

[9] C. E. York and K. C. Howell, "A two-level differential corrections algorithm for low-thrust spacecraft trajectory targeting," *29th AAS/AIAA Space Flight Mechanics Meeting*, 2019, pp. 1–20.

[10] B. P. Emma and H. J. Pernicka, "Algorithm for autonomous longitude and eccentricity control for geostationary spacecraft," *Journal of guidance, control, and dynamics*, Vol. 26, No. 3, 2003, pp. 483–490.

[11] H. J. Pernicka, B. A. Carlson, and S. Balakrishnan, "Spacecraft formation flight about libration points using impulsive maneuvering," *Journal of guidance, control, and dynamics*, Vol. 29, No. 5, 2006, pp. 1122–1130.

[12] A. Galántai, "The theory of Newton's method," *Journal of Computational and Applied Mathematics*, Vol. 124, No. 1-2, 2000, pp. 25–44.

[13] G. R. Hecht and E. M. Botta, "Heuristic optimization algorithms for initializing indirect minimum-fuel trajectory optimization," *AIAA SCITECH 2022 Forum*, 2022, p. 1628.

[14] M. Pontani and B. Conway, "Particle swarm optimization applied to space trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1429–1441.

[15] S. Sharan and R. G. Melton, "Design of low thrust trajectories from low earth orbit to distant retrograde orbits by particle swarm optimization," *AAS/AIAA Astrodynamics Specialist Conference, 2020*, Univelt Inc., 2021, pp. 4561–4579.

[16] E. Halley, "Methodus nova accurata & facilis inveniendi radices æqnationum quarumcumque generaliter, sine praviæ reductione," *Philosophical Transactions of the Royal Society of London*, Vol. 18, No. 210, 1707, pp. 136–148.

[17] T. Yamamoto, "On the method of tangent hyperbolas in Banach spaces," *Journal of computational and applied mathematics*, Vol. 21, No. 1, 1988, pp. 75–86.

[18] J. M. Gutierrez and M. A. Hernández, "An acceleration of Newton's method: super-Halley method," *Applied Mathematics and Computation*, Vol. 117, No. 2-3, 2001, pp. 223–239.

[19] J. Kou, Y. Li, and X. Wang, "A variant of super-Halley method with accelerated fourth-order convergence," *Applied mathematics and computation*, Vol. 186, No. 1, 2007, pp. 535–539.

[20] N. Adurthi, P. Singla, and T. Singh, "The conjugate unscented transform—an approach to evaluate multi-dimensional expectation integrals," *2012 American Control Conference (ACC)*, IEEE, 2012, pp. 5556–5561.

[21] N. Adurthi, P. Singla, and T. Singh, "Conjugate unscented transform and its application to filtering and stochastic integral calculation," *AIAA Guidance, Navigation, and Control Conference*, 2012, p. 4934.

[22] N. Adurthi, P. Singla, and M. Majji, "Conjugate Unscented Transform based approach for dynamic sensor tasking and Space Situational Awareness," *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 5218–5223.

[23] N. Adurthi and P. Singla, "Conjugate unscented transformation-based approach for accurate conjunction analysis," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 9, 2015, pp. 1642–1658.

[24] N. Adurthi, P. Singla, and T. Singh, "Conjugate unscented transformation: Applications to estimation and control," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 140, No. 3, 2018.

[25] S. Sharan, R. T. Eapen, P. Singla, and R. G. Melton, "Coordinate choice implications for uncertainty propagation in the CR3BP framework," *2022 AAS/AIAA Astrodynamics Specialist Conference, Charlotte, North Carolina, August 7-11*, 2022.

[26] B. Conway, *Spacecraft trajectory optimization*, Vol. 29. Cambridge University Press, 2010.

[27] D. B. Meade, B. S. Haran, and R. E. White, "The shooting technique for the solution of two-point boundary value problems," *Maple Technical Newsletter*, Vol. 3, No. 1, 1996, pp. 1–8.

[28] M. T. Ozimek, *Low-thrust trajectory design and optimization of lunar south pole coverage missions*. PhD thesis, Purdue University, 2010.

[29] W. Gander, "On Halley's iteration method," *The American Mathematical Monthly*, Vol. 92, No. 2, 1985, pp. 131–134.

[30] X. Wang, C. Gu, and J. Kou, "Semilocal convergence of a multipoint fourth-order super-Halley method in Banach spaces," *Numerical Algorithms*, Vol. 56, 2011, pp. 497–516.

[31] A. A. Cuyt and L. B. Rall, "Computational implementation of the multivariate Halley method for solving nonlinear systems of equations," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 11, No. 1, 1985, pp. 20–36.

[32] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Mathematics of computation*, Vol. 19, No. 92, 1965, pp. 577–593.

[33] C. G. Broyden, "Quasi-Newton methods and their application to function minimisation," *Mathematics of Computation*, Vol. 21, No. 99, 1967, pp. 368–381.

[34] R. Fletcher, "A new approach to variable metric algorithms," *The computer journal*, Vol. 13, No. 3, 1970, pp. 317–322.

[35] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Mathematics of computation*, Vol. 24, No. 109, 1970, pp. 23–26.

[36] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Mathematics of computation*, Vol. 24, No. 111, 1970, pp. 647–656.

[37] R. S. Park and D. J. Scheeres, "Nonlinear mapping of Gaussian statistics: theory and applications to spacecraft trajectory design," *Journal of guidance, Control, and Dynamics*, Vol. 29, No. 6, 2006, pp. 1367–1375.

[38] M. Majji, J. L. Junkins, and J. D. Turner, "A high order method for estimation of dynamic systems," *The Journal of the Astronautical Sciences*, Vol. 56, No. 3, 2008, pp. 401–440.

[39] J. D. Turner, "Automated generation of high-order partial derivative models," *AIAA journal*, Vol. 41, No. 8, 2003, pp. 1590–1598.

[40] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, Vol. 2, No. 2, 1944, pp. 164–168.

[41] F. W. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, *NIST handbook of mathematical functions hardback and CD-ROM*. Cambridge university press, 2010.

[42] N. Adurthi, P. Singla, and T. Singh, "Conjugate unscented transform rules for uniform probability density functions," *2013 American Control Conference*, IEEE, 2013, pp. 2454–2459.

[43] Z. Hall, *A Probabilistic Framework to Locate and Track Maneuvering Satellites*. PhD thesis, The Pennsylvania State University, 2021.

[44] E. Zermelo, "Über das Navigationsproblem bei ruhender oder veränderlicher Windverteilung," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, Vol. 11, No. 2, 1931, pp. 114–124.