Second-order optimal estimation of slip state for a simple slip-steered vehicle

T. M. Caldwell and T. D. Murphey

***Abstract*— We present a method to optimally estimate when a slip state transition occurs for a slip-steered vehicle. A slip-steered vehicle's contact state with the ground must slip sideways in order for the vehicle to turn. This slipping generates uncertainties for an autonomous controller. These uncertainties can be be reduced if an estimate of when the vehicle switches between slipping and sticking is provided to the controller.**

We present an estimator that optimally determines when a switch between slipping and sticking occurs by comparing simulations of the slip-steered vehicle with its measured configurations. We demonstrate that steepest descent-based optimization has slow convergence and show how this issue can be rectified by using Newton's Method. The paper concludes with the introduction of an algorithm that uses second-order optimization in a manner that is appropriate for online implementation.

I. Introduction

A slip-steered (or skid-steered) vehicle's wheels must slip sideways in order for the vehicle to turn. Slipping introduces error when odometry is used to dead-reckon the vehicle's position and orientation. In [3] this is addressed by putting constraints on the vehicle's instantaneous center of mass to minimize the slipping error. In [8], a model is created where the parameters are decided on after an extensive characterization of the terrain. In [1], Kalman filters are used to minimize the uncertainty caused by unknown ground characteristics. However, it is not clear that a stochastic method intended for linear, or at least smooth, systems is appropriate for these nonsmooth systems.

We propose a method that assigns to each slip state (i.e., all wheels sticking, all wheels slipping, front wheels sticking with back wheels slipping, and front wheels slipping with back wheels sticking) its own dynamic model. An estimation process determines when a switch occurs so the controller can handle these abrupt transitions in the physical system. The purpose of this paper is to provide such an estimator.

The estimator finds optimal switching times by minimizing a cost function of the error between the measured configuration trajectories of the physical system and the simulated trajectories found by integrating forward the piecewise equations of motion that are "pieced" together at guessed switching times $T = \{T_1, T_2, \ldots, T_{N-1}\}$. This approach does not rely on guessing the coefficients of friction needed for predicting stick/slip transitions, but instead relies only on the system's configuration data. We expect this approach to be less sensitive to uncertain ground friction characteristics than other approaches because it does not rely on explicitly characterizing stick/slip transitions in terms of surface properties.

It is natural to ask why numerical optimization tools have not been used to estimate stick/slip transitions before; we believe this is related to the slow convergence often observed in the application of first-order optimization. This is a key issue addressed in the present work, and we demonstrate that the second-order methods provide fast convergence suitable for real-time implementation.

This paper is organized as follows: Section II describes the Slip-Steered Vehicle and its dynamics for different slip states. Section III introduces optimization using steepest descent of a switched autonomous system by deriving the gradient of a cost function with respect to the switching times. Section III also provides a first-order iterative algorithm for finding optimal switching time estimates. Section IV presents an example of applying the estimator to a simulated slip-steered vehicle. The example illustrates that the first-order algorithm described in Section III converges slowly and we discuss why that is to be expected. This motivates the use of Newton's Method explained in Section V and applied to the example. Lastly, Section VI discusses a second-order algorithm appropriate for online implementation. We end with conclusions and future research directions in Section VII.

II. The Slip-Steered Vehicle

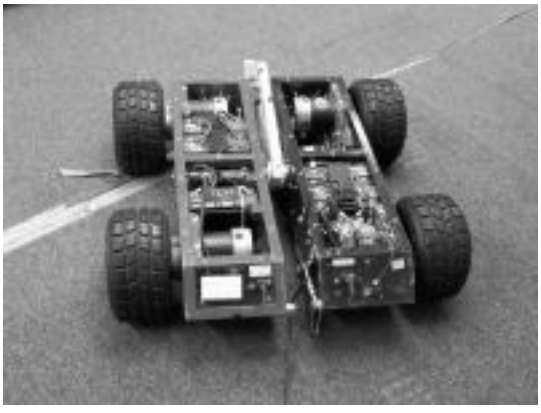

Fig. 1. The "Flexy Flyer". The author's slip-steered vehicle for testing

Our example slip-steered vehicle, pictured in Fig 1, is composed of four wheels supporting a body. The wheels

This work was supported by the National Science Foundation under CAREER award CMS-0546430

The authors are in the Department Electrical and Computer Engineering, University of Colorado at Boulder, Boulder, CO 80309, USA {timothy.caldwell,murphey}@colorado.edu

are locked in position and can only rotate. The vehicle is differentially driven by applying different torques to the wheels on the left and right sides of the vehicle. In order to turn, the difference between the torques must be large enough to break the sideways static friction between the wheels and ground. A zero-point turn is accomplished by applying equal but opposite torques to the right and left wheels when the vehicle is at rest.

The representation of the slip-steered vehicle is the same as in [6]. The configuration is composed of $q = (x, y, \theta, \phi_1, \phi_2, \phi_3, \phi_4)$. The Cartesian coordinates, x and y specify the center of geometry relative to the world frame. The orientation, θ, is the heading of the vehicle where θ increases counter-clockwise and $\theta = 0$ is when the heading is aligned with the x-axis of the world frame. The angle of rotation of each wheel respectively is $\phi_1, \phi_2, \phi_3, \phi_4$. The sub-script, 1, refers to the front left wheel and 2, 3, and 4 are ordered clockwise from wheel 1 as shown in Fig 2.

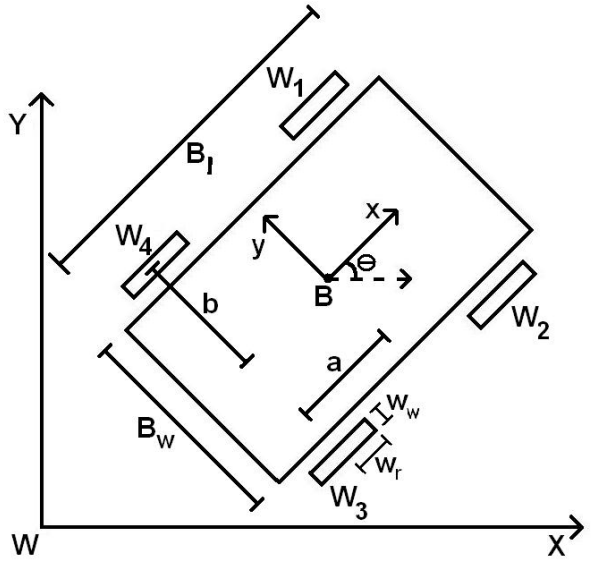

Fig. 2. Slip-steered vehicle with frame at its center of geometry.

The contact between the ground and wheels of the slip-steered vehicle at any time can be in one of four possible states. The combinations of whether the front and back wheels are sticking (i.e. constrained) or slipping (i.e. unconstrained) with the ground compose the states σ.

$$\sigma = \begin{cases} 1: & \text{Front and back wheels constrained} \\ 2: & \text{Front wheels constrained; back wheels not constrained} \\ 3: & \text{Front wheels not constrained; back wheels constrained} \\ 4: & \text{Front and back wheels not constrained} \end{cases}$$

As in [6], it is desirable to simplify the representation of the vehicle to a configuration of $q = (x, y, \theta)$. We'll call this reduced model the "Simplified Model" and the model that accounts for the wheel rotations (and associated inertias) as the "Full Model". This simplified model of the vehicle is a reduction of the full model at the cost of losing the impact of the rotational inertia of the wheels. This loss is negligible if the mass of the wheels is much less than the mass of the rest of the vehicle.

The inputs to the Full Model are torques on the wheels. Because the Simplified Model has no wheels, the inputs are transformed forces (i.e., using the $Ad(\cdot)$ operator [7]) to the center of mass of the vehicle. The inertia tensor for the Simplified Vehicle is $G = mdx \otimes dx + mdy \otimes dy + Jd\theta \otimes d\theta$, where m is the mass of the vehicle and J is the moment of inertia around the vehicle's center of mass.

The Euler Lagrange Equations are used to find the equations of motion for each slip state. Depending on the value of σ, the proper nonholonomic constraints are enforced.

The equations of motion for $\sigma = 1$ and $\sigma = 4$ are shown. Although we also compute the equations for $\sigma = 2, 3$, we do not reproduce them here because of their algebraic complexity.

$$\sigma = 1: \begin{cases} \ddot{x} = & \frac{(F_1+F_2+F_3+F_4)\cos\theta(t)}{m_B+4m_w} \\ & -\sin\theta(t)[\cos\theta(t)\dot{x}(t) + \sin\theta(t)\dot{y}(t)]\dot{\theta}(t) \\ \ddot{y} = & \frac{(F_1+F_2+F_3+F_4)\sin\theta(t)}{m_B+4m_w} \\ & +\cos\theta(t)[\cos\theta(t)\dot{x}(t) + \sin\theta(t)\dot{y}(t)]\dot{\theta}(t) \\ \ddot{\theta} = & 0 \end{cases} \tag{1}$$

$$\sigma = 4: \begin{cases} \ddot{x} = & \frac{(F_1+F_2+F_3+F_4)\cos\theta(t)}{m_B+4m_w} \\ & +g\mu_K \sin\theta(t)[-\sin\theta(t)\dot{x}(t) + \cos\theta(t)\dot{y}(t)] \\ \ddot{y} = & \frac{(F_1+F_2+F_3+F_4)\sin\theta(t)}{m_B+4m_w} \\ & -g\mu_K \cos\theta(t)[-\sin\theta(t)\dot{x}(t) + \cos\theta(t)\dot{y}(t)] \\ \ddot{\theta} = & \frac{12b(F1-F2-F3+F4)+12a^2g(m_B+4m_w)\mu_K\dot{\theta}(t)}{4m_w(12a^2+12b^2+w_w^2+3w_r^2)+m_B(B_l^2+B_w^2)} \end{cases} \tag{2}$$

where m_B is the mass of the body, m_w is the mass of a wheel, g is gravity, μ_K is the coefficient of kinetic friction and F_1, F_2, F_3 and F_4 are the respective input forces at the "wheels" that are transformed to the center of mass. The coefficient of kinetic friction shows up in the equations of motion when $\sigma = 4$ because there will be damping in the system when the wheels are slipping.

III. Finding Switching Times Using Steepest Descent

Steepest descent is a common approach in numerical optimization. It can be investigated further in [5]. The approach is an iterative one that uses the gradient of a cost function to descend to a local minimum.

We will use steepest descent to optimize the switching times of the switched autonomous system described by

$$\begin{cases} x(T_0) = x_0 \\ \dot{x}(t) = f_i(x(t)), \quad t \in (T_{i-1}, T_i), \quad i = 1, \ldots, N \end{cases} \tag{3}$$

where $x \in \mathbb{R}^n$, f_i is the ith function of a finite sequence of continuously differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}^n$, x_0 is the initial state, $T_i \in T$, the set of switching times, where $T_0 < T_1 < T_2 < \ldots < T_N$, T_0 is the initial time and T_N is the end time. The optimal switching times are denoted T^*.

The cost function that is to be minimized is:

$$J = \int_{T_0}^{T_N} L(x(t))dt \tag{4}$$

where $L(x(t))$ is a continuously differentiable function from $\mathbb{R}^n$ to $\mathbb{R}$. Often $L(x(t))$ is a norm. If the state is to be compared to some trajectory (i.e., measured data), the cost function may be changed to

$$J = \int_{T_0}^{T_N} L(x(t), h(t))dt \tag{5}$$

where $h(t)$ is the measured trajectory. If $L(x(t), h(t))$ is chosen as $\|x(t) - h(t)\|$, then J is the total integrated error from time T_0 to T_N.

The gradient is the direction of steepest descent *in the vector space of switching times*. The gradient is composed of the set of partials of the cost function with respect to the switching times.

$$\nabla J = (\frac{\partial J}{\partial T_1}, \frac{\partial J}{\partial T_2}, \ldots, \frac{\partial J}{\partial T_{N-1}}) \tag{6}$$

The gradient is derived from Eq. 4 using calculus of variations. Although the derivation can be found in [4], we recreate it here because the calculation is reasonably short and provides insight into how one calculates steepest descent iterations for optimally estimating switching times.

In order to do steepest descent, we wish to calculate $\frac{\partial J}{\partial T_i}$. Start with the definition of a derivative in the direction θ_i:

$$\frac{\partial J}{\partial T_i}\theta_i = \lim_{\epsilon \to 0} \frac{J(T+\epsilon\theta_i) - J(T)}{\epsilon} \quad i = 1, 2, \ldots, N-1 \tag{7}$$

where $T + \epsilon\theta_i = (T_1, \ldots, T_{i-1}, T_i + \epsilon\theta_i, T_{i+1}, \ldots, T_{N-1})$ and $\epsilon << 1$. Infinitesimally altering the T_i switching time causes a slight variation of the state trajectory such that $x(t) \to x(t) + \epsilon\eta(t)$ where η is the variation and $\epsilon\eta$ is an infinitesimal variation.

A Lagrange Multiplier λ is used to constrain the cost function to the trajectories defined by Eq. 3. The augmented cost function,

$$J_A(T) = \sum_{k=1}^{N} \int_{T_{k-1}}^{T_k} [L(x(t)) + \lambda(t)(f_k(x(t)) - \dot{x}(t))]dt \tag{8}$$

adjoins the Lagrangian and the constraints such that $J_A(T)$ is minimized when the constraints are satisfied. That is, minimizing $J_A(T)$ minimizes $J(T)$ while satisfying the constraints.

Now, for notational reasons, let

$$A_k(\eta(t)) = L(x(t)+\epsilon\eta(t)) + \lambda(t)[f_k(x(t)+\epsilon\eta(t)) - \dot{x}(t) - \epsilon\dot{\eta}(t)].$$

Therefore,

$$J_A(T) = \sum_{k=1}^{N} \int_{T_{k-1}}^{T_k} A_k(0)dt.$$

Taylor expand $A_k(\eta(t))$ around $x(t)$ by Taylor expanding $L(x(t) + \epsilon\eta(t))$ and $f_k(x(t) + \epsilon\eta(t))$ around $x(t)$:

$$A_k(\eta(t)) = L(x(t)) + \frac{\partial L}{\partial x}(x(t))\epsilon\eta(t) + \lambda(t)[f_k(x(t)) + \frac{\partial f_k}{\partial x}(x(t))\epsilon\eta(t) - \dot{x}(t) - \epsilon\dot{\eta}(t)] + O(\epsilon^2)$$

Now let us expand $J_A(T)$.

$$\begin{aligned}
J_A(T+\epsilon\theta_i) &= \sum_{k=1, k\neq i, i-1}^{N} \int_{T_{k-1}}^{T_k} A_k(\eta(t))dt \\
&\quad + \int_{T_{i-1}}^{T_i+\epsilon\theta_i} A_i(\eta(t))dt + \int_{T_i+\epsilon\theta_i}^{T_{i+1}} A_{i+1}(\eta(t))dt \\
&= \sum_{k=1}^{N} \int_{T_{k-1}}^{T_k} A_k(\eta(t))dt \\
&\quad - \int_{T_i}^{T_i+\epsilon\theta_i} A_i(\eta(t))dt + \int_{T_i}^{T_i+\epsilon\theta_i} A_{i+1}(\eta(t))dt \\
&= \sum_{k=1}^{N} \int_{T_{k-1}}^{T_k} \Big[A_k(0) + \frac{\partial L}{\partial x}(x(t))\epsilon\eta(t) \\
&\quad + \lambda(t)[\frac{\partial f_k}{\partial x}(x(t))\epsilon\eta(t) - \epsilon\dot{\eta}(t)] \Big] dt \\
&\quad + \int_{T_i}^{T_i+\epsilon\theta_i} [\lambda(t)\left(f_{i+1}(x(t)) - f_i(x(t))\right) \\
&\quad + O(\epsilon)]\, dt \\
&= J_A(T) + \epsilon \sum_{k=1}^{N} \int_{T_{k-1}}^{T_k} \Big[\frac{\partial L}{\partial x}(x(t))\eta(t) \\
&\quad + \lambda(t) \left(\frac{\partial f_k}{\partial x}(x(t))\eta(t) - \dot{\eta}(t) \right) \Big] dt \\
&\quad + \epsilon\theta_i [\lambda(t)(f_{i+1}(x(t)) - f_i(x(t)))]_{t=T_i^+} \\
&\quad + O(\epsilon^2)
\end{aligned}$$

Conduct integration by parts on $\epsilon \sum_{k=1}^{N} \int_{T_{k-1}}^{T_k} \lambda(t)\dot{\eta}(t)dt$ and pull $\eta(t)$ out for the summed integral expression.

$$\begin{aligned}
J_A(T+\epsilon\theta_i) &= J_A(T) + \epsilon \sum_{k=1}^{N} \int_{T_{k-1}}^{T_k} \Big[\frac{\partial L}{\partial x}(x(t)) \\
&\quad + \lambda(t)\frac{\partial f_k}{\partial x}(x(t)) + \dot{\lambda}(t) \Big] \eta(t)dt \\
&\quad - \sum_{k=1}^{N} [\lambda(t)\eta(t)]_{T_{k-1}}^{T_k} \\
&\quad + \epsilon\theta_i\lambda(T_i)[f_{i+1}(x(T_i)) - f_i(x(T_i))] + O(\epsilon^2)
\end{aligned}$$

Choose the Lagrangian Multipliers, λ, so that the solution does not depend on the η terms. In other words, the solution holds for all choices of η. With λ defined as

$$\begin{cases} \lambda(T_N) = 0 \\ \dot{\lambda}(t) = -\frac{\partial L}{\partial x}(x(t)) - \lambda(t)\frac{\partial f_i}{\partial x}(x(t)), \\ \qquad t \in (T_{i-1}, T_i), \quad i = 1, \ldots, N \end{cases} \tag{9}$$

then

$$J_A(T+\epsilon\theta_i) = J_A(T) + \epsilon\theta_i\lambda(T_i)[f_{i+1}(x(T_i)) - f_i(x(T_i))] + O(\epsilon)$$

Plug $J_A(T)$ and $J_A(T+\epsilon\theta_i)$ into the derivative equation,

Eq. 7:

$$\begin{aligned}\frac{\partial J_A}{\partial T_i}\theta_i &= \lim_{\epsilon \to 0}\left[J_A(T) + \epsilon\theta_i\lambda(T_i)[f_{i+1}(x(T_i)) - f_i(x(T_i))]\right. \\ &\quad \left. -J(T)\right]/\epsilon \\ &= \theta_i\lambda(T_i)[f_{i+1}(x(T_i)) - f_i(x(T_i))]\end{aligned}$$

Therefore, the system described by Eq. 3 with cost function Eq. 4 has direction of steepest descent in the direction of the gradient with components

$$\frac{\partial J}{\partial T_i} = \lambda(T_i)[f_{i+1}(x(T_i)) - f_i(x(T_i))] \tag{10}$$

where λ satisfies the backward differential equation in Eq. 9.

Using the gradient equations, an iterative algorithm for finding the optimal switching times is shown in Algorithm 1.

Algorithm 1: Steepest Descent

1) Start with some initial guess $\overline{T}$.
2) Compute the state $x(t)$ forward in time for $\overline{T}$ from T_0 to T_N from Eq. 3.
3) Compute the co-state $\lambda(t)$ backward in time for $\overline{T}$ from T_N to T_0 from Eq. 9.
4) Compute the gradient ∇J with components from Eq. 10.
5) Update $\overline{T}$ by taking a step in the direction of the gradient: $\overline{T} = \overline{T} - \gamma\nabla J$ where γ is the Armijo step size (see [2]).
6) Repeat steps 2 through 5 for new $\overline{T}$ until convergence criteria is met.

IV. Example of Applying Estimator to a Slip-Steered Vehicle

A simple example is shown to demonstrate applying Algorithm 1 to estimate the switching times of a slip-steered vehicle that transitions from $\sigma = 1$ (all wheels slipping constrained) to $\sigma = 4$ (no wheels slipping constrained) and back to $\sigma = 1$. These are by far the most common–see [6]. Note that all simulations were done in *Mathematica*.

The Simple Model of the slip-steered vehicle is used to both simulate the "measured" trajectories and as the model used by the estimator. With this setup, the Cost function evaluates to 0 when the optimal switching times are found because the estimator's system model is perfect. For the example, $h(t)$ is composed of the position and orientation trajectories as well as positional velocity and orientation velocity trajectories.

The initial state is $x(0) = 0$. The input forces used in the simulation are

$$\begin{bmatrix} u_{left} \\ u_{right} \end{bmatrix} = \begin{cases} \begin{bmatrix} 150 \\ 0 \end{bmatrix} & \text{(forward)} \quad ,0 < t < 2 \\ \begin{bmatrix} 255 \\ -255 \end{bmatrix} & \text{(turn)} \quad ,2 < t < 4 \\ \begin{bmatrix} 75 \\ 75 \end{bmatrix} & \text{(forward)} \quad ,4 < t < 6 \end{cases} \tag{11}$$

where $F_1 = F_4 = u_{left}$ and $F_2 = F_3 = u_{right}$. The difference between the left and right forces for the first two seconds is not large enough to break out of state $\sigma = 1$ so the vehicle goes straight. We chose not to have u_{left} and u_{right} to be equal initially because with a zero initial state and equal forces on all wheels, the dynamics of $\sigma = 1$ and $\sigma = 4$ are equivalent. This adds an extra complexity to the estimation and distracts from the purpose of the example.

These input forces result with simulated switching times of $T^\star = (2.00, 4.17)$ seconds.

For the example, the cost function was chosen to be

$$J(\overline{T}) = \int_{T_0}^{T_N} \|x(t) - h(t)\|^2 dt$$

where x(t) is the state calculated from integrating the Simplified Model's equations of motion for switching times $\overline{T}$. Therefore, the cost function is the norm squared of the error between $x(t)$ and $h(t)$.

Fig 3 shows a contour plot of the cost function for the first switching time T_1 versus the second switching time T_2 centered around T^*.

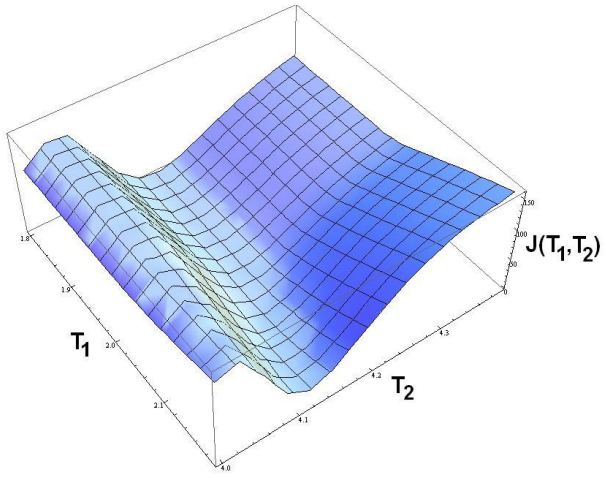

Fig. 3. Contour plot of cost versus switching times (T_1, T_2) where $T^* = (2.00, 4.17)$ seconds.

Algorithm 1 is used to find the optimal switching times starting from $\overline{T} = (1.6, 4.2)$ seconds. The initial guess for T_2 is very close to $T_2^\star$. We did this because later on it will be important for the initial guess to be in the basin of attraction of $T^\star$ and it also still illustrate the downfall of steepest descent.

Armijo step size, introduced in [2], is used to compute the the step size γ. The Armijo parameters α and β were

chosen to both be 0.4. The convergence criteria is met if the L_2 norm of the gradient is less than the tolerance 0.003. The results of using Algorithm 1 are shown in Fig 4.

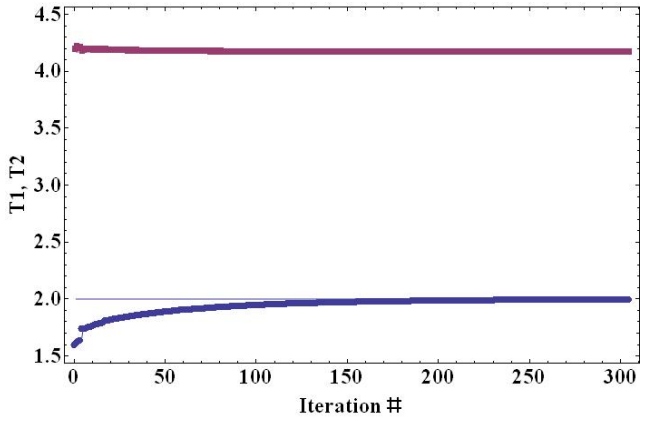

Fig. 4. Convergence of T_1 and T_2 to $T^\star = (2.00, 4.17)$ seconds using Algorithm 1. The algorithm converges to T^* in 304 iterations

The algorithm converged correctly to T* but it took 304 iterations of the algorithm to meet the convergence criteria of 0.003. This is due to lopsided behavior between the gradient in the T_2 direction compared to the T_1 direction. This can be seen in the contour plot of the cost where a "banana" shape is formed due to poor scaling of the cost function. This problem is discussed in [5]. To further illustrate this, the gradient field is plotted in Fig 5 and a "zoomed in" plot is shown in Fig 6.

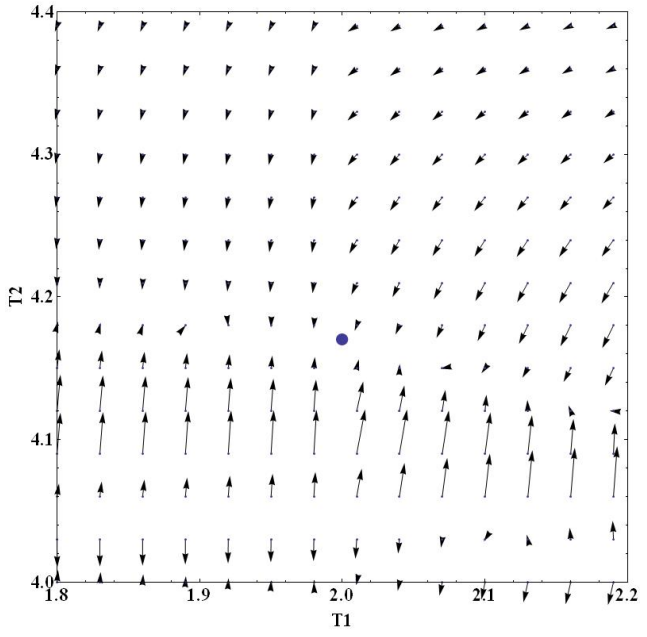

Fig. 5. Plot of the gradient field showing lopsided behavior of the gradient. The optimal switching times are $T^* = (2.00, 4.17)$ seconds and is designated with a circle.

Slow convergence, as seen in this example, would make it difficult to realize this estimator in real-time. The next section looks at using Newton's Method to find the switching times. Newton's Method doesn't have the same scaling problems as Steepest Descent and therefore has a greatly increased rate of convergence to the optimal switching times.

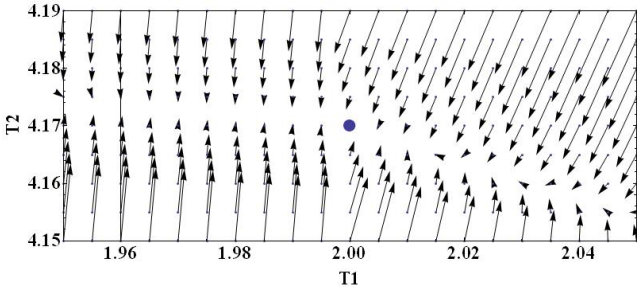

Fig. 6. A magnified plot of the gradient field showing lopsided behavior of the gradient near the converged switching times. The optimal switching times are $T^* = (2.00, 4.17)$ seconds and is designated with a circle.

V. Newton's Method and Example Continued

Newton's Method for finding switching times is investigated. The example started in Section IV is continued to show that Newton's Method does not suffer from the same slow convergence of Steepest Descent.

Newton's Method is an iterative method that converges to some local minimizer like Steepest Descent. Newton's Method though, does not have scaling problems like Steepest Descent and converges rapidly if the initial guess is within the local convergence of T^* (see [5]). Newton's Method is:

$$T^{k+1} = T^k - \gamma(\nabla^2 J(T^k))^{-1}\nabla J(T^k) \tag{12}$$

where T^k is the kth iteration, $\nabla J(T^k)$ is the gradient of the cost function, and $\nabla^2 J$ is the Hessian of the cost function with $\nabla^2 J = \text{Hess}_T J$. It is defined as:

$$\text{Hess}_T J = \begin{bmatrix} \frac{\partial^2 J}{\partial T_1^2} & \frac{\partial^2 J}{\partial T_1 \partial T_2} & \cdots & \frac{\partial^2 J}{\partial T_1 \partial T_{N-1}} \\ \frac{\partial^2 J}{\partial T_2 \partial T_1} & \frac{\partial^2 J}{\partial T_2^2} & \cdots & \frac{\partial^2 J}{\partial T_2 \partial T_{N-1}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 J}{\partial T_N \partial T_1} & \frac{\partial^2 J}{\partial T_N \partial T_2} & \cdots & \frac{\partial^2 J}{\partial T_{N-1}^2} \end{bmatrix} \tag{13}$$

A few notes:

- The Hessian must be positive definite to guarantee Newton's Method is stepping in a descending direction ($J(T^{k+1}) < J(T^k)\ \forall k$). It is often beneficial to evaluate a couple iterations of steepest descent before using Newton's Method because steepest descent tends to perform well globally while Newton's Method performs better near a minimizer. (see [5])
- The steepest descent direction is a special case of Newton's Method when the Hessian is the identity I_N.
- The Hessian is symmetric because mixed partials commute.

We would prefer to have a formula for finding the Hessian similar to the gradient in Eq.(10). For now, the second

partials will be calculated numerically to show why one would want to use Newton's Method. We have made progress on this front, but the functional analysis for this problem is both unnecessary for this application and is beyond the scope of the present paper.

Because the first partials can already be calculated from Eqs. 9 and 10, the second partials are numerically calculated using these equations.

Therefore,

$$\frac{\partial^2 J}{\partial T_i \partial T_j} = \begin{cases} \frac{\frac{\partial J}{\partial T_i}(T)\Big|_{T_j=T_j+\epsilon} - \frac{\partial J}{\partial T_i}(T)}{\epsilon} \\ \frac{\frac{\partial J}{\partial T_j}(T)\Big|_{T_i=T_i+\epsilon} - \frac{\partial J}{\partial T_j}(T)}{\epsilon} \end{cases} \quad i,j = 1,\ldots,N \tag{14}$$

are the approximate elements of the Hessian when $\epsilon << 1$. This leads us to an iterative algorithm for applying Newton's Method.

Algorithm 2: Newton's Method

1) Start with some initial guess $\overline{T}$.
2) Compute the state $x(t)$ forward in time for $\overline{T}$ from T_0 to T_N from Eq. 3.
3) Compute the co-state $\lambda(t)$ backward in time for $\overline{T}$ from T_N to T_0 from Eq. 9.
4) Compute the gradient ∇J with components from Eq. 10.
5) Compute the Hessian $\nabla^2 J$ from Eq. 13 with elements from Eq. 14 and insure the Hessian is positive definite.
6) Update $\overline{T}$ by taking a step: $\overline{T}^{k+1} = \overline{T} - \gamma(\nabla^2 J(\overline{T}))^{-1}\nabla J(\overline{T})$, where γ is the Armijo step size (see [2]).
7) Repeat steps 2 through 6 for new $\overline{T}$ until convergence criteria is met.

Continuing with the example from Section IV, Algorithm 2 is applied to find the optimal switching times. Again, our initial guess is $\overline{T} = (1.6, 4.2)$ seconds. As stated earlier, $\overline{T}_2$ is almost a perfect guess for $T_2^\star$. Right now we choose something in the basin of attraction or $T^\star$ because the Hessian for Newton's Method must be positive definite. Later we will see that a combination of steepest descent and Newton's Method allows us to efficiently estimate the switching times in a manner appropriate for online implementation

As in the earlier example, we used the Armijo step size with parameters $\alpha = \beta = 0.4$ and used a tolerance of 0.003 for the convergence criteria. We also chose ϵ for calculating the Hessian to be 0.0001. The results of using Algorithm 2 are shown in Fig 7.

Algorithm 2 converges to T* = (2.00,4.17) seconds in five iterations. This is a great increase in convergence speed compared to the 304 iterations from Algorithm 2. It took 12.4 seconds for *Mathematica* to realize this result. *Mathematica* is a rather slow platform when compared to an embedded one. We expect an embedded implementation will be able to run Algorithm 2 in real-time.

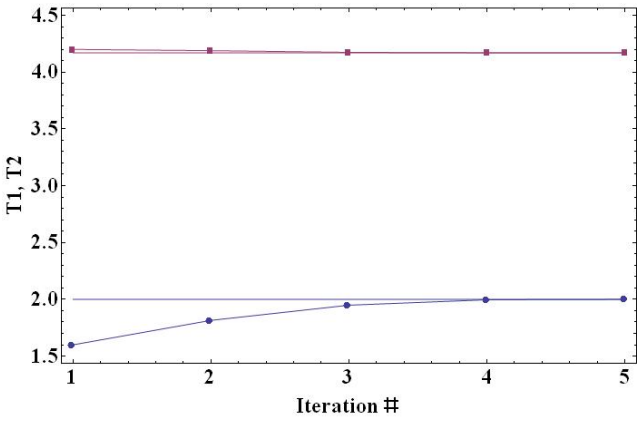

Fig. 7. Plot of the convergence of T_1 and T_2 to $T^* = (2.00, 4.17)$ using Algorithm 2. Newton's Method converges in 5 iterations compared to the 304 iterations using Steepest Descent.

To help illustrate why Algorithm 2 converged so quickly, the vector field composed of $(\nabla^2 J(T))^{-1}\nabla J(T)$ is shown in Fig 8. Fig 8 clearly shows the vector field aimed at the optimal switching times. This is made more obvious when compared to the gradient vector field shown in Fig 6.

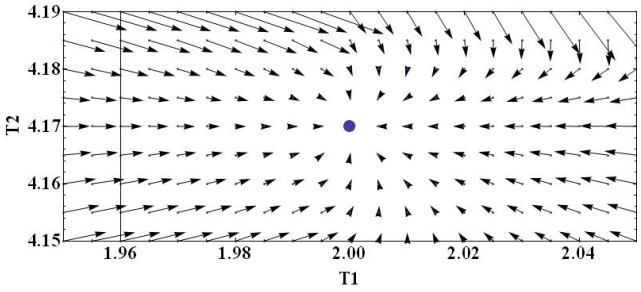

Fig. 8. Vector field computed from $(\nabla^2 J(T))^{-1}\nabla J(T)$. Vector field clearly converges to an obvious switching time, T^*. The optimal switching times are $T^* = (2.00, 4.17)$ seconds and is designated with a circle.

A. Estimation for the Full Model of the Slip-Steered Vehicle

Another example was conducted that used the Full Model of the slip-steered vehicle from Section II to generate the actual trajectories $h(t)$. All the same parameters were used including the input forces. Algorithms 1 and 2 were applied using the Simple Model. Estimating switching times with the Simple Model on trajectories found from the Full Model adds error from model disturbance.

The algorithms had similar results to the earlier, simpler example. It took 156 iterations of Steepest Descent to converge and 5 iterations of Newton's Method to converge. Instead of converging perfectly to T^*, the Algorithms converged slightly off to (1.957,4.183). Hence, the additional complexity of the Full Model did not cause the estimation algorithm to fail.

VI. Windowed Optimization for Online Implementation

A controller should know when the system's state switches as soon as possible after it occurs. We propose a way to do an online implementation. The algorithm is applied to the Simple Model from Section II.

The difficulty with determining the timing of a switch soon after it occurs is that sufficient error must accumulate before the estimator can converge to the correct time. Error develops when the actual trajectories and the modeled trajectories differ because the modeled trajectories are in the wrong state for some period of time.

The presented algorithm finds one switching time per timestep. A sliding time window is used to reduce the numerical integration required for each iteration. The window is of constant length except when a switching time is found, when the window's left bound is moved to the found switching time. The window's right bound is always the current time.

The initial guess for $\overline{T}$ is halfway between the left and right bounds of the window. Each iteration estimates a switching time by performing a couple iterations of Steepest Descent (Algorithm 1) followed by executing Newton's Method (Algorithm 2) until convergence. A switching time is found when the estimated switching times from consecutive iterations are within some tolerance of each other.

The algorithm uses the following constants:

- STEPSIZE = desired iteration step size
- WINDOWSIZE = desired window size
- BNDRY_TOL = the closest an estimated switching time can be to the edges of the window before being rejected
- SW_VERIFY_TOL = the maximum time between consecutive estimated switching times to be deemed a switching time

and the following variables at initial time:

- σ_{cur} = initial state (current state)
- σ_{next} = second state (next state)
- $T_b = T_0$+STEPSIZE (current time)
- $T_a = T_0$ (left time bound of window)
- $T_{prev} = T_0$ (time of previous switch)
- $T_{poss} = 0$ (possible switching time)

The algorithm is shown as pseudo-code for clarity.

(Pseudo) Algorithm 3: Online Implementation

```
while T_b < T_N
    T_a = T_b − WINDOWSIZE;
    if T_a < T_prev,
        T_a = T_prev;
    end if
    T̄ = (T_a + T_b)/2;
    T̄ = Perform 3 iterations of Steepest Descent
        using Algorithm 1 with initial guess T̄;
    T̄ = Perform Newton's Method from Algorithm
        2 with T̄ until convergence to find estimated
        switching time;
    if T_a + BNDRY_TOL < T̄ < T_b − BNDRY_TOL
        and (Hessians from Algorithm 2 are Positive
        Definite),
        if |T̄ − T_poss| < SW_VERIFY_TOL,
            T̄ is a switching time;
            σ_cur = σ_next;
            σ_next = the next expected state;
            T_poss = 0;
        else
            T_poss = T̄;
        end if
    else
        T_poss = 0;
    end if
    T_b = T_b + STEPSIZE;
end while
```

We applied Algorithm 3 to an example using the Simplified Model of the slip-steered vehicle from Section II. We chose inputs that resulted in the following eight switching times

$$T^\star = (0.5, 1.3, 1.9, 2.2, 4.0, 4.7, 6.5, 8.0)$$

with contact states

$$\sigma = (1, 4, 1, 4, 1, 4, 1, 4, 1)$$

We used an iteration step size of STEPSIZE $= 0.1$s, a window size of WINDOWSIZE $= 2$s, and tolerances of BNDRY_TOL $= 0.05$s and SW_VERIFY_TOL $= 0.01$s.

Algorithm 3 resulted in correctly estimating every switching time 0.2 seconds after the switch occurred except for the second and sixth switching times which were correctly estimated 0.5 seconds after their respective switches.

VII. Future Work and Conclusions

We presented an optimal estimator that estimates the slip state switching times of a slip-steered vehicle. A simple example to demonstrate how to use the estimator on a slip-steered vehicle had promising results with fast convergence when using Newton's Method. While there is much future work, the presented estimator is a good initial step to properly autonomously controlling systems with multiple states like the slip-steered vehicle.

There is much future research to do. An extensive investigation must be conducted to see how well the estimator

rejects disturbances such as noise and modeling errors. A derivation of an analytical formula for the Hessian of the cost function used for Newton's Method needs to be done This should reduce error from numerically calculating the second partial derivatives. An intelligent approach should be found for dealing with systems that have multiple states and the order of the states are not known. Also, the estimator needs to be made real-time and placed on an embedded platform for testing on the vehicle in Fig 1.

REFERENCES

[1] Georgia Anousaki and K. J. Kyriakopoulos. A dead-reckoning scheme for skid-steered vehicles in outdoor environments. *International Conference on Robotics and Automation*, 2004.

[2] L. Armijo. Minimization of functions having lipschitz continuous first-partial derivatives. *Pacific Journal of Mathematics*, 1966.

[3] Luca Caracciolo, Alessandro De Luca, and Stefano Iannitti. Trajectory tracking control of a four-wheel differentially driven mobile robot. *International Conference on Robotics and Automation*, 1999.

[4] Florent C. Delmotte. *Multi-Modal Control: From Motion Description Languages to Optimal Control*. PhD thesis, Georgia Institute of Technology, 2006.

[5] C.T. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, 1999.

[6] Todd D. Murphey. Kinematic reductions for uncertain mechanical contact. *Robotica*, 2007.

[7] R.M. Murray, Z. Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[8] T. H. Tran, N. M. Kwok, S. Scheding, and Q. P. Ha. Dynamic modelling of wheel-terrain interaction of a ugv. *IEEE Conference on Automation Science and Engineering*, 2007.