ACC01-IEEE1682

Switching-Time Computation for Bang–Bang Control Laws

Stephen K. Lucas & C. Yalçın Kaya

Centre for Industrial and Applicable Mathematics

School of Mathematics

Mawson Lakes, S.A. 5095 Australia

s.lucas@unisa.edu.au & y.kaya@unisa.edu.au

March 14, 2003

Abstract

In this paper we obtain improvements and give extensions to the Switching Time Computation (STC) method, which is used to compute the switching times for bang–bang control laws. We first report a considerable computational improvement on the STC method as applied to a nonlinear system with one control input. The second contribution of this paper is the extension and implementation of the STC method for two control inputs. In bang–bang control calculations it is usual practice to consider all possible combinations of the switchings and then carry out the computations with a large set of switching parameters. We introduce a novel scheme for the switching times for two inputs which results in far fewer switching parameters to calculate.

1   Introduction

In control systems, one of the most common types of input function is the piecewise-constant function, where a sequence of constant inputs is used to control a given system, with appropriate switchings. For instance, in the aerodynamic control of some rocket vehicles, the tail fins are suitably deflected to several angular positions to achieve the necessary control action. This type of input results in a concatenation of a sequence of constant-input trajectories, or arcs. In the case when the input is bounded, a most frequently encountered type of piecewise-constant input is bang–bang, which switches between the upper and lower bounds of the control input. It is a well-known fact that the nonsingular time-optimal control solution of linear-analytic systems with bounded control inputs is the bang–bang control. The bang–bang solution is also encountered in other optimal control problems[1]. This situation arises especially when the hamiltonian is linear in the control input and the solution is not singular.

As soon as the controls are assumed to be bang–bang, the problem of finding the required controls becomes one of finding the switching times. This special case has long attracted the attention of researchers in the area of optimal control[5],[7]. In references [11] and [16], the instants of switchings are considered to be the parameters of the optimal control problem. Mohler, in references [7] and [8], gives a bang–bang control algorithm called the Switching-Time-Variation-Method (STVM). The STVM requires the number of switchings and the switching times as the

1

initial guess. It generates a sequence of switching functions and computes the gradient of the cost with respect to the switching times. With this gradient, the switching times are corrected at each iteration.

Consider the nonlinear dynamical control system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \ , \tag{1}$$

where $\mathbf{x}(t) = (x_1(t), \ldots, x_n(t)) \in \mathbb{R}^n$, $\mathbf{u}(t) = (u_1(t), \ldots, u_m(t)) \in \mathbb{R}^m$, and the vector field $\mathbf{f}$ is C^1 almost everywhere on $\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}$. The control input is restricted to the space of bang–bang controls, namely $\mathbf{u} : \mathbb{R}^+ \cup \{0\} \to \mathcal{U} \subset \mathbb{R}^m$, where the input space $\mathcal{U}$ is the product of m copies of the set $\{-1, 1\}$. In other words, the ith component of $\mathbf{u}$, $u_i(t) \in \{-1, 1\}$.

The problem is to find a feasible bang–bang control solution which takes the system from a given initial point $\mathbf{x}(0) = \mathbf{x}_0$ to a given terminal point $\mathbf{x}(t_f) = \mathbf{x}_f$, where the time t_f is free.

It is assumed that solutions to differential equations of the form $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{v}, t)$ satisfying the boundary conditions $\mathbf{x}(0) = \mathbf{x}_0$ and $\mathbf{x}(t_f) = \mathbf{x}_f$ exist for some $t_f > 0$, for some $\mathbf{v} \in \mathcal{U}$, all $\mathbf{x}_0 \in \mathbb{R}^n$, all $\mathbf{x}_f \in \mathbb{R}^n$ and all $t > 0$.

A *trajectory* of the system (1) corresponding to a control $\mathbf{u}(\cdot)$ is a continuous curve $\mathbf{x}(\cdot)$ solving (1) for almost all t. We also refer to $\mathbf{x}(t) \in \mathbb{R}^n$ for some t as the *state*.

Wen and Desrochers [14] give an algorithm for obtaining bang-bang control laws for linear-analytic systems, namely systems of the form $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{G}(\mathbf{x}(t))\mathbf{u}(t)$, with given initial and terminal points. Here $\mathbf{G}(\mathbf{x}(t))$ is an $n \times m$ matrix with nonlinear function entries of $\mathbf{x}(t)$. Kaya and Noakes [3] give a similar algorithm, called the Switching Time Computation (STC) method, for general nonlinear systems with single control input, and consider a more general terminal condition. In both papers variational equations are derived for $\mathbf{x}$ with respect to the switching times, or equivalently *arc times*, which are defined as the differences between the consecutive switching times. These variational equations are then used to calculate the gradient of the final state with respect to the switching times.

The STC method has been implemented as a computer code and used to find feasible bang–bang solutions for some highly nonlinear control systems. The feasible solution was then used to find time optimal bang–bang controls by the Time Optimal Switchings (TOS) algorithm by Kaya & Noakes [4]. Feasible solutions obtained by the STC can also be used to find more general optimal controls other than minimum-time ones. Work on an algorithm for more general optimal switchings starting with an STC solution is in progress.

In this work we first report a considerable computational improvement in the STC method as applied to a nonlinear system with one control input. The existing STC method and the associated code involves the minimization of the norm of the distance between $\mathbf{x}(t_f)$ and the terminal desired point $\mathbf{x}_f$. In the new improved version the problem of minimization has been replaced by the computationally more efficient scheme of solving a nonlinear system of equations, where $\mathbf{x}(t_f) = \mathbf{x}_f$ is solved for the arc times.

The second contribution of this paper is the extension and implementation of the STC method for two control inputs. In bang–bang control calculations it is usual practice to consider all possible combinations of the switchings and then carry out the computations with a large set of switching parameters. See for example Lee et al. [6]. When the number of switchings increases, the number of possible combinations grows exponentially. We introduce a novel scheme for the switching times with two inputs which results in far fewer switching parameters to deal with during the calculations. The resulting number of parameters to deal with is one less than the total number of arcs for both control inputs, which grows only linearly with the growing number of arcs.
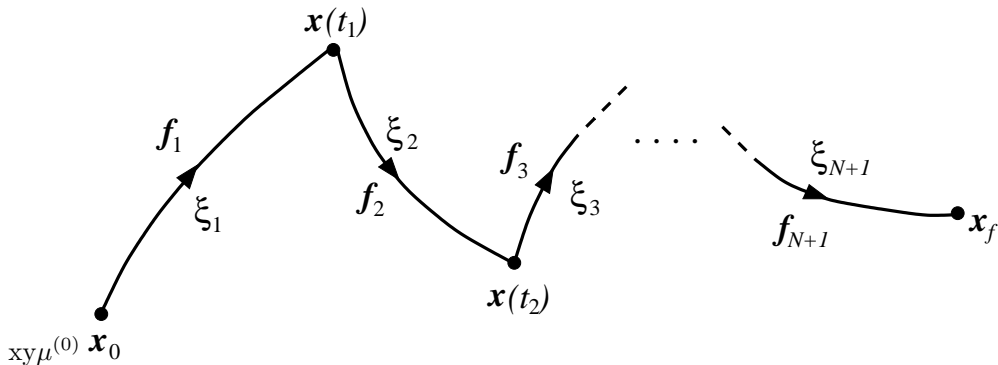
Figure 1: Concatenation of arcs from x_0 to x_f

2 STC Method

In this section, we briefly outline the Switching Time Control (STC) algorithm of Kaya & Noakes [3] for bang–bang control. A rigorous formulation of the variational equations for the STC method for one control input is given in [3]. In what follows, a review of this formulation is given with an appropriate notation. This leads to a variant aimed at reducing the computational effort and an extension to the case of two control inputs in the following sections.

Consider the system given in (1). Let the ith switching time be denoted by t_i, $i = 1, 2, \ldots, r-1$, such that $0 = t_0 \leq t_1 \leq t_2 \leq \cdots \leq t_{r-1} \leq t_r = t_f$. The initial and final times are given by $t_0 = 0$ and $t_r = t_f$. Given $\mathbf{x}_0$ and u_0, the trajectory $\mathbf{x}(t)$ is determined by the switching times. The segment of the trajectory $\mathbf{x}(t)$, where $t_{i-1} \leq t \leq t_i$, $i = 1, 2, \ldots, r$, is called the *ith arc*, or the *ith bang arc*, and denoted by $\mathbf{x}_i(t)$. Then the trajectory $\mathbf{x}(t)$, $0 \leq t \leq t_f$, is the concatenation of $\mathbf{x}_i(t)$. The time spent on the ith arc is called the *ith arc time* given by $\xi_i = t_i - t_{i-1}$. We also define the *arc times vector* $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_r)$. Note that $t_i = \sum_{j=1}^i \xi_j$, and $t_f = \sum_{j=1}^r \xi_j$. We require that each arc time ξ_i must be nonnegative for a physically realistic trajectory. So, while we have for the switching times the conditions $0 \leq t_1 \leq t_2 \leq \cdots \leq t_{r-1} \leq t_f$, for the arc times we have $\xi_i \geq 0$.

Let $\mathbf{f} = (f_1, \ldots, f_n)$. Then we write $\mathbf{f}_i = (f_{i1}, \ldots, f_{in})$ for the vector field in (1) on each arc, and similarly $\mathbf{x}_i = (x_{i1}, \ldots, x_{in})$. The ith arc $\mathbf{x}_i(t)$ satisfies the equations $\dot{\mathbf{x}}_i(t) = \mathbf{f}_i(\mathbf{x}_i(t), t)$, where $\mathbf{f}_i(\mathbf{x}_i(t), t) = \mathbf{f}(\mathbf{x}_i(t), u_i, t)$, $t_{i-1} \leq t \leq t_i$.

The number of arcs, r, is prescribed. We write $\mathbf{x}_i(t)$ as depending not only on time but also on the previous arc times, $\xi_1, \ldots, \xi_{i-1}$, namely $\mathbf{x}_i = \mathbf{x}_i(t; \xi_1, \ldots, \xi_{i-1})$. Then the problem can be stated as one of solving the following two-point boundary-value problem for the *parameters* $\xi_1, \ldots, \xi_r$ satisfying the boundary conditions $\mathbf{x}_1(0) = \mathbf{x}_0$ and $\mathbf{x}_1(t_f) = \mathbf{x}_f$:

$$\mathrm{P}_{TPBVP} : \begin{cases} \dfrac{\partial \mathbf{x}_i}{\partial t}(t; \xi_1, \xi_2, \ldots, \xi_{i-1}) = \mathbf{f}_i(\mathbf{x}_i(t; \xi_1, \xi_2, \ldots, \xi_{i-1}), t) \text{ on } [t_{i-1}, t_i]; \\ \mathbf{x}_1(0) = \mathbf{x}_0, \quad \mathbf{x}_i(t_{i-1}; \xi_1, \xi_2, \ldots, \xi_{i-1}) = \mathbf{x}_{i-1}(t_{i-1}; \xi_1, \xi_2, \ldots, \xi_{i-2}), \\ \mathbf{x}_r(t_f; \xi_1, \xi_2, \ldots, \xi_{r-1}) = \mathbf{x}_f, \quad i = 2, 3, \ldots, r \, . \end{cases} \tag{2}$$

Since t_f is the sum of ξ_i, the last line can be rewritten as

$$\mathbf{x}(\boldsymbol{\xi}) = \mathbf{x}_r(\boldsymbol{\xi}) = \mathbf{x}_f \, . \tag{3}$$

Problem (P_{TPBVP}) can also be viewed as the problem of solving the algebraic equation (3) subject to the ODEs that are sequentially given along each arc.

Kaya & Noakes [3] solved (3) using a minimisation technique—steepest descent followed by Newton's method for minimisation once the vector $\boldsymbol{\xi}$ gives a solution $\mathbf{x}(t_f; \boldsymbol{\xi})$ close enough to $\mathbf{x}_f$.

3

However, a more efficient technique is to directly apply Newton's method to the system of equations (3). This method will only require calculation of the Jacobian of the system, or the first partial derivatives $\partial \mathbf{x}(t_f)/\partial \boldsymbol{\xi}$, rather than $\partial \mathbf{x}(t_f)/\partial \boldsymbol{\xi}$ and $\partial^2 \mathbf{x}(t_f)/\partial \boldsymbol{\xi}^2$. However, the number of states n will typically be less than the number of arcs r, and so the standard Newton's method will not work.

The partial derivatives $\partial \mathbf{x}(t_f)/\partial \xi_i$, $i = 1, \ldots, r-1$, are calculated by solving the following differential equations simultaneously with the system equations given in Problem (P_{TPBVP}):

$$\frac{\partial}{\partial t} \frac{\partial \mathbf{x}_i}{\partial \xi_k}(t; \xi_1, \xi_2, \ldots, \xi_{i-1}) = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}(\mathbf{x}_i(t; \xi_1, \xi_2, \ldots, \xi_{i-1}), t) \frac{\partial \mathbf{x}_i}{\partial \xi_k}(t; \xi_1, \xi_2, \ldots, \xi_{i-1}) , \qquad (4)$$

with initial conditions

$$\frac{\partial \mathbf{x}_i}{\partial \xi_k}(t_{i-1}; \xi_1, \xi_2, \ldots, \xi_{i-1}) = \mathbf{f}_i(\mathbf{x}_i(t_{i-1}; \xi_1, \xi_2, \ldots, \xi_{i-1}), t_{i-1}) . \qquad (5)$$

The variation with respect to the final arc, namely $\partial \mathbf{x}(t_f)/\partial \xi_r$, is calculated from

$$\frac{\partial \mathbf{x}_r}{\partial \xi_r}(t_r; \xi_1, \xi_2, \ldots, \xi_r) = \mathbf{f}_r(\mathbf{x}_r(t_{i-1}; \xi_1, \xi_2, \ldots, \xi_{i-1}), t_r) . \qquad (6)$$

On the first arc we just solve for the state $\mathbf{x}_1(t)$, and form an initial condition for $\mathbf{x}^2$. On the second arc we solve for the state $\mathbf{x}_2(t)$ and $\partial \mathbf{x}(t_f)/\partial \xi_1$, and form an initial condition for $\mathbf{x}^3$ and $\partial \mathbf{x}(t)/\partial \xi_2$. This continues through to the last arc, where the required variations of $\mathbf{x}(t_f)$ with respect to all of the arcs are obtained. Note that the above calculations work perfectly well with arcs of zero length.

3   New Numerical Techniques for STC

3.1   Newton's method for underdetermined systems

Consider finding a solution to the nonlinear system of equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{F} = (f_1, f_2, \ldots, f_n)^T$, $\mathbf{x} = (x_1, x_2, \ldots, x_r)^T$, and $r > n$. Since there are more equations than unknowns, we expect an infinite number of solutions on some $(r-n)$-dimensional surface. If we are interested in finding just one of these possibly infinitely many solutions, we can follow the same technique as for Newton's method, and define the $n \times r$ matrix J with elements $J_{ij} = \partial f_i / \partial x_j$. The iteration then becomes

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J^+(\mathbf{x}^{(k)})\mathbf{F}(\mathbf{x}^{(k)}), \qquad (7)$$

where J^+ is the $r \times n$ Moore-Penrose generalised inverse, based on the singular value decomposition of J. If $J\mathbf{x} = \mathbf{b}$, then the solution $\mathbf{x} = J^+\mathbf{b}$ is the one where $||\mathbf{x}||_2$ takes its minimal value. This is particularly useful in this context, in that the update vector from the kth to $(k+1)$st iteration tries to find the point on $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ "close" to $\mathbf{x}^{(k)}$. This will hopefully help with the convergence difficulties of Newton's method.

This method is particularly suited to our problem. Walker [12, 13] gives an outline of this method, as well as some of its history. He also includes analysis of a Broyden style technique for underdetermined systems, which we will not pursue further here.

3.2   Modifications to Newton's method

Directly using (7) to solve (3) will not work unless the initial guess for $\boldsymbol{\xi}$ is sufficiently close to a feasible solution. The modified Newton's method for solving $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ (Press *et al.* [9], Dennis

& Schnabel [2]) involves choosing the Newton direction, then moving in that direction a distance such that $||\mathbf{F}||_2$ is decreased. While there are occasions where this technique leads to excessive numbers of iterations, it converges from almost anywhere, and is the technique we have currently implemented. An algorithm worth investigating in the future is the nonmonotone inexact Newton algorithm of Xiao & Chu [15], where $||\mathbf{F}||_2$ is allowed to increase in a controlled fashion. Numerical experiments in [15] indicate that the technique may be of use here.

A more important problem with using (7) to solve (3) is the constraints that $\xi_i \geq 0$, $i = 1, 2, \ldots, m$. Newton's method as described above is unconstrained, and there are situations where an initial guess of arc times will converge to a solution with negative arc times. While this is mathematically valid, it is not a physically realistic solution. Shacham [10] indicates that there are two main techniqes for dealing with constrained systems of this form: continuation, where one solves a series of problems from the starting guess through to a final solution that satisfies the constraints, and using penalty functions, which we will discuss further here.

Shacham suggests that instead of solving the system of r equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ with the constraint $x_i \geq 0$, $i = 1, 2, \ldots, r$, we form the penalty function $P = \sum_{j=1}^{r} \ln(x_j)$, and solve the system of equations $\mathbf{F_p}(\mathbf{x}) = \mathbf{F}(\mathbf{x})P = \mathbf{0}$. This has the same solution in the constrained region, and as long as the initial guess is in this region, Newton's method can now be used. While this technique is quite efficient, it has the disadvantage of adding superflous solutions when $P = 0$. We suggest a better choice for the penalty function as $P = \sum_{j=1}^{r} \ln(x_j) + (1/x_j)$, which is always positive in the constrained region. Solving $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ in this case changes the Jacobian of the system to

$$\hat{J}_{ij} = \frac{\partial(f_i P)}{\partial x_j} = P\frac{\partial f_i}{\partial x_j} + f_i\left(\frac{1}{x_j} - \frac{1}{x_j^2}\right). \tag{8}$$

We find that this technique is usually successful in solving (3) with all arc times nonnegative. However, there are times when it is unsuccessful. The modified Newton's method occasionally gets "stuck" near the constraint boundary, where the requirement of reducing $||\mathbf{F}||_2$ at each iteration leads to vanishingly small steps and an increasingly singular Jacobian.

There are two possible methods around this problem. The nonmonotone algorithm [15] may allow for escaping from near the boundary. Another possibility is to use the version of Newton's method described in Shacham [10], which does not require a reduction in $||\mathbf{F}||_2$ at each step. This version calculates the full Newton step, and if it is too large or the Jacobian is becoming singular it recalculates the step using the Levenberg/Marquardt (LM) algorithm. The LM algorithm involves a parameter allowing the step to vary between the Newton step and an infinitesimal step in the steepest descent direction. Further tests are required to determine the best combination of these ideas in an algorithm for solving (3).

3.3 Solving the ODEs

The ordinary differential equations in problem P_{TPBVP} were solved in [3] using a simple Runge-Kutta order 4 solver, where the step size was taken as 0.1, except for the step at the end of each arc, which was chosen so that a result was found precisely at the ends of arcs. A slightly more robust technique was followed here, where the same RKo4 solver was applied, but with 100 points on each arc. This ensures that even tiny arcs have sufficient function evaluations on them to satisfy reasoable accuracy. We emphasise here that the ODEs to be solved are inital value problems – far easier to solve than boundary value problems as found in many control algorithms.

A more efficient scheme could be to use an adaptive ODE solver, specifying the required accuracy. This does in many cases lead to a more efficient code with less function evaluations, particularly when close to a feasible solution. Unfortunately, many initial guesses lead to steps far away from the target point, where the ode's may be badly behaved. In these cases, an enormous amount of computer time is wasted in attempting to satisfy a required accuracy when the functions are

growing exponentially. While the modified Newton's method will bring the step back to more reasonable values, it still requires calculations out to these poorly behaved areas.

Our recommendation on solving general problems is to start with a simple 100 point RKo4 on each step with a modest accuracy requirement on reaching the target point. Once a set of arc lengths close to a feasible solution have been obtained, an adaptive ODE solver can be used with a high accuracy, to find a solution to as many digits accuracy as required.

4   STC Method with Two Control Inputs

Let us now consider the control problem $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u^1(t), u^2(t), t)$, where u^1 and u^2 are two independent controls. Assuming that the controls are associated with the arc times ξ_i^1, $i = 1, 2, \ldots, r_1$ and ξ_j^2, $j = 1, 2, \ldots, r_2$ (there is no requirement that $r_1 = r_2$). Then we can find the switching times $t_i^1 = \sum_{k=1}^i \xi_k^1$, $i = 1, 2, \ldots, r_1 - 1$, and $t_j^2 = \sum_{k=1}^j \xi_j^2$, $j = 1, 2, \ldots, r_2 - 1$. The initial and final times are given as $t_0^1 = t_0^2 = 0$ and $t_f = t_{r_1}^1 = \sum_{k=1}^{r_1} \xi_k^1$ or $t_f = t_{r_2}^2 = \sum_{k=1}^{r_2} \xi_k^2$. Note that we require that $t_{r_1}^1 = t_{r_2}^2$, so that both controls finish at the same time. For technical convenience, we refer to the initial and final times as switching times as well. Then, the $r_1 + r_2 + 1$ switching times can be merged together and relabeled $t_0 (= 0), t_1, \ldots, t_{r_1+r_2-1}$ (the repeated initial and final switching times are only required once here), and each time interval $[t_i - 1, t_i]$, $i = 1, 2, \ldots, r_1 + r_2 - 1$ can be considered as an arc, where both u^1 and u^2 are constants. We define new arc times based on these new switching times as $\boldsymbol{\eta}$, where $\eta_i = t_i - t_{i-1}$. The control values on each arc and which control changes from arc i to arc $i + 1$ can be easily determined, and we form the sequence ψ_i, $i = 1, 2, \ldots, r_1 + r_2 - 1$ such that ψ_i is exactly which ξ_j^1 or ξ_j^2 changes value from arc i to arc $i + 1$. Finally, we can also write the ξ's in terms of the η's: $\xi_i^j = \sum_{k=\alpha_{ij}}^{\alpha_{i+1,j}-1} \eta_k$, where α_{ij} is the arc upon which ξ_i^j starts in the original discretisation.

The system of odes we need to solve can now be written on the ith arc in a similar form as before:

$$
\begin{aligned}
&\frac{\partial \mathbf{x}_i}{\partial t}(t; \eta_1, \eta_2, \ldots, \eta_{i-1}) = \mathbf{f}_i(\mathbf{x}_i(t; \eta_1, \eta_2, \ldots, \eta_{i-1}), t), \quad i = 1, 2, \ldots, r_1 + r_2 - 1, \\
&\text{with} \quad \mathbf{x}_1(0) = \mathbf{x}_0, \quad \text{and} \\
&\mathbf{x}_i(t_{i-1}, \eta_1, \eta_2, \ldots, \eta_{i-1}) = \mathbf{x}_{i-1}(t_{i-1}, \eta_1, \eta_2, \ldots, \eta_{i-2}), \ i = 2, 3, \ldots, r_1 + r_2 - 1.
\end{aligned}
\tag{9}
$$

The system of equations we then need to solve, assuming the number of arcs r_1 and r_2 are prescribed, is

$$
\mathbf{x}(t_f; \boldsymbol{\eta}) - \mathbf{x}_f = \mathbf{0}
\tag{10}
$$

Equation (10) can be solved in exactly the same way as for (3) above, with the only changes being replacing ξ's by η's. The procedure required is to set up an iteration of Newton's method for (10) as for (3) previously, calculate the new $\xi's$ from the adjusted ηs, then form a new set of merged switching times, new ηs, and continue.

As a simple example to explain the above procedure and show its geometrical elegance, consider the situation in figure 2, with $r_1 = 2$, $\boldsymbol{\xi}^1 = (0.7, 1.4)^t$, $r_2 = 3$, and $\boldsymbol{\xi}^2 = (0.5, 0.8, 0.8)^t$. We see that $\mathbf{t} = (0, 0.5, 0.7, 1.3, 2.1)^t$, $\boldsymbol{\eta} = (0.5, 0.2, 0.6, 0.8)$, and assuming the control values are one and zero, the controls on each arc are (1,1), (0,1), (0,0), (0,1) respectively. Finally $\xi_1^1 = \eta_1 + \eta_2$, $\xi_2^1 = \eta_3 + \eta_4$, $\xi_1^2 = \eta_1$, $\xi_2^2 = \eta_2 + \eta_3$, and $\xi_3^2 = \eta_4$.

As a final comment before we leave this section, we note that the situation with more arcs, or even more control variables, is no more complicated than as described here. We simply work out the switching times for each control variable, merge these results into a current set of switching times, and label which control changes between arcs. If it turns out that the order of switchings are wrong, then at each step of Newton's method we simply recalculate the switching times and
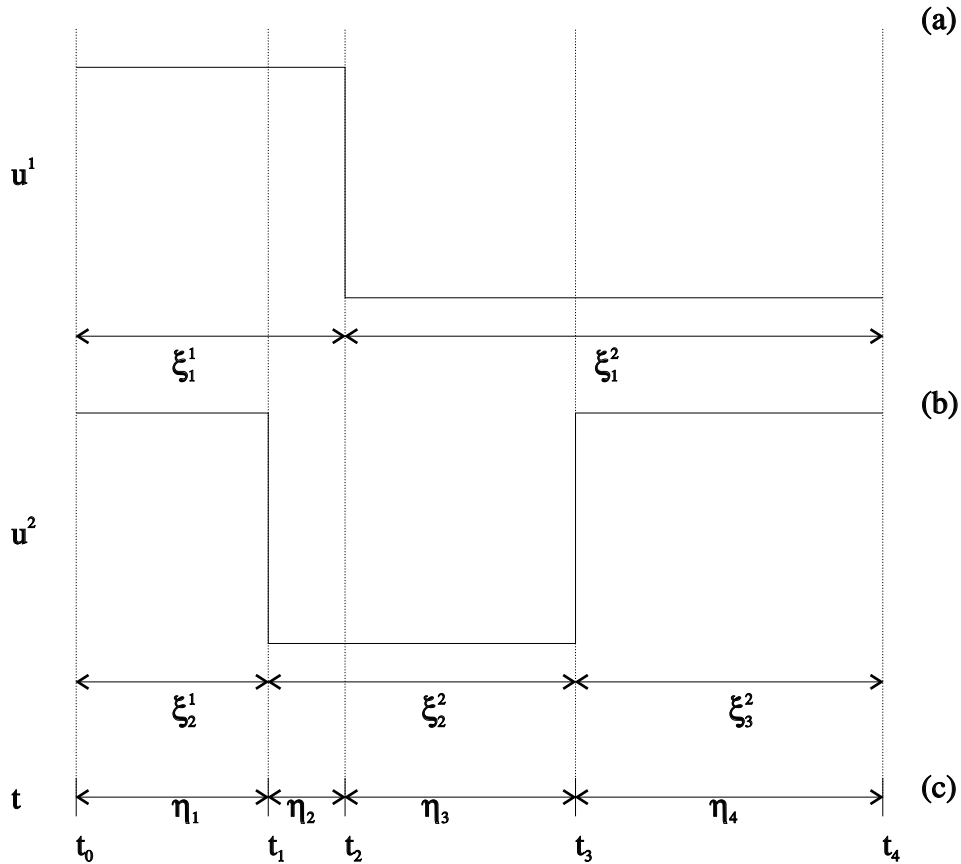
Figure 2: Merging two sets of control data (a, b) into a single set of arcs (c).

control changes as necessary. In multi-input bang–bang control calculations it is usual practice to consider all possible combinations of the switchings and then carry out the computations with a large set of switching parameters. See for example Lee et al. (1997). When the number of switchings increases, the number of possible combinations grows exponentially. In our case of two control inputs, the number of parameters to deal with is $r_1 + r_2 - 1$, which grows only linearly with the growing number of arcs.

5 Example Applications

5.1 One control input

We consider the van der Pol system studied in Reference [3]:

$$\dot{x}_1 = x_2 \ , \tag{11}$$

$$\dot{x}_2 = -x_1 - (x_1{}^2 - 1)x_2 + u \ , \tag{12}$$

where the control input $u(t)$ takes either the value 1 or -1. The initial and terminal points are $\mathbf{x}_0 = (-0.40, -0.60)$ and $\mathbf{x}_f = (0.60, 0.40)$, respectively. The reference [3] uses the version of the STC algorithm where the distance between $\mathbf{x}(t_f)$ and $\mathbf{x}_f$ is minimized. Figure 3 depicts the resulting iterations of that previous version of the STC algorithm as the trajectories run through the phase plane.
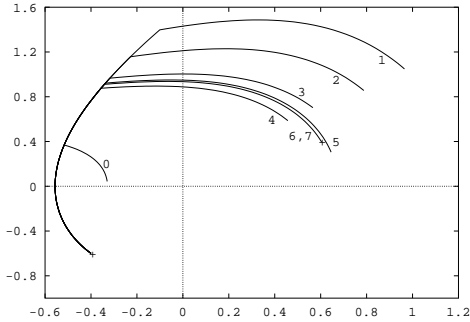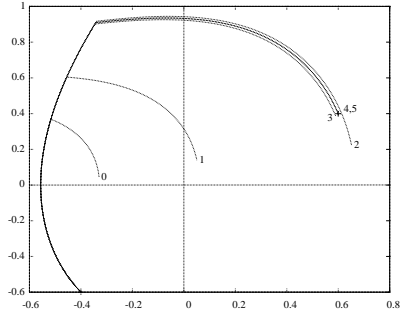
Figure 3: STC using minimization

Figure 4: STC solving nonlinear equations

Iterations	ξ_1^1	ξ_2^1	ξ_1^2	ξ_2^2	ξ_3^2	Distance
0	0.7	1.4	0.5	0.8	0.8	5.610373
1	0.464848	1.137783	0.312745	0.702649	0.587237	1.408439
2	0.329105	1.087859	0.216303	0.662390	0.538271	0.156110
3	0.312345	1.072819	0.202047	0.654472	0.528645	0.002418
4	0.312066	1.072706	0.201819	0.654441	0.528512	5.82×10^{-7}
5	0.312066	1.072707	0.201819	0.654442	0.528512	3.12×10^{-14}

Table 1: Arc lengths for the two control problem (to six dp)

An initial guess of $\boldsymbol{\xi} = (0.7000, 0.8000)$ results in a distance (between $\mathbf{x}(t_f)$ and $\mathbf{x}_f$) of 0.9958. The solution is obtained in the 8th step as $\boldsymbol{\xi} = (0.9766, 1.1637)$ with an accuracy of 10^{-6}. Using the same data, the STC algorithm using the modified Newton's method, shown in Figure 4, achieved this accuracy in five steps. This same speedup was seen for a wide range of initial guesses. We remind the reader that each step of the modified STC method is also much less computationally demanding.

5.2 Two control inputs

We will only consider here the simple problem

$$
\begin{aligned}
\dot{x}_1 &= x_2 + u_1, \\
\dot{x}_2 &= x_1 + u_2,
\end{aligned}
$$

with initial and terminal points $(0, 1)$ and $(1, 2)$ respectively. The controls can take the values $+1$ or -1, and we assume both start with the value $+1$. With the initial guess $\boldsymbol{\xi}^1 = (0.7, 1.4)$ and $\boldsymbol{\xi}^2 = (0.5, 0.8, 0.8)$, we get the results as shown in Table 1. Convergence is quadratic as expected from Newton's method. While this is a simple linear problem, we emphasise that the code we developed here is also applicable to nonlinear problems.

While the formulation as described in this paper for the two input case was successful for many initial guesses, it failed for some, where solutions with negative arcs were obtained. Techniques as described in the section on the one input case would need to be investigated for the multiple input case. This will be a more complicated problem, in that the constraints are no longer simply $\xi_i \geq 0$, but various combinations of η_i's being ≥ 0. For the example in Figure 2, the constraints would be $\xi_1^1 = \eta_1 + \eta_2 \geq 0$, $\xi_2^1 = \eta_3 + \eta_4 > 0$, $\xi_1^2 = \eta_1 \geq 0$, $\xi_2^2 = \eta_2 + \eta_3 \geq 0$, and $\xi_3^2 = \eta_4 \geq 0$.

6   Conclusion

In this work we have discussed and implemented new computational tools for the Switching Time Computation (STC) method for nonlinear systems with one control input. We have also extended and implemented the STC method for two control inputs. In achieving this we introduced a novel scheme for finding the switching times with two inputs which results in far fewer switching parameters to calculate, facilitating computational efficiency. While these contributions were illustrated through simple examples, these techniques are applicable to general systems.

References

[1] Belghith, S., F. Lamnabhi-Lagarrigue, and M.-M. Rosset, 'Bang–bang solutions for a class of problems arising in thermal control', in: *Algebraic and Geometric Methods in Nonlinear Control Theory* (Fliess, M., & Harzewinkel, M., eds), Dordrecht, D. Reichel Pub. Co., pp 623-632, 1986.

[2] J.E. Dennis & R.B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Englewood Cliffs, NJ: Prentice Hall (1983).

[3] C.Y. Kaya & J.L. Noakes, Computations and time-optimal controls, *Optimal Control Applications and Methods*, **17** 171–185 (1996).

[4] C.Y. Kaya & J.L. Noakes, Computational method for time-optimal switching control *Submitted to Journal of Optimization Theory and Applications.*

[5] Lastman, G. J., 'A shooting method for solving two-point boundary-value problems arising from nonsingular bang–bang optimal control processes', *International Journal of Control*, **27**, 513-524 (1978).

[6] Lee, H.W.J., Teo, K.L., Rehbock, V., and Jennings, L.S., Control Parameterization Enhancing Technique for Time Optimal Control Problems, *Dynamic Systems and Applications*, **6**243-262 (1997).

[7] Mohler, R. R., *Bilinear Control Processes*, Academic Press, New York, 1973, Chapter 3.

[8] Mohler, R .R., *Nonlinear Systems: V.2 Applications to Bilinear Control*, Prentice Hall: Englewood Cliffs, N.J., 1991, Chapter 7.

[9] W.H. Press, S.A. Teukolsky, W.T. Vetterling, & B.P. Flannery, *Numerical recipes in Fortran 77 2nd edition*, Cambridge University Press, 1992.

[10] M. Shacham, Numerical solution of constrained nonlinear algebraic equations, *Int. J. Numer. Meth. Eng.*, **23** 1455–1481 (1986).

[11] Teo, K. L., C. J. Goh, and K. H. Wong, *A Unified Computational Approach to Optimal Control Problems*, Longman Scientific and Technical, Essex, 1991.

[12] H.F. Walker, Newton-like methods for underdetermined systems, in *Computational solutions of non-linear systems of equations*, *Lect. Appl. Math.* **26** 679–699 (1990).

[13] H.F. Walker & L.T. Watson, Least-change secant update methods for underdetermined systems, *SIAM J. Numer. Anal.* **27** 1227–1262 (1990).

[14] Wen, J., & Desrochers, A.A., An algorithm for obtaining bang–bang control laws, *Journal of Dynamic Systems, Measurement, and Control*, **109** 171-175 (1987).

[15] Y. Xiao & E.K. Chu, A nonmonotone inexact Newton algorithm for nonlinear systems of equations, *J. Austral. Math. Soc. Ser. B*, **36** 460–492 (1995).

[16] Wong, K. H., D. J. Clements, and K. L. Teo, 'Optimal control computation for nonlinear time-lag systems', *Journal of Optimization Theory and Applications*, **47**, 91-107 (1985).