Proceedings of the
46th IEEE Conference on Decision and Control
New Orleans, LA, USA, Dec. 12-14, 2007

MPDopt: A versatile toolbox for adjoint-based model predictive control of smooth and switched nonlinear dynamic systems

Sean Summers and Thomas R. Bewley
Flow Control and Coordinated Robotics Labs, UC San Diego

***Abstract*— Over the years, adjoint-based Model Predictive Control (MPC) has proven to be a viable and effective tool for both offline and online optimization of control sequences for dynamic systems governed by differentiable nonlinear equations. This paper extends this versatile method to a class of nonsmooth systems in which the dynamical equation governing the physical system experiences sudden "switches" when the system changes operational modes. This extension involves only minimal alteration of the basic adjoint-based MPC algorithm. We then further improve this algorithm by introducing control trajectory smoothing via state augmentation. Finally, we present a new Matlab toolbox we have developed, MPDopt, that incorporates all of these developments while being user friendly, easily extensible, and capable of offline trajectory planning for wide range of smooth and switched systems of the present class.**

I. INTRODUCTION

The objective of Model Predictive Control (MPC) is the determination of an optimal sequence of control inputs $\mathbf{u}$ to a dynamic (possibly, nonlinear) system to minimize a user-defined (possibly, nonquadratic) cost function J, typically measuring both the state variable $\mathbf{x}$ and the control input $\mathbf{u}$ over a finite time interval $[0, T]$. As a mathematical optimization problem, as long as the dynamic system is differentiable, gradient-based approaches based on on steepest descent, conjugate gradient, or BFGS optimization methods are straightforward (see [2] and [5]). With such approaches, to compute the gradient of the cost function J with respect to the control inputs $\mathbf{u}$, a Lagrange multiplier λ (a.k.a. the adjoint state) times the equation governing the system may be appended to the cost function. By solving the state ODE forward in time and an appropriately-defined adjoint ODE backwards in time, the gradient of the cost function with respect to the control distribution $\mathbf{u}$ is revealed. Given such an efficient technique to compute the gradient, an iterative method (typically, steepest descent, conjugate gradient, or BFGS) may then be used to minimize J with respect to the sequence of control inputs $\mathbf{u}$ in a straightforward fashion.

A. The adjoint method for computing the gradient

We first derive the well-known adjoint method for computing the gradient of a constrained quadratic cost function, where the state constraint is the nonlinear ODE governing the system. Similar derivations for computing the gradient can be found in [1] and [3]. We first assume that the system of interest is differentiable and known and may be written in the form

$$\frac{d\mathbf{x}}{dt} = N(\mathbf{x}, \mathbf{u}) \quad 0 < t < T, \tag{1}$$

$$\mathbf{x} = \mathbf{x}_0 \qquad t = 0, \tag{2}$$

where $t = 0$ is the present time and

- $\mathbf{x}(t)$ is the state vector with $\mathbf{x}_0$ the known initial condition at $t = 0$
- $\mathbf{u}(t)$ is the control at time t
- $N(\mathbf{x}, \mathbf{u})$ is a known function, differentiable in $\mathbf{x}$ and $\mathbf{u}$, representing the dynamic system of interest.

We also define a cost function J (with weighted norm $|\mathbf{x}|_Q^2 \triangleq \mathbf{x}^T Q \mathbf{x}$) as

$$J = \frac{1}{2}\int_0^T \left[|\mathbf{x}|_{Q_\mathbf{x}}^2 + |\mathbf{u}|_{Q_\mathbf{u}}^2 \right] dt + \frac{1}{2} |\mathbf{x}(T)|_{Q_T}^2 , \tag{3}$$

where $t = T$ is the terminal time and

- $Q_\mathbf{x}$ is the penalty matrix on the state $\mathbf{x}$,
- $Q_\mathbf{u}$ is the penalty matrix on the control $\mathbf{u}$,
- Q_T is the penalty matrix on the terminal state $\mathbf{x}$(T).

By appending to (3) an adjoint state λ (a.k.a. Lagrangian multiplier) times the equation governing the system, (1), an augmented cost function is defined as

$$J = \frac{1}{2}\int_0^T \left[|\mathbf{x}|_{Q_\mathbf{x}}^2 + |\mathbf{u}|_{Q_\mathbf{u}}^2 \right] dt + \frac{1}{2} |\mathbf{x}(T)|_{Q_T}^2 - \int_0^T \lambda^T (\dot{\mathbf{x}} - N(\mathbf{x}, \mathbf{u})) dt. \tag{4}$$

We first set $\nabla_\lambda J = 0$, thereby enforcing the equality constraint (1). We also set $\nabla_\mathbf{x} J = 0$, thereby obtaining an auxiliary "adjoint" ODE

$$\begin{aligned} \dot{\lambda} &= -A^T \lambda - Q_\mathbf{x} \mathbf{x} \quad T > t > 0, \\ \lambda &= Q_T(T)\mathbf{x}(T) \quad t = T, \end{aligned} \tag{5}$$

where $A = \frac{\partial (N(\mathbf{x},\mathbf{u}))}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}(t),\mathbf{u}=\mathbf{u}(t)}$. Finally, the resultant gradient $\nabla_\mathbf{u} J$ is given by

$$\nabla_\mathbf{u} J = B^T \lambda + Q_\mathbf{u} \mathbf{u}, \tag{6}$$

where $B = \frac{\partial (N(\mathbf{x},\mathbf{u}))}{\partial \mathbf{u}}|_{\mathbf{x}=\mathbf{x}(t),\mathbf{u}=\mathbf{u}(t)}$. Note that, with this method, $\nabla_\mathbf{u} J$ is a simple function of λ, which is defined in (5) via a march from $t = T$ to $t = 0$ and which is itself a function of $\mathbf{x}$, which is defined in (1) via a march from $t = 0$ to $t = T$.

This work was supported by Los Alamos National Laboratory

Sean Summers, ssummers@ucsd.edu, and Professor Thomas R. Bewley, bewley@ucsd.edu, are affiliated with the Flow Control and Coordinated Robotics Labs at the Center for Control Systems and Dynamics at the Mechanical and Aerospace Engineering Department of the University of California, San Diego.

B. Nonsmooth optimization

Note that the application of standard gradient-based optimization methods to nonsmooth, or switched, systems commonly result in the optimization algorithm becoming stuck in a "kink" in the optimization surface. A kink is defined here as any point on the optimization surface at which multiple subgradients exist, and neither subgradient results in a viable direction of descent of the cost function. The theory of nonsmooth optimization is devoted exactly to such objective functions which are not continuously differentiable. Methods for solving such nonsmooth problems via subgradient methods, cutting plane methods, etc., have been introduced and provide an alternative to smooth optimization methods. For more in depth information on nonsmooth optimization analysis and algorithms, see [4]. Unfortunately, general nonsmooth optimization methods are considerably more cumbersome than the standard smooth optimization methods outlined above. Thus, in the present work, rather than resorting to general nonsmooth optimization methods, we instead convert our switched system into a smooth optimization problem.

II. Gradient computation in switched systems

In this section, we extend the adjoint based method of gradient computation presented in I-A to switched systems. That is, we will consider systems with one or more discrete switch points at which the continuous-time state equations governing the system change instantaneously. As mentioned above, this discontinuity in the state equation results in multiple subgradients of the cost function at specific points in the optimization surface. When faced with multiple subgradients, the adjoint based optimization algorithm may become stuck in a kink and fail to converge, even though a local optimal point has not yet been reached. Therefore, a new approach must be taken by either

(a) adjusting the problem definition such that J becomes well-conditioned [2], and therefore a smooth gradient based optimization algorithm may be applied, or
(b) using a nonsmooth optimization algorithm, such as a bundle method [4], to explicitly find a solution for the non-differentiable cost function.

Approach (b) may prove to be necessary for harder problems. Note that nonsmooth optimization approaches are significantly more cumbersome (and slow to converge) than optimization algorithms designed specifically for smooth optimization surfaces. For the present class of problems, however, approach (a) is shown viable through the use of multiple terminal constraints applied to the end of each phase. If such constraints are introduced correctly into the cost function (3), this makes the optimization problem smooth, well-conditioned, and amenable to standard gradient-based optimization methods, as detailed below.

For simplicity, we will initially assume that the system of interest is a single switch system with known switch time nominally set to $t = T_1$, terminal time nominally set to $t = T_2$ (see V-B for generalization), and nonlinear state equations of the form

$$\frac{d\mathbf{x}}{dt} = N_1(\mathbf{x}, \mathbf{u}) \quad 0 < t < T_1,, \tag{7}$$

$$\frac{d\mathbf{x}}{dt} = N_2(\mathbf{x}, \mathbf{u}) \quad T_1 < t < T_2,, \tag{8}$$

$$\mathbf{x} = \mathbf{x}_0 \qquad t = 0,. \tag{9}$$

1) Optimization of two decoupled phases: In (3) we introduced a weighted terminal constraint into the cost function. Defining the terminal weighting matrix Q_T appropriately and applying the iterative optimization algorithm described previously, this essentially forces the states at the terminal time $t = T$ to converge to a result very close to $\mathbf{x}(T) = 0$. By redefining the weighted terminal constraint as a general weighted constraint function at some time t,

$$|\mathbf{x}(t)|^2_Q \Rightarrow |f(\mathbf{x}(t), \mathbf{u}(t))|^2_Q, \tag{10}$$

we can split the optimization into two independent trajectory planning segments. Thus, our first attempt at obtaining a trajectory plan for the single switch system is to create two separate cost functions [the first defining the trajectory while governed by the equations of motion (7), and the second while governed by (8)] such that

$$J_1 = \frac{1}{2}\int_0^{T_1} \left[|\mathbf{x}|^2_{Q_{\mathbf{x}_1}} + |\mathbf{u}|^2_{Q_{\mathbf{u}_1}}\right] dt + \frac{1}{2}|f_{T_1}|^2_{Q_{T_1}}, \tag{11}$$

$$J_2 = \frac{1}{2}\int_{T_1}^{T_2} \left[|\mathbf{x}|^2_{Q_{\mathbf{x}_2}} + |\mathbf{u}|^2_{Q_{\mathbf{u}_2}}\right] dt + \frac{1}{2}|f_{T_2}|^2_{Q_{T_2}}, \tag{12}$$

with the terminal constraints defined as functions of both state and control variables

$$f_{T_1} = f_{T_1}(\mathbf{x}(T_1), \mathbf{u}(T_1)), \tag{13}$$

$$f_{T_2} = f_{T_2}(\mathbf{x}(T_2), \mathbf{u}(T_2)). \tag{14}$$

By separating the original optimization problem into two distinct, decoupled phases, we now have two smooth optimization problems that are easily solved. For each individual phase, we may compute the gradient as in section I-A and iterate via standard gradient-based optimization techniques until convergence. Unfortunately, there is an important flaw with this approach. By isolating each phase, the final position of the system during the first phase is unrelated to the initial position of the system during the second phase, and the two phases fail to match up with each other. This problem is eliminated in the following section.

2) Optimization of both phases simultaneously, with point constraints: We now address the shortcomings of the previous approach by summing the two individual cost functions (11) and (12) to create a single cost function,

$$J = \frac{1}{2}\int_0^{T_1} \left[|\mathbf{x}|^2_{Q_{\mathbf{x}_1}} + |\mathbf{u}|^2_{Q_{\mathbf{u}_1}}\right] dt + \frac{1}{2}|f_{T_1}|^2_{Q_{T_1}} + \frac{1}{2}\int_{T_1}^{T_2} \left[|\mathbf{x}|^2_{Q_{\mathbf{x}_2}} + |\mathbf{u}|^2_{Q_{\mathbf{u}_2}}\right] dt + \frac{1}{2}|f_{T_2}|^2_{Q_{T_2}}. \tag{15}$$

We now have multiple constraints, which we will refer to as "point constraints", incorporated in a single cost function. Note the importance of choosing an appropriate point

constraint at the switch time T_1. If an unsatisfactory function is chosen, the dynamic optimization will not be well-conditioned and thus a good solution will not be achieved. With the cost function defined in (15), and following the adjoint-based method presented in I-A, we obtain the nonsmooth adjoint system

$$\dot{\lambda} = \begin{cases} -A_1^T\lambda - Q_{\mathbf{x}_1}\mathbf{x} & T_1 > t > 0, \\ -A_2^T\lambda - Q_{\mathbf{x}_2}\mathbf{x} & T_2 > t > T_1, \end{cases} \quad (16)$$

$$\lambda = \begin{cases} \frac{1}{2}\nabla_{\mathbf{x}}\left|f_{T_1}\right|^2_{Q_{T_1}} + \lambda(T_1) & t = T_1, \\ \frac{1}{2}\nabla_{\mathbf{x}}\left|f_{T_2}\right|^2_{Q_{T_2}} & t = T_2, \end{cases} \quad (17)$$

where A_1 and A_2 are again Jacobian matrices with respect to $\mathbf{x}$ evaluated at $\mathbf{x} = \mathbf{x}(t)$ and $\mathbf{u} = \mathbf{u}(t)$. Further, we obtain a single, applicable gradient

$$\nabla_{\mathbf{u}} J = \begin{cases} Q_{\mathbf{u}_1}\mathbf{u} + B_1^T\lambda & 0 < t < T_1, \\ Q_{\mathbf{u}_1}\mathbf{u} + B_1^T\lambda + \frac{1}{2}\nabla_{\mathbf{u}}\left|f_{T_1}\right|^2_{Q_{T_1}} & t = T_1, \\ Q_{\mathbf{u}_2}\mathbf{u} + B_2^T\lambda & T_1 < t < T_2, \\ Q_{\mathbf{u}_2}\mathbf{u} + B_2^T\lambda + \frac{1}{2}\nabla_{\mathbf{u}}\left|f_{T_2}\right|^2_{Q_{T_2}} & t = T_2, \end{cases} \quad (18)$$

where B_1 and B_2 are again Jacobian matrices with respect to $\mathbf{u}$ evaluated at $\mathbf{x} = \mathbf{x}(t)$ and $\mathbf{u} = \mathbf{u}(t)$.

3) Example: Application to Multi-functional Ground Vehicle Trajectory Planning - Rover Uprighting Transition: In designing a multi-functional robotic ground vehicle, the ability to switch from one configuration to another is valuable. As such transitions often involve nondifferentiable maneuvers, a traditional smooth approach to offline trajectory optimization is not effective, and a nonsmooth approach is required. To illustrate this claim, we introduce the nonsmooth control problem depicting the uprighting of a three-wheeled rover as illustrated in Figure 1

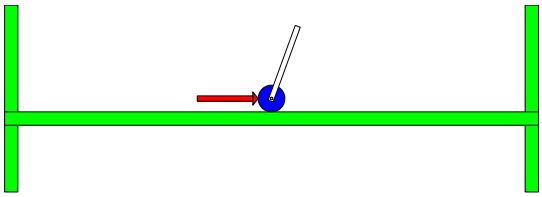


Fig. 1. Rover Upright

We assume that at $t = 0$ the rover has three wheels on the ground (two drive wheels and one follower wheel mounted at the top of the arm), with the arm, or pendulum, at an angle of $\theta = 5\pi/9$ radians. Our performance goal is to upright the vehicle with minimal terminal deviation from the starting horizontal position within the time interval $t = 0 : T$, with the time interval nominally set to $T = 3$ seconds. The terminal upright position is defined as $\theta = 0$ with the arm of the rover positioned vertically with the follower wheel at the peak. In deriving the state equations of the given system, it is obvious that this maneuver falls into the category of a switched, or hybrid, system as there is a discrete point at which the continuous time equations of motion governing the system change instantaneously. Thus, we have motivation to treat this as a nonsmooth system and approach the gradient derivation as presented in section II. The equation of motion for the vehicle in the horizontal roving mode (where the follower wheel at the end of the vehicle arm is in contact with the ground) is

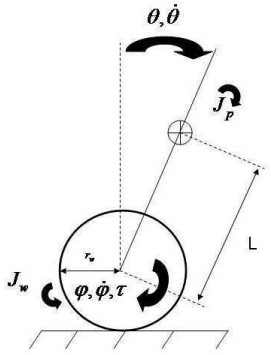


Fig. 2. Free Body Diagram of the Robotic Rover

$$(J_w + (M_p + M_w)r_w^2)\ddot{\phi} = -\frac{k^2}{R}\dot{\phi} + \frac{k}{R}\tau. \quad (19)$$

The equation of motion for the the vehicle during the second phase of the nonsmooth trajectory (as soon as the follower wheel lifts off the ground) is

$$(m_pL^2 + J_p)\ddot{\theta} + M_p r_w L\ddot{\phi}\cos\theta = M_p gL\sin\theta - \frac{k}{R}(\tau + k(\dot{\theta} - \dot{\phi})), \quad (20)$$

$$M_p r_w L\ddot{\theta}\cos\theta + (J_w + (M_p + M_w)r_w^2)\ddot{\phi} = M_p r_w L\dot{\theta}^2\sin\theta + \frac{k}{R}(\tau + k(\dot{\theta} - \dot{\phi})). \quad (21)$$

The parameters identified in the system ODE's (M_w,M_p,J_w,J_p,r_w,L,k,R,g) are assumed to be known and constant in time. The control and state variables of the given ODE's (see Figure 2) are τ (Voltage Applied to the Motors), ϕ (Angular Displacement of the Wheels (rad)), $\dot{\phi}$ (Angular Velocity of the Wheels (rad/s)), θ (Angular Displacement of Pendulum (rad))), and $\dot{\theta}$ (Angular Velocity of Pendulum (rad/s)). Putting the state and control variables into generalized form, $\mathbf{x} = \left[\theta, \dot{\theta}, \phi, \dot{\phi}\right]^T$ and $u = \tau$, we re-express (19), (20), and (21) in the nonlinear state-space form

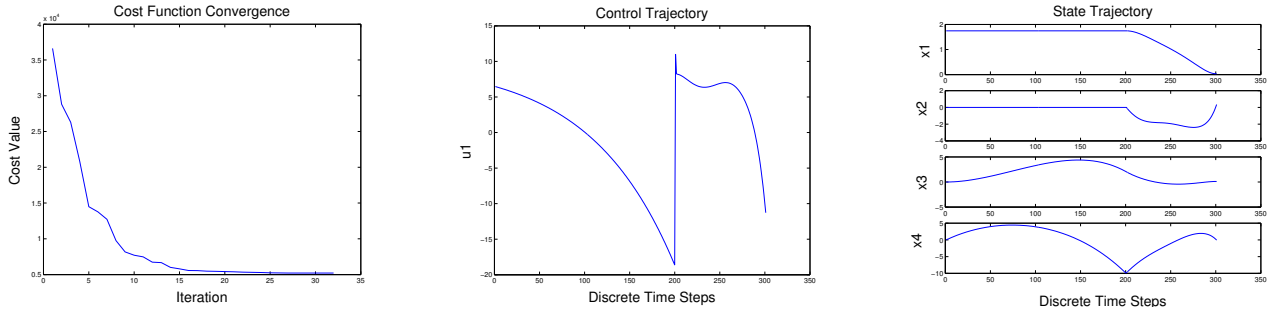
$$E_1(\mathbf{x})\frac{d\mathbf{x}}{dt} = N_1(\mathbf{x}, \mathbf{u}) \quad 0 < t < T_1, \quad (22)$$

$$E_2(\mathbf{x})\frac{d\mathbf{x}}{dt} = N_2(\mathbf{x}, \mathbf{u}) \quad T_1 < t < T_2. \quad (23)$$

Given that for the particular system $E_1(\mathbf{x})$ and $E_2(\mathbf{x})$ are both square and nonsingular matrices, we may express (22) and (23) in the nonlinear state-space form given in (7) and (8) with a switch at time $t = T_1$ such that

$$\frac{d\mathbf{x}}{dt} = \hat{N}_1(\mathbf{x}, \mathbf{u}) \quad 0 < t < T_1, \quad (24)$$

$$\frac{d\mathbf{x}}{dt} = \hat{N}_2(\mathbf{x}, \mathbf{u}) \quad T_1 < t < T_2, \quad (25)$$



(a) Trajectory Cost Value versus Iterations

(b) Control Trajectory versus Discrete Time Steps

(c) State Trajectory versus Discrete Time Steps

Fig. 3. Simulation Results for Nonsmooth Rover Upright Transition

where $\hat{N}_1(\mathbf{x},\mathbf{u}) = E_1^{-1}(\mathbf{x})N_1(\mathbf{x},\mathbf{u})$ and $\hat{N}_2(\mathbf{x},\mathbf{u}) = E_2^{-1}(\mathbf{x})N_2(\mathbf{x},\mathbf{u})$. The initial condition of the state variables at $t = 0$ is set to

$$\mathbf{x}_0 = \left[\begin{array}{cccc} \frac{5\pi}{9} & 0 & 0 & 0 \end{array} \right]^T,$$

and the switch time is nominally chosen as $T_1 = 2$ seconds. With the trajectory cost defined as (15), we choose appropriate values for the penalty matrices and define the point constraints

$$\begin{aligned} f_{T_1}(\mathbf{x}(T_1),\mathbf{u}(T_1)) &= \Gamma_{Total}(T_1), \\ &\cong \dot{x}_2(T_1), \\ f_{T_2}(\mathbf{x}(T_2),\mathbf{u}(T_2)) &= \mathbf{x}(T_2), \end{aligned}$$

where Γ_{Total} is the total torque applied to the arm of the vehicle assuming the normal force applied to follower wheel is zero (i.e., assuming it is not in contact with the ground). By attributing a very large penalty to this constraint at the predetermined switch time $t = T_1$, we ensure that the system passes through a state at $t = T_1$ where the overall torque applied to the arm is set to zero (i.e. the angular acceleration of the pendulum is zero at the switch), and therefore a smooth transition from the horizontal rover mode to the upright configuration may be accomplished. Also note the large dynamic penalty we place on the angle of the pendulum during the second phase of the uprighting transition, which in effect ensures that the arm accelerates in the *upward* direction at the start of the second phase. This keeps the system consistent with the fact that the arm may not move through the ground at any point during the trajectory. Using (18) as the gradient for the nonsmooth system maneuver, we iteratively update the discretized control sequence such that (15) is minimized. With the given parameter values and weighting matrices, we obtain a solution to the nonsmooth trajectory planning problem which is presented in terms of Cost Convergence, State Trajectory, and Control Trajectory in Figure 3.

III. Control Trajectory Smoothing via State Augmentation

In example II-.3 we produced a satisfactory result for the nonsmooth uprighting maneuver of a three wheeled rover. However, the control trajectory obtained via the Adjoint Based Dynamic Optimization algorithm for nonsmooth systems is less than ideal. An obvious flaw in this technique can be seen in the discontinuous tendency of the control trajectory solution in Figure 3(b). In a real world application, a control discontinuity this large would be very difficult to implement. Another apparent flaw in the original solution of the nonsmooth planned trajectory is the inability of the algorithm to 'force' the control effort at the initial/final time step to zero. Thus the motivation for a smooth, continuous control trajectory with initial and terminal constraints is presented and solved via state augmentation.

We begin by identifying the known, single switch system, as specified in (7), (8), (9), with chosen switch time T_1. We introduce an augmented state vector

$$\bar{\mathbf{x}} = \left[\begin{array}{c} \mathbf{x} \\ \mathbf{u} \end{array} \right], \tag{26}$$

and a control velocity term

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{v}. \tag{27}$$

The control velocity term becomes the optimized (forcing) variable in the augmented system state equation(s)

$$\frac{d\bar{\mathbf{x}}}{dt} = \bar{N}_1(\bar{\mathbf{x}},\mathbf{v}) \quad 0 < t < T_1, \tag{28}$$

$$\frac{d\bar{\mathbf{x}}}{dt} = \bar{N}_2(\bar{\mathbf{x}},\mathbf{v}) \quad T_1 < t < T_2, \tag{29}$$

$$\bar{\mathbf{x}} = \bar{\mathbf{x}}_0 \quad t = 0, \tag{30}$$

where

$$\bar{N}_1 = \left[\begin{array}{c} N_1(\bar{\mathbf{x}}) \\ \mathbf{v} \end{array} \right] \text{ and } N_2 = \left[\begin{array}{c} \hat{N}_2(\bar{\mathbf{x}}) \\ \mathbf{v} \end{array} \right].$$

We update the Trajectory Cost (15) to include the new control velocity term

$$\begin{aligned} J &= \frac{1}{2}\int_0^{T_1} \left[|\bar{\mathbf{x}}|^2_{Q_{\bar{\mathbf{x}}_1}} + |\mathbf{v}|^2_{Q_{\mathbf{v}_1}} \right] dt + \frac{1}{2}|f_{T_1}|^2_{Q_{T_1}} \\ &\quad + \frac{1}{2}\int_{T_1}^{T_2} \left[|\bar{\mathbf{x}}|^2_{Q_{\bar{\mathbf{x}}_2}} + |\mathbf{v}|^2_{Q_{\mathbf{v}_2}} \right] dt + \frac{1}{2}|f_{T_2}|^2_{Q_{T_2}}, \end{aligned} \tag{31}$$

where the point constraint functions are defined

$$f_{T_1} = f_{T_1}(\bar{\mathbf{x}}(T_1)), \quad (32)$$
$$f_{T_2} = f_{T_2}(\bar{\mathbf{x}}(T_2)). \quad (33)$$

The dynamic augmented state penalty matrices are redefined

$$Q_{\bar{\mathbf{x}}_1} = \begin{bmatrix} Q_{\mathbf{x}_1} & 0 \\ 0 & Q_{\mathbf{u}_1} \end{bmatrix} \text{ and } Q_{\bar{\mathbf{x}}_2} = \begin{bmatrix} Q_{\mathbf{x}_2} & 0 \\ 0 & Q_{\mathbf{u}_2} \end{bmatrix}, \quad (34)$$

and

- Q_{T_1} and Q_{T_2} are the point and terminal constraint optimization penalty matrices
- $Q_{\mathbf{v}_1}$ and $Q_{\mathbf{v}_2}$ are the optimization penalty matrices on the dynamic control velocity term $\mathbf{v}$

With this new configuration for the dynamic adjoint based optimization, we can now effectively apply start/end time constraints to the control variables, as well as restrict the speed at which the control evolves over time. We are capable of eliminating the control discontinuity at the initial time step by initializing the control vector to zero at the start of the planned trajectory. The terminal constraint may be defined such that the control effort at the end of the planned trajectory is pushed to zero, resulting in full control effort available to stabilize the system in the presence of deterministic or stochastic disturbances. In addition, for hybrid switched systems with known (desired) switch time, the state augmentation smooths the control trajectory such that undesirable discontinuities do not occur. Note that discontinuities may occur in the control velocity trajectory, however this does not adversely effect the implementable control trajectory.

4) Example: Application to Multi-functional Ground Vehicle Trajectory Planning - Rover Upright with Control Trajectory Smoothing: Given equivalent equations of motion and parameter values for the single switch Rover Upright problem as solved in Example II-.3, we wish to apply control trajectory smoothing via state augmentation as presented in Section III such that an improved control trajectory is obtained.

Following the approach of Section III, we augment the state vector and introduce a new control velocity term v. We redefine the system of ODE's in the form of (28) and (29), and set the initial condition of the augmented state to

$$\bar{\mathbf{x}}_0 = \begin{bmatrix} \frac{5\pi}{9} & 0 & 0 & 0 & 0 \end{bmatrix}^T.$$

Note that with the state augmentation, we have the ability to specify an initial condition for the control variable, in this case $u_0 = 0$. The trajectory cost is defined as equation (31) with the penalty matrices redefined and appropriately chosen, and the point constraints specified as

$$\begin{aligned} f_{T_1}(\bar{\mathbf{x}}(T_1)) &= \Gamma_{Total}(T_1), \\ &\cong \dot{x}_2(T_1), \\ f_{T_2}(\bar{\mathbf{x}}(T_2)) &= \bar{\mathbf{x}}(T_2). \end{aligned}$$

Note that the point constraint function for switch time T_1 remains as defined in example II-.3, and the point constraint at the terminal time T_2 has been updated where $u(T_2)$ is appended to the initial point constraint such that the control at the terminal time T_2 finalizes at zero. The solution presented in Figure 4 shows vast improvement over the prior result, especially with respect to the control trajectory 4(b).

IV. MPDOPT

MPDopt is a standalone MatLab package for Nonlinear Model Predictive Control offline trajectory planning. The package is fully automated and has the ability to obtain trajectory solutions for highly nonlinear systems, including both smooth and nonsmooth classes. The program obtains system information via user template which it uses to automatically generate files for the dynamic optimization. The following outlines the basic functionality of the standalone package:

A. MPDopt: User Defined Input

Augmented State ($t = T_0$)	$\bar{\mathbf{x}} = \bar{\mathbf{x}}_0 = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{u}_0 \end{bmatrix}$
Number of States/Controls	$\#\mathbf{x}/\#\mathbf{u}$
Number of Phases	P
Nonlinear ODE	$\{N_1(\mathbf{x},\mathbf{u}), ..., N_P(\mathbf{x},\mathbf{u})\}$
Point Constraint Functions	$\{f_{T_1}(\bar{\mathbf{x}}_{T_1}), ..., f_{T_P}(\bar{\mathbf{x}}_{T_P})\}$
Cost Penalty Matrices	$\{Q_{*1}, ..., Q_{*P}\}$
Time Window (Per Phase)	$\{T_1, ..., T_P\}$
Discretizations (Per Phase)	$\{N_1, ..., N_P\}$
Maximum Iterations	$MaxIter$
Residual Tolerance	tol

Given the single phase, or switched system, characteristics above, MPDopt automatically generates the required MatLab script files for the generalized adjoint based optimization. Note that the control trajectory smoothing via state augmentation is internal to the program such that the user may specify the system state matrices in terms of (7) and (8) rather than (28) and (29).

B. MPDopt: Generalized Algorithm

for k=1:MaxIter

• March the nonsmooth augmented state ODE forward in time from $t = T_0 : T_P$ saving the augmented state values at each discrete time step, where the generalized state equation can be expressed as

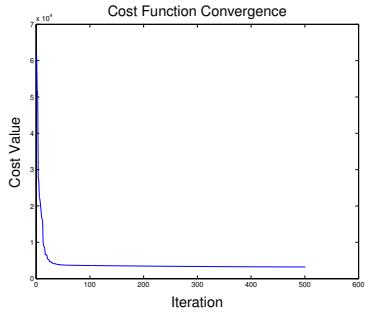
$$\begin{aligned} &\tfrac{d\bar{\mathbf{x}}}{dt} = \bar{N}_i(\bar{\mathbf{x}}, \mathbf{v}) \quad T_{i-1} < t < T_i \\ &i \in \{1, ..., P\} \\ &\bar{\mathbf{x}} = \bar{\mathbf{x}}_0 \qquad t = T_0 \end{aligned}$$

• Compute the trajectory cost value $J_k(\bar{\mathbf{x}}, \mathbf{v})$, expressed in the generalized form

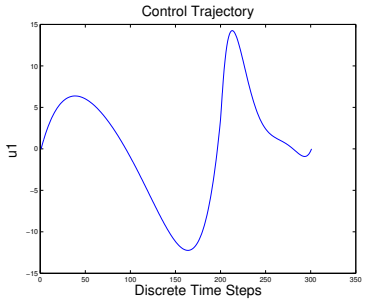
$$J_k = \sum_{i=1}^{P} \tfrac{1}{2} \int_{T_{i-1}}^{T_i} \left[|\bar{\mathbf{x}}|^2_{Q_{\bar{\mathbf{x}}_i}} + |\mathbf{v}|^2_{Q_{\mathbf{v}_i}} \right] dt + \tfrac{1}{2} |f_{T_i}(\bar{\mathbf{x}}_{T_i})|^2_{Q_{T_i}}$$

• March the nonsmooth augmented adjoint ODE backwards in time from $t = T_P : T_0$, where the adjoint ODE can be expressed

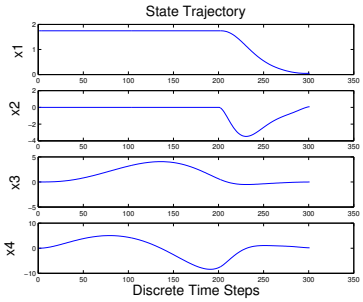
$$\begin{aligned} \dot{\lambda} &= -A_i^T \lambda - Q_{\bar{\mathbf{x}}_i} \bar{\mathbf{x}} && T_i > t > T_{i-1} \\ \lambda &= \tfrac{1}{2} \nabla_{\bar{\mathbf{x}}} |f_{T_i}(\bar{\mathbf{x}}(T_i))|^2_{Q_{T_i}} + \lambda(T_i) && t = T_i (\neq T_P) \\ \lambda &= \tfrac{1}{2} \nabla_{\bar{\mathbf{x}}} |f_{T_P}(\bar{\mathbf{x}}(T_P))|^2_{Q_{T_P}} && t = T_P \end{aligned}$$



(a) Trajectory Cost Value versus Iterations



(b) Control Trajectory versus Discrete Time Steps



(c) State Trajectory versus Discrete Time Steps

Fig. 4. Simulation Results for Nonsmooth Rover Upright with Control Trajectory Smoothing

- Compute the cost function gradient $\nabla_{\mathbf{v}} J_k$

$$\nabla_{\mathbf{v}} J_k = Q_{\mathbf{v}_i}\mathbf{v} + B_i^T \lambda \quad T_{i-1} < t < T_i$$

- Set search direction using steepest descent ($G_k = -\nabla_{\mathbf{v}} J_k$) or conjugate gradient method
- Use line search algorithm to find optimal step size [7]
- Compute new Trajectory Cost Value $J_{k+1}(\bar{\mathbf{x}}, \mathbf{v})$
- Check to see if Residual Tolerance is met

$$J_k(\bar{\mathbf{x}}, \mathbf{v}) - J_{k+1}(\bar{\mathbf{x}}, \mathbf{v}) < tol$$

if YES $\Rightarrow$ **STOP**
if NO $\Rightarrow$ advance iteration to $k = k + 1$
Loop to beginning

end(for)

C. MPDopt: Program Output

- State, Control, and Control Velocity Trajectory
- Cost Convergence
- System Animation (optional and user defined)

V. CONCLUSIONS AND CURRENT WORK

A. Conclusions

In this paper we have derived a method to determine a sequence of control inputs for a switched system such that a desired trajectory over a finite time horizon is achieved. We have extended the well known continuous time adjoint based gradient computation to a general class of nonsmooth systems by introducing point constraints into the trajectory cost equation. Further, we have introduced a method of control trajectory smoothing via state augmentation which substantially improves the control trajectory solution given by the optimization algorithm. Lastly, we have generalized the adjoint based dynamic optimization leading to the introduction of MPDopt, a standalone trajectory planning MatLab package for nonlinear, smooth, and nonsmooth classes of systems.

B. Current Work

Work is currently being done toward both the physical verification of the derived algorithm and the extension of MPDopt capability. Planned trajectory solutions will be applied to a cart and single pendulum swing-up, cart and double pendulum swing-up, and multiple rover maneuvers including upright, downright, and underactuated snatch and throw. Specific areas of interest toward improving MPDopt include the ability to incorporate hard constraints on the state and control variables, and, as alluded to in section II, to the optimization of the time intervals T_1 and T_2 themselves via time nondimensionalization of the system and subsequent optimization of the parameters involved in this time nondimensionalization. There is also interest in extending the algorithm to include Descriptor Systems.

REFERENCES

[1] T.R. Bewley, J. Kim, "A Linear Systems Approach to Flow Control" *Annual Review of Fluid Mechanics*, Vol. 39: 383-417, Jan. 2007.
[2] P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization*, Elsevier Academic Press, London, UK; 1986.
[3] C. Lemarechal, J.F. Bonnans, J.C. Gilbert, and C.A. Sagastizabal, *Numerical Optimization: Theoretical and Practical Aspects*, Springer-Verlag Berlin Heidelberg, 1997.
[4] C. Lemarechal and J.-B. Hiriart-Urruty, *Convex Analysis and Minimization Algorithms I:II*, Springer-Verlag Berlin Heidelberg, 1993.
[5] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, 1987.
[6] A. Ruszczynski, *Nonlinear Optimization*, Princeton University Press, Princeton, NJ, 2006.
[7] T.R. Bewley, *Numerical Renaissance: Simulation, Optimization, & Control*, http://renaissance.ucsd.edu/book/Welcome.html
[8] R. Fletcher *Practical Methods of Optimization*, John Wiley & Sons Ltd., 1987
[9] E.W. Kamen, B.S. Heck *Fundamentals of Signals and Systems*, Prentice Hall, Upper Saddle River, New Jersey, 1997
[10] S.J. Leon *Linear Algebra with Applications*, Prentice Hall, Upper Saddle River, New Jersey, 1998