Analytical first derivatives of the direct Hermite-Simpson collocation formulation of optimal control problems

Aravind Arunakirinathar^{1,*}, Jean-Francois de la Beaujardiere², and Michael Brooks²

¹Graduate Student, Mechanical Engineering, University of KwaZulu-Natal, 4041,

²Mechanical Engineering, University of KwaZulu-Natal, 4041

Abstract. In order to assess the capabilities of South Africa as a launch site for commercial satellites, an optimal control solver was developed. The developed solver makes use of direct Hermite-Simpson collocation methods, and can be applied to a general optimal control problem. Analytical first derivative information was obtained for direct Hermite-Simpson collocation methods. Typically, a numerical estimate of the derivative information is used. This paper will present the solver algorithm, and the formulation and derivation of the analytical first derivative information for this approach. A linear and non-linear sample problem are provided as validation of the solver.

1 Introduction

An optimal control solver making use of the direct Hermite-Simpson collocation formulation of optimal control problems was developed. Formulae for determining the analytical first derivative information of the defect constraints of the direct Hermite-Simpson collocation formulation of optimal control problems have been formulated. The control variables were represented by linear mean splines. Irrespective of this, the first derivative formulae obtained can be extended to other representations of the control variables. The formulae obtained can also be extended to solving ordinary two-point boundary value problems with Hermite-Simpson collocation.

Typically, these first derivatives are estimated using sparse finite-differencing, alternatively they can be obtained with automatic differentiation or symbolic differentiation. The formula provided allows for an easy and compact representation and calculation of analytical first derivatives. Provision of analytical derivative information, versus estimating this information, typically increases the robustness and accuracy of solvers using Newton-iterative techniques [1]. Symbolic differentiation algorithms tend to not present derivatives in compact forms, which has the potential to significantly increase computation time. Forward automatic differentiation is easy to use, but inefficient for large problems, whereas reverse automatic differentiation requires significant effort to implement, but is efficient for large problems [2].

Optimal control is a field of study with various application in fields such as aerospace, robotics and chemical engineering. It is concerned with the determination of trajectories which optimise some cost function of a given dynamic system. The techniques of optimal control have been used, successfully, to optimise a host of dynamic system trajectories. These

*This project was supported by the National Research Foundation (NRF). This is much appreciated.
e-mail: aravind519@yahoo.com,

trajectories include: minimum fuel orbital insertion trajectories, re-entry trajectories and minimum energy robot trajectories.

This paper first outlines the general transcription process of an optimal control problem to an Nonlinear Programming (NLP) problem using Hermite-Simpson collocation with linear mean control splines. The transcription process presented will assume the period of any phase of the state variables trajectory is a free variable. The derivation of the first derivative information of this formulation of the optimal control problem is then presented. Following which the first derivatives will be applied with example problems, using Matlab's Sequential Quadratic Programming (SQP) algorithm.

2 Problem Formulation

2.1 Transcription

Optimal control is the field of study concerned with determining the control history of a dynamic system which optimises a performance index or cost function. The cost can be a function of terminal values of the state variables or an integral function of the system, or both. The following summarises the optimal control problem, with terminal constraints to the state variables, and some of the state variables initial values defined. The systems analysed will be autonomous. It should be noted that other constraints such as path constraints can also be considered.

$$\min_{\mathbf{u}} \phi(\mathbf{x}_f, t_f) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t)) dt \quad (1)$$

subject to

$$\begin{aligned} (\mathbf{x}_f, t_f) &= 0 \quad (\text{end conditions}) \\ \dot{\mathbf{x}} &= g(\mathbf{x}(t), \mathbf{u}(t)) \end{aligned}$$

Where $\mathbf{x}_f$ is the terminal state vector, t_f is the unknown final time of the phase of the trajectory. $\mathbf{x} \in \mathbb{R}^{m \times 1}$ and $\mathbf{u} \in \mathbb{R}^{n \times 1}$ are the state and control vector respectively. The functions $g : \mathbb{R}^{(m+n) \times 1} \rightarrow \mathbb{R}^{m \times 1}$, outputs the first time derivative of the state vector, $\phi : \mathbb{R}^{(m+1) \times 1} \rightarrow \mathbb{R}^{m \times 1}$, outputs the component of the cost function which is dependent on terminal values of the system, and the function $L : \mathbb{R}^{(m+n) \times 1} \rightarrow \mathbb{R}^{m \times 1}$, outputs the integrand of the integral component of the cost function. $\psi : \mathbb{R}^{(m+n) \times 1} \rightarrow \mathbb{R}^{m \times 1}$, outputs the component of terminal boundary constraints, which is dependent on terminal values of the system.

In the event the final time is unknown, the following augmentation to the problem may be performed in order to allow for numerical solutions to be obtained.

Let $t = t_f \tau$ where $\tau \in [0, 1]$ then $\frac{dt}{d\tau} = t_f$.
Let there be a function $f : \mathbb{R}^{(m+n+1) \times 1} \rightarrow \mathbb{R}^{m \times 1}$, which outputs the first derivative of the state vector with respect to the new independent variable τ . That is

$$f(\mathbf{x}(\tau), \mathbf{u}(\tau), t_f) = \frac{d\mathbf{x}}{d\tau} \quad (2)$$

Then by the chain rule $f(\mathbf{x}(\tau), \mathbf{u}(\tau), t_f) = t_f g(\mathbf{x}(\tau), \mathbf{u}(\tau))$

The cost (or objective) function could, for example, be formulated in a manner so as to minimise the fuel usage of a particular rocket mission. The dynamic equations, along with any path constraints, essentially constrain the feasible solutions of the optimal objective function. In order to solve this problem using an NLP, the problem is converted, or transcribed, by Hermite-Simpson collocation into an NLP problem. This is achieved by discretising the independent variable of the system (typically time), in order to create a grid, or mesh. Each grid point of this grid represents discrete values of the dependent variables, in the case of optimal control these variables are a state and control vector. The differential equations which

represent the system are replaced with defect constraints for each independent variable interval, which relate the dependent variables of adjacent grid points by some suitable quadrature. In the case of this paper, Hermite-Simpson is the quadrature used.

C. Rosmann et al. [3] presented time-optimal solutions using direct Hermite-Simpson collocation. The solutions which used control splines with a free midpoint led to oscillations in the time-optimal solutions. The solutions which used linear mean control splines had no such oscillations. Since such oscillations are undesirable in practical application, and the ultimate goal of this solver is to deal with phases with time as a free variable, the control variables were represented using linear mean control splines.

Let $\mathbf{c}_k \in \mathbb{R}^{m \times 1}$ represent the collocation constraint vector at a grid point $k + 1$. Let there be N grid points. Then at a particular grid-point k : for $\mathbf{x}_k \in \mathbb{R}^{m \times 1}$, $\mathbf{u}_k \in \mathbb{R}^{(m+n) \times 1}$

$$\bar{\mathbf{z}}_k = \mathbf{I}_{m+n} \mathbf{z}_{k+1} + \frac{h_k}{8} \begin{bmatrix} \mathbf{f}_k - \mathbf{f}_{k+1} \\ \mathbf{0}_{n \times 1} \end{bmatrix} \quad (3)$$

where

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix} \in \mathbb{R}^{(m+n) \times 1} \quad \text{and} \quad \mathbf{f}_k = \mathbf{f}(\mathbf{z}_k, t_f).$$

The transcribed form of the optimal control problem, with terminal constraints to the state variables is now presented as follows:

$$\min_{\mathbf{u}_k, \forall k} \phi(\mathbf{x}_f, t_f) + \sum_{k=1}^{N-1} \frac{h_k}{6} \left(L(\mathbf{x}_k, \mathbf{u}_k, t_f) + 4L(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k, t_f) + L(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, t_f) \right) \quad (4)$$

subject to

$$\begin{aligned} (\mathbf{x}_f, t_f) &= 0 \quad (\text{end conditions}) \\ \mathbf{c}_k &= \mathbf{I}_m \mathbf{x}_k - \mathbf{I}_m \mathbf{x}_{k+1} - \frac{h_k}{6} (\mathbf{f}_k + 4\bar{\mathbf{f}}_k + \mathbf{f}_{k+1}) = \mathbf{0}, \quad \text{for all } k \end{aligned}$$

Here, $\mathbf{c}_k$ is referred to as a defect constraint. In the event that there are multiple trajectory phases, the following constraints must be imposed:

$$\mathbf{x}_f^i = \mathbf{x}_0^{i+1} \quad (5)$$

where the superscript i indicates the phase, and $\mathbf{x}_0^i$ indicates the initial state vector of phase i .

This new problem can be posed to an NLP solver in order for a solution to be obtained. It is often of importance to scale variables in order to improve robustness of an NLP solver [4]. In this work, state variables were scaled manually as the author saw fit.

2.2 Warm Starts

The developed optimal control solver will make use of warm starts. That is, obtaining the solution of simpler versions of the actual problem, in order to initialise the solver when attempting to obtain the solution of the actual problem. This provides solvers with good initial guesses. Provision of good initial guesses to Newton-iterative solvers increases speed and robustness of the solver, hence it is advantageous to make use of warm starts [5].

In this work, warm starts will be taken advantage of by obtaining a solution for a coarse grid. An estimate of this solution's discretisation error will be tested, if this estimate of the discretisation error is above a defined tolerance, the solver will make use of the coarse grids solution to determine an initial guess for a finer grid. The solver will then obtain a solution for this new grid, and the process will repeat.

2.3 Choice of Non-Linear Programming

The two popular NLP algorithms for solving direct optimal control problems are SQP and Interior point methods. SQP algorithms have the capacity to make use of warm starts. Thus the SQP algorithm can take advantage of a re-meshing strategy.

Interior point methods tend to increase the non-linearity of functions, which can present difficulties for already highly non-linear systems. Each iteration of the Interior point method must be strictly feasible, this prevents it from using warm starts. Hence interior point methods are unsuitable if a re-meshing strategy is to be used [6].

The class of problems the developed optimal control solver is intended to solve are highly non-linear. In addition to this, it is very desirable to take advantage of warm starts, hence a SQP algorithm was chosen.

2.4 First Derivative Information

The following section presents the analytical first derivative information of the defect constraints of the direct Hermite-Simpson collocation method, with linear mean control splines. In addition to this, first derivative information of the integral component of the cost function is provided.

For a particular phase of the trajectory:

Let

$$\bar{\mathbf{f}}_k = \mathbf{f}(\bar{\mathbf{z}}_k, t_k), \quad \text{and} \quad \mathbf{J}_k = \mathbf{J}_{\mathbf{f}_k}(\mathbf{z}_k), \quad \bar{\mathbf{J}}_k = \mathbf{J}_{\bar{\mathbf{f}}_k}(\bar{\mathbf{z}}_k)$$

Then the first derivative information of the defect constraint of a particular grid point is represented as follows:

$$\begin{bmatrix} \left(\frac{\partial \mathbf{c}_k}{\partial \mathbf{z}_k} \right)^T \\ \left(\frac{\partial \mathbf{c}_k}{\partial \mathbf{z}_{k+1}} \right)^T \end{bmatrix} = \begin{bmatrix} \mathbf{I}_m \\ \mathbf{O}_{n \times m} \\ \mathbf{I}_m \\ \mathbf{O}_{n \times m} \end{bmatrix} - \frac{h_k}{6} \begin{bmatrix} \left(\frac{\partial \mathbf{f}_k}{\partial \mathbf{z}_k} \right)^T \\ \left(\frac{\partial \mathbf{f}_{k+1}}{\partial \mathbf{z}_{k+1}} \right)^T \end{bmatrix} - \frac{2h_k}{3} \left(\frac{\partial \bar{\mathbf{z}}_k}{\partial \mathbf{p}_k} \right)^T \left(\frac{\partial \bar{\mathbf{f}}_k}{\partial \bar{\mathbf{z}}_k} \right)^T \quad (6)$$

$$= \begin{bmatrix} \mathbf{I}_m \\ \mathbf{O}_{n \times m} \\ \mathbf{I}_m \\ \mathbf{O}_{n \times m} \end{bmatrix} - \frac{h_k}{6} \begin{bmatrix} (\mathbf{J}_k)^T \\ (\mathbf{J}_{k+1})^T \end{bmatrix} - \frac{2h_k}{3} \left\{ \begin{bmatrix} \mathbf{I}_{m+n} \\ \mathbf{I}_{m+n} \end{bmatrix} + \begin{bmatrix} (\mathbf{J}_k)^T & \mathbf{O}_{n \times (m+n)} \\ (\mathbf{J}_{k+1})^T & \mathbf{O}_{n \times (m+n)} \end{bmatrix} \right\} (\bar{\mathbf{J}}_k)^T$$

$$\begin{aligned} \left(\frac{\partial \mathbf{c}_k}{\partial t_f} \right)^T &= \frac{-h_k}{6} \left(\left(\frac{\partial (\mathbf{f}_k + 4\bar{\mathbf{f}}_k + \mathbf{f}_{k+1})}{\partial t_f} \right)^T + 4 \left(\frac{\partial \bar{\mathbf{z}}_k}{\partial t_k} \right)^T \left(\frac{\partial \bar{\mathbf{f}}_k}{\partial \bar{\mathbf{z}}_k} \right)^T \right) \\ &= \frac{-h_k}{6} \left(\left(\frac{\partial (\mathbf{f}_k + 4\bar{\mathbf{f}}_k + \mathbf{f}_{k+1})}{\partial t_f} \right)^T + \frac{h_k}{2} \begin{bmatrix} \left(\frac{\partial (\mathbf{f}_k - \mathbf{f}_{k+1})}{\partial t_f} \right)^T \\ \mathbf{0}_{n \times m} \end{bmatrix}^T \right) (\bar{\mathbf{J}}_k)^T \end{aligned} \quad (7)$$

If the system is autonomous (not explicitly dependent on the independent variable):

$$\left(\frac{\partial \mathbf{c}_k}{\partial \mathbf{z}_k} \right)^T = \frac{-h_k}{6} \left(\left(\frac{\partial (\mathbf{f}_k + 4\bar{\mathbf{f}}_k + \mathbf{f}_{k+1})}{\partial t_f} \right)^T + \frac{h_k}{2} \begin{bmatrix} \left(\frac{\partial (\mathbf{f}_k - \mathbf{f}_{k+1})}{\partial t_f} \right)^T \\ \mathbf{0}_{n \times 1} \end{bmatrix}^T \right) (\bar{\mathbf{J}}_k)^T \quad (8)$$

Let

$$\mathbf{y}_1 = [\mathbf{z}_1^T, \mathbf{z}_2^T, \dots, \mathbf{z}_N^T, t_f]^T \in \mathbb{R}^{(N(m+n)+1) \times 1}, \quad \text{and} \quad \mathbf{c} = [\mathbf{c}_1^T, \mathbf{c}_2^T, \dots, \mathbf{c}_{N-1}^T]^T \in \mathbb{R}^{(N-1)m \times 1}$$

The transpose of the Jacobian of the nonlinear constraints is then

$$(\mathbf{J}_{y_1}(\mathbf{c}))^T = \begin{bmatrix} \left(\frac{\partial \mathbf{c}_1}{\partial \mathbf{z}_1}\right)^T & \mathbf{0}_{(m+n) \times m} & \mathbf{0}_{(m+n) \times m} & \cdots & \mathbf{0}_{(m+n) \times m} \\ \left(\frac{\partial \mathbf{c}_1}{\partial \mathbf{z}_2}\right)^T & \left(\frac{\partial \mathbf{c}_2}{\partial \mathbf{z}_2}\right)^T & \mathbf{0}_{(m+n) \times m} & \cdots & \mathbf{0}_{(m+n) \times m} \\ \mathbf{0}_{(m+n) \times m} & \left(\frac{\partial \mathbf{c}_2}{\partial \mathbf{z}_3}\right)^T & \left(\frac{\partial \mathbf{c}_3}{\partial \mathbf{z}_3}\right)^T & \cdots & \mathbf{0}_{(m+n) \times m} \\ \mathbf{0}_{(m+n) \times m} & \mathbf{0}_{(m+n) \times m} & \left(\frac{\partial \mathbf{c}_3}{\partial \mathbf{z}_4}\right)^T & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \left(\frac{\partial \mathbf{c}_{N-1}}{\partial \mathbf{z}_{N-1}}\right)^T \\ \mathbf{0}_{(m+n) \times m} & \mathbf{0}_{(m+n) \times m} & \mathbf{0}_{(m+n) \times m} & \cdots & \left(\frac{\partial \mathbf{c}_{N-1}}{\partial \mathbf{z}_N}\right)^T \end{bmatrix} \quad (9)$$

The gradient of the integral component of the cost function for a particular phase is as follows:

Let

$$L_k = L(\mathbf{z}_k, t_f), \text{ and } \bar{L}_k = L(\bar{\mathbf{z}}_k, t_f)$$

and let

$$\mathbf{G}_k = \left(\frac{\partial L_k}{\partial \mathbf{z}_k}\right)^T; \bar{\mathbf{G}}_k = \left(\frac{\partial \bar{L}_k}{\partial \bar{\mathbf{z}}_k}\right)^T$$

Then define $C_k = \frac{h_k}{6} (L(\mathbf{x}_k, \mathbf{u}_k, t_f) + 4L(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k, t_f) + L(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, t_f))$, and thus

$$\nabla \begin{bmatrix} \mathbf{z}_k \\ \mathbf{z}_{k+1} \end{bmatrix} C_k = \frac{h_k}{6} \left(\begin{bmatrix} \mathbf{G}_k \\ \mathbf{G}_{k+1} \end{bmatrix} + 4 \left(\frac{\partial \bar{\mathbf{z}}_k}{\partial \mathbf{t}_k} \right)^T \left(\frac{\partial \bar{\mathbf{f}}_k}{\partial \bar{\mathbf{z}}_k} \right)^T \right) \quad (10)$$

$$= \frac{h_k}{6} \left(\begin{bmatrix} \mathbf{G}_k \\ \mathbf{G}_{k+1} \end{bmatrix} + 4 \left(\begin{bmatrix} \mathbf{I}_{m+n} \\ \mathbf{I}_{m+n} \end{bmatrix} + \begin{bmatrix} (\mathbf{J}_k)^T & \mathbf{0}_{n \times (m+n)} \\ (\mathbf{J}_{k+1})^T & \mathbf{0}_{n \times (m+n)} \end{bmatrix} \right)^T \bar{\mathbf{G}}_k \right) \quad (11)$$

Now, we define

$$\mathbf{H}_k = \begin{cases} \begin{bmatrix} \nabla \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} C_1 \\ \mathbf{0}_{N-2m+1} \end{bmatrix} & \text{if } k = 1 \\ \begin{bmatrix} \mathbf{0}_{(m+n)(k-1)} \\ \nabla \begin{bmatrix} \mathbf{z}_k \\ \mathbf{z}_{k+1} \end{bmatrix} C_k \\ \mathbf{0}_{(m+n)(N-k-1)+1} \end{bmatrix} & \text{if } 1 < k \leq N-1 \end{cases} \quad (12)$$

Then:

$$\begin{aligned} \frac{\partial C_k}{\partial t_f} &= \sum_{k=1}^{N-1} \frac{h_k}{6} \left(\frac{\partial (L_k + 4\bar{L}_k + L_{k+1})}{\partial t_f} + \frac{h_k}{2} \left(\frac{\partial \bar{\mathbf{z}}_k}{\partial t_f} \right)^T \left(\frac{\partial \bar{L}_k}{\partial \bar{\mathbf{z}}_k} \right)^T \right) \\ &= \sum_{k=1}^{N-1} \frac{h_k}{6} \left(\frac{\partial (L_k + 4\bar{L}_k + L_{k+1})}{\partial t_f} + \frac{h_k}{2} \left[\left(\frac{\partial (\mathbf{f}_k - \mathbf{f}_{k+1})}{\partial t_f} \right)^T \right]^T \bar{\mathbf{G}}_k \right) \end{aligned}$$

If the system is autonomous, then $\frac{\partial C_k}{\partial t_f}$ becomes:

$$\frac{\partial C_k}{\partial t_f} = \sum_{k=1}^{N-1} \frac{h_k}{6} \left(\frac{\partial (L_k + 4\bar{L}_k + L_{k+1})}{\partial t_f} + \frac{h_k}{2} \left[\begin{bmatrix} (\mathbf{f}_k - \mathbf{f}_{k+1}) \\ t_f \end{bmatrix}^T \right] \bar{\mathbf{G}}_k \right) \quad (13)$$

The gradient of C is thus

$$\nabla_{y_1} C = \sum_{k=1}^{N-1} \mathbf{H}_k + \left[\mathbf{0}_{(m+n)N} \frac{\partial C}{\partial t_f} \right]$$

In the event that multiple trajectory phases exist, the derivative information is as follows:

If t_f^i represents the final time of phase i , $\mathbf{y}_k$ represents a vector of all the unknown variables of phase k , and $\mathbf{c}^i$ represents a vector of the collocation constraints of phase i , and P be the number of phases. Let

$$\mathbf{y} = [\mathbf{y}_1^T, \mathbf{y}_2^T, \dots, \mathbf{y}_P^T]^T \quad \text{and} \quad \mathbf{cc} = [\mathbf{c}^1, \mathbf{c}^2, \dots, \mathbf{c}^P]^T$$

Then

$$(\mathbf{J}_y(\mathbf{cc}))^T = \begin{bmatrix} (\mathbf{J}_{y_1}(\mathbf{c}^1))^T & 0 & 0 & 0 \\ 0 & (\mathbf{J}_{y_2}(\mathbf{c}^2))^T & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & (\mathbf{J}_{y_P}(\mathbf{c}^P))^T \end{bmatrix} \quad (14)$$

and $\nabla_y C = [(\nabla_{y_1} C)^T, (\nabla_{y_2} C)^T, \dots, (\nabla_{y_P} C)^T]^T$.

2.5 Interpolation of Solution and Error Estimation

In order to make use of coarse grids when using direct collocation, interpolation of the solution is used to both represent the solution, and estimate discretisation error. The interpolation method used must be consistent with the quadrature method used, in this case Hermite-Simpson.

The interpolation function of the state and control variables for a particular grid interval is as follows: First, if $p = t - t_k$, then

$$\tilde{\mathbf{x}}(t) = \mathbf{x}_k + \left(\frac{p}{h_k}\right) \mathbf{f}_k + \frac{1}{2} \left(\frac{p}{h_k}\right)^2 (-3\mathbf{f}_k + 4\bar{\mathbf{f}}_k - \mathbf{f}_{k+1}) + \frac{1}{3} \left(\frac{p}{h_k}\right)^3 (2\mathbf{f}_k - 4\bar{\mathbf{f}}_k + 2\mathbf{f}_{k+1}) \quad (15)$$

and

$$\frac{d\tilde{\mathbf{x}}(t)}{dt} = \mathbf{f}_k + \left(\frac{p}{h_k}\right) (-3\mathbf{f}_k + 4\bar{\mathbf{f}}_k - \mathbf{f}_{k+1}) + \left(\frac{p}{h_k}\right)^2 (2\mathbf{f}_k - 4\bar{\mathbf{f}}_k + 2\mathbf{f}_{k+1}) \quad (16)$$

with $\tilde{\mathbf{u}}(t) = \mathbf{u}_k + \frac{p}{h_k} (\mathbf{u}_{k+1} - \mathbf{u}_k)$.

The relative discretisation error for each mesh or grid interval is calculated as follows [6]:

$$\eta_k = \int_{t_k}^{t_{k+1}} \left| \frac{d\tilde{\mathbf{x}}(t)}{ds} - \mathbf{f}(\tilde{\mathbf{x}}(s), \tilde{\mathbf{u}}(s), s) \right| ds \quad (17)$$

$$w_j = \max_k \left\{ \left| \frac{d\tilde{x}_{j,k}}{dt} \right|, |\tilde{x}_{j,k}| \right\} \quad (18)$$

$$\epsilon = \max_{j,k} \frac{\eta_{j,k}}{w_j + 1} \quad (19)$$

Where j indicates the j th state variable, k indicates the k th grid interval, and ϵ is the maximum relative local error. The integral which is equivalent to η_k is solved with Romberg quadrature [6].

3 Examples

3.1 Linear case

The validation example presented is an exercise taken from page 60 of Lawden [7]. This problem was chosen as it has an analytical solution. The problem is as follows:

The dynamic system subject to the following state equation

$$\dot{x} = -x + u \quad (20)$$

Where x is the state variable, and u is the control variable. The system starts at a time of 0 and ends at a time of 1 second. The initial values for the state variable is 1. Then the optimization problem is set as follows:

Optimise the cost function, C ,

$$C = \int_0^1 \left(x^2 + \frac{1}{3}u^2 \right) dt$$

Then the Hamiltonian of the problem is

$$H = x^2 + \frac{1}{3}u^2 + \lambda(u - x)$$

and thus

$$\frac{\partial H}{\partial u} = \frac{2}{3}u + \lambda = 0 \quad (21)$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x} = -2x + \lambda \quad (22)$$

Substituting the value for λ from the equation (21) leads to the following system of ordinary differential equations

$$\dot{x} = u - x \quad (23)$$

$$\dot{u} = u + 3x \quad (24)$$

Since the final value of x is unconstrained, the value of λ , at the final time, must be 0. Therefore the value of u must be zero at the final time. This system of differential equations (24) has been solved for using the Laplace transform, providing the following solution for the optimal control and cost

$$u(t) = -\frac{3 \sinh 2(1-t)}{2 \cosh 2 + \sinh 2}$$

$$C = \frac{1}{1 + 2 \tanh 2} = 0.3252424$$

The cost, C , and control are both consistent with the solution provided in [7].

The following matrices were defined in order to solve the problem using the developed solver.

$$J_k = [-1, 1], \quad \text{and} \quad G_k = \begin{bmatrix} 2x_k \\ \frac{2}{3}u_k \end{bmatrix}$$

The problem was solved in 0.17 seconds with re-meshing, using the developed optimal control solver. Without the provision of analytical first derivative information, the solver required 3.41 seconds to obtain the same solution. This solver time is the sum of the time taken by the NLP to determine a solution for each mesh. The relative discretisation error of the presented

solution is 5.2845×10^{-10} . The maximum difference between the analytical and numerical solution of the control variable is 1.8201×10^{-4} . The cost found by the solver is 0.3252424, implying a 0% objective function error. The variables were unscaled. Figure 1 below depicts the time history of the numerical solution of the state and control variables, the control error, and the analytical optimal control. The control error is the difference between the analytical and numerical solution of the control history, at discrete points in time. It should be noted that only first order conditions were used to determine the analytical optimal control. Figure 2 depicts the phase trajectory.

Figure 1. Numerical results of the linear example

Figure 2. Phase trajectory of the linear example

3.2 Nonlinear case

The validation example presented is a minimum time co-planar circular orbital transfer. The system dynamics was augmented from Moyer & Pinkham [8], such that the problem was autonomous. The specific version of this problem solved for in Moyer & Pinkham will be solved. This problem attempts to find the minimum time taken for a spacecraft to transfer from the average circular orbit of the Earth around the Sun, to the average circular orbit of Mars around the Sun. The system of units used in Moyer & Pinkham were normalised. The time unit is equivalent to 58.18 days. The distance unit used is AU (astronomical unit). The mass unit is equivalent to 46.58 slugs, or 14.594 kg.

The following assumptions were used:

- Constant Thrust
- Gravity and Thrust are the only Forces acting on the rocket

- Earth and Mars have circular, co-planar orbits
- Only the gravitational attraction of the Sun is considered.

The system dynamics are as follows:

$$\begin{aligned}\dot{r} &= v_r \\ \dot{\theta} &= \frac{v_\theta}{r} = \omega \\ \dot{v}_r &= \frac{v_\theta^2}{r} - \frac{\mu}{r^2} + \frac{T u_1}{m} \\ \dot{v}_\theta &= -\frac{v_r v_\theta}{r} + \frac{T u_2}{m} \\ \dot{m} &= -|\dot{m}| \\ \text{subject to } u_1^2 + u_2^2 &= 1\end{aligned}$$

Where r is the radial distance of the spacecraft from the sun, θ is the angular position of the spacecraft, v_r is the radial speed, v_θ is the circumferential speed, ω is the angular speed, T is the thrust constant, m is the mass, μ is the standard gravitational parameter of the sun, and u_1 and u_2 are the components of the unit vector which represent the thrust direction. In this example, the aforementioned unit vector components are the control variables.

The normalised boundary constraints are:

$$\begin{aligned}r(0) &= 1; & m(0) &= 1; & r(t_f) &= 1.525; & v_r(0) &= 0; & v_r(t_f) &= 0 \\ v_\theta(0) - \sqrt{\frac{\mu}{r(0)}} &= 0; & v_\theta(t_f) - \sqrt{\frac{\mu}{r(t_f)}} &= 0\end{aligned}$$

The following constants are used:

$$T = 0.8098; \quad |\dot{m}| = 0.07487; \quad \mu = 1$$

The solution to this problem, found in [8] by using an indirect Newton-Raphson optimal control method, is a minimum time of 3.3207 time units or 193.2 days. This solution was found in 36 seconds. The following figure, obtained from [9], is a representation of the optimal trajectory, and the corresponding optimal control history. The arrows represent the thrust direction unit vectors.

Figure 3. Bryson & Ho Trajectory solution [9]

The matrix below was defined in order to solve the problem using the developed solver:

$$J = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{v_\theta}{r^2} & 0 & 0 & \frac{1}{r} & 0 & 0 & 0 \\ -\frac{v_\theta}{r^2} + \frac{2\mu}{r^3} & 0 & 0 & \frac{2v_\theta}{r} & -\frac{T u_1}{m^2} & \frac{T}{m} & 0 \\ \frac{v_\theta v_r}{r^2} & 0 & -\frac{v_\theta}{r} & -\frac{v_r}{r} & -\frac{T u_2}{m^2} & 0 & \frac{T}{m} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The problem was solved in 1.34 seconds with re-meshing, using the developed optimal control solver. Without the provision of analytical first derivative information, the solver required 14.82 seconds to obtain the same solution. This solver time is the sum of the time taken by the NLP to determine a solution for each mesh. The relative discretisation error of the presented solution is 7.659×10^{-7} . The cost found by the solver is 3.206 time units, implying a 0.00301 % objective function error. The variables were unscaled. The figure below is the optimal trajectory solution found by the optimal control solver.

Figure 4. Numerical results of the example

As stated in [9], the optimal radial thrust component is outward for roughly the first half of the trajectory, and inward for the rest. The results obtained are consistent with this observation. Figure 5 below compares the thrust direction history determined by Moyer & Pinkham [8], to the history determined by the developed optimal control solver. This figure shows that the optimal thrust direction history solved for is consistent with the optimal thrust direction history (represented by θ^*) found by Moyer & Pinkham. In this figure, the thrust direction is represented by the angle measured from the local horizontal to the thrust direction vector.

Figure 5. Control history found by solver (left) and Control history found in [8] (right)

The phase trajectory plots, are as follows:

Figure 6. Phase trajectory of state variables

4 Conclusion

Hermite-Simpson direct collocation methods have been effectively used to solve various optimal control problems. However typically the derivative information found using these methods have been estimated with finite difference methods. This paper presented a relatively simple formulae for determining first order analytical derivative information, for the Hermite-Simpson direct collocation method. This information allows for robust and faster solving of optimal control problems. The derivatives were tested in a developed solver, and have been proven to function in various validation examples, one of which was provided in this paper.

The first derivative information of the initial mesh was made to undergo a derivative information check by Matlab. Matlab tests the derivative information by comparing the analytically determine derivatives with numerical estimates (using finite differencing methods). The first derivative information passed Matlab’s test, further validating the obtained results. The provision of first order analytical derivative information had successfully reduced the solve time of the solver by a significant margin.

This work, was constrained by Matlab’s SQP algorithm, which estimates second derivative information using dense finite differencing methods. In order to improve the overall solver performance, and to compare it fairly to other optimal control solvers, the solver should be used in conjunction with an SQP algorithm capable of exploiting the sparsity of the second derivative information.

References

- [1] Betts J.T., *A Survey of Numerical Methods for Trajectory Optimization*, Journal of Guidance, Control, and Dynamics, **21** No. 2, 193-207 (1998)
- [2] Boudjemaa R., Cox M.G., Forbes A.B., and Harris P.M., *Automatic Differentiation Techniques and their Application in Metrology*, March 26,(2003)
http://www.npl.co.uk/ssfm/download/documents/cmsc26_03.pdf. (Accessed 30 April 2021).
- [3] Rosmann C., Makarow A., and Bertram T., *Time-optimal control with direct collocation and variable discretization*, arXiv preprint arXiv:2005.12136, 2020).
- [4] Nocedal J. and Wright S.J., *Numerical Optimization* (Springer, New York City, 1999)
- [5] Forsgen A., Ceragioli F., Dontchev A., Futura K., Marti K., and Pandolfi L., *System Modeling and Optimization* (Springer-Verlag, Boston, pp 51-66, 2006)
- [6] Betts J.T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming (2nd ed., (SIAM, Philadelphia, 2010)*
- [7] Lawden D.F., *Analytical Methods of Optimisation*, (Scottish academic press Ltd. Edinburgh, 1975)
- [8] Moyer, H.G. and Pinkham, G., *Several Trajectory Optimization Techniques Part II: Application*, (in Computing Methods in Optimization Problems, New York, 1964)
- [9] Bryson, A.E. and Ho, Y.C., *Applied Optimal Control* ,(Washington D.C.: Hemisphere Publishing Corporation, 1975)