International Journal of Control

ISSN: 0020-7179 (Print) 1366-5820 (Online) Journal homepage: https://www.tandfonline.com/loi/tcon20

Minimum-time speed optimisation over a fixed path

Thomas Lipp & Stephen Boyd

To cite this article: Thomas Lipp & Stephen Boyd (2014) Minimum-time speed optimisation over a fixed path, International Journal of Control, 87:6, 1297-1311, DOI: 10.1080/00207179.2013.875224

To link to this article: https://doi.org/10.1080/00207179.2013.875224

Accepted author version posted online: 17 Dec 2013.
Published online: 05 Feb 2014.

International Journal of Control, 2014
Vol. 87, No. 6, 1297–1311, http://dx.doi.org/10.1080/00207179.2013.875224

Taylor & Francis
Taylor & Francis Group

Minimum-time speed optimisation over a fixed path

Thomas Lipp[a,*] and Stephen Boyd[b]

[a]*Mechanical Engineering, Stanford University, Stanford, CA, USA;* [b]*Electrical Engineering, Stanford University, Stanford, CA, USA*

(*Received 6 June 2013; accepted 9 December 2013*)

In this paper we investigate the problem of optimising the speed of a vehicle over a fixed path for minimum time traversal. We utilise a change of variables that has been known since the 1980s, although the resulting convexity of the problem was not noted until recently. The contributions of this paper are three fold. First, we extend the convexification of the problem to a more general framework. Second, we identify a wide range of vehicle models and constraints which can be included in this expanded framework. Third, we develop and implement an algorithm that allows these problems to be solved in real time, on embedded systems, with a high degree of accuracy.

Keywords: minimum-time trajectory generation; optimal speed control; convex optimisation; interior point method; embedded control

1. Introduction

Interest in calculating minimum-time trajectories dates back at least to 1696 when Johann Bernoulli posed the Brachistochrone Problem in *Acta Eruditorum*. Although we may no longer be greatly concerned with the fastest path for a frictionless ball falling along a curve, faster trajectories can allow for the winning of races, interception of enemies, rapid surveillance of an area or getting aid more quickly to those who need it. Just a few of the applications that have desired minimum-time trajectories are: aircraft climbing (Bryson & Denham, 1962) and turning (Hedrick & Bryson, 1972), manipulator paths for robotics (Bobrow, Dubowsky, & Gibson, 1985), point-to-point travel for automated ground vehicles (Kalmár-Nagy, D'Andrea, & Ganguly, 2004; Pin & Vasseur, 1990) and air vehicles (Hehn & D'Andrea, 2011), optimal tracks for cars (Casanova, 2000; Hendrikx, Meijlink, & Kriens, 1996; Velenis & Tsiotras, 2005), paths through obstacles for automated vehicles (Donald, Xavier, Canny, & Reif, 1993; LaValle & Kuffner, 2001) and virtual gaming (Cardamone, Loiacono, Lanzi, & Bardelli, 2010).

In this paper, we focus on a subset of trajectory generation problems in which a specific path has already been determined. The problem we then solve is to find an achievable speed profile to produce the minimum time traversal of that path, given the dynamics of the vehicle. These problems primarily arise in two ways. First, the path may be completely determined by the problem. A vehicle may need to adhere closely to a pre-specified path to avoid obstacles in a highly cluttered environment, to avoid collisions by following a specified flight path or to perform a precise task as in the case of manufacturing. In these applications, there may be little room to modify the path for improved performance. Second, a higher level algorithm may have produced a series of collision-free paths that accomplish a task, which now need to be evaluated. A final path is then selected based on the minimum time required for traversal. This second scenario often occurs because the general minimum-time-trajectory problem is quite difficult and in some environments even finding a feasible path can be challenging. Therefore, the problem is split into two components where a higher level algorithm, like some discussed in Section 2, may be run to find collision-free paths through obstacles, while an algorithm like the one we develop is used to evaluate the paths to determine if they are feasible, given the vehicle dynamics, and the speed profile and control inputs necessary to traverse them in minimum time.

2. Previous work

2.1 *General minimum-time trajectories*

As the field of trajectory generation is a massive and long-lived field, we cannot mention all approaches but will touch on a few of the more common approaches to solving the minimum-time-trajectory problem used over the years. Since the minimum-time-trajectory problem is an optimal control problem, many of the same approaches developed for general optimal control are applied to the minimum-time problem. One of the oldest methods which arose from initial solutions to the Brachistochrone Curve problem was the calculus of variations which is still used on occasions (Bryson & Denham, 1962) and is the basis for many of the

* Corresponding author. Email: tlipp@stanford.edu

methods mentioned. Trajectory generation is also tackled from a nonlinear programming perspective with its direct methods of discretisation and multiple shooting (Betts, 1998, 2001; von Stryk & Bulirsch, 1997) and indirect methods (Bertolazzi, Biral, & Da Lio, 2005, 2007). Many of these nonlinear optimisation formulations can be approached with the ACADO package (Houska, Ferreau, & Diehl, 2011). Evolving from the nonlinear programming approaches is the pseudospectral methods approach in which Legendre, or similar, polynomials are used as basis functions to find solutions (Elnagar, Kazemi, & Razzaghi, 1995). This approach is supported by the GPOPS (General Pseudospectral Optimal Control Software) package (Darby, Hager, & Rao, 2011; Garg et al., 2011; Rao et al., 2010). Although many of these methods are promising, they are often hampered by speed and suffer under the constraint of exact path following.

There are also what we shall call graph methods that discretise the entire configuration space to create nodes and define edges as a minimum-time path between nodes. The algorithms then perform a random search through this graph in the hopes of finding a feasible, and then minimal, path. These approaches include potential field methods (Barraquand, Langlois, & Latombe, 1992; Caselli, Reggiani, & Rocchi, 2001) probabilistic road maps (Amato & Wu, 1996) and more general graph-based approaches (Donald & Xavier, 1989; Donald et al., 1993; Sahar & Hollerbach, 1985; Shiller & Dubowsky, 1989). Of course, as configuration spaces get larger, these methods become impractical, and for many vehicles, even calculating the minimum time between nodes is a substantial problem in its own right. A modification on these graph approaches, rapidly exploring random trees, removes the need to discretise initially, but is similar (Karaman & Frazzoli, 2011; LaValle & Kuffner, 2001). These methods are particularly popular when trying to find paths through environments containing many obstacles, and with vehicle dynamics removed, are well suited for finding collision-free paths. In order to evaluate and compare these paths, we need to know the minimum time required to traverse the path, which is the focus of this paper.

2.2 *Fixed-path minimum-time trajectories*

Our work builds upon a transformation known since at least 1985 in which the path is reparameterised by a single variable representing the distance travelled along the path. Once the path is reparameterised in this manner, using the vehicle dynamics, valid velocity regions are determined. The goal then becomes to find valid velocity profiles along the newly parameterised paths that stay below the maximum-velocity regions (Bobrow et al., 1985; Constantinescu & Croft, 2000; Dubowsky, Norris, & Shiller, 1986; Pfeiffer & Johanni, 1987; Shin & McKay, 1985). Another approach to the more general trajectory problem was to modify this method to allow modest changes to the trajectory (Bayer & Hauser, 2012; Shiller, 1994; Shiller & Dubowsky, 1989), but we will not be exploring those extensions. More recent research in this approach (sometimes called the Bobrow method) has focused on improving the robustness of the approach, and often merging it with the global search methods mentioned earlier, or allowing for more complex dynamics (Hauser & Saccon, 2006; Kunz & Stilman, 2012; Pham, 2013; Pham, Caron, & Nakamura, 2013; Zhao & Tsiotras, 2013). While the early implementations focused on iterative and geometric methods to find these velocity profiles, Verscheure, Demeulenaere, Swevers, Schutter, and Diehl (2009a) observed that under this transformation, the velocity profile for a standard six degree of freedom robotic manipulator is convex (see also Verscheure, Diehl, Schutter, & Swevers, 2009b). This allowed the problem to be approached with the many tools that have been developed for convex optimisation (Boyd & Vandenberghe, 2004), greatly improving the efficiency, reliability and flexibility of this formulation, and providing guarantees on optimality trivially. The convexity of the problem also allows for simple verification of the feasibility of the desired path. Variations and extensions on this method have allowed the path to vary, although this addition loses the benefits gained from the convexity of the problem (Dinh & Diehl, 2009). Although we will not be discussing it, alternative approaches have been looked at, rather than convex formulations, differentially flat formulations, which allow a broader class of vehicles to be modelled, but again lose the benefits that can be leveraged from convexity (Faiz, Agrawal, & Murray, 2001; Faulwasser, Hagenmeyer, & Findeisen, 2011).

2.3 *Overview*

In Section 3, we will review the problem formulation and formally state the problem we address. We extend the permissible constraints, giving a general form for constraints, rather than explicitly enumerating specific constraints. In Section 4, we introduce a series of examples that show how our model can be adapted to a wide range of vehicles and situations. Although some work has already been performed exploring how this model can be applied to vehicles beyond the robotic manipulator (Dinh & Diehl, 2009), our generalisation allows for new classes of vehicles and better models to be considered. We will investigate the simplest vehicle, the holonomic spacecraft, with modifications that would cover a ducted fan. We then explore a number of friction circle models for cars including front wheel drive, aerodynamic drag, a banked turn and independently actuated wheels. Finally, we conclude the section with a simple aircraft model. None of these models are possible in the torque-constrained model presented in Verscheure et al. (2009a). In Section 5, we demonstrate that the model is indeed convex. We then shift our focus in Section 6 to the development of an efficient algorithm for solving this

problem. In Section 6.1, we discretise the model making several observations to improve the behaviour of the solution. In Section 6.2, we discuss methods for solving the discretised problem and introduce a standard log barrier interior point solver that exploits structure. We then present results in Section 7 validating the algorithm, and demonstrating its improvement over existing solvers. We demonstrate solve times that make the algorithm feasible for implementation on embedded systems in real time.

3. Problem formulation

We address the problem of finding the control inputs that allow a vehicle to traverse a specified path in minimum time.

3.1 *Dynamics*

We consider vehicles that are represented by their generalised position or configuration vector $q \in \mathbf{R}^p$, which includes the position and orientation, and its time derivative $\dot{q} \in \mathbf{R}^p$, where p is the degree of freedom in the configuration. The vehicles can move under the power of their own actuators which take control inputs $u \in \mathbf{R}^r$.

In this paper, we concern ourselves exclusively with vehicles that have dynamics of the second-order form

$$R(q)u = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + d(q), \tag{1}$$

where $R : \mathbf{R}^p \to \mathbf{R}^{p\times r}$ is the control matrix, $M : \mathbf{R}^p \to \mathbf{S}^p_{++}$ is the mass matrix, $\mathbf{S}^p_{++}$ is the set of symmetric positive-definite matrices in $\mathbf{R}^{p\times p}$, $C : \mathbf{R}^{2p} \to \mathbf{R}^{p\times p}$ is the centrifugal matrix and is linear in $\dot{q}$ and $d : \mathbf{R}^p \to \mathbf{R}^p$ is the force dependent on the configuration.

3.2 *Control constraints*

The vehicle is further defined by constraints on its control inputs u which take the form

$$(\dot{q}^2, \ddot{q}, u) \in \mathcal{C}(q), \tag{2}$$

where $\mathcal{C}(q) \subseteq \mathbf{R}^{p\times p\times r}$ is a set valued mapping to convex sets. We use $\dot{q}^2$ to represent the element-wise square of $\dot{q}$. Previous formulations of this problem have only allowed for bounds on the elements of u dependent on q. This formulation allows the admissible control inputs to be coupled to each other, the configuration, acceleration and square of the velocity of the vehicle. Note that this representation means that the set of admissible $(q, \dot{q}^2, \ddot{q}, u)$ need not be convex. Although we will usually think of these as constraints on the control inputs, this formulation also allows for constraints on the acceleration, and the square of the velocity related to the position. For example, $\mathcal{C}$ could include speed limits in certain regions of the configuration space.

3.3 *Path*

The path to be traversed is defined as $s : [0, 1] \to \mathbf{R}^p$ in the generalised position space. We say a vehicle moves along path s when

$$s(\theta(t)) = q(t), \quad t \in [0, T], \tag{3}$$

where θ: $[0, T] \to [0, 1]$ satisfies $\theta(0) = 0, \theta(T) = 1, \dot{\theta} \geq 0$, and T is the time at which the vehicle reaches the end of the path. Thus, the function θ provides the speed of the vehicle along the path, specifically

$$\begin{aligned} \dot{q}(t) &= s'(\theta(t))\dot{\theta}(t), \\ \ddot{q}(t) &= s'(\theta(t))\ddot{\theta}(t) + s''(\theta(t))\dot{\theta}(t)^2, \end{aligned} \tag{4}$$

where $s' = \frac{ds}{d\theta}$ and $s'' = \frac{d^2s}{d\theta^2}$. Throughout this paper, we will continue to use $'$ to represent derivatives with respect to θ.

3.4 *Problem statement*

We can now represent our problem as

$$\begin{aligned} &\text{minimise} \quad T \\ &\text{subject to} \quad R(q(t))u(t) = M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) \\ &\qquad\qquad + d(q(t)), \quad t \in [0, T], \\ &\qquad\quad s(\theta(t)) = q(t), \quad t \in [0, T], \\ &\qquad\quad (\dot{q}(t)^2, \ddot{q}(t), u(t)) \in \mathcal{C}(q(t)), \quad t \in [0, T], \end{aligned} \tag{5}$$

where R, M, C, d, s and $\mathcal{C}$ are problem data and the functions θ and u are the optimisation parameters.

Although the dynamics model presented in (1) and problem formulation (5) may seem limiting, they include a wide range of vehicles and constraints. In Section 4, we will discuss robots, space vehicles, various car models and aircraft.

4. Examples

In this section, we will present a series of vehicle models which conform to the formulation presented in Section 3. Although some aspects of these models could have been handled in previous formulations, all of these models involve coupling between control inputs and could therefore not be handled in their entirety. The examples are by no means exhaustive, but were chosen to highlight aspects of this formulation and provide insight into how other vehicles and situations can be included in this formulation.

4.1 *Robotics*

The application of this model to robotics has already been well explored in Verscheure et al. (2009a, Verscheure et al. 2009b) which focused on a six degree of freedom robot. The dynamics formulation (1) is the standard form for robotics and will therefore accommodate almost all robotic manipulators (Asada & Slotine, 1986; Lewis, Dawson, & Abdallah, 2004). The model can handle many intricate

details of robotics, including centrifugal and Coriolis terms without any modification. In Verscheure et al. (2009a), they discuss how Coulomb friction can be included in the model, which is simple, but will not be covered here. We will forgo further discussion of robotic manipulators.

4.2 *Space vehicle*

A spacecraft can be modelled as a point mass that can be subjected to a force in any direction and gravity. This model corresponds to a vehicle with a primary thruster that can be oriented in any direction by the vehicle's attitude control. In this way, we have abstracted the details of actuation to produce a point-mass model with an allowable force envelope. In addition to the canonical spacecraft, the point-mass model and force envelope can cover most vehicles for which the details of the dynamics and actuators are abstracted. This is done for quadrocoptors in Hehn and D'Andrea (2011) and for ground vehicles in Kalmár-Nagy et al. (2004).

We take as the vehicle configuration its position in space, that is, $q = (x, y, z)$, and allow force inputs $u = (f_x, f_y, f_z)$ in the Newtonian frame. Therefore, $p = 3$ and $r = 3$. Although we are working in three dimensions, it is trivial to see that a similar model could be applied to a vehicle operating in two dimensions.

We assume that the spacecraft is near some gravitational body which is exerting a force in the negative z direction resulting in dynamics

$$M = mI, \quad C = 0, \quad d = -gme_3, \quad R = I, \tag{6}$$

where m is the mass of the vehicle and g is the acceleration due to gravity. The gravitational force d could be removed, or have its direction or magnitude changed without any violation of the problem formulation. Furthermore, the force could vary with the vehicle's position q for a more accurate model.

For the spacecraft model, the set $\mathcal{C}$ is invariant with respect to q and can be represented as

$$\mathcal{C} = \left\{(\dot{q}^2, \ddot{q}, u) \mid \|u\|_2 \leq u_{\max}\right\}, \tag{7}$$

where $u_{\max}$ is the maximum force the primary thruster can exert. In this example, we are able to represent the forces in the Newtonian frame because the vehicle is holonomic, thereby simplifying computation.

A spacecraft may have additional constraints such as a minimum force that can be applied; a thruster must produce some minimum force to operate. Such a constraint is non-convex and does not fit in our formulation, but Blackmore and Açikmeşe (2011) suggest methods for including such constraints.

4.3 *Space vehicle with vectored thrust*

One modification to the space-vehicle model is to limit the thrust to a cone aligned with the orientation of the vehicle. This limitation corresponds to the attitude control only being able to make modest modification to the orientation of the vehicle. Such a model would also be appropriate for handling ducted fans.

Rather than adding the orientation to our configuration parameters, we will instead introduce a function, $P : \mathbf{R}^3 \to \mathbf{R}^3$, which maps the position of the vehicle to an orientation vector. For ease of representation, we will assume that the position uniquely determines the orientation, but it is simple to allow the same position to correspond to multiple orientations at different points along the trajectory. The dynamics of the system are the same as in (6), but now the constraint set $\mathcal{C}$ has an additional constraint

$$\mathcal{C} = \left\{(\dot{q}^2, \ddot{q}, u) \mid \|u\|_2 \leq u_{\max}, \quad \|u\|_2 \cos(\phi) \leq P(q)^T u\right\}, \tag{8}$$

where ϕ is the maximum deviation angle from the primary orientation that the attitude control can achieve.

4.4 *Friction circle car model*

Whereas the spaceship had three dimensions, our first car model has only two. Although the model in (1) has been applied to a car in Dinh and Diehl (2009), we will use a different model based on the friction circle which has coupling between forces. We provide several variations on this model in order to provide insight into the ways various constraints, models and environmental factors can be incorporated into these models.

For our vehicle model, we will use a standard friction circle model with a limit on the available force in the orientation of the vehicle. The friction circle model derives from the observation that all forces that are applied to the car must transmit through the contact surface between the tyres and the road. As such, the maximum force that can be applied to the car, assuming a sufficiently powerful engine, is limited by the coefficient of friction between the tyres and the road. If the car attempts to produce more force than this limit, the tyres will slip. This will produce less force as the sliding coefficient of friction is less than the static coefficient of friction. Thus, the maximum achievable force is the force required to overcome static friction. This basic friction circle model assumes that by manipulating the steering wheel, accelerating and breaking, force can be produced in any direction desired (Milliken & Milliken, 1995; Rice & Alianello, 1970). With only these limitations we have created a two-dimensional space vehicle. We add a further constraint to model a front wheel drive vehicle. In this configuration, the force in the direction of motion is limited to the force required to overcome the static friction of the two front tyres, as the rear tyres provide no drive force. The front tyres typically support slightly more than 50% of the total mass of the car.

As in the spaceship model, we take as the vehicle configuration its position in space: $q = (x, y)$. The allowable force inputs are now provided relative to the orientation of

the car as $u = (f_{\text{long}}, f_{\text{lat}})$, where f_{long} is the longitudinal force in the direction the car is oriented and f_{lat} is the force perpendicular to the orientation of the car. Therefore, $p = 2$ and $r = 2$. The dynamics of the car are quite similar to those in (6) except u is now represented in the frame of the vehicle, so R is a rotation matrix:

$$M = mI, \quad C = 0, \quad d = 0,$$
$$R = \begin{bmatrix} \cos(\phi(q)) & -\sin(\phi(q)) \\ \sin(\phi(q)) & \cos(\phi(q)) \end{bmatrix}, \tag{9}$$

where $\phi : \mathbf{R}^2 \to \mathbf{R}$ maps the position of the vehicle to an orientation angle in the global frame, similar to the vectored thrust approach. Thus, R maps u from the vehicle frame back into the global frame. A typical implementation of ϕ will assume that the vehicle is always facing the direction of its instantaneous velocity, which is known from the path s. Although this is generally true at low speeds, at high speeds, the side slip angle (the angle between the heading and velocity of the vehicle) may become substantial. We will ignore these effects.

The constraint set $\mathcal{C}$ includes two constraints,

$$\mathcal{C} = \left\{ (\dot{q}^2, \ddot{q}, u) \mid \|u\|_2 \leq \mu_s F_N, \quad f_{\text{long}} \leq W_f \mu_s F_N \right\}, \tag{10}$$

where μ_s is the static coefficient of friction between the tyres and the road, F_N is the normal force of the vehicle, which on a flat surface is mg, where g is the acceleration due to gravity, and W_f is the percentage of vehicle weight on the front tyres. The first constraint represents the limits of the friction circle, while the second constraint encodes the acceleration limits from front wheel drive. Our constraint set, $\mathcal{C}$, is convex as it is the intersection of two convex sets; norms are convex, and our upper bounds are fixed.

We could have represented u in the Newtonian frame in which case the dynamics would have been the same as (6) and the rotations would have been incorporated in $\mathcal{C}$.

4.5 *Vehicle with aerodynamic drag*

To demonstrate the utility of the C term in the dynamics, we will present a model for our vehicle which incorporates aerodynamic drag. The drag, D, on the vehicle is

$$D = \frac{1}{2} \rho A C_D v^2,$$

where C_D is the coefficient of drag, ρ is the air density, v is the speed of the vehicle and A is the reference area (the front area of the vehicle). We have assumed that the vehicle is always oriented in the direction it is travelling, so we do not need to be concerned about the way the surface area varies based on the orientation of the vehicle. We then have dynamics

$$M = mI, \quad C = \begin{bmatrix} -\frac{1}{2}\rho A C_D \dot{x} & 0 \\ 0 & -\frac{1}{2}\rho A C_D \dot{y} \end{bmatrix}, \quad d = 0,$$
$$R = \begin{bmatrix} \cos(\phi(q)) & -\sin(\phi(q)) \\ \sin(\phi(q)) & \cos(\phi(q)) \end{bmatrix}, \tag{11}$$

and the same constraint set $\mathcal{C}$ as in the earlier examples.

4.6 *Banked turn*

We now consider an example of a vehicle traversing a banked curve. When a vehicle is on a banked road, there are two effects which take place. First, there is a force applied in the direction of descent of the road caused by gravity acting on the mass of the vehicle. Second, the normal force from the road decreases, and thus the available friction force at the tyres is reduced.

In order to enforce the constraint that the vehicle is on the road, rather than floating above it, we will take $q = (x, y)$ as before, where the height of the vehicle can be determined from its position. We will take $u = (f_{\text{long}}, f_{\text{lat}})$ which are still in the vehicle frame, but due to the slope of the road, may no longer be in the (x, y) plane.

The vehicle has dynamics

$$M = mI, \quad C = 0,$$
$$R = \begin{bmatrix} -\cos(\alpha(q)) & -\sin(\alpha(q)) \\ \sin(\alpha(q)) & -\cos(\alpha(q)) \end{bmatrix} \begin{bmatrix} \cos(\phi(q)) & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(\beta(q)) & \sin(\beta(q)) \\ -\sin(\beta(q)) & \cos(\beta(q)) \end{bmatrix}, \tag{12}$$
$$d = \begin{bmatrix} \cos(\alpha(q)) & \sin(\alpha(q)) \\ -\sin(\alpha(q)) & \cos(\alpha(q)) \end{bmatrix} \begin{bmatrix} \cos(\phi(q)) & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} mg \sin(\phi(q)) \\ 0 \end{bmatrix},$$

where $\beta : \mathbf{R}^2 \to \mathbf{R}$ is the angle between the orientation of the vehicle and the axis of maximum inclination (the line in the tangent plane to the road with the maximum vertical slope), $\phi : \mathbf{R}^2 \to \mathbf{R}$ is the angle of inclination and $\alpha : \mathbf{R}^2 \to \mathbf{R}$ is the angle between the axis of inclination and the x-axis in the (x, y) plane. To understand the formulation of R, we first transform the forces from the vehicle frame to a frame aligned with the axis of inclination (rotation by β). We then project those forces into the Newtonian (x, y) plane (projection by ϕ) and represent those forces in terms of their components in the x and y directions (rotation by α). The vector d is the force due to gravity from the inclination of the road represented along the x- and y-axes. The functions α, β and ϕ depend only on the state, q.

The constraint set $\mathcal{C}$ is similar to that seen in the friction circle approach, except for the aforementioned change in

the available normal force:

$$\mathcal{C} = \big\{(\dot{q}^2, \ddot{q}, u) \mid \|(f_{\text{long}}, f_{\text{lat}})\|_2 \leq \mu_s(q) mg \cos(\phi(q)), \quad f_{\text{long}} \leq W_f \mu_s(q) mg \cos(\phi(q))\big\}. \tag{13}$$

The constraint set as a function of q is still convex, because, as in the variable surface example, for a given q the upper bounds on the norms are all constant over $\dot{q}^2$, $\ddot{q}$ and u.

4.7 *Multiwheel car model*

In all models considered so far, orientation has either been irrelevant due to the holonomic nature of the vehicle (as with the spacecraft) or assumed in the direction of travel. Achieving the necessary orientation has been excluded from the dynamics. In this example, we include the orientation of our vehicle as one of the configuration states and consider a vehicle with four independently actuated wheels. As before, the orientation of the vehicle will be known *a priori*, but unlike the earlier examples, the control inputs chosen must achieve the proper orientation, not just the position of the vehicle.

The vehicle has a generalised position, $q = (x, y, \theta)$, where (x, y) are the Newtonian positions of the vehicle and θ is the orientation of the vehicle with respect to the x-axis. The vehicle has control inputs $u = (f_{xfl}, f_{yfl}, f_{xfr}, f_{yfr}, f_{xrl}, f_{ylr}, f_{xrr}, f_{yrr})$, where the x and y subscripts represent the direction of the force, *fl* means the force comes from the front (f) left (l) wheel and *rr* means the force comes from the rear (r) right (r) wheel. We will assume that the weight is evenly distributed between the right-hand and left-hand side of the vehicle, and that there is no weight transfer due to manoeuvres. This model has $p = 3$ and $r = 8$.

The vehicle has dynamics

$$M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}, \quad C = 0, \quad d = 0,$$
$$R = \begin{bmatrix} \mathrm{c}(\theta) & -\mathrm{s}(\theta) & \mathrm{c}(\theta) & -\mathrm{s}(\theta) & \mathrm{c}(\theta) & -\mathrm{s}(\theta) & \mathrm{c}(\theta) & -\mathrm{s}(\theta) \\ \mathrm{s}(\theta) & \mathrm{c}(\theta) & \mathrm{s}(\theta) & \mathrm{c}(\theta) & \mathrm{s}(\theta) & \mathrm{c}(\theta) & \mathrm{s}(\theta) & \mathrm{c}(\theta) \\ (1 - W_f)L & -w/2 & (1 - W_f)L & w/2 & -W_f L & -w/2 & -W_f L & w/2 \end{bmatrix}, \tag{14}$$

where m is the mass of the vehicle, J is the moment of inertia, $\mathrm{c}(\theta) = \cos(\theta)$, $\mathrm{s}(\theta) = \sin(\theta)$, W_f is the percentage of weight on the front tyres, L is the length of the vehicle and w is the width of the vehicle

The constraint set consists of limits to the force that can be applied by the front wheel,

$$\|(f_{xfi}, f_{yfi})\|_2 \leq \frac{W_f mg}{2} \quad i = l, r, \tag{15}$$

and that can be applied by the rear wheel,

$$\|(f_{xri}, f_{yri})\|_2 \leq \frac{(1 - W_f)mg}{2} \quad i = l, r, \tag{16}$$

where W_f is the percentage of weight supported by the front wheels, and the factor of $\frac{1}{2}$ appears because the weight is supported evenly by the left-hand and the right-hand wheels. This gives the constraint set

$$\mathcal{C} = \big\{(\dot{q}^2, \ddot{q}, u) \mid (15), (16)\big\}. \tag{17}$$

4.8 *Aircraft model*

In this example, we consider a simple aircraft model in two dimensions to demonstrate the advantages of this model allowing coupling, not just between control inputs as already shown, but between control inputs and the square of velocity. We will take as the state of our system $q = (x, h)$, where x is the lateral displacement of the vehicle and h is the altitude of the vehicle. We will take as our control inputs $u = (L, D, T)$, where L is the lift, D is the drag (both induced and from spoilers) and T is the thrust. Therefore, $p = 2$ and $r = 3$. As in our vectored thrust model, we assume a nominal angle of attack that is dependent on the lateral displacement, $\alpha : \mathbf{R} \to \mathbf{R}$, and allow the angle of attack to vary by an amount α_δ from the nominal angle of attack. However, we assume that the direction of the lift and drag forces are relative to the nominal angle of attack. The dynamics of our system are

$$M = mI, \quad C = 0,$$
$$R = \begin{bmatrix} -\sin(\alpha(x)) & -\cos(\alpha(x)) & \cos(\alpha(x) + \alpha_T) \\ \cos(\alpha(x)) & -\sin(\alpha(x)) & \sin(\alpha(x) + \alpha_T) \end{bmatrix},$$
$$d = -mge_2, \tag{18}$$

where m is the mass of the plane, α_T is the fixed angle of the thrust and g is the acceleration due to gravity. Although we have made the angle of attack dependent purely on the lateral displacement, we could have made it a function of q, that is, both x and h.

To determine our constraint set, we review some basic aerodynamics. The lift produced by the plane is

$$L = \frac{1}{2}\rho(h) S C_L v^2,$$

where L is the lift, $\rho : \mathbf{R} \to \mathbf{R}$ is the air density dependent on the altitude, S is the area of the lifting surface, v is the true airspeed (which we will simply consider to be the speed of the vehicle, $\|\dot{q}\|_2$, meaning, there is no wind) and C_L is the coefficient of lift. Although we have represented ρ as a function of h, it could also be represented as a function of q. The coefficient of lift can be approximated in the linear region as

$$C_L = a_0(\alpha - a_L),$$

where α is the angle of attack, and a_0 and a_L are coefficients representing the lifting curve. The lift that can be produced is therefore limited by the variation that we permit in the deviation from the nominal angle,

$$\begin{aligned}\frac{1}{2}\rho(h)S\|\dot{q}^2\|_1 a_0(\alpha(x) - \alpha_\delta - a_L) \le L \\ \le \frac{1}{2}\rho(h)S\|\dot{q}^2\|_1 a_0(\min(\alpha(x) + \alpha_\delta, \alpha_{\text{stall}}) - a_L),\end{aligned} \quad (19)$$

where α_{stall} is the angle of attack at which the linear model fails and the vehicle starts to stall. Although the lower bound is clearly convex, it is not immediately evident that the upper bound is convex. First, we observe that the minimum function is only operating on q and therefore its convexity is irrelevant. We next observe that $\|\dot{q}^2\|_1$ is linear in $\dot{q}^2$ since it is simply a sum of its components $\|\dot{q}^2\|_1 = x^2 + h^2 = v^2$.

In order to make our constraints set, $\mathcal{C}$, convex, we assume that the maximum drag that can be produced by the spoilers is greater than the induced drag from the lift, and that the spoiler drag is independent of the induced drag and lift. We will then constrain the drag to be less than this value. This means that in regions with high induced drag, it may be possible to achieve greater drag than we permit. From aerodynamics, we know that the induced drag is

$$D_{\text{ind}} = \frac{1}{2}\rho S C_{D_i} v^2,$$

where

$$C_{D_i} = C_{D_0} + \frac{C_L^2}{\pi e A R},$$

where C_{D_0} is the zero-lift drag coefficient, e is the Oswald efficiency number and AR is the aspect ratio of the wing. We can therefore define the induced drag as

$$D_{\text{ind}} = \frac{1}{2}\rho(h)SC_{D_0}\|\dot{q}^2\|_1 + \frac{2L^2}{\rho(h)S\pi e A R\|\dot{q}^2\|_1}.$$

However, this constraint is not convex, so we relax it by saying that we produce at least D_{ind}, but could produce up to $D_{\max}(\alpha(x), h)$ by engaging spoilers and other control surfaces where $D_{\max} : \mathbf{R}^2 \to \mathbf{R}$ is the maximum drag that can be produced by control surfaces independent of lift. We can now represent the drag produced as

$$\begin{aligned}\frac{1}{2}\rho(h)SC_{D_0}\|\dot{q}^2\|_1 + \frac{2L^2}{\rho(h)S\pi e A R\|\dot{q}^2\|_1} \\ \le D \le D_{\max}(\alpha(x), h).\end{aligned} \quad (20)$$

The first term on the left-hand side is linear in $\dot{q}^2$ as already discussed and is therefore convex. The second term on the left-hand side is in the form of a quadratic term (L^2) over a linear term ($\rho(h)S\pi e A R\|\dot{q}^2\|_1$) which is known to be convex (Boyd & Vandenberghe, 2004, , Section 3.1.5). As before, the upper bound is fixed by q.

We can therefore define the constraint set as

$$\mathcal{C} = \left\{(\dot{q}^2, \ddot{q}, u) \mid (19), (20), 0 \le T \le T_{\max}\right\}, \quad (21)$$

where $T_{\max}$ is the maximum thrust that can be produced, assumed independent of speed for simplification, although more complex models would work. Our constraint set is the intersection of three convex sets, and is therefore convex.

5. Convexification

Having seen the wide variety of models that fit this formulation, we now apply the change of variables used in Verscheure et al. (2009b) to reformulate (5), and thereby all of the examples, as convex problems. Although much of this analysis is performed in Verscheure et al. (2009a), we repeat it here for completeness. We represent the dynamics of (1) in terms of θ by applying (3) and (4) to produce

$$\tilde{R}(\theta)u = \tilde{m}(\theta)\ddot{\theta} + \tilde{c}(\theta)\dot{\theta}^2 + \tilde{d}(\theta), \quad (22)$$

where

$$\begin{aligned}\tilde{R}(\theta) &= R(s(\theta)), \\ \tilde{m}(\theta) &= M(s(\theta))s'(\theta), \\ \tilde{c}(\theta) &= M(s(\theta))s''(\theta) + C(s(\theta))s'^2(\theta), \\ \tilde{d}(\theta) &= d(s(\theta)).\end{aligned} \quad (23)$$

We next introduce two new functions,

$$a(\theta) = \ddot{\theta}, \quad b(\theta) = \dot{\theta}^2,$$

which are related by

$$\dot{b}(\theta) = b'(\theta)\dot{\theta} = \frac{d(\dot{\theta}^2)}{dt} = 2\ddot{\theta}\dot{\theta} = 2a(\theta)\dot{\theta}$$

or more simply

$$b'(\theta) = 2a(\theta). \quad (24)$$

Our objective function can be represented by

$$T = \int_0^T 1 d\, t = \int_{\theta(0)}^{\theta(T)} \dot{\theta}^{-1}\, d\theta = \int_0^1 \dot{\theta}^{-1} d\theta = \int_0^1 b(\theta)^{-1/2}\, d\theta. \tag{25}$$

Our formulation (5) can now be represented as

$$\begin{aligned} \text{minimise} \quad & \int_0^1 b(\theta)^{-1/2}\, d\theta \\ \text{subject to} \quad & \tilde{R}(\theta)u(\theta) = \tilde{m}(\theta)a(\theta) + \tilde{c}(\theta)b(\theta) + \tilde{d}(\theta), \\ & \qquad\qquad\qquad\qquad\qquad\qquad \theta \in [0, 1], \\ & b'(\theta) = 2a(\theta), \quad \theta \in [0, 1], \\ & (a(\theta), b(\theta), u(\theta)) \in \tilde{\mathcal{C}}_\theta, \quad \theta \in [0, 1], \end{aligned} \tag{26}$$

where

$$\tilde{\mathcal{C}}_\theta = \big\{(a(\theta), b(\theta), u(\theta)) \mid (s'(\theta)^2 b(\theta), s'(\theta)a(\theta) + s''(\theta)b(\theta), u(\theta)) \in C(s(\theta))\big\}.$$

The optimisation parameters are the functions a, b, and u while everything else is problem data. Our objective function is the integral of a negative power function and is therefore convex, as the integral of a convex function is convex. The dynamics constraint is affine in a, b, and u and is therefore convex. The relation between a and b is convex as the derivative is a linear operator. Finally, $\tilde{\mathcal{C}}_\theta$ is an affine transformation of a convex set and is therefore convex. We have now formulated the problem as an infinite dimensional convex optimisation problem. Therefore, locally optimal solutions to this problem are guaranteed to be globally optimal. Provided that $\tilde{\mathcal{C}}_\theta$ is numerically tractable, this problem can be easily solved using discretisation techniques.

6. Algorithm development

In the previous sections, our focus was on the general class of problems outlined in Section 3. In the subsequent sections, our attention shifts to our specific implementation, with a focus on developing an efficient algorithm for solving (26) that will rely on its convexity. In Section 6.1, we will discuss the impact of various discretisation schemes which will motivate the discretisation scheme used by our algorithm. Then, in Section 6.2, we will introduce a primal interior point method which exploits structure. The result is Algorithm **6.1**.

6.1 *Discretisation*

The problem stated in (26) can be readily solved using well-established methods of discretisation to address the differential constraints and continuous objective function. To create a finite-dimensional problem, discretise $a(\theta)$, $b(\theta)$ and $u(\theta)$ at the same $n + 1$ points in θ ranging from 0 to 1, replace the integral (25) with a sum and replace the derivative (24) with a finite difference approximation. Many discretisations of this problem, with an appropriately fine mesh, will find the solution, but we present some techniques we use to improve the accuracy of the results.

6.1.1 *Exact discretisation of the objective function*

Since it is often desirable to find a minimum-time trajectory, given an initial starting and or ending velocity, we will choose a discretisation that includes both $\theta = 0$ and $\theta = 1$. If either $b(0)$ or $b(1)$ is zero, we cannot evaluate the integrand at this value. We will resolve this issue by exactly evaluating the objective function as in Verscheure et al. (2009a). We will make the assumption that on the interval between two consecutive discretisation points, θ_i and θ_{i+1}, $a(\theta)$ is constant. From (24) we know, under this assumption, that $b'(\theta)$ is constant across the interval and thus

$$b(\theta) = b_i + (\theta - \theta_i)\left(\frac{b_{i+1} - b_i}{\theta_{i+1} - \theta_i}\right), \quad \theta \in [\theta_i, \theta_{i+1}], \tag{27}$$

where θ_i and θ_{i+1} are consecutive discretisation points, $b_i = b(\theta_i)$ and $b_{i+1} = b(\theta_{i+1})$. If both θ_i and θ_{i+1} are zero, the path is not traversable as under the a constant constraint, the vehicle is never able to progress through that segment. Under these assumptions, we have

$$\int_{\theta_i}^{\theta_{i+1}} b(\theta)^{-1/2}\, d\theta = \frac{2(\theta_{i+1} - \theta_i)}{b_i^{1/2} + b_{i+1}^{1/2}}. \tag{28}$$

To derive (28), we first apply (27) to our objective function to get

$$\begin{aligned} &\int_{\theta_i}^{\theta_{i+1}} \left(b_i + (\theta - \theta_i)\left(\frac{b_{i+1} - b_i}{\theta_{i+1} - \theta_i}\right)\right)^{-1/2} d\theta \\ &\quad = \frac{2\left(b_i + (\theta - \theta_i)\left(\frac{b_{i+1} - b_i}{\theta_{i+1} - \theta_i}\right)\right)^{1/2}}{\left(\frac{b_{i+1} - b_i}{\theta_{i+1} - \theta_i}\right)}\Bigg|_{\theta_i}^{\theta_{i+1}}. \end{aligned}$$

Carrying out algebra now yields

$$\int_{\theta_i}^{\theta_{i+1}} b(\theta)^{-1/2}\, d\theta = \frac{2\big(b_{i+1}^{1/2} - b_i^{1/2}\big)}{\left(\frac{b_{i+1} - b_i}{\theta_{i+1} - \theta_i}\right)} = \frac{2(\theta_{i+1} - \theta_i)}{b_i^{1/2} + b_{i+1}^{1/2}}.$$

Thus, assuming a is constant across the interval, (25) is equivalent to

$$2\sum_{i=0}^{n-1}\left(\frac{\theta_{i+1}-\theta_i}{b_i^{1/2}+b_{i+1}^{1/2}}\right), \tag{29}$$

where we have discretised θ and b at $n+1$ points with $\theta_0 = 0$, $\theta_n = 1$ and $\theta_i < \theta_{i+1}$ for $i = 0, \ldots, n$. Note that once the discretisation is decided upon, the θ values are all known, while the b values are found during the optimisation.

6.1.2 *Enforcing differential constraints at the midpoints*

Although we evaluated $b(\theta)$ at the discretisation points θ_i, according to the assumption that $a(\theta)$ is constant across the interval, the value of $a(\theta)$ at θ_i is discontinuous. Therefore, we choose to evaluate our transformed dynamics constraint at the midpoint of the interval, such that $a_i = a\left(\frac{\theta_{i-1}+\theta_i}{2}\right)$ and $u_i = u\left(\frac{\theta_{i-1}+\theta_i}{2}\right)$. Since $a(\theta)$ is constant across the interval from (24), we have $b\left(\frac{\theta_{i-1}+\theta_i}{2}\right) = \frac{b_{i-1}+b_i}{2}$. Thus, our dynamics constraint becomes

$$\tilde{R}\left(\bar{\theta}_i\right)u_i = \tilde{m}\left(\bar{\theta}_i\right)a_i + \tilde{c}\left(\bar{\theta}_i\right)\frac{b_{i-1}+b_i}{2} + \tilde{d}\left(\bar{\theta}_i\right), \quad i = 1, \ldots, n, \tag{30}$$

where $\bar{\theta}_i = \left(\frac{\theta_{i-1}+\theta_i}{2}\right)$.

6.1.3 *Enforcing $\mathcal{C}$ constraints*

We will impose the $\mathcal{C}$ constraints at the discretisation points θ_i. This is different from the enforcement of the dynamics constraints so that maximum velocity constraints are possible. Consider a velocity constraint of the form $b(\theta) \leq f(\theta)$, for example, $f(\theta) = k$. If we enforced this constraint at $\bar{\theta}_i$, if $b_{i+1} = 0$ and $b_{i-1} = 0$, then $b_i = 2k$ would satisfy our speed constraint, which is clearly antithetical to the goal of the speed constraint. Although we could apply the $\mathcal{C}$ constraints in different ways depending on the nature and goal of the constraint, for uniformity and ease of implementation, we will enforce constraint $\mathcal{C}_{\theta_i}$ on b_i, u_i, a_i. We note that a and u are discontinuous at θ_i and we therefore choose to use the values of u_i and a_i to maintain the block structure which we will exploit in Section 6.2. Under this discretisation, the $b(\theta)$ will always satisfy the maximum constraint given earlier.

One will note that this formulation does not permit constraints in $\mathcal{C}$ to apply to b_0. We will resolve this issue by requiring that b_0, the initial velocity, be prescribed. The problem can be solved as easily without prescribing an initial velocity, but we will not be examining that case in this paper. We implement discretised constraints

$$(a_i, b_i, u_i) \in \tilde{\mathcal{C}}_{\theta_i}, \quad i = 1, \ldots, n \tag{31}$$

$$b_0 = \left(\frac{\|v_{\text{init}}\|_2 d\theta}{\|s(\theta_1) - s(\theta_0)\|_2}\right)^2, \tag{32}$$

where v_{init} is the generalised initial velocity desired. Note that to be feasible $v_{\text{init}} = \alpha s'(0)$, where $\alpha \in \mathbf{R}_{++}$ so we could also write constraint (32) as

$$b_0 = \alpha^2.$$

6.1.4 *Calculating derivatives of b*

We estimate the first derivatives of b with a second-order central difference. From here on we will assume that the discretisation points are evenly spaced, that is, $\theta_{i+1} - \theta_i = d\theta$ for all i. This does not mean that the $s(\theta_i)$ need be evenly distributed. In the case of $b'(\theta)$, since we have assumed $a(\theta)$ is constant, the second-order approximation is exact:

$$b_i - b_{i-1} = 2a_i d\theta, \quad i = 1, \ldots, n. \tag{33}$$

6.1.5 *Estimating derivatives of the path*

If the function $s(\theta)$ is known and differentiable, then the calculation of $s'(\theta)$ is trivial. We will assume, in general, that this is not the case, and instead, that we are given the values of s evaluated at the discretisation points, that is, $s(\theta_i)$ for $i = 0, \ldots, n$. We assume that $s(\bar{\theta}_i)$ needed in (30) can be found as

$$s(\bar{\theta}_i) = \frac{s(\theta_{i-1}) + s(\theta_i)}{2}.$$

We calculate $s'(\bar{\theta})$ as

$$s'(\bar{\theta}_i) = \frac{s(\theta_i) - s(\theta_{i-1})}{d\theta}. \tag{34}$$

Although the order of the model used to compute s' appears to have little impact on the solution, testing has shown that the discretisation of $s''(\theta)$ has a substantial impact on the fidelity of the discretised solution, as shown in Figure 1. When we talk about the fidelity of the discretised solution, we are referring to the accuracy with which applying the control inputs determined by the optimisation successfully tracks the path desired. In particular when the derivative calculation has a low order and offset stencil (Equations (35) and (36)), results are poor. The derivative calculations compared in Figure 1 are

$$s''\left(\bar{\theta}_i\right) = \frac{s(\theta_{i-2}) - 2s(\theta_{i-1}) + s(\theta_i)}{d\theta^2}, \tag{35}$$

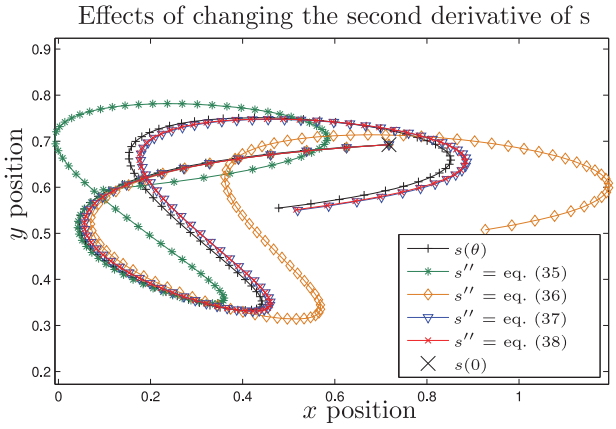

Figure 1. Trajectories are shown using the friction circle model from Section 4 to demonstrate the different methods for calculating s''. After determining the fixed control input u_i across the interval, we then apply a sixth-order Runge–Kutta scheme from Fehlberg (1968) to simulate the actual performance of the vehicle under the calculated inputs.

$$s''\left(\bar{\theta}_i\right) = \frac{s(\theta_{i-1}) - 2s(\theta_i) + s(\theta_{i+1})}{d\theta^2}, \tag{36}$$

$$s''\left(\bar{\theta}_i\right) = \frac{s(\theta_{i-2}) - s(\theta_{i-1}) - s(\theta_i) + s(\theta_{i+1})}{2d\theta^2}, \tag{37}$$

$$s''\left(\bar{\theta}_i\right) = \frac{-\frac{5}{48}s(\theta_{i-3}) + \frac{13}{16}s(\theta_{i-2}) - \frac{17}{24}s(\theta_{i-1}) - \frac{17}{24}s(\theta_i) + \frac{13}{16}s(\theta_{i+1}) - \frac{5}{48}s(\theta_{i+2})}{d\theta^2}, \tag{38}$$

where (35) and (36) are offset (more points on one side than the other) second-order models, and (37) and (38) are symmetric fourth- and sixth-order models, respectively. The fifth-order model, which is also offset, was omitted but has nearly identical performance to the sixth-order model. We implement the sixth-order model for its symmetry and improved performance.

In practice, these control laws would be implemented with feedback in order to avoid a compounding error, so the harm of a lower order discretisation would be diminished. Improved fidelity could also be achieved by using a finer mesh in the discretisation since all of the derivatives will converge in the limit for a smooth path. Whereas the computation time is linear in the size of the discretisation, the derivatives of s only need to be calculated once at the start of the algorithm, making their computation almost negligible. It is therefore better if higher fidelity can be achieved with a high-order discretisation than with a finer mesh.

6.1.6 *Calculating path derivatives at the start*

Figure 1 suggests there is one final discretisation technique we should employ, and which we have already used in Figure 1. Since the fidelity of our solution is heavily influenced by the offset of our stencil and the order of the model, for our calculation of $s''(\bar{\theta}_1)$ we will not use (36). Rather than using a higher order model with a larger offset, we will project back, and assume that prior to $s(\theta_1)$ the path was linear. Thus, we will create a ghost point at $s(\theta_{-1}) = 2s(0) - s(\theta_1)$, and use (37) to create

$$s''(\bar{\theta}_1) = \frac{\frac{1}{2}s(0) - s(\theta_1) + \frac{1}{2}s(\theta_2)}{d\theta^2}. \tag{39}$$

This approximation has the benefit of providing a symmetric stencil and being of higher order. The benefits of this approximation can be seen in Figure 2.

6.1.7 *Discretised problem*

Based on these results, our algorithm applies discretisations to produce the discrete optimisation problem

$$\begin{aligned} &\text{minimise} && (29) \\ &\text{subject to} && (30), \quad (31), \quad (32), \quad (33), \end{aligned} \tag{40}$$

where the derivatives of s are calculated according to (34), (38) and (39). The a_i, b_i and u_i are problem data, while everything else can be calculated from the given dynamics and s. The problem is now a finite-dimensional convex optimisation problem.

6.2 *Optimisation*

Problems like (40) are highly structured, and therefore can be solved quite efficiently. There is a long history of exploiting this structure in control problems dating back to shortly after the popularisation of interior point methods such as Wright (1993, 1997), Hansson and Boyd (1998), Rao, Wright, and Rawlings (1998), Hansson (2000), Potra and Wright (2000) and Wright and Nocedal (2006). While we focus on a log barrier, infeasible start, interior point method, targeted specifically at this problem, there are other approaches.

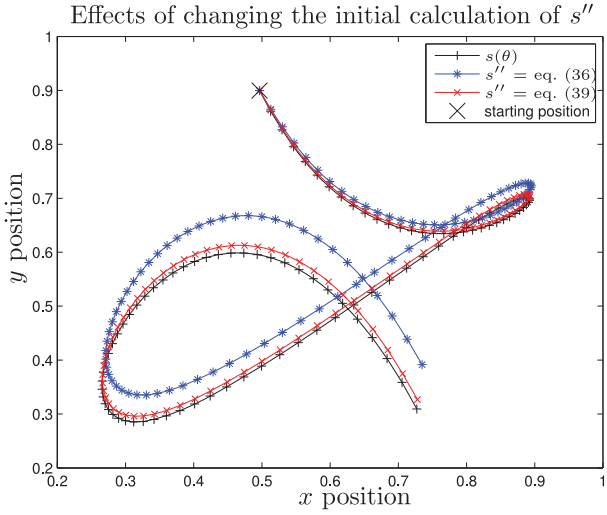

Figure 2. The general discretisation is provided by (38) but for the first point we have varied the scheme. As can be seen, the calculation of $s''(\bar{\theta}_1)$ can have a significant impact on fidelity.

As demonstrated in Verscheure et al. (2009a), this problem can be transformed into a Second Order Cone Program (SOCP) so that it may be solved by standard SOCP solvers such as SeDuMi (Sturm, 1999) or SDPT3 (Toh, Todd, & Tutuncu, 1999; Tutuncu, Toh, & Todd, 2003). In this formulation, the problem can benefit from robust preexisting solvers. However, these solvers must address all SOCP problems and therefore are not always able to take advantage of problem structure. One disadvantage of the SOCP form is that the conversion to an SOCP requires the addition of three variables per time step which may as much as double the size of the problem, diminishing the benefits of the optimised SOCP solvers.

In Hauser (2013), a method that involves solving a series of linear programs is suggested, although this approach is for a smaller class of problems (only velocity and acceleration constraints).

Another approach observes that different time steps are only coupled by b values, and therefore we could implement an operator splitting method (O'Donoghue, Stathopoulos, & Boyd, 2013). This would allow advantage to be taken of parallelisation allowing the computation time to scale linearly in the number of processors. This could be especially advantageous if truly massive speed optimisation problems were being considered.

6.2.1 *Direct interior point barrier method*

Our algorithm uses an infeasible start interior point method using log barrier functions as described in Boyd and Vandenberghe (2004). If faster solve times are needed, improvements could be made through implementation of some of the ideas in Wang and Boyd (2010) for fast model predictive control.

Interior point methods operate by solving a series of approximations of the optimisation problem desired using Newton's method. The approximations transform inequality constraints into costs added to the objective that more closely approach indicator functions in each subsequent step. Finding each Newton step requires solving a Karush–Kuhn–Tucker (KKT) system of equations representing a quadratic approximation of the optimality conditions. Thus, to facilitate rapid solving, we must increase the speed with which we can solve the KKT system.

6.2.2 *Banded system*

The KKT matrix for the optimisaiton problem

$$\begin{aligned} &\text{minimise} \;\; f_0(x) + \phi(x)/t \\ &\text{subject to} \;\; Ax = b, \end{aligned} \tag{41}$$

where f_0 is the desired function to minimise, ϕ is the barrier function, t represents the accuracy of the barrier function and the other constraints (dynamics and derivatives) are all affine equality constraints that can be encoded as $Ax = b$, has the structure

$$\begin{bmatrix} \nabla^2 f_0(x) + \nabla^2 \phi(x)/t & A^T \\ A & 0 \end{bmatrix}. \tag{42}$$

We have already observed that problem (40) is highly structured as between discretisation steps (corresponding to θ_i) only b_i appear in multiple steps. Therefore, by an appropriate ordering we can make the KKT system banded. By ordering our states $b_0, b_1, a_1, u_1, b_2, a_2, \ldots, u_n$ and interweaving the dual variables after each set of b, a, u, our KKT system is banded with bandwidth $6 + 2p + 2r$. We will in general ignore the b_0 term as it is fixed by the initial conditions and can therefore be eliminated before factoring the matrix. Therefore, without interweaving, the Hessian block of the KKT system has a predominantly block diagonal structure with of-block diagonal coupling on the b terms. Assuming that $r = 1$, the Hessian of $f(x) = f_0(x) + \phi(x)/t$ is

$$\begin{bmatrix}
\frac{\delta^2 f}{\delta b_1^2} & \frac{\delta^2 f}{\delta b_1 \delta a_1} & \frac{\delta^2 f}{\delta b_1 \delta u_1} & \frac{\delta^2 f}{\delta b_1 \delta b_2} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
\frac{\delta^2 f}{\delta b_1 \delta a_1} & \frac{\delta^2 f}{\delta a_1^2} & \frac{\delta^2 f}{\delta a_1 \delta u_1} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
\frac{\delta^2 f}{\delta b_1 \delta u_1} & \frac{\delta^2 f}{\delta a_1 \delta u_1} & \frac{\delta^2 f}{\delta u_1^2} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
\frac{\delta^2 f}{\delta b_1 \delta b_2} & 0 & 0 & \frac{\delta^2 f}{\delta b_2^2} & \frac{\delta^2 f}{\delta b_2 \delta a_2} & \frac{\delta^2 f}{\delta b_2 \delta u_2} & \frac{\delta^2 f}{\delta b_2 \delta b_3} & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\delta^2 f}{\delta b_2 \delta a_2} & \frac{\delta^2 f}{\delta a_2^2} & \frac{\delta^2 f}{\delta a_2 \delta u_2} & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\delta^2 f}{\delta b_2 \delta u_2} & \frac{\delta^2 f}{\delta a_2 \delta u_2} & \frac{\delta^2 f}{\delta u_2^2} & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\delta^2 f}{\delta b_2 \delta b_3} & 0 & 0 & \ddots & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \frac{\delta^2 f}{\delta b_n^2} & \frac{\delta^2 f}{\delta b_n \delta a_n} & \frac{\delta^2 f}{\delta b_n \delta u_n} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \frac{\delta^2 f}{\delta b_n \delta a_n} & \frac{\delta^2 f}{\delta a_n^2} & \frac{\delta^2 f}{\delta a_n \delta u_n} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \frac{\delta^2 f}{\delta b_n \delta u_n} & \frac{\delta^2 f}{\delta a_n \delta u_n} & \frac{\delta^2 f}{\delta u_n^2}
\end{bmatrix}. \tag{43}$$

The equality constraints of our system also have structure resulting in an A block of

$$\left[\begin{array}{ccc|ccc|ccc|ccc} \tilde{c}_1/2 & \tilde{m}_1 & -\tilde{R}_1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2d\theta & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \tilde{c}_2/2 & 0 & 0 & \tilde{c}_2/2 & \tilde{m}_2 & \tilde{R}_2 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 2d\theta & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \tilde{c}_n/2 & 0 & 0 & \tilde{c}_n/2 & \tilde{m}_n & \tilde{R}_n \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 & -1 & 2d\theta & 0 \end{array}\right], \tag{44}$$

where $\tilde{c}_i = \tilde{c}(\bar{\theta}_i)$, $\tilde{m}_i = \tilde{m}(\bar{\theta}_i)$ and $\tilde{R}_i = \tilde{R}(\bar{\theta}_i)$. By interweaving the blocks of A after the blocks of the Hessian, we can construct a system with bandwidth $6 + 2r + 2p$. The blocks and of-block elements in (43) contribute $4 + 2r$ to the bandwidth and the blocks and of-block elements in (44) contribute $2p + 2$ to the bandwidth. Given the slow growth of the number of iterations required for interior point methods with problem size (Boyd & Vandenberghe, 2004, , Section 11.5.3), the speed with which we can solve the algorithm is almost entirely dependent on how rapidly we can evaluate and solve the KKT system. By implementing a simple LU factorisation, we can exploit the banded structure to solve the system in only $4n(6 + 2m + 2p)^3$ flops. Thus, our solve time is linear in n. While other factorisations, such as an LDL^T factorisation, could be used, the LU factorisation is sufficiently fast for our purposes. It is worth noting that the KKT matrix is not positive definite, and therefore care must be taken during pivoting. We implement the LU factorisation using Tim Davis's sparse matrix libraries (Davis, 2006).

6.2.3 *An additional constraint*

To improve the performance of the algorithm, we also add a redundant constraint

$$b_i \geq 0, \quad i = 0, \ldots, n. \tag{45}$$

This constraint is already enforced by the domain of (29), and to keep the objective small, b_i must be large. However, because the denominator of the terms in the objective is

$$b_i^{1/2} + b_{i+1}^{1/2}, \quad i = 0, \ldots, n-1,$$

the algorithm will sometimes leave one b_i close to zero while pushing b_{i+1} larger. Once movement has begun in this direction, numerical instabilities from very small numbers can make it difficult for the algorithm to recover. By adding the redundant constraint $b_i \geq 0$, we encourage the algorithm to more evenly balance the b_i throughout its optimisation.

Algorithm. We now combine these results together to provide an overview of the algorithm. We do not delve into the details of interior point methods here; details can be found in Boyd and Vandenberghe (2004) and other optimisation texts.

Algorithm 6.1 *Speed optimisation algorithm*

Given the vehicle dynamics R, M, C, d, vehicle path s, constraint set $\mathcal{C}$ and v_{init}

(1) Discretise s at $n+1$ points, and construct discretised problem (40), adding (45) to $\mathcal{C}$.
(2) Solve for b_0 using (32).
(3) Create barrier functions for the constraint set $\mathcal{C}$.
(4) Choose a starting point for a_i, b_i, u_i, that satisfies the constraint set $\mathcal{C}$, $b_i > 0$.
In most cases, b_i small and positive, u_i and a_i zero will satisfy these constraints.
(5) Solve the infeasible start interior point problem.
Create KKT systems that interweave the blocks of (43) and (44).
Solve the KKT systems using an LU factorisation for solve times linear in n.

7. Results

7.1 *Timing*

In Figure 3, we compare our algorithm against SeDuMi and SDPT3 for problem sizes ranging from 10 time steps to 5000 time steps for the front wheel drive friction circle model given in Section 4. As a reminder, in this example $p = 2$, $r = 2$, and there are two constraints in $\mathcal{C}$. From this plot we see that the solve time is indeed linear in n and that our direct solve computes a solution on the order of 100 times faster. These times were computed on a 2.66 GHz Intel Core 2 Duo. It is worth noting that SeDuMi and SDPT3 are multithreaded, while our implementation is single threaded and still achieves a marked increase in performance.

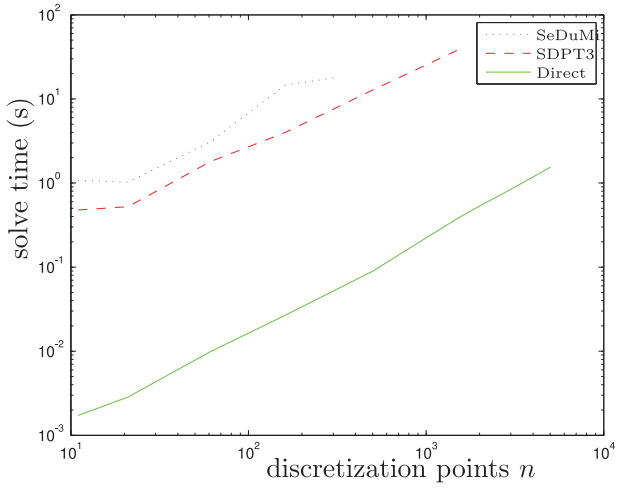

Figure 3. Timing for different solvers on the front wheel drive model given in Section 4 for problems with different values of n. For each value of n, 10 problems were run and their times were averaged. The solid line is Algorithm **1**. The dashed line is SDPT3. The dotted line is SeDuMi.

For problems with $n = 50$, we can find solutions in under a centisecond, allowing us to update our control inputs at a rate of 100 Hz. Given the high accuracy of our discretisation, we could use an even coarser discretisation (or shorter horizon) and take $n = 10$ in which case we can solve the problem in 1.7 milliseconds, corresponding to a rate of 580 Hz. At these speeds, the control could be implemented on a system in real time, allowing the path to be updated due to disturbances, drift in the vehicle or the need to avoid eminent obstacles.

7.2 *A trajectory example*

To verify that the algorithm is indeed working, we present an example with $n = 100$. As seen already, applying the calculated inputs tracks the desired path quite well. More importantly, in Figure 5 we see that the algorithm has indeed maximised the force used, which we would expect for a minimum time traversal. The few points on the interior arise from the discretisation.

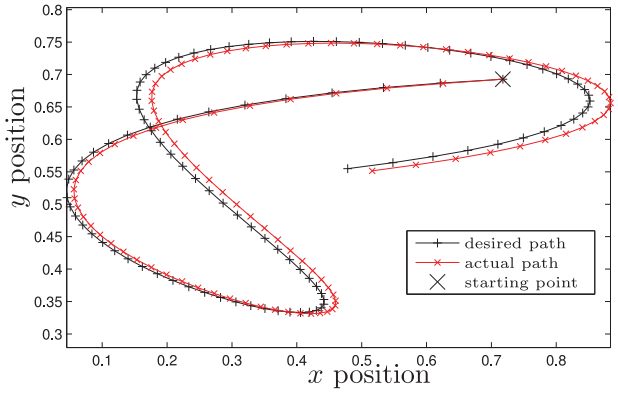

Figure 4. A sample trajectory for the front wheel drive model presented in Section 4 as calculated by Algorithm **1**. The performance of the vehicle under the calculated control inputs was then simulated using the sixth-order Runge–Kutta scheme from Fehlberg (1968).

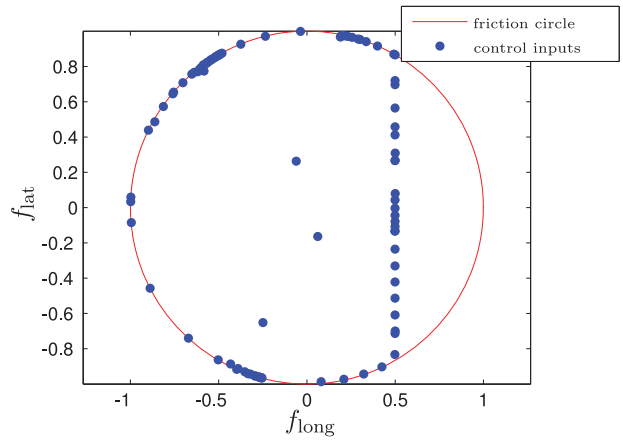

Figure 5. The force inputs found for the trajectory in Figure 4. As expected, the force limits are almost always achieved.

8. Summary

In this paper, we have presented a general framework for minimum-time speed optimisation of a vehicle along a fixed path. We showed the way in which this framework can be adapted for many types of vehicles, including cars, spacecraft and airplanes. We showed how various constraints can be incorporated which couple the position, acceleration, square of velocity and control inputs. Finally, we presented an algorithm for solving these problems that significantly outperforms SeDuMi and SDPT3 and is able to find solutions on a time scale that is feasible for embedded systems.

Acknowledgements

We would also like to thank our reviewers for their comments and suggested references which improved this paper.

Funding

This research was supported in part by a National Science Foundation Graduate Research Fellowship [grant number DGE-1147470] and by the Cleve B.Moler Stanford Graduate Fellowship.

References

Amato, N.M., & Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation* (Vol. 1, pp. 113–120). Minneapolis, MN.

Asada, H., & Slotine, J.J.E. (1986). *Robot analysis and control*. New York, NY: Wiley.

Barraquand, J., Langlois, B., & Latombe, J.C. (1992). Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics, 22*, 224–241.

Bayer, F., & Hauser, J. (2012). Trajectory optimization for vehicles in a constrained environment. In *Proceeding of the 51st IEEE Conference on Decision and Control* (pp. 5625–5630). Maui, HI.

Bertolazzi, E., Biral, F., & Da Lio, M. (2005). Symbolic–numeric indirect method for solving optimal control problems for large multibody systems. *Multibody system Dynamics, 13*, 233–252.

Bertolazzi, E., Biral, F., & Da Lio, M. (2007). Real-time motion planning for multibody systems. *Multibody System Dynamics, 17*, 119–139.

Betts, J.T. (1998). Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics, 21*, 193–207.

Betts, J.T. (2001). *Practical methods for optimal control using nonlinear programming*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Blackmore, L., & Açikmeşe, B. (2011). Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica, 47*, 341–347.

Bobrow, J.E., Dubowsky, S., & Gibson, J.S. (1985). Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research, 4*, 3–17.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Bryson, A.E., & Denham, W.F. (1962). A steepest-ascent method for solving optimum programming problems. *Journal of Applied Mechanics, 29*, 247–257.

Cardamone, L., Loiacono, D., Lanzi, P.L., & Bardelli, A.P. (2010). Searching for the optimal racing line using genetic algorithms. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games* (pp. 388–394). Copenhagen, Denmark.

Casanova, D. (2000). *On minimum time vehicle maneuvering: The theoretical optimal lap* (PhD thesis). Cranfield University, Bedford.

Caselli, S., Reggiani, M., & Rocchi, R. (2001). Heuristic methods for randomized path planning in potential fields. In *Proceedings of the 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation* (pp. 426–431). Banff, Alberta, Canada.

Constantinescu, D., & Croft, E.A. (2000). Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *Journal of Robotic Systems, 5*, 233–249.

Darby, C.L., Hager, W.W., & Rao, A.V. (2011). Direct trajectory optimization using a variable low-order adaptive pseudospectral method. *Journal of Spacecraft and Rockets, 48*, 433–445.

Davis, T.A. (2006). *Direct methods for sparse linear systems*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Dinh, Q.T., & Diehl, M. (2009). An application of sequential convex programming to time optimal trajectory planning for a car motion. In *Proceedings of the 2009 48th IEEE Conference on Decision and Control held jointly with the 2009 28th Chinese Control Conference* (pp. 4366–4371). Shanghai, China.

Donald, B., & Xavier, P. (1989). A provably good approximation algorithm for optimal-time trajectory planning. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation* (Vol. 2, pp. 958–963). Scottsdale, AZ.

Donald, B., Xavier, P., Canny, J., & Reif, J. (1993). Kinodynamic motion planning. *Journal of the ACM, 40*, 1048–1066.

Dubowsky, S., Norris, M., & Shiller, Z. (1986). Time optimal trajectory planning for robotic manipulators with obstacle avoidance: A CAD approach. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation* (Vol. 3, pp. 1906–1912). San Francisco, CA.

Elnagar, G., Kazemi, M.A., & Razzaghi, M. (1995). The pseudospectral Legendre method for discretizing optimal control problems. *IEEE Transactions on Automatic Controls, 40*, 1793–1796.

Faiz, N., Agrawal, S.K., & Murray, R.M. (2001). Trajectory planning of differentially flat systems with dynamics and inequalities. *Journal of Guidance, Control, and Dynamics, 24*, 219–227.

Faulwasser, T., Hagenmeyer, V., & Findeisen, R. (2011). Optimal exact path-following for constrained differentially flat systems. In *Proceedings of 18th IFAC World Congress* (pp. 9875–9880). Milano, Italy.

Fehlberg, E. (1968). *Classical fifth-, sixth-, seventh-, and eighth-order Runge-Kutta formulas with stepsize control* (Technical report). Washington, DC: National Aeronautics and Space Administration.

Garg, D., Patterson, M.A., Darby, C.L., Francolin, C., Huntington, G.T., Hager, W.W., & Rao, A.V. (2011). Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using a Radau pseudospectral method. *Computational Optimization and Applications, 49*, 335–358.

Hansson, A. (2000). A primal-dual interior-point method for robust optimal control of linear discrete-time systems. *IEEE Transactions on Automatic Control, 45*, 1639–1655.

Hansson, A., & Boyd, S. (1998). Robust optimal control of linear discrete-time systems using primal-dual interior-point methods. In *Proceedings of the 1998 American Control Conference* (Vol. 1, pp. 183–187). Philadelphia, PA.

Hauser, J., & Saccon, A. (2006). Motorcycle modeling for high-performance maneuvering. *IEEE Control Systems, 26*, 89–105.

Hauser, K. (2013). Fast interpolation and time-optimization on implicit contact submanifolds. In *Proceedings of Robotics: Science and Systems* (Vol. 8). Retrieved from http://www.roboticsproceedings.org/rss09/p22.pdf

Hedrick, J.K., & Bryson, A.E. (1972). Three-dimensional, minimum-time turns for a supersonic aircraft. *Journal of Aircraft, 9*, 115–121.

Hehn, M., & D'Andrea, R. (2011). Quadrocopter trajectory generation and control. In *Proceedings of the 18th IFAC World Congress* (Vol. 18, pp. 1485–1491). Milan, Italy. doi:10.3182/20110828-6-IT-1002.03178

Hendrikx, J.P.M., Meijlink, T.J.J., & Kriens, R.F.C. (1996). Application of optimal control theory to inverse simulation of car handling. *Vehicle System Dynamics, 26*, 449–461.

Houska, B., Ferreau, H.J., & Diehl, M. (2011). ACADO toolkit – an open source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods, 32*, 298–312.

Kalmár-Nagy, T., D'Andrea, R., & Ganguly, P. (2004). Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems, 46*, 47–64.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research, 30*, 846–894.

Kunz, T., & Stilman, M. (2012). Time-optimal trajectory generation for path following with bounded acceleration and velocity. In *Proceedings of the 2012 Robotics: Science and Systems Conference* (Vol. 8, pp. 9–13). Sydney, Australia.

LaValle, S.M., & Kuffner, J.J. Jr. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research, 10*, 378–400.

Lewis, F.L., Dawson, D.M., & Abdallah, C.T. (2004). *Robot manipulator control theory and practice*. New York, NY: Marcel Dekker.

Milliken, W.F., & Milliken, D.L. (1995). *Race car vehicle dynamics*. Warrendale, PA: SAE International.

O'Donoghue, B., Stathopoulos, G., & Boyd, S. (2013). A splitting method for optimal control. *IEEE Transactions on Control Systems Technology, 21*(6), 2432–2442.

Pfeiffer, F., & Johanni, R. (1987). A concept for manipulator trajectory planning. *IEEE Journal of Robotics and Automation, 3*, 115–123.

Pham, Q.C. (2013). Characterizing and addressing dynamic singularities in the time-optimal path parameterization algorithm. Retrieved from http://www.normalesup.org/pham/docs/robust.pdf

Pham, Q.C., Caron, S., & Nakamura, Y. (2013). Kinodynamic planning in the configuration space via velocity interval propagation. In *Proceedings of Robotics: Science and Systems*. Berlin, Germany.

Pin, F.G., & Vasseur, H.A. (1990). Autonomous trajectory generation for mobile robots with non-holonomic and steering angle constraints. In *Proceedings of the IEEE International Workshop on Intelligent Motion Control* (pp. 295–299). Instanbul, Turkey.

Potra, F.A., & Wright, S.J. (2000). Interior-point methods. *Journal of Computational and Applied Mathematics, 124*, 281–302.

Rao, A.V., Benson, D.A., Darby, C.L., Patterson, M.A., Francolin, C., Sanders, I., & Huntington, G.T. (2010). Algorithm 902: GPOPS, a MATLAB software for solving multiple phase optimal control problems using the Gauss pseudospectral method. *ACM Transactions on Mathematical Software, 37*, Article 22.

Rao, C.V., Wright, S.J., & Rawlings, J.B. (1998). Application of interior point methods to model predictive control. *Journal of Optimization Theory and Applications, 99*, 723–757.

Rice, R.S., & Alianello, D.A. (1970). *A driver characterizing function – The g–g diagram* (Technical report VJ-2882-K). Buffalo, NY: Cornell Aeronautical Laboratory.

Sahar, G., & Hollerbach, J.M. (1985). Planning of minimum-time trajectories for robot arms. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 751–758). St. Louis, MO.

Shiller, Z. (1994). Time-energy optimal control of articulated systems with geometric path constraints. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation* (Vol. 4, pp. 2680–2685). San Diego, CA.

Shiller, Z., & Dubowsky, S. (1989). Robot path planning with obstacles, acuator, gripper, and payload constraints. *The International Journal of Robotics Research, 8*, 3–18.

Shin, K.G., & McKay, N.D. (1985). Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control, AC-30*, 531–541.

Sturm, J.R. (1999). Using SeDuMi 1.02 a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software, 11–12*, 625–653. Retrieved from http://fewcal.kub.nl/sturm.

Toh, K.C., Todd, M.J., & Tutuncu, R.H. (1999). SDPT3 – A MATLAB software package for semidefinite programming. *Optimization Methods and Software, 11*, 545–581.

Tutuncu, R.H., Toh, K.C., & Todd, M.J. (2003). Solving semidefinite–quadratic–linear programs using SDPT3. *Mathematical Programming, 95*, 189–217.

Velenis, E., & Tsiotras, P. (2005). Minimum time vs maximum exit velocity path optimization during cornering. In *2005 IEEE International Symposium on Industrial Electronics* (pp. 335–360). Dubrovnik, Croatia.

Verscheure, D., Demeulenaere, B., Swevers, J., Schutter, J.D., & Diehl, M. (2009a). Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control, 54*, (10), 2318–2327.

Verscheure, D., Diehl, M., Schutter, J.D., & Swevers, J. (2009b). Recursive log-barrier method for on-line time-optimal robot path tracking. In *Proceedings of the 2009 American Control Conference* (pp. 4134–4140). St. Louis, MO.

von Stryk, O., & Bulirsch, R. (1997). Direct and indirect methods for trajectory optimization. *Annals of Operations Research, 37*, 357–373.

Wang, Y., & Boyd, S. (2010). Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology, 18*, 267–278.

Wright, S.J. (1993). Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications, 77*, 161–187.

Wright, S.J. (1997). Applying new optimization algorithms to model predictive control. In *Proceedings of the Fifth International Conference on Chemical Process Control – CPC V* (pp. 147–155). Tahoe City, CA.

Wright, S.J., & Nocedal, J. (2006). *Numerical optimization*. Berlin: Springer.

Zhao, Y., & Tsiotras, P. (2013). Speed profile optimization for optimal path tracking. In *Proceedings of the American Control Conference* (pp. 1171–1176). Washington, DC.