MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

# A Numerical Algorithm to Solve a Class of Multi-Point Boundary Value Problems

Wang, Y.; Zhao, Y.; Bortoff, S.A.

TR2013-067 July 2013

**Abstract**

This note presents a numerical algorithm to solve a class of multi-point boundary value problems (MBVPs). Compared to conventional multiple shooting, the proposed method uses less parameters to exploit advantages of single shooting approaches. Unknown parameters are updated by a two-step algorithm which improves the convergence of parameters over existing one-step algorithm. The main idea of the two-step algorithm is to decouple the effects on boundary conditions between two sets of parameters: state and costate, and switch times. The proposed algorithm can compute the solution of a class of MBVPs faster than various existing methods. An application example illustrates the effects of the algorithm.

*Chinese Control Conference (CCC)*

# A numerical algorithm to solve a class of multi-point boundary value problems

Yebin Wang[1], Yiming Zhao[1] and Scott A. Bortoff[1]
1. Mitsubishi Electric Research Laboratories
201 Broadway, Cambridge, MA 02139, U. S. A
E-mail: {yebinwang,yzhao,bortoff}@merl.com

**Abstract:** This note presents a numerical algorithm to solve a class of multi-point boundary value problems (MBVPs). Compared to conventional multiple shooting, the proposed method uses less parameters to exploit advantages of single shooting approaches. Unknown parameters are updated by a two-step algorithm which improves the convergence of parameters over existing one-step algorithm. The main idea of the two-step algorithm is to decouple the effects on boundary conditions between two sets of parameters: state and costate, and switch times. The proposed algorithm can compute the solution of a class of MBVPs faster than various existing methods. An application example illustrates the effects of the algorithm.

**Key Words:** boundary value problem, optimal control

## 1 Introduction

Optimal control theory has a wide range of applications such as minimum fuel or optimal landing problems in aerospace [1], time optimal problem in manufacturing [2–7] etc. Dynamic programming and minimum principle are fundamental theories to attack optimal control problems, albeit numerical optimization-based approaches have been widely used [8, 9]. When treated with dynamic programming, an optimal control problem is reduced to Hamiltonian-Jacobi-Bellman (HJB). The minimum principle yields a set of ordinary differential equations (ODEs) with boundary conditions. For a constrained optimal control problem, the minimum principle usually gives a set of piecewise ODEs, i.e., a multi-point boundary value problem (MBVP). Methods to solve an MBVP have been investigated for decades. This note focus on the method to achieve fast computation of the solution of a class of MBVPs.

Shooting methods [10–16] represent classic but effective methods to solve an MBVP as an initial value problem [17]. Recent work [18] proposes a continuous method to achieve fast computation of solutions of boundary value problems. Single shooting has some pros and cons such as a small number of parameters, easy implementation, difficulty to converge for problems having nonlinearity or long time horizon, difficulty to handle state constrained problems. Multiple shooting [10, 12] aims to relieve limitations of single shooting, and satisfactory in term of robustness, convergence of solutions etc. However, the computational load of conventional multiple shooting is high, which is mainly due to overparametrization of the problem. Since multiple shooting algorithms generally transform the original MBVP into a large nonlinear programming problem (NLP), it is difficult to meet the computation time target for real-time applications.

This note proposes a method which exploits the advantages of the single and multiple shooting to achieve fast computation of the solution of a class of MBVPs. The proposed method takes a different parametrization scheme to speed up the computation by introducing less parameters than multiple shooting, as well as allows the handling the state constrained problems. The proposed method treats the state and costate, and switch times as parameters, thus performs as single shooting for each segment of the system trajectory, but as multiple shooting for the entire trajectory. The new method suffers similar convergence issue due to the coupling effects on boundary conditions of the guesses of state and costate and switch times. A two-step algorithm, which updates the state and costate parameters, and switch times alternately, is proposed to decouple the interaction, thus improves the convergence property over single shooting.

This paper is organized as follows. Section 2 states an optimal control problem of a motivation example. In Section 3, the main algorithm is illustrated in details. Section 4 demonstrates the application of the proposed method to a MBVP associated with the motivation example. Conclusions are made in Section 5.

## 2 A Motivation Example

Consider the following model in the state space form

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= dx_2 + bu + c, \end{aligned} \tag{1}$$

where $d$ is the viscous friction coefficient, $c$ is the Coulomb friction, and $u$ is the control input. Assume $b, c, d$ are constant. We shall compute the optimal trajectory minimizing the following cost function

$$E = \int_0^T (Ru^2 + K_s|u| + K_\tau x_2 u)\mathrm{t}, \tag{2}$$

where $R, K_s, K_\tau$ are constant coefficients. This is equivalent to solve the following optimal control problem.

**Problem 2.1** *Given the plant* (1)*, the initial state* $x(0) = x_0 = (0,0)^T$*, the final state* $x(T) = (r,0)^T$*, and the final time* $T$*, find the control* $u^*$ *which minimizes the cost function* (2) *subject to acceleration and velocity constraints*

$$0 \le x_2 \le v_{max}, \quad |\dot{x}_2| \le a_{max}, \tag{3}$$

*where* $v_{\max}, a_{max}, r$ *are known constants.*

### 2.1 Benchmark: Direct Transcription with Mesh Refinement

Direct transcription methods solve optimal control problems by converting them directly into equivalent mathematical programming problems without applying any optimality conditions. In particular, consider the direct transcription of Problem 2.1 using trapezoidal integration rule on a mesh $0 = t_0 < t_1 < \cdots < t_N = T$. The control $u(t)$ is discretized on $(t_0 + t_1)/2, \ldots, (t_{N-1} + t_N)/2$ as $u_1, \ldots, u_N$, and the states $x_1(t)$, $x_2(t)$ are descritized as $x_{1,0}, x_{1,1}, \ldots, x_{1,N}$ and $x_{2,0}, x_{2,1}, \ldots, x_{2,N}$, respectively. Let $X = [x_{1,0}, \ldots, x_{1,N}, x_{2,0}, \ldots, x_{2,N}]$, $U = [u_1, \ldots, u_N]$, $\Delta_i = t_i - t_{i-1}$, $i = 1, \ldots, N$. The discretized problem is given by

**Problem 2.2 (Discretized problem)**

$$
\begin{aligned}
\min_{X,U} \quad & \sum_{i=1}^{N} \left( R u_i^2 + K_s |u_i| + K_\tau u_i \frac{x_{2,i-1} + x_{2,i}}{2} \right) \Delta_i \\
\text{s.t.} \quad & \frac{x_{1,i+1} - x_{1,i}}{\Delta_i} = \frac{x_{2,i} + x_{2,i+1}}{2}, \\
& \frac{x_{2,i+1} - x_{2,i}}{\Delta_i} = d \frac{x_{2,i} + x_{2,i+1}}{2} + b u_i + c \\
& \textit{for } i = 0, \ldots, N-1, \\
& 0 \le x_{2,i} \le v_{\max}, \\
& \left| d \frac{x_{2,i} + x_{2,i+1}}{2} + b u_i + c \right| \le a_{\max}, i = 0, \ldots, N \\
& x_{1,0} = 0,\ x_{1,N} = r, \\
& x_{2,0} = 0,\ x_{2,N} = 0.
\end{aligned}
$$

The mathematical programming problem 2.2 is then solved using optimization solvers to find approximate solutions to the original optimal control problems. Unlike shooting methods, direct transcription methods do not require the user's expertise in optimal control theory, and therefore, are easy to use.

The mesh refinement technique was introduced to enhance the accuracy, computational efficiency, and robustness of direct transcription methods [19]. In this paper, we use a recent direct transcription method with density function based mesh refinement scheme, which is introduced in [20], as a benchmark for evaluating the proposed method.

In that density function based mesh refinement method [20], the optimal control problem is first directly transcribed into a nonlinear programming problem (NLP) in the form of 2.2 by discretizing the integral cost function (2), system dynamics differential equations (1) and other constraints (3) on an uniform mesh ($t_0, \ldots, t_N$ are uniformly distributed on $[0, T]$).

After the first solution is obtained, a grid density function is computed based on the solution, and mesh points are added and redistributed according to the density function. Once a new mesh is obtained, the optimal control problem is discretized on the new mesh again to form another nonlinear program, and solved for a more accurate solution. Such a process is repeated until certain stopping criterion has been satisfied. The mesh refinement process allocates the grid points over the whole time interval while putting emphasis on the points of discontinuity of the state and control variables addressed by the density function. Hence, the size of the resulting NLP problems are typically much smaller than those obtained using direct transcription on an uniform mesh with similar resolution, which helps reducing the computation time.

### 2.2 Indirect Method

Problem 2.1 has been investigated thoroughly in [21, 22], where the minimum principle and the optimal principle are applied to establish necessary optimality conditions, the complete set of structures of optimal solutions, and the complete set of multi-point boundary value problems (MBVPs) corresponding to the set of structures. We cite a few results to make this note self-contained.

We express the Hamiltonian $H$ piecewisely,

$$
H = \begin{cases} H_{11} = R u^2 + K_s u + K_\tau x_2 u + \bar{H}_1, \ u \ge 0, \\ H_{12} = R u^2 - K_s u + K_\tau x_2 u + \bar{H}_1, \ u < 0, \end{cases} \tag{4}
$$

where

$$
\begin{aligned}
\bar{H}_1 =& \lambda^T (Ax + Bu + C) + \mu^T \begin{bmatrix} dx_2 + c + bu \\ -dx_2 - c - bu \end{bmatrix} \\
& + \nu^T \begin{bmatrix} dx_2 + c + bu - a_{max} \\ -a_{max} - dx_2 - c - bu \end{bmatrix}.
\end{aligned}
$$

Lagrange multipliers $\mu, \nu$ correspond to the velocity and acceleration constraints, respectively.

#### 2.2.1 Optimal Control

Assuming that at any time instant, only one constraint is active. Control on constrained segments is given by

$$
u = \begin{cases} -\frac{dx_2 + c}{b} = u_{vel}, \quad x_2 = v_{max}, \\ \frac{a_{max} - dx_2 - c}{b} = u_{pacc}, \quad \dot{x}_2 = a_{max}, \\ \frac{-a_{max} - dx_2 - c}{b} = u_{nacc}, \ \dot{x}_2 = -a_{max}. \end{cases} \tag{5}
$$

We solve unconstrained positive and negative controls, denoted by $u_1$ and $u_2$, from $\partial H_{11}/\partial u = 0, \partial H_{12}/\partial u = 0$ respectively

$$
u = \begin{cases} -\frac{K_\tau x_2 + K_s + b\lambda_2}{2R}, & u \ge 0, \\ -\frac{K_\tau x_2 - K_s + b\lambda_2}{2R}, & u < 0. \end{cases}
$$

If neither $\partial H_{11}/\partial u = 0$ has a positive solution nor $\partial H_{12}/\partial u = 0$ gives a negative solution, the optimal control is $u = 0$.

#### 2.2.2 Dynamics

Given the knowledge of $x, \lambda$, and the control, $x$-dynamics is well-defined. We therefore focus on the costate dynamics. Since the partial derivative of $H$ w.r.t. $x$ is well-defined, the costate dynamics can be readily obtained.

$$
\dot{\lambda} = -A^T \lambda - \begin{bmatrix} 0 \\ K_\tau u \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ d & -d \end{bmatrix} (\mu + \nu). \tag{6}
$$

When the system trajectory is along unconstrained arcs, $\mu = \nu = 0$. If an acceleration constraint is active, $\nu$ is solved from

$$
H_{11u}\big|_{u=u_{pacc}} = 0, \quad \dot{x}_2 - a_{max} \le 0 \text{ is active}, \tag{7a}
$$

$$H_{12u}\big|_{u=u_{nacc}} = 0, \quad -a_{max} - \dot{x}_2 \le 0 \text{ is active}, \tag{7b}$$

whose solutions are

$$\begin{aligned} \nu_1 &= \frac{-1}{b}\{2Ru + K_s + K_\tau x_2 + b\lambda_2\}, \\ \nu_2 &= \frac{1}{b}\{2Ru - K_s + K_\tau x_2 + b\lambda_2\}. \end{aligned} \tag{8}$$

For the velocity constraint $x_2 - v_{max} \le 0$, we have the Lagrange multiplier $\mu$ determined from $H_{11u} = 0$ with $u = u_{pvel}$

$$\mu_1 = \frac{-1}{b}\{2Ru + K_s + K_\tau x_2 + b\lambda_2\},\ x_2 = v_{max}. \tag{9}$$

#### 2.2.3 An MBVP

Assuming the knowledge of the structure of the optimal trajectory, the MBVP to be solved is well-defined and takes the following form

$$\begin{aligned} &\dot{x}_a = f(x_a, u_k, \nu, \mu),\ t \in [t_{k-1}, t_k], \\ &g(x_a(t_{k-1}^+), x_a(t_{k-1}^-), \pi) = 0,\ 1 \le k \le m-1, \\ &h(x_a(t_0), x_a(T)) = 0, \end{aligned} \tag{10}$$

where $x_a = [x, \lambda]^T$, $g, h$ are appropriate functions, $k$ denotes the $k$th segment of the optimal trajectory, $m$ is the number of segments the optimal trajectory includes, and $t_k$ is the entry time of the $k+1$th segment. The MBVP (10) can be established from applying the minimum principle [17, 23] to more general optimal control problems.

## 3 The Main Algorithm

We first make a few assumptions, then give the algorithm solving a class of MBVPs in the form of (10).

**Assumption 3.1** *The MBVP* (10) *admits a solution* $(x_a^*(t), u^*(t))$ *where* $u^*(t)$ *is continuous over* $[0, T]$.

**Remark 3.2** *Assumption 3.1 allows us to use the continuity of control as a part of Boundary Conditions (BCs) to determine switch times. This is without loss of generality, because without Assumption 3.1, the BCs determining switch times will still be well-defined.*

The main algorithm to solve the MBVP (10) is given in Figure 1, where $\beta_1, \beta_2$ are two sets of parameters, and $BC_1, BC_2$ are two sets of BCs. The main difference between the proposed and the one-step algorithm used in conventional multiple shooting solvers is that the former alternately updates two parameter sets $\beta_1, \beta_2$ on the basis of two sets of BCs $BC_1$ and $BC_2$, respectively. This is important to improve the convergence of the proposed algorithm which aims to achieve fast computation of the optimal trajectory at the expense of the convergence advantage of conventional multiple shooting methods.

**Remark 3.3** *It is not difficult to observe that the proposed algorithm actually tries to solve a sequence of optimal control problems, where at the qth iteration to update* $\beta_1^{(q)}$ *and* $\beta_2^{(q)}$*, the corresponding optimal control problem is*

$$\min_{t_k^{(q)}} \left( \min_{x^{(q)}, \lambda^{(q)}, u^{(q)}} E(x, \lambda, u, t_k) \right), \tag{11}$$

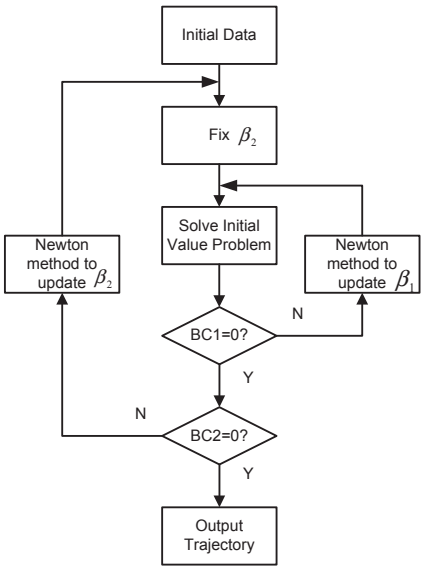


Fig. 1: Flow chart of the main algorithm

*for* $1 \le k \le m-1$. *The solution of Problem* (11) *at the qth iteration is the input of the algorithm to solve Problem* (11) *at the* $q+1$*th iteration. As a comparison, the original optimal control problem is*

$$\min_{t_k, x, \lambda, u} E(x, \lambda, u, t_k), \tag{12}$$

*for* $1 \le k \le m-1$. *Problems* (11) - (12) *clearly may have different solutions. The proposed algorithm tries to solve* (11) *instead of* (12) *is based on the realization that it is relatively difficult to compute the solution of* (12).

Without loss of generality, we assume that the solution to the MBVP has $m$ segments. Hence, the optimal trajectory switches at time instant $t_k, 1 \le k \le m-1$. We also denote $t_0 = 0, t_m = T$.

### 3.1 Parametrization of the MBVP

To simplify the presentation, we only consider parameters for the $k$th segment $(\beta_{k1}, \beta_{k2})$

$$\begin{aligned} \beta_{k1} &= ((x(t_{k-1}), \lambda(t_{k-1})), \\ \beta_{k2} &= t_k. \end{aligned}$$

Define $\beta_1 = (\beta_{11}, \ldots, \beta_{m1}), \beta_2 = (\beta_{12}, \ldots, \beta_{m2})$. The set $\beta_1$ consists of parameters representing the state and costate values at switch times $t_k, 1 \le k \le m-1$, and the set $\beta_2$ consists of parameters representing switch times $t_k, 1 \le k \le m-1$. Denote $\beta = (\beta_1, \beta_2)$ the entire set of parameters.

### 3.2 Boundary Conditions

For simplicity, we consider the BCs for the $k$th segment. If no state constraint is active, the BCs at the end of the $k$th segment take the form of

$$\begin{aligned} &\phi_x(x(t_{k-1}), \lambda(t_{k-1}), t_{k-1}, t_k^-) = x(t_k^+), \\ &\phi_\lambda(x(t_{k-1}), \lambda(t_{k-1}), t_{k-1}, t_k^-) = \lambda(t_k^+), \\ &u(t_k^-) = u(t_k^+), \end{aligned} \tag{13}$$

where $\phi_x$ and $\phi_\lambda$ represent the state and costate trajectories, respectively. BCs (13) can be partitioned into two sets $BC_{k1}$

and $BC_{k2}$ which correspond to parameter sets $\beta_{k1}$ and $\beta_{k2}$, respectively

$$
\begin{aligned}
&BC_{k1}:\\
&\qquad \phi_x(x(t_{k-1}),\lambda(t_{k-1}),t_{k-1},t_k^-)=x(t_k^+),\\
&\qquad \phi_\lambda(x(t_{k-1}),\lambda(t_{k-1}),t_{k-1},t_k^-)=\lambda(t_k^+),\\
&BC_{k2}:\\
&\qquad u(t_k^-)=u(t_k^+).
\end{aligned}
$$

The residue corresponding to $BC_{k1}, BC_{k2}$, denoted by $F_{k1}, F_{k2}$, are given by

$$
\begin{aligned}
&F_{k1}:\\
&\qquad x(t_k^+)-\phi_x(x(t_{k-1}),\lambda(t_{k-1}),t_{k-1},t_k^-),\\
&\qquad \lambda(t_k^+)-\phi_\lambda(x(t_{k-1}),\lambda(t_{k-1}),t_{k-1},t_k^-),\\
&F_{k2}:\\
&\qquad u(t_k^+)-u(t_k^-).
\end{aligned}
$$

Residues correspond to the two parameter sets $\beta_1, \beta_2$ are denoted by

$$
\begin{aligned}
F_1&=(F_{11},\ldots,F_{m1}),\\
F_2&=(F_{12},\ldots,F_{m2}).
\end{aligned}
$$

**Remark 3.4** *If a state constraint is active in the $k+1$th segment, the costate at $t=t_k$ will be discontinuous. Hence the BCs for the $k$th segment will be slightly different*

$$
\begin{aligned}
&\phi_x(x(t_{k-1}),\lambda(t_{k-1}),t_{k-1},t_k^-)=x(t_k^+),\\
&u(t_k^-)=u(t_k^+).
\end{aligned}
$$

*Meanwhile, $\lambda(t_k)$ is excluded from the parameter set.*

### 3.3 Sensitivity Equations

The procedure to find parameters satisfying BCs generally includes the computation of the jacobian of BCs or the residue $F=(F_1^T,F_2^T)^T$ w.r.t. parameters, which consists of the following components

$$
\frac{\partial x}{\partial \beta_1},\quad \frac{\partial \lambda}{\partial \beta_1},\quad \frac{\partial x}{\partial \beta_2},\quad \frac{\partial \lambda}{\partial \beta_2}.
$$

It is well-established that the gradient of BCs can be obtained from integrating sensitivity equations. For the completeness of the presentation, we recall how to compute the gradient of $F_1$ w.r.t. $\beta_1$. For the $k$th segment, the sensitivity equations of $x(t_k)$ w.r.t. $\lambda(t_{k-1})$ are computed from

$$
\begin{aligned}
&\frac{d}{dt}\frac{\partial x(t)}{\partial \lambda(t_{k-1})}\\
&=(f_x+f_u u_x)\frac{\partial x}{\partial \lambda(t_{k-1})}+f_u u_\lambda \frac{\partial \lambda}{\partial \lambda(t_{k-1})},
\end{aligned}
$$

where $t\in[t_{k-1},t_k)$, and

$$
\begin{aligned}
f_x&=\frac{\partial f(x,u(x,\lambda))}{\partial x}, & f_u&=\frac{\partial f(x,u(x,\lambda))}{\partial u},\\
u_x&=\frac{\partial u(x,\lambda)}{\partial x}, & u_\lambda&=\frac{\partial u(x,\lambda)}{\partial \lambda}.
\end{aligned}
$$

Similarly we have the sensitivity equations of $\lambda(t)$ w.r.t. $\lambda(t_{k-1})$.

$$
\begin{aligned}
\frac{d}{dt}\frac{\partial \lambda}{\partial \lambda(t_{k-1})}=&-H_{xx}^T\frac{\partial x}{\partial \lambda(t_{k-1})}-H_{x\lambda}^T\frac{\partial \lambda}{\partial \lambda(t_{k-1})}\\
&-H_{xu}^T(u_\lambda\frac{\partial \lambda}{\partial \lambda(t_{k-1})}+u_x\frac{\partial x}{\partial \lambda(t_{k-1})})\\
&-H_{x\mu}^T\left(\mu_\lambda\frac{\partial \lambda}{\partial \lambda(t_{k-1})}+\mu_x\frac{\partial x}{\partial \lambda(t_{k-1})}\right.\\
&\left.+\mu_u(u_x\frac{\partial x}{\partial \lambda(t_{k-1})}+u_\lambda\frac{\partial \lambda}{\partial \lambda(t_{k-1})})\right)\\
&-H_{x\nu}^T\left(\nu_\lambda\frac{\partial \lambda}{\partial \lambda(t_{k-1})}+\nu_x\frac{\partial x}{\partial \lambda(t_{k-1})}\right.\\
&\left.+\nu_u(u_x\frac{\partial x}{\partial \lambda(t_{k-1})}+u_\lambda\frac{\partial \lambda}{\partial \lambda(t_{k-1})})\right).
\end{aligned}
$$

where

$$
\begin{aligned}
H_{xx}^T&=\frac{\partial}{\partial x}\left(\frac{\partial H}{\partial x}\right)^T, \quad H_{x\lambda}^T=\frac{\partial}{\partial \lambda}\left(\frac{\partial H}{\partial x}\right)^T,\\
H_{xu}^T&=\frac{\partial}{\partial u}\left(\frac{\partial H}{\partial x}\right)^T, \quad H_{x\mu}^T=\frac{\partial}{\partial \mu}\left(\frac{\partial H}{\partial x}\right)^T,\\
H_{x\nu}^T&=\frac{\partial}{\partial \nu}\left(\frac{\partial H}{\partial x}\right)^T,\\
\mu_x&=\frac{\partial \mu}{\partial x},\quad \mu_\lambda=\frac{\partial \mu}{\partial \lambda},\quad \mu_u=\frac{\partial \mu}{\partial u},\\
\nu_x&=\frac{\partial \nu}{\partial x},\quad \nu_\lambda=\frac{\partial \nu}{\partial \lambda},\quad \nu_u=\frac{\partial \nu}{\partial u}.
\end{aligned}
$$

Similarly, we have the sensitivity equations on $\frac{\partial x}{\partial x(t_{k-1})}, \frac{\partial \lambda}{\partial x(t_{k-1})}$. We summarize the sensitivity equations as follows

$$
\begin{bmatrix}
\frac{d}{dt}\frac{\partial x}{\partial x(t_{k-1})}\\
\frac{d}{dt}\frac{\partial x}{\partial \lambda(t_{k-1})}\\
\frac{d}{dt}\frac{\partial \lambda}{\partial x(t_{k-1})}\\
\frac{d}{dt}\frac{\partial \lambda}{\partial \lambda(t_{k-1})}
\end{bmatrix}
=
\begin{bmatrix}
\alpha_1 I & 0 & \alpha_2 I & 0\\
0 & \alpha_1 I & 0 & \alpha_2 I\\
\alpha_3 I & 0 & \alpha_4 I & 0\\
0 & \alpha_3 I & 0 & \alpha_4 I
\end{bmatrix}
\begin{bmatrix}
\frac{\partial x}{\partial x(t_{k-1})}\\
\frac{\partial x}{\partial \lambda(t_{k-1})}\\
\frac{\partial \lambda}{\partial x(t_{k-1})}\\
\frac{\partial \lambda}{\partial \lambda(t_{k-1})}
\end{bmatrix}
\tag{14}
$$

where $I$ is the identity matrix with the same dimension as the state vector $x$, and

$$
\begin{aligned}
\alpha_1=&f_x+f_u u_x,\\
\alpha_2=&f_u u_\lambda,\\
\alpha_3=&-H_{xx}^T-H_{xu}^T u_x\\
&-H_{x\mu}^T(\mu_x+\mu_u u_x)-H_{x\nu}^T(\nu_x+\nu_u u_x),\\
\alpha_4=&-H_{x\lambda}^T-H_{xu}^T u_\lambda\\
&-H_{x\mu}^T(\mu_\lambda+\mu_u u_\lambda)-H_{x\nu}^T(\nu_\lambda+\nu_u u_\lambda).
\end{aligned}
$$

Initial conditions of sensitivity equations are

$$
\begin{aligned}
\left.\frac{\partial x(t)}{\partial x(t_{k-1})}\right|_{t=t_{k-1}}&=I, & \left.\frac{\partial x(t)}{\partial \lambda(t_{k-1})}\right|_{t=t_{k-1}}&=0,\\
\left.\frac{\partial \lambda(t)}{\partial x(t_{k-1})}\right|_{t=t_{k-1}}&=0, & \left.\frac{\partial \lambda(t)}{\partial \lambda(t_{k-1})}\right|_{t=t_{k-1}}&=I.
\end{aligned}
$$

Note that sensitivity equations for the $k$th segment is only defined over $[t_{k-1},t_k)$.

We compute the gradient of the residue $F_2$ w.r.t. the parameter set $\beta_2$ from $\partial x/\partial\beta_2, \partial\lambda/\partial\beta_2$, particularly

$$\frac{\partial x(t_k^-)}{\partial t_k}, \frac{\partial x(t_{k+1}^-)}{\partial t_k}, \frac{\partial \lambda(t_k^-)}{\partial t_k}, \frac{\partial \lambda(t_{k+1}^-)}{\partial t_k},$$

which take the following formula

$$\begin{aligned}
\frac{\partial x(t_k^-)}{\partial t_k} &= f(x(t),u(t))\Big|_{t=t_k^-},\\
\frac{\partial \lambda(t_k^-)}{\partial t_k} &= -H_x^T(x(t),\lambda(t),\mu(t),\nu(t))\Big|_{t=t_k^-},\\
\frac{\partial x(t_{k+1}^-)}{\partial t_k} &= -f(x(t),u(t))\Big|_{t=t_{k+1}^-},\\
\frac{\partial \lambda(t_{k+1}^-)}{\partial t_k} &= H_x^T(x(t),\lambda(t),\mu(t),\nu(t))\Big|_{t=t_{k+1}^-}.
\end{aligned}$$

### 3.4 Update Parameters

We need to find the parameter set $\beta^* = (\beta_1^*, \beta_2^*)$ s.t. $F_1(\beta_1^*) = 0, F_2(\beta_2^*) = 0$, which are a set of nonlinear algebraic equations. The newton method is used to update the guess of $\beta$

$$\begin{aligned}
\beta_1^{n+1} &= \beta_1^n + \rho_1^n \left(\frac{\partial F_1^n}{\partial \beta_1}\Big|_{\beta_1=\beta_1^n}\right)^{-1} F_1^n,\\
\beta_2^{(q+1)} &= \beta_2^{(q)} + \rho_2^{(q)} \left(\frac{\partial F_2^{(q)}}{\partial \beta_2}\Big|_{\beta_2=\beta_2^{(q)}}\right)^{-1} F_2^{(q)},
\end{aligned}$$

where $\rho_1^n, \rho_2^n$ are step lengths of parameter innovation, the superscript $n$ in $\beta_1, F_1, \rho_1$ denotes the $n$th step while solving (12) at the $q+1$th iteration, $0 \le q < +\infty$.

## 4 Solving the Motivation Example

The optimal solution to Problem 2.1 may consist of various segments, and its structure depends on problem data such as $T, r, a_{max}, v_{max}$. Different structure corresponds to a distinctive MBVP. To simplify the presentation, we illustrate the use of the proposed algorithm by solving the MBVP corresponding to the following problem data, $PDATA_1$ : $T = 0.608s, r = 164m, v_{max} = 314.16m/s, a_{max} = 3620m/s^2, d = -3.8258Ns/m, c = -241.2879N, b = 1031.4N/A$. The structure of the optimal solution to Problem 2.1 could be identified systematically as in [22]. Without loss of generality, we assume the knowledge of the structure of the optimal solution to Problem 2.1 with $PDATA_1$. That is: the optimal trajectory consists of 7 orderly segments

1) positive acceleration constrained arc;
2) unconstrained positive control arc;
3) velocity constrained arc;
4) unconstrained positive control arc;
5) zero control arc;
6) unconstrained negative control arc;
7) negative acceleration constrained positive control arc.

The corresponding MBVP is written as follows

### 4.1 Parameters and Boundary Conditions

We define parameters for the $k$ segment

$$(\beta_{k1}, \beta_{k2}) = \begin{cases} (\lambda_0, t_1)^T, \ k=1,\\ (x(t_k), \lambda(t_k), t_{k+1})^T, \ 2 \le k \le 6,\\ (x(t_6), \lambda(t_6))^T, \ k=7. \end{cases} \tag{15}$$

$$\begin{aligned}
\beta_1 &= (\lambda_0, x(t_1), \lambda(t_1), \ldots, x(t_6), \lambda(t_6)),\\
\beta_2 &= (t_1, \ldots, t_6) \in \mathbb{R}^6.
\end{aligned}$$

We also have the boundary conditions for the $k$th segment, $1 \le k \le 7$,

$$\begin{aligned}
&BC_1: \text{ on state and costate}\\
&\phi_x(x(t_{k-1}), \lambda(t_{k-1}), t_{k-1}, t_k) = x(t_k^+), \ 1 \le k \le 6\\
&\phi_\lambda(x(t_{k-1}), \lambda(t_{k-1}), t_{k-1}, t_k) = \lambda(t_k^+), \ k \ne 2, 1 \le k \le 6\\
&x_2(t_2^-) = v_{max},\\
&\lambda_1(t_2^-) = \lambda_1(t_2^+),\\
&\phi_x(x(t_6^+), \lambda(t_6^+), t_6, T) = x_f, \ k=7\\
&BC_2: \text{ on switch times}\\
&u(t_k^-) = u(t_k^+), \quad 1 \le k \le 6,
\end{aligned}$$

where $x(t_0) = x_0, \lambda(t_0) = \lambda_0$. The MBVP has 30 parameters and BCs.

**Remark 4.1** *The number of parameters can be further reduced by removing the costate at the entry of constrained segments. Thus the parameter sets can be written as*

$$(\beta_{k1}, \beta_{k2}) = \begin{cases} t_1, \ k=1,\\ (\lambda(t_1), t_2), \ k=2,\\ (x(t_2), t_3), \ k=3,\\ (x(t_3), \lambda(t_3), t_4), \ k=4,\\ (x(t_4), t_5)^T, \ k=5,\\ (x(t_5), \lambda(t_5), t_6)^T, \ k=6. \end{cases}$$

*The corresponding BCs can be similarly simplified.*

### 4.2 Sensitivity Equations

Next we derive the sensitivity equations defining

$$\frac{\partial x}{\partial \beta_1}, \quad \frac{\partial \lambda}{\partial \beta_1}, \quad \frac{\partial x}{\partial \beta_2}, \quad \frac{\partial \lambda}{\partial \beta_2}.$$

The sensitivity equations corresponding to (15) are summarized as follows

$$\frac{d}{dt}\frac{\partial x_a}{\partial \beta_1} = A_a \frac{\partial x_a}{\partial \beta_1} \tag{16}$$

where $x_a = [x, \lambda]$, and

$$\begin{aligned}
A_a &= \begin{bmatrix} \alpha_1 I_{48} & \alpha_2 I_{48} \\ \alpha_3 I_{48} & \alpha_4 I_{48} \end{bmatrix},\\
\frac{\partial x_a}{\partial \beta_1} &= \begin{bmatrix} \frac{\partial x}{\partial \beta_1} & \frac{\partial \lambda}{\partial \beta_1} \end{bmatrix} \in \mathbb{R}^{96}.
\end{aligned}$$

From (1), we have

$$f_x = \begin{bmatrix} 0 & 1 \\ 0 & d \end{bmatrix},$$

$$f_u = [0, b]^T,$$

$$u_x = \begin{cases} [0, -\frac{d}{b}], & |\dot{x}_2| = a_{max} \text{ or } x_2 = v_{max}, \\ [0, -\frac{K_\tau}{2R}], & u = u_1 \text{ or } u = u_2, \\ [0, 0], & \text{otherwise.} \end{cases}$$

$$u_\lambda = \begin{cases} [0, -\frac{b}{2R}], & u = u_1 \text{ or } u = u_2, \\ [0, 0], & \text{otherwise,} \end{cases}$$

$$H_{x\lambda}^T = A^T,$$

$$H_{xu}^T = \begin{cases} [0, K_\tau]^T, & \text{if } u \neq 0, \\ [0, 0]^T, & u = 0, \end{cases}$$

$$H_{xx}^T = 0,$$

$$H_{x\mu}^T = \begin{cases} \begin{bmatrix} 0 & 0 \\ d & -d \end{bmatrix}, & u \neq 0, \\ 0, & u = 0, \end{cases}$$

$$H_{x\nu}^T = H_{x\mu}^T,$$

$$\mu_x = \begin{cases} \begin{bmatrix} 0 & \frac{K_\tau}{-b} \\ 0 & 0 \end{bmatrix}, & x_2 = v_{max}, \\ \begin{bmatrix} 0 & 0 \\ 0 & \frac{K_\tau}{-b} \end{bmatrix}, & x_2 = 0, \\ 0, & \text{otherwise,} \end{cases}$$

$$\mu_\lambda = \begin{cases} \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}, & x_2 = v_{max}, \\ \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, & x_2 = 0, \\ 0, & \text{otherwise,} \end{cases}$$

$$\mu_u = \begin{cases} \begin{bmatrix} \frac{2R}{-b} & 0 \end{bmatrix}^T, & x_2 = v_{max}, \\ \begin{bmatrix} 0 & \frac{2R}{-b} \end{bmatrix}^T, & x_2 = 0, \\ 0, & \text{otherwise,} \end{cases}$$

$$\nu_x = \begin{cases} \begin{bmatrix} 0 & \frac{K_\tau}{-b} \\ 0 & 0 \end{bmatrix}, & \text{if } \dot{x}_2 = a_{max}, \\ \begin{bmatrix} 0 & 0 \\ 0 & \frac{K_\tau}{b} \end{bmatrix}, & \text{if } \dot{x}_2 = -a_{max}, \\ 0, & \text{otherwise,} \end{cases}$$

$$\nu_\lambda = \begin{cases} \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}, & \dot{x}_2 = a_{max}, \\ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, & \dot{x}_2 = -a_{max}, \\ 0, & \text{otherwise,} \end{cases}$$

$$\nu_u = \begin{cases} \begin{bmatrix} \frac{2R}{-b} & 0 \end{bmatrix}^T, & \text{if } \dot{x}_2 = a_{max}, \\ \begin{bmatrix} 0 & \frac{2R}{b} \end{bmatrix}^T, & \text{if } \dot{x}_2 = -a_{max}, \\ 0, & \text{otherwise.} \end{cases}$$

Thus we obtain $\alpha_1, \ldots, \alpha_4$. Following the procedure in Section 3, the sensitivity of $BC_2$ w.r.t. switch times can be readily computed.

### 4.3 Update Parameters

According to Section 3, we first use the residue $F_1$ and the gradient $\partial F_1/\partial\beta_1$ to update parameter set $\beta_1$ s.t. $F_1(\beta_1^*) = 0$, then use the residue $F_2$ and the gradient $\partial F_2/\partial\beta_2$ to update parameter set $\beta_2$ s.t. $F_2(\beta_2^*) = 0$.

### 4.4 Simulation

We code the algorithm in Matlab 7.0 to solve the MBVP associated with problem data $PDATA_1$. The solver performs 9 iterations to update the parameter set $\beta_2$. The benchmark mesh refinement direct transcription method in Section 2 is started from a uniform mesh containing 20 mesh points, and stopped after 4 iterations with a final mesh of 80 grid points. The final mesh has a finest local resolution equivalent to a uniform mesh containing 300 grid points. The NLP problem 2.2 is solved using the Matlab function fmincon with the SQP method. The control iteration and mesh refinement iterations of the benchmark method are shown in Figure 4 and Figure 5, respectively. The total computation time of the benchmark method is 5.4 seconds. The proposed algorithm finds the solution of the MBVP within 1s, which is more than 5 times faster than benchmark method. The optimal control trajectory is given in Figure 2. The iteration of updating $\beta_2$ is shown in Figure 3, where the iteration is illustrated by the control trajectory during each iteration of $\beta_2$.

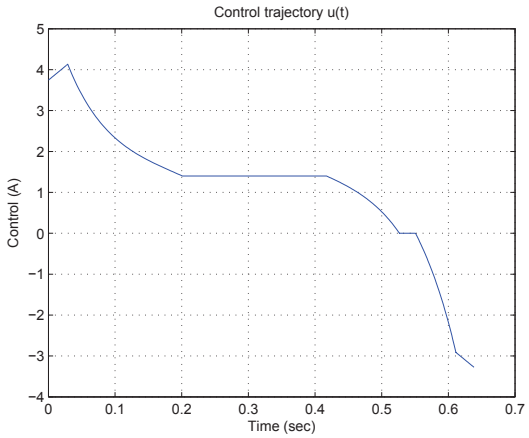


Fig. 2: Control trajectory

## 5 Conclusions

This note presents a method to solve a class of multi-point boundary value problems (MBVPs). The proposed method is a tradeoff between multiple and single shooting. A satisfactory computation speed and performance is achieved by requiring much less parameters, and the introduction of a two-step algorithm. Different from the existing one-step algorithm, the two-step algorithm improves the convergence property by decoupling the interaction between two sets of parameters: state and costate, and switch times. Simulation shows that the proposed method can compute the solution of a class of MBVPs faster than various existing methods.

## References

[1] Q. Gong, W. Kang, and I. M. Ross, "A pseudospectral method for the optimal control of constrained feedback linearizable

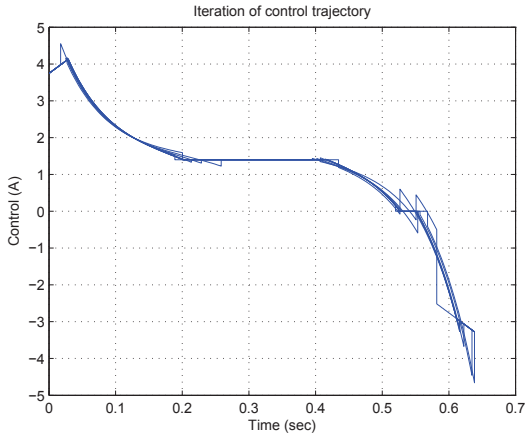


Fig. 3: Control trajectories over iterations

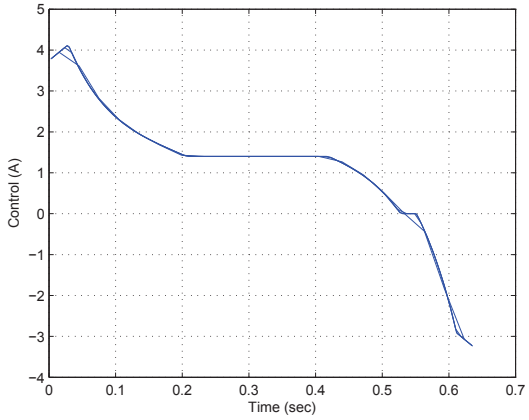


Fig. 4: Control iterations of the benchmark method

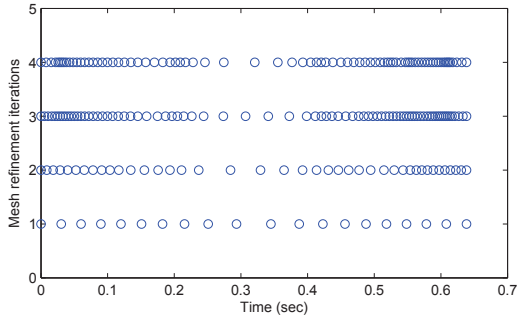


Fig. 5: Mesh refinement iterations of the benchmark method

systems," *IEEE Trans. Automat. Contr.*, vol. AC-51, no. 7, Jul. 2006.

[2] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, Sep. 1985.

[3] Z. Shiller and H.-H. Lu, "Computation of path constrained time optimal motions with dynamic singularities," *Trans. ASME, J. Dyn. Sys. Meas. Control*, vol. 114, pp. 34–40, Mar. 1992.

[4] K. G. Shin and N. D. Mckay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 6, pp. 531–541, Jun. 1985.

[5] ——, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Trans. Automat. Contr.*, vol. AC-31, no. 6, pp. 491–500, Jun. 1986.

[6] S. Singh and M. C. Leu, "Optimal trajectory generation for robotic manipulators using dynamics programming," *Trans. ASME, J. Dyn. Sys. Meas. Control*, vol. 109, pp. 88–96, Jun. 1987.

[7] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Automat. Contr.*, vol. AC-54, no. 10, pp. 2318–2327, Oct. 2009.

[8] G. Elnagar, M. A. Kazemi, and M. Razzaghi, "The pseudospectral legendre method for discretizing optimal control problems," *IEEE Trans. Automat. Contr.*, vol. AC-40, no. 10, pp. 1793–1796, Oct. 1995.

[9] F. Fahroo and I. M. Ross, "Direct trajectory optimization by a chebyshev pseudospectral method," in *Proc. 2000 ACC*, Chicago, IL, 2000, pp. 3860–3864.

[10] J. A. B. White, "Numberical solution of two-point boundary value problems," Ph.D. dissertation, Caltech, 1974.

[11] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solutions of Boundary Value Problems for Ordinary Differential Equation*. SIAM, 1995.

[12] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *Proceedings of the 9th IFAC World Congress*, Budapest, Hungary, 1984, pp. 242–247.

[13] G. Fraser-Andrews, "A multiple shooting technique for optimal control," *Journal of Optimization Theory and Applications*, vol. 102, no. 2, pp. 299–313, Aug. 1999.

[14] H. Pasic, "Multipoint boundary value solution of two point boundary value problem," *Journal of Optimization Theory and Applications*, vol. 100, no. 2, pp. 397–416, Feb. 1999.

[15] H. J. Pesch, "Real-time computation of feedback controls for constrained optimal control problems part 1: Neighboring extremals," in *Optimal Control Applications & Methods*, 1989.

[16] ——, "Real-time computation of feedback controls for constrained optimal control problems - part 2: A correction method based on multiple shooting," in *Optimal Control Applications & Methods*, 1989, pp. 147–171.

[17] ——, "A practical guide to the solution of real-life optimal control problems," *Control and Cybernetics*, vol. 23, pp. 7–60, 1994.

[18] T. Ohtsuka, "A continuation/GMRES method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563–574, Apr. 2004.

[19] J. T. Betts, *Practical Methods for Optimal Control using Nonlinear Programming*. PA: SIAM, 2001.

[20] Y. Zhao and P. Tsiotras, "Density functions for mesh refinement in numerical optimal control," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 1, pp. 271–277, Jan.-Feb. 2011.

[21] Y. Wang, K. Ueda, and S. A. Bortoff, "On the optimal trajectory generation for servomotors: a Hamiltonian approach," in *Proc. 51th CDC*, Maui, HI, 2012, pp. 7620–7625.

[22] ——, "A Hamiltonian approach to compute an energy efficient trajectory for a servomotor system," *Automatica*, 2012, revised.

[23] D. H. Jacobson, M. M. Lele, and J. L. Speyer, "New necessary conditions of optimality for control problems with state-variable inequality constraints," *Journal of Mathematical Analysis and Application*, vol. 35, pp. 255–284, 1971.